UMA ALTERNATIVA NA OBTENÇÃO DE MÉTODOS DE PONTOS INTERIORES ORIENTADOS A OBJETOS EXPLORANDO ESTRUTURAS ESPARSAS PARA PROBLEMAS DE OTIMIZAÇÃO

Anibal Tavares de Azevedo[1]
Aurelio Ribeiro Leite de Oliveira[2]
Secundino Soares Filho[1]
anibal@densis.fee.unicamp.br
aurelio@ime.unicamp.br
dino@densis.fee.unicamp.br
[1]Departamento de Engenharia de Sistemas, Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, Av. Albert Einstein, 400, Cidade Universitária, 13083-970, Campinas - SP - Brasil
[2]Departamento de Matemática Aplicada, Instituto de Matemática Estatística e Computação Científica, Universidade Estadual de Campinas, Praça Sérgio Buarque de Holanda, 651, Cidade Universitária, 13083-859, Campinas - SP - Brasil

Resumo. *Problemas de decisão relacionados a situações reais podem assumir grandes dimensões. Neste sentido, os métodos de pontos interiores são indicados na literatura como ferramentas rápidas e robustas, desde que aproveitem a estrutura esparsa associada ao problema em questão. Para a dedução dos métodos de pontos interiores duas abordagens podem ser utilizadas. Uma das abordagens apresenta reaproveitamento de código e rápida implementação computacional, mas depende da utilização de conceitos de programação orientada a objetos como polimorfismo. O propósito deste trabalho é o de fornecer uma extensão desta abordagem para a programação procedural. Além disso, são apresentadas aplicações da abordagem extendida para dois problemas reais. Um problema relacionado ao planejamento e controle da produção da manufatura e outro relacionado ao planejamento da operação de longo prazo de sistemas de potência.*

Keywords: *Otimização, Métodos de pontos interiores, Programação orientada a objetos*

1. INTRODUÇÃO

Problemas de tomada de decisão podem ser relacionados a modelos de otimização. Modelos de otimização relacionados a problemas reais podem assumir grandes dimensões. Neste caso, é necessário aproveitar a estrutura esparsa do problema. Como os métodos de pontos interiores são apresentados na literatura como métodos robustos e eficientes para problemas de grande porte, serão estudadas, neste trabalho, técnicas para dedução dos mesmos.

2. Métodos de Pontos Interiores

Historicamente, o desenvolvimento dos métodos de pontos interiores teve grande impulso devido à Karmarkar (Karmarkar, 1984), e aos bons resultados obtidos em (Adler et al., 1989). Hoje em dia os métodos de pontos interiores para programação linear estão bem estabelecidos tanto em termos de fundamentação teórica (Gonzaga, 1992; Wright, 1996; Tapia and Zhang, 1992), quanto prática (Gondzio, 1996; Mehrotra, 1994; Oliveira and Lyra, 1991), existindo uma série de códigos de boa qualidade disponíveis (Wright, 1996).

A eficiência de um método de pontos interiores (MPI) depende da exploração da estrutura esparsa do problema em questão. Para tanto, existem duas abordagens de dedução de um MPI.

A primeira abordagem consiste nas seguintes etapas:

1. Construir o lagrangeano associado ao problema.
2. Obter as condições de otimalidade.
3. Aplicar o método de Newton às condições de otimalidade.
4. Resolver o sistema linear resultante.

Os MPIs assim deduzidos serão denominados MPIs Tradicionais. Exemplos de utilização desta abordagem são fornecidos por (Azevedo et al., 2002a; Oliveira et al., 2004; Bhatia and Biegler, 1999; Castro, 2000).

A segunda abordagem é proposta por (Gondzio and Sarkissian, 2003) e consiste em:

1. Construir um método de pontos interiores, sem considerar estrutura esparsa particular, utilizando a primeira abordagem.
2. Identificar as operações algébricas relacionadas aos cálculos dos métodos de pontos interiores.
3. Definir as operações algébricas em função da estrutura esparsa do problema.

Os MPIs assim deduzidos serão denominados denominados de MPIs por blocos. Esta abordagem, porém, depende da utilização de conceitos de programação orientada a objetos. Ou seja, a implementação de métodos de pontos interiores será restrita a linguagens de programação como C++ e Java.

Este trabalho irá estender a abordagem de Gondzio and Sarkissian (2003), permitindo que esta possa ser utilizada em ambientes de programação procedural como Matlab.

Maiores detalhes sobre as abordagens serão fornecidos nas próximas seções.

2.1 MPI Tradicional

Para ilustrar a dedução de um MPI de acordo com primeira abordagem, temos o problema Eq. (1):

$$\begin{cases} Min\ f(x) \\ Ax = b \\ \underline{x} \leq x \leq \overline{x}, \end{cases} \tag{1}$$

que pode ser reformulado para Eq. (2):

$$\begin{cases} Min\ f(x) \\ Ax = b \\ x + s = \overline{x} - \underline{x} \\ x, s \geq 0 \end{cases} \tag{2}$$

A função Lagrangeana associada ao problema de otimização, com função objetivo não-linear e restrições lineares é dada por:

$$\mathcal{L} = f(x) - y^t h(x) - \tilde{w}^t(x + s - \overline{x}) - z^t(x - t) - \mu \left(\sum_{i=1}^{p} \log s_i + \sum_{i=1}^{p} \log t_i \right). \tag{3}$$

As condições de otimalidade de primeira ordem são dadas por:

$$\begin{array}{lcr} \nabla f(x) - A^t y + w - z & = & 0 \\ Ax - b & = & 0 \\ x + s - \overline{x} & = & 0 \\ SW & = & \mu e \\ XZ & = & \mu e, \end{array} \tag{4}$$

onde: S, X, W e Z são matrizes diagonais com elementos estritamente positivos e e é um vetor de uns de dimensão apropriada.

Para simplificar a notação, define-se $H(x) = \nabla^2 f(x)$. Aplicando o método de Newton às condições de otimalidade:

$$\begin{bmatrix} -H(x) & A^t & -I & I & 0 \\ A & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & I \\ 0 & 0 & S & 0 & W \\ Z & 0 & 0 & X & 0 \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dw \\ dz \\ ds \end{bmatrix} = \begin{bmatrix} r_d \\ r_p \\ r_a \\ r_b \\ r_c \end{bmatrix} \tag{5}$$

onde:

$$\begin{cases} r_d = \nabla f(x) - A^t y + w - z \\ r_p = b - Ax \\ r_a = \overline{x} - x - s \\ r_b = \mu e - SWe \\ r_c = \mu e - XZe \end{cases}$$

Reescrevendo o sistema linear:

$$\begin{array}{lcl} A^t dy - dw + dz - H(x)dx & = & r_d \\ Adx & = & r_p \\ dx + ds & = & r_a \\ Sdw + Wds & = & r_b \\ Xdz + Zdx & = & r_c \end{array} \tag{6}$$

A eliminação de variáveis do sistema linear, dado pela Eq. (6), fornecerá o MPI primal dual com variáveis canalizadas e restrições de igualdade:

Tabela 1: MPI primal dual para estrutura esparsa A.

MPI Primal Dual estrutura esparsa A

Dados $(x^0, s^0, w^0, z^0) > 0$, y^0 $livre$ e $\tau \in (0,1)$;

Para $k = 0, 1, ...$

$\mu^k = \sigma\gamma^k/n_p$,

onde: n_p =dimensão do vetor (x, s), γ^k é o GAP e γ^k/n_p é o GAP médio.

$r_d^k = \nabla f(x^k) - A^t y^k + w^k - z^k$

$r_p^k = b - Ax^k$

$r_a^k = \overline{x} - x^k - s^k$

$r_b^k = \mu^k e - S^k W^k e$

$r_c^k = \mu^k e - X^k Z^k e$

$r_k^k = r_d^k + S^{-1}(r_b^k - W r_a^k) - (X^k)^{-1} r_c^k$

$D^k = (H(x^k) + (S^k)^{-1} W^k + (X^k)^{-1} Z^k)$

$dy^k = (A(D^k)^{-1} A^t)^{-1}(r_p^k + A(D^k)^{-1} r_k^k)$

$dx^k = (D^k)^{-1}(A^t dy^k - r_k^k)$

$ds^k = r_a^k - dx^k$

$dw^k = (S^k)^{-1}(r_b^k - W^k ds^k)$

$dz^k = (X^k)^{-1}(r_c^k - Z^k dx^k)$

$e_{pd}^k = Min \left\{ \underset{\partial x_i^k < 0}{Min} \left(\frac{-x_i^k}{\partial x_i^k} \right), \underset{\partial s_i^k < 0}{Min} \left(\frac{-s_i^k}{\partial s_i^k} \right), \underset{\partial w_i^k < 0}{Min} \left(\frac{-w_i^k}{\partial w_i^k} \right), \underset{\partial z_i^k < 0}{Min} \left(\frac{-z_i^k}{\partial z_i^k} \right) \right\}$

$\alpha_{max}^k = Min(1, \tau e_{pd}^k)$

$\alpha_{pd}^k = \underset{\alpha^k \in [0, \alpha_{max}^k]}{Min} M(\alpha^k)$

$y^{k+1} = y^k + \alpha_{pd}^k dy^k$

$x^{k+1} = x^k + \alpha_{pd}^k dx^k$

$s^{k+1} = s^k + \alpha_{pd}^k ds^k$

$w^{k+1} = w^k + \alpha_{pd}^k dw^k$

$z^{k+1} = z^k + \alpha_{pd}^k dz^k$

$k \leftarrow k + 1$

Até Convergir

Cabe destacar que, a inversão de $AD^{-1}A^t$, para encontrar dy, envolve a maior parte do esforço computacional do nosso método. Mas, dado que A e $H(x)$ são $m \times p$, $m < p$, tendo m linhas linearmente independentes, então, $AD^{-1}A^t$ é simétrica, definida positiva e, na prática, é usada a decomposição de Cholesky para resolver o sistema linear (Wright, 1996).

Para problemas em que a matriz A possui uma estrutura esparsa bloco primal dada por:

$$\mathrm{A}= \begin{bmatrix} A_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & \cdots & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n & 0 \\ S_1 & S_2 & \cdots & S_n & A_{n+1} \end{bmatrix} \tag{7}$$

Os passos descritos anteriormente devem ser seguidos para se deduzir um MPI tradicional. Detalhes desta abordagem podem ser vistos em (Azevedo et al., 2002a; Bhatia and Biegler, 1999; Castro, 2000).

2.2 MPI por Blocos

A grande vantagem desta abordagem no desenvolvimento dos MPIs é a facilidade com que se pode explorar as estruturas esparsas da matriz A.

A idéia chave do processo é identificar quais são as operações matriciais relacionadas com os cálculos dos MPIs, como descrito na Fig. 1.

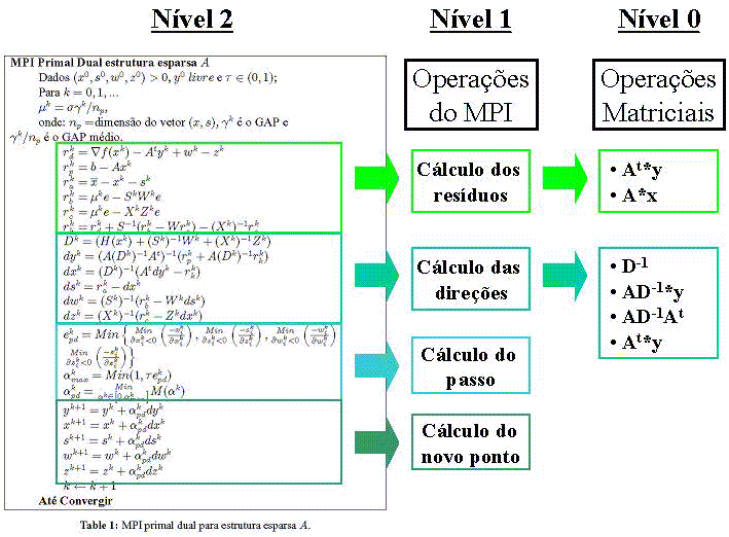

Figura 1: Identificação das operações matriciais de um MPI.

A Figura 1 mostra que para se definir um MPI de acordo com uma estrutura esparsa particular, basta se redefinir o significado das seguintes operações matriciais C_1:

- Dados A e D, calcular $\Phi = AD^{-1}A^t$.
- Calcular os fatores $\Phi = LL^t$.
- Resolver os sistemas lineares $\Phi x = b$, $Ly = b$ e $L^t x = y$.

- Produto de matriz com vetor: Nz, $N^t z$ para A, D e Φ.

Ou seja, para se deduzir um MPI por blocos, cuja matriz A possui a estrutura esparsa dada pela Eq. (7), basta utilizar o MPI da Tabela 1 com operações matriciais C_1 redefinidas de acordo com a Eq. (7).

Supondo definida uma classe *Vector*, que descreve os vetores associados às matrizes, uma possível declaração de uma classe abstrata *Matrix* seria:

```
class Matrix
{
  virtual void ComputeAThetaAt(Matrix theta);
  virtual void Factorize();
  virtual void SolveAThetaAt(Vector x, Vector y);
  virtual void SolveTriang(Vector x, Vector y);
  virtual void SolveTransTriang(Vector x, Vector y);
  virtual void MatrixVectProd(Vector x, Vector y);
  virtual void MatrixTransVectProd(Vector x, Vector y);
}
```

A classe *Matrix* pode ser estendida para classes *concretas* as quais serão responsáveis por fornecer uma implementação para as funções membro.

Ou seja, utilizando o *polimorfismo* de um objeto da classe *Matrix* (é possível representar diferentes objetos que compartilham os mesmos métodos) as operações apropriadas são implementadas de acordo com a estrutura esparsa da matriz.

A Figura 2 ilustra a aplicação deste conceito para matrizes A bloco primal.

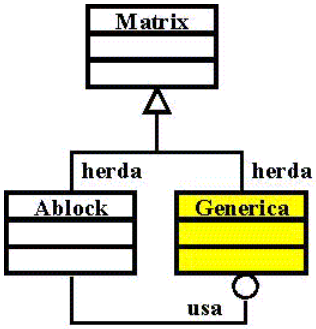

Figura 2: Classes para matriz A bloco primal.

A classe *Ablock* representa a matriz A bloco primal do problema. Como a matriz A é composta por matrizes supostas sem estrutura esparsa definida, denominada genéricas (matrizes A_i e S_i), então, ela utiliza classes que implementam estas matrizes e suas operações (classe *Generica*).

Isto só é possível, pois as classes *Ablock* e *Generica* herdam a mesma interface, da classe *Matrix*, que define as operações C_1.

Com isso uma multiplicação de um objeto matriz A da classe *Ablock* por um vetor x pode ser representada em termos das operações definidas na classe *Generica*.

Dessa forma, parte da declaração da classe *Ablock*, contendo dois vetores de matrizes da classe *Generica*, será:

```
class Ablock : public Matrix
{
  private:
    int  NumBlocks
    Generica[] Ai;
    Generica[] Si;
}
```

A Figura 3 fornece um esquema de cálcular $y = Ax$, matriz A como em Eq. (7), a partir das operações de multiplicação de matriz por vetor definidas na classe *Generica* (ou seja, $y = A_i x$ e $y = S_i x$).

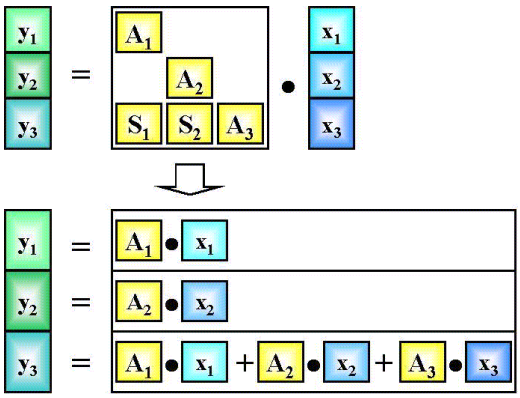

Figura 3: Operação $y = Ax$ para matriz A constituída por blocos de matrizes genéricas.

Como observado na apresentação do MPI da Tabela 1, a operação de maior custo computacional é a resolução do sistema (8):

$$(AD^{-1}A^t)dy = (r_p + AD^{-1}r_k) \quad \Rightarrow \quad \Phi x = b \tag{8}$$

onde: $D = H(x) + S^{-1}W + X^{-1}Z$.

Uma observação importante é que para problemas em que $f(x)$ é linear ou quadrática, $H(x)$ é matriz de zeros ou diagonal, respectivamente. Assim, D será matriz diagonal tal que o cálculo de D^{-1} é muito rápido.

Porém, para problemas em que a função objetivo é não-linear, a estrutura da matriz $H(x)$ poderá ser diferente de uma matriz diagonal.

Para os casos em que a função $f(x)$ é linear ou quadrática, tal que D é matriz diagonal separada por blocos, e a matriz A é bloco primal:

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n & 0 \\ S_1 & S_2 & \cdots & S_n & A_{n+1} \end{bmatrix} \quad D = \begin{bmatrix} D_1 & 0 & \cdots & 0 & 0 \\ 0 & D_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & D_n & 0 \\ 0 & 0 & \cdots & 0 & D_{n+1} \end{bmatrix} \tag{9}$$

Então, a matriz $AD^{-1}A^t$ será dada por:

$$\underbrace{\begin{bmatrix} A_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n & 0 \\ S_1 & S_2 & \cdots & S_n & A_{n+1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} D_1^{-1} & 0 & \cdots & 0 & 0 \\ 0 & D_2^{-1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & D_n^{-1} & 0 \\ 0 & 0 & \cdots & 0 & D_{n+1}^{-1} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} A_1^t & 0 & \cdots & 0 & S_1^t \\ 0 & A_2^t & \cdots & 0 & S_2^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n^t & S_n^t \\ 0 & 0 & \cdots & 0 & A_{n+1}^t \end{bmatrix}}_{A^t} =$$

$$= \begin{bmatrix} A_1 D_1^{-1} A_1^t & 0 & \cdots & 0 & A_1 D_1^{-1} S_1^t \\ 0 & A_2 D_2^{-1} A_2^t & \cdots & 0 & A_2 D_2^{-1} S_2^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n D_n^{-1} A_n^t & A_n D_n^{-1} S_n^t \\ S_1 D_1 A_1^t & S_2 D_2^{-1} A_2^t & \cdots & S_n D_n^{-1} A_n^t & \sum_{i=1}^{n+1} S_i D_i^{-1} S_i^t + A_{n+1} D_{n+1}^{-1} A_{n+1}^t \end{bmatrix} =$$

$$= \begin{bmatrix} \Phi_1 & 0 & \cdots & 0 & B_1^t \\ 0 & \Phi_2 & \cdots & 0 & B_2^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \Phi_n & B_n^t \\ B_1 & B_2 & \cdots & B_n & \Phi_{n+1} \end{bmatrix} = \Phi$$

onde: $B_i = S_i D_i^{-1} A_i^t$ e $\Phi_i = A_i D_i^{-1} A_i^t$, para $i = 1, \cdots, n$ e $\Phi_{n+1} = \sum_{i=1}^{n+1} S_i D_i^{-1} S_i^t + A_{n+1} D_{n+1}^{-1} A_{n+1}^t$.

A fatoração da matriz Φ é dada por:

$$\Phi = L \cdot L^t = \begin{bmatrix} L_1 & 0 & \cdots & 0 & 0 \\ 0 & L_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & L_n & 0 \\ L_{n,1} & L_{n,2} & \cdots & L_{n,n} & L_{n+1} \end{bmatrix} \begin{bmatrix} L_1^t & 0 & \cdots & 0 & L_{n,1}^t \\ 0 & L_2^t & \cdots & 0 & L_{n,2}^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & L_n^t & L_{n,n}^t \\ 0 & 0 & \cdots & 0 & L_{n+1}^t \end{bmatrix}$$

assim:

$$\begin{cases} L_i L_i^t = \Phi_i \\ L_{n,i} \cdot L_i^t = B_i \\ \sum_{i=1}^{n} L_{n,i} \cdot L_{n,i}^t + L_{n+1} L_{n+1}^t = \Phi_{n+1}. \end{cases} \tag{10}$$

Utilizando o sistema (10) para resolver o sistema $\Phi x = b$:

$$\begin{cases} z_i = L_i^{-1} b_i, \, i = 1, \cdots, n \\ z_{n+1} = L_{n+1}^{-1} (b_{n+1} - \sum_{i=1}^{n} B_i L_i^{-t} z_i) \\ x_{n+1} = L_{n+1}^{-t} z_{n+1} \\ x_i = L_i^{-t} (z_i - L_i^{-1} B_i^t x_{n+1}), \, i = 1, \cdots, n \end{cases} \tag{11}$$

A implementação do MPI apresentado na Tabela 1 considerando matrizes A e D, como descritas na Eq. (9), implica na construção de duas classes *Ablock* e *Dblock*. As operações matriciais dessas matrizes são como descritas anteriormente.

A Figura 4 apresenta como será a troca de mensagens entre as classes *Ablock* e *Dblock*.

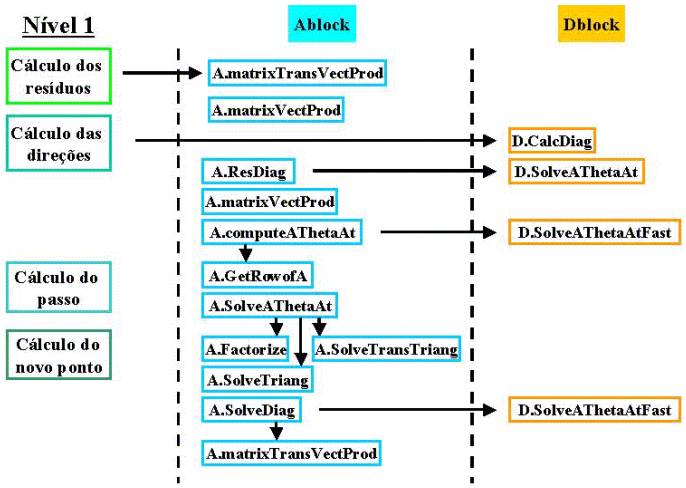

Figura 4: Mensagens entre as classes *Ablock* (matriz A) e *Dblock* (matriz D).

É importante observar que a Fig. 4 é um detalhamento, em termos das funções membro definidas pela interface da classe abstrata *matrix*, das operações do nível 1 da Fig. 1.

Além disso, as operações matriciais como $Ax = y$, representada por *A.matrixVectProd*, são definidas como descrito na Fig. 3. Isto só é possível com a utilização do polimorfismo, tal que o funcionamento das operações matriciais dependem do tipo de estrutura esparsa que é declarada para as matrizes A e D.

2.3 MPI por blocos modificado

Para deduzir um MPI, utilizando os conceitos ilustrados na seção anterior, é necessário observar que as mensagens entre as classes podem ser substituídas por funções. Estas funções, porém, estão associadas a um determinado tipo de estrutura esparsa.

O código da multiplicação de uma matriz A, com estrutura esparsa dada em Eq. (7), por um vetor x, cujo resultado é um vetor y, ou seja $y = Ax$, é dado, em Matlab (programando de maneira procedural), por:

```
function [y] = matrixVectProdAblock(Ai,Si,x,n)

   % Inicializacao de variaveis.
   y = [];
   fxi = 0;
   ys = 0;

   % Loop de construcao de yi, i = 1,...,n.
   for i = 1:n
     % Indices de particao de x.
     ixi = fxi + 1;
     fxi = fxi + size(Ai{i},2);

     % Particionando x.
```

```
  xi = x(ixi:fxi);
  % Calculo dos yi.
  [yi] = matrixVectProdDensa(Ai{i},xi);

  % Ultimo bloco yi.
  ys = ys + matrixVectProdDensa(Si{i},xi);

  % Armazenando em y os blocos yi.
  y = [y; yi];
end


% Novo valor de xi.
ixi = fxi + 1;
fxi = fxi + size(Ai{n+1},2);
xi = x(ixi:fxi);

% Obtendo o ultimo termo do bloco ys.
ys = ys + matrixVectProdDensa(Ai{n+1},xi);

% Armazenando o ultimo bloco na resposta.
y = [y; ys];
```

Ou seja, no lugar de se deduzir o MPI para a estrutura esparsa bloco angular por meio da orientação a objetos como descrito na Fig. 2, é promovida a reutilização de funções como descrito no código anterior.

Assim, a função *matrixVectProdAblock*, associada a $y = Ax$ da estrutura esparsa bloco angular, é elaborada utilizando a função *matricVectProdDensa*, que representa $y_i = A_i x_i$ para matrizes A_i densas.

Por meio desta abordagem, o MPI que era construído pela orientação a objetos como na Fig. 4, será codificado como indicado na Fig. 5.

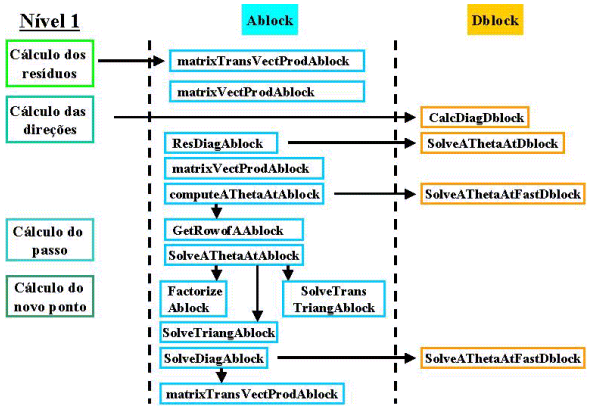

Figura 5: Funções a serem utilizadas para cada tipo de matriz (*Ablock* = A e *Dblock* = D).

Vale observar que a Fig. 5 mostra uma desvinculação entre as funções membro e suas respectivas classes. Portanto, a diferenciação entre um multiplicação $y = Ax$ para uma estrutura matricial bloco angular ou densa não é feita mais pelo polimorfismo, e sim pelo nome das funções.

Assim, é possível aliar o reaproveitamento de código já elaborado para outras estruturas esparsas do enfoque de Gondzio and Sarkissian (2003) com a capacidade de rápida prototipação de funções do Matlab.

3. Aplicações do MPI por blocos modificado

Para ilustrar a aplicação dos conceitos anteriores, nesta seção será mostrada a construção de MPI por blocos para dois tipos de problemas:

- O problema de planejamento e controle da produção (PCP) da manufatura.
- O problema de planejamento da operação de longo prazo (POLP) da geração hidrelétrica.

O objetivo é mostrar como aplicar a abordagem de dedução do MPI por blocos modificado de acordo com a estrutura esparsa do problema apresentado.

3.1 Aplicações em problemas de manufatura

O planejamento e controle da produção (PCP) é a forma pela qual as empresas gerenciam a transformação de matérias-primas em produtos. A função do PCP é de coordenar todas as atividades, desde a aquisição dos componentes necessários até a entrega de produtos acabados. O PCP pode ser dividido em 3 níveis de planejamento distintos: Planejamento estratégico, Planejamento tático e Planejamento Operacional e é especialmente importante para lidar com limitações como custos, capacidade, tempo e qualidade.

Com o advento dos avanços tecnológicos, tem sido dada uma maior ênfase a utilização de sistemas computacionais na manufatura. Um importante grupo dos sistemas computacionais são os sistemas de manufatura flexível (FMS). O FMS é uma coleção de máquinas, geralmente controladas numericamente, ligadas por um sistema automático de manipulação de materiais e dirigido por um computador central.

O FMS possui capacidade de se adaptar rapidamente a alterações nos produtos, resultando em *setups* desprezíveis e que devem ser incluídos nos tempos de processamento (Carvalho et al., 1999).

Considerando que existe uma tendência de adoção do FMS nas manufaturas e que no FMS os *setups* são desprezíveis, o modelo matemático proposto por Carvalho et al. (1999) pode ser adotado.

Assim, o problema de planejamento e controle da produção da manufatura pode utilizar uma modelagem de fluxos em redes com restrições de capacidade mútua que resultará em um problema de programação linear dado pela Eq. (12).

$$
\begin{aligned}
& Min \sum_{i=1}^{n+1} c_i x_i \\
& \begin{bmatrix} A_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_n & 0 \\ S_1 & S_2 & \cdots & S_n & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ d \end{bmatrix} \\
& l_i \leq x_i \leq u_i, i = 1, \cdots, n+1
\end{aligned}
\tag{12}
$$

onde: A_i representa a matriz associada a linha de montagem do produto i, b é um vetor cujos componentes b_i representam a disponibilidade de matéria-prima e a demanda para cada produto i, x_i representa o valor dos fluxos do produto i nos diversos estágios da linha de montagem, l_i representa a utilização mímima da máquina i, u_i expressa a capacidade máxima de produção da máquina i, S_i são as matrizes que representam as restrições de capacidade mútua

associadas aos fluxos x_i dos produtos i na linha de montagem, e d é um vetor cujos componentes representam que a capacidade máxima de processamento simultâneo de produtos uma máquina j não deve ultrapassar seu limite d_j, e a função objetivo é relacionada aos custos de transporte ou processamento dos produtos.

Esta representação utiliza a idéia intituitiva que de os componentes "fluem"pela manufatura, sendo processados por máquinas até se tornarem produtos finais. Maiores detalhes sobre a construção do modelo em (Azevedo et al., 2002a,b).

Devido às dimensões que o problema de PL, associado ao planejamento da produção, pode assumir, em aplicações práticas, torna-se necessário o desenvolvimento de um método de resolução que explore a estrutura do problema (Carvalho et al., 1999; Yamakami et al., 2000).

Para se deduzir um MPI por blocos para este problema, basta verificar que A_{n+1}, da matriz A dada por Eq. (9), é igual a matriz identidade I. Uma discussão detalhada entre a dedução do MPI tradicional e o MPI por blocos, para este problema, é fornecida em (Azevedo et al., 2004).

3.2 Aplicações em geração hidrelétrica

Na atual sociedade, a energia elétrica tem um amplo e importante papel. Desde a utilização para fins domésticos, até na indústria, não é possível conceber a falta deste importante insumo em qualquer atividade. Portanto, é imprescíndivel o planejamento otimizado deste insumo, considerando a etapas de geração, transmissão e distribuição. Como tal planejamento é tarefa complexa, a geração e a transmissão são consideradas por meio de uma Cadeia de Planejamento. A Cadeia de Planejamento divide o problema em dois horizontes de tempo:

- O problema da operação de longo prazo(POLP) considera aspectos hidraúlicos (geração) e um horizonte anual.
- O problema da programação de curto prazo (PPCP) considera aspectos elétricos (transmissão) e horizonte de uma semana.

Para resolver o POLP, uma modelagem de fluxos em redes com restrições lineares e função objetivo não-linear pode ser utilizada Carvalho and Soares (1987); Oliveira and Soares (1995); Cicogna (1999).

Para tanto, foi utilizado o esquema descrito na Fig. 6.

Assim, é obtido um problema de fluxos em redes não-linear com arcos capacitados, cuja formulação matricial simplificada é dada por Eq. (13):

$$\begin{array}{ll} Min & f(\tilde{x}) \\ S.a.: & A\tilde{x} = b \\ & l \leq \tilde{x} \leq u. \end{array} \tag{13}$$

Onde: $\tilde{x} = \left[\begin{array}{cc} x & u \end{array} \right]$ e $A = \left[\begin{array}{c|c} \tilde{A} & S \end{array} \right]$, sendo u o vetor associado as defluências das usinas e x o vetor associado aos volumes das usinas.

Para a rede da Figura 6, as matrizes $\tilde{A}$ e S serão dadas por:

$$\tilde{A} = \left[\begin{array}{ccc} I & 0 & 0 \\ -I & I & 0 \\ 0 & -I & I \end{array} \right] \text{ e } S = \left[\begin{array}{ccc} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{array} \right]. \tag{14}$$

Onde: I é matriz identidade, M é matriz de incidência nó-arco para as variáveis de defluência u.

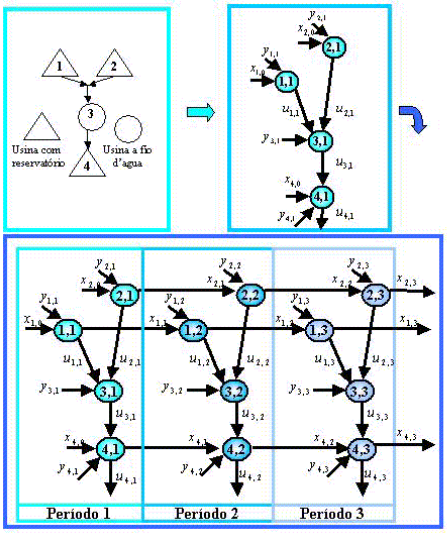

Figura 6: Grafo para 4 usinas e 3 períodos.

Para deduzir um MPI por blocos adaptado para este problema é necessário redefinir as operações matriciais C_1. Destacamos redefinição da operação dada por Eq. (8).

Inicialmente, é importante observar que a matriz Hessiana $H(x)$ do POLP possui a seguinte estrutura:

$$H(x) = \begin{bmatrix} H_1 & H_2 \\ H_2 & H_3 \end{bmatrix}. \tag{15}$$

Onde: H_1, H_2 e H_3 são matrizes diagonais.

A matriz D tem a mesma estrutura da matriz H, de maneira que D^{-1} pode ser rapidamente obtida e a matriz $AD^{-1}A^t$ terá uma estrutura tridiagonal.

Como exemplo, a matriz $AD^{-1}A^t$, para a rede da Fig. 6, será dada por:

$$\Phi = AD^{-1}A^t = \begin{bmatrix} \Phi_1 & B_1^t & 0 \\ B_1 & \Phi_2 & B_2^t \\ 0 & B_2 & \Phi_3 \end{bmatrix} \tag{16}$$

Onde: $B_i = (D_{1(i)}^{-1} + D_{2(i)}^{-1}M^t)$, $\Phi_1 = D_{1(1)}^{-1} + D_{2(1)}^{-1}M^t + M(D_{2(1)}^{-1} + D_{3(1)}^{-1}Mt)$ e $\Phi_i = D_{1(i-1)}^{-1} + D_{1(i)}^{-1} + D_{2(i)}^{-1}M^t + M(D_{2(i)}^{-1} + D_{3(i)}^{-1}Mt)$, $\forall i > 1$, tal que $D_{1(i)}^{-1}$, $D_{2(i)}^{-1}$ e $D_{3(i)}^{-1}$ são submatrizes de D_1^{-1}, D_2^{-1} e D_3^{-1} que por sua vez são submatrizes de D^{-1}.

A fatoração da matriz Φ é dada por:

$$\Phi = LL^t = \begin{bmatrix} L_1 & 0 & 0 \\ L_{n,1} & L_2 & 0 \\ 0 & L_{n,2} & L_3 \end{bmatrix} \begin{bmatrix} L_1^t & L_{n,1}^t & 0 \\ 0 & L_2^t & L_{n,2}^t \\ 0 & 0 & L_3^t \end{bmatrix}, \tag{17}$$

assim:

$$\begin{cases} L_1 L_1^t = \Phi_1 \\ L_{n,i} L_i = B_i, \forall i = 1, 2 \\ L_{n,i-1} L_{n,i-1}^t + L_i L_i^t = \Phi_i. \end{cases} \tag{18}$$

Utilizando a Eq. (17) para resolver o sistema (8):

$$\begin{cases} z_1 = L_1^{-1} b_1 \\ z_i = L_i^{-1}(b_i - B_{i-1} L_{i-1}^{-t} z_{i-1}), i = 2, \cdots, 3 \\ x_3 = L_3^{-t} z_3 \\ x_i = L_i^{-t}(z_i - L_i^{-1} B_i^t x_{i+1}), i = 1, \cdots, 2 \end{cases} \tag{19}$$

4. Conclusões e Trabalhos Futuros

Na tomada de decisão, relacionados a problemas reais é comum a obtenção de modelos matemáticos de grande porte. Para a resolução dos mesmos, os métodos de pontos interiores, que exploram a estrutura esparsa do problema, são indicados, na literatura, como uma alternativa rápida e robusta. O maior trabalho, porém, está relacionado à corretude e a tempo de desenvolvimento das deduções matemáticas que fornecem os MPIs. Por meio da abordagem exposta neste trabalho, é possível unir o rápido desenvolvimento de programas em Matlab, com a reutilização de código da programação orientada a objetivo. Isto resultará em uma maior facilidade na identificação de erros de programação, reduzindo o tempo de construção de programas de MPIs. Como trabalhos futuros é interessante aplicar a abordagem proposta para além dos problemas de PCP da manufatura e do POLP de sistemas de potência aqui descritos.

Agradecimentos

Este trabalho contou com o suporte financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

Adler, I., Resende, M. G. C., Veiga, G., & Karmarkar, N., 1989. An implementation of karmarkar's algorithm for linear programming. *Mathematical Programming*, vol. 44, pp. 297–335.

Azevedo, A. T., Carvalho, M. F. H., Oliveira, A. R. L., & Soares, S., 2002a. Métodos de pontos interiores aplicados a problema de multifluxo com restrições adicionais. *Tendências em Matemática Aplicada e Computacional*, vol. 3, n. 1, pp. 41–50.

Azevedo, A. T., Carvalho, M. F. H., Oliveira, A. R. L., & Soares, S., 2002b. Planejamento e controle da produção via fluxos em redes com métodos de pontos interiores. In *Revista do XIV Congresso Brasileiro de Automática*.

Azevedo, A. T., Carvalho, M. F. H., Oliveira, A. R. L., & Soares, S., 2004. Eficiência e robustez de métodos de pontos interiores no planejamento e controle da produção modelado por fluxos em redes. In *Submetido ao XXXVI Simpósio Brasileiro de Pesquisa Operacional*.

Bhatia, T. & Biegler, L., 1999. Multiperiod design and planning with interior point methods. *Computer and Chemical Engineering*, vol. 23, n. 7, pp. 919–932.

Carvalho, M. & Soares, S., 1987. An efficient hydrotermal scheduling algorithm. *IEEE Transactions on Power Systems*, vol. 2, n. 3, pp. 537–542.

Carvalho, M. F., Fernandes, C. A. O., & Ferreira, P. A. V., 1999. Multiproduct multistage production scheduling (mmps) for manufacturing systems. *Production Planning Control*, vol. 10, n. 7, pp. 671–681.

Castro, J., 2000. A specialized interior-point algorithm for multicommodity network flows. *SIAM J. Optimization*, vol. 10, n. 1, pp. 852–877.

Cicogna, M., 1999. *Modelo de Planejamento da Operação Energética de Sistemas Hidrotérmicos a Usinas Individualizadas Orientado por Objetos*. PhD thesis, Tese de Mestrado da FEEC - UNICAMP - Campinas.

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

Gondzio, J. & Sarkissian, R., 2003. Parallel interior point solver for structured linear programs. *Mathematical Programming*, vol. 96, n. 3, pp. 561–584.

Gonzaga, C. C., 1992. Path following methods for linear programming. *SIAM Review*, vol. 34, n. 2, pp. 167–227.

Karmarkar, N., 1984. A new polinomial-time algorithm for linear programming. *Combinatorica*, vol. 4, n. 4, pp. 373–395.

Mehrotra, S., 1994. On implementation of a primal-dual interior point methods. *SIAM Journal on Optimization*, vol. 2, n. 4, pp. 575–601.

Oliveira, A. & Lyra, C., 1991. Implementação de um método de pontos interiores para programação linear. *Revista SBA: Controle e Automação*, vol. 3, n. 2, pp. 370–382.

Oliveira, A., Soares, S., & Nepomuceno, L., 2004. Short term hydroeletric scheduling combining network flow and interior point approaches. Aceito para publicação na Eletrical Power & Energy Systems.

Oliveira, G. & Soares, S., 1995. A second-order network flow algorithm for hydrotermal scheduling. *IEEE Transactions on Power Systems*, vol. 10, n. 3, pp. 1635–1641.

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

Wright, S. J., 1996. *Primal-Dual Interior-Point Methods*. SIAM Publications.

Yamakami, A., Takahashi, M. T., & Carvalho, M. F., 2000. Comparison of some algorithms for manufacturing production planning. *IFAC-MIM 2000 Symposium on Manufacturing, Management and Control*, vol. , pp. 280–284.