Universidade Estadual de Campinas

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

Departamento de Matemática Aplicada

Tese de Doutorado

Uma Família de Algoritmos para Programação Linear Baseada no Algoritmo de Von Neumann

por

Jair da Silva [†]

Doutorado em Matemática Aplicada - Campinas - SP

Orientador: Prof. Dr. Aurelio R. Leite Oliveira

Co-orientador: Profa. Dra. Marta Ines Velazco

[†]Este trabalho contou com apoio financeiro do CNPq.

Uma Família de Algoritmos para Programação Linear Baseada no Algoritmo de Von Neumann

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por **Jair da Silva** e aprovada pela comissão julgadora.

Campinas, 31 de março de 2009.

Prof. Dr. Aurelio R. Leite de Oliveira
Orientador

Profa. Dra. Marta Ines Velazco
Co-orientadora

Banca examinadora:

Prof. Dr. Aurelio Ribeiro Leite de Oliveira.

Profa. Dra. Márcia Helena Costa Fampa.

Prof. Dr. Carlile Campos Lavor.

Prof. Dr. Marcius Fabius Henriques de Carvalho.

Prof. Dr. Antonio Carlos Moretti.

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP como requisito parcial para obtenção do título de **Doutor em Matemática Aplicada**.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Miriam Cristina Alves - CRB8/5094

Silva, Jair da

Si38u Uma família de algoritmos para programação linear baseada no algoritmo de Von Neumann / Jair da Silva -- Campinas, [S.P. :s.n.], 2009.

Orientadores : Aurélio Ribeiro Leite de Oliveira; Marta I. Velazco

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Programação linear. 2. Algoritmos. 3. Métodos de pontos interiores. 4. Algoritmo de Von Neumann. I. Oliveira, Aurélio Ribeiro Leite de. II. Velazco Fontova, Marta Inês. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Título em inglês: A family of linear programming algorithms based on the Von Neumann Algorithm.

Palavras-chave em inglês (Keywords): 1. Linear programming. 2. Algorithms. 3. Interior point method. 4. Von Neumann Algorithm.

Área de concentração: Programação linear.

Titulação: Doutor em Matemática Aplicada.

Banca examinadora: Prof. Dr. Aurélio Ribeiro Leite de Oliveira (IMECC-UNICAMP)
Profa. Dra. Márcia Helena Costa Fampa (UFRJ)
Prof. Dr. Carlile Campos Lavor (IMECC-UNICAMP)
Prof. Dr. Marcius Fabius Henriques de Carvalho (CENPRA)
Prof. Dr. Antonio Carlos Moretti (IMECC-UNICAMP)

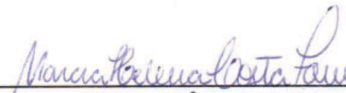
Data da defesa: 31/03/2009

Programa de pós-graduação: Doutorado em Matemática Aplicada.

Tese de Doutorado defendida em 31 de março de 2009 e aprovada

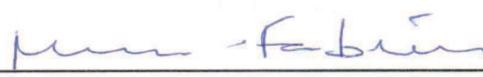
Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). AURÉLIO RIBEIRO LEITE DE OLIVEIRA

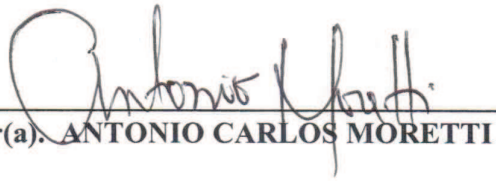


Prof(a). Dr(a). MÁRCIA HELENA COSTA FAMPA

Prof(a). Dr(a). CARLILE CAMPOS LAVOR



Prof(a). Dr(a). MARCIUS FABIUS HENRIQUES DE CARVALHO



Prof(a). Dr(a). ANTONIO CARLOS MORETTI

À minha mãe Lionete e ao
meu pai Jose (in memorian)

AGRADECIMENTOS

Este trabalho não seria possível sem inestimável orientação do Prof. Aurelio e da Profa. Marta. Agradeço imensamente o auxílio prestado através de suas valiosas sugestões e por sempre estarem dispostos à discutirem sobre os assuntos desta tese. Muito obrigado pela orientação!

À Janete, não apenas pelo apoio durante a fase final deste trabalho, mas também pelo seu carinho e companhia.

Ao Prof. Laércio Luis Vendite pelo apoio dado.

À Profa. Marcia Fampa e aos Profs Antonio Carlos Moretti, Carlile Campos Lavor e Marcius Carvalho por gentilmente terem aceito o convite para cômpor a banca examinadora.

As amigas Carla, Juliana, Daniele.

Agradeço também a todos meus amigos, tantos os feitos em Campinas como os de Maringá, pela ajuda que recebi em várias ocasiões e a minha família pelo apoio sempre que necessário.

À Cidinha, Ednaldo e Tânia por estarem sempre a disposição para solucionar as questões burocráticas.

À Deus por ter me amparado durante esta jornada.

Ao CNPq e a CAPES pelo imprescindível suporte financeiro.

RESUMO

Neste trabalho apresentamos uma nova família de algoritmos para resolver problemas de programação linear. A vantagem desta família de algoritmos é a sua simplicidade, a possibilidade de explorar a esparsidade dos dados do problema original e geralmente possuir raio de convergência inicial rápido. Esta família de algoritmos surgiu da generalização da idéia apresentada por João Gonçalves, Robert Storer e Jacek Gondzio, para desenvolver o algoritmo de ajustamento pelo par ótimo. Este algoritmo foi desenvolvido por sua vez tendo como base o algoritmo de Von Neumann. O algoritmo de Von Neumann possui propriedades interessantes, como simplicidade e convergência inicial rápida, porém, ele não é muito prático para resolver problemas lineares, visto que sua convergência é muito lenta. Do ponto de vista computacional, nossa proposta não é utilizar a família de algoritmos para resolver os problemas de programação linear até encontrar uma solução e sim explorar a sua simplicidade e seu raio de convergência inicial geralmente rápido e usá-la em conjunto com um método primal-dual de pontos interiores infactível, para melhorar a eficiência deste. Experimentos numéricos revelam que ao usar esta família de algoritmos em conjunto com um método primal-dual de pontos interiores infactível melhoramos o seu desempenho na solução de algumas classes de problemas de programação linear de grande porte.

ABSTRACT

In this work, we present a new family of algorithms to solve linear programming problems. The advantage of this family of algorithms relies in its simplicity, the possibility of exploiting the sparsity of the original problem data and usually to have fast initial ratio of convergence. This family of algorithms arose from the generalization of the idea presented by João Gonçalves, Robert Storer and Jacek Gondzio to develop the optimal pair adjustment algorithm. This algorithm was developed in its own turn based on the Von Neumann's algorithm. It has interesting properties, such as simplicity and fast initial convergence, but it is not very practical for solving linear problems, since its convergence is very slow. From the computational point of view, our suggestion is not to use the family of algorithms to solve problems of linear programming until optimality, but to exploit its simplicity and its fast initial ratio of convergence and use it together with a infeasible primal-dual interior point method to improve its efficiency. Numerical experiments show that using this family of algorithms with an infeasible primal-dual interior point method improves its performance in the solution of some classes of large-scale linear programming problems.

CONTEÚDO

Agradecimentos iv

Lista de Tabelas **ix**

Lista de Figuras **x**

Introdução **1**

1 Métodos de Pontos Interiores **4**

1.1 Problemas de Programação Linear 5

1.2 Métodos Primal-Dual 6

1.2.1 Método de Newton 6

1.2.2 Método de Pontos Interiores Primal-Dual 7

1.2.3 Método Preditor-Corretor 9

1.3 Programação Quadrática Convexa 10

2 Precondicionadores **12**

2.1 Equações de Newton 12

2.2 Precondicionador Separador 13

2.3 Precondicionador Fatoração Controlada de Cholesky 17

2.4 Mudança de Fases 19

3 Algoritmos Simples para Programação Linear **21**
3.1 Algoritmo de Von Neumann 22
3.1.1 Transformação do Problema de Programação Linear 22
3.1.2 Transformação do Problema de Programação Linear Canalizado . . . 26
3.1.3 Algoritmo de Von Neumann 27
3.2 Algoritmo de Redução por Pesos 34
3.3 Algoritmo de Ajustamento pelo Par Ótimo 36
3.3.1 Resolução do Subproblema 37

4 Uma Família de Novos Algoritmos **39**
4.1 Algoritmo de Ajustamento Ótimo para p Coordenadas 40
4.1.1 Resolução do Subproblema da Forma Clássica 41
4.1.2 Resolução do Subproblema Usando Métodos de Pontos Interiores . . 43
4.2 Propriedades Teóricas do Novo Método 45

5 Experimentos Numéricos **48**
5.1 O Código PCx 48
5.2 Implementação 49
5.3 Problemas Testes 50
5.4 Comparação do Desempenho para Diversos Valores de p Usando Matlab . . 54
5.5 Experimento em Conjunto com o Código PCx Modificado 55
5.5.1 Problemas que não Convergem na Abordagem Iterativa com Precondicionador Híbrido 57
5.5.2 Problemas que o PCx Modificado Necessita de Segunda Fase 58
5.6 Ponto Inicial 60
5.6.1 O Ponto Inicial do Código PCx Original 61
5.6.2 Experimento em Conjunto com o Código PCx Original 62

6 Conclusões e Trabalhos Futuros **67**

A Apêndice **70**
A.1 Multiplicadores de Lagrange 70
A.2 O Problema da Barreira Logarítmica 72
A.3 A Existência e Unicidade de Solução para o Problema da Barreira 74

LISTA DE TABELAS

5.1 Problemas Testes 50
5.2 Continuação da Tabela Problemas Testes 51
5.3 Continuação da Tabela Problemas Testes 52
5.4 Continuação da Tabela Problemas Testes 53
5.5 Continuação da Tabela Problemas Testes 54
5.6 Comparação do desempenho inicial para p=4,10,20 55
5.7 Problemas que não convergem na abordagem híbrida 58
5.8 PCxM × PCxMA$_{20}$ 59
5.9 Comparação do desempenho para p=2,10,20 60
5.10 PCxori × PCxoriM 62
5.11 Continuação da tabela PCxori × PCxoriM 63
5.12 Continuação da tabela PCxori × PCxoriM 64
5.13 Continuação da tabela PCxori × PCxoriM 65

LISTA DE FIGURAS

3.1 Ilustração do algoritmo de Von Neumann. 28
3.2 Ilustração do algoritmo de redução por pesos. 35

Introdução

Um problema de programação linear consiste em minimizar ou maximizar uma função linear, sujeito a restrições lineares de igualdade e/ou desigualdade. Este problema teve sua formulação nos anos de 1930 a 1940 [9]. Na metade da década de 1940 Dantzing apresentou o método simplex para programação linear. O método simplex tornou-se uma ferramenta fundamental em programação linear durante muitos anos. No entanto, este método possui convergência exponencial no número de variáveis, para problemas especialmente construídos [25].

Em 1979, Khachian [24] apresentou um método para programação linear com complexidade polinomial no pior caso, o método dos elipsóides. Este método não mostrou-se prático para resolver problemas de programação linear. Em 1984, Karmarkar [23] apresentou um outro algoritmo alternativo para programação linear, também com complexidade polinomial. A publicação deste algoritmo iniciou uma nova linha de pesquisa conhecida como métodos de pontos interiores, e uma década depois os métodos primais-duais se firmaram como os métodos mais importantes e úteis dentre os métodos de pontos interiores [47].

Neste trabalho apresentamos uma nova família de algoritmos para resolver problemas de programação linear. A vantagem desta família de algoritmos é a sua simplicidade, a possibilidade de explorar a esparsidade dos dados do problema original e geralmente possuir raio de convergência inicial rápido. Esta família de algoritmos surgiu da generalização da idéia apresentada por João Gonçalves, Robert Storer e Jacek Gondzio, para desenvolver o algoritmo de ajustamento pelo par ótimo [18]. Este algoritmo, por sua vez, foi desenvolvido tendo como base o algoritmo de Von Neumann [10, 11, 14]. O algoritmo de Von Neumann

possui propriedades interessantes como simplicidade, isto é, cada iteração sua é dominada por multiplicação matriz vetor, a possibilidade de explorar a esparsidade dos dados do problema original e geralmente possuir raio de convergência inicial rápida, porém, ele não é muito prático para resolver problemas lineares, visto que sua convergência é muito lenta.

Do ponto de vista computacional, nossa proposta não é utilizar a família de algoritmos para resolver os problemas de programação linear até encontrar uma solução e sim explorar a sua simplicidade e seu raio de convergência inicial que é geralmente rápido e usá-la em conjunto com um método de pontos interiores primal-dual infactível.

Vamos utilizar a família de algoritmos de duas formas. Na primeira, ela será usada em conjunto com o método de pontos interiores infactível dado em [45]. Em cada iteração desse método é necessário a resolução de um sistema de equações lineares no cálculo da direção de Newton, utilizando uma abordagem iterativa pelo método dos gradientes conjugados precondicionado inicialmente pelo precondicionador fatoração controlada de Cholesky [6] e posteriormente pelo precondicionador separador [37]. No entanto, esta abordagem de precondicionamento híbrida nem sempre é robusta e falha em algumas classes de problemas de programação linear. Isso ocorre devido a existência de uma faixa crítica na mudança dos precondicionadores, no sentido que, se mudarmos de etapa antes desta faixa ou no início dela, o precondicionador separador ainda não está preparado para assumir o processo e assim o método perde desempenho ou não converge. Utilizaremos a família de algoritmo simples nesta faixa crítica fazendo algumas iterações e devolvendo um ponto melhor para o método de pontos interiores. Com isso prolongamos a vida útil do precondicionador fatoração controlada de Cholesky ou melhoramos o desempenho do precondionador separador obtendo uma implementação robusta. Na segunda, como o ponto inicial para método de pontos interiores influencia muito em seu desempenho, vamos utilizar a família de algoritmos para melhorar o ponto inicial do método de pontos interiores dado em [8].

Este trabalho é constituído por mais cinco capítulos. No Capítulo 1 definimos o problema de programação linear e fazemos uma breve descrição de métodos de pontos interiores. O método primal-dual é apresentado assim como o método preditor-corretor, adicionalmente, é apresentada também uma breve descrição do problema de programação quadrática convexo. No Capítulo 2 são apresentados os precondicionadores separador e fatoração controlada de Cholesky. No Capítulo 3 encontra-se definido o problema de restrições lineares, uma vez que, os algoritmos apresentados neste capítulo resolvem os problemas neste formato. Como transformar qualquer problema de programação num problema de restrições lineares,

a apresentação do algoritmo de Von Neumann, do algoritmo de redução por pesos e do algoritmo de ajustamento pelo par ótimo. No Capítulo 4 apresentamos uma nova família de algoritmos para programação linear e propriedades de convergência do novo método. Os experimentos numéricos obtidos em conjunto com um método de pontos interiores infactível são apresentados no Capítulo 5. Finalmente as conclusões e trabalhos futuros encontram-se no Capítulo 6.

CAPÍTULO 1

Métodos de Pontos Interiores

Os métodos de pontos interiores surgiram nas décadas de 1950 e 1960. Mais precisamente, em 1955 Frisch [16] propôs o primeiro método de pontos interiores no contexto de programação não linear. Este método foi estudado por Fiacco e McCormick [15]. Em 1967, Dikin [12] publicou um trabalho que foi a base de muitos outros na área de programação linear. No entanto, somente em 1984 estes métodos tornaram-se um campo de pesquisa atrativo, com a publicação do trabalho de Karmarkar [23] que apresentou um método polinomial para resolver problemas de programação linear. Estes métodos, como o próprio nome sugere, buscam uma solução ótima do problema de programação linear percorrendo o interior de uma região.

Os métodos de pontos interiores podem ser classificados como: primal, dual e primal-dual dependendo do espaço em que estão sendo realizadas as iterações. Alternativamente, eles podem ser classificados em três categorias: métodos afim escala [12], métodos de redução de potencial [2] e métodos de trajetória central [20, 21]. A classe de métodos de pontos interiores que possui as melhores propriedades práticas e teóricas são os chamados métodos primais-duais pertencentes a categoria trajetória central. Por isso, em nosso trabalho nos restringiremos ao estudo de um método primal-dual infactível pertencente a esta classe.

Uma análise detalhada da teoria que fundamenta este capítulo pode ser encontrada nos livros Vanderbei [44] e Wright [47].

Neste capítulo, definiremos o problema de programação linear, apresentaremos uma breve

descrição do método primal-dual. Descreveremos também o método preditor-corretor, uma variante do método primal-dual.

1.1 Problemas de Programação Linear

Consideremos o seguinte problema de programação linear na forma padrão:

$$\begin{array}{lrl} \text{minimizar} & c^t x & \\ \text{s.a} & Ax & = b, \\ & x & \geq 0, \end{array} \tag{1.1}$$

onde $A \in \mathbb{R}^{m\times n}$, $c, x \in \mathbb{R}^n$, e $b \in \mathbb{R}^m$. O problema (1.1) é chamado primal, associado a este problema tem-se o problema dual, que é dado por

$$\begin{array}{lrl} \text{maximizar} & b^t y & \\ \text{s.a} & A^t y + z & = c, \\ & z & \geq 0, \end{array} \tag{1.2}$$

onde $y \in \mathbb{R}^m$ e $z \in \mathbb{R}^n$.

Os problemas (1.1) e (1.2) juntos são chamados de par primal-dual. As condições de otimalidade de primeira ordem (Karush-Kuhn-Tucker) dos problemas (1.1) e (1.2) são dadas por:

$$\begin{aligned} Ax - b &= 0, \\ A^t y + z - c &= 0, \\ XZe &= 0, \\ (x, z) &\geq 0. \end{aligned} \tag{1.3}$$

onde $X = diag(x)$, $Z = diag(z)$ e $e \in \mathbb{R}^n$ é o vetor com todas as coordenadas iguais a um.

Uma solução deste problema pode ser obtida resolvendo o sistema de equações não lineares (1.3). Se (x, y, z) for uma solução do problema (1.3), então x e (y, z) são soluções ótimas de (1.1) e (1.2), respectivamente. Um ponto (x, y, z) é dito ser factível se ele satisfaz o conjunto de restrições dos problemas primal e dual e, o ponto é dito ser interior se $(x, z) > 0$. O gap de um problema de programação linear é definido como a diferença entre os valores das funções objetivo do problema primal e dual, ou seja, $\gamma = c^t x - b^t y$ [47]. É possível mostrar que para um ponto primal e dual factível, o gap é dado por $\gamma = x^t z$.

Os problemas de programação linear quando são formulados na prática, geralmente não se encontram na forma (1.1). Em geral, apresentam variáveis que podem ser não negativas, livres ou limitadas e restrições na forma de equações ou inequações. Todos estes problemas podem ser reduzidos ao formato (1.1) acrescentando variáveis e/ou restrições.

1.2 Métodos Primal-Dual

Megiddo em [29] introduziu a teoria dos métodos do tipo primal-dual. A partir deste trabalho estes métodos foram desenvolvidos por Kojima, Mizuno e Yoshise [28] e são considerados os mais eficientes dentre as inúmeras variantes de métodos de pontos interiores. Estes métodos apresentam melhores propriedades teóricas para a análise de complexidade [35] e convergência [43]. A maioria destes métodos utiliza o método de Newton em suas iterações. Por isso, antes de descrevermos um método pertencente a esta classe vamos apresentar o método de Newton.

1.2.1 Método de Newton

Resolver um sistema de equações não lineares, consiste em determinar, simultaneamente o zero de um conjunto de funções de n variáveis. A resolução deste problema pode ser obtida pelo método de Newton, que é uma generalização do método de Newton (também conhecido como Newton-Raphson) para o cálculo de zero de função unidimensional.

Seja $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ um vetor de funções não lineares F_i definidas $F_i : \mathbb{R}^n \longrightarrow \mathbb{R}$, com as seguintes propriedades:

(i) existe $x^* \in \mathbb{R}^n$, tal que $F(x^*) = 0$,

(ii) a função F é de classe $C^1(\mathbb{R}^n)$,

(iii) para qualquer x diferente de x^*, a matriz das derivadas parciais de F, denominada matriz jacobiana (J) é não singular.

O método de Newton para determinar $F(x^*) = 0$ pode ser construído usando a série de Taylor para expandir F na vizinhança de um ponto x^k,

$$F(x^k + \triangle x^k) = F(x^k) + J(x^k) \triangle x^k + O(\triangle x^k)^2.$$

Eliminando os termos de ordem 2 e superiores, tem-se:

$$F(x^k + \triangle x^k) \approx L_k(x^k + \triangle x^k) = F(x^k) + J(x^k) \triangle x^k.$$

Fazendo $L_k(x^k + \triangle x^k) = 0$ temos

$$J(x^k) \triangle x^k = -F(x^k).$$

A solução deste sistema de equações lineares fornece a direção que move o ponto para um valor da função mais próximo do zero. As equações deste sistema são conhecidas como equações de Newton e $\triangle x^k$ como direção de Newton.

O novo ponto é dado por

$$x^{k+1} = x^k + \triangle x^k.$$

Este processo continua até que um critério de convergência seja atingido, por exemplo,

$$\sum_{j=1}^{n} |F_i(x^{k+1})| \leq \epsilon.$$

1.2.2 Método de Pontos Interiores Primal-Dual

Os métodos primais-duais de pontos interiores podem ser vistos como aplicações do método de Newton para calcular aproximações da solução de uma seqüência de sistemas não lineares (1.3) perturbados por um parâmetro μ, originado do uso da função barreira (ver apêndice A) para relaxar as restrições de não negatividade,

$$F_\mu(x, y, z) = \begin{bmatrix} Ax - b \\ A^t y + z - c \\ -XZe + \mu e \end{bmatrix} = 0, \;\; (x, z) \geq 0. \tag{1.4}$$

Notemos que se $\mu = 0$, então os problemas (1.3) e (1.4) são equivalentes. Assim a solução do problema (1.4) se aproxima da solução do problema (1.3) conforme $\mu \to 0$.

Um método primal-dual obtém uma solução aproximada para o problema gerando uma seqüência de pontos (x^k, y^k, z^k) e de parâmetros μ^k. O conjunto de pontos (x^k, y^k, z^k) satisfazendo (1.4) para todo $\mu > 0$ é denominada trajetória central.

A cada iteração do método é aplicado um passo do método de Newton para resolver o sistema (1.4), com um dado parâmetro μ^k. A direção de Newton $\triangle = (\triangle x, \triangle y, \triangle z)$ é obtida da solução do seguinte sistema de equações lineares

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{bmatrix} \triangle x^k \\ \triangle y^k \\ \triangle z^k \end{bmatrix} = \begin{bmatrix} b - Ax^k \\ c - z^k - A^t y^k \\ \mu^k e - X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_{\mu^k} \end{bmatrix}. \tag{1.5}$$

O novo ponto é dado por

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^{k1}) + (\alpha_p \triangle x^k, \alpha_d \triangle y^k, \alpha_d \triangle z^k),$$

onde α_p e α_d são os tamanhos do passo primal e dual, respectivamente e preservam a não negatividade das variáveis x e z. Esses valores são determinados da seguinte forma:

$$\alpha_p = \min(1, \nu\overline{\alpha}_p), \quad \alpha_d = \min(1, \nu\overline{\alpha}_d) \quad \text{com} \quad \nu \in (0,1), \tag{1.6}$$

onde $\overline{\alpha}_p = \dfrac{-1}{\min_i(\frac{\triangle x_i^k}{x_i^k})}$ e $\overline{\alpha}_d = \dfrac{-1}{\min_i(\frac{\triangle z_i^k}{z_i^k})}$.

O parâmetro μ^k é atualizado e o processo se repete até que seja encontrada uma solução suficientemente próxima de uma solução ótima ou que seja detectada a infactibilidade do problema. Descreveremos a seguir os passos deste método.

Método de Pontos Interiores Primal-Dual de Trajetória Central

Entradas: $(x^0, z^0) > 0$ e y^0 livre.

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Escolha $\sigma^k \in (0,1)$ e calcule $\mu^k = \sigma^k \dfrac{(x^k)^t z^k}{n}$.

[2] Calcule os resíduos r_p^k, r_d^k e $r_{\mu^k}^k$.

[3] Calcule a direção de Newton $\triangle^k$.

[4] Calcule o tamanho do passo (α_p^k, α_d^k) tal que $(x^{k+1}, z^{k+1}) > 0$, conforme (1.6).

[5] Calcule $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\alpha_p^k \triangle x^k, \alpha_d^k \triangle y^k, \alpha_d^k \triangle z^k)$.

Fim.

A cada iteração do método é necessário resolver um sistema de equações lineares para determinar a direção de Newton. Quando métodos diretos estão sendo utilizados é feita uma fatoração da matriz e em seguida são resolvidos dois sistemas lineares mais simples, os quais estão associados a matrizes triangulares. O processo de fatoração de uma matriz de ordem $\overline{n}$ apresenta complexidade $O(\overline{n}^3)$ enquanto que a solução de sistemas triangulares pode ser feita com complexidade $O(\overline{n}^2)$. Algumas variantes de métodos de pontos interiores resolvem a cada iteração vários sistemas com a mesma matriz dos coeficientes e com isso determinam uma melhor direção, reduzindo o números total de iterações e, conseqüentemente, o número de fatorações.

1.2.3 Método Preditor-Corretor

O método preditor-corretor foi proposto por Mehotra em [31], ele difere do método apresentado na seção anterior, no cálculo da direção de Newton. A direção de Newton $\triangle = (\triangle x, \triangle y, \triangle z)$ é decomposta em duas partes, $\triangle = \triangle_a + \triangle_c$.

O termo $\triangle_a$ é obtido resolvendo o sistema (1.5) para $\mu = 0$, ou seja,

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{bmatrix} \triangle_a x^k \\ \triangle_a y^k \\ \triangle_a z^k \end{bmatrix} = \begin{bmatrix} b - Ax^k \\ c - z^k - A^t y^k \\ -X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_a \end{bmatrix}. \tag{1.7}$$

A componente $\triangle_a$, conhecida como direção afim, é responsável por determinar uma melhor predição para o parâmetro da barreira. Este parâmetro é escolhido usando a seguinte heurística

$$\mu^k = \left(\frac{(x^k + \alpha_{pa} \triangle_a x^k)^t (z^k + \alpha_{da} \triangle_a z^k)}{(x^t) z^k} \right)^p \frac{(x^k + \alpha_{pa} \triangle_a x^k)^t (z^k + \alpha_{da} \triangle_a z^k)}{n}, \tag{1.8}$$

onde α_{pa} e α_{da} são os passos que preservam a não negatividade das variáveis x e z. Mehrotra sugere que o valor do expoente seja $p = 2$ ou $p = 3$.

A direção corretora, $\triangle_c$ é dada por,

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{bmatrix} \triangle_c x^k \\ \triangle_c y^k \\ \triangle_c z^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ r_m \end{bmatrix}. \tag{1.9}$$

onde, $\triangle_a X^k = diag(\triangle_a x^k)$, $\triangle_a X^k = diag(\triangle_a z^k)$ e $r_m = \mu^k e - \triangle_a X^k \triangle_a Z^k e$.

O termo $\triangle_c$ tem por objetivo fazer com que a nova iteração seja a mais central possível e fazer a correção do termo não linear. Note que, o vetor de termos independentes depende da solução do sistema (1.7).

A direção final pode ser obtida diretamente, resolvendo o sistema (1.9) com o vetor de termos independentes substituído por $[r_p \ r_d \ r_m]^t$. Assim, evita-se o cálculo $\triangle = \triangle_a + \triangle_c$. Os passos deste método são descrito a seguir:

Método Preditor-Corretor

Entradas: $(x^0, z^0) > 0$ e y^0 livre.

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Calcule os resíduos r_p^k, r_d^k e r_a^k, com $\mu^k = 0$.

[2] Calcule a direção afim-escala $\triangle_a^k = (\triangle_a x^k, \triangle_a y^k, \triangle_a z^k)$.

[3] Calcule o tamanho do passo $(\alpha_{pa}^k, \alpha_{da}^k)$ tal que $(\widetilde{x}^{k+1}, \widetilde{z}^{k+1}) > 0$.

[4] Calcule μ^k, como dado em (1.8).

[5] Calcule o resíduos r_m^k.

[6] Calcule a direção de Newton $\triangle^k = \triangle_a^k + \triangle_c^k$.

[7] Calcule o tamanho do passo $\alpha^k = (\alpha_p^k, \alpha_d^k)$ tal que $(x^{k+1}, z^{k+1}) > 0$.

[8] Calcule $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\alpha_p^k dx^k, \alpha_d^k dy^k, \alpha_d^k dz^k)$.

Fim.

O acréscimo de esforço ocorrido para resolução dos dois sistemas lineares a cada iteração com uma mesma matriz é compensado considerando a redução significativa no número de iterações. Esta variante está implementada na maioria dos softwares livres (acadêmicos) e comerciais e está descrita em muitos livros de métodos de pontos interiores.

1.3 Programação Quadrática Convexa

Nesta seção vamos apresentar o problema de programação quadrática convexa. Este conceito será utilizado na seção 4.1, onde apresentaremos uma família de algoritmos simples para programação linear, sendo necessário resolver um subproblema de programação quadrática convexa pequeno em cada iteração destes algoritmos.

Antes de definirmos o problema de programação quadrática convexa vamos definir o problema de programação convexa.

Definição 1.1. *Um conjunto K é convexo se o segmento de reta que liga qualquer par de pontos no conjunto está contido inteiramente no conjunto, ou seja, $\alpha x + (1-\alpha)y \in K$ para todo $x, y \in K$ e $\alpha \in [0,1]$.*

Definição 1.2. *Uma função $f(x)$ definida em um conjunto convexo K é uma função convexa se $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ para todo $x, y \in K$ e $\alpha \in [0,1]$.*

Definição 1.3. *Chamamos de problema de programação convexa o problema*

$$\begin{array}{ll} \textit{Minimizar} & f(x) \\ s.a & x \in K \end{array} \tag{1.10}$$

onde K é um conjunto convexo e $f(x)$ é uma função convexa.

Uma aplicação do conceito de convexidade, é que qualquer mínimo local do problema de programação convexa (1.10) é também um mínimo global. Além disso, calcular um mínimo global para problemas convexos é uma tarefa factível do ponto de vista computacional.

Agora vamos definir o problema de programação quadrática convexa. Este problema pode ser formulado como:

$$\begin{array}{ll} \text{minimizar} & x^tQx + \overline{w}^t x \\ \text{s.a} & Bx = \overline{b} \\ & x \geq 0 \end{array} \tag{1.11}$$

onde $Q \in \mathbb{R}^{n\times n}$ é uma matriz simétrica e semi-definida positiva, $B \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, e $\overline{b} \in \mathbb{R}^m$.

As condições KKT do problema (1.11) são dadas por

$$\begin{aligned} Qx + B^t\rho - \tau + \overline{w} &= 0, \\ Bx - b &= 0, \\ \tau^t x &= 0, \\ x &\geq 0, \end{aligned} \tag{1.12}$$

onde ρ é livre, $\tau \geq 0$, e são os multiplicadores de Lagrange de igualdade e desigualdade, respectivamente.

As equações KKT são condições suficientes como veremos abaixo.

Teorema 1.1. *Consideremos o problema de programação quadrática convexa (1.11), se para um dado x^* as equações KKT (1.12) são satisfeitas, então x^* é um minimizador global.*

Na realidade este resultado é mais geral e vale para todos os problemas de programação convexa.

Este resultado será utilizado na subseção 4.1.2. Com base nele, formularemos um método de pontos interiores para resolver um subproblema nesta subseção.

Uma etapa que influencia diretamente no desempenho de implementações de métodos primais-duais de pontos interiores é a solução dos sistemas de equações de Newton. No próximo capítulo veremos como os sistemas de equações de Newton podem ser reduzidos. Também veremos dois precondicionadores aplicados ao sistema reduzido quando usamos métodos iterativos para resolver este sistema.

CAPÍTULO 2

Precondicionadores

O precondicionamento é uma estratégia que deve ser aplicada aos métodos iterativos para melhorar as características de convergência de sistemas que possuam a matriz de coeficientes com autovalores dispersos. Precondicionar um sistema linear consiste em fazer com que a matriz apresente as condições desejadas para que o método que está sendo aplicado para resolver o sistema seja eficiente.

Neste capítulo, apresentaremos apenas os precondicionadores separador, desenvolvido por Oliveira e Sorensen [37, 38], e a fatoração controlada de Cholesky construída por Campos e Birkett [5, 6] que precondicionam os sistemas lineares dos métodos de pontos interiores em [37] e [3]. Antes de descrevermos estes precondicionadores faremos algumas considerações sobre as equações de Newton.

2.1 Equações de Newton

Como visto no capítulo anterior, para encontrar as direções de Newton é necessário resolver um ou mais sistemas lineares. Uma vez que esses sistemas compartilham a mesma matriz de coeficientes, podemos restringir ao estudo do sistema linear (1.5) dado por

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{bmatrix} \triangle x^k \\ \triangle y^k \\ \triangle z^k \end{bmatrix} = \begin{bmatrix} b - Ax^k \\ c - z^k - A^t y^k \\ \mu^k e - X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_{\mu^k} \end{bmatrix}. \tag{2.1}$$

O sistema linear (2.1) pode ser transformado no chamado sistema aumentado usando a terceira equação para eliminar $\triangle z^k$ da segunda equação. Definindo $D = Z^{-1}X$ o sistema resultante é dado por

$$\begin{pmatrix} -D^{-1} & A^t \\ A & 0 \end{pmatrix} \begin{bmatrix} \triangle x^k \\ \triangle y^k \end{bmatrix} = \begin{bmatrix} r_d - X^{-1}r_\mu \\ r_p \end{bmatrix}. \tag{2.2}$$

Este sistema pode ser reduzido às equações normais, utilizando a primeira equação para eliminar $\triangle x^k$ da segunda equação, ou seja,

$$ADA^t \triangle y^k = AD(r_d - X^{-1}r_\mu) + r_p. \tag{2.3}$$

As variáveis $\triangle x^k$ e $\triangle z^k$ são dadas por

$$\begin{aligned} \triangle x^k &= DA^t \triangle y^k - D(r_d - X^{-1}r_\mu) \\ \triangle z^k &= X^{-1}(r_\mu - Z\triangle x^k). \end{aligned}$$

A maioria das implementações de métodos de pontos interiores utiliza métodos diretos para resolver o sistema de equações normais [1, 26, 27]. A abordagem mais comum é a fatoração de Cholesky.

Outra abordagem para resolver o sistema de equações normais consiste em aplicar métodos iterativos. O método iterativo mais utilizado em pontos interiores é o gradiente conjugado precondicionado [30, 46]. A seguir veremos os dois precondicionadores, aplicados ao método dos gradientes conjugados na solução dos sistemas lineares do método de pontos interiores em uma abordagem híbrida.

2.2 Precondicionador Separador

O precondicionador separador foi proposto em [37, 38]. Na realidade foi apresentada uma classe de precondicionadores para o sistema aumentado (2.2). Essa classe de precondicionadores pode ser vista como uma generalização do precondicionador proposto por Resende e Veiga no contexto de problemas de fluxo de custo mínimo para redes [40]. A principal característica dessa classe de precondicionadores é que ela apresenta melhores resultados nas proximidades de uma solução ótima, quando os sistemas lineares já são muito mal condicionados. A seguir descreveremos este precondicionador.

Consideremos o seguinte precondicionador simétrico em blocos para o sistema aumentado (2.2)

$$M^{-1} = \begin{pmatrix} F & G \\ H & J \end{pmatrix}.$$

A matriz de (2.2) precondicionada por M^{-1} tem a seguinte forma,

$$\begin{pmatrix} -FD^{-1}F^t + FA^tG^t + GAF^t & -FD^{-1}H^t + FA^tJ^t + GAH^t \\ -HD^{-1}F^t + HA^tG^t + JAF^t & -HD^{-1}H^t + HA^tJ^t + JAH^t \end{pmatrix}.$$

Os blocos F, G, H, e J são construídos com o objetivo principal de evitar a presença de ADA^t na matriz precondicionada. Assim, fazendo $J = 0$ a matriz se reduz a,

$$\begin{pmatrix} -FD^{-1}F^t + FA^tG^t + GAF^t & -FD^{-1}H^t + GAH^t \\ -HD^{-1}F^t + HA^tG^t & -HD^{-1}H^t \end{pmatrix}.$$

Para que os blocos fora da diagonal sejam nulos é atribuído $G^t = (HA^t)^{-1}HD^{-1}F^t$,

$$\begin{pmatrix} -FD^{-1}F^t + FA^tG^t + GAF^t & 0 \\ 0 & -HD^{-1}H^t \end{pmatrix}.$$

Fazendo $H = [I\,0]P$, onde P é uma matriz de permutação tal que HA^t é não singular, o bloco $HD^{-1}H^t$ é equivalente a uma matriz diagonal, denotada por D_B^{-1}, sendo,

$$PD^{-1}P^t = \begin{pmatrix} D_N^{-1} & 0 \\ 0 & D_B^{-1} \end{pmatrix}.$$

Por fim, atribui-se $F = D^{\frac{1}{2}}$.

A matriz precondicionada por M^{-1} é dada por,

$$M^{-1}\begin{pmatrix} D^{-1} & A^t \\ A & 0 \end{pmatrix}M^{-t} = \begin{pmatrix} -I_n + D^{\frac{1}{2}}A^tG^t + GAD^{\frac{1}{2}} & 0 \\ 0 & -D_B^{-1} \end{pmatrix} \tag{2.4}$$

sendo,

$M^{-1} = \begin{pmatrix} D^{\frac{1}{2}} & G \\ H & 0 \end{pmatrix}, G = H^tD^{-\frac{1}{2}}B^{-1}, HP = [I\,0]$ e $AP^t = [BN]$.

O sistema aumentado (2.2) tem dimensão $n + m$, onde n é o número de variáveis e m o número de restrições do problema de programação linear. Então podemos considerar que o precondicionador reduz a dimensão do sistema aumentado para n, visto que a parte inferior é formada apenas pela matriz diagonal D_B^{-1} e, portanto facilmente resolvida.

O sistema de dimensão n envolve a matriz indefinida,

$$-I_n + D^{\frac{1}{2}}A^tG^t + GAD^{\frac{1}{2}}. \tag{2.5}$$

O Lema 2.1 mostra que esta matriz pode ser escrita na forma

$$K = -I_n + U^t V^t + VU, \tag{2.6}$$

onde $UV = V^T U^t = I_m$ e $U, V^t \in \mathbb{R}^{m\times n}$.

Lema 2.1. *Seja $A = [BN]$, com B não singular e $G = H^t D^{-\frac{1}{2}} B^{-1}$, onde $H = [I\ 0]$. Então $G^t D^{\frac{1}{2}} A^t = AD^{\frac{1}{2}} G = I$.*

A demonstração deste lema é encontrada em [37], Lema 5.1.

A seguir descreveremos as principais característica do precondicionador separador.

Redução a um sistema definido positivo

A matriz (2.5) pode ser reduzida a

$$I_m + D_B^{-\frac{1}{2}} B^{-1} N D_N N^t B^{-t} D_B^{-\frac{1}{2}} \tag{2.7}$$

ou

$$I_{n-m} + D_N^{\frac{1}{2}} N^t B^{-t} D_B^{-1} B^{-1} N D_N^{\frac{1}{2}}. \tag{2.8}$$

Essas matrizes possuem dimensões m e $n-m$ respectivamente, são simétricas, definidas positivas e apresentam propriedades importantes relacionada com a matriz K. Uma outra propriedade importante destas matrizes é que todos os autovalores são maiores ou iguais a um. Essa característica é interessante, visto que autovalores muito próximos de zero, geralmente prejudicam o desempenho de métodos iterativos na solução dos sistemas lineares.

Equivalência às equações normais

A matriz (2.7) pode ser obtida utilizando o sistema de equações normais. Para isto, seja $A = [BN]P$, onde $P \in \mathbb{R}^{n\times n}$ é uma matriz de permutação tal que $B \in \mathbb{R}^{m\times m}$ é não singular, então

$$ADA^t = [BN]PDP^t \begin{bmatrix} B^t \\ N^t \end{bmatrix} = [BN] \begin{bmatrix} D_B & 0 \\ 0 & D_N \end{bmatrix} \begin{bmatrix} B^t \\ N^t \end{bmatrix} = BD_B B^t + ND_N N^t. \tag{2.9}$$

O precondicionador é dado por $D_B^{-\frac{1}{2}} B^{-1}$ e a matriz precondicionada T é dada como segue

$$T = D_B^{-\frac{1}{2}} B^{-1} (ADA^t) B^{-t} D_B^{-\frac{1}{2}} = I_m + \widetilde{W}\widetilde{W}^t, \tag{2.10}$$

onde $\widetilde{W}^t = D_B^{-\frac{1}{2}} B^{-1} N D_N^{\frac{1}{2}} \in \mathbb{R}^{m \times n-m}$.

Notemos que próximo a uma solução ao menos $n - m$ elementos em D^{-1} são grandes. Dessa forma, uma escolha adequada das colunas de B, faz com que os elementos em $D_B^{-\frac{1}{2}}$ e D_N sejam muito pequenos nesta situação. Neste caso, $\widetilde{W}$ aproxima-se da matriz nula, T se aproxima da matriz identidade e ambos o menor autovalor de T e o maior autovalor de T se aproximam do valor 1 assim como $\kappa_2(T)$.

O preço pago por evitar o uso do sistema de equações normais ADA^t é determinar quais colunas de A devem ser escolhidas para formar B e resolver sistemas lineares com esta matriz. Entretanto, a fatoração $QB = LU$ é tipicamente mais fácil de calcular do que a fatoração de Cholesky da matriz ADA^t. A forma como as colunas de A devem ser escolhidas para formar B, tem influência no precondicionador. Uma estratégia para obter B seria minimizar $||\widetilde{W}||$. Como perto da solução a matriz precondicionada se aproxima da matriz identidade, uma heurística para formar B é selecionar as m colunas linearmente independentes de AD com menores normas.

Fatoração LU

Como o conjunto de colunas linearmente independentes não é conhecido no início do processo de construção de B, a estratégia mais econômica para esta aplicação é trabalhar com uma coluna por vez. Neste caso, quando uma coluna linearmente dependente aparece ela é eliminada da fatoração e o método continua com a próxima coluna dada pela heurística de reordenamento. Outro fator importante é o controle do preenchimento excessivo que pode ocorrer durante a fatoração. Na implementação desenvolvida em [37] é utilizada uma técnica que consiste em interromper a fatoração quando o número de elementos não nulos for superior ao de ADA^t e então reordenar as colunas encontradas para obter uma fatoração mais esparsa. Em seguida, a fatoração é reiniciada e o processo se repete até que as m colunas linearmente independentes sejam encontradas. Uma vez encontrada B, outra fatoração LU é calculada utilizando as técnicas tradicionais, sendo geralmente muito mais esparsa que a anterior, principalmente em problemas de maior dimensão. Assim, o trabalho de calcular uma nova decomposição é compensando na solução do sistema linear pelo método iterativo.

Conjunto de Colunas

O mesmo conjunto de colunas linearmente independentes pode ser usado durante algumas iterações para construir o precondicionador. Desse modo, essas iterações são mais rápidas,

uma vez que a maior parte do tempo utilizado no cálculo do precondicionador consiste em escolher as colunas e calcular a fatoração LU. Observe que como a matriz D^{-1} é alterada durante todas as iterações, o precondicionador não será o mesmo ainda que B seja mantida por algumas iterações.

2.3 Precondicionador Fatoração Controlada de Cholesky

O segundo precondicionador que veremos é a fatoração controlada de Cholesky (FCC), o qual foi proposto em 1995 [5, 6]. Esse precondicionador pode ser visto como uma variação da fatoração incompleta de Cholesky, cujo objetivo principal está em construir uma matriz precondicionada que possua os autovalores agrupados e próximos a unidade, de forma a acelerar a convergência do método dos gradientes conjugados. A seguir descreveremos esta fatoração.

Considere a matriz $ADA^t \in \mathbb{R}^{m\times m}$ fatorada em:

$$ADA^t = LL^t = \tilde{L}\tilde{L}^t + R \tag{2.11}$$

sendo:

L: o fator obtido pela fatoração completa de Cholesky;

$\tilde{L}$: o fator obtido pela fatoração incompleta de Cholesky;

R: matriz dos resíduos.

Usando $\tilde{L}$ como uma matriz precondicionadora para ADA^t, obtém-se

$$\tilde{L}^{-1}ADA^t\tilde{L}^{-t} = (\tilde{L}^{-1}L)(L^t\tilde{L}^{-t}) = (\tilde{L}^{-1}L)(\tilde{L}^{-1}L)^t.$$

Definindo $E = L - \tilde{L}$ e substituindo na igualdade acima temos

$$\tilde{L}^{-1}ADA^t\tilde{L}^{-t} = (I + \tilde{L}^{-1}E)(I + \tilde{L}^{-1}E)^t.$$

Observemos que, quando $\tilde{L} \to L$ então $E \to 0$ e portanto $\tilde{L}ADA^t\tilde{L}^{-t} \to I$.

Duff e Meurant [13] mostraram que o número de iterações necessárias para a convergência pelo método dos gradientes conjugados está diretamente relacionado com a norma de R, sendo

$$R = LE^t + E\tilde{L}^t.$$

A fatoração controlada de Cholesky é construída baseada na minimização da norma de Frobenius de E, visto que, quando $||E|| \to 0$ implica $||R|| \to 0$. Assim consideremos o seguinte problema

$$\min \|E\|_F^2 = \sum_{j=1}^{m} c_j \text{ onde, } c_j = \sum_{i=1}^{m} |l_{ij} - \tilde{l}_{ij}|^2. \tag{2.12}$$

Dividindo c_j em dois somatórios temos

$$c_j = \sum_{k=1}^{m_j+\eta} |l_{i_kj} - \tilde{l}_{i_kj}|^2 + \sum_{k=m_j+\eta+1}^{m} |l_{i_kj}|^2,$$

onde

m_j : representa o número de elementos não nulos abaixo da diagonal da coluna j da matriz original;

η : representa o número adicional de elementos não nulos permitido na fatoração incompleta.

No primeiro somatório estão os $m_j + \eta$ elementos não nulos da j-ésima coluna de $\tilde{L}$. No segundo, estão apenas os elementos do fator completo L que não estão em $\tilde{L}$. Baseado nesses fatos, o problema (2.12) pode ser resolvido usando a seguinte heurística:

- Aumentando o parâmetro η, permitindo maior preenchimento. Desta forma, c_j decresce, pois o primeiro somatório contém mais elementos.
- Escolhendo os $m_j + \eta$ maiores elementos de $\tilde{L}$ em valor absoluto para η fixo. Desta forma, os maiores elementos ficam no primeiro somatório e os menores no segundo produzindo um fator $\tilde{L}$ ótimo, para uma determinada quantidade de armazenamento.

É calculada uma coluna do precondicionador por vez, sendo armazenados os maiores elementos em valor absoluto. Dessa forma, uma melhor aproximação para a fatoração completa é obtida com a quantidade de memória disponível.

Escolha de elemento por valor

A FCC nunca considera o padrão de esparsidade da matriz original. Os elementos armazenados no precondicionador podem ou não estar nas posições correspondentes aos da matriz original, já que a escolha é feita por valor e não por posição. Este fato, produz melhores precondicionadores em comparação com precondicionadores que conservam o padrão de esparsidade.

Generalização do ICD (Improved Incomplete Cholesky Decomposition)

Jones e Plassmann [22] desenvolveram uma classe de precondicionadores na qual o padrão de esparsidade da matriz original é ignorado. Naquela abordagem é escolhido um número fixo de elementos não nulos em cada linha ou coluna do precondicionador. Este número é o mesmo da matriz original, mas os elementos armazenados são os maiores em valor absoluto. Dessa forma, a FCC pode ser vista como uma generalização do ICD, uma vez que, o número de elementos não nulos em cada linha ou coluna do precondicionador pode variar.

Precondicionador versátil

Esta classe de precondicionadores pode ser construída levando em consideração a disponibilidade de memória. O parâmetro η pode variar de $-m$ a m, onde m é a ordem da matriz. Se $\eta = -m$ será feito apenas o escalonamento diagonal da matriz, para que a matriz precondicionada possua diagonal unitária. Se $\eta = 0$, o fator L requer o mesmo espaço de armazenamento da matriz original, mas o padrão de esparsidade normalmente não será o mesmo, já que o critério de seleção é baseado nos valores numéricos dos elementos e não em suas posições. Quando $\eta = m$ ter-se-á a fatoração de Cholesky completa. Conforme η aumenta a matriz precondicionada tende a identidade, conseqüentemente, o número de iterações necessárias para determinar a solução pelo método dos gradientes conjugados é reduzido, mas o tempo de precondicionamento aumenta. Uma boa escolha de η garante um menor tempo para a solução do sistema linear.

2.4 Mudança de Fases

Em [3] Bocanegra, Campos e Oliveira propõem um precondicionador híbrido para problemas de programação linear usando os dois precondicionadores descritos nas seções anteriores. O precondicionador fatoração controlada de Cholesky é usado para precondicionar o método dos gradientes conjugados nas iterações iniciais (fase I) e nas últimas iterações (Fase II) é usado o precondicionador separador que apresenta melhores resultados nas proximidades da solução ótima. Nesta heurística, a troca dos precondicionadores ocorre quando uma das seguintes condições é satisfeita:

- o gap $(x_0^t z_0)$ inicial para o problema de programação linear é reduzido por um fator 10^6;

- o número de iterações do método dos gradientes conjugados para resolver o sistema linear é maior que $m/2$, onde m é a ordem do sistema ADA^t;

Em [45] Velazco, Oliveira e Campos propõem uma nova heurística para troca dos precondicionadores:

- Se o número de iterações necessário para convergência do método dos gradientes conjugados é maior que $m/6$, o parâmetro η na fatoração controlada de Cholesky é incrementado em $\eta = \eta + 10$. A mudança de precondicionador acontece quando η ultrapassa um η máximo fixado.

No entanto, a abordagem híbrida nem sempre é robusta e falha em algumas classes de problemas de programação linear. Isso ocorre devido a existência de uma faixa crítica na troca dos precondicionadores, no sentido que, se mudarmos de etapa antes desta faixa ou no início dela, o precondicionador separador ainda não está preparado para assumir o processo e assim o método perde desempenho ou não converge. Por outro lado, se avançarmos muito nesta faixa o precondicionador fatoração controlada de Cholesky perde em eficiência e o método pode ficar muito lento ou mesmo não convergir.

Nossa proposta é usar a família de algoritmos simples em conjunto com o método de pontos interiores dado em [45] fazendo algumas iterações do algoritmo simples na faixa crítica e devolvendo um ponto melhorado para o método. Com isso prolongamos a vida útil do precondicionador fatoração controlada de Cholesky ou melhoramos o desempenho do precondionador separador obtendo uma implementação robusta.

No próximo capítulo, definiremos o problema de restrições lineares e mostraremos como transformar qualquer problema de programação linear em um problema de restrições lineares. Apresentaremos também alguns algoritmos simples para resolver este problema.

CAPÍTULO 3

Algoritmos Simples para Programação Linear

O algoritmo de Von Neumann foi apresentado por Dantzig no início dos anos 1990 [10, 11] e mais tarde foi estudado por Epelman e Freund [14]. Este algoritmo possui propriedades interessantes, como simplicidade e convergência inicial rápida, porém, ele não é muito prático para resolver problemas lineares, visto que sua convergência é muito lenta. Gonçalves, Storer e Gondzio [18] apresentaram três novos algoritmos baseados no algoritmo de Von Neumann, sendo que o algoritmo de ajustamento pelo par ótimo obtém melhores resultados na prática.

O algoritmo de ajustamento pelo par ótimo herda as melhores propriedades do algoritmo de Von Neumann. Embora seja provado em [18] que o algoritmo de ajustamento pelo par ótimo é superior ao de Von Neumann, em termos de convergência, ainda assim ele também não é prático para resolver problemas lineares, visto que sua convergência também é muito lenta.

Ao generalizar a idéia em [18] para o algoritmo de ajustamento do par ótimo, desenvolvemos o algoritmo de ajustamento ótimo para p coordenadas. Na realidade para cada p temos um algoritmo diferente, onde p é limitado pela ordem do problema, assim desenvolvemos uma família de algoritmos.

Neste capítulo encontram-se definidos o problema de restrições lineares e como trans-

formar qualquer problema de programação linear em um problema de restrições lineares. Apresentaremos o algoritmo de Von Neumann, o algoritmo de redução por pesos e o algoritmo de ajustamento pelo par ótimo. Este último algoritmo servirá de alicerce para construção de uma nova família de algoritmos para programação linear.

3.1 Algoritmo de Von Neumann

Consideremos o problema de encontrar uma solução factível do conjunto de restrições lineares:

$$\begin{aligned} Px &= 0, \\ e^t x &= 1, \\ x &\geq 0, \end{aligned} \tag{3.1}$$

onde $P \in \mathbb{R}^{\overline{m}\times\overline{n}}$, $x \in \mathbb{R}^{\overline{n}}$ e $e \in \mathbb{R}^{\overline{n}}$ é o vetor com todas as coordenadas iguais a um, e as colunas de P tem norma um, isto é, $||P_j|| = 1$, para $j = 1, \ldots, \overline{n}$. Geometricamente as colunas P_j podem ser vistas como pontos sobre a hiperesfera $\overline{m}$-dimensional com raio unitário e centro na origem. O problema acima pode ser descrito então como atribuir ponderações x_j não negativas às colunas P_j de modo que depois de reescalado, seu centro de gravidade seja a origem.

Notemos que qualquer problema de programação linear pode ser reduzido ao problema (3.1), conforme demonstrado a seguir.

3.1.1 Transformação do Problema de Programação Linear

Nesta subseção mostraremos como transformar um problema de programação linear, no problema de encontrar uma solução factível do conjunto de restrições lineares (3.1). Para isso, consideremos o par primal-dual dados em (1.1) e (1.2).

Obter uma solução ótima para este par de problemas é equivalente a obter uma solução factível para o seguinte problema:

$$\begin{aligned} Ax &= b, \\ A^t y + z &= c, \\ c^t x - b^t y &= 0, \\ x, z &\geq 0. \end{aligned} \tag{3.2}$$

Como y é uma variável irrestrita, fazemos $y = y^+ - y^-$, onde $y^+ \geq 0$ e $y^- \geq 0$. Assim o problema (3.2) toma a forma

$$\begin{aligned} Ax &= b, \\ A^t y^+ - A^t y^- + z &= c, \\ c^y x - b^t y^+ + b^t y^- &= 0, \\ x, y^+, y^-, z &\geq 0. \end{aligned} \tag{3.3}$$

Reescrevendo (3.3) na forma matricial temos

$$\begin{aligned} \widetilde{A}\widetilde{x} &= \widetilde{b} \\ \widetilde{x} &\geq 0 \end{aligned} \tag{3.4}$$

onde

$$\widetilde{A} = \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & A^t & -A^t & I \\ c^t & -b^t & b^t & 0 \end{pmatrix}, \widetilde{x} = \begin{pmatrix} x \\ y^+ \\ y^- \\ z \end{pmatrix}$$

e $\widetilde{b} = \begin{pmatrix} b \\ c \\ 0 \end{pmatrix}$.

Agora consideremos o seguinte problema de restrições lineares

$$\begin{aligned} \widetilde{A}\widetilde{x} - \widetilde{b}\widetilde{v} &= 0, \\ e^t\widetilde{x} + \widetilde{v} &= 1, \\ \widetilde{x}, \widetilde{v} &\geq 0, \end{aligned} \tag{3.5}$$

onde $x, z \in \mathbb{R}^n$, $y^+, y^- \in R^m$ e $\widetilde{v} \in \mathbb{R}$.

Reescrevendo (3.5) na forma matricial temos

$$\begin{aligned} \widehat{A}\widehat{x} &= 0, \\ e^t\widehat{x} &= 1, \\ \widehat{x} &\geq 0, \end{aligned} \tag{3.6}$$

onde $\widehat{A} = [\widetilde{A} - \widetilde{b}]$ e $\widehat{x} = \begin{bmatrix} \widetilde{x} \\ \widetilde{v} \end{bmatrix}$.

O teorema a seguir mostra a relação entre as soluções dos problemas (3.4) e (3.5), mais precisamente, uma solução factível do problema (3.5) é solução do problema (3.4) e vice-versa. Este teorema é apresentado em [18], suplemento online.

Teorema 3.1. *Consideremos a região factível definida pelas restrições (3.4) e (3.5).*

a) Se $\widetilde{x}, \widetilde{v}$, *onde* $\widetilde{v} \neq 0$, *é solução factível de (3.5), então* $\overline{x} = \dfrac{\widetilde{x}}{\widetilde{v}}$ *é uma solução factível para (3.4).*

b) Reciprocamente se $\overline{x}$ *é uma solução factível para (3.4), então* $\widetilde{x} = \widetilde{v}\overline{x}$, *onde* $\widetilde{v} = \frac{1}{e^t\overline{x}+1}$ *é solução factível para (3.5).*

Demonstração.

a) Seja $\widetilde{x}, \widetilde{v}$, onde $\widetilde{v} \neq 0$, uma solução de (3.5). Então podemos escrever $\widetilde{A}\widetilde{x} - \widetilde{b}\widetilde{v} = 0 \Leftrightarrow \widetilde{A}\frac{\widetilde{x}}{\widetilde{v}} = \widetilde{b}$. Como $\widetilde{x} \geq 0$ e $\widetilde{v} \geq 0$, então concluímos que $\overline{x} = \frac{\widetilde{x}}{\widetilde{v}}$ é uma solução factível de (3.4).

b) Seja $\overline{x}$ uma solução factível de (3.4). Seja $\widetilde{x} = \widetilde{v}\overline{x}$, então $\widetilde{A}\overline{x}\widetilde{v} - \widetilde{v}\widetilde{b} = \widetilde{b}\widetilde{v} - \widetilde{v}\widetilde{b} = 0$. Também temos que $e^t\widetilde{v}\overline{x} + \widetilde{v} = (e^t\overline{x} + 1)\widetilde{v} = 1$. Portanto $\widetilde{x}$ é uma solução factível de (3.5). ■

Finalmente, apresentemos a última modificação para transformar o problema (3.5) no problema (3.1). Esta transformação é dada por Dantzig em [11]. Consideremos

$$P_j = \frac{\widehat{A}_j}{||\widehat{A}_j||},$$

onde $\widehat{A}_j$ são as colunas da matriz $\widehat{A}$, para $j = 1, \ldots, 2n + 2m + 1$. Então,

$$\begin{aligned} P\overline{x} &= 0, \\ e^t\overline{x} &= 1, \\ \overline{x} &\geq 0, \end{aligned} \tag{3.7}$$

onde

$$\overline{x} = \begin{bmatrix} x \\ y^+ \\ y^- \\ z \\ v \end{bmatrix},$$

$x, z \in \mathbb{R}^n, y^+, y^- \in \mathbb{R}^m, v \in \mathbb{R}$. O próximo teorema, demonstrado em [18] suplemento online, mostra que (3.6) é factível se, e somente se, (3.7) também é factível.

Teorema 3.2. *Consideremos a região factível definida pelas restrições (3.6) e (3.7).*

a) Se $\overline{x}$ é uma solução factível para (3.7), então $\widehat{x}$ é uma solução factível para (3.6), onde

$$\widehat{x}_j = \left(\frac{\overline{x}_j}{||\widehat{A}_j||}\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \frac{\overline{x}_k}{||\widehat{A}_k||}\right),$$

onde $\overline{n}$ é a dimensão das colunas de P.

b) Reciprocamente, se $\widehat{x}$ é uma solução factível para (3.6), então $\overline{x}$ é uma solução factível para (3.7), onde

$$\overline{x}_j = \left(\widehat{x}_j||\widehat{A}_j||\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \widehat{x}_k||\widehat{A}_k||\right).$$

Demonstração. a) Seja $\overline{x}$ uma solução factível de (3.7). Seja $\widehat{x}$ dado como no teorema acima. Então,

$$\begin{aligned}
P\overline{x} = 0 &\Leftrightarrow \sum_{j=1}^{\overline{n}} P_j \overline{x}_j = 0 \\
&\Leftrightarrow \sum_{j=1}^{\overline{n}} \frac{\widehat{A_j}}{||\widehat{A_j}||} \overline{x}_j = 0 \\
&\Leftrightarrow \sum_{j=1}^{\overline{n}} \left[\widehat{A_j}\left(\frac{\overline{x}_j}{||\widehat{A_j}||}\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \frac{\overline{x}_k}{||\widehat{A_k}||}\right)\right] = 0 \\
&\Leftrightarrow \widehat{A}\widehat{x} = 0.
\end{aligned}$$

Agora vamos mostrar que a restrição de convexidade em (3.6) também é satisfeita. Assim

$$e^t\widehat{x} = \sum_{j=1}^{\overline{n}} \left[\left(\frac{\overline{x}_j}{||\widehat{A_j}||}\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \frac{\overline{x}_k}{||\widehat{A_k}||}\right)\right] = 1.$$

Finalmente, como $\overline{x}_j \geq 0$, para todo j, então $\widehat{x}_j \geq 0$ para todo j. Portanto $\widehat{x}$ é solução de factível de (3.6).

b) Seja $\widehat{x}$ uma solução factível de (3.6). Seja $\overline{x}$ dada como no teorema. Então,

$$\begin{aligned}
\widehat{A}\widehat{x} = 0 &\Leftrightarrow \sum_{j=1}^{\overline{n}} \left[\frac{\widehat{A_j}}{||\widehat{A_j}||}\left(\widehat{x}_j||\widehat{A_j}||\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \widehat{x}_k||\widehat{A_k}||\right)\right] = 0 \\
&\Leftrightarrow \sum_{j=1}^{\overline{n}} \left[P_j\left(\widehat{x}_j||\widehat{A_j}||\right) \Bigg/ \left(\sum_{k=1}^{\overline{n}} \widehat{x}_k||\widehat{A_k}||\right)\right] = 0 \\
&\Leftrightarrow P\overline{x} = 0.
\end{aligned}$$

Agora

$$e^t\overline{x} = \sum_{j=1}^{\overline{n}} \left[\left(\widehat{x}_j ||\widehat{A_j}|| \right) \Big/ \left(\sum_{k=1}^{\overline{n}} \widehat{x}_k ||\widehat{A_k}|| \right) \right] = 1.$$

Finalmente, como $\widehat{x}_j \geq 0$, para todo j, então $\overline{x}_j \geq 0$ para todo j. Portanto $\overline{x}$ é solução de factível de (3.7).

3.1.2 Transformação do Problema de Programação Linear Canalizado

Nesta subseção apresentaremos a transformação do problema de programação linear canalizado, em um problema de restrições lineares. Para tanto, consideremos o seguinte problema

$$\begin{array}{rl} \text{minimizar} & c^t x \\ \text{s.a} & Ax = b, \\ & x + s = u, \\ x, s \geq 0, & \end{array} \tag{3.8}$$

onde $A \in \mathbb{R}^{m\times n}$, $c, x, s, u \in \mathbb{R}^n$, e $b \in \mathbb{R}^m$. O dual de (3.8) é dado por

$$\begin{array}{rll} \text{maximizar} & b^t y - u^T w & \\ \text{s.a} & A^t y - w + z & = c, \\ & z, w & \geq 0, \end{array} \tag{3.9}$$

onde $y \in \mathbb{R}^m$ e $z, w \in \mathbb{R}^n$.

Como na seção anterior, encontrar uma solução para o par de problemas primal-dual (3.8) e (3.9) é equivalente a encontrar uma solução factível para o seguinte problema de restrições lineares:

$$\begin{aligned} Ax - b &= 0, \\ x + s - u &= 0, \\ A^t y - w + z - c &= 0, \\ c^t x - b^t y + u^t w &= 0, \\ x, s, z, w &\geq 0. \end{aligned} \tag{3.10}$$

Seguindo o mesmo raciocínio da seção anterior, ou seja, fazendo as mesmas transformações só que agora para o problema (3.10) obtemos o problema

$$\begin{aligned} P\widehat{x} &= 0, \\ e^t\widehat{x} &= 1, \\ \widehat{x} &\geq 0. \end{aligned} \tag{3.11}$$

onde $P_j = \dfrac{\widehat{A_j}}{||\widehat{A_j}||}$, para $j = 1, \ldots, 4n + 2m + 1$ e, neste caso,

$$\widehat{A} = \begin{bmatrix} A & 0 & 0 & 0 & 0 & 0 & -b \\ I & I & 0 & 0 & 0 & 0 & -u \\ 0 & 0 & A^t & -A^t & -I & I & -c \\ c^t & 0^t & -b^t & b^t & u^t & 0^t & 0 \end{bmatrix}, \quad \widehat{x} = \begin{bmatrix} \overline{x} \\ \overline{s} \\ \overline{y}^+ \\ \overline{y}^- \\ \overline{w} \\ \overline{z} \\ \overline{v} \end{bmatrix}.$$

Notemos que antes de aplicar a transformação da subseção (1.2.1), substituímos a variável irrestrita y por $y = y^+ - y^-$, onde $y^+, y^- \geq 0$.

3.1.3 Algoritmo de Von Neumann

O algoritmo de Von Neumann foi proposto a Dantzig em 1948 por Von Neumann em comunicação privada e foi divulgado por Dantzig no início dos anos 1990 em [10, 11]. Este algoritmo é simples, a figura (3.1) descreve como ele trabalha em cada iteração. Primeiramente ele encontra a coluna P_s de P que forma o maior ângulo com o resíduo b^{k-1}, e então o próximo resíduo é a projeção da origem no segmento de reta ligando b^{k-1} a P_s. Descrevemos a seguir este algoritmo:

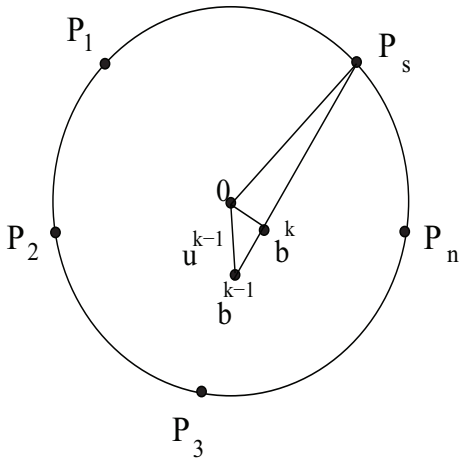


Figura 3.1: Ilustração do algoritmo de Von Neumann.

Algoritmo de Von Neumann

Dado: $\mathrm{x}^0 \geq 0$, com $e^t x^0 = 1$.

Calcule $b^0 = Px^0$.

Para $k = 1, 2, 3, ...$ **faça**

[1] Calcule

$$s^+ = \text{argmin}_{j=1,\ldots,\overline{n}} P_j^t b^{k-1},$$
$$v_{k-1} = P_{s^+}^t b^{k-1}.$$

[2] Se $v_{k-1} > 0$, então **PARE**; o problema (3.1) é infactível.

[3] Calcule

$$u^{k-1} = ||b^{k-1}||,$$
$$\lambda = \frac{1 - v_{k-1}}{(u^{k-1})^2 - 2v_{k-1} + 1}.$$

[4] Atualize

$$b^k = \lambda b^{k-1} + (1 - \lambda) P_{s^+},$$
$$x^k = \lambda x^{k-1} + (1 - \lambda) e_{s^+},$$

onde e_{s^+} é o vetor da base canônica com 1 na s^+-ésima coordenada.

$k = k + 1$.

Fim

Note que $b^k = Px^k$, para todo $k \geq 0$. No algoritmo acima x^0 é arbitrário, desde que $e^t x^0 = 1$, por exemplo podemos tomar $x^0 = \frac{1}{n} e$. Na iteração k a coluna P_{s^+} é a coluna que forma maior ângulo com o vetor b^{k-1}.

Notemos ainda que $v_{k-1} = P_{s^+}^t b^{k-1} \leq 0$, então $0 < 1 - v_{k-1} < (u^{k-1})^2 - 2v_{k-1} + 1$, assim, $0 < \lambda < 1$. Também temos que a norma do novo resíduo é menor que a anterior, ou seja, $u^k < u^{k-1}$. Podemos ver isto facilmente na figura (3.1), o triângulo $0b^{k-1}b^k$ possui hipotenusa $u^{k-1} = \overline{0b}^{k-1}$ e lado $u^k = \overline{0b}^k$.

O critério de parada usado para o algoritmo de Von Neumann em [18] é satisfeito quando a diferença relativa entre $||b^{k-1}||$ e $||b^k||$, isto é, $(||b^{k-1} - b^k||)/||b^{k-1}||$ é menor que uma certa tolerância especificada.

O esforço por iteração do algoritmo de Von Neumann é dominado por multiplicação matriz vetor, necessário na seleção da coluna P_{s^+} que é $O(\overline{mn})$. Notemos que o número de operações requeridas nesta multiplicação tem uma redução significativa, se a matriz P é esparsa.

A convergência do algoritmo de Von Neumann foi estudada por Dantzig [10, 11] e por Epelman e Freund [14]. A seguir apresentaremos os resultados de convergência deste algoritmo, incluindo suas demonstrações, visto que, Gonçalves não estuda a convergência do algoritmo de ajustamento pelo para ótimo, ele somente mostra a convergência deste baseada na teoria de convergência do algoritmo de Von Neumann. O primeiro resultado apresentado em [17, 18], fornece um limite superior para a norma do resíduo b^k, na k-ésima iteração considerando a norma $||.||_1$.

Lema 3.1. *Se o algoritmo de Von Neumann completou k iterações ($k \geq 1$), então* $||b^k|| = ||Px^k|| \leq 1/\sqrt{k}$.

Demonstração. A demonstração é feita por indução sobre k. Para $k = 1$, temos que

$$||b^1|| = ||Px^1|| \leq ||P||||x^1|| \leq 1/\sqrt{1},$$

pois

$$||P|| = \max\{||Px|| : ||x|| = 1\} = \max\{||P_j||, j = 1, \ldots, \overline{n}\} = 1,$$

e $||x^1|| = 1$.

Por hipótese de indução temos $||b^{k-1}|| = ||Px^{k-1}|| \leq 1/\sqrt{k-1}$. Agora na iteração k $||b^k|| = ||\lambda b^{k-1} - (1-\lambda)P_{s^+}||$. Como λ é o valor que minimiza $||b^k||$ sobre todos os valores $\lambda \in [0, 1]$. Então tomando $\overline{\lambda} = (k-1)/k \in [0, 1]$. Segue que

$$\begin{aligned} ||b^k||^2 &= ||\lambda b^{k-1} - (1-\lambda)P_{s^+}||^2 \\ &\leq ||\overline{\lambda} b^{k-1} - (1-\overline{\lambda})P_{s^+}||^2 \\ &= \left\| \frac{k-1}{k} b^{k-1} - \frac{1}{k} P_{s^+} \right\| \\ &= \frac{1}{k^2}\left[(k-1)^2||b^{k-1}||^2 + ||P_{s^+}||^2 + 2(k-1)P_{s^+}^t b^{k-1}\right]. \end{aligned}$$

Como o algoritmo não termina na k-ésima iteração, então $P_{s^+}^t b^{k-1} \leq 0$. Assim, temos que

$$||b^k||^2 \leq \frac{1}{k^2}[(k-1)^2||b^{k-1}||^2 + 1].$$

Substituindo a hipótese de indução $||b^{k-1}|| \leq 1/\sqrt{k-1}$ nesta expressão, obtemos

$$||b^k||^2 \leq \frac{1}{k^2}[(k-1)^2 \frac{1}{\sqrt{k-1}} + 1] = \frac{1}{k}.$$

Portanto $||b^k|| \leq 1/\sqrt{k}$.

■

O resultado acima também é valido para a $||.||_2$, é provado por Dantzig [11].

O próximo teorema é o resultado principal de convergência do algoritmo de Von Neumann e foi apresentado por Dantzig em [11]. Antes de apresentá-lo necessitamos de uma definição.

Definição 3.2. *Uma ϵ-solução de (3.1) é uma solução aproximada x^k tal que, $x^k \geq 0$, $e^t x^k = 1$ e $||b^k|| = ||Px^k|| \leq \epsilon$.*

Teorema 3.3. *Dado $\epsilon > 0$, se o problema (3.1) é factível, então o algoritmo de Von Neumann obtém uma ϵ-solução de (3.1) em não mais que $\lceil 1/\epsilon^2 \rceil$ iterações.*

Demonstração. Pelo lema(3.1), temos que após k iterações, $||b^k|| \leq 1/\sqrt{k}$. Então, para uma ϵ-solução é suficiente termos $\epsilon = ||b^k||$. Assim, seja $\epsilon = ||b^k||$. Então, temos que $k \leq 1/\epsilon^2$. Portanto, o algoritmo necessita de não mais que $\lceil 1/\epsilon^2 \rceil$ iterações para obter uma ϵ-solução. ■

Notemos que a convergência do algoritmo de Von Neumann independe do número de linhas $\overline{m}$ e do número de colunas $\overline{n}$ de P, o que é potencialmente uma vantagem. Notemos também, que o teorema (3.3) considera somente a convergência quando o problema (3.1) é factível. A análise feita por Epelman e Freund [14] que apresentaremos a seguir, trata de ambos os casos: factível e infactível.

Inicialmente precisaremos introduzir algumas definições adicionais, que serão usadas nesta análise. Seja $H = \{Px; e^t x = 1, x \geq 0\}$ a casca convexa de todas as colunas da

matriz P. Notemos que o problema (3.1) é factível quando 0 pertence a H, isto é, $0 \in H$. Definimos r como a distância de 0 até fronteira de H. Matematicamente isto pode ser formulado como

$$r = \inf\{||h - 0||; h \in \partial H\} = \inf\{||h||; h \in \partial H\}.$$

Observemos que $r = 0$ precisamente quando a origem 0 está sobre a fronteira da casca convexa das colunas de P. Assim, quando $r = 0$, o problema (3.1) tem solução factível, mas mudanças arbitrariamente pequenas nos dados $(P, 0)$ podem produzir exemplos de problemas (3.1) que não tem solução factível. Neste caso, Epelman e Freund chamam o problema (3.1) de instável, que é equivalente a dizer mal-posto na linguagem da perturbação de dados e números de condição. Em contraste, o problema (3.1) é dito bem-posto quando $r > 0$.

Quando (3.1) tem solução factível, r pode ser interpretado como o raio da maior bola centrada na origem 0 e inteiramente contida na casca convexa das colunas de P. Se (3.1) não é factível, r é a distância da origem 0 a casca convexa das colunas de P.

A análise de desempenho do algoritmo de Von Neumann feita por Epelman e Freund é baseada na quantidade r. A análise começa com o resultado técnico abaixo.

Proposição 3.3. *Suponha que o problema (3.1) tem uma solução factível. Seja b^{k-1} o resíduo no início da iteração k, e seja P_{s^+} a coluna selecionada na iteração k. Então $P_{s^+}^t b^{k-1} + r||b^{k-1}|| \leq 0$.*

Demonstração. Se $b^{k-1} = 0$, o resultado segue facilmente. Suponhamos que $b^{k-1} \neq 0$. Como (3.1) possui uma solução factível, r é o raio da maior bola centrada na origem 0 que está contida em H. Assim existe um $h \in H$ tal que $r\dfrac{-b^{k-1}}{||b^{k-1}||} = h$. Pela definição de H, $h = P\overline{x}$ para algum $\overline{x} \geq 0$, $e^t\overline{x} = 1$. Logo

$$P\overline{x} = r\frac{-b^{k-1}}{||b^{k-1}||}.$$

Também temos que

$$\min\{(b^{k-1})^t Px; x \geq 0, e^t x = 1\} =$$

$$\min\left\{\sum_{j=1}^{n}(b^{k-1})^t Px_j; x \geq 0, e^t x = 1\right\} = (b^{k-1})^t P_{s^+}.$$

Pela definição do algoritmo de Von Neumann,

$$b^{k-1}P_{s^+} \leq b^{k-1}P_j, j = 1, \ldots, n, j \neq s.$$

Combinando as expressões acima, obtemos

$$(b^{k-1})^t P_{s^+} \leq (b^{k-1})^t P\overline{x} = (b^{k-1})^t r \frac{-b^{k-1}}{||b^{k-1}||} = -r||b^{k-1}||.$$

Portanto,

$$(b^{k-1})^t P_{s^+} + r||b^{k-1}|| \leq 0.$$

■

A proposição (3.3) é usada para provar o seguinte raio de convergência do algoritmo de Von Neumann.

Lema 3.4. *Suponha que o problema (3.1) tem uma solução factível, e que* $r > 0$. *Se o algoritmo de Von Neumann completou* k *iterações* ($k \geq 1$), *então*

$$||b^k|| \leq e^{-kr^2/2}.$$

Demonstração. No final da iteração k temos que

$$||b^k|| = ||\lambda b^{k-1} + (1-\lambda)P_{s^+}|| = ||[1-(1-\lambda)b^{k-1}] + (1-\lambda)P_{s^+}||.$$

Elevando ao quadrado e substituindo a expressão de λ dada no algoritmo de Von Neumann, obtemos

$$\begin{aligned}
||b^k||^2 &= \left\| (1 - \frac{||b^{k-1}||^2 - v_{k-1}}{||b^{k-1}||^2 - 2v_{k-1} + 1}) b^{k-1} + (\frac{||b^{k-1}||^2 - v_{k-1}}{||b^{k-1}||^2 - 2v_{k-1} + 1}) P_{s^+} \right\|^2 \\
&= \left\| b^{k-1} - \frac{||b^{k-1}||^2 - v_{k-1}}{||b^{k-1}||^2 - 2v_{k-1} + 1} (b^{k-1} - P_{s^+}) \right\|^2 \\
&= ||b^{k-1}||^2 - 2\frac{||b^{k-1}||^2 - v_{k-1}}{||b^{k-1}||^2 - 2v_{k-1} + 1} (b^{k-1})^t (b^{k-1} - P_{s^+}) \\
&\quad + \frac{||b^{k-1}||^2 - v_{k-1}}{||b^{k-1}||^2 - 2v_{k-1} + 1} ||b^{k-1} - P_{s^+}||^2 \\
&= ||b^{k-1}||^2 - \frac{(||b^{k-1}||^2 - v_{k-1})^2}{||b^{k-1}||^2 - 2v_{k-1} + 1} \\
&= \frac{||b^{k-1}||^2 - (v_{k-1})^2}{||b^{k-1}||^2 - 2v_{k-1} + 1}.
\end{aligned}$$

A penúltima simplificação é possível porque $(b^{k-1})^t (b^{k-1} - P_{s^+}) = ||b^{k-1}||^2 - v_{k-1}$ e $||b^{k-1} - P_{s^+}|| = ||b^{k-1}||^2 - 2v_{k-1} + 1$.

Relembrando da proposição (3.3) que $v_{k-1} = P_{s^+}^t b^{k-1} \leq -r||b^{k-1}|| \leq 0$. Então,

$$||b^{k-1}||^2 - (v_{k-1})^2 \leq ||b^{k-1}||^2 - r^2||b^{k-1}||^2.$$

Usando a expressão acima e que $||b^{k-1}||^2 - 2v_{k-1} + 1 \geq 1$ obtemos

$$||b^{k-1}||^2 \leq \frac{||b^{k-1}||^2 - r^2||b^{k-1}||^2}{1} = ||b^{k-1}||^2(1 - r^2).$$

Usando a inequação $1 - t \leq e^{-t}$ para $t = r^2$, na inequação acima temos que

$$||b^k||^2 \leq ||b^{k-1}||^2 e^{-r^2}$$

Como $||b^k|| \leq ||b^{k-1}||$ para todo k, então

$$||b^k|| \leq ||b^0|| e^{-kr^2/2} \leq e^{-kr^2/2}.$$

■

Agora, apresentaremos o resultado de convergência do algoritmo de Von Neumann desenvolvido por Epelman e Freund.

Teorema 3.4. *Suponha que $r > 0$ e seja $\epsilon > 0$. Se o problema (3.1) é factível, então o algoritmo de Von Neumann obtém uma ϵ-solução de (3.1) em não mais que*

$$\lceil \frac{2}{r^2} \ln \frac{1}{\epsilon} \rceil$$

iterações. Se (3.1) é infactível, então o algoritmo de Von Neumann prova a infactibilidade em não mais que $\lfloor 1/r^2 \rfloor$ iterações.

Demonstração. Suponhamos que o problema (3.1) é factível. Relembrando que para uma ϵ-solução $||b^k|| \leq \epsilon$. Pelo lema (3.4) garantimos que

$$e^{-kr^2}/2 \leq \epsilon.$$

Aplicando o logaritmo na inequação acima temos que

$$k \geq 2r^2 \frac{1}{\epsilon}.$$

Assim, se (3.1) é factível, o algoritmo de Von Neumann necessita

$$\lceil \frac{2}{r^2} \ln \frac{1}{\epsilon} \rceil$$

iterações para encontrar uma ϵ-solução.

Suponhamos agora que o problema (3.1) é infactível. Relembramos que, em tal caso, r é caracterizado como a distância da origem 0 a casca convexa das colunas de P, isto é, $r = \inf\{||Px||; x \geq, e^t x = 1\}$. Assim $||b^k|| = ||Px|| \geq r$ para todo k. Por outro lado, pelo lema (3.1) se o algoritmo de Von Neumann completou k iterações, então $||b^k|| \leq 1/\sqrt{k}$. Assim, se k iterações foram completadas, temos que

$$r \leq ||b^k|| \leq \frac{1}{\sqrt{k}}$$

então

$$k \leq \frac{1}{r^2}.$$

Portanto, se (3.1) é infactível concluímos que precisamos de não mais que $\lfloor \frac{1}{r^2} \rfloor$ iterações do algoritmo de Von Neumann para provar a infactibilidade do problema 3.1.

■

3.2 Algoritmo de Redução por Pesos

O algoritmo de redução por pesos foi proposto por Gonçalves, Storer e Gondzio [18], como uma tentativa de melhorar a eficiência do algoritmo de Von Neumann, este é baseado na idéia que o resíduo b^{k-1} pode ser movido de tal forma a aproximar-se da origem 0, aumentando um peso x_j de alguma coluna P_j e reduzindo o peso x_i de uma outra coluna P_i. Em particular esperamos que o novo resíduo b^k esteja mais próximo da origem 0 que o resíduo b^{k-1}, se aumentarmos o peso correspondente da coluna P_{s^+}, que forma o maior ângulo com o resíduo b^{k-1} e reduzirmos o peso correspondente da coluna P_{s^-} que forma o menor ângulo com o resíduo b^{k-1}. Isto corresponde a fazer o resíduo b^{k-1} mover na direção $P_{s^+} - P_{s^-}$. O novo resíduo b^k é o ponto que minimiza a distância da origem 0 a esta linha. É claro que, a minimização da distância da linha a origem está restrita ao decréscimo máximo possível de x_{s^-}. Já que $x_j \geq 0$ para todo j, então podemos decrescer x_{s^-} até ele se anular. A figura (3.2) é uma ilustração geométrica de como o algoritmo trabalha em cada iteração. Descrevemos a seguir este algoritmo:

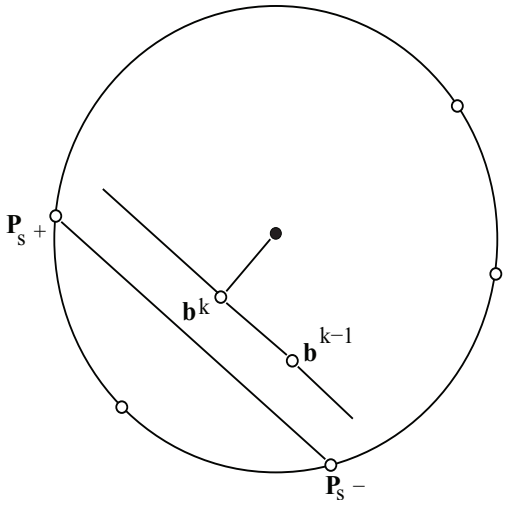


Figura 3.2: Ilustração do algoritmo de redução por pesos.

Algoritmo de Redução por Pesos

Dado: $\mathrm{x}^0 \geq 0$, com $e^t x^0 = 1$.

Calcule $b^0 = Px^0$.

Para $k = 1, 2, 3, ...$ **faça**

[1] Calcule

$$s^+ = \text{argmin}_{j=1,...,\overline{n}}\{P_j^t b^{k-1}\},$$
$$s^- = \text{argmax}_{j=1,...,\overline{n}}\{P_j^t b^{k-1} |\ x_j > 0\},$$
$$v_{k-1} = P_{s^+}^t b^{k-1}.$$

Seja $d = P_{s^+} - P_{s^-}$.

[2] Se $v_{k-1} > 0$, então **PARE**; o problema (3.1) é infactível.

[3] Calcule

$$u^{k-1} = ||b^{k-1}||,$$
$$\lambda = \frac{-d^t b^{k-1}}{||d||^2}.$$

[4] Se $\lambda > x_{s^-}$, então $\lambda = x_{s^-}$.

[5] Atualize

$$b^k = b^{k-1} + \lambda d,$$
$$x^k = x^{k-1} + \lambda(e_{s^+} - e_{s^-}),$$

$k = k + 1$.

Fim.

Uma iteração do algoritmo de redução por pesos não é garantidamente melhor que uma iteração do algoritmo de Von Neumann. Entretanto, o algoritmo de redução por pesos pode ser facilmente modificado tal que uma iteração redução por pesos é substituída por uma iteração do algoritmo de Von Neumann quando o último fornece uma maior melhoria.

O esforço por iteração do algoritmo de redução por pesos é dominado por multiplicação matriz vetor, necessário na seleção das colunas P_{s^+} e P_{s^-} que é $O(\overline{mn})$. Este é o maior esforço computacional como no algoritmo de Von Neumann.

3.3 Algoritmo de Ajustamento pelo Par Ótimo

O algoritmo de ajustamento pelo par ótimo é uma generalização do algoritmo de redução por pesos, também proposto por Gonçalves, Storer e Gondzio [18]. De um certo modo, podemos dizer que o algoritmo de ajustamento pelo par ótimo prioriza apenas duas variáveis em cada iteração, porque ele encontra o valor ótimo para duas coordenadas e ajusta o restante das coordenadas em função destes valores. Este algoritmo identifica os vetores P_{s^+} e P_{s^-} que tem o maior e o menor ângulo com o vetor b^{k-1}, respectivamente. Em seguida ele encontra os valores $x^k_{s^+}$, $x^k_{s^-}$ e λ onde $x^k_j = \lambda x^{k-1}_j$ para todo $j \neq s^+$ e $j \neq s^-$, que minimiza a distância de b^k a origem satisfazendo a convexidade e as restrições de não negatividade. Este problema de otimização tem a solução facilmente calculada examinando as condições de Karush-Kuhn-Tucker (KKT).

Descrevemos o algoritmo de ajustamento pelo par ótimo a seguir:

Algoritmo de Ajustamento pelo par Ótimo

Dado: $\mathrm{x}^0 \geq 0$, com $e^t x^0 = 1$.

Calcule $b^0 = Px^0$.

Para $k = 1, 2, 3, ...$ **faça**

[1] Calcule

$$s^+ = \mathrm{argmin}_{j=1,...,\overline{n}}\{P_j^t b^{k-1}\},$$
$$s^- = \mathrm{argmax}_{j=1,...,\overline{n}}\{P_j^t b^{k-1} |\ x_j > 0\},$$
$$v_{k-1} = P_{s^+}^t b^{k-1}.$$

[2] Se $v_{k-1} > 0$, então **PARE**; o problema (3.1) é infactível.

[3] Resolva o problema

$$\begin{array}{ll} \text{minimizar} & ||\overline{b}||^2 \\ \text{s.a} & \lambda_0(1 - x_{s^+}^{k-1} - x_{s^-}^{k-1}) + \lambda_1 + \lambda_2 = 1, \\ & \lambda_0 \geq 0, \\ & \lambda_1 \geq 0, \\ & \lambda_2 \geq 0. \end{array} \tag{3.12}$$

onde,

$\overline{b} = \lambda_0(b^{k-1} - x_{s^+}^{k-1}P_{s^+} - x_{s^-}^{k-1}P_{s^-}) + \lambda_1 P_{s^+} + \lambda_2 P_{s^-}$.

[4] Atualize

$b^k = \lambda_0(b^{k-1} - x_{s^+}^{k-1}P_{s^+} - x_{s^-}^{k-1}P_{s^-}) + \lambda_1 P_{s^+} + \lambda_2 P_{s^-}$,

$u^k = ||b^k||$,

$$x_j^k = \begin{cases} \lambda_0 x_j^{k-1}, & j \neq s^+ \text{ e } j \neq s^-, \\ \lambda_1, & j = s^+, \\ \lambda_2, & j = s^-. \end{cases}$$

$k = k + 1$.

Fim.

Como no algoritmo de Von Neumann devemos notar que $b^k = Px^k$, para todo $k \geq 0$, o ponto inicial x^0 é arbitrário desde que $e^t x^0 = 1$, por exemplo podemos tomar $x^0 = \frac{1}{n}e$. Na iteração k a coluna P_{s^+} é a coluna que forma maior ângulo com o vetor b^{k-1} e a coluna P_{s^-} é a coluna que forma o menor ângulo com o vetor b^{k-1} tal que $x_{s^-} > 0$.

3.3.1 Resolução do Subproblema

Para resolver o problema (3.12), inicialmente eliminamos a variável λ_0. Temos que

$$\lambda_0 = \frac{1 - \lambda_1 - \lambda_2}{1 - x_{s^+}^{k-1} - x_{s^-}^{k-1}} \tag{3.13}$$

então substituindo (3.13) em (3.12), podemos reescrever o problema como

$$\begin{array}{ll} \text{minimizar} & ||\overline{b}||^2 \\ \text{s.a} & 1 - \lambda_1 - \lambda_2 \geq 0, \\ & \lambda_1 \geq 0, \\ & \lambda_2 \geq 0. \end{array} \tag{3.14}$$

onde $\overline{b} = \frac{1-\lambda_1-\lambda_2}{1-x_{s^+}^{k-1}-x_{s^-}^{k-1}}(b^{k-1} - x_{s^+}^{k-1}P_{s^+} - x_{s^-}^{k-1}P_{s^-}) + \lambda_1 P_{s^+} + \lambda_2 P_{s^-}$.

Definindo $g_0(\lambda_1, \lambda_2) = \lambda_1 + \lambda_2 - 1$, $g_1(\lambda_1, \lambda_2) = -\lambda_1$, $g_2(\lambda_1, \lambda_2) = -\lambda_2$. As restrições do problema (3.14) podem ser reescritas como $g_0(\lambda_1, \lambda_2) \leq 0$, $g_1(\lambda_1, \lambda_2) \leq 0$ e $g_2(\lambda_1, \lambda_2) \leq 0$. Denotando a função objetivo por $f(\lambda_1, \lambda_2)$ e seja $(\overline{\lambda}_1, \overline{\lambda}_2)$ uma solução factível. Pela convexidade da função objetivo e das restrições, se $(\overline{\lambda}_1, \overline{\lambda}_2)$ é uma solução ótima local, ela também é uma solução ótima global. Assim $(\overline{\lambda}_1, \overline{\lambda}_2)$ satisfaz a condição necessária (KKT) dada por

$$\begin{aligned} \nabla f(\overline{\lambda}_1, \overline{\lambda}_2) + \sum_{i=0}^{2} \mu_i \nabla g_i(\overline{\lambda}_1, \overline{\lambda}_2) = 0, \\ g_i(\overline{\lambda}_1, \overline{\lambda}_2) \leq 0, \text{ para } i = 0, 1, 2, \\ \mu_i g_i(\overline{\lambda}_1, \overline{\lambda}_2) = 0, \text{ para } i = 0, 1, 2, \\ \mu_i \geq 0, \text{ para } i = 0, 1, 2, \end{aligned}$$

onde os escalares μ_i são os multiplicadores de Lagrange. Pela convexidade da função objetivo e das restrições, as condições KKT são também suficientes como vimos na seção 1.3. Resolvemos o problema (3.14) selecionando uma solução factível entre todas as possibilidades que satisfazem a condição KKT. Isto é feito analisando os seguintes casos:

(a)$\lambda_1 = \lambda_2 = 0$;

(b)$\lambda_1 = 0, 0 < \lambda_2 < 1$;

(c)$\lambda_1 = 0, \lambda_2 = 1$;

(d)$0 < \lambda_1 < 1, \lambda_2 = 0$;

(e)$0 < \lambda_1 < 1, 0 < \lambda_2 < 1, \lambda_1 + \lambda_2 - 1 = 0$;

(f)$\lambda_1 = 1, \lambda_2 = 0$;

(g)$0 < \lambda_1 < 1, 0 < \lambda_2 < 1, \lambda_1 + \lambda_2 - 1 \neq 0$;

Para cada caso acima substituímos os valores conhecidos na equação KKT e resolvemos o sistema linear resultante, obtendo assim a solução do subproblema.

No próximo capítulo apresentaremos uma nova família de algoritmos para programação linear a partir da generalização do algoritmo de ajustamento pelo par ótimo.

CAPÍTULO 4

Uma Família de Novos Algoritmos

O algoritmo de ajustamento pelo par ótimo construído em [18] prioriza duas coordenadas em cada iteração. Vamos nos referir a este, como o algoritmo para 2 variáveis. Utilizando a mesma idéia contida neste algoritmo podemos generalizá-lo e construir o algoritmo para p variáveis. A idéia central utilizada no algoritmo para 2 variáveis ao dar prioridade a duas coordenadas consiste em resolver o subproblema (3.12). Este subproblema pode ser generalizado se no lugar de utilizarmos duas colunas para formular o problema utilizarmos qualquer quantidade de colunas e assim dar importância a quantas variáveis desejarmos.

A forma de escolha das variáveis priorizadas é livre e podemos escolhe-las, de acordo com o problema que iremos resolver. Se vamos construir um algoritmo para p variáveis, uma escolha natural é tomarmos $p/2$ colunas que fazem o maior ângulo com o vetor b^k e as outras $p/2$ colunas são as que fazem o menor ângulo com o vetor b^k, se p for ímpar colocamos uma coluna a mais para o conjunto de vetores que formam o maior ângulo com o vetor b^k, por exemplo.

Neste capítulo, apresentaremos o algoritmo de ajustamento ótimo para p coordenadas. Na realidade para cada p temos um algoritmo diferente, onde p é limitado pela ordem do problema, assim apresentaremos uma nova família de algoritmos para programação linear. A grande vantagem desta família é a sua simplicidade e seu raio de convergência inicial rápido.

Em cada iteração do algoritmo de ajustamento ótimo para p coordenadas é necessário resolver um sistema não linear de ordem no máximo $(p+1) \times (p+1)$ sob algumas condições.

Veremos neste capítulo duas estratégias de como resolve-lo: na primeira (forma clássica) [41], testamos todos os casos possíveis de soluções factíveis, no entanto, o número destes casos cresce exponencialmente com o valor de p, mais precisamente eles são $2^{p+1}-1$ como veremos adiante. Na segunda [42], contornamos a dificuldade resolvendo este sistema não linear usando um método de pontos interiores. A grande vantagem de usar métodos de pontos interiores para resolver o subproblema é que o custo computacional para uma matriz de ordem 10×10 ou de ordem 100×100 não é muito importante considerando que o problema de programação linear a ser resolvido é de grande porte. Já verificar os possíveis $2^{10}-1$ casos ou os possíveis $2^{100}-1$ casos representam custos bem diferentes.

4.1 Algoritmo de Ajustamento Ótimo para p Coordenadas

Nesta seção, vamos descrever o algoritmo de ajustamento ótimo para p coordenadas, ele começa identificando as s_1 colunas que fazem o maior ângulo com o vetor b^{k-1}, em seguida ele encontra as s_2 colunas que fazem o menor ângulo com o vetor b^{k-1}, onde $s_1+s_2=p$ e p é o número de colunas que queremos priorizar. Depois, o subproblema de otimização é resolvido e é atualizado o resíduo e o ponto corrente. Este algoritmo segue a mesma idéia geral do algoritmo de ajustamento pelo par ótimo:

Algoritmo de Ajustamento Ótimo para p Coordenadas

Dado: $\mathrm{x}^0 \geq 0$, com $e^t x^0 = 1$.

Calcule $b^0 = Px^0$.

Para $k = 1, 2, 3, \ldots$ **faça**

[1] Calcule

$\{P_{\eta_1^+}, \ldots, P_{\eta_{s_1}^+}\}$ que fazem o maior ângulo com o vetor b^{k-1}.

$\{P_{\eta_1^-}, \ldots, P_{\eta_{s_2}^-}\}$ que fazem o menor ângulo com o vetor b^{k-1} e tal que $x_i^{k-1} > 0, i = \eta_1^-, \ldots, \eta_{s_2}^-$, onde $s_1 + s_2 = p$.

$v_{k-1} = \mathrm{maximo}_{i=1,\ldots,s_1} P_{\eta_i^+}^t b^{k-1}$.

[2] Se $v_{k-1} > 0$, então **PARE**; o problema (3.1) é infactível.

[3] Resolva o problema

$$\begin{aligned} \text{minimizar} \quad & ||\bar{b}||^2 \\ \text{s.a} \quad & \lambda_0 \left(1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} + \sum_{j=1}^{s} \lambda_{\eta_j^-} = 1, \\ & \lambda_{\eta_i^+} \geq 0, \ \text{para}, i = 1, \ldots, s_1, \\ & \lambda_{\eta_i^-} \geq 0, \ \text{para}, j = 1, \ldots, s_2, \end{aligned} \tag{4.1}$$

onde,

$$\bar{b} = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} P_{\eta_j^-}.$$

[4] Atualize

$$b^k = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} P_{\eta_j^-},$$

$$u^k = ||b^k||,$$

$$x_j^k = \begin{cases} \lambda_0 x_j^{k-1}, \ j \notin \{\eta_1^+, \ldots, \eta_{s_1}^+, \eta_1^-, \ldots, \eta_{s_2}^-\}, \\ \lambda_{\eta_i^+}, \qquad j = \eta_i^+, i = 1, \ldots, s_1, \\ \lambda_{\eta_j^-}, \qquad j = \eta_j^-, j = 1, \ldots, s_2. \end{cases}$$

$k = k + 1$.

Fim

4.1.1 Resolução do Subproblema da Forma Clássica

Em cada iteração do algoritmo de ajustamento ótimo para p coordenadas, é necessário resolver o subproblema (4.1). Este subproblema é resolvido encontrando uma solução no octante positivo, sujeito a factibilidade para um sistema linear de ordem no máximo $(p + 1) \times (p + 1)$. Em [41] resolvemos este subproblema verificando todos os possíveis casos de soluções factíveis. O inconveniente que surge naturalmente, ao resolver o subproblema desta forma é que o número de casos possíveis que temos que verificar cresce exponencialmente com o valor p.

De fato em [41], para resolver o subproblema (4.1) primeiramente eliminamos a variável λ_0 do problema. Assim temos que

$$\lambda_0 = \frac{1 - \sum_{i=1}^{s_1} \lambda_{\eta_i^+} - \sum_{j=1}^{s} \lambda_{\eta_j^+}}{1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1}} \tag{4.2}$$

então substuindo (4.2) em (4.1) podemos reescrever o problema como

$$\begin{aligned} \text{minimizar} \quad & ||\overline{b}||^2 \\ \text{s.a} \quad & 1 - \sum_{i=1}^{s_1} \lambda_{\eta_i^+} - \sum_{j=1}^{s_2} \lambda_{\eta_j^-} \geq 0, \\ & \lambda_{\eta_i^+} \geq 0 \text{ para } i = 1, \ldots, s_1, \\ & \lambda_{\eta_j^-} \geq 0 \text{ para } j = 1, \ldots, s_2, \end{aligned} \tag{4.3}$$

onde

$$\overline{b} = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} P_{\eta_j^-}.$$

Definindo

$g_0(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = \sum_{i=1}^{s_1} \lambda_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} - 1,$

$g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = -\lambda_{\eta_i^+}$, para $i = 1, \ldots, s_1$ e

$h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = -\lambda_{\eta_j^-}$, para $j = 1, \ldots, s_2$.

As restrições do problema (4.3) podem ser reescritas como

$g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leq 0$, para $i = 0, \ldots, s_1$ e

$h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leq 0$, para $j = 1, \ldots, s_2$. Denotando a função objetivo por $f(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-})$ então a condição necessária (KKT) é dada por

$$\begin{aligned} \nabla f(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) + \sum_{i=0}^{s_1} \mu_{g_i} \nabla g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) + \\ + \sum_{j=1}^{s_2} \mu_{h_j} \nabla h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \\ g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leq 0, \text{ para } i = 0, \ldots, s_1, \\ h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leq 0, \text{ para } j = 1, \ldots, s_2, \\ \mu_{g_i} g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \text{ para } i = 0, \ldots, s_1, \\ \mu_{h_j} h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \text{ para } j = 1, \ldots, s_2, \\ \mu_{g_i} \geq 0, \text{ para } i = 0, \ldots, s_1 \\ \mu_{h_j} \geq 0, \text{ para } j = 1, \ldots, s_2. \end{aligned} \tag{4.4}$$

Então resolvemos o problema (4.3) selecionando uma solução factível entre todas as possibilidades que satisfazem a condição KKT. Fazemos isto, analisando todos os casos possíveis de valores para as variáveis $\lambda_{\eta_i^+}$, para $i = 1, \ldots, s_1$ e $\lambda_{\eta_j^-}$, para $j = 1, \ldots, s_2$. Os casos a considerar são:

1) Caso $\lambda_{\eta_i^+} = 0$, $i = 1, \ldots, s_1$, e $\lambda_{\eta_j^-} = 0$, $j = 1, \ldots, s_1$.

2) Caso com um $\lambda_k \neq 0$, e $k \in \{\eta_1^+, \ldots, \eta_{s_1}^+, \eta_1^-, \ldots, \eta_{s_2}^-\}$ e os demais iguais a zero, o que nos fornece uma combinação C_1^p de casos, mas temos que considerar ainda o caso em que $\lambda_k = 1$, e $\lambda_k \neq 1$, assim temos $2C_1^p$ possibilidades de casos.

3) Caso com $\lambda_i \neq 0$ e $\lambda_j \neq 0$, e $i, j \in \{\eta_1^+, \ldots, \eta_{s_1}^+, \eta_1^-, \ldots, \eta_{s_2}^-\}$ e os demais iguais a zero, o que nos fornece uma combinação de C_2^p de casos, mas novamente temos que considerar os casos em que $\lambda_i + \lambda_j = 1$ e $\lambda_i + \lambda_j \neq 1$, assim temos $2C_2^p$ possibilidades.

Continuando com o raciocínio acima, concluímos que, o número total de casos quando queremos modificar p coordenadas é de

$$1 + 2C_1^p + 2C_2^p + 2C_3^p + \ldots + 2C_p^p = 2^{p+1} - 1. \tag{4.5}$$

A equação (4.5), mostra que o número de casos a ser considerado cresce exponencialmente com o valor de p. Este fato torna inviável a utilização do algoritmo de ajustamento ótimo para p coordenadas mesmo para valores razoavelmente pequenos de p através desta abordagem.

4.1.2 Resolução do Subproblema Usando Métodos de Pontos Interiores

Com a finalidade de contornar o problema do crescimento exponencial do números de casos com o valor de p, abordamos o subproblema (4.1) de outra forma e podemos resolvê-lo aplicando métodos de pontos interiores. A grande vantagem de usar métodos de pontos interiores para resolver o subproblema é que o custo computacional para uma matriz de ordem 10×10 ou de ordem 100×100 não é muito importante considerando que o problema de programação linear a ser resolvido é de grande porte. Já verificar os possíveis $2^{10} - 1$ casos ou os possíveis $2^{100} - 1$ casos representam custos bem diferentes.

A seguir reformularemos o subproblema (4.1), com objetivo de usar um método de pontos interiores para resolvê-lo.

Primeiramente reescrevemos o subproblema na forma matricial:

$$\begin{aligned} &\text{minimizar } \tfrac{1}{2}||W\lambda||^2 \\ &\text{s.a} \quad a^t\lambda = 1, \\ &\qquad -\lambda \leq 0, \end{aligned} \tag{4.6}$$

onde

$$\begin{aligned} W &= \left[\overline{w}\, P_{\eta_1^+} \dots P_{\eta_{s_1}^+} P_{\eta_1^-} \dots P_{\eta_{s_2}^-}\right], \\ \overline{w} &= b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-}, \\ \lambda &= \left(\lambda_0, \lambda_{\eta_1^+}, \dots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \dots, \lambda_{\eta_{s_2}^-}\right), \\ a &= (a_1, 1, \dots, 1), a_1 = 1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1}. \end{aligned} \tag{4.7}$$

As equações KKT do problema (4.6) são dadas por

$$\begin{aligned} W^t W\lambda + a\tau - \mu &= 0, \\ \mu^t\lambda &= 0, \\ a^t\lambda - 1 &= 0, \\ -\lambda &\leq 0, \end{aligned} \tag{4.8}$$

onde τ é livre, $0 \leq \mu$, são os multiplicadores de Lagrange de igualdade e desigualdade, respectivamente, e a matriz W^tW é de ordem $(p+1)\times(p+1)$. Estas são as equações onde aplicamos um método de pontos interiores.

O sistema linear que surge em cada iteração do método de pontos interiores aplicado as equações (4.8) tem a forma:

$$\begin{pmatrix} W^tW & a & -Id \\ U & 0 & \Lambda \\ a^t & 0 & 0 \end{pmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta\tau \\ \Delta\mu \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \tag{4.9}$$

onde, $U = diag(\mu)$, $\Lambda = diag(\lambda)$, $r_1 = \mu - a\tau - W^tW\lambda$, $r_2 = -\tau^t\lambda$, e $r_3 = 1 - a^t\lambda$.

Fazendo algumas substituições simples chegamos que as direções $\Delta\lambda$, $\Delta\tau$ e $\Delta\mu$ são dadas por

$$\begin{aligned} \Delta\mu &= \Lambda^{-1}r_2 - \Lambda^{-1}U\Delta\lambda, \\ \Delta\lambda &= (W^tW + \Lambda^{-1}U)^{-1}r_4 - (W^tW + \Lambda^{-1}U)^{-1}a\Delta\tau, \\ a^t(W^tW + \Lambda^{-1}U)^{-1}a\Delta\tau &= a^t(W^tW + \Lambda^{-1}U)^{-1}r_4 - r_3, \end{aligned}$$

onde $r_4 = r_1 + \Lambda^{-1}r_2$.

Assim, para calcular as direções precisamos resolver os seguintes sistemas lineares

$$\begin{aligned} (W^tW + \Lambda^{-1}U)l1 &= a \\ (W^tW + \Lambda^{-1}U)l2 &= r_4. \end{aligned}$$

No entanto, resolver estes sistemas lineares não causa dificuldades, porque fatoramos uma matriz definida positiva de ordem $(p+1)\times(p+1)$ e resolvemos os dois sistemas com a mesma fatoração.

Resolver o subproblema usando método de pontos interiores é bem vantajoso, visto que, por exemplo, se quisermos modificar 10 coordenadas é muito mais fácil fatorar uma matriz definida positiva de ordem 11×11 do que verificar 2^{11} casos.

4.2 Propriedades Teóricas do Novo Método

Nesta seção, provaremos o desempenho superior da família de algoritmos aqui desenvolvida, em relação ao algoritmo de Von Neumann. Também provaremos que se $p_2 \geq p_1$ então o algoritmo de ajustamento ótimo para p_2 coordenadas possui um desempenho superior em relação ao algoritmo de ajustamento ótimo para p_1 coordenadas.

Teorema 4.1. *O decréscimo em $||b^k||$ obtido por uma iteração do algoritmo de ajustamento ótimo para p coordenadas, com $1 \leq p \leq \overline{n}$, onde $\overline{n}$ representa o números de colunas de P, no pior caso é igual ao obtido por uma iteração do algoritmo Von Neumann.*

Demonstração. Dado $k \geq 1$, seja b^{k-1} o resíduo no início da iteração k e $\{P_{\eta_1^+}, \ldots, P_{\eta_{s_1}^+}\}$ e $\{P_{\eta_1^-}, \ldots, P_{\eta_{s_2}^-}\}$ o conjunto de vetores que fazem o maior e menor ângulo com o vetor b^{k-1}, respectivamente. Depois da k-ésima iteração no algoritmo de ajustamento ótimo para p coordenadas temos que

$$b^k = \overline{\lambda}_1 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \overline{\lambda}_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_2} \overline{\lambda}_{\eta_j^-} P_{\eta_j^-}.$$

onde $(\overline{\lambda}_1, \overline{\lambda}_{\eta_1^+}, \ldots, \overline{\lambda}_{\eta_{s_1}^+}, \overline{\lambda}_{\eta_1^-}, \ldots, \overline{\lambda}_{\eta_{s_2}^-})$ é a solução ótima do problema (4.1) priorizando p coordenadas. Temos que

$$(\lambda_{vn}, \lambda_{vn} x_{\eta_1^+}^{k-1} + 1 - \lambda_{vn}, \lambda_{vn} x_{\eta_2^+}^{k-1}, \ldots, \lambda_{vn} x_{\eta_{s_1}^+}^{k-1}, \lambda_{vn} x_{\eta_1^-}^{k-1}, \ldots, \lambda_{vn} x_{\eta_{s_2}^-}^{k-1}),$$

onde λ_{vn} é o λ da iteração Von Neumann, é uma solução factível para (4.1). Então

$$||\lambda_{vn} b^{k-1} + (1 - \lambda_{vn}) P_{\eta_1^+}|| = ||b_{vn}^k|| \geq ||b^k||,$$

onde b_{vn}^k é o resíduo obtido por aplicar uma iteração do algoritmo de Von Neumann. Com isto concluímos que a redução em $||b^k||$ por uma iteração do algoritmo de ajustamento ótimo para p coordenadas no pior caso é igual a redução por uma iteração do algoritmo de Von Neumann. ■

O teorema acima garante, que a família de algoritmos converge no pior caso com a mesma taxa de convergência do algoritmo de Von Neumann. No caso $p = 2$, Gonçalves, Storer e Gondzio usam os mesmos argumentos dado no teorema (4.1) para provar em [18], que este algoritmo converge no pior caso com a taxa de convergência do algoritmo de Von Neumann. Eles também mostram em testes numéricos que este algoritmo possui um desempenho superior ao algoritmo de Von Neumann.

Teorema 4.2. *O decréscimo em $||b^k||$ obtido por uma iteração do algoritmo de ajustamento ótimo para p_2 coordenadas, no pior caso é igual ao obtido por uma iteração do algoritmo de ajustamento ótimo para p_1 coordenadas com $p_1 \leq p_2 \leq \overline{n}$, onde $\overline{n}$ é a dimensão das colunas de P.*

Demonstração. Seja k, $k \geq 1$ dado e seja b^{k-1} o resíduo no início da iteração k. Seja $\{P_{\eta_1^+}, \ldots, P_{\eta_{s_1}^+}\}$ e $\{P_{\eta_1^-}, \ldots, P_{\eta_{s_2}^-}\}$ o conjunto de vetores que formam o maior e menor ângulo com o vetor b^{k-1}, respectivamente, para o algoritmo que prioriza p_2 coordenadas, onde $s_1 + s_2 = p_2$ e seja $\{P_{\eta_1^+}, \ldots, P_{\eta_{s_3}^+}\}$ e $\{P_{\eta_1^-}, \ldots, P_{\eta_{s_4}^-}\}$ o conjunto de vetores que formam o maior e menor ângulo com o vetor b^{k-1}, respectivamente, para o algoritmo que prioritiza p_1 coordenadas, onde $s_3 + s_4 = p_1$. Após a k-ésima iteração no algoritmo de ajustamento ótimo para p_2 coordenadas temos que

$$b_{p_2}^k = \overline{\lambda}_1 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} P_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \overline{\lambda}_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_2} \overline{\lambda}_{\eta_j^-} P_{\eta_j^-},$$

onde $(\overline{\lambda}_1, \overline{\lambda}_{\eta_1^+}, \ldots, \overline{\lambda}_{\eta_{s_1}^+}, \overline{\lambda}_{\eta_1^-}, \ldots, \overline{\lambda}_{\eta_{s_2}^-})$ é a solução ótima do problema (4.1) priorizando p_2 coordenadas. Temos que a solução ótima de (4.1) priorizando p_1 coordenadas

$(\widetilde{\lambda}_1, \widetilde{\lambda}_{\eta_1^+}, \ldots, \widetilde{\lambda}_{\eta_{s_3}^+}, \widetilde{\lambda}_{\eta_1^-}, \ldots, \widetilde{\lambda}_{\eta_{s_4}^-})$, é também uma solução factível para (4.1) priorizando p_2 coordenadas. Então

$$\left\| \widetilde{\lambda}_1 \left(b^{k-1} - \sum_{i=1}^{s_3} x_{\eta_i^+}^{k-1} P_{\eta_i^+} - \sum_{j=1}^{s_4} x_{\eta_j^-}^{k-1} P_{\eta_j^-} \right) + \sum_{i=1}^{s_3} \widetilde{\lambda}_{\eta_i^+} P_{\eta_i^+} + \sum_{j=1}^{s_4} \widetilde{\lambda}_{\eta_j^-} P_{\eta_j^-} \right\| = \left\| b_{p_1}^k \right\| \geq \left\| b_{p_2}^k \right\|,$$

onde $\mathrm{b}_{p_1}^k$ é o resíduo obtido ao aplicar uma iteração do algoritmo de ajustamento ótimo para p_1 coordenadas. Com isto concluímos que a redução em $||b^k||$ de uma iteração do algoritmo de ajustamento ótimo para p_2 coordenadas no pior caso é igual a redução de uma iteração do algoritmo de ajustamento ótimo para p_1 coordenadas. ■

Este teorema revela que para $p > 2$, os algoritmos da família possui um desempenho superior, do ponto de vista teórico, do que o algoritmo de ajustamento pelo par ótimo. Este teorema também revela que quanto maior o valor de p mais robusto será o algoritmo do ponto

de vista teórico. Os dois teoremas acima garantem a convergência da família de algoritmos. No próximo capítulo, apresentaremos os experimentos numéricos que confirmam este fato.

CAPÍTULO 5

Experimentos Numéricos

Neste capítulo são apresentados os experimentos numéricos deste trabalho. Nos experimentos iniciais, o algoritmo de ajustamento ótimo para p coordenadas é implementado em Matlab e comparamos o desempenho deste para os casos $p = 4$, $p = 10$ e $p = 20$. No segundo experimento o algoritmo de ajustamento ótimo para p coordenadas foi adaptado ao código PCx modificado e comparamos seu desempenho com esta abordagem. No terceiro experimento o algoritmo de ajustamento ótimo para p coordenadas foi adaptado ao código PCx original com o objetivo de melhorar o ponto inicial deste e comparamos seu desempenho com esta abordagem.

5.1 O Código PCx

O Código PCx Original: o código PCx [8] foi desenvolvido no Optimization Technology Center at Argonne National Laboratory and Northwestern University e implementa o método primal-dual preditor-corretor [31] com múltiplas correções [19]. Esta abordagem é considerada a mais eficiente das variantes de métodos de pontos interiores. As rotinas são implementadas em C exceto as rotinas responsáveis pela solução dos sistemas lineares, que utiliza uma biblioteca para fatoração esparsa de Cholesky, desenvolvida por Ng e Peyton [36] em FORTRAN77. O PCx trabalha com problemas de programação linear no formato MPS. Os problemas podem conter restrições de igualdade e desigualdade, variáveis livres e

limitadas. Uma rotina de preprocessamento converte os modelos em um formato padrão e após a solução ser encontrada é transformada em termos da formulação original.

O Código PCx Modificado: o código PCx, no qual agregamos a família de algoritmos possui as modificações descritas em [37] e [3]. Estas modificações são:

- A eliminação das múltiplas correções de centralidade [19] que implicam na solução de vários sistemas lineares a cada iteração, que é uma estratégia mais adequada para tratar com abordagens diretas, visto que a fatoração é feita uma única vez. Com a eliminação, tem-se o método primal-dual preditor-corretor e são resolvidos dois sistemas lineares a cada iteração.

- A resolução dos dois sistemas lineares é feita iterativamente pelo método dos gradientes conjugados pré-condicionado, inicialmente pela fatoração controlada de Cholesky (fase I), quando é satisfeito o critério de mudança de fase, o precondicionador utilizado é o separador (fase II).

5.2 Implementação

Para a implementação da família de algoritmos utilizamos o formato do problema dado na subseção 3.1.2, com uma pequena alteração, visto que, na implementação temos que levar em consideração que nem todas as variáveis do problema pode ser canalizadas.

Não é necessário construir a matriz P explicitamente. Para encontrar o resíduo em cada iteração, fazemos o produto da matriz P por um vetor utilizando a estrutura esparsa desta e também da matriz de restrições A.

A família de algoritmos pode produzir uma situação não prevista até aqui que deve ser evitada. Em cada iteração o algoritmo pode produzir uma solução do problema (3.1), que não é uma solução do problema de programação linear. Isto ocorre, porque na matriz P existem m pares de colunas opostas geradas quando fazemos a mudança da variável irrestrita $y = y^+ - y^-$. Então por exemplo, se uma destas colunas está no conjunto das que formam o maior ângulo com o resíduo, sua oposta está no conjunto das que formam o menor ângulo com o resíduo, assim o algoritmo fornece um peso de 0.5 para cada uma destas colunas e anula todas as demais. Esta não é uma solução do problema de programação linear, porque a última variável do problema foi anulada (ver teorema 3.2). A estratégia utilizada para não ocorrer este caso divide a matriz P em dois conjuntos, sendo o primeiro formado pelas

colunas de 1 até $n+m$ e o segundo contendo as demais colunas. Em cada iteração utilizamos alternadamente apenas um destes conjuntos para encontrar as colunas que formam o maior e menor ângulo com o resíduo. Uma vez que as colunas y^+ e y^- estão em conjuntos diferentes esta situação não ocorre mais.

Resolvemos o subproblema (4.8), necessário em cada iteração da família de algoritmos, por método de pontos interiores, com uma precisão relativamente alta. Isto, se deve ao fato que o vetor $\widehat{x}$ tem a propriedade que $e^t\widehat{x} = 1$, ou seja, cada fator $\widehat{x}_j$ pode ser pequeno, então se devolvemos uma solução com uma precisão relaxada, esta solução causa instabilidade no algoritmo.

Os experimento numéricos foram realizados em uma máquina com processador AMD Athlon 64X2 Dual Core 4200+ 2.20 GHz e com 2GB de RAM, em Linux usando os compiladores gcc e gfortran.

5.3 Problemas Testes

Os problemas usados como testes são apresentados na tabela 5.5, onde o número de linhas e colunas são referentes aos problemas após a fase de pré-processamento. Esses problemas foram selecionados como uma forma de identificar classes de problemas onde a família de algoritmos adaptada ao PCx original e modificado tem um bom desempenho. Todos os problemas testes são de domínio público. Eles são encontrados em NETLIB [7], MISC [33, 34]. Os modelos QAP são da coleção QAPLIB [4] com a letra M ao final quando a modificação descrita em [39] é aplicada.

Tabela 5.1: Problemas Testes

Problema	Linhas	Colunas	Nnulos	Coleção
SCR20	5079	15980	61780	QAP
CHR25A	8149	15325	53725	QAP
CHR22B	5587	10417	36520	QAP
SCR15	2234	6210	24060	QAP
NUG06-3rd	3972	4686	19670	QAP

Tabela 5.2: Continuação da Tabela Problemas Testes

Problema	Linhas	Colunas	Nnulos	Coleção
NUG05-3rd	1410	1425	5523	QAP
NUG07	602	931	3318	QAP
NUG05	210	225	740	QAP
NUG12M	2794	8856	33528	MISC
STOCFOR3	15362	22228	62960	NETLIB
FIT2P	3000	13525	50284	NETLIB
DFL001	5984	12143	35274	NETLIB
80BAU3B	2140	11066	21631	NETLIB
FIT2D	25	10524	129042	NETLIB
TRUSS	1000	8806	27836	NETLIB
MAROS-R7	2152	7440	100486	NETLIB
PILOT87	1971	6373	72163	NETLIB
D2Q06C	2132	5728	31965	NETLIB
D6CUBE	403	5444	34233	NETLIB
PILOT	1368	4543	41879	NETLIB
WOODW	708	5364	19809	NETLIB
SHIP12L	610	4171	9254	NETLIB
GREENBEA	1933	4164	23765	NETLIB
GREENBEB	1932	4154	23673	NETLIB
BNL2	1964	4008	14037	NETLIB
SCTAP3	1408	3268	9383	NETLIB
SHIP08L	470	3121	7122	NETLIB
NESM	654	2922	13244	NETLIB
STOCFOR2	1980	2868	8090	NETLIB
PILOTWE	701	2814	8841	NETLIB
CZPROB	671	2779	5531	NETLIB
CYCLE	1420	2773	15004	NETLIB
SCSD8	397	2750	8584	NETLIB

Tabela 5.3: Continuação da Tabela Problemas Testes

SIERRA	1212	2705	7771	NETLIB
DEGEN3	1501	2604	25425	NETLIB
SCTAP2	1033	2413	7052	NETLIB
PILOTNOV	848	2123	11922	NETLIB
MAROS	846	1966	6634	NETLIB
SHIP12S	340	1943	4297	NETLIB
SHIP04L	292	1905	4290	NETLIB
25FV47	798	1854	10538	NETLIB
PILOTJA	810	1810	11417	NETLIB
WOOD1P	171	1718	44575	NETLIB
SCFXM3	915	1704	8206	NETLIB
FIT1P	627	1677	9868	NETLIB
SHIP08S	276	1604	3644	NETLIB
MODSZK1	665	1599	3065	NETLIB
GANGES	1113	1510	6537	NETLIB
BNL1	610	1491	5256	NETLIB
SHELL	487	1451	2904	NETLIB
PEROLD	593	1389	5636	NETLIB
SCSD6	147	1350	4316	NETLIB
SHIP04S	216	1281	2875	NETLIB
SCRS8	421	1199	3036	NETLIB
STANDMPS	422	1192	2831	NETLIB
SCFMX2	610	1136	4919	NETLIB
GFRDPNC	590	1134	2393	NETLIB
FIT1D	24	1049	13427	NETLIB
PILOT4	396	1022	5001	NETLIB
GROW22	440	946	8252	NETLIB
FINNIS	438	935	2332	NETLIB
SEBA	448	901	8252	NETLIB
FFFFF800	322	826	5164	NETLIB

Tabela 5.4: Continuação da Tabela Problemas Testes

STANDATA	314	796	1403	NETLIB
STANDGUB	314	796	1403	NETLIB
SCSD1	78	760	2388	NETLIB
DEGEN2	442	757	4167	NETLIB
AGG2	514	750	4558	NETLIB
AGG3	514	750	4574	NETLIB
BOEING1	331	697	3104	NETLIB
SCAGR25	469	669	1715	NETLIB
ETAMACRO	334	669	1995	NETLIB
GROW15	300	645	5620	NETLIB
SCTAP1	284	644	1802	NETLIB
SCFXM1	305	568	2732	NETLIB
TUFF	257	567	4095	NETLIB
STAIR	356	532	3813	NETLIB
AGG	390	477	2055	NETLIB
FORPLAN	121	447	4415	NETLIB
CAPRI	241	436	1528	NETLIB
E226	198	429	2515	NETLIB
SCORPION	340	412	1534	NETLIB
BANDM	240	395	1894	NETLIB
LOTFI	133	346	867	NETLIB
ISRAEL	174	316	2443	NETLIB
SC205	203	315	663	NETLIB
GROW7	140	301	2612	NETLIB
BOEING2	125	264	979	NETLIB
SHARE1B	112	248	1148	NETLIB
BRANDY	133	238	1911	NETLIB
SCAGR7	127	183	455	NETLIB
BEACONFD	86	171	1316	NETLIB
SC105	104	162	339	NETLIB

Tabela 5.5: Continuação da Tabela Problemas Testes

SHARE2B	96	162	777	NETLIB
STOCFOR1	102	150	421	NETLIB
BORE3D	81	138	549	NETLIB
ADLITTLE	55	137	417	NETLIB
RECIPE	64	123	443	NETLIB
VTPBASE	72	111	283	NETLIB
BLEND	71	111	477	NETLIB
SC50A	49	77	159	NETLIB
SC50B	48	76	146	NETLIB
KB2	43	68	313	NETLIB
AFIRO	27	51	102	NETLIB

5.4 Comparação do Desempenho para Diversos Valores de p Usando Matlab

Neste experimento usamos os problemas E226, SCFMX2, SCSD6, SCSD8, DEGEN3 e 25FV47 da biblioteca Netlib. Realizamos 100 iterações do algoritmo de ajustamento ótimo para p coordenadas, para os casos $p = 4$, $p = 10$ e $p = 20$ e comparamos o desempenho destes. Os resultados são mostrados na tabela abaixo. As notações são as seguintes: $||b^0||$ representa o resíduo inicial, $||b^{100}||$ representa o resíduo na centésima iteração, (desemp) representa a redução relativa do resíduo.

Tabela 5.6: Comparação do desempenho inicial para p=4,10,20

E226		SCFMX2		SCSD6	
$\|\|b^0\|\| = 0,0341$	$\|\|b^{100}\|\|$	$\|\|b^0\|\| = 0,0204$	$\|\|b^{100}\|\|$	$\|b^0\|\| = 0,3575$	$\|b^{100}\|\|$
$p = 4$	0,0096	$p = 4$	0,0092	p=4	0,0327
$p = 10$	0,0041	$p = 10$	0,0068	p=10	0,0167
$p = 20$	0,0024	$p = 20$	0,0035	p=20	0,0068
algoritmos	desemp(%)	algoritmos	desemp(%)	algoritmos	desemp(%)
$p = 4$ ×p=10	134%	$p = 4$ ×p=10	44%	p=4 ×$p = 10$	96%
$p = 4$ ×p=20	300 %	$p = 4$ ×p=20	163 %	p=4 ×$p = 20$	380 %
$p = 10$ ×p=20	71%	$p = 10$ ×p=20	94%	p=10 ×$p = 20$	145%
SCSD8		DEGEN3		25FV47	
$\|\|b^0\|\| = 0,3332$	$\|\|b^{100}\|\|$	$\|\|b^0\|\| = 0,0651$	$\|\|b^{100}\|\|$	$\|b^0\|\| = 0,0396$	$\|b^{100}\|\|$
$p = 4$	0,0176	$p = 4$	0,0082	p=4	0,0079
$p = 10$	0,0120	$p = 10$	0,0060	p=10	0,0043
$p = 20$	0,0078	$p = 20$	0,0055	p=20	0,0038
algoritmos	desemp(%)	algoritmos	desemp(%)	algoritmos	desemp(%)
$p = 4$ ×p=10	47%	$p = 4$ ×p=10	37%	p=4 ×$p = 10$	84%
$p = 4$ ×p=20	126 %	$p = 4$ ×p=20	49 %	p=4 ×$p = 20$	108 %
$p = 10$ ×p=20	54%	$p = 10$ ×p=20	9%	p=10 ×$p = 20$	13%

Os seis experimentos indicam que aumentando o valor de p, o desempenho do algoritmo melhora para raio de convergência inicial, ou seja, que conseguimos reduzir mais rapidamente o resíduo $||b^k||$ do problema auxiliar no raio de convergência inicial. Na realidade os seis experimentos confirmam o Teorema 4.2 para o raio de convergência inicial.

5.5 Experimento em Conjunto com o Código PCx Modificado

Nesta seção vamos comparar o comportamento PCx modificado auxiliado pelo algoritmo simples (PCxMA$_p$), com $p = 2$, $p = 10$ e $p = 20$ e o PCx modificado (PCxM). Escolhemos 10 problemas em que o PCx alterado como em [45] não consegue obter uma solução e mais 12 problemas onde o PCx modificado necessita de segunda fase. Para os problemas onde

o PCx não necessita de segunda fase, não ocorreram melhoras significativas em relação a abordagem anterior, assim optamos por não apresentar estes experimentos.

Antes de apresentarmos os experimentos é necessário destacar uma observação relevante sobre a forma como o algoritmo de ajustamento ótimo para p coordenadas trabalha. Como vimos na seção 4.1, em cada iteração o algoritmo modifica p coordenadas e ajusta as demais em função destas. Entre as p coordenadas modificadas algumas podem ser nulas, entretanto no momento de transferir estas coordenadas ao método de pontos interiores, se elas fizerem parte das variáveis primais (vetor x e s) ou das variáveis duais de folga (vetor z e w) elas não podem ser nulas, então adotamos valores pequenos para elas, visto que provavelmente elas devem ser nulas em uma solução. Nos experimentos a seguir usamos o valor 10^{-5} para estas variáveis. Para os problemas onde o PCx modificado não consegue obter uma solução, a precisão que usamos para resolver o subproblema (4.8) é de 10^{-17} para complementariedade relativa e 10^{-14} para o restante das equações que compõe as condições KKT e para os problemas onde o PCx modificado necessita de segunda fase, a precisão maxima foi de 10^{-18} para complementariedade relativa e 10^{-15} para o restante das equações que compõe as condições KKT. O número máximo permitido de iterações do método de pontos interiores para resolução do subproblema é de 200.

No decorrer desta seção vamos utilizar as seguintes notações nas tabelas: (MPI*) indica o número de iterações que o PCx modificado necessita para detectar que o problema não converge, iteração melhorada (itMelh) é onde realizamos iterações com o algoritmo de ajustamento ótimo para p coordenadas, número de iterações auxiliares em cada iteração melhorada (itAux) é a quantidade de iterações realizadas com o algoritmo de ajustamento ótimo para p coordenadas, (MPI_p) indica o número total de iterações do PCx modificado auxiliado pelo algoritmo simples para $p = 2, 10$ e 20, (T) indica o tempo total para a solução dos problemas em segundos, (MPI) indica o número de iterações que o PCx modificado necessita para detectar que o problema converge, (Itgc) indica o número de iterações internas do método dos gradientes conjugados nas direções preditora e corretora após o auxílio do algoritmo simples e (.) indica a iteração da mudança de fase.

5.5.1 Problemas que não Convergem na Abordagem Iterativa com Precondicionador Híbrido

Para os problemas onde a abordagem híbrida não consegue obter uma solução, o experimento foi realizado da seguinte forma: nos problemas onde o método necessita de segunda fase foram realizadas algumas iterações com algoritmo simples com $p = 2$, $p = 10$ e $p = 20$ na faixa crítica da mudança de fase e devolvido um ponto melhorado para o método prosseguir.

A justificativa para esta estratégia é que para algumas classes de problemas o precondicionador fatoração controlada de Cholesky perde eficiência e o método muda de fase, no entanto, o precondicionador separador ainda não esta preparado para assumir o processo e assim o método não converge.

Para os problemas que o método não necessita de segunda fase, na iteração onde o gap ou a infactibilidade primal ou dual aumenta, realizamos iterações com o algoritmo de ajustamento ótimo para p coordenadas com $p = 2$, $p = 10$ e $p = 20$ e devolvemos um ponto melhorado para o método de pontos interiores prosseguir. Se mesmo assim, o método de pontos interiores não encontra uma solução para o problema, então mudamos a estratégia e realizamos algumas iterações do algoritmo de ajustamento para p coordenadas com $p = 2$, $p = 10$ e $p = 20$ nas iterações iniciais do método de pontos interiores e devolvemos um ponto melhorado.

A justificativa para usarmos a primeira estratégia é que se o gap ou a infactibilidade primal ou dual aumenta, provavelmente o método de pontos interiores está tomando um caminho que não é muito bom, então fazendo algumas iterações com o algoritmo para p coordenadas podemos alterar a sequência gerada pelo método de pontos interiores para um caminho que o faça convergir. Se a primeira tática falhar, então mudamos a estratégia e tentamos alterar o caminho do método de pontos interiores nas iterações iniciais. Os experimentos foram realizados sempre procurando fazer o mínimo de iterações possíveis do algoritmo simples.

Apresentamos na tabela 5.7 uma comparação dos resultados obtidos com e sem o auxílio do algoritmo para p coordenadas.

Tabela 5.7: Problemas que não convergem na abordagem híbrida

Nome	MPI*	itMelh	itAux	MPI_2	T	MPI_{10}	T	MPI_{20}	T
NUG05	7(3)	4	50	-	-	6(3)	0.10	6(3)	0.14
NUG05-3rd	7(3)	3	50	-	-	9(3)	6.14	7(3)	3.72
NUG06-3rd	9(4)	2 a 4	50	-	-	-	-	12(-)	26.9
FINNIS	25(-)	16	100	-	-	34	0.80	27	0.73
AGG2	26(-)	13	50	-	-	31	1.70	29	1.15
AGG3	21(-)	12	100	47	2.9	39	1.45	36	1.85
SCRS8	23(-)	2 a 8	50	-	-	-	-	38	2.04
BNL2	37(-)	11 a 12	300	-	-	47	14.1	40	25.1
D2Q06C	16(-)	-	-	-	-	-	-	-	-
80BAU3B	46(-)	-	-	-	-	-	-	-	-

Com esta abordagem conseguimos obter uma solução para os problemas AGG2, AGG33, SCRS8, BNL2, FINNIS, NUG05, NUG05-3rd e NUG06-3rd. Os resultados da tabela indicam que o algoritmo simples com $p = 20$ é superior ao algoritmo simples com $p = 2$ e $p = 10$, ou seja, aumentando o valor de p o desempenho do algoritmo melhora para esta classe de problemas. Em [45] para que os problemas NUG05, NUG05-3rd e NUG06-3rd convergissem, o parâmetro η foi alterado e um novo parâmetro foi calculado explorando a estrutura dos problemas, já em nossa abordagem o parâmetro η original foi mantido.

5.5.2 Problemas que o PCx Modificado Necessita de Segunda Fase

Como no PCx modificado existe uma faixa crítica na troca dos precondicionadores, no sentido que, se mudarmos de etapa antes desta faixa ou no início dela, o precondicionador separador ainda não esta preparado para assumir o trabalho e assim o método pode perder robustez. Então, para os problemas onde o PCx modificado necessita de segunda fase, a estratégia utilizada foi fazer algumas iterações do algoritmo simples com $p = 2$, $p = 10$ e $p = 20$ nesta faixa crítica e devolver um ponto melhorado para o método de pontos interiores prosseguir. Os experimentos foram realizados sempre procurando fazer o mínimo de iterações do algoritmo simples.

Na tabela (5.8) apresentamos uma comparação do desempenho do PCx modificado e do PCx modificado com auxílio do algoritmo simples com $p = 20$.

Tabela 5.8: PCxM $\times$ PCxMA$_{20}$

nome	MPI	Itgc	T	itMelh	itAux	MPI$_{20}$	Itgc	T
E226	32(8)	5202	0.64	7	50	20(12)	655	0.50
MAROS	40(9)	28282	8.62	9	30	26(9)	15778	7.14
25FV47	29(17)	10231	6.25	17	50	28(17)	6814	5.30
SCR15	24(11)	4860	30.63	13	100	23(11)	3997	28.7
CHR22b	29(16)	2880	80.0	15	100	28(16)	10855	99.80
CHR25a	29(15)	12698	181.4	14-15	50	28(15)	17362	219.2
NUG07	11(4)	465	0.72	6	30	10(4)	729	0.79
NUG12M	20(7)	9372	232.8	8	50	20(7)	7351	209.7
SCR20	21(15)	5207	244.6	16	20	21(15)	4284	241.4
STOCFOR2	21(15)	2025	3.98	17	30	21(15)	747	3.78
DEGEN3	16(8)	1426	19.91	9	50	16(8)	1126	19.2
DEGEN2	12(5)	598	0.71	5	30	12(5)	409	0.69

Ao usarmos o algoritmo simples para 20 coordenadas em conjunto com o PCx modificado, conseguimos reduzir o número de iterações deste, para uma classe de problemas e para outra classe conseguimos reduzir o número de iterações internas dos gradientes conjugados, tornando o método mais robusto para estas classes de problemas.

No problema E226 ocorre que ao usarmos o algoritmo simples com $p = 20$ antes da troca, ele prolonga a vida útil do precondicionador fatoração controlada de Cholesky, o que é bom porque o precondicionador separador possui um melhor desempenho perto da solução. No problema 25FV47 ocorre que o algoritmo simples devolve um ponto mais próximo da solução, e pelo mesmo motivo acima esta abordagem possui um bom desempenho.

De uma forma geral podemos dizer que existem dois tipos de problemas em que esta abordagem possui um bom desempenho, o primeiro tipo é a que ela prolonga a vida útil do precondicionador fatoração controlada de Cholesky e o segundo tipo é aquele que usamos o algoritmo simples após a mudança de fase e ele devolve um ponto suficientemente próximo da solução, para que o precondionador separador tenha um melhor desempenho.

Na tabela 5.9 exibimos uma comparação do algoritmo simples com $p = 2$, $p = 10$ e $p = 20$

para os mesmos problemas da tabela 5.8 e nas mesmas condições.

Tabela 5.9: Comparação do desempenho para p=2,10,20

nome	MPI_2	Itgc	T	MPI_{10}	Itgc	T	MPI_{20}	Itgc	T
E226	22	1014	0.28	22	853	0.25	20	655	0.50
MAROS	31	16433	6.3	24	13484	5.26	26	15778	7.14
25FV47	33	13322	7.86	28	6216	4.9	28	6814	5.30
SCR15	24	4967	31.69	24	4021	30.20	23	3997	28.70
CHR22b	29	10802	101.9	28	11155	102.1	28	10855	99.80
chr25a	28	17697	220.4	29	19308	222.2	28	17362	219.2
NUG07	11	565	0.82	11	619	0.85	10	729	0.79
NUG12M	20	8242	221.3	20	9214	230.1	20	7351	209.7
SCR20	-	-	-	21	4621	242.7	21	4284	241.4
STOCFOR2	21	813	3.79	21	784	3.80	21	747	3.78
DEGEN3	16	1288	19.7	16	1413	19.15	16	1126	19.20
DEGEN2	12	477	0.65	12	518	0.67	12	409	0.69

Novamente podemos observar que aumentando o valor de p o desempenho algoritmo melhora, visto que, na maioria dos problemas da tabela acima, o algoritmo simples com $p = 20$ agregado ao PCx modificado ganha no número de iterações e no número de iterações internas do método dos gradientes conjugados.

5.6 Ponto Inicial

O ponto inicial para método de pontos interiores influencia muito em seu desempenho. Como a família de algoritmos possui um custo computacional barato por iteração, uma idéia natural é utilizar a família de algoritmos para encontrar um ponto inicial para métodos de pontos interiores ou melhorar o ponto inicial já dado. Nesta seção vamos usar a família de algoritmos para melhorar o ponto inicial do código PCx original.

5.6.1 O Ponto Inicial do Código PCx Original

O ponto inicial do código PCx original [8] é descrito por Mehrotra em [32]. Primeiramente Mehrotra utiliza mínimos quadrados para calcular os pontos

$$\begin{aligned} \tilde{y} &= (AA^t)^{-1}Ac, \\ \tilde{z} &= c - A^t\tilde{y}, \\ \tilde{x} &= A^t(AA^t)^{-1}b. \end{aligned} \tag{5.1}$$

Depois são calculados os valores δ_x e δ_z

$$\begin{aligned} \delta_x &= \max(-1.5\min\{x_i\}, 0), \\ \delta_z &= \max(-1.5\min\{z_i\}, 0). \end{aligned} \tag{5.2}$$

Os valores δ_x e δ_z são encontrados de forma que os pontos $\tilde{x} + \delta_x$ e $\tilde{z} + \delta_z$ sejam maiores que zero. Então são calculados

$$\tilde{\delta}_x = \delta_x + 0.5\frac{(x + \delta_x e)^t(z + \delta_z e)}{\sum_{i=1}^{n}(z_i + \delta_z)}, \tag{5.3}$$

$$\tilde{\delta}_z = \delta_z + 0.5\frac{(x + \delta_x e)^t(z + \delta_z e)}{\sum_{i=1}^{n}(x_i + \delta_x)}. \tag{5.4}$$

Finalmente o ponto inicial é dado por

$$\begin{aligned} y^0 &= \tilde{y}, \\ s^0 &= \tilde{s} + \tilde{\delta}_s e, \\ x^0 &= \tilde{x} + \tilde{\delta}_x e. \end{aligned} \tag{5.5}$$

Os valores $\tilde{\delta}_x$ e $\tilde{\delta}_z$ são encontrados de maneira que os pontos x^0 e z^0 sejam maiores que zero e centralizados.

De uma forma geral podemos ver que Mehrotra encontra o ponto inicial em três passos.

- Primeiro, encontra os pontos:

$$\begin{aligned} \tilde{y} &= (AA^t)^{-1}Ac, \\ \tilde{z} &= c - A^t\tilde{y}, \\ \tilde{x} &= A^t(AA^t)^{-1}b. \end{aligned} \tag{5.6}$$

- Segundo, calcula os valores δ_x e δ_z de modo que os pontos $\tilde{x} + \delta_x$ e $\tilde{z} + \delta_z$ sejam maiores que zero.

- Terceiro, calcula os valores $\tilde{\delta}_x$ e $\tilde{\delta}_z$ de maneira a centralizar os pontos $x^0 = \tilde{x} + \delta_x$ e $z^0 = \tilde{z} + \delta_z$.

5.6.2 Experimento em Conjunto com o Código PCx Original

A estratégia utilizada para realizar este experimento foi fazer algumas iterações com o algoritmos simples com $p = 20$ antes da centralização dos pontos. A justificativa para esta estratégia é que se usarmos a família de algoritmos simples depois de centralizar os pontos melhoramos estes, mas podemos perder sua centralidade que é importante para os métodos de pontos interiores.

Nos experimentos, usamos uma heurística dependente do número de colunas do problema para determinar o número máximo de iterações auxiliares do algoritmos simples usadas em cada problema. Esta heurística foi desenvolvida baseada em experimentos numéricos e é dada por $\max(4, 2\min(16, n/200))$, onde n é o número de colunas do problema. Utilizamos 95 problemas da coleção NETLIB. Na tabela (5.10) apresentamos uma comparação do desempenho do PCx original e do PCx modificado com auxílio do algoritmo simples com $p = 20$. As notações são as seguintes: (itAux) é a quantidade de iterações realizadas com o algoritmo de ajustamento ótimo para p coordenadas, (PCxori) indica o número total de iterações do PCx original, (PCxoriM) indica o número total de iterações do PCx original modificado auxiliado pelo algoritmo simples com $p = 20$.

Tabela 5.10: PCxori × PCxoriM

nome	itAux	PCxori	PCxoriM
25FV47	18	25	25
80BAU3B	32	36	35
ADLITTLE	4	11	11
AFIRO	4	7	7
AGG	4	18	18
AGG2	7	21	21
AGG3	7	19	19
BANDM	4	16	16
BEACONFD	4	10	10
BLEND	4	9	9
BNL1	14	33	33
BNL2	32	33	33

Tabela 5.11: Continuação da tabela PCxori × PCxoriM

nome	itAux	PCxori	PCxoriM
BOEING1	6	18	19
BOEING2	4	12	12
BORE3D	4	15	19
BRANDY	4	17	17
CAPRI	4	17	17
CYCLE	27	23	23
CZPROB	27	26	26
D2Q06C	32	27	27
D6CUBE	32	17	17
DEGEN2	7	11	11
DEGEN3	26	29	15
DFL001	32	44	39
E226	4	18	18
ETAMACRO	6	26	23
FFFFF800	8	29	26
FINNIS	9	22	22
FIT1D	10	17	17
FIT1P	16	16	16
FIT2D	32	22	22
FIT2P	32	20	20
FORPLAN	4	21	21
GANGES	15	16	16
GFRDPNC	11	16	16
GREENBEA	32	49	48
GREENBEB	32	39	38

Tabela 5.12: Continuação da tabela PCxori × PCxoriM

nome	itAux	PCxori	PCxoriM
GROW15	6	19	19
GROW22	9	21	21
GROW7	4	16	15
ISRAEL	4	19	20
KB2	4	11	12
LOTFI	4	13	17
MAROS	14	18	19
MAROS-R7	32	12	13
MODSZK1	15	20	32
NESM	29	25	25
PEROLD	13	32	35
PILOT	32	34	35
PILOT4	10	46	43
PILOT87	32	25	24
PILOTJA	18	30	30
PILOTWE	28	45	44
PILOTNOV	21	16	16
RECIPE	4	8	8
SC105	4	9	9
SC205	4	10	16
SC50A	4	7	7
SC50B	4	6	6
SCAGR25	6	17	17
SCAGR7	4	13	13
SCFXM1	5	17	16
SCFXM2	11	19	19
SCFXM3	17	19	18
SCORPION	4	11	11
SCRS8	11	21	22

Tabela 5.13: Continuação da tabela PCxori × PCxoriM

nome	itAux	PCxori	PCxoriM
SCSD1	7	8	8
SCSD6	13	11	12
SCSD8	27	10	11
SCTAP1	6	14	14
SCTAP2	24	12	12
SCTAP3	32	14	14
SEBA	9	13	14
SHARE1B	4	18	18
SHARE2B	4	17	16
SHELL	14	20	20
SHIP04L	19	13	12
SHIP04S	12	12	13
SHIP08L	31	14	14
SHIP08S	16	11	11
SHIP12L	32	15	14
SHIP12S	19	12	12
SIERRA	27	18	18
STAIR	5	13	16
STANDATA	7	13	12
STANDGUB	7	13	12
STANDMPS	11	23	24
STOCFOR1	4	11	11
STOCFOR2	28	19	19
STOCFOR3	32	30	29
TRUSS	32	17	19
TUFF	5	18	18
VTPBASE	4	10	11
WOOD1P	17	22	22
WOODW	32	30	28

Com essa abordagem, para 20 problemas conseguimos diminuir o número de iterações do código PCx original. A quantidade de iterações auxiliares utilizada do algoritmo simples foi pequena. Para 55 problemas o número de iterações permaneceram inalteradas e para 20 problemas aumentaram as iterações do PCx original. Dos 95 problemas analisados, 28 possuem mais de 2000 variáveis. Desses, conseguimos diminuir o número de iterações em 10 problemas, ou seja, em 36% deles e perdemos apenas em 4 problemas. O número total de iterações para esses 28 problemas utilizando o PCx original foi 701 e usando essa abordagem foi 678, uma redução de 23 iterações. Desta forma o experimento indica que essa abordagem tornou a implementação mais robusta para os problemas maiores dessa coleção.

CAPÍTULO 6

Conclusões e Trabalhos Futuros

Apresentamos uma nova família de algoritmos para programação linear. A grande vantagem desta família de algoritmos é a sua simplicidade e seu raio de convergência inicial rápido.

Em cada iteração dos algoritmos desta família precisamos resolver um subproblema. No caso $p = 2$, que é o algoritmo de ajustamento pelo par ótimo, Gonçalves, Storer e Gondzio resolvem este subproblema verificando as condições KKT, mais precisamente, ele verifica todas as possíveis soluções factíveis das equações KKT, que neste caso são 7. No caso geral que é o algoritmo para p coordenadas, se resolvermos o subproblema seguindo o mesmo raciocínio, o número de casos possíveis de soluções factíveis cresce exponencialmente com o valor de p e este número de casos é exatamente $2^{p+1} - 1$. Contornamos este problema resolvendo as equações KKT do subproblema por métodos de pontos interiores. Resolver o subproblema usando método de pontos interiores é bem vantajoso, visto que, por exemplo, se quisermos usar o algoritmo com $p = 10$ é muito mais fácil fatorar uma matriz definida positiva de ordem 11×11 do que verificar $2^{11} - 1$ casos possíveis.

Uma das propriedades que estamos explorando é o raio de convergência inicial rápido da família de algoritmos. Os resultados numéricos indicam que na prática aumentando o valor de p o desempenho do algoritmo melhora confirmando o Teorema 4.2 para o raio de convergência inicial.

Ao usarmos a família de algoritmos simples em conjunto com um método de pontos interiores infactível em uma classe de problemas, onde o método não converge, conseguimos fazer o método convergir para quase todos os problemas desta classe. Para a maioria dos

problemas a quantidade de iterações auxiliares utilizada do algoritmo simples foi pequena, visto que, o seu custo por iteração é baixo. Para que os problemas NUG05, NUG05-3rd e NUG06-3rd convergissem em [45], o parâmetro η foi alterado e um novo parâmetro foi calculado explorando a estrutura dos problemas, já em nossa abordagem o parâmetro η original foi mantido. Tornando a implementação menos sensível ao ajuste de parâmetros.

Para outra classe de problemas onde o método de pontos interiores infactível necessita de segunda fase, com a abordagem utilizada conseguimos reduzir o número de iterações do método para uma subclasse desta e, conseguimos reduzir o número de iterações internas do método dos gradientes conjugados para outra subclasse. Novamente para a maioria dos problemas a quantidade de iterações auxiliares utilizada do algoritmo simples foi pequena, e tendo em vista que, o seu custo por iteração é baixo o tempo computacional utilizado pelo novo algoritmo não é significativo em relação a uma iteração dos métodos de pontos interiores.

Os resultados numéricos indicam que aumentando o valor de p, o desempenho da família de algoritmos simples melhora, tanto para classe de problemas que o método de pontos interiores infactível não consegue obter uma solução, quanto para a classe de problemas que o método necessita da segunda fase.

Em particular pudemos identificar dois tipos de problemas onde a abordagem utilizada possui um melhor desempenho. O primeiro tipo é a que ela prolonga a vida útil do precondicionador fatoração controlada de Cholesky e o segundo tipo é aquele em que usamos o algoritmo simples após a mudança de fase e ele devolve um ponto suficientemente próximo da solução, para que o precondionador separador tenha um melhor desempenho. Um trabalho futuro que pode ser explorado é montar um conjunto de problemas de grande porte do primeiro e segundo tipo e fazer testes neste conjunto.

Ao utilizarmos a família de algoritmos de um outro modo, melhorando o ponto inicial de um método de pontos interiores, conseguimos diminuir o número de iterações deste para 20 problemas. A quantidade de iterações auxiliares utilizada do algoritmo simples foi pequena. Para 55 problemas o número de iterações permaneceram inalteradas e para 20 problemas aumentaram as iterações. Dos 95 problemas analisados, 28 possuem mais de 2000 variáveis. Desses, conseguimos diminuir o número de iterações em 10 problemas, ou seja, em 36% deles e perdemos apenas em 4 problemas. O número total de iterações para esses 28 problemas utilizando o PCx original foi 701 e usando essa abordagem foi 678, uma redução de 23 iterações.

Podemos concluir que os resultados numéricos indicam que ao usarmos a família de algoritmos simples em conjunto com métodos de pontos interiores infactível, melhoramos sua robustez e portanto seu desempenho.

Problemas de grande porte possuem um custo computacional relativamente alto em cada iteração de métodos de pontos interiores. Assim uma pesquisa futura pode ser utilizar a família de algoritmo simples em cada iteração de pontos interiores para reduzir o custo computacional desta.

Finalizando, ao usarmos a família de algoritmos simples em cada iteração de um método de pontos interiores podemos mudar o algoritmo simples de uma iteração para outra aumentando ou diminuindo valor de p. Um trabalho futuro pode ser explorado é determinar o valor de p dinamicamente em função dos resultados da iteração anterior.

APÊNDICE A

Apêndice

A.1 Multiplicadores de Lagrange

Nesta seção, analisaremos o problema geral de minimizar uma função $f(x)$ sujeita a uma ou mais restrições de igualdade. A única exigência sobre $f(x)$ é que ela seja, no mínimo, duas vezes diferenciável. Assim, consideremos o seguinte problema de otimização:

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & g(x) = 0 \\ & x \geq 0 \end{array} \tag{A.1}$$

em que $g(x) = 0$ corresponde a um conjunto de m restrições.

O gradiente de $f(x)$, denotado por ∇f, é um vetor que aponta na direção de maior crescimento de $f(x)$. Se o problema de otimização fosse irrestrito bastaria fazermos $\nabla f(x) = 0$ para determinarmos os pontos críticos de $f(x)$ e, o mínimo, se existisse, deveria pertencer a este conjunto (de pontos críticos). Para o caso particular em que $m = 1$, devido a presença da restrição isolada $g(x) = 0$, o gradiente de $g(x)$ deve ser ortogonal ao conjunto de soluções factíveis $F = \{x; g(x) = 0\}$, ou seja, $\nabla g(x) \cdot F = 0$, para cada $x \in F$. Assim, para que um ponto x^* seja um ponto crítico é necessário que ele pertença ao conjunto factível F, $g(x^*) = 0$ e que $\nabla f(x^*)$ seja proporcional a $\nabla g(x^*)$, ou seja,

$$g(x^*) = 0 \tag{A.2}$$

$$\nabla f(x^*) = \lambda \nabla g(x^*) \tag{A.3}$$

em que $\lambda \in \mathbb{R}$ é a constante de proporcionalidade conhecida como multiplicador de Lagrange.

No caso em que $m > 1$, a região de factibilidade F é a intersecção de m hiperplanos e o sistema constituído pelas equações (A.2) e (A.3) deve ser reescrito como:

$$g(x^*) = 0 \tag{A.4}$$

$$\nabla f(x^*) = \sum_{i=1}^{m} \lambda_i \nabla g(x^*). \tag{A.5}$$

Em termos formais é possível obter o sistema de equações anteriores introduzindo a seguinte função, conhecida como função Lagrangeana:

$$L(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i g_i(x). \tag{A.6}$$

Derivando (A.6) parcialmente com relação a x e λ_i e igualando a zero, temos:

$$\frac{\partial L}{\partial x_j} = \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i}{\partial x_j} = 0, \;\; j = 1, 2, \ldots, n \tag{A.7}$$

$$\frac{\partial L}{\partial \lambda_i} = -g_i(x) = 0, \;\; i = 1, 2, \ldots, m. \tag{A.8}$$

Em notação vetorial observemos que as equações (A.7) e (A.8) correspondem exatamente às equações (A.4) e (A.5). Estas equações são conhecidas como condições de otimalidade de primeira ordem. Determinar se uma solução das equações (A.7) e (A.8) é um mínimo global do problema (A.1) pode ser algo muito difícil. Todavia, se as restrições forem todas lineares, o Teorema A.1, enunciado a seguir indica um meio de calcular um mínimo local de $f(x)$.

Teorema A.1. *Se o conjunto de restrições for linear, um ponto crítico x^* é um mínimo local se*

$$\xi' H f(x^*) \xi > 0 \tag{A.9}$$

para todo $\xi \neq 0$ satisfazendo

$$\xi' \nabla g_i(x^*) = 0, \quad i = 1, 2, \ldots, m \tag{A.10}$$

em que

$$H f(x^*) = \left[\frac{\partial^2 f}{\partial x_i^* \partial x_j^*} \right] \tag{A.11}$$

é a matriz Hessiana de f no ponto x^*.

A versão deste teorema para o máximo está enunciado em Vanderbei [44].

A.2 O Problema da Barreira Logarítmica

Métodos de pontos interiores são intimamente relacionados com métodos de barreira, que foram desenvolvidos por Fiacco e McCormick [15] no final da década de 1960. A idéia do método está baseada na resolução aproximada de uma sequência de subproblemas sem qualquer restrição de desigualdade. Sob certas condições, a sequência de soluções aproximadas converge para a solução do problema original.

Consideremos o seguinte problema de maximização com restrições:

$$\begin{array}{ll} \text{maximizar} & f(x) = c^t x \\ \text{sujeito a} & Ax + w = b \\ & x, w \geq 0 \end{array} \tag{A.12}$$

Neste problema, podemos remover a restrição de não negatividade para cada variável adicionando-a na função objetivo $f(x)$, uma função que é escolhida de tal forma que se aproxima do infinito quando a variável em questão aproxima-se de zero. A função mais simples com esta propriedade é o logaritmo. Portanto, para cada variável, introduzimos um novo termo na função objetivo que corresponde a uma constante multiplicada pelo logaritmo da variável, ou seja:

$$\begin{array}{ll} \text{maximizar} & f(x) + \mu \sum_j \log x_j + \mu \sum_i \log w_i \\ \text{sujeito a} & Ax + w = b \end{array} \tag{A.13}$$

em que o parâmetro μ é maior que zero.

O problema (A.13) é chamado de problema da barreira logarítmica associado com (A.12). Observemos que ele não é realmente um problema, mas uma família de problemas associada ao parâmetro μ. Cada um destes problemas é um problema de programação não linear porque a função objetivo é não linear. Esta função não linear é conhecida como função barreira logarítmica.

Se o problema original for um problema de programação linear, então, a região factível F é um poliedro, com cada face sendo caracterizada pela propriedade que, nela, cada uma das variáveis é nula. Portanto, a função barreira tende a menos infinito sobre cada face do poliedro. Além disto, ela é finita no interior do poliedro e se aproxima de menos infinito nas proximidades do contorno. Para cada $\mu > 0$, o máximo é atingido em um ponto interior e quando μ está próximo de zero este ponto interior move-se na direção da solução ótima do problema original. Visto como uma função de μ o conjunto de soluções ótimas para o

problema da barreira forma um caminho através do interior do poliedro que representa o conjunto das soluções factíveis. Este caminho é denominado caminho central. O objetivo daqui em diante é estudar este caminho central, Vanderbei [44].

No que segue, mostraremos que para cada valor do parâmetro barreira μ existe uma única solução para o problema da barreira. Além disso, se $\mu \to 0$, a solução do problema tende para a solução do problema de programação linear original. O problema (A.13) é um problema de otimização com restrição de igualdade e assim, podemos associar a ele o seu Lagrangeano, ou seja,

$$L(x, w, \lambda) = c^t x + \mu \sum_j \log x_j + \mu \sum_i \log w_i + \lambda^t (b - Ax - w) \tag{A.14}$$

Derivando (A.14) parcialmente com relação a cada variável, obtemos as seguintes condições de otimalidade:

$$\begin{aligned} \frac{\partial L}{\partial x_j} = c_j + \mu \frac{1}{x_j} - \textstyle\sum_i \lambda_i a_{ij} &= 0 \quad j = 1, 2, \ldots, n \\ \frac{\partial L}{\partial w_i} = \mu \frac{1}{w_i} - \lambda_i &= 0 \quad i = 1, 2, \ldots, m \\ \frac{\partial L}{\partial \lambda_i} = b_i - \textstyle\sum_j a_{ij} x_j - w_i &= 0 \quad i = 1, 2, \ldots, m. \end{aligned}$$

Na forma matricial, estas condições de otimalidade são dadas por

$$\begin{aligned} A'\lambda - \mu X^{-1} e &= c \\ \lambda &= \mu W^{-1} e \\ Ax + w &= b \end{aligned} \tag{A.15}$$

em que X e W são matrizes diagonais cujos elementos são, respectivamente, as componentes dos vetores x e w e e denota o vetor que possui todos os elementos iguais a 1.

Definindo $z = \mu X^{-1} e$, as condições de otimalidade (A.15) podem ser reescritas como:

$$\begin{aligned} Ax + w &= b \\ A'\lambda - z &= c \\ z &= \mu X^{-1} e \\ \lambda &= \mu W^{-1} e. \end{aligned}$$

Multiplicando a terceira e quarta equações, respectivamente, por X e W, obtemos finalmente a seguinte forma simétrica primal-dual das condições de otimalidade de primeira

ordem:

$$\begin{aligned} Ax + w &= b \\ A'\lambda - z &= c \\ XZe &= \mu e \\ \Lambda We &= \mu e. \end{aligned} \tag{A.16}$$

A.3 A Existência e Unicidade de Solução para o Problema da Barreira

Uma solução para o problema da barreira nem sempre existe. O Teorema A.2 a seguir, cuja demonstração pode ser encontrada em Vanderbei [44], fornece uma condição necessária e suficiente para a existência de solução para este problema.

Teorema A.2. *Uma solução para o problema da barreira existe se, e somente se, as regiões factíveis para o primal e dual tem interior não vazio.*

Para mostrar que se uma solução existe ela deve ser única, devemos analisar o comportamento da Hessiana da função barreira

$$f(x, w) = c^t x + \mu \sum_j \log x_j + \mu \sum_i \log w_i$$

que é dada por:

$$H = \begin{bmatrix} -\frac{\mu}{x_j^2} & 0 \\ 0 & -\frac{\mu}{w_i^2} \end{bmatrix}, i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, m.$$

Observemos que H é diagonal com elementos estritamente negativos. Portanto, pelo Teorema A.1 existe no máximo um ponto crítico e, se existe, é um máximo global.

O Corolário A.1 a seguir resume estes resultados:

Corolário A.1. *Se um conjunto factível primal (ou seu dual) possui um interior não vazio e é limitado, então para cada $\mu > 0$ existe uma única solução $(x_\mu, w_\mu, \lambda_\mu, z_\mu)$ para as condições de otimalidade de primeira ordem definidas em (A.16).*

O caminho $\{(x_\mu, w_\mu, \lambda_\mu, z_\mu); \mu > 0\}$ é chamado caminho central primal-dual e desempenha um papel importante em métodos de pontos interiores para programação linear.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] I. Adler, M. G. C. Resende, G. Veiga, and N. Karmarkar. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[2] K. M. Anstreicher. Potencial reduction methods. *In T. Terlaky, editor, Interior Point Methods in Mathematical Programming*, Kluwer Academic Publishers, 1996.

[3] S. Bocanegra, F. F. Campos, and A. R. L. Oliveira. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, 36:149–164, 2007.

[4] R. S. Burkard, S. Karisch, and F. Rendl. Qaplib - a quadratic assignment problem library. *European Journal of Operations Research*, pages 55:115–119, 1991.

[5] F. F. Campos. *Analysis of Conjugate Gradients - type methods for solving linear equations*. PhD thesis, Oxford University Computing Laboratory, Oxford, 1995.

[6] F. F. Campos and N. R. C. Birkett. An efficient solver for multi-right hand side linear systems based on the CCCG(η) method with applications to implicit time-dependent partial differential equations. *SIAM J. Sci. Comput.*, 19(1):126–138, 1998.

[7] NETLIB collection LP test sets. Netlib lp repository. *Online at http://www.netlib.org/lp/data*.

[8] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright. PCx an interior point code for linear programming. *Optimization Methods & Software*, 11-2(1-4):397–430, 1999.

[9] G. B. Dantzig. Linear programming and extensions. *Princeton University Press, Princeton, NJ*, 1963.

[10] G. B. Dantzig. Converting a converging algorithm into a polynomially bounded algorithm. Technical report, Stanford University, SOL 91-5, 1991.

[11] G. B. Dantzig. An ϵ-precise feasible solution to a linear program with a convexity constraint in $\frac{1}{\epsilon^2}$ iterations independent of problem size. Technical report, Stanford University, SOL 92-5, 1992.

[12] I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Soviets Math. Doklady*, 8:674–675, 1967.

[13] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29(4):635–657, 1989.

[14] M. Epelman and R. M. Freund. Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. *Mathematical Programing*, 88:451–485, 2000.

[15] A. V. Fiacco and G. P. McCormick. Nonlinear programming: sequential unconstrained minimization techniques. In *John Willey Sons, Inc.*, 1968.

[16] K. R. Frisch. The logarithmic potential method of convex programming. Technical report, University Institute of Economics, Oslo, 1955.

[17] J. P. M. Gonçalves. *A Family of Linear Programming Algorithms Based on the Von Neumann Algorithm*. PhD thesis, Lehigh University, Bethlehem, 2004.

[18] J. P. M. Gonçalves, R. H. Storer, and J. Gondzio. A family of linear programming algorithms based on an algorithm by von neumann. *Aceito para publicação na revista Optimization Methods and Software*, 2007.

[19] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[20] C. C. Gonzaga. Path following methods for linear programming. *SIAM Review*, pages 34(2):167–224, 1992.

[21] P. Huard. Resolution of mathematical programming with nonlinear constraints by the method of centres. *In J. Abadie, editor, Nonlinear Programming,pages 209-219*, Wiley-Sons, Inc, 1967.

[22] M. T. Jones and P. E. Plassmann. An improved incomplete Cholesky factorization. *ACM Trans. Math. Software*, 21:5–17, 1995.

[23] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[24] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, pages 20:191–194, 1979.

[25] V. Klee and G. J. Minty. How good is the simplex algorithm? In *O. Shisha, Editor, Inequalities-III*, pages 159–175, 1972.

[26] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior-point method for linear programming. *Linear Algebra Appl.*, 152:191–222, 1991.

[27] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization*, 2:435–449, 1992.

[28] S. Mizuno M. Kojima and A. Yoshise. A primal-dual interior point method for linear programming. *In N. Megiddo, editor, Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, Springer Verlag:29–47, 1989.

[29] N. Megiddo. Pathways to the optimal set in linear programming. *In N. Megiddo, editor, Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, Springer Verlag:131–158, 1989.

[30] S. Mehrotra. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. *ORSA Journal on Computing*, 4:103–118, 1992.

[31] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[32] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, page 2:575601, 1992.

[33] Miscellaneous LP models. Hungarian Academy of Sciences OR Lab. Online at http://www.sztaki.hu/ meszaros/puplic-ftp/lptestset/mist.

[34] Mittelmann LP models. Miscellaneous LP models collect by Hans D. Mittelmann. Online at ftp://plato.asu.edu/pub/lptestset/pds.

[35] R. D. C. Monteiro and I. Adler. Interior path-following primal-dual algorithms, part1:linear programming. *Mathematical Programming*, pages 44:27–41, 1989.

[36] Esmond Ng and B. P. Peyton. Block sparse Cholesky algorithms on advanced uniprocessors computers. *SIAM Journal on Scientific Stat. Computing*, 14:1034–1056, 1993.

[37] A. R. L. Oliveira and D. C. Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra and Its Applications*, 394:1–24, 2005.

[38] Aurelio Ribeiro Leite Oliveira. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Technical report, PhD Thesis, TR97-11, Department of Computational and Applied Mathematics, Rice University, Houston TX, 1997.

[39] M. Padberg and M. P. Rijal. Location, scheduling, design and integer programming. *Kluwer Academic, Boston*, 1996.

[40] Mauricio G. C. Resende and Geraldo Veiga. An efficient implementation of a network interior point method. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 12:299–348, 1993.

[41] J. Silva, A. R. L. Oliveira, and M. I. Velazco. Um algoritmo novo baseado no algoritmo von neumann para programação linear. In *XXX Congresso Nacional de Matemática Aplicada e Computacional*, 2007.

[42] J. Silva, A. R. L. Oliveira, and M. I. Velazco. Uma família de novos algoritmos para programação linear baseada no algoritmo von neumann. In *XXXI Congresso Nacional de Matemática Aplicada e Computacional*, 2008.

[43] R. A. Tapia and Y. Zhang. Superlinear and quadratic convergence of primal-dual interior point methods for linear programming revisited. *Journal of Optimization Theory and Applications*, 73:229–242, 1992.

[44] R. J. Vanderbei. *Linear Programming – Foundations and Extensions*. Kluwer Academics Publishers, Boston, USA, 1996.

[45] M. I. Velazco, A. R. L. Oliveira, and F. F. Campos. A note on hybrid preconditioners for large-scale normal equations arising from interior-point methods. *Aceito para publicação na revista Optimization Methods and Software*, 2008.

[46] W. Wang and D. P. O'Leary. Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming. *Numerical Algorithms*, 25(1–4):387–406, 2000.

[47] S. J. Wright. *Primal–Dual Interior–Point Methods*. SIAM Publications, SIAM, Philadelphia, PA, USA, 1996.