versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

MÉTODOS TIPO DUAL SIMPLEX PARA PROBLEMAS DE OTIMIZAÇÃO LINEAR CANALIZADOS E ESPARSOS

Carla Taviane Lucke da Silva
Marcos Nereu Arenales *
Inst. de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
São Carlos – SP
carla@icmc.usp.br
arenales@icmc.usp.br

Ricardo Silveira Sousa
Faculdade Redentor
Itaperuna – RJ
rsilveiras@hotmail.com

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 06/2006; aceito em 10/2007 após 1 revisão
Received June 2006; accepted October 2007 after one revision

Resumo

Os problemas de otimização linear canalizados e esparsos, objeto principal deste trabalho, surgem em várias aplicações, como por exemplo, problemas de planejamento da produção, problemas de mistura, entre outras. Métodos tipo dual simplex com busca linear por partes foram propostos e analisados em Sousa *et al.* (2005), com resultados efetivos para problemas densos pequenos e agora são analisados para problemas esparsos maiores. Algumas heurísticas de pivotamento foram implementadas para tentar manter a esparsidade e reduzir o tempo total de resolução dos problemas. Um conjunto de exemplos com estruturas esparsas que tipicamente ocorrem na prática foram gerados aleatoriamente para analisar o desempenho dos métodos. Os resultados computacionais demonstram a eficiência da abordagem.

Palavras-chave: otimização linear; otimização linear por partes; dualidade; esparsidade.

Abstract

Two-side constraint and sparse linear optimization problems, the main object of this work, appear in several applications, such as, production planning problems, mix problems among others. Dual Simplex-typed methods, called two-side constraint dual simplex methods with piecewise linear search were proposed and analyzed in Sousa *et al.* (2005), which showed effective results for dense and small problems and now they are analyzed for larger and sparse problems. These methods were implemented together with some pivoting heuristics to maintain sparsity and to reduce the running time. Sets of linear optimization randomly generated problems with sparse structures that occur in real world were used to analyze the performance of the methods. The computational results show the efficiency of the approaches.

Keywords: linear optimization; piecewise linear optimization; duality; sparsity.

1. Introdução

Em Sousa *et al.* (2005), foi estudado o problema de otimização linear com restrições canalizadas e desenvolvidos métodos tipo dual simplex com busca linear por partes, explorando a característica do problema dual ser linear por partes. Foram examinadas buscas unidimensionais exatas e inexatas, pois a simples aplicação do método simplex clássico é muito dispendiosa em relação ao número de iterações e o tempo de resolução. Neste trabalho, focaliza-se problemas de otimização linear canalizados e esparsos e algumas técnicas (heurísticas de pivotamento e formas de atualização da decomposição LU) baseadas em métodos diretos para a solução dos sistemas lineares básicos que surgem a cada iteração do método dual simplex linear por partes.

Maros (2003) aborda a heurística de Markowitz como uma ferramenta útil para uma implementação do método simplex para variáveis canalizadas. Vanderbei (1997) relata o uso das heurísticas de Grau Mínimo e de Markowitz, mas não apresenta nenhum experimento computacional comparativo entre tais técnicas.

A classe dos problemas de otimização linear esparsos e com restrições canalizadas é de grande interesse prático, uma vez que representa vários problemas reais, como por exemplo, problemas de planejamento e controle da produção, planejamento de recursos hídricos, transportes, fluxos em uma rede de telecomunicações, cálculo estrutural e muitos outros problemas industriais e econômicos. A forma canalizada (ou forma geral conforme Vanderbei (1997)) não só representa prontamente qualquer problema de otimização linear, como também surge espontaneamente na modelagem de problemas reais e após a aplicação de técnicas de pré-processamento para eliminar redundâncias, fixar variáveis, ou apertar limitantes.

A maioria dos problemas de otimização linear esparsos possui estrutura particular que pode ser explorada. Nestes casos, a exploração da estrutura particular leva a grandes economias, tanto de memória quanto de tempo de processamento, pois se evita armazenamento e operações com os elementos nulos. Outro modo de diminuir o tempo de resolução do problema consiste em reduzir o número de iterações do método (sem aumentar significativamente os cálculos intermediários), conforme exposto em Sousa *et al.* (2005).

A solução de um problema de grande porte exige muito esforço computacional e, ao utilizar a aritmética do ponto flutuante, problemas numéricos podem surgir. Não é verdade que todo problema de grande porte é numericamente difícil, mas a tendência é que sejam mais propensos a dificuldades numéricas que os problemas menores (Maros, 2003). Nos métodos tipo simplex, as fontes de imprecisão são, principalmente, as operações feitas com a inversa da matriz básica ou formas equivalentes. Em tais operações, os erros de arredondamento ocorrem e são acumulados, propagando-se de uma iteração para outra. As conseqüências podem ser uma solução numericamente inexata, uma base ótima errada, ou mesmo conclusões equivocadas, como por exemplo, problema factível declarado como infactível, problema ilimitado com solução ótima, ou vice-versa. Além disso, os cálculos podem se degenerar em qualquer momento se um elemento muito pequeno for erroneamente escolhido como pivô (quando, na verdade, era apenas um lixo numérico ao invés do zero).

A suscetibilidade dos métodos do tipo simplex em relação aos problemas numéricos tem motivado a pesquisa sobre algoritmos que são numericamente mais estáveis e também, eficientes (Maros, 2003). Neste aspecto, progressos notáveis têm sido alcançados ao se utilizar algumas técnicas para refatorar a matriz básica. Tais técnicas buscam produzir

soluções numericamente estáveis e, ao mesmo tempo, lidar com a esparsidade presente nos problemas. Algumas dessas técnicas são revisadas neste artigo.

O artigo está organizado da seguinte forma: na seção 2, são apresentados brevemente métodos do tipo dual simplex para resolver problemas de otimização linear canalizados e esparsos, enfocando principalmente a etapa de resolução dos sistemas de equações lineares que surgem a cada iteração. A seção 3 traz duas heurísticas de pivotamento e a seção 4 descreve duas formas de fazer a atualização da matriz básica. Na seção 5, são apresentados os experimentos computacionais realizados e a seção 6 contém as conclusões do artigo.

2. Métodos do tipo dual simplex e sistemas lineares esparsos

Os problemas de otimização linear com restrições canalizadas, ou na forma geral, podem ser formulados por:

$$\begin{aligned} &\text{Minimizar } f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{Sujeito a} \quad \boldsymbol{d} \le \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{e} \end{aligned} \tag{1}$$

em que $\boldsymbol{A} \in R^{m \times n}$; $\boldsymbol{d}$, $\boldsymbol{e} \in R^m$, com $d_i \le e_i$; $\boldsymbol{c}$, $\boldsymbol{x} \in R^n$. Sem perda de generalidade, supomos posto(**A**) = n (veja Sousa *et al.*, 2005).

O dual do problema (1) é dado por (Sousa *et al.*, 2005):

$$\begin{aligned} &\text{Maximizar } \boldsymbol{h}(\boldsymbol{\lambda}) = \sum_{i=1}^{m} \boldsymbol{h}_i(\boldsymbol{\lambda}_i) \\ &\text{Sujeito a} \quad \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{c} \end{aligned}$$

em que $h_i(\lambda_i)$ é uma função côncava linear por partes dada por:

$$h_i(\lambda_i) = \begin{cases} d_i\lambda_i, & \text{se } \lambda_i \ge 0. \\ e_i\lambda_i, & \text{se } \lambda_i \le 0. \end{cases}$$

O maior esforço computacional realizado pelos métodos do tipo simplex, consiste na resolução de sistemas lineares em cada iteração. No caso do método dual simplex é preciso resolver os três seguintes sistemas lineares (considerando a partição básica $\boldsymbol{A}^{\mathrm{T}} = (\boldsymbol{B}^{\mathrm{T}}\ \boldsymbol{N}^{\mathrm{T}})$, $\boldsymbol{B}$ é a matriz básica de dimensão $n \times n$):

$\boldsymbol{B}\hat{\boldsymbol{x}} = \hat{\boldsymbol{y}}_{\boldsymbol{B}}$ (solução básica primal, em que: $\hat{y}_{B_i} = e_{B_i}$ ou d_{B_i}).

$\boldsymbol{B}^{\boldsymbol{T}}\hat{\lambda}_{\boldsymbol{B}} = \boldsymbol{c}$ (solução básica dual).

$\boldsymbol{B}^{\boldsymbol{T}}\eta_{\boldsymbol{B}} = -\boldsymbol{a}^{\boldsymbol{T}}_{N_k}$ (direção dual simplex, N_k é o índice da equação primal violada).

Os cálculos realizados para a resolução de um dos sistemas poderão ser utilizados para a resolução dos outros, já que têm a mesma matriz de coeficientes $\boldsymbol{B}$ ou a sua transposta.

Diversos métodos podem ser utilizados para resolver tais sistemas. Implementações simples do método simplex trabalham com a inversa da matriz básica $\boldsymbol{B}$, produzindo resultados satisfatórios para problemas com até poucas centenas de restrições e variáveis. Se tal matriz inversa é calculada explicitamente, diz-se que está sendo usada a inversa explícita, ou se é armazenada como o produto de matrizes elementares, então, diz-se que a *forma produto da*

inversa é utilizada. Uma outra opção para resolver os sistemas lineares é a utilização da decomposição LU da matriz $\boldsymbol{B}$. É importante que sejam usados procedimentos eficientes para atualizar, de uma iteração para outra, a decomposição LU da matriz $\boldsymbol{B}$, uma vez que apenas uma coluna de $\boldsymbol{B}$ é alterada por iteração, conforme Bartels (1971) e Bartels & Golub (1969).

A decomposição LU é uma técnica muito utilizada pela maioria dos pacotes para otimização linear, equivalente à eliminação de Gauss, procedimento útil quando se trata de sistemas lineares gerais esparsos de grande porte e heurísticas para a escolha do pivô são usadas para minimizar preenchimentos (*fill-in*). A seguir são apresentadas duas delas.

3. Heurísticas de pivotamento

Existem diferentes estratégias que tentam minimizar o aparecimento de novos elementos não-nulos na operação de pivotamento. O surgimento desses novos elementos acarreta em demanda por mais memória para armazená-los e mais tempo computacional nas operações aritméticas em que são envolvidos. Fato que ocorre com muita freqüência nas operações de pivotamento na eliminação tradicional, pois 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 pivotamento são utilizadas com a decomposição LU com o objetivo de preservar a esparsidade existente nas matrizes básicas no método simplex. A primeira heurística de pivotamento descrita é a de Markowitz (1957). Esta heurística é bastante simples e, apesar de muitas outras heurísticas terem sido desenvolvidas e implementadas, ela ainda é considerada muito eficiente (Duff *et al.*, 1986). A segunda heurística de pivotamento é a Grau Mínimo (Vanderbei, 1997), que também é simples e de fácil implementação.

3.1 Heurística de Markowitz

A heurística de Markowitz consiste numa estratégia eficiente para manter a esparsidade na decomposição LU. Trata-se de uma das mais usadas na prática e apresenta resultados bem satisfatórios, apesar de ser computacionalmente cara (Duff *et al.*, 1986).

Considere a matriz original como *ativa* inicialmente: $b_{ij}^{(1)} = b_{ij}$. A heurística de Markowitz consiste em:

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. Trocar linhas e colunas para que o pivô $a_{pq}^{(t)}$ seja colocado na posição (t, t).

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

Exemplo 1: Considere a seguinte matriz $\boldsymbol{B}$ esparsa, quadrada e de ordem 5 a ser decomposta no produto LU utilizando a heurística de Markowitz:

$$\boldsymbol{B} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & & & 2 \\ 2 & & 4 & 1 & \\ 1 & -1 & & & 3 \\ & & 1 & & 1 \\ & 1 & & & 2 \end{bmatrix} \end{array}.$$

- $t = 1$, pivô na posição (1, 1):

1. Determine $r_i^{(1)}$, $i = 1,2,3,4,5$.

$$r_1^{(1)} = 3; \quad r_2^{(1)} = 3; \quad r_3^{(1)} = 3; \quad r_4^{(1)} = 2; \quad r_5^{(1)} = 2.$$

2. Determine $c_j^{(1)}$, $\quad j = 1,2,3,4,5$.

$$c_1^{(1)} = 3; \quad c_2^{(1)} = 3; \quad c_3^{(1)} = 2; \quad c_4^{(1)} = 1; \quad c_5^{(1)} = 4.$$

3. Determine: $\min_{i,j} \left\{ (r_i^{(1)} - 1)(c_j^{(1)} - 1) \;\; i = 1,2,3,4,5, \;\; j = 1,2,3,4,5 \right\} = (r_2^{(1)} - 1)(c_4^{(1)} - 1) = 0.$

4. Como $c_4^{(1)} = 1$, basta identificar o único elemento não nulo na coluna 4: a_{24}. Assim, a linha 2 deve ser trocada de posição com a linha pivô 1 e coluna 4 com a coluna pivô 1.

$$\begin{array}{c} \\ 2 \\ 1 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 4 & 2 & 3 & 1 & 5 \end{array} \\ \begin{bmatrix} 1 & & 4 & 2 & \\ & 2 & & 1 & 2 \\ & -1 & & 1 & 3 \\ & & 1 & & 1 \\ & 1 & & & 2 \end{bmatrix} \end{array}.$$

5. Eliminações não são necessárias e a nova matriz ativa, em destaque a seguir, é obtida prontamente.

$$\begin{array}{c} \\ 2 \\ 1 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 4 & 2 & 3 & 1 & 5 \end{array} \\ \begin{bmatrix} 1 & & 4 & 2 & \\ & 2 & & 1 & 2 \\ & -1 & & 1 & 3 \\ & & 1 & & 1 \\ & 1 & & & 2 \end{bmatrix} \end{array}.$$

- $t = 2$, pivô na posição (2, 2):

1. Determine $r_i^{(2)}$, $i = 1,3,4,5$.

$$r_1^{(2)} = 3; \quad r_3^{(2)} = 3; \quad r_4^{(2)} = 2; \quad r_5^{(2)} = 2.$$

2. Determine $c_j^{(2)}$, $j = 1,2,3,5$.

$$c_1^{(2)} = 2; \quad c_2^{(2)} = 3; \quad c_3^{(2)} = 1; \quad c_5^{(2)} = 4.$$

3. Determine: $\min_{i,j}\left\{(r_i^{(2)} - 1)(c_j^{(2)} - 1) \;\; i = 1,3,4,5, \;\; j = 1,2,3,5\right\} = (r_4^{(2)} - 1)(c_3^{(2)} - 1) = 0.$

4. Como $c_3^{(2)} = 1$, basta identificar o único elemento não nulo, na coluna 3: a_{43}. Assim, a linha 4 deve ser trocada de posição com a linha pivô 1 e coluna 3 deve ser trocada de posição com a coluna pivô 2.

	4	3	2	1	5
2	1	4		2	
4		1			1
3			-1	1	3
1			2	1	2
5			1		2

5. Eliminações não são necessárias e a nova matriz ativa é obtida prontamente.

	4	3	2	1	5
2	1	4		2	
4		1			1
3			-1	1	3
1			2	1	2
5			1		2

Continuando a realizar os passos de 1 a 4 da heurística de Markowitz para $t = 3$ e 4 obtém-se, finalmente, a seguinte matriz:

	4	3	2	1	5
2	1	4		2	
4	*1*	1			1
5			1		2
1			*-2*	1	-2
3			*-1*	*-1*	7

Observação: Os valores em itálico, abaixo da diagonal principal, são os multiplicadores utilizados para fazer as eliminações que fazem parte da matriz triangular inferior ***L***.

Para garantir estabilidade numérica, é importante que o elemento pivô não seja muito pequeno em relação ao tamanho dos outros elementos da submatriz ativa. Sabe-se que o pivotamento parcial produz, em geral, uma certa estabilidade numérica, mas quando a esparsidade está presente, apenas este método não é suficiente, pois ele não se preocupa com a geração de novos elementos não nulos. Então, para que a estabilidade seja

considerada na heurística de Markowitz, o candidato a pivô deve também satisfazer a desigualdade:

$$\left|b_{it}^{(t)}\right| \geq u \max_{p} \left|b_{pt}^{(t)}\right|, \; t = 1,..., n$$

em que u é um parâmetro no intervalo $0 < u \leq 1$.

Este procedimento para obter estabilidade pode não ser muito útil se os elementos de $\boldsymbol{B}$ são de magnitudes extremamente diferentes (Suhl & Suhl, 1990). Quando isso ocorre, faz-se uma mudança de escala nos elementos da matriz.

Finalmente, a decomposição LU da matriz $\boldsymbol{B}$ é a seguinte:

$$\boldsymbol{PBQ} = \boldsymbol{LU} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -2 & 1 & \\ & & -1 & -1 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 4 & & 2 & \\ & 1 & & & 1 \\ & & 1 & & 2 \\ & & & 1 & -2 \\ & & & & 7 \end{bmatrix} .$$

em que as matrizes $\boldsymbol{P}$ e $\boldsymbol{Q}$ são as matrizes de permutações de linhas e colunas respectivamente, realizadas durante a decomposição que podem ser armazenadas como vetores de inteiros.

3.2 Heurística de Grau Mínimo

A Grau Mínimo é uma outra heurística de pivotamento bastante simples que pode ser utilizada com o objetivo de manter a esparsidade presente. Esta heurística também apresenta bons resultados e reduz o trabalho de encontrar o elemento que será o pivô.

A heurística de Grau Mínimo consiste em:

1. Procurar entre as linhas da submatriz ativa aquela que é mais esparsa (a linha que possui menos elementos diferentes de zero), por exemplo, a linha i.
2. Trocar esta linha com a linha do pivô.
3. Para os elementos não nulos na linha escolhida, selecionar aquele cuja coluna é a mais esparsa (ou seja, a coluna que possui menos elementos não nulos na submatriz), por exemplo, a coluna j.
4. Trocar esta coluna com a coluna pivô.
5. Efetuar a eliminação Gaussiana.

Exemplo 2: Considere a mesma matriz $\boldsymbol{B}$ do exemplo 1 a ser decomposta no produto **LU** utilizando a heurística de Grau Mínimo:

$$\boldsymbol{B} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & & & 2 \\ 2 & & 4 & 1 & \\ 1 & -1 & & & 3 \\ & & 1 & & 1 \\ & 1 & & & 2 \end{bmatrix} \end{array} .$$

- Pivô na posição (1, 1):

1. As linhas 4 e 5 possuem o mesmo grau, ou seja, o número de elementos não nulos na submatriz é igual em ambas as linhas. Quando isso ocorre, escolhe-se a linha de menor índice, no caso é a linha 4.
2. A linha 4 deve ser trocada de posição com a linha 1.
3. Nesta linha, o elemento 1 na coluna 3 pertence à coluna que possui menos elementos não nulos.
4. A coluna 3 deve ser trocada com a coluna 1.

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 3 & 2 & 1 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & & 1 \\ 4 & & 2 & 1 & \\ & -1 & 1 & & 3 \\ & 2 & 1 & & 2 \\ & 1 & & & 2 \end{array}\right] \end{array}.$$

5. Efetuando a eliminação Gaussiana, tem-se:

$$\begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 3 & 2 & 1 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & & 1 \\ -4 & & 2 & 1 & -4 \\ & -1 & 1 & & 3 \\ & 2 & 1 & & 2 \\ & 1 & & & 2 \end{array}\right] \end{array}.$$

- Pivô na posição (2, 2):

1. A linha 5 possui menos elementos não nulos na submatriz ativa;
2. A linha 5 deve ser trocada de posição com a linha 2;
3. Nesta linha, o elemento 1, na coluna 2 pertence à coluna mais esparsa da submatriz.
4. Troca de colunas não é necessária.

$$\begin{array}{c} \\ 4 \\ 5 \\ 3 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccccc} 3 & 2 & 1 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & & 1 \\ & 1 & & & 2 \\ & -1 & 1 & & 3 \\ & 2 & 1 & & 2 \\ -4 & & 2 & 1 & -4 \end{array}\right] \end{array}.$$

5. Efetuando a eliminação Gaussiana, tem-se:

$$\begin{array}{c} \\ 4 \\ 5 \\ 3 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccccc} 3 & 2 & 1 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & & 1 \\ & 1 & & & 2 \\ & 1 & 1 & & 5 \\ & -2 & 1 & & -2 \\ -4 & & 2 & 1 & -4 \end{array}\right] \end{array}.$$

Continuando a realizar os passos de 1 a 4 da heurística de Grau Mínimo para t = 3 e 4 obtém-se a seguinte matriz:

$$\begin{array}{c} \\ 4 \\ 5 \\ 3 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} 3 & 2 & 1 & 5 & 4 \end{array} \\ \begin{bmatrix} 1 & & & 1 & \\ & 1 & & 2 & \\ & \textit{1} & 1 & 5 & \\ & \textit{-2} & \textit{-1} & \textit{-7} & \\ \textit{-4} & & \textit{-2} & \textit{-2} & 1 \end{bmatrix} \end{array}.$$

Finalmente, a decomposição LU da matriz $\boldsymbol{B}$ é:

$$\boldsymbol{PBQ} = \boldsymbol{LU} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & 1 & 1 & & \\ & -2 & -1 & 1 & \\ -4 & & -2 & -2 & 1 \end{bmatrix} . \begin{bmatrix} 1 & & & 1 & \\ & 1 & & 4 & \\ & & 1 & 5 & \\ & & & -7 & \\ & & & & 1 \end{bmatrix}.$$

Apesar da matriz $\boldsymbol{U}$, produzida pela heurística de Grau Mínimo (GM), ter apenas 3 não-nulos fora da diagonal principal (na heurística de Markowitz, a matriz $\boldsymbol{U}$ tem 5 elementos não-nulos) é preciso chamar a rotina de eliminação de Gauss 4 vezes, ao passo que a heurística de Markowitz chama esta rotina apenas 3 vezes. Para este exemplo, a GM teria o pior desempenho em termos do número de operações aritméticas. Por outro lado, no momento de fazer a atualização da matriz básica, em que a matriz $\boldsymbol{U}$ é utilizada, a situação pode se inverter.

A próxima seção descreve os procedimentos de atualização da base que foram utilizados nos experimentos computacionais reportados na seção 5.

4. Atualização da base

4.1 Decomposição LU de Bartels-Golub

Bartels & Golub (1969) propuseram uma técnica para atualizar a decomposição LU da matriz básica $\boldsymbol{B}$, a qual utiliza o pivotamento parcial, que consiste em escolher como pivô o maior elemento em módulo da coluna (Duff *et al.*, 1986), para evitar que os erros de arredondamento se propaguem de forma descontrolada (o pivotamento parcial é uma heurística para controlar o crescimento rápido dos erros de arredondamento, que funciona bem, em geral).

Considere o sistema linear $\boldsymbol{B}\hat{\boldsymbol{x}} = \hat{\boldsymbol{y}}_{\boldsymbol{B}}$ (os demais sistemas lineares que aparecem em cada iteração simplex são tratados de forma análoga). Para resolvê-lo suponha que a matriz básica $\boldsymbol{B}$ tenha sido decomposta no produto $\boldsymbol{LU}$.

Assim, $\boldsymbol{PB} = \boldsymbol{LU}$ para alguma matriz de permutação $\boldsymbol{P}$.

Portanto, a resolução do sistema $\boldsymbol{B}\hat{\boldsymbol{x}} = \hat{\boldsymbol{y}}_{\boldsymbol{B}}$ pode ser obtida resolvendo os dois sistemas triangulares:

$$\boldsymbol{L}\boldsymbol{z} = \boldsymbol{P}\hat{\boldsymbol{y}}_{\boldsymbol{B}} ;$$

$$\boldsymbol{U}\hat{\boldsymbol{x}} = \boldsymbol{z} .$$

Seja $\boldsymbol{B}^{(0)}$ a matriz básica da iteração inicial do método simplex. Para a próxima iteração será requerida a construção de uma nova matriz básica $\boldsymbol{B}^{(1)}$, dado que a ℓ-ésima variável deixa a base e a k-ésima variável entra na base. A nova matriz básica $\boldsymbol{B}^{(1)}$ na próxima iteração do método simplex terá a forma:

$$\boldsymbol{B}^{(1)} = \left[\boldsymbol{a}_{B_1} \cdots \boldsymbol{a}_{B_{\ell-1}} \boldsymbol{a}_{N_k} \boldsymbol{a}_{B_{\ell+1}} \cdots \boldsymbol{a}_{B_n} \right].$$

A posição em que a nova coluna $\boldsymbol{a}_{N_k}$ entra na base, não precisa ser necessariamente na ℓ-ésima posição. Um ordenamento pré-definido nas colunas originais pode ser conveniente para facilitar a decomposição LU (Oliveira & Cantane, 2007).

A decomposição de $\boldsymbol{B}^{(1)}$ a partir de $\boldsymbol{B}^{(0)}$ é particularmente fácil e pode ser também estável.

Se a ℓ-ésima coluna $\boldsymbol{a}_{N_k}$ de $\boldsymbol{B}^{(1)}$ for colocada na última posição da matriz e todas as colunas subseqüentes a ela forem movidas uma posição à esquerda tem-se:

$$\boldsymbol{B}^{(1)} = \left[\boldsymbol{a}_{B_1} \cdots \boldsymbol{a}_{B_{\ell-1}} \boldsymbol{a}_{B_{\ell+1}} \cdots \boldsymbol{a}_{B_n} \boldsymbol{a}_{N_k} \right].$$

Assim,

$$\boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{B}^{(1)} = \left[\boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{B_1} \ldots \boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{B_{\ell-1}} \boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{B_{\ell+1}} \ldots \boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{B_n} \boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{N_k} \right] = \left[\boldsymbol{u}_1 \boldsymbol{u}_2 \ldots \boldsymbol{u}_{\ell-1} \boldsymbol{u}_{\ell+1} \ldots \boldsymbol{u}_n \boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{N_k} \right] = \boldsymbol{H}^{(1)}. \quad (15)$$

em que $\boldsymbol{u}_i$ são as colunas da matriz $\boldsymbol{U}^{(0)}$, uma vez que $\boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{B}^{(0)} = \boldsymbol{U}^{(0)}$.

A matriz $\boldsymbol{H}^{(1)}$ tem a forma subtriangular superior (isto é, os elementos abaixo da subdiagonal inferior são nulos) com zeros na subdiagonal nas primeiras ℓ - 1 colunas como mostra a Figura 1.

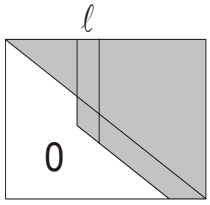


Figura 1 – Matriz $\mathbf{H}^{(1)}$ subtriangular superior.

É importante ressaltar que o vetor $\boldsymbol{L}^{-1}\boldsymbol{P}\boldsymbol{a}_{N_k}$ já foi obtido anteriormente quando a direção simplex foi calculada, antes da atualização da base. Assim, a matriz $\boldsymbol{H}^{(1)}$ pode ser construída sem qualquer esforço adicional.

A matriz $\boldsymbol{H}^{(1)}$ pode ser reduzida a uma matriz triangular superior $\boldsymbol{U}^{(1)}$ usando a eliminação de Gauss para zerar os elementos da subdiagonal nas colunas ℓ até n-1. O pivotamento parcial é realizado em cada passo da eliminação podendo ter ou não troca entre duas linhas adjacentes. As trocas são representadas pelas matrizes $\boldsymbol{P}_i^{(1)}$, $i = \ell, \ldots, n$-1, em que cada uma delas é simplesmente a matriz identidade ou a matriz identidade com a i-ésima e i+1-ésima linhas trocadas, a qual é simbolicamente armazenada.

Assim, $\boldsymbol{U}^{(1)}$ é obtida aplicando uma seqüência de operações elementares por linha em $\boldsymbol{H}^{(1)}$, ou seja,

$$\boldsymbol{U}^{(1)} = \boldsymbol{E}_{n-1}^{(1)}\boldsymbol{P}_{n-1}^{(1)}\cdots\boldsymbol{E}_{\ell}^{(1)}\boldsymbol{P}_{\ell}^{(1)}\boldsymbol{H}^{(1)} \tag{16}$$

em que cada $\boldsymbol{E}_i^{(1)}$ tem a forma:

$$\begin{array}{c} \\ i \\ i+1 \\ \\ \\ \\ \end{array}\!\!\begin{array}{c} \begin{array}{ccccccc} & & & i & i+1 & & \end{array} \\ \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & -g_i^{(1)} & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{bmatrix} \end{array}.$$

A escolha do pivô deve ser feita de forma que cada multiplicador seja menor ou igual a 1, ou seja, $\left|g_i^{(1)}\right| \le 1$.

Substituindo (16) em (15) e observando que $\left(\boldsymbol{P}_i^{(1)}\right)^{-1} = \boldsymbol{P}_i^{(1)}$ tem-se:

$$\boldsymbol{PB}^{(1)} = \boldsymbol{LP}_{\ell}^{(1)}\boldsymbol{E}_{\ell}^{(1)^{-1}}\cdots\boldsymbol{P}_{n-1}^{(1)}\boldsymbol{E}_{n-1}^{(1)^{-1}}\boldsymbol{U}^{(1)}.$$

Se outra iteração do método simplex for realizada, a matriz básica $\boldsymbol{B}^{(2)}$ resultante pode ser decomposta a partir da decomposição de $\boldsymbol{B}^{(1)}$, da mesma forma que $\boldsymbol{B}^{(1)}$ foi obtida a partir da decomposição de $\boldsymbol{B}^{(0)}$. Embora as matrizes $\boldsymbol{P}_i^{(t)}$ e $E_i^{(t)}$ possam ser armazenadas de forma bastante econômica, depois de muitas iterações do método simplex é conveniente refazer a decomposição LU. Para mais detalhes veja Silva (2002).

4.2 Variantes de Bartels-Golub

As duas variantes do algoritmo de Bartels-Golub tentam manter a esparsidade existente nas matrizes básicas e controlar o acúmulo dos erros de arredondamento. A segunda variante é um aperfeiçoamento da primeira e ambas foram propostas por Reid (1982).

4.2.1 Variante 1 – Algoritmo esparso de Bartels-Golub

Seja $\boldsymbol{B}^{(0)}$ a matriz básica inicial no método simplex. Aplicando a eliminação de Gauss em $\boldsymbol{B}^{(0)}$, com linhas e colunas permutadas, obtém-se a decomposição LU desta base, que pode ser expressa da seguinte forma:

$$\boldsymbol{E}_n^{(0)}\boldsymbol{E}_{n-1}^{(0)}...\boldsymbol{E}_1^{(0)}\boldsymbol{P}^{(0)}\boldsymbol{B}^{(0)}\boldsymbol{Q}^{(0)} = \boldsymbol{U}^{(0)}$$

em que, $\boldsymbol{E}_n^{(0)}\boldsymbol{E}_{n-1}^{(0)}...\boldsymbol{E}_1^{(0)} = (\boldsymbol{L}^{(0)})^{-1}$;

$\boldsymbol{E}_i$ é uma matriz elementar que difere da matriz identidade $\boldsymbol{I}$ em apenas um elemento fora da diagonal (os multiplicadores) e, portanto, representa uma seqüência de operações elementares por linha;

$\boldsymbol{P}$ e $\boldsymbol{Q}$ são matrizes de permutação.

Cada coluna da matriz $\boldsymbol{L}$ pode ser armazenada como um produto de matrizes elementares. Tal formação, pode simplificar bastante as manipulações 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 extra.

Denotando por $\boldsymbol{B}^{(1)}$ a matriz básica da iteração seguinte, que difere de $\boldsymbol{B}^{(0)}$ em apenas uma coluna, tem-se que $\boldsymbol{B}^{(1)}$ satisfaz a equação:

$$\boldsymbol{E}_n^{(0)}\boldsymbol{E}_{n-1}^{(0)}...\boldsymbol{E}_1^{(0)}\boldsymbol{P}^{(0)}\boldsymbol{B}^{(1)}\boldsymbol{Q}^{(0)} = \boldsymbol{S}^{(1)}$$

sendo que a matriz $\boldsymbol{S}^{(1)}$ difere da matriz $\boldsymbol{U}^{(0)}$ também por somente uma coluna, na mesma posição em que $\boldsymbol{B}^{(0)}$ e $\boldsymbol{B}^{(1)}$ diferem.

A estrutura da matriz $\boldsymbol{S}^{(1)}$ é mostrada na Figura 2.

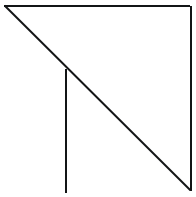


Figura 2 – Matriz $\boldsymbol{S}^{(1)}$ com coluna espeto.

De acordo com a Figura 2, a coluna da matriz $\boldsymbol{S}^{(1)}$, que difere da matriz $\boldsymbol{U}^{(0)}$, possui elementos não nulos abaixo da diagonal principal destruindo a triangularidade de $\boldsymbol{U}^{(0)}$. Tal coluna é chamada de espeto. Conforme a seção 4.1, Bartels e Golub sugeriram fazer permutações de colunas movendo a coluna espeto para a última coluna da matriz e empurrando as colunas posteriores a ela uma posição à esquerda, obtendo uma matriz na forma subtriangular superior, mostrada na Figura 3. Depois, deve-se realizar operações elementares por linha, com trocas de linhas adjacentes, se necessárias, para restaurar a forma triangular superior.

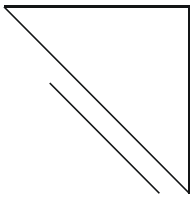


Figura 3 – Matriz subtriangular superior.

Se as operações elementares por linha são escritas como $\boldsymbol{E}_{n\text{-}1}^{(1)}\boldsymbol{E}_{m\text{-}2}^{(1)}...\boldsymbol{E}_{\ell}^{(1)}$, a nova fatoração da matriz básica é dada por:

$$\boldsymbol{E}_{n\text{-}1}^{(1)}...\boldsymbol{E}_{\ell}^{(1)}\boldsymbol{E}_{n\text{-}1}^{(0)}...\boldsymbol{E}_1^{(0)}\boldsymbol{P}^{(1)}\boldsymbol{P}^{(0)}\boldsymbol{B}^{(1)}\boldsymbol{Q}^{(0)}\boldsymbol{Q}^{(1)} = \boldsymbol{U}^{(1)}.$$

O objetivo principal desta variante é tentar manter a esparsidade presente nas matrizes básicas, algo que a decomposição Bartels e Golub não leva em consideração. Para isto, Reid

propôs que, ao fazer a decomposição LU da matriz básica inicial, deve-se utilizar uma heurística de pivotamento. A heurística que ele utilizou foi a heurística de Markowitz, pelo fato de ter obtido bons resultados com o uso desta heurística em trabalhos anteriores. A heurística de Grau Mínimo também pode ser utilizada.

Existe também a necessidade de se utilizar uma estratégia de pivotamento para manter a esparsidade no momento de reduzir a matriz na forma subtriangular superior à forma triangular superior, para evitar que preenchimentos excessivos ocorram. Todos os elementos da subdiagonal são não nulos, pois eles eram elementos da diagonal principal 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 de apenas trocas de linhas. Agora, se os elementos da diagonal e subdiagonal são não nulos, a princípio, deve ser escolhido como pivô o elemento que está na linha mais esparsa. Mas, por causa de instabilidade numérica, não é desejável ter um pivô muito pequeno. Dessa forma, o pivô escolhido será o maior entre os elementos da diagonal e subdiagonal, em módulo, se o menor deles é menor que o produto de uma constante u $(0 < u \leq 1)$ com o maior elemento (veja observação final na seção 3.1). Caso contrário, o pivô será o elemento que se encontra na linha mais esparsa.

Exemplo 3: Considere a seguinte matriz $\boldsymbol{S}^{(1)}$ a ser colocada na forma triangular superior (a coluna espeto é a segunda coluna):

$$\boldsymbol{S}^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left[\begin{array}{ccccccccc} 1 & -1 & & 3 & & & 1 & & 2 \\ & & 1 & & & 1 & & & \\ & & 2 & -1 & & 5 & & & \\ & & & 1 & & & 2 & & -1 \\ & & & & 1 & & -1 & 1 & \\ & & & & & 3 & & 4 & \\ & & & & & & 1 & & 1 \\ & 1 & & & & & & 5 & \\ & & & & & & & & 1 \end{array}\right] \end{array}.$$

1- *Colocar* $\boldsymbol{S}^{(1)}$ *na forma subtriangular superior*.

Colocando a coluna espeto (segunda coluna) na posição da coluna 9 e movendo as colunas 3, 4, 5, 6, 7, 8 e 9 uma posição para a esquerda, tem-se a seguinte matriz na forma subtriangular superior:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 2 \end{array} \\ \left[\begin{array}{ccccccccc} 1 & & 3 & & & 1 & & 2 & -1 \\ & 1 & & & 1 & & & & \\ & 2 & -1 & & 5 & & & & \\ & & 1 & & & 2 & & -1 & \\ & & & 1 & & -1 & 1 & & \\ & & & & 3 & & 4 & & \\ & & & & & 1 & & 1 & \\ & & & & & & 5 & & 1 \\ & & & & & & & 1 & \end{array}\right] \end{array}.$$

2- *Fazer as eliminações para colocar a matriz na forma triangular superior tentando preservar a esparsidade*.

Lembrando que, antes de fazer a eliminação propriamente dita, é preciso escolher qual será o elemento pivô levando em consideração tanto a questão da estabilidade numérica, quanto a questão da esparsidade que deve ser mantida.

Dessa forma, para a escolha do pivô têm-se duas opções:

(a) O elemento da diagonal principal é nulo e o elemento da subdiagonal é não nulo.
Neste caso, o pivô será o elemento da subdiagonal. Basta fazer a troca de linhas.

(b) Os elementos da diagonal principal e da subdiagonal são não nulos.
Neste caso, será escolhido como pivô o maior elemento entre os elementos da diagonal e subdiagonal, em módulo, se o menor deles for menor que o produto de u pelo maior elemento (u.maior). Caso contrário, o pivô será o elemento que se encontra na linha mais esparsa. Um valor típico para u é 0,1 (Reid, 1982).

- Pivô na posição (2, 2):

(b) Os elementos diagonal e subdiagonal são não nulos.

- ♦ maior = $|s_{32}| = 2$, menor = $|s_{22}| = 1$ (ambos servem como pivô, considerando u=0,1).
  $|s_{22}| = 1 > u.|s_{32}| = (0,1).(2) = 0,2$.
- ♦ Procurando a linha mais esparsa:
  Linha 2 tem 2 elementos não nulos.
  Linha 3 tem 3 elementos não nulos.

A linha 2 é mais esparsa. Portanto, não é preciso fazer troca de linhas, pois o pivô já está na posição correta. Anulando com uma operação elementar o elemento na posição (3, 2) da matriz, tem-se:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}\!\!\begin{array}{c} \begin{array}{ccccccccc} 1 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 2 \end{array} \\ \left[\begin{array}{ccccccccc} 1 & & 3 & & & 1 & & 2 & -1 \\ & 1 & & & 1 & & & & \\ & & -1 & & 3 & & & & \\ & & 1 & & & 2 & & -1 & \\ & & & 1 & & -1 & 1 & & \\ & & & & 3 & & 4 & & \\ & & & & & 1 & & 1 & \\ & & & & & & 5 & & 1 \\ & & & & & & & 1 & \end{array}\right] \end{array}.$$

- Pivô na posição (3, 3):

(b) Os elementos diagonal e subdiagonal são não nulos.

- ♦ maior = $|s_{33}| = 1$, menor = $|s_{43}| = 1$ (ambos servem como pivô).
  $|s_{43}| = 1 > u.|s_{33}| = (0,1).(1) = 0,1$.

♦ Procurando a linha mais esparsa:
Linha 3 tem 2 elementos não nulos.
Linha 4 tem 3 elementos não nulos.

A linha 3 é mais esparsa. Portanto, não é preciso trocar as linhas, pois o pivô já está na posição correta. Anulando o elemento na posição (4, 3) da matriz, tem-se:

	1	3	4	5	6	7	8	9	2
1	1		3			1		2	−1
2		1			1				
3			−1		3				
4					3	2		−1	
5				1		−1	1		
6					3		4		
7						1		1	
8							5		1
9								1	

.

- Pivô na posição (4, 4):

(a) O elemento diagonal é nulo e subdiagonal é não nulo.

Trocando a linha 5 com a linha 4 tem-se:

	1	3	4	5	6	7	8	9	2
1	1		3			1		2	−1
2		1			1				
3			−1		3				
5				1		−1	1		
4					3	2		−1	
6					3		4		
7						1		1	
8							5		1
9								1	

.

Realizando os mesmos passos para os pivôs nas posições (5, 5), (6, 6), (7, 7) e (8, 8), a matriz na forma triangular superior obtida é:

	1	3	4	5	6	7	8	9	2
1	1		3			1		2	−1
2		1			1				
3			−1		3				
5				1		−1	1		
6					3		4		
7						1		1	
4							−4	−3	
8								1	
9									3.75

.

4.2.2 Variante 2 – Aperfeiçoamento do algoritmo esparso de Bartels-Golub

A segunda variante do algoritmo de Bartels-Golub é um aperfeiçoamento da variante 1 descrita na seção anterior. Esta variante utiliza permutações de linhas e colunas adicionais antes de fazer as eliminações tentando evitá-las e, conseqüentemente, diminuir os preenchimentos, ou então, reduzir o número de elementos na subdiagonal inferior a serem eliminados.

Segundo Saunders (1976), alguma vantagem deveria ser tomada do fato que, no caso esparso, a coluna espeto também é bastante esparsa. Dessa forma, ao invés de colocar a coluna espeto na última posição da matriz e mover as colunas posteriores a ela uma posição à esquerda, (como é feito na variante 1), a coluna espeto deve ser colocada na posição da linha em que se encontra o último elemento desta coluna.

Reid (1982), aproveitando a sugestão de Saunders (1976) e tentando melhorar ainda mais esta idéia, sugeriu fazer mais algumas permutações de linhas e colunas antes de realizar as eliminações na matriz subtriangular.

Suponha que a coluna espeto esteja na coluna ℓ e que seu último elemento não nulo esteja na linha q. Considere a submatriz não triangular formada pelas linhas e colunas de ℓ a q. O objetivo, agora, é reduzir o tamanho desta submatriz não triangular para colocar a matriz original na forma triangular superior ou na forma subtriangular superior com menos elementos não nulos na subdiagonal e diminuir, com isso, o número de eliminações a serem realizadas.

O próximo passo é procurar, dentro desta submatriz não triangular e 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, ..., p-1 são movidas uma posição à direita e a p-ésima linha deve ser colocada na posição da linha ℓ e as demais linhas ℓ+1,..., p-1 são movidas uma posição para baixo. Com estas permutações, a forma da matriz foi preservada e o comprimento da coluna espeto, agora na coluna ℓ+1, foi reduzido por um, e com isso, 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 este procedimento, nenhuma das colunas ℓ+2, ..., p possui apenas um elemento não nulo e 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).

Suponha que a nova, 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 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, ..., 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, ..., q são movidas uma posição à esquerda. A nova procura por outras linhas que tenham um único 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 por 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 e 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 mais algumas permutações de colunas para colocar a submatriz não triangular novamente na forma subtriangular superior, como antes.

Neste momento, a única coluna na submatriz não triangular que pode ter apenas um elemento não nulo e 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 maneira descrita anteriormente. A última coluna da nova submatriz não triangular pode, de novo, possuir um único não nulo e na diagonal. Neste caso, o procedimento anterior deve ser repetido, ou seja, o elemento deve ser colocado na primeira posição da submatriz após as permutações necessárias. 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 juntas posteriormente).

Após a realização de todas permutações possíveis, a matriz $\boldsymbol{U}$ está pronta para que as eliminações sejam feitas se estas forem necessárias, com o objetivo de tornar a matriz $\boldsymbol{U}$ triangular superior novamente, caso isto ainda não tenha acontecido. Com a realização de todos estes passos, obtém-se a atualização da decomposição LU feita anteriormente, evitando uma nova decomposição da matriz básica, que representa um grande esforço computacional.

Exemplo 4: Considere a matriz $\boldsymbol{S}^{(1)}$ do exemplo anterior a ser colocada na forma triangular superior, agora, aplicando a variante 2:

$$\boldsymbol{S}^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} 1 & -1 & & 3 & & & 1 & & 2 \\ & & 1 & & & 1 & & & \\ & & 2 & -1 & & 5 & & & \\ & & & 1 & & & 2 & & -1 \\ & & & & 1 & & -1 & 1 & \\ & & & & & 3 & & 4 & \\ & & & & & & 1 & & 1 \\ & 1 & & & & & & 5 & \\ & & & & & & & & 1 \end{bmatrix} \end{array}.$$

A matriz $\boldsymbol{S}^{(1)}$ não está na forma triangular superior por causa da coluna 2, que é a coluna que acabou de entrar na base.

1- *Determinar a coluna espeto e a submatriz não triangular.*

A coluna espeto é coluna 2 e seu elemento não nulo está na linha 8. Assim, a submatriz não triangular vai ser formada pelas linhas e colunas de 2 até 8. Observe que a coluna espeto é a primeira coluna da submatriz não triangular.

	1	2	3	4	5	6	7	8	9
1	1	–1		3			1		2
2			1			1			
3			2	–1		5			
4				1			2		–1
5					1		–1	1	
6						3		4	
7							1		1
8		1						5	
9									1

2- *Procurar por colunas que possuam um único elemento não nulo e este elemento esteja na diagonal e fazer as permutações de colunas e linhas para colocar tal elemento no topo da submatriz.*

Analisando a submatriz não triangular a partir da sua segunda coluna, vê-se que a coluna 5 possui um único elemento não nulo e na diagonal. Colocando esta coluna na posição da coluna 2, movendo as colunas 2, 3 e 4 uma posição à direita e fazendo a mesma troca para as linhas (colocando a linha 5 na posição da linha 2 e movendo as linhas 2, 3 e 4 uma posição para baixo), tem-se:

	1	5	2	3	4	6	7	8	9
1	1		–1		3		1		2
5		1					–1	1	
2				1		1			
3				2	–1	5			
4					1		2		–1
6						3		4	
7							1		1
8			1					5	
9									1

Com estas permutações, a forma da submatriz foi preservada e o comprimento da coluna espeto, agora na terceira coluna da matriz, foi reduzido por um e, conseqüentemente, o tamanho da submatriz não triangular também foi reduzido.

Continuando a procura, na nova submatriz, por colunas que possuem um único elemento não nulo e na diagonal, primeiro entre as colunas 3 e 4 e depois entre as colunas 6, 7 e 8, vê-se que não existe mais nenhuma coluna com esta característica. Assim, pode-se ir para o próximo passo.

3- *Procurar por linhas que possuam um único elemento não nulo e que este elemento esteja na diagonal e fazer as permutações de linhas e colunas para colocar este elemento na última posição da submatriz não triangular.*

Iniciando a procura por linhas que possuam um único elemento não nulo na diagonal, a partir da penúltima até a segunda linha da submatriz não triangular, encontra-se a linha 7. Colocando esta linha na posição da linha 8, movendo a linha 8 uma posição para cima e realizando a mesma troca para as colunas (colocando a coluna 7 na posição da 8 e movendo a coluna 8 uma posição para a esquerda), tem-se:

$$\begin{array}{c|cccccccc} & 1 & 5 & 2 & 3 & 4 & 6 & 8 & 7 & 9 \\ 1 & 1 & & -1 & & 3 & & & 1 & 2 \\ 5 & & 1 & & & & & 1 & -1 & \\ 2 & & & & 1 & & 1 & & & \\ 3 & & & & 2 & -1 & 5 & & & \\ 4 & & & & & 1 & & & 2 & -1 \\ 6 & & & & & & 3 & 4 & & \\ 8 & & & 1 & & & & 5 & & \\ 7 & & & & & & & & 1 & 1 \\ 9 & & & & & & & & & 1 \end{array}.$$

Com estas permutações, a forma da matriz foi preservada e a ordem da submatriz não triangular pode ser reduzida por uma unidade retirando-se a linha e a coluna 7.

A nova procura por outras linhas que tenham um único não nulo na diagonal deve ser feita a partir da linha 6 até a linha 2. A linha 4 possui tal característica. Fazendo as trocas adequadas de linhas e colunas, tem-se:

$$\begin{array}{c|cccccccc} & 1 & 5 & 2 & 3 & 6 & 8 & 4 & 7 & 9 \\ 1 & 1 & & -1 & & & & 3 & 1 & 2 \\ 5 & & 1 & & & & 1 & & -1 & \\ 2 & & & & 1 & 1 & & & & \\ 3 & & & & 2 & 5 & & -1 & & \\ 6 & & & & & 3 & 4 & & & \\ 8 & & & 1 & & & 5 & & & \\ 4 & & & & & & & 1 & 2 & -1 \\ 7 & & & & & & & & 1 & 1 \\ 9 & & & & & & & & & 1 \end{array}.$$

Continuando a busca na submatriz triangular por linhas com um único elemento na diagonal a partir da linha 6 até a linha 2 nenhuma linha é encontrada. Então, pode-se ir ao próximo passo.

4- *Colocar a submatriz não triangular na forma subtriangular superior e verificar se a última coluna possui um único elemento não nulo e que esteja na diagonal. Se isto ocorrer, fazer a permutação simétrica para colocar tal elemento no topo da submatriz. Repetir este passo até que a última coluna não possua apenas um elemento não nulo na diagonal ou não se tenha mais a submatriz não triangular.*

Para deixar a submatriz não triangular na forma subtriangular superior é preciso colocar a coluna 2 na posição da coluna 8 e mover as colunas 3, 6 e 8 uma posição para a esquerda, obtendo-se:

$$
\begin{array}{c}
\\ 1 \\ 5 \\ 2 \\ 3 \\ 6 \\ 8 \\ 4 \\ 7 \\ 9
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc} 1 & 5 & 3 & 6 & 8 & 2 & 4 & 7 & 9 \end{array} \\
\left[\begin{array}{ccccccccc}
1 & & & & & -1 & 3 & 1 & 2 \\
 & 1 & & & 1 & & & -1 & \\
 & & 1 & 1 & & & & & \\
 & & 2 & 5 & & & -1 & & \\
 & & & 3 & 4 & & & & \\
 & & & & 5 & 1 & & & \\
 & & & & & & 1 & 2 & -1 \\
 & & & & & & & 1 & 1 \\
 & & & & & & & & 1
\end{array}\right]
\end{array}.
$$

A última coluna da submatriz não triangular (coluna 2) possui um único elemento não nulo na diagonal. Colocando a coluna 2 na posição da coluna 3, movendo as colunas 3, 6 e 8 uma posição à direita e fazendo a mesma troca para as linhas, (colocando a linha 8 na posição da linha 2 e movendo as linhas 2, 3 e 6 uma posição para baixo), tem-se:

$$
\begin{array}{c}
\\ 1 \\ 5 \\ 8 \\ 2 \\ 3 \\ 6 \\ 4 \\ 7 \\ 9
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc} 1 & 5 & 2 & 3 & 6 & 8 & 4 & 7 & 9 \end{array} \\
\left[\begin{array}{ccccccccc}
1 & & -1 & & & & 3 & 1 & 2 \\
 & 1 & & & & 1 & & -1 & \\
 & & 1 & & & 5 & & & \\
 & & & 1 & 1 & & & & \\
 & & & 2 & 5 & & -1 & & \\
 & & & & 3 & 4 & & & \\
 & & & & & & 1 & 2 & -1 \\
 & & & & & & & 1 & 1 \\
 & & & & & & & & 1
\end{array}\right]
\end{array}.
$$

Novamente, a última coluna da submatriz não triangular (coluna 8) possui um único elemento não nulo e na diagonal. Realizando as movimentações de colunas e linhas necessárias, tem-se:

$$
\begin{array}{c}
\\ 1 \\ 5 \\ 8 \\ 6 \\ 2 \\ 3 \\ 4 \\ 7 \\ 9
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc} 1 & 5 & 2 & 8 & 3 & 6 & 4 & 7 & 9 \end{array} \\
\left[\begin{array}{ccccccccc}
1 & & -1 & & & & 3 & 1 & 2 \\
 & 1 & & 1 & & & & -1 & \\
 & & 1 & 5 & & & & & \\
 & & & 4 & & 3 & & & \\
 & & & & 1 & 1 & & & \\
 & & & & 2 & 5 & & & \\
 & & & & & & 1 & 2 & -1 \\
 & & & & & & & 1 & 1 \\
 & & & & & & & & 1
\end{array}\right]
\end{array}.
$$

Neste momento, a submatriz não triangular não possui colunas com somente um elemento não nulo e na diagonal. Isso significa que o processo terminou. A matriz resultante está praticamente na forma triangular superior, faltando eliminar apenas um elemento abaixo da diagonal principal.

5- *Fazer as eliminações necessárias para colocar a matriz na forma triangular superior.*

Realizando uma única eliminação na matriz, obtém-se a atualização da matriz $\boldsymbol{U}$:

$$\boldsymbol{U}^{(1)} = \begin{bmatrix} 1 & & -1 & & & & 3 & 1 & 2 \\ & 1 & & 1 & & & & -1 & \\ & & 1 & 5 & & & & & \\ & & & 4 & & 3 & & & \\ & & & & 1 & 1 & & & \\ & & & & & 3 & & & \\ & & & & & & 1 & 2 & -1 \\ & & & & & & & 1 & 1 \\ & & & & & & & & 1 \end{bmatrix}.$$

5. Experimentos Computacionais

5.1 Estrutura de dados

A estrutura de dados utilizada foi baseada em Reid (1982). Para armazenar uma matriz esparsa são usados 2 vetores de ponteiros (*rows* e *cols*) alocados dinamicamente, de tamanhos iguais, respectivamente, ao número de linhas (*nr*) e ao número colunas (*nc*) da matriz. Um destes vetores contém ponteiros que apontam para listas (*list*) duplamente encadeadas que representam as linhas ou colunas da matriz e o outro vetor, também contém ponteiros que apontam para listas duplamente encadeadas que representam as colunas da matriz. Estas listas, por sua vez, contêm ponteiros que apontam para o primeiro elemento não nulo de cada linha ou coluna (*head*) e um inteiro (*count*) que indica quantos elementos não nulos existem na linha ou coluna. E finalmente, cada elemento não nulo da matriz é representado por um nó (*list_element*) que contém um ponteiro que aponta para o elemento anterior (*prev*), um ponteiro que aponta para o próximo elemento (*next*), um ponteiro para o valor do elemento (*value*) e um inteiro (*pos*) que guarda a posição que o elemento ocupa na linha ou coluna.

A Figura 4 exibe o esquema de representação da matriz, lista e nó da estrutura de dados utilizada quando os elementos são acessados por coluna. A representação por linha é semelhante.

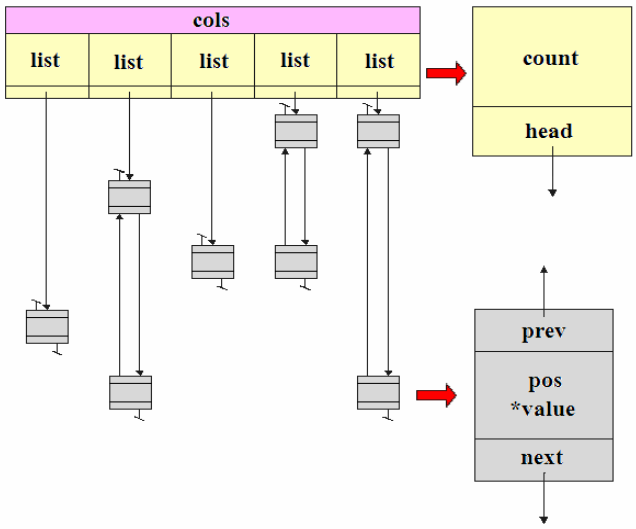
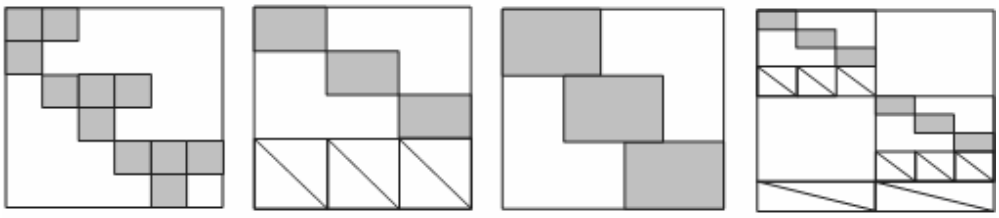


Figura 4 – Representação da matriz, lista e nó acessando os elementos por coluna.

5.2 Geradores

Nesta seção, são apresentados os quatro geradores implementados para obter os exemplares de problemas de otimização linear canalizados e esparsos utilizados nos testes computacionais: produção (a), bloco angular (b), bloco escada (c) e bloco angular em vários níveis (d). As estruturas das matrizes dos coeficientes criadas pelos geradores estão representadas na Figura 5.

- **Gerador Produção:** Um problema na literatura de Pesquisa Operacional muito pesquisado é o problema de dimensionamento de lotes, que consiste basicamente em determinar o quanto e quando produzir de certos produtos manufaturados (itens) em um horizonte de planejamento finito dividido em períodos. A matriz dos coeficientes do modelo matemático que representam estes problemas tem a forma da Figura 5 (a). Para gerar exemplares desse tipo de problema, deve-se informar o número de tipos de itens a ser produzido e o número de períodos em que a produção será feita.
- **Gerador Bloco Angular:** A matriz dos coeficientes do modelo matemático do problema de corte de estoque unidimensional com vários objetos em estoque e em quantidades limitadas tem a forma bloco angular conforme a Figura 5 (b). Para gerar exemplares deste tipo de problema, deve-se informar a quantidade de blocos e o número de linhas e colunas de cada bloco. O número de linhas de acoplamento (linhas abaixo do último bloco angular) é igual ao número de colunas de cada bloco.
- **Gerador Escada:** O modelo matemático do problema de corte de estoque unidimensional multiperíodo é representado matricialmente por matrizes que tem a forma bloco angular em vários níveis conforme a Figura 5 (c). Para gerar exemplares deste tipo de problema, deve-se informar a quantidade de blocos, o número de linhas e colunas de cada bloco e também o número de colunas de interseção.
- **Gerador Bloco Angular em Vários Níveis:** O modelo matemático do problema de rotações de culturas é representado matricialmente por matrizes que tem a forma bloco angular em vários níveis conforme a Figura 5 (d). Para gerar exemplares deste tipo de problema, deve-se informar a quantidade de blocos maiores, a quantidade de blocos menores e também o número de linhas e colunas de cada bloco menor. O número de linhas de acoplamento é igual ao número de blocos menores de cada bloco maior.



(a) produção (b) bloco angular (c) escada (d) bloco angular 2-níveis

Figura 5 – Representação matricial dos problemas gerados.

5.3 Resultados computacionais

Todas as implementações (geradores e método dual simplex com busca linear por partes, heurísticas de pivotamento e formas de atualização) foram feitas em linguagem C.

Os testes foram realizados usando os problemas obtidos aleatoriamente pelos quatro geradores descritos anteriormente. O número de linhas (m) e colunas (n) varia entre 50 e 3000 aproximadamente e a densidade (nz) varia entre 0,2% e 6% para problemas na forma geral. São 12 classes problemas com o número de colunas maior que o número de linhas ($n > m$) e 12 classes de problemas com o número de linhas maior que o número de colunas ($m > n$). Para cada classe foram resolvidos 10 problemas e calculada a média aritmética de alguns resultados, tais como, número de iterações, tempo computacional, entre outros.

Todos os testes foram realizados em um computador AMD Atlhon 1800+ (1500 MHz) com 1GB de RAM e um *hard disk* de 40GB utilizando o sistema operacional Windows 2000 e o compilador C++ da Intel.

Os primeiros testes foram realizados com os problemas construídos pelo gerador escada, os quais foram resolvidos algumas vezes mudando-se apenas os valores para o número de atualizações a serem feitas antes de uma nova decomposição LU da matriz básica. Isto porque era importante saber o quanto o número de atualizações realizadas influencia, principalmente, no tempo de resolução e, assim, poder determinar um valor a ser usado nos demais testes. Os valores considerados foram: 100, $\sqrt{m}$, $3\sqrt{m}/2$ e $0{,}1m$. Os itens analisados foram: número de iterações, tempo total de resolução, preenchimento por atualização, tempo por atualização, preenchimento por decomposição e tempo por decomposição.

A Tabela 1 exibe o desempenho dos quatro valores utilizados sendo do melhor para o pior desempenho. O valor 100 apresentou melhores resultados em quase todos os itens analisados para a maioria dos problemas nos casos em que $n > m$ e $m > n$.

Outros testes foram feitos para valores menores que $\sqrt{m}$ e maiores que $0{,}1m$ e os resultados obtidos foram piores que estes apresentados para a maioria dos itens analisados.

Tabela 1 – Valores para o número de atualizações.

Classificação dos valores para o número de atualizações								
	n > m				***m > n***			
	1º	**2º**	**3º**	**4º**	**1º**	**2º**	**3º**	**4º**
Número de iterações	100	$\sqrt{m}$	$0{,}1m$	$3\sqrt{m}/2$	100	$0{,}1m$	$\sqrt{m}$	$3\sqrt{m}/2$
Tempo de resolução	100	$0{,}1m$	$3\sqrt{m}/2$	$\sqrt{m}$	$\sqrt{m}$	$3\sqrt{m}/2$	$0{,}1m$	100
Preenchimento/atualização	$\sqrt{m}$	$3\sqrt{m}/2$	100	$0{,}1m$	100	$3\sqrt{m}/2$	$\sqrt{m}$	$0{,}1m$
Tempo/atualização	100	$3\sqrt{m}/2$	$(0{,}1)(m)$	$\sqrt{m}$	todos			
Preenchimento/decomposição	100	$\sqrt{m}$	$3\sqrt{m}/2$	$0{,}1m$	100	$3\sqrt{m}/2$	$\sqrt{m}$	$0{,}1m$
Tempo/decomposição	100	$3\sqrt{m}/2$	$0{,}1m$	$\sqrt{m}$	100	$3\sqrt{m}/2$	$\sqrt{m}$	$0{,}1m$

A seguir, foram realizados novos testes utilizando os problemas construídos pelos quatro geradores com o objetivo de identificar, entre as combinações das heurísticas de pivotamento (Markowitz (M) e Grau Mínimo (GM) descritas nas seções 3.1 e 3.2) com as duas formas de atualizações (Variante 1 e Variante 2 descritas nas seções 4.2.1 e 4.2.2), as combinações que apresentam melhor e pior desempenho em relação aos itens analisados anteriormente.

De acordo com os resultados a combinação heurística de Grau Mínimo com variante 2 foi a melhor em relação ao tempo de decomposição e a combinação heurística de Markowitz com variante 1 foi a pior. Para o tempo de resolução e de atualização a melhor combinação foi heurística de Markowitz com variante 2 e a pior combinação foi heurística de Markowitz com variante 1. Os preenchimentos por atualização e decomposição, em geral, foram menores com a combinação heurística de Markowitz com variante 2 e maiores com a combinação heurística de Grau Mínimo com variante 1. De um modo geral, pode-se dizer que a combinação heurística de Markowitz com variante 2 apresentou melhores resultados na maioria dos itens analisados, considerando os casos em que $n > m$ e $m > n$ e em relação aos quatro geradores utilizados. As combinações heurísticas de Markowitz com variante 1 e heurística de Grau Mínimo com variante 2 apresentaram os piores resultados na maioria dos itens considerados para os quatro geradores, sendo que a primeira obteve desempenho ligeiramente inferior.

Entre os itens analisados, o mais importante para este trabalho foi o tempo total de resolução. Mas, não se pode esquecer que os demais itens influenciam diretamente neste e por eles é possível identificar onde esta abordagem deve ser melhorada.

Nas Tabelas 2 e 3, estão os tempos de resolução e nas Tabelas 4 e 5 os preenchimentos ocorridos por atualização dos problemas construídos pelo gerador bloco angular para cada combinação entre as heurísticas de pivotamento e formas de atualização.

Tabela 2 – Tempo total de resolução de problemas usando gerador bloco angular (caso $n>m$).

Tempo Total de Resolução (em segundos)						
n* > *m			**Grau Mínimo**		**Markowitz**	
Problemas	**Iterações**	**%nz**	**Variante 1**	**Variante 2**	**Variante 1**	**Variante 2**
60x400	78,70	5,00	0,62	**0,39**	0,63	0,39
90x400	116,30	3,33	0,99	0,82	0,96	**0,80**
158x400	159,60	2,53	1,50	1,12	1,33	**1,03**
204x400	87,00	1,47	0,61	0,25	0,61	**0,25**
120x1000	324,63	2,50	65,34	71,32	**29,30**	49,83
258x1000	175,80	1,16	11,20	11,04	**10,33**	11,03
504x1000	210,70	0,60	11,40	9,60	12,01	**7,60**
220x2000	377,60	1,36	421,24	374,42	317,62	**317,30**
508x2000	308,60	0,59	269,68	192,38	268,04	**115,57**
1004x2000	395,80	0,30	305,86	163,26	347,30	**93,81**
320x3000	448,75	0,94	1164,75	1081,82	981,26	**866,42**
762x3000	740,75	0,52	1862,50	1419,5	1514,00	**1105,15**

Tabela 3 – Tempo total de resolução de problemas usando gerador bloco angular (caso $m>n$).

Tempo Total de Resolução (em segundos)						
m* > *n			**Grau Mínimo**		**Markowitz**	
Problemas	**Iterações**	**%nz**	**Variante 1**	**Variante 2**	**Variante 1**	**Variante 2**
402x50	77,30	4,23	0,03	0,02	0,03	**0,01**
402x100	134,70	2,24	0,12	**0,06**	0,11	**0,06**
403x150	176,80	2,23	0,40	**0,20**	0,41	**0,20**
402x200	197,70	1,24	0,39	**0,23**	0,40	**0,23**
1002x100	172,20	2,10	0,16	**0,09**	0,16	**0,09**
1002x250	336,00	0,90	1,42	**0,73**	1,42	**0,73**
1002x500	418,20	0,50	5,72	2,38	5,81	**2,27**
2002x200	357,40	1,05	1,19	**0,71**	1,13	**0,71**
2002x500	662,00	0,45	10,24	**4,04**	10,34	4,25
2002x1000	856,40	0,25	57,57	27,21	59,15	**26,16**
3002x300	573,60	0,70	4,04	2,30	3,88	**2,25**
3003x750	1269,00	0,43	72,31	**34,60**	62,44	39,74

Tabela 4 – Preenchimento por atualização usando gerador bloco angular (caso $n>m$).

Preenchimento por atualização						
n* > *m			**Grau Mínimo**		**Markowitz**	
Problemas	**Iterações**	**%nz**	**Variante 1**	**Variante 2**	**Variante 1**	**Variante 2**
60x400	78,70	5,00	10,98	**10,69**	10,98	**10,69**
90x400	116,30	3,33	14,93	13,27	10,27	**8,59**
158x400	159,60	2,53	17,29	15,31	9,62	**7,27**
204x400	87,00	1,47	7,45	**6,89**	7,45	**6,89**
120x1000	324,63	2,50	58,94	61,53	16,69	**13,77**
258x1000	175,80	1,16	23,60	24,66	11,72	**10,56**
504x1000	210,70	0,60	11,40	15,45	7,91	**7,36**
220x2000	377,60	1,36	89,94	86,71	19,29	**15,51**
508x2000	308,60	0,59	43,75	43,24	13,21	**12,01**
1004x2000	395,80	0,30	29,54	29,40	8,24	**7,87**
320x3000	448,75	0,94	112,26	118,99	20,63	**17,16**
762x3000	740,75	0,52	89,66	87,15	15,65	**12,41**

Tabela 5 – Preenchimento por atualização usando gerador bloco angular (caso $m>n$).

Preenchimento por atualização						
m* > *n			**Grau Mínimo**		**Markowitz**	
Problemas	**Iterações**	**%nz**	**Variante 1**	**Variante 2**	**Variante 1**	**Variante 2**
402x50	77,30	4,23	1,85	**0,75**	1,85	0,76
402x100	134,70	2,24	4,19	2,81	3,39	**1,80**
403x150	176,80	2,23	4,08	3,30	3,06	**2,06**
402x200	197,70	1,24	7,38	4,79	6,69	**3,09**
1002x100	172,20	2,10	3,05	1,68	2,52	**1,15**
1002x250	336,00	0,90	5,46	4,86	3,09	**2,19**
1002x500	418,20	0,50	7,26	6,89	3,13	**2,82**
2002x200	357,40	1,05	4,86	4,37	2,24	**1,30**
2002x500	662,00	0,45	8,31	8,27	2,98	**2,49**
2002x1000	856,40	0,25	13,37	13,24	3,55	**3,01**
3002x300	573,60	0,70	6,22	5,43	2,32	**0,93**
3003x750	1269,00	0,43	18,22	16,77	4,79	**3,07**

De acordo com as Tabelas 2 e 3, para a maioria dos problemas nos dois casos em que $n > m$ e $m > n$ o tempo total de resolução das combinações Grau Mínimo com variante 1 e Markowitz com variante 1 são próximos. O mesmo acontece com as combinações Grau Mínimo com variante 2 e Markowitz com variante 2. Estas duas últimas combinações apresentaram melhores desempenhos e a combinação Markowitz com variante 2 ligeiramente melhor.

Os resultados das combinações da heurística de Markowitz com as variantes 1 e 2 são melhores e mais estáveis que os resultados obtidos com as combinações Grau Mínimo com as variantes 1 e 2 em relação aos preenchimentos por atualização. E de acordo com as Tabelas 4 e 5 a melhor combinação para este item foi heurística de Markowitz com variante 2.

Os resultados obtidos com os problemas construídos pelos demais geradores foram semelhantes a estes (Silva, 2002).

A implementação do método dual simplex com busca linear por partes que utiliza técnicas de esparsidade (coluna "Esparsa" da Tabela 6) foi comparada, em relação ao tempo total de resolução, com uma implementação do mesmo método (Sousa, 2000), que utiliza estruturas de dados estáticas e não utiliza técnicas de esparsidade (coluna "Densa" da Tabela 6). Para os testes foram utilizadas as melhores combinações entre as heurísticas de pivotamento e as duas formas de atualização para cada gerador. Os problemas construídos pelos quatro geradores possuem valores para m e n inferiores a 400, uma vez que a implementação feita por Sousa (2000) é limitada com relação ao tamanho das matrizes (ordem inferior a 500).

Na Tabela 6, são apresentados somente os resultados obtidos para os problemas do gerador bloco escada. Os resultados obtidos com os demais geradores foram semelhantes a estes.

Tabela 6 – Esparsa x Densa – Tempo de Resolução (gerador escada).

Tempo total de resolução (em segundos)									
n > m					***m > n***				
Problemas	**Iterações**	**%nz**	**Esparsa**	**Densa**	**Problemas**	**Iterações**	**%nz**	**Esparsa**	**Densa**
50x101	49,70	4,95	**0,02**	0,22	100x51	66,20	5,88	**0,01**	0,05
50x201	62,90	4,48	**0,03**	1,92	200x51	86,20	5,88	**0,02**	0,08
100x201	102,30	2,49	**0,09**	3,22	200x101	131,40	2,97	**0,03**	0,58
50x401	64,50	4,25	**0,14**	18,62	400x51	96,70	5,88	**0,02**	0,08
100x401	121,30	2,24	**0,34**	34,89	400x101	163,50	2,97	**0,05**	0,78
200x401	223,10	1,25	**0,66**	63,95	400x201	247,30	1,49	**0,20**	8,33

De acordo com a Tabela 6, os problemas cujas matrizes dos coeficientes estão na forma escada foram resolvidos em um tempo muito menor pelo método dual que explora a esparsidade, em alguns casos mais de 100 vezes mais rápido.

5.3.1 Extensões na busca unidimensional

No trabalho de Sousa *et al*. (2005) foram desenvolvidos métodos tipo dual simplex para problemas na forma geral explorando o problema dual linear por partes com buscas exatas e inexatas. Foram propostos 3 novos procedimentos de busca. O primeiro procedimento de busca inexata baseia-se na estrutura linear por partes da função objetivo e examina somente uma parcela dos pontos de não-diferenciação. O segundo procedimento é de busca exata e não faz a ordenação completa do vetor de pontos de não-diferenciação. E o último procedimento, também de busca inexata, é uma adaptação da clássica regra de Armijo (Luenberger, 1984) e faz um teste para evitar passos 'muito grandes'.

Os resultados apresentados nas Tabelas 7 e 8 são dos problemas construídos pelo gerador bloco angular utilizando a combinação heurística de Markowitz com variante 2, o valor 100 para as atualizações e quatro formas diferentes de determinar a direção de busca: Busca Completa 1 (BC_1), Busca Completa 2 (BC_2), Regra de Armijo (AR) e Busca Parcial (BP) (Sousa *et al*., 2005). Na Busca Parcial a quantidade de pontos de não-diferenciação ordenados foi $0{,}25n$ ($n > m$) e $0{,}25m$ ($m > n$).

Analisando os resultados da Tabela 7, vê-se que ao usar a Regra de Armijo foram necessárias mais iterações para se obter a solução ótima em praticamente todos os problemas. As outras três buscas realizaram o mesmo número de iterações em todos os problemas no caso em que $n > m$. No caso em que $m > n$ as Buscas Completa 1 e 2 realizaram o mesmo número de iterações, como era esperado, pois as duas são exatas. Elas apresentaram resultados melhores que a Busca Parcial em poucos casos.

Tabela 7 – Diferentes buscas – Iterações para $n > m$ e $m > n$ (bloco angular).

Número de Iterações											
n > m						***m > n***					
Problemas	**%nz**	**BC_1**	**BC_2**	**AR**	**BP**	**Problemas**	**%nz**	**BC_1**	**BC_2**	**AR**	**BP**
60x400	5,00	**74,00**	**74,00**	85,00	**74,00**	402x50	4,23	**80,80**	**80,80**	81,30	**80,80**
90x400	3,33	114,00	114,00	**113,80**	114,00	402x100	2,24	138,30	138,30	**138,00**	138,30
158x400	2,53	**144,50**	**144,50**	157,80	**144,50**	403x150	2,23	**201,30**	**201,30**	201,80	**201,30**
204x400	1,47	**88,50**	**88,50**	105,30	**88,50**	402x200	1,24	**181,80**	**181,80**	**181,80**	**181,80**
120x1000	2,50	**320,00**	**320,00**	384,50	**320,00**	1002x100	2,10	171,80	171,80	177,50	**170,00**
258x1000	1,16	**189,50**	**189,50**	210,50	**189,50**	1002x250	0,90	**346,30**	**346,30**	353,80	348,50
504x1000	0,60	**203,00**	**203,00**	215,00	**203,00**	1002x500	0,50	**417,50**	**417,50**	426,50	**417,50**
220x2000	1,36	**369,50**	**369,50**	496,50	**369,50**	2002x200	1,05	**374,00**	**374,00**	376,80	**374,00**
508x2000	0,59	**315,80**	**315,80**	380,70	**315,80**	2002x500	0,45	**686,80**	**686,80**	706,80	693,50
1004x2000	0,30	**414,00**	**414,00**	538,80	**414,00**	2002x1000	0,25	**883,80**	**883,80**	939,80	899,50
320x3000	0,94	**466,50**	**466,50**	830,00	**466,50**	3002x300	0,70	566,50	566,50	577,50	**565,30**
762x3000	0,52	**765,30**	**765,30**	1050,80	**765,30**	3003x750	0,43	**1266,30**	**1266,30**	1285,80	**1266,30**

Tabela 8 – Diferentes buscas – Tempo de resolução para caso $n > m$ (bloco angular).

Tempo Total de Resolução (em segundos)											
n > m						***m > n***					
Problemas	**%nz**	**BC_1**	**BC_2**	**AR**	**BP**	**Problemas**	**%nz**	**BC_1**	**BC_2**	**AR**	**BP**
60x400	5,00	**0,31**	0,32	0,40	0,32	402x50	4,23	**0,02**	**0,02**	**0,02**	**0,02**
90x400	3,33	0,81	0,67	0,68	**0,63**	402x100	2,24	**0,05**	0,08	0,07	0,06
158x400	2,53	0,90	0,85	0,96	**0,79**	403x150	2,23	**0,23**	0,29	0,25	0,26
204x400	1,47	**0,25**	0,34	0,40	0,29	402x200	1,24	**0,20**	0,26	0,21	0,21
120x1000	2,50	46,41	**42,83**	60,28	44,02	1002x100	2,10	**0,10**	0,11	0,11	**0,10**
258x1000	1,16	11,45	9,99	11,93	**9,95**	1002x250	0,90	**0,73**	0,93	0,82	0,78
504x1000	0,60	6,08	5,47	6,28	**5,17**	1002x500	0,50	**2,14**	2,69	2,23	2,15
220x2000	1,36	316,05	**264,05**	403,94	265,84	2002x200	1,05	**0,73**	0,82	0,77	0,75
508x2000	0,59	114,35	**107,71**	155,02	107,83	2002x500	0,45	**4,32**	5,15	4,81	4,44
1004x2000	0,30	96,23	95,25	181,85	**94,38**	2002x1000	0,25	**26,89**	28,67	31,71	27,71
320x3000	0,94	848,08	**834,13**	1771,43	841,44	3002x300	0,70	**2,23**	2,40	2,33	2,26
762x3000	0,52	1105,15	**1035,61**	1645,50	1036,07	3003x750	0,43	39,15	39,02	42,02	**38,50**

Observando a Tabela 8 pode-se dizer que os resultados obtidos com relação ao tempo total de resolução foram também piores para a Regra de Armijo em todos os problemas testados. Para os problemas em que $n > m$ a Busca Completa 2 e a Busca Parcial apresentaram resultados melhores e muitos próximos havendo uma alternância entre elas. E no caso em que $n > m$ a Busca Completa 1 apresentou melhor desempenho.

6. Conclusões

Considerando os resultados obtidos com todos os geradores implementados, em ambos os casos em que $n > m$ e $m > n$, constatou-se que houve uma considerável diferença em relação ao número de preenchimentos entre as combinações da heurística de Markowitz com as variantes 1 e 2 e as combinações da heurística de Grau Mínimo com as variantes 1 e 2, sendo que as duas primeiras combinações apresentaram resultados melhores e mais estáveis, ou seja, sem diferenças significativas entre as ordens dos problemas.

Em todos os testes realizados o tempo total de resolução dos problemas usando o método dual simplex foi maior para os problemas no caso em que $n > m$. Isto ocorreu porque o número de preenchimentos feitos também foi maior para este caso. Assim, fica claro o quanto é importante manter a esparsidade para reduzir o tempo de resolução do problema.

Vale salientar que, o número de atualizações da decomposição LU realizadas nas iterações seguintes a ela influenciam no tempo total de resolução dos problemas, nos preenchimentos ocorridos e até mesmo no número de iterações. O valor 100 para o número de atualizações obteve um melhor desempenho em geral e valores menores que $\sqrt{m}$ e maiores que $0{,}1m$ não apresentam bons resultados.

Com relação aos procedimentos de busca testados, pode-se concluir que a Regra de Armijo teve o pior desempenho, pois fez mais iterações e também levou mais tempo para chegar à solução ótima.

De acordo com os resultados obtidos pela implementação do método dual simplex linear por partes que utiliza técnicas de esparsidade, comparados com os resultados obtidos pela implementação do mesmo método que não faz uso dessas técnicas e, portanto, trata o problema como denso, pode-se dizer que houve uma grande evolução em relação ao tempo total de resolução. Em alguns exemplos a implementação esparsa foi aproximadamente 100 vezes mais rápida que a implementação densa e, quanto maior o tamanho dos problemas a diferença no tempo de resolução também aumentou.

Reconhecimento

Os autores são gratos aos três revisores anônimos deste artigo, que contribuíram com sugestões importantes. Este trabalho contou com apoio da FAPESP e CNPq.

Referências Bibliográficas

(1) Bartels, R.H. (1971). A Stabilization of the Simplex Method. *Numerische Mathematik*, **16**, 414-434.

(2) Bartels, R.H. & Golub, G.H. (1969). The Simplex Method of Linear Programming Using the LU Decomposition. *Communications of the Association for Computing Machinery*, **12**, 266-268.

(3) Duff, I.S.; Erisman, A.M. & Reid, J.K. (1986). *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford.

(4) Luenberger, D.G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley.

(5) Markowitz, H.M. (1957). The Elimination Form of the Inverse and its Applications to Linear Programming. *Management Science*, **3**, 255-269.

(6) Maros, I. (2003). A Generalized Dual Phase-2 Simplex Algorithm. *European Journal Operational Research*, **149**, 1-16.

(7) Oliveira, A.R.L. & Cantane, D.R. (2007). An Efficient Simplex LU Factorization Update. *International Journal of Pure and Applied Mathematics*, **1**, 1-10.

(8) Reid, J.K. (1982). A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases. *Mathematical Programming*, **24**, 55-69.

(9) Saunders, M.A. (1976). The Complexity of LU Updating in the Simplex Method. **In:** *The complexity of computational problem solving* [edited by R.S. Anderssen and R.P. Brent], University Press, Queensland, 214-230.

(10) Silva, C.T.L. (2002). Problemas de Otimização Linear Canalizados e Esparsos. Dissertação de Mestrado, ICMC - USP - São Carlos.

(11) Sousa, R.S. (2000). Estudos em Otimização Linear. Dissertação de Mestrado, ICMC - USP - São Carlos.

(12) Sousa, R.S.; Silva, C.L.T. & Arenales, M. (2005). Métodos do tipo dual simplex para problemas de otimização linear canalizados. *Pesquisa Operacional*, **25**(3), 349-382.

(13) Suhl, U.H. & Suhl L.M. (1990). Computing Sparse (LU) Factorizations for Large-Scale Linear Programming. *ORSA Journal on Computing*, **2**, 325-335.

(14) Vanderbei, R.J. (1997). *Linear Programming: Foundations and Extensions*. Kluwer Academic Publisher.