Universidade Estadual de Campinas

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

Departamento de Matemática Aplicada

Dissertação de Mestrado

Detecção de Linhas Redundantes em Problemas de Programação Linear de Grande Porte

por

Daniele Costa Silva †

Mestrado em Matemática Aplicada - Campinas - SP

Orientador: Prof. Dr. Aurelio R. Leite Oliveira

†Este trabalho contou com apoio financeiro da FAPESP.

DETECÇÃO DE LINHAS REDUNDANTES EM PROBLEMAS DE PROGRAMAÇÃO LINEAR DE GRANDE PORTE

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por **Daniele Costa Silva** e aprovada pela comissão julgadora.

Campinas, 28 de maio de 2010.

Aurelio R. L. de Oliveira

Prof. Dr. Aurelio R. Leite de Oliveira
Orientador

Banca Examinadora:

Prof. Dr. Aurelio Ribeiro Leite de Oliveira
Prof. Dr. Francisco de Assis Magalhães Gomes Neto
Prof. Dr. Frederico Ferrera Campos Filho

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

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Silva, Daniele Costa
Si38d Detecção de linhas redundantes em problemas de programação linear de grande porte / Daniele Costa Silva -- Campinas, [S.P. : s.n.], 2010.

Orientador : Aurélio Ribeiro Leite de Oliveira
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Programação linear. 2. Redundância (Engenharia). 3. Métodos de pontos interiores. I. Oliveira, Aurélio Ribeiro Leite de. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Título em inglês: Finding all linearly dependent rows in large-scale linear programming.

Palavras-chave em inglês (Keywords): 1. Linear programming. 2. Redundancy (Engineering). 3. Interior point methods.

Área de concentração: Programação linear

Titulação: Mestre em Matemática Aplicada

Banca examinadora: Prof. Dr. Aurélio Ribeiro Leite de Oliveira (IMECC-Unicamp)
Prof. Dr. Francisco de Assis Magalhães Gomes Neto (IMECC-Unicamp)
Prof. Dr. Frederico Ferreira Campos (DCC-UFMG)

Data da defesa: 28/05/2010

Programa de Pós-Graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 28 de maio de 2010 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Aurelio R L de Oliveira

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

Prof. (a). Dr (a). FREDERICO FERREIRA CAMPOS FILHO

Prof. (a). Dr (a). FRANCISCO DE ASSIS MAGALHÃES GOMES NETO

AGRADECIMENTOS

Este trabalho é fruto de um processo de pesquisa e superação pessoal, o qual não teria sido possível sem a colaboração de diversas pessoas, as quais eu não posso deixar de mencionar neste espaço. Assim agradeço:

À Rosana, minha primeira orientadora e grande amiga. Pessoa pela qual tenho grande apreço e que despertou meu gosto pela pesquisa. Sem ela, os rumos de minha vida seriam bem diferentes.

À Cecilia (Ciça), minha eterna veterana, por ter me indicado o Aurelio como orientador. Realmente ela estava certa, eu não poderia encontrar orientador melhor. Muitíssimo obrigada pela sugestão.

Ao Professor Aurelio, por ter depositado uma grande confiança em mim. Mesmo sem me conhecer e com meu "atrapalhado" desempenho acadêmico, acreditou numa possível parceria. Sempre disposto, com muita paciência, pois para aguentar o desespero, os dramas e os "chiliques" de sua orientanda, realmente é necessária muita calma. Muitíssimo obrigada pela orientação e confiança, sem esses fatores, com certeza não seria possível a conclusão deste trabalho.

Aos colegas de curso, em especial, Akira, Camila, Denise, Livia e Kally, por terem compartilhado os dramas acadêmicos, virar noites, perder lindos dias de sol e claro por terem comemorado todas as nossas superações.

Ao Professor Moretti, pelo apoio dado durante o curso.

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

À minha mãe Gizelia e ao meu irmão Jonas, pelo carinho, força e apoio. Sem vocês

também não seria possível a conclusão deste trabalho.

A todos os meus amigos, que de alguma forma sempre estavam presentes nos momentos mais difíceis, me apoiando, e nos momentos mais alegres, comemorando minhas conquistas. Em especial, Cynthia (Cycy), Gilberto (Visconde), Greice (Gre), Jair (Seu Jair), Janete, Juliana (Ju), Keli, Luana (Baiana), Michele (Mi), Nathan, Nazarete (Naza) e Rafael (Rafa), por terem acompanhado mais de perto meus dramas acadêmicos, pessoais e as inúmeras vezes em que pensei em jogar tudo pro alto...

Aos Professores Frederico e Francisco (Chico), por gentilmente terem aceito o convite para cômpor a banca examinadora.

As "Forças Superiores", por terem me amparado durante esta jornada.

À FAPESP pelo imprescíndivel suporte financeiro.

RESUMO

A presença de linhas redundantes na matriz de restrições não é incomum em problemas reais de grande porte. A existência de tais linhas deve ser levada em consideração na solução destes problemas. Se o método de solução adotado for o método simplex, existem procedimentos eficientes e de fácil implementação que contornam este problema. O mesmo se aplica quando métodos de pontos interiores são adotados e os sistemas lineares resultantes são resolvidos por métodos diretos. No entanto, existem problemas de grande porte cuja única forma possível de solução é resolver os sistemas lineares por métodos iterativos. Nesta situação as linhas redundantes representam uma dificuldade considerável pois geram uma matriz singular e os métodos iterativos não convergem. A única alternativa viável consiste em detectar tais linhas e eliminá-las antes da aplicação dos métodos de pontos interiores. Este trabalho propõe uma implementação eficiente de um procedimento de detecção de linhas redundantes, que incluímos em uma adaptação própria do PCx que resolve os sistemas lineares por métodos iterativos.

ABSTRACT

The presence of dependent rows in the constraint matrix is frequent in real large-scale problems. If the method of solution adopted is the simplex method, there are efficient procedures easy to implement that circumvent this problem. The same applies when interior point methods are adopted and the resulting linear systems are solved for directed methods. However, there are large-scale problems whose only possible solution is to solve linear systems by iterative methods. In this situation, the dependent rows create a singular matrix and the iterative method does not converge. The only viable alternative is to find and remove these rows before applying the method. This dissertation proposes an efficient implementation of a procedure for detection dependent rows, include in a PCx modification that solves linear systems by iterative methods.

CONTEÚDO

LISTA DE TABELAS

Introdução

É amplamente reconhecida a importância do preprocessamento na resolução de problemas de programação linear de grande porte, independente do método de solução escolhido. Embora os códigos de programação linear e os computadores tenham se tornado muito mais rápidos, os modelos aumentaram significativamente de tamanho. Portanto, técnicas de execução mais eficazes são de grande importância.

Os modelos práticos de programação linear são, em sua maioria, gerados por programas computacionais, quer seja diretamente ou pela inserção em sistemas de modelagem. Tais geradores, de forma geral, possuem capacidades muito limitadas para a análise de dados. Como consequência, o modelo pode apresentar diversos tipos de redundância.

Neste trabalho, nos concentraremos na detecção e remoção de restrições linearmente dependentes, denominadas linhas redundantes. A redundância não é incomum nos problemas reais de grande porte, e deve ser levada em consideração na solução destes, principalmente quando são resolvidos por métodos de pontos interiores e seus sistemas lineares por métodos iterativos.

Apresentamos uma proposta eficiente de implementação de um algoritmo bastante conhecido (Chvátal (1983)), cuja finalidade é a detecção deste tipo de redundância, tomando como partida as idéias apresentadas e discutidas em Andersen (1995).

O algoritmo consiste na construção de uma matriz base constituída por colunas da matriz de restrições e colunas relacionadas a variáveis artificiais que podem ser agregadas ao problema. Por meio de resolução de sistemas lineares que envolvem esta base e suas possíveis

atualizações, as linhas redundantes são detectadas.

Para a representação e atualização da base, assim como para a resolução dos sistemas lineares, Andersen (1995) propõe a utilização da decomposição LU e sua atualização baseada no procedimento de Bartels-Golub (Bartels *et al.*). Alternativamente, propomos um procedimento inspirado na forma produto da inversa (Dantzig e Orchard-Hays (1954)).

Integramos o algoritmo a uma versão modificada do código PCx, a qual implementa o método primal-dual preditor-corretor (Mehrotra (1992)) resolvendo os sistemas lineares iterativamente pelo método dos gradientes conjugados precondicionado.

Este trabalho é constituído por mais cinco capítulos. No Capítulo 1, definimos o problema de programação linear e descrevemos brevemente os métodos primais-duais de pontos interiores assim como uma de suas mais importantes variações, o método preditor-corretor. No Capítulo 2, apresentamos o método dos gradientes conjugados precondicionado como alternativa para a resolução dos sistemas lineares oriundos dos métodos de pontos interiores. Adicionalmente, é feita uma breve descrição de dois precondicionadores; o separador e a fatoração controlada de Cholesky. No Capítulo 3, é apresentado o algoritmo de detecção de linhas redundantes e propostas para sua implementação. No Capítulo 4, são apresentados os resultados obtidos através dos experimentos numéricos realizados com o propósito de verificar a eficácia do algoritmo. Finalmente as conclusões e trabalhos futuros são apresentados no Capítulo 5.

CAPÍTULO 1

Métodos Primais Duais de Pontos Interiores

O trabalho de Karmarkar, apresentado em 1984, revolucionou a área da programação linear, com a publicação de um algoritmo com complexidade polinomial e bom desempenho quando aplicado a problemas práticos. Esse fato desencadeou um novo campo de pesquisa: os métodos de pontos interiores. Desde então, códigos computacionais baseados nessas idéias vêm se apresentando como uma alternativa eficiente para problemas de programação linear, em especial em problemas de grande porte (Adler *et al.* (1989); Bocanegra *et al.* (2007); Czyzyk *et al.* (1999); Gondzio (1996); Lustig *et al.* (1992); Oliveira e Sorensen (2005)).

Podemos dividir os métodos de pontos interiores em três grupos, a saber, primal, dual e primal-dual; de acordo com o espaço em que ocorrem as iterações. Alternativamente, estes grupos podem ser classificados em três categorias: métodos afim escala (Dikin (1967)), métodos de redução de potencial (Anstreicher (1996)) e métodos de trajetória central (Gonzaga (1992); Huard (1967)). Neste trabalho, serão abordados os métodos primais-duais pertencentes a categoria de trajetória central.

No presente capítulo, definiremos o problema de programação linear e apresentaremos uma breve descrição do método primal-dual e de uma de suas mais importantes variações, o método preditor-corretor.

1.1 Problemas de Programação Linear

Consideremos o seguinte problema de programação linear na forma padrão:

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

onde $A \in \mathbb{R}^{mxn}$, $c, x \in \mathbb{R}^n$, e $b \in \mathbb{R}^m$.

Associado a este problema temos o problema dual:

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

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

As soluções primal-dual de (1.1) e (1.2) são caracterizadas pelas condições de Karush-Kunch-Tucker (condições de otimalidade), as quais são definidas a seguir:

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

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

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 factível se satisfaz o conjunto de restrições dos problemas primal e dual, e o ponto é dito interior se $(x, z) > 0$. O gap dual é dado por $\gamma = c^t x - b^t y$, que se reduz a $\gamma = x^t z$ para pontos primais e duais factíveis.

Os problemas de programação linear, em geral, não estão na forma definida em (1.1). Estes podem apresentar variáveis negativas, livres ou limitadas, e restrições de desigualdades. Entretanto, todos estes problemas podem ser reduzidos a forma padrão (1.1) acrescentando variáveis e/ou restrições.

1.2 Métodos Primais-Duais

Os métodos de pontos interiores primais-duais são considerados os métodos mais eficientes dentro da gama de variações dos métodos de pontos interiores, além de apresentarem as melhores propriedades teóricas para a análise de complexidade (Adler e Monteiro (1989)) e convergência (Tapia e Zhang (1992)). Consistem, em sua maioria, na aplicação do método de Newton modificado (uma generalização do método de Newton-Raphson para o cálculo do zero de uma função unidimensional), às condições de otimalidade (1.3) do problema de programação linear, com o objetivo de obter aproximações para a solução destas.

Nesta seção, descreveremos o método de Newton e alguns métodos primais-duais, destacando os métodos de trajetória central.

1.2.1 Método de Newton

Considere o seguinte sistema não linear:

$$\begin{aligned} F(x) &= 0, \\ F : \mathbb{R}^n &\longrightarrow \mathbb{R}^n \end{aligned} \tag{1.4}$$

com $F \in C^1(\mathbb{R}^n)$.

Entre os métodos clássicos para a resolução do sistema não linear (1.4), encontra-se o método de Newton. Este consiste em, a cada iteração, partindo de um ponto $x \in \mathbb{R}^n$, encontrar uma direção Δx tal que $F(x + \Delta x) \simeq 0$. Isto pode ser feito utilizando os dois primeiros termos da expansão da função F em série de Taylor em torno de um ponto x^k, ou seja:

$$F(x^k + \Delta x^k) \simeq F(x^k) + J(x^k)\Delta x^k,$$

onde J é a matriz jacobiana de F.

Igualando a aproximação de $F(x^k + \Delta x^k)$ a zero, obtemos:

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

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 Δx^k como a direção de Newton.

O novo ponto é dado por $x^{k+1} = x^k + \Delta x^k$.

1.2.2 Método de Pontos Interiores Primais-Duais

Considere as condições de otimalidade (1.3).

Aplicaremos o método de Newton à estas condições, desprezando $(x, z) \geq 0$. Assim, sejam:

- a função $F(x, y, z) = \begin{bmatrix} Ax - b \\ A^t y + z - c \\ XZe \end{bmatrix}$;
- sua matriz jacobiana $J(x, y, z) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{bmatrix}$;
- e o vetor de resíduos $r(x, y, z) = \begin{bmatrix} r_p \\ r_d \\ r_c \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^t y - z \\ -XZe \end{bmatrix}$,

em que $r_p = b - Ax^k$, $r_d = c - A^t y^k - z^k$ e $r_c = -X^k Z^k e$ são os resíduos primal, dual e de complementariedade, respectivamente.

Pelo método de Newton, dado um ponto $w^k = (x^k, y^k, z^k)^t$, determinamos um novo ponto $w^{k+1} = (x^{k+1}, y^{k+1}, z^{k+1})^t$ da seguinte forma:

$$w^{k+1} = w^k + \Delta w^k = (x^k, y^k, z^k)^t + (\Delta x^k, \Delta y^k, \Delta z^k)^t,$$

onde $\Delta w^k = (\Delta x^k, \Delta y^k, \Delta z^k)^t$ é a direção de Newton, obtida através da solução do sistema linear $J(w^k)\Delta w^k = -F(w^k) = r(w^k)$, ou seja:

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

Agora, considere que o ponto $w^k = (x^k, y^k, z^k)^t$ é um ponto interior, isto é, $(x^k, y^k) > 0$. Gostariamos de utilizar o método de Newton, de tal forma que o próximo ponto

$w^{k+1} = (x^{k+1}, y^{k+1}, z^{k+1})^t$ também fosse interior ($(x^{k+1}, y^{k+1}) > 0$). Para isto, utilizamos o vetor de passo $\alpha^k = (\alpha_p^k, \alpha_d^k, \alpha_d^k)^t$, tal que $w^{k+1} = w^k + \alpha^k \Delta w^k$ é um ponto interior.

O vetor α^k pode ser calculado da seguinte forma:

$$\alpha_p^k = \min(1, \delta\overline{\alpha}_p^k), \quad \alpha_d^k = \min(1, \delta\overline{\alpha}_d^k) \quad , \tag{1.6}$$

onde $\overline{\alpha}_p^k = \dfrac{-1}{\min_i(\frac{\Delta x_i^k}{x_i^k})}$, $\overline{\alpha}_d^k = \dfrac{-1}{\min_i(\frac{\Delta z_i^k}{z_i^k})}$ e $\delta \in (0,1)$.

A idéia central dos métodos de pontos interiores primal-dual consiste em aplicar o procedimento descrito acima, ou seja, a partir de um ponto interior $w^k = (x^k, y^k, z^k)^t$, gerar uma sequência de pontos interiores tal que:

$$w^{k+1} = (x^{k+1}, y^{k+1}, z^{k+1})^t = w^k + \alpha^k \Delta w^k = (x^k, y^k, z^k) + (\alpha_p^k \Delta x^k, \alpha_d^k \Delta y^k, \alpha_d^k \Delta z^k),$$

onde α^k é o vetor de passo obtido de acordo com (1.6) e Δw^k é a direção de busca obtida pela aplicação do método de Newton às condições de otimalidade perturbadas por um parâmetro de barreira $\mu > 0$, ou seja, às seguintes condições:

$$\begin{aligned} Ax - b &= 0, \\ A^t y + z - c &= 0, \\ XZe &= \mu e, \\ (x, z) &\geq 0. \end{aligned} \tag{1.7}$$

Esta perturbação é feita com o objetivo de tornar os pontos mais centrais.

Observe que, para $\mu = 0$, as equações (1.6) são equivalentes a (1.3).

A cada iteração, o parâmetro $\mu > 0$ é reduzido por um determinado fator. E a cada valor de μ obtemos uma única solução $(x(\mu), y(\mu), z(\mu))^t$ para o sistema (1.7), a qual define o caminho central da região factível primal-dual. Conforme μ se aproxima de zero, esta solução se aproxima de uma solução ótima primal-dual.

Sendo assim, dado um ponto $w^k = (x^k, y^k, z^k)^t$, uma nova iteração do método é realizada, determinando um novo ponto,

$$w^{k+1} = w^k + \alpha^k \Delta w^k,$$

onde $\Delta w^k = (\Delta x^k, \Delta y^k, \Delta z^k)^t$ é a solução do seguinte sistema de equações de Newton:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{pmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_c \end{bmatrix}, \tag{1.8}$$

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

Método de Pontos Interiores Primal-Dual de Trajetória Central

Entradas: $w^0 = (x^0, y^0, z^0)$, tal que $(x^0, z^0) > 0$.

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_c^k.

[3] Calcule a direção de Newton Δw^k, de acordo com (1.8).

[4] Calcule o tamanho do passo $\alpha^k = (\alpha_p^k, \alpha_d^k, \alpha_d^k)$ de acordo com (1.6).

[5] Calcule o novo ponto $w^{k+1} = w^k + \alpha^k \Delta w^k$.

Fim.

A cada iteração do método, é necessário resolver um sistema de equações lineares para determinar a direção de Newton. Quando métodos diretos são 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 n apresenta complexidade $O(n^3)$, enquanto que a solução de sistemas triangulares pode ser feita com complexidade $O(n^2)$. Algumas variantes de métodos de pontos interiores resolvem a cada iteração vários sistemas com a mesma matriz de coeficientes e, com isso, determinam uma melhor direção, reduzindo o número total de iterações e, consequentemente, o número de fatorações.

1.2.3 Método Preditor-Corretor

Uma das mais importantes variantes do método apresentado na subseção anterior é o método preditor-corretor. Este, foi proposto por Mehrotra (1992) e difere do método anterior no cálculo da direção de Newton, a qual é decomposta em duas partes: $\Delta w^k = \Delta_a(w^k) + \Delta_c(w^k)$.

A componente $\Delta_a(w^k) = (\Delta_a x^k, \Delta_a y^k, \Delta_a z^k)^t$, denominada direção afim, é utilizada para determinar uma melhor predição para o cálculo do parâmetro de barreira (μ), o qual é escolhido usando a seguinte heurística:

$$\mu^k = \left(\frac{(x^k + \alpha_{pa}\Delta_a x^k)^t (z^k + \alpha_{da}\Delta_a z^k)}{(x^t) z^k} \right)^p \frac{(x^k + \alpha_{pa}\Delta_a x^k)^t (z^k + \alpha_{da}\Delta_a z^k)}{n}, \tag{1.9}$$

onde α_{pa} e α_{da} são os passos que preservam a não negatividade das variáveis x e z.

Já a componente $\Delta_c(w^k) = (\Delta_c x^k, \Delta_c y^k, \Delta_c z^k)^t$, conhecida como direção corretora, tem por objetivo fazer com que a nova iteração seja a mais central possível e fazer a correção do termo não linear.

A seguir discutiremos aspectos envolvidos no cálculo destas componentes.

Considere novamente as condições de otimalidade (1.3) e os resíduos relacionados à estas, ou seja:

$$r(x, y, z) = \begin{bmatrix} r_p \\ r_d \\ r_c \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^t y - z \\ -XZe \end{bmatrix}. \tag{1.10}$$

Vimos anteriormente que os métodos de pontos interiores primais-duais, consistem na geração da seguinte sequência de pontos interiores:

$$w^{k+1} = w^k + \alpha^k \Delta w^k.$$

Vamos analisar o que ocorre com os resíduos (1.10) se tomarmos $\alpha^k = (1, 1, 1)^t$, desconsiderando a perturbação dada pelo parâmetro de barreira (μ). Sendo assim:

- Resíduo Primal (r_p)
  Da equação (1.10),
  temos $r_p^{k+1} = b - Ax^{k+1} = b - A(x^k + \alpha_p^k \Delta x^k) = b - A(x^k + \Delta x^k)$,
  uma vez que $\alpha^k = (1, 1, 1)^t$.
  Assim, $r_p^{k+1} = (b - Ax^k) - A\Delta x^k$.

  Das equações de Newton (1.5) temos que $A\Delta x^k = r_p^k$. Portanto, $r_p^{k+1} = (b - Ax^k) - A\Delta x^k = r_p^k - r_p^k = 0$.

- Resíduo Dual (r_d)
  Da equação (1.10), temos

$r_d^{k+1} = c - A^t y^{k+1} - z^{k+1} = c - A^t(y^k + \alpha_d^k \Delta y^k) - (z^k + \alpha_d^k \Delta z^k) = c - A^t(y^k + \Delta y^k) - (z^k - \Delta z^k)$,

uma vez que $\alpha^k = (1,1,1)^t$.

Daí, temos

$r_d^{k+1} = (c - A^t y^k - z^k) - (A^t \Delta y^k + \Delta z^k)$.

Das equações de Newton (1.5), temos que

$A^t \Delta y^k + \Delta z^k = r_d^k$. Logo,

$r_d^{k+1} = (c - A^t y^k - z^k) - (A^t \Delta y^k + \Delta z^k) = r_d^k - r_d^k = 0$.

- Resíduo de Complementariedade (r_c)

  Da equação (1.10) temos,

  $r_c^{k+1} = -X^{k+1} Z^{k+1} e = -X^{k+1} z^{k+1} = -(X^k + D_x^k)(z^k + \Delta z^k)$,

  onde $D_x = diag(\Delta x)$.

  Assim,

  $r_c^{k+1} = -(X^k z^k + D_x^k z^k + X^k \Delta z^k + D_x^k \Delta z^k) = -(X^k Z^k e + Z^k \Delta x^k + X^k \Delta z^k + D_x^k \Delta z^k)$.

  Pelas equações de Newton (1.5), temos que $Z^k \Delta x^k + X^k \Delta z^k = r_c^k$. Portanto,

  $r_c^{k+1} = -(X^k Z^k e + Z^k \Delta x^k + X^k \Delta z^k + D_x^k \Delta z^k) =$

  $- (-r_c^k + r_c^k + D_x^k \Delta z^k) =$

  $- D_x^k D_z^k e$.

  Onde $D_z = diag(\Delta z)$.

Podemos perceber que, ao tomar $\alpha^k = (1,1,1)^t$, os resíduos primal (r_p) e dual (r_d) se anulam e o resíduo de complementariedade (r_c) passa a ser dado pelo produto $D_x D_z e$. Com base nestas informações, damos início ao cálculo das componentes Δ_a e Δ_c.

Primeiramente, obtemos a direção afim, Δ_a (passo preditor), por meio da solução do sistema de equações de Newton (1.5), ou seja:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{pmatrix} \begin{bmatrix} \Delta_a x^k \\ \Delta_a y^k \\ \Delta_a z^k \end{bmatrix} = \begin{bmatrix} b - Ax^k \\ c - z^k - A^t y^k \\ -X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_c^k \end{bmatrix}. \tag{1.11}$$

Posteriormente, determinamos a direção de correção Δ_c (passo corretor). Para isto, resolvemos o sistema de equações de Newton (1.7), partindo da direção afim (Δ_a) já determinada e tomando o vetor de passo $\alpha^k = (1,1,1)^t$, ou seja,

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{pmatrix} \begin{bmatrix} \Delta_c x^k \\ \Delta_c y^k \\ \Delta_c z^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mu^k e - \Delta_a X^k \Delta_a Z^k e \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_c^k \end{bmatrix}, \qquad (1.12)$$

onde $\Delta_a X^k = diag(\Delta_a x^k)$ e $\Delta_a Z^k = diag(\Delta_a z^k)$.

Observe que neste método é necessária a resolução de dois sistemas de equações lineares, sendo a matriz de coeficientes a mesma para ambos os passos (preditor e corretor), mas com diferentes vetores no lado direito. Isto nos leva a um tempo computacional menor para a resolução da maioria dos problemas.

Note, também, que a direção final ($\Delta w^k = \Delta_a(w^k) + \Delta_c(w^k)$) pode ser obtida diretamente da soma dos sistemas lineares (1.11) e (1.12). Assim, evita-se o cálculo $\Delta w^k = \Delta_a(w^k) + \Delta_c(w^k)$.

Devido a seu desempenho superior na prática, este método é usado em diversas implementações de otimização linear.

Descreveremos a seguir os passos deste método.

Método Preditor-Corretor

Entradas: $w^0 = (x^0, y^0, z^0)$ tal que $(x^0, y^0) > 0$

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Calcule os resíduos r_p^k, r_d^k e r_c^k, com $\mu^k = 0$ de acordo com a equação (1.10).

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

[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, de acordo com (1.9).

[5] Calcule o resíduo $r_c^k = \mu^k e - \Delta_a X^k \Delta_a Z^k e$.

[6] Calcule a direção final de Newton, Δw^k.

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

[8] Calcule $w^{k+1} = (x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\alpha_p^k \Delta x^k, \alpha_d^k \Delta y^k, \alpha_d^k \Delta z^k) = w^k + \alpha^k \Delta w^k$.

Fim.

CAPÍTULO 2

Métodos Iterativos para Sistemas Lineares

O maior esforço computacional exigido nos método de pontos interiores primais-duais é determinar as direções de Newton através da solução de um ou mais sistemas lineares. Umas das abordagens mais utilizadas para resolução destes sistemas nas implementações existentes consistem em reduzi-los às equações normais e aplicar a fatoração de Cholesky (Adler *et al.* (1989); Czyzyk *et al.* (1999); Gondzio (1996); Lustig *et al.* (1992)). No entanto, por limitações de tempo e memória, o uso dessa estratégia torna-se proibitivo em muitos problemas de grande porte, fazendo com que abordagens iterativas sejam mais adequadas.

Neste capítulo, nos concentraremos na resolução destes sistemas através da redução às equações normais e a aplicação do método dos gradientes conjugados precondicionado.

2.1 Redução às Equações Normais

Conforme visto no capítulo anterior, para determinarmos as direções de Newton é necessária a resolução de um ou mais sistemas lineares. Como estes sistemas compartilham a mesma matriz de coeficientes, nos restringiremos ao estudo do sistema linear (1.8), dado por:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_c \end{bmatrix}, \tag{2.1}$$

onde $r_p = b - Ax$, $r_d = c - A^t y - z$ e $r_c = \mu e - XZe$.

Por simplicidade de notação, retiramos os índices referentes as iterações (k).

Substituindo $\Delta z = X^{-1}(r_c - Z\Delta x)$ na segunda equação de (2.1), obtemos o seguinte sistema linear, denominado sistema aumentado ou indefinido:

$$\begin{pmatrix} -D^{-1} & A^t \\ A & 0 \end{pmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1}r_c \\ r_p \end{bmatrix}, \tag{2.2}$$

onde: $D = Z^{-1}X$.

Utilizando a primeira equação de (2.2) para eliminar Δx, obtemos o seguinte sistema de equações normais:

$$(AZ^{-1}XA^t)\Delta y = r_p + A(Z^{-1}Xr_d + x - \mu Z^{-1}e). \tag{2.3}$$

Uma vez determinada Δy pela resolução do sistema anterior, as demais soluções podem ser facilmente calculadas:

$$\Delta z = r_d - A^t \Delta y,$$

$$\Delta x = Z^{-1}(r_c - X\Delta z).$$

Comparando o sistema de equações normais (2.3) ao sistema aumentado (2.2), pode-se observar que a matriz do sistema de equações normais é simétrica e definida-positiva (uma vez que A tenha posto completo), o que possibilita a resolução de (2.3) através da fatoração de Cholesky ou do método dos gradientes conjugados. Por outro lado, o sistema aumentado (2.2) pode ser resolvido através de métodos diretos ou iterativos, os quais não serão discutidos neste trabalho.

2.2 Método dos Gradientes Conjugados

Proposto em 1952 por Hestenes e Stiefel, o método dos gradientes conjugados tornou-se uma das técnicas mais utilizadas para a resolução de sistemas lineares definidos-positivos;

principalmente quando estamos tratando de problemas de grande porte. Nestes casos, o método é mais vantajoso em relação aos métodos diretos (decomposição de matrizes, por exemplo), pois não altera a matriz de coeficientes.

Nesta e na próxima seção, apresentaremos o método dos gradientes conjugados e conceitos relacionados a este. Uma análise detalhada da teoria que fundamenta estas seções podem ser vistas em Nocedal e Wright (1960); Pulino (2008).

2.2.1 O Problema de Minimização

Considere o sistema linear definido-positivo:

$$Ax = b^1 \tag{2.4}$$

onde $A \in \mathbb{R}^{nxn}$ é uma matriz simétrica definida-positiva; x, $b \in \mathbb{R}^n$.

O sistema (2.4) é equivalente ao seguinte problema de minimização: encontrar $x^* \in \mathbb{R}^n$ tal que

$$J(x^*) = min[J(x), x \in \mathbb{R}^n], \tag{2.5}$$

onde o funcional $J : \mathbb{R}^n \longmapsto \mathbb{R}$ é definido da seguinte forma:

$$J(x) = \frac{1}{2} \langle Ax, x \rangle - \langle b, x \rangle , \tag{2.6}$$

com $< ., . >$ indicando o produto interno usual em $\mathbb{R}^n$.

Ou seja, (2.4) e (2.5) possuem a mesma solução.

2.2.2 Descrição do Método

O método dos gradientes conjugados consiste em obter uma solução numérica para o sistema linear definido-positivo (2.4) por meio de um procedimento iterativo para encontrar o ponto de mínimo do funcional J de (2.6), ou seja, através da solução numérica para o problema de minimização (2.5).

Para isso, são consideradas sucessivas minimizações do funcional J ao longo de direções $[p^0, p^1, ..., p^k]$. Isto é:

[1] Note que o sistema (2.4) é distinto do sistema apresentado no problema de programação linear (1.1).

$$J(x^{k+1}) = \min[J(x^k + \lambda p^k); \lambda \in \mathbb{R}] \tag{2.7}$$

com

$$x^{k+1} = x^k + \lambda_k p^k. \tag{2.8}$$

Logo, o parâmetro λ deve ser escolhido de modo a obtermos o ponto de mínimo na variedade linear S_k definida da seguinte forma:

$$S_k = [z \in \mathbb{R}^n; z = x^k + \lambda p^k; \lambda \in \mathbb{R}].$$

Sendo assim, considere a função auxiliar $\varphi : \mathbb{R} \longmapsto \mathbb{R}$ definida como:

$$\varphi(\lambda) = J(x^k + \lambda p^k); \lambda \in \mathbb{R}.$$

Fazendo $\varphi^{'}(\lambda) = 0$, obtemos o parâmetro λ_k que realiza o mínimo de J na variedade linear S_k. Assim, tem-se:

- $\varphi(\lambda) = J(x^k + \lambda p^k) = \frac{1}{2}\left\langle A(x^k + \lambda p^k), x^k + \lambda p^k\right\rangle - \left\langle b, x^k + \lambda p^k\right\rangle =$
  $\frac{1}{2}\left[\left\langle Ax^k + \lambda Ap^k, x^k\right\rangle + \left\langle Ax^k + \lambda Ap^k, \lambda p^k\right\rangle\right] - \left\langle b, x^k\right\rangle - \lambda\left\langle b, p^k\right\rangle =$
  $\frac{1}{2}\left[\left\langle Ax^k, x^k\right\rangle + \lambda\left\langle Ap^k, x^k\right\rangle + \lambda\left\langle Ax^k, p^k\right\rangle + \lambda^2\left\langle Ap^k, p^k\right\rangle\right] - \left\langle b, x^k\right\rangle - \lambda\left\langle b, p^k\right\rangle =$
  $\frac{1}{2}\left[\left\langle Ax^k, x^k\right\rangle + 2\lambda\left\langle Ax^k, p^k\right\rangle + \lambda^2\left\langle Ap^k, p^k\right\rangle\right] - \left\langle b, x^k\right\rangle - \lambda\left\langle b, p^k\right\rangle.$

- Derivando:
  $\varphi^{'}(\lambda) = \left\langle Ax^k, p^k\right\rangle + \lambda\left\langle Ap^k, p^k\right\rangle - \left\langle b, p^k\right\rangle.$

- Igualando a derivada a zero:
  $\varphi^{'}(\lambda) = 0 \Longrightarrow \left\langle Ax^k, p^k\right\rangle + \lambda\left\langle Ap^k, p^k\right\rangle - \left\langle b, p^k\right\rangle = 0 \Longrightarrow$
  $\left\langle Ax^k - b, p^k\right\rangle + \lambda\left\langle Ap^k, p^k\right\rangle = 0 \Longrightarrow$
  $\lambda = \frac{-\left\langle Ax^k - b, p^k\right\rangle}{\left\langle Ap^k, p^k\right\rangle} = -\frac{\left\langle -r^k, p^k\right\rangle}{\left\langle Ap^k, p^k\right\rangle} = \frac{\left\langle r^k, p^k\right\rangle}{\left\langle Ap^k, p^k\right\rangle},$
  onde $r^k = Ax^k - b$ é denominado resíduo.

Portanto, temos:

$$\lambda_k = \frac{\left\langle r^k, p^k\right\rangle}{\left\langle Ap^k, p^k\right\rangle}. \tag{2.9}$$

Podemos escolher as direções p^k, $k = 0, 1, 2...$, de modo que o processo iterativo (2.7) convirja em n passos, onde n é a dimensão do sistema linear.

Primeiramente, construimos direções consecutivas p^k e p^{k+1} A-conjugadas, isto é

$$\left\langle Ap^k, p^{k+1} \right\rangle = 0 \quad \forall k$$

com $p^0 = -\nabla J(x^0) = r^0$,

onde $\nabla J(x)$ é o gradiente do funcional J no ponto x.

Com isto, determinamos as direções da seguinte forma:

$$p^{k+1} = r^{k+1} + \beta_k p^k, \tag{2.10}$$

onde o parâmetro β_k é obtido impondo a condição que as direções p^k e p^{k+1} sejam A-conjugadas:

$$\beta_k = \frac{-\left\langle Ar^{k+1}, p^k \right\rangle}{\left\langle Ap^k, p^k \right\rangle} = \frac{-\left\langle r^{k+1}, Ap^k \right\rangle}{\left\langle Ap^k, p^k \right\rangle}. \tag{2.11}$$

Assim, temos que a direção p^{k+1} é a projeção ortogonal do resíduo r^{k+1} sobre o complemento ortogonal do subespaço gerado pela direção p^k.

O resíduo r^{k+1} pode ser calculado como segue:

$$r^{k+1} = b - Ax^{k+1} = b - Ax^k - \lambda_k Ap^k.$$

Com isto temos:

$$r^{k+1} = r^k - \lambda_k Ap^k. \tag{2.12}$$

Apresentamos a seguir uma versão inicial do método dos gradientes conjugados.

Método dos Gradientes Conjugados

Entradas: x^0, A, b.

[0] Calcule o resíduo e a direção inicial: $r^0 = b - Ax^0$; $p^0 = r^0$.

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Calcule λ_k de acordo com (2.9).

[2] Atualize o ponto de mínimo (x^{k+1}) de acordo com (2.8).

[3] Atualize o valor do resíduo (r^{k+1}) de acordo com (2.12).

[4] Calcule β_k de acordo com (2.11).

[5] Atualize a direção (p^{k+1}) de acordo com (2.10).

Fim.

2.2.3 Versão Econômica

Podemos escrever os parâmetros λ_k e β_k de forma mais econômica, tornando o método mais eficaz.

Observe que:

$$\left\langle r^k, p^k \right\rangle = \left\langle r^k, r^k + \beta_{k-1} p^{k-1} \right\rangle = \left\langle r^k, r^k \right\rangle + \beta_{k-1} \left\langle r^k, p^{k-1} \right\rangle = \left\langle r^k, r^k \right\rangle,$$

pois $\left\langle r^k, p^{k-1} \right\rangle = 0$, tendo em vista que $r_k = -\nabla J(x_k)$, onde x_k é o ponto de mínimo do funcional J na direção p_{k-1}. Assim, temos que:

$$\lambda_k = \frac{\left\langle r^k, r^k \right\rangle}{\langle Ap^k, p^k \rangle}. \tag{2.13}$$

Através da relação (2.12), obtemos uma expressão para Ap^k que, substituída em (2.11) e com a nova expressão (2.13), implica em:

$$\beta_k = \frac{\left\langle r^{k+1}, r^{k+1} \right\rangle}{\left\langle r^k, r^k \right\rangle}. \tag{2.14}$$

Com as novas expressões para os parâmetros λ_k (2.13) e β_k (2.14), podemos apresentar o método dos gradientes conjugados otimizado:

Método dos Gradientes Conjugados Otimizado

Entradas: x^0, A, b.

[0] Calcule o resíduo e a direção inicial: $r^0 = b - Ax^0$; $p^0 = r^0$.

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Calcule λ_k de acordo com (2.13).

[2] Atualize o ponto de mínimo (x^{k+1}) de acordo com (2.8).

[3] Atualize o valor do resíduo (r^{k+1}) de acordo com (2.12).

[4] Calcule β_k de acordo com (2.14).

[5] Atualize a direção (p^{k+1}) de acordo com (2.10).

Fim.

2.3 Método dos Gradientes Conjugados Precondicionado

Um dos fatores que determinam o sucesso de um método iterativo é a sua velocidade de convergência. Nesta seção, apresentaremos o precondicionamento como uma alternativa de melhora na taxa e velocidade de convergência do método dos gradientes conjugados.

2.3.1 Precondicionamento

Conforme visto na Subseção 2.2.2, o método dos gradientes conjugados é construído de forma que a conversão ocorra em no máximo n iterações, onde n é a dimensão do sistema linear. Entretanto, esta taxa de convergência pode ser reduzida de acordo com a distribuição dos autovalores da matriz de coeficientes do sistema linear. Uma alternativa é o precondicionamento do sistema linear (2.4).

O precondicionamento consiste em transformar o sistema linear simétrico e definido-positivo $Ax = b$ em um sistema linear também simétrico e definido-positivo $\widetilde{A}\widetilde{x} = \widetilde{b}$, onde a matriz $\widetilde{A}$ tem a propriedade de que $\mathbb{K}_2(\widetilde{A})$ é bem menor que $\mathbb{K}_2(A)$.

Para tanto, devemos encontrar uma matriz C para ser o precondicionador da matriz A, de modo que, $\widetilde{A} = C^{-t}AC^{-1}$. Em seguida, aplicamos o método dos gradientes conjugados otimizado no sistema linear transformado:

$$\widetilde{A}\widetilde{x} = \widetilde{b}, \tag{2.15}$$

onde $\widetilde{x} = Cx$ e $\widetilde{b} = C^{-t}b$.

Sendo assim, dados uma aproximação inicial $\widetilde{x}^0 \in \mathbb{R}^n$; $\widetilde{A} = C^{-t}AC^{-1}$, tal que $A \in \mathbb{R}^{nxn}$, $C \in \mathbb{R}^{nxn}$ não singular; $\widetilde{b} = C^{-t}b$, $b \in \mathbb{R}^n$, temos:

- **Método dos gradientes conjugados otimizado aplicado ao sistema** $\widetilde{A}\widetilde{x} = \widetilde{b}$

$\widetilde{r}^0 = \widetilde{b} - \widetilde{A}\widetilde{x}^0$;

$\widetilde{p}^0 = \widetilde{r}^0$.

Para $k = 0, 1, 2, \ldots$ **faça**

$\widetilde{\lambda}_k = \frac{\langle \widetilde{r}^k, \widetilde{r}^k \rangle}{\langle \widetilde{A}\widetilde{p}^k, \widetilde{p}^k \rangle}$;

$\widetilde{x}^{k+1} = \widetilde{x}^k + \widetilde{\lambda}_k \widetilde{p}^k$;

$\widetilde{r}^{k+1} = \widetilde{r}^k - \widetilde{\lambda}_k \widetilde{A}\widetilde{p}^k$;

$\widetilde{\beta}_k = \frac{\langle \widetilde{r}^{k+1}, \widetilde{r}^{k+1} \rangle}{\langle \widetilde{r}^k, \widetilde{r}^k \rangle}$;

$\widetilde{p}^{k+1} = \widetilde{r}^{k+1} + \widetilde{\beta}_k \widetilde{p}^k$.

Fim.

Observe que necessitamos de uma mudança de variável adequada para resolvermos o problema com os dados originais. Desta forma, fazemos:

$x^k = C^{-1}\widetilde{x}^k$;

$p^k = C^{-1}\widetilde{p}^k$;

$z^k = C^{-t}\widetilde{r}^k$;

$r^k = C^t\widetilde{r}^k = b - Ax^k$;

$M = C^tC$.

Com isto, obtemos:

- **Resíduo e direção inicial**

$$r^0 = C^t\widetilde{r}^0 = b - Ax^0 \quad p^0 = z^0 \quad \text{onde} \quad Mz^0 = r^0. \tag{2.16}$$

- **Parâmetros**

Tomando $Mz^k = r^k$, $k = 0, 1, 2...$, temos:

$$\widetilde{\lambda}_k = \frac{\langle \widetilde{r}^k, \widetilde{r}^k \rangle}{\left\langle \widetilde{A}\widetilde{p}^k, \widetilde{p}^k \right\rangle} = \frac{\langle C^{-t}r^k, C^{-t}r^k \rangle}{\langle C^{-t}AC^{-1}p^k, Cp^k \rangle} = \frac{\langle C^{-1}C^{-t}r^k, r^k \rangle}{\langle C^{-t}Ap^k, Cp^k \rangle} = \frac{\langle M^{-1}r^k, r^k \rangle}{\langle Ap^k, C^{-1}Cp^k \rangle} = \frac{\langle z^k, r^k \rangle}{\langle Ap^k, p^k \rangle}. \tag{2.17}$$

$$\widetilde{\beta}_k = \frac{\langle \widetilde{r}^k, \widetilde{r}^k \rangle}{\langle \widetilde{r}^k, \widetilde{r}^k \rangle} = \frac{\langle z^{k+1}, r^{k+1} \rangle}{\langle z^k, r^k \rangle}. \tag{2.18}$$

- **Atualizações**

$\widetilde{x}^{k+1} = \widetilde{x}^k + \widetilde{\lambda}_k \widetilde{p}^k \Leftrightarrow Cx^{k+1} = Cx^k + \widetilde{\lambda}_k Cp^k \Leftrightarrow x^{k+1} = x^k + \widetilde{\lambda}_k p^k$;

$\widetilde{r}^{k+1} = \widetilde{r}^k - \widetilde{\lambda}_k \widetilde{A}\widetilde{p}^k \Leftrightarrow C^{-t}r^{k+1} = C^{-t}r^k - \widetilde{\lambda}_k C^{-t}AC^{-1}Cp^k \Leftrightarrow r^{k+1} = r^k - \widetilde{\lambda}_k Ap^k$;

$\widetilde{p}^{k+1} = \widetilde{r}^{k+1} + \widetilde{\beta}_k \widetilde{p}^k = C^{-t}r^{k+1} + \widetilde{\beta}_k Cp^k = C^{-t}Mr^k + \widetilde{\beta}_k Cp^k = C^{-t}C^t r^k + \widetilde{\beta}_k Cp^k = Cz^{k+1} + \widetilde{\beta}_k Cp^k$. Observe que esta última equação implica em

$$p^{k+1} = z^{k+1} + \widetilde{\beta}_k p^k. \tag{2.19}$$

Apresentamos a seguir o método dos gradientes conjugados precondicionado para o sistema linear (2.4):

Método dos Gradientes Conjugados Precondicionado

Entradas: x^0, A, b, M, onde $M = C^tC$, C não singular.

[0] Calcule o resíduo e as direções iniciais: $r^0 = b - Ax^0$; $Mz^0 = r^0$; $p^0 = z^0$.

Para $k = 0, 1, 2, \ldots$ **faça**

[1] Calcule λ_k de acordo com (2.17).

[2] Atualize o ponto de mínimo (x^{k+1}) de acordo com (2.8).

[3] Atualize o valor do resíduo (r^{k+1}) de acordo com (2.12).

[4] Resolva o sistema linear $Mz^{k+1} = r^{k+1}$.

[5] Calcule β_k de acordo com (2.18).

[6] Atualize a direção (p^{k+1}) de acordo com (2.19).

Fim.

Note que, através da mudança de variável proposta, não utilizamos C explicitamente, mas a matriz $M = C^tC$, a qual é simétrica e definida positiva (por construção), tornando a resolução do sistema mais eficaz.

Visto que a matriz do sistema de equações normais (2.3), $AZ^{-1}XA^t$, é simétrica e definida-positiva, podemos utilizar as técnicas apresentadas neste capítulo para determinar as direções de Newton. Trabalhos interessantes neste sentido podem ser vistos em Mehrotra (1992); Wang e O'Leary (2000).

2.3.2 Precondicionadores

A escolha de um bom precondicionador pode fazer uma grande diferença na velocidade de convergência de um método. Em Oliveira e Sorensen (2005) é desenvolvido o precondicionador separador que é específico para os sistemas lineares oriundos dos métodos de pontos interiores e, quando aplicado ao método dos gradientes conjugados, evita o cálculo da matriz de equações normais e sua fatoração. Este precondicionador apresenta bons resultados próximo a uma solução do problema, mas não é eficiente nas primeiras iterações.

Por outro lado, a fatoração controlada de Cholesky (Campos (1995)) apresenta bons resultados nas iterações iniciais dos métodos de pontos interiores quando as matrizes não são muito mal condicionadas.

Estes precondicionadores foram agregados ao código PCx, resultando em uma abordagem eficiente que obteve melhores resultados onde a fatoração de Cholesky falha ou é mais lenta (Bocanegra *et al.* (2007)).

Nas próximas subseções, descreveremos estes precondicionadores.

2.3.3 Precondicionador Separador

Em Oliveira e Soresen (2005); Oliveira (1997) foi apresentada uma classe de precondicionadores para o sistema aumentado (2.2), a qual pode ser considerada uma generalização do precondicionador proposto por Resende e Veiga (1993) no contexto de problemas de fluxo de custo mínimo para redes. Dentro desta classe, destaca-se o precondicionador separador que será descrito a seguir.

Seja C um precondicionador simétrico em blocos para o sistema aumentado (2.2), tal que:

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

Ao precondicionar a matriz de (2.2) por C, obtemos a seguinte matriz:

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

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

$$PD^{-1}P^t = \begin{pmatrix} D_N^{-1} & 0 \\ 0 & D_B^{-1} \end{pmatrix}. \tag{2.20}$$

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

A matriz precondicionada por C é dada por,

$$C^{-t}\begin{pmatrix} D^{-1} & A^t \\ A & 0 \end{pmatrix}C^{-1} = \begin{pmatrix} -I_n + D^{\frac{1}{2}}A^tG^t + GAD^{\frac{1}{2}} & 0 \\ 0 & -D_B^{-1} \end{pmatrix}, \tag{2.21}$$

sendo $C^{-t} = \begin{pmatrix} D^{\frac{1}{2}} & G \\ H & 0 \end{pmatrix}, G = H^tD^{-\frac{1}{2}}B^{-1}, HP = [I0], AP^t = [BN]$.

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 e, portanto, pode ser facilmente resolvida.

O sistema de dimensão n envolve a matriz indefinida

$$-I_n + D^{\frac{1}{2}}A^tG^t + GAD^{\frac{1}{2}}, \tag{2.22}$$

a qual pode ser escrita na forma

$$K = -I_n + U^tV^t + VU, \tag{2.23}$$

onde $UV = V^tU^t = I_m$ e $U, V^t \in \mathbb{R}^{mxn}$.

A demonstração desta equivalência pode ser vista em Oliveira e Sorensen (2005).

Na sequência, descreveremos as principais características do precondicionador separador.

Redução a um sistema definido positivo

A matriz (2.22) pode ser reduzida a

$$I_m + D_B^{-\frac{1}{2}} B^{-1} N D_N N^t B^{-t} D_B^{-\frac{1}{2}} \tag{2.24}$$

ou

$$I_{n-m} + D_N^{\frac{1}{2}} N^t B^{-t} D_B^{-1} B^{-1} N D_N^{\frac{1}{2}}. \tag{2.25}$$

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 dos métodos iterativos na solução de sistemas lineares.

Equivalência às equações normais

A matriz (2.24) pode ser obtida utilizando o sistema de equações normais. Para isto, seja $A = [BN]P$, onde $P \in \mathbb{R}^{nxn}$ é uma matriz de permutação tal que $B \in \mathbb{R}^{mxm}$ é não singular. Então

$$ADA^t = [BN]PDP^t \begin{bmatrix} B^t \\ N^t \end{bmatrix} = [BN] \begin{bmatrix} D_B & 0 \\ 0 & D_N \end{bmatrix} \begin{bmatrix} B^t \\ N^t \end{bmatrix} = BD_B B^t + ND_N N^t. \tag{2.26}$$

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

$$T = D_B^{-\frac{1}{2}} B^{-1} (ADA^t) B^{-t} D_B^{-\frac{1}{2}} = I_m + \widetilde{W}\widetilde{W}^t, \tag{2.27}$$

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

Note que, próximo a uma solução, ao menos $n-m$ elementos em D^{-1} são grandes. Desta 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 tanto o menor autovalor de T como o maior autovalor de T se aproximam do valor 1, assim como $\mathbb{K}_2(T)$. 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 alternativa seria minimizar $||\widetilde{W}||_2$. 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.

2.3.4 Fatoração Controlada de Cholesky

A fatoração controlada de Cholesky é uma variação da fatoração incompleta de Cholesky. O objetivo deste precondicionador é construir uma matriz precondicionada que possua os autovalores agrupados e próximos do valor 1, de forma a acelerar a convergência do método dos gradientes conjugados. A seguir descreveremos esta fatoração.

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

$$ADA^t = LL^t = \widetilde{L}\widetilde{L}^t + R, \tag{2.28}$$

onde:
L é o fator obtido pela fatoração completa de Cholesky;
$\widetilde{L}$ é o fator obtido pela fatoração incompleta de Cholesky;
R é a matriz de resíduos.

Usando $\widetilde{L}^t$ como uma matriz precondicionadora para ADA^t, temos

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

Definindo $E = L - \widetilde{L}$ e substituindo na equação acima, obtemos

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

Observe que, quando $\widetilde{L} \longrightarrow L$ tem-se $E \longrightarrow 0$ e, portanto, $\widetilde{L}^{-1}ADA^t\widetilde{L}^{-t} \longrightarrow I$.

Duff e Meurant (1989) mostraram que o número de iterações necessárias para a convergência pelo método dos gradientes conjugados está diretamente relacionado com a norma de R, sendo

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

A fatoração controlada de Cholesky é baseada na minimização da norma de Frobenius de E, visto que, $||E||_2 \longrightarrow 0$ implica $||R||_2 \longrightarrow 0$. Assim, consideremos o seguinte problema

$$\text{minimizar} \quad ||E||_F^2 = \sum_{j=1}^{m} c_j, \quad \text{onde} \quad c_j = \sum_{i=1}^{m} |l_{ij} - \widetilde{l}_{ij}|^2. \tag{2.29}$$

Dividindo c_j em dois somatórios temos

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

onde

m_j representa o número de elementos não nulos abaixo da diagonal da coluna j da matriz original; e
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 $\widetilde{L}$. Já no segundo somatório, estão apenas os elementos do fator completo L que não estão em $\widetilde{L}$. Baseado nesses fatos, o problema (2.29) pode ser resolvido através da 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 $\widetilde{L}$ em valor absoluto para η fixo. Desta forma, os maiores elementos ficam no primeiro somatório e os menores no segundo, produzindo um melhor fator $\widetilde{L}$ 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.

CAPÍTULO 3

Detecção de Linhas Redundantes

Antes da aplicação do método de solução escolhido a um problema de programação linear, os problemas são preprocessados para reduzir sua dimensão, detectar possível infactibilidade e aprimorar a estabilidade numérica (Andersen e Andersen (1995); Gondzio (1996); Suhl (1994); Tomlin e Welch (1986)). No preprocessamento, são aplicadas regras, quase sempre simples, que permitem encontrar variáveis com valor fixo, linhas duplicatas, entre outras situações.

Embora as técnicas utilizadas nos trabalhos citados acima encontrem algumas linhas redundantes, nem todas são necessariamente detectadas, por ser este considerado um processo computacionalmente caro.

Neste capítulo, apresentaremos um algoritmo bastante conhecido (Chvátal (1983)), para a detecção de todas as linhas redundantes de uma matriz de restrições, e propostas de implementação deste.

3.1 Linha Redundante

Definição 3.1. *Uma combinação linear de um conjunto de vetores S é uma soma finita $a_1v_1 + \ldots + a_nv_n$, onde:*
$v_1, \ldots, v_n \in S$ e $a_1, \ldots, a_n$ são escalares.

Uma linha da matriz de restrições é dita redundante quando pode ser expressa como uma combinação linear de um subconjunto das demais linhas.

3.2 O Algoritmo

Considere o seguinte problema de programação linear em sua forma padrão:

$$\begin{aligned} \text{Minimizar} \quad & c^t x \\ \text{s.a} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

ou, equivalentemente,

$$\begin{aligned} \text{Minimizar} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.a} \quad & \sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, 2, ..., m) \\ & x_j \geq 0 \quad (j = 1, 2, ..., n) \end{aligned} \tag{3.2}$$

onde $A = [a_{ij}] \in \mathbb{R}^{mxn}$; $x = [x_j]$, $c = [c_j] \in \mathbb{R}^n$ e $b = [b_i] \in \mathbb{R}^m$.

Podemos detectar as linhas redundantes da matriz de restrições A do problema acima (3.1 e 3.2) através da eliminação Gaussiana. Utilizando operações e permutações elementares sobre as linhas de A, a reduzimos para a seguinte forma:

$$\begin{bmatrix} x & : & : & : & : & : \\ 0 & x & : & : & : & : \\ 0 & 0 & x & : & : & : \\ 0 & 0 & 0 & x & : & : \\ 0 & 0 & 0 & 0 & x & : \end{bmatrix} \tag{3.3}$$

Se no decorrer deste processo surgirem linhas nulas na matriz transformada (3.3) estas correspondem as linhas redundantes. Observe que, ao atualizar o vetor b, ou seja, ao aplicar as mesmas operações e permutações realizadas sobre as linhas de A neste vetor, as componentes deste referentes às linhas nulas de (3.3) também devem ser iguais a zero, caso contrário, o problema é infactível.

A desvantagem deste procedimento é que as operações sobre as linhas modifica toda a matriz A, além do grande número de preenchimentos que podem ser gerados durante o procedimento, interferindo na esparsidade e estabilidade numérica do problema (Andersen 1995).

Alternativamente, podemos utilizar uma matriz base B, a qual é gerada a partir de colunas da matriz de restrições. Para isso, adicionamos variáveis artificiais ao problema (3.2) obtendo o problema equivalente:

$$\begin{aligned} & \text{Minimizar} \quad \sum_{j=1}^{n} c_j x_j \\ \text{s.a} \quad & \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i \quad (i = 1, 2, ..., m) \\ & 0 \leq x_{n+i} \leq 0 \quad (i = 1, 2, .., m) \\ & x_j \geq 0 \quad (j = 1, 2, ..., n) \end{aligned} \tag{3.4}$$

onde $A = [a_{ij}] \in \mathbb{R}^{mxn}$; $x = [x_j]$, $c = [c_j] \in \mathbb{R}^n$; $b = [b_i] \in \mathbb{R}^m$ e $x_{n+1}, x_{n+2}, .., x_{n+m}$ são variáveis artificiais.

Observe que os problemas (3.2) e (3.4) são equivalentes, pois as variáveis artificiais são fixadas em zero.

Seja S o conjunto de índices i tais que x_{n+i} é uma variável básica, (inicialmente todas as variáveis artificiais são básicas, logo a matriz base inicial é a matriz identidade de ordem m). Apresentamos então, o algoritmo para detecção de linhas redundantes.

Algoritmo para Detecção de Linhas Redundantes

Entradas: A, B e S.

Repita o procedimento abaixo até que todos os índices pertencentes a S tenham sido utilizados.

[1] Escolha um índice $k \in S$.

[2] Resolva o sistema $B^t r = e_k$, onde e_k é a k-ésima coluna da matriz identidade.

[3] Efetue o produto $r^t A$. Se $r^t A$ for um vetor nulo, volte para o passo 1. Do contrário, existe uma variável não básica x_j tal que $r^t a_{.j} \neq 0$, onde $a_{.j}$ é a j-ésima coluna de A. Substitua a k-ésima coluna de B por $a_{.j}$, substitua x_{n+k} por x_j na base e retorne para o passo 1.

Note que, ao fim deste procedimento, algumas variáveis artificiais x_{n+k} podem permanecer na base. Esta permanência indica que a k-ésima linha de A é uma linha redundante e pode ser retirada do problema. Mais precisamente, seja J o conjunto de índices k tais que x_{n+k} permanece na base, e I o conjunto de índices $1, 2, ..., m$ que não pertencem a J; então toda solução de

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad (i \in I) \tag{3.5}$$

satisfaz todas as equações

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad (i = 1, 2, ..., m).$$

Para demonstrar esta afirmação, considere um índice arbritário $k \in J$. Considere também, r a solução do sistema $B^t r = e_k$. Uma vez que $r^t A$ é um vetor nulo, temos que:

$$\sum_{i=1}^{m} r_i a_{ij} = 0 \quad \forall j = 1, 2, ..., n. \tag{3.6}$$

E como $B^t r = e_k$, temos que $r_k = 1$ e $r_j = 0$ para todo $j \in J$, $j \neq k$. Assim a equação (3.6) pode ser escrita da seguinte forma:

$$a_{kj} = -\sum_{i \in I} r_i a_{ij}$$

e toda solução de (3.5) satisfaz

$$\sum_{j=1}^{n} a_{kj}x_j = \sum_{j=1}^{n} \left(-\sum_{i \in I} r_i a_{ij} \right) x_j = \sum_{i \in I} \left(-r_i \sum_{j=1}^{n} a_{ij}x_j \right) = \sum_{i \in I} (-r_i b_i). \tag{3.7}$$

Em particular, (3.7) deve ser satisfeita por qualquer solução factível $x_1^*, x_2^*, ..., x_n^*$ de (3.2), isto é

$$\sum_{i \in I} (-r_i b_i) = \sum_{j=1}^{n} a_{kj}x_j^* = b_k.$$

Com isto, concluímos que toda solução de (3.5) satisfaz

$$\sum_{j=1}^{n} a_{kj} x_j = b_k,$$

como queríamos demonstrar.

O algoritmo e a demonstração apresentada nesta seção podem ser vistos mais detalhadamente em Chvátal (1983).

Observe que estamos partindo do presuposto que o problema em questão é factível, ou seja, que todas as restrições podem ser satisfeitas. Porém, assim como na eliminação Gaussiana, quando a redundância é detectada, devemos verificar a factibilidade do problema.

Embora este algoritmo seja bastante conhecido, sua implementação direta pode ter um custo computacional bastante elevado, o que o torna menos atraente. Tendo isto em vista, Andersen (1995) apresenta uma proposta de implementação eficiente para este algoritmo. Nas próximas seções, discutiremos esta e outras idéias que visam tornar o algoritmo mais eficaz.

3.3 Construção da Base Inicial

Dois dos fatores que interferem no custo computacional do algoritmo para detecção de linhas redundantes são a dimensão da base inicial e o número de elementos não nulos contidos nesta. Podemos reduzir estes fatores através de algumas observações.

Primeiramente, observamos se há alguma linha nula (linha na qual todos elementos são iguais a zero) na matriz de restrições. Se isso ocorre, estas devem ser declaradas redundantes e retiradas do problema. Geralmente, estas linhas são identificadas nas técnicas tradicionais de preprocessamento.

Posteriormente, verificamos se há alguma coluna única (coluna que contém apenas um elemento não nulo) na matriz de restrições. Claramente, a linha na qual esta coluna ocorre é linearmente independente das demais e pode ser retirada temporariamente do problema e não participar do procedimento de verificação de linhas redundantes. Observe que, quando uma linha é removida, novas colunas únicas podem ser geradas e mais linhas podem ser removidas, o que reduz a dimensão da base inicial.

Outro fator determinante no custo computacional do algoritmo é o número de variáveis artificiais na base inicial, uma vez que o número de iterações depende desta quantidade. Em Andersen (1995) é proposta uma heurística cujo objetivo é construir uma base inicial com o

maior número possível de colunas estruturais. A seguir, apresentaremos tal heurística; porém é importante notar que a heurística deve ser aplicada somente após feitos os procedimentos descritos acima.

3.3.1 Heurística de Andersen

Nesta subseção, descreveremos a heurística proposta por Andersen (1995) para a construção da base inicial.

Inicialmente, o número de elementos não nulos em cada linha e coluna da matriz de restrições é calculado. Associamos uma variável artificial à linha mais densa. Removemos temporariamente esta linha do problema e atualizamos a contagem de elementos não nulos em cada coluna. Caso ocorra alguma coluna única, atribuímos a linha na qual ela ocorre à variável básica estrutural correspondente. Repetimos este procedimento até que cada linha esteja associada a uma variável (artificial ou estrutural). Através desta construção, obtemos uma base inicial triangular superior e não singular.

3.4 Representação da Base

Em todo o algoritmo de detecção de linhas redundantes, a principal operação de cada iteração consiste na resolução de um sistema linear do tipo $B^t r = e_k$. Portanto, é imprescindível manter uma representação da base B, ou de sua inversa, que seja capaz de contribuir para a eficiência do algoritmo.

Uma alternativa seria calcular e armazenar explicitamente a inversa da base (B^{-1}). Embora esta técnica seja conceitualmente simples e correta, torna-se computacionalmente iniviável para problemas de grande porte, visto que é uma técnica muito cara computacionalmente, pois apresenta complexidade $O(m^4)$, onde m é a dimensão da base. A quantidade de memória utilizada depende da dimensão do problema, uma vez que todos os elementos de B^{-1} devem ser armazenados. Além disto, o grande número de preenchimentos que podem ser gerados nos faz perder caracteristícas importantes tais como a esparsidade e estabilidade numérica. Portanto, esta é uma técnica impraticável e deve-se buscar abordagens mais eficientes de representação.

Em Andersen (1995) é proposta a utilização da decomposição LU e sua atualização baseada nas idéias de Bartels-Golub (Bartels *et al.*). Alternativamente, neste trabalho, propomos uma representação baseada na forma produto da inversa.

Nas próximas seções, apresentaremos com mais detalhes tais propostas.

3.5 Proposta 1: Decomposição e Atualização LU

Andersen (1995) propõe a utilização da decomposição LU da base B, ou seja, a representação de B como um produto de duas matrizes triangulares L e U, onde L é uma matriz triangular inferior e U uma matriz triangular superior, ambas não singulares, uma vez que B é não singular.

Assim, o sistema $B^t r = e_k$ pode ser equivalentemente expresso por $(LU)^t r = e_k$ e resolvido em duas etapas:

$$U^t s = e_k, \tag{3.8}$$

$$L^t r = s. \tag{3.9}$$

Para que estas etapas sejam realizadas eficientemente, Andersen (1995) propõe que a inversa de L (L^{-1}) seja armazenada na forma produto (ver próxima seção) e que sejam realizadas permutações na base inicial para que, inicialmente, U tenha a seguinte forma:

$$\begin{bmatrix} x & : & : & : & 0 & 0 & 0 \\ 0 & x & : & : & 0 & 0 & 0 \\ 0 & 0 & x & : & 0 & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Segundo o autor, (3.10) auxilia na resolução dos sistemas (3.8) e (3.9), além de facilitar o processo de atualização da decomposição LU.

Ao final de cada iteração, identificamos se a linha é redundante ou não. Nos casos em que a redundância não é detectada, se faz necessária a atualização da base e, consequentemente, dos fatores L e U. Para que não seja realizada uma nova decomposição LU, Andersen (1995) propõe atualizar a decomposição LU conforme a atualização de Bartels - Golub, apresentada a seguir.

Seja B a base atual. Considere a sua decomposição LU:

$$B = LU \quad \text{ou} \quad L^{-1}B = U. \tag{3.11}$$

Podemos escrever (3.11) da seguinte forma:

$$L^{-1}(b_1...b_p...b_m) = (u_1...u_p...u_m),$$

onde b_i, u_i, $i = 1, ..., m$ são as colunas de B e U, respectivamente, e m a dimensão da base.

Suponha que a coluna b_p de B será substituída por um vetor de mesma dimensão a_q. Com isto, temos:

$$L^{-1}(b_1...a_q...b_m) = (u_1...\alpha_p...u_m)$$

onde $\alpha_p = L^{-1}a_q$.

Observe que, ao substituirmos a p-ésima coluna de U (u_p) por $\alpha_p = L^{-1}a_q$, atualizamos a decomposição LU sem reefetuá-la. Porém, com esta substituição o fator U pode perder sua forma triangular e consequentemente esta deve ser restaurada. As operações necessárias para a restauração da triangularidade de U podem ser armazenadas em forma produto juntamente com a L^{-1}.

Trabalhos interessantes sobre a utilização da decomposição e atualização LU podem ser vistos em Cantante e Oliveira (2007), Reid (1982) e Suhl e Suhl (1990, 1993).

3.6 Proposta 2: Forma Produto

A forma produto, ou forma produto da inversa, foi proposta por Dantzig e Orchard-Hays (1954) ao observarem que, em uma troca de base, a inversa da nova base pode ser obtida a partir da inversa da base atual por uma matriz de transformação elementar.

Uma matriz de transformação elementar é toda matriz que se diferencia da matriz identidade de mesma ordem por uma única linha ou coluna não trivial. Seja E uma matriz de transformação elementar de ordem m, com uma coluna não trivial η na posição k. Então,

$$E = [e_1, e_2, .., \eta, ...e_m] = \begin{bmatrix} 1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & \eta_k & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{bmatrix}, \tag{3.12}$$

onde e_i é a i-ésima coluna da matriz identidade de ordem m.

Seja $B = [b_1, ..., b_m]$ a matriz base de uma dada iteração. Suponha que a j-ésima variável não básica x_j tenha sido selecionada para substituir a k-ésima variável básica. Denotando $\tilde{B}$ como a nova base, tem-se:

$$\tilde{B} = [b_1, .., b_{k-1}, a_{.j}, b_{k+1}, ...b_m],$$

onde $a_{.j}$ é a coluna da matriz de restrições relacionada à variável x_j.

Sendo $a_{.j}$ um vetor m-dimensional, é possível escrevê-lo como a combinação linear das colunas de B:

$$a_{.j} = \sum_{i=1}^{m} v_i b_i = Bv. \tag{3.13}$$

Uma vez que B é não singular temos $v = B^{-1}a_{.j}$.

Se $v_k \neq 0$, então b_k pode ser expresso de acordo com a equação (3.13):

$$b_k = \frac{1}{v_k} a_{.j} - \sum_{i \neq k} \frac{v_i}{v_k} b_i,$$

que é a representação de b_k como combinação linear das colunas de $\tilde{B}$. Definindo o vetor

$$\eta = [-\frac{v_1}{v_k}, ..., -\frac{v_{k-1}}{v_k}, \frac{1}{v_k}, -\frac{v_{k+1}}{v_k}, ..., -\frac{v_m}{v_k}]^t, \tag{3.14}$$

tem-se que $b_k = \tilde{B}\eta$.

Sendo E uma matriz de transformação elementar cuja coluna não trivial é dada por η (3.14), temos:

$$B = \tilde{B}E. \tag{3.15}$$

Invertendo ambos os lados da equação (3.15) temos $B^{-1} = E^{-1}\tilde{B}^{-1}$. Pré-multiplicando esta equação por E obtemos:

$$\tilde{B}^{-1} = EB^{-1}.$$

Assim, a inversa da nova base é obtida pela pré-multiplicação da base atual por uma matriz de transformação elementar.

Tomando como base as idéias descritas acima, propomos uma representação para a base, a qual será descrita na sequência.

Da equação (3.15), temos que:

$$\tilde{B} = BE^{-1}. \tag{3.16}$$

Com isto, podemos escrever também a nova base em função da base atual. Optamos por esta representação e não diretamente pela forma produto pois, desta forma, não é necessária a inversão da base inicial.

Sendo assim, após p atualizações da matriz base, temos:

$$B = B_0E_1^{-1}E_2^{-1}...E_p^{-1},$$

onde: B_0, é a base inicial; $E_i = [e_1, ..., e_{k-1}, \eta_k, e_{k+1}, ..., e_m]^t$; $\eta_k = [-\frac{v_1}{v_k}, ..., -\frac{v_{k-1}}{v_k}, \frac{1}{v_k}, -\frac{v_{k+1}}{v_k}, ..., -\frac{v_m}{v_k}]^t$.

Portanto, o sistema $B^tr = e_k$ pode ser resolvido em duas etapas:

$$(E_1^{-1}E_2^{-1}...E_p^{-1})^t y = e_k, \tag{3.17}$$

$$B_0^t r = y. \tag{3.18}$$

Onde (3.17) pode ser expresso como $y^t = e_k(E_p...E_2E_1)$.

A técnica escolhida deve levar em conta a esparsidade e a estabilidade numérica da representação, e o tempo computacional para obtê-la e atualiza-la deve ser o menor possível. Em ambas as propostas apresentadas neste capítulo, tais fatores são considerados. Entretanto, a proposta 2 nos parece mais vantajosa.

Em geral, a forma produto da inversa é vista como uma representação menos esparsa e estável em relação a decomposição LU e sua atualização. Porém, neste caso, não há grandes

diferenças em relação a estes fatores (esparsidade e estabilidade numérica) entre as técnicas propostas, visto que a base é atualizada poucas vezes, em no máximo o número de variáveis artificiais presentes na base inicial.

Logo, a utilização de uma técnica de implementação relativamente simples como a apresentada na proposta 2 torna o algoritmo mais eficiente computacionalmente, pois o esforço e o tempo computacional serão menores.

Outro ponto favorável é a utilização da estrutura triangular da base inicial. Com as permutações propostas por Andersen (1995) para que o fator U inicialmente tenha a forma (3.10), pode fazer com que a base inicial perca sua forma triangular, sendo necessária, de fato, a decomposição LU desta. Em nossa proposta (proposta 2) a forma triangular da base inicial é preservada e com isto temos $B = LU = IU$, onde I é a matriz identidade e U uma matriz triangular superior conhecida.

Por fim, devido aos fatores descritos acima utilizaremos a proposta 2 em nossas implementações, as quais serão descritas nos próximos capítulos.

CAPÍTULO 4

Experimentos Numéricos

Neste capítulo, apresentaremos os experimentos numéricos realizados e seus respectivos resultados. Integramos o algoritmo de detecção de linhas redundantes às rotinas de pre-processamento do código PCx modificado. A fim de verificar a eficácia do algoritmo, este foi comparado com o procedimento implementado em Bocanegra *et al.* (2007); Oliveira e Sorensen (2005).

Os detalhes são apresentados na sequência.

4.1 O Código PCx

O código PCx (Czyzyk *et al.* (1999)) foi desenvolvido no Optimization Technology Center at Argonne National Laboratory and Northwestern University e implementa o método primal-dual preditor-corretor (Mehrotra (1992)) com múltiplas correções (Gondzio (1996)). 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 utilizam uma biblioteca para fatoração esparsa de Cholesky, desenvolvida por Ng e Peyton (1993) 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 das rotinas de preprocessamento converte os modelos em

um formato padrão e após ser encontrada a solução é transformada em termos da formulação original.

4.1.1 O Código PCx Modificado

O código PCx modificado possui as alterações descritas em Oliveira e Soresen (2005); Bocanegra *et al.* (2007). Estas modificações são:

- A eliminação das múltiplas correções de centralidade (Gondzio (1996)) 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 corretor e são resolvidos dois sistemas a cada iteração.
- A resolução dos dois sistemas lineares é feita pelo método dos gradientes conjugados, precondicionado inicialmente pela fatoração controlada de Cholesky e posteriormente pelo precondicionador separador.

4.2 O Preprocessamento

Modelos de programação linear frequentemente contêm informações redundantes, as quais podem ser identificadas com técnicas não muito sofisticadas, conhecidas como preprocessamento.

O código PCx possue um conjunto de rotinas de preprocessamento, as quais fazem uso das técnicas descritas por Andersen e Andersen (1995). Para isto, os problemas são convertidos para a seguinte forma:

$$\begin{array}{lrl} \text{minimizar} & c^t x & \\ \text{s.a} & Ax & = b, \\ & x_i & \geq 0 \quad i \in N, \\ & 0 \leq x_i \leq u_i & \quad i \in U, \\ x_i & \text{livre} \quad i \in F & \end{array} \tag{4.1}$$

onde: $A \in \mathbb{R}^{mxn}$, $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $N \cup U \cup F$ é uma partição do conjunto de índices 1,2,...n.

Após esta conversão, o problema é preprocessado, ou seja, é aplicada ao problema uma série de rotinas cujo objetivo é a identificação dos seguintes fatores:

- **Infactibilidade:** verifica-se para cada limite superior u_i, $i \in U$, se $u_i \geq 0$ e se, a cada linha nula de A, há um zero correspondente no vetor b.

- **Linha nula:** se a matriz A possui uma linha nula e um zero correspondente no vetor b, esta pode ser removida do problema.

- **Linha duplicata:** quando uma linha de A (e seu elemento correspondente no vetor b) é múltiplo de outra linha (e seu respectivo elemento no vetor b), uma destas linhas pode ser removida do problema.

- **Coluna duplicata:** quando uma coluna de A é múltiplo de outra coluna e as variáveis correspondentes são maiores ou iguais a zero, estas colunas podem ser combinadas. A variável primal resultante desta combinação poderá ser maior ou igual a zero ou livre de sinal.

- **Coluna nula:** se há uma coluna em que todos os elementos são nulos, a variável correspondente pode ser fixada em alguns de seus limitantes ou o problema é ilimitado.

- **Variável fixa:** quando ambos os limites (inferior e superior) de uma variável são iguais. Neste caso, o valor da variável é facilmente determinado podendo esta ser retirada do problema.

- **Linha única:** quando uma linha contém um único elemento (a_{ij}) não nulo, temos $x_j = b_i/a_{ij}$ e, portanto, a variável x_j pode ser removida do problema, assim como a linha única.

- **Coluna única:** quando uma coluna contém um único elemento não nulo a_{ij}, e a variável correspondente a esta coluna (x_j) é uma variável livre, podemos expressá-la em função das demais variáveis e eliminá-la do problema. Se x_j não é livre, mas seus limitantes são mais "fracos" em relação aos demais limitantes das variáveis presentes na i-ésima linha de A, esta também pode ser eliminada do problema.

- **Restrição dominante:** às vezes, uma restrição linear representada pela i-ésima linha de A "força" todas as variáveis para seu limite inferior ou superior. Por exemplo, considere a restrição $10x_3 - 4x_{10} + x_{12} = -4$ tal que, $x_3 \in [0, +\infty)$, $x_{10} \in [0, 1]$, $x_{12} \in [0, +\infty)$. Neste caso, necessariamente $x_3 = 0$, $x_{10} = 1$, $x_{12} = 0$. Então, estas três variáveis e a linha correspondente à restrição podem ser retiradas do problema.

Este procedimento é repetido até que nenhum destes fatores sejam identificados.

4.3 Implementação

Para avaliar o desempenho do algoritmo de detecção de linhas redundantes, o integramos às rotinas de preprocessamento do código PCx modificado. Após o encerramento do procedimento descrito na seção anterior, dá-se início ao processo de detecção de linhas redundantes.

Primeiramente, são identificadas e retiradas temporariamente do problema as colunas únicas assim como as linhas onde elas ocorrem. A seguir, é aplicada a heurística descrita na Subseção 3.3.1 e construída a base inicial. A escolha dos índices pertencentes ao conjunto S, passo [1] do algoritmo de detecção de linhas redundantes (ver Seção 3.2), foi feita em ordem decrescente. E para representação e atualização da base, assim como para a resolução dos sistemas lineares presentes no algoritmo, foi utilizada a proposta 2, descrita na Seção 3.6, devidos aos motivos descritos naquela seção.

Devido aos erros de arredondamento no passo [3] do algoritmo substituímos a verificação de redundância $r^t a_{.j} \neq 0$ por $|r^t a_{.j}| \leq \varepsilon$. Onde ε é um parâmetro a ser escolhido. Utilizamos $\varepsilon = 10^{-9}$. Caso aja mais de uma coluna tal que $|r^t a_{.j}| \leq 10^{-9}$, optamos pela coluna mais esparsa.

A fim de verificar a interferência da heurística de Andersen (Subseção 3.3.1) no desempenho do algoritmo, também implementamos o procedimento descrito acima, alterando a heurística da seguinte forma; ao invés de associarmos variáveis artificiais as linhas mais densas, estas foram associadas as linhas mais esparsas. A base inicial gerada permanece triangular superior e não singular.

Por fim, como parâmetro de comparação integramos o procedimento descrito em (Bocanegra *et al.* (2007); Oliveira e Sorensen (2005)) ao preprocessamento do código PCx modificado. Este é um procedimento pouco sofisticado que detecta as linhas redundantes por meio de uma decomposição LU retangular da matriz de restrições, após eliminar recursivamente as linhas de elementos de colunas únicas.

Nas próximas seções, apresentaremos e avaliaremos o desempenho destes procedimentos. As implementações foram feitas em linguagem C e testadas em uma plataforma Intel(R) Pentium(R) D 3.40GHz e com 2Gb de memória RAM, em Linux, utilizando compiladores gcc e gfortran.

4.4 Problemas Testes

Os problemas usados como testes são apresentados nas tabelas a seguir. Todos os problemas são de domínio públicos podem ser encontrados em NETLIB (NETLIB collection LP test sets), MISC (Miscellaneous LP models) e QAP (Burkard *et al.* (1991)). Optamos por alguns dos problemas de maior porte destas coleções, uma vez que, o foco do trabalho são problemas de grande porte. Também foi feita a escolha de alguns problemas de médio e pequeno porte, com o intuito de avaliar o desempenho do algoritmo nestes tipos de problemas.

Tabela 4.1: Problemas testes

Problema	Linhas	Colunas	Nnulos	Coleção
PDS-30	49941	158489	340635	MISC
PDS-40	66844	217531	466800	MISC
PDS-50	83060	275814	590833	MISC
PDS-60	99431	336421	719557	MISC
PDS-70	114944	390005	833465	MISC
PDS-80	129181	434580	927826	MISC
PDS-90	142823	475448	1014136	MISC
DEGEN2	444	757	4201	NETLIB
DEGEN3	1503	2604	25432	NETLIB
DFL001	6701	12230	35632	NETLIB
PDS-10	16558	49932	107605	NETLIB
PDS-20	33874	108175	232647	NETLIB
QAP8	912	1632	7296	NETLIB
QAP12	3192	8856	38304	NETLIB
QAP15	6330	22275	94950	NETLIB
SCORPION	388	466	1534	NETLIB
SIERRA	1227	2735	8001	NETLIB
KRA30A	18059	85725	337920	QAP
NUG05	210	255	1050	QAP
NUG05-3rd	1410	1425	6450	QAP

Tabela 4.2: Continuação da Tabela Problemas testes

Problema	Linhas	Colunas	Nnulos	Coleção
NUG06	372	486	2232	QAP
NUG06-3rd	3972	4686	22032	QAP
NUG07	602	931	4214	QAP
NUG07-3rd	9422	12691	61544	QAP
NUG08	912	1632	7296	QAP
NUG12	3192	8856	38304	QAP
NUG15	6330	22275	94950	QAP
NUG30	21899	141405	568320	QAP
ROU20	7359	37640	152980	QAP
STE36A	27683	131076	512640	QAP

4.5 Resultados Computacionais

Nesta seção, apresentaremos os resultados obtidos através dos experimentos numéricos realizados.

Embora o preprocessamento seja imprescindível no processo de resolução de problemas de programação linear de grande porte, claramente, há uma perda quanto ao fator tempo. Detectar redundâncias é um processo computacionalmente caro, que pode adicionar um excessivo tempo computacional na resolução dos problemas, comprometendo a eficiência do método escolhido. O ideal seria detectar e remover os tipos de redundância que conduzem a uma redução no tempo total de solução. No entanto, nem sempre isso é possível, portanto, a estratégia conservadora de detectar simples formas de redundância, de forma rápida, pode ser a melhor estratégia. Porém, esta não é suficiente em alguns casos, como por exemplo, no contexto em que estamos inseridos: a resolução dos problemas por métodos de pontos interiores solucionando seus sistemas lineares por métodos iterativos. Neste caso, as linhas redundantes que não são detectadas pelos métodos tradicionais de preprocessamento impossibilitam a resolução dos problemas. É importante enfatizar que existem problemas de grande porte em que esta é a única forma de solução, destacando a importância da detecção

e remoção de linhas redundantes.

Tendo isto em vista, em nossos experimentos numéricos, priorizamos o tempo computacional necessário para a realização dos procedimentos como fator de análise de eficiência. Nos propomos a apresentar um procedimento rápido e eficaz de detecção de linhas redundantes e que conduzisse a uma redução do tempo total de solução. Portanto, para avaliar o desempenho de nossa proposta, para cada problema teste, foram realizados experimentos numéricos, nos quais, foi medido o tempo de processamento e solução, o número de linhas redundantes detectadas e dados referentes a dimensão dos problemas. O mesmo foi feito para o procedimento descrito em Bocanegra *et al.* (2007); Oliveira e Sorensen (2005) que nos servirá de parâmetro de comparação.

Denominamos as rotinas de detecção de linhas redundantes implementadas neste trabalho como, LU retangular (procedimento descrito em Bocanegra *et al.* (2007); Oliveira e Sorensen (2005)), Redundante e Redundante Modificado (procedimentos descritos no presente trabalho). A rotina Redundante Modificado é o procedimento no qual a heurística de Andersen é modificada (ver Seção 4.3).

Na Tabela 4.3 são apresentados dados sobre a dimensão dos problemas testes após o preprocessamento. Nesta tabela as colunas PLinhas, PColunas e PNnulos representam respectivamente, o número de linhas, colunas e elementos não nulos após a aplicação das rotinas de preprocessamento do código PCx (Seção 4.2). Já as colunas DNnulos e PRNnulos representam, respectivamente, a diferença e o percentual de redução de elementos não nulos em relação à formulação original dos problemas.

Na sequência, a Tabela 4.4 apresenta dados de dimensão dos problemas testes após a detecção de linhas redundantes. A coluna RLinhas representa o número de linhas após a aplicação das rotinas de detecção de linhas redundantes. As colunas PRNnulos1, PRNnulos2 e PRNnulos3, representam, respectivamente, o percentual de redução de elementos não nulos em relação à formulação original dos problemas quando são utilizadas as rotinas LU Retangular, Redundante e Redundante Modificado. Por fim, a coluna LR representa o número de linhas redundantes identificadas.

A Tabela 4.5 apresenta a dimensão da matriz base (B) e o número de iterações do algoritmo de detecção de linhas redundantes quando este é implementado diretamente e quando a heurística de Andersen (com e sem alterações) é utilizada. Nesta tabela, a coluna B1 indica a dimensão da base e o número de iterações do algoritmo caso este fosse implementado diretamente. B2 representa a dimensão da base quando a heurística é utilizada (com e

sem alterações). Já as colunas VA1 e VA2 mostram, respectivamente, o número de colunas artificiais presentes na base inicial quando a heurística é aplicada diretamente e com modificações. Observe que, o número de colunas artificiais presentes na base inicial é o número de iterações do algoritmo.

Na Tabela 4.6 são apresentados dados sobre o tempo de preprocessamento. A coluna TP representa o tempo (em segundos) utilizado pelas rotinas de preprocessamento do código PCx. TLR1, TLR2 e TLR3 representam, respectivamente, o tempo (em segundos) utilizado para detectar as linhas redundantes quando são aplicadas as rotinas LU Retangular, Redundante e Redundante Modificado. Já as colunas TTP1, TTP2 e TTP3 representam, respectivamente, a soma dos tempos de preprocessamento e detecção de linhas redundantes quando são utilizadas as rotinas LU Retangular, Redundante e Redundante Modificado.

Na sequência, a Tabela 4.7 apresenta dados sobre o tempo de solução. Nesta tabela, as colunas TS1, TS2 e TS3 representam, respectivamente, o tempo (em segundos) utilizado pelo método preditor-corretor para solucionar os problemas quando para a detecção de linhas redundantes aplicamos as rotinas LU Retangular, Redundante e Redundante Modificado. IT1, IT2 e IT3 representam, respectivamente, o número de iterações do método preditor-corretor necessárias para solucionar os problemas quando são utilizadas as rotinas LU Retangular, Redundante e Redundante Modificado para a identificação das linhas redundantes.

Por fim, a Tabela 4.8 apresenta o tempo total de solução. As colunas TT1, TT2 e TT3 representam, respectivamente, o tempo total (em segundos) utilizado para solucionar os problemas quando identificamos as linhas redundantes através das rotinas LU Retangular, Redundante e Redundante Modificado.

Através das Tabelas 4.3 e 4.4 podemos perceber a importância da detecção e remoção das linhas redundantes, em especial, no contexto em que estamos inseridos. A maioria dos problemas teste apresentaram linhas redundantes e em alguns casos, (QAP8, QAP12, QAP15, NUG05, NUG05-3rd, NUG06, NUG06-3rd, NUG07, NUG07-3rd, NUG08, NUG12 e NUG15) nenhum tipo de redundância foi detectada pelas rotinas de preprocessamento (ver Tabela 4.3), porém, foi identificado um número elevado de linhas redundantes (Tabela 4.4), as quais geram uma série de complicações no processo de solução. Enfatizando que as técnicas tradicionais de preprocessamento não são suficientes em todos os contextos.

Atém disso, com a detecção de linhas redundantes reduzimos em média 8% o número de elementos não nulos.

Já a comparação apresentada na Tabela 4.5 retoma a discussão feita no Capítulo 3, de

que a implementação direta do algoritmo de detecção de linhas redundantes possui um custo computacional elevado. Conforme discutido neste capítulo, fatores como a dimensão da base inicial e o número de colunas artificiais presentes nesta interferem diretamente no custo computacional do algoritmo. Tendo isso em vista, Andersen (1995) propõe uma heurística que tem por objetivo diminuir estas quantidades (dimensão da base inicial e número de colunas artificiais) e consequentemente tornar o método mais eficaz. Por meio dos dados da Tabela 4.5 podemos observar que houve uma redução significativa destes fatores, o que mostra que a heurística proposta (mesmo com as alterações realizadas neste trabalho) de fato contribui para a eficácia do algoritmo.

Analisando agora o fator tempo, que conforme discutido anteriormente será nosso parâmetro de desempenho, podemos verificar que a rotina tradicional de detecção de linhas redundantes (LU Retangular) mostrou-se pouco eficiente. Apresenta elevados tempos de execução (Tabela 4.6) e para a maioria dos problemas, o tempo utilizado pelo método preditor-corretor para solucionar os problemas é maior quando este procedimento é utilizado (Tabela 4.7), comprometendo a eficiência do método de solução. Em contrapartida, nossas propostas (Redundante e Redundante Modificado) apresentaram um desempenho bastante satisfatório. Em comparação com a rotina LU Retangular, mostaram-se em média, três vezes mais rápidas, chegando a ser sete vezes mais rápidas no caso do problema NUG07-3rd. Entretanto, houve uma pequena queda de desempenho quando o problema não apresenta linhas redundantes (KRA30A, ROU20 e STE3EA). Uma perda pequena comparada com os ganhos nos casos em que estas linhas estão presentes. Em relação ao tempo de solução, o método preditor-corretor dispõe de um tempo menor para solucionar a maioria dos problemas teste quando nossas rotinas de detecção de linhas redundantes são aplicadas, em especial, a Redundante Modificado (Tabela 4.7). Reduzindo o tempo total de solução, conforme apresentado na Tabela 4.8.

Quanto ao número de iterações realizadas para solucionar os problemas teste não houve grandes variações independente do procedimento de detecção de linhas redundantes adotado. Exceto para os problemas DEGEN3 e NUG15. Nestes casos, quando nossas rotinas são utilizadas para a detecção de linhas redundantes, o método preditor-corretor necessita de um número de iterações bem maior em relação ao número de iterações realizadas quando a LU Retangular é utilizada para a identificação das linhas redundantes. O que justifica o desempenho superior da LU retangular em relação às rotinas Redundante e Redundante Modificado nestes problemas.

Tabela 4.3: Dados dos problemas testes após o preprocessamento

Problema	PLinhas	PColunas	PNnulos	DNnulos	PRNnulos
PDS-30	47968	156042	334310	6325	2%
PDS-40	64276	214385	458591	8209	2%
PDS-50	80339	272513	582206	8627	1%
PDS-60	96514	332862	710234	9323	1%
PDS-70	111896	386238	823582	9883	1%
PDS-80	126120	430800	917908	9918	1%
PDS-90	139752	471538	1003958	10178	1%
DEGEN2	444	757	4201	0	0%
DEGEN3	1503	2604	25432	0	0%
DFL001	5984	12143	35338	294	1%
PDS-10	15648	48780	104770	2835	3%
PDS-20	32287	106180	227541	5106	2%
QAP8	912	1632	7296	0	0%
QAP12	3192	8856	38304	0	0%
QAP15	6330	22275	94950	0	0%
SCORPION	340	412	1339	195	13%
SIERRA	1212	2705	7931	70	1%
KRA30A	18059	85725	337920	0	0%
NUG05	210	225	1050	0	0%
NUG05-3rd	1410	1425	6450	0	0%
NUG06	372	486	2232	0	0%
NUG06-3rd	3972	4686	22032	0	0%
NUG07	602	931	4214	0	0%
NUG07-3rd	9422	12691	61544	0	0%
NUG08	912	1632	7296	0	0%
NUG12	3192	8856	38304	0	0%
NUG15	6330	22275	94950	0	0%
NUG30	21899	141405	568320	0	0%
ROU20	7359	37640	152980	0	0%
STE36A	27683	131076	512640	0	0%

PLinhas: número de linhas após o preprocessamento.

PColunas: número de colunas após o preprocessamento.

PNnulos: número de elementos não nulos após o preprocessamento.

DNnulos: diferença de elementos não nulos em relação à formulação original.

PRNnulos: percentual de redução de elementos não nulos em relação à formulação original.

Tabela 4.4: Dados dos problemas testes após a detecção de linhas redundantes

Problema	RLinhas	PRNnulos1	PRNnulos2	PRNnulos3	LR
PDS-30	47957	2%	2%	2%	11
PDS-40	64265	2%	2%	2%	11
PDS-50	80328	2%	2%	1%	11
PDS-60	96503	1%	1%	1%	11
PDS-70	111885	1%	1%	1%	11
PDS-80	126109	1%	1%	1%	11
PDS-90	139741	1%	1%	1%	11
DEGEN2	442	1%	1%	0%	2
DEGEN3	1501	0%	1%	0%	2
DFL001	5971	1%	1%	1%	13
PDS-10	15637	4%	3%	3%	11
PDS-20	32276	3%	3%	2%	11
QAP8	742	19%	19%	19%	170
QAP12	2794	12%	12%	12%	398
QAP15	5698	10%	10%	10%	632
SCORPION	317	19%	19%	19%	23
SIERRA	1202	3%	4%	1%	10
KRA30A	18059	0%	0%	0%	0
NUG05	148	30%	30%	30%	62
NUG05-3rd	1208	14%	14%	14%	202
NUG06	280	25%	25%	25%	92
NUG06-3rd	3540	11%	11%	11%	432
NUG07	474	21%	21%	21%	128
NUG07-3rd	8594	9%	9%	9%	828
NUG08	742	19%	19%	19%	170
NUG12	2794	12%	12%	12%	398
NUG15	5698	10%	10%	10%	632
NUG30	21899	0%	0%	0%	0
ROU20	7359	0%	0%	0%	0
STE36A	27683	0%	0%	0%	0

RLinhas: número de linhas após a detecção de linhas redundantes.

PRNnulos1: percentual de redução de elementos não nulos (LU Retangular).

PRNnulos2: percentual de redução de elementos não nulos (Redundante).

PRNnulos3: percentual de redução de elementos não nulos (Redundante Modificado).

LR: número de linhas redundantes detectadas.

Tabela 4.5: Comparação entre dimensões e número de iterações

Problema	B1	B2	VA1	VA2
PDS-30	47968	44795	13	44
PDS-40	64276	60030	13	44
PDS-50	80339	75070	13	44
PDS-60	96514	90226	13	44
PDS-70	111896	104724	13	44
PDS-80	126120	118342	13	44
PDS-90	139752	131499	13	44
DEGEN2	444	97	3	4
DEGEN3	1503	159	2	4
DFL001	5984	4275	15	19
PDS-10	15648	14597	13	44
PDS-20	32287	30088	13	44
QAP8	912	912	394	394
QAP12	3192	3192	1190	1190
QAP15	6330	6330	2207	2207
SCORPION	340	116	87	87
SIERRA	1212	515	13	15
KRA30A	18059	5009	165	167
NUG05	210	210	117	117
NUG05-3rd	1410	810	362	417
NUG06	372	372	203	203
NUG06-3rd	3972	2172	949	1043
NUG07	602	602	324	324
NUG07-3rd	9422	5012	2142	2284
NUG08	912	912	486	486
NUG12	3192	3192	1664	1664
NUG15	6330	6330	3272	3272
NUG30	21899	8849	293	323
ROU20	7359	3559	176	196
STE36A	27683	6226	172	173

B1: dimensão da base e número de iterações (algoritmo implementado diretamente).

B2: dimensão da base (algoritmo implementado juntamente com a heurística).

VA1: número de colunas artificiais na base inicial (heurística implementada diretamente).

VA2: número de colunas artificiais na base inicial (heurística com alterações).

Tabela 4.6: Resultados de tempo de preprocessamento para os problemas teste

Problema	TP	TLR1	TLR2	TLR3	TTP1	TTP2	TTP3
PDS-30	0,28	42,71	**19,19**	19,45	42,99	19,47	19,73
PDS-40	0,29	78,99	**35,33**	35,62	79,18	35,62	35,91
PDS-50	0,37	134,58	**55,92**	56,08	134,95	56,29	56,45
PDS-60	1,16	213,72	**82,56**	83,12	214,88	83,72	84,28
PDS-70	0,64	308,85	111,42	**111,35**	309,49	112,06	111,99
PDS-80	0,24	405,13	**141,11**	142,27	405,37	141,35	142,51
PDS-90	0,27	531,87	172,08	**170,56**	532,14	172,35	170,83
DEGEN2	0,00	**0,00**	**0,00**	**0,00**	0,00	0,00	0,00
DEGEN3	0,02	**0,00**	**0,00**	**0,00**	0,02	0,02	0,02
DFL001	0,05	0,25	**0,18**	**0,18**	0,30	0,23	0,23
PDS-10	0,08	4,28	**1,99**	2,07	4,36	2,07	2,15
PDS-20	0,10	18,38	**8,82**	9,04	18,48	8,92	9,14
QAP8	0,01	0,35	**0,06**	0,08	0,36	0,07	0,09
QAP12	0,05	3,87	**0,74**	0,90	3,92	0,79	0,95
QAP15	0,08	18,59	**3,55**	4,24	18,67	3,63	4,32
SCORPION	0,00	**0,00**	**0,00**	**0,00**	0,00	0,00	0,00
SIERRA	0,01	**0,01**	**0,01**	**0,01**	0,02	0,02	0,02
KRA30A	0,14	**0,16**	1,76	1,75	0,30	1,90	1,89
NUG05	0,01	**0,00**	**0,00**	**0,00**	0,01	0,01	0,01
NUG05-3rd	0,00	0,10	**0,06**	0,07	0,10	0,06	0,07
NUG06	0,01	0,02	0,02	**0,01**	0,03	0,03	0,02
NUG06-3rd	0,00	1,54	0,38	**0,37**	1,54	0,38	0,37
NUG07	0,00	0,06	**0,04**	**0,04**	0,06	0,04	0,04
NUG07-3rd	0,00	17,34	2,56	**2,46**	17,34	2,56	2,46
NUG08	0,01	0,16	**0,10**	**0,10**	0,17	0,11	0,11
NUG12	0,01	5,17	1,31	**1,23**	5,18	1,32	1,24
NUG15	0,00	46,45	6,37	**6,20**	46,45	6,37	6,20
NUG30	0,25	**0,48**	5,20	5,39	0,73	5,45	5,64
ROU20	0,02	**0,09**	0,66	0,72	0,11	0,68	0,74
STE36A	0,15	**0,26**	3,16	3,11	0,41	3,31	3,26

TP: tempo (em segundos) de preprocessamento (PCx).

TLR1, TLR2, TLR3: tempo (em segundos) de detecção de linhas redundantes (LU Retangular, Redundante e Redundante Modificado).

TTP1, TTP2, TTP3: tempo total de preprocessamento (LU Retangular, Redundante e Redundante Modificado).

Tabela 4.7: Tempo e número de iterações utilizados pelo método preditor-corretor para solucionar os problemas teste

Problema	TS1	TS2	TS3	IT1	IT2	IT3
PDS-30	209,94	206,86	**202,21**	37	36	39
PDS-40	348,61	364,92	**341,65**	33	35	39
PDS-50	586,72	572,54	**515,53**	36	36	36
PDS-60	855,07	816,81	**762,21**	35	37	35
PDS-70	1095,48	1129,07	**1035,76**	34	34	37
PDS-80	**1511,35**	1524,60	1528,57	37	37	37
PDS-90	**1831,40**	1892,86	1948,33	37	37	37
DEGEN2	**0,73**	0,74	0,89	12	12	12
DEGEN3	**19,01**	31,19	30,12	16	41	41
DFL001	**25,64**	25,66	26,41	16	16	16
PDS-10	96,50	**94,40**	97,63	42	42	42
PDS-20	182,02	182,74	**135,76**	42	42	42
QAP8	2,72	3,03	**2,66**	10	9	9
QAP12	208,43	220,65	**176,32**	13	13	13
QAP15	12733,22	12105,41	**11348,63**	42	41	41
SCORPION	0,10	**0,08**	**0,08**	12	11	11
SIERRA	0,24	0,24	**0,23**	13	13	13
KRA30A	**1207,07**	**1207,07**	**1207,07**	41	41	41
NUG05	0,08	**0,07**	0,09	7	7	7
NUG05-3rd	**2,87**	5,32	6,48	6	7	7
NUG06	**0,25**	0,27	0,26	6	6	6
NUG06-3rd	292,79	**194,47**	321,50	8	7	8
NUG07	1,35	0,97	**0,96**	12	10	10
NUG07-3rd	13871,49	**6481,37**	15394,99	10	8	8
NUG08	**1,83**	2,19	2,19	9	9	9
NUG12	**142,95**	155,03	157,32	13	13	13
NUG15	**2170,77**	11273,28	11086,19	13	41	41
NUG30	**3164,08**	**3164,08**	**3164,08**	41	41	41
ROU20	**18549,69**	**18549,69**	**18549,69**	41	41	41
STE36A	**787,41**	**787,41**	**787,41**	40	40	40

TS1, TS2, TS3:tempo (em segundos) de solução
(LU Retangular, Redundante e Redundante Modificado).
IT1, IT2, IT3:número de iterações
(LU Retangular, Redundante e Redundante Modificado).

Tabela 4.8: Tempo total utilizado para solucionar os problemas teste

Problema	TT1	TT2	TT3
PDS-30	252,93	226,33	**221,94**
PDS-40	472,79	400,54	**377,56**
PDS-50	721,67	628,83	**571,98**
PDS-60	1069,95	900,53	**846,49**
PDS-70	1404,97	1241,13	**1147,75**
PDS-80	1916,72	**1665,95**	1671,08
PDS-90	2363,54	**2065,21**	2119,66
DEGEN2	**0,73**	0,74	0,89
DEGEN3	**19,03**	31,21	30,14
DFL001	25,94	**25,89**	26,64
PDS-10	100,86	**96,47**	99,78
PDS-20	200,50	191,66	**144,90**
QAP8	3,08	3,10	**2,75**
QAP12	212,35	221,44	**177,27**
QAP15	12751,89	12109,04	**11352,95**
SCORPION	0,10	**0,08**	**0,08**
SIERRA	0,26	0,26	**0,25**
KRA30A	1207,37	1208,97	**1208,96**
NUG05	0,09	**0,08**	0,10
NUG05-3rd	**2,97**	5,38	6,55
NUG06	**0,28**	0,30	**0,28**
NUG06-3rd	294,33	**194,85**	321,87
NUG07	1,41	1,01	**1,00**
NUG07-3rd	13888,83	**6483,93**	15397,45
NUG08	**2,00**	2,30	2,30
NUG12	**148,13**	156,35	158,56
NUG15	**2217,22**	11279,65	11092,39
NUG30	**3164,81**	3169,53	3169,72
ROU20	**18549,80**	18550,37	18550,43
STE36A	**787,82**	790,72	790,67

TT1:tempo (em segundos) total de solução (LU Retangular).

TT2:tempo (em segundos) total de solução (Redundante).

TT3:tempo (em segundos) total de solução (Redundante Modificado).

CAPÍTULO 5

Conclusões e Trabalhos Futuros

Neste trabalho, foram apresentadas propostas de implementação de um procedimento de detecção de linhas redundantes (Chvátal (1983)) a uma adaptação própria do código PCx que resolve problemas de programação linear pelo método primal-dual preditor corretor e seus respectivos sistemas lineares através do método dos gradientes conjugados precondicionado, tendo em vista, a importância dos problemas que estas linhas podem gerar no processo de solução de modelos de programação linear, em especial os de grande porte, modelos estes com vasta aplicação prática; e do preprocessamento de forma geral.

As propostas mostraram-se eficientes em termos de tempo computacional, fator de extrema importância quando estamos tratando de preprocessamento em programação linear (Andersen e Andersen (1995)), ao ser comparada com um procedimento tradicional de detecção de linhas redundantes (Bocanegra *et al.* (2007); Oliveira e Sorensen (2005)), baseado na eliminação gaussiana.

A diferença de desempenho entre os procedimentos, aumentou de acordo com a dimensão do problema, tornando-se cada vez mais significativa nos problemas teste de maior porte. De forma geral, nossas propostas mostraram-se cerca de três vezes mais rápida em comparação ao outro procedimento testado, enfatizando que o procedimentos apresentados no presente trabalho pode ser uma alternativa eficaz para a resolução do problema em questão: detecção e remoção de linhas redundantes em problemas de grande porte.

Além disto, nossos procedimentos interferiram de forma positiva no processo de solução dos problemas. Ao aplica-los para a detecção de linhas redundantes, reduzimos o tempo

utilizado pelo método preditor-corretor para solucionar os problemas, na maioria dos casos, conduzindo a uma redução do tempo total de solução.

Como trabalho futuro, propomos a implementação do procedimento de detecção de linhas redundantes descrito no artigo de Andersen (1995) para uma possível comparação entre este e os procedimentos descritos neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ADLER, I.; MONTEIRO, R. D. C. Interior path-following primal-dual algorithms, part 1: linear programming. *Mathematical Programming*, v. 44, p. 27-41, 1989.

ADLER, I.; RESENDE, M. G. C.; VEIGA, G.; KARMARKAR, N. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, v. 44, p. 297-335, 1989.

ANDERSEN, E. D. Finding all linearly dependent rows in large-scale linear programming. *Optimization Methods and Softwares*, v. 6, p. 219-227, 1995.

ANDERSEN, E. D.; ANDERSEN, K. D. Presolving in linear programming. *Mathematical Programming*, v. 71, p. 221-245, 1995.

ANSTREICHER, K. M. Potentional reduction methods. *In T. Terlaky, editor, Interior Point Methods in Mathematical Programming*, Klewer Academic Publishers, 1996.

BARTELS, R. H.; GOLUB, G. H.; SAUNDERS, M. A. Numerical techniques in mathematical programing. p. 4-6, 1970.

BOCANEGRA, S.; CAMPOS, F. F.; OLIVEIRA, A. R. L. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, v. 36, p. 149-167, 2007.

BURKARD, R. S.; KARISCH, S.; RENDL, F. Qaplib - a quadratic assignment problem library. *European Journal of Operations Research*, v. 55, p. 115-119, 1991.

CAMPOS, F. F. *Analysis of Conjugate Gradients - type methods for solving linear*

equations." PhD thesis, Oxford University Computing Laboratory, Oxford, 1995.

CANTANTE, D. R.; OLIVEIRA, A. R. L. An efficient simplex LU factorization update. *Aceito para publicação na International Journal of Pure and Applied Mathematics*, 2007.

CHVÁTAL, V. *Linear Programming*, W. H. Freeman and Company, New York, 1983.

CZYZYK, J.; MEHROTRA, S.; WAGNER, M.; WRIGHT, S.J. PCx an interior point code for linear programming. *Optimization Methods and Software*, v. 11, n. 2, p. 397-430, 1999.

DANTIZIG, G. B.; ORCHARD-HAYS, W. The product form for the inverse in the simplex method. *Mathematical Tables and Others Aids to Computation*, v. 8, n. 46, p. 64-67, 1954.

DUFF, I. S.; MEURANT, G. A. The effect of ordering on preconditioned conjugate gradients. *BIT*, v. 29, p. 635-657, 1989.

DIKIN, I. I. Iterative solution of problems of linear and quadratic programming. *Soviets Math Doklady*, v. 8, p. 674-675, 1967.

GONDZIO, J. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, v. 6, p. 137-156, 1996.

GONZAGA, C. C. Path following methods for linear programming. *SIAM Review*, v. 34, n. 2, p. 167-224, 1992.

HESTENES, M. R.; STIEFEL, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, v. 49, p. 409-436.

HUARD, P. Resolution of mathematical programming with nonlinear constraints by the method of centres. *In J. Abadie, editor, Non Linear Programming*, p. 209-219, 1967.

KARMAKAR, N. A new polynomial algorithm for linear programming. *Combinatorica*, v.4, p. 373-395, 1984.

LUSTIG, I. J.; MARSTEN, R. E.; SHANNO, D. F. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization*, v. 2, p. 435-449, 1992.

MEHROTRA, S. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, v. 2, n. 4, p. 575-601, 1992.

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

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

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

NG, E.; PEYTON, B. P. Block sparse Cholesky algorithms on advanced uniprocessors computers. *SIAM Journal on Scientific Stat. Computing*, v. 14, p.1034-1056, 1993.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*, Springer series in operations research, 1960.

OLIVEIRA, A. R. L.; SORENSEN, D. C. A new class of preconditioners for large-scale linear systems from interior point methods for linear proagramming. *Linear Algebra Its Applications*, v. 394, p. 1-24, 2005.

OLIVEIRA, A. R. L. *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.

PULINO, P. *Álgebra linear e suas Aplicações - Notas de Aula*, 2008. http://www.ime.unicamp.br/ pulino/ALESA/Texto/.

REID, J. A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Mathematical Programming*, v. 24, p. 55-69, 1982.

RESENDE, M. G. C.; VEIGA, G. An efficient implementation of a network interior point method. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, v. 12, p. 299-348, 1993.

SUHL, U. H. MPOS Mathematical optimization system. *European Journal of Operations Research*, v. 72, p. 312-322, 1994.

SUHL, U. H.; SUHL, L. M. Computing sparse LU factorizations for large-scale linear programming bases. *ORSA Journal on Computing*, v.2, n. 4, p. 325-335, 1990.

SUHL, U. H.; SUHL, L. M. A fast LU upadate for linear programming. *Annals of Operations Research*, v. 47, p. 33-47, 1993.

TAPIA, R. A.; ZHANG, Y. Superlinear and quadratic convergence of primal-dual interior point methods for linear programming revisited. *Journal of Optimization*

Theory and Applications, v. 73, p. 229-242, 1992.

TOMLIN, J. A.; WELCH, J. S. Finding duplicate rows in a linear program. *Operations Research Letters*, v. 5, p. 7-11, 1986.

WANG, W.; O'LEARY, D. P. Adaptative use of iterative methods in predictor-corrector interior point methods for linear programming. *Numerical Algorithms*, v. 25 (1-4), p. 387-406, 2000.