UM SISTEMA COMPUTACIONAL PARA SOLUÇÕES DE PROBLEMAS DE PROGRAMAÇÃO LINEAR INTEIRA: MÉTODO DE BRANCH AND BOUND.

Antônio Luiz Pagani

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Nelson Maculan Filho
(Presidente)

Sueli Mendes

Jair Koiller

Rio de Janeiro, RJ - Brasil
Julho de 1979

PAGANI, ANTÔNIO LUIZ

Um Sistema Computacional para Soluções de Problemas de Programação Linear Inteira: Método de BRANCH AND BOUND [Rio de Janeiro] 1979.

V, 120p. 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1979)

Tese - Univ. Fed. Rio de Janeiro. Centro de Tecnologia.

1.Programação Linear Inteira I.COPPE/UFRJ II.Título(série).

À Angela

AGRADECIMENTOS

Ao Professor Nelson Maculan Filho pelo incentivo e orientação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao Banco de Desenvolvimento do Estado do Espírito Santo (BANDES) pelo apoio financeiro e confiança depositada.

À Fundação Instituto Brasileiro de Geografia e Estatística (IBGE).

RESUMO

O trabalho apresenta um sistema computacional para soluções de problemas de programação linear inteira, usando o método de BRANCH AND BOUND.

A estrutura de armazenamento e a modularização utilizados na implementação possibilitam ao usuário a alteração, de modo fácil, dos critérios de corte adotados, permitindo-lhe assim aproveitar-se do conhecimento acerca do problema a resolver, ou mesmo o estudo de novos critérios de corte.

ABSTRACT

This work describes a computational system for problem solving of integer linear programming using the BRANCH AND BOUND method.

The cutting policy can be easily changed by the user due to the storage structure and to the modularity adopted in the implementation. This approach enables the user to take advantage of his knowledge of the target problem or to study new cutting policies.

Í N D I C E

I. INTRODUÇÃO

Um problema de Programação Linear Inteira pode ser considerado como uma extensão de um problema de Programação Linear onde, além das restrições a que podem estar sujeitas as variáveis, temos também que uma ou mais variáveis (até mesmo todas) devem ser inteiras, isto é, assumir valores inteiros.

No Método de "Branch and Bound" esta idéia de extensão é utilizada:

- o problema é resolvido como se fosse de Programação Linear simples (sem levar em consideração que alguma variável deve ser inteira);

- se na solução encontrada as variáveis que devem ser inteiras não o forem, escolhemos uma destas variáveis;

- substituimos o problema que originou esta solução por dois outros, iguais ao problema anterior a menos de uma nova restrição introduzida em cada, restrição esta que nos levará mais próximos de uma solução inteira, se existir, com relação à variável escolhida. A esta operação damos o nome de "corte";

- para cada um dos novos problemas o processo é repetido, até que se encontre a solução desejada ou se esgotem todas as possibilidades.

Para solução de problemas de Programação Linear Simples, o Método Simplex, e sua variante o Método Simplex Revisado, que é mais adaptado para utilização em computadores, são os mais comumente utilizados. Aqui utilizamos o Método Simplex Revisado.

Do exposto acima vemos que a cada iteração do processo temos que nos decidir com relação a duas questões:

- dentre os diversos problemas pendentes, qual deve ser escolhido para prosseguir?
- escolhido um problema, qual das variáveis deve ser cortada?

Ambas as decisões devem ser tomadas visando minimizar o número de iterações necessárias para a conclusão, evitando também a proliferação de problemas pendentes, que requerem espaço adicional em memória principal ou secundária.

Existem alguns métodos gerais para efetuar os dois tipos de decisão, porém o conhecimento do problema particular que se deseja resolver pode, muitas vezes, ditar soluções mais vantajosas que os métodos genéricos.

Para permitir a adoção de critérios (para escolha do problema e da variável a ser cortada) mais adequados ao problema específico, bem como facilitar o estudo de novos métodos, procuramos aqui criar um sistema computacional modular, de modo que as modificações necessárias para alteração dos critérios de escolha ficam restritas aos respectivos módulos.

No capítulo II é apresentado o Método Simplex Revisado e são definidos alguns dos termos aqui utilizados.

No capítulo III é apresentado o Método de BRANCH AND BOUND.

No capítulo IV é descrito o modelo implementado.

No capítulo V é descrito o modo de utilização do sistema e são exemplificadas as modificações necessárias para alterar o critério de escolha do problema e da variável.

Finalmente, no capítulo VI, são apresentadas algumas conclusões.

II. MÉTODO SIMPLEX REVISADO

Um Problema de Programação Linear pode ser descrito como sendo a otimização (maximização ou minimização) de uma função linear sujeita a restrições lineares de desigualdade ou igualdade.

Como exemplo de Problema de Programação Linear temos:

maximizar $z = 3x_1 + 4x_2 + x_3$

sujeita ao seguinte conjunto de restrições:

$$8x_1 + 3x_2 + 4x_3 \leq 7$$

$$2x_1 + 6x_2 + x_3 \leq 3$$

$$x_1 + 4x_2 + 5x_3 \leq 8$$

$$x_1, x_2, x_3 \geq 0$$

Damos à função que se quer otimizar (no caso z) o nome de função objetivo.

O método Simplex tem como premissa e utiliza-se da particularidade de que todas as variáveis do problema devem ser não-negativas[1], motivo pelo qual as restrições do tipo $x \geq 0$ não serão consideradas no conjunto das restrições.

Para solucionar um problema, como o exposto acima, pelo método Simplex, inicialmente precisamos eliminar as desigualdades que apareçam nas restrições. Para isso, introduzimos em cada restrição uma variável, que chamaremos de variável de folga, com coeficiente unitário positivo (se a desigualdade for do tipo $\leq$) ou negativo (no caso contrário).

(1) se tal não ocorrer, o problema pode ser transformado, por meio de substituição de variáveis, num problema equivalente, nas condições desejadas.

As variáveis de folga também são restritas a valores não negativos.

Deste modo o problema em questão ficaria:

$$\text{maximizar } z = 3x_1 + 4x_2 + x_3$$

$$\begin{aligned} \text{sujeito a:} \quad & 8x_1 + 3x_2 + 4x_3 + x_4 = 7 \\ & 2x_1 + 6x_2 + x_3 + x_5 = 3 \\ & x_1 + 4x_2 + 5x_3 + x_6 = 8 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

É fácil verificar que devido a $x_4 \geq 0$, qualquer conjunto de valores de x_1, x_2, x_3 e x_4 que satisfaça a:

$$8x_1 + 3x_2 + 4x_3 + x_4 = 7$$

satisfará também a:

$$8x_1 + 3x_2 + 4x_3 \leq 7.$$

O mesmo ocorre para as demais restrições.

Se fizermos:

$\underset{\sim}{c}^T = (3\ 4\ 1\ 0\ 0\ 0)$, vetor de coeficientes da função objetivo;

$\underset{\sim}{x}^T = (x_1\ x_2\ x_3\ x_4\ x_5\ x_6)$, vetor de variáveis do problema;

$$A = \begin{bmatrix} 8 & 3 & 4 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 4 & 5 & 0 & 0 & 1 \end{bmatrix}$$, matriz de coefientes das restrições;

$\underset{\sim}{b}^T$ ▫ (7 3 8), vetor de restrições;

$\underset{\sim}{0}^T$ ▫ (0 0 ..... 0 0), vetor de zeros, poderemos, utilizando a notação matricial, descrever o problema acima como:

maximizar $z = \underset{\sim}{c}^T \underset{\sim}{x}$

sujeito a:

$$A\underset{\sim}{x} = \underset{\sim}{b}$$

$$\underset{\sim}{x} \geq \underset{\sim}{0}, \text{ onde } \underset{\sim}{x} \in \mathbb{R}^n$$

$$\underset{\sim}{c} \in \mathbb{R}^n$$

$$\underset{\sim}{b} \in \mathbb{R}^m$$

$$A = (a_{ij})_{mxn}$$

Seja m o número de colunas independentes de A.

Tomemos uma matriz B_{mxm}, formada por m colunas de A, tal que o determinante de B seja diferente de 0[1]. As demais colunas de A formarão a matriz N_{mxn-m}.

De maneira análoga dividiremos o vetor $\underset{\sim}{x}$ em dois outros:

$\underset{\sim}{x}_B$ - formado pelos elementos de $\underset{\sim}{x}$ correspondentes às colunas de B;

$\underset{\sim}{x}_N$ - formado pelos elementos de $\underset{\sim}{x}$ correspondentes às colunas de N.

(1) B é uma base.

As variáveis que compõem $\underset{\sim}{x}_B$ são chamadas variáveis básicas, por estarem na base, e as que compõem $\underset{\sim}{x}_N$ são chamadas variáveis não básicas.

Podemos substituir então $A\,\underset{\sim}{x} = \underset{\sim}{b}$ por

$$B\,\underset{\sim}{x}_B + N\,\underset{\sim}{x}_N = \underset{\sim}{b}$$

Como determinante de B é diferente de 0, isto significa que existe B^{-1}. Podemos então fazer:

$$B^{-1}\,(B\,\underset{\sim}{x}_B + N\,\underset{\sim}{x}_N) = B^{-1}\underset{\sim}{b}$$

$$\therefore\ \underset{\sim}{x}_B + B^{-1}\,N\,\underset{\sim}{x}_N = B^{-1}\,\underset{\sim}{b}$$

$$\therefore\ \underset{\sim}{x}_B = B^{-1}\,\underset{\sim}{b} - B^{-1}\,N\,\underset{\sim}{x}_N \qquad \text{(II.1)}$$

Se fizermos $\underset{\sim}{x}_N = \underset{\sim}{0}$, então obteremos para o sistema de restrições a solução $\underset{\sim}{x}_B = B^{-1}\,\underset{\sim}{b}$, que é dita uma solução básica. Além disso, se $\underset{\sim}{x}_B \geq \underset{\sim}{0}$, então $\underset{\sim}{x}_B$ é dita uma solução básica viável.

Vamos então verificar essa solução, com referência à função objetivo. Para isso dividiremos o vetor $\underset{\sim}{c}$ em $\underset{\sim}{c}_B$ e $\underset{\sim}{c}_N$, da mesma forma como fizemos com o vetor $\underset{\sim}{x}$.

Nossa função objetivo, $z = \underset{\sim}{c}^T\,\underset{\sim}{x}$ (função que queremos maximizar), pode então ser escrita como:

$$z = \underset{\sim}{c}_B^T\,\underset{\sim}{x}_B + \underset{\sim}{c}_N^T\,\underset{\sim}{x}_N$$

Substituindo aqui o valor de $\underset{\sim}{x}_B$ obtido em (II.I), teremos:

$$z = \underset{\sim}{c}_B^T\,(B^{-1}\,\underset{\sim}{b} - B^{-1}\,N\,\underset{\sim}{x}_N) + \underset{\sim}{c}_N^T\,\underset{\sim}{x}_N$$

$$\therefore\ z = \underset{\sim}{c}_B^T\,B^{-1}\,\underset{\sim}{b} - (\underset{\sim}{c}_B^T\,B^{-1}N - \underset{\sim}{c}_N^T)\,\underset{\sim}{x}_N \qquad \text{(II.2)}$$

Considerando $\underset{\sim}{x}_B$ uma solução básica viável, isto é, $\underset{\sim}{x}_B = B^{-1} \underset{\sim}{b} \geq \underset{\sim}{0}$, então $\underset{\sim}{x}_B$ será uma solução ótima se $(\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T) \geq \underset{\sim}{0}^T$, pois queremos maximizar z.

Justificativa 1:

Existe um teorema que diz que num Problema de Programação Linear, se houver um ótimo finito, ao menos uma solução ótima é básica[1].

Como o conjunto de restrições

$X = \{x \mid A \underset{\sim}{x} = \underset{\sim}{b} \text{ e } \underset{\sim}{x} \geq \underset{\sim}{0}\}$ é convexo e a função objetivo é convexa e côncava, então um ótimo local é um ótimo global.

Justificativa 2:

Tomemos $z = \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{b} - (\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T) \underset{\sim}{x}_N$.

$\underset{\sim}{x}_N$ é o conjunto das variáveis não básicas que fizemos igual a $\underset{\sim}{0}$ para esta solução. Portanto $z = \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{b}$.

Se $(\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T) \geq \underset{\sim}{0}^T$, isto significa que para qualquer das variáveis não básicas (pertencente ao conjunto $\underset{\sim}{x}_N$), x_j por exemplo, se tentarmos atribuir-lhe um valor diferente de 0 (maior que 0, para obedecer à restrição $x_j \geq 0$), teremos:

$z = \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{b} - k\, x_j$, onde k é um componente de $(\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T)$ e portanto não-negativo. Decorre daí que o valor de z ficará inalterado ou diminuirá. Portanto z já está num máximo (ótimo).

(1) Ver DANTZIG, G. B. - Linear Programming and Extensions - Princeton University Press - Princeton, New Jersey - 1963.

Temos então a condição para que a solução básica viável seja ótima. Como a solução inicial arbitrada dificilmente será ótima, temos que ir por tentativas, substituindo de cada vez uma variável de $\underset{\sim}{x}_B$ por uma de $\underset{\sim}{x}_N$, até que encontremos o ótimo desejado.

Mostraremos a seguir o método para escolha da variável que entrará na base e da variável que sairá.

Sejam:

I = {variáveis básicas}.
J = {variáveis não básicas}.

Podemos dizer que:

$N = (\underset{\sim}{a}_j) \quad \forall \quad j \in J$

$B = (\underset{\sim}{a}_i) \quad \forall \quad i \in I$

$\underset{\sim}{x}_N = (x_j) \quad \forall \quad j \in J$

$\underset{\sim}{x}_B = (x_i) \quad \forall \quad i \in I$

De (II.2) temos que:

$$z = \underset{\sim}{c}_B^T \; B^{-1} \; \underset{\sim}{b} - (\underset{\sim}{c}_B^T \; B^{-1} \; N - \underset{\sim}{c}_N^T) \; \underset{\sim}{x}_N$$

Quando fazemos $\underset{\sim}{x}_N = \underset{\sim}{0}$, temos:

$z = \underset{\sim}{c}_B^T \; B^{-1} \; \underset{\sim}{b}$, que chamaremos de $\bar{z}$ (valor corrente de z). (II.3)

$\underset{\sim}{c}_B^T \; B^{-1} \; N$ pode ser escrito como:

$$\underset{\sim}{c}_B^T \; B^{-1} \; (\underset{\sim}{a}_{j_1} \; \underset{\sim}{a}_{j_2} \cdots \underset{\sim}{a}_{j_{n-m}}), \; j_1, \; j_2, \ldots j_{n-m} \; \varepsilon J$$

Então:

$$\underset{\sim}{c}_B^T B^{-1} N = (\underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_1} \quad \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_2} \cdots \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_{n-m}})$$

Por sua vez, $\underset{\sim}{c}_N^T$ pode ser escrito como:

$$(c_{j_1} \quad c_{j_2} \cdots\cdots c_{j_{n-m}})$$

levando a:

$$\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T = (\underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_1} - c_{j_1} \quad \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_2} - c_{j_2} \cdots \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_{n-m}} - c_{j_{n-m}})$$

$\underset{\sim}{x}_N^T$ pode ser escrito como:

$$(x_{j_1} \quad x_{j_2} \cdots\cdots x_{j_{n-m}})$$

e o produto $(\underset{\sim}{c}_B^T B^{-1} N - \underset{\sim}{c}_N^T)\ \underset{\sim}{x}_N$ pode ser escrito como:

$$(\underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_1} - c_{j_1} \quad \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_2} - c_{j_2} \cdots \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{a}_{j_{n-m}} - c_{j_{n-m}}) \begin{bmatrix} \underset{\sim}{x}_{j_1} \\ \underset{\sim}{x}_{j_2} \\ \vdots \\ \underset{\sim}{x}_{j_{n-m}} \end{bmatrix}$$

ou $\sum_{j\varepsilon J} \left[(c_{\sim B}^T B^{-1} a_{\sim j} - c_j) x_j \right]$ e teremos então:

$$z = c_{\sim B}^T B^{-1} b - \sum_{j\varepsilon J} \left[(c_{\sim B}^T B^{-1} a_{\sim j} - c_j) x_j \right]$$

Da mesma forma como chamamos $c_{\sim B}^T B^{-1} b$ de $\bar{z}$, chamaremos $c_{\sim B}^T B^{-1} a_{\sim j}$ de z_j e teremos finalmente:

$$z = \bar{z} - \sum_{j\varepsilon J} \left[(z_j - c_j) x_j \right] \qquad \text{(II.4)}$$

Se todos os $z_j - c_j \geq 0$, então $x_{\sim B}$ é ótimo. Caso contrário, para pelo menos um j, digamos j=r, $z_r - c_r < 0$, e x_r deverá ir para a base.

Novamente de (II.1) temos que:

$$x_{\sim B} = B^{-1} b - B^{-1} N x_{\sim N}.$$

Faremos:

$$B^{-1} b = \bar{x}_{\sim B} \qquad \text{(II.5)}$$

$$\text{e } B^{-1} N = Y = (y_{ij})$$

A i-ésima componente de $x_{\sim B}$ pode ser escrita como:

$$x_i = \bar{x}_i - \sum_{j\varepsilon J} \left[y_{ij} x_j \right]$$

$$\therefore \quad x_i = \bar{x}_i - \sum_{j\varepsilon J, j\neq r} \left[y_{ij} x_j - y_{ir} x_r \right]$$

Fazendo $\underset{\sim}{x}_N = \underset{\sim}{0}$ teremos:

$$x_i = \bar{x}_i - y_{ir}\, x_r$$

Para que a solução seja viável, deveremos ter:

$$x_i = \bar{x}_i - y_{ir}\, x_r \geq 0 \quad \forall\, i \in I$$

$$\therefore \quad y_{ir}\, x_r \leq \bar{x}_i$$

$y_{ir} = 0$ ou $y_{ir} < 0$ não nos interessam pois não limitam x_r.

Para $y_{ir} > 0$ temos $x_r \leq \dfrac{\bar{x}_i}{y_{ir}}$

Tomemos:

$\dfrac{\bar{x}_s}{y_{sr}} = \underset{y_{ir} > 0}{\text{mínimo}} \dfrac{\bar{x}_i}{y_{ir}}$; x_s então será a variável que sairá da base.

Mostraremos agora o exposto acima aplicado ao nosso exemplo numérico, cuja proposição repetiremos aqui para recordar:

$$\text{maximizar} \quad z = 3x_1 + 4x_2 + x_3$$

$$\begin{aligned} \text{sujeito a:} \quad & 8x_1 + 3x_2 + 4x_3 \leq 7 \\ & 2x_1 + 6x_2 + x_3 \leq 3 \\ & x_1 + 4x_2 + 5x_3 \leq 8 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Introduzindo as variáveis de folga obtivemos:

$$
\begin{aligned}
z - 3x_1 - 4x_2 - x_3 &= 0 \\
8x_1 + 3x_2 + 4x_3 + x_4 &= 7 \\
2x_1 + 6x_2 + x_3 + x_5 &= 3 \\
x_1 + 4x_2 + 5x_3 + x_6 &= 8
\end{aligned}
$$

Donde:

$$
A = \begin{bmatrix} 8 & 3 & 4 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 4 & 5 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\underset{\sim}{c}^T = (3\ 4\ 1\ 0\ 0\ 0)
$$

$$
\underset{\sim}{b}^T = (7\ 3\ 8)
$$

$$
\underset{\sim}{x}^T = (x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6)
$$

Tomando x_4, x_5 e x_6 como variáveis básicas:

$$
\underset{\sim}{x}_B^T = (x_4 \quad x_5 \quad x_6) \qquad \underset{\sim}{x}_N^T = (x_1 \quad x_2 \quad x_3)
$$

$$
B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad N = \begin{bmatrix} 8 & 3 & 4 \\ 2 & 6 & 1 \\ 1 & 4 & 5 \end{bmatrix}
$$

$$
\underset{\sim}{c}_B^T = (\ 0 \quad 0 \quad 0\ ) \qquad \underset{\sim}{c}_N^T = (\ 3 \quad 4 \quad 1\ )
$$

Determinante de B = 1 e $B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Lembrando que:

$\underset{\sim}{x}_B = B^{-1}\ \underset{\sim}{b}$, para $\underset{\sim}{x}_N = \underset{\sim}{0}$, teremos:

$$\underset{\sim}{x}_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ 8 \end{bmatrix} \geq \underset{\sim}{0}\ ,$$

portanto $\underset{\sim}{x}_B^T = (x_4 \quad x_5 \quad x_6)$ é uma solução básica viável, levando a:

$$z = \underset{\sim}{c}_B^T\ B^{-1}\ \underset{\sim}{b} = (0\ 0\ 0) \begin{bmatrix} 7 \\ 3 \\ 8 \end{bmatrix} = 0.$$

Verificaremos então a condição de otimalidade $z_j - c_j \geq 0\ \forall j \in J$, lembrando que:

$z_j = \underset{\sim}{c}_B^T\ B^{-1}\ \underset{\sim}{a}_j$ e $J = \{1 \quad 2 \quad 3\}$

$$z_1 = (0 \quad 0 \quad 0) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \\ 1 \end{bmatrix} = 0$$

$c_1 = 3$

$\therefore \quad z_1 - c_1 = 0 - 3 = -3$

$$z_2 = (0 \quad 0 \quad 0) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix} = 0$$

$c_2 = 4$

$\therefore \quad z_2 - c_2 = 0 - 4 = -4$

$$z_3 = (0 \quad 0 \quad 0) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix} = 0$$

$c_3 = 1$

$\therefore z_3 - c_3 = 0 - 1 = -1$

Como para pelo menos um j, $z_j - c_j < 0$, concluimos que esta solução não é ótima e escolhemos a variável correspondente ao menor valor de $z_j - c_j$ para entrar na base, no caso x_2.

Lembrando que:

$Y = B^{-1} N$.

$$Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 & 3 & 4 \\ 2 & 6 & 1 \\ 1 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 8 & 3 & 4 \\ 2 & 6 & 1 \\ 1 & 4 & 5 \end{bmatrix}$$

$\underset{\sim}{y}_r^T = (3 \quad 6 \quad 4)$

Calcularemos então $\dfrac{\bar{x}_s}{y_{sr}} = \underset{y_{ir} > 0}{\text{mínimo}} \dfrac{\bar{x}_i}{y_{ir}}$

Para i=4, $\dfrac{\bar{x}_i}{y_{ir}} = \dfrac{7}{3}$;

para i=5, $\dfrac{\bar{x}_i}{y_{ir}} = \dfrac{3}{6}$ e

para i = 6, $\dfrac{\bar{x}_i}{y_{ir}} = \dfrac{8}{4}$, portanto o mínimo corresponde a i=5 e x_5 sairá da base.

Neste ponto deveríamos:

. Substituir em B a coluna correspondente à variável que deixará a base (no caso x_5) pela coluna de A correspondente à variável que entrará na base (no caso x_2);

. Calcular a inversa da nova matriz B;

. Verificar a condição de otimalidade para a nova base. Se não for satisfeita, repetir o processo. Caso contrário, calcular $\underset{\sim}{x}_B = B^{-1} \underset{\sim}{b}$ e $\bar{z} = \underset{\sim}{c}_B^T B^{-1} \underset{\sim}{b}$, usando $\underset{\sim}{x}_B$, B^{-1} e $\underset{\sim}{c}_B^T$ correspondentes a esta base.

Para evitar o processo de inversão matricial a cada iteração, chamando a base anterior de B_0 e a nova base de B_1, calculamos B_1^{-1} a partir de B_0^{-1}.

B_1 difere de B_0 de apenas uma coluna, que chamaremos de $\underset{\sim}{b}_i$ (no nosso caso a segunda coluna, que corresponde a x_2).

Para obter B_1^{-1}:

. Calculamos $B_0^{-1} \underset{\sim}{b}_i$;

. Formamos uma nova matriz, acrescentando à matriz B_o^{-1} o vetor resultante do produto acima;

. Efetuamos o pivotamento desta matriz, sobre o i-ésimo elemento do vetor produto. Ao final do pivotamento, no lugar de B_0^{-1} teremos B_1^{-1}.

Indo além nesta linha de raciocínio, formemos as matrizes:

$$B_i' = \begin{bmatrix} 1 & \tilde{c}_{B_i}^T B_i^{-1} \\ \hline \overset{\sim}{0} & B_i^{-1} \end{bmatrix} \qquad A' = \begin{bmatrix} -\tilde{c}^T \\ \hline A \end{bmatrix}$$

Desta forma o produto $B_i' \tilde{a}_j'$ nos dará como primeiro elemento $\tilde{c}_{B_i}^T B_i^{-1} \tilde{a}_j - c_j$, isto é, $z_j - c_j$, que usamos para verificar a otimalidade. Por outro lado, ao acrescentarmos $B_i' \tilde{a}_j'$ a B_i' e efetuarmos o pivotamento, estaremos substituindo em B_i' os valores de $\tilde{c}_{B_i}^T B_i^{-1}$ pelos correspondentes à nova base.

Se fizermos

$$\overset{\sim}{\bar{x}} = \begin{bmatrix} \bar{z} \\ \hline \overset{\sim}{\bar{x}}_B \end{bmatrix}$$

e incluirmos também $\overset{\sim}{\bar{x}}$ no pivotamento, teremos os valores correspondentes à nova base.

Representando sob a forma de um quadro:

variáveis básicas

		x_{i_1}	x_{i_2}	...	x_{i_m}		
	z	1	$\tilde{c}_{B_i}^T B_i^{-1}$			$\bar{z}$	$z_j - c_j$
variáveis básicas	x_{i_1}	0	B_i^{-1}			$\overset{\sim}{\bar{x}}_{B_i}$	$B_i^{-1} \tilde{a}_j$
	x_{i_2}	0					
	$\vdots$	$\vdots$					
	x_{i_m}	0					

Colocando nosso exemplo numérico na forma da página anterior, teremos:

	z	x_4	x_5	x_6	$\bar{x}$	$B_0' \; a_2'$
z	1	0	0	0	0	-4
x_4	0	1	0	0	7	3
x_5	0	0	1	0	3	6
x_6	0	0	0	1	8	4

Para obter o quadro relativo a nova base, efetuaremos o pivotamento sobre o elemento da linha de x_5, cujo resultado apresentamos a seguir:

	z	x_4	x_2	x_6	$\bar{x}$	
z	1	0	0.667	0	2.0	0
x_4	0	1	-0.500	0	5.5	0
x_2	0	0	0.167	0	0.5	1
x_6	0	0	-0.667	1	6.0	0

Para verificar a otimalidade desta nova solução, basta multiplicar o vetor formado pela primeira linha de B_1' pelos vetores coluna a_j' $\forall$ $j \in J$.

Assim teremos:

$$\text{para } j=1,\ z_1-c_1 = (1 \quad 0 \quad 0.667 \quad 0) \begin{bmatrix} -3 \\ 8 \\ 2 \\ 1 \end{bmatrix} = -1.667$$

$$\text{para } j = 3, \quad z_3 - c_3 = (1 \quad 0 \quad 0.667 \quad 0) \begin{bmatrix} -1 \\ 4 \\ 1 \\ 5 \end{bmatrix} = -0.333;$$

$$\text{para } j = 5, \quad z_5 - c_5 = (1 \quad 0 \quad 0.667 \quad 0) \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 0.667.^{1}$$

Portanto x_1 entrará na base. Calculando o resto do vetor $B_1' \underset{\sim}{a}_1'$ completaremos o quadro para a próxima iteração, conforme mostrado abaixo.

	z	x_4	x_2	x_6	$\bar{x}$	$B_1' \underset{\sim}{a}_1'$
z	1	0	0.667	0	2.000	-1.667
x_4	0	1	-0.500	0	5.500	7.000
x_2	0	0	0.167	0	0.500	0.333
x_6	0	0	-0.667	1	6.000	-0.333

Para determinar qual variável deixará a base, basta dividir cada elemento de $\bar{x}$ pelo correspondente de $B_1' \underset{\sim}{a}_1'$ (exceto o elemento da 1a. linha e lembrando que só interessam os valores de $B_1' \underset{\sim}{a}_1'$ maiores que zero).

(1) Este cálculo não precisava ser feito para j=5, pois x_5 acabou de sair da base na iteração anterior.

Feito isso, concluimos que x_4 deverá deixar a base. Efetuado o pivotamento obteremos:

	z	x_1	x_2	x_6	$\bar{x}$	
z	1	0.238	0.548	0	3.310	0
x_1	0	0.143	-0.071	0	0.786	1
x_2	0	-0.048	0.190	0	0.238	0
x_6	0	0.048	-0.690	1	6.262	0

quando então devemos encontrar todos os $z_j - c_j$ positivos, o que indica que estamos num ótimo com:

$\bar{Z} = 3.310$, as variáveis básicas $\bar{x}_1$, $\bar{x}_2$ e $\bar{x}_6$ respectivamente iguais a 0.786, 0.238 e 6.262 e as variáveis não básicas $\bar{x}_3$, $\bar{x}_4$ e $\bar{x}_5$ iguais a 0.

III - MÉTODO DE BRANCH AND BOUND

Na descrição do método de BRANCH AND BOUND utilizaremos o exemplo numérico introduzido no capítulo anterior. Por questões de apresentação, estamos utilizando um problema pequeno (na prática ocorrem problemas com até algumas centenas de variáveis inteiras) e por esse motivo, algumas das situações que ocorrem na prática não se verificarão no nosso exemplo. Para efetuar uma descrição completa do método sem recorrer a um exemplo maior, a possibilidade de ocorrência dessas outras situações será ressaltada, nos pontos adequados ao longo da apresentação, bem como suas implicações no método.

Para que um problema de programação linear seja considerado também inteiro, basta que para pelo menos uma das variáveis do problema exista a restrição de só poder assumir valores inteiros, isto é, pelo menos uma das variáveis deve ser inteira. Para aumentar a representatividade do nosso exemplo numérico, consideraremos que todas as variáveis do problema são inteiras.

No método de BRANCH AND BOUND, inicialmente resolvemos o problema como se o mesmo fosse de programação linear simples, isto é, sem levar em consideração que alguma variável seja inteira. No nosso exemplo isso foi feito no capítulo anterior, conduzindo a um ótimo com $\bar{z} = 3.310$; $\bar{x}_1=0.786$, $\bar{x}_2=0.238$ e $\bar{x}_6=6.262$, variáveis básicas; $\bar{x}_3=\bar{x}_4=\bar{x}_5=0$, variáveis não básicas.

Prosseguindo, devemos verificar na solução encontrada se todas as variáveis inteiras estão com valores inteiros[1]. Caso isso ocorra, a solução encontrada além de ser ótima é também inteira, isto é, estamos num ótimo inteiro.

(1) Apenas para as variáveis básicas pois as não básicas são iguais a 0.

No nosso caso, temos x_1, x_2 e x_6 variáveis inteiras com valores fracionários. Escolhemos uma dessas variáveis para ser cortada . Como neste ponto pretendemos apenas descrever a operação de corte , adotaremos como critério para a escolha da variável o menor índice, deixando para mais tarde as considerações sobre os critérios usuais. Portanto, efetuaremos o corte sobre a variável x_1.

A operação de corte consiste em substituir o problema que originou a presente solução por dois outros, semelhantes ao original, a menos de uma nova restrição introduzida em cada. Os dois novos problemas são por sua vez resolvidos e se suas soluções não forem inteiras efetuam-se novos cortes e assim por diante.

Se fossemos seguir nesta fase, sem nenhuma alteração, o esquema desenvolvido no capítulo anterior, a introdução desta nova restrição no problema significaria acrescentar uma nova equação no conjunto de restrições, com o correspondente acréscimo de linha nas matrizes A e B e etc. e a solução do novo problema desde o início.

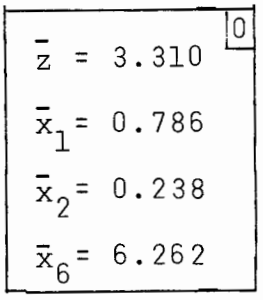
Veremos entretanto que a introdução desta nova restrição pode ser feita sem ser preciso aumentar as matrizes, e que a solução pode ser obtida a partir da solução anterior.

As restrições a serem introduzidas são da forma:

$x_k \leq \lfloor \bar{x}_k \rfloor$ e $x_k \geq \lfloor \bar{x}_k \rfloor + 1$, considerando que o corte está sendo efetuado sobre a variável x_k e que $\lfloor \bar{x}_k \rfloor$ significa o maior inteiro menor ou igual a $\bar{x}_k$.

Pelo desdobramento de um problema em dois outros através do corte, representaremos esta situação por meio de uma árvore, onde cada nó é um problema otimizado, caracterizado pela situação final do quadro do simplex revisado, que por comodidade substituiremos aqui pelo valor da função objetivo e das variáveis básicas.

Assim, o quadro a que chegamos ao final do capítulo anterior daria origem ao primeiro nó (nó 0), que na representação simplificada que adotamos é mostrado abaixo:



(o número do nó está indicado no canto superior direito da figura)

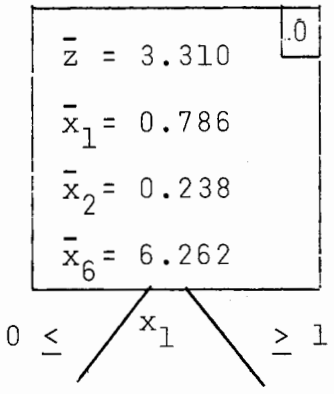
Arbitrariamente, representaremos como nó esquerdo o problema gerado pela introdução da restrição do tipo $x_k \leq \lfloor \bar{x}_k \rfloor$, que será por isso denominada corte à esquerda. Da mesma forma teremos o nó direito, gerado pelo corte à direita ($x_k \geq \lfloor \bar{x}_k \rfloor + 1$).

Um nó como o apresentado acima é dito um nó aberto, pois dele podem ser gerados novos nós. Em oposição, um nó fechado é aquele do qual não serão gerados outros nós. Um nó será considerado fechado quando:

- for efetivamente substituido por seus dois descendentes;
- representar um ótimo inteiro;
- representar um problema sem solução viável;
- ou não interessar a geração de seus descendentes[1].

(1) O motivo pelo qual não interessará a geração dos descendentes, será mostrado adiante.

Os cortes serão indicados conforme mostrado na figura abaixo para o corte da variável x_1 do nó 0:



Embora existam diferenças, as operações de corte à esquerda e à direita são bastante semelhantes. Para ambas as restrições devemos eliminar as desigualdades por meio de variáveis de folga[1]. Assim, $x_k \leq \lfloor \bar{x}_k \rfloor$ e $x_k \geq \lfloor \bar{x}_k \rfloor + 1$ tornam-se respectivamente:

$$x_k + x'_k = \lfloor \bar{x}_k \rfloor \quad \text{e} \quad x_k - x''_k = \lfloor \bar{x}_k \rfloor + 1,$$

x'_k , $x''_k \geq 0$; x'_k e x''_k são as variáveis de folga originadas respectivamente pelo corte à esquerda e pelo corte à direita sobre x_k. Chamaremos x'_k de <u>variável complementar esquerda</u> de x_k e x''_k de <u>variável complementar direita</u> de x_k.

A introdução destas restrições, sem o correspondente aumento das matrizes, é obtida por meio de substituição de variáveis. Explicitando x_k teremos:

$$x_k = \lfloor \bar{x}_k \rfloor - x'_k \qquad \text{(III.1)}$$

$$\text{e} \quad x_k = \lfloor \bar{x}_k \rfloor + 1 + x''_k \qquad \text{(III.2)}$$

(1) Ver capítulo II.

Portanto x_k será substituido no nó esquerdo por $\lfloor \bar{x}_k \rfloor - x'_k$ e no nó direito por $\lfloor \bar{x}_k \rfloor + 1 + x''_k$. Resolvidos os dois problemas, os valores encontrados para x'_k e x''_k serão substituidos em (III.1) e (III.2) para obtermos os valores de x_k que nos interessam. Considerando que:

$x_k, \bar{x}_k, x'_k, x''_k \geq 0$, temos de (III.2) que qualquer valor encontrado para x''_k numa solução básica viável será válido, porém o mesmo não ocorre para x'_k. De (III.1) temos que:

$$x_k = \lfloor \bar{x}_k \rfloor - x'_k \geq 0, \text{ portanto } x'_k \leq \lfloor \bar{x}_k \rfloor \quad \text{(III.3)}$$

Como as restrições introduzidas limitam ainda mais os valores possíveis para x_k, as soluções dos novos problemas (feita a ressalva expressa em (III.3)), serão também soluções do problema original. Por outro lado, as limitações sobre os valores de x_k introduzidas por estas restrições são as mínimas necessárias, visto que se situam nos valores inteiros mais próximos da solução anteriormente encontrada ($\lfloor \bar{x}_k \rfloor$ e $\lfloor \bar{x}_k \rfloor + 1$), excluindo tão somente um intervalo fracionário não válido como solução.

Lembrando da proposição do nosso problema que $A\underset{\sim}{x} = \underset{\sim}{b}$, tomemos a i-ésima linha deste sistema:

$$a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{ik} x_k + \ldots + a_{in} x_n = b_i \quad \text{(III.4)}$$

Substituindo alternadamente (III.1) e (III.2) em (III.4) teremos:

$$a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{ik} (\lfloor \bar{x}_k \rfloor - x'_k) + \ldots + a_{in} x_n = b_i$$

$$a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{ik} (\lfloor \bar{x}_k \rfloor + 1 + x''_k) + \ldots + a_{in} x_n = b_i$$

que podem ser escritas como:

$$a_{il}\, x_1 + a_{i2}\, x_2 + \ldots - a_{ik}\, x'_k + \ldots + a_{in}\, x_n = b_i - a_{ik}(\lfloor \bar{x}_k \rfloor) \qquad \text{(III.5)}$$

$$a_{il}\, x_1 + a_{i2}\, x_2 + \ldots + a_{ik}\, x''_k + \ldots + a_{in}\, x_n = b_i - a_{ik}(\lfloor \bar{x}_k \rfloor + 1) \qquad \text{(III.6)}$$

donde podemos concluir que a única alteração verificada na matriz A será a inversão do sinal da coluna correspondente à variável cortada, no caso do corte à esquerda.

Também da proposição do problema temos que:

$$z = \underset{\sim}{c}^T \underset{\sim}{x}$$

ou seja:

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_k x_k + \ldots + c_n x_n \qquad \text{(III.7)}$$

Substituindo alternadamente (III.1) e (III.2) em (III.7) teremos:

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_k\, (\lfloor \bar{x}_k \rfloor - x'_k) + \ldots + c_n x_n$$

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_k\, (\lfloor x_k \rfloor + 1 + x''_k) + \ldots + c_n x_n,$$

que podem ser escritas como:

$$z - c_k\, (\lfloor \bar{x}_k \rfloor) = c_1 x_1 + c_2 x_2 + \ldots - c_k x'_k + \ldots + c_n x_n \qquad \text{(III.8)}$$

$$z - c_k\, (\lfloor \bar{x}_k \rfloor + 1) = c_1 x_1 + c_2 x_2 + \ldots + c_k x''_k + \ldots + c_n x_n \qquad \text{(III.9)}$$

donde podemos concluir que no vetor $\underset{\sim}{c}$ apenas o elemento correspondente a x_k mudará de sinal no corte à esquerda.

Também a matriz B, o vetor $\underset{\sim}{b}$, etc, sofrerão alterações devido à substituição de variáveis efetuada. Porém como no Método Simplex Revisado não trabalhamos com estes valores e sim com outros, deles derivados (B^{-1}, $\underset{\sim}{x}_B$, etc), que formam o quadro do Simplex, apresentaremos direto no quadro essas alterações.

Conforme mostrado anteriormente, o quadro que estamos usando tem a composição representada abaixo:

Variáveis básicas

	z	x_{i_1}	x_{i_2}	...	x_{i_m}	$\bar{x}$	
z	1	$c_{\sim B_i}^T \; B_i^{-1}$				$\bar{z}$	
Variáveis básicas: x_{i_1}	0						
x_{i_2}	0						
.	.	B_i^{-1}				$\bar{x}_{B_i}$	
.	.						
x_{i_m}	0						

B_i^{-1} é a inversa de B_i que por sua vez é composta de colunas de A. Portanto as alterações em B_i^{-1} serão reflexo das alterações em A. Como o corte à direita não afeta a matriz A, neste caso B_i^{-1} também não será afetada.

No corte à esquerda, a coluna de A que troca de sinal corresponde a uma variável básica, portanto a coluna correspondente em B_i também trocará de sinal.

Seja k o índice da coluna de B_i que trocará de sinal. Chamemos, para facilitar a representação, de B à matriz B_i e de C à sua inversa.

O produto B x C é constante e igual à matriz Identidade (I). Tomemos a expressão de um elemento qualquer deste produto:

$b_{il}\, c_{1j} + b_{i2}\, c_{2j} + \ldots + b_{ik}\, c_{kj} + \ldots + b_{in}\, c_{nj} = i_{ij} = \text{constante}$.

A inversão de sinal da k-ésima coluna de B se traduz na inversão de sinal do elemento b_{ik}. Para que o produto acima permaneça constante, é necessário e suficiente que o elemento c_{kj} também troque de sinal. Portanto a troca de sinal da k-ésima coluna de B_i provoca a troca de sinal da k-ésima linha de sua inversa.

O produto $\underset{\sim}{c}^T_{B_i} B_i^{-1}$ não sofre alteração devido ao corte à direita, pois vimos que neste caso não se alteram nem o vetor $\underset{\sim}{c}$ nem a matriz A (e por conseguinte os vetores $\underset{\sim}{c}_{B_i}$ e a matriz B_i^{-1}). No corte à esquerda, veremos que as alterações ocorridas não afetam o produto em questão.

Novamente consideraremos que:

. o corte foi efetuado sobre a variável x_k;

. x_k é a k-ésima variável da base.

Conforme mostramos acima, sendo o corte à esquerda, teremos a troca de sinal do k-ésimo elemento de $\underset{\sim}{c}_{B_i}$ e da k-ésima linha de B_i^{-1}.

Chamando de c_i a um elemento genérico de $\underset{\sim}{c}_{B_i}$ e de b_{ij} a um elemento genérico de B_i^{-1}, a expressão do i-ésimo elemento do produto $\underset{\sim}{c}^T_{B_i} B_i^{-1}$ que antes do corte era:

$c_1 b_{1i} + c_2 b_{2i} + \ldots + c_k b_{ki} + \ldots + c_m b_{mi}$, após o corte será:

$c_1 b_{1i} + c_2 b_{2i} + \ldots + (-c_k)(-b_{ki}) + \ldots + c_m b_{mi}$ que comprova a afirmativa acima.

De (III.5) e (III.6), respectivamente, vemos que $\underset{\sim}{b}$ transforma-se em $\underset{\sim}{b} - \underset{\sim}{a}_k (\lfloor \bar{x}_k \rfloor)$ devido ao corte à esquerda e em $\underset{\sim}{b} - \underset{\sim}{a}_k (\lfloor \bar{x}_k \rfloor + 1)$ devido ao corte à direita.

De (II.5) temos que $\bar{x}_{B_i} = B_i^{-1} \underset{\sim}{b}$. Já vimos que B_i^{-1} não sofre alteração devido ao corte à direita e portanto neste caso:

$$\bar{x}_{B_i} = B_i^{-1} (\underset{\sim}{b} - \underset{\sim}{a}_k (\lfloor \bar{x}_k \rfloor + 1)), \text{ donde:}$$

$$\bar{x}_{B_i} = B_i^{-1} \underset{\sim}{b} - B_i^{-1} \underset{\sim}{a}_k (\lfloor \bar{x}_k \rfloor + 1).$$

Note-se que $\underset{\sim}{a}_k$ é a coluna de A correspondente a x_k, e que x_k é uma das variáveis básicas nesta solução. Portanto $\underset{\sim}{a}_k$ será também uma coluna de B_i e por isso o produto $B_i^{-1}\,\underset{\sim}{a}_k$ corresponderá à coluna da matriz Identidade correspondente a x_k, isto é, terá todos seus elementos nulos, exceto o k-ésimo que será igual a 1.

Portanto o único elemento de $\bar{x}_{B_i}$ que sofrerá alteração devido ao corte à direita será o correspondente a x_k, que será decrementado de $(\lfloor \bar{x}_k \rfloor + 1)$.

No caso do corte à esquerda, teremos:

$$\bar{x}_{B_i} = B_i^{-1}\,\underset{\sim}{b} - B_i^{-1}\,\underset{\sim}{a}_k\,(\lfloor \bar{x}_k \rfloor) = B_i^{-1}\,\underset{\sim}{b} + B_i^{-1}\,(-\underset{\sim}{a}_k)\,(\lfloor \bar{x}_k \rfloor).$$

A troca de sinal da linha de B_i^{-1} correspondente a x_k, no produto $B_i^{-1}\,\underset{\sim}{b}$ afetará apenas o elemento correspondente a x_k que também trocará de sinal. tQuanto ao termo $B_i^{-1}\,(-\underset{\sim}{a}_k)\,(\lfloor \bar{x}_k \rfloor)$, lembrando que $(-\underset{\sim}{a}_k)$ é que é a coluna de B_i após o corte, podemos afirmar que terá todos os elementos nulos, exceto o correspondente a x_k que será igual a $\lfloor \bar{x}_k \rfloor$. Portanto o único elemento de $\bar{x}_{B_i}$ que sofrerá alteração devido ao corte à esquerda será o correspondente a x_k que trocará de sinal e será acrescido de $\lfloor \bar{x}_k \rfloor$.

De (II.3) temos que $\bar{z} = \underset{\sim}{c}_{B_i}^T\,B_i^{-1}\,\underset{\sim}{b}$

De (III.9) temos que após o corte à direita:

$$z - c_k\,(\lfloor \bar{x}_k \rfloor + 1) = \underset{\sim}{c}^T\,\underset{\sim}{x} \quad \text{donde} \quad z = \underset{\sim}{c}^T\,\underset{\sim}{x} + c_k(\lfloor \bar{x}_k \rfloor + 1)$$

Por outro lado, de (III.6) vimos que $\underset{\sim}{b}$ torna-se $\underset{\sim}{b} - \underset{\sim}{a}_k\,(\lfloor \bar{x}_k \rfloor + 1)$, enquanto $\underset{\sim}{c}_{B_i}^T$ e B_i^{-1} mantem-se inalterados. Portanto:

$$\bar{z} = \underset{\sim}{c}_{B_i}^T\,B_i^{-1}\,(\underset{\sim}{b} - \underset{\sim}{a}_k\,(\lfloor \bar{x}_k \rfloor + 1)) + c_k\,(\lfloor \bar{x}_k \rfloor + 1)$$

donde $\bar{z} = \underset{\sim}{c}_{B_i}^T\,B_i^{-1}\,\underset{\sim}{b} - \underset{\sim}{c}_{B_i}^T\,B_i^{-1}\,\underset{\sim}{a}_k\,(\lfloor \bar{x}_k \rfloor + 1) + c_k\,(\lfloor \bar{x}_k \rfloor + 1)$.

Como vimos anteriormente, o produto $B_i^{-1}\ a_k$ terá todos os elementos nulos, exceto o correspondente à variável x_k, e deste modo o produto $c_{B_i}^T\ B_i^{-1}\ a_k\ (\lfloor \bar{x}_k \rfloor + 1)$ se reduzirá à $c_k\ (\lfloor \bar{x}_k \rfloor + 1)$. Portanto:

$\bar{z} = c_{B_i}^T\ B_i^{-1}\ b - c_k\ (\lfloor \bar{x}_k \rfloor + 1) + c_k\ (\lfloor \bar{x}_k \rfloor + 1)$, donde se pode concluir que $\bar{z}$ não se altera devido ao corte à direita e, por raciocínio semelhante, que o mesmo ocorre para o corte à esquerda.

Resumindo, as alterações devidas aos cortes sobre x_k são:

a) Corte à esquerda:

- Troca de sinal da coluna correspondente a x_k na matriz A;
- Troca de sinal do elemento correspondente a x_k no vetor c;
- Troca de sinal da linha correspondente a x_k na matriz B_i^{-1};
- Substituição de $\bar{x}_k$ por $(-\bar{x}_k + \lfloor \bar{x}_k \rfloor)$.

b) Corte à direita:

- Substituição de $\bar{x}_k$ por $(\bar{x}_k - (\lfloor \bar{x}_k \rfloor + 1))$.

Voltando ao exemplo numérico, tínhamos como nó 0[1]:

0	z	x_1	x_2	x_6	$\bar{x}$
z	1	0.238	0.548	0	3.310
x_1	0	0.143	-0.071	0	0.786
x_2	0	-0.048	0.190	0	0.238
x_6	0	0.048	-0.690	1	6.262

(o número do nó está indicado no canto superior esquerdo da figura)

e decidimos cortar x_1.

(1) Ver final do capítulo II.

Efetuando as alterações definidas acima, teremos:

. para o corte à esquerda:

1	z	x_1'	x_2	x_6	$\bar{x}$
z	1	0.238	0.548	0	3.310
x_1'	0	-0.143	0.071	0	-0.786
x_2	0	-0.048	0.190	0	0.238
x_6	0	0.048	-0.690	1	6.262

. para o corte à direita:

2	z	x_1''	x_2	x_6	$\bar{x}$
z	1	0.238	0.548	0	3.310
x_1''	0	0.143	-0.071	0	-0.214
x_2	0	-0.048	0.190	0	0.238
x_6	0	0.048	-0.690	1	6.262

Estes dois últimos quadros serão os quadros iniciais para o nó esquerdo e direito, respectivamente. Ambos representam soluções básicas, pois $\underset{\sim}{x}_N = \underset{\sim}{0}$, porém não viáveis, pois temos uma variável básica (x_1' e x_1'') com valor negativo em cada.

Temos então que otimizar estes nós, isto é, resolver os problemas que êles representam.

Na otimização dêstes nós, em lugar de decidir qual variável deverá entrar na base e, em função desta, qual sairá, faremos ao contrário: escolhemos qual variável deixará a base e, em função desta, qual entrará.

Para sair da base, escolhemos qualquer uma das variáveis que tornam esta solução não viável, isto é, qualquer variável que esteja com valor negativo. Seja x_k esta variável.

Para definir a variável que entrará na base, tomemos de (II.1) a expressão de $\underset{\sim}{x}_{B_i}$:

$$\underset{\sim}{x}_{B_i} = B_i^{-1}\ \underset{\sim}{b} - B_i^{-1}\ N_i\ \underset{\sim}{x}_{N_i}.$$

Faremos:

$B_i^{-1} \underset{\sim}{b} = \underset{\sim}{\bar{x}}_{B_i}$, $B_i^{-1} N_i = Y = (y_{ij})$ e

J = {índices das variáveis não básicas desta solução}

A k-ésima componente de $\underset{\sim}{x}_{B_i}$ pode ser escrita como:

$$x_k = \bar{x}_k - \sum_{j \in J} (y_{kj}\, x_j)$$

$$\therefore \quad x_k = \bar{x}_k - \sum_{j \in J, j \neq r} (y_{kj}\, x_j) - y_{kr}\, x_r$$

Sairá da base a variável x_k e entrará x_r. Fazendo, para a nova solução, as variáveis não básicas iguais a 0, teremos:

$$0 = \bar{x}_k - y_{kr}\, x_r \quad \therefore \quad x_r = \frac{\bar{x}_k}{y_{kr}}. \qquad \text{(III.10)}$$

Para que esta solução seja viável deveremos ter $x_r \geq 0$, e como $\bar{x}_k < 0$, deveremos então ter $y_{kr} < 0$.

Por outro lado, de (II.4) temos que:

$$z = \bar{z} - \sum_{j \in J} \left[(z_j - c_j)\, x_j \right]$$

$$\therefore \quad z = \bar{z} - \sum_{j \in J, j \neq r} \left[(z_j - c_j)\, x_j \right] - (z_r - c_r) x_r$$

Fazendo $\underset{\sim}{x}_{N_i} = 0$ teremos:

$$z = \bar{z} - (z_r - c_r)\, x_r.$$

Como $(z_r - c_r) \geq 0$ (pois antes do corte estávamos num ótimo) e $x_r \geq 0$ (para que tenhamos uma solução viável), podemos concluir que o valor de z sofrerá um decréscimo. Como estamos maximizando, este decréscimo deverá ser o menor possível. Assim, estamos interessados no:

$$\text{mínimo} \left[(z_r - c_r)\, x_r \right]$$

Substituindo x_r pela sua expressão em (III.10):

$$\text{mínimo} \left[(z_r - c_r) \frac{\bar{x}_k}{y_{kr}} \right], \; y_{ir} < 0.$$

Como $\bar{x}_k$ é constante e negativo, podemos escrever então:

$$\text{mínimo}_{y_{kr} < 0} \left[\frac{z_r - c_r}{-y_{kr}} \right], \; r \in J,$$ que é a condição para que a variável x_r entre na base.

Em termos do quadro do Simplex Revisado, estes critérios se traduzem em:

. sai da base uma das variáveis cujo valor na coluna $\bar{x}$ seja negativo. Se não houver nenhuma variável nesta situação, então estaremos num ótimo;

. calculamos para cada variável não básica o produto da linha de $B_i^!$ correspondente à variável escolhida para deixar a base pela coluna de A' correspondente à variável não básica em questão;

. as variáveis para as quais o produto acima for não negativo serão abandonadas (se todas estiverem nesta situação, o problema não terá solução e o nó será fechado);

. para as demais, calculamos $z_j - c_j$ (produto da primeira linha do quadro pela j-ésima coluna de A'), e o quociente entre este valor e o obtido acima;

. será escolhida para entrar na base a variável que apresentar o menor valor absoluto para o quociente.

Determinada a variável que entra e a variável que sai, o processo segue normalmente, calculando-se a coluna $B_i^! \, a_j^!$, efetuando-se o pivotamento, etc...

Pelo que vimos até agora, no decorrer de todo o processo, trabalhamos apenas com o quadro do Simplex Revisado e

a matriz A'. Repetiremos aqui esta última[1].

$$A' = \begin{bmatrix} -3 & -4 & -1 & 0 & 0 & 0 \\ 8 & 3 & 4 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 4 & 5 & 0 & 0 & 1 \end{bmatrix}$$

No exemplo que estamos apresentando havíamos chegado, para o corte à esquerda, a:

1	z	x_1'	x_2	x_6	$\bar{x}$
z	1	0.238	0.548	0	3.310
x_1'	0	-0.143	0.071	0	-0.786
x_2	0	-0.048	0.190	0	0.238
x_6	0	0.048	-0.690	1	6.262

Prosseguindo de acordo com o esquema desenvolvido, escolhemos x_1' para deixar a base.

Para x_3 teremos:

$$(0 \quad -0.143 \quad 0.071 \quad 0) \begin{bmatrix} -1 \\ 4 \\ 1 \\ 5 \end{bmatrix} = -0.501,$$

que nos interessa por ser negativo.

Prosseguindo:

$$(1 \quad 0.238 \quad 0.548 \quad 0) \begin{bmatrix} -1 \\ 4 \\ 1 \\ 5 \end{bmatrix} = 0.5$$

e o quociente (em valor absoluto) 0.501 / 0.5 = 1.002.

(1) No decorrer do processo a única modificação que ocorre em A' é a troca de sinal das colunas correspondentes às variáveis cortadas à esquerda. Porisso manteremos a matriz A' referente ao problema original, invertendo durante os cálculos, quando for o caso, o sinal das colunas.

Para x_4:

$$(0 \quad -0.143 \quad 0.071 \quad 0) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = -0.143,$$

que tambem serve.

$$(1 \quad 0.238 \quad 0.548 \quad 0) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 0.238$$

$0.238 / 0.143 = 1.664.$

Para x_5:

$$(0 \quad -0.143 \quad 0.071 \quad 0) \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 0.071,$$

que não interessa e portanto será abandonado.

Comparando os quocientes, concluimos que x_3 deverá entrar na base, levando à:

1	z	x_1'	x_2	x_6	$\bar{x}$	$B_i' \underset{\sim}{a}_3'$
z	1	0.238	0.548	0	3.310	0.500
x_1'	0	-0.143	0.071	0	-0.786	-0.501
x_2	0	-0.048	0.190	0	0.238	-0.002
x_6	0	0.048	-0.690	1	6.262	4.502

Efetuando o pivotamento sôbre o elemento na interseção de x_1' com $B_i' \underset{\sim}{a}_3'$, teremos:

1	z	x_3	x_2	x_6	$\bar{x}$	$B_i' \underset{\sim}{a}_3'$
z	1	0.095	0.619	0	2.524	0
x_3	0	0.286	-0.143	0	1.571	1
x_2	0	-0.048	0.190	0	0.238	0
x_6	0	-1.238	-0.048	1	-0.810	0

Prosseguindo, sai x_6 e entra x_4 levando, após o pivotamento à:

1	z	x_3	x_2	x_4	$\bar{x}$
z	1	0	0.615	0.077	2.462
x_3	0	0	-0.154	0.231	1.385
x_2	0	0	0.192	-0.038	0.269
x_4	0	1	0.038	-0.808	0.654

Como todas as variáveis básicas são positivas, estamos novamente num ótimo, que entretanto ainda não é inteiro.

Quanto ao corte à direita havíamos chegado à:

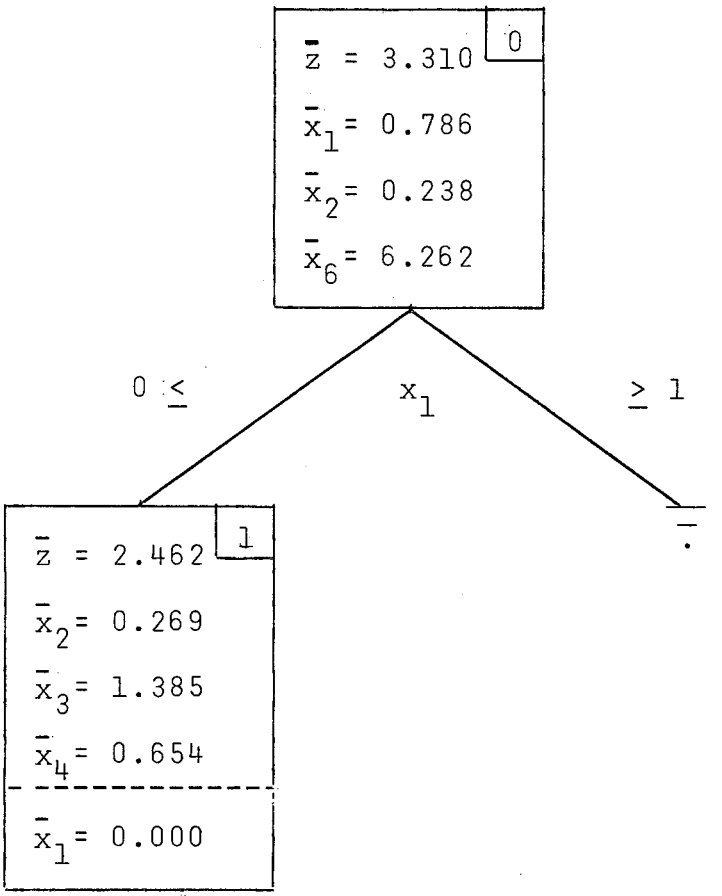
2	z	x_1''	x_2	x_6	$\bar{x}$
z	1	0.238	0.548	0	3.310
x_1''	0	0.143	-0.071	0	-0.214
x_2	0	-0.048	0.190	0	0.238
x_6	0	0.048	-0.690	1	6.262

Como primeiro passo na otimização do problema acima, sairá da base x_1'', sendo substituido por x_5. Após o pivotamento teremos:

2	z	x_5	x_2	x_6	$\bar{x}$
z	1	1.333	0	0	1.667
x_5	0	-2.000	1	0	3.000
x_2	0	0.333	0	0	-0.333
x_6	0	-1.333	0	1	8.333

que é uma solução não viável devido a $\bar{x}_2$. De acordo com o esquema desenvolvido, x_2 deve sair da base. Entretanto, ao efetuarmos o produto da linha de x_2 em $B_i^!$ por $\underset{\sim}{a}_j^!$, $\forall$ x_j não básico, só encontraremos valores positivos, o que indica que este problema não tem solução limitada, sendo porisso abandonado, e o nó correspondente, fechado.

Representando sob forma de árvore os resultados até agora obtidos, teríamos:



Na figura acima temos que:

. o nó 0 é um nó fechado, pois dele já foram gerados os dois problemas possíveis;

. o corte à direita resultou num nó fechado, pois o problema não tinha solução;

. o nó 1 é um nó aberto pois representa um ótimo não inteiro.

No nó 1, a variável x_1 aparece separada pela linha pontilhada para indicar que seu valor foi obtido indiretamente deste quadro, pois ela foi substituida neste problema por x'_1. De (III.1) temos que para o corte à esquerda sôbre x_k, $x_k = \lfloor \bar{x}_k \rfloor - x'_k$. Na presente situação, $k = 1$, $\lfloor \bar{x}_1 \rfloor = 0$ e $x'_1 = 0$ (variável não básica nesta solução), donde resulta que $x_1 = 0$.

Como só temos um nó aberto, somos obrigados à continuar a partir dêle. Porém, frequentemente ocorre têrmos diversos nós abertos, surgindo a necessidade de decidir qual será cortado inicialmente. Abster-nos-emos por enquanto de discutir as políticas usualmente adotadas na solução deste problema, indicando simplesmente que caminharemos na árvore em pré-ordem.

A partir deste ponto omitiremos os resultados intermediários, só nos detendo em detalhes quando para complementar a exposição do método.

Corte de x_2 (nó 1):

. nó esquerdo:

3	z	x_3	x_2'	x_4	$\bar{x}$
z	1	0	0.615	0.077	2.462
x_3	0	0	-0.154	0.231	1.385
x_2'	0	0	-0.192	0.038	-0.269
x_4	0	1	0.038	-0.808	0.654

Sai variável x_2', entra x_5.

3	z	x_3	x_5	x_4	$\bar{x}$
z	1	0	0	0.200	1.600
x_3	0	0	0	0.200	1.600
x_5	0	0	1	-0.200	1.400
x_4	0	1	0	-0.800	0.600

Ótimo não inteiro.

. nó direito:

4	z	x_3	x_2''	x_4	$\bar{x}$
z	1	0	0.615	0.077	2.462
x_3	0	0	-0.154	0.231	1.385
x_2''	0	0	0.192	-0.038	-0.731
x_4	0	1	0.038	-0.808	0.654

Sai variável x_2, entra x_6; sai variável x_3, entra x'_1.

4	z	x'_1	x_6	x_4	$\bar{x}$
z	1	0	1.500	0	-0.500
x'_1	0	0	-0.500	0	1.500
x_6	0	0	-0.500	1	5.500
x_4	0	1	-4.000	0	16.000

Independentemente, e mesmo apesar de, aparentemente, termos alcançado um ótimo,pelo fato de ter entrado na base uma variável complementar esquerda (x'_1), devemos verificar se seu valor atual está dentro do limite válido, conforme estabelecido em (III.3).

No presente caso deveríamos ter $x'_1 \leq 0$. Como tal não ocorre, esta solução é básica viável para o problema em que as variáveis são x'_1, x_2, x_3, x_4, x_5 e x_6, porém não é viável para o problema original, em que as variáveis eram x_1, x_2, x_3, x_4, x_5 e x_6.

Para solucionar este problema, substituimos agora x'_1 pela variável original, x_1. De (III.1) temos que:

$$x_k = \lfloor \bar{x}_k \rfloor - x'_k.$$

Portanto, agora faremos $x'_k = \lfloor \bar{x}_k \rfloor - x_k$, que no caso atual será: $x'_1 = 0 - x_1$. As alterações decorrentes desta operação são idênticas às verificadas por ocasião do corte à esquerda:

. $\underset{\sim}{a}'_1$ troca de sinal, voltando portanto à condição inicial;

. a linha de B'_i correspondente a x'_1, troca de sinal;

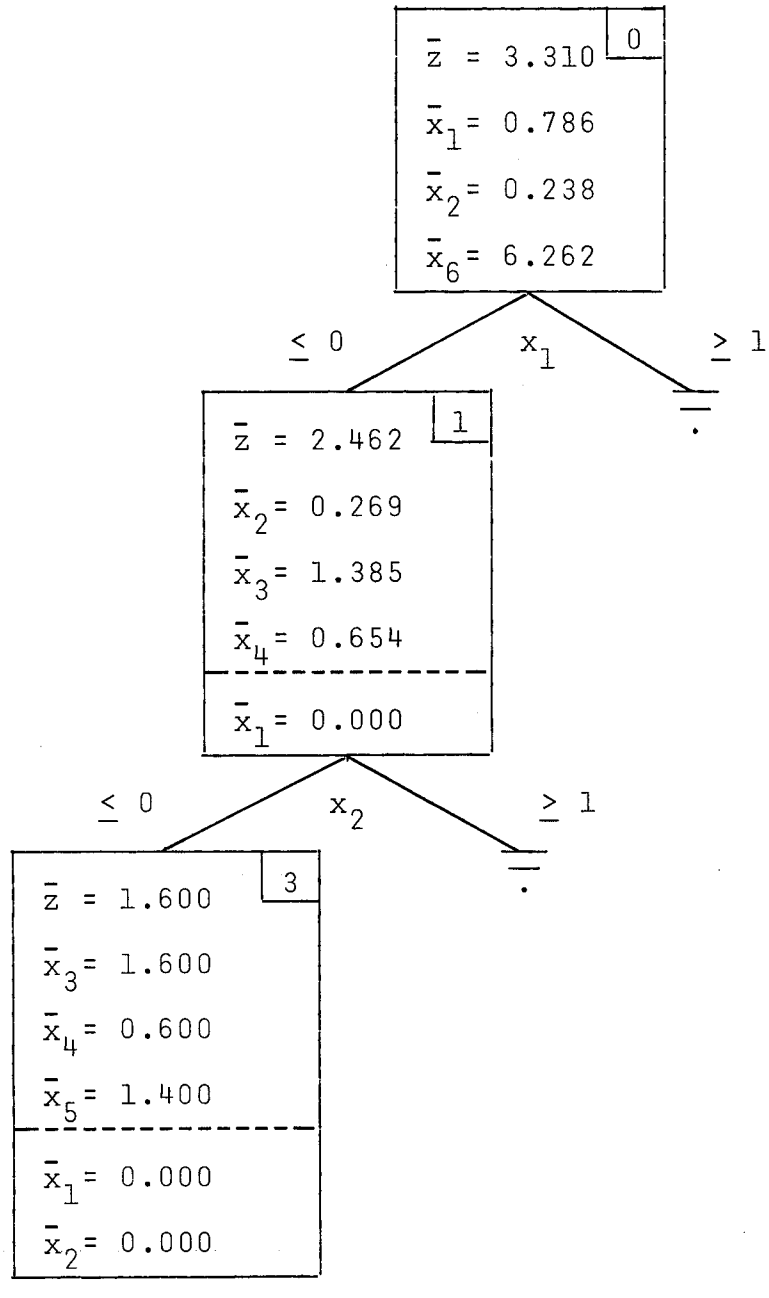
. $\bar{x}'_1$ será substituido por $(-\bar{x}'_1 + \lfloor \bar{x}_1 \rfloor)$.

Efetuando a substituição:

4	z	x_1	x_6	x_4	$\bar{x}$
z	1	0	1.500	0	-0.500
x_1	0	0	0.500	0	-1.500
x_6	0	0	-0.500	1	5.500
x_4	0	1	-4,000	0	16.000

quando concluimos que o problema incidentalmente não tem solução.

Representando na árvore:



Mais uma vez no nosso exemplo temos apenas um nó aberto. Cortando x_3:

. nó esquerdo:

5	z	x_3'	x_5	x_4	$\bar{x}$
z	1	0	0	0.200	1.600
x_3'	0	0	0	-0.200	-0.600
x_5	0	0	1	-0.200	1.400
x_4	0	1	0	-0.800	0.600

Sai variável x_3', entra x_6.

5	z	x_6	x_5	x_4	$\bar{x}$
z	1	0	0	0	1.000
x_6	0	0	0	1	3.000
x_5	0	0	1	0	2.000
x_4	0	1	0	0	3.000

Embora tenhamos encontrado um ótimo inteiro, nada nos garante que esta seja a melhor solução. Devemos portanto prosseguir pois podemos encontrar outras soluções inteiras com valor maior para a função objetivo (estamos maximizando).

. nó direito:

6	z	x_3''	x_5	x_4	$\bar{x}$
z	1	0	0	0.200	1.600
x_3''	0	0	0	0.200	-0.400
x_5	0	0	1	-0.200	1.400
x_4	0	1	0	-0.800	0.600

Sai variável x_3'', entra x_2'.

6	z	x_2'	x_5	x_4	$\bar{x}$
z	1	0	0	1.000	0.000
x_2'	0	0	0	-0.250	0.500
x_5	0	0	1	-1.500	4.000
x_4	0	1	0	-0.750	0.500

Se estivéssemos interessados apenas na melhor solução inteira, êste nó poderia ser abandonado (fechado), pois:

. já temos uma solução inteira com $\bar{z} = 1.000$;

. o valor de $\bar{z}$ neste problema já está em 0.000;

. qualquer solução, inteira ou não, obtida a partir deste nó será menor ou no máximo igual à solução corrente, porque a cada corte a tendência é restringir cada vez mais o valor de z.

Entretanto, muitas vezes estamos interessados em todas as soluções inteiras do problema, e então não abandonaremos nenhum nó nesta situação. Outras vêzes queremos as soluções inteiras cujo valor da função objetivo seja maior que determinado valor (quando maximizando), e então abandonamos os nós que não satisfaçam esta condição.

Consideraremos aqui que nos interessam todas as soluções inteiras.

Prosseguindo, temos x_2' com valor superior ao limite válido e portanto substituimos x_2' por x_2:

6	z	x_2	x_5	x_4	$\bar{x}$
z	1	0	0	1.000	0.000
x_2	0	0	0	0.250	-0.500
x_5	0	0	1	-1.500	4.000
x_4	0	1	0	-0.750	0.500

Sai variável x_2, entra x_1':

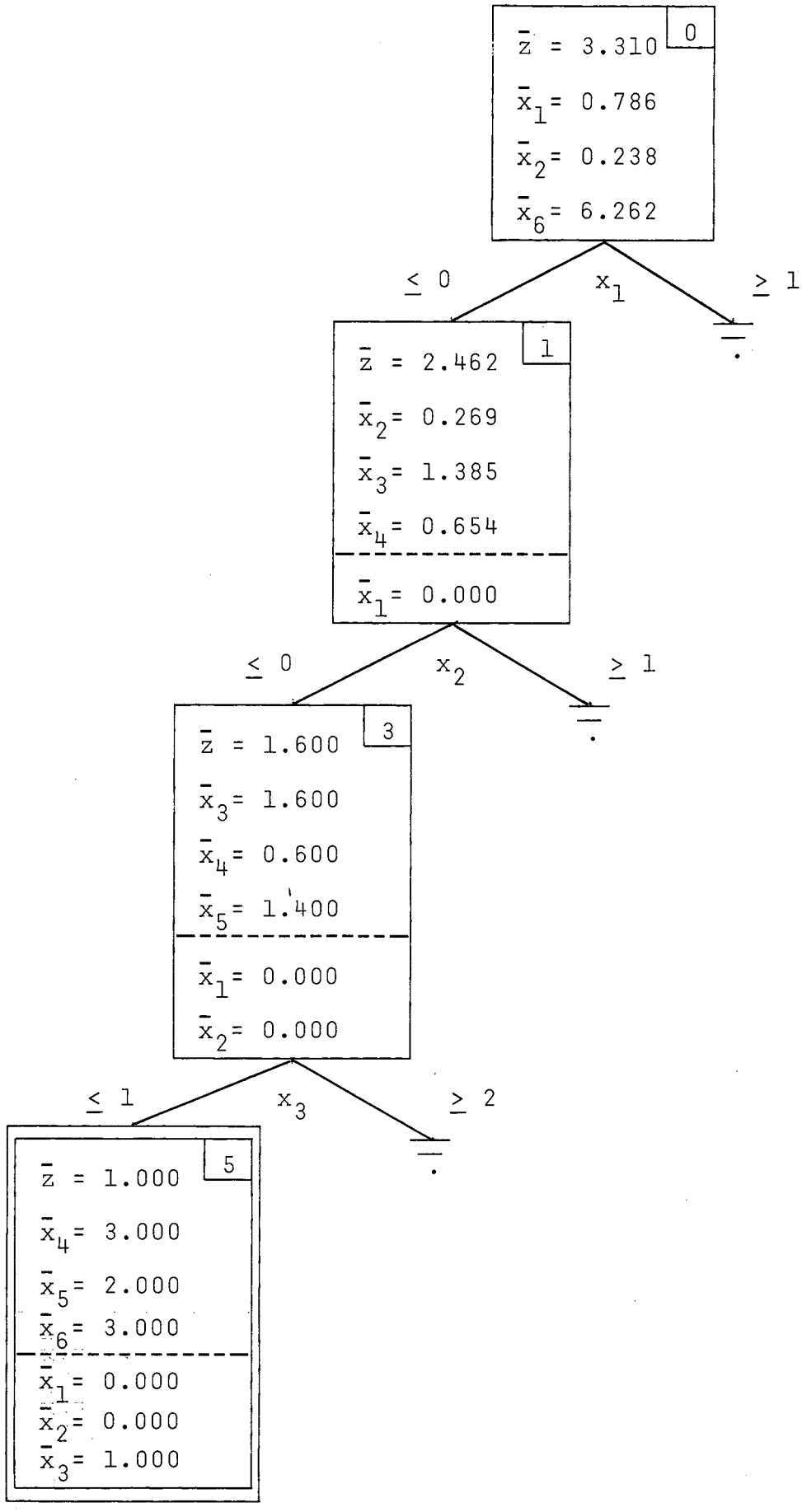
6	z	x_1'	x_5	x_4	$\bar{x}$
z	1	0	0	3.000	-4.000
x_1'	0	0	0	-1.000	2.000
x_5	0	0	1	-2.000	5.000
x_4	0	1	0	-8.000	15.000

x_1' ultrapassou o limite; substituimos x_1' por x_1:

6	z	x_1	x_5	x_4	$\bar{x}$
z	1	0	0	3.000	-4.000
x_1	0	0	0	1.000	-2.000
x_5	0	0	1	-2.000	5.000
x_4	0	1	0	-8.000	15.000

O problema não tem solução.

Na árvore:


0
$\bar{z}$ = 3.310
$\bar{x}_1$ = 0.786
$\bar{x}_2$ = 0.238
$\bar{x}_6$ = 6.262
≤ 0
x_1
≥ 1
1
$\bar{z}$ = 2.462
$\bar{x}_2$ = 0.269
$\bar{x}_3$ = 1.385
$\bar{x}_4$ = 0.654
$\bar{x}_1$ = 0.000
≤ 0
x_2
≥ 1
3
$\bar{z}$ = 1.600
$\bar{x}_3$ = 1.600
$\bar{x}_4$ = 0.600
$\bar{x}_5$ = 1.400
$\bar{x}_1$ = 0.000
$\bar{x}_2$ = 0.000
≤ 1
x_3
≥ 2
5
$\bar{z}$ = 1.000
$\bar{x}_4$ = 3.000
$\bar{x}_5$ = 2.000
$\bar{x}_6$ = 3.000
$\bar{x}_1$ = 0.000
$\bar{x}_2$ = 0.000
$\bar{x}_3$ = 1.000

Por falta de nós abertos o problema é então encerrado, tendo como única solução inteira a do nó 5.

Mais um ponto cabe ser ressaltado no método. Ocasiões há em que, por força dos critérios adotados para escolha do nó e da variável a ser cortada, vimo-nos na situação de ter que cortar uma variável complementar. Ora, como as variáveis complementares não são variáveis do problema original, tendo sido introduzidas como auxiliares no método para evitar o aumento das áreas de dados utilizadas na solução, antes de efetuarmos o corte devemos substituir a variável complementar pela original correspondente.

As alterações decorrentes do retorno de uma variável original ao problema, em substituição a uma variável complementar esquerda, já foram mostradas no decorrer do exemplo (quando uma variável complementar esquerda ultrapassa o limite válido). Para o caso de substituição de variável complementar direita, a única modificação será a substituição do elemento de $\bar{\underset{\sim}{x}}$, digamos $\bar{x}''_k$, por $\bar{x}''_k + \lfloor \bar{x}_k \rfloor + 1$.

O critério que adotamos para escolha do nó a ser cortado tem a vantagem de termos a inversa da base já calculada, pelo menos enquanto estivermos descendo por um mesmo ramo da árvore de nós. Suponhamos um caso extremo em que só tenhamos espaço na memória principal para armazenar uma inversa. Somente após alcançar o fim do ramo é que necessitaremos refazer uma inversa. Porém, se por um lado ganhamos tempo minimizando o número de vêzes que teremos que efetuar inversão de matrizes, por outro lado podemos estar efetuando muitos cortes além do necessário, pois num outro ramo da árvore talvez pudesse ser encontrada a solução em um número menor de iterações[1].

(1) Outros critérios para escolha do nó e da variável a ser cortada são mostrados em Maculan, N. F. - Programação Linear Inteira - capítulo IV - PDD - 17/78 - COPPE/UFRJ.

IV - MODELO IMPLEMENTADO

Na implementação do sistema computacional, procuramos não apenas desenvolver um sistema para solução de problemas de Programação Linear Inteira, mas principalmente um sistema que permitisse ao usuário aproveitar-se do seu conhecimento acerca do problema a resolver a fim de reduzir o custo da obtenção da solução e tambem testar novas políticas para seleção de nó e variável a ser cortada.

Assim, independentemente da modularização ditada pelas regras da boa programação, tomamos especial cuidado em manter em módulos isolados os trechos de programa que realizam a escolha do nó e da variável, de modo a facilitar suas alterações.

Desta forma, o usuário tanto pode usar o sistema para resolver seus problemas de Programação Linear Inteira, aceitando os critérios de corte que foram implementados[1], ou reprogramar o módulo correspondente ao critério que deseja alterar e utilizar o resto do sistema sem precisar fazer alterações..

Outro ponto em que o usuário pode exercer alguma influência sôbre o sistema é com relação às áreas para armazenamento dos dados. Esta influência é exercida por meio de parâmetros, sem necessidade de programação adicional.

ESTRUTURA DE ARMAZENAMENTO

Na descrição do Método de BRANCH AND BOUND, vimos a necessidade de armazenar um número variável de nós durante a solução do problema. Embora só precisemos armazenar os nós abertos, o número destes geralmente alcança valores tais que torna-se impraticável mante-los todos na memória principal. Como solução, adotamos 3 níveis de armazenamento: memória, disco e fita.

(1) Os critérios implementados foram os usados no exemplo do capítulo anterior.

Para armazenamento na memória principal, utilizamos uma lista simplesmente encadeada. Isto porque:

- a estrutura arborescente usada para representar a geração dos novos nós atraves dos cortes pouca utilidade tem para a escolha do nó a ser cortado;
- trabalhamos sempre com as folhas da árvore;
- a simplicidade da estrutura adotada dá ao modelo a flexibilidade necessária para permitir alterações no critério de escolha do nó, com modificações localizadas.

A cada nó (otimizado) é atribuido um valor, que chamaremos de peso do nó, que indica a prioridade para escolha. A cada iteração é cortado o nó de maior peso.

A lista de nós na memória é mantida ordenada por valores decrescentes de peso e cada elemento da lista contém um nó pronto para ser cortado.

No disco, armazenamos uma versão simplificada do nó, contendo o mínimo de informações necessárias para refazelo. Físicamente no disco, os nós não obedecem a nenhuma ordenação. Na memória principal é mantida uma lista simplesmente encadeada, ordenada por peso, com cada elemento associado a uma posição no disco e por conseqüinte ao nó correspondente no disco. A esta lista daremos o nome de mapa do disco.

Na fita armazenamos tambem uma versão simplificada do nó. Não existe nenhuma ordenação: nem diretamente na fita nem na memória principal.

Se considerarmos a memória como o nível mais alto de armazenamento, seguida de disco e finalmente fita, podemos dizer que em cada nível, nenhum nó tem peso superior ao menor peso existente no nível imediatamente superior. Isto implica em que, um nó não pode ser armazenado num determinado nível, mesmo que haja espaço neste nível, se existir num dos níveis abaixo dele algum nó com peso superior ao nó em questão. Isto é, o nó será então armazenado num dos níveis inferiores.

Quando um nó é cortado, dá origem a dois outros que serão então otimizados. Há portanto necessidade de armazená-los, mesmo que temporariamente, na memória principal, até que êles sejam otimizados e tenham sido decididos seus destinos (se serão fechados, armazenados na memória principal, armazenados em disco ou em fita). Para um destes novos nós é aproveitado o espaço do nó pai, que será fechado. Quanto ao outro, se não houver espaço na memória principal, o nó de menor peso na memória será tranferido para disco. Esta ressalva é feita para mostrar que sempre haverá espaço na memória principal para armazenar o nó quando formos decidir aonde armazená-lo. Devido ao exposto no parágrafo anterior, não podemos simplesmente deixar o nó na memória principal.

Considerando Pd e Pf o maior peso presente no disco e na fita, respectivamente, e P o peso do nó N que se quer armazenar, o seguinte algorítmo determina onde N deve ser armazenado:

Algorítmo I:

1. Se P < Pf, então grava N na fita. Fim.
2. Se P < Pd, então percorre o mapa do disco para determinar onde N deve ser inserido;
3. Se houver espaço no disco, grava N no disco na posição determinada acima; atualiza o mapa de disco. Fim.
4. Se não houver espaço no disco e no passo 2 foi determinado que N deve ser inserido no final da lista (mapa do disco), então grava N na fita. Fim.
5. Se não houver espaço no disco e no passo 2 foi determinado que N deve ser inserido no início ou no meio da lista, então procura o último nó da lista (mapa do disco); grava este nó na fita e volta ao passo 3 (agora há espaço no disco).

Dissemos acima que a cada iteração é cortado o nó de maior peso. O algorítmo abaixo localiza êste nó e completa a tarefa de gerenciar as áreas de armazenamento. Neste algorítmo consideramos que existe em disco espaço para N nós.

No algorítmo I acima referimo-nos simplesmente a "fita", dando a idéia de apenas um arquivo em fita. Para esse algorítmo, tudo se passa como se só existisse um arquivo em fita. Já para o próximo algorítmo, veremos que são necessários dois arquivos em fita, porém na prática apenas um de cada vez estará em atividade. Para identificar os dois arquivos, chamaremos de fita ativa o arquivo onde estão sendo gravados os nós (este é o arquivo do algorítmo anterior), e fita inativa o outro.

Algorítmo II:

1. Verifica se a lista de nós na memória está vazia. Em caso afirmativo, continue a partir do passo 2. Caso contrário, siga com o passo 8;
2. Verifica se o disco está vazio (mapa do disco). Em caso afirmativo continue a partir do passo 3. Caso contrário, siga com o passo 7;
3. Verifica se a fita ativa está vazia. Em caso afirmativo não existem mais nós abertos e a solução final do problema terá sido alcançada. Caso contrário, prosseguir com o passo 4;
4. Ler a fita ativa selecionando (guardando a posição relativa do registro) os N/2 nós de maior peso;
5. Copiar a fita ativa para a fita inativa, excluindo os N/2 nós selecionados no ítem anterior, que serão gravados no disco (mantendo atualizado o mapa do disco);
6. Inverter as fitas, isto é, a fita inativa passa a ser considerada fita ativa e vice-versa;
7. Ler do disco o nó de maior peso (de acordo com o mapa do disco) e colocá-lo na fila de nós na memória (o nó deverá ser refeito);
8. O primeiro nó da lista de nós na memória é o nó procurado. Fim.

A estrutura de armazenamento adotada, juntamente com os dois algorítmos descritos têm por objetivo:

. selecionar para o corte sempre o nó de maior peso;

. manter os nós de maior peso o mais facilmente acessíveis e vice-versa.

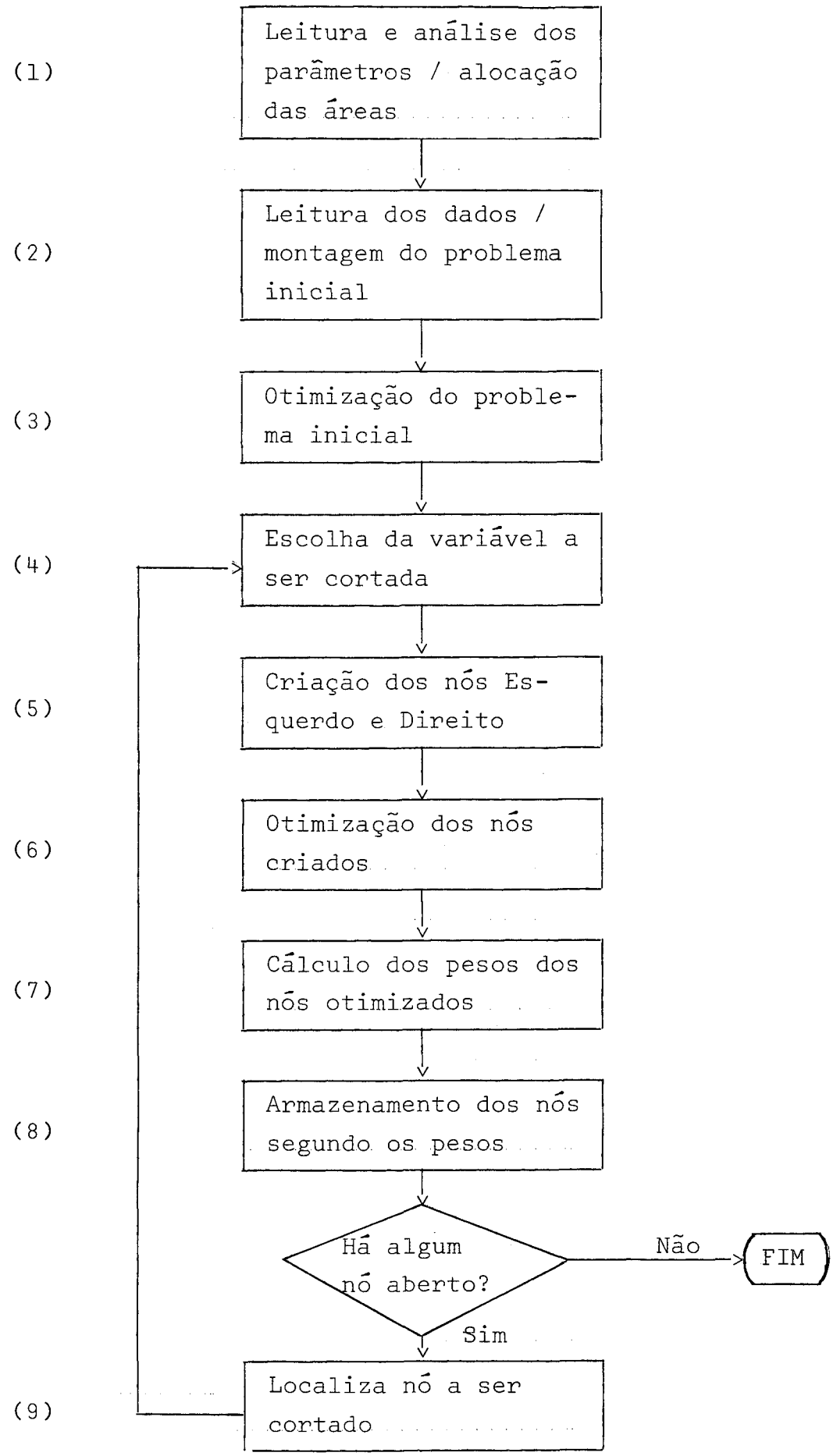
No último algorítmo dois pontos merecem ser comentados: a passagem de nós de disco para memória e a passagem de nós de fita para disco.

Na passagem de disco para memória, apenas um nó é passado de cada vez. Isto porque o acesso ao nó no disco é feito de maneira direta e o processo requer apenas a leitura do nó. Por outro lado, este nó na memória ao ser cortado gerará novos nós, podendo inclusive encher novamente toda a lista na memória.

Já a passagem de fita para disco envolve um número maior de operações, exigindo inclusive duas leituras na fita. Procuramos então diminuir estas passagens, transferindo de cada vez um grupo de nós. Arbitrariamente escolhemos transferir o número suficiente para ocupar a metade do espaço reservado no disco.

Pelo que foi até agora exposto, podemos ver que a definição dos critérios para escolha do nó será feita no módulo que calcula o peso do nó.

Na figura a seguir apresentamos os principais módulos lógicos do sistema, bem como suas ligações, passando então a descrever cada um dêles.


(1)
Leitura e análise dos parâmetros / alocação das áreas
(2)
Leitura dos dados / montagem do problema inicial
(3)
Otimização do problema inicial
(4)
Escolha da variável a ser cortada
(5)
Criação dos nós Esquerdo e Direito
(6)
Otimização dos nós criados
(7)
Cálculo dos pesos dos nós otimizados
(8)
Armazenamento dos nós segundo os pesos
Há algum nó aberto?
Não
FIM
Sim
(9)
Localiza nó a ser cortado

DESCRIÇÃO DOS MÓDULOS

.Módulo 1

Neste módulo são lidos os parâmetros que definem o problema a ser resolvido (número de variáveis do problema, número de variáveis inteiras, etc.), os recursos a serem utilizados na execução (área de memória principal, área em disco, etc.) e formato para leitura e apresentação dos dados (número de colunas no cartão de entrada, número de casas decimais dos valores fracionários, etc.). É também verificada a consistência dos mesmos.

A relação completa dos parâmetros, junto com as instruções para sua codificação, será apresentada no próximo capítulo.

Para os parâmetros opcionais, a ausência de expecificação implicará na adoção de valores pré-definidos[1].

Procuramos, com a utilização de parâmetros, dar um pouco de flexibilidade à codificação dos dados para o sistema e para a apresentação dos resultados, porém, para transformar este num sistema comercial, uma flexibilidade maior deve ser provida.

Também neste módulo são feitas a alocação e a "inicialização" das áreas de trabalho, de acordo com os valores dos parâmetros lidos ou assumidos.

.Módulo 2

Neste módulo são lidos os dados do problema usando as descrições definidas nos parâmetros, e é feita a montagem do problema inicial. Bàsicamente, são preenchidos a matriz A' e o nó 0 do problema[2].

(1) Ver também no próximo capítulo.

(2) Ver capítulo anterior.

.Módulo 3

Neste módulo o nó 0 é otimizado, de acordo com o esquema apresentado no capítulo II.

Já nesta etapa da implementação deparamo-nos com problemas de perda de precisão devidos à manipulação de valores em ponto flutuante.

Uma das manifestações destes problemas é o fato de valores que deveriam resultar nulos ou inteiros apresentarem resíduos fracionários. A solução adotada foi definir um valor (que no sistema chamamos de EPSI e que pode ser alterado na leitura dos parâmetros) que indica a precisão desejada. Assim, diferenças inferiores, em valor absoluto, a EPSI não serão consideradas, e em certos casos corrigidas.

Exemplificando, se temos EPSI = 0.005, uma variável com valor 0.004 será considerada nula e com valor 0.995 será considerada inteira.

Outra manifestação destes problemas é a existência de diferença entre o valor da função objetivo obtido diretamente do quadro ($\bar{z}$) e o valor calculado por meio dos valores correntes das variáveis básicas ($\bar{x}_B$). Neste caso a solução adotada foi recalcular B_i^{-1} sempre que este fato se verifica.

.Módulo 4

Neste módulo é escolhida a variável a ser cortada. Na fase de leitura dos dados (módulo 2, acima), são também lidos e armazenados os índices das variáveis inteiras. Na presente implementação, a ordem em que são lidos os índices define a prioridade para corte destas variáveis. A execução deste módulo reduz-se a percorrer a lista de variáveis inteiras, verificando para cada uma se a mesma está na base com valor fracionário. A primeira variável que satisfizer a essas condições será escolhida para ser cortada. Como só são armazenados nós abertos, pelo menos uma variável estará nesta condição.

Para modificar o critério de escolha da variável, este módulo deverá ser reprogramado.

.Módulo 5

Neste módulo são reservadas as áreas para os novos nós na memória principal[1] e introduzidas as alterações devidas aos cortes, esquerdo e direito, respectivamente, conforme descrito no capítulo anterior.

.Módulo 6

Neste módulo os novos nós são otimizados, de acordo com o esquema desenvolvido no capítulo III.

Da otimização de cada novo nó pode resultar que:

1) O nó representa um problema sem solução.

Neste caso o nó é fechado.

2) O nó apresenta uma solução viável, porém não inteira.

Neste caso é verificado se este nó pode conduzir a uma solução que interesse, dependendo do número de soluções inteiras desejadas, do valor limite da função objetivo imposto para estas soluções e do número e valores (função objetivo) das soluções inteiras já encontradas. Em suma, o valor obtido para a função objetivo nesta solução é comparado com os limites definidos pelas condições acima e caso não os satisfaça, o nó será fechado. Caso contrário o nó será passado para o próximo módulo.

3) O nó apresenta uma solução inteira.

Neste caso há também uma dependência das condições referidas no ítem anterior. Para maior clareza, recorreremos à exemplificação. Suponhamos que estamos maximizando, desejamos as 3 melhores soluções e o nó em questão apresenta solução inteira com $\overline{z} = 5$. As situações que podem ocorrer enqua-

(1) Ver "Estrutura de Armazenamento", neste capítulo, acima.

dram-se num dos casos seguintes:

a) Já foram encontradas três soluções inteiras, todas com valor (função objetivo) maior que 5. Neste caso o nó em questão é simplesmente fechado.

b) Mesmo com a presente solução o número de soluções inteiras é inferior a 3. Neste caso o nó é fechado e guardado para definição posterior (podem ou não surgir soluções melhores).

c) Com a presente solução o número de soluções inteiras fica igual ou maior que 3. Tomamos apenas as três melhores soluções, e destas a de menor valor. Eliminamos do problema todos os nós abertos com valor inferior a esse. Para os nós na memória principal ou no disco a eliminação é feita imediatamente. Para os nós em fita a eliminação é feita quando for preciso ler a fita[1]. O nó em questão é também fechado e guardado para definição posterior.

Para o caso de minimização, raciocínio semelhante se aplica.

.Módulo 7

Neste módulo são calculados os pesos dos nós otimizados no módulo anterior e que não foram fechados. Na presente implementação o cálculo do peso se resume em atribuir ao nó o peso atribuido ao nó anterior, incrementado de 1. Como este módulo é executado primeiro para o nó direito (quando existir), isto garante o percurso da árvore em pré-ordem.

Os critérios para escolha do nó a ser cortado são definidos neste módulo e convertidos no peso do nó. Para modificação destes critérios, este módulo deverá ser reprogramado.

(1) Ver o algorítmo II apresentado no tópico "Estrutura de Armazenamento" acima, neste capítulo.

.Módulo 8

Neste módulo é decidido e efetuado o armazenamento dos nós em questão, na memória principal, em disco ou em fita, de acordo com o algorítmo I apresentado no tópico "Estrutura de Armazenamento" acima, no início deste capítulo.

.Módulo 9

Neste módulo é localizado o nó aberto de maior peso, se existir algum nó aberto. A execução deste módulo corresponde à execução do algorítmo II apresentado no tópico "Estrutura de Armazenamento".

V - UTILIZAÇÃO DO SISTEMA E ALTERAÇÃO DOS CRITÉRIOS PARA ESCOLHA DO NÓ E DA VARIÁVEL A SER CORTADA

Para descrever a forma de utilização do sistema, recorreremos novamente ao exemplo numérico introduzido no capítulo II.

Inicialmente mostraremos como codificar os dados e os parâmetros, usando ao máximo os valores pré-definidos do sistema, isto é, como utilizar o sistema na sua forma básica. Em seguida apresentaremos a relação completa dos parâmetros e suas funções. E finalmente, exemplificaremos as alterações de critérios para escolha do nó e da variável a ser cortada.

Para facilitar o acompanhamento da apresentação repetiremos aqui a proposição do problema que queremos resolver:

maximizar $z = 3\,x_1 + 4\,x_2 + x_3$

sujeita ao seguinte conjunto de restrições:

$$8\,x_1 + 3\,x_2 + 4\,x_3 \leq 7$$
$$2\,x_1 + 6\,x_2 + x_3 \leq 3$$
$$x_1 + 4\,x_2 + 5\,x_3 \leq 8$$

$$x_1, x_2, x_3 \geq 0$$

x_1, x_2, x_3 inteiras.

Para utilizarmos o sistema precisamos introduzir manualmente as variáveis de folga[1] e colocar a função objetivo sob outra forma.

(1) Já nos referimos anteriormente à necessidade de maiores facilidades de utilização para transformá-lo num sistema comercial. Este é mais um exemplo. Outros serão indicados posteriormente.

O problema a resolver se apresenta então como:

maximizar

$$z - 3x_1 - 4x_2 - x_3 + 0x_4 + 0x_5 + 0x_6 = 0$$

sujeito a:

$$8x_1 + 3x_2 + 4x_3 + x_4 + 0x_5 + 0x_6 = 7$$
$$2x_1 + 6x_2 + x_3 + 0x_4 + x_5 + 0x_6 = 3$$
$$x_1 + 4x_2 + 5x_3 + 0x_4 + 0x_5 + x_6 = 8$$

x_1, x_2, x_3 inteiras.

Como todas as restrições eram do tipo $\leq$, as variáveis de folga nos fornecem a solução básica viável que precisamos para iniciar o processo.

Antes dos dados do problema, precisamos apresentar ao sistema os parâmetros que controlarão a execução. Todos os parâmetros são parâmetros por palavra chave, separados por uma vírgula ou por pelo menos um espaço e após o último deve vir um ponto e vírgula (;). Para o presente caso, codificaríamos os seguintes parâmetros[1]:

NVAR=6, NBASE=3 NINT=3 ;

A seguir codificaríamos os dados do problema, por coluna:

```
-3  8  2  1
-4  3  6  4
-1  4  1  5
 0  1  0  0
 0  0  1  0
 0  0  0  1
 0  7  3  8
```

seguido dos índices das variáveis que comporão a solução ini-

(1) Ver adiante a descrição.

cial, isto é, das variáveis básicas da solução inicial:

4 5 6

e finalmente os índices das variáveis inteiras:

1 2 3

Note-se que cada valor é alinhado à direita do campo. Para os dados do problema cada campo contem 3 colunas e para os índices das variáveis básicas e inteiras, 2. Estes valores podem ser alterados por meio dos parâmetros. Caso qualquer destes dados não caiba num único cartão, usa-se o cartão seguinte como continuação, com espacejamento igual ao anterior (isto é, um campo não pode começar num cartão e terminar em outro).

A tabela abaixo mostra um resumo dos parâmetros, seus valores pré-definidos e as restrições a que estão sujeitos:

NOME	VALOR PRÉ-DEFINIDO	RESTRIÇÕES
NVAR		Obrigatório, inteiro e > 0
NBASE		Obrigatório, inteiro e > 0
FASE	2	FASE=1 ou FASE=2
NINT	NVAR	Inteiro e 0 < NINT $\leq$ NVAR
NNO	3	Inteiro e > 2
NDISCO	6	Inteiro e = 0 ou > 2
NFITA	15	Inteiro. Se NDISCO=0 então NFITA=0
NCOLDAT	3	Inteiro e > NDECIN+2
NDECIN	0	Inteiro e $\geq$ 0
NCOLBAS	2	Inteiro e > 0 se FASE=2
NCOLINT	2	Inteiro e > 0
NCOLIMP	7	Inteiro.NCOLIMP>NDECOUT+3 e NCOLIMP>6
NDECOUT	3	Inteiro e $\geq$ 0
LTAM	80	Inteiro e 71 < LTAM < 133
EPSI	0.0005	0 < EPSI $\leq$ 0.05
NUMSOLU	1	Inteiro e 0 $\leq$ NUMSOLU < 11
MINMAX	0	MINMAX=0 ou MINMAX=1
MAXSOLU	-99999.E70	
INVSOLU	0	INVSOLU=0 ou INVSOLU=1

Os parâmetros restritos a valores inteiros terão seus valores arredondados pelo truncamento da parte fracionária, caso a restrição não seja obedecida.

Tanto os parâmetros como os demais dados do problema serão lidos, pelo sistema, do arquivo SYSIN ("DDNAME" da Job Control Language do Sistema IBM/370) em registros de tamanho fixo e igual a 80, motivo pelo qual nos referiremos adiante a cartão e coluna.

DESCRIÇÃO DOS PARÂMETROS:

.NVAR:

Número de variáveis do problema, incluindo as variáveis de folga.

.NBASE:

Número de restrições do problema, que indica tambem quantas variáveis comporão a base.

.FASE:

Este parâmetro admite apenas dois valores:

1 - Indica que não é conhecida nenhuma solução inicial. Neste caso o sistema calculará tambem a solução inicial, usando o método das duas fases[1]. O parâmetro NCOLBAS pode ser omitido e no conjunto de dados não podem estar os índices das variáveis básicas da solução inicial;

2 - Indica que o conjunto de variáveis de folga fornece a solução básica inicial. Neste caso os índices destas variáveis devem estar codificados no conjunto de dados do problema.

.NINT:

Número de variáveis inteiras do problema. Se omitido indicará que todas as variáveis do problema são inteiras, e neste caso não constarão do conjunto de dados do problema os índices das variáveis inteiras.

.NNO:

Número máximo desejado de nós na memória principal.

(1) Ver G. HADLEY - Linear Programming - capítulo 5.

.NDISCO:

Número máximo desejado de nós em disco. Para indicar que o sistema deve trabalhar apenas com a memória principal, este parâmetro deve ser codificado com valor zero (e neste caso será assumido tambem valor zero para o parâmetro NFITA). Caso o valor (codificado ou assumido) deste parâmetro seja diferente de zero, deverá estar disponível para o sistema um arquivo em disco, com nome DISCO. A fórmula a seguir deve ser utilizada para calcular o espaço necessário em disco, em função do valor de NDISCO:

$$N = \left\lceil NDISCO \;/\; \left\lfloor TRK \;/\; (IBG + 10 * (NVAR + NBASE) + 8) \right\rfloor \right\rceil$$

onde: N = número de trilhas necessárias;

TRK = tamanho da trilha em "bytes" (depende do modelo do disco usado);

IBG = "Inter Block Gap" em "bytes" (depende do modelo do disco usado);

$\lceil x \rceil$ = menor inteiro maior ou igual a x;

$\lfloor x \rfloor$ = maior inteiro menor ou igual a x;

.NFITA:

Número máximo desejado de nós em fita. Para indicar que o sistema não deverá trabalhar com fita, este parâmetro deve ser codificado com valor zero. Caso o valor (codificado ou assumido) deste parâmetro seja diferente de zero, deverão estar disponíveis para o sistema dois arquivos em fita, com nomes FITA1 e FITA2. Se durante a execução o espaço colocado à disposição do sistema (número de nós na memória principal + nú-

mero de nós em disco + número de nós em fita) não for suficiente (de acordo com os critérios definidos no algorítmo I, capítulo IV) o nó de menor peso existente neste instante será abandonado e será emitida uma mensagem notificando o fato e identificando o nó abandonado.

.NCOLDAT:

Número de colunas ocupadas por cada dado do problema, incluindo sinal e o ponto decimal, se existirem.

.NDECIN:

Número de casas decimais dos dados do problema (não considerando o ponto decimal).

.NCOLBAS:

Número de colunas ocupadas por cada índice de variável básica (quando FASE = 2).

.NCOLINT:

Número de colunas ocupadas por cada índice de variável inteira (quando for codificado NINT).

.NCOLIMP:

Número de colunas que deve ser reservado para impressão de cada valor, incluindo o sinal e o ponto decimal[1].

.NDECOUT:

Número desejado de casas decimais para impressão dos resultados.

.LTAM:

Tamanho, em caracteres, da linha para impressão dos resultados.

(1) Para todo e qualquer tipo de impressão - dados, mensagens e resultados - o sistema usa o arquivo SYSPRINT.

.NUMSOLU:

Número de soluções desejadas. Para indicar que se quer todas as soluções possíveis, deve-se codificar êste parâmetro com valor zero.

.EPSI:

Valor a ser usado para arredondamentos. Deve ser codificado como um número decimal sem sinal.

.MINMAX:

Tipo de otimização desejada. Admite apenas dois valores:

0 - para indicar maximização

1 - para minimização.

.MAXSOLU:

Valor limite para as soluções desejadas. Soluções iguais ou piores que MAXSOLU serão abandonadas.

.INVSOLU:

Indica se é (=1) ou não (=0) para incluir as inversas das bases (quadros) na apresentação dos resultados.

EXEMPLO DE ALTERAÇÃO DO CRITÉRIO PARA ESCOLHA DO NÓ:

Suponhamos que em lugar do critério adotado na implementação, desejamos utilizar o critério baseado no valor da função objetivo, isto é, queremos cortar a cada iteração o nó que apresentar o maior valor para a função objetivo.

Para o sistema, a escolha do nó a ser cortado é função do peso atribuido ao nó. Portanto a alteração do critério de escolha do nó será implementado por meio da alteração da rotina que calcula o peso do nó.

Usaremos então como peso do nó o valor da função objetivo. Como o peso do nó deve ser um valor inteiro compreendido entre 0 e 32767, atribuiremos, para simplificar o exemplo,

peso 0 aos nós com valor negativo para a função objetivo e 32767 para os nós com valor da função objetivo superior a 32767[1].

Na implementação do sistema, a rotina que calcula o peso dos nós é a função CALCULA[2].

Esta função recebe 6 parâmetros (na ordem apresentada):

NO - nó para o qual se quer calcular o peso;
NBASE - número de variáveis básicas;
NVAR - número de variáveis do problema;
INTEIRAS - vetor de índices das variáveis inteiras do problema;
NINT - número de variáveis inteiras do problema;
EPSI - valor a ser usado para arredondamentos.

A definição destes parâmetros em linguagem PL/I é mostrada abaixo (na codificação real, as dimensões das matrizes devem ser substituidas por * por serem parâmetros):

```
DCL 1 NO ,
      2 QUADRO(0:NBASE , 0:NBASE+2)   BIN FLOAT(53) ,
      2 BASE(NBASE)                   BIN FIXED(15,0) ,
      2 PESO                          BIN FIXED(15,0) ,
      2 LINK                          BIN FIXED(15,0) ,
      2 FLAG(NVAR)                    BIN FIXED(15,0) ,
      2 LIMITE(NVAR)                  BIN FLOAT(53) ;

DCL NBASE                             BIN FIXED(15,0) ;

DCL NVAR                              BIN FIXED(15,0) ;

DCL INTEIRAS(NINT)                    BIN FIXED(15,0) ;

DCL NINT                              BIN FIXED(15,0) ;

DCL EPSI                              BIN FLOAT(53) ;
```

(1) Diversos artifícios podem ser usados para contornar esta restrição, como por exemplo atribuir peso por faixas de valores da função objetivo.
(2) A versão implementada é mostrada no anexo A2.

A matriz QUADRO contém o quadro simplex revisado para êste nó, que pode ser resumido como:

	0	1 ..... NBASE	NBASE+1	NBASE+2
0	1	$\underset{\sim}{c}_{B_i}^T \; B_i^{-1}$	$\bar{z}$	
1				
⋮	$\underset{\sim}{0}$	B_i^{-1}	$\bar{\underset{\sim}{x}}_{B_i}$	
NBASE				

O vetor BASE contém os índices das variáveis básicas, na ordem em que aparecem no quadro.

A variável PESO indica o peso deste nó e a variável LINK é utilizada pelo sistema para efetuar o encadeamento dos nós.

O vetor FLAG tem cada elemento associado a uma variável do problema e indica se esta variável ainda não foi cortada (valor 0), se foi cortada à direita (valor 1) ou à esquerda (valor 2).

O vetor LIMITE tem tambem cada elemento associado a uma variável do problema e contém, para as variáveis já cortadas, o valor da parte inteira da variável na ocasião do corte.

Nenhuma alteração deve ser efetuada nos parâmetros desta função, para que o sistema não perca sua integridade.

Para implementar o novo critério de escolha do nó, descrito acima, podemos substituir a função CALCULA original pela mostrada no anexo A4.

EXEMPLO DE ALTERAÇÃO DO CRITÉRIO PARA ESCOLHA DA VARIÁVEL:

Suponhamos que em lugar do critério adotado na implementação, queremos cortar a cada iteração a variável que estiver mais distante de um valor inteiro.

Na implementação do sistema, a função DEFINEV[1] é usada para determinar o índice da variável a ser cortada.
Os parâmetros recebidos por esta função são os mesmos da função CALCULA, passados na mesma ordem.

Considerando que uma variável estará tanto mais distante de um valor inteiro quanto sua parte fracionária estiver mais próxima de 0.5, uma possível implementação deste critério pode ser obtida pela substituição da função DEFINEV original pela mostrada no anexo A5.

Tambem aqui cabe ressaltar que na reprogramação desta função deve-se cuidar que nenhuma alteração seja efetuada nos parâmetros desta função, para que o sistema não perca sua integridade.

(1) A versão implementada é mostrada no anexo A3.

VI. CONCLUSÕES

1. O trabalho apresentado tem conotação acadêmica, no sentido de permitir ao usuário testar e desenvolver possibilidades de escolha do nó e da variável a ser cortada. Para isso, procuramos dar ao usuário maior flexibilidade e facilidade para alteração dos critérios de escolha referidos acima;

2. A estruturação dos dados, no que se refere à armazenagem dos nós em memória principal, (e/ou) disco e (ou) fita, obedeceu ao critério de se ter mais rapidamente acessíveis os nós com maior prioridade para corte;

3. No tocante à entrada de dados não houve nenhuma tentativa de compatibilização com os códigos MPS-TEMPO da BURROUGHS, MPSX da IBM, UMPIRE da UNIVAC e APEX da CDC, que utilizam o formato já muito divulgado, do MPS;

4. O algorítmo de Programação Linear utilizado não leva em consideração as facilidades decorrentes de restrições especiais do tipo $a \leq x_j \leq b$ na fase inicial do Simplex. Este tipo de restrição só é considerada na fase de cortes. No entanto sua implementação não é dificil;

5. Quanto à programação propriamente dita, em muitos casos a otimização foi sacrificada em função de maior clareza. Portanto uma revisão efetuada no programa levará forçosamente a um código mais eficiente.

BIBLIOGRAFIA

1. DANTZIG, G. B. - Linear Programming and Extensions, Princeton - New Jersey, Princeton University Press, 1963.

2. HADLEY, G. - Linear Programming, Reading - Massachusetts, Addison-Wesley Publishing Company, Inc., 1965. 520p.

3. MACULAN FILHO, Nelson - Programação Linear Inteira, Rio de Janeiro, PDD - 17/78 - COPPE/UFRJ, 1978. 288p.

A N E X O S

A1 - Programa Principal

```
SIMPLER : PROC OPTIONS(MAIN) ;
/*****************************************************************/
/* VARIAVEIS QUE CONTROLAM A EXECUCAO:                           */
/* NOME        DESCRICAO                        DEFAULT  RESTRICOES      */
/* NVAR        N.VARIAVEIS DO PROBLEMA          -------  OBRIGATORIO    */
/* NBASE       N.VARIAVEIS NA BASE              -------  OBRIGATORIO    */
/* FASE        FASE EM QUE DEVE SER INICIA-     2                       */
/*             DA A EXECUCAO:                                           */
/*               1 = DEVE SER CALCULADA A                               */
/*                   SOLUCAO INICIAL INTRO-                             */
/*                   DUZINDO-SE VARIAVEIS                               */
/*                   ARTIFICIAIS                                        */
/*               2 = DEVEM SER LIDOS OS IN-           DEVEM SER INFOR-*/
/*                   DICES DAS VARIAVEIS              MADOS OS INDI-  */
/*                   QUE FORMAM A BASE DA             CES DAS VARIA-  */
/*                   SOLUCAO INICIAL.                 VEIS BASICAS DA */
/*                                                    SOLUCAO INICIAL.*/
/* MINMAX      TIPO DE OTIMIZACAO               0                       */
/*               0 = MAXIMIZACAO                                        */
/*               1 = MINIMIZACAO                                        */
/* NINT        N.VARIAVEIS INTEIRAS             NVAR                    */
/* NNO         N.NOS NA MEMORIA                 3        NNO>2          */
/* NDISCO      N.NOS NO DISCO                   6        0=NDISCO>2     */
/* NFITA       N.NOS NA FITA                    15       0 SE NDISCO=0  */
/* NCOLDAT     N.COLUNAS DO CARTAO PARA         3        NCOLDAT>NDECIN */
/*             CADA DADO                                 +2             */
/* NDECIN      N.CASAS DECIMAIS DO DADO DE      0        NDECIN>=0      */
/*             ENTRADA                                                  */
/* NCOLBAS     N.COLUNAS DO CARTAO PARA         2        NCOLBAS>0      */
/*             INDICE DE VARIAVEL QUE                                   */
/*             ESTEJA NA BASE                                           */
/* NCOLINT     N.COLUNAS DO CARTAO PARA         2        NCOLINT>0      */
/*             CADA INDICE DE VARIAVEL                                  */
/*             INTEIRA                                                  */
/* NCOLIMP     N.COLUNAS PARA IMPRESSSAO        7        NCOLIMP>NDECOUT */
/*             DE CADA VALOR                             +3             */
/*                                                       NCOLIMP>6      */
/* NDECOUT     N.CASAS DECIMAIS P/IMPRESSAO     3        NDECOUT>=0     */
/* LTAM        TAMANHO DA LINHA P/IMPRESSAO     80       71<LTAM<133    */
/* EPSI        VALOR LIMITE PARA ARREDONDA-     0.0005   0<EPSI<=0.05   */
/*             MENTO                                                    */
/* MAXSOLU     VALOR LIMITE PARA AS SOLU-     -99999.E70                */
/*             COES DESEJADAS. SOLUCOES                                 */
/*             IGUAIS OU PIORES QUE MAXSOLU                             */
/*             SERAO ABANDONADAS                                        */
/* NUMSOLU     NUMERO DE SOLUCOES DESEJADAS     1        SE NUMSOLU<=0  */
/*                                                       OU NUMSOLU>10, */
/*                                                       SERAO CALCULA- */
/*                                                       DAS TODAS AS SO-*/
/*                                                       LUCOES POSSIVEIS*/
/* INVSOLU     INDICA SE E (=1) OU NAO          0                       */
/*             (=0) PARA IMPRIMIR AS IN-                                */
/*             VERSAS (QUADROS) DAS SOLU-                               */
/*             COES INTEIRAS                                            */
/*****************************************************************/
/* TRACES :                                                      */
/*  1 - IMPRIME NOS ANTES E APOS OTIMIZACAO, E MENSAGENS INTERMEDIA- */
/*      RIAS.                                                    */
/*  2 - IMPRIME A SITUACAO DOS NOS NA MEMORIA IMEDIATAMENTE ANTES DE */
/*      CADA CORTE                                               */
```

```
/*  3 - IMPRIME NOS APOS CADA MODIFICACAO E MENSAGENS INTERMEDIARIAS */
/*  4 - AINDA NAO ESTA SENDO UTILIZADO                               */
/*  5 - AINDA NAO ESTA SENDO UTILIZADO                               */
/*  6 - AINDA NAO ESTA SENDO UTILIZADO                               */
/*  7 - AINDA NAO ESTA SENDO UTILIZADO                               */
/*  8 - AINDA NAO ESTA SENDO UTILIZADO                               */
/*  9 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 10 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 11 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 12 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 13 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 14 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 15 - AINDA NAO ESTA SENDO UTILIZADO                               */
/* 16 - MISCELANEA                                                   */
/*******************************************************************/
/* FLAG - USADO PARA INDICAR O ESTADO DA VARIAVEL                    */
/*        0 - NAO FOI CORTADA                                        */
/*        1 - CORTADA A DIREITA ( >= )                               */
/*        2 - CORTADA A ESQUERDA ( <= )                              */
/*******************************************************************/
DCL CALCULA ENTRY EXTERNAL RETURNS(BIN FIXED(15,0)) ;
DCL DEFINEV ENTRY EXTERNAL RETURNS(BIN FIXED(15,0)) ;
DCL (FITAIN,FITAOUT) FILE RECORD ENV(FB BLKSIZE(BLOCK) RECSIZE(REC)) ;
DCL (BLOCK,REC)          BIN FIXED(31,0) STATIC ;
DCL DISCO FILE RECORD DIRECT KEYED ENV(F REGIONAL(1) BLKSIZE(DISCBLK));
DCL DISCBLK              BIN FIXED(31,0) STATIC ;
DCL CHAVE                BIN FIXED(15,0) ;
DCL (FIN,FOUT,FAUX)      CHAR(8) ;
DCL I                    BIN FIXED ;
DCL II                   BIN FIXED(15,0) ;
DCL J                    BIN FIXED ;
DCL K                    BIN FIXED ;
DCL M                    BIN FIXED ;
DCL N                    BIN FIXED ;
DCL ENTRA                BIN FIXED ;
DCL SAI                  BIN FIXED ;
DCL MINMAX               BIN FIXED(15,0) INIT(0) ;
DCL NVAR                 BIN FIXED INIT(0) ;
DCL NBASE                BIN FIXED INIT(0) ;
DCL NINT                 BIN FIXED INIT(0) ;
DCL NNO                  BIN FIXED INIT(3) ;
DCL NDISCO               BIN FIXED INIT(6) ;
DCL NFITA                BIN FIXED INIT(15) ;
DCL NFITMAP              BIN FIXED ;
DCL FASE                 BIN FIXED(15,0) INIT(2) ;
DCL FUNCOBJ(NVAR+1)      BIN FLOAT(53) CONTROLLED ;
DCL OTIMO                BIN FIXED INIT(0) ;
DCL FIM                  BIT(1) INIT('0'B) ;
DCL ABRE                 BIT(1) INIT('0'B) ;
DCL MENOR                BIN FLOAT(53) ;
DCL MAX                  BIN FLOAT(53) INIT(99999.E70) ;
DCL MAXSOLU              BIN FLOAT(53) INIT(-99999.E70) ;
DCL AUX                  BIN FLOAT(53) ;
DCL AUXO                 BIN FLOAT(53) ;
DCL AUXSAI               BIN FLOAT(53) ;
DCL FASECHAR             CHAR(8) INIT('CONTINUO') ;
DCL CONTINUA             CHAR(16) INIT(' ') ;
DCL ATU                  BIN FIXED INIT(1) ;
DCL INT                  BIN FIXED ;
DCL EPSI                 BIN FLOAT(53) INIT(0.0005) ;
```

```
DCL IMPBASE              PIC '999' ;
DCL IMPCHAR(3)           CHAR(1) DEF IMPBASE ;
DCL DISP                 BIN FIXED INIT(2) ;
DCL DISCDISP             BIN FIXED ;
DCL HEADISC              BIN FIXED INIT(0) ;
DCL MAXDISCO             BIN FIXED INIT(-1) ;
DCL PFITAMAX             BIN FIXED INIT(-1) ;
DCL ULTFITA              BIN FIXED ;
DCL CONTFIN              BIN FIXED ;
DCL CONTFOUT             BIN FIXED INIT(0) ;
DCL CABEC                BIN FIXED ;
DCL DIR                  BIN FIXED INIT(2) ;
DCL ESQ                  BIN FIXED ;
DCL NCOLIMP              BIN FIXED INIT(7) ;
DCL NDECOUT              BIN FIXED INIT(3) ;
DCL NCOLDAT              BIN FIXED INIT(3) ;
DCL NDECIN               BIN FIXED INIT(0) ;
DCL NCOLBAS              BIN FIXED INIT(2) ;
DCL NCOLINT              BIN FIXED INIT(2) ;
DCL LTAM                 BIN FIXED INIT(80) ;
DCL NUMSOLU              BIN FIXED INIT(1) ;
DCL INVSOLU              BIN FIXED(15,0) INIT(0) ;
DCL DISPSOLU             BIN FIXED(15,0) ;
DCL CABECSOLU            BIN FIXED(15,0) INIT(0) ;
DCL N2                   BIN FIXED ;
DCL N3                   BIN FIXED ;
DCL TRACE(16)            BIT(1) INIT((16)(1) '0'B) ;
DCL MSGFINAL             BIT(1) INIT('1'B) ;
GET DATA ;
IF LTAM < 72
THEN LTAM = 72 ;
ELSE IF LTAM > 132
     THEN LTAM = 132 ;
OPEN FILE(SYSPRINT) LINESIZE(LTAM) ;
IF NVAR = 0
THEN DO ;
       PUT SKIP EDIT('*** PROGRAMA ENCERRADO. NUMERO DE VARIAVEIS ' ,
                     'NULO OU NAO INFORMADO.')(A,A) ;
       STOP ;
     END ;
IF NBASE = 0
THEN DO ;
       PUT SKIP EDIT('*** PROGRAMA ENCERRADO. NUMERO DE VARIAVEIS ' ,
                     'NA BASE NULO OU NAO INFORMADO.')(A , A) ;
       STOP ;
     END ;
IF EPSI <= 0 | EPSI > 0.05
THEN EPSI = 0.0005 ;
IF MINMAX ¬= 0
THEN MINMAX = 1 ;
IF INVSOLU ¬= 0
THEN INVSOLU = 1 ;
IF NINT = 0
THEN DO ;
       NINT = NVAR ;
       ABRE = '1'B ;
     END ;
IF NCOLDAT-3 < NDECIN
THEN NCOLDAT = NDECIN + 3 ;
IF NCOLIMP - 4 < NDECOUT
```

```
THEN NCOLIMP = NDECOUT + 4 ;
IF NCOLIMP < 7
THEN NCOLIMP = 7 ;
IF NNO < 3
THEN NNO = 3 ;
IF NDISCO < 3
THEN DO ;
       NDISCO = 0 ;
       NFITA = 0 ;
     END ;
ELSE DISCBLK = 2 * NBASE + 2 * NVAR + 8 * NVAR + 8 * ( NBASE + 1 ) ;
IF NFITA <= 0
THEN DO ;
       NFITA = 0 ;
       NFITMAP = 0 ;
     END ;
ELSE DO ;
       NFITMAP = NDISCO / 2 + 1 ;
       REC = DISCBLK + 2 ;
       IF REC >= 4000
       THEN BLOCK = REC ;
       ELSE DO ;
              BLOCK = 4000 / REC ;
              BLOCK = BLOCK * REC ;
            END ;
     END ;
IF EPSI <= 0. | EPSI > 0.05
THEN EPSI = 0.0005 ;
IF NUMSOLU < 0 | NUMSOLU > 10
THEN NUMSOLU = 0 ;
BEGIN ;
  DCL MATRIZ(0:NBASE,NVAR+1)          BIN FLOAT(53) ; /*DADOS INICIAIS */
  DCL BASINIT(NBASE)                  BIN FIXED ;
  DCL BASINOR(0:NBASE)                BIN FIXED ;
  DCL INVBO(0:NBASE,NBASE+1)          BIN FLOAT(53) ; /*INVERSA INICIAL*/
  DCL 1 GUARDSOLU(NUMSOLU) ,
        2 BASE(NBASE)                     BIN FIXED(15,0) ,
        2 FLAG(NVAR)                      BIN FIXED(15,0) ,
        2 LIMITE(NVAR)                    BIN FLOAT(53) ,
        2 XBARRA(0:NBASE)                 BIN FLOAT(53) ,
        2 LINK                            BIN FIXED(15,0) ;
  DCL 1 NO(NNO) ,
        2 QUADRO(0:NBASE,0:NBASE+2) BIN FLOAT(53) ,
        2 BASE(NBASE)               BIN FIXED ,
        2 PESO                      BIN FIXED ,
        2 LINK                      BIN FIXED ,
        2 FLAG(NVAR)                BIN FIXED ,
        2 LIMITE(NVAR)              BIN FLOAT(53) ;
  DCL 1 RDISCO ,
        2 BASE(NBASE)               BIN FIXED ,
        2 FLAG(NVAR)                BIN FIXED ,
        2 LIMITE(NVAR)              BIN FLOAT(53) ,
        2 XBARRA(0:NBASE)           BIN FLOAT(53) ;
  DCL 1 RFITA ,
        2 BASE(NBASE)               BIN FIXED ,
        2 FLAG(NVAR)                BIN FIXED ,
        2 LIMITE(NVAR)              BIN FLOAT(53) ,
        2 XBARRA(0:NBASE)           BIN FLOAT(53) ,
        2 PESO                      BIN FIXED ;
  DCL 1 DISCMAP(NDISCO) ,
```

```
      2 PDISCO                          BIN FIXED ,
      2 XBARDISC                        BIN FLOAT(53) ,
      2 LDISCO                          BIN FIXED ;
DCL 1 FITAMAP(NFITMAP) ,
      2 PFITA                           BIN FIXED ,
      2 POSFITA                         BIN FIXED ,
      2 LFITA                           BIN FIXED ;
DCL INTEIRAS(NINT)                      BIN FIXED ;
IF NDISCO ¬= 0
THEN DO ; /* FORMATA DISCO */
       OPEN FILE(DISCO) OUTPUT ;
       DO CHAVE = 0 TO NDISCO - 1 ;
         WRITE FILE(DISCO) FROM(RDISCO) KEYFROM(CHAVE) ;
       END ;
       CLOSE FILE(DISCO) ;
       OPEN FILE(DISCO) UPDATE ;
     END ;
IF NFITA ¬= 0
THEN DO ;
       FIN = 'FITA1' ;
       FOUT = 'FITA2' ;
       OPEN FILE(FITAOUT) TITLE(FOUT) OUTPUT ;
       OPEN FILE(FITAIN) TITLE(FIN) INPUT ;
     END ;
DO I = NUMSOLU TO 1 BY -1 ;
  GUARDSOLU(I).LINK = I - 1 ;
END ;
DISPSOLU = NUMSOLU ;
NO(1).FLAG(*) = 0 ;
NO(1).PESO = 0 ;
NO(1).LINK = 0 ;
DO I = 2 TO NNO-1 ;
  NO(I).LINK = I + 1 ;
END ;
NO(NNO).LINK = 0 ;
DO I = NDISCO TO 1 BY -1 ;
  LDISCO(I) = I - 1 ;
END ;
DISCDISP = NDISCO ;
PFITA(*) = -1 ;
N2 = 80 / NCOLDAT ;
N3 = (NBASE + 1) / N2 + 1 ;
DO J = 1 TO NVAR+1 ;
  GET EDIT((MATRIZ(I,J) DO I = 0 TO NBASE))
          ((N3)(COL(1) , (N2)F(NCOLDAT,NDECIN))) ;
END ;
PUT EDIT('TIPO DE OTIMIZACAO : ')(COL(37) , A) ;
IF MINMAX = 1
THEN PUT EDIT('MINIMIZACAO')(A) ;
ELSE PUT EDIT('MAXIMIZACAO')(A) ;
PUT EDIT('NUMERO DE VARIAVEIS DO PROBLEMA : ' , NVAR ,
         'NUMERO DE VARIAVEIS NA BASE : ' , NBASE ,
         'INICIAR PELA FASE : ' , FASE ,
         'NUMERO DE VARIAVEIS INTEIRAS : ' , NINT ,
         'NUMERO DE NOS NA MEMORIA : ' , NNO ,
         'NUMERO DE NOS NO DISCO : ' , NDISCO ,
         'NUMERO DE NOS NA FITA : ' , NFITA ,
         'NUMERO DE COLUNAS DOS DADOS DE ENTRADA : ' , NCOLDAT ,
         'NUMERO DE CASAS DECIMAIS DOS DADOS DE ENTRADA : ' , NDECIN)
        (COL(24) , A , F(3) ,
```

```
          COL(28) , A , F(3) ,
          COL(38) , A , F(1) ,
          COL(27) , A , F(3) ,
          COL(31) , A , F(3) ,
          COL(33) , A , F(3) ,
          COL(34) , A , F(3) ,
          COL(17) , A , F(3) ,
          COL(10) , A , F(3)) ;
IF FASE = 2
THEN PUT EDIT('NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS NA BASE',
                ' : ' , NCOLBAS)(COL(4) , A , A , F(3)) ;
PUT EDIT('NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS INTEIRAS : ' ,
          NCOLINT ,
          'NUMERO DE COLUNAS PARA IMPRESSAO DOS RESULTADOS : ' ,
          NCOLIMP ,
          'NUMERO DE CASAS DECIMAIS PARA IMPRESSAO DOS RESULTADOS : ',
          NDECOUT ,
          'TAMANHO DA LINHA PARA IMPRESSAO : ' , LTAM ,
          'PRECISAO UTILIZADA (EPSI) : ' , EPSI ,
          'VALOR LIMITE PARA AS SOLUCOES : ' , MAXSOLU ,
          'NUMERO DE SOLUCOES PROCURADAS : ')
         (COL(3) , A , F(3) ,
          COL(8) , A , F(3) ,
          COL(1) , A , F(3) ,
          COL(24) , A , F(3) ,
          COL(30) , A , F(15,13) ,
          COL(26) , A , E(15,7) ,
          COL(26) , A) ;
IF NUMSOLU = 0
THEN PUT EDIT('TODAS')(A) ;
ELSE PUT EDIT(NUMSOLU)(F(3)) ;
PUT EDIT('IMPRESSAO DAS INVERSAS :')(COL(33) , A) ;
IF INVSOLU = 1
THEN PUT EDIT('SIM')(A) ;
ELSE PUT EDIT('NAO')(A) ;
PUT SKIP ;
DO I = 1 TO 16 ;
  IF TRACE(I)
  THEN PUT EDIT('TRACE(' , I , ') ')(A , F(2) , A) ;
END ;
IF NCOLDAT < 7
THEN NCOLDAT = 7 ;
N2 = LTAM / NCOLDAT ;
DO K = 1 TO NVAR+1 BY N2 ;
  PUT SKIP(2) EDIT('PROBLEMA INICIAL ' , CONTINUA)(COL(20) , A , A) ;
  CONTINUA = 'CONTINUACAO' ;
  N3 = K + N2 - 1 ;
  IF N3 >= NVAR+1
  THEN DO ;
         N3 = NVAR + 1 ;
         N2 = N3 - K + 1 ;
         PUT SKIP(2) EDIT(('X' , IMPBASE DO IMPBASE = K TO N3-1) ,
                          'XBARRA')((N2-1)(X(NCOLDAT-4) , A , A) ,
                          X(NCOLDAT-5) , A) ;
       END ;
  ELSE PUT SKIP(2) EDIT(('X' , IMPBASE DO IMPBASE = K TO N3))
                       ((N2)(X(NCOLDAT-4) , A , A)) ;
  DO I = 0 TO NBASE ;
    PUT EDIT((MATRIZ(I,J) DO J = K TO N3))
             (COL(1) , (N2)F(NCOLDAT,NDECIN)) ;
```

```
   END ;
 END ;
 /* TROCA SINAL DA FUNCAO OBJETIVO SE FOR MINIMIZACAO */
 IF MINMAX = 1
 THEN DO K = 1 TO NVAR + 1 ;
        MATRIZ(0,K) = -MATRIZ(0,K) ;
      END ;
 IF FASE = 2
 THEN DO ; /* LE INDICES DAS VARIAVEIS DA BASE */
        N2 = 80 / NCOLBAS ;
        GET EDIT(BASINIT)(COL(1) , (N2)F(NCOLBAS)) ;
        /* ORDENA BASE INICIAL */
        CALL ORDENA ;
ORDENA : PROC ;
  DCL (I,J,K,N)            BIN FIXED(15,0) ;
  BASINOR(0) = 1 ;
  BASINOR(1) = 0 ;
  DO I = 2 TO NBASE ;
    J = BASINOR(0) ;
    K = 0 ;
    N = BASINIT(I) ;
    DO WHILE(J ¬= 0 & BASINIT(J) < N) ;
      K = J ;
      J = BASINOR(J) ;
    END ;
    BASINOR(K) = I ;
    BASINOR(I) = J ;
  END ;
END ORDENA ;
      END ;
 ELSE DO ;
        /* GUARDA FUNCAO OBJETIVO */
        ALLOC FUNCOBJ ;
        DO I = 1 TO NVAR + 1 ;
          FUNCOBJ(I) = MATRIZ(0,I) ;
          MATRIZ(0,I) = 0. ;
        END ;
        /* GUARDA ORDEM DA BASE INICIAL */
        DO I = 1 TO NBASE ;
          BASINIT(I) = NVAR + I ;
          BASINOR(I-1) = I ;
        END ;
        BASINOR(NBASE) = 0 ;
      END ;
 IF ABRE /* TODAS AS VARIAVEIS SAO INTEIRAS ? */
 THEN DO I = 1 TO NVAR ;
        INTEIRAS(I) = I ;
      END ;
 ELSE DO ;
        N2 = 80 / NCOLINT ;
        GET EDIT(INTEIRAS)(COL(1) , (N2)F(NCOLINT)) ;
      END ;
 /* PREPARA QUADRO INICIAL */
 DO I = 1 TO NBASE ;
   NO(1).QUADRO(I,0) = 0. ;
   NO(1).QUADRO(I,NBASE+1) = MATRIZ(I,NVAR+1) ;
   IF TRACE(1)
   THEN NO(1).QUADRO(I,NBASE+2) = 0. ;
 END ;
 NO(1).QUADRO(0,0) = 1. ;
```

```
IF FASE = 2
THEN DO ;
       NO(1).QUADRO(0,NBASE+1) = MATRIZ(0,NVAR+1) ;
       DO J = 1 TO NBASE ;
         N = BASINIT(J) ;
         NO(1).BASE(J) = N ;
         DO I = 0 TO NBASE ;
           NO(1).QUADRO(I,J) = MATRIZ(I,N) ;
         END ;
       END ;
     END ;
ELSE DO ;
       AUX = 0. ;
       DO J = 1 TO NBASE ;
         AUX = AUX - NO(1).QUADRO(J,NBASE+1) ;
         NO(1).BASE(J) = NVAR + J ;
         NO(1).QUADRO(0,J) = -1. ;
         DO I = 1 TO NBASE ;
           IF I = J
           THEN NO(1).QUADRO(I,J) = 1. ;
           ELSE NO(1).QUADRO(I,J) = 0. ;
         END ;
       END ;
       NO(1).QUADRO(0,NBASE+1) = AUX ;
     END ;
/* GUARDA QUADRO INICIAL */
DO I = 0 TO NBASE ;
  DO J = 1 TO NBASE + 1 ;
    INVBA(I,J) = NO(1).QUADRO(I,J) ;
  END ;
END ;
IF TRACE(1) & ¬ TRACE(3)
THEN CALL IMPRIME(NO(1) , 1) ;
/* PROCURA DA SOLUCAO OTIMA PARA O PROBLEMA CONTINUO */
DO WHILE(OTIMO = 0) ;
  MENOR = 0. ;
  DO J = 1 TO NVAR ;
    AUX = 0. ;
    DO N = 0 TO NBASE ;
      AUX = AUX + NO(1).QUADRO(0,N) * MATRIZ(N,J) ;
    END ;
    IF AUX < MENOR
    THEN DO ;
           ENTRA = J ;
           MENOR = AUX ;
         END ;
  END ;
  IF ABS(MENOR) < EPSI
  THEN OTIMO = 1 ; /* ESTAMOS NUM OTIMO */
  ELSE DO ; /* VARIAVEL INDICE ENTRA IRA PARA A BASE */
         /* CALCULA RESTO DO VETOR */
         NO(1).QUADRO(0,NBASE+2) = MENOR ;
         DO I = 1 TO NBASE ;
           AUX = 0. ;
           DO N = 0 TO NBASE ;
             AUX = AUX + NO(1).QUADRO(I,N) * MATRIZ(N,ENTRA) ;
           END ;
           NO(1).QUADRO(I,NBASE+2) = AUX ;
         END ;
         IF TRACE(3)
```

```
        THEN CALL IMPRIME(NO(1) , 1) ;
        /* ESCOLHA DA VARIAVEL QUE SAI DA BASE */
        MENOR = MAX ;
        SAI = 0 ;
        DO I = 1 TO NBASE ;
          IF NO(1).QUADRO(I,NBASE+2) > 0.
          THEN DO ;
                 AUX = NO(1).QUADRO(I,NBASE+1) /
                       NO(1).QUADRO(I,NBASE+2) ;
                 IF AUX < MENOR
                 THEN DO ;
                        SAI = I ;
                        MENOR = AUX ;
                      END ;
               END ;
        END ;
        IF SAI = 0
        THEN DO ;
               IF ¬TRACE(3)
               THEN CALL IMPRIME(NO(1) , 1) ;
               PUT SKIP LIST('O PROBLEMA NAO TEM SOLUCAO LIMITADA');
               STOP ;
             END ;
        IF TRACE(1) | TRACE(3)
        THEN DO ;
               IMPBASE = ENTRA ;
               PUT EDIT('VARIAVEL X',IMPBASE,' ENTRA NA BASE, SAI ',
                        'VARIAVEL X')(COL(1) , (4) A) ;
               IMPBASE = NO(1).BASE(SAI) ;
               PUT EDIT(IMPBASE)(A) ;
             END ;
        NO(1).BASE(SAI) = ENTRA ;
        CALL PIVOTA(NO(1),SAI) ;
      END ;
IF FASE = 1 & (OTIMO = 1 | ABS(NO(1).QUADRO(0,NBASE+1)) < EPSI)
THEN DO ; /* TERMINOU FASE 1 */
        IF ABS(NO(1).QUADRO(0,NBASE+1)) > EPSI
        THEN DO ;
               PUT SKIP EDIT('O PROBLEMA NAO TEM SOLUCAO VIAVEL')
                            (4) ;
               STOP ;
             END ;
        /* PREPARA INICIO DA FASE 2 */
        IF TRACE(1) | TRACE(3)
        THEN DO ;
               PUT SKIP LIST('****  OTIMO  FASE  1  ****') ;
               CALL IMPRIME(NO(1) , 1) ;
             END ;
        /* RETIRA VARIAVEIS ARTIFICIAIS DA BASE, SE HOUVER (E FOR */
        /* POSSIVEL)                                              */
        DO I = 1 TO NBASE ;
          IF NO(1).BASE(I) > NVAR
          THEN DO ; /* VARIAVEL ARTIFICIAL NA BASE */
                 AUX = 0. ;
                 DO J = 1 TO NVAR WHILE(ABS(AUX) <= EPSI) ;
                   AUX = 0. ;
                   /* MULTIPLICA LINHA I DO QUADRO PELA COLUNA J */
                   /* DA MATRIZ                                  */
                   DO K = 1 TO NBASE ;
                     AUX = AUX + NO(1).QUADRO(I,K) * MATRIZ(K,J) ;
```

```
            END ;
          END ;
          IF ABS(AUX) > EPSI
          THEN DO ; /* SAI DA BASE BASE A VARIAVEL ARTIFI- */
                    /* CIAL BASE(I) E ENTRA A VARIAVEL J-1 */
                 /* CALCULA RESTO DO VETOR P/PIVOTAMENTO */
                 NO(1).QUADRO(I,NBASE+2) = AUX ;
                 DO I1 = 0 TO I-1 , I+1 TO NBASE ;
                   AUX = 0. ;
                   /* MULTIPLICA LINHA I1 DO QUADRO PELA */
                   /* COLUNA J-1 DA MATRIZ               */
                   DO K = 0 TO NBASE ;
                     AUX = AUX + NO(1).QUADRO(I1,K) *
                           MATRIZ(K,J-1) ;
                   END ;
                   NO(1).QUADRO(I1,NBASE+2) = AUX ;
                 END ;
                 NO(1).BASE(I) = J - 1 ;
                 CALL PIVOTA(NO(1) , I) ;
               END ;
        END ;
END ;
/* PREPARA QUADRO INICIAL PARA FASE 2 */
/* REFAZ PRIMEIRA LINHA = (C(B(I)) * INVERSA(B(I))) */
DO J = 1 TO NBASE ;
  AUX = 0. ;
  DO I = 1 TO NBASE ;
    IF NO(1).BASE(I) > NVAR
    THEN AUX = AUX - NO(1).QUADRO(I,J) ;
    ELSE AUX = AUX - FUNCOBJ(NO(1).BASE(I)) *
               NO(1).QUADRO(I,J) ;
  END ;
  IF ABS(AUX) < EPSI
  THEN NO(1).QUADRO(0,J) = 0. ;
  ELSE NO(1).QUADRO(0,J) = AUX ;
END ;
/* RECALCULA O VALOR DA FUNCAO OBJETIVO PARA ESTA SOLUCAO */
/* = C(B(I)) * INVERSA(B(I)) * VETOR B                    */
AUX = 0. ;
DO J = 1 TO NBASE ;
  AUX = AUX + NO(1).QUADRO(0,J) * MATRIZ(J,NVAR+1) ;
END ;
IF ABS(AUX) < EPSI
THEN NO(1).QUADRO(0,NBASE+1) = 0. ;
ELSE NO(1).QUADRO(0,NBASE+1) = AUX ;
/* REFAZ PRIMEIRA LINHA DA MATRIZ */
DO I = 1 TO NVAR+1 ;
  MATRIZ(0,I) = FUNCOBJ(I) ;
END ;
FREE FUNCOBJ ; /* LIBERA AREA */
/* GUARDA O NOVO QUADRO INICIAL */
DO I = 0 TO NBASE ;
  DO J = 1 TO  NBASE + 1 ;
    INVBO(I,J) = NO(1).QUADRO(I,J) ;
  END ;
END ;
/* GUARDA NOVA BASE INICIAL */
DO I = 1 TO NBASE ;
  BASINIT(I) = NO(1).BASE(I) ;
END ;
```

```
        /* GUARDA ORDEM DA NOVA BASE INICIAL */
        CALL ORDENA ;
        OTIMO = 0 ;
        FASE = 2 ;
        PUT SKIP LIST('****  FIM  DA  FASE  1  ****') ;
        CALL IMPRIME(NO(1) , 1) ;
      END ;
END ;
CALL IMPRIME(NO(1) , 1) ;
PUT SKIP(2) EDIT('SOLUCAO OTIMA PARA O PROBLEMA ' , FASECHAR)(A,A) ;
FASECHAR = 'INTEIRO' ;
ABRE = '0'B ;
DO I = 1 TO NINT WHILE(¬ABRE) ;
  INT = INTEIRAS(I) ;
  DO SAI = 1 TO NBASE WHILE(INT ¬= NO(1).BASE(SAI)) ;
  END ;
  IF SAI <= NBASE & ABS(NO(1).QUADRO(SAI,NBASE+1)
     - FIXED(NO(1).QUADRO(SAI,NBASE+1) + EPSI)) > EPSI
  THEN ABRE = '1'B ;
END ;
IF ¬ABRE
THEN DO ;
       PUT EDIT('OTIMO INTEIRO')(COL(1) , A) ;
       MSGFINAL = '0'B ;
       FIM = '1'B ;
     END ;
/* PROCURA SOLUCOES PARA O PROBLEMA INTEIRO */
DO WHILE(¬FIM) ;
  IF TRACE(2)
  THEN DO ;
         NTRACE1 = LTAM / (NCOLIMP + 8) ;
         BEGIN ;
           DCL 1 A(NTRACE1) ,
                 2 NO                    BIN FIXED ,
                 2 PESO                  BIN FIXED ,
                 2 BASE(0:NBASE)         BIN FIXED ,
                 2 VALOR(0:NBASE)        BIN FLOAT(53) ;
           DCL (NTRACE2 , NTRACE3 , NTRACE4 , NTRACE5) BIN FIXED ;
           PUT SKIP(2) EDIT('DISP = ' , DISP , ' ATU = ' , ATU)
                           (A , F(3) , A , F(3)) ;
           NTRACE2 = ATU ;
           DO WHILE(NTRACE2 ¬= 0) ;
             DO NTRACE3 = 1 TO NTRACE1 WHILE(NTRACE2 ¬= 0) ;
               A(NTRACE3).NO = NTRACE2 ;
               A(NTRACE3).PESO = NO(NTRACE2).PESO ;
               A(NTRACE3).BASE(0) = 0 ;
               A(NTRACE3).VALOR(0) = NO(NTRACE2).QUADRO(0,NBASE+1) ;
               DO NTRACE4 = 1 TO NBASE ;
                 A(NTRACE3).BASE(NTRACE4)=NO(NTRACE2).BASE(NTRACE4) ;
                 A(NTRACE3).VALOR(NTRACE4) =
                                 NO(NTRACE2).QUADRO(NTRACE4,NBASE+1) ;
               END ;
               NTRACE2 = NO(NTRACE2).LINK ;
             END ;
             PUT SKIP EDIT(('* NO = ' , A(NTRACE4).NO DO NTRACE4=1
                           TO NTRACE3-1) , ('* BASE' , 'XBARRA '
                           DO NTRACE4=1 TO NTRACE3-1))((NTRACE3-1)
                           (A , F(3) , X(NCOLIMP-2)) , COL(1) ,
                           (NTRACE3-1)(A , X(NCOLIMP-5) , A)) ;
             DO NTRACE5 = 0 TO NBASE ;
```

```
            PUT SKIP EDIT(('* ' , A(NTRACE4).BASE(NTRACE5) ,
                          A(NTRACE4).VALOR(NTRACE5) DO NTRACE4=1
                          TO NTRACE3-1))((NTRACE3-1)(A , F(4) ,
                          F(NCOLIMP,NDECOUT) , X(16-NCOLIMP))) ;
          END ;
          PUT EDIT(('* PESO = ' , A(NTRACE4).PESO DO NTRACE4=1
                  TO NTRACE3-1))(COL(1) , (NTRACE3-1)(A , F(3) ,
                  X(NCOLIMP-4))) ;
        END ;
      END ; /* BEGIN */
    END ; /* TRACE(2) */
/* ESCOLHA DA VARIAVEL QUE VAI CORTAR */
SAI = DEFINEV(NO(ATU),NBASE,NVAR,INTEIRAS,NINT,EPSI) ;
/* CORTA SAI-ESIMA VARIAVEL DA BASE */
J = NO(ATU).BASE(SAI) ;
IF TRACE(1) | TRACE(3)
THEN DO ;
      IMPBASE = J ;
      PUT EDIT('CORTA VARIAVEL X' , IMPBASE)(COL(1) , A , A) ;
    END ;
/* RESERVA UM NO PARA PROBLEMA A DIREITA */
IF DISP ¬= 0
THEN DO ; /* HA NO DISPONIVEL */
      DIR = DISP ;
      DISP = NO(DISP).LINK ;
    END ;
ELSE DO ; /* NAO HA NO DISPONIVEL */
      /* JOGA NO DE MENOR PESO PARA DISCO */
      DIR = ATU ;
      DO WHILE(NO(DIR).LINK ¬= 0) ;
        K = DIR ;
        DIR = NO(DIR).LINK ;
      END ;
      NO(K).LINK = 0 ;
      CALL GRAVA(DIR) ; /* JOGA PARA DISCO LISTA DE NOS APONTA- */
                        /* DA POR DIR                          */
    END ;
/* DESLIGA NO ESQUERDO DA LISTA E APONTA POR ESQ */
ESQ = ATU ;
ATU = NO(ATU).LINK ;
IF NO(ESQ).FLAG(J) ¬= 0
THEN DO ; /* VARIAVEL COMPLEMENTAR NA BASE */
      IF TRACE(1) | TRACE(3)
      THEN PUT SKIP EDIT('VARIAVEL COMPLEMENTAR NA BASE. FOI FE' ,
                         'ITA SUBSTITUICAO DE VARIAVEL')(A , A) ;
      IF NO(ESQ).FLAG(J) = 1 /* SUBSTIT.DEVIDO CORTE DIREITA ? */
      THEN NO(ESQ).QUADRO(SAI,NBASE+1)=NO(ESQ).QUADRO(SAI,NBASE+1)
                       + NO(ESQ).LIMITE(J) ;
      ELSE DO ; /* SUBSTITUICAO DEVIDO A CORTE A ESQUERDA */
             NO(ESQ).QUADRO(SAI,NBASE+1) =
                 NO(ESQ).QUADRO(SAI,NBASE+1) - NO(ESQ).LIMITE(J) ;
             DO I = 0 TO NBASE+1 ;
               NO(ESQ).QUADRO(SAI,I) = - NO(ESQ).QUADRO(SAI,I) ;
             END ;
           END ;
    END ;
/* CRIA NO A DIREITA */
AUX = FIXED(NO(ESQ).QUADRO(SAI,NBASE+1) + EPSI) ;
NO(ESQ).QUADRO(SAI,NBASE+1) = NO(ESQ).QUADRO(SAI,NBASE+1) - AUX ;
NO(DIR) = NO(ESQ) ;
```

```
NO(DIR).QUADRO(SAI,NBASE+1) = NO(DIR).QUADRO(SAI,NBASE+1) - 1. ;
NO(DIR).FLAG(J) = 1 ;
NO(DIR).LIMITE(J) = AUX + 1 ;
/* COMPLETA NO ESQUERDO */
NO(ESQ).FLAG(J) = 2 ;
NO(ESQ).LIMITE(J) = AUX ;
DO J = 0 TO NBASE + 1 ;
  NO(ESQ).QUADRO(SAI,J) = - NO(ESQ).QUADRO(SAI,J) ;
END ;
IF TRACE(1) | TRACE(3)
THEN CALL IMPRIME(NO(DIR) , DIR) ;
CALL OTIMIZA(NO(DIR)) ;
CABEC = 0 ;
IF OTIMO = 1 & NO(DIR).QUADRO(0,NBASE+1) - MAXSOLU > EPSI
THEN DO ; /* SOLUCAO VIAVEL NAO INTEIRA */
       /* CALCULA PESO NO DIREITO */
       NO(DIR).PESO = CALCULA(NO(DIR) , NBASE , NVAR , INTEIRAS ,
                              NINT , EPSI) ;
       CABEC = DIR ;
       NO(DIR).LINK = 0 ;
     END ;
ELSE DO ; /* SOLUCAO INTEIRA OU NAO VIAVEL OU PIOR QUE MAXSOLU */
       IF OTIMO = 2 & NO(DIR).QUADRO(0,NBASE+1) - MAXSOLU > EPSI
       THEN DO ; /* SOLUCAO INTEIRA MELHOR QUE MAXSOLU */
              IF TRACE(1) | TRACE(3) | NUMSOLU = 0
              THEN DO ;
              PUT EDIT('OTIMO INTEIRO')(COL(1) , A) ;
              PUT SKIP(2) EDIT('SOLUCAO OTIMA PARA O PROBLEMA ' ,
                               FASECHAR , ' :')((3)A) ;
              CALL IMPSOLU(NO(DIR)) ;
                   END ;
              IF NUMSOLU ^= 0
              THEN IF GUARDA(NO(DIR)) /* GUARDA ESTA SOLUCAO */
                   THEN CALL ELIMINA ; /* ELIMINA NOS COM SOLU- */
                                       /* COES PIORES QUE AS ME-*/
                                       /* LHORES ENCONTRADAS    */
            END ;
       /* DEVOLVE NO PARA LISTA DE NOS DISPONIVEIS */
       NO(DIR).LINK = DISP ;
       DISP = DIR ;
     END ;
IF TRACE(1) | TRACE(3)
THEN CALL IMPRIME(NO(ESQ) , ESQ) ;
CALL OTIMIZA(NO(ESQ)) ;
IF OTIMO = 1 & NO(ESQ).QUADRO(0,NBASE+1) - MAXSOLU > EPSI
THEN DO ; /* SOLUCAO VIAVEL NAO INTEIRA */
       /* CALCULA PESO NO ESQUERDO */
       NO(ESQ).PESO = CALCULA(NO(ESQ) , NBASE , NVAR , INTEIRAS ,
                              NINT , EPSI) ;
       IF CABEC = 0
       THEN DO ;
              CABEC = ESQ ;
              NO(ESQ).LINK = 0 ;
            END ;
       ELSE IF NO(CABEC).PESO >= NO(ESQ).PESO
            THEN DO ;
                   NO(ESQ).LINK = 0 ;
                   NO(CABEC).LINK = ESQ ;
                 END ;
            ELSE DO ;
```

```
                    NO(ESQ).LINK = CABEC ;
                    CABEC = ESQ ;
                  END ;
      END ;
ELSE DO ; /* SOLUCAO INTEIRA OU NAO VIAVEL OU PIOR QUE MAXSOLU */
        IF OTIMO = 2 & NO(ESQ).QUADRO(0,NBASE+1) - MAXSOLU > EPSI
        THEN DO ; /* SOLUCAO INTEIRA MELHOR QUE MAXSOLU */
               IF TRACE(1) | TRACE(3) | NUMSOLU = 0
               THEN DO ;
               PUT EDIT('OTIMO INTEIRO')(COL(1) , A) ;
               PUT SKIP(2) EDIT('SOLUCAO OTIMA PARA O PROBLEMA ' ,
                                FASECHAR , ' :')((3)A) ;
               CALL IMPSOLU(NO(ESQ)) ;
                    END ;
               IF NUMSOLU ¬= 0
               THEN IF GUARDA(NO(ESQ)) /* GUARDA ESTA SOLUCAO */
                    THEN CALL ELIMINA ; /* ELIMINA NOS COM SOLU-  */
                                        /* COES PIORES QUE AS ME- */
                                        /* LHORES ENCONTRADAS.    */
             END ;
        /* DEVOLVE NO PARA LISTA DE NOS DISPONIVEIS */
        NO(ESQ).LINK = DISP ;
        DISP = ESQ ;
      END ;
IF CABEC ¬= 0
THEN DO ; /* COLOCA NOS NA LISTA POR PESO */
        IF HEADISC = 0
        THEN IF NO(CABEC).PESO < PFITAMAX
             THEN MAXDISCO = PFITAMAX ;
             ELSE MAXDISCO = -1 ;
        ELSE MAXDISCO = PDISCO(HEADISC) ;
        J = NO(CABEC).LINK ;
        IF NO(CABEC).PESO < MAXDISCO
        THEN DO ; /* JOGA OS DOIS NOS PARA DISCO */
               CALL GRAVA(CABEC) ;
               /* DEVOLVE NOS PARA LISTA DE NOS DISPONIVEIS */
               IF J ¬= 0
               THEN NO(J).LINK = DISP ;
               ELSE NO(CABEC).LINK = DISP ;
               DISP = CABEC ;
               CABEC = 0 ;
             END ;
        ELSE IF J ¬= 0 & NO(J).PESO < MAXDISCO
             THEN DO ; /* JOGA NO J PARA DISCO */
                    CALL GRAVA(J) ;
                    NO(J).LINK = DISP ;
                    DISP = J ;
                    NO(CABEC).LINK = 0 ;
                  END ;
        /* COLOCA NOS NA LISTA POR ORDEM DE PESO */
        DO WHILE(CABEC ¬= 0) ;
          J = ATU ;
          N = NO(CABEC).PESO ;
          DO WHILE(J ¬= 0 & N <= NO(J).PESO) ;
            K = J ;
            J = NO(J).LINK ;
          END ;
          I = CABEC ;
          CABEC = NO(CABEC).LINK ;
          NO(I).LINK = J ;
```

```
          IF J = ATU
          THEN ATU = I ;
          ELSE NO(K).LINK = I ;
        END ;
      END ;
IF ATU = 0 /* FILA NA MEMORIA ESTA VAZIA ? */
THEN DO ;
       IF HEADISC = 0 /* DISCO ESTA VAZIO ? */
       THEN IF PFITAMAX = -1 /* FITA ESTA VAZIA ? */
            THEN FIM = '1'B ;
            ELSE DO ; /* LE FITA */
                   /* TROCA AS FITAS */
                   CLOSE FILE(FITAIN) ;
                   CLOSE FILE(FITAOUT) ;
                   FAUX = FIN ;
                   FIN = FOUT ;
                   FOUT = FAUX ;
                   OPEN FILE(FITAIN) TITLE(FIN) INPUT ;
                   OPEN FILE(FITAOUT) TITLE(FOUT) OUTPUT ;
                   CONTFIN = CONTFOUT ;
                   CONTFOUT = 0 ;
                   CABEC = NFITMAP ;
                   /* PEGA OS NDISCO/2+1 MELHORES (COM RELACAO    */
                   /* AO PESO) NOS DA FITA, C/VALOR DA FUNCAO     */
                   /* OBJETIVO NAO PIOR QUE MAXSOLU.              */
                   DO I = 1 TO CABEC ;
                     PFITA(I) = -1 ;
                     LFITA(I) = I - 1 ;
                   END ;
                   DO I = 1 TO CONTFIN ;
                     READ FILE(FITAIN) INTO(RFITA) ;
                     IF RFITA.XBARRA(0) - MAXSOLU > EPSI
                     THEN DO ;
                            ULTFITA = I ;
                            J = CABEC ;
                            DO WHILE(J¬=0 & RFITA.PESO>PFITA(J)) ;
                              K = J ;
                              J = LFITA(J) ;
                            END ;
                            IF J ¬= CABEC
                            THEN DO ; /* NO MELHOR QUE ALGUM SE- */
                                      /* LECIONADO ANTERIORMENTE.*/
                                   PFITA(CABEC) = RFITA.PESO ;
                                   POSFITA(CABEC) = I ;
                                   IF K ¬= CABEC
                                   THEN DO ; /* ACERTA POSICAO   */
                                             /* NA LISTA         */
                                          LFITA(K) = CABEC ;
                                          K = CABEC ;
                                          CABEC = LFITA(K) ;
                                          LFITA(K) = J ;
                                        END ;
                                 END ;
                          END ;
                   END ;
                   PFITAMAX = PFITA(CABEC) ; /* GUARDA MAIOR PE- */
                                             /* SO EM FITA       */
                   CABEC = LFITA(CABEC) ;
                   /* ASSOCIA MAPA DE FITA C/MAPA DE DISCO */
                   J = CABEC ;
```

```
DO I = NDISCO/2 TO 1 BY -1 ;
  IF PFITA(J) ¬= -1
  THEN DO ;
         PFITA(J) = I ;
         J = LFITA(J) ;
       END ;
  ELSE DO ;
         CABEC = LFITA(J) ;
         J = CABEC ;
       END ;
END ;
IF CABEC = 0 /* TEM ALGUM NO APROVEITAVEL NA   */
             /* FITA ?                          */
THEN FIM = '1'B ; /* NENHUM NO APROVEITAVEL */
ELSE DO ; /* COPIA MELHORES NOS DE FITA PARA   */
          /* DISCO.                            */
       J = PFITA(CABEC) ;
       DO I = 1 TO J-1 , J+1 TO NDISCO-1 ;
         LDISCO(I) = I + 1 ;
       END ;
       HEADISC = 1 ;
       DISCOISP = J + 1 ;
       LDISCO(J) = 0 ;
       LDISCO(NDISCO) = 0 ;
       /* ORDENA FITAMAP POR POSFITA */
       J = LFITA(CABEC) ;
       LFITA(CABEC) = 0 ;
       DO WHILE(J ¬= 0) ;
         K = J ;
         J = LFITA(J) ;
         IF POSFITA(K) < POSFITA(CABEC)
         THEN DO ;
                LFITA(K) = CABEC ;
                CABEC = K ;
              END ;
         ELSE DO ;
                M = CABEC ;
                DO WHILE(LFITA(M) ¬= 0 &
                         POSFITA(K) >
                         POSFITA(LFITA(M))) ;
                  M = LFITA(M) ;
                END ;
                LFITA(K) = LFITA(M) ;
                LFITA(M) = K ;
              END ;
       END ;
       CLOSE FILE(FITAIN) ;
       OPEN FILE(FITAIN) INPUT TITLE(FIN) ;
       /* COPIA NOS SELECIONADOS DA FITA PARA */
       /* DISCO                               */
       I = 0 ;
       DO WHILE(CABEC ¬= 0) ;
         DO I = I+1 TO POSFITA(CABEC)-1 ;
           READ FILE(FITAIN) INTO(RFITA) ;
           IF RFITA.XBARRA(0) - MAXSOLU > EPSI
           THEN DO ; /* GRAVA NA OUTRA FITA */
                  CONTFOUT = CONTFOUT + 1 ;
                  WRITE FILE(FITAOUT)
                        FROM(RFITA) ;
                END ;
```

```
                                    END ;
                                    READ FILE(FITAIN) INTO(RFITA) ;
                                    RDISCO = RFITA , BY NAME ;
                                    J = PFITA(CABEC) ;
                                    /* GRAVA NO DISCO */
                                    CHAVE = J - 1 ;
                                    WRITE FILE(DISCO) FROM(RDISCO)
                                                      KEYFROM(CHAVE) ;
                                    /*COMPLETA POSICAO J DO MAPA DE DISCO*/
                                    PDISCO(J) = RFITA.PESO ;
                                    XBARDISC(J) = RDISCO.XBARRA(0) ;
                                    CABEC = LFITA(CABEC) ;
                                  END ;
                                  DO I = I+1 TO ULTFITA ; /* COPIA RESTO */
                                                          /* DA FITA     */
                                    READ FILE(FITAIN) INTO(RFITA) ;
                                    CONTFOUT = CONTFOUT + 1 ;
                                    WRITE FILE(FITAOUT) FROM(RFITA) ;
                                  END ;
                                END ;
                      END ;
          IF ¬FIM
          THEN DO ; /* LE DISCO */
                 CHAVE = HEADISC - 1 ;
                 READ FILE(DISCO) INTO(RDISCO) KEY(CHAVE) ;
                 /* TIRA REFERENCIA A ESTE NO DO MAPA DE DISCO */
                 ATU = HEADISC ;
                 HEADISC = LDISCO(HEADISC) ;
                 LDISCO(ATU) = DISCDISP ;
                 DISCDISP = ATU ;
                 /* PEGA NO NA LISTA DOS DISPONIVEIS */
                 ATU = DISP ;
                 DISP = NO(DISP).LINK ;
                 NO(ATU).LINK = 0 ;
                 /* REFAZ NO A PARTIR DO DISCO */
                 DO J = 1 TO NVAR ;
                   NO(ATU).FLAG(J) = RDISCO.FLAG(J) ;
                   NO(ATU).LIMITE(J) = RDISCO.LIMITE(J) ;
                 END ;
                 DO I = 1 TO NBASE ;
                   NO(ATU).BASE(I) = RDISCO.BASE(I) ;
                 END ;
                 IF TRACE(1) | TRACE(3)
                 THEN PUT SKIP EDIT('REFAZ NO A PARTIR DO DISCO')(A) ;
                 CALL RECINV(NO(ATU)) ;
                 NO(ATU).PESO = PDISCO(DISCDISP) ;
                 IF TRACE(1) | TRACE(3)
                 THEN CALL IMPRIME(NO(ATU) , ATU) ;
               END ;
  END ;
END ;
IF MSGFINAL
THEN PUT SKIP LIST('O PROBLEMA NAO TEM SOLUCAO INTEIRA') ;
ELSE DO WHILE(CABECSOLU ¬= 0) ;
       DO I = 1 TO NBASE ;
         NO(1).BASE(I) = GUARDSOLU(CABECSOLU).BASE(I) ;
         NO(1).QUADRO(I,NBASE+1) = GUARDSOLU(CABECSOLU).XBARRA(I) ;
       END ;
       NO(1).QUADRO(0,NBASE+1) = GUARDSOLU(CABECSOLU).XBARRA(0) ;
       DO I = 1 TO NVAR ;
```

```
            NO(1).FLAG(I) = GUARDSOLU(CABECSOLU).FLAG(I) ;
            NO(1).LIMITE(I) = GUARDSOLU(CABECSOLU).LIMITE(I) ;
          END ;
          IF INVSOLU = 1
          THEN DO ; /* IMPRIME INVERSA */
                 CALL RECINV(NO(1)) ;
                 CALL IMPRIME(NO(1) , 0) ;
               END ;
          PUT SKIP EDIT('SOLUCAO INTEIRA : ')(COL(24) , A) ;
          CALL IMPSOLU(NO(1)) ;
          CABECSOLU = GUARDSOLU(CABECSOLU).LINK ;
        END ;
OTIMIZA : PROC(NO) ;
  DCL 1 NO ,
        2 QUADRO(*,*)              BIN FLOAT(53) ,
        2 BASE(*)                  BIN FIXED ,
        2 PESO                     BIN FIXED ,
        2 LINK                     BIN FIXED ,
        2 FLAG(*)                  BIN FIXED ,
        2 LIMITE(*)                BIN FLOAT(53) ;
  DO WHILE('1'B) ;
    /* ESCOLHA DA VARIAVEL QUE SAI DA BASE */
    DO SAI = 1 TO NBASE WHILE(QUADRO(SAI,NBASE+1) >= 0.) ;
    END ;
    IF SAI > NBASE
    THEN DO ;
           IF TRACE(1) | TRACE(3)
           THEN DO ;
                  IF TRACE(1)
                  THEN CALL IMPRIME(NO , 0) ;
                  PUT SKIP(2) EDIT('SOLUCAO OTIMA PARA O PROBLEMA ' ,
                                   FASECHAR , ' :')((3)A) ;
                  CALL IMPSOLU(NO) ;
                END ;
           ABRE = '0'B ;
           DO I = 1 TO NINT WHILE(¬ABRE) ;
             INT = INTEIRAS(I) ;
             DO SAI = 1 TO NBASE WHILE(INT ¬= BASE(SAI)) ;
             END ;
             IF SAI <= NBASE & ABS(QUADRO(SAI,NBASE+1)
                - FIXED(QUADRO(SAI,NBASE+1) + EPSI)) > EPSI
             THEN ABRE = '1'B ;
           END ;
           IF ABRE
           THEN OTIMO = 1 ;
           ELSE DO ;
                  OTIMO = 2 ;
                  MSGFINAL = '0'B ;
                END ;
           IF TRACE(15) & QUADRO(0,NBASE+1) < 4000.
           THEN DO ;
                  TRACE(15) = '0'B ;
                  TRACE(1) = '1'B ;
                END ;
           RETURN ;
         END ;
IF TRACE(16)
THEN PUT SKIP DATA(SAI) ;
    /* ESCOLHA DA VARIAVEL QUE ENTRA NA BASE */
    MENOR = MAX ;
```

```
      ENTRA = 0 ;
      DO J = 1 TO NVAR ;
        /* VERIFICA SE J ESTA NA BASE */
        DO K = 1 TO NBASE WHILE(BASE(K) ¬= J) ;
        END ;
        IF K > NBASE
        THEN DO ; /* J NAO ESTA NA BASE */
               AUXSAI = 0. ;
               DO K = 0 TO NBASE ;
                 AUXSAI = AUXSAI + QUADRO(SAI,K) * MATRIZ(K,J) ;
               END ;
IF TRACE(16)
THEN PUT SKIP DATA (J,AUXSAI) ;
               IF FLAG(J) = 2
               THEN AUXSAI = - AUXSAI ;
IF TRACE(16)
THEN PUT SKIP DATA (J,AUXSAI) ;
               IF AUXSAI < 0.
               THEN DO ;
                      AUX0 = 0. ;
                      DO K = 0 TO NBASE ;
                        AUX0 = AUX0 + QUADRO(0,K) * MATRIZ(K,J) ;
                      END ;
                      AUX = ABS(AUX0 / AUXSAI) ;
                      IF AUX < MENOR
                      THEN DO ;
                             ENTRA = J ;
                             MENOR = AUX ;
                             IF FLAG(J) = 2
                             THEN AUX0 = - AUX0 ;
                             QUADRO(0,NBASE+2) = AUX0 ;
                             QUADRO(SAI,NBASE+2) = AUXSAI ;
                           END ;
                    END ;
             END ;
      END ;
      IF ENTRA = 0
      THEN DO ;
             IF TRACE(1) | TRACE(3)
             THEN DO ;
                    IF TRACE(1)
                    THEN CALL IMPRIME(NO , 0) ;
                    PUT EDIT('O PROBLEMA NAO TEM SOLUCAO')(COL(1) , A) ;
                  END ;
             OTIMO = 0 ;
             RETURN ;
           END ;
      ELSE DO ;
             /* COMPLETA COLUNA PIVOT */
             DO I = 1 TO SAI-1 , SAI+1 TO NBASE ;
               AUX = 0. ;
               DO J = 0 TO NBASE ;
                 AUX = AUX + QUADRO(I,J) * MATRIZ(J,ENTRA) ;
               END ;
               IF FLAG(ENTRA) = 2
               THEN AUX = - AUX ;
               QUADRO(I,NBASE+2) = AUX ;
             END ;
             IF TRACE(1) | TRACE(3)
             THEN DO ;
```

```
                    IMPBASE = ENTRA ;
                    PUT EDIT('VARIAVEL X' , IMPBASE , ' ENTRA NA BASE,' ,
                             ' SAI VARIAVEL X')(COL(1) , (4) A) ;
                    IMPBASE = BASE(SAI) ;
                    PUT EDIT(IMPBASE)(A) ;
                  END ;
              BASE(SAI) = ENTRA ;
              CALL PIVOTA(NO,SAI) ;
              IF TRACE(3)
              THEN CALL IMPRIME(NO , 0) ;
IF TRACE(16)
THEN PUT SKIP LIST('LIMITE=' , LIMITE(ENTRA)) ;
              IF FLAG(ENTRA) = 2 & QUADRO(SAI,NBASE+1) > LIMITE(ENTRA)
              THEN DO ;
                    QUADRO(SAI,NBASE+1) = QUADRO(SAI,NBASE+1)
                                          - LIMITE(ENTRA) ;
                    FLAG(ENTRA) = 0 ;
                    DO J = 0 TO NBASE + 1 ;
                      QUADRO(SAI,J) = - QUADRO(SAI,J) ;
                    END ;
                    IF TRACE(1) | TRACE(3)
                    THEN DO ;
                          IMPBASE = ENTRA ;
                          PUT EDIT('VARIAVEL X' , IMPBASE , ' ULTRAPA' ,
                                   'SSOU LIMITE. FOI FEITA SUBSTITUIC' ,
                                   'AO DE VARIAVEL')(COL(1) , (5) A) ;
                          IF TRACE(3)
                          THEN CALL IMPRIME(NO , 0) ;
                        END ;
                  END ;
            END ;
  END ;
END OTIMIZA ;
GRAVA : PROC(PT1) ;
  DCL PT1                 BIN FIXED ;
  DCL PT                  BIN FIXED ;
  DCL (I,J,K,L,M)         BIN FIXED ;
  DCL IMP                 BIT(1) ;
  DCL IMPBASE             PIC '999' ;
  DCL (INT , N)           BIN FIXED ;
  PT = PT1 ;
  DO WHILE(PT ¬= 0) ;
    IF NO(PT).PESO < PFITAMAX /* EXISTE NO NA FITA C/PESO MAIOR ? */
    THEN DO WHILE(PT ¬= 0) ; /* JOGA NOS PARA FITA */
            IF CONTFOUT >= NFITA
            THEN DO ; /* NUMERO MAXIMO DE NOS NA FITA FOI ULTRAPASSADO*/
                  PUT EDIT('NUMERO MAXIMO DE NOS NA FITA FOI ULTRAPA' ,
                           'SSADO.' , ' O NO DESCRITO A SEGUIR FOI ' ,
                           'ELIMINADO DO PROBLEMA')
                          (SKIP , A , A , SKIP , A , A) ;
                  CALL IMPSOLU(NO(PT)) ;
                END ;
            ELSE DO ;
                  RFITA.XBARRA(J) = NO(PT).QUADRO(0,NBASE+1) ;
                  DO K = 1 TO NBASE ;
                    RFITA.BASE(K) = NO(PT).BASE(K) ;
                    RFITA.XBARRA(K) = NO(PT).QUADRO(K,NBASE+1) ;
                  END ;
                  DO K = 1 TO NVAR ;
                    RFITA.FLAG(K) = NO(PT).FLAG(K) ;
```

```
                RFITA.LIMITE(K) = NO(PT).LIMITE(K) ;
              END ;
              RFITA.PESO = NO(PT).PESO ;
              CONTFOUT = CONTFOUT + 1 ;
              WRITE FILE(FITAOUT) FROM(RFITA) ;
            END ;
          PT = NO(PT).LINK ;
        END ;
      ELSE DO ; /* NAO EXISTE NA FITA NO COM PESO > DO NO(PT) */
        IF DISCDISP ¬= 0
        THEN DO ; /* TEM ESPACO NO DISCO */
              /* JOGA NO(PT) PARA DISCO */
              K = DISCDISP ;
              DISCDISP = LDISCO(K) ;
              RDISCO.XBARRA(0) = NO(PT).QUADRO(0,NBASE+1) ;
              DO I = 1 TO NBASE ;
                RDISCO.BASE(I) = NO(PT).BASE(I) ;
                RDISCO.XBARRA(I) = NO(PT).QUADRO(I,NBASE+1) ;
              END ;
              DO I = 1 TO NVAR ;
                RDISCO.FLAG(I) = NO(PT).FLAG(I) ;
                RDISCO.LIMITE(I) = NO(PT).LIMITE(I) ;
              END ;
              /* GRAVA NO DISCO */
              CHAVE = K - 1 ;
              WRITE FILE(DISCO) FROM(RDISCO) KEYFROM(CHAVE) ;
              /* ACERTA MAPA DE DISCO */
              PDISCO(K) = NO(PT).PESO ;
              XBARDISC(K) = NO(PT).QUADRO(0,NBASE+1) ;
              I = HEADISC ;
              DO WHILE(I ¬= 0 & PDISCO(I) >= PDISCO(K)) ;
                J = I ; I = LDISCO(I) ;
              END ;
              LDISCO(K) = I ;
              IF I = HEADISC
              THEN HEADISC = K ;
              ELSE LDISCO(J) = K ;
              PT = NO(PT).LINK ;
            END ;
        ELSE DO ; /* NAO TEM ESPACO NO DISCO */
              /* PROCURA POSICAO P/NO(PT) NO MAPA DE DISCO */
              I = HEADISC ;
              DO WHILE(I ¬= 0 & PDISCO(I) >= NO(PT).PESO) ;
                J = I ; I = LDISCO(I) ;
              END ;
              IF I = 0 /* PESO NO(PT) <= MENOR PESO DO DISCO ? */
              THEN DO WHILE(PT ¬= 0) ; /* JOGA NOS PARA FITA */
                    IF CONTFOUT >= NFITA
                    THEN DO ; /* NUMERO MAXIMO DE NOS NA FITA ***/
                              /* FOI ULTRAPASSADO                */
                          PUT EDIT('NUMERO MAXIMO DE NOS NA FI' ,
                                   'TA FOI ULTRAPASSADO.' ,
                                   'O NO DESCRITO A SEGUIR FOI' ,
                                   ' ELIMINADO DO PROBLEMA')
                                  (SKIP , A , A , SKIP , A , A) ;
                          CALL IMPSOLU(NO(PT)) ;
                        END ;
                    ELSE DO ;
                          RFITA.XBARRA(0) =
                                NO(PT).QUADRO(0,NBASE+1) ;
```

```
                    DO K = 1 TO NBASE ;
                      RFITA.BASE(K) = NO(PT).BASE(K) ;
                      RFITA.XBARRA(K) =
                                NO(PT).QUADRO(K,NBASE+1) ;
                    END ;
                    DO K = 1 TO NVAR ;
                      RFITA.FLAG(K) = NO(PT).FLAG(K) ;
                      RFITA.LIMITE(K) = NO(PT).LIMITE(K) ;
                    END ;
                    RFITA.PESO = NO(PT).PESO ;
                    CONTFOUT = CONTFOUT + 1 ;
                    /* GRAVA FITA */
                    WRITE FILE(FITAOUT) FROM(RFITA) ;
                  END ;
            PT = NO(PT).LINK ;
          END ;
ELSE DO ; /* HA NO DISCO NO C/PESO < DO NO(PT) */
          /* JOGA ULTIMO NO DA LISTA P/FITA */
          /* CAMINHA NA LISTA P/PEGAR ULTIMO NO */
          L = I ;
          DO WHILE(LDISCO(L) ¬= 0) ;
            M = L ; L = LDISCO(L) ;
          END ;
          /* PEGA NO L DO DISCO E GRAVA NA FITA */
          CHAVE = L - 1 ;
          READ FILE(DISCO) INTO(RDISCO) KEY(CHAVE) ;
          IF CONTFOUT >= NFITA
          THEN DO ; /* NUMERO MAXIMO DE NOS NA FITA    */
                    /* FOI ULTRAPASSADO                */
                 PUT EDIT('NUMERO MAXIMO DE NOS NA FI' ,
                          'TA FOI ULTRAPASSADO.' ,
                          'O NO DESCRITO A SEGUIR SER' ,
                          'A ELIMINADO DO PROBLEMA')
                         (SKIP , A , A , SKIP , A , A) ;
                 PUT EDIT('X000 = ')(SKIP,COL(42),A) ;
                 IF MINMAX = 1
                 THEN PUT EDIT(-RDISCO.XBARRA(0))
                              (F(NCOLIMP,NDECOUT)) ;
                 ELSE PUT EDIT(RDISCO.XBARRA(0))
                              (F(NCOLIMP,NDECOUT)) ;
                 DO K = 1 TO NBASE ;
                   IMPBASE = RDISCO.BASE(K) ;
                   PUT EDIT('X' , IMPBASE , ' = ' ,
                            RDISCO.BASE(K))(COL(42) ,
                            (3)A , F(NCOLIMP,NDECOUT)) ;
                   IF RDISCO.FLAG(IMPBASE) ¬= 0
                   THEN DO ;
                          PUT EDIT(' (COMPLEMENTAR) ')
                                   (A) ;
                          IF RDISCO.FLAG(IMPBASE) = 1
                          THEN PUT EDIT(RDISCO.XBARRA(K)
                               + RDISCO.LIMITE(IMPBASE))
                               (F(NCOLIMP,NDECOUT)) ;
                          ELSE PUT EDIT(RDISCO.XBARRA(K)
                               - RDISCO.LIMITE(IMPBASE))
                               (F(NCOLIMP,NDECOUT)) ;
                        END ;
                 END ;
                 IMP = '0'B ;
                 DO K = 1 TO NINT ;
```

```
                                    INT = INTEIRAS(K) ;
                                    IF RDISCO.FLAG(INT) ¬= 0
                                    THEN DO ;
                                           DO N = 1 TO NBASE
                                              WHILE(INT¬=RDISCO.BASE(N));
                                           END ;
                                           IF N > NBASE
                                           THEN DO ;
                                                  IF ¬IMP
                                                  THEN DO ;
                                                         IMP = '1'B ;

PUT SKIP EDIT('VARIAVEIS COMPLEMENTARES FORA DA BASE :')(A) ;

                                                       END ;
                                                  IMPBASE = INT ;

PUT EDIT('X',IMPBASE,' = ',RDISCO.LIMITE(INT))(COL(42),(3)A,
                                                F(NCOLIMP,NDECOUT)) ;

                                             END ;
                                      END ;
                                  END ;
                               END ;
                            ELSE DO ;
                                  RFITA = RDISCO , BY NAME ;
                                  RFITA.PESO = PDISCO(L) ;
                                  CONTFOUT = CONTFOUT + 1 ;
                                  /* GRAVA FITA */
                                  WRITE FILE(FITAOUT) FROM(RFITA) ;
                                  /* ATUALIZA PFITAMAX */
                                  IF RFITA.PESO > PFITAMAX
                                  THEN PFITAMAX = RFITA.PESO ;
                               END ;
                            /* GRAVA NO(PT) NA POSICAO L DO DISCO */
                            RDISCO.XBARRA(0) = NO(PT).QUADRO(0,NBASE+1) ;
                            DO K = 1 TO NBASE ;
                              RDISCO.BASE(K) = NO(PT).BASE(K) ;
                              RDISCO.XBARRA(K) = NO(PT).QUADRO(K,NBASE+1);
                            END ;
                            DO K = 1 TO NVAR ;
                              RDISCO.FLAG(K) = NO(PT).FLAG(K) ;
                              RDISCO.LIMITE(K) = NO(PT).LIMITE(K) ;
                            END ;
                            /* GRAVA NA POSICAO L-1 DO DISCO */
                            CHAVE = L - 1 ;
                            WRITE FILE(DISCO) FROM(RDISCO) KEYFROM(CHAVE);
                            /* ACERTA MAPA DE DISCO */
                            PDISCO(L) = NO(PT).PESO ;
                            XBARDISC(L) = RDISCO.XBARRA(0) ;
                            IF L ¬= 1
                            THEN DO ; /* ESTA FORA DE POSICAO */
                                   LDISCO(M) = 0 ;
                                   LDISCO(L) = I ;
                                   IF I = HEADISC
                                   THEN HEADISC = L ;
                                   ELSE LDISCO(J) = L ;
                                 END ;
                            PT = NO(PT).LINK ;
                          END ;
```

```
                END ;
          END ;
  END ;
END GRAVA ;
IMPSOLU : PROC(NO) ;
  DCL 1 NO ,
        2 QUADRO(*,*)    BIN FLOAT(53) ,
        2 BASE(*)        BIN FIXED ,
        2 PESO           BIN FIXED ,
        2 LINK           BIN FIXED ,
        2 FLAG(*)        BIN FIXED ,
        2 LIMITE(*)      BIN FLOAT(53) ;
  DCL IMPBASE         PIC '999' ;
  DCL (I,J,INT)       BIN FIXED ;
  DCL IMP             BIT(1) INIT('0'B) ;
  DCL AUX             BIN FLOAT(53) ;
  AUX = QUADRO(0,NBASE+1) ;
  IF MINMAX = 1
  THEN AUX = -AUX ;
  PUT SKIP EDIT('X000 = ' , AUX)(COL(42) , A , F(NCOLIMP,NDECOUT)) ;
  DO I = 1 TO NBASE ;
    IMPBASE = BASE(I) ;
    PUT EDIT('X' , IMPBASE , ' = ' , QUADRO(I , NBASE+1))
          (COL(42) , (3) A , F(NCOLIMP,NDECOUT)) ;
    IF FLAG(IMPBASE) ¬= 0
    THEN DO ;
           PUT EDIT(' (COMPLEMENTAR) ')(A) ;
           IF FLAG(IMPBASE) = 1
           THEN AUX = QUADRO(I,NBASE+1) + LIMITE(IMPBASE) ;
           ELSE AUX = LIMITE(IMPBASE) - QUADRO(I,NBASE+1) ;
           IF ABS(AUX) < EPSI
           THEN AUX = 0. ;
           PUT EDIT(AUX)(F(NCOLIMP,NDECOUT)) ;
         END ;
  END ;
  DO I = 1 TO NINT ;
    INT = INTEIRAS(I) ;
    IF FLAG(INT) ¬= 0
    THEN DO ;
           DO J = 1 TO NBASE WHILE(INT ¬= BASE(J)) ;
           END ;
           IF J > NBASE
           THEN DO ;
                  IF ¬ IMP
                  THEN DO ;
                         IMP = '1'B ;
                         PUT SKIP EDIT('VARIAVEIS COMPLEMENTARES FORA',
                                       ' DA BASE  : ')(A , A) ;
                       END ;
                  IMPBASE = INT ;
                  PUT EDIT('X' , IMPBASE , ' = ' , LIMITE(INT))
                        (COL(42) , (3) A , F(NCOLIMP,NDECOUT)) ;
                END ;
         END ;
  END ;
END IMPSOLU ;
PIVOTA : PROC(NO,LPIVOT) ;
  DCL 1 NO ,
        2 QUADRO(*,*)      BIN FLOAT(53) ,
        2 BASE(*)          BIN FIXED ,
```

```
        2 PESO                BIN FIXED ,
        2 LINK                BIN FIXED ,
        2 FLAG(*)             BIN FIXED ,
        2 LIMITE(*)           BIN FLOAT(53) ;
  DCL LPIVOT                  BIN FIXED ;
  DCL CPIVOT                  BIN FIXED ;
  DCL PIVOT                   BIN FLOAT(53) ;
  DCL I1                      BIN FIXED ;
  DCL J1                      BIN FIXED ;
  CPIVOT = NBASE + 2 ;
  PIVOT = QUADRO(LPIVOT,CPIVOT) ;
  /* PROCESSA DEMAIS LINHAS */
  DO I1 = 0 TO LPIVOT-1 , LPIVOT+1 TO NBASE ;
    DO J1 = 0 TO CPIVOT-1 ;
      QUADRO(I1,J1) = QUADRO(I1,J1) - QUADRO(LPIVOT,J1) *
                      (QUADRO(I1,CPIVOT) / PIVOT) ;
      IF ABS(QUADRO(I1,J1)) < EPSI
      THEN QUADRO(I1,J1) = 0. ;
    END ;
    QUADRO(I1,CPIVOT) = 0. ;
  END ;
  /* PROCESSA LINHA DO PIVOT */
  DO J1 = 0 TO CPIVOT ;
    IF QUADRO(LPIVOT,J1) ¬= 0.
    THEN DO ;
           QUADRO(LPIVOT,J1) = QUADRO(LPIVOT,J1) / PIVOT ;
           IF ABS(QUADRO(LPIVOT,J1)) < EPSI
           THEN QUADRO(LPIVOT,J1) = 0. ;
         END ;
  END ;
  /* CONFERE VALOR CALCULADO DA FUNCAO OBJETIVO */
  PIVOT = 0. ;
  IF FASE = 2
  THEN DO I1 = 1 TO NVAR ;
         IF FLAG(I1) ¬= 0 & LIMITE(I1) ¬= 0. & MATRIZ(0,I1) ¬= 0.
         THEN PIVOT = PIVOT - LIMITE(I1) * MATRIZ(0,I1) ;
       END ;
  DO I1 = 1 TO NBASE ;
    IF FASE = 2
    THEN DO ;
           IF MATRIZ(0,BASE(I1)) ¬= 0. & QUADRO(I1,NBASE+1) ¬= 0.
           THEN IF FLAG(BASE(I1)) = 2 /* CORTE A DIREITA ? */
                THEN PIVOT = PIVOT + QUADRO(I1,NBASE+1) *
                             MATRIZ(0,BASE(I1)) ;
                ELSE PIVOT = PIVOT - QUADRO(I1,NBASE+1) *
                             MATRIZ(0,BASE(I1)) ;
         END ;
    ELSE IF BASE(I1) > NVAR
         THEN PIVOT = PIVOT - QUADRO(I1,NBASE+1) ;
  END ;
  IF ABS(QUADRO(0,NBASE+1) - PIVOT) > EPSI
  THEN CALL RECINV(NO) ; /* RECALCULA A INVERSA */
END PIVOTA ;
GUARDA : PROC(NO) RETURNS(BIT(1)) ;
  /*****************************************************************/
  /* ESTA FUNCAO E CHAMADA PARA GUARDAR UMA SOLUCAO INTEIRA. E CHAMA-*/
  /* DA SE AS SEGUINTES CONDICOES FOREM SATISFEITAS :               */
  /*    - O NUMERO DE SOLUCOES DESEJADAS (NUMSOLU) E > 0 E < 11     */
  /*    - O NUMERO DE SOLUCOES JA ENCONTRADAS (E GUARDADAS) E MENOR */
  /*      QUE O NUMERO DE SOLUCOES DESEJADAS (NUMSOLU) OU O VALOR DES-*/
```

```
/*      TA SOLUCAO E MAIOR QUE O VALOR DA PIOR SOLUCAO JA ENCONTRA- */
/*      DA (E GUARDADA - MAXSOLU)                                   */
/* RETORNA ('1'B) SE MAXSOLU FOR ALTERADO (PORTANTO DEVE SER CHAMA-*/
/* DA A PROCEDURE ELIMINA PARA ELIMINAR OS NOS COM VALOR MENOR QUE */
/* ESTE NOVO LIMITE - MAXSOLU) E ('0'B) EM CASO CONTRARIO          */
/*******************************************************************/
DCL 1 NO ,
      2 QUADRO(*,*)              BIN FLOAT(53) ,
      2 BASE(*)                  BIN FIXED(15,0) ,
      2 PESO                     BIN FIXED(15,0) ,
      2 LINK                     BIN FIXED(15,0) ,
      2 FLAG(*)                  BIN FIXED(15,0) ,
      2 LIMITE(*)                BIN FLOAT(53) ;
DCL RETCODE                   BIT(1) ;
DCL (I , J , K , L)           BIN FIXED(15,0) ;
DCL SOLUCAO                   BIN FLOAT(53) ;
IF DISPSOLU ¬= 0
THEN DO ;
       K = DISPSOLU ;
       DISPSOLU = GUARDSOLU(K).LINK ;
       L = K ;
     END ;
ELSE DO ;
       K = CABECSOLU ;
       L = K ;
       DO WHILE(GUARDSOLU(K).LINK ¬= 0) ;
         L = K ;
         K = GUARDSOLU(L).LINK ;
       END ;
       IF L = K
       THEN CABECSOLU = 0 ;
       ELSE GUARDSOLU(L).LINK = 0 ;
     END ;
SOLUCAO = QUADRO(0,NBASE+1) ;
IF CABECSOLU = 0 | SOLUCAO > GUARDSOLU(CABECSOLU).XBARRA(0)
THEN DO ;
       GUARDSOLU(K).LINK = CABECSOLU ;
       CABECSOLU = K ;
     END ;
ELSE DO ;
       I = CABECSOLU ;
       DO WHILE(I ¬= 0 & GUARDSOLU(I).XBARRA(0) >= SOLUCAO) ;
         J = I ;
         I = GUARDSOLU(J).LINK ;
       END ;
       GUARDSOLU(K).LINK = I ;
       GUARDSOLU(J).LINK = K ;
     END ;
/* GUARDA SOLUCAO */
DO I = 1 TO NBASE ;
  GUARDSOLU(K).BASE(I) = NO.BASE(I) ;
  GUARDSOLU(K).XBARRA(I) = NO.QUADRO(I,NBASE+1) ;
END ;
GUARDSOLU(K).XBARRA(0) = SOLUCAO ;
DO I = 1 TO NVAR ;
  GUARDSOLU(K).FLAG(I) = NO.FLAG(I) ;
  GUARDSOLU(K).LIMITE(I) = NO.LIMITE(I) ;
END ;
RETCODE = '0'B ;
IF DISPSOLU = 0
```

```
  THEN DO ;
         DO WHILE(GUARDSOLU(L).LINK ¬= 0) ;
           L = GUARDSOLU(L).LINK ;
         END ;
         IF GUARDSOLU(L).XBARRA(0) - MAXSOLU > EPSI
         THEN DO ;
                MAXSOLU = GUARDSOLU(L).XBARRA(0) ;
                RETCODE = '1'B ;
              END ;
       END ;
  RETURN(RETCODE) ;
END GUARDA ;
ELIMINA : PROC ;
  DCL PT                  BIN FIXED ;
  DCL PTI                 BIN FIXED ;
  /* ELIMINA NOS NA MEMORIA COM SOLUCAO PIOR QUE MAXSOLU */
  PTI = ATU ;
  PT = PTI ;
  DO WHILE(PT ¬= 0) ;
    IF NO(PT).QUADRO(0,NBASE+1) - MAXSOLU > EPSI
    THEN DO ; /* MANTEM ESTE NO */
           PTI = PT ;
           PT = NO(PT).LINK ;
         END ;
    ELSE DO ; /* ELIMINA ESTE NO */
           IF PT ¬= ATU
           THEN DO ;
                  NO(PTI).LINK = NO(PT).LINK ;
                  /* DEVOLVE ESTE NO P/LISTA DE NOS DISPONIVEIS */
                  NO(PT).LINK = DISP ;
                  DISP = PT ;
                  PT = NO(PTI).LINK ;
                END ;
           ELSE DO ;
                  ATU = NO(PT).LINK ;
                  PTI = ATU ;
                  /* DEVOLVE ESTE NO P/LISTA DE NOS DISPONIVEIS */
                  NO(PT).LINK = DISP ;
                  DISP = PT ;
                  PT = PTI ;
                END ;
         END ;
  END ;
  /* ELIMINA NOS DO DISCO COM SOLUCAO PIOR QUE MAXSOLU */
  PTI = HEADISC ;
  PT = HEADISC ;
  DO WHILE(PT ¬= 0) ;
    IF XBARDISC(PT) - MAXSOLU > EPSI
    THEN DO ; /* MANTEM ESTE NO */
           PTI = PT ;
           PT = LDISCO(PT) ;
         END ;
    ELSE DO ; /* ELIMINA ESTE NO */
           IF PT ¬= HEADISC
           THEN DO ;
                  LDISCO(PTI) = LDISCO(PT) ;
                  /*DEVOLVE NO (MAPA DE DISCO) P/LISTA DE DISPONIVEIS*/
                  LDISCO(PT) = DISCDISP ;
                  DISCDISP = PT ;
                  PT = LDISCO(PTI) ;
```

```
                END ;
            ELSE DO ;
                   HEADISC = LDISCO(PT) ;
                   PT1 = HEADISC ;
                   /*DEVOLVE NO (MAPA DE DISCO) P/LISTA DE DISPONIVEIS*/
                   LDISCO(PT) = DISCDISP ;
                   DISCDISP = PT ;
                   PT = PT1 ;
                 END ;
          END ;
  END ;
END ELIMINA ;
RECINV : PROC(NO) ;
  DCL 1 NO ,
        2 QUADRO(*,*)              BIN FLOAT(53) ,
        2 BASE(*)                  BIN FIXED ,
        2 PESO                     BIN FIXED ,
        2 LINK                     BIN FIXED ,
        2 FLAG(*)                  BIN FIXED ,
        2 LIMITE(*)                BIN FLOAT(53) ;
  DCL (I , J , N , I1 , J1 , K1 , L1) BIN FIXED ;
  DCL BASEORD(0 : NBASE)           BIN FIXED ;
  DCL ORDENA(NBASE)                BIN FIXED ;
  DCL AUX                          BIN FLOAT(53) ;
  DCL NETA(0:NBASE)                BIN FLOAT(53) ;
  IF TRACE(1) | TRACE(3)
  THEN DO ;
         IF TRACE(3)
         THEN CALL IMPRIME(NO , 0) ;
         PUT SKIP EDIT('RECALCULADA A INVERSA')(A) ;
       END ;
  /* REFAZ QUADRO INICIAL */
  DO I1 = 0 TO NBASE ;
    DO J1 = 1 TO NBASE + 1 ;
      QUADRO(I1,J1) = INVBO(I1,J1) ;
    END ;
  END ;
  /* ORDENA BASE DO QUADRO ATUAL */
  BASEORD(0) = 1 ;
  BASEORD(1) = 0 ;
  DO I1 = 2 TO NBASE ;
    J1 = BASEORD(0) ;
    K1 = 0 ;
    N = BASE(I1) ;
    DO WHILE(J1 ¬= 0 & BASE(J1) < N) ;
      K1 = J1 ;
      J1 = BASEORD(J1) ;
    END ;
    BASEORD(K1) = I1 ;
    BASEORD(I1) = J1 ;
  END ;
  IF TRACE(16)
  THEN CALL IMPRIME(NO , 0) ;
  /* REFAZ A INVERSA */
  I1 = BASEORD(0) ; /* I1 APONTARA VARIAVEL A SER INSERIDA */
  J1 = I1 ;
  K1 = BASINOR(0) ; /* K1 APONTARA VARIAVEL A SER SUBSTITUIDA */
  L1 = K1 ;
  DO WHILE(I1 ¬= 0) ;
    DO WHILE(L1 ¬= 0 & BASE(I1) > BASINIT(L1)) ;
```

```
    L1 = BASINOR(L1) ;
  END ;
  IF L1 ¬= 0 & BASE(I1) = BASINIT(L1)
  THEN DO ; /* VARIAVEL BASE(I1) JA ESTA NA INVERSA */
         ORDENA(I1) = L1 ;
         I1 = BASEORD(I1) ;
         L1 = BASINOR(L1) ;
       END ;
  ELSE DO ; /* INSERE VARIAVEL BASE(I1) NA INVERSA */
         DO WHILE(J1 ¬= 0 & BASINIT(K1) >= BASE(J1)) ;
           IF BASINIT(K1) = BASE(J1)
           THEN K1 = BASINOR(K1) ;
           J1 = BASEORD(J1) ;
         END ;
         /*INSERE VARIAVEL BASE(I1) NO LUGAR DA VAR. BASINIT(K1)*/
         N = BASE(I1) ;
         DO I = 0 TO NBASE ; /* PREPARA VETOR A */
           QUADRO(I,NBASE+2) = MATRIZ(I,N) ;
         END ;
         DO I = 0 TO NBASE ; /* NETA = INV(B) * A */
           AUX = 0. ;
           DO J = 0 TO NBASE ;
             AUX = AUX + QUADRO(I,J) * QUADRO(J,NBASE+2) ;
           END ;
           NETA(I) = AUX ;
         END ;
         AUX = NETA(K1) ;
         DO I = 0 TO K1-1 , K1+1 TO NBASE ;
           NETA(I) = - NETA(I) / AUX ;
         END ;
         NETA(K1) = 1. / AUX ;
         DO I = 0 TO K1-1 , K1+1 TO NBASE ; /* B = E(K1) * INV(B) */
           DO J = 0 TO NBASE ;
             QUADRO(I,J) = QUADRO(I,J) + NETA(I) * QUADRO(K1,J) ;
             IF ABS(QUADRO(I,J)) < EPSI
             THEN QUADRO(I,J) = 0. ;
           END ;
         END ;
         DO J = 0 TO NBASE ;
           QUADRO(K1,J) = NETA(K1) * QUADRO(K1,J) ;
           IF ABS(QUADRO(K1,J)) < EPSI
           THEN QUADRO(K1,J) = 0. ;
         END ;
         ORDENA(I1) = K1 ;
         I1 = BASEORD(I1) ;
         K1 = BASINOR(K1) ;
         IF TRACE(16)
         THEN CALL IMPRIME(NO , 0) ;
       END ;
END ;
/* RECRDENA A INVERSA */
DO I1 = 1 TO NBASE ;
  IF ORDENA(I1) ¬= I1
  THEN DO ;
         DO J1 = 0 TO NBASE ;
           QUADRO(J1,NBASE+2) = QUADRO(I1,J1) ;
         END ;
         K1 = ORDENA(I1) ;
         L1 = I1 ;
         DO WHILE(K1 ¬= I1) ;
```

```
            DO J1 = 0 TO NBASE ; /* MOVE LINHA K1 P/LINHA L1 */
              QUADRO(L1,J1) = QUADRO(K1,J1) ;
            END ;
            ORDENA(L1) = L1 ;
            L1 = K1 ;
            K1 = ORDENA(K1) ;
          END ;
          DO J1 = 0 TO NBASE ;
            QUADRO(L1,J1) = QUADRO(J1,NBASE+2) ;
          END ;
          ORDENA(L1) = L1 ;
        END ;
  END ;
  IF FASE = 2
  THEN DO ;
        DO I1 = 1 TO NBASE ;
          /* TROCA SINAL DAS LINHAS COMPLEMENTARES ESQ */
          IF FLAG(BASE(I1)) = 2 /* CORTE A ESQUERDA ? */
          THEN DO J1 = 1 TO NBASE ;
                 QUADRO(I1,J1) = - QUADRO(I1,J1) ;
               END ;
        END ;
        DO I1 = 0 TO NBASE ; /* RECALCULA VETOR B */
          NETA(I1) = QUADRO(I1,NBASE+1) ;
          DO J1 = 1 TO NVAR ;
            IF FLAG(J1) ¬= 0
            THEN NETA(I1) = NETA(I1) - LIMITE(J1) * MATRIZ(I1,J1) ;
          END ;
        END ;
      END ;
  ELSE DO I1 = 0 TO NBASE ;
        NETA(I1) = QUADRO(I1,NBASE+1) ;
      END ;
  DO I1 = 0 TO NBASE ; /* RECALCULA XBARRA */
    AUX = 0. ;
    DO J1 = 0 TO NBASE ;
      AUX = AUX + QUADRO(I1,J1) * NETA(J1) ;
    END ;
    IF ABS(AUX) < EPSI
    THEN QUADRO(I1,NBASE+1) = 0. ;
    ELSE QUADRO(I1,NBASE+1) = AUX ;
  END ;
END RECINV ;
END ; /* BEGIN */
IMPRIME : PROC(NO , NUM) ;
  DCL 1 NO ,
        2 QUADRO(*,*)     BIN FLOAT(53) ,
        2 BASE(*)         BIN FIXED ,
        2 PESO            BIN FIXED ,
        2 LINK            BIN FIXED ,
        2 FLAG(*)         BIN FIXED ,
        2 LIMITE(*)       BIN FLOAT(53) ;
  DCL NUM               BIN FIXED ;
  DCL NUM1              PIC '999' ;
  DCL (I,K,N4)          BIN FIXED ;
  DCL IMPBASE           PIC '999' ;
  IF MINMAX = 1
  THEN QUADRO(0,NBASE+1) = -QUADRO(0,NBASE+1) ;
  IF NUM = 0
  THEN CONTINUA = ' ' ;
```

```
  ELSE DO ;
         IMPBASE = PESO ;
         NUM1 = NUM ;
         CONTINUA = NUM1 || ' - PESO = ' || IMPBASE ;
       END ;
  N2 = (LTAM - 4) / NCOLIMP ;
  DO K = 0 TO NBASE+2 BY N2 ;
    PUT SKIP(2) EDIT('IMPRESSAO DO QUADRO ' , CONTINUA)(COL(20),A,A) ;
    CONTINUA = 'CONTINUACAO' ;
    IF K = 0
    THEN DO ;
           IMPBASE = 0 ;
           PUT SKIP(2) EDIT('X' , IMPBASE)(COL(4+NCOLIMP-3) , A , A) ;
           N4 = 1 ;
         END ;
    ELSE DO ;
           N4 = K ;
           PUT SKIP(2) EDIT(' ')(COL(4) , A) ;
         END ;
    N3 = K + N2 - 1 ;
    IF N3 > NBASE
    THEN DO ;
           DO I = N4 TO NBASE ;
             IMPBASE = BASE(I) ;
             PUT EDIT('X' , IMPBASE)(X(NCOLIMP-4) , A , A) ;
           END ;
           IF K < NBASE+2
           THEN PUT EDIT('XBARRA')(X(NCOLIMP-6) , A) ;
           IF N3 > NBASE+1
           THEN DO ;
                  N3 = NBASE + 2 ;
                  N2 = N3 - K + 1 ;
                  PUT EDIT('PIVOT')(X(NCOLIMP-5) , A) ;
                END ;
         END ;
    ELSE DO I = N4 TO N3 ;
           IMPBASE = BASE(I) ;
           PUT EDIT('X' , IMPBASE)(X(NCOLIMP-4) , A , A) ;
         END ;
    IMPBASE = 0 ;
    PUT SKIP(2) EDIT('X' , IMPBASE , (QUADRO(0,J) DO J = K TO N3))
                     (A , A , (N2)F(NCOLIMP,NDECOUT)) ;
    DO I = 1 TO NBASE ;
      IMPBASE = BASE(I) ;
      PUT EDIT('X' , IMPBASE , (QUADRO(I,J) DO J = K TO N3)) (COL(1) ,
               A , A , (N2)F(NCOLIMP,NDECOUT)) ;
    END ;
  END ;
  IF MINMAX = 1
  THEN QUADRO(0,NBASE+1) = -QUADRO(0,NBASE+1) ;
END IMPRIME ;
END SIMPLER ;
```

A2 - Função CALCULA original

```
CALCULA : PROC(NO , NBASE , NVAR , INTEIRAS , NINT , EPSI)
                RETURNS(BIN FIXED(15,0)) ;
DCL (NBASE , NVAR , NINT)          BIN FIXED(15,0) ;
DCL 1 NO ,
      2 QUADRO(*,*)                    BIN FLOAT(53) ,
      2 BASE(*)                        BIN FIXED(15,0) ,
      2 PESO                           BIN FIXED(15,0) ,
      2 LINK                           BIN FIXED(15,0) ,
      2 FLAG(*)                        BIN FIXED(15,0) ,
      2 LIMITE(*)                      BIN FLOAT(53) ;
DCL EPSI                           BIN FLOAT(53) ;
DCL INTEIRAS(*)                    BIN FIXED(15,0) ;
DCL PMAX                           STATIC BIN FIXED(15,0)
                                          INIT(0) ;

PMAX = PMAX + 1 ;
RETURN(PMAX) ;
END CALCULA ;
```

A3 - Função DEFINEV original

```
DEFINEV : PROC(NO , NBASE , NVAR , INTEIRAS , NINT , EPSI)
                RETURNS(BIN FIXED(15,0)) ;
DCL (NBASE,NVAR,NINT)                    BIN FIXED ;
DCL 1 NO ,
      2 QUADRO(*,*)                          BIN FLOAT(53) ,
      2 BASE(*)                              BIN FIXED ,
      2 PESO                                 BIN FIXED ,
      2 LINK                                 BIN FIXED ,
      2 FLAG(*)                              BIN FIXED ,
      2 LIMITE(*)                            BIN FLOAT(53) ;
DCL EPSI                                 BIN FLOAT(53) ;
DCL INTEIRAS(*)                          BIN FIXED ;
DCL AUX                                  BIN FLOAT(53) ;
DCL (SAI,I,INT)                          BIN FIXED ;
SAI = NBASE + 1 ;
I = 0 ;
DO WHILE(SAI > NBASE) ;
  I = I + 1 ;
  INT = INTEIRAS(I) ;
  DO SAI = 1 TO NBASE WHILE(INT ¬= BASE(SAI)) ;
  END ;
  IF SAI <= NBASE
  THEN DO ;
         AUX = FIXED(QUADRO(SAI,NBASE+1) + EPSI) ;
         IF ABS(QUADRO(SAI,NBASE+1) - AUX) <= EPSI
         THEN SAI = NBASE + 1 ;
       END ;
END ;
RETURN(SAI) ;
END DEFINEV ;
```

A4 - Função CALCULA alterada

```
CALCULA : PROC(NO , NBASE , NVAR , INTEIRAS , NINT , EPSI)
                RETURNS(BIN FIXED(15,0)) ;
DCL 1 NO ,
      2 QUADRO(*,*)                          BIN FLOAT(53) ,
      2 BASE(*)                              BIN FIXED(15,0) ,
      2 PESO                                 BIN FIXED(15,0) ,
      2 LINK                                 BIN FIXED(15,0) ,
      2 FLAG(*)                              BIN FIXED(15,0) ,
      2 LIMITE(*)                            BIN FLOAT(53) ;
DCL (NBASE,NVAR,INTEIRAS(*),NINT)     BIN FIXED(15,0) ;
DCL (EPSI,AUX)                        BIN FLOAT(53) ;
DCL FIXED                             BUILTIN ;
AUX = QUADRO(0,NBASE+2) ;
IF AUX < 0.
THEN AUX = 0. ;
ELSE IF AUX > 32767.
     THEN AUX = 32767. ;
RETURN(FIXED(AUX)) ;
END CALCULA ;
```

A5 - Função DEFINEV alterada

```
DEFINEV : PROC(NO , NBASE , NVAR , INTEIRAS , NINT , EPSI)
          RETURNS(BIN FIXED(15,0)) ;
DCL 1 NO ,
      2 QUADRO(*,*)                                  BIN FLOAT(53) ,
      2 BASE(*)                                      BIN FIXED(15,0) ,
      2 PESO                                         BIN FIXED(15,0) ,
      2 LINK                                         BIN FIXED(15,0) ,
      2 FLAG(*)                                      BIN FIXED(15,0) ,
      2 LIMITE(*)                                    BIN FLOAT(53) ;
DCL (NBASE , NVAR , INTEIRAS(*) , NINT)  BIN FIXED(15,0) ;
DCL EPSI                                 BIN FLOAT(53) ;
DCL (I , INDMENOR)                       BIN FIXED(15,0) ;
DCL (AUX , MENOR)                        BIN FLOAT(53) ;
DCL (ABS , FIXED)                        BUILTIN ;
MENOR = 1. ;
DO I = 1 TO NBASE WHILE(MENOR ¬= 0.) ;
  AUX=ABS(.5-(QUADRO(I,NBASE+1)-FIXED(QUADRO(I,NBASE+1))));
  IF AUX < MENOR
  THEN DO ;
         MENOR = AUX ;
         INDMENOR = I ;
       END ;
END ;
RETURN(INDMENOR) ;
END DEFINEV ;
```

A6 - Execução do sistema para o problema apresentado no texto

```
                                TIPO DE OTIMIZACAO : MAXIMIZACAO
                      NUMERO DE VARIAVEIS DO PROBLEMA :   6
                          NUMERO DE VARIAVEIS NA BASE :   3
                                INICIAR PELA FASE : 2
                         NUMERO DE VARIAVEIS INTEIRAS :   3
                             NUMERO DE NOS NA MEMORIA :   3
                               NUMERO DE NOS NO DISCO :   6
                                NUMERO DE NOS NA FITA :  15
           NUMERO DE COLUNAS DOS DADOS DE ENTRADA :   3
     NUMERO DE CASAS DECIMAIS DOS DADOS DE ENTRADA :   0
NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS NA BASE :   2
NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS INTEIRAS :   2
  NUMERO DE COLUNAS PARA IMPRESSAO DOS RESULTADOS :   7
NUMERO DE CASAS DECIMAIS PARA IMPRESSAO DOS RESULTADOS :   3
                      TAMANHO DA LINHA PARA IMPRESSAO :  80
                           PRECISAO UTILIZADA (EPSI) : 0.000500000000000
                     VALOR LIMITE PARA AS SOLUCOES :  -9.9999000E+74
                     NUMERO DE SOLUCOES PROCURADAS :   1
                           IMPRESSAO DAS INVERSAS :NAO


                          PROBLEMA INICIAL

   X001    X002    X003    X004    X005    X006  XBARRA
     -3      -4      -1       0       0       0       0
      8       3       4       1       0       0       7
      2       6       1       0       1       0       3
      1       4       5       0       0       1       8

                    IMPRESSAO DO QUADRO 001 - PESO = 000

       X000    X001    X002    X006 XBARRA  PIVOT

X000  1.000   0.238   0.548   0.000  3.310  0.000
X001  0.000   0.143  -0.071   0.000  0.786  1.000
X002  0.000  -0.048   0.190   0.000  0.238  0.000
X006  0.000   0.048  -0.690   1.000  6.262  0.000

SOLUCAO OTIMA PARA O PROBLEMA CONTINUO
                          SOLUCAO INTEIRA :
                                                 X000 =   1.000
                                                 X006 =   3.000
                                                 X005 =   2.000
                                                 X004 =   3.000
VARIAVEIS COMPLEMENTARES FORA DA BASE  : X001 =   0.000
                                                 X002 =   0.000
                                                 X003 =   1.000
```

A7 - Execução do sistema para um problema de 15 variáveis e 10 restrições - 1 solução

```
                                TIPO DE OTIMIZACAO : MAXIMIZACAO
                    NUMERO DE VARIAVEIS DO PROBLEMA :  25
                        NUMERO DE VARIAVEIS NA BASE :  10
                                INICIAR PELA FASE : 2
                       NUMERO DE VARIAVEIS INTEIRAS :  25
                           NUMERO DE NOS NA MEMORIA :   4
                             NUMERO DE NOS NO DISCO :  35
                              NUMERO DE NOS NA FITA :  15
                NUMERO DE COLUNAS DOS DADOS DE ENTRADA :   6
          NUMERO DE CASAS DECIMAIS DOS DADOS DE ENTRADA :   0
     NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS NA BASE :   3
   NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS INTEIRAS :   2
        NUMERO DE COLUNAS PARA IMPRESSAO DOS RESULTADOS :   9
NUMERO DE CASAS DECIMAIS PARA IMPRESSAO DOS RESULTADOS :   5
                    TAMANHO DA LINHA PARA IMPRESSAO :  72
                          PRECISAO UTILIZADA (EPSI) : 0.0005000000000 (
                      VALOR LIMITE PARA AS SOLUCOES :  -9.9999000E+74
                      NUMERO DE SOLUCOES PROCURADAS :   1
                            IMPRESSAO DAS INVERSAS :NAO
```

PROBLEMA INICIAL

X001	X002	X003	X004	X005	X006	X007	X008	X009	X010
-100	-220	-90	-400	-300	-400	-205	-120	-160	-580
8	24	13	80	70	80	45	15	28	90
8	44	13	100	100	90	75	25	28	120
3	6	4	20	20	30	8	3	12	14
5	9	6	40	30	40	16	5	18	24
5	11	7	50	40	40	19	7	18	29
5	11	7	55	40	40	21	9	18	29
0	0	1	10	4	10	0	6	0	6
3	4	5	20	14	20	6	12	10	18
3	6	9	30	29	20	12	12	10	30
3	8	9	35	29	20	16	15	10	30

PROBLEMA INICIAL CONTINUACAO

X011	X012	X013	X014	X015	X016	X017	X018	X019	X020
-400	-140	-100	-1300	-650	0	0	0	0	0
130	32	20	120	40	1	0	0	0	0
130	32	40	160	40	0	1	0	0	0
40	6	3	20	5	0	0	1	0	0
60	16	11	30	25	0	0	0	1	0
70	21	17	30	25	0	0	0	0	1
70	21	17	35	25	0	0	0	0	0
32	3	0	70	10	0	0	0	0	0
42	9	12	100	20	0	0	0	0	0
42	18	18	110	20	0	0	0	0	0
42	20	18	120	20	0	0	0	0	0

PROBLEMA INICIAL CONTINUACAO

X021	X022	X023	X024	X025	XBARRA
0	0	0	0	0	0
0	0	0	0	0	550
0	0	0	0	0	700
0	0	0	0	0	130
0	0	0	0	0	240
0	0	0	0	0	280

1	0	0	0	0	310
0	1	0	0	0	110
0	0	1	0	0	205
0	0	0	1	0	260
0	0	0	0	1	275

IMPRESSAO DO QUADRO 001 - PESO = 000

	X000	X024	X016	X018	X002	X020	X021
X000	1.00000	0.00000	0.23006	0.00000	20.42945	0.00000	0.00000
X024	0.00000	0.00000	-0.07055	0.00000	0.13497	0.00000	0.00000
X016	0.00000	1.00000	-0.36810	0.00000	-0.68712	0.00000	0.00000
X018	0.00000	0.00000	-0.13957	1.00000	0.00613	0.00000	0.00000
X002	0.00000	0.00000	0.02914	0.00000	-0.01227	0.00000	0.00000
X020	0.00000	0.00000	-0.05828	0.00000	-0.97546	1.00000	0.00000
X021	0.00000	0.00000	-0.06442	0.00000	-0.92025	0.00000	1.00000
X014	0.00000	0.00000	0.00123	0.00000	-0.01104	0.00000	0.00000
X022	0.00000	0.00000	0.03374	0.00000	0.19632	0.00000	0.00000
X015	0.00000	0.00000	-0.01196	0.00000	0.05767	0.00000	0.00000
X025	0.00000	0.00000	-0.14110	0.00000	0.26994	0.00000	0.00000

IMPRESSAO DO QUADRO CONTINUACAO

	X014	X022	X015	X025	XBARRA	PIVOT
X000	0.00000	6.50307	0.00000	0.00000	397.23926	0.00000
X024	0.00000	-1.02761	1.00000	0.00000	32.34663	0.00000
X016	0.00000	-0.40491	0.00000	0.00000	44.41718	1.00000
X018	0.00000	0.02147	0.00000	0.00000	38.17485	0.00000
X002	0.00000	-0.04294	0.00000	0.00000	8.65031	0.00000
X020	0.00000	0.08589	0.00000	0.00000	22.69939	0.00000
X021	0.00000	0.02914	0.00000	0.00000	50.02301	0.00000
X014	0.00000	0.01135	0.00000	0.00000	0.53528	0.00000
X022	1.00000	-0.81288	0.00000	0.00000	14.09509	0.00000
X015	0.00000	0.00184	0.00000	0.00000	5.84356	0.00000
X025	0.00000	-1.05521	0.00000	1.00000	24.69325	0.00000

SOLUCAO OTIMA PARA O PROBLEMA CONTINUO

SOLUCAO INTEIRA :

X000 = 190.00000
X024 = 62.00000
X016 = 151.00000
X018 = 76.00000
X017 = 271.00000
X020 = 37.00000
X021 = 65.00000
X023 = 9.00000
X022 = 14.00000
X019 = 1.00000
X025 = 72.00000

VARIAVEIS COMPLEMENTARES FORA DA BASE :

X001 = 0.00000
X002 = 1.00000
X003 = 0.00000
X004 = 0.00000
X005 = 0.00000
X006 = 0.00000
X007 = 0.00000
X008 = 1.00000
X009 = 0.00000

X010 = 0.00000
X011 = 0.00000
X012 = 0.00000
X013 = 0.00000
X014 = 0.00000
X015 = 9.00000

A8 - Execução do sistema para o problema anterior - 2 soluções

```
                                TIPO DE OTIMIZACAO : MAXIMIZACAO
                    NUMERO DE VARIAVEIS DO PROBLEMA :  25
                        NUMERO DE VARIAVEIS NA BASE :  10
                                INICIAR PELA FASE : 2
                        NUMERO DE VARIAVEIS INTEIRAS :  25
                            NUMERO DE NOS NA MEMORIA :   4
                             NUMERO DE NOS NO DISCO :  35
                               NUMERO DE NOS NA FITA :  15
          NUMERO DE COLUNAS DOS DADOS DE ENTRADA :   6
     NUMERO DE CASAS DECIMAIS DOS DADOS DE ENTRADA :   0
NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS NA BASE :   3
NUMERO DE COLUNAS DOS INDICES DAS VARIAVEIS INTEIRAS :   2
    NUMERO DE COLUNAS PARA IMPRESSAO DOS RESULTADOS :   9
NUMERO DE CASAS DECIMAIS PARA IMPRESSAO DOS RESULTADOS :   5
                    TAMANHO DA LINHA PARA IMPRESSAO :  72
                          PRECISAO UTILIZADA (EPSI) : 0.0005000000000
                       VALOR LIMITE PARA AS SOLUCOES :  -9.9999000E+74
                       NUMERO DE SOLUCOES PROCURADAS :   2
                            IMPRESSAO DAS INVERSAS :NAO
```

PROBLEMA INICIAL

X001	X002	X003	X004	X005	X006	X007	X008	X009	X010
-100	-220	-90	-400	-300	-400	-205	-120	-160	-580
8	24	13	80	70	80	45	15	28	90
8	44	13	100	100	90	75	25	28	120
3	6	4	20	20	30	8	3	12	14
5	9	6	40	30	40	16	5	18	24
5	11	7	50	40	40	19	7	18	29
5	11	7	55	40	40	21	9	18	29
0	0	1	10	4	10	0	6	0	6
3	4	5	20	14	20	6	12	10	18
3	6	9	30	29	20	12	12	10	30
3	8	9	35	29	20	16	15	10	30

PROBLEMA INICIAL CONTINUACAO

X011	X012	X013	X014	X015	X016	X017	X018	X019	X020
-400	-140	-100	-1300	-650	0	0	0	0	0
130	32	20	120	40	1	0	0	0	0
130	32	40	160	40	0	1	0	0	0
40	6	3	20	5	0	0	1	0	0
60	16	11	30	25	0	0	0	1	0
70	21	17	30	25	0	0	0	0	1
70	21	17	35	25	0	0	0	0	0
32	3	0	70	10	0	0	0	0	0
42	9	12	100	20	0	0	0	0	0
42	18	18	110	20	0	0	0	0	0
42	20	18	120	20	0	0	0	0	0

PROBLEMA INICIAL CONTINUACAO

X021	X022	X023	X024	X025	XBARRA
0	0	0	0	0	0
0	0	0	0	0	550
0	0	0	0	0	700
0	0	0	0	0	130
0	0	0	0	0	240
0	0	0	0	0	280

1	0	0	0	0	310
0	1	0	0	0	110
0	0	1	0	0	205
0	0	0	1	0	260
0	0	0	0	1	275

IMPRESSAO DO QUADRO 001 - PESO = 000

	X000	X024	X016	X018	X002	X020	X021
X000	1.00000	0.00000	0.23006	0.00000	20.42945	0.00000	0.00000
X024	0.00000	0.00000	-0.07055	0.00000	0.13497	0.00000	0.00000
X016	0.00000	1.00000	-0.36810	0.00000	-0.68712	0.00000	0.00000
X018	0.00000	0.00000	-0.13957	1.00000	0.00613	0.00000	0.00000
X002	0.00000	0.00000	0.02914	0.00000	-0.01227	0.00000	0.00000
X020	0.00000	0.00000	-0.05828	0.00000	-0.97546	1.00000	0.00000
X021	0.00000	0.00000	-0.06442	0.00000	-0.92025	0.00000	1.00000
X014	0.00000	0.00000	0.00123	0.00000	-0.01104	0.00000	0.00000
X022	0.00000	0.00000	0.03374	0.00000	0.19632	0.00000	0.00000
X015	0.00000	0.00000	-0.01196	0.00000	0.05767	0.00000	0.00000
X025	0.00000	0.00000	-0.14110	0.00000	0.26994	0.00000	0.00000

IMPRESSAO DO QUADRO CONTINUACAO

	X014	X022	X015	X025	XBARRA	PIVOT
X000	0.00000	6.50307	0.00000	0.00000	397.23926	0.00000
X024	0.00000	-1.02761	1.00000	0.00000	32.34663	0.00000
X016	0.00000	-0.40491	0.00000	0.00000	44.41718	1.00000
X018	0.00000	0.02147	0.00000	0.00000	38.17485	0.00000
X002	0.00000	-0.04294	0.00000	0.00000	8.65031	0.00000
X020	0.00000	0.08589	0.00000	0.00000	22.69939	0.00000
X021	0.00000	0.02914	0.00000	0.00000	50.02301	0.00000
X014	0.00000	0.01135	0.00000	0.00000	0.53528	0.00000
X022	1.00000	-0.81288	0.00000	0.00000	14.09509	0.00000
X015	0.00000	0.00184	0.00000	0.00000	5.84356	0.00000
X025	0.00000	-1.05521	0.00000	1.00000	24.69325	0.00000

SOLUCAO OTIMA PARA O PROBLEMA CONTINUO

SOLUCAO INTEIRA :

X000 = 190.00000
X024 = 62.00000
X016 = 151.00000
X018 = 76.00000
X017 = 271.00000
X020 = 37.00000
X021 = 65.00000
X023 = 9.00000
X027 = 14.00000
X019 = 1.00000
X025 = 72.00000

VARIAVEIS COMPLEMENTARES FORA DA BASE :

X001 = 0.00000
X002 = 1.00000
X003 = 0.00000
X004 = 0.00000
X005 = 0.00000
X006 = 0.00000
X007 = 0.00000
X008 = 1.00000
X009 = 0.00000

X010 = 0.00000
X011 = 0.00000
X012 = 0.00000
X013 = 0.00000
X014 = 0.00000
X015 = 9.00000

SOLUCAO INTEIRA :

X006 = 170.00000
X024 = 71.00000
X016 = 158.00000
X018 = 76.00000
X017 = 288.00000
X020 = 39.00000
X021 = 69.00000
X023 = 18.00000
X022 = 20.00000
X019 = 1.00000
X025 = 84.00000

VARIAVEIS COMPLEMENTARES FORA DA BASE : X001 = 1.00000
X002 = 1.00000
X003 = 0.00000
X004 = 0.00000
X005 = 0.00000
X006 = 0.00000
X007 = 0.00000
X008 = 0.00000
X009 = 0.00000
X010 = 0.00000
X011 = 0.00000
X012 = 0.00000
X013 = 0.00000
X014 = 0.00000
X015 = 9.00000