Resumo

Neste trabalho é apresentada uma revisão do método simplex com geração de colunas e sua aplicação ao problema de corte de estoque. É apresentado o problema combinado, que acopla os problemas de dimensionamento de lotes e de corte de estoque, incluindo uma formulação matemática deste problema. Em seguida consideramos algumas propriedades da matriz de restrições e como construir uma base esparsa para ela, utilizando um reordenamento estático das colunas básicas. Resultados numéricos de uma implementação em MATLAB que realiza trocas de colunas da base e verifica sua esparsidade, simulando o método simplex são apresentados. Após uma troca de colunas básicas, estas são atualizadas de forma eficiente, de modo que cause o menor preenchimento da matriz. Foram realizados também testes computacionais para verificar a robustez do método, através de operações inversas à decomposição e comparação com as colunas originais. Concluímos que a proposta de construção da base estática esparsa leva a bons resultados computacionais com relação à velocidade e robustez em comparação com abordagens que não consideram a estrutura esparsa da matriz de restrições.

Abstract

In this work the combined problem is considered, which connects the lot sizing and the cutting stock problems. We study some properties of the matrix of constraints and how to factorize the bases without loosing sparsity in the simplex method context, by a static reording of the basic columns. Numerical results of a MATLAB that implementation simulate simplex iterations and verify the sparsity of the factorizations are presentd. After one exchange of basic columns, the factorization is updated efficiently, in order to keep the sparsity of the factorized matrix. Robustness is verified by undoing the factorization in order to simulate the basis update. We conclude that the approach of construction of the static sparse base reording leads to very good computational results for both: speed and robustness, in comparison with approaches which do not consider the sparse structure of the matrix of constraints.

Conteúdo

Capítulo 1

Introdução

As indústrias de manufatura têm sido estimuladas a tornar seus processos mais eficientes devido a aspectos econômicos e avanços computacionais. Isto incentiva o crescimento de modelos de otimização para o controle e planejamento de sistemas produtivos, motivando pesquisas acadêmicas.

O gerenciamento da produção dentro de uma indústria é responsável pelo planejamento e controle da transformação de matérias-primas em produtos finais. O sistema responsável por este gerenciamento denomina-se *Planejamento e Controle da Produção*, que coordena as atividades, desde a aquisição de matérias-primas até a entrega dos produtos finais.

Este problema é bem conhecido na literatura e tem motivado várias pesquisas acadêmicas [28, 15, 21], mas as suas implementações geralmente não exploram a estrutura matricial do problema.

Uma linha de pesquisa importante na área de otimização considera classes específicas de problemas e explora as particularidades da estrutura matricial destas classes com o objetivo de obter implementações mais eficientes. Esta exploração da estrutura pode levar a métodos de solução muito diversificados. Assim, dependendo do problema, a melhor opção pode ser a utilização de métodos iterativos [29], decomposição LU [13, 25, 24, 26], uma combinação entre decomposição e métodos iterativos [6], ou mesmo a redução do problema de tal forma que a esparsidade apareça apenas implicitamente

[22, 23]. Todas estas aplicações têm como resultado a obtenção de implementações mais eficientes que a utilização de métodos genéricos de otimização.

O objetivo deste trabalho consiste na resolução dos sistemas lineares oriundos do método simplex, explorando a estrutura matricial inerente ao problema de programação de lotes e cortes. Este é um modelo cuja matriz de restrições possui uma estrutura bloco angular com acoplamento entre os blocos. Estes blocos matriciais têm alto grau de esparsidade e pouco acoplamento, indicando que esta abordagem deve produzir bons resultados em termos de eficiência computacional, conforme observado neste trabalho.

A solução eficiente de sistemas lineares de grande porte é de fundamental importância na resolução de problemas de otimização. A solução pode ser obtida através de métodos genéricos, via decomposição LU das matrizes que formam as bases no método simplex, ou através da exploração da estrutura da classe de problemas a ser resolvida [32].

No Capítulo 2 é feita uma revisão sobre os principais conceitos do método simplex e da técnica de geração de colunas, aplicando-a ao problema de corte. Também é descrita a forma como é realizada a decomposição LU da base e a atualização das decomposições utilizando o pivoteamento parcial com a versão GAXPY [9].

No Capítulo 3 é apresentado o problema combinado, que acopla dois conhecidos problemas da otimização linear: o problema de dimensionamento de lotes e o problema de corte de estoque. Também é feita sua formulação matemática e a formulação matricial da matriz de restrições. Esta matriz é reordenada de forma que adquira um formato bloco diagonal para facilitar sua futura decomposição.

Algumas considerações relevantes sobre a matriz de restrições são descritas no Capítulo 4. Dentre elas, a base inicial da matriz de restrições é construída para que sua decomposição seja feita posteriormente. É descrito um algoritmo para identificarmos a qual bloco matricial pertencem as colunas que sai e que entra na base.

No Capítulo 5 são apresentados resultados dos nossos experimentos numéricos, onde é simulado o método simplex, testando a singularidade e a esparsidade. Foi implementado em MATLAB uma decomposição LU da base esparsa da matriz de restrições do

problema combinado, que explora sua estrutura esparsa, mostrando-se mais eficiente se comparada à decomposição que não considera esta estrutura específica.

No Capítulo 6 são apresentadas as perspectivas de continuidade e de melhoria deste trabalho.

Capítulo 2

Método Simplex e Decomposição LU

Neste capítulo, é apresentada uma revisão do método simplex com a técnica de geração de colunas, por ser o mais adequado na resolução de problemas de corte de peças, para o qual um modelo de otimização linear com um número muito grande de colunas é formado. Também são descritas formas para atualização da decomposição LU da base utilizando o pivoteamento parcial.

2.1 O Método Simplex com Geração de Colunas e Aplicação ao Problema de Corte

O método simplex foi desenvolvido para a resolução dos problemas de otimização linear. Entretanto, existem problemas que apresentam uma característica que inviabiliza a utilização de métodos numéricos diretamente: o número extremamente grande de variáveis. Porém, nesta aplicação, os coeficientes das variáveis podem ser calculados eficientemente, uma vez que representam padrões de cortes e, consequentemente, coeficientes da função objetivo, viabilizando a utilização do método simplex.

2.1.1 Breve Revisão do Método Simplex

Considere o problema primal de otimização linear na forma padrão [5, 32]:

$$\min f(x) = c^T x$$

$$\text{s.a: } Ax = b$$

$$x \geq 0$$

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

A solução geral do sistema em $Ax = b$ pode ser descrita considerando uma partição nas colunas de A:

$$A = (B, N)$$

tal que $B \in \mathbb{R}^{m \times m}$, formada por m colunas da matriz A, seja não singular. A partição equivalente é feita no vetor das variáveis:

$$x = (x_B, x_N),$$

onde x_B é chamado *vetor de variáveis básicas* e x_N *vetor de variáveis não básicas.* Assim,

$$Ax = b \Leftrightarrow Bx_B + Nx_N = b \Leftrightarrow$$

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

Dada uma escolha qualquer para x_N, temos x_B bem determinado, de modo que o sistema está verificado.

Definição 2.1 *A solução particular x obtida por $x_B^0 = B^{-1}b$, $x_N^0 = 0$ é chamada solução básica. Se $x_B^0 = B^{-1}b \geq 0$, então a solução básica é primal factível.*

Considere também a partição nos coeficientes do gradiente da função objetivo c:

$$c^T = (c_B, c_N)^T.$$

Definição 2.2 *Chamamos o vetor $y \in \mathbb{R}^m$, dado por*

$$y^T = c_B^T B^{-1}$$

por vetor das variáveis duais ou vetor multiplicador simplex. Se

$$c_j - y^T a_j \geq 0,$$

para $j = 1, \ldots, n$ então y é uma solução básica dual factível, e dizemos que a partição é dual factível, onde a_j representa a coluna j da matriz de restrições A.

Teorema 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: [32].

Teorema 2.2 *Se o problema primal tiver uma solução ótima, então existe uma partição básica ótima.*

Prova: [32].

Definição 2.3 *Chamamos de estratégia simplex a seguinte perturbação da solução básica: escolha $k \in N$, onde N é o conjunto de índices de variáveis não básicas, tal que $c_k - y^T a_k < 0$; faça $x_k = \epsilon \geq 0$, $x_j = 0, \forall j \in N - k$.*

A estratégia simplex produz uma nova solução dada por

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

e o valor da função objetivo dado por:

$$f(x) = f(x^0) + (c_k - y^T a_k)\epsilon$$

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

Note que a direção $d \in \mathbb{R}^n$, dada por $d = (d_B, d_N)^T = (d_B, e_k)^T$, define uma perturbação da solução básica e é chamada *direção simplex*. Se a solução básica for não-degenerada, isto é, $x_B^0 > 0$, então d é uma direção factível. Note ainda que o produto escalar entre d e o gradiente da função objetivo é $c^T d = c_k - y^T a_k < 0$. Portanto d é uma direção de descida.

Da estratégia simplex, podemos determinar o maior valor de ϵ, impondo $x_B \geq 0$:

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

onde $x_{B_e}^0$ é a e-ésima componente de 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

fase I

Encontre uma partição básica primal-factível: $A = (B, N)$.

Faça PARE=FALSO, IT=0

(Será FALSO até que a condição de otimalidade seja verificada. IT indica o número da iteração.)

fase II

Enquanto NÃO PARE faça:

- Determine a solução básica primal factível: $x_B = B^{-1}b$.
- Teste de otimalidade:
  Determine a solução básica dual: $y^T = c_B^T B^{-1}$;
  Encontre x_k com custo relativo: $c_k - y^T a_k < 0$.
  Se $c_k - y^T a_k \geq 0, \quad \forall \ k = 1, \ldots, n-m$, então a solução na iteração IT é ótima. PARE=VERDADE.
  Senão:
- Determine a direção simplex: $d_B = -B^{-1}a_k$, de mudança nos valores das variáveis básicas.
- Determine o passo: $\epsilon^0 = min\left\{-\frac{x^0_{B_e}}{d_{B_e}} | d_{B_e} < 0, i = 1, \ldots, m\right\}$.
  Se $d_B \geq 0$, o problema não tem solução ótima finita. PARE=VERDADE.
  Senão:
- Atualize a partição básica: $a_{B_l} \leftrightarrow a_k, IT \leftarrow IT + 1$.
  Fim enquanto.

Detalhes sobre a Fase I e determinação da variável que entra na base podem ser vistos em [16, 5, 32].

2.1.2 Geração de Colunas

Suponha que em um problema de otimização linear o número de variáveis seja muito maior que o número de restrições. Ao aplicarmos o método simplex, necessitamos de uma base primal factível (B). Para verificar a otimalidade da solução básica x, determinamos x_k com o menor custo relativo:

$$c_k - y^T a_k = min\left\{c_j - y^T a_j \geq 0, j = 1, 2, \ldots, n - m\right\}.$$

Em alguns problemas, as colunas de A são calculadas de modo a satisfazer algumas propriedades, isto é, a é uma coluna de A se e somente se $a \in X \subseteq \mathbb{R}^m$ (para o problema deste trabalho, X é um conjunto de padrões de corte), bem como o coeficiente na função objetivo correspondente é determinado pelos elementos da coluna: $c(a)$.

Observação: A matriz A neste caso, representa somente as colunas do padrão de corte, e não toda a matriz de restrições, pois optamos por utilizar a notação mais comum da literatura de cortes.

A técnica de geração de colunas pode ser aplicada a problemas lineares de grandes dimensões, no caso de não se dispor de todas as colunas a priori, ou quando se pretende resolver um problema utilizando a decomposição de Dantzig-Wolfe, onde as colunas correspondem aos pontos extremos do conjunto convexo de soluções factíveis do problema. Neste caso, o método de resolução alterna entre um subproblema e um problema mestre restrito. A partir de um conjunto inicial de colunas, resolve-se o problema mestre, obtendo-se as variáveis duais que serão utilizadas pelo subproblema para determinar novas colunas a serem consideradas no problema mestre. Os métodos de geração de colunas e o de decomposição de Dantzig-Wolfe ficaram conhecidos como métodos eficientes para o tratamento de problemas de otimização linear com grande número de variáveis. Um problema mestre restrito é identificado e novas colunas são geradas através de um subproblema [14].

O motivo para utilizar o método simplex com geração de colunas neste trabalho, vem do fato de existir um número elevado de colunas correspondentes aos padrões de corte, o que proporciona uma matriz de restrições extremamente grande.

Sabemos que apenas m colunas de A serão necessárias para descrever uma solução ótima, isto é, uma solução básica, de modo que com a geração de colunas, não necessitamos armazenar toda a matriz A.

Considere $a = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, uma coluna de A e suponha que coeficiente na função objetivo seja dado por

$$c(a) = \gamma_0 + \sum_{i=1}^{m} \gamma_i \alpha_i,$$

onde γ_i são conhecidos. Assim, o custo relativo da variável, cuja coluna é dada por a, pode ser determinado por:

$$\phi(a) = c(a) - y^T a = \gamma_0 + \sum_{i=1}^{m} \gamma_i \alpha_i - \sum_{i=1}^{m} y_i \alpha_i = \gamma_0 + \sum_{i=1}^{m} (\gamma_i - y_i)\alpha_i$$

onde y_i é o vetor multiplicador de uma dada iteração do método simplex. Um sub-problema, para determinar uma coluna de A a entrar na base, pode ser reescrito por:

$$\min \phi(a) = \gamma_0 + \sum_{i=1}^{m} (\gamma_i - y_i)\alpha_i$$

$$\text{s.a: } (\alpha_1, \alpha_2, \ldots, \alpha_m) \in X.$$

As colunas de A, não estando disponíveis, podem ser geradas, à medida que necessário, pela solução deste sub-problema.

2.1.3 Aplicação de Geração de Colunas ao Problema de Corte

Consideremos o problema mais simples, onde barras de comprimento L devem ser cortadas em pedaços $l_i, i = 1, \ldots, P$. As colunas de A são bem determinadas por

$$X = \{a = (\alpha_1, \alpha_2, \ldots, \alpha_P) | l_1\alpha_1 + l_2\alpha_2 + \ldots + l_P\alpha_P \leq L_i, \alpha_i \geq 0 \text{ e inteiro}\}$$

Como já observamos anteriormente, algumas colunas de A são facilmente construídas, considerando-se padrões homogêneos, ou seja, cada padrão de corte produz quantidades de um único tipo de peça. Os vetores associados a eles constituem P vetores LI de X:

$$a_i = (0, \ldots, a_{ii}, \ldots, 0), i = 1, \ldots, P,$$

onde $a_{ii} = \lfloor L/l_i \rfloor$.

Assim, podemos construir a seguinte matriz diagonal [3]:

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{PP} \end{bmatrix}.$$

Desta forma, o sub-problema pode ser escrito por

$$\min \phi(a) = l - \sum_{i=1}^{p} y_i \alpha_i$$

$$\text{s.a: } \sum_{i=1}^{p} l_i \alpha_i \leq L,$$

$$\alpha_i \geq 0 \quad \text{e inteiro, } i = 1, \ldots, P.$$

que pode ser reescrito como

$$\max \sum_{i=1}^{p} y_i \alpha_i$$

$$\text{s.a: } \sum_{i=1}^{p} l_i \alpha_i \leq L,$$

$$\alpha_i \geq 0 \quad \text{e inteiro, } i = 1, \ldots, P.$$

Este é o modelo de um problema conhecido na literatura como *Problema da Mochila*, cuja motivação decorre da situação hipotética onde um muambeiro deseja carregar sua mochila com itens, cujos valores de compra são l_i. O valor total da compra não pode ultrapassar L. O muambeiro deseja maximizar seu lucro total. A solução do problema da mochila fornece uma coluna $(\alpha_1, \ldots, \alpha_P)$ tal que o custo relativo é mínimo. Portanto, resolvendo o problema de corte de estoque unidimensional, a cada iteração do método simplex será necessário resolver um problema da mochila para determinar a coluna com o menor custo relativo. Mais detalhes sobre este problema podem ser vistos em [12, 2, 20, 18, 8].

Para o caso bidimensional com dois estágios, ou seja, quando são consideradas duas dimensões no processo de corte e apenas uma mudança no sentido do corte é permitida: horizontal/vertical, a cada iteração do método simplex será necessário resolver $P + 1$ problemas da mochila, pois haverá alocação dos itens em P faixas mais a alocação das faixas no retângulo, somando mais um problema da mochila. Já para o plano de corte bidimensional de uma maneira geral, é utilizada uma abordagem por grafo e/ou para que as colunas sejam determinadas. Este procedimento pode ser visto em [1].

Neste trabalho, estudaremos a aplicação do método Simplex com geração de colunas no problema de lotes e cortes, apresentado no próximo capítulo.

2.2 A Decomposição LU da Base

Por ser inviável inverter a matriz básica B, do ponto de vista computacional, por razões de esparsidade e eficiência computacional, usualmente realiza-se a decomposição LU da base, onde L é uma matriz triangular inferior e U triangular superior, proporcionando um algoritmo mais eficiente e rápido.

Para decompor a matriz, podemos considerar a *decomposição de Crout* [7], que assume U unitária (possui apenas 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 é a primeira coluna de L, a primeira linha de U, em seguida, a segunda coluna de L, a segunda linha de U e assim por diante. Isto pode ser comparado à eliminação Gaussiana, 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 é a divisão da linha do pivô (e não a coluna) pelo pivô b_{kk}^{k}.

Mais geralmente, podemos desenvolver a decomposição

$$B = LDU,$$

onde D é uma matriz diagonal e ambas L e U são unitárias. Por exemplo, a decomposição de Crout corresponde à associação $(LD)U$ e a decomposição de Dolittle está associada à $L(DU)$, sendo portanto, a matriz L unitária. Esta decomposição é única, enquanto que a decomposição LU não é.

É possível armazenar a decomposição LU diretamente na mesma área ocupada pela matriz B. Mais precisamente, U é armazenada na parte triangular superior de B (a diagonal de U não é armazenada, pois seus elementos são assumidos ser 1) e L ocupa a parte triangular inferior de B (incluindo a diagonal).

2.2.1 Decomposição com Pivoteamento Parcial: Versão GAXPY

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, estaremos utilizando o pivoteamento parcial por nos oferecer a melhor combinação entre estabilidade numérica e eficiência computacional.

Seja B a base que pretendemos decompor. Esta decomposição calcula a matriz triangular inferior unitária L, a triangular superior U e a permutação P tal que $PB = LU$. $B(i,j)$ terá $L(i,j)$ em sua parte superior se $i > j$ e terá $U(i,j)$ caso contrário. $P = E_{n-1} \dots E_1$ está representada no vetor inteiro $piv(1 : n-1)$. Em particular, E_j é uma permutação envolvendo a linha j e $piv(j)$, com $j = 1 : n-1$.

Esta decomposição é um procedimento de três laços que podem ser dispostos de várias maneiras. 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 algoritmo a seguir. O nome vem de uma generalização de SAXPY (Scalar $a \times$ 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.

for j=1:n
  for k=1:j-1
    for i=k+1:n
      $B(i,j) = B(i,j) - B(i,k)$*$B(k,j)$;
    end
  end
  for i=j+1:n
    $B(i,j)$=$B(i,j)/B(j,j)$
  end
end

A base original $B(:,j)$ é mantida até o passo j. No ponto em que a parte superior de $B(:,j)$ for escrita por $M_{j-1}\ldots M_1 B(:,j)$, onde M são as matrizes dos multiplicadores, a j-ésima transformação Gaussiana é calculada [9].

As permutações podem ser usadas para garantir que nenhum multiplicador é maior que um em valor absoluto, obtendo decomposições mais estáveis [9]. 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 esta matriz resultante pela matriz dos multiplicadores 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 $|(E_k M_{k-1}\ldots M_1 E_1 B)_{kk}| = max_{k\leq i\leq n}|(E_k M_{k-1}\ldots M_1 E_1 B)_{ik}|$ para $k = 1 : n-1$. Portanto, o pivoteamento parcial

efetivamente protege contra multiplicadores maiores.

Limiar de pivoteamento [1] é uma estratégia de pivoteamento às vezes utilizada em substituição ao pivoteamento parcial. Nesta estratégia, qualquer entrada não eliminada $A(p,k)$ na coluna corrente k pode ser selecionada como pivô fornecendo $\mid A(p,k) \mid \geq u.max \mid (A(k:n,k)) \mid$ 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 [2] limitado. Esta estratégia de pivoteamento oferece pequeno benefício para o caso onde a matriz é densa e pode estar preso na memória principal, 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 *fill-in* (preenchimento), mas que ainda contém um pivô suficientemente grande, onde "suficientemente"grande significa que o pivô que selecionamos é bastante grande por alguma medida ou *limiar*, por exemplo, ele é pelo menos 10% tão grande quanto o maior elemento absoluto nesta coluna.

A seguir, veremos métodos para atualizar a decomposição LU da matriz básica, buscando manter a esparsidade original dos dados.

2.2.2 Atualização da Base: Esparsidade

Nesta seção, vamos apresentar a técnica para atualizar a decomposição LU da matriz básica, proposta por Bartels-Golub [4], a qual utiliza o pivoteamento parcial. Duas variantes deste algoritmo tentam manter tanto a esparsidade existente nas matrizes básicas quanto a estabilidade numérica. A segunda variante é um aperfeiçoamento da primeira e ambas foram propostas por Reid [27].

[1] do inglês *Threshold pivoting*

[2] do inglês *growth*

Decomposição de Bartels-Golub

Esta técnica [4] foi desenvolvida para atualizar a decomposição LU da matriz básica B em cada iteração do método simplex, e utiliza o pivoteamento parcial, que consiste em utilizar o elemento pivô como o maior elemento em módulo da coluna [7], evitando assim que os erros de arredondamento se propaguem de forma descontrolada.

Suponha que desejamos resolver um sistema do tipo $Bv = q$ e que a matriz B tenha sido decomposta no produto de uma matriz triangular superior U e uma matriz triangular inferior L, onde os elementos da diagonal de L são iguais a 1. Tal decomposição é sempre possível, pois a permutação nas linhas de B é permitida.

Desta forma, temos $PB = LU$ para alguma matriz de permutação P. Logo, a resolução do sistema inicial $Bv = q$ pode ser obtida resolvendo dos dois sistemas triangulares: $Lt = Pq$ e $Uv = t$.

Seja $B^{(0)} = [a_{B_1} \dots a_{B_l} \dots a_{B_m}]$ a matriz básica inicial. No final de uma iteração foi requerida a construção de uma nova matriz básica $B^{(1)}$, pois a l-ésima coluna deixará a base e a k-ésima coluna da matriz N entrará na base. $B^{(1)}$ terá a seguinte forma: $B^{(1)} = [a_{B_1} \dots a_{B_{l-1}} a_{B_{l+1}} \dots a_{B_m} a_{N_k}]$.

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

$$L^{-1}PB^{(1)} = [L^{-1}Pa_{B_1} \dots L^{-1}Pa_{B_{l-1}} L^{-1}Pa_{B_{l+1}} \dots L^{-1}Pa_{B_m} L^{-1}Pa_{N_k}]$$

$$L^{-1}PB^{(1)} = [u_1, u_2, \dots u_{r_j-1}, u_{r_j+1}, \dots, u_m, L^{-1}Pa_{N_k}] = H^{(1)}.$$

onde u_i são as colunas da matriz U e $H^{(1)}$ é uma matriz de Hessenberg superior com zeros abaixo da diagonal principal nas primeiras l-1 colunas, isto é, $H^{(1)}$ tem a forma subtridiagonal superior (onde os elementos abaixo da subdiagonal inferior são nulos) com

zeros abaixo da diagonal principal nas primeiras l-1 colunas. O vetor $L^{-1}Pa_{N_k}$ já foi obtido quando a direção simplex foi calculada, antes da atualização da base.

De fato, ao resolver o sistema $Bd_B = -a_{N_k}$ temos

$$PBd_B = -Pa_{N_k} \Rightarrow LUd_B = -Pa_{N_k} \Rightarrow t = -L^{-1}Pa_{N_k}.$$

Assim, $H^{(1)}$ pode ser construída sem qualquer esforço adicional, e pode ser reduzida a uma matriz triangular superior $U^{(1)}$ usando a eliminação Gaussiana para zerar 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, $U^{(1)}$ é obtida de $H^{(1)}$ aplicando uma sequência de operações elementares à esquerda de $H^{(1)}$.

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

Variantes de Bartels-Golub

As duas variantes do algoritmo de Bartels-Golub tentam manter tanto a esparsidade existente nas matrizes básicas quanto a estabilidade numérica. Estas variantes podem ser usadas quando equações da forma: $By = a$ e $B^t z = d$ precisam ser resolvidas em toda iteração e cada matriz B, da sequência obtida, difere de sua antecessora em apenas uma coluna, como ocorre no método simplex.

Assim como o algoritmo de Bartels-Golub, estas duas variantes são usadas para que a decomposição LU da matriz básica não seja feita em todas iterações. Elas fazem atualizações da decomposição 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.

Variante 1 - Algoritmo Esparso de Bartels-Golub

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

$$E_m^{(0)} E_{m-1}^{(0)} \dots E_1^{(0)} B^{(0)} = P^{(0)} U^{(0)} Q^{(0)}$$

onde,

- $E_m^{(0)} E_{m-1}^{(0)} \dots E_1^{(0)} = (L^{(0)})^{-1}$;
- E_i é uma matriz elementar que difere da matriz identidade em apenas um elemento fora da diagonal (os multiplicadores) e, portanto, representa uma sequência de operações elementares por linha;
- P e Q são matrizes de permutação;
- L é uma matriz triangular inferior e U é uma matriz triangular superior.

Cada coluna da matriz triangular inferior L pode ser armazenada como um produto de matrizes elementares. Tal formação, pode simplificar bastante muitas 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 adicional.

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

$$E_m^{(0)} E_{m-1}^{(0)} \dots E_1^{(0)} B^{(1)} = P^{(0)} S^{(1)} Q^{(0)}$$

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

A coluna da matriz $S^{(1)}$ que difere da matriz $U^{(0)}$ possui elementos não nulos abaixo da diagonal, destruindo a triangularidade de $U^{(0)}$. Esta coluna é comumente chamada de *coluna espeto* [3], que por definição, é a coluna que contém pelo menos um elemento não nulo abaixo da diagonal (Figura 2.1(a)).

[3] do inglês *spike column*

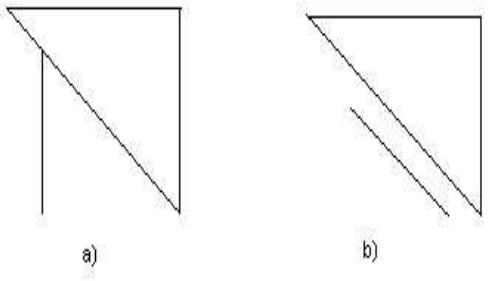

Figura 2.1: (a) Matriz $S^{(1)}$ com coluna espeto (b) Matriz subtriangular superior

Bartels e Golub sugeriram fazer permutações de colunas movendo a coluna espeto para a última coluna da matriz e empurrando as colunas posteriores a espeto uma posição para a esquerda, obtendo uma matriz na forma subtriangular superior (Figura 2.1(b)), e depois, realizar 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 LU da matriz básica inicial, deve-se utilizar uma heurística de pivoteamento. A heurística que ele utilizou foi a *heurística de Markowitz* [17], pelo fato de existirem bons resultados obtidos por esta heurística em trabalhos anteriores. Isto difere a variante proposta por Reid do algoritmo de Bartels-Golub, o qual utiliza o pivoteamento parcial, que não preserva a esparsidade.

A heurística de grau mínimo proposta por Markowitz pode ser resumida nos seguintes passos:

1. Antes de eliminar os elementos não nulos abaixo do pivô, procure entre as linhas que ainda não foram eliminadas aquela que é mais esparsa, ou seja, a linha que possui menos elementos diferentes de zero na parte que ainda não foi eliminada.

2. Troque essa linha com a linha do pivô.

3. Para os elementos não nulos na linha escolhida, selecione aquele cuja coluna possui menos elementos não nulos em sua parte não eliminada.

4. Troque essa coluna com a coluna pivô.

5. Efetue a eliminação Gaussiana.

O número de elementos não nulos na sub-matriz ativa de uma linha e/ou coluna é chamado grau da linha e/ou coluna. Portanto, o nome da heurística é grau mínimo.

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.

Agora, se os elementos da diagonal e subdiagonal são não nulos, do ponto de vista de esparsidade, deveria ser escolhido como pivô aquele que possui menos elementos não nulos em sua linha. Mas, como é indesejável por motivo de estabilidade numérica ter um pivô muito pequeno, deve-se então, tomar como pivô, o maior dos dois elementos (diagonal e subdiagonal) se o menor dos elementos é menor que uma constante u, $0 < u \leq 1$, vezes o maior elemento, caso contrário, o pivô será o elemento que se encontra na linha que possui um número menor 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 descrita anteriormente, porque 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, ou então, reduzir o número de elementos na sub-diagonal inferior a serem eliminados.

Segundo [30], é possível explorar o fato que, no caso esparso, a coluna espeto também é bastante esparsa e que, portanto, 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 deve ser colocada na posição da linha em que se encontra o último elemento desta coluna.

Por exemplo, suponha que a nova matriz $U^{(1)}$ a ser colocada na forma triangular superior é de ordem 10 e que a coluna espeto está na coluna 2 e 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 menos elementos não nulos na subdiagonal, ou seja, apenas 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 coluna 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, diminuindo 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 $l+1, \ldots, p-1$ são movidas uma posição para a direita e 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 foi preservada e o comprimento da

coluna espeto, agora na coluna $l+1$, foi reduzido por um, e 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 $l+2, \ldots, p$ podem possuir 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).

Em seguida, deve ser aplicado um procedimento análogo a este acima 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 outras linhas que tenham um único elemento não nulo 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 que esteja na diagonal é feito até que não sejam encontradas linhas com estas características (As permutações novamente 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 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 juntas posteriormente).

Após todas as permutações possíveis serem realizadas, a matriz U está pronta para que as eliminações sejam feitas se estas forem necessárias, com o objetivo de tornar a matriz 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 que uma nova decomposição da matriz básica seja feita, o que representa um grande esforço computacional.

No trabalho [31] foram implementadas algumas formas de atualização da decomposição LU da matriz básica, com o objetivo de manter a esparsidade original dos dados. Também foi apresentada a decomposição LU proposta por Bartels-Golub, como 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 a ela, influenciam bastante no tempo de resolução dos problemas, nos preenchimentos ocorridos e até mesmo no número de iterações.

Capítulo 3

O Problema Combinado

O processo de programar a produção num ambiente onde ocorre um estágio de corte de peças em estoque (por exemplo, placas), pode ser dividido em três etapas:

1. A primeira delas define uma carteira de pedidos para um horizonte de planejamento finito (como dias, meses, etc.) especificando as quantidades dos produtos finais demandados e suas respectivas datas de entrega.

2. A segunda etapa converte a demanda de produtos finais em demanda de peças. Deste modo, um tipo de peça pode ser utilizado por vários produtos finais diferentes, e as quantidades das peças necessárias são produzidas.

3. Finalmente, a terceira etapa decide a quantidade de produtos finais que devem ser produzidos em cada período do horizonte de planejamento, minimizando os custos e as perdas ocorridas no processo de corte das placas em peças.

Frequentemente existem perdas de material no corte de peças. Esta perda tende a ser relativamente menor conforme a demanda de peças aumenta, devido a um melhor rearranjo dos padrões de corte nas placas. Portanto surge uma pressão econômica para fabricar alguns produtos antecipadamente com objetivo de minimizar as perdas. Por outro lado, os custos de estoque exercem pressão oposta no sentido de retardar a produção. Baseado nesta decisão de antecipar ou não a produção de certos produtos finais surge o *problema combinado* [10], que acopla dois importantes problemas de otimização, o *dimensionamento de lotes* e o *corte de estoque*.

Em situações reais, a maioria das indústrias aborda esses dois problemas de forma separada, devido à alta complexidade do problema combinado. Inicialmente, são determinados para cada período do horizonte de planejamento, as quantidades de cada produto final (tamanho do lote) a serem produzidas. A partir desta informação, determina-se, para cada período, a quantidade de peças de cada tipo a serem cortadas e os melhores padrões de corte são gerados. Entretanto, tratá-los de forma separada pode elevar os custos globais, principalmente se uma parcela significativa do custo do produto final é formada pelo material a ser cortado.

Apesar dos problemas de corte serem tratados tradicionalmente de forma isolada, eles surgem na prática dentro de um contexto da programação da produção, onde ordens de serviço com data de entrega, limitações de capacidades, etc, definem condições de contorno que influenciam na otimização dos planos de cortes. O problema de selecionar objetos e definir como devem ser cortados consiste por si só um problema de otimização combinatória que tem motivado intensas pesquisas ([8, 11, 19, 10] entre outros). Sua combinação com o planejamento da produção é ainda pouco explorada na literatura, mas a constatação de sua relevância em diversas situações na prática o elege como um importante problema a ser pesquisado.

3.1 Problema de Dimensionamento de Lotes e Problema de Corte de Estoque

Vejamos os problemas de corte de estoque e de dimensionamento de lotes descrevendo separadamente cada um deles.

1. *Problema de Dimensionamento de Lotes (PDL):* Este problema consiste em planejar a quantidade dos itens a ser produzida em vários estágios, em cada período ao longo de um horizonte de tempo finito, de modo a atender a demanda e otimizar uma função objetivo, como minimizar os custos de produção e de estocagem. Um PDL pode ser classificado como *monoestágio*, onde os itens são produzidos independentemente, e *multiestágio*, em que a produção de um item depende da produção

de um outro item.

Para resolvermos este problema, podemos decompô-lo em M subproblemas com apenas um produto final, que podem ser resolvidos, por exemplo, por programação dinâmica [34].

2. *Problema de Corte de Estoque (PCE):* Este problema consiste na otimização do processo de corte de placas em peças menores nas quantidades e dimensões demandadas. Define-se *padrão de corte* como o arranjo das peças dentro de cada placa, isto é, a forma como um objeto (peça) é cortado para a produção de itens demandados. Algumas regras são necessárias para defini-lo, como cortes do tipo *guilhotinado* (onde cada corte feito sobre uma placa retangular produz dois novos retângulos), limitação de peças (cortes restritos ou irrestritos), número de estágios (é dito ser 2-estágios quando apenas uma mudança no sentido dos cortes guilhotinados é permitida: horizontal/vertical ou vertical/horizontal). Além disso, o problema será *bidimensional* quando duas dimensões são relevantes para cortagem.

   Para resolver este problema, podemos aplicar o método simplex em conjunto com a técnica de geração de colunas, conforme descrito no capítulo anterior.

3.1.1 Problema de Corte de Estoque

Suponha que várias barras estejam disponíveis para serem cortadas na produção dos diversos itens. As barras são suficientes para a produção de todos os itens demandados. Temos então que escolher quais barras devem ser cortadas, ou seja, temos um problema de seleção de objetos a serem cortados.

Os objetos em estoque de mesmo tipo são disponíveis em grande quantidade, e podem ser de apenas um único tipo ou de vários tipos, havendo ou não limitação de estoque. A solução deste problema terá muitas peças em estoque igualmente cortadas para a produção dos diferentes tipos de itens.

Para o caso em que o estoque é composto de *apenas um tipo de barra*, em *quantidade*

ilimitada, toda a demanda será atendida. O objetivo é então atender a demanda ao custo mímimo, ou seja, minimizar o número de objetos (barras) cortados.

Para o caso em que o estoque é composto de *diversos tipos de barras*, em *quantidades ilimitadas*, além dos dados de demanda considerados inicialmente, devemos ter também os dados de estoque, e os padrões de cortes devem ser definidos para cada barra em estoque. Esta aplicação ocorre em indústrias onde os objetos são produzidos por máquinas diferentes e a capacidade de produção é suficientemente grande, ou ainda, os objetos de vários tamanhos são adquiridos no mercado, onde a oferta é grande.

A situação em que o estoque é composto de vários tipos de objetos, porém em *quantidades disponíveis limitadas*, ocorre em indústrias onde os objetos são adquiridos com antecedência e estocados. Um novo dado deve ser adicionado ao problema: a disponibilidade do objeto em estoque.

3.2 Resolução Prática do Problema Combinado

Para resolvermos os problemas de corte de estoque e de dimensionamento de lotes *separadamente*, podemos aplicar uma *heurística de decomposição*, usual na prática, conforme [10], que consiste em:

- Inicialmente, resolver o PDL, em que dada uma carteira de pedidos realizamos o planejamento da produção, decidindo a quantidade de cada tipo de produto final a ser produzido em cada período do horizonte de planejamento, minimizando os custos de produção, estoque e preparação.

- Como segundo passo, para cada período do horizonte de planejamento, resolvemos um problema de corte de estoque com restrições de capacidade, e assim determinamos a quantidade de placas cortadas conforme um determinado padrão.

- Finalmente, a função objetivo do problema combinado é dada pela soma das funções objetivos dos problemas PDL e PCE.

Embora a resolução dos problemas de forma independente seja mais simples pelo fato de já existirem métodos clássicos na literatura que os resolvem de forma eficiente, tratar os problemas separadamente pode fazer crescer os custos de produção global. Apesar das dificuldades do modelo combinado quanto à integralidade das variáveis que representam a quantidade de placas cortadas num certo padrão e a grande quantidade de padrões de corte que pode ser gerado, esta abordagem fornece um ganho significativo nos custos globais, incentivando estudos nesta linha.

3.3 Formulação Matemática do Problema Combinado

O problema combinado consiste em decidir a quantidade de produtos finais a serem produzidos em cada período do horizonte de planejamento tal que minimize os custos da produção, preparação e estocagem (PDL) e a quantidade de placas a serem cortadas, bem como os padrões de corte, para compor produtos finais (PCE). Este problema pode ser formulado como um modelo inteiro-misto que analisa o compromisso entre antecipar a produção de certos lotes de produtos finais para minimizar os custos no processo de corte e preparação, e o aumento dos custos de estocagem [10]. No mesmo trabalho também foi proposta a resolução do problema combinado via método simplex com geração de colunas.

Existe uma relação importante entre a demanda de um produto final e a demanda de seus itens (peças). A demanda de um produto final é um dado externo e as quantidades são determinadas pela carteira de pedidos. Conhecida a demanda de produtos finais, obtemos a demanda interna dos itens. Uma outra consideração importante nos modelos reais é a disponibilidade da serra em cada período. Temos de assegurar que o tempo gasto para cortar as placas não excede o tempo disponível.

3.3.1 O Modelo Matemático

Uma abordagem para o problema combinado desconsidera a ocorrência de custos de preparação e relaxa a integralidade das variáveis que representam a quantidade de placas cortadas num certo padrão, o que pressupõe grandes quantidades de demanda. Esta abordagem pode ser aplicada na indústria de móveis, onde placas de madeira devem ser cortadas na produção de itens. É necessário examinar se é vantajoso antecipar a produção de certos lotes de peças e produtos finais, aumentando os custos de estoque, mas obtendo um ganho no processo de corte com um melhor arranjo das peças nas placas. Por simplicidade, consideramos que haja apenas um tipo de placa em estoque, suficiente para atender a demanda. Definimos então o modelo da seguinte forma:

$$\min \sum_{i=1}^{M}\sum_{t=1}^{T}(c_{it}x_{it} + h_{it}e_{it}) + \sum_{j=1}^{N}\sum_{t=1}^{T} cpy_{jt} + \sum_{p=1}^{P}\sum_{t=1}^{T} hp_{pt}ep_{pt}$$

$$\text{s.a: } x_{it} + e_{i,t-1} - e_{it} = d_{it} \tag{3.1}$$

$$\sum_{j=1}^{N} a_{pj}y_{jt} + ep_{p,t-1} - ep_{pt} = \sum_{i=1}^{M} r_{pi}x_{it} \quad \forall\ t = 1 \ldots T \tag{3.2}$$

$$\sum_{j=1}^{N} v_j y_{jt} \leq u_t \tag{3.3}$$

$$x_{it}, e_{it}, y_{jt}, ep_{pt} \geq 0 \tag{3.4}$$

onde:

Índices:

t=1, ..., T número de períodos.

p=1, ..., P número de diferentes tipos de peças a serem cortadas.

j=1, ..., N número de diferentes padrões de corte.

i=1, ... ,M número de diferentes produtos finais demandados.

Parâmetros:

c_{it}: custo de produção do produto final i no período t.

h_{it}: custo de estocagem do produto final i no período t.

hp_{pt}: custo de estocagem da peça tipo p no período t.

d_{it}: demanda do produto final i no período t.

r_{pi}: número de peças tipo p necessárias para formar um produto i.

v_j: tempo gasto para cortar uma placa no padrão de corte j.

a_{pj}: número de peças tipo p no padrão j.

u_t: tempo máximo de operação da serra.

cp: custo da placa a ser cortada.

Variáveis:

x_{it}: quantidade do produto final i produzido no período t.

e_{it}: quantidade do produto final i em estoque no fim do período t.

ep_{pt}: quantidade da peça tipo p em estoque no fim do período t.

y_{jt}: quantidade de placas cortadas usando o padrão j no período t.

As restrições (3.1) se referem às equações de balanço de estoque com relação aos produtos finais, o que garante que a demanda de itens de cada período será atendida. As restrições (3.2) se referem às equações de balanço de estoque com relação às peças, o que asseguram que a demanda de peças será satisfeita. Estas restrições são as que acoplam os problemas de dimensionamento de lotes e de corte de estoque, pois ambas incluem as variáveis x_{it}, que definem o tamanho dos lotes e y_{jt}, que definem a quantidade de placas cortadas num certo padrão de corte. As restrições (3.3) se referem à capacidade da serra, o que garante que o tempo gasto no processo de corte das placas nos diversos padrões de corte não ultrapassa a capacidade disponível da serra, ou seja, seu tempo máximo de operação.

Este é um modelo linear, por não considerar a integralidade das variáveis. Neste trabalho, caso a integralidade fosse considerada, esta discussão continuaria válida, pois, estamos interessados somente na decomposição LU da base no método simplex. Permanece porém a dificuldade referente à grande quantidade de padrões de corte que podem ser

gerados.

Por simplicidade de notação reescrevemos o modelo da seguinte forma matricial:

$$\min \sum_{t=1}^{T}(c_t x_t + h_t e_t) + \sum_{t=1}^{T} cp.y_t + \sum_{t=1}^{T} hp_t ep_t$$

$$\begin{aligned} \text{s.a:} \quad & x_t + e_{t-1} - e_t = d_t \\ & -Rx_t + Ay_t + ep_{t-1} - ep_t = 0 \quad \forall\ t = 1 \ldots T \\ & v^T y_t \leq u_t \\ & x_t, e_t, y_t, ep_t \geq 0 \end{aligned}$$

onde ep_0, e_0 são conhecidos, $c_t = (c_{1t}, c_{2t}, ..., c_{Mt})$; e os demais parâmetros e variáveis $h_t, e_t, hp_t, ep_t, d_t, x_t$ e y_t são definidos de forma similar,
$v^T = (v_1 \quad v_2 \quad ... \quad v_N)$,

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{P1} & r_{P2} & \cdots & r_{PM} \end{bmatrix}$$

e A é uma matriz $P \times N$ tal que cada coluna corresponde a um padrão de corte.

3.3.2 Formulação Matricial

Acrescentando as *variáveis de folga*

$$f_t = u_t - v^T y_t$$

para $t = 1 \ldots T$, a matriz dos coeficientes para o modelo, tem a seguinte forma:

x_1	x_2	...	x_T	e_1	e_2	...	e_{t-1}	e_t	...	e_T	y_1	y_2	...	y_T	ep_1	ep_2	...	ep_{t-1}	ep_t	...	ep_T	f_1	f_2	...	f_T	d_t
I	0	...	0	-I	0	...	0	0	...	0	0	...		0	0	...					0	0	0	...	0	$d_1 - e_0$
0	I	...	0	I	-I	...	0	0	...	0	0	...		0	0	...					0	0	0	...	0	d_2
					...						0	...		0	0	...					0			...		...
0	0	...	0	0	0	...	-I	0	...	0	0	...		0	0	...					0	0	0	...	0	d_{t-1}
0	0	...	0	0	0	...	I	-I	...	0	0	...		0	0	...					0	0	0	...	0	d_t
...						...			...			...				...								...		...
0	0	...	I	0	0	...	0	0	...	-I	0	...		0	0	...					0	0	0	...	0	d_T
-R	0	...	0	0		...				0	A	0	...	0	-I	0	...	0	0	...	0	0	0	...	0	0
0	-R	...	0	0		...				0	0	A	...	0	I	-I	...	0	0	...	0	0	0	...	0	0
		...		0		...				0			...				...			...				...		0
0	0	...	-R	0		...				0	0	0	...	A	0	0	...	0	0	...	-I	0	0	...	0	0
0	0	...	0	0		...				0	v^T	0	...	0	0	0	...	0	0	0	0	1	0	...	0	u_1
0	0	...	0	0		...				0	0	v^T	...	0	0	0	...	0	0	0	0	0	1	...	0	u_2
		...				...							...				...					...		...		...
0	0	...	0	0		...				0	0	0	...	v^T	0	0	...	0	0	0	0	0	0	...	1	u_T

Tabela 3.1: Matriz de Restrições para o Modelo Combinado

A última coluna representa o termo independente. Note que a matriz é muito esparsa e cresce consideravelmente com o número de peças, produtos finais e períodos.

As matrizes envolvidas têm as seguintes dimensões: $A : P \times N$, $v^T : 1 \times N$, $R : P \times M$. Portanto, a matriz de restrições tem os respectivos números de linhas e colunas:

linhas: $MT + PT + T = T(M + P + 1)$.
colunas: $2MT + NT + PT + T = T(2M + N + P + 1)$, considerando as variáveis de folga.

Este modelo acopla um problema de dimensionamento de lotes com o problema de corte. Apresenta uma estrutura esparsa e tem um grande número de colunas, devido ao número de padrões de corte N. Observe que, embora tenha sido enunciado como um problema bidimensional, é suficientemente geral para abrigar outras dimensões.

3.3.3 Reordenamento da matriz de restrições

Podemos reordenar a matriz de restrições acima de modo que a matriz adquira formato bloco diagonal, a fim de facilitar a decomposição das futuras bases, evitando assim, o preenchimento da matriz.

Pelo 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, e o das colunas da base obedecem esta ordem. Desta forma, o reordenamento inicial não tem custo de inicialização, nem de atualização.

As colunas da matriz de padrões de corte não são reordenadas entre si a priori, mesmo porque elas são geradas durante a execução do método simplex, e portanto, não são conhecidas. Mas se torna fácil a tarefa de encontrar a posição de tais colunas quando entrarem na base com respeito às outras colunas de padrão no mesmo período de tempo. O critério mais prático seria colocá-las em ordem crescente do número de elementos não nulos.

A matriz R, que representa a demanda de peças, pode conter esparsidade, pois, como cada uma de suas colunas representa um produto final, este pode não ser constituído de algumas peças que serão utilizadas para compôr um outro produto. Desta forma, as variáveis x_{it}, que definem o tamanho do lote, podem ser reordenadas de acordo com a matriz R, reordenada também em ordem crescente do número de elementos não nulos. Isto equivale a numerar os itens finais em uma ordem conveniente.

e_1	e_2	...	e_{t-1}	e_t	...	e_T	x_1	x_2	...	x_T	y_1	y_2	...	y_T	ep_1	ep_2	...	ep_{t-1}	ep_t	...	ep_T	f_1	f_2	...	f_T	d_t
-I	0	...	0	0	...	0	I	0	...	0	0	...		0	0	...					0	0	0	...	0	$d_1 - e_0$
I	-I	...	0	0	...	0	0	I	...	0	0	...		0	0	...					0	0	0	...	0	d_2
					...						0	...		0	0	...					0			...		...
0	0	...	-I	0	...	0	0	0	...	0	0	...		0	0	...					0	0	0	...	0	d_{t-1}
0	0	...	I	-I	...	0	0	0	...	0	0	...		0	0	...					0	0	0	...	0	d_t
...						...			...			...				...								...		...
0	0	...	0	0	...	-I	0	0	...	I	0	...		0	0	...					0	0	0	...	0	d_T
0		...				0	-R	0	...	0	A	0	...	0	-I	0	...	0	0	...	0	0	0	...	0	0
0		...				0	0	-R	...	0	0	A	...	0	I	-I	...	0	0	...	0	0	0	...	0	0
0		...				0							...				...			...				...		...
0		...				0	0	0	...	-R	0	0	...	A	0	0	...	0	0	...	-I	0	0	...	0	0
0		...				0	0	0	...	0	v^T	0	...	0	0	0	...	0	0	0	0	1	0	...	0	u_1
0		...				0	0	0	...	0	0	v^T	...	0	0	0	...	0	0	0	0	0	1	...	0	u_2
		...				...							...				...					...		...		...
0		...				0	0	0	...	0	0	0	...	v^T	0	0	...	0	0	0	0	0	0	...	1	u_T

Tabela 3.2: Matriz de Restrições Reordenada para o Modelo Combinado

Para uma melhor visualização dos blocos matriciais, consideraremos um exemplo com $T = 2$ períodos. Obtemos então a seguinte matriz:

e_1	e_2	x_1	x_2	y_1	y_2	ep_1	ep_2	f_1	f_2	d_t
-I	0	I	0	0	0	0	0	0	0	$d_1 - e_0$
I	-I	0	I	0	0	0	0	0	0	d_2
0	0	-R	0	A	0	-I	0	0	0	0
0	0	0	-R	0	A	I	-I	0	0	0
0	0	0	0	v^T	0	0	0	1	0	u_1
0	0	0	0	0	v^T	0	0	0	1	u_2

Para a matriz acima, temos os respectivos números de linhas e de colunas:
linhas: $2M + 2P + 2 = 2(M + P + 1)$,
colunas: $4M + 2N + 2P + 2 = 2(2M + N + P + 1)$.

3.4 Exemplo Numérico

Considere dois produtos finais constituídos por três tipos de peças diferentes e três períodos. Então, temos os seguintes valores:

$t = 1, 2, 3$ número de períodos ($T = 3$).
$p = 1, 2, 3$ número de tipos de peças a serem cortadas (P=3).
$i = 1, 2$ número de produtos finais (M=2).
$j = 1, 2, 3, 4, 5$ número de padrões de corte ($N = 5$).

Seja a matriz R

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{bmatrix}.$$

Sendo que cada coluna da matriz representa um produto final.

r_{11}: quantidade de peças tipo 1 necessária para formar o produto final 1.

r_{21}: quantidade de peças tipo 2 necessária para formar o produto final 1.

r_{31}: quantidade de peças tipo 3 necessária para formar o produto final 1.

r_{pi}: quantidade de peças tipo p necessária para formar o produto final i.

Neste caso, a matriz $R_{P\times M}$ tem dimensão 3×2, e suponha que seja dada por:

$$R = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}.$$

Para a matriz A vamos definir os seguintes padrões de corte:

Para o *produto final 1*: 1 peça do tipo 1 e 2 peças do tipo 2.

Para o *produto final 2*: 1 peça do tipo 2 e 2 peças do tipo 3.

Lembrando que a matriz A possui dimensão $P \times N$, e que estamos considerando 5 padrões de corte (N=5), então, para este caso, a dimensão de A é 3×5, e pode ser escrita como

$$A = \begin{bmatrix} 1 & 3 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 4 \\ 1 & 0 & 2 & 2 & 0 \end{bmatrix}.$$

Sendo que cada coluna da matriz representa um padrão de corte e a_{pj} representa o número de peças tipo p no padrão j.

padrão 1: 1 peça do tipo 1, 1 peças do tipo 2, 1 peça do tipo 3.

padrão 2: apenas 3 peças do tipo 1.

padrão 3: 2 peças do tipo 1, 1 peça do tipo 2, 2 peças do tipo 3.

padrão 4: apenas 2 peças do tipo 3.

padrão 5: apenas 4 peças do tipo 2.

Sabemos que o vetor v^T possui dimensão $1 \times N$. Como estamos considerando $N = 5$, então, para este caso, a dimensão de v é 1×5, e pode ser o seguinte:

$$v^T = \begin{bmatrix} 4 & 2 & 3 & 2 & 5 \end{bmatrix}$$

Sendo que v_j^T representa o tempo gasto para cortar a placa no padrão j.

3.4.1 Matriz de Restrições

Variáveis

e_{it}: quantidade de produto final i em estoque no final do período t.

x_{it}: quantidade de produto final i no período t.

y_{jt}: quantidade de placas cortadas usando o padrão j no período t.

ep_{pt}: quantidade de peças tipo p em estoque no período t.

d_t	e_1	e_2	e_3	x_1	x_2	x_3	y_1	y_2	y_3	ep_1	ep_2	ep_3	f
$d_1 - I_0$	$-I_{2\times2}$	$0_{2\times2}$	$0_{2\times2}$	$I_{2\times2}$	$0_{2\times2}$	$0_{2\times2}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
d_2	$I_{2\times2}$	$-I_{2\times2}$	$0_{2\times2}$	$0_{2\times2}$	$I_{2\times2}$	$0_{2\times2}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
d_3	$0_{2\times2}$	$I_{2\times2}$	$-I_{2\times2}$	$0_{2\times2}$	$0_{2\times2}$	$I_{2\times2}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times5}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
0	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$-R_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$A_{3\times5}$	$0_{3\times5}$	$0_{3\times5}$	$-I_{3\times3}$	$0_{3\times3}$	$0_{3\times3}$	$0_{3\times3}$
0	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$-R_{3\times2}$	$0_{3\times2}$	$0_{3\times5}$	$A_{3\times5}$	$0_{3\times5}$	$I_{3\times3}$	$-I_{3\times3}$	$0_{3\times3}$	$0_{3\times3}$
0	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$0_{3\times2}$	$-R_{3\times2}$	$0_{3\times5}$	$0_{3\times5}$	$A_{3\times5}$	$0_{3\times3}$	$I_{3\times3}$	$-I_{3\times3}$	$0_{3\times3}$
C_1	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$v_{1\times5}$	$0_{1\times5}$	$0_{1\times5}$	$0_{1\times3}$	$0_{1\times3}$	$0_{1\times3}$	
C_2	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times5}$	$v_{1\times5}$	$0_{1\times5}$	$0_{1\times3}$	$0_{1\times3}$	$0_{1\times3}$	$I_{3\times3}$
C_3	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times2}$	$0_{1\times5}$	$v_{1\times5}$	$0_{1\times5}$	$0_{1\times3}$	$0_{1\times3}$	$0_{1\times3}$	

Tabela 3.3: Matriz de Restrições do Exemplo dado

Esta matriz tem as seguintes dimensões:

número de linhas: 18 $\quad T(M + P + 1) = 3(2 + 3 + 1) = 18$

número de colunas: 36 $\quad T(2M + N + P) = 3(4 + 5 + 3) = 36$, mais as três colunas das variáveis de folga, totalizando 39 colunas.

Capítulo 4

Considerações sobre a Matriz de Restrições

Neste capítulo apontaremos algumas considerações relevantes da matriz de restrições para construir, em seguida, sua base inicial e apresentar um exemplo numérico desta base. Por fim, analisamos as propriedades da base da matriz de restrições, verificando quais conjuntos de colunas da matriz podem pertencer à base e o que pode ocorrer com respeito à decomposição LU.

Podemos considerar 5 blocos referentes à matriz de restrições, os quais denominamos:

1. *bloco produção*: é o bloco referente às colunas das variáveis de produção x_{it}

2. *bloco item*: é o bloco referente às colunas das variáveis de estoque de itens (produtos finais)e_{it}.

3. *bloco geração*: é o bloco correspondente aos padrões de corte.

4. *bloco peça*: é o bloco referente às colunas das variáveis de estoque de peças ep_{pt}.

5. *bloco folga*: é o bloco de dimensão $T \times T$ referente às colunas das variáveis de folga f, ou variáveis artificiais.

4.1 Base inicial para a matriz de restrições

Uma base inicial pode ser tomada considerando vazios os estoques de peças e de produtos finais, e também considerando a matriz A como uma matriz diagonal, através da escolha dos padrões homogêneos. A partir dessas hipóteses obtemos a seguinte base inicial:

x_1	x_2	...	x_T	y_1	y_2	...	y_T	f
$I_{M\times M}$	0	...	0	0	0	...	0	0
0	$I_{M\times M}$	...	0	0	0	...	0	0
		...				...		
0	0	...	$I_{M\times M}$	0	0	...	0	0
$-R_{P\times M}$	0	...	0	$D_{P\times P}$	0	...	0	0
0	$-R_{P\times M}$	...	0	0	$D_{P\times P}$	...	0	0
		...				...		
0	0	...	$-R_{P\times M}$	0	0	...	$D_{P\times P}$	0
0	0	...	0	$v^T_{B_{1\times P}}$	0	...	0	
0	0	...	0	0	$v^T_{B_{1\times P}}$	...	0	I
0	0	...	0	0	0	...	$v^T_{B_{1\times P}}$	

Sendo D uma submatriz diagonal de A de dimensão $P\times P$ e $v^T_{B_{1\times P}}$ representa uma partição do vetor v^T cujas colunas são correspondentes às colunas de D.

4.1.1 Propriedades da matriz base inicial

1. A matriz base inicial é quadrada:
   número de linhas: $TM + TP + T = T(M + P + 1)$,
   número de colunas: $T(M + P + 1)$.

2. As colunas são todas linearmente independentes, ou seja, a matriz base inicial é não singular, pois esta matriz é triangular inferior e todos os elementos da diagonal são diferentes de zero. Portanto, não é necessário decompor esta base.

3. o *bloco item* não pertence à base inicial acima, pois o estoque de produtos finais é inicialmente considerado vazio.

4. o *bloco peça* não pertence à base inicial acima, pois o estoque de peças também é inicialmente considerado vazio.

5. as variáveis de folga agora podem ser tomadas como *variáveis artificiais*, caso a solução não seja factível (Fase I). As colunas das variáveis artificiais possuem então, sinal negativo, ao contrário das variáveis de folga.

6. Logo podemos iniciar o método simplex com esta base.

4.2 Exemplo Numérico de Base Inicial

Retomando o exemplo numérico do capítulo anterior, tínhamos a seguinte matriz A:

$$A = \begin{bmatrix} 1 & 3 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 4 \\ 1 & 0 & 2 & 2 & 0 \end{bmatrix}$$

Para construirmos a base inicial deste exemplo, devemos tomar a matriz D diagonal, a partir das colunas de A. Portanto, consideramos a submatriz:

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

e o vetor v_B^T com as colunas correspondentes:

$$v_B^T = \begin{pmatrix} 2 & 5 & 2 \end{pmatrix}$$

Podemos então escrever a matriz base inicial para o exemplo numericamente, da seguinte forma:

x_1	x_2	x_3	y_1	y_2	y_3	f
$I_{2\times2}$	0	0	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
0	$I_{2\times2}$	0	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
0	0	$I_{2\times2}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$	$0_{2\times3}$
$-R_{3\times2}$	0	0	$D_{3\times3}$	0	0	$0_{3\times3}$
0	$-R_{3\times2}$	0	0	$D_{3\times3}$	0	$0_{3\times3}$
0	0	$-R_{3\times2}$	0	0	$D_{3\times3}$	$0_{3\times3}$
$0_{1\times2}$	0	0	$v^T_{1\times3}$	0	0	
0	$0_{1\times2}$	0	0	$v^T_{1\times3}$	0	$I_{3\times3}$
0	0	$0_{1\times2}$	0	0	$v^T_{1\times3}$	

Esta matriz nos dá a visualização da dimensão de cada um dos blocos. Ao adotarmos os valores numéricos de cada elemento da matriz, obtemos a seguinte base inicial numérica:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & -1 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 & 2 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 & 2 & 0 & 0 & 1
\end{bmatrix}.$$

4.3 Propriedades da matriz de restrições

A seguir, podemos apontar algumas propriedades relevantes da matriz de restrições especialmente quanto à sua decomposição e, observando a forma geral da matriz de restrições, podemos apontar algumas propriedades para a construção da base.

1. Cada coluna da variável y pode ter no máximo $P + 1$ colunas na base. Isto nos sugere o seguinte teorema:

   Teorema 4.1 *Um subconjunto de colunas das matrizes*

   $$\begin{pmatrix} A \\ v^T \end{pmatrix}$$

tem no máximo $T(P+1)$ colunas na base.

Prova: Cada matriz A possui dimensão $P \times N$ e cada vetor v^T possui dimensão $1 \times N$. Então cada coluna y possui $P+1$ linhas e N colunas, e todo o conjunto possui $T(P+1)$ linhas e NT colunas. Assim, se cada variável y tiver $P+1$ colunas, este bloco será quadrado. Caso tenha menos de $P+1$ colunas, podemos tomar o bloco de folga como pivô, e o conjunto de matrizes acima é retangular em pé, possuindo mais linhas que colunas. Se cada coluna da variável y tiver mais que $P+1$ colunas, a base seria singular. Portanto, cada coluna de y tem no máximo $P+1$ colunas na base. Logo, todo o conjunto acima deve ter no máximo $T(P+1)$ colunas na base.

2. As matrizes formadas pelas colunas ep_{pt} são sempre triangulares inferiores pela restrição $-Rx_t + Ay_t + ep_{t-1} - ep_t = 0$, e no reordenamento de colunas proposto nunca são decompostas, por conterem apenas dois blocos matrizes identidades.

3. O *bloco folga* também não é afetado pela decomposição, pois possui apenas colunas unitárias e o elemento não nulo de cada coluna básica será tomado como pivô pela decomposição.

4. Consequentemente, o esforço computacional da decomposição LU para o reordenamento proposto existe de fato apenas em $\begin{pmatrix} A \\ v^T \end{pmatrix}$ e em R.

5. Entre e_t e x_t deve haver pelo menos M colunas na base. Caso contrário, teríamos uma linha de zeros na matriz, e então ela seria linearmente dependente. Esta afirmação sugere o seguinte teorema:

Teorema 4.2 *Se $e_t(i)$ não está na base $\Rightarrow$ $x_t(i)$ está na base e podemos tomar $x_t(i)$ como próximo pivô, onde i é a coluna desta matriz, onde existem estoque e produção simultaneamente, caso $x_t(i)$ pertença à base, para $i = 1, \ldots, j$, sendo j o primeiro estágio de tempo. Para os estágios de tempo seguintes, se $x_t(k)$ não está na base, podemos tomar o estoque deste mesmo produto no tempo anterior $e_t(k-1)$, para $k = j+1, \ldots, M$.*

Prova: Se nenhuma das colunas $e_t(i)$ e $x_t(i)$ estiver na base, teremos uma linha de zeros na matriz. Se ambas estiverem, tomamos o primeiro elemento de $-R$ como pivô. E se apenas a coluna $x_t(i)$ estiver na base, e não a coluna $e_t(i)$, o próximo pivô será $x_t(i)$. Quanto aos estágios de tempo seguintes, se $x_t(k)$ não está na base, podemos tomar a coluna de estoque deste produto final corresponde ao período anterior, pois esta conterá uma matriz identidade, devido à restrição $x_t + e_{t-1} - e_t = d_t$, para $t = 1 \ldots T$, o que evitará que a base seja singular.

6. Como consequência do teorema acima, podemos escrever o seguinte corolário:

Corolário 4.1 *A base deve conter pelo menos TM colunas entre os blocos item e produção.*

Na seção seguinte veremos que é possível desenvolver um algoritmo onde, dadas a coluna que sai da base e a coluna que entra, temos como descobrir a qual bloco pertencem estas colunas.

4.4 Identificação do Tipo das Colunas

Descrevemos a seguir um algoritmo onde, dada a coluna que sai da base e a coluna que entra, podemos identificar facilmente a qual bloco pertencem estas colunas, utilizando sempre a posição do primeiro e do último elemento não nulo. Portanto, são necessárias poucas alterações em uma implementação pré-existente que não considera a estrutura particular da matriz de restrições. Pode-se incluir a atualização da decomposição LU proposta em uma implementação, bastando que as colunas que entram e saem da base sejam fornecidas.

4.4.1 Algoritmo

Dada uma coluna c, obtemos a posição do seu primeiro e último elementos não nulos k_1 e k_2:

se $K_1 \leq TM$

então se $K_2 > TM$

então $c \in$ produção

senão $c \in$ estoque de item

senão se $K_1 > T(P + M)$

então $c \in$ folga

senão se $K_2 > T(P + M)$

$c \in$ padrão de corte

senão $c \in$ estoque de peça.

4.4.2 Teoremas sobre trocas de colunas

Analisando quais colunas podem entrar na base a partir da saída de uma determinada coluna, podemos deduzir os seguintes teoremas:

Teorema 4.3 *Se a coluna a sair da base pertence ao estoque de item, apenas a coluna de produção correspondente pode entrar na base, ou a coluna de produção do período anterior.*

Prova: Se uma coluna do estoque de item sai da base, pode ser inserida uma coluna correspondente à produção, pois teremos estoque e produção simultaneamente. Se não entrar uma coluna de produção correspondente à coluna de estoque de item que saiu da base, esta pode se tornar singular. Para o primeiro estágio de tempo, se a coluna de estoque de item sai da base, deve entrar a coluna de produção correspondente, senão teremos uma linha de zeros na base. Para os estágios de tempo seguintes, podemos tomar a coluna de produção do período anterior, pois esta conterá elementos da matriz identidade, evitando que a base seja singular. Uma coluna de estoque de item não pode ser trocada por uma coluna de padrão, estoque de peças ou folga, pois estas contém elementos não nulos na metade inferior da matriz, justamente onde o bloco estoque de item contém apenas elementos nulos, estando os não nulos em sua parte superior. Assim, a base teria linhas de zeros, sendo portanto, singular.

Teorema 4.4 *Se a coluna a sair da base pertence ao padrão de corte, colunas pertencentes a produção, estoque de peças e folga podem entrar na base. Colunas pertencentes ao estoque de item podem ser inseridas contanto que pelo menos uma coluna de produção já esteja na base.*

Prova: Se uma coluna de padrão de corte sai da base, pode ser inserida uma coluna pertencente à produção, pois esta conterá elementos não nulos nas partes superior e inferior da matriz, impedindo que ocorra linhas nulas na base. Também podem ser inseridas colunas de estoque de peças e de folga, pois, apesar de possuírem elementos não nulos apenas na metade inferior da matriz, a base não será singular, devido as colunas de produção ou estoque de item que obrigatoriamente estarão na base, impedindo que as linha superiores sejam nulas. Uma coluna do estoque de item só poderá ser inserida se houver pelo menos uma coluna de produção na base, pois esta conterá elementos não nulos na parte superior e inferior da matriz, impedindo que ocorra uma linha de zeros na base, ao contrário das colunas de estoque de item que possuem elementos não nulos apenas na parte superior.

Teorema 4.5 *Se a coluna a sair da base pertence à produção, pode entrar na base a coluna de estoque de item correspondente ao período anterior. Podem ser inseridas colunas pertencentes ao padrão, estoque de peças ou folga, contanto que a coluna de estoque de item correspondente à coluna de produção que saiu da base, ou a correspondente ao período anterior pertença à base.*

Prova: Se sai da base uma coluna de produção, podemos tomar a coluna de estoque de item correspondente ao período anterior, pois esta conterá elementos da matriz identidade não nulos, evitando a singularidade da base. Podem ser inseridas colunas de padrão, estoque de peças ou folga, contanto que a coluna de estoque de item correspondente à coluna de produção que saiu da base, ou a correspondente ao período anterior pertença à base, pois caso contrário, teremos uma linha de zeros fazendo com que a base se torne singular.

Teorema 4.6 *Se a coluna a sair da base pertence ao estoque de peças, podem ser inseridas na base colunas pertencentes a: folga ou estoque de item se uma coluna de padrão ou*

produção referentes ao mesmo estágio de tempo da coluna que saiu, esteja na base. Produção ou padrão se a coluna de produção a entrar na base for correspondente ao mesmo período, ou já estiver na base uma coluna de padrão correspondente ao mesmo período, ou uma coluna de estoque de peças correspondente ao período anterior estiver na base.

Prova: Se sai da base uma coluna de estoque de peças, podem ser inseridas colunas de folga ou estoque de item, com a condição de que uma coluna referente ao padrão ou à produção que seja do mesmo estágio de tempo da coluna de estoque de peças que foi retirada pertença à base. Caso contrário, seria inevitável a ocorrência de uma linha nula na base, tornando-a singular. Uma vez que devem haver pelo menos TM colunas na base entre os blocos item e produção (Corolário 4.1), e como está entrando na base uma coluna de estoque de item, é muito provável que esta coluna correspondente de produção esteja na base. Podem ser inseridas colunas pertencentes ao padrão ou produção, se obedecidas uma das restrições citadas no teorema, pois se nenhuma delas for satisfeita, teremos uma linha nula na base, o que a torna singular.

Teorema 4.7 *Se a coluna a sair da base pertence à folga, podem ser inseridas na base colunas pertencentes ao estoque de item, produção ou estoque de peças se uma coluna pertencente ao padrão correspondente ao mesmo período estiver na base. Podem entrar colunas de padrão se a coluna que entra for do mesmo período de tempo da coluna que sai.*

Prova: Se a coluna a sair da base pertence à folga, podem ser inseridas na base colunas pertencentes ao estoque de item, produção ou estoque de peças se uma coluna pertencente ao padrão correspondente ao mesmo período estiver na base, pois só desta forma podemos garantir que não haverá uma linha de zeros na base. Isto porque as colunas de folga possuem elementos não nulos apenas nas últimas linhas da matriz e as colunas de estoque de item, estoque de peças e produção possuem elementos não nulos nas linhas onde o bloco folga só possui zeros. Desta forma, para que a base seja não singular, deve existir uma coluna de padrão de corte do mesmo período da coluna de folga que saiu, pois esta conterá elementos não nulos na mesma linha desta coluna de folga. Podem entrar colunas de padrão contanto que a coluna que entrar seja do mesmo período da coluna de folga que sair, pois caso contrário, teremos uma linha nula na base, tornando-a singular.

Com base nestes teoremas, dada a coluna que sai, podemos saber a quais blocos pertencem as colunas que podem ser inseridas na base.

Capítulo 5

Experimentos Numéricos

O objetivo destes experimentos é verificar a viabilidade das idéias apresentadas neste trabalho. Tomamos como teste a matriz de restrições considerando:
$T = 6$ períodos de tempo;
$M = 2$ produtos finais;
$P = 5$ quantidade de peças diferentes;
$N = 60$ número de padrões de corte.

Com esses parâmetros, a matriz de restrições possui dimensão 48×420, sendo 360 colunas correspondentes aos padrões de corte. Para desenvolvermos os experimentos, definimos as matrizes B_1, B_2 e B_3. Inicialmente todas elas correspondem à base inicial. Vamos simular iterações do *método simplex* escolhendo aleatoriamente uma coluna para sair da base e uma para entrar na base. A cada iteração as três matrizes contém as mesmas colunas básicas, embora em ordem diferente. As colunas em B_1 respeitam o ordenamento estático proposto neste trabalho, descrito na Tabela 3.2. Na matriz B_2 sempre colocamos a coluna que entra na base na posição da coluna que sai e na matriz B_3 a coluna que entra na base ocupa a última posição da matriz.

Os experimentos foram realizados no MATLAB 5.3, em um microcomputador pentium 4 com processador INTEL 1.8GHz e 512MB de memória RAM.

5.1 Critério para troca de colunas

Realizamos trocas das colunas básicas por colunas não básicas, testando a singularidade e a esparsidade. Sorteamos então uma coluna que sairá da base e uma coluna que será inserida. Se a nova base for não singular, atualizamos as posições das colunas básicas em B_1, B_2 e B_3, da forma descrita anteriormente; caso contrário, sorteamos outra coluna para entrar na base até que encontremos uma matriz não singular. Efetuamos a seguir a decomposição LU de B_1, B_2 e B_3, e calculamos o número de elementos não nulos dessas decomposições.

Esperamos que B_1 seja mais esparsa que B_2 e B_3, porque ela é ordenada de forma que sua decomposição cause menor preenchimento na matriz.

É relevante ressaltar que para um número pequeno de produtos finais pode não ser vantajoso manter as colunas ordenadas, pois o número de elementos nulos das decomposições é elevado, não apresentando diferenças consideráveis.

5.2 Resultados Observados

Apresentamos nas tabelas seguintes os resultados obtidos pelos experimentos numéricos. Os valores *min*, *max* e *média* nas Tabelas 5.1 e 5.2 representam, respectivamente, o mínimo, o máximo e a média do número de elementos não nulos das matrizes L e U (nnz(L+U)) da decomposição das bases correspondentes. A singularidade da matriz é verificada pela função $rcond(\,)$ do MATLAB, e o cálculo da decomposição LU foi realizado usando o comando interno $lu(B, 0.1)$, que reordena as linhas para a seleção do pivô, com *threshold* $u = 0,1$. Antes da primeira troca de colunas, as matrizes B_1, B_2 e B_3 são iguais, e neste exemplo, $nnz(L + U) = 282$.

Os resultados apresentados consideram diversos números de simulação de iterações do simplex.

Os resultados obtidos para as 500 primeiras iterações foram:

nnz(L+U)	B_1	B_2	B_3
min	310	301	389
media	351.4	377.6	351.6
max	393	556	401

Tabela 5.1: Número de elementos não nulos da decomposição LU das bases para as 500 primeiras iterações.

Os resultados obtidos para as 10000 primeiras iterações foram:

nnz(L+U)	B_1	B_2	B_3
min	293	293	140
media	349.5	382.6	354.1
max	416	648	491

Tabela 5.2: Número de elementos não nulos da decomposição LU das bases para as 10000 primeiras iterações.

Observamos que a partir de 2000 iterações a diferença entre os números de elementos não nulos das decomposições das bases B_1, B_2 e B_3 atingem um certo equilíbrio, não sofrendo alterações importantes para um número maior de iterações.

Finalmente a Tabela 5.3 compara os números de elementos não nulos da decomposição de B1, já que esta se mostrou mais esparsa, para as 500 primeiras iterações utilizando diversos valores de limiar de pivoteamento.

nnz(L+U)	$B_1, 1$	$B_1, 0.1$	$B_1, 0.01$
min	304	310	257
max	429	393	415
média	388.2	351.4	356.9

Tabela 5.3: Número de elementos não nulos da decomposição LU de B1 para 500 iterações.

Para este nosso exemplo, relativamente pequeno, o valor do limiar $(B_1, 0.1)$ apresentou melhores resultados que os demais, por ter se mostrado mais esparso.

5.3 Decomposição LU com Troca de Colunas

Implementamos em MATLAB uma decomposição LU que considera a estrutura esparsa da matriz de restrições, e comparamos nossos resultados com o cálculo da decomposição LU completa, realizada através do comando interno do MATLAB $lu(\,)$, que não explora a esparsidade da matriz.

Como a base B_1 havia se mostrado mais esparsa que as bases B_2 e B_3 nos experimentos anteriores, comparamos então o número de operações necessárias para decompormos esta base, através da contagem de "flops", ou seja, *contagem de operações de ponto flutuante*, das formas de decomposição. A comparação foi feita usando três formas de decomposição diferentes.

A primeira é a decomposição LU completa, que não explora a estrutura matricial, ou seja, a esparsidade da matriz, que chamamos de "lu(B)". Na segunda forma de decomposição, a base da matriz de restrições é redecomposta a partir da posição da primeira coluna que entrar ou sair da base. Para tanto, tomamos a posição na base da coluna que sai (s) e da que entra (e), e escolhemos a menor delas (j). Na Tabela 5.4 , esta decomposição está indicada como "$min(e, s)$". A terceira e última forma, que chamamos "dec.esparsa(B)", redecompõe apenas as colunas necessárias, pois a saída de uma coluna pode não alterar as demais colunas pela sua estrutura esparsa.

Nesta nova decomposição, se $s < e$, estamos interessados em saber quantas colunas a partir de $s+1$ não são afetadas pela saída desta coluna s. A decomposição é feita apenas nas colunas afetadas, isso porque a saída de uma coluna pode não afetar colunas depois dela pela sua estrutura *esparsa*. Ou seja, as posições dos elementos não nulos da coluna s podem não coincidir com os elementos não nulos de alguma coluna seguinte, e então, essa coluna não sofrerá qualquer alteração de s, e então não será decomposta.

As operações que foram causadas pela coluna que saiu são *desfeitas*. Para tanto, efetuamos as operações na ordem inversa da decomposição de k, sendo $k = s+1 \ldots e-1$, sempre considerando a esparsidade. Também decompomos e, a coluna que entra na base, e contamos o número de flops desta operação.

Agora, se $e < s$, isto é, se a coluna a entrar na base for inserida em alguma posição anterior à posição da coluna a sair da base, se necessário, decompomos novamente k, neste caso, para $k = e + 1 \ldots s - 1$, e calculamos o número de operações.

Finalmente, resta-nos decompor as últimas colunas da base, ou seja, devemos decompor a partir de $max(e, s)$ até a última coluna da base, m. Então, se necessário, fazemos a decomposição da coluna k, neste caso, para $k = max(e, s) \ldots m$, e calculamos o número de flops desta atualização.

5.3.1 Comparação dos Resultados

A Tabela 5.4 abaixo exibe a comparação do número de "flops"(contagem de operações de ponto flutuante) entre a decomposição proposta neste trabalho (dec.esparsa(B)), a decomposição a partir da primeira coluna afetada por j, e a decomposição completa, utilizando o comando do MATLAB.

Os resultados a seguir mostram o mínimo, a média e o máximo de flops entre as duas decomposições, para 500 iterações.

flops	$lu(B)$	$min(e, s)$	$dec.esparsa(B)$
min	1241	37	7
max	1477	391	71
média	1352.9	185.1	33.7

Tabela 5.4: Contagem de operações entre as decomposições de B_1

Podemos concluir que a decomposição LU proposta neste trabalho reduz o número de operações em aproximadamente 97% se comparada à versão que não explora a decomposição da base anterior, podendo realizá-la de maneira mais rápida e eficiente. Mesmo explorando parcialmente a esparsidade, como acontece na decomposição indicada por $min(e, s)$, onde a base é redecomposta a partir da posição da primeira coluna que entrar ou sair da base, já temos um ganho significativo de em média, 86% se comparada à decom-

posição completa. Como vimos, podemos obter um resultado ainda melhor quando exploramos totalmente a esparsidade, como é mostrado na tabela 5.4 com a dec.esparsa(B).

5.3.2 Estimativa do Erro da Atualização

Testes computacionais adicionais foram realizados com o objetivo de verificar a robustez do método proposto. Para isso, todas as operações da decomposição foram "desfeitas" em todas as iterações do método simplex. Ao efetuarmos as operações na ordem inversa da decomposição LU obtemos uma matriz que após a troca das colunas que entra e sai, simulará a atualização da decomposição da base ao ser decomposta desde a primeira coluna. Optamos por desfazer toda a decomposição, com o objetivo de ilustrar a robustez do método de atualização proposto pois estamos simulando o pior caso em termos do número de operações realizadas, após cada iteração.

Como medida do erro introduzido por estas operações, calculamos a norma entre a matriz, obtida a partir da função que desfaz as operações da decomposição, e a base reordenada, obtida diretamente da matriz de restrições, para várias iterações do simplex.

Outra maneira de recuperar a matriz base inicial é efetuar a operação L*U, onde L é a matriz triangular inferior com elementos da base, e U a matriz triangular superior.

Os erros máximo e médio obtidos para 100, 1000, 2000 e 10000 iterações são apresentados na Tabela 5.5. As duas primeiras linhas exibem o erro acumulado para a operação que desfaz toda a decomposição, e as duas últimas linhas exibem do erro da recuperação da base através da multiplicação de L por U.

erro	100	1000	2000	10000
max (desfaz)	6,8 e-13	1,3 e-12	1,8 e-12	4,0 e-12
média (desfaz)	1,6 e-13	6,1 e-13	5,8 e-13	7,0 e-13
max (L*U)	1,1398 e-12	2,2397 e-12	2,8235 e-12	3,6005 e-12
media (L*U)	4,2917 e-13	1,1003 e-12	9,9500 e-13	1,1265 e-12

Tabela 5.5: Estimativa do erro de aproximação da atualização da base

Podemos verificar 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 da ordem de 10^{-12} em relação às colunas originais. Estes resultados são ainda mais significativos ao verificarmos que o erro ao recuperarmos as matrizes utilizando o produto L*U é na média ligeiramente superior que a operação de desfazer as decomposição. Isto significa que não é necessário calcular periodicamente uma decomposição da base, para esta aplicação, como em outros métodos de atualização pois a abordagem proposta é extremamente robusta.

5.4 Conclusões

Neste trabalho foi apresentado o problema combinado, que acopla dois importantes e conhecidos problemas de otimização linear, o problema de dimensionamento de lotes e o de corte de estoque. Uma formulação matemática deste problema foi feita, e também foi exibida sua matriz de restrições e uma base inicial para ela.

Construímos uma base esparsa para esta matriz de restrições através de um reordenamento estático das colunas básicas, o que nos proporcionou uma decomposição esparsa para qualquer base, e por ser estático, não requer esforço computacional para obter a ordem das colunas. Pelo reordenamento, a matriz adquire um formato bloco diagonal, que facilita a decomposição das bases futuras e evita, desta forma, o preenchimento (*fill-in*) da matriz.

Através dos experimentos numéricos descritos no capítulo 5, concluímos que a estratégia B_1 apresentou melhores resultados em comparação com as estratégias B_2 e B_3, pois, por exemplo, para 10000 iterações, uma das respostas obtidas foi que o máximo de elementos não nulos na decomposição de B_1 foi 416, enquanto que para B_2 foi 648 e para B_3 foi 491. Como a base possui 2304 elementos, esses valores correspondem, respectivamente, a 82%, 72% e 79% de esparsidade. Podemos observar que a estratégia B_3 assume uma posição intermediária entre as estratégias B_1 e B_2.

Além disso, a utilização de $lu(B1, 0.1)$ geralmente apresenta melhores resultados

se comparados com $lu(B1, 1)$ e com $lu(B1, 0.01)$, obtendo uma diferença maior entre os números de elementos não nulos da decomposição da matriz reordenada B1 e a não reordenada. Desta forma, B_1 contribui para a redução de tempo computacional, podendo resolver problemas de otimização linear de maneira mais eficiente.

Concluímos que o reordenamento estático das colunas da matriz de restrições apresenta várias vantagens computacionais:

- A base inicial, descrita na Seção 4.1 é triangular para a solução inicial proposta. Desta forma, as decomposições se tornam baratas, em termos computacionais, inclusive a primeira decomposição, que não requer nenhuma operação de ponto flutuante.

- Desta forma, a matriz reordenada se torna bloco diagonal, o que evita seu preenchimento durante a decomposição.

- O reordenamento estático leva à decomposições mais esparsas, conforme indicam as tabelas dos experimentos deste capítulo, e à atualização de decomposições extremamente baratas.

- O reordenamento inicial também não tem custo de inicialização e tampouco de atualização, pois como é estático, podemos obter uma decomposição esparsa para qualquer base, sem esforço computacional para ordenar as colunas.

- Por este reordenamento, a decomposição LU esparsa pode ser facilmente integrada a outros códigos já existentes, que não estejam explorando a estrutura específica da matriz. O software GLPK - *Gnu Linear Programming Kit*, por exemplo, consiste em uma biblioteca de programação linear com código fonte aberto, portanto, se torna mais simples a tarefa de integrá-lo à idéia proposta de reordenamento estático neste trabalho.

  Este software pode ser encontrado em www.gnu.org/software/glpk/glpk.html.

- A Tabela 5.5 apresenta resultados estáveis, onde o erro absoluto entre a matriz obtida desfazendo-se repetidamente toda a decomposição LU e a base reordenada resultante das iterações do método simplex é totalmente aceitável. Dessa forma,

concluímos que o método proposto é bastante robusto não necessitando de decomposições periódicas da base.

- Esta última conclusão implica que por este método de atualização nenhuma decomposição LU necessita ser calculada durante toda a execução do método simplex se partimos da base inicial descrita na Seção 4.1.

Assim, a proposta de construção da base estática esparsa leva a excelentes resultados computacionais, com respeito tanto a robustez quanto à eficiência computacional em comparação com abordagens que não consideram a estrutura esparsa da matriz de restrições.

Capítulo 6

Perspectivas de Continuidade

Como perspectivas futuras, podemos sugerir para continuidade e melhoria deste trabalho:

1. As implementações podem ser feitas em linguagem C, Pascal, ou Fortran, para um melhor desempenho em termos computacionais.

2. Pode-se integrar a decomposição LU proposta neste trabalho a códigos já existentes, que não estejam explorando a estrutura específica da matriz de restrições, como por exemplo o GLPK. Através desta comparação deve-se comprovar a eficiência do método na resolução de problemas de otimização linear.

3. Pode-se considerar o problema de encontrar uma solução factível para problemas de dimensionamento de lotes com capacidade limitada que considere tempo de preparação *(setup time)*. No caso do problema de dimensionamento de lotes monoestágio com capacidade infinita de produção, a minimização dos custos trata do balanceamento dos custos de produção e estocagem. Como citado neste trabalho, a referência clássica para resolução deste problema é [34]. Entretanto, se a quantidade de recursos a serem utilizados for limitada e incluído tempos de preparação, obtém-se o *problema de dimensionamento de lotes monoestágio capacitado com tempo de preparação*. Para este problema, em [33] foi formulado um método heurístico baseado na relaxação lagrangiana das restrições de capacidade. Em sua modelagem matemática, a função objetivo representará custos de produção, estoque e preparação; sujeita às restrições de balanço de estoque e limites de capacidade, onde se levam em consideração o tempo despendido para a produção dos itens e

preparação das máquinas. A matriz de restrições deste problema diferirá da matriz do problema combinado apenas em mais um novo grupo de variáveis, referentes às restrições de capacidade. Estas variáveis, que podem ser chamadas de z_{it} são iguais a 1 se existir produção do item i no período t, e serão iguais a 0 caso contrário. Assim, teremos mais restrições à função objetivo, as que denotam em quais períodos ocorre produção.

4. Podem ser realizados experimentos estabelecendo comparações entre a atualização da base proposta neste trabalho, onde, após uma troca de colunas básicas, atualiza-se tais colunas eficientemente, causando o menor preenchimento da matriz e obtendo-se uma base estática esparsa, e a atualização da decomposição LU proposta por Bartels - Golub, a qual utiliza o pivoteamento parcial, e suas variantes propostas por Reid, que procuram manter tanto a esparsidade original dos dados quanto a estabilidade numérica. Como o método proposto neste trabalho se apresentou bastante robusto, esperamos comprovar a eficiência do método de atualização da base esparsa, que causa o menor preenchimento na matriz.

A idéia geral do reordenamento estático proposto neste trabalho pode ser aplicada a outras classes de problemas com estrutura matricial particular.

Bibliografia

[1] M.N. Arenales A.G. Vianna. O problema de corte bidimensional com placa defeituosa. *Anais do XXXIV Simpósio Brasileiro de Pesquisa Operacional, IME - Rio de Janeiro, 8-11 de outubro*, 2002.

[2] M.N. Arenales and R. S. Hoto. Um problema de corte unidimensional com restrições de agrupamento e aplicações industriais. *I Oficina Nacional em Problemas de Corte e Empacotamento, São Paulo, SP*, pages 31–36, 1996.

[3] M.N. Arenales, R. Morábito, and H. Yanasse. *O problema de Corte e Empacotamento e Aplicações Industriais.* XX Congresso Nacional de Matemática Aplicada e Computacional, Gramado, 8 a 12 de setembro, 1997.

[4] R.H. Bartels. A stabilization of the simplex method. *Numer. Math.*, 16:414–434, 1969.

[5] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows.* John Wiley & Sons, 1990.

[6] Jordi Castro. A specialized interior-point algorithm for multcommodity network flows. *SIAM J. Optimization*, 10(3):852–877, 2000.

[7] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices.* Clarendon Press, Oxford, 1986.

[8] P. Gilmore and R. Gomory. Multistage cutting sotck problems of two and more dimensions. *Operation Research*, 14:1045–1074, 1965.

[9] G. H. Golub and C. V. Loan. *Matrix Computations.* Johns Hopkins, Segunda Edição, 1989.

[10] M.C.N Gramani. *Otimização do Processo de Cortagem Acoplado ao Planejamento da Produção*. Tese de Doutorado, Densis-Unicamp, 2001.

[11] R. W. Haessler. Selection and design of heuristic procedures for solving roll trim loss problems. *Management Science*, 34(12):1460–1471, 1988.

[12] R. S. Hoto. *Otimização no Corte de Peças Unidimensionais com Restrições de Agrupamento*. Dissertação de Mestrado, ICMC-USP, 1996.

[13] J. L. Kennington and R. V. Helgason. *Algorithms for Network Programming*. Wiley, New York, 1980.

[14] M. A. Pereira L. A. Lorena and S. N. Salomão. A relaxação lagrangeana/surrogate e o método de gerção de colunas: Novos limitantes e novas colunas. *Anais da V Oficina Nacional de Problemas de Corte e Empacotamento, São José dos Campos, SP*, pages 19–37, 2001.

[15] K. K. Fok L. C. Hendry and K. W. Shek. A cutting stock scheduling problem in the copper industry. *Operations Research*, 47:38–47, 1996.

[16] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

[17] H.M. Markowitz. The elimination form of the inverse and its applications to linear programming. *Management Science*, 3:255–269, 1957.

[18] F. P. Marques. *O Problema da Mochila Compartimentada*. Dissertação de Mestrado, ICMC-USP, 2000.

[19] R. Morabito and M. Arenales. Staged and constrained two-dimensional guillotine cutting problems: And and/or-graph approach. *European J. Operational Research*, 94:548–560, 1996.

[20] H. Yanasse N. Soma and N. Maculan. O problema de corte e empacotamento e aplicações industriais: O problema da mochila. *XX Congresso Nacional de Matemática Aplicada e Computacional, Gramado, RS*, pages 24–58, 1997.

[21] S. L. Nonas and A. Thorstenson. A combined cutting-stock and lot-sizing problem. *Operations Research*, 120(2):327–342, 2000.

[22] A.R.L. Oliveira and C. Lyra. Interior point methods for the polynomial L_∞ fitting problem. *Trabalho submetido à revista International Transaction in Operational Research*, 2002.

[23] A.R.L. Oliveira, M.A. Nascimento, and C. Lyra. Efficient implementation and benchmark of interior point methods for the polynomial L_1 fitting problem. *Statistics & Data Analysis*, 35(2):119–135, 2000.

[24] A.R.L. Oliveira, L. Nepomuceno, and S. Soares. Short term hydroelectric scheduling combining network flow and interior point approaches. *Trabalho submetido à Electrical Power & Energy Systems*, 2001.

[25] A.R.L. Oliveira, L. Nepomuceno, and S. Soares. Optimal active power dispatch combining network flow and interior point approaches. *Aceito para publicação, IEEE Transactions on Power Systems*, 2003.

[26] A.R.L. Oliveira and S. Soares. Métodos de pontos interiores para problema de fluxo de potência ótimo. *Anais do XIII Congresso Brasileiro de Automática, em CD-ROM, Florianópolis, SC*, pages 790–795, 2000.

[27] J.K. Reid. A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Mathematical Programming*, 24:55–69, 1982.

[28] M. P. Reinders. Cutting stock optimization and integral productions planning for centralized wood processing. *Mathematical Computer Modeling*, 16(1):37–55, 1992.

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

[30] M.A. Saunders. The complexity of lu upin the simplex method. *R.S. Anderssen and R.P. Brent, eds, The complexity of computacional problem solving, University Press, Queenslang*, pages 214–230, 1976.

[31] C.T.L. Silva. *Problemas de Otimização Linear Canalizados e Esparsos.* Dissertação de Mestrado, ICMC - USP, 2002.

[32] R. J. Vanderbei. *Linear Programming Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996.

[33] J.O. Mcclain W. Trigeiro, L.J. Thomas. Capacited lot sizing with setup times. *Management Science*, 35(3):353–366, 1989.

[34] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.