Daniela Renata Cantane

Contribuição da Atualização da Decomposição LU no Método Simplex

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Automação. Aprovação em 17/08/2009.

Orientador:
Prof. Dr. Christiano Lyra Filho - FEEC/UNICAMP

Co-orientador:
Prof. Dr. Aurelio Ribeiro Leite de Oliveira - IMECC/UNICAMP

Campinas, SP
2009

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

Cantane, Daniela Renata

C166c Contribuição atualização da decomposição **LU** no método Simplex / Daniela Renata Cantane. – Campinas, SP:[s.n.],2009.

Orientadores: Aurelio Ribeiro Leite de Oliveira, Christiano Lyra Filho.

Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Programação Linear. 2. Simplex (Matemática). 3. Decomposição **LU**. I. Oliveira, Aurelio Ribeiro Leite. II. Lyra Filho, Christiano. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título

Título em Inglês:	Contribution of the **LU** factorization update in the Simplex method
Palavras-chave em Inglês:	Linear Programming, Simplex Method, **LU** Factorization Update
Área de concentração:	Automação
Titulação:	Doutor em Engenharia Elétrica
Banca Examinadora:	Frederico Ferreira Campos Filho, Marcos Nereu Arenales, Paulo Augusto Valente Ferreira, Akebo Yamakami
Data da defesa:	17/08/2009
Programa de Pós Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidata: Daniela Renata Cantane

Data da Defesa: 17 de agosto de 2009

Título da Tese: "Contribuição da Atualização da Decomposição LU no Método Simplex"

Prof. Dr. Christiano Lyra Filho (Presidente): _____

Prof. Dr. Frederico Ferreira Campos Filho: _____

Prof. Dr. Marcos Nereu Arenales: _____

Prof. Dr. Paulo Augusto Valente Ferreira: _____

Prof. Dr. Akebo Yamakami: _____

À minha família...

Agradecimentos

À Deus, pelo dom da vida e por iluminar meu caminho. À Nossa Senhora, por interceder por mim nos momentos de aflições.
Aos meus orientadores, prof. Aurelio Ribeiro Leite de Oliveira, pela dedicação, compreensão e amizade ao longo de todos os anos do meu mestrado e doutorado; prof. Christiano Lyra Filho, pela confiança depositada em mim e no desenvolvimento deste trabalho.
Ao meu marido Daniel, pelo amor, compreensão, carinho, amizade e por estar sempre ao meu lado durante todos esses anos.
Aos meus pais, Cantane e Cidinha, pelo amor, apoio, amizade, carinho e dedicação que sempre tiveram por mim.
Aos meus irmãos, Daniel e Diego, à toda minha família (avó, tios, tias, primos(as), cunhado e sogra), pelo incentivo e ótimos momentos juntos.
Aos amigos de São José do Rio Preto, pelos momentos de descontração.
À todos amigos e colegas de São Carlos, em especial, à "irmãzinha" Kelly e Carla (e também ao Edson) pela ajuda e dedicação no desenvolvimento da última etapa da tese.
Aos amigos da UNICAMP, Luci Any, Ana Camila, Luciana, Aline, Marina, Maristela, Alana, Aníbal, Adriano e a todos os colegas do Laboratório de Otimização de Redes de Energia, FEEC/UNICAMP.
Aos amigos da república, Clecios, Ana Camila, Juliana, Haroldo e Luis, pelos bons anos de convivência e amizade.
Aos amigos do curso de Inglês, em especial, Cassandra, Graziela, Fernanda e Patrick, pela alegria e risadas aos sábados.
Aos professores Marcos Nereu Arenales (USP/ICMC) e Franklina Maria Bragion de Toledo (USP/ICMC), pela contribuição na minha formação e companhia nos congressos.
Ao professor Francisco A. Magalhães Gomes (IMECC/UNICAMP), pelas sugestões e correções que foram de grande contribuição para a versão final da Tese.
À todos os professores do IBILCE/UNESP, ICMC/USP, IMECC/UNICAMP e FEEC/UNICAMP que contribuíram para minha formação.
À FAPESP, pelo apoio financeiro.
Enfim, agradeço a todos que direta ou indiretamente fizeram parte destes anos de trabalho. Obrigada!

Resumo

A solução eficiente de sistemas lineares é fundamental em problemas de otimização linear e o primeiro método a obter êxito nesta classe de problemas foi o método Simplex. Com o objetivo de desenvolver alternativas eficientes para sua implementação, são apresentadas nesta tese técnicas de atualização da decomposição **LU** da base para aperfeiçoar a solução dos sistemas lineares oriundos do método Simplex, utilizando um reordenamento estático nas colunas da matriz. Uma simulação do método Simplex é implementada, realizando troca de bases obtidas pelo MINOS e verificando sua esparsidade. Somente os elementos afetados pela mudança de base são considerados para obter uma atualização da decomposição **LU** eficaz. As colunas da matriz são reordenadas de acordo com três estratégias: mínimo grau; forma bloco triangular e estratégia de Björck. Assim, obtém-se uma decomposição esparsa para qualquer base sem esforço computacional para obter a ordem das colunas, pois o reordenamento da matriz é estático e as colunas da base obedecem esta ordem. A forma bloco triangular obteve os melhores resultados, para os maiores problemas testados, em relação ao mínimo grau e a estratégia de Björck. Resultados computacionais para problemas da Netlib mostram a robustez e um bom desempenho computacional do método de atualização da decomposição **LU** proposto, pois não são necessárias refatorações periódicas da base como nos métodos de atualização tradicionais. O método proposto obteve uma redução do número de elementos não nulos da base em relação ao MINOS. Esta abordagem foi aplicada em problemas de corte de estoque e a atualização da decomposição **LU** proposta obteve uma redução do tempo computacional na solução destes problemas em relação ao GPLK.

Palavras-chave: otimização linear, matriz esparsa, atualização da decomposição, método Simplex

Abstract

Finding efficient solution of linear systems is fundamental in the linear programming problems and the first method to obtain success for this class of problems was the Simplex method. With the objective to develop efficient alternatives to its implementation, techniques of the simplex basis $\mathbf{LU}$ factorization update are developed in this thesis to improve the solution of the Simplex method linear systems towards a matrix columns static reordering. A simulation of the Simplex method is implemented, carrying through the change of basis obtained from MINOS and verifying its sparsity. Only the factored columns actually modified by the change of the base are carried through to obtain an efficient $\mathbf{LU}$ factorization update. The matrix columns are reordered according to three strategies: minimum degree; block triangular form and the Björck strategy. Thus, sparse factorizations are obtained for any base without computational effort to obtain the order of columns, since the reordering of the matrix is static and base columns follow this ordering. The application of the block triangular form achieved the best results, for larger scale problems tested, in comparison to minimum degree method and the Björck strategy. Computational results for Netlib problems show the robustness of this approach and good computational performance, since there is no need of periodical factorizations as used in traditional updating methods. The proposed method obtained a reduction of the nonzero entries of the basis with respect to MINOS. This approach was applied in the cutting stock problems and the proposed method achieved a reduction of the computational time in the solution of such problems with respect to the GLPK.

Keywords: linear optimization, sparse matrix, factorization update, Simplex method

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho apresenta inovações para aperfeiçoar a solução dos grandes sistemas lineares que compõem o núcleo computacional do método Simplex, o primeiro método eficiente para a solução de problemas de otimização linear, proposto por Dantzig [15], em 1947. Um algoritmo numérico teoricamente correto pode ser implementado de diversas formas diferentes. A eficiência destas diferentes versões pode variar muito, incluindo até variantes incapazes de resolver alguns problemas pequenos. Isto ocorre também com o método Simplex. Portanto, é importante desenvolver alternativas eficientes para sua implementação.

Outros métodos surgiram ao longo do tempo; o mais importante entre eles é o método de pontos interiores para otimização linear (Adler et al. [1], Mehrotra [42]), mas o método de Dantzig sobreviveu como uma das principais ferramentas para resolver problemas de otimização linear. Logo após a descoberta do método Simplex, tornou-se óbvio que o número de operações aritméticas necessárias para encontrar uma solução ótima pelo método é extremamente grande e que ele não teria êxito sem utilizar ferramentas computacionais de alta velocidade. Após a criação do Simplex, os pesquisadores desenvolveram outras variações do método (Orchard-Hays [43]), computacionalmente melhores e, em aproximadamente 15 anos, surgiram diversos programas eficientes.

A otimização linear é a área de otimização mais frequentemente usada. Uma das razões para sua popularidade decorre do fato de representar problemas práticos importantes e existirem algoritmos e implementações poderosas para a resolução de sistemas de grande porte. As duas grandes famílias de métodos para solucionar problemas de otimização linear são as variantes do método Simplex e os métodos de pontos interiores. Eles proporcionaram uma competição intensa nos anos 90 para desenvolver o "melhor método", mas a disputa, do ponto de vista prático, está encerrada agora.

Resulta que ambas as variantes têm propriedades distintas que definem a escolha mais apropriada para abordar diferentes classes de problemas.

Implementações baseadas no método Simplex, ou no método de pontos interiores, são capazes de resolver problemas de grande porte com alta confiabilidade e em um tempo razoável. Modelos são construídos com esta consciência e, frequentemente, procura-se formular problemas reais dentro desta estrutura para garantir que eles possam ser resolvidos eficientemente.

Muitos problemas de otimização reais podem ser formulados como modelos lineares e outros problemas podem ser aproximados por linearização. Além disso, a necessidade de resolver problemas de otimização linear pode surgir durante a solução de outros problemas de otimização, como a otimização linear inteira.

Os métodos de pontos interiores para otimização linear foram propostos na metade dos anos 80 e alcançaram sua maturidade, no sentido teórico e computacional, em uma década e meia. O método Simplex, no entanto, parece continuar susceptível a aperfeiçoamentos. Em 1970, acreditava-se que ele tinha alcançado sua maturidade, mas o surgimento dos métodos de pontos interiores teve um efeito estimulante na pesquisa sobre o método Simplex. Estima-se que a eficiência de implementações do Simplex tenha aumentado duas ordens de grandeza desde o surgimento dos métodos de pontos interiores.

A solução eficiente de sistemas lineares de grande porte é fundamental em problemas de otimização linear. A solução destes sistemas pode ser obtida por meio de métodos genéricos para lidar com as matrizes que formam a base e realizar suas atualizações, ou pela exploração da estrutura da classe de problemas a serem resolvidos. Um exemplo de abordagem genérica é a decomposição $\mathbf{LU}$; exemplos de exploração de estrutura em particular são os métodos específicos para abordar os problemas de fluxos em redes (Kennington e Helgason [38]).

O foco principal deste trabalho consiste no desenvolvimento de um reordenamento estático das colunas da base do método Simplex, em conjunto com uma atualização eficiente das colunas que entram e saem da base, para problemas de otimização linear em geral. A decomposição $\mathbf{LU}$ é realizada somente nas colunas realmente afetadas pela entrada e/ou saída de colunas na base.

Na atualização da decomposição $\mathbf{LU}$, as operações causadas pela coluna que sai da base são

desfeitas, ou seja, as operações são efetuadas na ordem inversa da decomposição. As operações para decomposição da matriz básica, obtida com a coluna que entra na base são, então, realizadas. Finalmente, é necessário efetuar as operações sobre as colunas localizadas após a coluna que entra na base, sempre considerando a esparsidade.

Este trabalho concentra-se em aplicar esta atualização da decomposição $\mathbf{LU}$ em problemas gerais de otimização linear. Como uma abordagem específica, esta atualização foi aplicada em problemas de corte de estoque (Poldi [47]).

Nos trabalhos de Bressan e Oliveira [7, 8], o reordenamento das colunas é feito de modo que a matriz resultante adquire formato bloco diagonal, facilitando a decomposição das bases, reduzindo assim o preenchimento da matriz no problema de corte e empacotamento. Por meio do reordenamento das colunas, é possível obter uma decomposição esparsa para qualquer base sem nenhum esforço computacional para obter a ordem das colunas, pois o reordenamento da matriz de restrições é estático, ou seja, a matriz é ordenada inicialmente por um vetor auxiliar e as colunas da base obedecem este ordenamento.

Para o problema de lotes e cortes, os resultados obtidos em Bressan e Oliveira [7] mostram que esta abordagem além de computacionalmente barata, também é robusta introduzindo erros de arredondamento insignificantes em situações de pior caso, após milhares de iterações.

No Capítulo 2, encontram-se uma breve revisão do método Simplex e da decomposição $\mathbf{LU}$ da base. Apresenta-se, no Capítulo 3, os métodos de atualização da decomposição $\mathbf{LU}$. A definição de base crash e os métodos de triangularização da matriz são apresentados no Capítulo 4. Apresenta-se, no Capítulo 5, o modelo do problema de corte de estoque utilizado na aplicação do método proposto. Desenvolve-se no Capítulo 6 um método de atualização de colunas, principal contribuição do trabalho. Os resultados numéricos obtidos são apresentados no Capítulo 7. As conclusões e discussões dos próximos passos na pesquisa encontram-se no Capítulo 8, seguidas pelas referências bibliográficas e uma lista do conjunto dos artigos já divulgados, como parte da pesquisa desenvolvida na tese.

No Apêndice A encontra-se o trabalho que fez parte do estágio no exterior realizado em Málaga - Espanha, com orientação dos professores Pablo Guerrero-Garcia e Àngel Santos-Palomo. O trabalho consiste na aplicação do ordenamento de colunas e atualização da decomposição $\mathbf{LU}$ desenvolvidas nesta tese no cálculo da solução de problemas de mínimos quadrados usando gradiente reduzido. No

Apêndice B, encontram-se os arquivos de entrada e saída do MINOS necessários tanto para solucionar os problemas da Netlib quanto para fornecer as informações utilizadas nos testes computacionais do Capítulo 7.

Capítulo 2

Método Simplex e Decomposição LU

2.1 Introdução

O método Simplex proposto por Dantzig tem recebido evoluções, embora seus princípios permaneçam os mesmos. Do ponto de vista teórico, o método Simplex não é necessariamente um algoritmo promissor, pois seu desempenho no pior caso é exponencial para toda regra de entrada na base conhecida. Isto significa que o número de iterações necessárias para resolver um problema é limitado por uma função exponencial de m (quantidade de linhas da matriz) e n (quantidade de colunas da matriz). Na prática, no entanto, o método Simplex mostra um desempenho médio que é uma função linear de m e é altamente eficiente para resolver problemas da vida real.

Entre 1951 e a metade de 1970, o principal alvo dos pesquisadores em alternativas para solução de problemas de otimização linear era o aumento do desempenho computacional do método Simplex. No fim deste período, acreditava-se que o método tinha alcançado sua maturidade. No entanto, na metade de 1980, com a introdução dos métodos de pontos interiores para resolver problemas de otimização linear, ressurgiu o interesse no Simplex. Desde então, ele progrediu muito.

Neste capítulo é apresentada a decomposição **LU** da base utilizando o pivoteamento parcial e uma revisão do método Simplex utilizado como uma aplicação da atualização da decomposição **LU** proposta.

2.2 Método Simplex

Considere o problema primal de otimização linear na forma padrão (Bazaraa et al. [3], Vanderbei [60])

$$\begin{aligned} \min f(\mathbf{x}) &= \mathbf{c^t x} \\ \text{s.a } \mathbf{Ax} &= \mathbf{b}, \\ \mathbf{x} &\geq \mathbf{0}, \end{aligned} \tag{2.1}$$

onde $\mathbf{A} \in I\!R^{m \times n}$ e, sem perda de generalidade, assuma que $posto(\mathbf{A}) = m$.

A solução geral do sistema em $\mathbf{Ax} = \mathbf{b}$ pode ser descrita considerando uma partição nas colunas de $\mathbf{A}$

$$\mathbf{A} = (\mathbf{B}, \mathbf{N}),$$

onde $\mathbf{B} \in I\!R^{m \times m}$ é formada por m colunas linearmente independentes da matriz $\mathbf{A}$, ou seja, $\mathbf{B}$ é não singular. $\mathbf{B}$ é denominada matriz das varíaveis básicas e $\mathbf{N}$ a matriz das variáveis não básicas. Uma partição equivalente é feita no vetor das variáveis

$$\mathbf{x} = (\mathbf{x_B}, \mathbf{x_N}),$$

onde $\mathbf{x_B}$ é chamado "vetor de variáveis básicas" e $\mathbf{x_N}$ "vetor de variáveis não básicas". Considere também a partição nos coeficientes da função objetivo $\mathbf{c}$

$$\mathbf{c}^t = (\mathbf{c_B}, \mathbf{c_N})^t.$$

Temos que,

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow \mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b} \Leftrightarrow \mathbf{x_B} = \mathbf{B^{-1}}(\mathbf{b} - \mathbf{Nx_N}). \tag{2.2}$$

Esta equação mostra que as variáveis não básicas $\mathbf{x_N}$ determinam os valores das variáveis básicas $\mathbf{x_B}$, de modo que o sistema seja satisfeito.

Definição 2.2.1 *A solução particular* $\mathbf{x}$ *obtida por* $\mathbf{x_B}^0 = \mathbf{B}^{-1}\mathbf{b},\ \mathbf{x_N}^0 = \mathbf{0}$ *é chamada "solução básica". Se* $\mathbf{x_B}^0 = \mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$*, então a solução básica é primal factível. Temos ainda que, se* $\mathbf{x_B}^0 > \mathbf{0}$ *a solução é não degenerada.*

Substituindo (2.2) na função objetivo, temos

$$f(\mathbf{x}) = \mathbf{c_B}\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c_N}^t - \mathbf{c_B}^t\mathbf{B}^{-1}\mathbf{N})\mathbf{x_N}.$$

Uma componente j de $\mathbf{c_N}^t - \mathbf{c_B}^t\mathbf{B}^{-1}\mathbf{N}$ é chamada "custo reduzido de $\mathbf{x_j}$", é denotada por

$$\mathbf{d_j} = \mathbf{c_j} - \mathbf{c_B}^t\mathbf{B}^{-1}\mathbf{a_j},$$

onde $\mathbf{a_j}$ representa a coluna j da matriz de restrições A. Uma condição suficiente para que a base factível B seja ótima, ou seja, a "condição de otimalidade" é

$$\mathbf{d_j} \geq \mathbf{0} \text{ para todo } j \in I\!N,$$

onde $I\!N$ é o conjunto dos índices das colunas da matriz $\mathbf{N}$ (colunas não-básicas da matriz $\mathbf{A}$).

Para o problema primal de otimização linear (2.1), temos que o problema dual é dado por

$$\begin{aligned} &\max f(\mathbf{y}) = \mathbf{b}^t\mathbf{y} \\ &\text{s.a } \mathbf{A}^t\mathbf{y} \leq \mathbf{c}. \end{aligned} \tag{2.3}$$

Neste caso, as variáveis duais $\mathbf{y}$ são irrestritas. Se o problema primal (2.1) possuir restrição matricial de desigualdade, as variáveis duais serão restritas, $\mathbf{y} \geq \mathbf{0}$ se $\mathbf{Ax} \geq \mathbf{b}$ e $\mathbf{y} \leq \mathbf{0}$ se $\mathbf{Ax} \leq \mathbf{b}$.

Seja $\mathbf{y} \in I\!R^m$, dado por

$$\mathbf{y} = \mathbf{B}^{-t}\mathbf{c_B},$$

o vetor das variáveis duais ou "vetor multiplicador simplex". Se

$$\mathbf{c_j} - \mathbf{y}^t\mathbf{a_j} \geq \mathbf{0},$$

para $j = 1, \ldots, n$. Então $\mathbf{y}$ é uma "solução básica dual factível", e dizemos que a partição é dual factível.

Assim, a função objetivo $f(\mathbf{x})$ (em termos das variáveis não-básicas) é dada por

$$f(\mathbf{x}) = f(\mathbf{x}^0) + \sum_{j \in I\!N} (\mathbf{c_j} - \mathbf{y}^t\mathbf{a_j})\mathbf{x_j}.$$

Isto mostra como a função objetivo se altera quando alterações são feitas nas variáveis não-básicas.

Teorema 2.2.1 *Se uma partição básica for primal e dual factível, então as soluções básicas associadas resolvem os problemas primal e dual, respectivamente, e dizemos que a partição básica é ótima.*

Prova: A demonstração é trivial. Pode ser encontrada, por exemplo, em Vanderbei [60].

Teorema 2.2.2 *Se o problema primal tiver uma solução ótima, então existe uma partição básica ótima.*

Prova: Vanderbei [60].

Definição 2.2.2 *Chamamos de "estratégia simplex" a seguinte perturbação da solução básica: escolha $k \in I\!N$, onde $I\!N$ é o conjunto de índices de variáveis não básicas, tal que $\mathbf{c_k} - \mathbf{y}^t\mathbf{a_k} < \mathbf{0}$, pois queremos minimizar $f(\mathbf{x})$ e $\mathbf{x_j} \geq \mathbf{0}$ para todo $j \in I\!N$; faça $\mathbf{x_k} = \epsilon \geq \mathbf{0}$, $\mathbf{x_j} = \mathbf{0}, \forall j \in I\!N - \{k\}$.*

A estratégia simplex produz uma nova solução dada por

$$\begin{cases} \mathbf{x_B} = \mathbf{x_B}^0 + \epsilon\mathbf{d_B} \\ \mathbf{x_N} = \epsilon\mathbf{e_k} \end{cases}$$

e o valor da função objetivo dado por

$$f(\mathbf{x}) = f(\mathbf{x}^0) + (\mathbf{c_k} - \mathbf{y}^t\mathbf{a_k})\epsilon,$$

onde $\mathbf{d_B} = -\mathbf{B}^{-1}\mathbf{a_k}$ e $\mathbf{e_k} = (0, \ldots, 1, \ldots, 0)^t \in I\!R^{n-m}$ com 1 na k-ésima componente.

Note que a direção $\mathbf{d} \in I\!R^n$, dada por $\mathbf{d} = (\mathbf{d_B}, \mathbf{d_N})^t = (\mathbf{d_B}, \mathbf{e_k})^t$, define uma perturbação da solução básica e é chamada "direção simplex". Note ainda que o produto escalar entre $\mathbf{d}$ e o gradiente da função objetivo é $\mathbf{c}^t\mathbf{d} = \mathbf{c_k} - \mathbf{y}^t\mathbf{a_k} < 0$. Portanto $\mathbf{d}$ é uma direção de descida.

Da estratégia Simplex, podemos determinar o maior valor de ϵ para que a nova base seja factível, impondo $\mathbf{x_B} \geq \mathbf{0}$

$$\epsilon^0 = \min\left\{ -\frac{\mathbf{x_{B_i}}^0}{\mathbf{d_{B_i}}} \mid \mathbf{d_{B_i}} < 0, i = 1, \ldots, m \right\},$$

onde $\mathbf{x_{B_i}}^0$ é a i-ésima componente de $\mathbf{x_B}^0$, que sai da base.

Para uma melhor visualização sobre a organização dos conceitos acima, descrevemos a seguir todos os passos necessários para a realização do método Primal Simplex.

Método Primal Simplex

1- Encontre uma partição básica primal factível $\mathbf{A} = (\mathbf{B}, \mathbf{N})$.

2- Determine a solução básica primal factível $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$.

3- Calcule os multiplicadores Simplex $\mathbf{y} = \mathbf{B}^{-t}\mathbf{c_B}$.

4- Teste de otimalidade: Encontre $\mathbf{x_k}$ com custo relativo $\mathbf{c_k} - \mathbf{y}^t\mathbf{a_k}$. Se $\mathbf{c_k} - \mathbf{y}^t\mathbf{a_k} \geq 0,\ \forall\ k = 1, \ldots, n-m$, então a solução é ótima. Senão,

5- Determine a direção Simplex $\mathbf{d_B} = -\mathbf{B}^{-1}\mathbf{a_k}$, das variáveis básicas.

6- Determine o passo: $\epsilon^0 = \min\left\{-\frac{\mathbf{x_{B_i}}^0}{\mathbf{d_{B_i}}} \mid \mathbf{d_{B_i}} < 0, i = 1, \ldots, m\right\}$.

7- Se $\mathbf{d_B} \geq 0$, o problema não tem solução ótima finita. Senão,

8- Atualize a partição básica: $\mathbf{a_{B_i}} \leftrightarrow \mathbf{a_k}$ e retorne ao passo 2.

Método Simplex Duas Fases

O problema de otimização linear (2.1) pode ser resolvido em duas fases. Na Fase I, variáveis artificiais são introduzidas na matriz de restrições (somente nas equações que não possuem variáveis de introduzidas na matriz de restrições (somente nas equações que não possuem variáveis de folga) e procura-se uma solução básica factível para o problema. Se a busca pela solução básica factível não é bem sucedida podemos concluir que o problema é infactível. Se este é o caso, a solução ótima do problema da Fase I indica um (dentre os muitos) conjunto de restrições infactíveis.

A Fase II começa quando a Fase I termina bem-sucedida com a função objetivo do problema auxiliar da Fase I igual a zero. O algoritmo da Fase II usa a solução básica factível resultante da Fase I. Durante a Fase II, as variáveis artificiais e a função objetivo da Fase I são omitidas. O algoritmo pode terminar com uma solução ótima ou com uma indicação que a solução é ilimitada.

Detalhes sobre o método Primal Simplex e o método Simplex Duas Fases podem ser vistos em Bazaraa et al. [3], Luenberger [39], Maros [41], Vanderbei [60].

2.3 Forma Produto da Inversa

A forma produto da inversa consiste em armazenar a inversa de uma certa matriz básica $\mathbf{B}$ como o produto de matrizes elementares, ou seja, a matriz inversa $\mathbf{B}^{-1}$ é expressa como o produto de matrizes elementares $\mathbf{E}_i$, que representam as operações linhas realizadas na coluna i da matriz $\mathbf{B}$ para torná-la uma matriz identidade. Por exemplo, suponha que a matriz $\mathbf{B}$ seja de ordem m. A forma produto da inversa desta matriz pode ser representada da seguinte forma

$$\mathbf{B}^{-1} = \mathbf{E}_m \mathbf{E}_{m-1} \dots \mathbf{E}_1.$$

Definição 2.3.1 *Uma matriz é dita elementar se ela difere da matriz identidade por apenas uma coluna, a qual representa as operações linhas realizadas em uma certa matriz.*

Considere a matriz elementar

$$\mathbf{E}_i = \begin{pmatrix} 1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & \eta_i & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{pmatrix}.$$

O efeito de multiplicar a matriz elementar $\mathbf{E}_i$ à esquerda de uma matriz $\mathbf{A}$ qualquer é

$$\mathbf{E}_i A = \begin{pmatrix} 1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & \eta_i & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{pmatrix} \begin{pmatrix} a^1 \\ \vdots \\ a^i \\ \vdots \\ a^m \end{pmatrix} = \begin{pmatrix} a^1 + \eta_1 a^i \\ \vdots \\ \eta_i a^i \\ \vdots \\ a^m + \eta_m a^i \end{pmatrix}.$$

Os valores η_j, $j = 1, \dots, m$, são adequadamente escolhidos de modo a transformar a coluna i da matriz $\mathbf{A}$ na i-ésima coluna da matriz identidade. Considere as matrizes elementares da seguinte forma

$$\mathbf{E}_i = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \eta_i & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{pmatrix}.$$

As matrizes elementares com a estrutura representada acima podem ser usadas para atualizar a inversa da matriz $\mathbf{B}$ de uma iteração para a outra. Seja $\mathbf{B}^{(t)} = (b_1, \ldots, b_l, \ldots, b_m)$ a matriz básica da iteração t, onde cada b_i é uma coluna de $\mathbf{A}$ tal que $b_i = a_{p_i}$ e $(\mathbf{B}^{(t)})^{-1}$ a inversa da matriz $\mathbf{B}$ representada como o produto de matrizes elementares.

A matriz básica $\mathbf{B}^{(t+1)}$ difere da matriz $\mathbf{B}^{(t)}$ em apenas uma coluna, pois quando a $\mathbf{B_l}$-ésima variável básica deixa a base e a $\mathbf{N_k}$-ésima variável não básica entra para a base somente uma das colunas de $\mathbf{B}^{(t)}$ se modifica, enquanto as demais permanecem como antes. Assim, é de se esperar que a inversa de $\mathbf{B}^{(t+1)}$ seja obtida de forma simples a partir da inversa de $\mathbf{B}^{(t)}$. Multiplicando $(\mathbf{B}^{(t)})^{-1}\mathbf{B}^{(t+1)}$, temos

$$\begin{aligned} (\mathbf{B}^{(t)})^{-1}\mathbf{B}^{(t+1)} &= \left((\mathbf{B}^{(t)})^{-1}b_1 \ldots (\mathbf{B}^{(t)})^{-1}a_k \ldots (\mathbf{B}^{(t)})^{-1}b_m\right) \\ &= \begin{pmatrix} 1 & 0 & & p_1 & & 0 \\ 0 & 1 & & p_2 & & 0 \\ & & \ddots & \vdots & & \\ \vdots & \vdots & \ldots & p_l & \ldots & \vdots \\ & & & \vdots & \ddots & \\ 0 & 0 & & p_m & & 1 \end{pmatrix} = \bar{I}. \end{aligned}$$

ou seja, $(\mathbf{B}^{(t)})^{-1}\mathbf{B}^{(t+1)} = \bar{\mathbf{I}}$. Observe que $\bar{\mathbf{I}}$ difere da matriz identidade em apenas uma coluna, no caso, a l-ésima coluna, isto porque, $(\mathbf{B}^{(t)})^{-1}a_i = e_i,\ i = 1, \ldots, l-1, l+1, \ldots, m$, onde e_i é a i-ésima coluna da matriz identidade. Observe também que p_l é (a menos do sinal) a l-ésima coordenada básica da direção Simplex, que define o tamanho do passo numa iteração Simplex e, portanto, $p_l \neq 0$.

A inversa de $\bar{\mathbf{I}}$ está bem definida e é facilmente obtida por

$$(\bar{\mathbf{I}})^{-1} = \begin{pmatrix} 1 & 0 & & -\frac{p_1}{p_l} & & 0 \\ 0 & 1 & & -\frac{p_2}{p_l} & & 0 \\ & & \ddots & \vdots & & \\ \vdots & \vdots & \cdots & \frac{1}{p_l} & \cdots & \vdots \\ & & & \vdots & \ddots & \\ 0 & 0 & & -\frac{p_m}{p_l} & & 1 \end{pmatrix},$$

ou seja, $(\bar{\mathbf{I}})^{-1}$ é uma matriz elementar que transforma a l-ésima coluna de $\bar{\mathbf{I}}$ na l-ésima coluna da matriz identidade (e_l). Assim,

$$(\bar{\mathbf{I}})^{-1}(\mathbf{B}^{(t)})^{-1}\mathbf{B}^{(t+1)} = (\bar{\mathbf{I}})^{-1}\bar{\mathbf{I}} = \mathbf{I}.$$

Portanto,

$$(\mathbf{B}^{(t+1)})^{-1} = (\bar{\mathbf{I}})^{-1}(\mathbf{B}^{(t)})^{-1}.$$

Isso significa que para determinarmos a inversa de $\mathbf{B}^{(t+1)}$ basta multiplicarmos a esquerda da matriz $(\mathbf{B}^{(t)})^{-1}$, que já é conhecida, por $(\bar{\mathbf{I}})^{-1}$. Mas, multiplicar $(\bar{\mathbf{I}})^{-1}$ a esquerda de uma matriz é equivalente a estender para $(\mathbf{B}^{(t)})^{-1}$ as operações elementares necessárias para transformar o vetor $(p_1, p_2, \ldots, p_m)^t$ em e_l. Denotando $\mathbf{E}^{t+1} = (\bar{\mathbf{I}})^{-1}$ e, supondo que a matriz básica da iteração inicial $\mathbf{B}^{(0)} = \mathbf{I}$, e $\mathbf{E}^{(1)}, \mathbf{E}^{(2)}, \ldots, \mathbf{E}^{(t)}$ são as matrizes elementares obtidas ao atualizar as inversas das bases nas iterações $1, 2, \ldots, t$, a matriz $(\mathbf{B}^{(t+1)})^{-1}$ pode ser escrita

$$(\mathbf{B}^{(t+1)})^{-1} = \mathbf{E}^{(t+1)}\mathbf{E}^{(t)} \ldots \mathbf{E}^{(1)}.$$

Assim, na iteração t,

- Solução básica primal: $\mathbf{x_B} = (\mathbf{B}^{(t+1)})^{-1}b = \mathbf{E}^{(t+1)}(\mathbf{E}^{(t)} \ldots (\mathbf{E}^{(1)}b))$.

- Vetor multiplicador simplex: $\mathbf{y}^t = \mathbf{c_B}^t(\mathbf{B}^{(t+1)})^{-1} = ((\mathbf{c_B}^t\mathbf{E}^{t+1}) \ldots \mathbf{E}^{(2)})\mathbf{E}^{(1)}$.

- Coordenadas básicas da direção simplex: $\mathbf{d_B} = -(\mathbf{B}^{(t+1)})^{-1}a_k = \mathbf{E}^{(t+1)}(\mathbf{E}^{(t)} \ldots (\mathbf{E}^{(1)}a_k))$.

Observação 2.3.1 *Para armazenar uma matriz elementar* $\mathbf{E}^{(t)}$*, é necessário armazenar apenas* $m+1$ *números: além da posição* t*,* $(p_1/p_l, p_2/p_l, \ldots, p_l - 1/p_l, 1/p_l, p_{l+1}/p_l, \ldots, p_m/p_l)$*, onde o* p_i *é o elemento da linha* i *na coluna* l *da matriz* $\mathbf{E}$*.*

A solução básica primal, o vetor multiplicador simplex e as coordenadas básicas da direção simplex são facilmente obtidos, pois eles são uma sequência de produtos de matrizes elementares por vetores.

2.4 Decomposição LU da Base

Em problemas grandes, normalmente é inviável inverter-se a matriz básica $\mathbf{B}$. Por isso, usualmente realiza-se a decomposição $\mathbf{LU}$ da base, onde $\mathbf{L}$ é uma matriz triangular inferior e $\mathbf{U}$ triangular superior, procurando-se obter um algoritmo mais eficiente em termos de rapidez de solução, uso de memória e estabilidade numérica.

Para decompor a matriz, podemos usar a *decomposição de Crout* (Duff et al. [20]), que define $\mathbf{U}$ triangular superior unitária (possui apenas elementos uns na diagonal principal). Assim, obtemos as relações

$$l_{ij} = b_{ij} - \sum_{p=1}^{j-1} l_{ip} u_{pj}, \quad i \geq j$$

$$u_{ij} = (b_{ij} - \sum_{p=1}^{i-1} l_{ip} u_{pj})/l_{ii}, \quad i < j.$$

A sequência de cálculos usualmente associada com a decomposição de Crout consiste em trabalhar com a primeira coluna de $\mathbf{L}$, a primeira linha de $\mathbf{U}$, em seguida, a segunda coluna de $\mathbf{L}$, a segunda linha de $\mathbf{U}$ e assim por diante. Isto pode ser comparado à eliminação Gaussiana (Duff et al. [20]), através da equação

$$b_{ij}^{k+1} = b_{ij}^{k} - b_{ik}^{k}(b_{kj}^{k}/b_{kk}^{k})$$

e fazendo as associações

$$l_{ik} = b_{ik}^{k} \qquad u_{kj} = b_{kj}^{k}/b_{kk}^{k},$$

que correspondem à divisão da linha do pivô (e não a coluna) pelo pivô b_{kk}^{k}.

De forma mais geral, podemos desenvolver a decomposição

$$\mathbf{B} = \mathbf{LDU},$$

onde $\mathbf{D}$ é uma matriz diagonal e ambas $\mathbf{L}$ e $\mathbf{U}$ possuem elementos uns em suas diagonais principais. Por exemplo, a decomposição de Crout corresponde à associação $(\mathbf{LD})\mathbf{U}$ e a decomposição de Dolittle (Duff et al. [20]) está associada à $\mathbf{L}(\mathbf{DU})$, sendo portanto, a matriz $\mathbf{L}$ triangular inferior unitária. Esta decomposição é única, enquanto que a decomposição $\mathbf{LU}$ não é.

Neste trabalho, a decomposição $\mathbf{LU}$ é armazenada na mesma área ocupada pela matriz $\mathbf{B}$. Mais precisamente, $\mathbf{U}$ é armazenada na parte triangular superior de $\mathbf{B}$ (a diagonal de $\mathbf{U}$ não é armazenada, pois seus elementos são unitários) e $\mathbf{L}$ ocupa a parte triangular inferior de $\mathbf{B}$ (incluindo a diagonal).

2.4.1 Decomposição com Pivoteamento Parcial: Versão GAXPY

Existem diferentes estratégias que tentam minimizar o aparecimento de novos elementos não-nulos nas operações de pivoteamento. O surgimento desses novos elementos acarreta em necessidade de mais memória para armazená-los e mais tempo computacional nas operações aritméticas em que são envolvidos. Este fato ocorre com muita freqüência nas operações de pivoteamento quando o elemento pivô é selecionado sem levar em conta que podem ser gerados novos elementos (não-nulos) ao eliminar os elementos que estão nas linhas abaixo dele.

As heurísticas de pivoteamento são utilizadas com a decomposição $\mathbf{LU}$ com o objetivo de preservar a esparsidade existente nas matrizes básicas e para evitar problemas de estabilidade numérica no método Simplex (Duff et al. [20]). Dentre as estratégias de pivoteamento, temos o *pivoteamento completo*, onde linhas e colunas são permutadas, e o *pivoteamento parcial*, onde apenas as linhas são permutadas. Neste trabalho, utilizaremos o pivoteamento parcial, por oferecer para os problemas estudados a melhor combinação entre estabilidade numérica e eficiência computacional.

Seja $\mathbf{B}$ a base que pretendemos decompor. Nesta decomposição, é preciso calcular uma matriz triangular inferior unitária $\mathbf{L}$, uma triangular superior $\mathbf{U}$ e uma permutação $\mathbf{P}$ tal que $\mathbf{PB} = \mathbf{LU}$. No armazenamento, b_{ij} terá l_{ij} em sua parte inferior se $i > j$ e terá $u_{i,j}$ caso contrário. $\mathbf{P} = \mathbf{E}_{n-1} \ldots \mathbf{E}_1$ pode ser representada pelo vetor inteiro $piv(1 : n-1)$. Em particular, $\mathbf{E}_j$ é uma permutação envolvendo a linha j e $piv(j)$, com $j = 1 : n-1$.

Esta decomposição é um procedimento computacional de três laços, que podem ser dispostos de várias formas. Vale notar que a sequência das operações desta decomposição é a mesma que é realizada pela eliminação Gaussiana; por isso, podemos nos referir a ela como tal. Na versão GAXPY

os ciclos ocorrem na ordem jki, como no pseudo-código apresentado a seguir. O nome vem de uma generalização de SAXPY (Scalar $a \times \mathbf{x}$ Plus y), onde os ciclos ocorrem na ordem kji, isto é, a operação básica do algoritmo, que consiste em multiplicar um escalar a por um vetor X, somando um outro vetor y e então armazenando o resultado. Na versão GAXPY, o escalar a é substituído pela matriz. Assim, sua operação básica é um produto matriz-vetor. Adotamos esta forma de decomposição pelo fato de cada coluna ser atualizada apenas no momento de seu *próprio pivoteamento*, o que apresenta uma grande utilidade para o método Simplex, já que, a cada iteração deste, as matrizes básicas obtidas são diferentes da anterior em apenas uma coluna.

Pseudo-Código de Algoritmo GAXPY

for j=1:n
  for k=1:j-1
    for i=k+1:n
      $b_{ij} = b_{ij} - b_{ik}$*b_{kj};
    end
  end
  for i=j+1:n
    b_{ij}=b_{ij}/b_{jj}
  end
end

A coluna j da base original $\mathbf{B}$ é mantida até o passo j. No ponto em que a parte superior de $\mathbf{B}(:, j)$ for sobrescrita por $\mathbf{M}_{j-1} \ldots \mathbf{M}_1\mathbf{B}(:, j)$, onde $\mathbf{M}$ são as matrizes dos multiplicadores, a j-ésima transformação Gaussiana é calculada (Golub e Loan [30]).

As permutações podem ser usadas para garantir que nenhum multiplicador é maior que um em valor absoluto, obtendo decomposições mais estáveis (Golub e Loan [30]). Para que os multiplicadores sejam os menores possíveis na primeira transformação de Gauss usando troca de linhas, precisamos que b_{11} seja o maior elemento em módulo na primeira coluna. Se necessário, realizamos a primeira permutação, e multiplicamos a matriz resultante pela matriz dos multiplicadores $\mathbf{M}_1$, e assim por diante. Esta estratégia de trocas de linhas é chamada de *pivoteamento parcial*. Como

consequência, nenhum multiplicador é maior que um em valor absoluto, isto porque

$$|(\mathbf{E}_k\mathbf{M}_{k-1}\ldots\mathbf{M}_1\mathbf{E}_1\mathbf{B})_{kk}| = max_{k\leq i\leq n}|(\mathbf{E}_k\mathbf{M}_{k-1}\ldots\mathbf{M}_1\mathbf{E}_1\mathbf{B})_{ik}|, \text{ para } k = 1 : n-1.$$

Portanto, observando a equação acima, o pivoteamento parcial protege contra erros de arredondamentos causados por multiplicadores maiores.

Limiar de pivoteamento[1] é uma estratégia de pivoteamento utilizada às vezes em substituição ao pivoteamento parcial. Nesta estratégia, qualquer entrada não eliminada $\mathbf{B}(p,k)$ na coluna corrente k pode ser selecionada como pivô, desde que satisfaça

$$\mid \mathbf{B}(p,k) \mid \geq u.max \mid (\mathbf{B}(k:n,k)) \mid,$$

onde u é um parâmetro entre 0 e 1. Por exemplo, $u = 1$ resulta no pivoteamento parcial. Se u é menor que 1, podemos admitir vários candidatos a pivô; cada um deles pode introduzir um crescimento limitado (Duff et al. [20]). Esta estratégia de pivoteamento oferece pequeno benefício para o caso onde a matriz é densa, mas é muito útil no caso de matrizes esparsas.

A idéia é relaxar a regra do pivoteamento parcial que escolhe o maior pivô em cada iteração. Deste modo, escolhemos a linha do pivô que será a melhor em termos de minimizar o *fill-in* (preenchimento), mas que ainda contém um pivô suficientemente grande, ou seja, o pivô que selecionamos é grande por alguma medida ou *limiar*, por exemplo, ele é pelo menos 10% tão grande quanto o maior elemento absoluto nesta coluna.

No próximo Capítulo, veremos métodos para atualizar a decomposição $\mathbf{LU}$ da matriz básica, buscando manter a esparsidade.

[1] do inglês *Threshold pivoting*

Capítulo 3

Métodos de Atualização da Decomposição LU

3.1 Introdução

A idéia de que as rotinas para inversão de matrizes de otimização linear poderiam ser melhoradas em eficiência utilizando a forma eliminação da inversa ao invés da forma produto da inversa foi primeiramente defendida por Markowitz e, depois, por Dantzig para matrizes especialmente estruturadas na forma "escada". Mais tarde, em experimentos com matrizes gerais de otimização linear, Dantzig et al. [16], segundo Forrest-Tomlin [24], demonstraram a superioridade da forma eliminação da inversa, a qual utiliza a decomposição **LU**, sobre a forma produto da inversa em termos de velocidade, precisão e esparsidade.

Dadas as vantagens da forma eliminação da inversa, o próximo passo era tentar usar os fatores triangulares resultantes na atualização da inversa da base nas iterações seguintes. De acordo com Forrest-Tomlin [24], isso foi feito por Dantzig [15] em seu algoritmo com matrizes "escada", de forma a preservar a estrutura especial dos fatores triangulares, e também por Bartels-Golub [2], cujo esquema de atualização tem muitas propriedades numéricas eficientes.

Foram propostos vários métodos para atualizar os fatores triangulares de uma matriz modificada e após uma análise feita por Forrest-Tomlin [24] sobre suas conveniências com relação à implementação em um problema de otimização linear de grande porte, o método devido a Brayton et al. [6], pareceu ser o mais conveniente para implementações práticas. Experimentos computacionais com

este método em alguns problemas de otimização linear de tamanho médio foram muito promissores. A redução no crescimento de elementos não nulos na inversa foi de 50%, comparada com a forma produto padrão, e a redução global no tamanho do espaço de armazenamento que leva à inversa da base foi de 30%. O método que veremos na Seção 3.3, que produz uma matriz inversa esparsa, é uma extensão do método devido a Brayton et al. [6] e foi implementado por Forrest-Tomlin [24].

Com o objetivo de manter a esparsidade original dos dados, algumas formas de decomposição **LU** da matriz básica foram implementadas por Silva [56]. Também foi apresentada a decomposição **LU** proposta por Bartels-Golub, fomas de resolver sistemas lineares quando a atualização da decomposição é feita e algumas considerações sobre implementações. Neste trabalho, concluiu-se que o número de atualizações da decomposição **LU** realizadas nas iterações seguintes influenciam consideravelmente no tempo de resolução dos problemas, nos preenchimentos ocorridos e até mesmo no número de iterações do método Simplex.

Em Maros [41], a atualização da decomposição **LU** e sua aplicação na atualização da base no método Simplex proposta por Markowitz [40] é descrita em detalhe. A seguir, são apresentados diferentes métodos para atualizar a decomposição **LU** da matriz básica, buscando manter a esparsidade da matriz. O primeiro método apresentado foi proposto por Bartels-Golub [2] e Reid [51] propôs suas variantes. O segundo foi proposto por Forrest-Tomlin [24]. Logo após, são apresentadas idéias de uma implementação eficiente proposta por Suhl e Suhl [58], que é uma combinação de uma decomposição simbólica e numérica e apresenta um bom compromisso entre esparsidade e estabilidade numérica. Outras técnicas de atualização são descritas em Duff et al. [20]. Por último, encontra-se a atualização proposta nesta tese.

3.2 Atualização de Bartels-Golub

A técnica de atualização da decomposição **LU** da matriz básica com pivoteamento parcial foi proposta por Bartels-Golub [2]. Duas variantes deste algoritmo propostas por Reid [51] procuram manter tanto a esparsidade existente na decomposição quanto a estabilidade numérica. A segunda variante é um aperfeiçoamento da primeira.

Esta técnica proposta por Bartels-Golub [2] foi desenvolvida para atualizar a decomposição **LU** da matriz básica $\mathbf{B}$ em cada iteração do método Simplex. Utiliza o pivoteamento parcial, que consiste em utilizar o elemento pivô como o maior elemento em módulo da coluna (Duff et al. [20]), evitando

assim que os erros de arredondamento se propaguem de forma descontrolada.

Suponha que desejamos resolver um sistema do tipo $\mathbf{Bv} = \mathbf{q}$ e que a matriz $\mathbf{B}$ tenha sido decomposta no produto de uma matriz triangular superior unitária $\mathbf{U}$ e uma matriz triangular inferior $\mathbf{L}$.

Desta forma, temos $\mathbf{PB} = \mathbf{LU}$ para alguma matriz de permutação $\mathbf{P}$. Logo, a resolução do sistema inicial $\mathbf{Bv} = \mathbf{q}$ pode ser obtida resolvendo dos dois sistemas triangulares

$$\mathbf{Lt} = \mathbf{Pq} \qquad \text{e} \qquad \mathbf{Uv} = \mathbf{t}.$$

Seja $\mathbf{B}^{(0)} = [a_{\mathbf{B}_1} \dots a_{\mathbf{B}_l} \dots a_{\mathbf{B}_m}]$ a matriz básica inicial. No final da primeira iteração foi requerida a construção de uma nova matriz básica $\mathbf{B}^{(1)}$, pois a l-ésima coluna deixará a base e a k-ésima coluna da matriz $\mathbf{N}$ entrará na base. $\mathbf{B}^{(1)}$ terá a seguinte forma

$$\mathbf{B}^{(1)} = [a_{\mathbf{B}_1} \dots a_{\mathbf{B}_{l-1}} a_{\mathbf{N}_k} a_{\mathbf{B}_{l+1}} \dots a_{\mathbf{B}_m}].$$

A decomposição de $\mathbf{B}^{(1)}$ a partir de $\mathbf{B}^{(0)}$ é particularmente fácil como também estável. Considere a matriz básica $\mathbf{B}^{(1)}$ obtida de $\mathbf{B}^{(0)}$ retirando a l-ésima coluna, deslocando todas as colunas subsequentes uma posição à esquerda e colocando a k-ésima coluna de $\mathbf{N}$ à direita. Consequentemente,

$$\mathbf{B}^{(1)} = [a_{\mathbf{B}_1} \dots a_{\mathbf{B}_{l-1}} a_{\mathbf{B}_{l+1}} \dots a_{\mathbf{B}_m} a_{\mathbf{N}_k}].$$

Assim,

$$\mathbf{L}^{-1}\mathbf{PB}^{(1)} = [\mathbf{L}^{-1}\mathbf{P}a_{\mathbf{B}_1} \dots \mathbf{L}^{-1}\mathbf{P}a_{\mathbf{B}_{l-1}} \mathbf{L}^{-1}\mathbf{P}a_{\mathbf{B}_{l+1}} \dots \mathbf{L}^{-1}\mathbf{P}a_{\mathbf{B}_m} \mathbf{L}^{-1}\mathbf{P}a_{\mathbf{N}_k}]$$

$$\mathbf{L}^{-1}\mathbf{PB}^{(1)} = [u_1, u_2, \dots u_{r_j-1}, u_{r_j+1}, \dots, u_m, \mathbf{L}^{-1}\mathbf{P}a_{\mathbf{N}_k}] = \mathbf{H}^{(1)}.$$

onde u_i são as colunas da matriz $\mathbf{U}^{(0)}$, uma vez que

$$\mathbf{L}^{-1}\mathbf{PB}^{(0)} = \mathbf{U}^{(0)},$$

e $\mathbf{H}^{(1)}$ tem a forma Hessenberg (os elementos abaixo da subdiagonal inferior são nulos) com zeros abaixo da diagonal principal nas primeiras l-1 colunas. O vetor $\mathbf{L}^{-1}\mathbf{P}a_{\mathbf{N}_k}$ já foi obtido quando a

direção simplex foi calculada, antes da atualização da base.

De fato, ao resolver o sistema $\mathbf{Bd_B} = -a_{\mathbf{N}_k}$ temos

$$\mathbf{PBd_B} = -\mathbf{P}a_{\mathbf{N}_k} \Rightarrow \mathbf{LUd_B} = -\mathbf{P}a_{\mathbf{N}_k} \Rightarrow \mathbf{t} = -\mathbf{L}^{-1}\mathbf{P}a_{\mathbf{N}_k}.$$

Assim, $\mathbf{H}^{(1)}$ pode ser construída sem qualquer esforço adicional, e pode ser reduzida a uma matriz triangular superior $\mathbf{U}^{(1)}$ usando a eliminação Gaussiana para anular os elementos sub-diagonais nas colunas l até m. A seleção do pivô é feita em cada passo da eliminação, podendo ter ou não troca entre duas linhas adjacentes. Assim, $\mathbf{U}^{(1)}$ é obtida de $\mathbf{H}^{(1)}$ aplicando uma sequência de operações elementares à esquerda de $\mathbf{H}^{(1)}$.

Após outra iteração do método Simplex, a matriz básica $\mathbf{B}^{(2)}$ resultante poderia ser decomposta a partir da decomposição de $\mathbf{B}^{(1)}$, da mesma forma que $\mathbf{B}^{(1)}$ foi decomposta a partir de $\mathbf{B}^{(0)}$.

3.2.1 Variantes de Bartels-Golub

Assim como o algoritmo de Bartels-Golub, estas duas variantes são usadas para que a decomposição $\mathbf{LU}$ da matriz básica não seja realizada em todas iterações. Elas fazem atualizações da decomposição $\mathbf{LU}$ inicial até que uma nova decomposição da base seja requerida e, em seguida, passam a fazer novamente as atualizações desta nova decomposição, e assim sucessivamente, até chegar à solução do problema.

As duas variantes do algoritmo de Bartels-Golub tentam manter tanto a esparsidade existente nas matrizes básicas quanto a estabilidade numérica.

Variante 1 - Algoritmo Esparso de Bartels-Golub

Seja $\mathbf{B}^{(0)}$ a matriz básica inicial no método Simplex. Aplicando a eliminação de Gauss, com linhas e colunas permutadas, obtemos a decomposição $\mathbf{LU}$ desta base, que pode ser expressa algebricamente da seguinte forma

$$\mathbf{E}_m^{(0)}\mathbf{E}_{m-1}^{(0)}\ldots\mathbf{E}_1^{(0)}\mathbf{B}^{(0)} = \mathbf{P}^{(0)}\mathbf{U}^{(0)}\mathbf{Q}^{(0)}$$

onde,

- $\mathbf{E}_m^{(0)}\mathbf{E}_{m-1}^{(0)}\ldots\mathbf{E}_1^{(0)} = (\mathbf{L}^{(0)})^{-1}$;
- $\mathbf{P}$ e $\mathbf{Q}$ são matrizes de permutação.

Cada coluna da matriz triangular inferior $\mathbf{L}$ pode ser armazenada como um produto de matrizes elementares. Tal formação pode simplificar bastante muitas operações algébricas feitas com a inversa da matriz triangular inferior. É importante ressaltar que decompor uma matriz triangular inferior em um produto de matrizes elementares não envolve nenhum cálculo adicional.

Definição 3.2.1 *Uma coluna p é chamada coluna espeto se contém pelo menos um elemento não nulo abaixo da diagonal (Figura 3.1). Geralmente, um vetor espeto também é um vetor esparso e, consequentemente, o último elemento não nulo pode estar em uma posição $r < m$ com mostra a Figura 3.2.*

Denotamos por $\mathbf{B}^{(1)}$ a matriz básica da iteração seguinte, que difere de $\mathbf{B}^{(0)}$ em apenas uma coluna. Assim, $\mathbf{B}^{(1)}$ satisfaz a equação

$$\mathbf{E}_m^{(0)}\mathbf{E}_{m-1}^{(0)}\ldots\mathbf{E}_1^{(0)}\mathbf{B}^{(1)} = \mathbf{P}^{(0)}\mathbf{S}^{(1)}\mathbf{Q}^{(0)}$$

onde a matriz $\mathbf{S}^{(1)}$ difere da matriz $\mathbf{U}^{(0)}$ também por somente uma coluna (coluna espeto), a mesma em que $\mathbf{B}^{(0)}$ e $\mathbf{B}^{(1)}$ diferem. A coluna da matriz $\mathbf{S}^{(1)}$ que difere da matriz $\mathbf{U}^{(0)}$ possui elementos não nulos abaixo da diagonal, destruindo a triangularidade de $\mathbf{U}^{(0)}$.

Bartels e Golub sugeriram fazer permutações de colunas movendo a coluna espeto para a última coluna da matriz e deslocando as posteriores à espeto uma posição para a esquerda. Com isso, obtém-se uma matriz na forma subtriangular superior e, depois, realizar-se operações elementares por linha, juntamente com trocas de linhas adjacentes, se necessárias, para restaurar a forma triangular superior.

O objetivo principal desta variante é tentar manter a esparsidade presente nas matrizes básicas. Para isto, Reid propôs que, ao fazer a decomposição $\mathbf{LU}$ da matriz básica inicial, deve-se utilizar uma heurística de pivoteamento. A heurística que ele utilizou foi a *heurística de Markowitz* (Duff et al. [20]), pelo fato de existirem bons resultados obtidos por esta heurística em trabalhos anteriores. Esta estratégia difere a variante proposta por Reid, para o algoritmo de Bartels-Golub, o qual utiliza

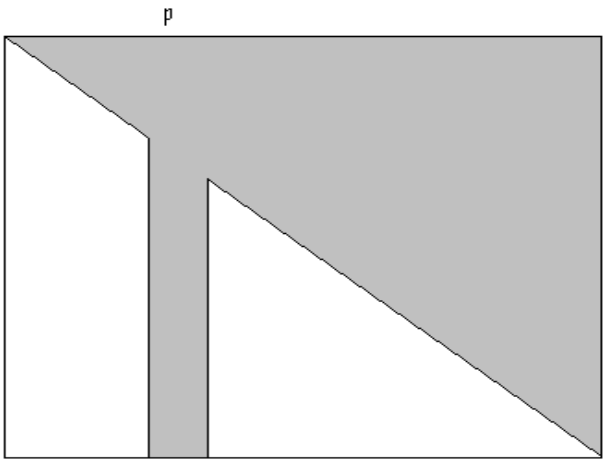

Fig. 3.1: A coluna espeto na posição p destrói a triangularidade de $\mathbf{U}$.

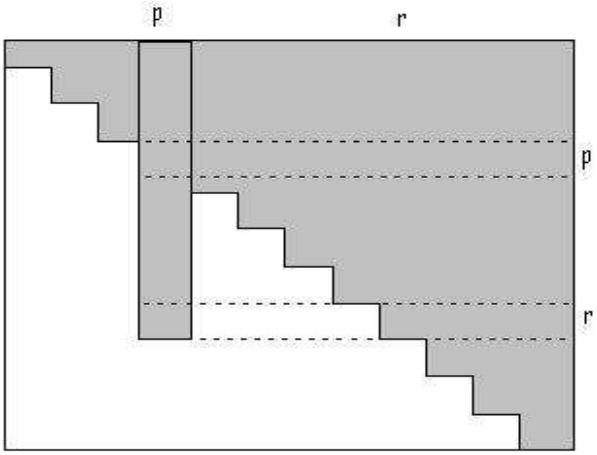

Fig. 3.2: Uma coluna espeto p tem seu último elemento não nulo na linha r.

o pivoteamento parcial, que não preserva a esparsidade.

A heurística proposta por Markowitz consiste numa estratégia eficiente para manter a esparsidade na decomposição $\mathbf{LU}$. Trata-se de uma das mais usadas em implementações e apresenta bons resultados, apesar de ser computacionalmente cara (Duff et al. [20]). Considere a matriz original como ativa inicialmente: $b_{ij}^{(1)} = b_{ij}$. A heurística de Markowitz (Duff et al. [20]) consiste na sequência de passos descrita a seguir.

Para $t = 1$ até $n - 1$ faça:

1. Determine $r_i^{(t)}$ (número de elementos não nulos na linha i) para todas as linhas i da matriz ativa na iteração t.

2. Determine $c_j^{(t)}$ (número de elementos não nulos na coluna j) para todas as colunas j da matriz ativa na iteração t.

3. Determine $(r_p^{(t)} - 1)(c_q^{(t)} - 1) = min\{(r_i^{(t)} - 1)(c_j^{(t)} - 1),$ para todo i, j tal que $b_{ij}^{(t)} \neq 0\}$. Se $r_p^{(t)} = 1$ ou $c_q^{(t)} = 1$ este único elemento da linha ou coluna será o pivô.

4. Troque linhas e colunas para que o pivô $\alpha_{pq}^{(t)}$ seja colocado na posição (t, t).

5. Efetue a eliminação de Gauss, se necessária (se $c_q^{(t)} = 1$, eliminações não são necessárias) e redefina a matriz ativa, excluindo a linha t e a coluna t (que correspondem na matriz original à linha p e à coluna q).

Existe também a necessidade de se utilizar uma estratégia de pivoteamento para manter a esparsidade no momento de reduzir a matriz na forma subtriangular superior à forma triangular superior, tentando evitar que preenchimentos excessivos ocorram. Todos os elementos da subdiagonal são não nulos, pois eles eram elementos da diagonal na matriz triangular superior anterior, mas é provável que muitos elementos da diagonal sejam nulos devido à esparsidade. Se isto ocorrer, muitos passos da redução consistirão apenas de trocas de linhas.

Se os elementos da diagonal e subdiagonal são não nulos, do ponto de vista de esparsidade, deveria ser escolhido como pivô aquele que possui o menor número de elementos não nulos em sua linha. Mas, como é indesejável, por motivo de estabilidade numérica, ter um pivô muito pequeno,

deve-se tomar como pivô o maior dos dois elementos (diagonal e subdiagonal), se o menor dos elementos é menor que uma constante u vezes o maior elemento, $0 < u \leq 1$. Caso contrário, o pivô será o elemento que se encontra na linha que possui o menor número de elementos não nulos.

Variante 2 - Aperfeiçoamento do Algoritmo Esparso de Bartels-Golub

Esta segunda variante do algoritmo de Bartels-Golub, também proposta por Reid, é um aperfeiçoamento da Variante 1. Com um pouco mais de permutações de linhas e colunas, antes de fazer as eliminações para colocar a matriz novamente na forma triangular superior, tenta-se evitar ao máximo tais eliminações e, conseqüentemente, os preenchimentos.

Segundo Saunders [55], é possível explorar o fato que, no caso esparso, a coluna espeto também é bastante esparsa. Ao invés de colocar a coluna espeto na última posição da matriz e mover as posteriores a ela uma posição para a esquerda, como é feito na Variante 1, a coluna espeto pode ser colocada na posição da linha em que se encontra o seu último elemento.

Por exemplo, suponha que a nova matriz $\mathbf{U}^{(1)}$ a ser colocada na forma triangular superior é de ordem 10, que a coluna espeto está na coluna 2 e que seu último elemento não nulo está na linha 5. Ao invés de colocar a coluna espeto no lugar da coluna 10 e mover as colunas 3 até 9 uma posição para a esquerda, deve-se colocá-la no lugar da coluna 5 e mover as colunas 3 e 4 uma posição para a esquerda. Desta forma, será obtida uma matriz na forma subtriangular superior com um número menor de elementos não nulos na subdiagonal; ou seja, existem apenas elementos não nulos nas colunas 2 a 4.

Reid, aprimorando esta idéia, sugeriu fazer mais algumas permutações de linhas e colunas antes de realizar as eliminações. Vejamos como isto pode ser feito de uma forma mais geral.

Suponha que a coluna espeto esteja na posição l e que seu último elemento não nulo esteja na linha q. Restringimos nossa atenção para a submatriz não triangular formada pelas linhas e colunas de l a q. O objetivo agora é ir reduzindo o tamanho desta submatriz não triangular para colocar a matriz original na forma triangular superior ou, então, na forma subtriangular superior com menos elementos não nulos na subdiagonal. Com isso, reduz-se o número de eliminações a serem realizadas.

O próximo passo consiste em procurar dentro desta submatriz não triangular, a partir da sua

segunda coluna (pois a primeira é a coluna espeto), colunas que possuam apenas um elemento não nulo e que este elemento esteja na diagonal.

Suponha que a coluna p da submatriz possua tais características. Em seguida, deve-se fazer a permutação simétrica (a mesma troca para linhas e colunas) colocando o p-ésimo elemento diagonal na l-ésima posição, ou seja, a p-ésima coluna deve ser colocada na posição da coluna l e as demais colunas $l+1, \ldots, p-1$ são movidas uma posição para a direita. Assim, a p-ésima linha deve ser colocada na posição da linha l e as demais linhas $l+1, \ldots, p-1$ são movidas uma posição para baixo.

Com estas permutações, a forma da matriz é preservada e o comprimento da coluna espeto, agora na coluna $l+1$, é reduzido de um. Conseqüentemente, o tamanho da submatriz não triangular também foi reduzido.

A coluna espeto passa a ser novamente a primeira coluna da submatriz não triangular. O processo de procurar por colunas que possuem apenas um elemento não nulo na diagonal é repetido na nova submatriz não triangular de linhas e colunas $l+1$ até p. Após isso, nenhuma das colunas $l+2, \ldots, p$ pode possuir apenas um elemento não nulo na diagonal. Então, a procura deve ser iniciada a partir da coluna $p+1$. Tal procura deve ser feita até que não sejam encontradas colunas com estas características (as permutações podem ser feitas após todas as colunas serem encontradas).

Em seguida, deve ser aplicado um procedimento análogo a este às linhas da submatriz não triangular. Suponha que a nova, e possivelmente menor, submatriz não triangular seja composta por linhas e colunas de l até q. A procura vai ser feita por linhas que possuam apenas um elemento não nulo e este elemento se encontra na diagonal a partir das linhas $q-1$ até $l+1$. Se a linha p da submatriz possuir tais características, deve-se fazer a permutação simétrica (a mesma troca para linhas e colunas), colocando o p-ésimo elemento diagonal na q-ésima posição, ou seja, a p-ésima linha deve ser colocada na posição da linha q e as demais linhas $p+1, \ldots, q$ são movidas uma posição para cima e a p-ésima coluna deve ser colocada na posição da coluna q e as demais colunas $p+1, \ldots, q$ são movidas uma posição para esquerda.

A nova procura por mais linhas que tenham um único elemento não nulo e que esteja na diagonal deve ser feita a partir da linha $p-1$ até l. Com estas permutações, a forma da matriz também foi preservada e o tamanho da submatriz não triangular poderá ser reduzido de um, retirando-se a linha q e coluna q da submatriz não triangular. O processo de procurar por linhas que contêm apenas

um elemento não nulo que esteja na diagonal é feito até que não sejam encontradas linhas com estas características (as permutações podem ser feitas após todas as linhas serem encontradas).

A seguir, deve-se fazer permutações de colunas adicionais, para colocar a submatriz não triangular na forma Hessenberg superior.

Neste momento, a única coluna na submatriz não triangular que pode ter apenas um elemento não nulo na diagonal é a última, pois senão a última linha teria estas características. A permutação simétrica novamente deve ser feita para colocar este elemento diagonal no topo da submatriz, da mesma forma descrita anteriormente.

A última coluna da nova submatriz não triangular pode novamente possuir um único elemento não nulo e na diagonal. Neste caso, o procedimento anterior deve ser repetido. Este último passo continua até que não se tenha mais uma submatriz não triangular, ou a última coluna não possua apenas um elemento não nulo e na diagonal (mais uma vez todas estas permutações podem ser realizadas posteriormente).

Após todas as permutações possíveis serem realizadas, a matriz $\mathbf{U}$ está pronta para que as eliminações sejam feitas, se forem necessárias, com o objetivo de tornar a matriz $\mathbf{U}$ triangular superior novamente, caso isto ainda não tenha acontecido. Com a realização de todos estes passos, obtém-se a atualização da decomposição $\mathbf{LU}$ feita anteriormente.

3.3 Método de Forrest-Tomlin

Primeiramente, supomos que a matriz básica $\mathbf{B}^{(0)}$ foi decomposta no produto $\mathbf{LU}$ utilizando a heurística de Markowitz [40] com o objetivo de reduzir o preenchimento, procurando manter a estrutura esparsa da matriz. Seja $\mathbf{B}^{(1)}$ a matriz básica da iteração seguinte, a qual difere da anterior por apenas uma coluna. Digamos que a coluna p da base anterior foi substituída pela coluna não básica k de $\mathbf{N}$, obtendo-se $\mathbf{B}^{(1)}$.

Considere $a^p = \mathbf{L}^{-1}\mathbf{P}a_k$ a forma atualizada do vetor básico a_k. A matriz $\mathbf{L}^{-1}\mathbf{P}\mathbf{B}^{(1)}$ não é mais triangular superior, embora difira da matriz $\mathbf{U}$ somente na coluna p.

De acordo com o método de Bartels-Golub descrito a seguir, multiplica-se $\mathbf{L}^{-1}\mathbf{P}\mathbf{B}^{(1)}$ à direita

por uma matriz de permutação $\mathbf{Q}$, a qual coloca a coluna a^p na posição m da matriz e move as colunas $p+1$ até m uma posição para a esquerda, ou seja,

$$\mathbf{L}^{-1}\mathbf{P}\mathbf{B}^{(1)}\mathbf{Q} = \mathbf{H}, \tag{3.1}$$

onde $\mathbf{H}$ é uma matriz de Hessenberg (Golub e Loan [30]).

A matriz $\mathbf{H}$ resultante pode ser colocada na forma triangular superior de várias formas. Bartels-Golub sugerem que seja utilizado o pivoteamento parcial. Este procedimento busca controlar o crescimento dos erros de arredondamento (estabilidade numérica), mas tem a desvantagem de que, devido às trocas de linhas, podem ocorrer preenchimentos excessivos. Forrest-Tomlin [24] adotaram a estratégia de eliminar os elementos da linha p, nas colunas p até $m-1$ da matriz $\mathbf{H}$. Para isto, deve-se multiplicar a esquerda de $\mathbf{H}$ pela matriz $\mathbf{R}^{-1}$, que veremos como calcular em seguida e, então, realizar permutação de linhas para torná-la uma matriz triangular superior.

A matriz $\mathbf{R}$ possui a seguinte forma

$$\mathbf{R} = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ 0 & \dots & 0 & 1 & r_{p,p+1} & \dots & r_{p,m} & \\ & & & & \ddots & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \end{pmatrix}.$$

Algebricamente, $\mathbf{R}$ pode ser escrita da seguinte forma:

$$\mathbf{R} = \mathbf{I} + e_p r^t,$$

onde e_p é um vetor de zeros com o elemento 1 somente na posição p e $r^t = (0, \dots, r_{p+1}, \dots, r_m)$.

A inversa da matriz $\mathbf{R}$ é facilmente obtida e pode ser expressa como

$$\mathbf{R}^{-1} = \mathbf{I} - e_p r^t.$$

As colunas p até $m-1$ de $\mathbf{H}$ são as mesmas colunas de $p+1$ até m da matriz $\mathbf{U}$, então a matriz

$\mathbf{R}^{-1}$ elimina os elementos $h_{p,p}, \ldots, h_{p,m-1}$ ou $u_{p,p+1}, \ldots, u_{p,m}$. Este fato pode ser usado para simplificar a determinação do vetor r^t.

Sejam,

$$\tilde{u}^t = (0, \ldots, 0, u_{p,p+1}, \ldots, u_{p,m}) \quad \text{e} \quad r^t = \tilde{u}^t \mathbf{U}^{-1}. \tag{3.2}$$

Então,

$$\mathbf{R}^{-1}\mathbf{U} = (\mathbf{I} - e_p r^t)\mathbf{U} = \mathbf{U} - e_p \tilde{u}^t \mathbf{U}^{-1}\mathbf{U} = \mathbf{U} - e_p \tilde{u}^t$$

e a última expressão é simplesmente a matriz $\mathbf{U}$ com os elementos $u_{p,p+1}, \ldots, u_{p,m}$ nulos.

Agora seja,

$$\mathbf{V} = \mathbf{R}^{-1}\mathbf{H}. \tag{3.3}$$

Então, $\mathbf{V}$ difere de $\mathbf{H}$ tendo os elementos $h_{p,p}, \ldots, h_{p,m-1}$ substituídos por zeros e a última coluna é $\mathbf{R}^{-1}a^p$. A matriz $\mathbf{V}$ é claramente uma matriz triangular permutada. Se multiplicarmos $\mathbf{V}$ à esquerda pela matriz de permutação $\mathbf{Q}^{-1}$ (a matriz de permutação $\mathbf{Q}$ foi usada anteriormente para colocar a matriz $\mathbf{U}$ na forma subtriangular superior), obtemos a matriz triangular superior $\mathbf{U}^{(1)}$, ou seja, $\mathbf{U}^{(-1)} = \mathbf{Q}^{-1}\mathbf{V}$. Para um melhor entendimento, estes últimos passos são ilustrados a seguir.

Seja $\mathbf{Q}$ a matriz de permutação utilizada para colocar a matriz $\mathbf{U}$ na forma subtriangular superior e suponha que a seguinte matriz $\mathbf{H}$ seja o resultado do produto $\mathbf{QU}$

$$\mathbf{H} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

Ao multiplicarmos a esquerda da matriz $\mathbf{H}$ pela matriz $\mathbf{R}^{-1}$ obtemos a seguinte matriz $\mathbf{V}$:

$$
\begin{array}{c}
\\
\\
\mathbf{V} = \\
\\
\\
\\
\\
\\
\end{array}
\begin{array}{c}
p-\text{ésima coluna} \\
\downarrow \\
\begin{pmatrix}
\times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times \\
 & & & & & \times \\
 & & \times & \times & \times & \times \\
 & & & \times & \times & \times \\
 & & & & \times & \times
\end{pmatrix}
\end{array}
\begin{array}{l}
\\
\\
\\
\\
\leftarrow l-\text{ésima linha} \\
\\
\\
\\
\end{array}
$$

Finalmente, deve-se fazer uma permutação nas linhas da matriz $\mathbf{V}$, para colocar a p-ésima linha na posição da m-ésima linha e mover as demais linhas, $p+1$ até m, uma posição para cima. Para isto, basta multiplicar a esquerda de $\mathbf{V}$ pela matriz de permutação $\mathbf{Q}^{-1}$, obtendo a seguinte matriz

$$
\mathbf{U}^{(1)} = \begin{pmatrix}
\times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times \\
 & & & \times & \times & \times \\
 & & & & \times & \times \\
 & & & & & \times
\end{pmatrix}.
$$

Este procedimento faz com que a esparsidade seja preservada na matriz triangular superior, pois somente a linha p é alterada. De (3.1) e (3.3) pode-se escrever

$$
\mathbf{P}(\mathbf{B}^{(1)})^{-1} = \mathbf{Q}\mathbf{V}^{-1}\mathbf{R}^{-1}\mathbf{L}^{-1}. \tag{3.4}
$$

Alternativamente, utilizando a definição $\mathbf{V} = \mathbf{Q}\mathbf{U}^{(1)}$ temos:

$$
\mathbf{P}(\mathbf{B}^{(1)})^{-1} = \mathbf{Q}(\mathbf{U}^{(1)})^{-1}\mathbf{Q}^{-1}\mathbf{R}^{-1}\mathbf{L}^{-1}. \tag{3.5}
$$

De forma geral, sejam $\mathbf{U} = \mathbf{V}^{(0)}$ e $\mathbf{R} = \mathbf{R}^{(1)}$. Então, na iteração t do método Simplex, a inversa

da base $\mathbf{B}^{(t)}$ é representada como

$$\mathbf{P}(\mathbf{B}^{(1)})^{-1} = \mathbf{Q}^{(t)} \dots \mathbf{Q}^{(1)}(\mathbf{V}^{(t)})^{-1}(\mathbf{R}^{(t)})^{-1} \dots (\mathbf{R}^{(1)})^{-1}\mathbf{L}^{-1}.$$

3.4 Método de Atualização de Suhl e Suhl

Supomos que o vetor coluna da variável que entra a_q é trocado pela p-ésima variável básica (com o vetor coluna correspondente $\mathbf{B}e_p$). A nova base pode ser escrita formalmente como

$$\bar{\mathbf{B}} = \mathbf{B} + (a_q - \mathbf{B}e_p)e_p^t. \tag{3.6}$$

Se (3.6) é multiplicada por $\mathbf{L}^{-1}$, obtemos

$$\mathbf{L}^{-1}\bar{\mathbf{B}} = \mathbf{U} + (\mathbf{L}^{-1}a_q - \mathbf{U}e_p)e_p^t. \tag{3.7}$$

Nesta forma, as permutações são omitidas para uma compreensão mais fácil da essência do procedimento. A equação (3.7) diz que, para obter um novo $\mathbf{L}^{-1}\bar{\mathbf{B}}$, a coluna p de $\mathbf{U}$ deve ser trocada por um vetor g derivado do vetor que entra a_q, $g = \mathbf{L}^{-1}a_q$. O vetor g é a coluna espeto representada na Figura 3.1; a Figura 3.2 representa $\mathbf{L}^{-1}\bar{\mathbf{B}}$.

A idéia chave na atualização $\mathbf{LU}$ esparsa é a utilização da esparsidade da coluna espeto. Shul e Suhl [58] permutam a coluna espeto r, que corresponde a posição do seu último elemento diferente de zero. Como resultado, pode-se obter uma matriz de Hessenberg superior, como mostra a Figura 3.3.

Isto está em contraste com o método de Forrest-Tomlin [24], que permuta a coluna espeto para a última coluna e, então permuta, a linha p para a última linha, levando a uma estrutura de linha espeto, como mostra a Figura 3.4, que não é uma matriz de Hessenberg.

Retornando ao método de Suhl e Suhl [58], durante a decomposição $\mathbf{LU}$ as colunas da base devem ser permutadas de forma que se obtenha um pivô abaixo da diagonal. A ordem do pivô atual é mantida em uma matriz de permutação $\mathbf{R}$. Durante a atualização $\mathbf{LU}$, são realizadas permutações de linhas e colunas para que os elementos diagonais permaneçam pivôs. Dessa forma, é suficiente apenas uma permutação da matriz $\mathbf{R}$.

Se os índices originais da linha são mantidos na parte $\mathbf{L}$ e as permutações são aplicadas na parte

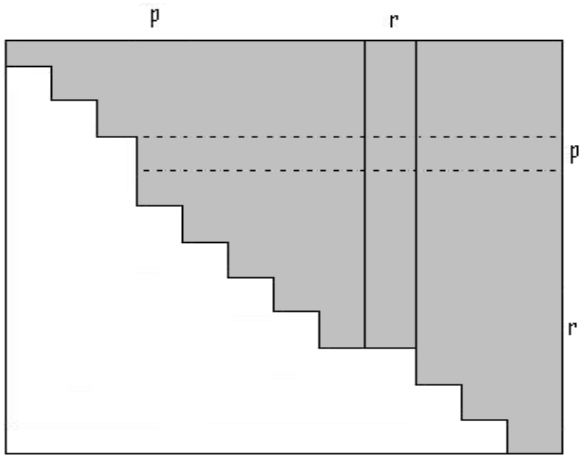

Fig. 3.3: A matriz de Hessenberg superior obtida movendo a coluna espeto para a posição r enquanto desloca as outras colunas para esquerda para preencher a abertura.

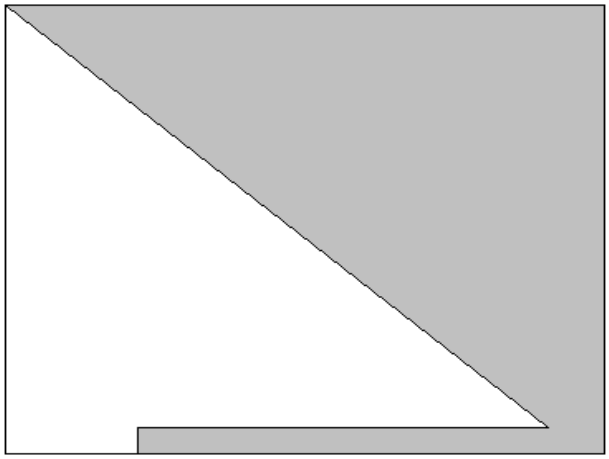

Fig. 3.4: Estrutura da linha espeto do método de atualização de Forrest-Tomlin.

$\mathbf{U}$, então a coluna espeto $\mathbf{L}^{-1}a_q$ pode ser calculada e inserida sem permutações. Se há uma troca de base, uma permutação adicional da matriz é aplicada para a matriz espetada $\mathbf{U}$, seguida por algumas eliminações para restaurar sua forma triangular. Uma iteração k é representada por

$$(\mathbf{L}^k)^{-1}\mathbf{B}^k = \mathbf{R}^k\mathbf{U}^k(\mathbf{R}^k)^t,$$

onde $\mathbf{B}^k$ é a matriz básica, $\mathbf{R}^k$ é a matriz de permutação, e $\mathbf{L}^k$ e $\mathbf{U}^k$ são matrizes triangulares inferior e superior, respectivamente.

Agora, impõe-se que os índices e valores de elementos não nulos de $\mathbf{U}$ podem ser guardados na forma linha e coluna, enquanto $\mathbf{L}^{-1}$ é guardada na forma coluna. Pode ser suficiente guardar os valores apenas uma vez, mas isso poderia resultar num acesso não sequencial dos elementos o que pode deteriorar seriamente o desempenho da implementação.

Em resumo, o método de atualização $\mathbf{LU}$ de Suhl e Suhl pode ser descrito na sequência de passos a seguir (mais detalhes encontra-se em Maros [41]).

1- Inicialização. Suponha que a coluna espeto $\mathbf{L}^{(-1)}a_q$ foi calculada e a linha pivô p determinada.

2- Remova a coluna p de $\mathbf{U}$.

3- Faça permutação de coluna. Primeiro, determine a posição r, o último elemento não nulo da coluna espeto (Figura 3.2). Mova as colunas $p, p+1, \ldots, r$ uma posição para a esquerda, deixando a posição r livre. Coloque a coluna espeto na posição r, obtendo a estrutura mostrada na Figura 3.3.

4- Use as linhas $p+1, \ldots, r$ para eliminar entradas na posição $p, p+1, \ldots, r-1$ da linha p. O resultado é mostrado na Figura 3.5. Armazene multiplicadores na parte $\mathbf{L}$, como uma matriz de transformação de linha elementar triangular.

5- Faça permutação de linhas. Coloque a linha p na posição r, depois mova as linhas $p+1, \ldots, r$ uma posição acima. Desta forma, a triagularidade de $\mathbf{U}$ é restaurada (veja Figura 3.6).

6- Armazene a linha modificada (inicialmente a linha p, agora a linha r) com representação esparsa, na forma linha e coluna de $\mathbf{U}$.

Geralmente, uma iteração k após uma decomposição obtém-se uma matriz triangular superior $\mathbf{U}^k$ e uma triangular infeiror $(\mathbf{L}^k)^{(-1)}$. A matriz é armazenada como uma coleção de vetores de linhas e colunas esparsos.

3.5 Método de Atualização Proposto

O método de atualização da decomposição $\mathbf{LU}$ proposto neste trabalho consiste em realizar operações somente nas colunas realmente afetadas pela troca de base, procurando usar da melhor forma possível a estrutura esparsa da matriz.

Pode-se não ter alteração em colunas situadas após a coluna que entra (e) na base, e/ou a que sai (s), devido à estrutura esparsa das colunas envolvidas. Ou seja, se o elemento que encontra-se na linha k e na coluna que entra e da matriz $\mathbf{LU}$ é igual a zero ($\mathbf{LU}(k, e) = 0$), ou o elemento que encontra-se na linha k e na coluna que sai s da matriz $\mathbf{LU}$ é igual a zero ($\mathbf{LU}(k, s) = 0$), então a coluna k não é afetada pelas colunas e ou s.

Neste sentido, com o objetivo obter maior eficiência, ordena-se as colunas da base segundo algum dos critérios de triangularização descritos no Capítulo 4.

Na atualização da decomposição $\mathbf{LU}$ proposta, as operações causadas pela coluna que saiu da base s são desfeitas, efetuando-se as operações na ordem inversa da decomposição, a partir da última coluna até a coluna $s + 1$, considerando-se a esparsidade.

O próximo passo consiste em verificar quais as colunas a partir da coluna que entrou (coluna $e + 1$) são afetadas pela entrada da coluna e na base. A operação será feita apenas nestas colunas, após a operação da própria coluna que entra na base e.

A seguir, apresenta-se os métodos de triangularização utilizados neste trabalho com o objetivo de aperfeiçoar a decomposição $\mathbf{LU}$ da base, reduzindo o preenchimento da matriz.

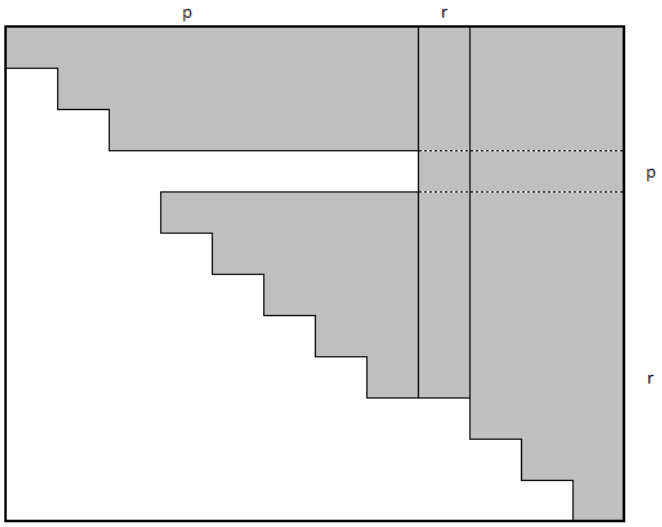

Fig. 3.5: A matriz após as eliminações das posições $p, p+1, \ldots, r-1$ da linha p.

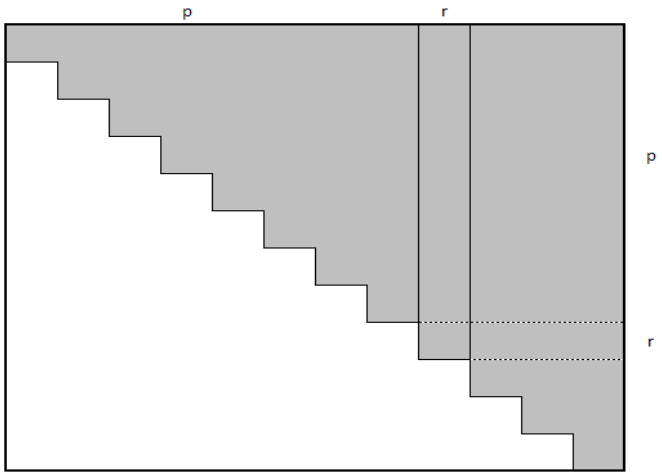

Fig. 3.6: A triangularidade de **U** é recuperada após mover $p+1, \ldots, r$ uma posição.

Capítulo 4

Métodos de Reordenamento das Matrizes

4.1 Introdução

Uma base inicial é necessária para facilitar o método Simplex. A matriz de restrições de (2.1) tem posto completo, então existe pelo menos um conjunto de m colunas linearmente independentes que formam uma base. Um conjunto de colunas linearmente independente pode ser escolhido de diversas formas para formar uma base. Com o objetivo de obter implementações mais eficientes, utilizam-se procedimentos que reduzem o esforço computacional. Neste trabalho, utilizam-se as bases iniciais obtidas pelo MINOS e GLPK.

Na prática, as bases de otimização linear de grande porte quase sempre possuem uma grande parte triangular "escondida", ou seja, existe uma parte triangular que não é óbvia. No entanto, por meio de permutações apropriadas de linhas e colunas da matriz da base, formas triangulares podem ser obtidas.

Neste Capítulo são apresentados alguns procedimentos como a base *Crash* descrita em Maros [41], o método de triangularização de Björck [5] e os métodos de reordenamentos desenvolvidos neste trabalho, mínimo grau e forma bloco triangular, os quais são utilizados no Capítulo 6 com o objetivo de facilitar a decomposição $\mathbf{LU}$ das bases, diminuindo o preenchimento da matriz e sua posterior atualização.

Como estes reordenamentos são estáticos, obtém-se uma decomposição esparsa sem nenhum esforço computacional para obter a ordem das colunas. Os métodos de triangularização podem ser comparados a uma base *Crash* (usada pelo MINOS), pois a triangularização da base não varia durante

as iterações. Uma outra variante desta base é a base CPLEX, proposta por Bixby [4].

4.2 Base *Crash*

Experiências computacionais mostram que uma base inicial com variáveis artificiais geralmente leva a uma solução inicial melhor que uma base com variáveis de folga. Particularmente, as bases triangulares possuem características como:

a. Não exigem inversão;

b. São numericamente estáveis;

c. Encontrá-la é geralmente muito rápido;

d. Operações com elas são rápidas.

Assim, desenvolvem-se algoritmos que criam uma base mais próxima possível de triangular, cujas colunas são colunas artificiais de $\mathbf{A}$ ou vetores unitários de variáveis de folga. Para identificar se uma base é triangular, são necessárias apenas operações lógicas. O mesmo é válido para a seleção de uma matriz triangular de um conjunto de vetores.

Para criar uma base triangular, pode-se começar com uma matriz identidade e trocar suas colunas por colunas de variáveis artificiais de tal forma que a nova matriz torna-se triangular. Dois tipos de bases triangulares $\mathbf{B}$ podem ser definidas na otimização linear. A primeira, triangular inferior e a segunda, a triangular superior. Em ambos os casos, a diagonal principal é diferente de zero.

Considere a partição $\mathbf{A} = [\hat{\mathbf{A}} \mid \mathbf{I}]$. A maneira mais simples de obter uma estrutura triangular é utilizar contadores de linhas e colunas, $contl_i$ e $contc_j$, nos quais definem o número de elementos não nulos na linha i de $\hat{\mathbf{A}}$, e o número de elementos não nulos na coluna j de $\hat{\mathbf{A}}$, respectivamente.

Uma base triangular inferior tem todos elementos nulos à sua direita. Entretanto, para o primeiro elemento da diagonal (canto superior esquerdo), pode-se selecionar uma linha pivô i tal que $contl_i = min_k\{contl_k\}$. A coluna do pivô nesta linha é determinado se $contl_i = 1$. No entanto, se $contl_i > 1$, existem algumas alternativas. Se queremos manter um número baixo de elementos não nulos em $\mathbf{B}$, então uma coluna com elementos não nulos na linha i deve ter o menor número de elementos não nulos para ser escolhida. A seleção é geralmente baseada em alguns critérios adicionais se existe algum

empate. O elemento selecionado torna-se o pivô atual e o posição do pivô é permutada (pela permutação da linha e da coluna) para o canto superior esquerdo. Todas as colunas que têm um número de elementos não nulos na linha i são marcados como indisponíveis. A contagem das linhas e colunas são recalculadas (atualizadas) para o restante das linhas e colunas (a parte ativa) de $\hat{\mathbf{A}}$. Estes passos são repetidos para a parte ativa de $\hat{\mathbf{A}}$. Este procedimento simples é uma base *Crash* triangular inferior. A ordem do pivô é a ordem natural de como as posições do pivô são identificadas.

Uma coluna selecionada depois neste procedimento é tal que ela possui um elemento zero em todas as posições antes do pivô. Isto pode impor um limite no sucesso de encontrar mais componentes triangulares. Em alguns casos, quando o procedimento termina temos vetores que podem ser permutados na forma triangular inferior. Eles podem não fazer uma base cheia mas as posições do pivô que não utilizadas ficam preenchidas pelos vetores unidade correspondentes resultando em uma base triangular inferior $\mathbf{B}$.

Uma (sub) matriz triangular superior pode ser obtida da seguinte forma. Selecione uma coluna com $contc_j$ mínimo. Se este é igual a 1, então a linha pivô é determinada e esta posição é permutada para a posição superior esquerda, o pivô linha e coluna são marcados como indisponíveis, a contagem da coluna é atualizada e busca por um novo $contc_j = min_k\{contc_k\}$. Se $contc_j > 1$, então temos que determinar qual posição na coluna j será pivoteada. Isto geralmente é feito na base da contagem mínima da linha, para as linhas com $a^i_j \neq 0$. Novamente, alguns critérios adicionais podem ser aplicados em caso de empate. Selecionada uma posição do pivô na coluna dada, marcamos esta coluna como indisponível; todas as linhas que têm elementos não nulos nesta coluna são marcados como indisponíveis. Contadores das linhas e colunas são atualizados para a parte restante de $\hat{\mathbf{A}}$ e o procedimento é repetido.

Neste procedimento, colunas selecionadas depois são caracterizadas por terem pelo menos um elemento não nulo em alguma das linhas que ainda está disponível para pivoteamento. Se não temos uma base triangular superior cheia, então as posições do pivô não utilizadas ficam preenchidas pelos vetores unidade correspondentes de $\mathbf{A}$, levando a uma base triangular inferior completa $\mathbf{B}$. A ordem do pivô no caso triangular superior é inversa, ou seja, a posição do pivô encontrada por último é utilizada primeiro.

Em cada passo do procedimento triangular inferior, uma linha é marcada indisponível (a linha pivô) e todas as colunas que cruzam com esta linha. No procedimento triangular superior, uma co-

luna (a coluna pivô) torna-se indisponível e todas as linhas que cruzam com esta coluna. Os dois procedimentos podem encontrar partes triangulares de tamanhos diferentes. A qualidade da base é determinada por diversos fatores, como a densidade, número de condição, factibilidade, e o valor objetivo correspondente. Estas características ou suas combinações desempenham um papel importante na escolha de uma linha ou coluna empatada durante a seleção.

A descrição acima do procedimento *Crash* triangular fornece uma estrutura conceitual e os procedimentos atuais são baseados nestas idéias mas envolve outras considerações.

Se um procedimento não contabilizar os valores numéricos reais dos coeficientes mas utiliza somente a localização de elementos não nulos, dizemos que ele é um procedimento *Crash simbólico*. Se os valores numéricos dos coeficientes são contabilizados então ele é definido como um procedimento *Crash numérico*.

A base *Crash* é utilizada pelo programa MINOS (Capítulo 6). Mais detalhes sobre a base *Crash* encontra-se em Maros [41]. Outros tipos de bases encontram-se em Bixby [4].

4.3 Método do Mínimo Grau

O primeiro reordenamento estático da matriz, o método do mínimo grau (Cantane e Oliveira [9, 10]), inspirado em Bressan e Oliveira [8] e Markowitz (Duff et al. [20]), está implementado no Capítulo 6.

Seja $contc_j$ o número de elementos não nulos da coluna j da matriz $\mathbf{A}$. O primeiro passo é encontrar o número de elementos não nulos da matriz $\mathbf{A}$, coluna por coluna. Selecione o $contc_j = min_k\{contc_k\}$. A coluna $contc_j$ é permutada para a primeira posição da matriz $\mathbf{A}$. Selecionada a coluna, a mesma é marcada como indisponível. O procedimento é repetido sucessivamente buscando o novo $contc_j$ e permutando-o para a próxima posição da matriz $\mathbf{A}$.

Dessa forma, as colunas da matriz $\mathbf{A}$ são ordenadas de acordo com o número de elementos não nulos, em ordem não decrescente.

4.4 Forma Bloco Triangular

A forma bloco triangular (Cantane et al. [12]) é o segundo método de reordenamento estático implementado no Capítulo 6 e obtém uma matriz triangular superior.

Primeiramente, como no método do mínimo grau (Seção 4.3), encontra-se o número de elementos não nulos da matriz $\mathbf{A}$, coluna por coluna. O próximo passo consiste em encontrar a coluna que possui o menor número de elementos não nulos, ou seja, $contc_j = min_k\{contc_k\}$.

Se $contc_j = 1$ então a linha do pivô é determinada nesta coluna e esta posição é permutada para a posição superior esquerda. Encontra-se todas as colunas que possuem um elemento não nulo na linha da coluna escolhida e retira-se um grau de cada uma delas. A contagem da coluna é atualizada e busca-se um novo $contc_j$.

Se $contc_j > 1$ então temos que determinar qual posição na coluna j será pivoteada. A linha que possui o menor número de elementos não nulos, na coluna j, será escolhida. Selecionada uma posição do pivô na coluna dada, marcamos esta coluna como indisponível e retira-se um grau de cada coluna que possui um elemento não nulo na linha escolhida. Assim, os contadores de linhas e colunas são atualizados e o procedimento é repetido.

4.5 Triangularização de Björck

Este método encontra-se em Björck [5]. Considere as matrizes $\mathbf{A}$ e $\mathbf{PA}$ onde o bloco superior e inferior possuem m e n linhas respectivamente.

$$\mathbf{A} = \left(\begin{array}{ccccc} \times & \times & \times & \times & \times \\ \times & & & & \\ \vdots & & & & \\ \times & & & & \\ \hline \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{array}\right), \mathbf{PA} = \left(\begin{array}{ccccc} \times & & & & \\ \times & & & & \\ \vdots & & & & \\ \times & & & & \\ \hline \times & \times & \times & \times & \times \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{array}\right).$$

Assumimos que as linhas de $\mathbf{A}$ não tem normas muito distintas, o ordenamento das linhas não afeta a estabilidade numérica e pode ser escolhido baseado somente na esparsidade. Considera-se a seguinte heurística para determinar um ordenamento de linhas, que é uma extensão do ordenamento de linhas recomendado para matrizes esparsas.

Sejam $p_i(\mathbf{A})$ e $u_i(\mathbf{A})$ os índices do primeiro e do último elementos não nulos na i-ésima linha de $\mathbf{A}$, respectivamente. Primeiramente, são escolhidas as linhas após incremento de $p_i(\mathbf{A})$, tal que $p_i(\mathbf{A}) \leq p_k(\mathbf{A})$ se $i \leq k$. Então para cada grupo de linhas com $p_i(\mathbf{A}) = k,\ k = 1, \ldots, max_i p_i(\mathbf{A})$, escolhemos todas as linhas após incremento de $u_i(\mathbf{A})$.

No Capítulo 6, foi utilizada esta estratégia com ordenamento de colunas no lugar de linhas, pois temos matrizes que possuem mais colunas que linhas. Dessa forma, os papéis dos vetores $p_i(\mathbf{A})$ e $u_i(\mathbf{A})$ foram invertidos, ou seja, $p_i(\mathbf{A})$ e $u_i(\mathbf{A})$ são os índices do primeiro e do último elementos não nulos na i-ésima coluna de $\mathbf{A}$, respectivamente.

4.6 Forma Bloco Triangular com Grafos Bipartidos

Este método foi proposto por Photen e Fan [46] e encontra-se em Björck [5].

Definição 4.6.1 *Uma matriz $\mathbf{A} \in I\!R^{m\times n}$, $m \geq n$, tem propriedade forte de Hall se para cada subconjunto de k colunas, $0 < k < m$, a submatriz correspondente tem elementos não nulos em pelo menos $k+1$ linhas. (Então, quando $m > n$, cada subconjunto de $k \leq n$ tem esta propriedade, e quando $m = n$, cada subconjunto de colunas $k < n$ tem esta propriedade.)*

A propriedade forte de Hall implica em posto completo.

Teorema 4.6.1 *Seja $\mathbf{A} \in I\!R^{m\times n}$, $m \geq n$ com uma propriedade forte de Hall. Então a estrutura de $\mathbf{A}^t\mathbf{A}$ irá predizer corretamente a estrutura de $\mathbf{R}$, onde $\mathbf{A} = \mathbf{QR}$ e $\mathbf{A}^t\mathbf{A} = \mathbf{R}^t\mathbf{R}$, excluindo cancelamentos numéricos.*

Obviamente, a matriz $\mathbf{A}$ na Figura 4.6 tem a propriedade forte de Hall, uma vez que a primeira coluna tem somente um elemento não nulo. No entanto, a matriz $\tilde{\mathbf{A}}$ obtida eliminando a primeira coluna tem esta propriedade.

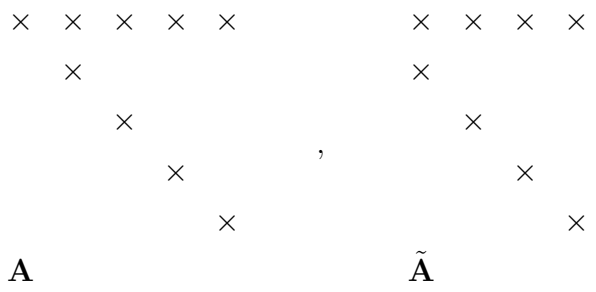

Fig. 4.1: Propriedade forte de Hall das matrizes $\mathbf{A}$ e $\tilde{\mathbf{A}}$.

Uma matriz retangular arbitrária $\mathbf{A} \in I\!R^{m\times n}$, $m \geq n$, por meio de permutações de linhas e colunas, pode ter a forma bloco triangular

$$\mathbf{PAQ} = \begin{pmatrix} \mathbf{A}_h & \mathbf{U}_{hs} & \mathbf{U}_{hv} \\ & \mathbf{A}_s & \mathbf{U}_{sv} \\ & & \mathbf{A}_v \end{pmatrix}, \tag{4.1}$$

onde $\mathbf{A}_h$ é bloco diagonal e indeterminada (isto é, possui mais colunas que linhas), $\mathbf{A}_s$ é quadrada e $\mathbf{A}_v$ é sobredeterminada (isto é, possui mais linhas que colunas), e todas as três tem uma diagonal de elementos não nulos (veja o exemplo na Figura (4.6)). As submatrizes $\mathbf{A}_v$ e $\mathbf{A}_h^t$ têm a propriedade forte de Hall. Os blocos sem diagonais denotados por $\mathbf{U}$ são matrizes não nulas de dimensões apropriadas. A forma bloco triangular (4.1) de uma matriz esparsa é baseada em uma decomposição canônica de grafos bipartidos proposto por Dulmage et al. [21, 22, 23, 37].

De acordo com as notações de Pothen e Fan [46], a decomposição de $\mathbf{A}$ em submatrizes $\mathbf{A}_h$, $\mathbf{A}_s$ e $\mathbf{A}_v$ denomina-se *decomposição grosseira*. Um ou dois blocos diagonais podem estar ausentes na decomposição grosseira. Pode ser possível decompor mais o bloco diagonal em (4.1) para obter a *decomposição fina* destas submatrizes. Cada um dos blocos $\mathbf{A}_h$ e $\mathbf{A}_v$ pode ser decomposto na forma bloco diagonal,

$$\mathbf{A}_h = \begin{pmatrix} \mathbf{A}_{h1} & & \\ & \ddots & \\ & & \mathbf{A}_{hp} \end{pmatrix}, \mathbf{A}_v = \begin{pmatrix} \mathbf{A}_{v1} & & \\ & \ddots & \\ & & \mathbf{A}_{vq} \end{pmatrix},$$

onde cada $\mathbf{A}_{h1}, \ldots, \mathbf{A}_{hp}$ é indeterminada e cada $\mathbf{A}_{v1}, \ldots, \mathbf{A}_{vq}$ é sobredeterminada. A submatriz $\mathbf{A}_s$

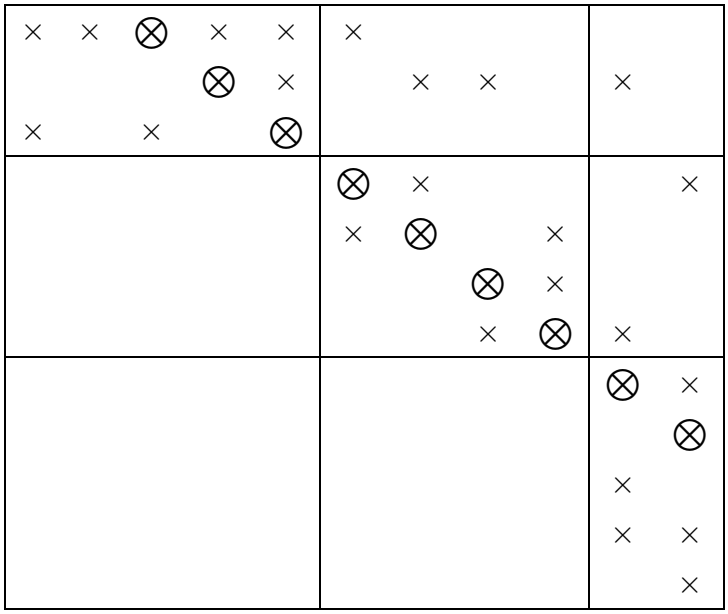

Fig. 4.2: A decomposição bloco triangular grosseira de $\mathbf{A}$

.

pode ser decomposta na forma triangular superior

$$\mathbf{A}_s = \begin{pmatrix} \mathbf{A}_{s1} & \mathbf{U}_{12} & \dots & \mathbf{U}_{1,t} \\ & \mathbf{A}_{s2} & \dots & \mathbf{U}_{2,t} \\ & & \ddots & \vdots \\ & & & \mathbf{A}_{st}, \end{pmatrix} \tag{4.2}$$

com blocos diagonais $\mathbf{A}_{s1}, \dots, \mathbf{A}_{st}$ que têm elementos não nulos na diagonal. Pode-se mostrar que o resultado da decomposição é único. Qualquer forma bloco triangular pode ser obtida de qualquer outra aplicando permutações de linhas que envolvem as linhas de um único bloco de linha, permutações de colunas que envolvem as colunas de um único bloco de coluna, e permutações que reordenam os blocos.

Uma matriz quadrada que pode ser permutada na forma (4.2), com $t > 1$, é chamada redutível; caso contrário é chamada irredutível [1]. Todos os blocos diagonais na decomposição fina são irredutíveis; isto implica que todos $\mathbf{A}_{s1}, \dots, \mathbf{A}_{st}$ tem a propriedade forte de Hall (Coleman et al. [14]).

[1] Alguns autores reservam o termo redutível para o caso $Q = P^t$, e usa o termo biredutível e bi-irredutível

Para o caso em que $\mathbf{A}$ é uma matriz não singular e quadrada, há um algoritmo de duas fases que permuta $\mathbf{A}$ para a forma bloco triangular superior (Tarjan [59], Gustavson [36] e Duff [18, 19]). Um algoritmo para a forma bloco triangular mais geral descrito a seguir é dado por Pothen e Fan [46]. Este algoritmo depende de um conceito de combinação em grafos bipartidos.

O grafo bipartido associado com $\mathbf{A}$ é denotado por $G(\mathbf{A}) = \{L, C, E\}$, onde $L = (l_1, \ldots, l_m)$ representa um conjunto de vértices correspondentes as linhas de $\mathbf{A}$, $C = (c_1, \ldots, c_m)$ representa um conjunto de vértices correspondentes as colunas de $\mathbf{A}$, E representa o conjunto de arestas, e $\{l_i, c_j\} \in E$ se e somente se a_{ij} é não nulo. Uma combinação em $G(\mathbf{A})$ é um subconjunto de arestas que não possuem pontos finais em comum. Na matriz $\mathbf{A}$, isto corresponde ao subconjunto de elementos não nulos, em que dois destes não pertencem a mesma linha ou coluna. Uma combinação máxima é uma combinação com um número máximo $l(\mathbf{A})$ de arestas. O posto estrutural de $\mathbf{A}$ é $l(\mathbf{A})$. Em (4.6), o posto estrutural de $\mathbf{A} \in I\!R^{12\times 11}$ é 9. Note que o posto numérico é sempre menor ou igual a seu posto estrutural.

O algoritmo de Pothen e Fan [46] consiste nos seguintes passos:

1- Encontre a combinação máxima no grafo bipartido $G(A)$ com um conjunto de linhas L e um conjunto de colunas C.

2- De acordo com a combinação, defina a partição L nos conjuntos VL, SL, HL e C nos conjuntos VC, SC, HC, correspondentes aos blocos verticais $\mathbf{A}_v$, quadrados $\mathbf{A}_s$, horizontais $\mathbf{A}_h$.

3- Encontre os blocos diagonais das submatrizes $\mathbf{A}_v$ e $\mathbf{A}_h$ das componentes conexas nos subgrafos $G(\mathbf{A}_v)$ e $G(\mathbf{A}_h)$. Encontre a forma bloco triangular superior da submatriz $\mathbf{A}_s$ das componentes fortemente conexas no subgrafo orientado associado $G(\mathbf{A}_s)$, com arestas orientadas das colunas para as linhas.

O reordenamento na forma bloco triangular na fase de pré-processamento pode reduzir esforço computacional e armazenamento na resolução de problemas de mínimos quadrados. Se $\mathbf{A}$ tem posto estrutural igual a n, então o primeiro bloco de linhas de 4.6 deve ser vazio, e o problema de mínimos quadrados original, após o reordenamento, pode ser resolvido por substituição da última variável para a primeira. Primeiramente calcula-se a solução de

$$\min_{\tilde{x_v}} \|\mathbf{A}_v \tilde{x_v} - \tilde{b_v}\|_2, \tag{4.3}$$

onde $\tilde{x} = \mathbf{Q}^t x$ e $\tilde{b_v} = \mathbf{P}b$ particionados conforme $\mathbf{PAQ}$ em 4.1. A parte restante da solução de $\tilde{x_k}, \ldots, \tilde{x_1}$ é, então, determinada por

$$\mathbf{A_{si}}\tilde{x_i} = \tilde{b_i} - \sum_{j=i+1}^{k} \mathbf{U}_{ij}\tilde{x_j},\ i = k, \ldots, 2, 1. \tag{4.4}$$

Finalmente, temos que $x = \mathbf{Q}\tilde{x}$. Pode-se resolver os subproblemas (4.3) e (4.4) calculando a decomposição $\mathbf{QR}$ de $\mathbf{A}_v$ e $\mathbf{A}_{si},\ i = 1, \ldots, k$. Desde que $\mathbf{A}_{s1}, \ldots, \mathbf{A}_{sk}$ e $\mathbf{A}_v$ tenham a propriedade forte de Hall, as estruturas das matrizes $\mathbf{R}_i$ são dadas pelas estruturas das matrizes normais correspondentes.

Se $\mathbf{A}$ tem posto estrutural n mas é numericamente posto deficiente, não será possível decompor todos os blocos diagonais em (4.2). Neste caso, a estrutura bloco triangular dada pela forma Dulmage-Mendelsohn [21, 22, 23] pode não ser preservada, ou alguns blocos podem tornarem-se mal-condicionados.

Se a matriz $\mathbf{A}$ tem posto estrutural menor que n, então temos um bloco indeterminado $\mathbf{A}_h$. Neste caso, ainda pode-se obter a forma 4.2 com um bloco quadrado $\mathbf{A_{11}}$ permutando as colunas extras do primeiro bloco para o fim. A solução dos mínimos quadrados, então, não é única, mas uma única solução de tamanho mínimo pode ser encontrada Björck [5].

Este método não foi utilizado neste trabalho. O desenvolvimento para a decomposição $\mathbf{LU}$, implementação e comparação com esta abordagem é um dos temas propostos para continuidade deste trabalho.

Capítulo 5

Problemas de Corte de Estoque

5.1 Introdução

Problemas de corte de estoque são essenciais no planejamento da produção em várias indústrias, tais como indústrias de papel, vidro, móveis, plástico, chapas de madeira, aço, têxtil etc. Estes problemas consistem em cortar objetos grandes para produzir um conjunto de itens menores, visando atender uma certa demanda, a partir do corte de peças em estoque, otimizando uma função objetivo, como por exemplo, minimizar o número total de peças em estoque cortadas ou as perdas. Em geral, estes problemas são formulados como problemas de otimização linear inteira.

Um dos focos da tese foi a aplicação do método de atualização da decomposição $\mathbf{LU}$ proposto ao problema de corte de estoque, embora o método tenha aplicação geral para qualquer problema de otimização linear. Neste capítulo, encontra-se uma breve definição do problema de corte de estoque unidimensional (Poldi [47], Silva [57]). Apresenta-se também o método Simplex com geração de colunas utilizado para solucionar este problema de corte de estoque. O objetivo não foi estudar detalhamente este problema, mas simplesmente apresentar o problema que foi estudado em Poldi [47], Poldi e Arenales [49, 50], Silva [57] e utilizado nos testes computacionais do Capítulo 7.

5.2 Classificação dos Problemas de Corte de Estoque

Os problemas de corte de estoque são classificados de acordo com as dimensões que são relevantes do objeto a ser cortado da seguinte forma:

• **Problema de Corte Unidimensional**

Um problema é dito "unidimensional" quando apenas uma dimensão (comprimento) é relevante no processo de corte. Este tipo de problema aparece, por exemplo, nas indústrias de papel e aço, em que bobinas grandes são cortadas em bobinas com comprimentos menores e mesmo diâmetro.

Algumas restrições podem surgir do processo de corte, tais como, limitação do número de facas, compartimentação de itens, entre outras.

• **Problema de Corte Bidimensional**

O problema é dito "bidimensional" quando duas dimensões (comprimento e largura) do objeto a ser cortado são significativas no processo de corte. Resolver este tipo de problema consiste em combinar geometricamente os itens ao longo do comprimento e da largura das peças em estoque.

Encontramos este tipo de problema nas indústrias de placas de vidro, móveis, metalúrgica, de esquadrias de alumínio, entre outras, em que placas retangulares grandes são cortadas em peças retangulares menores (itens), as quais, geralmente, compõem produtos demandados.

Várias restrições surgem no processo de corte, como por exemplo, o tipo de corte usado, a limitação no número de itens, o número máximo de tipos de itens, o número máximo de estágios, entre outras. Na próxima seção encontra-se diversos tipos de padrões de corte classificados conforme tais restrições.

• **Problema de Corte Tridimensional**

No problema "'tridimensional", as três dimensões (comprimento, largura e altura) do objeto a ser cortado são significativas no processo de corte. Este tipo de problema é aplicado, por exemplo, em indústrias de colchões e travesseiros.

• **Problema de Corte** 1.5 **Unidimensional**

É um problema de corte bidimensional com uma das dimensões variável (por exemplo, largura fixa e comprimento suficientemente grande). Um exemplo desse tipo de problema é o de um rolo de tecido, que tem largura fixa e comprimento suficientemente grande para a produção de roupas.

• **Problema de Corte** 2.5 **Dimensional**

É um problema de corte tridimensional com uma das dimensões variável (por exemplo, largura e comprimento fixos e altura suficientemente grande). Um exemplo seria um contêiner com largura e comprimento fixas e suficientemente alto para acomodar um volume da carga.

• **Problema de Corte Multidimensional**

É o problema em que mais de três dimensões do objeto a ser cortado são relevantes no processo de corte. Este problema surge quando a produção de um item requer simultaneamente diferentes recursos independentes e limitados.

5.3 Classificação do Padrão de Corte

Definição 5.3.1 *Padrão de corte é a maneira como um objeto em estoque deve ser cortado para produzir os itens demandados.*

Considerando m o número de tipos de itens demandados, a um padrão de corte j é associado um vetor m-dimensional a_j, em que cada coordenada α_{ij} define a quantidade de itens do tipo i presentes no padrão j

$$a_j = (\alpha_{1j}, \alpha_{2j}, \ldots, \alpha_{mj})^t.$$

Definição 5.3.2 *Um padrão de corte que produza apenas um tipo de item é chamado "padrão de corte homogêneo".*

Em outras palavras, um padrão de corte é homogêneo se o vetor associado tem apenas uma coordenada não-nula: $(0, \ldots, \alpha_{ij}, \ldots, 0), \alpha_{ij} \neq 0$.

Além das dimensões relevantes do objeto, os padrões de corte podem ser classificados de acordo com o tipo de corte usado, a limitação no número de itens, o número de estágios, entre outros. A seguir, são apresentados alguns tipos de padrões de corte. Mais detalhes podem ser encontrados em Silva [57].

- **Padrão de Corte Guilhotinado:** quando todos os cortes realizados no objeto para obter os itens são guilhotinados. Um corte guilhotinado é o corte feito paralelamente a um dos lados do objeto e por toda sua extensão.

- **Padrão de Corte $n-$Estágios:** quando são feitas $n-1$ rotações de 90° no objeto durante o corte do padrão.

- **Padrão de Corte Restrito:** quando existe uma limitação nas quantidades de cada tipo de item que compõe o padrão de corte.

5.4 Problema de Corte de Estoque Unidimensional

O problema de corte de estoque unidimensional pode ser definido como segue.

Considere que temos em estoque um número suficientemente grande de objetos (barras, bobinas, etc) de um determinado comprimento L e um conjunto de pedidos de barras menores de comprimentos l_i, $i = 1, \ldots, m$, os quais chamaremos de itens. Cada item i deve ser produzido na quantidade d_i, $i = 1, \ldots, m$. O problema, então, consiste em produzir os itens a partir do corte das peças em estoque de modo a atender a demanda, otimizando uma certa função. Pode-se, por exemplo, minimizar o número total de peças a serem cortadas, ou minimizar a perda, ou maximizar o lucro, etc.

A modelagem matemática do problema de corte de estoque envolve duas etapas: definir os padrões de corte para os objetos estocados e decidir quantas vezes cada padrão de corte será utilizado para atender à demanda. A primeira etapa pode ser realizada independentemente da demanda dos itens. Neste modelo, considera-se os dados dos itens (comprimento e demanda) e vários tipos de barras em estoque e em quantidade limitada. Detalhes desta formulação matemática encontra-se em Poldi [47].

$$\text{minimizar } \sum_{j=1}^{N_1} c_{j1}x_{j1} + \sum_{j=1}^{N_2} c_{j2}x_{j2} + \ldots + \sum_{j=1}^{N_K} c_{jK}x_{jK}$$

$$\text{sujeito a } \sum_{j=1}^{N_1} a_{j1}x_{j1} + \sum_{j=1}^{N_2} a_{j2}x_{j2} + \ldots + \sum_{j=1}^{N_K} a_{jK}x_{jK} = d$$

$$\begin{aligned} \sum_{j=1}^{N_1} x_{j1} &\leq e_1 \\ \sum_{j=1}^{N_2} x_{j2} &\leq e_2 \\ \ddots &\quad \vdots \\ \sum_{j=1}^{N_K} x_{jK} &\leq e_K \end{aligned} \tag{5.1}$$

$$x_{jk} \geq 0, \text{ inteiros}, j = 1, \ldots, N_k, k = 1, \ldots, K.$$

Índices:

- $k = 1, \ldots, K$: número de tipos (comprimentos) de objetos disponíveis em estoque;
- $j = 1, \ldots, N_k$: o número de padrões de corte para o objeto k, $k = 1, \ldots, K$;
- $i = 1, \ldots, m$: o número de tipos de itens a serem cortados;

Dados:

- m: número de tipos de itens;
- d_i: demanda do item tipo i, $i = 1, \ldots, m$;
- K: número de objetos em estoque;
- e_k: disponibilidade em estoque do objeto k, $k = 1, \ldots, K$;
- α_{ijk}: número de itens do tipo i no j-ésimo padrão de corte para o objeto tipo k (a_{jk}: vetor de dimensão m, cujas componentes são α_{ijk}).

Parâmetros:

- c_{jk}: custo de cortar o objeto tipo k segundo o padrão de corte j, $j = 1, \ldots, N_k$, $k = 1, \ldots, K$;

Observe que c_{jk} pode ser a perda $c_{jk} = L_k - \sum_{i=1}^{m} l_i \alpha_{ijk}$, onde l_i é a dimensão do item i, $i = 1, \ldots, m$ e L_k é a dimensão do objeto k, $k = 1, \ldots, K$; ou simplesmente um custo do objeto k (que independe do padrão de corte): $c_{jk} = c_k$. Algumas vezes, minimizar o custo é mais interessante do que minimizar a perda, pois os objetos podem ser comprados com descontos (por ter dimensões fora dos padrões) ou são retalhados de cortes anteriores que retornaram ao estoque e deseja-se que estes sejam utilizados antes que os outros objetos.

Variáveis de decisão:

- x_{jk}: número de objetos do tipo k cortados conforme o padrão de corte j, $j = 1, \ldots, N_k, k = 1, \ldots, K$.

Observações:

1- As primeiras restrições de igualdade garantem que as quantidades de itens produzidos sejam exatamente igual à demanda. Outras restições (de $\leq$) asseguram que a quantidade de cada tipo de objeto que será cortado para atender à demanda de todos os tipos de itens seja, no máximo, igual à quantidade de cada tipo de objeto disponíveis em estoque. E a última restrição garante que a repetição de cada padrão de corte j seja um número inteiro não-negativo.

2- Cada padrão de corte a_{jk} para o objeto tipo k deve satisfazer as seguintes restrições do problema da mochila:

$$\begin{aligned} & l_1\alpha_{1jk} + l_2\alpha_{2jk} + \ldots + l_m\alpha_{mjk} \leq L_k \\ & \alpha_{ijk} \geq 0 \text{ e inteiro}, i = 1, \ldots, m, j = 1, \ldots, N_k, k = 1, \ldots, K. \end{aligned} \tag{5.2}$$

3- Outras restrições do processo de corte podem ser incluídas em (5.2), como por exemplo, a limitação no número de facas.

4- Este tipo de problema ocorre em indústrias nas quais os objetos ou são adquiridos com antecedência e estocados, ou são produzidos. Entretanto, a capacidade de produção é limitada.

Este modelo foi proposto por Gilmore e Gomory [29] e um método de geração de colunas foi desenvolvido. Os métodos de solução para este modelo encontram-se em Poldi [47] e Silva [57]. Como proposta futura de outra aplicação do método de atualização da decomposição **LU** desenvolvido, pretende-se utilizar o problema de corte de estoque multiperíodo estudado em Poldi [48], Silva [57] e descrito na Seção 5.6.

5.5 Reordenamento para o Problema de Corte de Estoque

Utiliza-se um reordenamento específico das colunas da base para o problema de corte de estoque. Na base inicial, as colunas de estoque são permutadas para as primeiras posições e as de padrões de corte para as últimas posições da matriz.

Ao entrar uma coluna na base, verifica-se se a mesma é de estoque ou um padrão de corte. A busca da posição em que a coluna será inserida na base é realizada somente nas posições das colunas de estoque ou de padrão de corte. A coluna será inserida na posição correta na base de acordo com o número de elementos não nulos. Os índices das colunas de estoque e de corte são atualizados e o procedimento é repetido.

5.6 Problema de Corte de Estoque Multiperíodo

O problema de corte multiperíodo foi motivado pelo estudo de problemas acoplados encontrados na prática. Neste problema, as demandas dos itens ocorrem em períodos diversos de um horizonte de planejamento finito, sendo possível antecipar ou não a produção de itens. Os objetos não utilizados em um período ficam disponíveis no próximo, juntamente com possíves novos objetos adquiridos ou produzidos pela própria empresa. A seguir, apresenta-se um modelo de otimização linear inteira de grande porte, cujo objetivo pondera as perdas nos cortes e os custos de estoque de objetos e itens:

$$
\begin{aligned}
\text{minimizar} & \sum_{t=1}^{T}\left(\sum_{j=1}^{N_1} c_{j1t}x_{j1t} + \sum_{j=1}^{N_2} c_{j2t}x_{j2t} + \ldots + \sum_{j=1}^{N_K} c_{jKt}x_{jKt} + \sum_{i=1}^{m} cr_{it}r_{it} + \sum_{k=1}^{K} cs_{kt}s_{kt}\right) \\
\text{sujeito a} & \sum_{j=1}^{N_1} a_{j1t}x_{j1t} + \sum_{j=1}^{N_2} a_{j2t}x_{j2t} + \ldots + \sum_{j=1}^{N_K} a_{jKt}x_{jKt} + r_{t-1} - r_t = d_t, \quad t = 1,\ldots,T \\
& \sum_{k=1}^{T}\sum_{j=1}^{N_k} x_{jkt} - s_{t-1} + s_t = e_t, \qquad\qquad t = 1,\ldots,T \quad (5.3) \\
& x_{jkt} \geq 0, \text{ inteiros}, r_{it} \geq 0, \qquad\qquad j = 1,\ldots,N_k, k = 1,\ldots,K, t = 1,\ldots,T.
\end{aligned}
$$

Índices:

- $t = 1,\ldots,T$: número de períodos no horizonte de planejamento;
- $k = 1,\ldots,K$: número de tipos (comprimentos) de objetos disponíveis em estoque;
- $j = 1,\ldots,N_k$: o número de padrões de corte para o objeto k, $k = 1,\ldots,K$;
- $i = 1,\ldots,m$: número de tipos de itens a serem cortados.

Dados:

- L_k : comprimento do objeto tipo k, $k = 1,\ldots,K$;
- e_{kt}: quantidade adquirida/produzida de objetos tipo k no período t, $k = 1,\ldots,K, t = 1,\ldots,T$ (e_t: vetor de dimensão K cujas componentes são e_{kt});
- l_i: comprimento do item tipo i, $i = 1,\ldots,m$;
- d_{it}: demanda do item tipo i no período t, $i = 1,\ldots,m$, $t = 1,\ldots,T$ (d_t: vetor de dimensão m, cujas componentes são d_{it});
- α_{ijkt}: número de itens do tipo i no j-ésimo padrão de corte para o objeto tipo k, no período t (a_{jkt}: vetor de dimensão m, cujas componentes são α_{ijkt}).

Parâmetros:

- c_{jkt}: custo de cortar o objeto tipo k segundo o j-ésimo padrão de corte no período t, $j = 1,\ldots,N_k, k = 1,\ldots,K, t = 1,\ldots,T$ (tipicamente proporcional à perda de material);

- cr_{it}: custo de estocar o item tipo i no final do período t, $i = 1, \ldots, m$, $t = 1, \ldots, T$;

- cs_{kt}: custo de estocar o objeto tipo k no final do período t, $k = 1, \ldots, K$, $t = 1, \ldots, T$;

Variáveis de decisão:

- x_{jkt}: número de objetos tipo k cortados conforme o padrão de corte j, no período t;

- r_{it}: número de itens do tipo i que são antecipados para o período t (r_t: vetor de dimensão m, cujas componentes são r_{it});

- s_{kt}: número de objetos do tipo k que sobram no final do período t.

Observações:

1- Sem perda de generalidade, pode ser considerado os estoques iniciais nulos, $r_{i0} = 0$, $i = 1, \ldots, m$ e $s_{k0} = 0$, $k = 1, \ldots, K$;

2- O parâmetro custo associado a um padrão de corte é $c_{jkt} = \gamma \left(L_k - \sum_{i=1}^{m} l_i \alpha_{ijkt} \right)$, ou seja, é a perda do j-ésimo padrão de corte para o objeto tipo k, no período t, multiplicado pelo custo unitário de material perdido, γ;

3- O parâmetro custo de adiantar a produção de itens de um período para outro é definido por: $cr_{it} = \alpha l_i$, em que α é o custo por comprimento de item estocado;

4- O parâmetro custo de estocar objetos de um período para outro é definido por $cs_{kt} = \beta L_k$, em que β é o custo por comprimento de item estocado;

5- O modelo para o problema de corte de estoque multiperíodo apresentado pode ser utilizado para resolver problemas de corte unidimensional, bidimensional, tridimensional. A diferença consiste, principalmente, na geração das colunas/padrões de corte, quando surge um problema de corte, que pode ser unidimensional, bidimensional etc. Os métodos para solucionar esses problemas encontram-se em Poldi [48];

6- O método Simplex com geração de colunas também é utilizado neste caso.

5.7 Método Simplex com Geração de Colunas e Aplicação ao Problema de Corte

O método Simplex pode ser usado na resolução de problemas de corte de estoque. A condição de integralidade sobre as variáveis x_j torna os problemas de corte (5.1) difíceis, senão impossíveis, de serem resolvidos computacionalmente. Para contornar este problema relaxa-se a condição de integralidade sobre as variáveis x_j e resolve-se o problema de otimização linear resultante pelo método Simplex.

Em problemas práticos, m (que é a quantidade de tipos de itens) é da ordem de dezenas, enquanto que n (o qual depende de m, L e $l_i, i = 1, \ldots, m$) pode ser da ordem de milhões, o que inviabiliza a resolução direta do problema. Para contornar esta dificuldade, utiliza-se o processo de geração de colunas na resolução da relaxação por otimização linear de (5.1). Para a solução do problema de corte de estoque inteiro, encontra-se em Poldi [47] alguns métodos heurísticos para arredondamento da solução.

Cada coluna representa um padrão de corte. Entretanto, durante as iterações do método Simplex, temos de determinar uma nova coluna para entrar na base. Este problema torna-se impraticável devido ao grande número de colunas que devem ser investigadas. Para contornar este problema, Gilmore e Gomory [28] propuseram uma técnica de geração de colunas que é bastante eficiente para resolver o problema.

Para descrevermos o método Simplex com geração de colunas para resolver problemas de corte de estoque, vamos considerar, por simplicidade, o problema de corte de estoque com um único tipo de objeto disponível em estoque para ser cortado e a função objetivo é a minimização do número total de objetos cortados. O modelo matemático a ser considerado é o seguinte

$$\begin{aligned} \min & \sum_{j=1}^{n} x_j \\ \text{s.a} & \sum_{j=1}^{n} \alpha_{ij} x_j = d_i, i = 1, \ldots, m \\ & x_j \geq \mathbf{0}, \text{inteiro}, j = 1, \ldots, n. \end{aligned}$$

O procedimento consiste em gerar uma coluna k, isto é, um novo padrão de corte, utilizando o critério de Dantzig, que procura a variável x_k com o menor custo relativo, o que sugere o seguinte

sub-problema

$$c_k - y^t\alpha_k = \text{min } \{c_j - y^t\alpha_j,\ j = 1, 2, \ldots\},$$

onde y é o vetor "multiplicador Simplex" de uma determinada iteração.

Se o custo relativo, $c_k - y^t\alpha_k \geq 0$ (as variáveis básicas, incluídas no cálculo do mínimo acima, têm custos relativos nulos), então a solução básica atual é ótima, caso contrário, a coluna α_k entra na base. Temos que

$$\text{min } \{c_j - y^t\alpha_j\} = 1 - \text{max } \{y^t\alpha_j\}.$$

Com isso, encontrar uma coluna não-básica, digamos $(\alpha_1, \alpha_2, \ldots, \alpha_m)^t$ que substituirá uma coluna básica, corresponde a resolver o problema

$$\begin{array}{ll} \text{max} & y_1\alpha_1 + y_2\alpha_2 + \ldots + y_m\alpha_m \\ \text{s.a} & (\alpha_1, \alpha_2, \ldots, \alpha_m)^t \text{ corresponda a um padrão de corte.} \end{array}$$

A coluna $\alpha_j = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ é gerada pela resolução de um problema de corte (problema da mochila, no unidimensional, sem restrições adicionais). Como a solução obtida por esse procedimento nos fornece a coluna $\alpha_j = (\alpha_1, \alpha_2, \ldots, \alpha_m)^t$ com o menor custo relativo, podemos garantir que:

- Se $c_k - y^t\alpha_k = 0$, a solução é ótima (o menor custo relativo nunca será positivo, pois os custos relativos dos padrões de corte básicos são nulos);
- Se $c_k - y^t\alpha_k < 0$, a coluna $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ entra na base (a coluna a sair da base segue os passos usuais do método Simplex).

A seguir, apresenta-se o algoritmo do método Primal-Simplex com geração de colunas utilizado na resolução dos problemas de corte de estoque.

Método Simplex com geração de colunas

FASE I:

1- Determine uma matriz básica inicial **B** (composta por padrões de corte homogêneos, ou seja, padrões com apenas um tipo de item).

2- Encontre uma solução básica factível.

FASE II:

1- Determine a solução básica corrente: $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{d}$.

2- Determine a solução dual (vetor multiplicador Simplex): $\mathbf{y} = \mathbf{B}^{-t}\mathbf{c_B}$.

3- Calcule os custos relativos (gera uma coluna para entrar na base, α, com custo $\hat{c}$):

$$\begin{aligned} \hat{c} = \max \quad & \hat{y_1}\alpha_1 + \hat{y_2}\alpha_2 + \ldots + \hat{y_m}\alpha_m \\ \text{s.a} \quad & (\alpha_1, \alpha_2, \ldots, \alpha_m)^t \text{ vetor associado a um padrão de corte.} \end{aligned}$$

onde $\hat{y_i}$ é o valor de utilidade do item tipo i, e é uma função do vetor multiplicador y calculado no passo 2.

4- Teste de otimalidade: Se $\hat{c} \geq 0$, a solução atual é ótima. Senão,

4.1- Determine π, as coordenadas básicas da direção Simplex: $\pi = -\mathbf{B}^{-1}\alpha$.

4.2- Determine o tamanho do passo:

$$\epsilon = \min \left\{ -\frac{x_{B_i}}{\pi_i} \mid \pi_i < 0,\ i = 1, \ldots, m \right\}.$$

4.3- Atualize a base, substituindo a i-ésima coluna da matriz $\mathbf{B}$ por α (obtida no passo 3). Volte ao passo 1.

5.8 Problema da Mochila

O problema descrito a seguir é conhecido na literatura como "problema da mochila (knapsack problem)". Associando-o ao problema de corte unidimensional, temos a situação onde uma barra deve ser cortada ao longo de seu comprimento L em itens de tamanhos $l_1, l_2, \ldots, l_m$. Considerando a limitação na quantidade de itens a serem selecionados, (essa situação surge, especialmente, em problemas de corte com baixa demanda), o problema é dito "problema da mochila restrito ou limitado". Para este problema o limite máximo para cada item, vetor d, deve ser respeitado.

Variável de decisão:

- α_i : quantidade de itens do tipo i selecionados, $i = 1, \ldots, m$.

Modelagem Matemática:

$$g(a) = \text{ máximo } v_1\alpha_1 + v_2\alpha_2 + \ldots + v_m\alpha_m$$
$$\text{sujeito a: } \begin{cases} l_1\alpha_1 + l_2\alpha_2 + \ldots + l_m\alpha_m \leq L \\ 0 \leq \alpha_i \leq d_i, i = 1, \ldots, m \text{ e inteiro,} \end{cases} \tag{5.4}$$

onde:

L = comprimento da peça em estoque;

m = número de tipos de itens;

l_i = comprimento do item i, $i = 1, \ldots, m$;

v_i = valor de utilidade do item i, $i = 1, \ldots, m$;

d_i = quantidade máxima de itens do tipo i que pode ser cortado, $i = 1, \ldots, m$.

O problema da mochila surge como sub-problema na solução de um problema de corte de estoque usando-se o método Simplex. No passo 3 da Fase II do algoritmo Simplex, temos de determinar uma coluna a entrar na base, esta coluna é a solução de um problema da mochila, para o qual o valor de utilidade do item i é o $y_i + l_i$ (y é o multiplicador Simplex). O método computacional utilizado para resolução do problema da mochila foi o método de enumeração implícita (branch-and-bound), proposto por Gilmore e Gomory [29].

Capítulo 6

O Método de Atualização da Decomposição LU Proposto

6.1 Introdução

Neste capítulo, encontra-se o método de atualização da decomposição **LU** proposto, que consiste em realizar operações somente nas colunas realmente afetadas pela troca de base devido a estrutura esparsa da matriz. Neste sentido, com o objetivo de obter maior eficiência, ordena-se as colunas da base segundo os critérios de reordenamentos descritos no Capítulo 3.

Na Seção 6.3, as implementações desenvolvidas em Matlab são descritas. No primeiro caso (6.3.1), as bases são escolhidas aleatoriamente, assim como as colunas que entram e saem da base, evitando a singularidade da nova base. Para o segundo caso (6.3.2) são escolhidas sequências de bases obtidas pelo MINOS ("Modular In-Core Nonlinear Optimizations System"). MINOS é um sistema de otimização que soluciona problemas lineares e não lineares de grande porte, implementado em FORTRAN.

No entanto, com o objetivo de obter maior eficiência computacional para problemas de grande porte e comparar o método em questão com os códigos computacionais existentes que não exploram a estrutura específica da matriz, a implementação foi desenvolvida também na linguagem de programação C. Tal implementação encontra-se descrita na Seção 6.4. Na Seção 6.4.1, a simulação é realizada com as sequências de bases obtidas pelo MINOS.

Com a implementação na linguagem C foi possível realizar testes computacionais comparando-a com o GLPK ("Gnu Linear Programming Kit"), na Seção 6.4.2. O GLPK é livre e consiste num conjunto de rotinas escritas na linguagem de programação ANSI C e organizadas na forma de bibliotecas. Destina-se a solucionar problemas de otimização linear, otimização inteira mista e outros problemas relacionados. Para resolver problemas de programação linear, utiliza uma rotina baseada no método Simplex revisado com duas fases. Neste trabalho, aplica-se o método proposto no problema de corte de estoque monoperíodo, utilizando o método Simplex com geração de colunas. O MINOS e o GLPK utilizam a atualização de Bartels-Golub (Capítulo 3).

6.2 Estrutura de Dados

De acordo com Maros [41], a matriz de restrição $\mathbf{A}$ de problemas de otimização linear de grande porte é geralmente muito esparsa. As colunas possuem, na média, de 5 a 10 elementos não nulos, independentemente da dimensão da matriz. Como uma base $\mathbf{B}$ é formada por colunas de $\mathbf{A}$, ela tem esparsidade similar. Provavelmente, a única diferença entre a esparsidade das duas é que $\mathbf{A}$ geralmente tem uma estrutura de elementos não nulos, enquanto em $\mathbf{B}$ as colunas aparecem intercaladas, sem uma estrutura particular. Portanto, propomos um ordenamento inicial das colunas de $\mathbf{A}$ e que esse ordenamento se mantenha nas bases obtidas durante a resolução do método Simplex.

Esparsidade é a característica mais importante a ser explorada nos problemas de otimização linear de grande porte. O sucesso da implementação do método Simplex depende de como matrizes e vetores esparsos são armazenados e utilizados na solução do problema.

Aparentemente, não existe uma estrutura de dados superior para todos os casos. De fato, as estruturas de dados devem ser desenvolvidas considerando as operações que serão realizadas. A estrutura de dados utilizada é estática e permanece sem alteração durante os cálculos.

Para armazenar uma matriz esparsa são utilizados três vetores *col, row* e *val* (Figura 6.1)

- *col*: vetor de ponteiros do início das colunas da matriz $\mathbf{A}$, que aponta *row* e para *val*. Se temos n colunas, precisamos de $n + 1$ posições, o último apontador aponta para a primeira posição depois do último elemento da última coluna,
- *row*: vetor dos índices das linhas da matriz $\mathbf{A}$,
- *val*: vetor dos elementos não nulos da matriz $\mathbf{A}$.

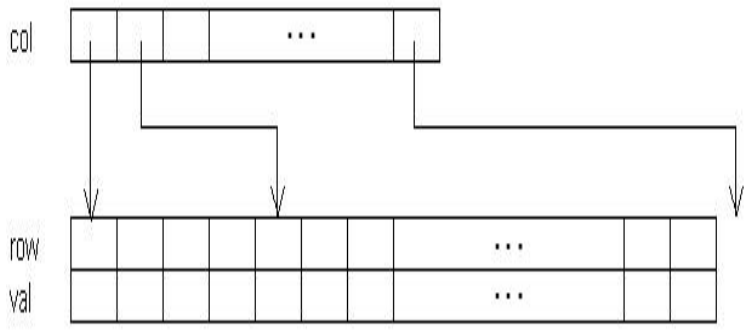

Fig. 6.1: Estrutura de dados

A matriz básica $\mathbf{B}$ é obtida por meio de um vetor de índices das colunas da matriz $\mathbf{A}$. As matrizes da decomposição $\mathbf{LU}$, $\mathbf{L}$ e $\mathbf{U}$, são armezenadas em uma única matriz, do mesmo modo da matriz $\mathbf{A}$.

Para facilitar os cálculos da atualização da decomposição $\mathbf{LU}$ e resolução dos sistemas triangulares, utiliza-se também o vetor

- *diag*: vetor de ponteiros dos elementos da diagonal da matriz $\mathbf{U}$.

6.3 Desenvolvimento do Método em Matlab

O objetivo desta implementação é simular as iterações do método Simplex utilizando uma decomposição $\mathbf{LU}$ que considera a estrutura esparsa da matriz. Inicialmente, o método de atualização da decomposição $\mathbf{LU}$ é desenvolvido em Matlab.

A simulação é realizada de duas formas distintas. No primeiro caso (6.3.1), as bases são escolhidas aleatoriamente, assim como as colunas que entram e saem da base, evitando a singularidade da nova base. Para o segundo caso (6.3.2), são escolhidas sequências de bases obtidas pelo MINOS.

Três tipos de reordenamento estático da matriz de restrições são utilizados, levando a uma decomposição esparsa sem nenhum esforço computacional para obter a ordem das colunas. Um vetor auxiliar armazena as posições antigas e atuais das colunas da matriz $\mathbf{A}$. O reordenamento equivale a multiplicar a matriz $\mathbf{A}$ à direita por uma matriz de permutação $\mathbf{P}$. Portanto, podemos recuperar e

atualizar (estaticamente) as posições das colunas na base. O número de elementos não nulos de cada um dos três tipos de reordenamentos é comparado:

R_1- As colunas são reordenadas de acordo com o número de elementos não nulos, em ordem não decrescente (Seção 4.3);

R_2- A matriz é reordenada na forma bloco triangular (Seção 4.4);

R_3- As colunas são reordenadas de acordo com a estratégia de Björck [5] (Seção 4.5).

Após escolhido um dos reordenamentos acima, quatro formas para a entrada de uma coluna na base são consideradas (Cantane e Oliveira [9]):

AC_1- A coluna entra na base respeitando um dos três tipos de reordenamentos citados anteriormente;

AC_2- A coluna entra na base na posição da coluna que sai da base;

AC_3- A coluna entra na base na última posição da matriz;

AC_{MINOS}- A coluna entra na base de acordo com o MINOS.

A decomposição **LU** da base é realizada para cada uma das atualizações citadas acima e os resultados são comparados. O comando interno do Matlab lu(.) é utilizado para realizar a decomposição **LU** tanto para escolher os três tipos de reordenamento, quanto para a atualização de colunas mais eficiente (esparsa).

O próximo passo consiste em comparar a eficiência de três decomposições **LU** distintas (Cantane et al. [11]). Neste caso, a comparação é realizada por meio do número de *flops*, ou seja, do número de operações de ponto flutuante de cada uma das decomposições:

D_1- Decomposição **LU** completa, que não aproveita a decomposição da base anterior, utilizando o comando interno lu(.) do Matlab;

D_2- Decomposição a partir da posição da coluna que entra ou sai da base, ou seja, é escolhida a menor posição na base entre a coluna que entra e e a coluna que sai s e a decomposição é realizada a partir do $min(e, s)$;

D_p- Decomposição proposta (Seção 3.5). Operação somente das colunas realmente afetadas pela troca de base devido à estrutura esparsa da matriz.

Com o objetivo de verificar a robustez do método de decomposição **LU** considerando a estrutura esparsa, é calculada a estimativa do erro introduzido pela decomposição D_p. Todas as operações da decomposição são desfeitas em todas as iterações do método Simplex. Assim, efetuando as operações na ordem inversa da decomposição **LU**, obtém-se uma matriz que simulará a atualização da decomposição da base ao ser decomposta desde a primeira coluna. Dessa forma, a base original é obtida com algum erro numérico e simula-se o pior caso em termos do erro de arredondamento, após cada iteração.

6.3.1 Simulação Aleatória

Neste caso, uma base inicial aleatória é calculada antes de iniciar a simulação para os problemas de otimização linear kb2 e beaconfd da Netlib. As colunas da matriz são reordenadas de acordo com o ordenamento R_1 (Seção 6.3), ou seja, de acordo com o número de elementos não nulos, em ordem não decrescente.

A coluna que entra e a coluna que sai da base são escolhidas aleatoriamente, evitando a singularidade da nova base. Utiliza-se as atualizações de colunas AC_1, AC_2 e AC_3 (6.3). Após a escolha da atualização de colunas mais eficiente, realiza-se as decomposições **LU** D_1, D_2 e D_p e o número de *flops* é calculado.

Por último, é calculada a estimativa do erro introduzido pela decomposição D_p. Todos os resultados desta simulação encontram-se na Seção 7.4.1.

6.3.2 Simulação com Sequências de Bases do MINOS

Nesta implementação, a simulação das iterações do método Simplex utilizando uma decomposição **LU** que considera a estrutura esparsa da matriz é realizada com as sequências de bases gerada pelo MINOS. Alguns problemas (pequenos) da Netlib são utilizados nos testes computationais, pois é inviável utilizar problemas maiores em Matlab devido ao tempo computacional.

Inicialmente, foi implementada uma função para encontrar a base inicial, a base *Crash* (Maros

[41]) (Seção 4.2), as colunas que entram e saem da base, o número de elementos não nulos das matrizes **L** e **U** e o número de *flops* da decomposição do arquivo de saída gerado pelo MINOS.

A função implementada faz a leitura dos dados do arquivo "problema.out" e, então, testes computacionais com os reordenamentos R_1, R_2 e R_3 são realizados. Depois de escolhido um dos três tipos de reordenamento, as formas de atualização da base AC_1 e AC_2 descritas na Seção 6.3 são consideradas, sendo que, na forma de atualização da base AC_1, a coluna entra na base respeitando o reordenamento escolhido. Ademais, a forma de atualização da base realizada pelo MINOS, denominada AC_{MINOS}, também foi considerada.

Novamente, depois de escolhida uma das três formas de atualização, são utilizados dois tipos de decomposição **LU** da base: a decomposição proposta D_p e a decomposição D_{MINOS}, em que é realizada uma decomposição **LU** completa da base para simular a decomposição do MINOS, e o número de *flops* é comparado.

A estimativa do erro introduzido pela decomposição D_p é calculada para todos os problemas testados. Os resultados desta simulação encontram-se na Seção 7.4.2.

6.4 Desenvolvimento do Método na Linguagem C

Com o objetivo de conseguir realizar experimentos computacionais para problemas de grande porte, as implementações foram desenvolvidas também na linguagem de programação C. Neste caso, as simulações do método Simplex que utiliza uma decomposição **LU** que considera a estrutura esparsa da matriz também foram implementadas na linguagem C.

Na Seção 6.4.1, são escolhidas as sequências de bases obtidas pelo MINOS. Assim como na Seção 6.3, três tipos de reordenamento estático da matriz de restrições são utilizados, levando a uma decomposição esparsa sem nenhum esforço computacional para obter a ordem das colunas, e compara-se o número de elementos não nulos de cada um deles.

Após escolhido um dos reordenamentos R_1, R_2 e R_3 descritos na Seção 6.3, são considerados três tipos de atualização da base, AC_1, AC_2 e AC_{MINOS}, também descritos na Seção 6.3.

Para cada uma das duas atualizações, realiza-se a decomposição **LU** da base proposta neste

trabalho descrita na Seção 3.5 e o número de elementos não nulos são comparados.

6.4.1 Simulação com Sequências de Bases do MINOS

Nesta implementação, a simulação das iterações do método Simplex utilizando uma decomposição $\mathbf{LU}$ que considera a estrutura esparsa da matriz é realizada com as sequências de bases geradas pelo MINOS. A maioria dos problemas da Netlib (os não canalizados) é utilizada nos testes computationais.

Como na Seção 6.3.2, foi implementada uma função para encontrar a base inicial, as colunas que entram e saem da base, o número de elementos não nulos das matrizes $\mathbf{L}$ e $\mathbf{U}$ da decomposição da matriz fornecida pelo arquivo de saída gerado pelo MINOS. Esta função, faz a leitura dos dados do arquivo problema.out e então realiza testes computacionais com os reordenamentos R_1, R_2 e R_3.

Depois de escolhido um dos três tipos de reordenamento, as formas de atualização da base AC_1 e AC_2 descritas na Seção 6.3 foram consideradas, sendo que, na forma de atualização da base AC_1, a coluna entra na base respeitando o reordenamento escolhido. A forma de atualização da base realizada pelo MINOS, denominada AC_{MINOS}, também foi considerada.

Para comparar o número de elementos não nulos dos reordenamentos e das atualizações de colunas, foi utilizada a decomposição $\mathbf{LU}$ da base proposta neste trabalho (descrita na Seção 3.5). O tempo computacional não foi utilizado para comparação, pois o programa desenvolvido na linguagem C não está integrado ao MINOS, implementado em FORTRAN.

6.4.2 Simulação com o GLPK

O objetivo desta simulação é comparar a implementação da atualização da decomposição $\mathbf{LU}$ proposta nesta tese com o GLPK. Esta comparação utiliza o problema de corte de estoque unidimensional (5.1) descrito no Capítulo 5.

Em Silva [57] realizou-se uma implementação do método Simplex com geração de colunas (Capítulo 2). Utiliza-se uma implementação do problema da mochila para gerar as colunas que entram na base, pois existe um número elevado de colunas correspondentes aos padrões de corte, o que proporciona uma matriz de restrições muito grande.

Este método resolve apenas a relaxação linear do problema de corte. A solução encontrada é contínua e não utilizam-se heurísticas para encontrar uma solução inteira, pois o objetivo não é encontrar tal solução, mas sim solucionar o problema de corte de estoque unidimensional utilizando a estrutura esparsa da matriz, na qual não foi considerada em Poldi [47].

A implementação realizada em Silva [57] utiliza as funções do GLPK para realizar a decomposição **LU** da base e as atualizações das colunas que entram na base no método Simplex. A implementação da atualização da decomposição **LU** proposta nesta tese foi integrada a esta implementação. Dessa forma, foi possível realizar testes computacionais utilizando as funções do GPLK e utilizando a atualização da decomposição **LU** proposta.

Utilizou-se um reordenamento específico das colunas do problema de corte (Seção 5.5) e a comparação foi realizada baseada no do tempo computacional obtido de cada uma das atualizações. Os problemas de corte utilizados nos testes computacionais foram gerados aleatoriamente. Os detalhes utilizados para gerar os dados encontram-se na Seção 7.3 e os resultados obtidos encontram-se na Seção 7.5.2.

Capítulo 7

Resultados Computacionais

7.1 Introdução

Na prática, os problemas de otimização linear geralmente são de grande porte. Nesses problemas, deve-se levar em consideração sua esparsidade, estrutura, preenchimentos e problemas numéricos. Note que o significado de "grande porte" muda constantemente; um problema considerado grande décadas atrás é pequeno hoje em dia. Enquanto a solução de problemas com centenas de restrições era uma realização nos anos 60, hoje problemas com milhares de linhas são resolvidos cotidianamente.

Neste capítulo, apresenta-se os resultados computacionais obtidos com relação ao desenvolvimento do método de atualização da decomposição $\mathbf{LU}$ (Capítulo 6). Apresenta-se também alguns dados dos problemas de otimização linear da Netlib e o gerador aleatório utilizados nos testes computacionais.

Nas implementações da Seção 7.4, utiliza-se as versões 7.1 e 5.3 do Matlab, pois o número de *flops* não é calculado na versão 7.1. Na Seção 7.5, as implementações são realizadas na linguagem C, utilizando o compilador gcc no Linux. Nos testes computacionais realizados em 7.4.1, 7.4.2 e 7.5.1, utilizam-se os problemas da Netlib. Em 7.5.2, utiliza-se a linguagem C e o Microsoft Visual C++ 2008 Express Edition no Windows XP. Nesta Seção, os problemas de corte são gerados aleatoriamente de acordo com o gerador descrito na Seção 7.3.

7.2 Problemas da Netlib

Existem diversas coleções de problemas de otimização linear. A mais conhecida está na Netlib, que pode ser obtida em http://www.netlib.org/lp/data. Alguns destes problemas são utilizados neste capítulo para comparar a eficiência computacional dos métodos implementados.

Muitos destes problemas estão na Netlib porque eles causaram dificuldades para o método Simplex na época em que surgiram. Assim, os problemas são utilizados com o objetivo de desenvolver códigos robustos. Informações mais detalhadas sobre os problemas da Netlib encontram-se em Bixby [4].

As Tabelas 7.1 e 7.2 mostram alguns dados dos problemas de otimização linear da Netlib solucionados utilizando o MINOS e adotados nos testes computacionais deste capítulo. Os dados da Tabela 7.2 foram utilizados somente nos testes computacionais em 7.5.1, pois não foi possível testá-los no Matlab devido às suas dimensões. Todos os testes computacionais referem-se somente à Fase 2.

Problema	Tamanho da Matriz	Elementos Não Nulos	Iterações sem *Crash*		Iterações com *Crash*	
			Total	Fase 2	Total	Fase 2
kb2	43×41	441	82	82	55	55
adlittle	56×97	465	107	67	116	67
sc205	205×203	552	208	208	52	52
scsd1	77×760	2388	325	223	364	314
israel	174×142	2358	199	193	277	235
bandm	305×472	2659	920	197	476	297
scfxm1	330×457	2612	525	173	408	138
beaconfd	173×262	3476	316	27	45	26
scsd6	147×1350	5666	1298	972	1138	917

Tab. 7.1: Dados dos problemas de otimização linear.

Problema	Tamanho da Matriz	Elementos Não Nulos	Iterações sem *Crash*		Iterações com *Crash*	
			Total	Fase 2	Total	Fase 2
sc50b	51×48	119	49	49	49	16
sc50a	51×48	131	50	50	14	14
sc105	106×103	281	106	106	27	27
scagr7	130×140	553	195	50	85	22
share2b	97×79	730	153	72 (infactível)	97	2
lotfi	154×308	1086	352	147	229	51
share1b	118×225	1182	326	135	189	54
scorpion	388×358	1708	386	59	156	48
scagr25	472×500	2029	1097	367	602	75
sctap1	300×480	2052	368	95	265	34
brandy	220×249	2150	666	178	443	40
agg	489×163	2541	128	69	117	67
grow7	140×301	2633	229	227	111	28
e226	224×282	2767	548	410	468	27
scrs8	491×1169	4029	1069	383	722	64
agg2	516×302	4515	195	156	153	75
agg3	516×302	4531	190	151	171	65
seba	515×1028	4874	1238	89	294	80
forplan	161×421	4916	514	185	348	55
scfxm2	660×914	5229	1154	383	729	32
grow15	300×645	5665	407	396	285	42
ship04s	402×1458	5810	419	118	180	92
bnl1	643×1175	6129	1849	284	1167	69
ganges	1309×1681	7021	1821	119	693	81
scfxm3	990×1371	7846	1562	588	1094	88
sctap2	1090×1880	8124	1791	644	809	41
ship04l	402×2118	8450	463	169	266	92
ship08s	778×2387	9501	582	126	256	89
sierra	1227×2036	9552	1051	435	1122	17
ship12s	1151×2763	10941	995	211	417	88
scsd8	397×2750	11334	4311	3011	2686	100
stockfor3	16675×15695	74004	*	*	*	*
maros-r7	3137×9408	151120	*	*	*	*

Tab. 7.2: Dados dos problemas de otimização linear.

Obs.: O símbolo $*$ significa que o MINOS não encontrou a solução do problema.

7.3 O Gerador Aleatório

O gerador aleatório descrito a seguir foi utilizado para gerar classes de problemas de corte de estoque. Tais classes foram usadas nos testes computacionais da Seção 7.5.2. Mais detalhes encontram-se em Poldi [47].

- *Número de barras em estoque:* $K = 50, 60, 100$ e 200;
- *Dimensão das barras em estoque:* Os valores de $L_k, k = 1, \ldots, K$ foram gerados aleatoriamente no intervalo $[1, 100]$;
- *Estoque disponível:* Os valores de $e_k, k = 1, \ldots, K$ foram gerados aleatoriamente no intervalo $[1, 100\frac{m}{2}]$;
- *Número de tipos de itens:* Foram utilizados diferentes valores de $m = 100, 150, 200$ e 300;
- *Dimensão dos itens a serem produzidos:* os comprimentos dos itens l_i foram gerados aleatoriamente nos intervalos $[v_1 L, v_2 L]$, onde v_1 e v_2 definem o menor e o maior valor que $l_i, i = 1, \ldots, m$ pode ter. Utilizou-se: $v_1 = 0,01$ e $0,2$ e $v_2 = 0,2$ e $0,8$. Combinando estes valores, gerou-se classes com problemas *pequenos* ($v_1 = 0,01$ e $v_2 = 0,2$), *médios* ($v_1 = 0,01$ e $v_2 = 0,8$) e *grandes* ($v_1 = 0,2$ e $v_2 = 0,8$).
- *Demanda:* Foram utilizados dois geradores diferentes para os valores da demanda dos itens; o primeiro com valores próximos a 10, ($d_i \in [7, 12], i = 1, \ldots, m$) e o segundo com valores próximos a 100, ($d_i \in [70, 120], i = 1, \ldots, m$).

Foram criadas 13 classes de problemas combinando (K, m, l_i) e gerados 20 exemplos para cada classe.

7.4 Resultados Computacionais Utilizando Matlab

A seguir, encontram-se os resultados computacionais da simulação aleatória apresentada em 6.3.1 e da simulação com sequências de bases do MINOS apresentada na Seção 6.3.2, implementadas em Matlab.

7.4.1 Simulação Aleatória

Inicialmente, foram realizados experimentos numéricos para verificar a eficiência das três formas de atualização das colunas na base. Nas Tabelas 7.3 e 7.4, encontram-se os números de elementos não nulos da decomposição da base, obtidos para os problemas kb2 e beaconfd da Netlib.

iteração	AC_1			AC_2			AC_3		
	Máx	Min	Med	Máx	Min	Med	Máx	Min	Med
1000	374	148	274	615	155	405	610	146	458
5000	409	139	285	735	139	445	708	144	476
10000	416	142	285	710	149	446	774	137	479

Tab. 7.3: Número de elementos não nulos da base para kb2 em Matlab.

iteração	AC_1			AC_2			AC_3		
	Máx	Min	Med	Máx	Min	Med	Máx	Min	Med
1000	5133	1628	3910	8273	1627	6546	6542	1628	5631
5000	5280	1632	3872	8062	1631	6674	6997	1804	6177
10000	5279	1632	3892	8479	1631	6613	7075	1611	6096

Tab. 7.4: Número de elementos não nulos da base para beaconfd em Matlab.

Pode-se verificar que a forma AC_1 de atualização das colunas obtém uma base mais esparsa que AC_2 e AC_3. No problema kb2, utilizando AC_1 para 10000 iterações, o máximo de elementos não nulos foi de 416, enquanto utilizando AC_2 e AC_3 foram de 710 e 774 respectivamente. Agora, no problema beaconfd, para 10000 iterações, o máximo de elementos não nulos utilizando AC_1 foi de 5279 e para AC_2 e AC_3 foi de 8479 e 7075 respectivamente.

Somente o método de atualização das colunas da base AC_1 será utilizado nos experimentos a seguir (nesta seção), pois obteve uma base mais esparsa. O próximo passo é obter o número de flops, ou seja, o número de operações de ponto flutuante das decomposições D_1, D_2 e D_p mencionadas em 6.3. Os resultados obtidos encontram-se na Tabelas 7.5 e 7.6.

iteração	D_1			D_2			D_p		
	Máx	Min	Med	Máx	Min	Med	Máx	Min	Med
1000	2224	365	1091	2363	117	965	435	0	70
5000	2151	325	1103	2485	68	979	657	0	80
10000	2560	353	1123	2378	66	972	688	0	81

Tab. 7.5: Número de flops das decomposições para kb2 em Matlab.

iteração	D_1			D_2			D_p		
	Máx	Min	Med	Máx	Min	Med	Máx	Min	Med
1000	108464	5583	55252	143247	871	55928	30873	0	96
5000	101921	5579	52034	132011	1138	52336	105965	0	25782

Tab. 7.6: Número de flops das decomposições para beaconfd em Matlab.

Pode-se verificar que a decomposição D_p, proposta neste trabalho, possui, na média, um número de operações reduzido em relação às decomposições D_1 e D_2 nos testes realizados. Utilizando o problema beaconfd, maior que o problema kb2, obtém-se um ganho significativo em relação ao número de flops, chegando a, em alguns casos, não realizar operações.

Finalmente, a estimativa do erro introduzido por estas operações é verificada calculando a norma entre a base obtida a partir da função que desfaz as operações da decomposição e a base reordenada obtida diretamente da matriz de restrições. Os resultados encontram-se nas Tabelas 7.7 e 7.8.

Verifica-se que o método de atualização da base é extremamente robusto, pois mesmo realizando 10000 iterações, o erro absoluto acumulado na matriz da base no pior caso é, na média, da ordem de 10^{-13} em relação às colunas originais.

iteração	Erro	
	Máx	Med
1000	$2,2 \times 10^{-13}$	9,4 $\times 10^{-14}$
5000	$3,2 \times 10^{-13}$	9,5 $\times 10^{-14}$
10000	$2,3 \times 10^{-13}$	7,4 $\times 10^{-14}$

Tab. 7.7: Estimativa do erro de aproximação da atualização da base para kb2 em Matlab.

iteração	Erro	
	Máx	Med
500	3,6 $\times 10^{-14}$	4,1 $\times 10^{-15}$
1000	2,8 $\times 10^{-14}$	6,1 $\times 10^{-15}$
5000	7,5 $\times 10^{-14}$	5,9 $\times 10^{-15}$

Tab. 7.8: Estimativa do erro de aproximação da atualização da base para beaconfd em Matlab.

7.4.2 Simulação com Sequências de Bases do MINOS

Na Tabela 7.9 (e no seu correspondente Gráfico 7.1) verifica-se o número de elementos não nulos utilizando os três reordenamentos estáticos considerados, não utilizando a base *Crash*.

Apesar de R_1 e R_3 obterem uma base um pouco mais esparsa que R_2, escolhemos utilizar o reordenamento R_2 em todos os testes a seguir, uma vez que ele obtém melhores resultados para os maiores problemas testados.

As Tabelas 7.10 e 7.11 mostram o número de elementos não nulos da decomposição da base para as três atualizações de colunas mencionadas anteriormente, utilizando e não utilizando a base *Crash*, respectivamente. Para uma melhor visualização, encontram-se em 7.2 e 7.3 os gráficos referente às mesmas.

Nas atualizações AC_1 e AC_2 utiliza-se a decomposição D_p e na atualização AC_{MINOS} a decomposição D_{MINOS}. A forma de atualização de colunas AC_1 reduz o número de elementos não nulos em até 52% para o problema scfxm1 e 78% para o problema bandm em relação à forma de atualização AC_2 nas Tabelas 7.10 e 7.11, respectivamente. Dessa forma, AC_1 é utilizada para realizar a decomposição **LU** proposta neste trabalho.

A atualização AC_1 é um pouco mais densa que a atualização AC_{MINOS} em alguns casos, mas como pode-se verificar mais adiante, nesta abordagem não são necessárias decomposições periódicas da base. Então, provavelmente [1] ela será mais rápida que outras estratégias de atualização.

A Tabela 7.12 e o Gráfico 7.4 mostram o número de *flops* de cada decomposição apresentada. A opção de base *Crash* não foi utilizada. A decomposição D_p, proposta neste trabalho, reduz o número de flops em relação à decomposição D_{MINOS}.

A estimativa do erro introduzido por estas operações foi verificada calculando a norma entre a base obtida a partir da função que desfaz as operações da decomposição e a base reordenada obtida diretamente da matriz de restrições. Os resultados encontram-se na Tabela 7.13 e no Gráfico 7.5. Pode-se concluir que o método de atualização da decomposição proposto é robusto. O erro máximo acumulado no pior caso é da ordem de 10^{-12}, isto é, não é necessário decompor periodicamente a base como é usualmente feito nos métodos tradicionais (da ordem de 10^{-8}) devido à robustez e esparsidade considerada.

[1] ver Seção 7.5.2

Elem. Não Nulos	Máximo			Mínimo			Média		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
kb2	331	357	331	86	86	86	231	241	231
adlittle	353	399	408	267	255	272	324	348	352
sc205	1150	1273	1266	411	411	411	712	767	723
israel	1949	1994	2989	625	625	625	1594	1641	2090
scsd1	1148	859	1058	433	407	457	738	611	785
bandm	6016	5287	7613	3923	3963	4872	4853	4572	6420
scfxm1	2228	2316	2604	1774	1782	2199	2056	2067	2408
beaconfd	1471	1615	1471	1410	1442	1410	1443	1482	1442
scsd6	2527	1744	2274	945	709	1029	1674	1064	1664

Tab. 7.9: Número de elementos não nulos não utilizando a base *Crash* em Matlab.

Elem. Não Nulos	Máximo			Média		
	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
kb2	353	460	406	271	332	257
adlittle	406	501	501	348	393	367
sc205	1242	1795	897	898	1109	644
israel	2187	6116	3536	1957	4382	2173
scsd1	689	1274	907	537	881	602
bandm	5508	9674	4163	4439	6422	3076
scfxm1	2250	4721	2059	2096	4006	1692
beaconfd	1423	1845	1217	1396	1610	1194
scsd6	1618	2997	1934	1053	1968	1168

Tab. 7.10: Número de elementos não nulos utilizando a base *Crash* em Matlab.

Elem. Não Nulos	Máximo			Média		
	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
kb2	357	612	387	241	407	230
adlittle	399	697	528	348	521	374
sc205	1273	2094	1221	767	917	560
israel	1994	5840	2933	1641	4020	1796
scsd1	859	1215	1013	611	820	625
bandm	5287	22422	4173	4572	20415	3121
scfxm1	2316	4718	2090	2067	4227	1654
beaconfd	1615	3744	1485	1482	3655	1458
scsd6	1744	3644	1975	1064	2426	1194

Tab. 7.11: Número de elementos não nulos não utilizando a base *Crash* em Matlab.

Número de Flops	Máximo		Média		Total	
	D_p	D_{MINOS}	D_p	D_{MINOS}	D_p	D_{MINOS}
kb2	2289	3624	1462	1693	119873	138836
adlittle	2492	5358	2103	2646	140909	177309
sc205	7655	12486	4590	4663	954724	969842
israel	12585	30456	10010	12039	1922010	2311507
scsd1	5279	11829	3635	4346	810552	969254
bandm	41178	88358	21238	27961	4183852	5508392
scfxm1	13919	21071	12436	11979	2151433	2072447
beaconfd	9701	9948	8898	9785	240233	264184
scsd6	10298	23511	6199	8158	6025711	7930057

Tab. 7.12: Número de flops da decomposição $\mathbf{LU}$ em Matlab.

Erro	Máximo	Mínimo	Média
kb2	$1{,}7 \times 10^{-14}$	0	$1{,}0 \times 10^{-14}$
adlittle	$3{,}4 \times 10^{-16}$	$1{,}1 \times 10^{-16}$	$2{,}5 \times 10^{-16}$
scs205	$3{,}5 \times 10^{-16}$	0	$1{,}2 \times 10^{-16}$
israel	$4{,}6 \times 10^{-12}$	0	$2{,}4 \times 10^{-13}$
scsd1	$7{,}3 \times 10^{-16}$	$1{,}4 \times 10^{-16}$	$3{,}3 \times 10^{-16}$
bandm	$1{,}9 \times 10^{-13}$	$5{,}7 \times 10^{-14}$	$1{,}1 \times 10^{-14}$
scfxm1	$1{,}1 \times 10^{-14}$	$2{,}1 \times 10^{-15}$	$7{,}5 \times 10^{-15}$
beaconfd	$2{,}4 \times 10^{-16}$	$2{,}8 \times 10^{-17}$	$7{,}0 \times 10^{-17}$
scsd6	$1{,}0 \times 10^{-15}$	$1{,}4 \times 10^{-16}$	$3{,}2 \times 10^{-16}$

Tab. 7.13: Estimativa do erro da atualização da base não utilizando *Crash* em Matlab.

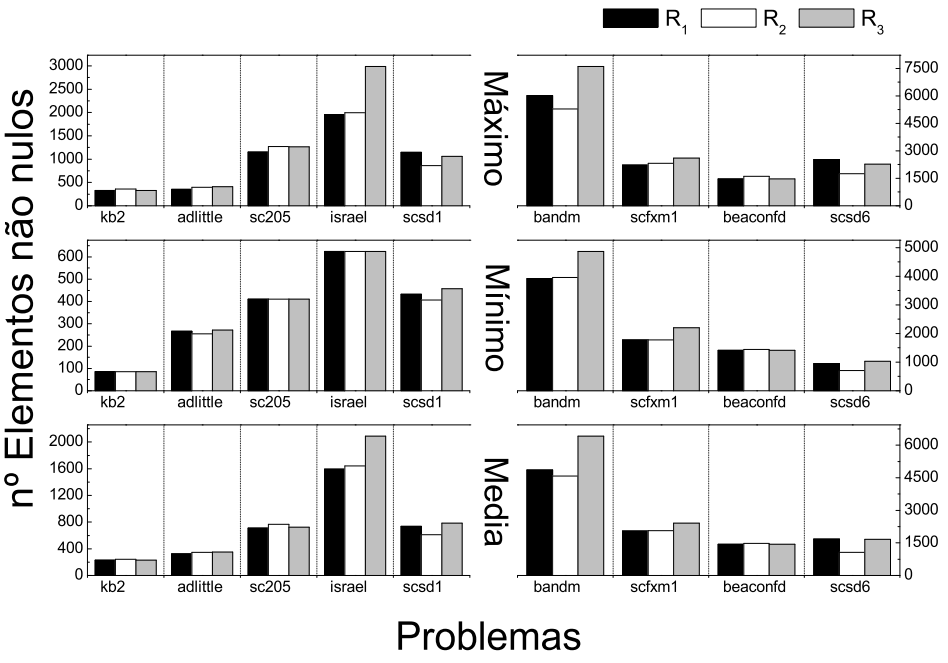

Fig. 7.1: Gráfico da Tabela 7.9

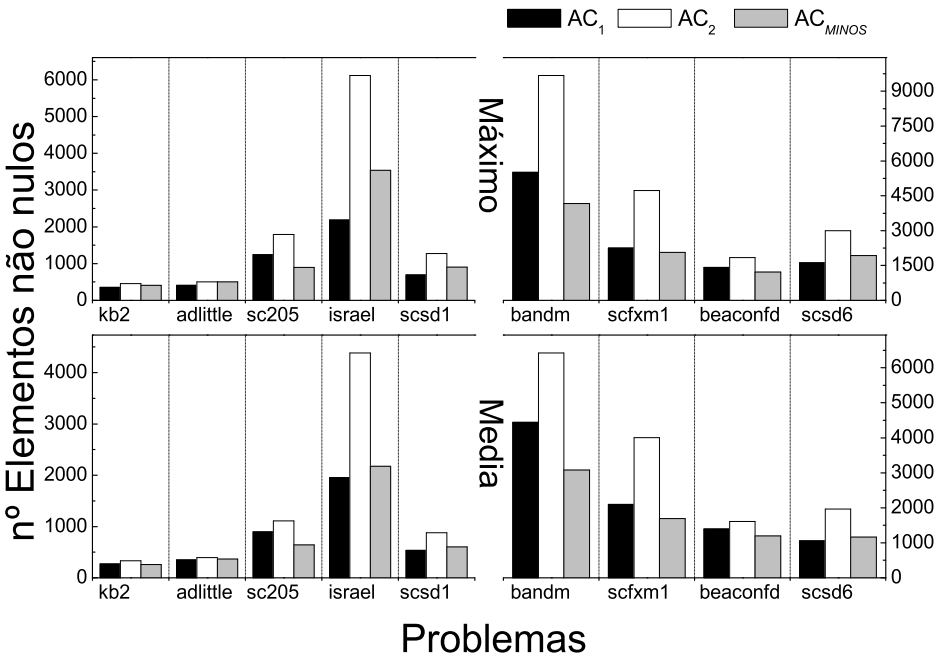

Fig. 7.2: Gráfico da Tabela 7.10

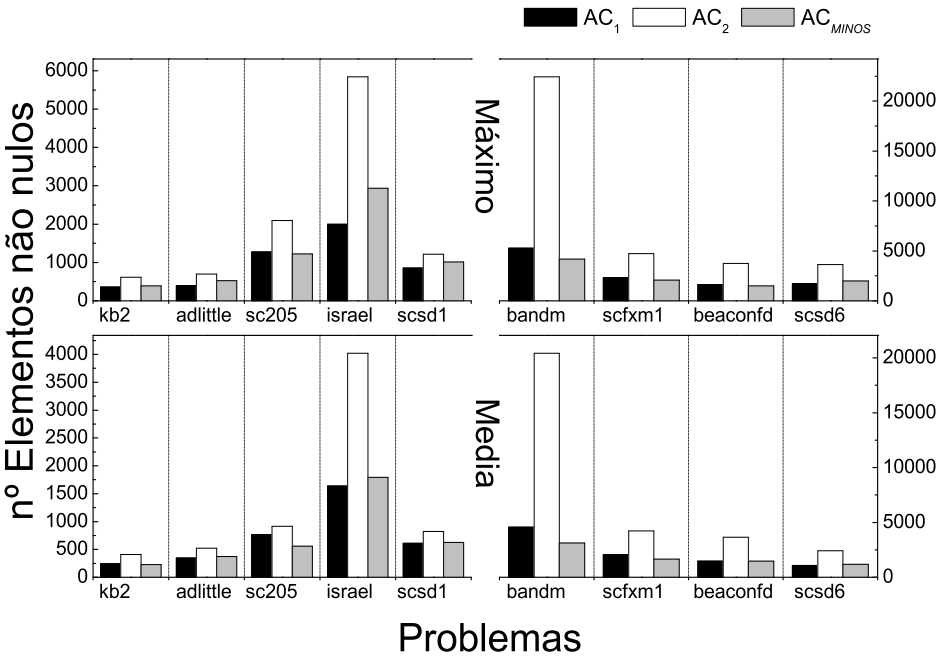

Fig. 7.3: Gráfico da Tabela 7.11

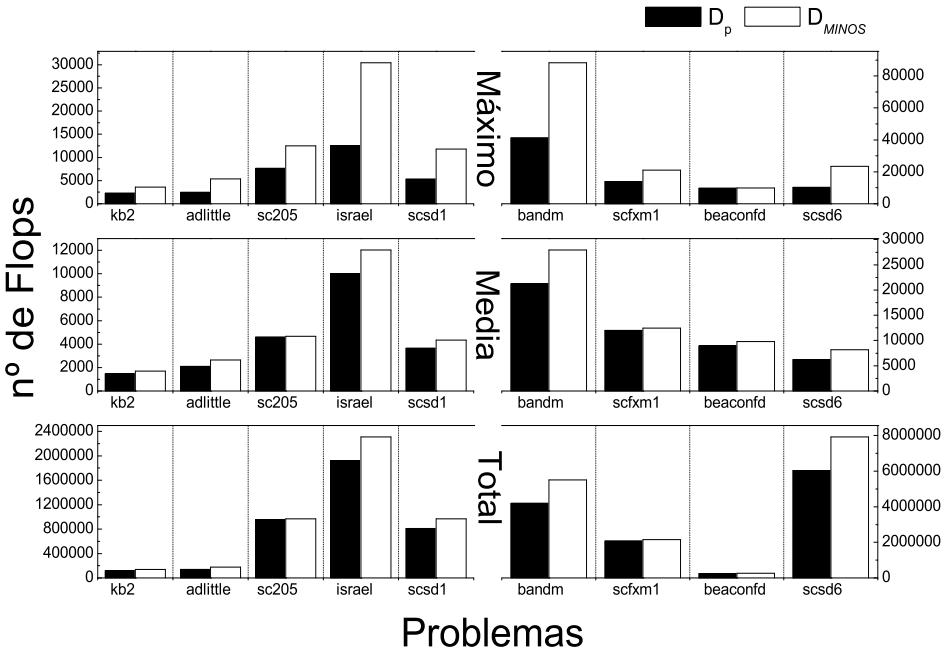

Fig. 7.4: Gráfico da Tabela 7.12

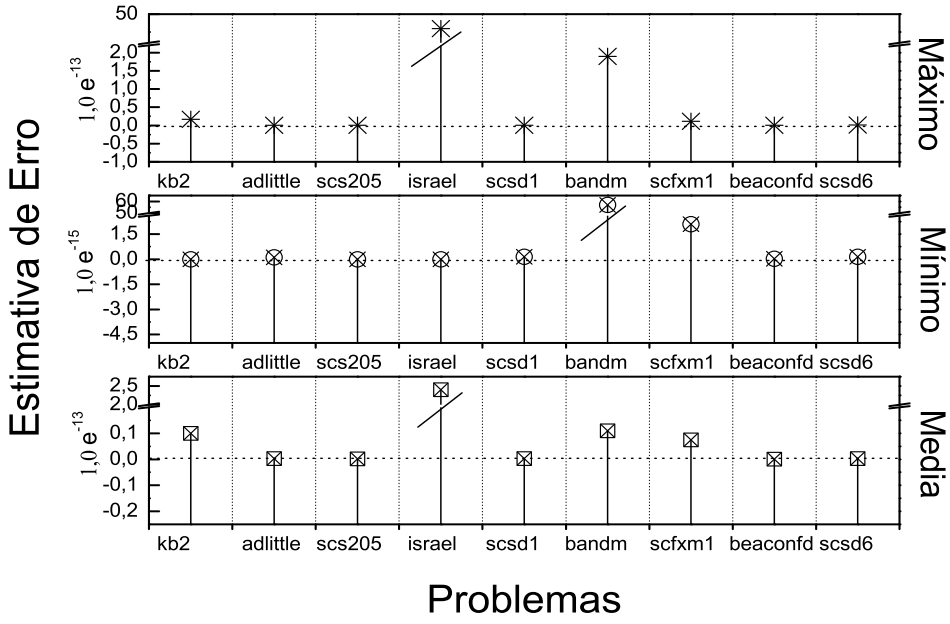

Fig. 7.5: Gráfico da Tabela 7.13

7.5 Resultados Computacionais Utilizando a Linguagem C

Os resultados computacionais da simulação com sequências de bases do MINOS apresentada na Seção 6.4.1 e da simulação com o GLPK apresentada na Seção 6.4.2, implementados na linguagem C, encontram-se a seguir.

7.5.1 Simulação com Sequências de Bases do MINOS

Nas Tabelas 7.14 e 7.15 e nos Gráficos 7.6, 7.7, 7.8 e 7.9 correspondentes, podemos verificar o número de elementos não nulos utilizando os três tipos de reordenamento estático considerados em 6.4, sem utilizar a base *Crash*.

Pode-se verificar que em apenas $20,45\%$ dos casos o reordenamento R_3 obteve uma base mais esparsa; R_1 e R_2 obtiveram uma base mais esparsa na maioria dos casos. Entretanto, para os maiores problemas testados, o reordenamento R_2 é mais eficiente. Assim, o reordenamento R_2 é utilizado nos testes computacionais a seguir.

O número de elementos não nulos da decomposição $\mathbf{LU}$ da base utilizando as três atualizações de colunas conforme 6.4 encontram-se nas Tabelas 7.16 à 7.19. Nas atualizações AC_1 e AC_2, utiliza-se a decomposição D_p, e na atualização AC_{MINOS}, a decomposição D_{MINOS} (Seção 6.3).

Agora, sem utilizar a base *Crash*, nas Tabelas 7.16 e 7.17, a atualização AC_1 proposta neste trabalho obtém uma base mais esparsa em 85% dos casos, enquanto a atualização obtida pelo MINOS obtém uma base mais esparsa em apenas $12,5\%$ dos casos. A atualização AC_2 obteve melhor esparsidade da base em apenas um caso. A atualização AC_1 proposta neste trabalho obteve, para o problema scsd8, uma redução de $48,11\%$ do número de elementos não nulos em relação à AC_2; para o problema scagr7, $43,89\%$ em relação a AC_{MINOS}.

Utilizando a base *Crash*, nas Tabelas 7.18 e 7.19, a atualização AC_1 proposta neste trabalho obtém uma base mais esparsa em $87,5\%$ dos casos, enquanto a atualização obtida pelo MINOS obtém uma base mais esparsa em apenas 10% dos casos. A atualização AC_2 obteve melhor esparsidade da base em apenas um caso, para o problema ship08s. A atualização AC_1 proposta neste trabalho obteve, para o problema brandy, uma redução de $46,43\%$ de número de elementos não nulos em relação à AC_2 e para o problema sctap1, $43,67\%$ em relação à AC_{MINOS}. Os dados obtidos das atualizações de colunas podem ser melhor visualizados nos Gráficos 7.10 à 7.15.

Elementos	Máximo			Mínimo			Média		
Não Nulos	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
sc50b	195	211	198	50	50	50	108	114	106
sc50a	193	216	204	50	50	50	109	119	110
sc105	412	444	436	105	105	105	241	254	245
kb2	266	299	263	43	43	43	176	186	176
adlittle	284	280	305	183	189	186	248	254	250
blend	555	614	634	74	74	74	354	393	383
scagr7	590	467	448	406	372	387	462	418	413
sc205	980	961	985	205	205	205	486	518	492
share2b	557	519	544	414	407	415	479	464	482
lotfi	621	711	842	512	506	537	552	577	653
share1b	838	760	929	604	580	622	700	661	729
scorpion	1793	2848	1755	1499	2494	1487	1639	2677	1593
scagr25	3276	1769	1766	1461	1371	1362	1829	1488	1471
sctap1	954	848	1060	839	764	819	881	800	876
brandy	1913	2169	2296	1585	1702	1672	1763	1937	1966
scsd1	692	652	746	287	298	305	476	442	513
israel	1654	1672	1920	355	355	355	1378	1386	1493
bandm	3246	3499	4314	2322	2684	2829	2780	3161	3492
agg	2052	2021	1880	1449	1454	1419	1811	1807	1709
grow7	3373	2379	2502	140	140	140	1767	1417	1486
scfxm1	1696	1735	1868	1342	1355	1495	1550	1579	1698
e226	1943	1881	1892	895	884	902	1477	1396	1382
beaconfd	1329	1404	1374	1234	1232	1234	1269	1274	1278

Tab. 7.14: Número de elementos não nulos dos reordenamentos em C não utilizando a base *Crash*.

Elementos	Máximo			Mínimo			Média		
Não Nulos	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
scrs8	2301	2170	2609	1912	1476	1823	2078	1695	2075
scsd6	1398	1400	1330	637	442	596	924	724	914
degen2	4797	6472	6079	3837	4532	3983	4381	5250	4738
agg2	2942	2862	2700	855	852	852	2168	2139	2053
agg3	2806	2983	2636	958	958	957	2183	2178	2097
seba	2711	2727	2722	2460	2476	2469	2589	2602	2597
forplan	1724	1507	1840	1045	1138	1244	1268	1283	1435
scfxm2	3563	3449	3768	3072	2994	3441	3254	3190	3582
grow15	7084	4969	5415	300	300	300	3350	2957	3170
ship04s	1269	1053	1232	1054	1021	1025	1088	1039	1123
bnl1	4248	5309	4146	3126	3569	3150	3529	4096	3573
fffff800	3041	3274	4946	2756	2817	3958	2884	2999	4408
ganges	5732	6788	6025	5526	5996	5811	5641	6423	5928
scfxm3	5532	5170	5763	4433	4191	4730	4922	4690	5196
sctap2	3548	4281	3499	2533	2499	2680	3114	2942	3099
grow22	9320	7259	7979	440	440	440	4521	4204	4590
ship04l	1061	1037	1083	996	984	1029	1033	1011	1045
ship08s	1860	1797	2016	1786	1746	1662	1831	1781	1817
sierra	3195	2890	3032	2845	2722	2568	3013	2780	2857
ship12s	3178	3005	3253	2972	2861	3192	3035	2935	3230
scsd8	8430	3380	3772	2435	1300	1674	3765	2069	2603

Tab. 7.15: Número de elementos não nulos dos reordenamentos em C não utilizando a base *Crash*.

Elementos	Máximo			Mínimo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
sc50b	211	282	198	50	51	51	114	127	112
sc50a	216	245	222	50	51	51	119	134	121
sc105	444	578	560	105	106	106	254	297	270
kb2	299	492	387	43	43	44	186	291	230
adlittle	280	402	528	189	228	262	254	299	374
scagr7	467	515	899	372	417	599	418	474	745
sc205	961	1409	1221	205	206	206	518	624	560
share2b	519	778	925	407	507	503	464	655	691
lotfi	711	1370	1098	506	576	496	577	971	727
share1b	760	1533	1414	580	786	633	661	1115	977
scorpion	2848	2178	2413	2494	1933	1827	2677	2056	2160
scagr25	1769	2574	3207	1371	1808	1669	1488	2132	2229
sctap1	820	1200	1345	764	955	903	800	1068	1163
brandy	2040	3288	2862	1702	2422	1426	1937	2770	2143
scsd1	652	867	1013	298	332	342	442	542	625
israel	1672	2521	2933	355	493	363	1386	1875	1796
bandm	3499	5877	4173	2684	3846	2229	3161	4728	3121
agg	2021	2715	2030	1454	1632	1449	1807	2462	1813
grow7	2406	3163	3237	140	140	141	1417	1848	1719
scfxm1	1735	2636	2090	1355	2001	1279	1579	2353	1654
e226	1881	3860	2722	884	1167	947	1396	2278	1739
beaconfd	1404	1776	1485	1232	1345	1421	1374	1582	1458

Tab. 7.16: Número de elementos não nulos das atualizações de colunas em C e da atualização do MINOS não utilizando *Crash*.

Elementos	Máximo			Mínimo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
scrs8	2170	3426	3188	1476	2640	1586	1695	3012	2328
scsd6	1400	1832	1975	442	751	615	724	1173	1194
agg2	2835	3152	2895	852	920	890	2139	2302	2136
agg3	2727	3055	2899	958	1021	989	2178	2374	2213
seba	2700	3453	3276	2476	2896	2529	2602	3060	2770
forplan	1507	2094	2751	1138	1487	1000	1283	1753	1588
scfxm2	3297	5830	4134	2994	4604	2768	3190	5208	3293
grow15	4748	6104	5966	300	300	301	2957	3449	3301
ship04s	1053	1354	1570	1021	1073	1265	1039	1208	1409
bnl1	5309	5435	3807	3569	3918	2637	4096	4580	3119
ganges	6712	7871	6148	5996	7340	5461	6423	7525	5836
scfxm3	4988	7835	5933	4191	6073	3839	4690	7002	4721
sctap2	3647	4452	5018	2499	2941	3082	2942	3761	3762
ship04l	996	1221	1318	984	1073	1260	1011	1106	1299
ship08s	1795	2170	2622	1746	1757	2104	1781	1908	2356
sierra	2890	3145	3154	2722	2829	2814	2780	2956	2961
ship12s	2969	3305	3789	2861	3103	3422	2935	3167	3541
scsd8	3264	6486	6125	1300	2374	1809	2069	3987	3164

Tab. 7.17: Número de elementos não nulos das atualizações de colunas em C e da atualização do MINOS não utilizando *Crash*.

Elementos	Máximo			Mínimo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
sc50b	223	261	182	90	96	92	134	151	126
sc50a	197	218	177	100	106	102	137	151	134
sc105	375	639	418	220	231	222	298	379	309
kb2	205	298	283	126	150	130	165	207	201
adlittle	307	290	489	219	255	386	248	270	437
scagr7	468	487	854	427	446	754	445	468	786
sc205	856	1764	892	437	459	439	614	899	625
share2b	489	675	723	482	645	710	485	660	717
lotfi	598	1343	959	519	599	613	556	819	773
share1b	670	1131	1218	598	760	884	618	923	1047
scorpion	2715	2584	2202	2096	2154	1628	2454	2354	1918
scagr25	1619	2128	2868	1491	1787	2181	1547	1944	2507
sctap1	843	1098	1549	802	1027	1373	823	1063	1461
brandy	2042	3954	2704	1695	2910	1614	1809	3377	2110
scsd1	455	658	784	371	447	383	418	560	583
israel	1702	2893	2266	1394	1500	1612	1581	2334	1904
bandm	2931	4320	3974	2054	2390	1977	2343	3083	2935
agg	2197	2701	2190	1901	2011	1898	2088	2422	2093
grow7	1790	2237	2701	1031	1178	1312	1552	1829	1980
scfxm1	1633	2721	2059	1539	2008	1524	1587	2317	1781
e226	1333	1551	2041	1238	1318	1784	1286	1422	1883
beaconfd	1143	1380	1217	1123	1149	1194	1132	1295	1206

Tab. 7.18: Número de elementos não nulos das atualizações de colunas em C e da atualização do MINOS utilizando *Crash*.

Elementos	Máximo			Mínimo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
scrs8	1982	3125	2924	1665	2354	1887	1833	2672	2354
scsd6	1184	1461	1890	924	1169	911	1052	1295	1383
agg2	2716	2812	2803	1541	1693	1601	2249	2260	2305
agg3	2695	3003	2734	1696	1806	1743	2346	2528	2369
seba	2896	3570	2990	2655	3169	2725	2800	3366	2878
forplan	1483	2440	2276	1259	1666	1739	1363	2019	2061
scfxm2	3196	4594	3833	3060	4309	3480	3131	4445	3669
grow15	4177	4694	5723	2127	2314	2841	3419	3626	4226
ship04s	1184	1381	1705	1115	1055	1386	1151	1183	1543
bnl1	5529	5678	3684	3485	4107	3047	4263	4872	3324
ganges	4016	5521	3926	3272	5174	3134	3655	5342	3529
scfxm3	4904	8195	5923	4721	7246	4501	4816	7570	5162
sctap2	3818	3866	4484	2974	3601	4030	3140	3663	4268
ship04l	1110	1865	1536	1076	1049	1380	1091	1201	1448
ship08s	2217	2318	3043	2187	2030	2766	2202	2116	2881
sierra	3641	3999	4433	3618	3899	4413	3629	3942	4424
ship12s	3466	3587	4587	3426	3512	4213	3445	3547	4522
scsd8	2707	3385	5147	2181	2937	2559	2530	3197	3763

Tab. 7.19: Número de elementos não nulos das atualizações de colunas em C e da atualização do MINOS utilizando *Crash*.

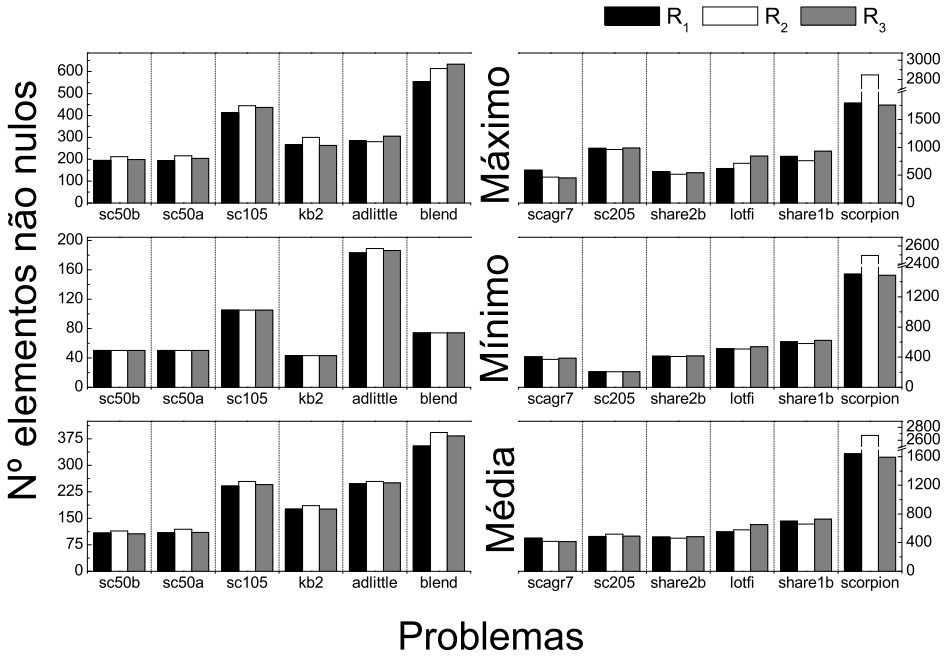

Fig. 7.6: Gráfico referente às Tabelas 7.14 e 7.15

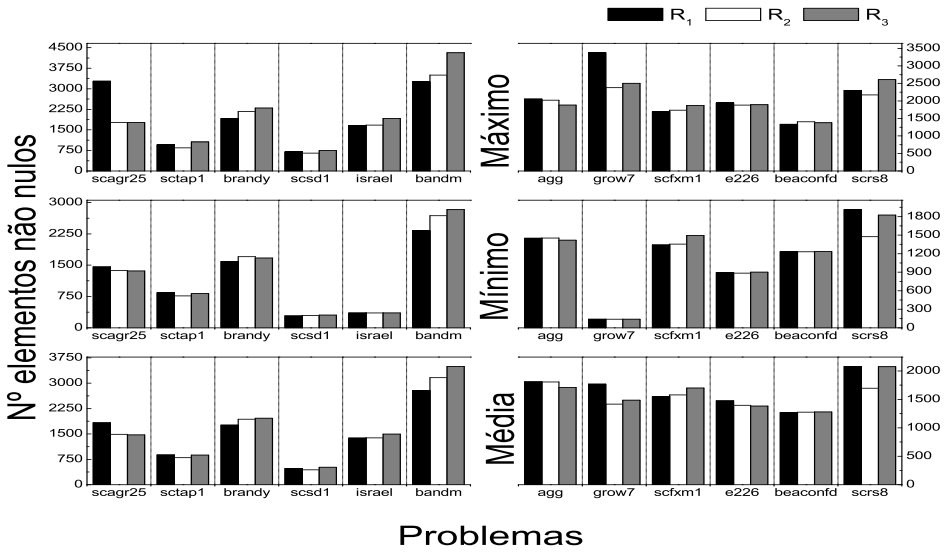

Fig. 7.7: Gráfico referente às Tabelas 7.14 e 7.15

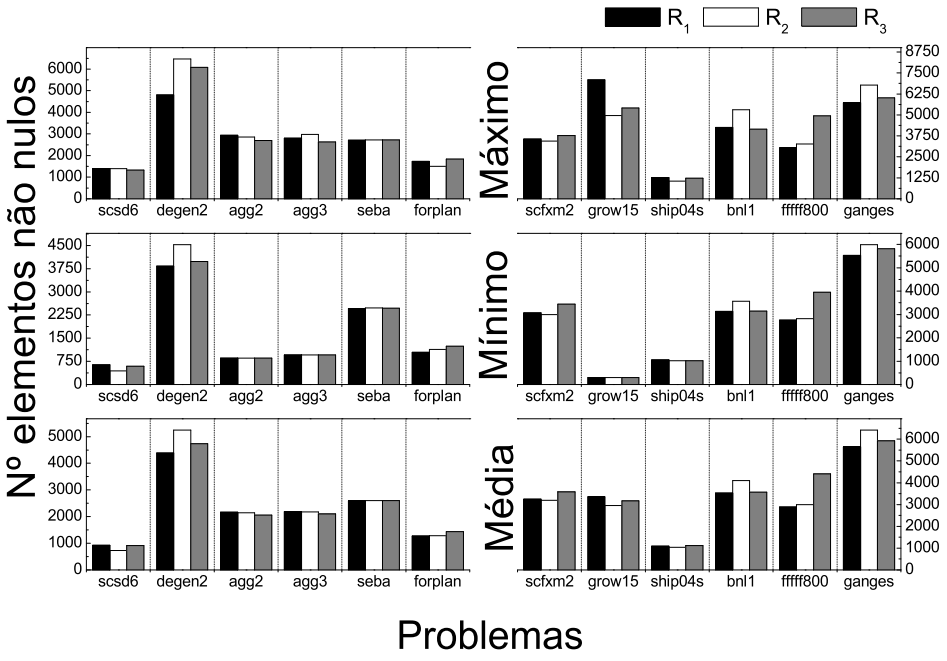

Fig. 7.8: Gráfico referente às Tabelas 7.14 e 7.15

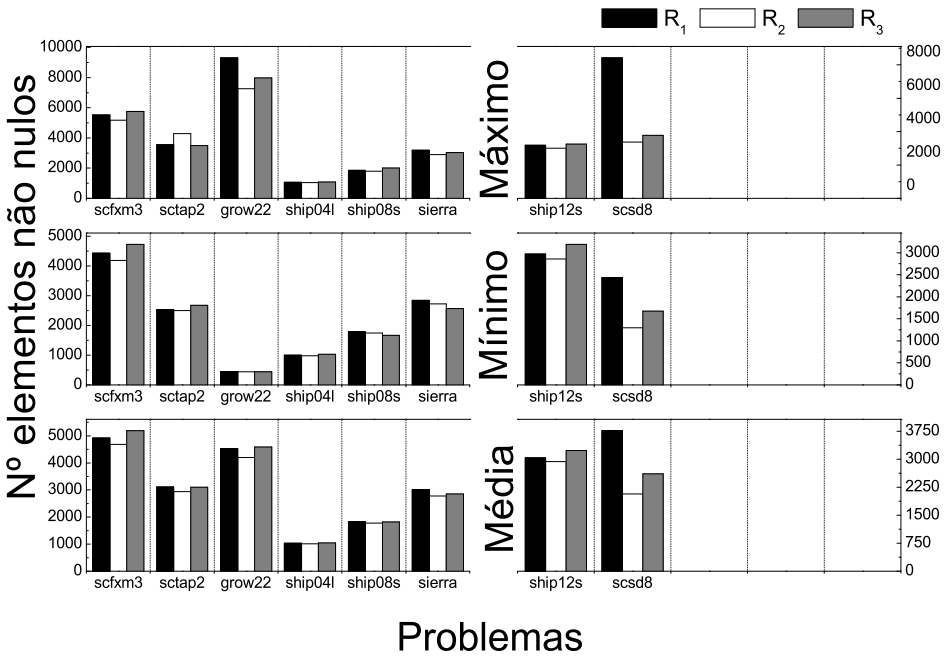

Fig. 7.9: Gráfico referente às Tabelas 7.14 e 7.15

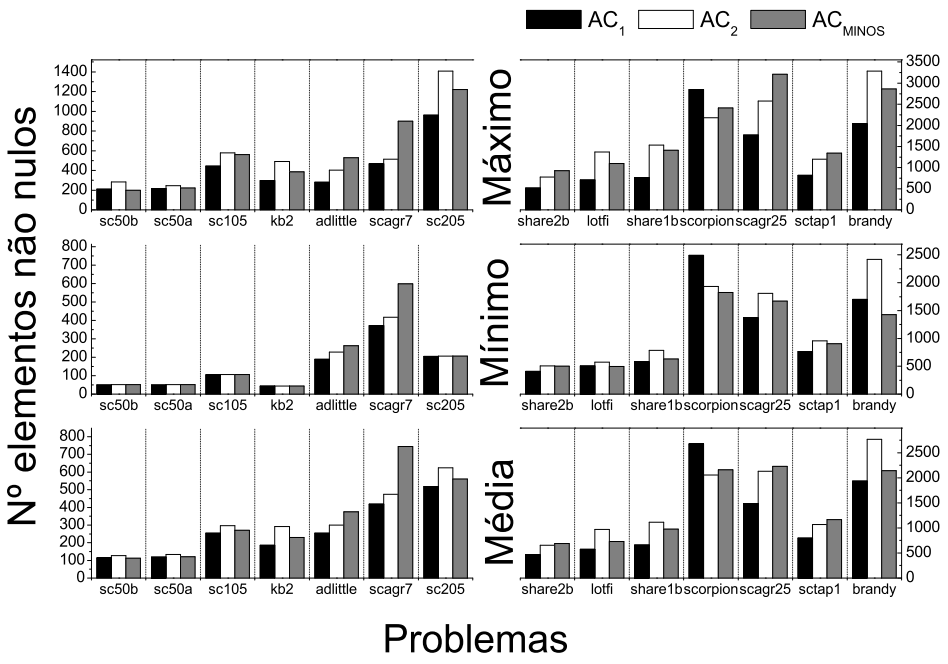

Fig. 7.10: Gráfico referente às Tabelas 7.16 e 7.17

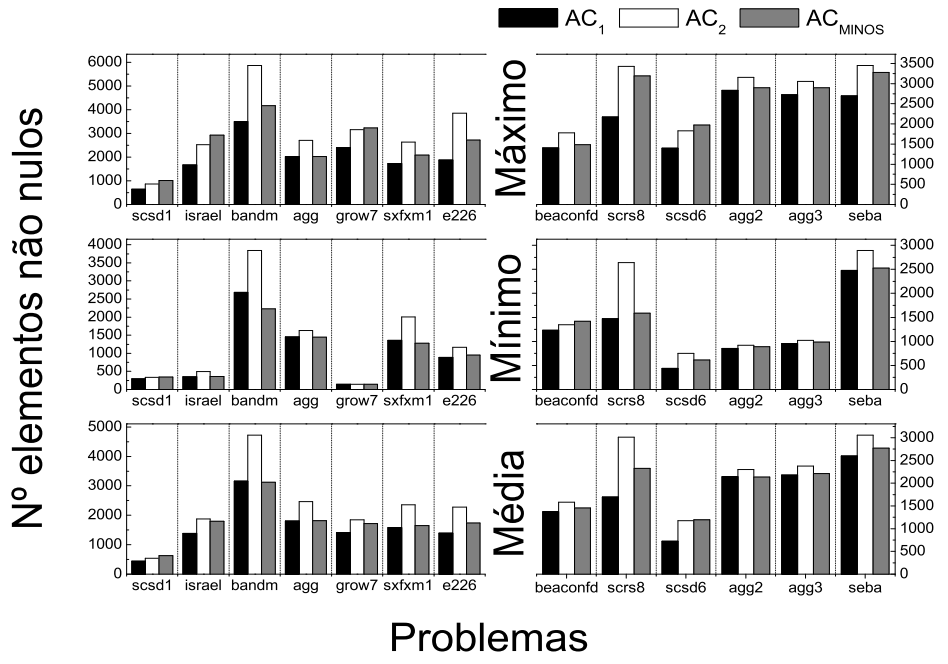

Fig. 7.11: Gráfico referente às Tabelas 7.16 e 7.17

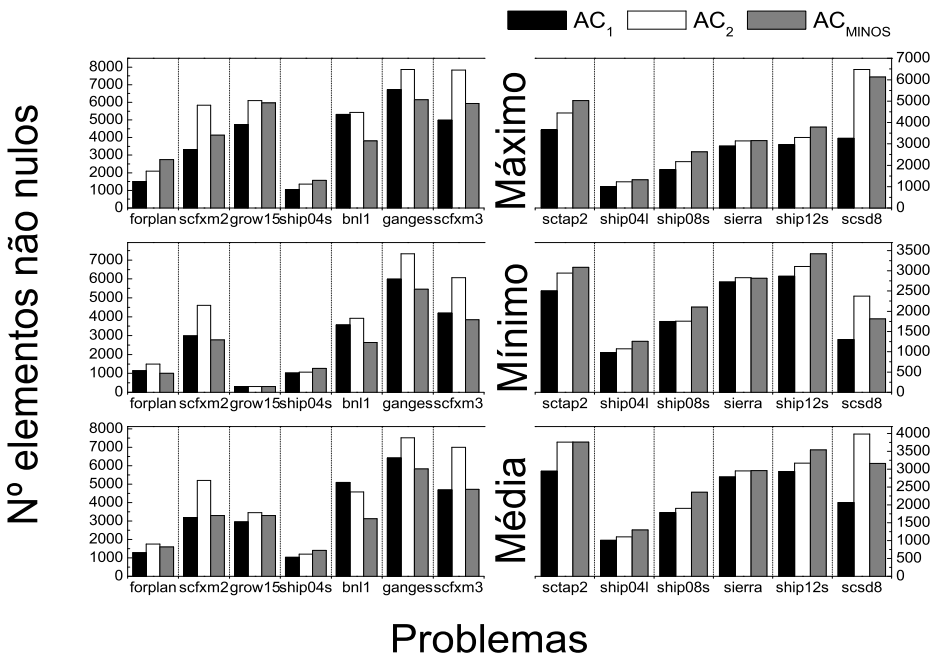

Fig. 7.12: Gráfico referente às Tabelas 7.16 e 7.17

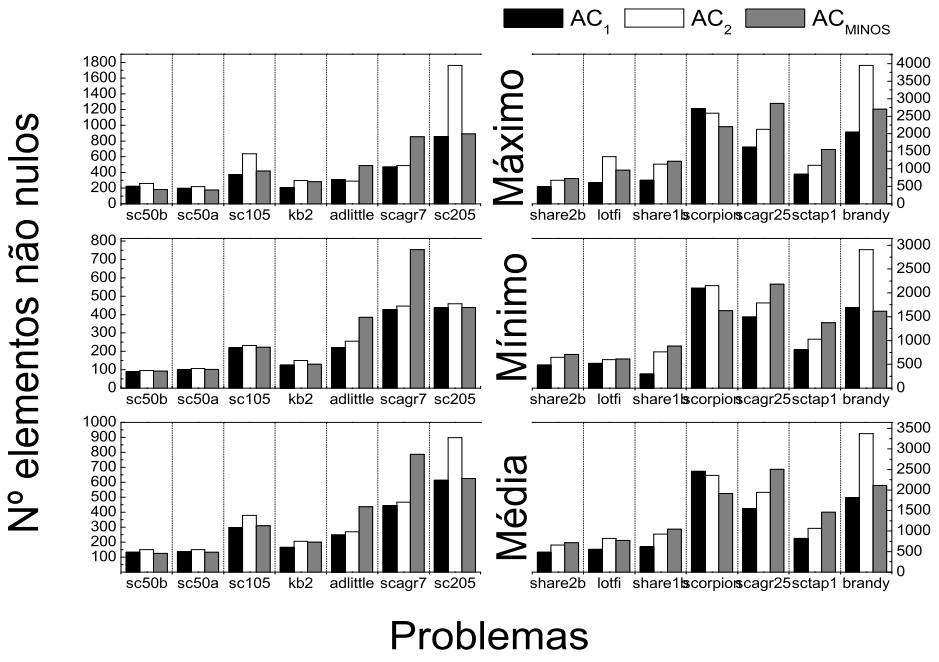

Fig. 7.13: Gráfico referente às Tabelas 7.18 e 7.19

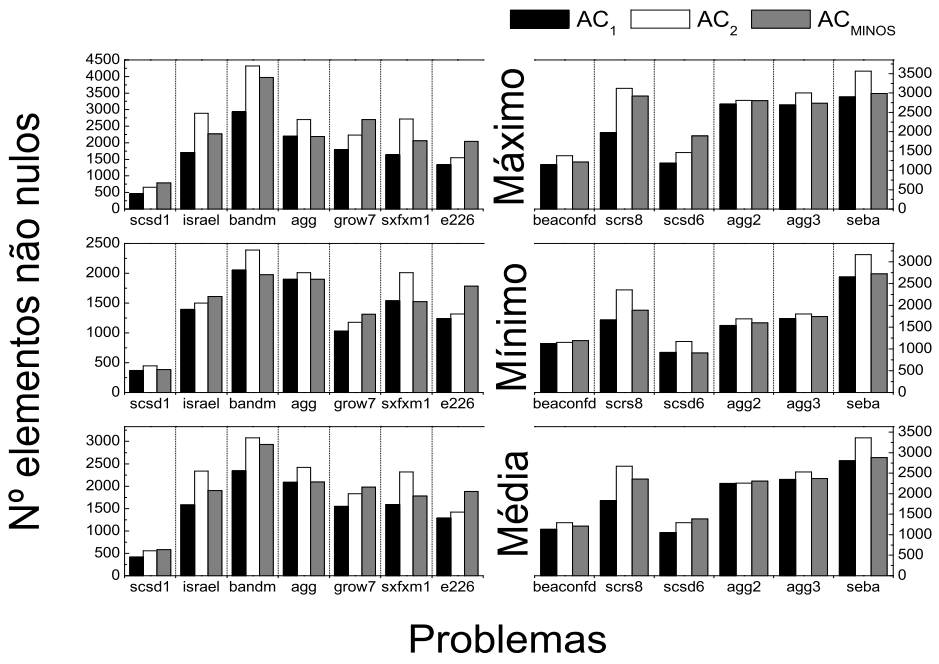

Fig. 7.14: Gráfico referente às Tabelas 7.18 e 7.19

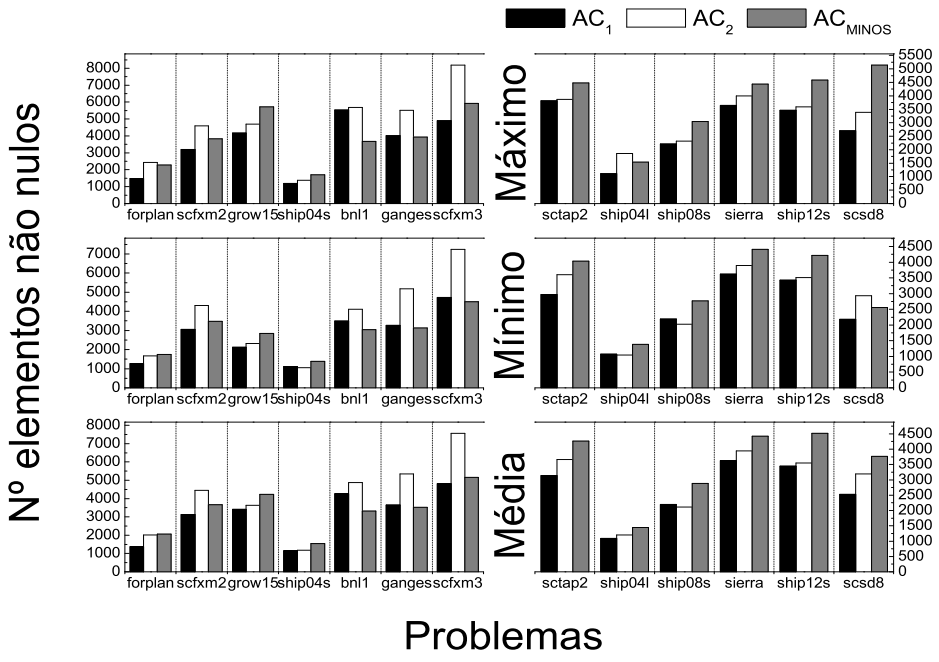

Fig. 7.15: Gráfico referente às Tabelas 7.18 e 7.19

7.5.2 Simulação com o GLPK

Utiliza-se o tempo computacional para relizar os testes nesta simulação com o objetivo de comparar a implementação da atualização da decomposição **LU** proposta com o GLPK (Seção 6.4.2). O problema de corte de estoque unidimensional, descrito no Capítulo 5, foi utilizado como uma aplicação da atualização proposta.

Classes de problemas foram geradas aleatoriamente. Mais especificamente, 13 classes com 20 exemplos em cada classe, totalizando 280 exemplos. Os critérios usados para gerar os dados encontram-se na Seção 7.3. Todos os testes computacionais foram realizados em um computador Intel com $3,4$GHz e 3Gb de memória RAM.

Pode-se verificar que a atualização proposta obteve uma redução no tempo computacional na maioria dos casos. Os melhores resultados encontram-se nas classes $C7, C10$ e $C13$, com uma redução de tempo computacional de 35%, 30% e 21%, respectivamente.

Somente nas classes $C1, C3, C5$, cujas demandas são pequenas, a atualização do GLPK obtém melhor tempo computacional que a atualização proposta. Entretanto, o tempo computacional é apenas ligeiramente menor, de quase 3% para as classes $C1$ e $C3$ e de quase $0,5\%$ para a classe $C5$.

Classe	Dados				Tempo Computacional (em segundos)	
	K	m	Itens	Demanda	Atualização Proposta	Atualização GLPK
C1	50	100	médios	pequena	86,346	83,817
C2	50	100	médios	grande	23,426	23,872
C3	50	100	grandes	pequena	16,239	15,823
C4	50	100	grandes	grande	15,681	16,152
C5	50	150	grandes	pequena	211,928	211,278
C6	50	200	médios	grande	201,190	269,705
C7	50	200	grande	grande	314,61	482,57
C8	50	300	médios	grande	2194,415	2592,531
C9	60	100	grandes	grande	64,724	67,187
C10	60	300	médios	grande	2413,172	3496,573
C11	100	100	médios	grande	25,740	25,763
C12	100	200	médios	grande	325,529	357,591
C13	200	300	médios	grande	3005,991	3816,47

Tab. 7.20: Tempo computacional da atualização de colunas proposta e do GLPK.

Capítulo 8

Conclusões e Trabalhos Futuros

Utilizando de um reordenamento estático das colunas básicas, foi construída uma base esparsa. Assim, foi obtida uma decomposição $\mathbf{LU}$ esparsa sem nenhum esforço computacional para obter a ordem das colunas devido ao reordenamento estático da base. A implementação simula as iterações do método Simplex utilizando esta decomposição $\mathbf{LU}$.

Dentre os tipos de ordenamento utilizados, o método que obtém uma matriz resultante na forma bloco triangular obteve melhores resultados, para os maiores problemas testados, em comparação ao método do mínimo grau e a estratégia de Björck. Assim, este método foi utilizado na atualização de colunas da decomposição $\mathbf{LU}$ proposta neste trabalho.

Para os testes computacionais utilizando o desenvolvimento do método proposto em Matlab, foram analisadas duas situações. Primeiramente, foi utilizado um gerador aleatório. Resultou que a forma de atualização na qual a coluna entra na base respeitando o reordenamento obtém uma base mais esparsa em comparação à forma de atualização em que a coluna entra na base na posição da coluna que sai da base. Isso ocorre também em relação à atualização na qual a coluna entra na base na última posição da matriz.

Assim, utilizando a forma de atualização nas decomposições descrita acima, verifica-se que o método de decomposição proposto neste trabalho é mais eficiente que os métodos que realizam a decomposição $\mathbf{LU}$ completa e a decomposição a partir da menor posição entre a coluna que entra e a que sai da base. Verifica-se também que o método de atualização da base é extremamente robusto pois, após 10000 iterações o erro absoluto acumulado na matriz da base no pior caso é, na média, da ordem de 10^{-13} em relação às colunas originais. Assim, para estes problemas não é preciso decompor

periodicamente a base como nos métodos tradicionais (Bartels e Golub [2]).

Na segunda situação, a simulação das iterações do método Simplex é realizada com as sequências de bases gerada pelo MINOS. Alguns problemas (pequenos) da Netlib são utilizados nos testes computationais. A forma de atualização de colunas que respeita o reordenamento reduz o número de elementos não nulos em relação à forma de atualização na qual a coluna entra no lugar da coluna que sai da base. Apesar desta atualização obter uma base um pouco mais densa que a atualização de colunas do MINOS em alguns casos, nesta abordagem não são necessárias decomposições periódicas da base. Então, ela cria a possibilidade de maior rapidez em relação à outras estratégias de atualização, como pode-se verificar na simulação com o GLPK.

O número de *flops* da decomposição proposta neste trabalho é menor que o número de *flops* em relação à decomposição do MINOS. Verifica-se também a robustez do método de atualização da base; calculando-se a norma da diferença entre os elementos da base obtida a partir da função que desfaz as operações da decomposição e a base reordenada obtida diretamente da matriz, tem-se uma média de 10^{-12}, no pior caso. Assim, como a estimativa do erro introduzido por essas operações é pequena, para estes problemas não é preciso decompor periodicamente a base como nos métodos tradicionais (Bartels e Golub [2]).

Para os testes computacionais utilizando o desenvolvimento do método proposto na linguagem C, também foram analisadas duas situações. Na primeira, a simulação das iterações do método Simplex é realizada com as sequências de bases gerada pelo MINOS, utilizando-se problemas de grande porte.

Sem utilizar a base *Crash*, a atualização de colunas que respeita o ordenamento, em conjunto com a decomposição proposta neste trabalho, obtém uma base mais esparsa em 85% dos casos, enquanto a atualização obtida pelo MINOS obtém uma base mais esparsa em apenas $12,5\%$ dos casos. A atualização de base mais simples para o método Simplex, na qual a coluna entra na base sempre na mesma posição da coluna que sai da base, obteve melhor esparsidade em apenas um caso.

Utilizando a base *Crash*, a atualização proposta neste trabalho obtém uma base mais esparsa em quase 90% dos casos, enquanto a atualização obtida pelo MINOS obtém uma base mais esparsa em apenas 10% dos casos. Novamente, a atualização na qual a coluna entra na base sempre na mesma posição da coluna que sai da base obteve melhor esparsidade em apenas um caso.

Embora o método proposto tenha aplicação em problemas de otimização linear em geral, um dos focos foi a aplicação no problema de corte de estoque monoperíodo. Neste caso, faz-se a comparação do tempo computacional com o solver livre GLPK.

Um reordenamento específico das colunas da matriz do problema de corte de estoque foi utilizado. A decomposição $\mathbf{LU}$ proposta reduziu em até 35% o tempo computacional, em comparação ao GLPK.

Os bons resultados obtidos com o trabalho realizado abre perspectiva para a continuidade de investigações na mesma linha. Em particular, pretende-se:

- Implementar a forma bloco triangular com grafos bipartidos, como apresentado na Seção 4.6;
- Integrar a decomposição $\mathbf{LU}$ esparsa ao GLPK, que ainda não explora a estrutura específica da matriz;
- Aplicar o método proposto ao problema de corte de estoque multiperíodo (Poldi [48]).

Referências Bibliográficas

[1] I. ADLER, M. RESENDE, G. VEIGA, AND N. KARMARKAR, *An implementation of Karmarkar's algorithm for linear programming*, Mathematical Programming, 44 (1989), pp. 297–335.

[2] R. BARTELS AND G. GOLUB, *The Simplex method of linear programming using the LU decomposition*, Communications of the Association for Computing Machinery, 12 (1969), pp. 266–268.

[3] M. BAZARAA, J. JARVIS, AND H. SHERALI, *Linear Programming and Network Flows*, John Wiley & Sons, 1990.

[4] R. BIXBY, *Implementing the Simplex method: the inicial basis*, ORSA Journal on Computing, 4 (1992), pp. 267–284.

[5] A. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.

[6] R. BRAYTOM, F. GUSTAVSON, AND R. WILLOUGHBY, *Some Results on Sparse Matrices*, IBM Research Centre, Yorktown Heights, N.Y., 1969.

[7] G. BRESSAN AND A. OLIVEIRA, *Fast iterations for the combined cutting-stock and lot-sizing problems*, Anais da IX International Conference on Industrial Enginneringand Operations Management, Arq. TI0601, (2003), pp. 1–8.

[8] ——, *Reordenamento eficiente das colunas básicas na programação de lotes e cortes*, Pesquisa Operacional, 4 (2004), pp. 323–337.

[9] D. CANTANE AND A. OLIVEIRA, *An efficient Simplex LU factorization update - Aceito para publicação no International Journal of Pure and Applied Mathematics*, Anais do 19th International Symposium on Mathematical Programming, 1 (2006), p. 136.

[10] ——, *Técnicas de atualização da decomposição LU da base no método Simplex*, Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional, 1 (2006).

[11] D. CANTANE, A. OLIVEIRA, AND C. LYRA, *Techniques of the Simplex base LU factorization update*, Anais do XIII Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas, 1 (2006).

[12] ——, *Reordenamento estático na atualização da decomposição LU no método Simplex*, in TEMA. Tendências em Matemática Aplicada e Computacional, 8 (2007), pp. 211–218.

[13] J. CARDENAL, I. DUFF, AND J. JIMÉNEZ, *Solution of sparse quasi-square rectangular systems by Gaussian elimination*, IMA J. Numer. Anal., 18 (1998), pp. 165–177.

[14] T. COLEMAN, A. EDENBRANDT, AND J. GILBERT, *Predictin fill for sparse orthogonal factorization*, J. Assoc. Comput. Mach., 33 (1986), pp. 517–532.

[15] G. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, 1963.

[16] G. DANTZIG, R. HARVEY, MCKNIGHT, AND S. SMITH, *Sparse Matrix Techniques in Two Mathematical Programming Codes*, Sparse matrix proceedings, Ed. R. Willoughby, RA-I, IBM Research Centre, Yorktown Heights, N.Y., 1969.

[17] T. DAVIS, *Direct Methods for Sparse Linear Systems*, SIAM, 2006.

[18] I. DUFF, *On permutations to block triangular form*, J. Inst. Maths. Applic., 19 (1977), pp. 339–342.

[19] ——, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Software, 7 (1981), pp. 315–330.

[20] I. DUFF, A. ERISMAN, AND J. REID, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

[21] A. DULMAGE AND N. MENDELSOHN, *Coverings of bipartite graphs*, Canad. J. Math., 10 (1958), pp. 517–534.

[22] ——, *A structure theory of bipartite graphs of finite exterior dimension*, Trans. Roy. Soc. Canada, 53, III (1959), pp. 1–13.

[23] ——, *Two algorithms for bipartite graphs*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 183–194.

[24] J. FORREST AND J. TOMLIN, *Updating triangular factors of the basis to maintain sparsity in the product form Simplex method*, Mathematical Programming, 2 (1972), pp. 263–278.

[25] A. FORSGREN, P. GILL, AND J. GRIFFIN, *Iterative solution of augmented system arising in interior methods*, tech. report, NA-05-03, UCSD Dept. Math., Univ. California, San Diego, August, 2005.

[26] J. GEORGE, K. IKRAMOV, AND A. KUCHEROV, *Some properties of symmetric quasi-definite matrices*, SIAM J. Matrix Anal. Appl., 4 (2000), pp. 1318–1323.

[27] P. GILL, W. MURRAY, M. SUNDERS, AND M. WRIGHT, *Maintaining LU factors of a general sparse matrix.*, Lin. Alg. App., 88/89 (1987), pp. 239–270.

[28] P. GILMORE AND R. GOMORY, *A linear programming approach to the cutting stock problem*, Operations Research, 9 (1961), pp. 848–859.

[29] ——, *A linear programming approach to the cutting stock problem - part ii*, Operations Research, 11 (1963), pp. 863–888.

[30] G. GOLUB AND C. LOAN, *Matrix Computations*, Johns Hopkins, Segunda Edição, 1989.

[31] G. GOLUB AND C. V. LOAN, *Matrix Computations*, North Oxford Academic Publishing, Baltimore, 1996.

[32] P. GUERRERO-GARCÍA, *Range-Space Methods for Sparse Linear Programs (Spanish)*, PhD thesis, Ph.D. thesis, 2002.

[33] P. GUERRERO-GARCÍA AND A. SANTOS-PALOMO, *A comparison of three sparse linear program solvers*, tech. report, MA-03-04, Dept. Appl. Math., Univ. Málaga, 2003.

[34] ——, *A non-Simplex active-set framework for basis-deficiency-allowing Simplex variations*, tech. report, MA-03-02, Dept. Appl. Math., Univ. Málaga, 2003.

[35] ——, *A sparse implementation of Lawson and Hanson's convex nnls*, tech. report, MA-05-03, Dept. Appl. Math., Univ. Málaga, 2005.

[36] F. GUSTAVSON, *Finding the block lower triangular form of a matrix*, in Sparse Matrix Computations, D.J. Rose, eds., Academic Press, New York, (1976), pp. 275–289.

[37] D. JOHNSON, A. DULMAGE, AND N. MENDELSOHN, *Connectivity and reducibility of graphs*, Canad. J. Math., 14 (1963), pp. 529–539.

[38] J. L. KENNINGTON AND R. V. HELGASON, *Algorithms for Network Programming*, Wiley, New York, 1980.

[39] D. LUENBERGER, *Linear and Nonlinear Programming*, Addison-Wesley, 1984.

[40] H. MARKOWITZ, *The elimination form of the inverse and its applications to linear programming*, Management Science, 3 (1957), pp. 255–269.

[41] I. MAROS, *Computational Techniques of the Simplex Method*, Kluwer Academic Publishers, 2003.

[42] S. MEHROTRA, *On the Implementation of a Primal-Dual Interior Point Method*, vol. 2, SIAM Journal on Optimization, 1992.

[43] W. ORCHARD-HAYS, *Advanced Linear-Programming Computing Techniques*, McGraw-Hill, 1968.

[44] P. PAN, *A basis-deficiency-allowing variation of the Simplex method for linear programming*, Computers Math. Applic., 36 (1998), pp. 33–53.

[45] ——, *A revised dual projective pivot algorithm for linear programming*, SIAM Optim., 16 (2005), pp. 49–68.

[46] A. PHOTEN AND C. FAN, *Computing the block triangular form of a sparse matrix*, ACM Trans. Math. Software, 16 (1990), pp. 303–324.

[47] K. POLDI, *Algumas extensões do problema de corte de estoque*, master's thesis, Dissertação de Mestrado, ICMC - USP, 2003.

[48] ——, *O Problema de Corte de Estoque Multiperíodo*, PhD thesis, Tese de Doutorado, ICMC - USP, 2007.

[49] K. POLDI AND M. ARENALES, *Heurísticas para o problema de corte de estoque unidimensional inteiro*, Pesquisa Operacional, 6 (2006), pp. 473–492.

[50] ——, *Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths*, Computers and Operations Research, 36 (2009), pp. 2074–2081.

[51] J. REID, *A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases*, Mathematical Programming, 24 (1982), pp. 55–69.

[52] . SANTOS-PALOMO AND P. GUERRERO-GARCÍA, *Computational Netlib experience with a dense projected gradient Sagitta method*, tech. report, MA-05-05, Dept. Appl. Math., Univ. Málaga, 2005.

[53] ——, *Updating and downdating an upper trapezoidal sparse orthogonal factorization*, IMA J. Numer. Anal., 26 (2006), pp. 1–10.

[54] A. SANTOS-PALOMO AND P. GUERRERO-GARCÍA, *Solving a sequence of sparse least squares problems*, tech. report, MA-03-03, Dept. Appl. Math., Univ. Málaga, 2003.

[55] M. SAUNDERS, *The complexity of LU updating in the Simplex method*, R.S. Anderssen and R.P. Brent, eds, *The complexity of computacional problem solving*, University Press, Queenslang, (1976), pp. 214–230.

[56] C. SILVA, *Problemas de Otimização Linear Canalizados e Esparsos*, Dissertação de Mestrado, ICMC - USP, 2002.

[57] ——, *Otimização de Processos Acoplados: Programação da Produção e Corte de Estoque*, Tese de Doutorado, ICMC - USP, 2008.

[58] U. SUHL AND L. SUHL, *A fast LU update for linear programming*, Annals of Operations Research, 43 (1993), pp. 33–47.

[59] R. TARJAN, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–159.

[60] R. VANDERBEI, *Linear Programming Foundations and Extensions*, Kluwer Academic Publishers, Boston, 1996.

Trabalhos Publicados Pelo Autor

1. D.R. Cantane and A.R.L. Oliveira, "Ordenamento estático e atualização da decomposição LU da base no método Simplex", *Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional - SBPO*, pg. 142-142, Gramado, Brasil, 2005.

2. D.R. Cantane and A.R.L. Oliveira, "An efficient Simplex LU factorization update - Aceito para publicação no International Journal of Pure and Applied Mathematics", *Anais do 19th International Symposium on Mathematical Programming - ISMP*, pg. 136-136, Rio de Janeiro, Brasil, 2006.

3. D.R. Cantane and A.R.L. Oliveira and C. Lyra, "Techniques of the Simplex base LU factorization update, *Anais do XIII Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas - CLAIO*, Montevidéu, Uruguai, 2006.

4. D.R. Cantane and A.R.L. Oliveira, "Técnicas de atualização da decomposição LU da base no método Simplex", *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional - SBPO*, Gioiânia, Brasil, 2006.

5. D.R. Cantane and A.R.L. Oliveira and C. Lyra, "Decomposição LU da Base no Método Simplex Utilizando uma Atualização Eficiente", *XII Escuela Latinoamericana de Verano en Investigación Operativa - ELAVIO*, Itaipava, Brasil, 2007.

6. D.R. Cantane and A.R.L. Oliveira, "Reordenamento estático na atualização da decomposição LU no método Simplex", *Tendências em Matemática Aplicada e Computacional - TEMA*, pg. 211-218, v. 8, 2007.

7. D.R. Cantane and A.R.L. Oliveira and C. Lyra, "Contribution of the LU Factorization Update in the Simplex Method", *XXII European Conference on Operational Research - EURO*, Book of Abstracts, pg. 202-202, Praga, República Tcheca, 2007.

8. D.R. Cantane and A.R.L. Oliveira and C. Lyra, "Block Triangular Form and Efficient Update for LU Factorization in the Simplex Method", *Optimization 2007*, Book of Abstracts, pg. 107-107, Porto, Portugal, 2007.

9. D.R. Cantane and A.R.L. Oliveira and P. Guerrero-Garcia and À. Santos-Palomo, "Towards a Static Data Structure for a Sequence of Sparse LU Factors with Partial Pivoting", *VII BRrazilian Workshop on Continuous Optimization*, Book of Abstracts, Campinas, Brasil, 2008.

Apêndice A

Solução de mínimos quadrados utilizando gradiente conjugado reduzido

A.1 Introdução

Este trabalho fez parte do estágio no exterior realizado durante o doutorado em Málaga - Espanha, em conjunto com os professores Pablo Guerrero-Garcia e Àngel Santos-Palomo. O trabalho consiste na aplicação do ordenamento de colunas e atualização da decomposição $\mathbf{LU}$ desenvolvidas nesta tese no cálculo da solução de problemas de mínimos quadrados usando gradiente reduzido. Primeiramente, encontra-se soluções de mínimos quadrados de uma sequência de problemas sobredeterminados e subdeterminados quando suas matrizes correspondentes estão longe de ser quase quadradas. Um método do gradiente reduzido é implementado, assim estas matrizes podem ser eficientemente mantidas com uma decomposição $\mathbf{LU}$ dinâmica esparsa como LUSOL ou com uma decomposição ortogonal trapezoidal estática esparsa. Uma decomposição em bloco de Saunders de uma permutação de um sistema aumentado particionado é combinada com a aplicação segura do método do gradiente conjugado para um sistema com uma matriz quase definida simétrica e um subvetor zero do lado direito. Resultados preliminares mostram a adequação do método quando os elementos da matriz de trabalho são os únicos disponíveis para armazenamento, e sua relação com problemas habituais e gradiente conjugado pré-condicionado $\mathbf{LU}$ é tratada.

A.2 Motivação

Em certas aplicações precisamos resolver uma *sequência* de problemas de minimização da norma-2

$$\begin{aligned} &\min_x \|x\|_2 \\ &\text{s.a } \mathbf{A}_k^t x = b_k, \end{aligned} \tag{A.1}$$

e problemas de mínimos quadrados linear

$$\min_x \|\mathbf{A}_k y - c\|_2, \tag{A.2}$$

onde b_k é um vetor, c é um vetor fixo, e x e y são vetores desconhecidos. A matriz $\mathbf{A}_k \in I\!R^{n \times m_k}$ é uma matriz posto coluna completo "alta e magra", com $m_k \leq n$ e $posto(\mathbf{A}_k) = m_k$. As matrizes $\mathbf{A}_k$ são subconjuntos de uma matriz posto linha completo "baixa e gorda" fixa e $\mathbf{A}_{k+1}$ é obtida adicionando / removendo colunas para / de $\mathbf{A}_k$.

Por ter sido implementado um método do gradiente reduzido, $\mathbf{A}_k$ pode ser eficientemente mantida com uma decomposição $\mathbf{LU}$ dinâmica esparsa como LUSOL (Gill et al. [27]) de $\mathbf{A}_k$

$$\mathbf{\Pi}_k \mathbf{A}_k \mathbf{P}_k = \mathbf{L}_k \mathbf{U}_k = [\mathbf{L}_1; \mathbf{L}_2] \mathbf{U}_k \tag{A.3}$$

onde a matriz triangular superior $\mathbf{U}_k$ é mantida na forma produto, ou com uma decomposição ortogonal trapezoidal estática esparsa de $\mathbf{A}_k^t$ (Santos e Guerreiro [53])

$$\mathbf{\Pi}_k \mathbf{A}_k = \mathbf{R}_k^t \mathbf{Q}_k^t = [\mathbf{R}_i^t; \mathbf{R}_d^t] \mathbf{Q}_k^t \tag{A.4}$$

onde a matriz ortogonal $\mathbf{Q}_k$ não é mantida. Na prática, a matriz anterior pode ser obtida da decomposição $\mathbf{LU}$ de $\mathbf{A}_k^t$ (Gill et al. [27])

$$\mathbf{P}_k^t \mathbf{A}_k^t \mathbf{\Pi}_k^t = \mathbf{U}_k^t [\mathbf{L}_1^t; \mathbf{L}_2^t].$$

Pode-se assumir que $\mathbf{A}_k$ está longe de ser quadrada (Cardenal et al. [13]), ou seja, que m_k não é igual a n. De agora em diante, para um melhor entendimento, trocamos o subíndice k, sinalizando o contador da iteração e omitindo a permutação das matrizes $\mathbf{P}_k$ ou $\mathbf{\Pi}_k$. Além disso, a apresentação é feita em termos da decomposição $\mathbf{LU}$, mas permanece o mesmo quando trocamos $\mathbf{L}_1$ por $\mathbf{R}_i^t$, $\mathbf{L}_2$ por $\mathbf{R}_d^t$, e $\mathbf{U}$ por $\mathbf{Q}^t$.

A motivação principal surge da implementação esparsa de métodos de conjunto-ativo não-simplex (também normalmente conhecidos como variação simplex base-deficiente, Guerrero e Santos [34]) para programação linear, que mostraram bom desempenho computacional em várias implementações do método Simplex, no caso denso (Pan [44], Guerrero e Santos [52]) e no caso esparso (Guerrero e Santos [33], Pan [45]). Embora uma decomposição estática $\mathbf{QR}$ esparsa de uma matriz pode ser usada em uma implementação de gradiente projetado (Guerrero [32], Guerrero e Santos [35, 54]), as implementações de gradiente reduzido são consideradas mais adequadas devido à esparsidade, apesar que, neste caso, a solução acima torna-se não trivial. Este é o objetivo do estudo para mostrar como podemos proceder quando soluções de mínimos quadrados são obrigatórias para o comportamento correto do programa.

Em 1998, Cardenal et al. [13] usam a decomposição em bloco de Saunders de uma permutação do sistema aumentado particionado para obter um sistema condensado (Forsgren et al. [25]) com uma matriz simétrica definida positiva de dimensão $n - m$. Eles assumem que m é quase igual a n, e portanto propõem usar uma decomposição de Cholesky densa para resolvê-lo. Em nosso caso, com m está longe de ser igual a n, devemos proceder de um modo diferente porque os programas de programação linear envolvidos tentam obter vantagem de soluções altamente degeneradas. A proposta é combinar decomposição de Saunders com a aplicação segura (Forsgren et al. [25]) do método do gradiente conjugado (Golub e Loan [31]) para um sistema (obtido do sistema condensado mencionado acima) com uma matriz simétrica semi-definida positiva (George et al. [26]) e um subvetor zero do lado direito.

A.3 O método proposto

Se $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2] = [\mathbf{L}_1; \mathbf{L}_2]U$ é uma partição de $\mathbf{A}$ determinado por suas decomposições $\mathbf{LU}$, então a matriz do sistema aumentado particionado pode ser decomposto como segue:

$$\begin{bmatrix} \mathbf{I}_m & 0 & \mathbf{A}_1 \\ 0 & \mathbf{I}_{n-m} & \mathbf{A}_2 \\ \mathbf{A}_1^t & \mathbf{A}_2^t & \mathbf{0}_m \end{bmatrix} = \begin{bmatrix} \mathbf{I}_m & \mathbf{I}_m & \\ & -\mathbf{Z} & \\ \mathbf{A}_1^t & & \mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & -\mathbf{Z}^t & \\ & \mathbf{Z}^t & \mathbf{A}_1 \\ & \mathbf{S} & \end{bmatrix},$$

onde $\mathbf{Z} = -\mathbf{L}_2\mathbf{L}_1^{-1} \in I\!R^{(n-m)\times m}$ é tal que as colunas de $[\mathbf{Z}^t; \mathbf{I}_{n-m}]$ formam uma base do espaço nulo de $\mathbf{A}^t$ e $\mathbf{S} = \mathbf{I}_{n-m} + \mathbf{Z}\mathbf{Z}^t$ é uma matriz semi-definida positiva. Seguindo Cardenal et al. [13], a decomposição bloco triangular de Saunders surge depois que são trocadas as colunas 2 e 3, linhas 2 e

3, e linhas 1 e 2:

$$\begin{bmatrix} \mathbf{A}_1^t & 0_m & \mathbf{A}_2^t \\ I_m & \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 & I_{n-m} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1^t & & \\ \mathbf{I}_m & \mathbf{I}_m & \\ & -\mathbf{Z} & \mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & & -\mathbf{Z}^t \\ & \mathbf{A}_1 & \mathbf{Z}^t \\ & & \mathbf{S} \end{bmatrix}.$$

Depois de particionar e permutar o correspondente sistema aumentado para o problema de norma-2 $(A.1)$, obtém-se

$$\begin{bmatrix} \mathbf{A}_1^t & 0_m & \mathbf{A}_2^t \\ \mathbf{I}_m & \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 & \mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} x_1 \\ -z \\ x_2 \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix},$$

onde z é o vetor multiplicador que satisfaz $x = \mathbf{A}z$. A decomposição de Saunders obtém a solução do seguinte *sistema condensado*, (Björck [5], Forsgren et al. [25]):

$$\mathbf{S}x_2 = -\mathbf{Z}w, \text{ onde } w = \mathbf{A}_1^{-t}b.$$

A solução de $(A.1)$ é dada por $[w + \mathbf{Z}^t x_2; x_2]$, onde w pode ser obtido diretamente da decomposição $\mathbf{LU}$ de $\mathbf{A}_1$ (Björck [5]) ou de

$$\mathbf{L}_1\mathbf{U}\mathbf{U}^t\mathbf{L}_1^t w = \mathbf{A}_1\mathbf{A}_1^t w = \mathbf{A}_1 b,$$

onde pode-se aproveitar o fato que $\mathbf{U}$ será ortogonal se empregarmos $(A.4)$ ao invés de $(A.3)$. Cardenal et al. [13] propuseram usar uma decomposição de Cholesky densa para obter x_2 depois de construir e decompor $\mathbf{S}$; a proposta é aplicar gradiente conjugado sem precondicionador para o sistema

$$\begin{bmatrix} \mathbf{I}_m & -\mathbf{Z}^t \\ -\mathbf{Z} & -\mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w \\ 0 \end{bmatrix}. \tag{A.5}$$

Esta aplicação é segura (Forsgren et al. [25]) porque este sistema (obtido diretamente do sistema condensado mencionado acima) tem uma matriz semi-definida simétrica e um subvetor zero no lado direito. Se z é necessário nesta aplicação, precisa-se fazer isto para resolver o sistema $\mathbf{A}_1 z = x_1$ diretamente da decomposição $\mathbf{LU}$ de $\mathbf{A}_1$ ou fazendo $z = \mathbf{A}_1^t v$, com v satisfazendo

$$\mathbf{L}_1\mathbf{U}\mathbf{U}^t\mathbf{L}_1^t v = \mathbf{A}_1\mathbf{A}_1^t v = x_1,$$

onde poderia novamente obter vantagem de uma eventual ortogonalidade de $\mathbf{U}$. Note que, ao invés de $(A.5)$, pode-se escolher aplicar gradiente conjugado sem precondicionamento para o sistema

equivalente

$$\begin{bmatrix} \mathbf{I}_{n-m} & \mathbf{Z} \\ \mathbf{Z}^t & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ -w \end{bmatrix}. \tag{A.6}$$

Agora com o problema de mínimo quadrados $(A.2)$, o sistema aumentado, permutado e particionado é

$$\begin{bmatrix} \mathbf{A}_1^t & 0_m & \mathbf{A}_2^t \\ \mathbf{I}_m & \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 & \mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} -d_1 \\ y \\ -d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ c_1 \\ c_2 \end{bmatrix},$$

onde $[d_1; d_2]$ é a projeção ortogonal de $-c$ no espaço nulo de $\mathbf{A}^t$, isto é, o oposto do resíduo de $(A.2)$. Então a decomposição de Saunders precisa resolver um outro *sistema condensado* de Saunders (Björck [5]):

$$\mathbf{S}d_2 = -(\mathbf{Z}c_1 + c_2), \text{ então } y = \mathbf{A}_1^{-1}(c_1 + \mathbf{Z}^t d_2).$$

Cardenal et al. [13] propuseram usar uma decomposição de Cholesky densa para obter d_2 depois de construir e decompor $\mathbf{S}$. Entretanto, a aplicação do gradiente conjugado sem precondicionamento no sistema (obtido diretamente do sistema condensado)

$$\begin{bmatrix} \mathbf{I}_m & -\mathbf{Z}^t \\ -\mathbf{Z} & -\mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 y \\ d_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

poderia não ser segura. Por outro lado, notando que $\mathbf{A}_1^t d_1 + \mathbf{A}_2^t d_2 = 0$ implica $d1 = -\mathbf{A}_1^{-t}\mathbf{A}_2^t d_2 = \mathbf{Z}^t d_2$, e pode-se trocar $\mathbf{Z}^t d_2$ por d_1 no cálculo de y acima para obter o sistema

$$\begin{bmatrix} \mathbf{I}_m & -\mathbf{Z}^t \\ -\mathbf{Z} & -\mathbf{I}_{n-m} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{Z}c_1 + c_2 \end{bmatrix}, \tag{A.7}$$

e então aplicar seguramente gradiente conjugado sem precondicionamento para o sistema equivalente

$$\begin{bmatrix} \mathbf{I}_{n-m} & \mathbf{Z} \\ \mathbf{Z}^t & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} d_2 \\ d_1 \end{bmatrix} = \begin{bmatrix} -(\mathbf{Z}c_1 + c_2) \\ 0 \end{bmatrix}. \tag{A.8}$$

Como no caso de norma-2 mínima, y pode ser obtido diretamente da decomposição $\mathbf{LU}$ de $\mathbf{A}_1$ ou como $\mathbf{A}_1^t u$ com u satisfazendo

$$\mathbf{L}_1\mathbf{U}\mathbf{U}^t\mathbf{L}_1^t u = \mathbf{A}_1\mathbf{A}_1^t u = c_1 + d_1,$$

onde pode-se aproveitar o fato que $\mathbf{U}$ será ortogonal se $(A.4)$ ao invés de $(A.3)$ for empregado. Refinamentos iterativos podem melhorar a precisão de u, v e w neste último caso.

A.4 Resultados comparativos preliminares

Foram realizados testes computacionais em Matlab 6.1 (com Pentium Intel 4, 2.8Ghz e 1Gb de RAM) para comprovar a eficiência do método descrito na seção anterior. O problema da Netlib STOCFOR2 com $n = 2157$ foi escolhido na forma padrão com o maior valor de n como descrito por Guerrero e Santos [35]. Os detalhes de escalonamento e ordenamento deste problema pode ser encontrado em Guerrero [32], Guerrero e Santos [33]. Para este problema, o método descrito em Guerrero e Santos [35] relizou 1205 iterações para obter uma matriz final com 878 colunas; a solução de um problema de mínimos quadrados foi realizada em cada iteração, mas também poderia ter resolvido um problema correspondente de minimizar norma-2 com um subvetor apropriado da função objetivo linear original do lado direito. As linhas dos problemas foram ordenadas de acordo com o critério do mínimo grau para fazer $(A.4)$ mais esparsa e, então, cada cálculo de mínimo quadrado foi repetido com a versão esparsa do método de Cardenal et al. [13], e o método proposto na seção $A.3$ com decomposição esparsa $\mathbf{LU}$ e decomposição ortogonal trapezoidal esparsa.

A implementação em Matlab usa o comando **qr** para calcular a decomposição ortogonal trapezoidal esparsa permutada de $\mathbf{A}^t$, e o comando **chol** para calcular a decomposição de Cholesky esparsa da matriz do sistema condensado $\mathbf{I}_{n-m} + \mathbf{L}_2(\mathbf{L}_1^{-1}(\mathbf{L}_1^{-t}\mathbf{L}_2^t))$ (através de duas soluções triangulares esparsa). No entanto, não é possível utilizar o comando lu para calcular diretamente a decomposição $\mathbf{LU}$ esparsa permutada. Para mostrar isto, considere

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 3 \\ 5 & 0 & 4 \end{bmatrix}, P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \bar{L} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 3 & 0 \\ 0 & 4 & 5 \end{bmatrix}, U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note que $\mathbf{AP} = \bar{\mathbf{L}}\mathbf{U}$ é válido, de modo que $\mathbf{\Pi AP} = \mathbf{LU}$ com $\mathbf{\Pi}^t = [e_2, e_4, e_5, e_1, e_3]$ e $\mathbf{L} = \mathbf{\Pi}\bar{\mathbf{L}}$. Mas o que se obtém com os comandos do Matlab

```
>> A=[0 0 0 0 5; 0 1 0 2 0; 0 0 0 3 4]'
>> [U,Lb,P]=lu(A'); U=U', Lb=Lb', P=P'
```

é $\mathbf{U} = \mathbf{P} = \mathbf{I}_3$ e $\bar{\mathbf{L}} = \mathbf{A}$, porque o comando lu não trabalha corretamente com matrizes retangulares "baixas e gordas". Portanto, matrizes retangulares "altas e magras"são usadas

```
>> [L,U,Pi]=lu(A); L=L*diag(diag(U)),
>> U=diag(diag(U).^(-1))*U, PiT=rem(find(Pi')'-1,size(A,1))+1
```

para obter $\mathbf{\Pi AP} = \mathbf{LU}$ com $\mathbf{\Pi}^t = [e_5, e_4, e_2, e_3, e_1], \mathbf{P} = \mathbf{I}_3$ e

$$\mathbf{L} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & -3/2 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 1 & 0 & 4/5 \\ 0 & 1 & 3/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Esta decomposição é utilizada para comparação, embora $\mathbf{U}$ não esteja na forma produto em Matlab. Como não existe uma maneira de atualizar/desfazer $\mathbf{LU}$ em Matlab, não utiliza-se técnicas de atualizar/desfazer para as outras decomposições. Não é necessário refinamento iterativo, e a solução do sistema da matriz $\mathbf{A}_1$ foi obtida diretamente no caso $\mathbf{LU}$. Foi implementado um gradiente conjugado sem precondicionamento e um produto auxiliar matriz-vetor dado por $K_1(-\mathbf{Z}^t)$ e $K_1(\mathbf{Z})$.

O vetor nulo foi usado como ponto inicial, n como o número máximo de iterações e 10^{-8} como tolerância. O comando **pcg** não funcionou porque a quantidade de escalar calculado torna-se muito pequeno ou muito grande para continuar calculando.

A técnica mais rápida foi a versão esparsa de Cardenal et al. [13], com 26 segundos, enquanto o método proposto na seção $A.3$ utilizou 219 e 1293 segundos com a decomposição esparsa $\mathbf{LU}$ e ortogonal trapezoidal, respectivamente. Entretanto, o caso $\mathbf{LU}$ é claramente competitivo se levarmos em conta o fato que o comando **chol** é compilado e a função do gradiente conjugado é interpretada; sendo assim ele pode ser considerado como uma alternativa quando o armazenamento do fator Cholesky de $\mathbf{S}$ tornar-se inviável. A grande diferença entre o desempenho dos dois procedimentos iterativos (o caso $\mathbf{LU}$ foi seis vezes mais rápido) pode ser explicado em termos da necessidade total de armazenamento (3635666 elementos não nulos no caso $\mathbf{LU}$ contra 7017897) e do número de iterações do gradiente conjugado exigidas na média (80 e 62 para $(A.1)$ e $(A.2)$ no caso $\mathbf{LU}$ contra 245 e 182). Resultados similares foram obtidos utilizando outros problemas da Netlib.

O número de iterações do gradiente conjugado necessárias para resolver os problemas $(A.1)$ e $(A.2)$ de $(A.3)$ e $(A.4)$ encontra-se na Figura $A.1$; todos os problemas de mínimos quadrados foram resolvidos com êxito. Na Figura $A.2$ é feita uma comparação do número de condição de $\mathbf{S}$, $K_1(\mathbf{Z})$ de $(A.3)$ e $K_1(\mathbf{Z})$ de $(A.4)$ com relação ao desempenho de $\mathbf{A}$. Pode-se verificar que o crescimento do mal condicionamento de $\mathbf{A}$ é quase quadrado no caso iterativo ortogonal (isto não prejudica a eficiência do resultado, mas implica na necessitade de efetuar mais que 700 iterações do gradiente

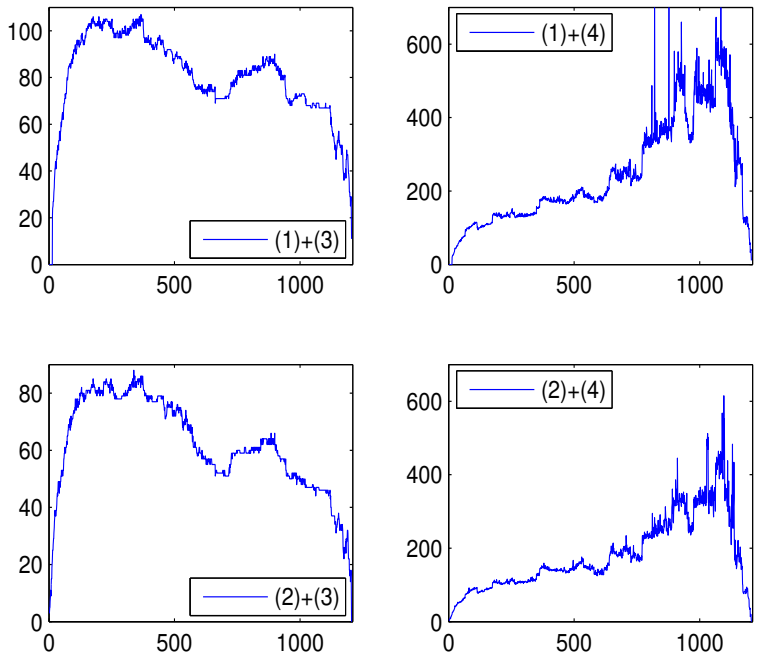

Fig. A.1: Número de iterações do gradiente conjugado para resolver A.1 e A.2 de A.3 e A.4 para stocfor2

conjugado para $k \in \{820, 877, 1084\}$), no entanto ele permanece sobre controle no caso iterativo e direto $\mathbf{LU}$ (não é maior que 30 no pior caso durante todo o processo). Resumindo, conclui-se que

- A versão esparsa da técnica direta de Cardenal et al. [13] trabalha com eficiência até mesmo quando as matrizes envolvidas estão longe de ser quase quadradas, mas necessita de armazenamento adicional;

- Quando o armazenamento do fator de Cholesky adicional torna-se inviável, o método iterativo descrito na seção $A.3$ é uma alternativa segura, ao menos em sua versão $\mathbf{LU}$ (e talvez em sua versão ortogonal também se a facilidade das técnicas de atualizar/desfazer compensar, na média, duas vezes o total de necessidade de armazenamento e três vezes as iterações necessárias do gradiente conjugado).

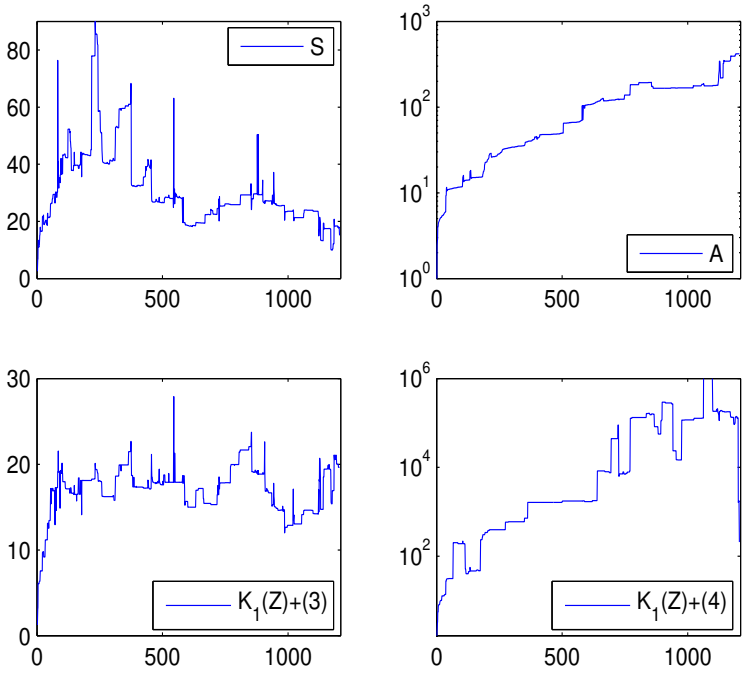

Fig. A.2: Número de condição de $\mathbf{S}$, $K_1(\mathbf{Z})$ de $(A.3)$ e $K_1(\mathbf{Z})$ de $(A.4)$ para stocfor2

A.5 Aplicação do ordenamento estático

Primeiramente, elege-se uma matriz $\mathbf{P}$ de permutação de linhas de $\mathbf{A}$

$$\mathbf{PAA}^t\mathbf{P}^t = \mathbf{R}^t\mathbf{R}, \tag{A.9}$$

onde $\mathbf{R}$ é fator de Cholesky. O próximo passo é renomear $\mathbf{A}$ como $\mathbf{PA}$ e reordenar as colunas de $\mathbf{A}$ de acordo com Cantane et al. [11].

$$\mathbf{A}\tilde{\mathbf{P}}\tilde{\mathbf{P}}^t\mathbf{A}^t = \tilde{\mathbf{A}}\tilde{\mathbf{A}}^t,$$

isto não afeta o fator de Cholesky.

Elege-se uma matriz $\mathbf{B}_k$, submatriz de $\tilde{\mathbf{A}} = \mathbf{A}\tilde{\mathbf{P}}$ de posto completo por colunas, não necessariamente quadrada. Realiza-se uma decomposição $\mathbf{LU}$ com pivotagem parcial no estilo Dolittle (Davis [17] e Cantane et al. [11]),

$$\mathbf{\Pi}_k\mathbf{B}_k = \mathbf{L}_k\mathbf{U}_k \text{ ou } \mathbf{B}_k^t\mathbf{\Pi}_k^t = \mathbf{U}_k^t\mathbf{L}_k^t. \tag{A.10}$$

Deve-se comprovar que, para qualquer $\mathbf{\Pi}_k$, $\mathbf{L}_k \subset \mathbf{R}^t$ (ou $\mathbf{L}_k^t \subset \mathbf{R}$) e $\mathbf{U}_k \subset \mathbf{R}$ (ou $\mathbf{U}_k^t \subset \mathbf{R}^t$). $\mathbf{\Pi}_k$ está definida implicitamente pela estrutura, apesar de ser determinada para garantir a estabilidade

numérica. Esse método é ilustrado por alguns exemplos numéricos pequenos a seguir.

Dada uma matriz $\mathbf{A}$, ordena-se por colunas de acordo com Cantane et al. [11], encontra-se uma base qualquer com posto completo e realiza-se as decomposições $(A.9)$ e $(A.3)$. Observamos que o fator de Cholesky de $\mathbf{B}^t\mathbf{B}$ cabe no fator de $\mathbf{A}^t\mathbf{A}$, pois a permutação $\mathbf{\Pi}$ realizada por questões numéricas não vai influir na estrutura

$$\mathbf{B}^t\mathbf{\Pi}^t\mathbf{\Pi}\mathbf{B} = \mathbf{B}^t\mathbf{B}.$$

Exemplo A.5.1 *Os gráficos nas Figuras $A.3$, $A.4$ e $A.5$ ilustram os elementos sobrepostos das matrizes $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$, não utilizando ordenamento de colunas e utilizando reordenamento de colunas pelos métodos do mínimo grau e bloco triangular, respectivamente.*

```
A_original =
     1     0     2     0     1     0    -2     0     0
     0     3     2     0     0     1     0     0     1
     1     0     0    -1     0     2     0     0     1
     0     0     0     0     2     0     1     1     0

A_minimograu =
     0     0     0     1     2     1     0    -2     0
     3     0     0     0     2     0     1     0     1
     0    -1     0     1     0     0     2     0     1
     0     0     1     0     0     2     0     1     0

A_blocotriangular =
     0     2     0     0     1     0     1    -2     0
     3     2     1     1     0     0     0     0     0
     0     0     2     1     1    -1     0     0     0
     0     0     0     0     0     0     2     1     1

B_original =
     1     2     0     1
```

```
     0     2     0     0
     1     0    -1     0
     0     0     0     2

B_minimograu =
     0     1     2     1
     0     0     2     0
    -1     1     0     0
     0     0     0     2

B_blocotriangular =
     2     1     0     1
     2     0     0     0
     0     1    -1     0
     0     0     0     2
```

Exemplo A.5.2 *Os gráficos nas Figuras* $A.6$*,* $A.7$ *e* $A.8$ *ilustram os elementos sobrepostos das matrizes* $\mathbf{L}$ *e* $\mathbf{R}^t$ *e* $\mathbf{U}$ *e* $\mathbf{R}$*, não utilizando ordenamento de colunas e utilizando reordenamento de colunas pelos métodos do mínimo grau e bloco triangular, respectivamente.*

```
A_original =
     1    -2     0     0     0     0     0     1     2
     0     0     0     3     0     1     1     0     2
     0     0     0     0    -1     2     1     1     0
     2     1     1     0     0     0     0     0     0
     0     0     0     1     1     0     0     0     0
     0     1     1     0     0     0     1     0     1

A_minimograu =
     1     0     0     0     0     0     1    -2     2
     0     0     3     0     1     1     0     0     2
     0     0     0    -1     2     0     1     0     0
     2     1     0     0     0     0     0     1     0
     0     0     1     1     0     0     0     0     0
     0     1     0     0     0     1     0     1     1
```

```
A_blocotriangular =
     1     1    -2     0     2     0     0     0     0
     0     0     0     0     2     1     3     1     0
     0     1     0     0     0     0     0     2    -1
     2     0     1     1     0     0     0     0     0
     0     0     0     0     0     0     1     0     1
     0     0     1     1     1     1     0     0     0

B_original =
    -2     0     0     0     1     2
     0     0     3     1     0     2
     0     0     0     2     1     0
     1     1     0     0     0     0
     0     0     1     0     0     0
     1     1     0     0     0     1

B_minimograu =
     0     0     0     1    -2     2
     0     3     1     0     0     2
     0     0     2     1     0     0
     1     0     0     0     1     0
     0     1     0     0     0     0
     1     0     0     0     1     1

B_blocotriangular =
     1    -2     0     2     0     0
     0     0     0     2     3     1
     1     0     0     0     0     2
     0     1     1     0     0     0
     0     0     0     0     1     0
     0     1     1     1     0     0
```

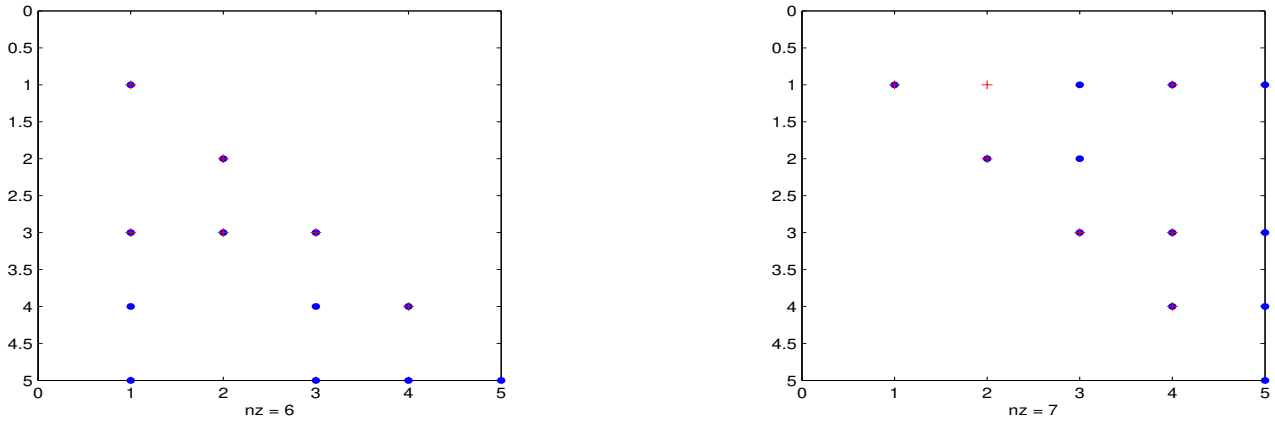

Fig. A.3: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - sem ordenamento

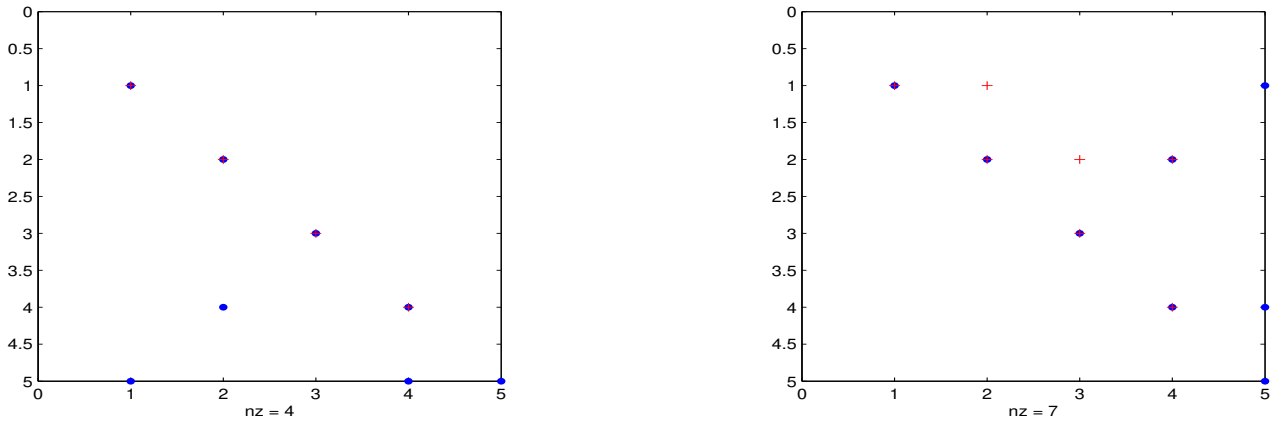

Fig. A.4: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - ordenamento mínimo grau

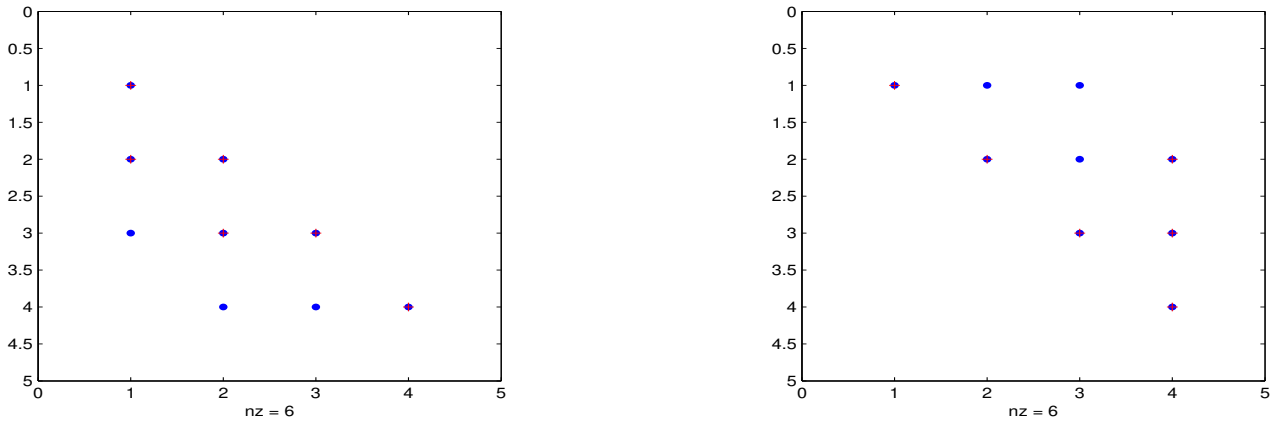

Fig. A.5: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - ordenamento bloco triangular

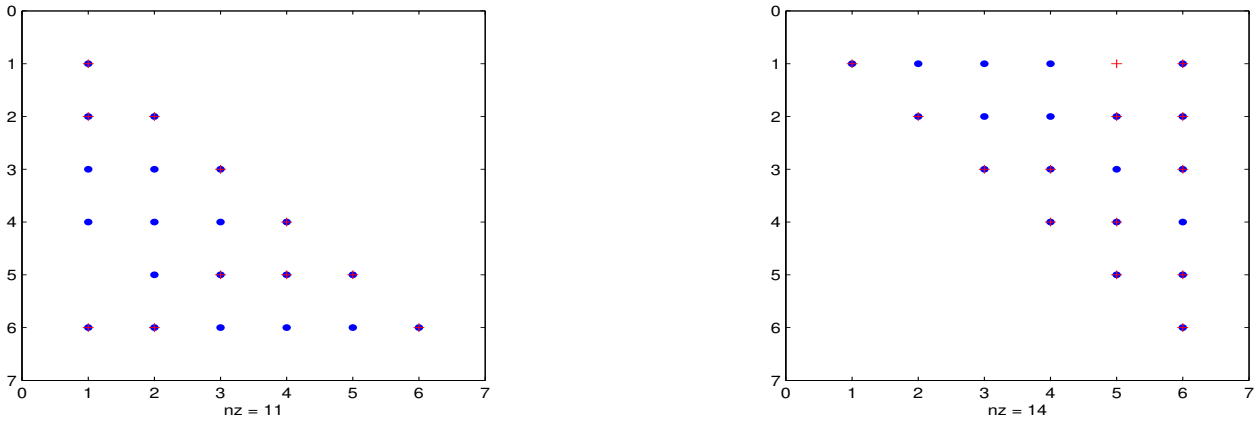

Fig. A.6: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - sem reordenamento

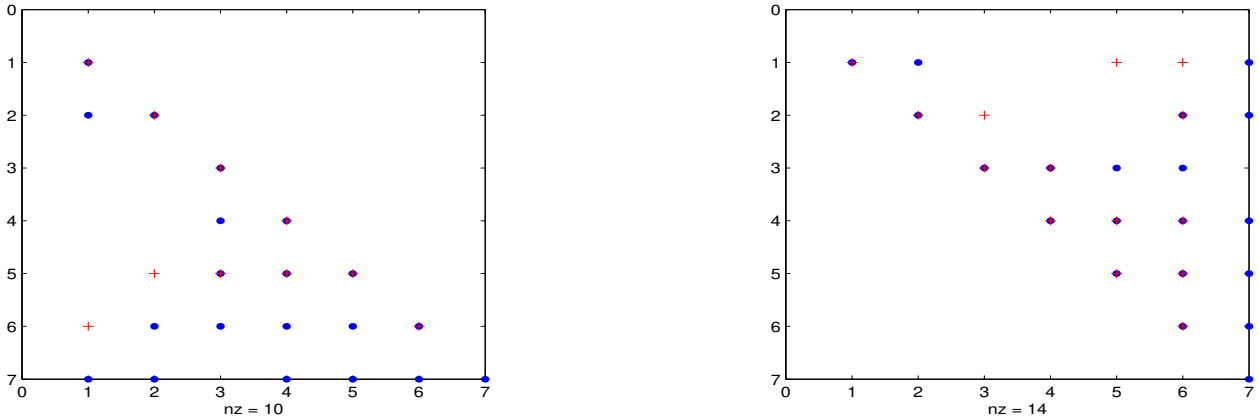

Fig. A.7: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - ordenamento mínimo grau

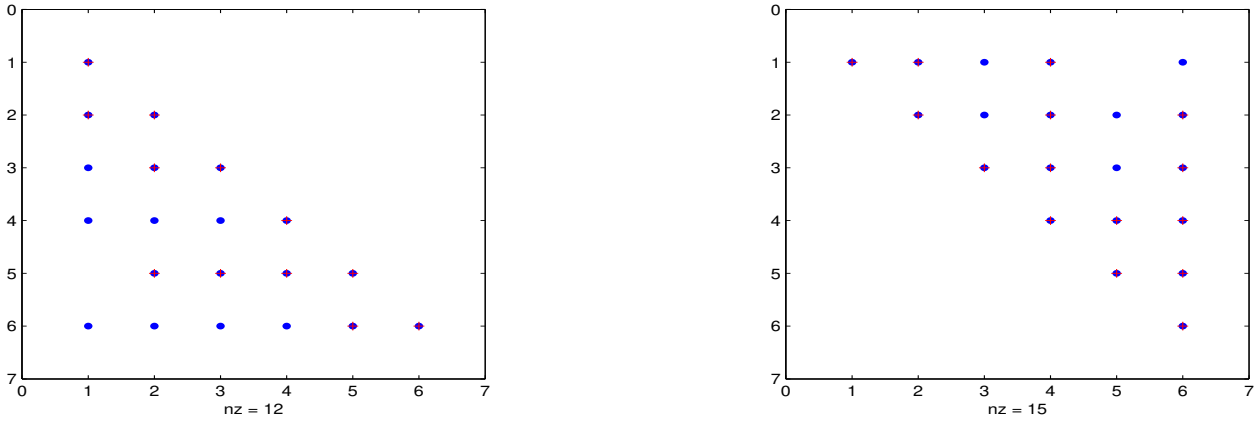

Fig. A.8: Sobreposição dos elementos de $\mathbf{L}$ e $\mathbf{R}^t$ e $\mathbf{U}$ e $\mathbf{R}$ - ordenamento bloco triangular

Os símbolos • são os fatores de Cholesky e os símbolos + são os fatores da decomposição $\mathbf{L}$ e $\mathbf{U}$. Dessa forma, não pode haver o símbolo + sem o símbolo • por baixo e observa-se esse fato somente utilizando o ordenamento bloco triangular.

Foram realizados vários exemplos e observou-se que, se as colunas são ordenadas de acordo com Cantane et al. [11], $\mathbf{L}_k \subset \mathbf{R}^t$ (ou $\mathbf{L}_k^t \subset \mathbf{R}$) e que $\mathbf{U}_k \subset \mathbf{R}$ (ou $\mathbf{U}_k^t \subset \mathbf{R}^t$).

Apêndice B

Arquivos do MINOS

Neste Apêndice, encontram-se os arquivos de entrada e saída do MINOS necessários tanto para solucionar os problemas da Netlib quanto para fornecer as informações utilizadas nos testes computacionais do Capítulo 7.

Do ponto de vista prático, problemas de otimização linear têm duas representações distintas. A versão final de um modelo criado por um sistema de modelagem ou um programa especial é apresentado de uma forma legível, geralmente em um arquivo, denominado representação externa. Este arquivo é submetido ao programa que o converte primeiro em uma representação interna.

A representação externa mais utilizada é o formato MPS (Mathematical Programming System), considerado padrão. A única característica em comum entre eles é que têm vantagens sobre esparsidade da matriz de restrições usando algumas técnicas combinadas (ver Capítulo 5 de Maros [41]).

O formato MPS foi introduzido pelo pacote de programação matemática da IBM em 1950, quando os dados de entrada do computador eram fornecidos com cartões. Desde então um grande número de modelos importantes são armazenados neste formato.

Sua filosofia básica, que é correta até hoje, é armazenar somente os elementos não nulos do problema juntamente com suas posições para reduzir a quantidade de dados de entrada. O formato MPS contém seis campos posicionados como na Tabela B.1.

Os arquivos no formato MPS são utilizados como dados de entrada para o MINOS para solucinar os problemas de otimização linear utilizados como testes no Capítulo 7. Além do formato MPS, o

NAME
ROWS
COLUMNS
RHS
RANGES
BOUNDS
ENDATA

	Campo-1	Campo-2	Campo-3	Campo-4	Campo-5	Campo-6
Localização	2-3	5-12	15-22	25-36	40-47	50-61
Conteúdo	Indicador	Nome	Nome	Valor	Nome	Valor

Tab. B.1: Formato MPS.

MINOS requer um outro arquivo de entrada, no formato SPC, arquivo no qual o usuário estima um número de iterações inicial suficiente para que o problema seja resolvido. Encontra-se a seguir um exemplo de arquivo SPC.

O próximo passo é executar o programa MINOS para obter a resolução do problema. Além do resultado na tela, o MINOS grava todos os dados em um arquivo de saída no formato OUT. Este arquivo contém todas as informações necessárias, como por exemplo, número de linhas, colunas e elementos não nulos da decomposição, a base e as colunas que entram e que saem da base na primeira fase, a base e as colunas que entram e saem da base na segunda fase (neste trabalho utiliza-se somente a segunda fase), as bases *Crash*, as variáveis artificiais utilizadas e muitas outras informações que não são utilizadas neste trabalho. A seguir, encontra-se um exemplo do arquivo de saída do MINOS.

Exemplo 1: Arquivo de entrada kb2.spc.

```
Begin kb2
   Minimize
   Rows                  44
   Columns               41
   Elements             441

   MPS file              10
```

```
   Iterations            55
   Debug level          100
   New basis file        11
   Backup file           12

   Crash option           0
   Print level            1  * OK for small problems
   Print frequency        1
   Summary frequency      1
End kb2
```

Exemplo 2: Arquivo de saída kb2.out:

```
1

     ==============================
     M I N O S  5.51     (Apr 2004)
     ==============================

     Begin kb2
        Minimize
        Rows                44
        Columns             41
        Elements           441

        MPS file            10

        Iterations         500
        Debug level        100
        New basis file      11
        *Backup file        12
        *Punch file         20
        *Dump file          30

        Print level              1  * OK for small problems
        Print frequency          1
        Summary frequency        1
     End kb2

 Reasonable Workspace limits are        0 ...   26985
 Actual     Workspace limits are        0 ... 1000000 ... 1000000  words of  z.
1

 Parameters
 ----------
 MPS INPUT DATA.
 Row limit.............       44       List limit.............        0       Lower bound default....  0.00E+00
 Column limit...........       41       Error message limit....       10       Upper bound default....  1.00E+20
 Elements limit ........      441       Phantom elements.......        0       Aij tolerance..........  1.00E-10

 FILES.
 MPS file ..............       10       Old basis file ........        0       (Card reader)..........        5
 Solution file..........        0       New basis file ........       11       (Printer)..............        9
 Insert file............        0       Backup basis file......        0       (Specs file)...........        4
 Punch file.............        0       Load file..............        0       Dump file..............        0
```

```
 FREQUENCIES.
 Log frequency.........         1       Check row error........        60       Save new basis map.....       100
 Summary frequency......        1       Factorize basis........       100       Expand frequency.......     10000

 LP PARAMETERS.
 Scale option...........        2       Feasibility tolerance..  1.00E-06       Iteration limit........       500
 Scale tolerance........    0.900       Optimality tolerance...  1.00E-06       Partial  price.........        10
 Crash option...........        3       Pivot tolerance........  3.25E-11       Multiple price.........         1
 Crash tolerance........    0.100       Weight on objective....  0.00E+00

 NONLINEAR PROBLEMS.
 Nonlinear constraints..        0       Hessian dimension......         1       Function precision.....  3.00E-13
 Nonlinear Jacobian vars        0       Superbasics limit......         1       Difference interval....  5.48E-07
 Nonlinear objectiv vars        0       Linesearch tolerance...   0.10000       Central difference int.  6.69E-05
 Problem number.........        0       Subspace tolerance.....   0.50000       Derivative level.......         3
 Unbounded objective val  1.00E+20       Unbounded step size....  1.00E+10       Verify level...........         0

 MISCELLANEOUS.
 LU factor tolerance....    100.00       Workspace (user).......         0       Debug level............       100
 LU update tolerance....     10.00       Workspace (total)......   1000000       Linesearch debug after.   9999999
 LU singularity tol.....  3.25E-11       eps (machine precision)  2.22E-16       nwordr, nwordi, nwordh.   2  2  2
 LU swap tolerance......  1.22E-04       Timing level...........         3
1

 MPS file
 --------
      1    NAME          KB2
      2    ROWS
     47    COLUMNS
    208    RHS
    209    BOUNDS
 ===>  Warning - the RHS is zero
    219    ENDATA



 Names selected
 --------------
 Objective      FAT7..J. (Min)       1
 RHS                                 0
 RANGES                              0
 BOUNDS         77BOUND              9



 Length of row-name hash table           101
 Collisions during table lookup          144

 Nonzeros allowed for in LU factors   499164

 Scale option  2,      Partial price       4
 Partial price section size (A)           10
 Partial price section size (I)           11



 Matrix Statistics
 -----------------
               Total      Normal        Free       Fixed     Bounded
 Rows             44          27           1          16           0
 Columns          41          32           0           0           9

 No. of matrix elements                  291     Density      16.131
 Biggest                          1.1300E+02  (excluding fixed columns,
 Smallest                         1.7000E-01   free rows, and RHS)

 No. of objective coefficients             5
 Biggest                          1.6500E+01  (excluding fixed columns)
 Smallest                         8.7570E-02
```

```
 Nonlinear constraints       0     Linear constraints      44
 Nonlinear variables         0     Linear variables        41
 Jacobian  variables         0     Objective variables      0

 Initial basis
 -------------
 No basis file supplied


 Scaling
 -------
             Min elem    Max elem       Max col ratio
 After  0    1.70E-01    1.13E+02              190.04
 After  1    7.79E-02    1.28E+01              164.90
 After  2    1.21E-01    8.29E+00               68.76
 After  3    1.46E-01    6.86E+00               47.02
 After  4    1.46E-01    6.86E+00               47.02

            Min scale                       Max scale      Between 0.5 and 2.0
 Col     39   1.6E-02            Col     19   9.6E+00               12    29.3
 Row      6   1.8E-01            Row     32   4.6E+01                6    13.6

 Norm of fixed columns and slacks             0.0E+00
 (before and after row scaling)               0.0E+00


 Crash option  3

 Crash on linear E  rows:
 Slacks     0  Free cols     0  Preferred     0
 Unit      11  Double        4  Triangle      0  Pad     1
1

 Iterations
 ----------

 Factor      1  Demand      0  Itn           0  Infeas       0
 Nonlin      0  Linear     15  Slacks       29  Objective       0.00000000E+00
 m          44 =n          44  Elems       110  Amax   6.9E+00  Density   5.68
 Merit     0.0  lenL         0  L+U         110  Cmpressns    0  Incres    0.00
 Utri       44  lenU       110  Ltol   1.0E+02  Umax   6.9E+00  Ugrwth 1.0E+00
 Ltri        0  dense1       0  Lmax   0.0E+00
 bump        0  dense2       0  DUmax  1.7E+00  DUmin  7.0E-01  condU  2.4E+00
 Itn       0 -- row check.   Max residual = 0.0E+00  on row    1.   Norm x = 1.00E+00

 BS and SB values:

     42    0.0000000          55    0.0000000          57    0.0000000          58    0.0000000          59    0.0000000
     60    0.0000000          61    0.0000000          62    0.0000000          63    0.0000000          64    0.0000000
     65    0.0000000          66    0.0000000          67    0.0000000          68    0.0000000          69    0.0000000
     70    0.0000000          71    0.0000000          74    0.0000000          75    0.0000000          76    0.0000000
     77    0.0000000          78    0.0000000          79    0.0000000          80    0.0000000          81    0.0000000
     82    0.0000000          83    0.0000000          84    0.0000000          85    0.0000000           2    0.0000000
      5    0.0000000          15    0.0000000          22    0.0000000          23    0.0000000          24    0.0000000
     25    0.0000000          26    0.0000000          28    0.0000000          29    0.0000000          32    0.0000000
     33    0.0000000          35    0.0000000          38    0.0000000          39    0.0000000

 Itn       0 -- linear E rows feasible.   Obj =   0.000000000E+00

 Crash on linear LG rows:
 Slacks     1  Free cols     0  Preferred     0
 Unit       0  Double        0  Triangle      5  Pad    22

 Factor      2  Demand      0  Itn           0  Infeas       1
 Nonlin      0  Linear     20  Slacks       24  Sum infeas      0.00000000E+00
 m          44 =n          44  Elems       130  Amax   6.9E+00  Density   6.71
 Merit     0.0  lenL         0  L+U         130  Cmpressns    0  Incres    0.00
```

```
Utri       44  lenU      130  Ltol   1.0E+02  Umax   6.9E+00  Ugrwth 1.0E+00
Ltri        0  dense1      0  Lmax   0.0E+00
bump        0  dense2      0  DUmax  3.2E+00  DUmin  7.0E-01  condU  4.7E+00
Itn      0 -- row check.   Max residual = 0.0E+00  on row    1.   Norm x = 1.00E+00

BS and SB values:

    42    0.0000000         55    0.0000000         57    0.0000000         59    0.0000000         60    0.0000000
    62    0.0000000         63    0.0000000         65    0.0000000         66    0.0000000         68    0.0000000
    69    0.0000000         70    0.0000000         71    0.0000000         75    0.0000000         76    0.0000000
    77    0.0000000         78    0.0000000         79    0.0000000         80    0.0000000         81    0.0000000
    82    0.0000000         83    0.0000000         84    0.0000000         85    0.0000000          2    0.0000000
     5    0.0000000         15    0.0000000         22    0.0000000         23    0.0000000         24    0.0000000
    25    0.0000000         26    0.0000000         28    0.0000000         29    0.0000000         31    0.0000000
    32    0.0000000         33    0.0000000         35    0.0000000         36    0.0000000         37    0.0000000
    38    0.0000000         39    0.0000000         40    0.0000000         41    0.0000000

Itn      0 -- feasible solution.  Objective =   0.000000000E+00

Itn      0 -- toldj = 7.5E+00    Norm pi = 2.7E+01    wtobj =  0.0E+00

   Itn ph pp     dj    +sbs  -sbs   -bs  step    pivot   ninf  sinf,objective     L     U ncp
     1  2  3 -3.7E+01    30    30    70 2.8E-11 -1.7E+00    0 -1.05047480E-09     0   130   0

BS and SB values...
    42   0.45237847E-10     55   0.21964872E-10     57   0.12740412E-10     59   0.11435545E-10     60   0.68066773E-11
    62   0.56070086E-11     63  -0.79788757E-11     65  -0.36128977E-11     66  -0.76125867E-11     68  -0.34914242E-11
    69  -0.22954023E-11     30   0.28119687E-10     71  -0.75406784E-11     75   0.89577172E-10     76  -0.21406257E-10
    77   0.83385771E-10     78   0.87309939E-10     79   0.44899820E-10     80   0.44552893E-10     81   0.49641103E-10
    82   0.10907909E-09     83   0.24955154E-10     84   0.12532545E-09     85   0.26732240E-10      2   0.24859301E-10
     5    0.0000000         15   0.13131007E-09     22   0.38716058E-10     23   0.10399324E-09     24   0.16455737E-10
    25    0.0000000         26    0.0000000         28    0.0000000         29    0.0000000         31   0.23843970E-10
    32   0.12689677E-10     33   0.24262655E-10     35   0.33141317E-10     36   0.69510010E-10     37   0.59263685E-10
    38    0.0000000         39    0.0000000         40   0.14895480E-09     41   0.93037990E-10
     2  2  1 -1.5E+02     9     9    76 1.6E-11  3.1E+00    0 -3.45506451E-09     3   130   0

BS and SB values...
    42   0.14878956E-09     55   0.17925096E-10     57   0.41903855E-10     59   0.37612084E-10     60   0.22387504E-10
    62   0.18441734E-10     63  -0.26242924E-10     65  -0.11883003E-10     66  -0.25038181E-10     68  -0.11483470E-10
    69  -0.75496939E-11     30   0.92487066E-10     71  -0.24801671E-10     75   0.29462383E-09      9   0.15791216E-10
    77   0.27426000E-09     78   0.28716678E-09     79   0.16155726E-09     80   0.14194596E-09     81   0.19466669E-09
    82   0.35876661E-09     83   0.82078758E-10     84   0.41220171E-09     85   0.87923683E-10      2   0.81763491E-10
     5    0.0000000         15   0.43188544E-09     22   0.12733906E-09     23   0.34203901E-09     24   0.77979171E-10
    25    0.0000000         26    0.0000000         28    0.0000000         29    0.0000000         31   0.78424017E-10
    32   0.41736986E-10     33   0.79801095E-10     35   0.10900346E-09     36   0.22862192E-09     37   0.19492125E-09
    38   0.64552442E-10     39   0.40403573E-10     40   0.48991983E-09     41   0.30600663E-09
     3  2  3 -2.3E+01    74    74    69 7.9E-10 -6.2E-02    0 -2.12235032E-08     8   135   0

BS and SB values...
    42   0.91397300E-09     55  -0.11766412E-10     57   0.83288398E-10     59   0.78796476E-10     60   0.46398712E-10
    62   0.39711369E-10     63  -0.16047983E-09     65  -0.72666530E-10     66  -0.15311263E-09     68  -0.70223323E-10
    74   0.78787034E-09     30   0.56557374E-09     71  -0.15166633E-09     75   0.18016735E-08      9   0.13185332E-09
    77   0.16771453E-08     78   0.17560725E-08     79   0.10189648E-08     80   0.85776447E-09     81   0.12605742E-08
    82   0.21939173E-08     83   0.50192521E-09     84   0.25206818E-08     85   0.53766789E-09      2   0.49999730E-09
     5    0.0000000         15   0.26410511E-08     22   0.77869947E-09     23   0.20916252E-08     24   0.53016344E-09
    25    0.0000000         26    0.0000000         28    0.0000000         29    0.0000000         31   0.21026926E-09
    32   0.25522859E-09     33   0.48799692E-09     35   0.66657423E-09     36   0.12526604E-08     37   0.11087796E-08
    38   0.53899928E-09     39   0.33736132E-09     40   0.29959409E-08     41   0.18712813E-08
     4  2  1 -5.1E+01     7     7    57 2.5E-10 -2.0E-01    0 -3.40179101E-08    16   128   0

BS and SB values...
    42   0.14649538E-08     55  -0.37165358E-10      7   0.25067616E-09     59   0.12542218E-09     60   0.78307963E-10
    62   0.67604291E-10     63  -0.27530983E-09     65  -0.12466246E-09     66  -0.26267109E-09     68  -0.12047104E-09
    74   0.14618367E-08     30   0.97026530E-09     71  -0.26018990E-09     75   0.30506496E-08      9   0.23113609E-09
    77   0.28362155E-08     78   0.28229603E-08     79   0.17524152E-08     80   0.14700956E-08     81   0.21723813E-08
    82   0.37637565E-08     83   0.86107360E-09     84   0.43243345E-08     85   0.92239167E-09      2   0.65697185E-09
     5    0.0000000         15   0.45308331E-08     22   0.13358914E-08     23   0.35882702E-08     24   0.78405771E-09
    25    0.0000000         26    0.0000000         28    0.0000000         29   0.17022993E-09     31   0.32305336E-09
    32   0.43785527E-09     33   0.83717904E-09     35   0.11435359E-08     36   0.21855109E-08     37   0.19479157E-08
```

```
    38   0.94485438E-09     39   0.59138728E-09     40   0.51396615E-08     41   0.32102610E-08
     5  2  2  1.8E+02    58    58    60 9.3E-11 -5.3E-01    0 -5.08609157E-08    22   136   0

BS and SB values...
    42   0.21902842E-08     55  -0.72524056E-10      7   0.70058998E-09     59   0.11951466E-09     58  -0.92798072E-10
    62   0.11034013E-09     63  -0.43516840E-09     65  -0.19704768E-09     66  -0.41519098E-09     68  -0.19042251E-09
    74   0.24000871E-08     30   0.15336496E-08     71  -0.41126909E-09     75   0.47732038E-08      9   0.36935086E-09
    77   0.44332863E-08     78   0.42318413E-08     79   0.27734754E-08     80   0.23225417E-08     81   0.34417376E-08
    82   0.59491806E-08     83   0.13610557E-08     84   0.68352581E-08     85   0.14579781E-08      2   0.79464766E-09
     5    0.0000000        15   0.71616600E-08     22   0.21115763E-08     23   0.56717982E-08     24   0.10839907E-08
    25    0.0000000        26    0.0000000         28    0.0000000         29   0.47575877E-09     31   0.48006377E-09
    32   0.69209581E-09     33   0.13232868E-08     35   0.18075296E-08     36   0.34674819E-08     37   0.31392160E-08
    38   0.15098584E-08     39   0.94502507E-09     40   0.81240044E-08     41   0.50742981E-08
     6  2  3 -1.6E+01    27    27    77 1.3E-09  3.5E+00    0 -7.15134651E-08    28   135   0

BS and SB values...
    42   0.30796695E-08     55  -0.11241356E-09      7   0.74456586E-09     59   0.12833433E-09     58  -0.10804793E-09
    62   0.11411651E-09     63  -0.61551101E-09     65  -0.27870823E-09     66  -0.58725454E-09     68  -0.26933745E-09
    74   0.34585634E-08     30   0.21692250E-08     71  -0.58170734E-09     75   0.16805293E-08      9   0.52527626E-09
    27   0.12799915E-08     78   0.43363735E-08     79   0.39253726E-08     80   0.32842190E-08     81   0.48737473E-08
    82   0.84146417E-08     83   0.19251048E-08     84   0.96679277E-08     85   0.20621938E-08      2   0.10099930E-08
     5   0.18359874E-08     15   0.10129597E-07     22   0.29866563E-08     23   0.80223064E-08     24   0.14620926E-08
    25    0.0000000        26    0.0000000         28    0.0000000         29   0.50562204E-09     31   0.65719326E-09
    32   0.97891434E-09     33   0.18716837E-08     35   0.25566066E-08     36   0.49676945E-08     37   0.44083877E-08
    38   0.21472613E-08     39   0.13439775E-08     40   0.11490757E-07     41   0.71771900E-08
     7  2  4 -9.7E+01    76    76    55 2.3E-10  2.1E-01    0 -9.36019707E-08    33   138   0

BS and SB values...
    42   0.40308931E-08     76   0.15841832E-09      7   0.14029361E-08     59   0.13996261E-09     58  -0.80431237E-10
    62   0.11899708E-09     63  -0.83704265E-09     65  -0.37901950E-09     66  -0.79861625E-09     68  -0.36627604E-09
    74   0.52593836E-08     30   0.29499616E-08     71  -0.79107253E-09     75   0.22744782E-08      9   0.71681397E-09
    27   0.17277298E-08     78   0.56203114E-08     79   0.53403554E-08     80   0.44655368E-08     81   0.66328182E-08
    82   0.11443197E-07     83   0.26179789E-08     84   0.13147560E-07     85   0.28044083E-08      2   0.10639477E-08
     5   0.24782120E-08     15   0.13775391E-07     22   0.40615987E-08     23   0.10909655E-07     24   0.17871597E-08
    25    0.0000000        26    0.0000000         28    0.0000000         29   0.95271014E-09     31   0.70366662E-09
    32   0.13312403E-08     33   0.25453307E-08     35   0.34767676E-08     36   0.67687158E-08     37   0.60253862E-08
    38   0.29302425E-08     39   0.18340479E-08     40   0.15626452E-07     41   0.97603683E-08
     8  2  1 -5.8E+01     8     8    58 1.1E-09 -7.3E-02    0 -1.57932849E-07    38   143   0

BS and SB values...
    42   0.68012503E-08     76   0.82485026E-09      7   0.33203831E-08     59   0.17382895E-09      8   0.11076256E-08
    62   0.13321130E-09     63  -0.14822345E-08     65  -0.67116747E-09     66  -0.14141891E-08     68  -0.64860135E-09
    74   0.10504118E-07     30   0.52237900E-08     71  -0.14008307E-08     75   0.40043037E-08      9   0.94630192E-09
    27   0.30317294E-08     78   0.93596703E-08     79   0.84163725E-08     80   0.73343189E-08     81   0.92642799E-08
    82   0.20263606E-07     83   0.46359152E-08     84   0.23281690E-07     85   0.49660442E-08      2   0.12210859E-08
     5   0.43486360E-08     15   0.24393453E-07     22   0.71922762E-08     23   0.20789550E-07     24   0.22378602E-08
    25    0.0000000        26    0.0000000         28    0.0000000         29   0.22548159E-08     31   0.83901629E-09
    32   0.23573594E-08     33   0.45072698E-08     35   0.61566576E-08     36   0.12014036E-07     37   0.10734755E-07
    38   0.52493246E-08     39   0.31966377E-08     40   0.27671310E-07     41   0.17283653E-07
     9  2  2  1.4E+03    61    61    59 9.4E-11 -5.2E-01    0 -2.88107923E-07    40   145   0

BS and SB values...
    42   0.12407135E-07     76   0.18674717E-08      7   0.69553721E-08     61  -0.93860751E-10      8   0.33186630E-08
    62   0.14654610E-09     63  -0.27701638E-08     65  -0.12543520E-08     66  -0.26429929E-08     68  -0.12121779E-08
    74   0.20344222E-07     30   0.97627963E-08     71  -0.26180273E-08     75   0.74627510E-08      9   0.14044048E-08
    27   0.56411568E-08     78   0.16960724E-07     79   0.14556705E-07     80   0.13060969E-07     81   0.14517192E-07
    82   0.37870867E-07     83   0.86641109E-08     84   0.43511396E-07     85   0.92810924E-08      2   0.16874936E-08
     5   0.80915328E-08     15   0.45589181E-07     22   0.13441721E-07     23   0.40511753E-07     24   0.32386465E-08
    25    0.0000000        26    0.0000000         28    0.0000000         29   0.47232752E-08     31   0.13243434E-08
    32   0.44056938E-08     33   0.84236841E-08     35   0.11506242E-07     36   0.22489320E-07     37   0.20129206E-07
    38   0.98786662E-08     39   0.59166338E-08     40   0.51715204E-07     41   0.32301602E-07
    10  2  2  2.3E+03    57    57    62 1.4E-10 -3.5E-01    0 -6.07533569E-07    49   148   0

BS and SB values...
    42   0.26162942E-07     76   0.47980178E-08      7   0.16433564E-07     61  -0.23134917E-09      8   0.88143086E-08
    57  -0.59756335E-11     63  -0.59713773E-08     65  -0.27038866E-08     66  -0.56972472E-08     68  -0.26129761E-08
    74   0.45544051E-07     30   0.21044727E-07     71  -0.56434311E-08     75   0.16046523E-07      9   0.25430427E-08
    27   0.12112318E-07     78   0.35539267E-07     79   0.29818811E-07     80   0.27294849E-07     81   0.27573572E-07
    82   0.81634608E-07     83   0.18676396E-07     84   0.93793356E-07     85   0.20006364E-07      2   0.24953075E-08
```

```
     5   0.17373603E-07     15   0.98272240E-07     22   0.28975033E-07     23   0.89532291E-07     24   0.54934968E-08
    25    0.0000000         26    0.0000000         28    0.0000000         29   0.11159755E-07     31   0.22771069E-08
    32   0.94969330E-08     33   0.18158131E-07     35   0.24802907E-07     36   0.48488430E-07     37   0.43531881E-07
    38   0.21385129E-07     39   0.12677322E-07     40   0.11147752E-06     41   0.69629476E-07
    11  2  3  3.7E+01    69    69     2 4.5E-10  5.6E+00    0 -6.24033307E-07    58   145   0

BS and SB values...
    42   0.26873489E-07     76   0.56013430E-08      7   0.21151973E-07     61  -0.34217441E-09      8   0.96370708E-08
    57  -0.18865778E-10     63  -0.64506362E-08     65  -0.29208989E-08     66  -0.61545047E-08     68  -0.28226919E-08
    74   0.50114361E-07     30   0.22733763E-07     71  -0.60963693E-08     75   0.17239503E-07      9   0.27135100E-08
    27   0.12971680E-07     78   0.35981396E-07     79   0.32103727E-07     80   0.29425827E-07     81   0.29528264E-07
    82   0.88186550E-07     83   0.20175351E-07     84   0.10132115E-06     85   0.21612062E-07     69  -0.40776258E-09
     5   0.18606251E-07     15   0.10615951E-06     22   0.31300551E-07     23   0.96871237E-07     24   0.40992368E-08
    25    0.0000000         26    0.0000000         28    0.0000000         29   0.14363946E-07     31   0.21471134E-08
    32   0.10259151E-07     33   0.19615491E-07     35   0.26793572E-07     36   0.52971646E-07     37   0.47676302E-07
    38   0.23107781E-07     39   0.13689476E-07     40   0.12042464E-06     41   0.75217894E-07
    12  2  2  6.9E+03    60    60    23 2.8E-02 -1.1E+03    0 -1.89617262E+02    65   146   0

BS and SB values...
    42    8.1657140         76    1.8628563          7    6.4257428         61  -0.14201560          8    3.3641790
    57  -0.83162700E-01     63   -1.9596342         65  -0.88733781         66   -1.8696726         68  -0.85750358
    74    15.576922         30    6.9062739         71   -1.8520117         75    5.2371764          9   0.69702123
    27    3.9406576         78    10.930763         79    9.3427530         80    8.7133204         81    7.9925095
    82    26.790130         83    6.1290558         84    30.780281         85    6.5655130         69  -0.81042301E-01
     5    5.6523798         15    32.250122         22    9.5087722         60  -0.27553435E-01     24    1.0529738
    25    0.0000000         26    0.0000000         28    0.0000000         29    4.3636130         31   0.53170616
    32    3.1166202         33    5.9589761         35    8.1396004         36    16.027126         37    14.434588
    38    7.0437224         39    4.1385792         40    36.583716         41    22.850391
    13  2  1 -1.2E+01     2     2    69 4.7E-01 -1.7E-01    0 -1.95255612E+02    75   145   0

BS and SB values...
    42    8.4085249         76    1.7972259          7    5.8318076         61  -0.12765401          8    3.3641790
    57  -0.84568523E-01     63   -1.9596342         65  -0.88733781         66   -1.8696726         68  -0.85750358
    74    15.433344         30    6.9062739         71   -1.8520117         75    5.2537575          9   0.69702123
    27    3.9603592         78    11.351865         79    9.3427530         80    8.7133204         81    7.9925095
    82    26.790130         83    6.1290558         84    30.780281         85    6.5655130          2   0.47095688
     5    5.6806393         15    32.250122         22    9.5087722         60  -0.28825262E-01     24    1.3647261
    25    0.0000000         26    0.0000000         28    0.0000000         29    3.9602816         31   0.58078353
    32    3.1166202         33    5.9589761         35    8.1396004         36    15.920766         37    14.318662
    38    7.0437224         39    4.1385792         40    36.583716         41    22.850391
    14  2  2  1.6E+01    58    58    60 2.8E-01 -1.0E-01    0 -1.99641529E+02    76   143   0

BS and SB values...
    42    8.5974009         76    1.2107520          7    5.3549367         61  -0.27053798          8    3.3641790
    57  -0.48358809E-01     63   -1.9596342         65  -0.88733781         66   -1.8696726         68  -0.85750358
    74    14.150327         30    6.9062739         71   -1.8520117         75    5.2670705          9   0.69702123
    27    3.9761777         78    11.689969         79    9.3427530         80    8.7133204         81    7.9925095
    82    26.790130         83    6.1290558         84    30.780281         85    6.5655130          2   0.84908844
     5    5.7033288         15    32.250122         22    9.5087722         58  -0.27580473         24    1.6150321
    25    0.0000000         26    0.0000000         28    0.0000000         29    3.6364467         31    1.0193392
    32    3.1166202         33    5.9589761         35    8.1396004         36    15.933251         37    14.301364
    38    7.0437224         39    4.1385792         40    36.583716         41    22.850391
    15  2  2  1.4E+02    62    62    57 4.0E-02 -1.2E+00    0 -2.05111862E+02    77   147   0

BS and SB values...
    42    8.8329764         76   0.43339614          7    4.7588860         61  -0.51896191          8    3.3641790
    62  -0.39982013E-01     63   -1.9596342         65  -0.88733781         66   -1.8696726         68  -0.85750358
    74    12.449721         30    6.9062739         71   -1.8520117         75    5.2837107          9   0.69702123
    27    3.9959495         78    12.112571         79    9.3427530         80    8.7133204         81    7.9925095
    82    26.790130         83    6.1290558         84    30.780281         85    6.5655130          2    1.3217228
     5    5.7316890         15    32.250122         22    9.5087722         58  -0.64343386         24    1.9278947
    25    0.0000000         26    0.0000000         28    0.0000000         29    3.2316788         31    1.6006333
    32    3.1166202         33    5.9589761         35    8.1396004         36    15.956981         37    14.268485
    38    7.0437224         39    4.1385792         40    36.583716         41    22.850391
    16  2  2  9.4E+01    59    59    76 2.8E-02  1.5E+01    0 -2.07760160E+02    78   150   0

BS and SB values...
    42    8.9470232         59  -0.28024478E-01      7    4.4687416         61  -0.66035186          8    3.3641790
    62  -0.62637317E-01     63   -1.9596342         65  -0.88733781         66   -1.8696726         68  -0.85750358
```

```
    74    11.501589         30    6.9062739         71   -1.8520117         75    5.2918108          9   0.69702123
    27    4.0055739         78    12.318284         79    9.3427530         80    8.7133204         81    7.9925095
    82    26.790130         83    6.1290558         84    30.780281         85    6.5655130          2    1.5517908
     5    5.7454941         15    32.250122         22    9.5087722         58  -0.88529291         24    2.0801894
    25    0.0000000         26    0.0000000         28    0.0000000         29    3.0346466         31    1.9247199
    32    3.1166202         33    5.9589761         35    8.1396004         36    15.963908         37    14.258887
    38    7.0437224         39    4.1385792         40    36.583716         41    22.850391
    17  2  2 -1.6E+01    16    16    71 8.8E+00 -2.1E-01    0 -3.50470149E+02    83   147   0

BS and SB values...
    42    15.092713         59  -0.47274429E-01      7    7.5383102         61   -1.1139461          8    5.6750261
    62  -0.10566275         63   -2.4779230         65   -1.1255846         66   -2.2170476         68   -1.0056091
    74    19.402005         30    11.650178         16    8.8065432         75    8.9267437          9    1.1758036
    27    6.7569937         78    20.779686         79    15.760269         80    14.698481         81    13.482546
    82    62.488658         83    10.339090         84    62.791015         85    43.708756          2    2.6177124
     5    9.6920611         15    28.375697         22    16.040327         58   -1.4933986         24    9.5825597
    25    0.0000000         26    0.0000000         28    0.0000000         29    5.1191386         31    3.2468057
    32    5.2574196         33    10.052183         35    13.730674         36    26.929481         37    24.053284
    38    11.882040         39    6.9813601         40    56.808624         41    37.400410
    18  2  3 -7.6E+00    21    21    29 5.7E-01 -9.6E-01    0 -3.54758824E+02    94   156   0

BS and SB values...
    42    15.277402         59  -0.48611638E-01      7    7.7515396         61   -1.1454553          8    5.8355504
    62  -0.10865154         63   -2.4871689         65   -1.1258964         66   -2.2178374         68   -1.0018266
    74    19.950812         30    11.979716         16    8.9675848         75    9.1792465          9    1.2090624
    27    6.9481227         78    21.367462         79    16.206066         80    15.114243         81    13.863915
    82    64.857825         83    10.631543         84    64.903419         85    45.766855          2    2.6917572
     5    9.9662117         15    28.106557         22    16.494045         58   -1.5356410         24    9.7928809
    25    0.0000000         26    0.0000000         28    0.0000000         21    0.56622570         31    3.3386452
    32    5.4061315         33    10.336520         35    14.119061         36    27.691211         37    24.733657
    38    12.218136         39    7.1788355         40    58.368417         41    38.531968
    19  2  4 -3.7E+01    34    34    21 8.8E-01  6.4E-01    0 -3.87366740E+02   103   168   0

BS and SB values...
    42    16.681635         59  -0.52287978E-01      7    8.3377634         61   -1.2320823          8    6.2768741
    62  -0.11686850         63   -2.0266343         65  -0.80227371         66   -1.7439013         68  -0.67474054
    74    19.213133         30    12.885703         16    11.100161         75    9.8734430          9    1.3004999
    27    7.4735866         78    22.983414         79    17.431678         80    16.257284         81    14.912398
    82    71.786153         83    10.122381         84    71.128044         85    53.382520          2    2.8953261
     5    10.719924         15    27.366623         22    17.741436         58   -1.6517766         24    11.536513
    25    0.0000000         26    0.0000000         28    0.0000000         34    0.88201345         31    3.5911360
    32    5.8149796         33    11.118238         35    15.186840         36    29.785407         37    26.604183
    38    13.142154         39    7.7217475         40    62.709478         41    41.560396
    20  2  1 -3.4E+01     1     1    68 1.9E+00 -3.6E-01    0 -4.51242749E+02   103   163   0

BS and SB values...
    42    19.432404         59  -0.29838585E-01      7    8.3377634         61   -1.2210069          8    7.1382987
    62  -0.10423684         63   -1.0523738         65  -0.13597527         66  -0.74102698          1    1.8989628
    74    17.909868         30    14.654109         16    15.008187         75    10.755728          9    1.4789777
    27    8.4023373         78    25.174252         79    19.823963         80    18.488399         81    16.958943
    82    86.321516         83    9.2084040         84    83.832327         85    69.545198          2    3.5814718
     5    12.052100         15    24.074697         22    20.176232         58   -1.7194051         24    14.955529
    25    0.0000000         26    0.0000000         28    0.0000000         34    2.5499864         31    4.0839758
    32    6.6130149         33    12.644081         35    17.271049         36    33.961848         37    30.132309
    38    14.945755         39    8.7814634         40    71.098436         41    47.603598
    21  2  2 -2.2E+01    13    13    59 3.8E+00 -7.8E-03    0 -5.36699957E+02   109   174   0

BS and SB values...
    42    23.112549         13    3.8385921          7    8.3377634         61   -1.2062860          8    3.3704354
    62  -0.87447494E-01     63   -1.2211714         65  -0.15778529         66  -0.85988547          1    4.4229670
    74    20.782557         30    17.004584         16    17.415455         75    11.928416          9    1.5066971
    27    9.6367853         78    28.086203         79    23.569571         80    21.372181         81    21.961047
    82    100.16723         83    10.685404         84    97.278779         85    80.700037          2    4.4934616
     5    13.822761         15    27.936206         22    23.412439         58   -1.8092934         24    17.261289
    25    0.0000000         26    0.0000000         28    5.6564885         34    2.9589965         31    4.7390332
    32    7.6737225         33    14.672153         35    20.041273         36    39.512960         37    34.821712
    38    17.232731         39    10.374267         40    82.502410         41    55.239071
LU file has increased by a factor of   2.2
```

```
Factor      3  Demand      2  Itn         21  Infeas       0
Nonlin      0  Linear     27  Slacks      17  Objective      -5.36699957E+02
m          44 =n          44  Elems      189  Amax   6.9E+00  Density   9.76
Merit     0.5  lenL       32  L+U        200  Cmpressns    0  Incres    5.82
Utri       23  lenU      168  Ltol   1.0E+02  Umax   2.8E+01  Ugrwth 4.1E+00
Ltri        1  dense1     13  Lmax   1.8E+01
bump       20  dense2      9  DUmax  7.5E+00  DUmin  2.8E-02  condU  2.7E+02
Itn     21 -- row check.   Max residual = 1.4E-13  on row   20.   Norm x = 1.00E+02

BS and SB values:

    42    23.112549         58   -1.8092934         61   -1.2062860         62  -0.87447494E-01     63   -1.2211714
    65  -0.15778529         66  -0.85988547         74    20.782557         75    11.928416         78    28.086203
    79    23.569571         80    21.372181         81    21.961047         82    100.16723         83    10.685404
    84    97.278779         85    80.700037          1    4.4229670          2    4.4934616          5    13.822761
     7    8.3377634          8    3.3704354          9    1.5066971         13    3.8385921         15    27.936206
    16    17.415455         22    23.412439         24    17.261289         25    0.0000000         26    0.0000000
    27    9.6367853         28    5.6564885         30    17.004584         31    4.7390332         32    7.6737225
    33    14.672153         34    2.9589965         35    20.041273         36    39.512960         37    34.821712
    38    17.232731         39    10.374267         40    82.502410         41    55.239071
Itn     21 -- feasible solution.  Objective =  -5.366999570E+02

   Itn ph pp     dj    +sbs  -sbs   -bs  step    pivot   ninf  sinf,objective     L     U ncp
    22  2  1 -3.6E+01     6     6    24 4.7E-01 -8.9E-01    0 -5.53613440E+02    32   168   0

BS and SB values...
    42    23.840915         58   -1.8572819         61   -1.2348595         62  -0.89334433E-01     63   -1.2545343
    65  -0.16209605         66  -0.88337789         74    21.350344         75    12.163163         78    28.647318
    79    24.286313         80    21.945566         81    22.854634         82    102.90384         83    10.977334
    84    99.936475         85    82.904795          1    4.5690994          2    4.6036562          5    14.157741
     7    8.3377634          8    2.8304155          9    1.5209052         13    4.4373511         15    28.699434
    16    17.891252         22    24.052076          6   0.47340005         25    0.0000000         26    0.0000000
    27    9.8703224         28    6.8626102         30    17.469156         31    4.8685055         32    7.8833717
    33    15.073002         34    3.0398375         35    20.588808         36    40.602562         37    35.759073
    38    17.689348         39    10.681407         40    84.756409         41    56.748225
    23  2  2  2.3E+02    59    59     1 7.9E-02  5.8E+01    0 -5.71706559E+02    43   177   0

BS and SB values...
    42    24.620080         58   -2.3123908         61   -1.6864343         62  -0.16101306         63   -1.2896210
    65  -0.16662953         66  -0.90808409         74    21.947468         75    12.449727         78    29.044183
    79    24.724386         80    22.594154         81    22.521369         82    105.78184         83    11.284346
    84    102.73149         85    85.223465         59  -0.78962630E-01      2    3.7815087          5    14.309301
     7    8.3377634          8    5.0032007          9    1.6527229         13    2.9256193         15    29.502096
    16    18.391632         22    24.724761          6    7.3095789         25    0.0000000         26    0.0000000
    27    9.9759853         28    9.3108137         30    17.957731         31    5.0046672         32    8.1038528
    33    15.494562         34    3.1248553         35    21.164633         36    41.647015         37    36.885420
    38    18.231077         39    10.901610         40    87.126865         41    58.335352
    24  2  3  1.7E+02    69    69     2 7.3E-01  5.2E+00    0 -6.92838871E+02    45   170   0

BS and SB values...
    42    29.836545         58   -3.1041888         61   -2.2757056         62  -0.22991069         63   -1.5248284
    65  -0.19702025         66   -1.0737049         74    25.950355         75    11.976431         78    24.858453
    79    29.708524         80    26.646440         81    28.543435         82    125.07485         83    13.342441
    84    121.46816         85    100.76694         59  -0.15716912         69  -0.73268637          5    15.428354
     7    8.3377634          8    1.7939352          9    1.7783843         13    6.6797195         15    34.882833
    16    21.745988         22    29.234185          6    23.173301         25    0.0000000         26    0.0000000
    27    10.756153         28    25.693399         30    21.232950         31    5.9174429         32    9.5818732
    33    18.320536         34    3.6947817         35    25.024743         36    50.486507         37    44.269923
    38    21.463629         39    13.044509         40    103.01749         41    68.974839
    25  2  1 -1.2E+01    10    10     8 5.6E+00  3.2E-01    0 -7.59663571E+02    49   164   0

BS and SB values...
    42    32.714297         58   -3.3694772         61   -2.4614473         62  -0.24881278         63   -1.6563061
    65  -0.21400823         66   -1.1662847         74    28.187913         75    12.749366         78    26.451075
    79    28.378072         80    26.577622         81    21.963584         82    135.85937         83    14.492887
    84    131.94170         85    109.45553         59  -0.17127803         69  -0.78651797          5    16.635903
     7    8.3377634         10    5.5796863          9   0.92238831         13    7.8880870         15    37.890591
    16    23.621027         22    31.754890          6    26.630510         25    6.8691371         26    0.0000000
    27    11.598017         28    29.838721         30    23.063752         31    6.4276720         32    10.408066
```

```
    33    19.900217         34    4.0133627         35    27.182491         36    54.854773         37    48.035816
    38    23.258415         39    14.262691         40    111.90013         41    74.922166
    26  2  2 -1.6E+01    20    20     9 7.6E+00  1.2E-01    0 -8.77321565E+02    55   161   0

BS and SB values...
    42    37.781143         58   -3.8361222         61   -2.7881689         62  -0.28206174         63   -1.8079153
    65  -0.24038509         66   -1.2634227         74    33.056027         75    14.108966         78    29.252513
    79    27.668708         80    27.639835         81    13.742607         82    153.82630         83    17.061467
    84    149.22063         85    123.69576         59  -0.19609574         69  -0.88120830          5    18.759995
     7    8.3377634         10    12.691125         20    7.5514938         13    7.9793098         15    37.890591
    16    25.641496         22    36.188836          6    32.711777         25    15.624010         26    0.0000000
    27    13.078866         28    39.478818         30    26.284152         31    7.3251701         32    11.861348
    33    22.678891         34    4.2077418         35    30.977992         36    62.538599         37    54.660060
    38    26.492188         39    16.277270         40    127.42623         41    85.537675
    27  2  3  3.1E+02    68    68    28 1.0E+00 -3.8E+01    0 -1.19302132E+03    57   169   0

BS and SB values...
    42    51.376498         58   -5.0849373         61   -3.6625272         62  -0.37104117         63   -3.7302646
    65   -1.3320398         66   -3.0511064         74    52.939877         75    17.747470         78    36.749601
    79    36.740984         80    36.702645         81    18.248663         82    193.94378         83    27.943082
    84    189.55168         85    148.27495         59  -0.26251183         69   -1.1346145          5    24.444397
     7    8.3377634         10    16.852410         20    41.069747         13    10.595641         15    37.890591
    16    25.641496         22    48.054773          6    48.986199         25    20.746957         26    0.0000000
    27    17.041848         68   -1.0152564         30    34.902447         31    9.7270158         32    15.750559
    33    30.115060         34    2.0361371         35    41.135347         36    83.101721         37    72.387577
    38    35.178695         39    21.614414         40    169.09012         41    113.76880
Itn     27 -- toldj = 9.3E-01    Norm pi = 1.0E+02    wtobj =  0.0E+00
    28  2  2 -4.6E+00    17    17    68 3.9E+01 -2.6E-02    0 -1.37240680E+03    67   170   0

BS and SB values...
    42    59.101589         58   -5.8010576         61   -4.1639190         62  -0.42206552         63   -2.5121004
    65  -0.32060811         66   -1.7652979         74    50.694022         75    19.833932         78    41.048729
    79    41.943388         80    41.899620         81    20.832614         82    207.56518         83    26.205053
    84    208.63617         85    141.59401         59  -0.30059746         69   -1.2799277          5    27.704059
     7    8.3377634         10    19.238657         20    28.930526         13    12.095949         15    37.890591
    16    25.641496         22    54.859172          6    58.318600         25    55.146951         26    0.0000000
    27    19.314380         17    38.760591         30    39.844520         31    11.104330         32    17.980787
    33    34.379254         34    6.1493180         35    46.959979         36    94.893433         37    82.553240
    38    40.159884         39    24.674945         40    192.51300         41    130.69074
    29  2  3  1.4E+02    67    67    65 3.6E-01 -9.0E-01    0 -1.42292431E+03    67   179   0

BS and SB values...
    42    61.277084         58   -6.0028275         61   -4.3051883         62  -0.43644184         63   -2.1146667
    67  -0.35501947         66   -1.6599278         74    49.853518         75    20.421802         78    42.260026
    79    43.409188         80    43.363890         81    21.560653         82    211.40307         83    25.593929
    84    214.01331         85    139.71163         59  -0.31132825         69   -1.3208703          5    28.622483
     7    8.3377634         10    19.910992         20    25.510249         13    12.518667         15    37.890591
    16    25.641496         22    56.776340          6    60.948043         25    64.839294         26    0.0000000
    27    19.954676         17    49.681550         30    41.236970         31    11.492394         32    18.609163
    33    35.580709         34    7.3897807         35    48.601093         36    98.215797         37    85.417458
    38    41.563355         39    25.537262         40    199.17106         41    135.36700
    30  2  1 -3.3E+01     2     2    25 2.3E+00 -6.2E+00    0 -1.49826973E+03    76   185   0

BS and SB values...
    42    64.521774         58   -6.1137391         61   -4.3616882         62  -0.43677749         63   -2.2340644
    67  -0.35977198         66   -1.7592913         74    51.800016         75    22.430526         78    48.345402
    79    45.612032         80    45.564435         81    22.654771         82    214.00499         83    26.551710
    84    219.72159         85    131.54400         59  -0.29534301         69  -0.99658087          5    30.633959
     7    8.3377634         10    20.921395         20    26.039495         13    13.153940         15    37.890591
    16    23.440979         22    59.657515          6    59.031618          2    2.2926061         26    0.0000000
    27    21.357012         17    65.781321         30    43.329583         31    12.075587         32    19.553504
    33    37.386289         34    7.9938243         35    51.067406         36    102.58740         37    89.329329
    38    43.672531         39    26.833177         40    209.13976         41    142.45279
    31  2  2 -4.6E+00    11    11    10 1.7E+01  1.2E+00    0 -1.57723977E+03    80   192   0

BS and SB values...
    42    67.922555         58   -5.9488006         61   -4.1924222         62  -0.40578728         63   -2.4035203
    67  -0.35215594         66   -1.9064688         74    52.884204         75    26.235378         78    61.133849
    79    42.446786         80    44.763053         81    16.530486         82    208.49054         83    26.993091
```

```
    84    220.76058         85    106.91413         59  -0.23093342         69  -0.81419029E-01      5    33.713631
     7    8.3377634         11    17.148829         20    22.834025         13    19.658934         15    37.890591
    16    17.846379         22    62.736089          6    48.335077          2    8.1213320         26    15.985445
    27    23.504060         17    97.512271         30    45.565569         31    12.698737         32    20.562546
    33    39.315576         34    9.0301717         35    53.702695         36    106.34272         37    92.930631
    38    46.361751         39    27.490059         40    219.68178         41    150.19555
    32  2  3  1.7E+00    65    65    69 1.0E-01 -8.1E-01    0 -1.57741190E+03    83   192   0

BS and SB values...
    42    67.929967         58   -5.9092711         61   -4.1602682         62  -0.40136925         63   -2.5527103
    67  -0.24244172         66   -1.9603408         74    53.557606         75    26.476951         78    62.043562
    79    42.454326         80    44.771006         81    16.533422         82    207.84219         83    27.386032
    84    220.28941         85    105.77184         59  -0.22419948         65  -0.99985026E-01      5    33.852570
     7    8.3377634         11    17.151876         20    24.011879         13    19.662426         15    37.890591
    16    17.380649         22    62.747234          6    47.108420          2    8.6065528         26    15.988285
    27    23.600924         17    97.512271         30    45.573664         31    12.700993         32    20.566199
    33    39.322560         34    8.7710750         35    53.712236         36    106.23052         37    92.864199
    38    46.369987         39    27.494943         40    219.71260         41    150.23507
    33  2  1 -3.9E+00     4     4    65 2.0E+00 -4.9E-02    0 -1.58535387E+03    88   186   0

BS and SB values...
    42    68.271982         58   -5.8418203         61   -4.0328751         62  -0.38338818         63   -2.4109140
    67  -0.35406239         66   -1.9119310         74    53.193156         75    25.505881         78    59.041928
    79    42.644257         80    44.971300         81    16.607389         82    209.68525         83    27.155450
    84    221.90690         85    107.87195         59  -0.21030785          4    2.0482729          5    33.610072
     7    8.3377634         11    17.228609         20    23.310960         13    19.750391         15    37.890591
    16    17.987556         22    63.027950          6    47.644392          2    7.9742470         26    17.671259
    27    23.431862         17    97.512271         30    45.777549         31    12.757814         32    20.658207
    33    39.498480         34    9.0474773         35    53.952531         36    106.77051         37    93.189952
    38    46.577435         39    27.617949         40    220.70392         41    150.89409
    34  2  2 -2.8E+00    19    19    26 4.7E+01 -4.9E-01    0 -1.71756305E+03    88   197   0

BS and SB values...
    42    73.965463         58   -5.2591485         61   -2.5631762         62  -0.16664448         63   -2.7847675
    67  -0.60486880         66   -2.1390236         74    56.146726         75    13.890076         78    22.251312
    79    47.349622         80    49.933432         81    18.439848         82    126.45340         83    28.447370
    84    105.85539         85    111.43114         59  -0.39400161E-01      4    29.078450          5    32.080406
     7    8.3377634         11    19.129613         20    8.6547753         13    21.929648         15    37.890591
    16    25.600795         22    69.982452          6    58.119241          2   0.42404936E-01     19    47.414878
    27    49.446838         17    97.512271         30    50.828642         31    14.165511         32    22.937632
    33    43.856741         34    11.190555         35    59.905652         36    119.41687         37    102.27362
    38    51.716788         39    30.665312         40    245.42206         41    166.97197
    35  2  3  1.9E+01    69    69     2 8.0E-03  5.3E+00    0 -1.71771576E+03    90   211   0

BS and SB values...
    42    73.972039         58   -5.2654448         61   -2.5682646         62  -0.16728432         63   -2.7867210
    67  -0.60647820         66   -2.1401153         74    56.137643         75    13.878283         78    22.192570
    79    47.363993         80    49.948586         81    18.445444         82    125.74049         83    28.440734
    84    104.91220         85    111.36289         59  -0.40194666E-01      4    29.071571          5    32.079587
     7    8.3377634         11    19.135419         20    8.5031548         13    21.936304         15    37.890591
    16    25.641496         22    70.003691          6    58.261836         69  -0.79673371E-02     19    47.754405
    27    49.640191         17    97.512271         30    50.844068         31    14.169810         32    22.944594
    33    43.870051         34    11.204208         35    59.923833         36    119.46556         37    102.31329
    38    51.732484         39    30.674619         40    245.49904         41    167.01874
    36  2  1 -1.6E+00     8     8    84 1.1E+01  9.7E+00    0 -1.73541533E+03    93   205   0

BS and SB values...
    42    74.734257         58   -5.7633196         61   -3.0572299         62  -0.22767023         63   -2.7447659
    67  -0.76710671         66   -2.0410553         74    58.561675         75    15.834325         78    26.980221
    79    44.999501         80    50.283813         81    8.1862951         82    46.067006         83    29.707493
     8    10.810702         85    100.70581         59  -0.93344507E-01      4    25.728246          5    33.562251
     7    8.3377634         11    21.957156         20    18.409112         13    14.535085         15    19.765090
    16    25.641496         22    72.407425          6    63.953885         69  -0.24277516         19    84.257148
    27    71.522708         17    97.512271         30    52.589913         31    14.656362         32    23.732448
    33    45.376428         34    11.394026         35    61.981452         36    123.82348         37    106.23399
    38    53.800422         39    31.240632         40    253.85771         41    172.86485
Itn     36 -- toldj = 1.4E-01    Norm pi = 2.7E+01    wtobj =  0.0E+00
    37  2  2 -7.2E-01    12    12    81 8.8E+00  9.3E-01    0 -1.74179063E+03   106   211   0
```

```
BS and SB values...
    42    75.008804         58   -5.8485252         61   -3.1138168         62  -0.23319645         63   -2.7416705
    67  -0.77542609         66   -2.0343319         74    59.753844         75    16.071524         78    27.452407
    79    20.602823         80    22.130618         12    8.8244697         82    46.908762         83    30.350604
     8    12.037103         85    102.71486         59  -0.97292777E-01      4    25.801358          5    33.967228
     7    8.3377634         11    21.895450         20    22.234504         13    12.173748         15    17.708876
    16    25.641496         22    73.267913          6    65.081687         69  -0.25713935         19    85.225507
    27    79.356211         17    97.512271         30    53.214891         31    14.830538         32    24.014484
    33    45.915680         34    11.334606         35    62.718038         36    125.31046         37    107.51234
    38    54.511757         39    31.491624         40    256.84024         41    174.97282
    38  2  3  9.2E+00    65    65    80 5.7E-01  3.9E+01    0 -1.74701923E+03   122   218   0

BS and SB values...
    42    75.233969         58   -5.7994845         61   -3.0105644         62  -0.21747216         63   -3.7444906
    67  -0.16233374         66   -2.4685779         74    62.487622         75    15.680497         78    26.274937
    79    3.1384573         65  -0.57097039         12    16.020881         82    46.153623         83    31.906368
     8    6.4047842         85    99.897673         59  -0.81590772E-01      4    27.442716          5    34.128691
     7    8.3377634         11    20.510158         20    16.085913         13    15.242952         15    27.152165
    16    25.641496         22    73.944325          6    64.833478         69  -0.18151054         19    85.817639
    27    85.513958         17    97.512271         30    53.706172         31    14.967454         32    24.236186
    33    46.339575         34    10.582505         35    63.297053         36    126.38821         37    108.36109
    38    54.894250         39    31.984157         40    259.34318         41    176.38212
    39  2  4 -2.6E-01    76    76    59 1.1E+00 -7.2E-02    0 -1.74732058E+03   139   223   0

BS and SB values...
    42    75.246947         58   -5.1101026         61   -2.5864512         62  -0.15129457         63   -3.7444906
    67  -0.16233374         66   -2.4685779         74    64.981287         75    15.680497         78    26.274937
    79    3.1384573         65  -0.57097039         12    16.020881         82    46.153623         83    31.906368
     8    6.4047842         85    99.897673         76    1.1398674          4    27.442716          5    34.128691
     7    8.3377634         11    20.510158         20    16.085913         13    15.242952         15    27.152165
    16    25.641496         22    73.944325          6    64.833478         69  -0.69761358E-01     19    85.817639
    27    85.513958         17    97.512271         30    53.706172         31    14.115080         32    24.236186
    33    46.339575         34    10.582505         35    63.297053         36    126.22201         37    108.22832
    38    54.894250         39    31.984157         40    259.34318         41    176.38212
    40  2  2  3.8E+00    57    57    69 4.3E-02 -1.6E+00    0 -1.74748227E+03   146   223   0

BS and SB values...
    42    75.253910         58   -4.7940742         61   -2.3588820         62  -0.11578524         63   -3.7444906
    67  -0.16233374         66   -2.4685779         74    66.319329         75    15.680497         78    26.274937
    79    3.1384573         65  -0.57097039         12    16.020881         82    46.153623         83    31.906368
     8    6.4047842         85    99.897673         76    1.7514934          4    27.442716          5    34.128691
     7    8.3377634         11    20.510158         20    16.085913         13    15.242952         15    27.152165
    16    25.641496         22    73.944325          6    64.833478         57  -0.42783926E-01     19    85.817639
    27    85.513958         17    97.512271         30    53.706172         31    13.657716         32    24.236186
    33    46.339575         34    10.582505         35    63.297053         36    126.10986         37    108.15707
    38    54.894250         39    31.984157         40    259.34318         41    176.38212
Itn     40 -- toldj = 1.4E-02    Norm pi = 2.7E+01    wtobj =  0.0E+00
    41  2  2 -7.1E-02    18    18    62 7.5E+00 -1.5E-02    0 -1.74801187E+03   147   220   0

BS and SB values...
    42    75.276717         58   -3.3864298         61   -1.7009757         18    7.5017595         63   -3.2246033
    67  -0.51908674         66   -2.3360444         74    70.701714         75    18.221155         78    33.088242
    79    3.1384573         65  -0.19992316         12    16.020881         82    43.162962         83    29.972962
     8    6.4047842         85    91.877593         76    5.2665825          4    20.474360          5    34.818894
     7    8.3377634         11    20.510158         20    11.262593         13    15.242952         15    27.152165
    16    25.641496         22    73.944325          6    69.825840         57  -0.20781415         19    84.975163
    27    85.513958         17    97.512271         30    53.706172         31    11.029189         32    24.236186
    33    46.339575         34    11.881090         35    63.297053         36    125.90963         37    108.43449
    38    54.894250         39    31.984157         40    259.57641         41    176.01745
    42  2  4 -2.0E-02    81    81    67 6.2E+00 -8.4E-02    0 -1.74813357E+03   165   233   0

BS and SB values...
    42    75.281958         58   -3.3864298         61   -1.7009757         18    9.1452850         63   -4.0796392
    81    6.2025843         66   -2.7306601         74    71.708820         75    18.221155         78    33.088242
    79    4.7898927         65  -0.66897463         12    16.271186         82    41.142114         83    30.561666
     8   0.46731031E-01     85    85.882888         76    5.2665825          4    20.474360          5    34.818894
     7    8.3377634         11    19.221667         20    1.3809997         13    19.990407         15    37.812240
    16    25.641496         22    73.944325          6    69.825840         57  -0.20781415         19    84.627624
    27    85.513958         17    97.512271         30    53.706172         31    11.029189         32    24.236186
```

```
    33    46.339575         34    11.485683         35    63.297053         36    125.90963         37    108.43449
    38    54.723961         39    32.268725         40    259.78021         41    175.69878
    43  2  1 -1.3E-01     2     2     8 1.7E-01  2.7E-01    0 -1.74815528E+03   174   223   0

BS and SB values...
    42    75.282893         58   -3.3454310         61   -1.7092678         18    10.027340         63   -4.0759914
    81    6.2481727         66   -2.7363076         74    71.888478         75    18.600310         78    34.198715
    79    4.8020306         65  -0.66336770         12    16.273026         82    40.680950         83    30.476340
     2   0.17219406         85    84.531667         76    5.4154280          4    19.666244          5    34.946348
     7    8.3377634         11    19.212197         20    1.1748281         13    20.025300         15    37.890591
    16    25.476219         22    73.944325          6    69.964063         57  -0.21120645         19    84.469497
    27    85.513958         17    97.512271         30    53.706172         31    10.917885         32    24.236186
    33    46.339575         34    11.542993         35    63.297053         36    125.88593         37    108.46733
    38    54.722710         39    32.270816         40    259.80553         41    175.65920
    44  2  2  1.2E+01    62    62    20 1.8E-02  6.6E+01    0 -1.74837248E+03   178   221   0

BS and SB values...
    42    75.292246         58   -2.9735339         61   -1.9289802         18    21.347623         63   -4.0133176
    81    6.2481727         66   -2.7960010         74    73.685811         75    23.645667         78    49.071842
    79    4.8020306         65  -0.59046050         12    16.273026         82    34.741957         83    29.487629
     2    2.6404298         85    67.024206         76    7.0101472          4    9.1508757          5    36.667477
     7    8.3377634         11    19.212197         62  -0.17749052E-01     13    20.025300         15    37.890591
    16    23.107126         22    73.944325          6    71.180066         57  -0.23453390         19    82.368650
    27    85.513958         17    97.512271         30    53.706172         31    9.7253800         32    24.236186
    33    46.339575         34    12.207067         35    63.297053         36    125.57694         37    108.89545
    38    54.722710         39    32.270816         40    260.11112         41    175.18138
    45  2  3  1.2E+00    68    68     4 7.3E-02  1.3E+02    0 -1.74845764E+03   187   202   0

BS and SB values...
    42    75.295914         58   -2.9434045         61   -2.3232808         18    31.198967         63   -4.0659183
    81    6.2481727         66   -2.9567026         74    74.390501         75    28.239253         78    62.784136
    79    4.8020306         65  -0.59953877         12    16.273026         82    29.334753         83    28.919459
     2    5.2018205         85    50.883184         76    7.7765610         68  -0.72657263E-01      5    38.279105
     7    8.3377634         11    19.212197         62  -0.65434470E-01     13    20.025300         15    37.890591
    16    20.648621         22    73.944325          6    71.180066         57  -0.21702785         19    80.401464
    27    85.513958         17    97.512271         30    53.706172         31    9.1522682         32    24.236186
    33    46.339575         34    12.588683         35    63.297053         36    125.30690         37    109.26960
    38    54.722710         39    32.270816         40    260.38418         41    174.75443
    46  2  1 -2.1E-01     9     9    62 1.2E+00 -5.5E-02    0 -1.74870473E+03   193   201   0

BS and SB values...
    42    75.306555         58   -2.3476867         61   -1.9110649         18    38.439079         63   -4.1045765
    81    20.423430         66   -3.0748081         74    76.435170         75    27.827393         78    61.223081
    79    8.4457268         65  -0.60621072         12    17.517407         82    25.360802         83    28.501889
     2    4.3626929         85    39.020558         76    9.0377196         68  -0.12605574          5    38.048061
     7    8.3377634         11    13.536093          9    1.1925569         13    19.028371         15    37.890591
    16    18.841775         22    73.944325          6    73.327850         57  -0.29376125         19    78.955707
    27    85.513958         17    97.512271         30    53.706172         31    8.2091943         32    24.236186
    33    46.339575         34    12.869146         35    63.297053         36    125.30923         37    109.26638
    38    54.602132         39    32.472313         40    260.58485         41    174.44065
    47  2  2  3.9E+00    60    60    11 1.4E-01  9.6E+01    0 -1.74924826E+03   194   212   0

BS and SB values...
    42    75.329961         58   -1.0164055         61   -1.2005721         18    55.704942         63   -4.1967667
    81    54.227887         66   -3.3564604         74    80.932865         75    26.845208         78    57.500355
    79    17.135034         65  -0.62212167         12    20.484946         82    15.883918         83    27.506090
     2    2.3615826         85    10.731152         76    11.872331         68  -0.25339779          5    37.497077
     7    8.3377634         60  -0.14061788          9    4.0365078         13    16.650942         15    37.890591
    16    14.532899         22    73.944325          6    78.449780         57  -0.46465412         19    75.507936
    27    85.513958         17    97.512271         30    53.706172         31    6.0895179         32    24.236186
    33    46.339575         34    13.537982         35    63.297053         36    125.34649         37    109.21475
    38    54.314583         39    32.952832         40    261.06342         41    173.69237
LU file has increased by a factor of   2.0

Factor      4  Demand      2  Itn         47  Infeas       0
Nonlin      0  Linear     26  Slacks      18  Objective      -1.74924826E+03
m          44 =n          44  Elems      219  Amax   6.9E+00  Density  11.31
Merit     0.7  lenL       40  L+U        235  Cmpressns    0  Incres    7.31
Utri       20  lenU      195  Ltol   1.0E+02  Umax   1.6E+01  Ugrwth 2.4E+00
```

```
Ltri        4  dense1     13  Lmax   6.5E+00
bump       20  dense2      7  DUmax  4.4E+00  DUmin  7.3E-02  condU  6.1E+01
Itn     47 -- row check.   Max residual = 4.3E-13  on row   15.   Norm x = 2.61E+02

BS and SB values:

    42    75.329961         57  -0.46465412         58   -1.0164055         60  -0.14061788         61   -1.2005721
    63   -4.1967667         65  -0.62212167         66   -3.3564604         68  -0.25339779         74    80.932865
    75    26.845208         76    11.872331         78    57.500355         79    17.135034         81    54.227887
    82    15.883918         83    27.506090         85    10.731152          2    2.3615826          5    37.497077
     6    78.449780          7    8.3377634          9    4.0365078         12    20.484946         13    16.650942
    15    37.890591         16    14.532899         17    97.512271         18    55.704942         19    75.507936
    22    73.944325         27    85.513958         30    53.706172         31    6.0895179         32    24.236186
    33    46.339575         34    13.537982         35    63.297053         36    125.34649         37    109.21475
    38    54.314583         39    32.952832         40    261.06342         41    173.69237
Itn     47 -- feasible solution.  Objective =  -1.749248264E+03

   Itn ph pp     dj    +sbs  -sbs   -bs  step    pivot   ninf  sinf,objective     L     U ncp
    48  2  1 -1.4E-01     8     8    68 1.5E+00 -1.7E-01    0 -1.74944928E+03    40   195   0

BS and SB values...
    42    75.338618         57  -0.54639058         58  -0.37966625         60  -0.20787408         61  -0.86075003
    63   -3.7846667         65  -0.36688783         66   -2.9461821          8    1.4535788         74    82.596254
    75    26.375438         76    13.228099         78    55.719811         79    16.946581         81    53.545511
    82    16.667502         83    26.744655         85    13.486574          2    1.4044703          5    37.233547
     6    80.899551          7    8.3377634          9    4.0983984         12    20.492302         13    15.513840
    15    35.453483         16    15.315997         17    97.512271         18    55.704942         19    75.819394
    22    73.944325         27    85.513958         30    53.706172         31    5.0756965         32    24.236186
    33    46.339575         34    14.049404         35    63.297053         36    125.36431         37    109.19005
    38    54.347257         39    32.898231         40    260.99195         41    173.80413
    49  2  3  5.7E-01    67    67    58 1.9E-01 -2.0E+00    0 -1.74955588E+03    49   199   0

BS and SB values...
    42    75.343208         57  -0.59512728         67  -0.18752070         60  -0.24797670         61  -0.65812556
    63   -3.5103113         65  -0.19587918         66   -2.8372473          8    2.3202990         74    83.478357
    75    26.095330         76    14.036498         78    54.658131         79    16.834213         81    53.138633
    82    17.134727         83    26.226500         85    15.129540          2   0.83377639          5    37.076413
     6    82.360267          7    8.3377634          9    4.1353017         12    20.496687         13    14.835825
    15    34.000318         16    15.782931         17    97.512271         18    55.704942         19    76.005106
    22    73.944325         27    85.513958         30    53.706172         31    4.4711889         32    24.236186
    33    46.339575         34    14.397427         35    63.297053         36    125.37494         37    109.17533
    38    54.366739         39    32.865675         40    260.98026         41    173.82240
    50  2  2  1.3E-01    59    59    65 2.0E-01 -9.8E-01    0 -1.74958112E+03    59   192   0

BS and SB values...
    42    75.344295         57  -0.82032520         67  -0.40231336         60  -0.27342468         61  -0.53202477
    63   -3.1960552         59  -0.19954134         66   -2.7124697          8    3.3130701         74    83.687199
    75    25.774485         76    14.596073         78    53.442048         79    16.705502         81    52.672580
    82    17.669902         83    25.632988         85    17.011450          2   0.18008399          5    36.896426
     6    84.033421          7    8.3377634          9    4.1775720         12    20.501711         13    14.059202
    15    32.335812         16    16.317774         17    97.512271         18    55.704942         19    76.217827
    22    73.944325         27    85.513958         30    53.706172         31    4.0527475         32    24.236186
    33    46.339575         34    14.796063         35    63.297053         36    125.34957         37    109.21048
    38    54.389055         39    32.828383         40    260.96688         41    173.84332
Itn     50 -- toldj = 2.8E-03    Norm pi = 2.7E+01    wtobj =  0.0E+00
    51  2  1 -1.4E-02    10    10     2 5.1E+00  3.5E-02    0 -1.74965274E+03    71   189   0

BS and SB values...
    42    75.347380         57  -0.88236438         67  -0.41405424         60  -0.28043528         61  -0.49728560
    63   -3.1922581         59  -0.25451245         66   -2.7050038          8    2.6962966         74    84.279881
    75    25.686096         76    14.750229         78    53.107032         79    14.869519         81    44.471502
    82    20.020624         83    25.782305         85    25.332024         10    5.1215820          5    36.846842
     6    84.494354          7    8.3377634          9    3.3096528         12    19.846719         13    13.845253
    15    33.369911         16    18.391781         17    89.744497         18    55.704942         19    77.187781
    22    73.944325         27    85.513958         30    53.706172         31    3.9374722         32    24.236186
    33    46.339575         34    14.695774         35    63.297053         36    125.34258         37    109.22016
    38    54.359989         39    32.876955         40    261.05226         41    173.70982
    52  2  2 -4.3E-02    11    11     9 3.4E+00  9.6E-01    0 -1.74979926E+03    74   191   0
```

```
 BS and SB values...
     42    75.353689         57  -0.88236438         67  -0.48036342         60  -0.28043528         61  -0.49728560
     63   -3.1643234         59  -0.25451245         66   -2.6453566          8   0.81888258         74    85.492313
     75    25.686096         76    14.750229         78    53.107032         79    7.1970778         81    11.470033
     82    29.409328         83    26.491033         85    57.227450         10    20.371028          5    36.846842
      6    84.494354          7    8.3377634         11    3.4444802         12    17.141142         13    13.845253
     15    36.517631         16    25.641496         17    66.616049         18    51.311361         19    80.944369
     22    73.944325         27    85.513958         30    53.706172         31    3.9374722         32    24.236186
     33    46.339575         34    14.219751         35    63.297053         36    125.34258         37    109.22016
     38    54.345695         39    32.900842         40    261.19188         41    173.49151
     53  2  3  4.8E-01    65    65     8 1.1E-01  7.3E+00    0 -1.74985310E+03    79   196   0

 BS and SB values...
     42    75.356008         57  -0.88236438         67  -0.34423509         60  -0.28043528         61  -0.49728560
     63   -3.3530717         59  -0.25451245         66   -2.7348346         65  -0.11291848         74    85.937892
     75    25.686096         76    14.750229         78    53.107032         79    7.6961104         81    12.929983
     82    28.798666         83    26.751497         85    55.816431         10    22.607818          5    36.846842
      6    84.494354          7    8.3377634         11    1.5289731         12    17.272241         13    13.845253
     15    37.890591         16    25.641496         17    63.223567         18    53.754672         19    80.762343
     22    73.944325         27    85.513958         30    53.706172         31    3.9374722         32    24.236186
     33    46.339575         34    14.044809         35    63.297053         36    125.34258         37    109.22016
     38    54.279957         39    33.010695         40    261.30061         41    173.32150
     54  2  2 -3.0E-02    20    20    11 1.1E+00  1.4E+00    0 -1.74988646E+03    82   184   0

 BS and SB values...
     42    75.357445         57  -0.88236438         67  -0.28369619         60  -0.28043528         61  -0.49728560
     63   -3.4217831         59  -0.25451245         66   -2.7686149         65  -0.15880559         74    86.213936
     75    25.686096         76    14.750229         78    53.107032         79    7.9463458         81    13.507723
     82    28.492455         83    26.912858         85    55.258054         10    24.562594          5    36.846842
      6    84.494354          7    8.3377634         20    1.1122344         12    17.358638         13    13.310897
     15    37.890591         16    25.641496         17    60.258808         18    55.704942         19    80.642385
     22    73.944325         27    85.513958         30    53.706172         31    3.9374722         32    24.236186
     33    46.339575         34    13.936430         35    63.297053         36    125.34258         37    109.22016
     38    54.241674         39    33.074669         40    261.37269         41    173.20880
     55  2  1 -4.3E-03     1     1    65 3.2E+00 -5.0E-02    0 -1.74990013E+03    82   186   0

 BS and SB values...
     42    75.358033         57   -1.1901627         67  -0.45603618         60  -0.33542036         61  -0.23497024
     63   -3.1421892         59  -0.50536063         66   -2.6463668          1    3.1614936         74    86.327037
     75    25.351325         76    15.030346         78    52.068227         79    7.9463458         81    13.507723
     82    28.886520         83    26.620756         85    56.480853         10    24.562594          5    36.808317
      6    81.858622          7    8.3377634         20    3.6587205         12    17.358638         13    13.310897
     15    34.814543         16    25.641496         17    60.258808         18    55.704942         19    80.689409
     22    73.944325         27    85.513958         30    53.706172         31    3.7280053         32    24.236186
     33    46.339575         34    14.132623         35    63.297053         36    125.37453         37    109.17590
     38    54.241674         39    33.074669         40    261.32863         41    173.27770
 Itn     55 -- row check.   Max residual = 9.6E-14  on row   22.   Norm x = 2.61E+02
 Itn     55 --      2  nonbasics set on bound, basics recomputed

 Itn     55 -- feasible solution.  Objective =  -1.749900130E+03


 Biggest dj =  4.586E-03 (variable     14)   norm rg =  0.000E+00   norm pi =  2.735E+01
 Itn     55 -- row check.   Max residual = 9.6E-14  on row   22.   Norm x = 2.61E+02
1

 EXIT -- optimal solution found

 NEW BASIS file saved on file  11    itn =     55

 Problem name                KB2
 No. of iterations                55   Objective value     -1.7499001299E+03
 No. of degenerate steps           8   Percentage                     14.55
 Max x       (scaled)       40 2.6E+02   Max pi      (scaled)       12 2.7E+01
 Max x                      41 6.3E+03   Max pi                      8 1.7E+01
 Max Prim inf(scaled)        0 0.0E+00   Max Dual inf(scaled)       35 7.9E-15
 Max Primal infeas           0 0.0E+00   Max Dual infeas            35 7.6E-15
1
```

NAME KB2 OBJECTIVE VALUE -1.7499001299E+03

STATUS OPTIMAL SOLN ITERATION 55 SUPERBASICS 0

OBJECTIVE FAT7..J. (Min)
RHS
RANGES
BOUNDS 77BOUND

SECTION 1 - ROWS

NUMBER	...ROW..	STATE	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY	..I
42	FAT7..J.	BS	-1749.90013	1749.90013	None	None	-1.0	1
43	BAL...BW	EQ	.	.	.	.	17.26921	2
44	BHC...BW	EQ	.	.	.	.	17.21410	3
45	BLC...BW	EQ	.	.	.	.	16.94205	4
46	BLV...BW	EQ	.	.	.	.	16.65984	5
47	BN4...BW	EQ	.	.	.	.	12.00000	6
48	BP8...BW	EQ	.	.	.	.	17.20937	7
49	BTO...BW	EQ	.	.	.	.	17.42578	8
50	B3E...BW	EQ	.	.	.	.	16.59347	9
51	B3P...BW	EQ	.	.	.	.	16.46019	10
52	B3R...BW	EQ	.	.	.	.	16.46234	11
53	B3T...BW	EQ	.	.	.	.	16.50000	12
54	B3E.VOBW	EQ	.	.	.	.	14.49066	13
55	B3P.VOBW	EQ	.	.	.	.	15.04097	14
56	B3R.VOBW	EQ	.	.	.	.	15.14876	15
57	HMH.3EBW	BS	16.39029	-16.39029	.	None	.	16
58	HML.3EBW	LL	.	.	.	None	0.02037	17
59	HMM.3EBW	BS	6.80126	-6.80126	.	None	0.00000	18
60	HRH.3EBW	BS	5.03696	-5.03696	.	None	.	19
61	HRL.3EBW	BS	2.38322	-2.38322	.	None	.	20
62	HRM.3EBW	LL	.	.	.	None	0.01144	21
63	HMH.3RBW	BS	44.33774	-44.33774	.	None	.	22
64	HML.3RBW	LL	.	.	.	None	0.01172	23
65	HMM.3RBW	LL	.	.	.	None	0.00639	24
66	HRH.3RBW	BS	38.94628	-38.94628	.	None	.	25
67	HRL.3RBW	BS	5.84199	-5.84199	.	None	0.00000	26
68	HRM.3RBW	LL	.	.	.	None	0.00693	27
69	NOI.3EBW	LL	.	.	.	None	0.02790	28
70	NOI.3PBW	LL	.	.	.	None	0.02416	29
71	NOI.3RBW	LL	.	.	.	None	0.02237	30
72	WMO.3PBW	EQ	.	.	.	.	0.02029	31
73	WRO.3PBW	EQ	.	.	.	.	0.00652	32
74	XPB.3ABW	BS	-119.13097	119.13097	None	.	.	33
75	XCV.3EBW	BS	-94.80838	94.80838	None	.	.	34
76	XPB.3EBW	BS	-45.37646	45.37646	None	.	.	35
77	XRV.3EBW	UL	.	.	None	.	-0.07901	36
78	X12.3EBW	BS	-850.17588	850.17588	None	.	.	37
79	XCV.3PBW	BS	-30.89333	30.89333	None	.	.	38
80	XRV.3PBW	UL	.	.	None	.	-0.07782	39
81	X12.3PBW	BS	-181.08913	181.08913	None	.	.	40
82	XCV.3RBW	BS	-91.77415	91.77415	None	.	.	41
83	XPB.3RBW	BS	-62.84572	62.84572	None	.	.	42
84	XRV.3RBW	UL	.	.	None	.	-0.07927	43
85	X12.3RBW	BS	-783.45913	783.45913	None	.	.	44

1

SECTION 2 - COLUMNS

NUMBER	.COLUMN.	STATE	...ACTIVITY...	.OBJ GRADIENT.	..LOWER LIMIT.	..UPPER LIMIT.	REDUCED GRADNT	M+J
1	BAL.3EBW	BS	0.81182	.	.	None	0.00000	45
2	BHC.3EBW	LL	.	.	.	10.00000	0.06381	46
3	BLC.3EBW	LL	.	.	.	None	0.04896	47
4	BLV.3EBW	LL	.	.	.	None	0.10208	48
5	BN4.3EBW	BS	4.67255	.	.	None	0.00000	49
6	BP8.3EBW	BS	25.06112	.	.	None	0.00000	50

7	BTO.3EBW	BS	5.00000	.	.	None	0.00000	51
8	BAL.3PBW	LL	.	.	.	None	0.04643	52
9	BHC.3PBW	LL	.	.	.	None	0.03395	53
10	BLC.3PBW	BS	9.55097	.	.	None	0.00000	54
11	BLV.3PBW	LL	.	.	.	None	0.08880	55
12	BN4.3PBW	BS	2.50655	.	.	None	0.00000	56
13	BP8.3PBW	BS	8.77950	.	.	None	0.00000	57
14	BTO.3PBW	LL	.	.	.	None	0.00352	58
15	BAL.3RBW	BS	9.18818	.	.	None	0.00000	59
16	BHC.3RBW	BS	20.00000	.	.	None	0.00000	60
17	BLC.3RBW	BS	15.44903	.	.	None	0.00000	61
18	BLV.3RBW	BS	12.00000	.	.	None	0.00000	62
19	BN4.3RBW	BS	8.39159	.	.	None	0.00000	63
20	BP8.3RBW	BS	1.15938	.	.	None	0.00000	64
21	BTO.3RBW	LL	.	.	.	None	0.01634	65
22	D3T...BW	BS	122.57069	-16.50000	.	200.00000	0.00000	66
23	EAL...BW	UL	10.00000	.	.	10.00000	-17.26921	67
24	EHC...BW	UL	20.00000	.	.	20.00000	-17.21410	68
25	ELC...BW	UL	25.00000	.	.	25.00000	-16.94205	69
26	ELV...BW	UL	12.00000	.	.	12.00000	-16.65984	70
27	EN4...BW	BS	15.57069	12.00000	.	100.00000	0.00000	71
28	EP8...BW	UL	35.00000	.	.	35.00000	-17.20937	72
29	ETO...BW	UL	5.00000	16.00000	.	5.00000	-1.42578	73
30	M3..3TBW	BS	122.57069	.	.	None	0.00000	74
31	QPB73EBW	BS	15.05089	0.08757	.	None	0.00000	75
32	QVO73EBW	BS	35.54550	.	.	None	0.00000	76
33	QVO73PBW	BS	20.83702	.	.	None	0.00000	77
34	QPB73RBW	BS	49.67417	0.08757	.	None	0.00000	78
35	QVO73RBW	BS	66.18817	.	.	None	0.00000	79
36	WMO73EBW	BS	3214.88918	.	.	None	0.00000	80
37	WRO73EBW	BS	3597.51965	.	.	None	0.00000	81
38	WMO73PBW	BS	1770.36101	.	.	None	0.00000	82
39	WRO73PBW	BS	2009.74296	.	.	None	0.00000	83
40	WMO73RBW	BS	5651.99315	.	.	None	0.00000	84
41	WRO73RBW	BS	6262.64687	.	.	None	0.00000	85

```
Time for MPS input                      0.00 seconds
Time for solving problem                0.01 seconds
Time for solution output                0.00 seconds
Time for constraint functions           0.00 seconds
Time for objective function             0.00 seconds
Endrun
```