UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM ALGORITMO DE PROGRAMAÇÃO LINEAR INTEIRA ZERO-UM UTILIZANDO A TÉCNICA LEXICOGRÁFICA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

FREDERICO AGENOR ALVAREZ

FLORIANÓPOLIS

SANTA CATARINA - BRASIL

DEZEMBRO - 1979

UM ALGORITMO DE PROGRAMAÇÃO LINEAR INTEIRA ZERO-UM UTILIZANDO A TÉCNICA LEXICOGRÁFICA

FREDERICO AGENOR ALVAREZ

ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE "MESTRE EM ENGENHARIA" ESPECIALIDADE ENGENHARIA DE PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO.

PROF. LEONARDO ENSSLIN, Ph.D.
COORDENADOR

APRESENTADA PERANTE A BANCA EXAMINADORA COMPOSTA DOS PROFESSORES:

PROF. WILHELM RÖDDER, Ph.D.
PRESIDENTE

PROF. PAULO RENECIO NASCIMENTO, M.Sc.

PROF. GEOFREY MICHAEL BUXEY, Ph.D.

PROF. LEONARDO ENSSLIN, Ph.D.

Aos meu pais

Frederico e

Thereza em memória

Aos meus irmãos

Attilio

Casimiro e

Maria Elza

A cunhada

Elena

A G R A D E C I M E N T O S

Manifesto meus sinceros agradecimentos às seguintes pessoas e instituições:

- ao Prof. WILHEM RÖDDER, pela orientação no desenvolvimento deste trabalho.

- aos Professores LEONARDO ENSSLIN e PAULO RENECIO NASCIMENTO pelo interesse com que acompanharam este trabalho e pelas suas proveitosas sugestões.

- aos diretores e funcionários do NÚCLEO DE PROCESSAMENTO DE DADOS DA UFSC, em especial a MARCIO NEI FERRARI e VÂNIA ABREU DEKKER, pelo eficiente atendimento.

- aos amigos ALCEU RIBEIRO ALVES, EDVARDO BONFIM RODRIGUES JUNIOR, DOMINGOS FERNANDES CAMPOS, FERNANDO GUERRA, JOVELINO FALQUETO e MASANAO OHIRA pelo apoio e amizade demonstrados.

- aos colegas professores e funcionários do DEPARTAMENTO DE CIÊNCIAS ESTATÍSTICAS E DA COMPUTAÇÃO DA UFSC pelo incentivo.

- a todas as pessoas, que direta e indiretamente, contribuiram para a realização deste trabalho.

R E S U M O

O objetivo do presente trabalho é melhorar a eficiência computacional do algoritmo de programação linear inteira "LEXS", baseado na técnica lexicográfica.

Inicialmente, situa-se a técnica lexicográfica dentre os métodos de solução de programação inteira. Segue-se a apresentação do desenvolvimento da teoria na qual está fundamentado o algoritmo "LEXS".

Posteriormente, apresenta-se uma variante do algoritmo "LEXS", proveniente das alterações na sua estrutura básica e de incorporações de técnicas heurísticas.

Finalmente, é desenvolvida uma análise comparativa de performance para avaliar o incremento de eficiência obtida.

A B S T R A C T

The objective of the work presented here is to improve the computational efficiency of the integer linear programming algorithm "LEXS", which is based upon the lexicographic technique.

Firstly, the lexicographic technique is discussed and compared with other methods of solving the integer linear programming problem. There follows an outline of the development of the theory on which the algorithm "LEXS" is founded.

Afterwards, a variation of the algorithm "LEXS" is presented, arrived at by both alterations in the basic structure and by the incorporation of heuristic solution techniques.

Finally, a comparative analysis of performance is made to evaluate the increment of computational efficiency obtained.

S U M Á R I O

Pag.

Pag.

"O âmago da ciência, que quase sempre se apresenta como o mais importante no que se refere a resultados práticos, é a pesquisa altamente teórica e abstrata, nascida das infatigáveis curiosidade, flexibilidade e força da razão humana".

Andrei Sakharov.

C A P Í T U L O I

1. INTRODUÇÃO

Programação linear inteira é um instrumento muito valioso para orientar os decisores em problemas de distribuição de recursos limitados entre atividades competitivas, com a finalidade de atender a um determinado objetivo.

Para tanto, torna-se necessário saber "a priori" quais as atividades que consomem cada recurso, e em que proporção é feito esse consumo. Tais informações serão fornecidas por equações ou inequações lineares, uma para cada recurso. Ao conjunto destas dá-se o nome de restrições do modelo e ao objetivo proposto é associado uma função denominada de função objetivo.

Geralmente existem inúmeras maneiras de distribuir os escassos recursos entre as diversas atividades, bastando para isso que essas distribuições sejam coerentes com as equações de consumo de cada recurso, ou seja, que elas satisfaçam as restrições do problema. Entretanto, deseja-se achar a distribuição que satisfaça as restrições do problema, e que alcance o objetivo maximizado ou minimizado.

1.1. Formulação do Modelo Matemático

O modelo matemático de problemas de programa

ção linear inteira tem a seguinte formulação:

(I)

$$\text{Max(Min)}\ Z = \sum_{j=1}^{n} c_j x_j$$

Sujeito às restrições:

$$\text{(I.1)}\quad \sum_{j=1}^{n} a_{ij} x_j \gtreqless b_i,\ i = 1(1)m(*)$$

$$\text{(I.2)}\quad x_j \in N \cup \{0\},\ j=1(1)n$$

ou em notação compacta:

(II)

$$\text{Max(Min)} g_o(x_1,\ldots,x_n)$$

Sujeito às restrições:

$$\text{(II.1)}\quad g_i(x_1,x_2,\ldots,x_n) - b_i \gtreqless 0,\ i=1(1)m$$

$$\text{(II.2)}\quad x_j \in N \cup \{0\},\ j=1(1)n$$

onde:

$$g_o(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j x_j$$

$$g_i(x_1,\ldots,x_n) = \sum_{j=1}^{n} a_{ij} x_j,\ i=1(1)m$$

Onde:

m - Número de restrições

n - Número de atividades

b_i - Quantidade de recurso i disponível para as n atividades

---

(*) $j=n_1(n_3)n_2$: leia-se j variando de n_1 até n_2, passo n_3.

c_j - Retorno unitário da atividade j.

a_{ij} - Quantidade do recurso i consumida na execução de uma unidade da atividade j.

x_j - Valor do nível da atividade j, j=1(1)n.

Quando $x_j \in \{0,1\}$, j=1(1)n, o problema é dito de programação linear inteira zero-um, "PPLI01".

1.2. Objetivo do Trabalho

O presente trabalho propõe-se a melhorar a eficiência computacional de um modelo básico de programação linear inteira, utilizando a técnica lexicográfica, visando com isto proporcionar uma melhor performance computacional, através da incorporação de técnicas heurísticas e alterações na estrutura do modelo.

1.3. Conceitos Básicos

No desenvolvimento deste trabalho é utilizada uma terminologia, relativa aos problemas de programação linear inteira, que deve ser convenientemente entendida para permitir um melhor acompanhamento deste estudo.

Solução: Diz-se que $X = (x_1,x_2,\ldots,x_n)$ é uma solução de um problema de programação inteira, se e somente se, $x_j \in N \cup \{0\}$, $j = 1(1)n$.

Solução viável: $X = (x_1,x_2,\ldots,x_n)$ é uma solução viável de um pro

blema de programação inteira se não violar as suas restrições.

Solução ótima: Uma solução é ótima e denota-se por $\hat{\hat{X}}$ se:

- $\hat{\hat{X}}$ é uma solução viável e,
- Não existe solução viável X, tal que $g_o(X) > g_o(\hat{\hat{X}})$ ou $g_o(X) < g_o(\hat{\hat{X}})$, se o problema for de maximização ou de minimização, respectivamente.

Solução sub-ótima: Seja X' o conjunto das soluções já enumeradas. Uma solução $X \in X'$ é sub-ótima e denota-se por $\hat{X}$, se:

- $\hat{X}$ é uma solução viável de (I) e,
- Não existe solução viável $X \in X'$, tal que: $g_o(X) > g_o(\hat{X})$ ou $g_o(X) < g_o(\hat{X})$, se o problema for de maximização ou de minimização, respectivamente.

Ordenação Parcial: As soluções $X=(x_1,x_2,\ldots,x_n)$ e $Y=(y_1,y_2,\ldots,y_n)$ estão parcialmente ordenadas e são representadas por $X \leq Y$, se e somente se:

$$x_j \leq y_j \quad \text{para} \quad j = 1(1)n$$

Ordenação lexicográfica: As soluções $X = (x_1,x_2,\ldots,x_n)$ e $Y = (y_1,y_2,\ldots,y_n)$ estão lexicograficamente ordenadas e representa-se por $X \lesssim Y$ se e somente se:

$X = Y$ ou

Para todo j, tal que $x_j > y_j$, existe $j' < j$, tal que $x_{j'} < y_{j'}$.

Solução descendente: Seja $X^{r(*)}$ uma solução de PPLI01 caracterizada pelo conjunto J_r dos índices das variáveis x_j^r iguais a 1. Se X^s for uma solução do mesmo PPLI01 tal que $J_r \subset J_s$, dir-se-á que X^s é uma solução descendente de X^r.

Solução lexicograficamente descendente: Seja X^1 uma solução de PPLI01 e:

. L = {j tais que $x_j^1 = 1$, j = 1(1)n}

. r o maior índice pertencente a L, tal que exista s, $r < s \leq n$, onde $x_s = 0$.

Toda solução X^2 do mesmo PPLI01 será denominada lexicograficamente descendente de X^1 se:

. Existir r

. Para todo j, $j \in \{1,2,\ldots,n\}$ tal que $x_j^2=1$; então: $j \in L$ ou $j > r$.

X* - 1 - Seja X' uma solução de um PPLI01. A maior solução lexicograficamente descendente e parcialmente ordenada a esta solução será denotada por X* - 1.

X* - Dada uma solução qualquer X' de um PPLI01, a primeira solução parcialmente ordenada X, tal que, X > X', será denominada de X*.

Método para obter-se X* dado X'

- Considere X como sendo um número binário.

(*) X^r : o índice r denota a ordem da solução.

- Subtraia 1 de X obtendo X-1 (usando a operação usual de subtração dos números binários).
- Adicione X com X-1 obtendo X^*-1 (usando aritmética booleana, onde 1+1=1).
- Adicione 1 a X^*-1 obtendo X^* (usando a operação de adição usual dos números binários).

Exemplos:

1) Seja: X = (0,1,0,1,1,0,0)

X - 1 = (0,1,0,1,0,1,1)

X^* - 1 = (0,1,0,1,1,1,1)

X^* = (0,1,1,0,0,0,0)

2) Seja: X = (0,1,0,1,0,0,0)

X - 1 = (0,1,0,0,1,1,1)

X* - 1 = (0,1,0,1,1,1,1)

X^* = (0,1,1,0,0,0,0)

C A P Í T U L O I I

2. MÉTODOS DE SOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO LINEAR INTEIRA

Pode-se classificar os métodos de solução de problemas de programação inteira de uma maneira esquemática em quatro grupos, de acordo com Maculan[1]:

- Métodos de enumeração explícita;
- Métodos de cortes ou dos planos de corte;
- Métodos de "branch and bound";
- Métodos de enumeração implícita.

2.1. Enumeração Explícita

Se o problema a ser resolvido tiver n variáveis, o número de possíveis soluções será K, onde K é dado por:

$$K = \prod_{j=1(1)n} (I_j+1)$$

Onde I_j é o limite superior da variável j.

Um algoritmo que teste todas estas soluções, que são:

$$X^0 = (0,0,0,\ldots,0)$$
$$X^1 = (0,0,0,\ldots,1)$$
$$\vdots$$
$$X^K = (I_1,I_2,I_3,\ldots,I_n)$$

---

[1] MACULAN FILHO, N.. Programação linear inteira, p.2.8 - 2.10).

é chamado de enumeração explícita e o valor ótimo de g_o é assegurado.

Algoritmos de enumeração explícita, na prática, não são usados. Por exemplo:

Se o problema Max $g_o(x)$, sujeito a

$$g_i(x)-b_i \geq 0, \quad i=1(1)m$$

tiver 28 variáveis, 4 restrições e todas as variáveis forem zero-um, tem-se que o número de possíveis soluções será:

$$K = \prod_{j=1(1)28} I_j = 268.435.456 \text{ possíveis soluções}$$

Considerando uma média de 150 operações para cada solução testada e que o computador IBM-360/40 faz, em média, 250000 operações por segundo, tem-se um tempo de C.P.U. de:

$$\text{TEMPO MÉDIO DE CPU} = \frac{268435456.150}{250000} \simeq 45 \text{ HORAS.}$$

2.2. Métodos de Cortes ou de Planos de Corte

A idéia básica dos métodos de cortes segundo Maculan[2], é o acréscimo sistemático de restrições (cortes), sem que nenhuma solução inteira seja eliminada do conjunto de restrições. A seguir é apresentado o diagrama de blocos geral desta classe de algoritmos, conforme figura 1.

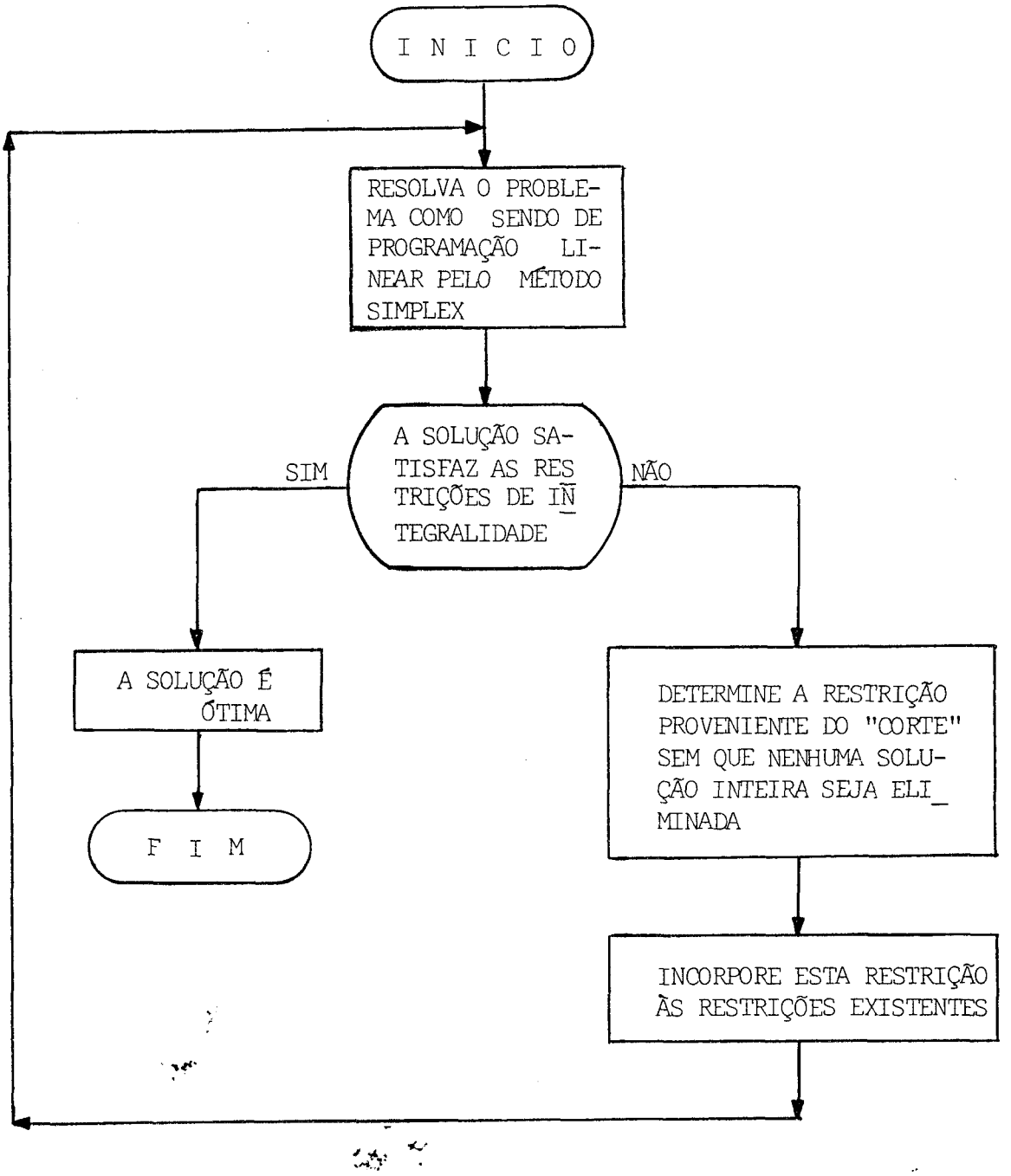


FIGURA 1 - DIAGRAMA DE BLOCOS DOS MÉTODOS DE CORTE

[2]MACULAN FILHO, N.. op. cit..

2.3. Métodos de "Branch and Bound"

Nos métodos de "branch and bound" de acordo com Maculan[3], desenvolve-se uma arborescência, cuja raíz é a solução do problema de programação linear, formado do problema de programação linear inteira retirando as restrições de integralidade. Um fluxograma simplificado deste método é apresentado, conforme figura 2.

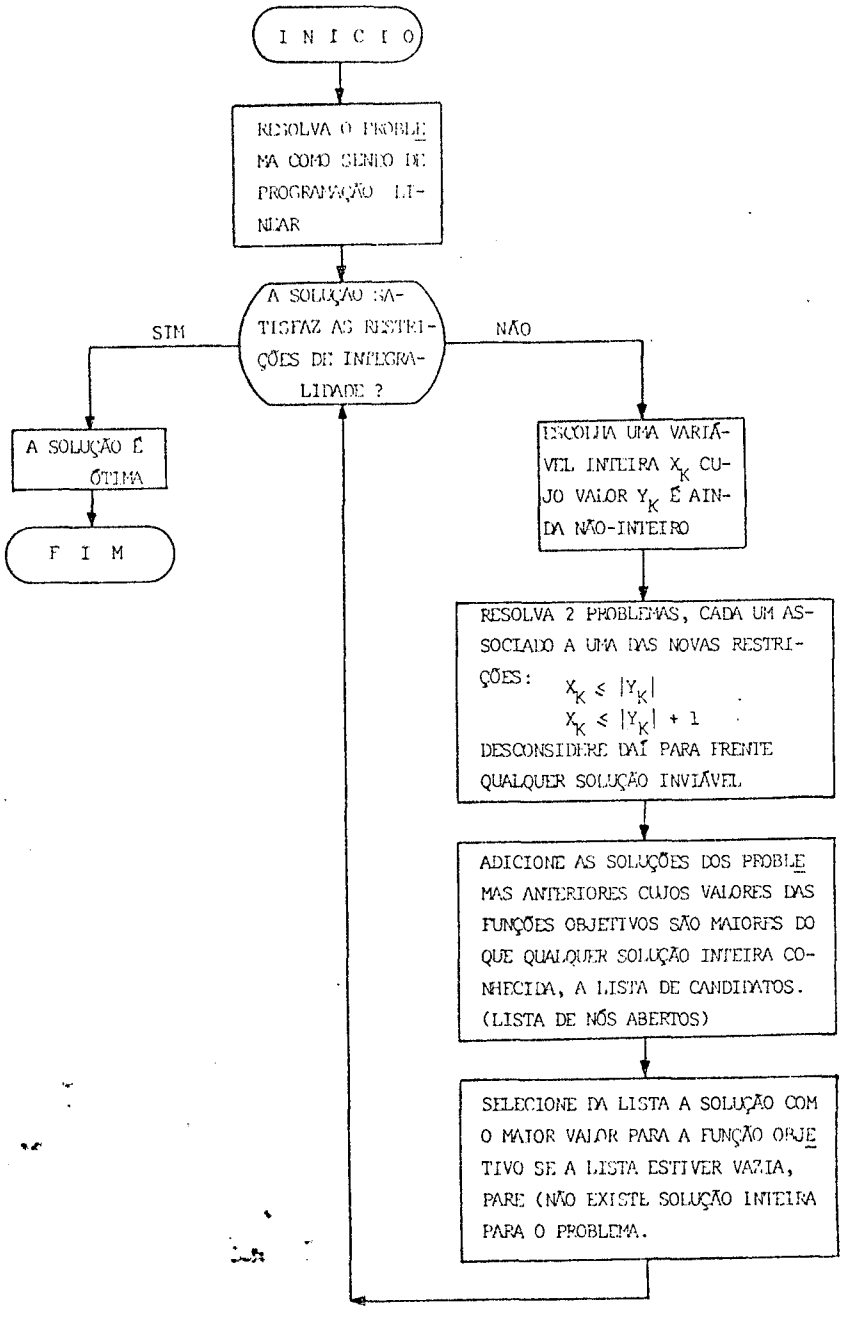


FIGURA 2 - DIAGRAMA DE BLOCOS DOS MÉTODOS DE "BRANCH E BOUND"

[3]MACULAN FILHO, N.. op. cit..

2.4. Métodos de Enumeração Implícita

Qualquer algoritmo que assegure a determinação da solução ótima, sem que seja necessário examinar todo o conjunto de soluções, é classificado como sendo de enumeração implícita.

Nos métodos de enumeração implícita, segundo Maculan[4] desenvolve-se também uma arborescência, porém de maneira diferente. Parte-se de uma solução inicial, viável ou não, e dela deriva-se uma solução ótima. Assim, esta classe de algoritmos permite que, a partir de uma solução X, desconsidere-se um subconjunto de soluções, sem que a viabilidade destas necessite ser testada. Toda solução cujo exame é evitado é dita implicitamente enumerada. Os melhores métodos são evidentemente aqueles que conduzem à geração de informações que permitem a exclusão do maior número possível de soluções do campo de investigação. Pode-se observar que a eficiência desses algoritmos está diretamente relacionada com a eficiência dos critérios estabelecidos para determinar os saltos.

---

[4]MACULAN FILHO, N.. op. cit..

C A P Í T U L O I I I

3. MÉTODOS LEXICOGRÁFICOS PARA SOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO INTEIRA

Dentre os métodos de enumeração implícita encontram-se aqueles baseados em técnicas lexicográficas, sendo que o algoritmo básico utilizando estas técnicas, foi inicialmente desenvolvido por Lawler e Bell[5] (1966), sendo seguido dos apresentados por Wallingford[6] (1969), Dragan[7] (1968-1969) e Rödder[8] (1972), que são variantes do algoritmo básico. A variação introduzida por Rödder[9], foi pelo próprio autor denominada "LEXS".

Dado que o presente trabalho desenvolve-se totalmente baseado em "LEXS", apresentar-se-á a seguir a técnica inicial de Lawler e Bell[10] e a de Rödder[11], que dela se originou.

---

[5]LAWLER, E. & BELL, M.. A method for solving... p.1098-1112.

[6]WALLINGFORD, B.A., & MAO, James C.T.. An entension... p.351-360.

[7]DRAGAN, Irinel. An algorithme lexicographique... p.246-252.

[8]RÖDDER, Wilhelm. Ein lexikographischer ... p.209-217.

[9]Id. ibid.

[10]LAWLER, E., & BELL, M., op. cit..

[11]RÖDDER, Wilhelm, op. cit..

3.1. Algoritmo Lexicográfico de Lawler e Bell[12]

3.1.1. Formulação do problema

Especificamente, este método é aplicado a qualquer problema que possa ser colocado na forma:

Minimize $g_o(X)$

Sujeito a:
$$g_{11}(X) - g_{12}(X) \geq 0$$
$$g_{21}(X) - g_{22}(X) \geq 0$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$g_{m1}(X) - g_{m2}(X) \geq 0$$

Onde:

- $X = (x_1, x_2, \ldots, x_n)$;
- $x_j = 0$ ou 1, $j = 1(1)n$ e
- As funções $g_o, g_{11}, \ldots, g_{m1}$ são monótonas e não decrescentes em cada uma das variáveis $x_1, x_2, \ldots, x_n$.

3.1.2. Critério de Lawler e Bell[13] para enumerar solução implicitamente - "Saltos"

No algoritmo de Lawler e Bell[14], salta-se de uma dada solução $X^{\cdot}$ para a solução X^{*}, se:

---

[12] LAWLER, E., & BELL, M., op.cit..

[13] Id. ibid.

[14] Id. ibid.

- Regra 1: $g_o(X^{\cdot}) \geq g_o(X)$, onde X é a solução sub-ótima até o momento.

- Regra 2: $X^{\cdot}$ é uma solução viável e produz o melhor valor da função objetivo até aqui conhecido.

- Regra 3: existir $i \in \{1,2,\ldots,m\}$, tal que $g_{i1}(X^{*}-1) - g_{i2}(X^{\cdot}) < 0$.

Justificação:

- Regra 1: A função g_o é monótona não decrescente, portanto, nenhuma das soluções X tal que $X^{\cdot} \leq X \leq X^{*} - 1$, pode ter um custo menor, dado que as soluções estão parcialmente ordenadas.

- Regra 2: Faz-se $\hat{X}=X^{\cdot}$, e aplica-se a justificativa da Regra 1.

- Regra 3: Temos que g_{i1} é monótona não decrescente e g_{i2} é monótona não crescente, para i=1(1)m; então:

$$g_{i1}(X^{*}-1)-g_{i2}(X^{\cdot}) = \max_{X^{\cdot}\leq X\leq X^{*}} g_{i1}(X) -$$

$$- \min_{X^{\cdot}\leq X\leq X^{*}} \{g_{i2}(X)\} < 0$$

3.1.3. Diagrama de blocos do algoritmo de Lawler e Bell[15]

Para se ter uma visão geral do algoritmo desenvolvido por Lawler e Bell[16], apresenta-se o diagrama de blocos correspondente, conforme figura 3.

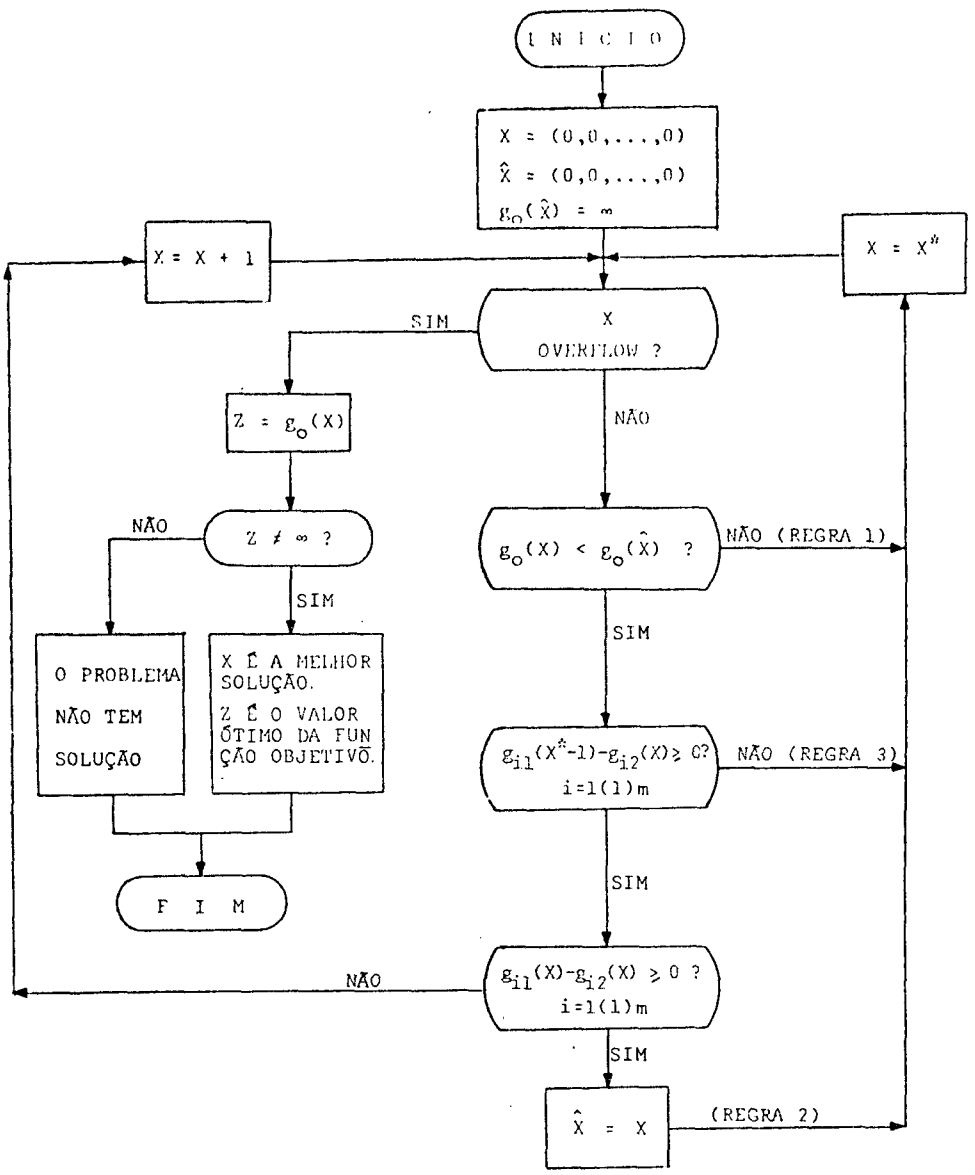


FIGURA 3 - DIAGRAMA DE BLOCOS DO ALGORITMO DE LAWLER E BELL

---

[15] LAWLER, E., & BELL, M., op. cit..

[16] Id. ibid.

3.2. Algoritmo Lexicográfico de Rödder[17] - "LEXS"

O trabalho de Rödder[18] é totalmente desenvolvido com base no trabalho de Lawler e Bell[19], apresentando as seguintes variações:

- Vários dos saltos são condensados em um único salto. Assim, torna-se supérfluo o cálculo repetido de g_{i1}, g_{i2}, respectivamente.
- Seja aplicável em problemas de otimização nos quais as variáveis de decisão podem assumir valores inteiros limitados, sem ter que transformá-las em variáveis binárias.

3.2.1. Forma padrão do problema

Para utilização do algoritmo lexicográfico (LEXS) para resolver problemas de programação linear inteira, o problema analisado deverá estar na seguinte forma:

---

[17] RÖDDER, Wilhelm, op. cit..

[18] Id. ibid..

[19] LAWLER, E., & BELL, M., op. cit.

(III)

Max $g_o(x_1,\ldots,x_n)$, sujeito às seguintes restrições.

$g_i(x_1,\ldots,x_n) - b_i \geq 0$, $i = 1(1)m$

onde

$g_i(x_1,\ldots,x_n) = a_{i1}x_1+\ldots+a_{in}x_n$

x_j pertencente $D_j = \{0,1,\ldots,I_j\}$, $j = 1(1)n$

a_{ij} pertencente ao conjunto dos números inteiros

I_j = limite superior da variável x_j,

I_j pertencente $N \cup \{0\}$

D_j = conjunto de valores que a variável x_j pode assumir

m = número de restrições

n = número de variáveis

3.2.2. Obtenção do limite superior de cada variável

É necessário que cada variável tenha um limite superior, pois a eficiência do algoritmo é inversamente proporcional à ordem do limite superior.

Ao modelar-se um problema, é muito vantajoso considerar-se o menor limite superior permitido a cada variável.

3.2.3. Incorporação da função objetivo com uma restrição

O algoritmo LEXS incorpora às m restrições existentes do problema (III) a função objetivo como sendo mais uma restrição, passando o problema a ter m+1 restrições.

Para que essa incorporação não afete o problema original, o valor inicial de b_o será:

$$b_o \leq - \sum_{j=1}^{n} |a_{oj}| \cdot I_j, \text{ então } g_o(X) - b_o \geq 0, \quad \text{pois}$$

$$g_o(X) \leq -b_o \text{ para qualquer } X.$$

Assim, quando se obtiver a primeira solução $X^{\cdot}$ que não viole as restrições do problema, ela necessariamente verificará g_o, pois cada variável x_j é limitada e:

$$g_o(X^{\cdot}) - b_o = g_o(X^{\cdot}) + \sum_{j=1}^{n} |a_{oj}| \cdot I_j \geq 0 \text{ pelas consi-}$$

derações feitas acima.

Após cada solução viável determinada, a restrição decorrente da função objetivo torna-se mais atuante, pela redefinição do valor de b_o como segue:

Seja X^1 uma solução viável, então:

$b_o = - (g_o(X^1) + \delta$, δ pertencente aos números naturais e suficientemente pequeno.

Portanto, o algoritmo só testará nas restrições específicas do problema as soluções X lexicograficamente maiores que X^1, que verificam a restrição decorrente da função objetivo, isto é, somente as soluções X que satisfaçam as condições:

$$- X \succ X^1 \quad e$$

$$- g_o(X) - b_o \geq 0$$

3.2.4. "Salto"

3.2.4.1. Critério de salto para uma restrição

Apresenta-se agora como é estabelecido um salto de X^1 para X^2, assegurando que a solução ótima $\hat{\hat{X}}$ não esteja compreendida entre X^1 e X^2.

Sejam:

- $X^{\cdot} = (x_1^{\cdot}, x_2^{\cdot}, \ldots, x_n^{\cdot})$ pertencente ao conjunto de soluções que não satisfaçam as restrições (III).

- $V \subset \{0,1,2,3,4, \ldots, m\}$, conjunto ordenado dos índices correspondentes às restrições as quais não foram satisfeitas para o vetor $X^{\cdot}$.

- $J = \{1,2,3,\ldots,n\}$, conjunto ordenado dos índices correspondentes às variáveis do problema.

Para cada $i \in V$ é associado um, e somente um, índice $j_i \in J$ que tenha a seguinte propriedade:

Para qualquer $j > j_i$ tem-se

$$\max_{x_j \in D_j, \ldots, x_n \in D_n} g_i(x_1^{\cdot}, \ldots, x_{j-1}^{\cdot}, x_j, \ldots, x_n) - b_i < 0$$

mas

$$\max_{x_{j_i} \in D_{j_i}, \ldots, x_n \in D_n} g_i(x_1^{\cdot}, \ldots, x_{j_i-1}^{\cdot}, x_{j_i}, \ldots, x_n) - b_i \geq 0$$

Assim distinguem-se dois casos:

1 - Se g_i em x_{j_i} é monótona não decrescente, todos os vetores X tais que:

$$X^{\cdot} \lesssim X \lesssim Z = (x_1^{\cdot},\ldots,x_{j_i}^{\cdot}-1,x_{j_i}+1,0,\ldots,0)$$

são inadimissíveis para a i-ésima restrição.

Portanto, pode-se saltar para Z, e denota-se o índice j_i por j_i^*.

2 - Se g_i em x_{j_i} for monótona não crescente, um aumento na componente j_i não tornaria viável a restrição i.

Por esse motivo, eleva-se de uma unidade a componente j_i^* à esquerda de j_i, para a qual $x_{j_i^*} \neq I_{j_i}$.

Assim pode-se dizer que:

Se j_i^* existe, tem-se que todo X tal que:

$X^{\cdot} \lesssim X \lesssim Z = (x_1^{\cdot},\ldots,x_{j_i^*}^{\cdot}-1,x_{j_i^*}^{\cdot} + 1,0,\ldots,0)$ é inadimissível para a i-ésima restrição.

Portanto, pode-se saltar a partir de $X^{\cdot}$, no mínimo até Z.

Se ocorrer a não existência de j_i para a i-ésima restrição, isso indica que todo $X \gtrsim X^{\cdot}$ são inviáveis para esta restrição.

3.2.4.2. Agregação de saltos de restrições não satisfeitas

O procedimento de análise mencionado em (3.2.4.1) pode ser estendido como segue:

Para um salto não se considera apenas a estrutura de uma restrição, mas todas as restrições tais que:

$$g_i(x) - b_i < 0 \quad \text{com } i \in V$$

É evidente que com $j^* = \min_{i \in V} \{j_i^*\}$ as afirmações mencionadas em (3.2.4.1) para j^* em lugar de j_i^* são válidas.

Para problemas com função objetivo e restrição lineares, o cálculo seqüencial do máximo para determinar-se j_i^*, partindo de uma solução inviável $X^\cdot$, é consideravelmente simplificado, como segue:

Seja $S_i = \sum_{j=1}^{n} a_{ij} x_j^\cdot - b_i < 0$ a causa da não admissibilidade de $X^\cdot$.

Então é válido:

$$\max_{x_j \in D_j, \ldots, x_n \in D_n} g_i(x_1^\cdot, \ldots, x_{j-1}^\cdot, x_j, \ldots, x_n) =$$

$$= S_i + \max_{x_j \in D_j, \ldots, x_n \in D_n} \sum_{k=j}^{n} a_{ik} x_k =$$

$$= S_i + \sum_{k=j}^{n} (a_{ik}^+ \bar{x}_k^\cdot + a_{ik}^- x_k^\cdot), \text{ para qualquer } j \in J$$

Notação:

$a^{+}_{ik} = \max(a_{ik}, 0)$

$a^{-}_{ik} = \max(-a_{ik}, 0)$

$\bar{x}_k = I_k - x_k$

3.2.5. Caso especial

Em problemas de programação linear inteira zero-um tem-se:

$$I_j = 1 \text{ para } j = 1(1)n$$

Seja:

$X^{\cdot} = (x^{\cdot}_1, \ldots, x^{\cdot}_n)$ não admissível, portanto, $S_i = \sum_{j=1}^{n} a_{ij}\, x^{\cdot}_j - b_i < 0$ e j^{*}_i será o primeiro índice j que possuir a propriedade:

$$S_i + \sum_{k=j}^{n} (a^{+}_{ik}\, x^{\cdot}_k + a^{-}_{ik}\, x^{\cdot}_k) \geq 0$$

onde,

$\bar{x}_j = 1 - x^{\cdot}_j$

$a^{+}_{ik} = \max(0, a_{ik})$

$a^{-}_{ik} = \max(-a_{ik}, 0)$

3.2.6. Exemplo

Antes de ser demonstrada a resolução de (III)

em forma de um algoritmo apresenta-se o exemplo de Rödder[20], para mostrar como partir de um vetor X' inadmissível e encontrar o primeiro vetor lexicograficamente maior que X', utilizando a técnica de salto exposta em (3.2.4.2).

Exemplo:

Ache, partindo de X' = (0,0,0), o primeiro vetor lexicograficamente maior que X', utilizando a técnica de salto exposta em (3.2.4.2), do seguinte sistema de inequações.

$$4x_1 + 3x_2 + \frac{1}{2}x_3 - 4 \geq 0$$

$$3x_1 - x_3 - 3 \geq 0$$

$$x_j \in \{0,1,2,3,4\},\ j = 1(1)3$$

Solução:

Para x' = (0,0,0), tem-se que:

$$\left.\begin{array}{l} S_1 = -4 \\ S_2 = -3 \end{array}\right\} \Longrightarrow \quad V = \{1,2\}$$

Segue o cálculo de j^*

Para i = 1, tem que:

Para j = 3

$$-4 + \sum_{j=3}^{3} (a_{1j}^{+} \bar{x}_j + a_{1j}^{-} x_j) = -4 + \frac{1}{2}(4) = -2 < 0$$

Para j = 2

$$-4 + \sum_{j=2}^{3} (a_{1j}^{+} \bar{x}_j + a_{1j}^{-} x_j) = -4 + \frac{1}{2}.4 = 10 > 0$$

---

[20] RÖDDER, Wilhel, op. cit..

Portanto $j_1^* = 2$

Para i = 2, tem-se que:

Para j = 3

$$-3 + \sum_{j=3}^{3} (a_{2j}^{+} \bar{x}_j + a_{2j}^{-} x_j) = -3+4.0 = -3 < 0$$

Para j = 2

$$-3 + \sum_{j=2}^{3} (a_{2j}^{+} \bar{x}_j + a_{2j}^{-} x_j) = -3+0.4+4.0=-3<0$$

Para j = 1

$$-3 + \sum_{j=1}^{3} (a_{2j}^{+} \bar{x}_j + a_{2j}^{-} x_j) = -3+3.4+0.4+4.0=9>0$$

tem-se que $j_2^* = 1$

Portanto $j^* = \min_{i \in V} \{j_i^*\} = \min \{j_1^*, j_2^*\} =$

$\min \{2,1\} = 1$

Assim, pode-se saltar de $x^{\cdot} = (0,0,0)$ para o primeiro vetor admissível da desigualdade na ordenação lexicográfica que é Z = (1,0,0)

$x^{\cdot} =$ (0,0,0)
(0,0,1)
⋮ } *
(0,4,4)
Z = (1,0,0)

Pelo método de cálculo de j*, pode-se afirmar que nenhuma solução pertencente ao conjunto * é viável.

3.2.7. Diagrama de blocos do algoritmo LEXS

Para se ter uma visão global do algoritmo LEXS, é apresentado o diagrama de blocos correspondente, conforme figura 4.

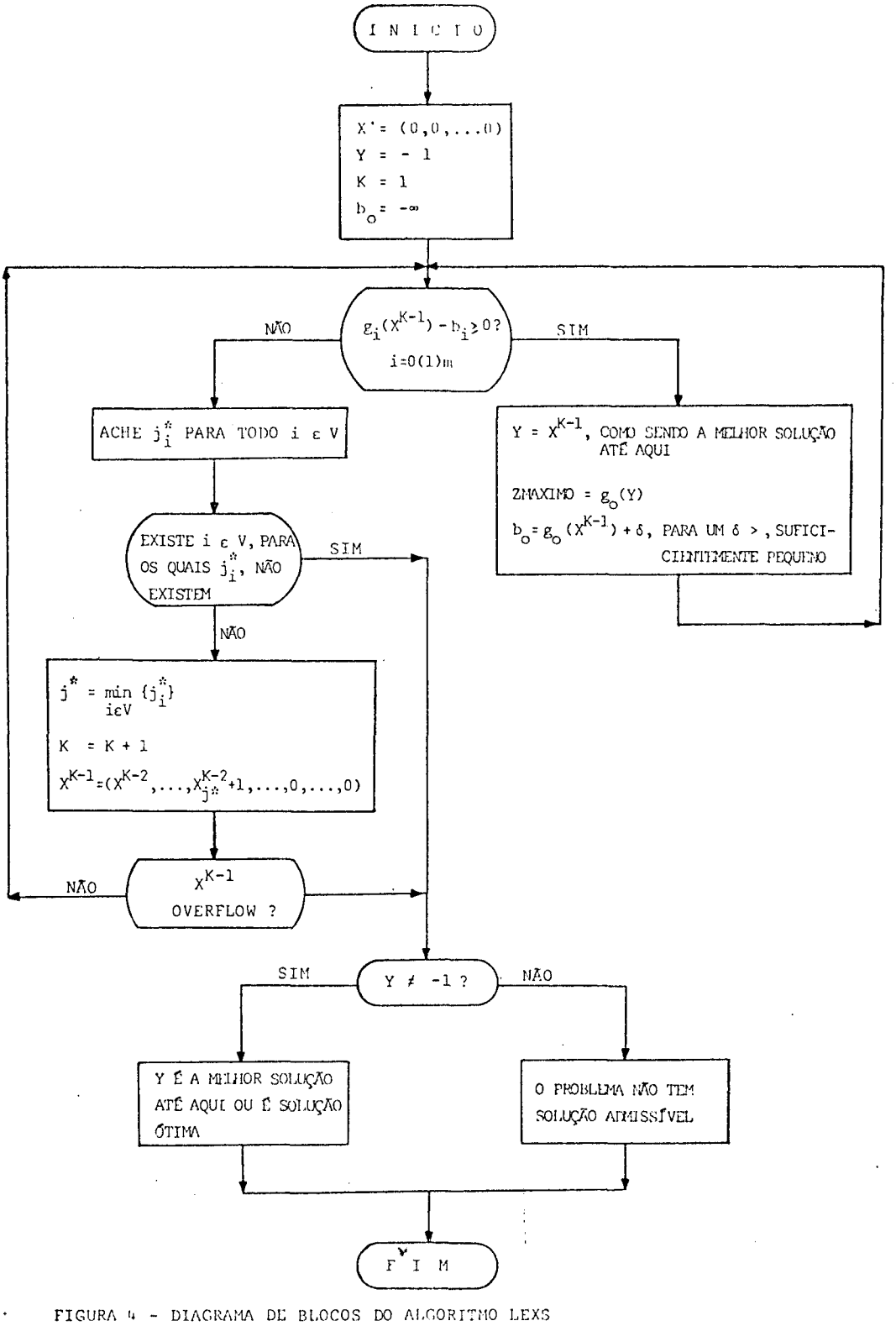


FIGURA 4 - DIAGRAMA DE BLOCOS DO ALGORITMO LEXS

C A P Í T U L O I V

4. MODELO PROPOSTO

Para melhor compreensão do modelo, apresenta-se o seu desenvolvimento em três etapas, a saber:

a) Apresentação do algoritmo LEXSM

b) Estudo das heurísticas

c) Modelo proposto propriamente dito.

4.1. O Algoritmo LEXSM

Tomando-se por base o algoritmo LEXS e mantendo os seus critérios de enumeração lexicográfica e de enumeração implícita, respectivamente, e realizando modificações em sua estrutura básica, obteve-se um novo algoritmo denominado de LEXSM. Apresenta-se a seguir um conjunto de considerações para poder efetuar as alterações, o fluxo de LEXSM e um estudo comparativo dos resultados obtidos com LEXS e LEXSM.

Primeira consideração: Forma padrão do problema

No desenvolvimento da teoria de LEXSM, considerou-se que o problema esteja na forma padrão

(IV)

$$\text{Max } Z = \sum_{j=1}^{n} cc_j \cdot y_j \qquad \text{(IV.1)}$$

Sujeito a:

$$\sum_{j=1}^{n} aa_{ij}\, y_j \leq b_i, \quad i = 1(1)m \qquad \text{(IV.2)}$$

$$y_j \in \{0,1\}, \quad j = 1(1)n \qquad \text{(IV.3)}$$

$$cc_j \geq 0, \quad \forall_j = 1(1)n \qquad \text{(IV.4)}$$

$$\forall_{j,k} \text{ tais que: } 1 \leq j < k \leq n \Longrightarrow cc_j \leq cc_k \qquad \text{(IV.5)}$$

As hipóteses (IV.4) e (IV.5) não invalidam a generalização do problema, pois qualquer problema de programação linear inteira binária pode ser transformado na forma padrão (IV), como segue:

Obtenção da hipótese (IV): '$c_j \geq 0$, qualquer $j = 1(1)n$'

Para cada $c_k < 0$ do problema original, faz-se a seguinte mudança de variável.

$x_k = 1 - x'_k$, onde $x'_k \in \{0,1\}$, logo:

$$c_k x_k = c_k(1-x'_k) = c_k - c_k x'_k$$

Seja $c'_k = -c_k$, então $c'_k > 0$ e a função objetivo ficará:

$$g_o(X) = c_k + \sum_{\substack{j=1 \\ j\neq k}}^{n} c_j x_j + c'_k x'_k, \text{ fazendo}$$

$F(X) = g_o(X) - c_k$, tem-se:

$F(X) = \sum_{\substack{j=1 \\ j\neq k}}^{n} c_k x_j + c'_k x'_k$ que é uma função com todos os coeficientes não negativos equivalentes à função inicial $g_o(X)$.

Obtenção da hipótese (IV.5):

"Quaisquer j e k, tais que:

$$1 \leq j < k \leq n \Longrightarrow cc_j \leq cc_k"$$

Para obter-se esta segunda hipótese, define-se uma transformação T, que, aplicada ao problema com a hipótese, (IV.4) já verificada, o transformará na forma padrão (IV).

A transformação T é definida como segue:

$T(1,2,\ldots,n)=(j_1,j_2,\ldots,j_n)$, tal que $c_{j_1} \leq c_{j_2} \leq \ldots \leq c_{j_n}$, passando o problema a ter a seguinte notação:

$y_k = x_{j_k}$ para $k=1(1)n$

$cc_k = c_{j_k}$ para $k=1(1)n$

$aa_k = a_{j_k}$ para $k=1(1)n$ e para $i=1(1)m$

Como o algoritmo LEXSM utiliza o problema na forma (IV), a cada solução viável Y obtida, necessário se faz que se determine a solução correspondente X, pela aplicação da transformação inversa da transformação T, obtendo-se a solução viável correspondente do problema original.

Segunda consideração: Definição do vetor SOFO, com a função de armazenar acumulativamente os valores dos coeficientes da função objetivo, onde cada elemento é definido por:

$$SOFO_k = \sum_{j=k}^{n} cc_j, \quad k = 1(1)n$$

Terceira consideração: Ordenar as restrições heuristicamente, segundo a sua sensibilidade.

Assume-se aqui como sensibilidade a folga de cada restrição.

Se todos os b_i forem positivos, a sensibilidade de cada restrição será definida por

$$SENSIBILIDADE_i = \left(\sum_{j=1}^{n} aa_{ij}^{+} - b_i \right) / b_i; \quad i = 1(1)m$$

onde $aa_{ij}^{+} = \max \{0, aa_{ij}\}$

Caso $\sum_{j=1}^{n} aa_{ij}^{+} - b_i \leq 0$, esta restrição será desconsiderada do problema.

Senão, se algum b_i for não positivo, a sensibilidade de cada restrição será dada por:

$$SENSIBILIDADE_i = b_i - \sum_{j=1}^{n} aa_{ij}^{+}$$

Passando-se a considerar, no desenvolvimento do algoritmo LEXSM, que as restrições (IV.2) estejam sempre ordenadas em ordem decrescente de sensibilidade.

4.1.1. Modificações introduzidas no algoritmo LEXS

Estas considerações têm por finalidade servirem de suporte para que as modificações estruturais do algoritmo produzam os melhores resultados, como segue:

Primeira modificação: Determinação j_o^*

Seja X uma solução viável e $X^{\cdot} \nsubseteq X$ uma solução inadmissível para a restrição da função objetivo, então:

$$S_o = g_o(X^{\cdot}) - b_o < 0$$

O algoritmo LEXS utiliza o seguinte processo para determinar j_o^*.

- j_o^* será o primeiro índice j, tal que:

$$\sum_{k=1}^{j-1} c_k x_k^{\cdot} + \sum_{k=j}^{n} (c_k^+ \bar{x}_k^{\cdot} + c_k^- x_k^{\cdot}) - b_o \geq 0$$

onde:

$$c_k^+ = \max (0, c_k)$$

$$c_k^- = \max (-c_k\ 0)$$

$$\bar{x}_j = 1 - x_j$$

Este processo utilizando o algoritmo LEXSM é modificado para:

- j_o^* será o primeiro índice i, tal que:

$$SOFO_i - b_o \geq 0, \text{ onde } SOFO_i = \sum_{k=i}^{n} cc_i$$

Este procedimento é justificado pelo fato de todos os cc_j serem não negativos e devido à estrutura do vetor SOFO.

Por outro lado, este processo garante que j_o^* seja o 'salto' otimizado, pois os coeficientes da função objetivo estão ordenados em ordem crescente.

Portanto, evidentemente, um número substancial de testes e operações de soma para a determinação de j^* será suprimido no cômputo geral da aplicação do algoritmo.

Segunda modificação: Consideração das restrições

O algoritmo LEXS para enumerar uma solução considera todas as restrições. Este procedimento acarreta um número elevado de operações para se efetuar a enumeração, acentuando-se em problemas com muitas restrições. O método proposto e utilização em LEXSM minimiza o número de operações envolvidas na enumeração de uma solução, fazendo uma filtragem. Este procedimento é apresentado a seguir.

Método de enumeração de uma solução de LEXSM

A enumeração de uma solução será realizada em duas etapas:

Primeira etapa: Filtragem

Consideram-se atuantes somente duas restrições que são:

a) A restrição proveniente da função objetivo.

b) Uma restrição flutuante.

A restrição flutuante será a restrição mais sensível do conjunto de restrições. Uma solução somente atingirá a segunda etapa se for viável às restrições a) e b), simultaneamente.

Segunda etapa: Viabilidade geral de uma solução

Após a obtenção de uma solução viável na primeira etapa, a mesma será testada nas restrições complementares, pela ordem decrescente de sensibilidade. Se, ao testar-se a viabilidade da solução em uma restrição, ocorrer a violação da mesma, esta passa a ser a restrição flutuante e volta-se à etapa um. Se todas as restrições forem verificadas, atualiza-se o valor de b_o, e a restrição flutuante volta a ser a mais sensível, retornando-se à etapa um.

Estas são as modificações que foram realizadas no algoritmo LEXS gerando o algoritmo LEXSM.

4.1.2. Diagrama de blocos do algoritmo LEXSM

Para uma visualização geral do algoritmo LEXSM, apresenta-se o diagrama de blocos correspondente, conforme figura 5.

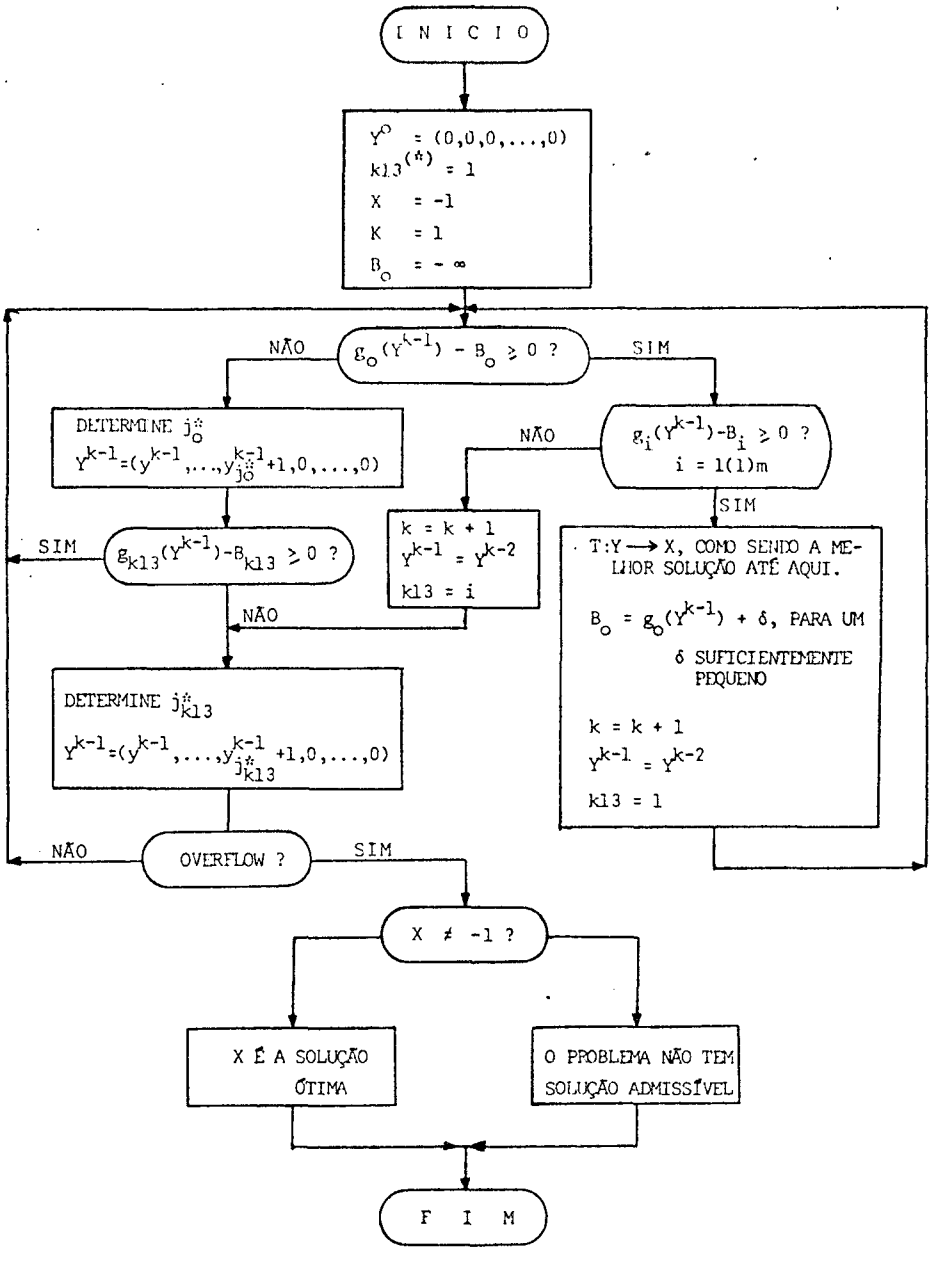


FIGURA 5 - DIAGRAMA DE BLOCOS DO ALGORITMO LEXSM

(*) k13 indica a restrição mais sensível do conjunto de restrições, para uma solução específica.

4.1.3. Análise comparativa: LEXS e LEXSM

Para realizar a análise comparativa, os algoritmos LEXS e LEXSM foram programados em FORTRAN, anexo 1 e anexo 3, respectivamente. Em seguida, foram resolvidos no computador IBM-360/40, da Universidade Federal de Santa Catarina, os problemas apresentados no Anexo 2. Os resultados obtidos são apresentados na tabela 1.

TABELA 1 - PERFORMANCE COMPARATIVA DOS ALGORITMOS LEXS E LEXSM

PROBLEMAS	NV	NR	SOLUÇÃO ÓTIMA	VALOR OBTIDO		TEMPO DE CPU EM SEGUNDOS	
				LEXS	LEXSM	LEXS	LEXSM
1	6	10	3800	3800	3800	4	1
2	10	4	70	70	70	5	1
3	10	7	-23	-23	-23	82	2
4	15	10	4015	4015	4015	21	13
5	15	15	10	10	10	18	6
6	15	50	9	9	9	109	137
7	20	3	26	26	26	25	37
8	20	10	6120	6120	6120	371	41
9	28	10	12400	11970*	12400	3500	280
10	31	31	18	18*	18*	3466	2438
11	39	5	10618	9777*	10618	3333	2357
12	50	5	16537	10756*	15148*	3226	2457

NV = NÚMERO DE VARIÁVEIS
NR = NÚMERO DE RESTRIÇÕES
* = NÃO FORAM ENUMERADAS TODAS AS SOLUÇÕES

Os dados tabulados na tabela 1 evidenciam claramente a performance superior do algoritmo LEXSM.

4.2. Seleção e Incorporação de Heurísticas

Os algoritmos LEXS e LEXSM incorporam às m restrições do problema, outra, decorrente da função objetivo, ficando-se então com m+1 restrições. A adição desta nova restrição terá a finalidade de, uma vez encontrada a primeira solução viável, tornar todas as demais soluções viáveis que advirem melhores que a anterior, melhorando-se passo a passo a função objetivo até a sua maximização.

Note-se que a nova restrição introduzida só passa a ser ativa após ter-se encontrado a primeira solução viável.

Portanto, se, antes de ser aplicada a rotina iterativa dos algoritmos, incorporar-se ao mesmo um processo que forneça uma solução viável que, aplicada na função objetivo, produza um valor na vizinhança do ótimo da mesma, iniciar-se-á com a restrição incorporada extremamente atuante. Este fato acarreta um número substancial de soluções que serão enumeradas implicitamente.

Observe-se que a eficiência deste processo é inversamente proporcional ao tempo de CPU utilizado para a obtenção da solução acima, e diretamente proporcional ao valor que essa solução produz na função objetivo.

Esse processo é possível de ser realizado através de uso de heurística.

Assim sendo, foram estudadas as seguintes heurísticas:

- Incremento Absoluto
- Incremento Relativo
- Senju e Toyoda
- Kochenberger

O estudo destas heurísticas consta de um procedimento padrão que se resume em:

- Programação
- Testes
- Análises

4.2.1. Heurística de incremento absoluto

Esta heurística leva em consideração somente os valores de c_j, incluindo, se possível, os de maiores valores, desconsiderando totalmente as restrições.

O processo inicia com todas variáveis x_j iguais a zero.

Iterativamente a variável x_k (onde k é o índice de c_k, tal que, $c_k = \underset{j=1(1)n}{\text{Max}} \{c_j\}$) assume o valor um, sendo testada a viabilidade da nova solução desta decorrência. Se viável, x_k permanece com o valor um; caso contrário, volta a ter o valor zero. O processo termina quando todas as variáveis forem testadas.

Este processo encontra-se na figura 6.

4.2.2. Heurística de incremento relativo

O processo inicia com todas as variáveis x_j iguais a um e verifica a viabilidade desta solução. Se viável, o processo termina; caso contrário, iterativamente a variável x_k (onde k é o índice da variável G_k definida como:

$$G_k = \min_{j=1(1)n} \{G_j = c_j / \sum_{i=1}^{n} a_{ij}^{+}\}, a_{ij}^{+} = \max\{0, a_{ij}\})$$

assumirá o valor zero. O processo termina quando uma solução viável for encontrada.

Este método apresenta como ponto negativo o fato de considerar todas as restrições com pesos iguais, não detectando as influências acentuadas de algumas restrições. Em outras palavras, considera somente o incremento relativo, desconsiderando o custo relativo, isto é, a relação b_i/a_{ij}.

Esta heurística encontra-se na figura 7.

4.2.3. Heurística Senju e Toyoda

Esta heurística leva em consideração as restrições do problema, procurando eliminar a influência acentuada de alguma restrição pela seguinte normalização:

$$S_i = \text{Max } (0, \sum_{P} (a_{iP}/b_i) - 1), \quad i=1(1)m$$

onde, P é o conjunto de variáveis iguais a 1.

O processo inicia com todas as variáveis x_j

iguais a um e verifica a viabilidade desta solução. Caso não seja viável, a variável x_k (onde k é o índice de G_k, tal que:

$$G_k = \min_{j=1(1)n} \{G_j = c_j / \sum_{i=1}^{m} S_i(a_{ij}/b_i)\})$$

assumirá o valor zero. O processo termina quando uma solução viável for encontrada.

Apresenta como limitação a necessidade de todos os b_i serem positivos.

O processo completo será exposto na figura 8.

4.2.4. Heurística Kochenberger

Inicia o processo com todos os x_j iguais a zero, como também leva em consideração as restrições. A cada solução viável determinada, faz a seguinte normalização.

$$b_i^* = b_i - \sum_{j=1}^{n} a_{ij}\, x_j$$

Em seguida, a variável x_k (onde k é o índice de G_k, tal que, $G_k = \max_{j=1(1)n} \{G_j = c_j / \sum_{i=1}^{n} (a_{ij}/b_i^*)\}$) assume o valor um. Se a solução decorrente deste procedimento for viável, então x_k permanece igual a um; caso contrário, assume o valor zero. O procedimento termina quando todas as variáveis forem testadas.

Tem como limitação a obrigatotiedade de todos os b_i serem positivos.

O fluxo completo desta heurística é apresenta

do na figura 9.

4.2.5. Análise e seleção das heurísticas

As heurísticas aqui citadas foram programadas em FORTRAN e testadas nos problemas que vêm sendo utilizados. Os resultados obtidos encontram-se na tabela 2.

Pela análise das heurísticas, nota-se que as de incremento absoluto e incremento relativo apresentam o maior erro. Este fato é justificável, pois as mesmas não fazem restrição à estrutura do problema, sendo que a mais precisa destas duas é a heurística de incremento relativo, enquanto as específicas apresentam uma performance superior, sobressaindo-se, entre estas, a de Kochenberger.

Devido ao modelo proposto ser geral, não tendo nenhuma restrição quanto à estrutura do problema, a heurística geral engloba as heurísticas de incremento relativo e de Kochenberger, sendo que a primeira somente será usada se houver algum b_i não positivo, como se pode notar na figura 10.

TABELA 2 - ANÁLISE COMPARATIVA DAS HEURÍSTICAS

P	NV	NR	MZ	VALOR FUNÇÃO OBJETIVO				TEMPO DE CPU EM SEGUNDOS				ERRO %			
				H1	H2	H3	H4	H1	H2	H3	H4	H1	H2	H3	H4
1	10	4	70	70	70	70	70	1	2	1	1	0	0	0	0
2	6	10	3800	2400	3800	3300	3800	1	1	1	1	36.0	0	13.1	0
3	15	10	4015	3950	3825	3325	4005	1	3	2	2	1.6	4.7	17.1	0.2
4	20	10	6120	6070	6010	6010	6010	2	4	3	4	0.8	1.7	1.7	1.7
5	28	10	12400	12180	12400	11950	12400	13	5	4	5	1.7	0	3.6	0
6	39	5	10618	9209	9888	10512	10313	2	5	4	6	13.2	6.8	0.9	2.8
7	50	5	16537	13863	15832	16263	16031	3	22	5	9	16.1	4.2	1.6	3.0

P = Número do Problema
NV = Número de Variáveis
NR = Número de Restrições
MZ = Valor Máximo da Função Objetivo
H1 = Incremento Absoluto
H2 = Incremento Relativo
H3 = Senju and Toyoda
H4 = Kochenberger

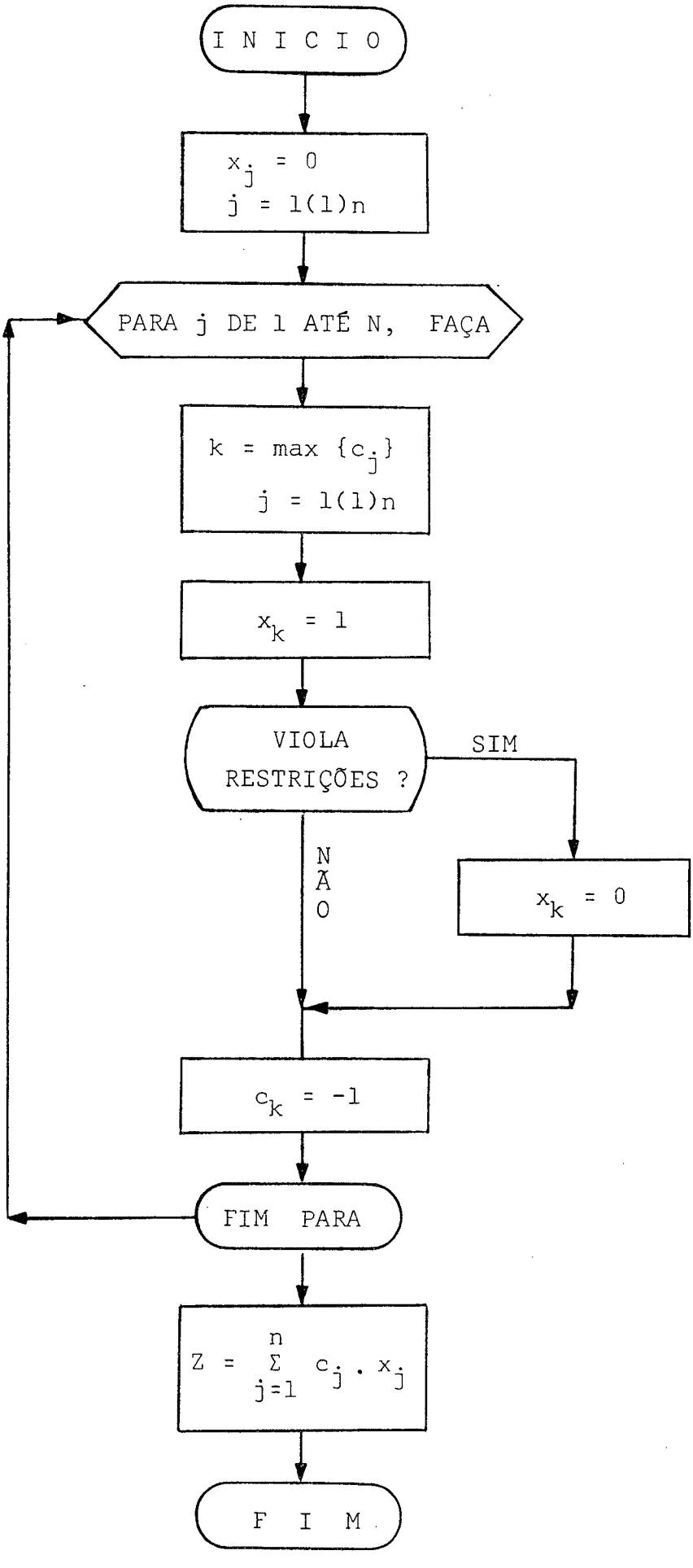


FIGURA 6 - DIAGRAMA DE BLOCOS DA HEURÍSTICA DE INCREMENTO ABSOLUTO

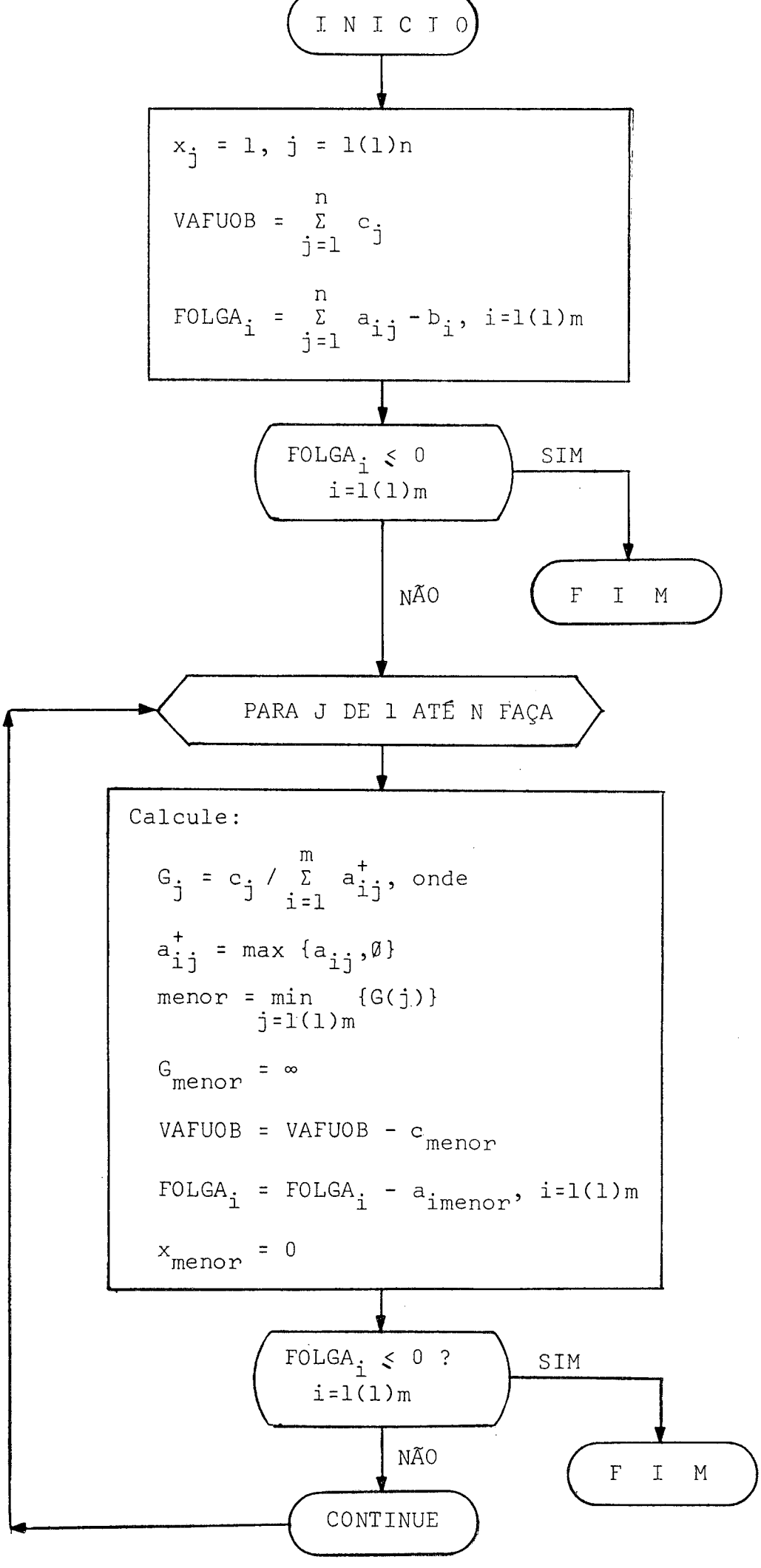


FIGURA 7 - DIAGRAMA DE BLOCOS DA HEURÍSTICA DE INCREMENTO RELATIVO

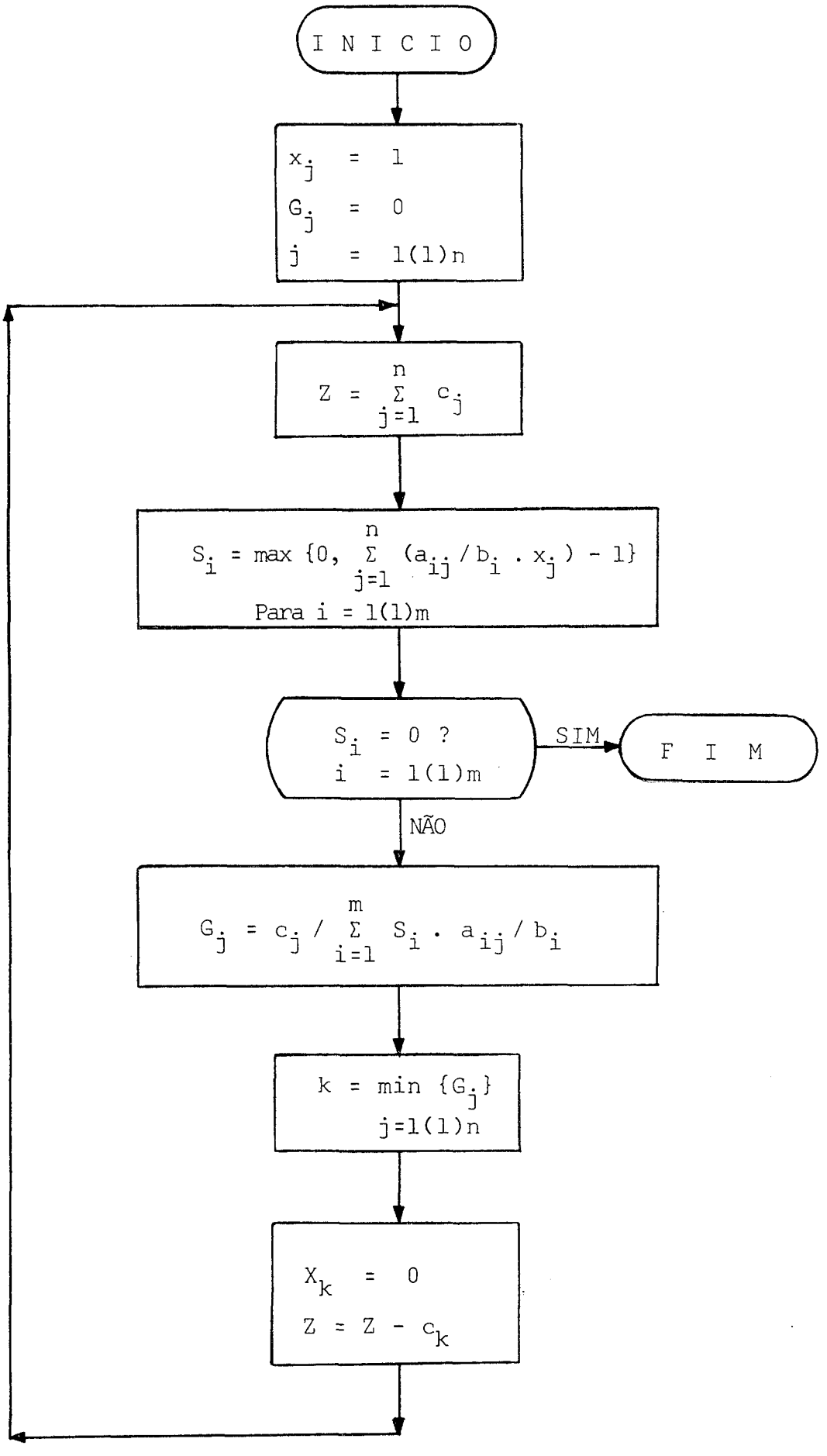


FIGURA 8 - DIAGRAMA DE BLOCOS DA HEURÍSTICA SENJU E TOYODA

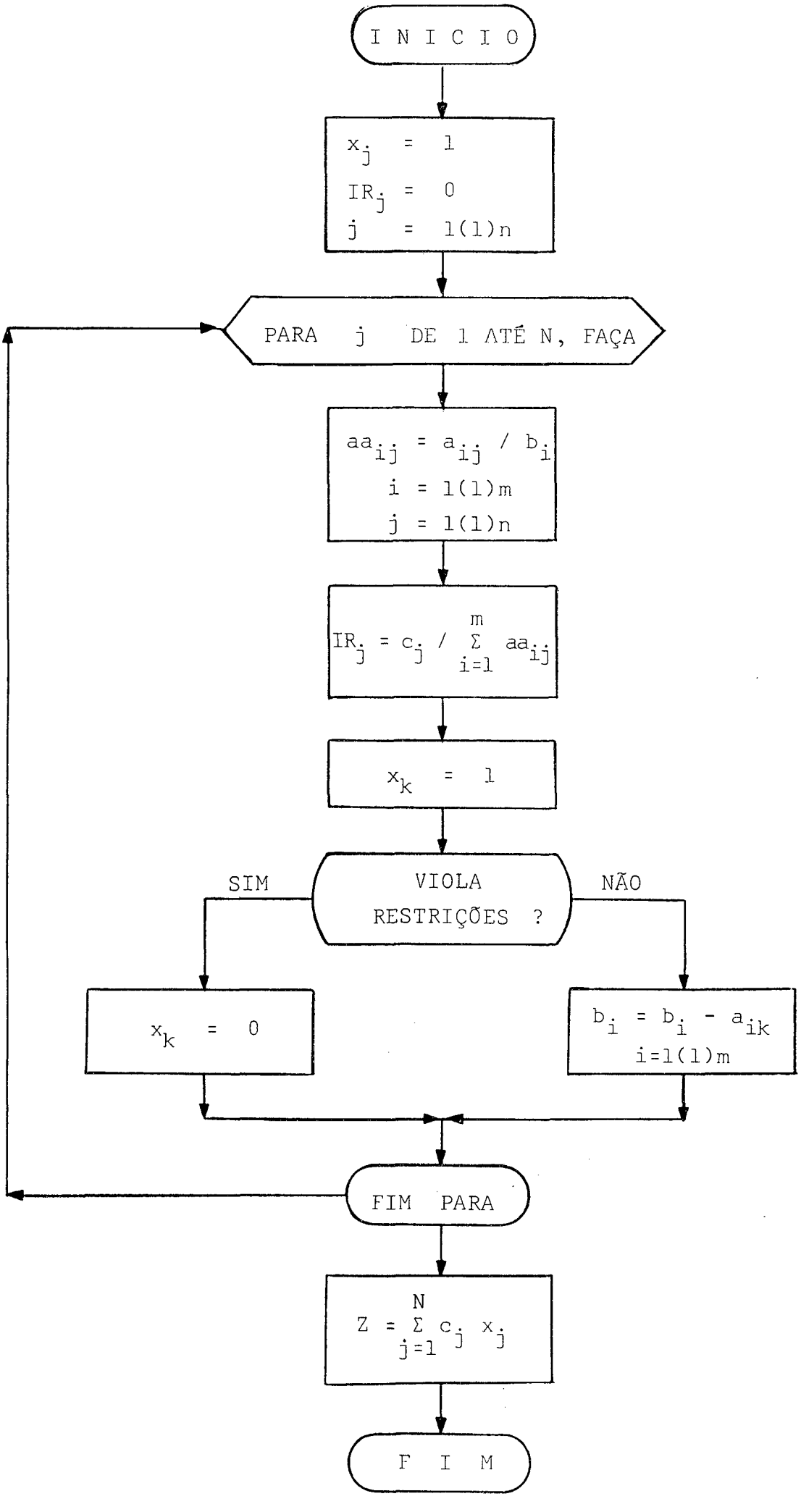


FIGURA 9 - DIAGRAMA DE BLOCOS DA HEURÍSTICA KOCHENBERGER

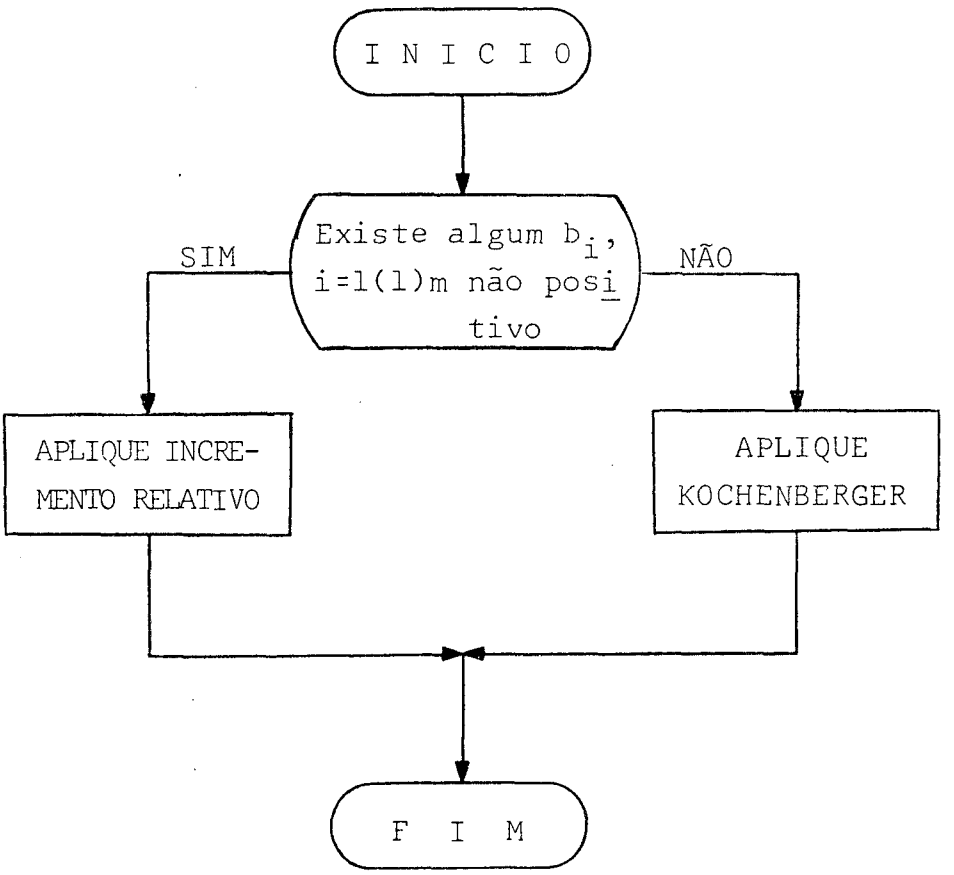


FIGURA 10 - DIAGRAMA DE BLOCOS DA HEURÍSTICA GERAL

4.3. Primeira Fase do Modelo Proposto

O modelo proposto, que a seguir será apresentado, constitui-se em duas fases distintas. Na primeira fase, através da aplicação da heurística geral, vide figura 10, obtém-se uma solução inicial viável, que originará uma série de subproblemas nos quais serão aplicados o algoritmo LEXSM, melhorando, se possível, a solução inicial. Todavia, apesar da solução resultante ser viável, não é garantida a sua otimalidade.

Na segunda fase, aplica-se o algoritmo LEXSM ao problema original, informando-o dos resultados obtidos, na primeira fase. Ao final desta fase obtém-se a solução ótima.

Para aplicação do modelo, será considerado, sem perda de generalidade, que o problema, após as transformações necessárias, esteja na forma padrão (IV), seguindo-se os seguintes passos.

Passo 1:

Aplicação da heurística geral do problema, resultando na obtenção de uma solução viável Y^*.

Substituindo-se Y^* no problema, obtém-se:

a) O valor inicial da função objetivo Z^*, onde

$$Z^* = \sum_{j=1}^{n} cc_j \, y_j^*$$

b) As folgas $BAUX_i$ referente à solução Y^* são obtidas por:

$$BAUX_i = b_i - \sum_{j=1}^{n} aa_{ij}\, y_i^*, \text{ para } i = 1(1)m$$

Passo 2: Montagem dos subproblemas.

Para cada componente y_{I2}^* de Y^*, I2=1(1)n, tal que $y_{I2}^* = 0$, monta-se um subproblema com a rotina a seguir:

a) Seleção das variáveis

Percorre-se a solução $Y^* = (y_1^*,\ldots,y^*)$ em ordem crescente de seu índice, até detectar a primeira componente igual a zero. Seja I2 o índice com a característica acima:

Irão compor este subproblema a variável y_{I2}^* unida com todas as variáveis $y_j^* = 1$, j=1(1)I2-1, obtendo-se o conjunto K dos índices das variáveis que compõem o subproblema.

b) Cálculo dos recursos do subproblema

Com as variáveis y_i, i ε K, definem-se os recursos do subproblema, levando-se em consideração as folgas do problema original decorrente da solução Y^*. O procedimento acima, computacionalmente formulado, é:

$$BAUX_i = \sum_{K} aa_{iK} \cdot y_K + BAUX_i, \text{ para } i=1(1)m.$$

c) Determinação do valor inicial da função objetivo do subproblema

O valor inicial da função objetivo do subproblema I2 é dado por

$$Z^{*}_{I2} = \sum_{K} cc_{iK} \cdot y_K - cc_{I2}$$

passando o subproblema a ter a seguinte forma padrão

$$\text{Max } Z = \sum_{K} cc_K \cdot y_K - Z^{*}_{I2}$$

Sujeito a:

$$\sum_{K} aa_{iK}\, y_K \leq BAUX_i \qquad i = 1(1)m$$

Passo 3:

A cada subproblema montado é aplicado o algoritmo LEXSM. Para cada solução viável advinda da aplicação do algoritmo, aplica-se a transformação T^{-1}, para se obter a solução viável do problema que se deseja resolver.

Quando se repetir o procedimento, para a montagem de outro subproblema, o conjunto Y^{*} é percorrido a partir do último índice I2 acrescido de uma unidade.

Conclui-se esta primeira fase quando o valor do apontador I2 for maior que o número de variáveis do problema.

4.4. Segunda Fase do Modelo Proposto

Nesta segunda fase, aplica-se o algoritmo LEXMS ao problema que se pretende resolver.

Para que as informações obtidas na primeira fase sejam totalmente incorporadas ao modelo, deve-se utilizar o valor da função objetivo já determinado na primeira fase, tornando-se, assim, a restrição proveniente da função objetivo extremamen-

te atuante, como segue:

Seja X^* a solução obtida na fase um. Então, na fase dois, a restrição correspondente à função objetivo será

$g_o(X) - g_o(X^*) \geq 0$, forçando LEXSM a só testar as soluções X, tais que $X \rangle X^*$.

Devido ao próprio desenvolvimento do algoritmo LEXSM, a última solução viável obtida é garantidamente ótima.

4.5. Diagrama de Blocos do Modelo Proposto

Para uma visualização global do modelo proposto, apresenta-se o diagrama de blocos correspondente, conforme figura 11, e uma listagem do programa conforme anexo 3 e manual de uso anexo 4.

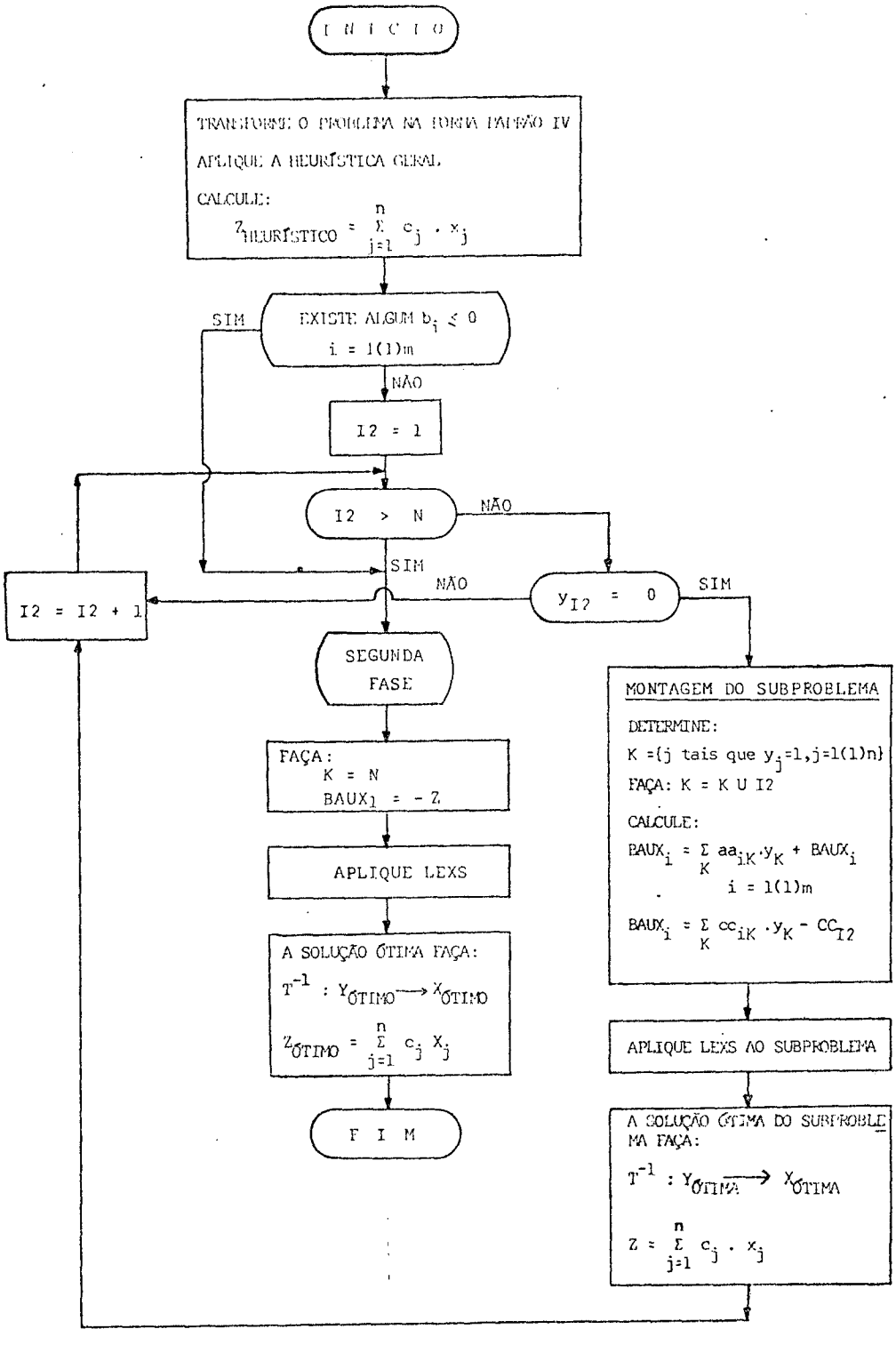


FIGURA 11 - DIAGRAMA DE BLOCOS DO MODELO PROPOSTO

4.6. Análise da Performance Computacional do Modelo

Para facilitar a análise, elaborou-se a tabela 3.

TABELA 3 - ANÁLISE COMPARATIVA DO MODELO PROPOSTO COM O ALGORITMO LEXS

PROBLEMAS	NV	NR	SOLUÇÃO ÓTIMA	MODELO PROPOSTO				LEXS	
				FASE 1		FASE 2			
				$\overline{Z}$	TEMPO	$\overline{Z}$	TEMPO	$\overline{Z}$	TEMPO
1	6	10	3800	3800	1	3800	1	3800	4
2	10	4	70	70	1	70	1	70	5
3	10	7	-23	∞	1	-23	2	-23	82
4	15	10	4015	4015	1	4015	2	4015	21
5	15	15	10	10	1	10	5	10	18
6	15	50	9	9	1	9	136	9	109
7	20	3	26	40	1	26	26	26	25
8	20	10	6120	6120	3	6120	4	6120	371
9	28	10	12400	12400	4	12400	17	11970*	3500
10	31	31	18	19	16	18*	3200	18*	3466
11	39	5	10618	10588	27	10618	1488	9777*	3333
12	50	5	16537	16360	25	16360*	3200	10756*	3226

NOTA:

NV = Número de Variáveis

NR = Número de Restrições

$\overline{Z}$ = Valor Obtido da Função Objetivo

TEMPO = Tempo de CPU em Segundos

* = Não Foram Enumeradas todas as Soluções.

Nota-se através da tabela 3, a eficiência do modelo proposto em sua primeira fase, que tem como principal característica um crescimento linear de tempo de solução, em função do número de variáveis do problema, apresentando um valor da função objetivo em alguns casos ótimo e nos demais com um erro praticamente desprezível.

Na segunda fase, quando se deseja obter a solução ótima, observa-se que o crescimento do tempo de resolução também é linear, para problemas que tenham em torno de trinta variáveis. Para problemas maiores, o crescimento do tempo passa a ser uma função exponencial do número de variáveis.

C A P Í T U L O V

5. CONCLUSÃO

O desenvolvimento de pesquisas e estudos sobre as técnicas lexicográficas, na tentativa de se conseguir modelos computacionalmente viáveis, para a resolução de problemas de programação linear inteira, ainda se encontram num estágio embrionário.

Entretanto, na prática, independente do método utilizado, o conhecimento da solução ótima de programação linear inteira é freqüentemente impraticável, devido aos seguintes aspectos:

a) pelo excessivo tempo e memória requeridos de computador, se o problema é de médio ou grande porte.

b) quando os aspectos anteriores não são considerados, a modelagem do problema real requer algumas abstrações que se farão sentir na solução ótima do modelo, a qual certamente pertencerá a uma vizinhança da solução ótima do problema real.

Tendo em mente as considerações acima, pode-se afirmar que o modelo proposto é dotado de uma potencialidade de abrangência suficientemente ampla para ter aplicabilidade a problemas de qualquer porte.

Em problemas de pequeno porte, como foi exaustivamente demonstrado ao longo deste trabalho, o modelo proposto

tem uma excelente performance, obtendo-se a solução ótima em um tempo muito bom, quando comparado com os tempos de literatura existente.

Finalmente, em problemas de médio e grande porte, quando surgem durações maiores para a obtenção da solução ótima, observa-se que a primeira fase do modelo proposto tem a virtude de trazer uma solução surpreendentemente próxima da ótima, com um erro desprezível, em um curto espaço de tempo. Com esta solução, o usuário dispõe de excelente ferramenta de trabalho e pode optar por ela ou ainda despender mais algum tempo computacional na busca da otimização, bastando para tal que se aplique a segunda fase do modelo proposto.

Entretanto, volta-se a enfatizar que o atingimento da solução ótima pode significar um incremento de informações muito pequeno, tendo em vista as limitações de modelagem do problema, contra um custo que pode tornar-se muito significativo.

BIBLIOGRAFIA CONSULTADA

BALAS, E. An additive algorithm for solving linear programs with zero-one variables. Operations Research, 13(4):517-546, 1965.

_____. Discrete programming by the filter method. Operations Research, 15(5):915-957, 1967.

DRAGAN, Irinel. An algorithme lexicographique pour la resolution des programmes linéaires en variables binaires. Management Science, 16(3):246-252, Nov. 1969.

GLOVER, F. Surrogate constrainsts. Operations Research, 16(4): 741-749, 1968.

HALDI, J. 25 Integer programming test problems. Working Paper nº 43, Graduate School of Business, Stanford University, December 1964.

LAWLER, E. & BELL, M. A method for solving discrete optimization problems. Operations Research, 14(6):1098-1112, 1966.

MACULAN FILHO, Nelson. Programação linear inteira. Rio de Janeiro, COPPE, UFRJ, 1978.

NASCIMENTO, Paulo R. Anotações de aula de pesquisa operacional III. Pós-Graduação Engenharia de Produção, UFSC, 1978.

PETERSEN, Clifford C. Computacional experience with variantes of the balas algorithm applied to the seletion of R&D projects. Management Science, 13(9):736-750, May 1967.

RÖDDER, Wilhelm. Ein lexikographischer suchalgorithmus zur ganzzahligen Programmierung: LEXS. Zor, Bd. 20, 1972. p.209-217.

SALKIN, Harvey M. Integer Programming. Cleveland, Ohio, Addison-Wesley Publishing, 1975.

SIMONNARD, M.A. Programmation linéaire. Paris, Dunod, 1973. v.2.

WALLINGFORD, B.A., & MAO, James C.T. An extesion of Lawler and Bell's method of discrete capital budgeting. Management Science, 15(2): Oct, 1968.

ZANAKIS, Stelios H. Heuristic 0-1 linear programming an experimental comparison of three methods. Management Science, 24(1):91-104, Sep 1977.

ZIONTS, S. & KENDALL, K.E. Solving integer programming problems by aggregating constraints. Operations Research, 25(2); Mar-Apr , 1977.

A N E X O 1

PROGRAMAÇÃO DO ALGORITMO LEXS

DESCRIÇÃO DA FUNÇÃO DE CADA SUB-ROTINA

ATUAL:

A cada solução viável determinada, a sub-rotina atual tem a função de:

- armazenar a solução
- redefinir o valor da função objetivo
- imprimir a solução, o valor da função objetivo e o número de soluções enumeradas explicitamente.

CMJE e CALMJE:

Estas duas sub-rotinas têm a mesma finalidade: determinar o salto j^*. A sub-rotina CMJE somente será usada se todas as variáveis do problema forem binárias; caso contrário, usa-se a sub-rotina CALMJE.

ENTRA:

Realiza a leitura e impressão de todos os dados referentes ao problema que se deseja resolver.

INICIO:

Todas as variáveis do problema serão inicializadas nesta sub-rotina.

LIMITE:

Tem a finalidade de fornecer os limites superior e inferior de cada variável.

PREPA:

Muda todas as restrições da forma $g_i(X) \gtrless b_i$, para a forma $g_i(X) - b_i \geq 0$, com i=1(1)m, e transforma os problemas de minimização em problemas de maximização.

PULO e PULO1:

Estas duas sub-rotinas tem a mesma finalidade: determinar a próxima solução lexicográfica que será enumerada. A sub-rotina PULO somente será usada se todas as variáveis do problema forem binárias; caso contrário, usa-se a sub-rotina PULO1.

SAÍDA:

Têm por função a impressão dos resultados finais, que são:

- valor ótimo da função objetivo
- solução ótima
- número de soluções enumeradas explicitamente
- folgas referentes a solução ótima.

VERIFI:

Determina as restrições que estão sendo violadas para uma determinada solução.

```
*****************************************************************
**                                                             **
      PROGRAMA PRINCIPAL
**                                                             **
*****************************************************************
    IMPLICIT  INTEGER(A-Z)
    COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
    COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,JMJE
    COMMON IMP,LEI
    COMMON LS(55),PO1,NIRUS

    CONTA=0
    NUPRO=12
7755 CONTINUE
    CONTA=CONTA+1
    CALL TIM300(TIME)
    TINICI=TIME
    CALL ENTRA
    CALL PREPA
    CALL INICIO
    IF(PO1.EQ.1)GO TO 1
*********PROGRAMACAO INTEIRA
    CALL LIMITE
 75 CONTINUE
    CALL VERIFI
    IF(K)50,50,60
 50   CONTINUE
      CALL ATUAL
      GO TO 75
 60   CONTINUE
      CALL CMJE
      IF(JENE.EQ.1)GO TO 40
      CALL PULO1
      IF(FIM.EQ.1)GO TO 40
      GO TO 75

********  PROBLEMA ZERO-UM


  1 CONTINUE
    CALL VERIFI
    IF(K)20,20,30
 20   CONTINUE
      CALL ATUAL
      GO TO 1
 30   CONTINUE
      CALL CALMJE
      IF(JENE.EQ.1)GO TO 40
        CALL PULO
      IF(FIM.EQ.1)GO TO 40
        GO TO 1
 40 CONTINUE
    CALL SAIDA
    CALL TIM300(TIME)
    TFINAL=TIME
    TGASTO=(TFINAL-TINICI)/300.
    WRITE(IMP,7758)TGASTO
7758 FORMAT(///,40X,'TEMPO DE CPU UTILIZADO = ',I10,2X, '  SEGUNDOS')
    IF(CONTA.LT.NUPRO)GO TO 7755
    STOP
    END
```

```
**********************************************************************
**                                                                  **
      SUBROUTINE ATUAL
**                                                                  **
**********************************************************************
      IMPLICIT  INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,JMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
           ATUALIZA O VALOR DA FUNCAO OBJETIVA
           ARMAZENA O MELHOR SOLUCAO ENCONTRADA
           Y = MELHOR SOLUCAO ENCONTRADA

      Z=0


      DO 20 J=1,N
         Y(J)=X(J)
         Z=Z  + A(1,J)*X(J)
20    CONTINUE
           PARA FORCAR A SOLUCAO ATUALNAO MAIS SERVIR

      S(1)=-1
      B(1)=-(Z+1)
      DICASO=1

           PARA LISTAR A MELHOR SOLUCAO ATE O MOMENTO

      WRITE(IMP,13)Z,NIRUS,(X(I),I=1,N)
13    FORMAT(2I10,10X,30I3,/,30X,30I3)
      RETURN
      END


**********************************************************************
**                                                                  **
      SUBROUTINE CALMJE
**                                                                  **
**********************************************************************
      IMPLICIT  INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,JMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
           CALCULA E ARMAZENA O MAIOR SALTO

      MJE=0
      DO 25 I=1,K
         LL=RNS(I)
         SA=S(LL)
         DO 20 J=1,N
            IF(A(LL,J))35,45,45
45             CONTINUE
               SA=SA+A(LL,J)*(1-X(J))
               GO TO 85
35             CONTINUE


               SA=SA-A(LL,J) * X(J)
85             CONTINUE
            IF(SA.GE.0)GO TO 95
20       CONTINUE
         JENE=1
         RETURN
95       CONTINUE
         IF(MJE.GT.J)GO TO 25
            MJE=J
25    CONTINUE
      RETURN
            END
```

```
**********************************************************************
**                                                                  **
      SUBROUTINE ENTRA
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,JMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRJS
      DIMENSION NOME(80),NOME1(52),NVP(100)
      DATA NOME1/'   J',' CJ',' A1J',' A2J',' A3J',' A4J',' A5J',' A6J'
     $          ,' A7J',' A8J',' A9J','A10J','A11J','A12J','A13J','A14J'
     $          ,'A15J','A16J','A17J','A18J','A19J','A20J','A21J',
     $          'A22J','A23J','A24J','A25J','A26J','A27J','A28J','A29J'
     *          ,'A30J','A31J','A32J','A33J','A34J','A35J','A36J','A37J'
     *          ,'A38J','A39J','A40J','A41J','A42J','A43J','A44J','A45J'
     *          ,'A46J','A47J','A48J','A49J','A50J'/
      LEI=1
      IMP=3
          NUMERO DE RESTRICOES - M
          NUMERO DE VARIVEIS - N
          TIPO DO PROBLEMA  TP=1 MAXIMIZAR
                            TP=2 MINIMIZAR
          COEFICIENTES DAS RESTRICOES E DA FUNCAO OBJETIVA - A
          DISPONIBILIDADES - B
          TIPO DA RESTRICAO  -  TR = 1 MAIOR OU IGUAL

                                TR = 2 MENOR OU IGUAL
                                TR = 3 IGUAL
          P01=1  ***  PROBLEMA ZERO-UM

      DO 85 J=1,100
           NVP(J)=J
 85   CONTINUE
      READ(LEI,5)NOME
      READ(LEI,10)M,N,TP,P01
      WRITE(IMP,15)NOME
      WRITE(IMP,11)M,N
      KL=M+2
      M=M+1
      DO 25 J=1,N
         READ(LEI,30,ERR=755)(A(I,J),I=1,M)
 25   CONTINUE
      READ(LEI,40)(TR(I),I=1,M)
      TR(1)=1
      READ(LEI,50)(B(I),I=1,M)
      B(1)=99999
      KII=1
      KI=1
      KL=12
      KF=11
      IF(KF.GT.M)GO TO 215
315   CONTINUE
          WRITE(IMP,20)(NOME1(I),I=KII,KL)
          DO 205 J=1,N
               WRITE(IMP,35)J,(A(I,J),I=KI,KF)
205       CONTINUE
          KI=KI+1
          WRITE(IMP,45)(TR(I),I=KI,KF)
          WRITE(IMP,55)(B(I),I=KI,KF)
          IF(KF.EQ.M)GO TO 753
295       CONTINUE
          KII=KL+1
          KL=KL+10
          KI=KF+1
          KF=KF+10
      IF(KF.GT.M)GO TO 415
```

```
425 CONTINUE
      WRITE(IMP,220)(NOME1(I),I=KI,KL)
      DO 275 J=1,N
         WRITE(IMP,235)J,(A(I,J),I=KI,KF)
275   CONTINUE
      WRITE(IMP,245)(TR(I),I=KI,KF)
      WRITE(IMP,255)(B(I),I=KI,KF)
      IF(KF.EQ.M)GO TO 753
   GO TO 295
215 CONTINUE
      KF=M
      KL=M+1
   GO TO 315
415 CONTINUE
      KF=M
      KL=M+1
   GO TO 425
 20 FORMAT(///,12(6X,A4))
 35 FORMAT(/,12I10)
 45 FORMAT(///,1X,'TIPO DAS RESTRICOES',10I10)
 55 FORMAT(///,1X,'VALOR DOS RECURSOS ',10I10)
220 FORMAT(///,9X,'J',10X,10(6X,A4))
235 FORMAT(/,I10,10X,10I10)
245 FORMAT(///,1X,'TIPO DAS RESTRICOES',10I10)
255 FORMAT(///,1X,'VALOR DOS RECURSOS ',10I10)
755 CONTINUE
   WRITE(IMP,785)J
785 FORMAT(10X,'***** ERRO NO ',2X,I2,2X,'CARTAO DE DADOS DA MATRIZ A'
  *)
  5 FORMAT(80A1)
 10 FORMAT(4I2)
 15 FORMAT(1H1,80A1)
 11 FORMAT(///,10X,'NUMERO DE RESTRICOES = ',I5,
  $        ///,10X,'NUMERO DE VARIAVEIS = ',I5)
 30 FORMAT(16I5)
 40 FORMAT(16I5)
 50 FORMAT(16I5)
 60 FORMAT(///,10X,'SOLUCAO DO PROBLEMA',/,10X,19('*'),////,
  *           1X,'ZOTIMOPAR',1X,'NUMEROITE',20X,'SOLUCAO PARCIAL',//,
  *           30X,31I3)


753 CONTINUE
   WRITE(IMP,60)(NVP(J),J=1,N)
   RETURN
   END

*************************************************************************
**                                                                     **
   SUBROUTINE INICIO
**                                                                     **
*************************************************************************
   IMPLICIT INTEGER(A-Z)
   COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
   COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
   COMMON IMP,LEI
   COMMON LS(55),P01,NIPUS
        INICIALIZA AS VARIAVEIS NECESSARIAS DO PROGRAMA

        PARA INICIAR COM O VETOR NULO

   DO 15 J=1,N
      X(J)=0
15 CONTINUE

        VALOR DE CADA RESTRICAO PARA O VETOR NULO

   DO 20 I=1,M
      S(I)=B(I)
20 CONTINUE

        DICASO=BANDEIRA PARA SABER SE EXISTE PELO MENOS UMA SOLUCAO
                              DICASO = 0 - NAO TEM SOLUCAO
                              DICASO = 1 -     TEM SOLUCAO
   DICASO=0

        UMJE - ULTIMO-VALOR MAXIMO DE J-ESTRELA
   UMJE=0

        JENE - PARA DETECTAR SE J-ESTRELA NAO EXISTE
                        JENE=0 - NAO EXISTE
                        JENE=1       EXISTE
```

```
      JENE=0
C        NIRUS - NUMERO DE SOLUCOES TESTADAS
      NIRUS=0
C
C        FIM - INDICA OVERFLOW NO VETOR SOLUCAO
C              FIM = 1 - OVERFLOW
C
      FIM=0
      RETURN
      END
C*****************************************************************
C**                                                             **
      SUBROUTINE PREPA
C**                                                             **
C*****************************************************************
      IMPLICIT  INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
C
C        COLOCA O PROBLEMA NA FORMA PADRAO UTILIZADA POR HOLDI
C
C        TRANSFORMA PROBLEMA DE MINIMIZAR EM MAXIMIZAR
C
      IF(.NOT.(TP.EQ.2))GO TO 50
        DO 15 J=1,N
          A(1,J)=-A(1,J)
15      CONTINUE
      SENAO
50      DO 18 I=2,M
          IF(TR(I).EQ.2)GO TO 60
            B(I)=-B(I)
            GO TO 18
          SENAO
60          DO 25 J=1,N
              A(I,J)=-A(I,J)
25          CONTINUE
C
18      CONTINUE
      RETURN
      END
C*****************************************************************
C**                                                             **
      SUBROUTINE PULO
C**                                                             **
C*****************************************************************
      IMPLICIT INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
C
      IF(MJE.GT.UMJE)GO TO 15
        IF(X(MJE).EQ.0)GO TO 20
50        MJE=MJE+1
          IF(MJE.GT.N)GO TO 30
            IF(X(MJE).EQ.1)GO TO 50
              IF(MJE.GT.UMJE)GO TO 15
20            X(MJE)=1
                DO 38 I=1,M
                  S(I)=S(I)+A(I,MJE)
38              CONTINUE
                IF(MJE.EQ.1)RETURN
                MP=MJE-1
                DO 47 J=1,MP
                  IF(X(J).NE.1)GO TO 47
                    DO 57 I=1,M
                      S(I)=S(I)-A(I,J)
57                  CONTINUE
                    X(J)=0
47              CONTINUE
                RETURN
30          CONTINUE
            FIM=1
            RETURN
15    DO 58 J=1,MJE
        X(J)=0
```

```
68      CONTINUE
        X(MJE)=1
        UMJE=MJE
        DO 78 I=1,M
           S(I)=B(I)+A(I,MJE)
 78     CONTINUE
     RETURN
           END
 ***********************************************************************
 **                                                                   **
     SUBROUTINE SAIDA
 **                                                                   **
 ***********************************************************************
     IMPLICIT INTEGER(A-Z)
     COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
     COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
     COMMON IMP,LEI
     COMMON LS(55),PO1,NIRUS
     IF(DICASO.EQ.0)GO TO 300
          DETERMINAR QUAIS VARIAVEIS PERTENCEM A MELHOR SOLUCAO
          KKK - NUMERO DE VARIAVEIS DIFERENTE DE ZERO DA MELHOR
                SOLUCAO
     KKK=0
     DO 15 J=1,N
        IF(Y(J).EQ.0)GO TO 15
          KKK=KKK+1
          VB(KKK)=J
 15  CONTINUE
     WRITE(IMP,200)(VB(JF),JF=1,KKK)
200  FORMAT(//,20X,'VARIAVEIS BASICAS',//,10X,' J = ',10I5)

     IF(TP.EQ.2)Z=-Z
     WRITE(IMP,400)Z
     WRITE(IMP,115)NIRUS
115  FORMAT(10X,'NUMERO DE SOLUCOES TESTADAS  =  ',I10)
400  FORMAT(//,20X,'SOLUCAO OTIMA - Z = ',I5)
     RETURN
300  WRITE(IMP,100)
100  FORMAT(20X,'PROBLEMA IMPOSSIVEL')
     RETURN


     END

 ***********************************************************************
 **                                                                   **
     SUBROUTINE VERIFI
 **                                                                   **
 ***********************************************************************
     IMPLICIT  INTEGER(A-Z)
     COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
     COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
     COMMON IMP,LEI
     COMMON LS(55),PO1,NIRUS


         K = NUMERO DE RESTRICOES NAO SATISFEITA

     NIRUS=NIRUS+1
     K=1
     RNS(1)=1
     IF(S(1).LT.0)RETURN
               K=0
     CONTINUE
     DO 25 I=2,M
        IF(S(I))10,20,20
 10        CONTINUE
           K=K+1
           RNS(K)=I
 20        CONTINUE
 25  CONTINUE
     RETURN
     END

 ***********************************************************************
 **                                                                   **
     SUBROUTINE PULO1
 **                                                                   **
 ***********************************************************************
     IMPLICIT INTEGER(A-Z)
     COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
     COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
```

```
      COMMON IMP,LEI
      COMMON LS(55),P01,NIPUS
      IF(MJE.GT.UMJE)GO TO 15
        IF(X(MJE).LT.LS(MJE))GO TO 20
50        MJE=MJE+1
          IF(MJE.GT.N)GO TO 30
            IF(X(MJE).EQ.LS(MJE))GO TO 50
              IF(MJE.GT.UMJE)GO TO 15
              X(MJE)=X(MJE)+1
                DO 38 I=1,M
                   S(I)=S(I)+A(I,MJE)
38              CONTINUE
                MP=MJE-1
                DO 47 J=1,MP
                   IF(X(J).EQ.0)GO TO 47
                     DO 57 I=1,M
                        S(I)=S(I)-A(I,J)*X(J)
57                   CONTINUE
                        X(J)=0
47              CONTINUE
                RETURN
30            CONTINUE
              FIM=1
              RETURN
 20       X(MJE)=X(MJE)+1
          DO 18 I=1,M
          S(I)=S(I)+A(I,MJE)
 18        CONTINUE
          RETURN
 15     DO 68 J=1,MJE
          X(J)=0
68       CONTINUE
         X(MJE)=1
         UMJE=MJE
         DO 78 I=1,M
           S(I)=B(I)+A(I,MJE)
 78      CONTINUE
      RETURN
            END


*************************************************************************
**                                                                     **
      SUBROUTINE CMJE
**                                                                     **
*************************************************************************
      IMPLICIT  INTEGER(A-Z)
      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
           CALCULA E ARMAZENA O MAIOR SALTO
      MJE=0
      DO 25 I=1,K
         LL=RNS(I)
         SA=S(LL)
         DO 20 J=1,N
            IF(A(LL,J))35,45,45
 45           CONTINUE
              SA=SA+A(LL,J)*(LS(J)-X(J))
              GO TO 85
 35           CONTINUE
              SA=SA-A(LL,J) * X(J)
 85           CONTINUE
            IF(SA.GE.0)GO TO 95
 20      CONTINUE
         JENE=1
         RETURN
 95      CONTINUE
         IF(MJE.GT.J)GO TO 25
           MJE=J
 25   CONTINUE
      RETURN
            END
```

```
**********************************************************************
**                                                                  **
      SUBROUTINE LIMITE
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-Z)



      COMMON A(55,55),TR(55),B(55),X(55),Y(55),S(55),VB(55),RNS(55)
      COMMON K,JENE,FIM,Z,DICASO,MJE,TP,M,N,UMJE
      COMMON IMP,LEI
      COMMON LS(55),P01,NIRUS
      READ(LEI,5)(LS(I),I=1,N)
    5 FORMAT(16I5)
      RETURN
      END
```

A N E X O 2

OS PROBLEMAS DE TESTES

OS PROBLEMAS DE TESTES:

Foram selecionados doze problemas de programação linear inteira zero-um. Os exemplos mais representativos desta classe são os de seleção de portafólio e correspondem, aos problemas 1,2,4,8,9,11 e 12, que foram apresentados por Petersen[21].

Geralmente, neste tipo de problema deseja-se selecionar alguns de N projetos dados, onde c_j é o lucro estimado para o j-ésimo projeto j=1(1)n. O limite de recurso para o i-ésimo período de tempo i=1(1)m é dado por b_i, e o gasto necessário para realizar o j-ésimo projeto no i-ésimo período de tempo é a_{ij}.

Os problemas 3 e 7 estão propostos no livro INTEGER PROGRAMMING de Salkin[22].

Para finalizar, foram incorporados os problemas clássicos da literatura desenvolvidos por Haldi[23], que correspondem aos problemas 5, 6 e 10.

---

[21] PETERSEN, Clifford C., Computacional..., p.736-780.

[22] SALKIN, Harvey M., Integer programming..., p.235-236.

[23] HALDI, J., 25 integer programming.

*** PROBLEMA 1 ***

NUMERO DE RESTRICOES = 10

NUMERO DE VARIAVEIS = 6

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	100	8	8	3	5	5	5	0	3	5	3
2	600	12	12	6	10	13	13	0	0	2	2
3	1200	13	13	4	8	8	8	0	4	4	4
4	2400	64	75	18	32	42	48	0	0	0	8
5	500	22	22	6	6	6	6	8	8	8	8
6	2000	41	41	4	12	20	20	0	0	4	4
TIPO DAS RESTRICOES		2	2	2	2	2	2	2	2	2	2
VALOR DOS RECURSOS		80	96	20	36	44	48	10	18	22	24

Solução: ZMAX = 3800; x_j = 1(j=2,3,6), outros x_j = 0

*** PROBLEMA 2 ***

NUMERO DE RESTRICOES = 4

NUMERO DE VARIAVEIS = 10

J	CJ	A1J	A2J	A3J	A4J
1	14	12	3	3	0
2	17	54	7	0	0
3	17	6	6	4	3
4	15	6	2	1	1
5	40	30	35	4	2
6	12	6	6	3	3
7	14	48	4	4	4
8	10	36	3	4	2
9	12	18	3	5	7
10	10	30	20	10	10
TIPO DAS RESTRICOES		2	2	2	2
VALOR DOS RECURSOS		50	20	20	15

Solução: ZMAX = 70; x_j = 1(j=1,3,4,6,9), outros x_j = 0

*** PROBLEMA 3 ***

NUMERO DE RESTRICOES = 7

NUMERO DE VARIAVEIS = 10

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J
1	-10	-3	0	-5	5	0	0	8
2	7	-12	1	-3	3	0	9	5
3	-1	8	10	1	-1	4	0	-2
4	12	1	0	0	0	-2	-12	-7
5	-2	0	5	0	0	0	7	1
6	-8	0	-1	0	0	5	-6	0
7	3	0	7	0	0	1	0	-5
8	1	0	1	-2	2	-4	2	0
9	-5	7	0	0	0	2	15	10
10	-3	-2	0	-1	1	0	3	0
TIPO DAS RESTRICOES		2	2	2	2	2	2	2
VALOR DOS RECURSOS		8	13	-6	6	8	12	16

Solução: ZMIN = -23, x_j = 1(j=1,5,6,10),outros x_j = 0

*** PROBLEMA 4 ***

NUMERO DE RESTRICOES = 10

NUMERO DE VARIAVEIS = 15

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	100	8	8	3	5	5	5	0	3	3	3
2	220	24	44	6	9	11	11	0	4	6	8
3	90	13	13	4	6	7	7	1	5	9	9
4	400	80	100	20	40	50	55	10	20	30	35
5	300	70	100	20	30	40	40	4	14	29	29
6	400	80	90	30	40	40	40	10	20	20	20
7	205	45	75	8	16	19	21	0	6	12	16
8	120	15	25	3	5	7	9	6	12	12	15
9	160	28	28	12	18	16	18	0	10	10	10
10	580	90	120	14	24	29	29	6	18	30	30
11	400	130	130	40	60	70	70	32	42	42	42
12	140	32	32	6	16	21	21	3	9	18	20
13	100	20	40	3	11	17	17	0	12	18	18
14	1300	120	160	20	30	30	35	70	100	110	120
15	650	40	40	5	25	25	25	10	20	20	20
TIPO DAS RESTRICOES		2	2	2	2	2	2	2	2	2	2
VALOR DOS RECURSOS		550	700	130	240	280	310	110	205	260	275

Solução: ZMAX = 4015; x_j = 1(j=1,2,4,6,7,9,10,14,15); outros x_j = 0

*** PROBLEMA 5 ***

NUMERO DE RESTRICOES = 15

NUMERO DE VARIAVEIS = 15

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	1	1	0	0	0	1	1	1	0	0	0
2	1	0	1	0	0	1	0	0	1	1	0
3	1	0	0	1	0	0	1	0	1	0	1
4	1	0	0	0	1	0	0	1	0	1	1
5	1	1	1	0	0	0	1	1	1	1	0
6	1	1	0	1	0	1	0	1	1	0	1
7	1	1	0	0	1	1	1	0	0	1	1
8	1	0	1	1	0	1	1	0	0	1	1
9	1	0	1	0	1	1	0	1	1	0	1
10	1	0	0	1	1	0	1	1	1	1	0
11	1	1	1	1	0	0	0	1	0	1	1
12	1	1	1	0	1	0	1	0	1	0	1
13	1	1	0	1	1	1	0	0	1	1	0
14	1	0	1	1	1	1	1	1	0	0	0
15	1	1	1	1	1	0	0	0	0	0	0
TIPO DAS RESTRICOES		1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS		5	5	5	5	5	5	5	5	5	5

J	A11J	A12J	A13J	A14J	A15J
1	1	1	1	0	1
2	1	1	0	1	1
3	1	0	1	1	1
4	0	1	1	1	1
5	0	0	1	1	0
6	0	1	0	1	0
7	1	0	0	1	0
8	0	1	1	0	0
9	1	0	1	0	0
10	1	1	0	0	0
11	1	0	0	0	1
12	0	1	0	0	1
13	0	0	1	0	1
14	0	0	0	1	1
15	1	1	1	1	0
TIPO DAS RESTRICOES	1	1	1	1	1
VALOR DOS RECURSOS	4	4	4	4	5

Solução: ZMIN = 10, x_j = 1(5,6,7,8,9,10,11,12,13,14); outros

x_j = 0

*** PROBLEMA 6 ***

NUMERO DE RESTRICOES = 50

NUMERO DE VARIAVEIS = 15

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	1	0	0	1	1	0	0	1	0	0	1
2	1	0	0	0	1	1	1	0	1	0	0
3	1	1	0	0	0	1	0	1	0	1	0
4	1	1	1	0	0	0	0	0	1	0	1
5	1	0	1	1	0	0	1	0	0	1	0
6	1	1	0	0	0	0	1	0	0	0	0
7	1	0	1	0	0	0	0	1	0	0	0
8	1	0	0	1	0	0	0	0	1	0	0
9	1	0	0	0	1	0	0	0	0	1	0
10	1	0	0	0	0	1	0	0	0	0	1
11	1	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0
15	1	0	0	0	0	0	0	0	0	0	0
TIPO DAS RESTRICOES		1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS		1	1	1	1	1	1	1	1	1	1

J	A11J	A12J	A13J	A14J	A15J	A16J	A17J	A18J	A19J	A20J
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	1	1	0	0	1	0	0	1
7	0	0	0	1	1	1	0	1	0	0
8	1	0	0	0	1	0	1	0	1	0
9	1	1	0	0	0	0	0	1	0	1
10	0	1	1	0	0	1	0	0	1	0
11	1	0	0	0	0	1	0	0	0	0
12	0	1	0	0	0	0	1	0	0	0
13	0	0	1	0	0	0	0	1	0	0
14	0	0	0	1	0	0	0	0	1	0
15	0	0	0	0	1	0	0	0	0	1
TIPO DAS RESTRICOES	1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS	1	1	1	1	1	1	1	1	1	1

J	A21J	A22J	A23J	A24J	A25J	A26J	A27J	A28J	A29J	A30J
1	1	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0
3	0	0	1	0	0	0	0	1	0	0
4	0	0	0	1	0	0	0	0	1	0
5	0	0	0	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	1	1	0	0	1	0	0	1
12	0	0	0	1	1	1	0	1	0	0
13	1	0	0	0	1	0	1	0	1	0
14	1	1	0	0	0	0	0	1	0	1
15	0	1	1	0	0	1	0	0	1	0
TIPO DAS RESTRICOES	1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS	1	1	1	1	1	1	1	1	1	1

J	A31J	A32J	A33J	A34J	A35J	A36J	A37J	A38J	A39J	440J
1	1	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0
3	0	0	1	0	0	0	0	1	0	0
4	0	0	0	1	0	0	0	0	1	0
5	0	0	0	0	1	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	0
13	0	0	1	0	0	0	0	0	0	0
14	0	0	0	1	0	0	0	0	0	0
15	0	0	0	0	1	0	0	0	0	0
TIPO DAS RESTRICOES	1	1	1	1	1	2	2	2	2	2
VALOR DOS RECURSOS	1	1	1	1	1	1	1	1	1	1

J	A41J	A42J	A43J	A44J	A45J	A46J	A47J	A48J	A49J	A50J
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	1
TIPO DAS RESTRICOES	2	2	2	2	2	2	2	2	2	2
VALOR DOS RECURSOS	1	1	1	1	1	1	1	1	1	1

Solução: ZMIN = 9, $x_j = 1(1,2,3,4,5,6,7,8,9)$, outros $x_j = 0$

*** PROLBEMA 7 ***

NUMERO DE RESTRICOES = 3

NUMERO DE VARIAVEIS = 20

J	CJ	A1J	A2J	A3J
1	3	-6	-1	3
2	2	5	3	6
3	5	-3	3	1
4	8	-3	4	-3
5	6	0	1	-5
6	9	-1	0	6
7	11	-3	4	-9
8	4	-8	-1	6
9	5	-9	-6	3
10	6	3	0	-9
11	11	-8	8	-6
12	2	6	0	-3
13	8	-3	1	-6
14	5	-8	-5	-6
15	8	-6	-4	6
16	7	7	-1	2
17	3	6	-9	-7
18	9	-2	-7	-6
19	2	3	2	0
20	4	7	2	7
TIPO DAS RESTRICOES		2	2	2
VALOR DOS RECURSOS		-21	-10	-14

Solução: ZMIN = 26, x_j = 1(3,9,13,14,17),outros x_j = 0

*** PROBLEMA 8 ***

NUMERO DE RESTRICOES = 10

NUMERO DE VARIAVEIS = 20

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	100	8	8	3	5	5	5	0	3	3	3
2	220	24	44	6	9	11	11	0	4	6	8
3	90	13	13	4	6	7	7	1	5	9	9
4	400	80	100	20	40	50	55	10	20	30	35
5	300	70	100	20	30	40	40	4	14	29	29
6	400	80	90	30	40	40	40	10	20	20	20
7	205	45	75	8	16	19	21	0	6	12	16
8	120	15	25	3	5	7	9	6	12	12	15
9	160	28	28	12	18	18	18	0	10	10	10
10	580	90	120	14	24	29	29	6	18	30	30
11	400	130	130	40	60	70	70	32	42	42	42
12	140	32	32	6	16	21	21	3	9	18	20
13	100	20	40	3	11	17	17	0	12	18	18
14	1300	120	160	20	30	30	35	70	100	110	120
15	650	40	40	5	25	25	25	10	20	20	20
16	320	30	60	0	10	15	20	0	5	15	20
17	480	20	55	5	13	25	25	0	6	18	22
18	80	6	10	3	5	5	5	0	4	7	7
19	60	3	6	0	1	1	2	0	1	2	3
20	2550	180	240	20	80	100	110	0	20	40	50
TIPO DAS RESTRICOES		2	2	2	2	2	2	2	2	2	2
VALOR DOS RECURSOS		550	700	130	240	280	310	110	205	260	275

Solução: ZMAX = 6120; x_j = 1(j=1,10,14,15,16,17,18,19,20); outros x_j = 0

*** PROBLEMA 9 ***

NUMERO DE RESTRICOES = 10

NUMERO DE VARIAVEIS = 28

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	100	8	8	3	5	5	5	0	3	3	3
2	220	24	44	6	9	11	11	0	4	6	8
3	90	13	13	4	6	7	7	1	5	9	5
4	400	80	100	20	40	50	55	10	20	30	35
5	300	70	100	20	30	40	40	4	14	29	29
6	400	80	90	30	40	40	40	10	20	20	20
7	205	45	75	8	16	19	21	0	6	12	16
8	120	15	25	3	5	7	9	6	12	12	15
9	160	28	28	12	18	18	18	0	10	10	10
10	580	90	120	14	24	29	29	6	18	30	30
11	400	130	130	40	60	70	70	32	42	42	42
12	140	32	32	6	16	21	21	3	9	18	20
13	100	20	40	3	11	17	17	0	12	18	18
14	1300	120	160	20	30	30	35	70	100	110	120
15	650	40	40	5	25	25	25	10	20	20	20
16	320	30	60	0	10	15	20	0	5	15	20
17	480	20	55	5	13	25	25	0	6	18	22
18	80	6	10	3	5	5	5	0	4	7	7
19	60	3	6	0	1	1	2	0	1	2	3
20	2550	180	240	20	80	100	110	0	20	40	50
21	3100	220	290	30	60	70	70	30	50	60	60
22	1100	50	80	40	50	55	55	10	30	50	55
23	950	30	90	10	20	20	20	0	5	25	25
24	450	50	70	0	30	50	50	10	20	15	30
25	300	12	27	5	10	15	20	10	20	25	25
26	220	5	17	0	5	15	15	5	10	15	15
27	200	8	8	0	3	6	6	0	10	10	10
28	520	18	28	10	20	20	20	10	20	28	28
TIPO DAS RESTRICOES		2	2	2	2	2	2	2	2	2	2
VALOR DOS RECURSOS		930	1210	272	462	532	572	240	400	470	490

Solução: ZMAX = 12400; x_j = 1(j=1,2,3,9,14,15,16,17,18,19,20,21,22,23,25,26,27,28);
outros x_j = 0

*** PROBLEMA 10 ***

NUMERO DE RESTRICOES = 31

NUMERO DE VARIAVEIS = 31

J	CJ	A1J	A2J	A3J	A4J	A5J	A6J	A7J	A8J	A9J	A10J
1	1	1	0	1	0	1	0	1	0	1	0
2	1	0	1	1	0	0	1	1	0	0	1
3	1	1	1	0	0	1	1	0	0	1	1
4	1	0	0	0	1	1	1	1	0	0	0
5	1	1	0	1	1	0	1	0	0	1	0
6	1	0	1	1	1	1	0	0	0	0	1
7	1	1	1	0	1	0	0	1	0	1	1
8	1	0	0	0	0	0	0	0	1	1	1
9	1	1	0	1	0	1	0	1	1	0	1
10	1	0	1	1	0	0	1	1	1	1	0
11	1	1	1	0	0	1	1	0	1	0	0
12	1	0	0	0	1	1	1	1	1	1	1
13	1	1	0	1	1	0	1	0	1	0	1
14	1	0	1	1	1	1	0	0	1	1	0
15	1	1	1	0	1	0	0	1	1	0	0
16	1	0	0	0	0	0	0	0	0	0	0
17	1	1	0	1	0	1	0	1	0	1	0
18	1	0	1	1	0	0	1	1	0	0	1
19	1	1	1	0	0	1	1	0	0	1	1
20	1	0	0	0	1	1	1	1	0	0	0
21	1	1	0	1	1	0	1	0	0	1	0
22	1	0	1	1	1	1	0	0	0	0	1
23	1	1	1	0	1	0	0	1	0	1	1
24	1	0	0	0	0	0	0	0	1	1	1
25	1	1	0	1	0	1	0	1	1	0	1
26	1	0	1	1	0	0	1	1	1	1	0
27	1	1	1	0	0	1	1	0	1	0	0
28	1	0	0	0	1	1	1	1	1	1	1
29	1	1	0	1	1	0	1	0	1	0	1
30	1	0	1	1	1	1	0	0	1	1	0
31	1	1	1	0	1	0	0	1	1	0	0
TIPO DAS RESTRICOES		1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS		10	10	9	10	9	9	8	10	9	9

J	A11J	A12J	A13J	A14J	A15J	A16J	A17J	A18J	A19J	A20J
1	1	0	1	0	1	0	1	0	1	0
2	1	0	0	1	1	0	0	1	1	0
3	0	0	1	1	0	0	1	1	0	0
4	0	1	1	1	1	0	0	0	0	1
5	1	1	0	1	0	0	1	0	1	1
6	1	1	1	0	0	0	0	1	1	1
7	0	1	0	0	1	0	1	1	0	1
8	1	1	1	1	1	0	0	0	0	0
9	0	1	0	1	0	0	1	0	1	0
10	0	1	1	0	0	0	0	1	1	0
11	1	1	0	0	1	0	1	1	0	0
12	1	0	0	0	0	0	0	0	0	1
13	0	0	1	0	1	0	1	0	1	1
14	0	0	0	1	1	0	0	1	1	1
15	1	0	1	1	0	0	1	1	0	1
16	0	0	0	0	0	1	1	1	1	1
17	1	0	1	0	1	1	0	1	0	1
18	1	0	0	1	1	1	1	0	0	1
19	0	0	1	1	0	1	0	0	1	1
20	0	1	1	1	1	1	1	1	1	0
21	1	1	0	1	0	1	0	1	0	0
22	1	1	1	0	0	1	1	0	0	0
23	0	1	0	0	1	1	0	0	1	0
24	1	1	1	1	1	1	1	1	1	1
25	0	1	1	0	0	1	0	1	0	1
26	0	1	1	0	0	1	1	0	0	1
27	1	1	0	0	1	1	0	0	1	1
28	1	0	0	0	0	1	1	1	1	0
29	0	0	1	0	1	1	0	1	0	0
30	0	0	0	1	1	1	1	0	0	0
31	1	0	1	1	0	1	0	0	1	0
TIPO DAS RESTRICOES	1	1	1	1	1	1	1	1	1	1
VALOR DOS RECURSOS	8	9	8	8	7	10	9	9	8	9

J	A21J	A22J	A23J	A24J	A25J	A26J	A27J	A28J	A29J	A30J	A31J
1	1	0	1	0	1	0	1	0	1	0	1
2	0	1	1	0	0	1	1	0	0	1	1
3	1	1	0	0	1	1	0	0	1	1	0
4	1	1	1	0	0	0	0	1	1	1	1
5	0	1	0	0	1	0	1	1	1	0	0
6	1	0	0	0	0	1	1	1	1	0	0
7	0	0	1	0	1	1	0	1	0	0	1
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	1	1	0	1	0	1	0	1	0
10	0	1	1	1	1	0	0	1	1	0	0
11	1	1	0	1	0	0	1	1	0	0	1
12	1	1	1	1	1	1	1	0	0	0	0
13	0	1	0	1	0	1	0	0	1	0	1
14	1	0	0	1	1	0	0	0	0	1	1
15	0	0	1	1	0	0	1	0	1	1	0
16	1	1	1	1	1	1	1	1	1	1	1
17	0	1	0	1	0	1	0	1	0	1	0
18	1	0	0	1	1	0	0	1	1	0	0
19	0	0	1	1	0	0	1	1	0	0	1
20	0	0	0	1	1	1	1	0	0	0	0
21	1	0	1	1	0	1	0	0	1	0	1
22	0	1	1	1	1	0	0	0	0	1	1
23	1	1	0	1	0	0	1	0	1	1	0
24	1	1	1	0	0	0	0	0	0	0	0
25	0	1	0	0	1	0	1	0	1	0	1
26	1	0	0	0	0	1	1	0	0	1	1
27	0	0	1	0	1	1	0	0	1	1	0
28	0	0	0	0	0	0	0	1	1	1	1
29	1	0	1	0	1	0	1	1	0	1	0	1
30	0	1	1	0	0	1	1	1	1	0	0
31	1	1	0	0	1	1	0	1	0	0	1
DAS RESTRICOES	1	1	1	1	1	1	1	1	1	1	1
LOR DOS RECURSOS	8	8	7	9	8	8	7	8	7	7	6

Solução: ZMIN = 18, 1(3,5,7,9,11,13,14,15,17,19,21,22,23,25,26,27,29,31),
outros $x_j = 0$

ou

Solução: ZMIN = 18, 1(3,5,6,9,10,13,14,15,17,18,21,22,23,25,26,27,29,30),
outros $x_j = 0$

*** PROBLEMA 11 ***

NUMERO DE RESTRICOES = 5

NUMERO DE VARIAVEIS = 39

J	CJ	A1J	A2J	A3J	A4J	A5J
1	560	40	16	38	8	36
2	1125	91	92	39	71	52
3	300	10	41	32	30	30
4	620	30	16	71	60	42
5	2100	160	150	60	200	170
6	431	20	23	26	13	7
7	68	3	4	5	6	7
8	328	12	18	40	30	20
9	47	3	6	8	4	0
10	122	18	0	12	3	3
11	322	9	12	30	31	21
12	196	25	8	15	6	4
13	41	1	2	0	3	1
14	25	1	1	1	0	2
15	425	10	0	23	18	14
16	4260	280	200	100	60	310
17	416	10	20	0	21	8
18	115	3	6	20	4	4
19	82	1	2	3	0	6
20	22	1	1	0	2	1
21	631	49	70	40	32	18
22	132	8	9	6	15	15
23	420	21	22	8	31	36
24	86	6	4	0	2	10
25	42	1	1	6	2	4
26	103	5	5	4	7	8
27	215	10	10	22	3	6
28	81	3	6	4	2	0
29	91	2	4	6	8	0
30	26	1	0	1	0	3
31	49	0	4	5	2	0
32	420	10	12	14	6	10
33	316	42	8	8	6	6
34	72	6	4	2	7	1
35	71	4	3	3	1	0
36	49	3	0	0	0	0
37	108	0	10	20	0	3
38	116	10	0	0	20	5
39	90	1	6	0	8	4
TIPO DAS RESTRICOES		2	2	2	2	2
VALOR DOS RECURSOS		600	500	500	500	600

Solução: ZMAX = 10618; x_j = 1(j=1,2,4,6,8,9,11,13,15,16,17,18,19,20,23,25,27,28,29,31,32,34,35,36,37,38,39); outros x_j = 0

*** PROBLEMA 12 ***

NUMERO DE RESTRICOES = 5

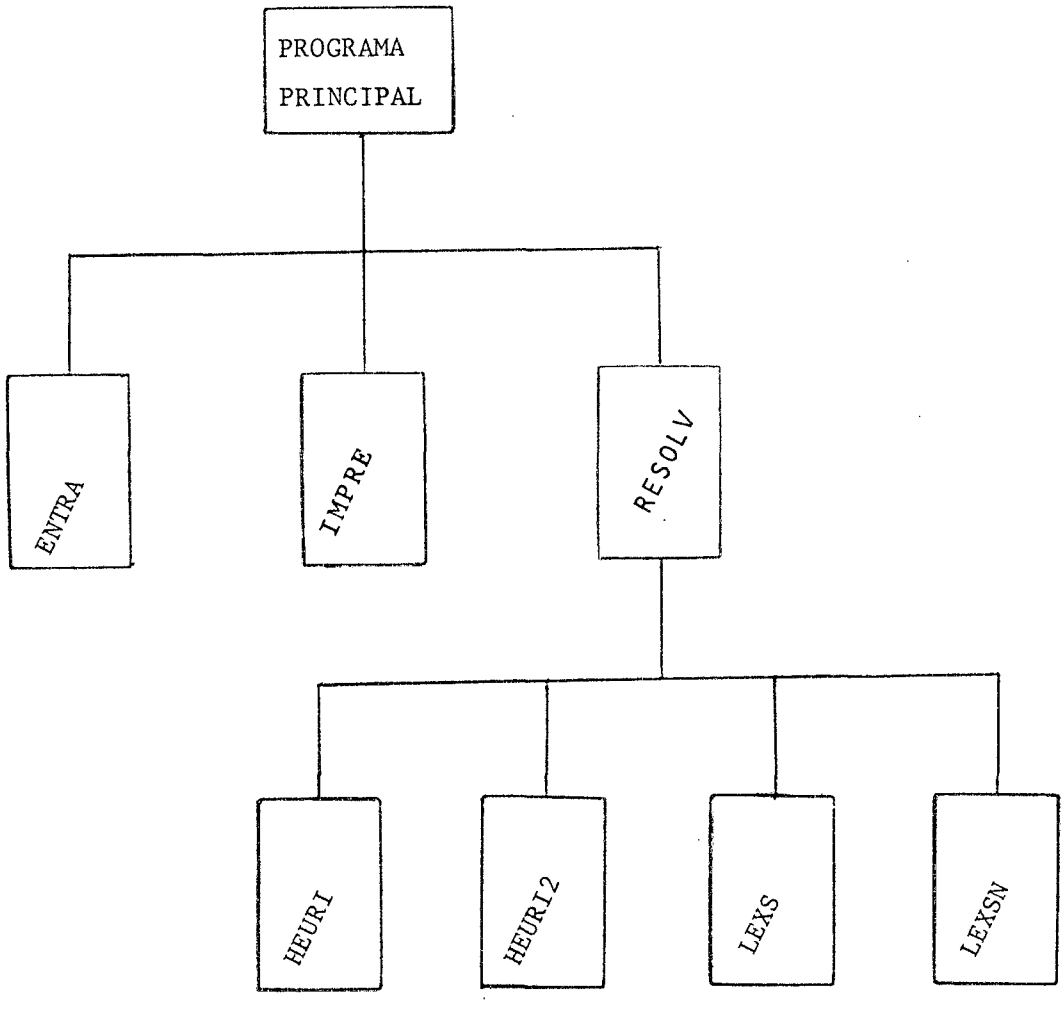
NUMERO DE VARIAVEIS = 50

J	CJ	A1J	A2J	A3J	A4J	A5J
1	560	40	16	38	8	38
2	1125	91	92	39	71	52
3	300	10	41	32	30	30
4	620	30	16	71	60	42
5	2100	160	150	80	200	170
6	431	20	23	26	18	9
7	68	3	4	5	6	7
8	328	12	18	40	30	20
9	47	3	6	8	4	0
10	122	18	0	12	8	3
11	322	9	12	30	31	21
12	196	25	8	15	6	4
13	41	1	2	0	3	1
14	25	1	1	1	0	2
15	425	10	0	23	18	14
16	4260	280	200	100	60	310
17	416	10	20	0	21	8
18	115	8	6	20	4	4
19	82	1	2	3	0	6
20	22	1	1	0	2	1
21	631	49	70	40	32	18
22	132	8	9	6	15	15
23	420	21	22	7	31	38
24	86	6	4	0	2	10
25	42	1	1	6	2	4
26	103	5	5	4	7	8
27	215	10	10	22	8	6
28	81	6	6	4	2	0
29	91	2	4	6	8	0
30	26	1	0	1	0	3
31	49	0	4	5	2	0
32	420	10	12	14	8	10
33	316	42	8	8	6	6
34	72	6	4	2	7	1
35	71	4	3	8	1	3
36	49	8	0	0	0	0
37	108	0	10	20	0	8
38	116	10	0	0	20	5
39	90	1	6	0	8	4
40	738	40	28	6	14	0
41	1811	86	93	12	20	30
42	430	11	9	6	2	12
43	3060	120	30	80	40	16
44	215	8	22	13	6	18
45	58	3	0	6	1	3
46	296	32	36	22	14	16
47	620	28	45	14	20	22
48	418	13	13	0	12	30
49	47	2	2	1	0	4
50	81	4	2	2	1	0
TIPO DAS RESTRICOES		2	2	2	2	2
VALOR DOS RECURSOS		800	650	550	550	650

Solução: ZMAX = 16537; x_j = 1(j=4,6,8,9,11,12,13,15,16,17,19, 20,23,25,26,27,28,29,31,32,34,35,36,37,38,39,40,41,42,43, 44,47,48,49,50); outros x_j = 0

A N E X O 3

PROGRAMAÇÃO DO MODELO PROPOSTO


PROGRAMA
PRINCIPAL
ENTRA
IMPRE
RESOLV
HEURI
HEURI2
LEXS
LEXSN

DESCRIÇÃO DA FUNÇÃO DE CADA SUB-ROTINA

ENTRA:

Realiza a leitura de todos os dados referentes aos problema que se deseja resolver.

IMPRE:

Realiza a impressão de todos os dados do problema que será resolvido.

HEURI e HEURI2:

Estas duas sub-rotinas têm a mesma finalidade: determinar um valor inicial da função objetivo. A sub-rotina HEURI sempre será usada quando todos os valores dos recursos forem positivos; caso contrário, usa-se a sub-rotina HEURI2.

LEXS e LEXSN:

Ambas têm a mesma finalidade:

- Ordenar as restrições por sensibilidade
- Determinar o vetor SOFO
- Resolver o problema utilizando o algoritmo LEXSN.

A sub-rotina LEXS sempre será aplicada aos problemas que possuirem todos os recursos positivos; caso contrário, aplica-se a sub-rotina LEXSN.

RESOLV:

Esta sub-rotina tem por finalidade transformar:

- todos $c_j < 0$ em $cc_j > 0$.
- todas as restrições $g_i(X) \gtrless b_i$ em $g_i(X) - b_i \leq 0$
- problemas de minimização em problemas de maximização.

Se o problema transformado possuir todos os recursos positivos, seguem-se os seguintes passos:

- aplica-se a sub-rotina HEURI
- montam-se os subproblemas
- aplica-se a sub-rotina LEXS ao problema transformado.

Caso contrário:

- aplica-se a sub-rotina HEURI2
- aplica-se a sub-rotina LEXSM ao problema transformado.

```
**********************************************************************
**                                                                  **
                    PROGRAMA PRINCIPAL
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      CONTA=0
      READ(1,1)NUPRO
7755  CONTINUE
        CALL ENTRA
        CALL IMPRE
        CALL RESOLV
      CONTA=CONTA+1
      IF(CONTA.LT.NUPRO)GO TO 7755
      STOP
    1 FORMAT(I5)
      END
**********************************************************************
**                                                                  **
      SUBROUTINE ENTRA
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION NOME(80),NOME1(52),NVP(100)
      LEI=1
      IMP=3
      READ(LEI,5)NOME
      READ(LEI,10)M,N,TP,PO
      WRITE(IMP,15)NOME
      WRITE(IMP,11)M,N
      M=M+1
      DO 25 J=1,N
         READ(LEI,30,ERR=755)(A(I,J),I=1,M)
   25 CONTINUE
      READ(LEI,40)(TR(I),I=1,M)
      READ(LEI,50)(B(I),I=1,M)
      I2=N
      RETURN
  755 CONTINUE
      WRITE(IMP,785)J
      STOP
    5 FORMAT(80A1)
   10 FORMAT(4I2)
   15 FORMAT(1H1,80A1)
   11 FORMAT(//,10X,'NUMERO DE RESTRICOES = ',I5,
     *       //,10X,'NUMERO DE VARIAVEIS  = ',I5)
   30 FORMAT(16I5)
   40 FORMAT(16I5)
   50 FORMAT(16I5)
  785 FORMAT(10X,'***** ERRO NO ',2X,I2,2X,'CARTAO DE DADOS DA MATRIZ A'
     *)
      END
```

```
**  ************************************************************************
**                                                                        **
      SUBROUTINE IMPRE
**                                                                        **
**  ************************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),

     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION NOME1(52)
      DATA NOME1/'    J',' CJ',' A1J',' A2J',' A3J',' A4J',' A5J',' A6J'
     $          ,' A7J',' A8J',' A9J','A10J','A11J','A12J','A13J','A14J'
     $          ,'A15J','A16J','A17J','A18J','A19J','A20J','A21J',
     *           'A22J','A23J','A24J','A25J','A26J','A27J','A28J','A29J'
     *          ,'A30J','A31J','A32J','A33J','A34J','A35J','A36J','A37J'
     *          ,'A38J','A39J','A40J','A41J','A42J','A43J','A44J','A45J'
     *          ,'A46J','A47J','A48J','A49J','A50J'/
      KII=1
      KI=1
      KL=12
      KF=11
      IF(KF.GT.M)GO TO 215
315   CONTINUE
        WRITE(IMP,20)(NOME1(I),I=KII,KL)
        DO 205 J=1,I2
              WRITE(IMP,35)J,(A(I,J),I=KI,KF)
205     CONTINUE
        KI=KI+1
        WRITE(IMP,45)(TR(I),I=KI,KF)
        WRITE(IMP,55)(B(I),I=KI,KF)
        IF(KF.EQ.M)GO TO 753
295     CONTINUE
        KII=KL+1
        KL=KL+10
        KI=KF+1
        KF=KF+10
      IF(KF.GT.M)GO TO 415
425   CONTINUE
        WRITE(IMP,220)(NOME1(I),I=KII,KL)
        DO 275 J=1,I2
              WRITE(IMP,235)J,(A(I,J),I=KI,KF)
275     CONTINUE
        WRITE(IMP,245)(TR(I),I=KI,KF)
        WRITE(IMP,255)(B(I),I=KI,KF)
        IF(KF.EQ.M)GO TO 753



      GO TO 295
215   CONTINUE
        KF=M
        KL=M+1
      GO TO 315
415   CONTINUE
        KF=M
        KL=M+1
      GO TO 425
110   CONTINUE
753   CONTINUE
      WRITE(IMP,15)
      RETURN
 15   FORMAT(/////,10X,'VALOR PARCIAL DA FO',8X,'VARIAVEIS BASICAS',//)
 20   FORMAT(//,2(6X,A4),10(1X,A4))
 35   FORMAT(2I10,10I5)
 45   FORMAT(/,1X,'TIPO DAS RESTRICOES',10I5)
 55   FORMAT(/,1X,'VALOR DOS RECURSOS ',10I5)
220   FORMAT(///,9X,'J',10X,10(1X,A4))
235   FORMAT(I10,10X,10I5)
245   FORMAT(/,1X,'TIPO DAS RESTRICOES',10I5)
255   FORMAT(/,1X,'VALOR DOS RECURSOS ',10I5)
      END
```

```
**  ********************************************************************
**                                                                      **
      SUBROUTINE HEURI
**                                                                      **
**  ********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION G(55),GSC(55)
      VAFUOB=0
      DO 217 J=1,N
             G(J)=0.
             X(J)=0
217   CONTINUE
      DO 213 JJ=1,N
              DO 273 I=2,M
                     IF(BAUX(I))613,613,273
273           CONTINUE
              DO 201 J=1,N
                     IF(G(J).EQ.-1.)GO TO 201
                     GSC(J)=0
                     DO 203 I=2,M
                        IF(A(I,J))203,203,215
215                     CONTINUE
                           GSC(J)=GSC(J)+FLOAT(A(I,J))/BAUX(I)
203                  CONTINUE

                     IF(GSC(J))225,227,229
225                  CONTINUE
                       WRITE(IMP,235)
                       STOP
227                  CONTINUE
                       G(J)=99999
                     GO TO 201
229                CONTINUE
                     G(J)=A(1,J)/GSC(J)
201           CONTINUE
              K=1
              DO 205 J=2,N
                     IF(G(K).LT.G(J))K=J
205           CONTINUE
              G(K)=-1.
              DO 207 I=2,M
                     BAUX(I)=BAUX(I)-A(I,K)
                     IF(BAUX(I))209,207,207
207           CONTINUE
              X(K)=1
              VAFUOB=VAFUOB+A(1,K)
              GO TO 213
209           CONTINUE
              DO 211 J=2,I
                     BAUX(J)=BAUX(J)+A(J,K)
211           CONTINUE
213   CONTINUE
613   CONTINUE
      IF(VAFUOB.EQ.0)RETURN
        VAUX=VAFUOB+VAFO
      IF(J1.EQ.0)GO TO 91
      DO 66 J=1,J1
             K=VT(J)
             X(K)=1-X(K)
 66   CONTINUE
 91   CONTINUE
      R1=0
      DO 85 J=1,N
             IF(X(J))85,85,83
 83            CONTINUE

               R1=R1+1
               VB(R1)=J
 85   CONTINUE
      WRITE(IMP,5455)VAUX,(VB(J),J=1,R1)
      RETURN
5455  FORMAT(/,15X,I6,15X,30I3,/,36X,30I3)
 235  FORMAT(10X,'A T E N C O A ** ERRO DE CORRENTE DO ALGORITMO ')
      END
```

```
**********************************************************************
**                                                                  **
      SUBROUTINE HEURI2
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION G(55),SC(55),BB(55)
      VAFUOB=0
      DO 1 J=1,N
      X(J)=1
      VAFUOB=VAFUOB+A(1,J)
    1 CONTINUE
      DO 5 I=2,M
           BB(I)=-B(I)
           DO 3 J=1,N
                BB(I)=BB(I)+A(I,J)
    3      CONTINUE
    5 CONTINUE
      DO 7 I=2,M
           IF(BB(I))7,7,9
    7 CONTINUE
      GO TO 49
    9 CONTINUE
      DO 10 J=1,N
           SC(J)=0
           DO 15 I=2,M

                IF(A(I,J).LE.0)GO TO 15
                  SC(J)=SC(J)+A(I,J)
   15 CONTINUE
           IF(SC(J))13,13,14
   14        CONTINUE
                G(J)=FLOAT(A(1,J))/SC(J)
             GO TO 10
   13        CONTINUE
                G(J)=1000000
   10 CONTINUE
      DO 25 JJ=1,N
           MENOR=1
           DO 17 J=2,N
                IF(G(J).LT.G(MENOR))MENOR=J
   17      CONTINUE
           G(MENOR)=1100000
           VAFUOB=VAFUOB-A(1,MENOR)
           X(MENOR)=0
           DO 19 I=2,M
                BB(I)=BB(I)-A(I,MENOR)
   19      CONTINUE
           DO 21 I=2,M
                IF(BB(I))21,21,25
   21      CONTINUE
      GO TO 49
   25 CONTINUE
   49 CONTINUE
      IF(VAFUOB.EQ.0)RETURN
        VAUX=VAFUOB+VAFO
      IF(J1.EQ.0)GO TO 91
      DO 66 J=1,J1
           K=VT(J)
           X(K)=1-X(K)
   66 CONTINUE
   91 CONTINUE
      R1=0
      DO 85 J=1,N
           IF(X(J))85,85,83
   83        CONTINUE
             R1=R1+1

             VB(R1)=J
   85 CONTINUE
      IF(TP.EQ.2)VAUX=-VAUX
      WRITE(IMP,5455)VAUX,(VB(J),J=1,R1)
      RETURN
 5455 FORMAT(/,15X,I6,15X,30I3,/,36X,30I3)
      END
```

```
**  ***********************************************************************
**                                                                       **
      SUBROUTINE RESOLV
**                                                                       **
**  ***********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON  A(55,55),AA(55,55),
     *        B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *        S(55),TR(55),VB(55),X(55),Y(55),
     *        I1,I2,I3,IMP,LEI,
     *        M,N,NIRUS,PO,POA,
     *        TP,VAFUO8,Z
      COMMON  VT(55),VAFO,J1,NUPRO,CHAVE
      CALL TIM300(TIME)
      DO 97 J=1,N
            IVP(J)=0
            ICO(J)=0


          X(J)=0
 97   CONTINUE
      VAFUO8=0
      POA=0
      TINICI=TIME
            TP=2 * PROBLEMA DE MINIMIZAR *
      IF(TP-2)3,25,3
 25       CONTINUE
            DO 2 J=1,N
                  A(1,J)=-A(1,J)
  2         CONTINUE
  3       CONTINUE
            TR=1   * RESTRICAO MAIOR OU IGUAL
      DO 6 I=2,M
      IF(TR(I)-1)6,7,6
  7           DO 4 J=1,N
                  A(I,J)=-A(I,J)
  4           CONTINUE
              B(I)=-B(I)
  6   CONTINUE
            TRANSFORMA TODOS C(J) MENOR QUE ZERO EM C(J) MAIOR QUE ZERO
            J1 - INDICADOR DO NJMERO DE C(J) MENORES QUE ZERO
            VT - VETOR DAS VARIAVEIS C(J))O TRANSFORMADA EM C(J)MAIOR
                 QUE ZERO
      VAFO=0
      J1=0
      DO 40 J=1,N
            IF(A(1,J))9,40,40
  9           J1=J1+1
              VT(J1)=J
              VAFO=VAFO+A(1,J)
              A(1,J)=-A(1,J)
              DO 8 I=2,M
                   B(I)=B(I)-A(I,J)
                   A(I,J)=-A(I,J)
  8           CONTINUE
 40   CONTINUE
      DO 1115 I=2,M
              BAUX(I)=B(I)
1115  CONTINUE
```

```
      ORDENAR OS COEFICIENTES DA FO EM ORDEM CRESCENTE
      DO 5 J=1,N
           C(J)=A(1,J)
    5 CONTINUE
      DO 15 J=1,N
           MENOR=1
           DO 10 JJ=2,N
                IF(C(JJ).LT.C(MENOR))MENOR=JJ
   10      CONTINUE
           ICO(J)=MENOR
           C(MENOR)=100000
   15 CONTINUE
      DO 71 J=1,N
           IVP(J)=ICO(J)
   71 CONTINUE
      DO 177 I=2,M
           IF(B(I))173,173,177
  173        CONTINUE
               POA=2
      CALL HEURI2
               GO TO 7751
  177 CONTINUE
      DO 225 I=2,M
           DO 223 J=1,N
                IF(A(I,J))222,223,223
  222           CONTINUE
                  POA=2
                  CALL HEURI2
                  GO TO 7751
  223      CONTINUE
  225 CONTINUE
      DETECTAR AS VARIAVEIS QUE IRAO COMPETIR
      CALL HEURI
      I1=2
    1 CONTINUE
      IF(I1.GT.N)GO TO 7751
      IF(X(ICO(I1)).EQ.0)GO TO 30
      I1=I1+1
      GO TO 1
      MONTAGEM DO PROBLEMA


   30 I2=0
      I3=I1-1
      DO 12 J=1,I3
           IF(X(ICO(J)).EQ.0)GO TO 12
              I2=I2+1
              IVP(I2)=ICO(J)
   12 CONTINUE
      IF(I2)16,16,17
   16 CONTINUE
         I1=I1+1
      GO TO 1
   17 CONTINUE
      I2=I2+1
      IVP(I2)=ICO(I1)
      DO 42 J=1,I2
           DO 32 I=1,M
                AA(I,J)=A(I,IVP(J))
                BAUX(I)=BAUX(I)+AA(I,J)
   32      CONTINUE
   42 CONTINUE
      DO 43 I=1,M
         BAUX(I)=BAUX(I)-AA(I,I2)
   43 CONTINUE
      I1=I1+1
      CALL LEXS
      R1=0
      DO 35 J=1,N
           IF(X(J))35,35,33
   33        CONTINUE
             R1=R1+1
             VB(R1)=J
   35 CONTINUE
      GO TO 1
 7751 CONTINUE
      CALL TIM300(TIME)
      TFINAL=TIME
      TGASTO=(TFINAL-TINICI)/300.
      WRITE(IMP,7758)TGASTO
       RETURN
   57 CONTINUE
```

```
      DO 76 J=1,N
            IVP(J)=ICO(J)
   76 CONTINUE
      PO=2
        I2=N
        DO 67 J=1,N
              DO 65 I=1,M
                    AA(I,J)=A(I,IVP(J))
   65         CONTINUE
   67   CONTINUE
        DO 69 I=2,M
              BAUX(I)=B(I)
   69   CONTINUE
      IF(POA.EQ.2)GO TO 155
        CALL LEXS
  153 CONTINUE
      VAUX=VAFUOB+VAFO
      R1=0
      DO 85 J=1,N
            IF(X(J))85,85,83
   83         CONTINUE
              R1=R1+1
              VB(R1)=J
   85 CONTINUE
      IF(TP.EQ.2)VAUX=-VAUX
      IF(VAFUOB.EQ.0)GO TO 167
      WRITE(IMP,3007)VAUX
      WRITE(IMP,37)(VB(J),J=1,R1)
      GO TO 126
  167 CONTINUE
      WRITE(IMP,555)
  126 CONTINUE
      WRITE(IMP,2003)NIRUS
      CALL TIM300(TIME)
      TTFIM=TIME
      TTFIM=(TTFIM-TINICI)/300.
      WRITE(IMP,81)TTFIM
      RETURN
  155 CONTINUE
        CALL LEXSN


      GO TO 153
   37 FORMAT(' VARIAVEIS BASICAS =',25I4,//,20X,25I4)
   81 FORMAT(///,40X,'TEMPO DE CPU UTILIZADO',I10,2X,' SEGUNDOS')
  555 FORMAT(20X,'P RO B L E M A   I M P O S S I V E L')
 2003 FORMAT(///,' NIRUS',I10)
 3007 FORMAT(///,20X,'S O L U C A O  O T I M A',///,20X,'VALOR DA FUNCAO
     * OBJETIVO = ', I10,//)
 7758 FORMAT(///,40X,'TEMPO DE CPU UTILIZADO = ',I10,2X, '  SEGUNDOS')
      END
```

```
**********************************************************************
**                                                                  **
      SUBROUTINE LEXS
**                                                                  **
**********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION G(55),SL(55),LMA(55),SOFO(55)
      Z=0
      NIRUS=0
      DO 1113 J=1,I2
              Y(J)=0
1113  CONTINUE
      IF(M-2)80,80,84
   84 CONTINUE
      ORDENAR AS RESTRICOES
      DO 5 I=2,M
           SL(I)=0
           DO 50 J=1,I2
                SL(I)=SL(I)+AA(I,J)
   50      CONTINUE
           G(I)=FLOAT(SL(I))/BAUX(I)
    5 CONTINUE
           DO 75 J=2,M
                K=2
                DO 10 I=2,M
                     IF(G(I).GT.G(K))K=I


   10           CONTINUE
                LMA(J)=K
                G(K)=-1.
   75      CONTINUE
           GO TO 87
   80 CONTINUE
        LMA(2)=2
   87 CONTINUE
      K13=LMA(2)
      S(K13)=-BAUX(K13)
      SOFO(1)=AA(1,1)
      DO 1111 J=2,I2
              SOFO(J)=SOFO(J-1)+AA(1,J)
1111  CONTINUE
      IF(PO.EQ.2)GO TO 605
      BAUX(1)=-SOFO(I2-1)
  607 CONTINUE
      VERIFICA FUNCAO OBJETIVO
      VERIFICA RESTRICAO FLUTUANTE
    1 CONTINUE
        IF(BAUX(1))11,20,20
   11     CONTINUE
            J=1
  169       CONTINUE
            IF(BAUX(1)+SOFO(J))30,40,40
   30          CONTINUE
                J=J+1
                GO TO 169
   40           CONTINUE
                IF(S(K13)+AA(K13,J))7,7,170
    7             CONTINUE
                  BAUX(1)=BAUX(1)+AA(1,J)
                  S(K13)=S(K13)+AA(K13,J)
                  Y(J)=1
                  UJE=J
                  GO TO 1
      VERIFICAR AS RESTRICOES
   20 CONTINUE
      IJT=K13
      NIRUS=NIRUS+1
```

```
      DO 105 I=2,M
             K13=LMA(I)
             IF(K13.EQ.IJT)GO TO 105
             S(K13)=0
             DO 101 J=1,I2
                    S(K13)=AA(K13,J)*Y(J)+S(K13)
 101         CONTINUE
             S(K13)=S(K13)-BAUX(K13)
             IF(S(K13))105,105,102
 105  CONTINUE
    C SOLUCAO VIAVEL
      VRES=VAFUOB
      Z=0
      DO 27 J=1,I2
             Z=Z+AA(1,J)*Y(J)
  27  CONTINUE
      BAUX(1)=-1
         ATUALIZACAO DAS VARIAVEIS BASICAS
      DO 52 J=1,I2
             X(IVP(J))=Y(J)
  52  CONTINUE
      IF(PO.EQ.2)GO TO 5555
      VAFUOB=0
      DO 51 J=1,N
             VAFUOB=VAFUOB+A(1,J)*X(J)
  51  CONTINUE
5556  CONTINUE
      IF(VRES.EQ.VAFUOB)GO TO 95
      VAUX=VAFUOB+VAFO
      IF(J1.EQ.0)GO TO 91
      DO 66 J=1,J1
             K=VT(J)
             X(K)=1-X(K)
  66  CONTINUE
  91  CONTINUE
      R1=0
      DO 85 J=1,N
             IF(X(J))85,85,83
  83           CONTINUE
               R1=R1+1


               VB(R1)=J
  85  CONTINUE
      IF(TP.EQ.2)VAUX=-VAUX
      WRITE(IMP,5455)VAUX,(VB(J),J=1,R1)
  95  CONTINUE
      DO 301 J=1,I2
             BAUX(1)=BAUX(1)+(1-Y(J))*AA(1,J)
             IF(BAUX(1))301,302,302
 301  CONTINUE
3005  CONTINUE
      DO 3001 I=2,M
             DO 3003 J=1,I2
                    BAUX(I)=BAUX(I)-AA(I,J)*X(IVP(J))
3003           CONTINUE
3001  CONTINUE
      RETURN
 302  CONTINUE
      Y(J)=1
      K13=LMA(2)
      S(K13)=S(K13)+AA(K13,J)
      IF(J-JJE)303,305,305
 303     CONTINUE
           IF(S(K13))71,71,72
  71          CONTINUE
                JJE=J
              GO TO 20
  72          CONTINUE
                 Y(J)=0
                 S(K13)=S(K13)-AA(K13,J)
                 BAUX(1)=BAUX(1)-AA(1,J)
              GO TO 170
 305     CONTINUE
           JJE=J
           K15=J-1
           DO 307 J=1,K15
                  BAUX(1)=BAUX(1)-AA(1,J)*Y(J)
                  S(K13)=S(K13)-AA(K13,J)*Y(J)
                  Y(J)=0
 307       CONTINUE
        GO TO 1
```

```
      UJE=J
      K15=J-1
      DO 307 J=1,K15
             BAUX(1)=BAUX(1)-AA(1,J)*Y(J)
             S(K13)=S(K13)-AA(K13,J)*Y(J)
             Y(J)=0
307   CONTINUE
      IF(S(K13))1,1,102
   CALCULAR O PULO
102 CONTINUE
   AUX=S(K13)
   DO 155 J=1,I2
         IF(AA(K13,J))154,155,158
154        CONTINUE
           AUX=AUX+(1-Y(J))*AA(K13,J)
           GO TO 153
158        CONTINUE
           AUX=AUX-AA(K13,J)*Y(J)
153        CONTINUE
           IF(AUX)172,172,155
155 CONTINUE

   STOP
172 CONTINUE
   DO 173 K=1,J
          BAUX(1)=BAUX(1)-AA(1,K)*Y(K)
          S(K13)=S(K13)-AA(K13,K)*Y(K)
          Y(K)=0
173 CONTINUE
   CALCULAR O INDICE A DIREITA DE J
170 CONTINUE
   IF(AA(K13,J))220,220,230
230   CONTINUE
      J=J+1
220   CONTINUE
   IF(J.GT.I2)GO TO 3005
      IF(Y(J))205,205,210
210      CONTINUE
         BAUX(1)=BAUX(1)-AA(1,J)
         S(K13)=S(K13)-AA(K13,J)
         Y(J)=0
         GO TO 230
205      CONTINUE
         BAUX(1)=BAUX(1)+AA(1,J)
         S(K13)=S(K13)+AA(K13,J)
         UJE=J
         Y(J)=1
         IF(S(K13))1,1,102
         PULO ESTABELECIDO PELA RESTRICAO PROVENIENTE DA FO
495 CONTINUE
      DO 490 J=1,I2
            BAUX(1)=BAUX(1)+(1-Y(J))*AA(1,J)
            S(K13)=S(K13)+AA(K13,J)*(1-Y(J))
            IF(BAUX(1))490,485,485
485            CONTINUE
               DO 480 L=1,J
                      S(K13)=S(K13)-AA(K13,L)
                      BAUX(1)=BAUX(1)-AA(1,L)
                      Y(L)=0
480            CONTINUE
               S(K13)=S(K13)+AA(K13,J)
               BAUX(1)=BAUX(1)+AA(1,J)




               Y(J)=1
               UJE=J
               IF(S(K13))1,1,102
490   CONTINUE
      RETURN
5455 FORMAT(/,15X,I6,15X,30I3,/,36X,30I3)
   END
```

```
**  ********************************************************************
**                                                                    **
      SUBROUTINE LEXSN
**                                                                    **
**  ********************************************************************
      IMPLICIT INTEGER(A-F,H-Z)
      COMMON A(55,55),AA(55,55),
     *       B(55),BAUX(55),C(55),ICO(55),IVP(55),
     *       S(55),TR(55),VB(55),X(55),Y(55),
     *       I1,I2,I3,IMP,LEI,
     *       M,N,NIRUS,PO,POA,
     *       TP,VAFUOB,Z
      COMMON VT(55),VAFO,J1,NUPRO,CHAVE
      DIMENSION G(55),SL(55),LMA(55),SOFO(55)
      NIRUS=0
      DO 1113 J=1,I2
              Y(J)=0
1113  CONTINUE
      IF(M-2)80,80,84
   84 CONTINUE
      ORDENAR AS RESTRICOES
      DO 5 I=2,M
           SL(I)=0
           DO 50 J=1,I2
                IF(AA(I,J))50,50,705
  705             CONTINUE
                  SL(I)=SL(I)+AA(I,J)
   50      CONTINUE
           G(I)=SL(I)-BAUX(I)
    5 CONTINUE
           DO 75 J=2,M
                K=2




                DO 10 I=2,M
                     IF(G(I).GT.G(K))K=I
   10           CONTINUE
                LMA(J)=K
                G(K)=-800000
   75      CONTINUE
           GO TO 87
   80 CONTINUE
        LMA(2)=2
   87 CONTINUE

      SOFO(1)=AA(1,1)
      DO 1111 J=2,I2
              SOFO(J)=SOFO(J-1)+AA(1,J)
1111  CONTINUE
        BAUX(1)=-VAFUOB
      VERIFICA FUNCAO OBJETIVO
      VERIFICA RESTRICAO FLUTUANTE
      K13=LMA(2)
      S(K13)=-BAUX(K13)
      UJE=I2+1
    1 CONTINUE
         IF(BAUX(1))11,20,20
   11      CONTINUE
              K25=UJE-1
              IF(K25.EQ.0)GO TO 495
              DO 2 J=1,K25
                   IF(BAUX(1)+SOFO(J))2,40,40
   40                CONTINUE
                        Y(J)=1
                        UJE=J
                        S(K13)=S(K13)+AA(K13,J)
                        BAUX(1)=BAUX(1)+AA(1,J)
                        IF(S(K13))1,1,102
    2         CONTINUE
              GO TO 495
      VERIFICAR AS RESTRICOES
```

```
 20 CONTINUE
    IJT=K13
    NIRUS=NIRUS+1


    DO 105 I=2,M
           K13=LMA(I)
           IF(K13.EQ.IJT)GO TO 105
           S(K13)=0
           DO 101 J=1,I2
                  S(K13)=AA(K13,J)*Y(J)+S(K13)
101        CONTINUE
    S(K13)=S(K13)-BAUX(K13)
           IF(S(K13))105,105,102
105 CONTINUE
  C SOLUCAO VIAVEL
    Z=0
    DO 27 J=1,I2
          Z=Z+AA(1,J)*Y(J)
 27 CONTINUE
    BAUX(1)=-1
       ATJALIZACAO DAS VARIAVEIS BASICAS
    DO 52 J=1,I2
          X(IVP(J))=Y(J)
 52 CONTINUE
    VAFUOB=Z
      VAUX=VAFUOB+VAFO
    IF(J1.EQ.0)GO TO 91
    DO 66 J=1,J1
          K=VT(J)
          X(K)=1-X(K)
 66 CONTINUE
 91 CONTINUE
    R1=0
    DO 85 J=1,N
          IF(X(J))85,85,83
 83         CONTINUE
            R1=R1+1
            VB(R1)=J
 85 CONTINUE
    IF(TP.EQ.2)VAUX=-VAUX
    WRITE(IMP,5455)VAUX,(VB(J),J=1,R1)
    DO 301 J=1,I2
           BAUX(1)=BAUX(1)+(1-Y(J))*AA(1,J)
           IF(BAUX(1))301,302,302




301 CONTINUE
3005 CONTINUE
    DO 3001 I=2,M
          DO 3003 J=1,I2
                 BAUX(I)=BAUX(I)-AA(I,J)*X(IVP(J))
3003        CONTINUE
3001 CONTINUE
    RETURN
 302 CONTINUE
    Y(J)=1
    K13=LMA(2)
    S(K13)=S(K13)+AA(K13,J)
    IF(J-JJE)303,305,305
 303    CONTINUE
          IF(S(K13))71,71,102
  71        CONTINUE
              UJE=J
            GO TO 20
 305    CONTINUE
```

```
      CALCULAR O PULO
  102 CONTINUE
      DO 155 J=1,I2
             S(K13)=S(K13)-AA(K13,J)*Y(J)
             BAUX(1)=BAUX(1)-AA(1,J)*Y(J)
             Y(J)=0
             IF(S(K13))170,170,155
  155 CONTINUE
      GO TO 3005
      CALCULAR O INDICE A DIREITA DE J
  170 CONTINUE
      J=J+1
      IF(J.GT.I2)GO TO 411
         IF(Y(J))205,205,210
  210       CONTINUE
             BAUX(1)=BAUX(1)-AA(1,J)
             S(K13)=S(K13)-AA(K13,J)
             Y(J)=0
             GO TO 170
  205       CONTINUE
             IF(S(K13)+AA(K13,J))475,475,170
  475 CONTINUE
             BAUX(1)=BAUX(1)+AA(1,J)
             S(K13)=S(K13)+AA(K13,J)
             UJE=J
             Y(J)=1
               GO TO 1
  411 CONTINUE
      GO TO 3005
  605 CONTINUE
           BAUX(1)=-VAFUOB
      GO TO 607
 5555 CONTINUE
      VAFUOB=Z
      GO TO 5556
 5455 FORMAT(/,15X,I6,15X,30I3,/,36X,30I3)
      END
```

A N E X O 4

MANUAL DO MODELO PROPOSTO

MANUAL DO MODELO PROPOSTO

1. Condicionamento do problema

Para a utilização do programa, o problema não necessita estar em uma forma padrão pré-definida. Poderá estar numa das formas abaixo:

$$\text{Max (min)} \sum_{j=1}^{n} c_j x_j$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \gtreqless b_j, \text{ com } i = 1(1)m$$

Onde $x_j \in \{0,1\}$ e todos os coeficientes têm que ser inteiros. Caso existam alguns que não o sejam, de acordo com a precisão desejada pelo usuário, poder-se-á arredondar todos os coeficientes fracionários para o inteiro mais próximo, ou multiplicá-los por um fator que os torne inteiros.

2. Preparação dos dados

O programa permite processar diversos problemas a cada vez que for executado, sendo que o "DECK" de cartões de entrada estará disposto de acordo com a figura 12.

O primeiro cartão de entrada dos dados terá a finalidade de informar o número de problemas que serão executados, seguido de tantos blocos de cartões quantos forem os problemas a serem processados.

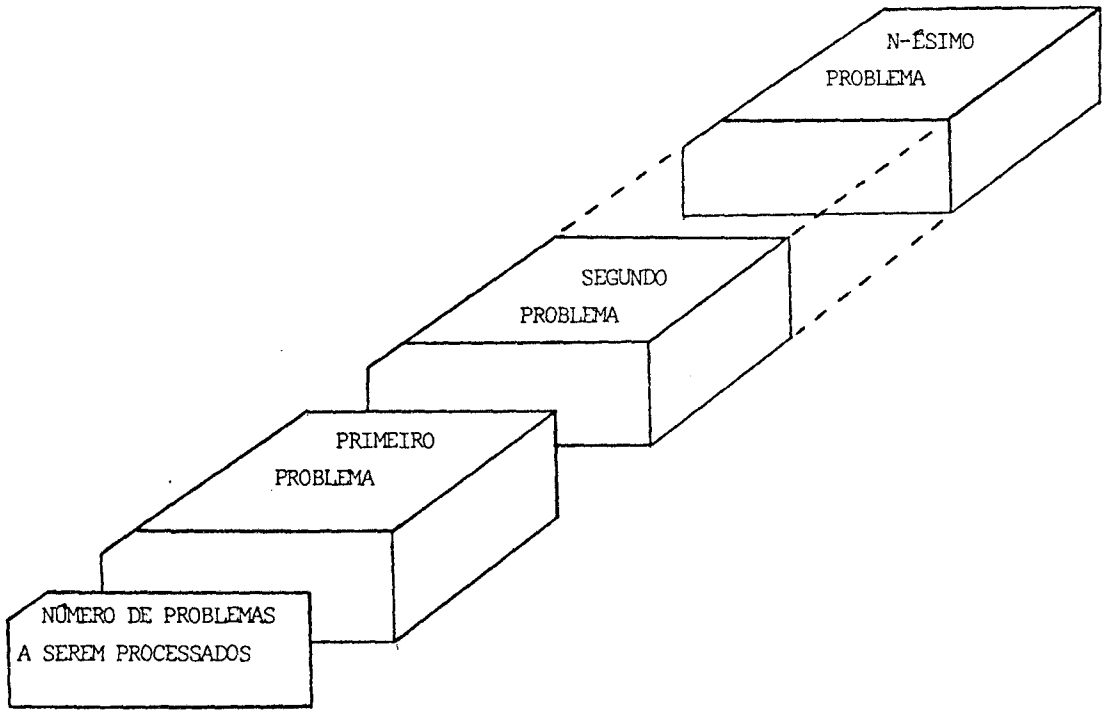


FIGURA 12 - "DECK" DE CARTÕES DE ENTRADA

3. Perfuração dos dados

3.1. Primeiro cartão do "DECK" de cartões de entrada

Neste cartão deverá constar o número de problemas a serem processados pelo programa. Este número corresponde a uma variável inteira (NUPRO), que ocupará um campo de cinco colunas do cartão. O último dígito necessariamente será alocado na quinta coluna do cartão.

3.2. Cartões de cada bloco

Cada bloco de cartões subseqüentes ao primeiro cartão do "DECK" de dados de entrada corresponderá a um problema e constituir-se-á dos cartões indicados na figura 13.

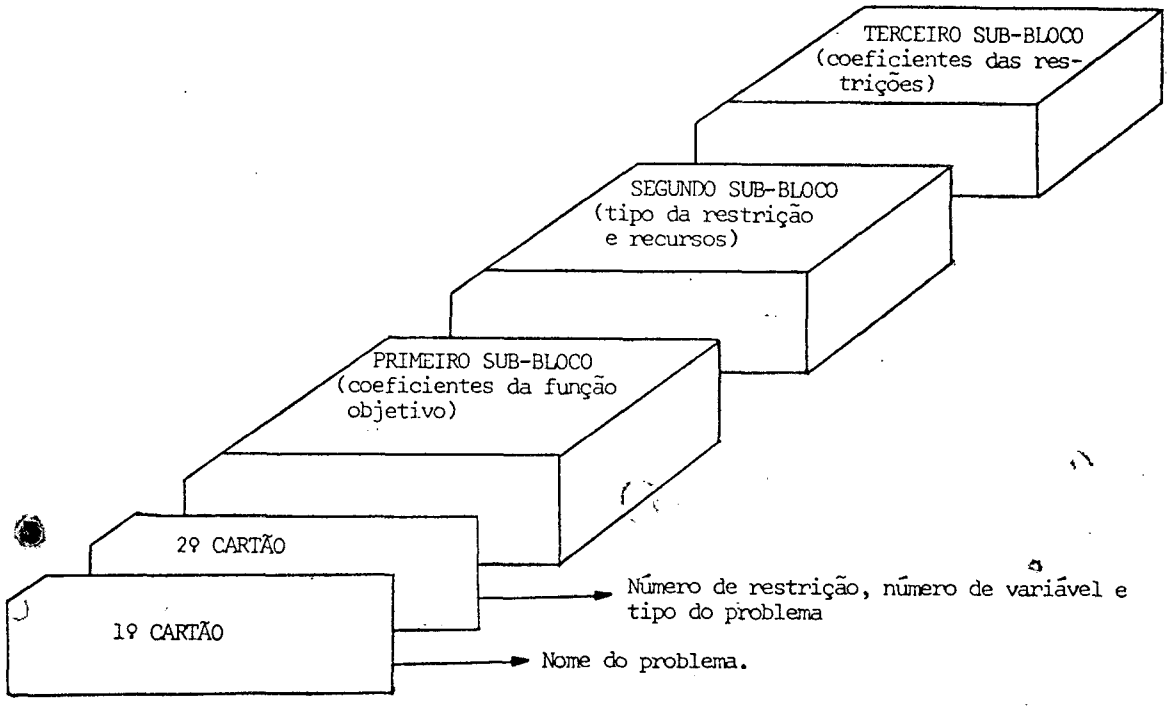


FIGURA 13 - "DECK" DE CARTÕES DE CADA BLOCO

CADA CARTÃO DO BLOCO TERÁ A SEGUINTE FINALIDADE:

- Primeiro cartão do bloco:

Neste cartão deverá constar um nome para o problema, perfurado em quaisquer posições das 80 colunas do cartão.

- Segundo cartão do bloco:

Neste cartão deverá constar o número de restrições, o número de variáveis e o tipo do problema do respectivo bloco.

- Número de restrições:

Variável inteira (M), que ocupará um campo de cinco colunas do cartão. O último dígito necessariamente será alocado na quinta coluna do cartão.

- Número de variável:

Variável inteira (N) que ocupará um campo de cinco colunas do cartão. O último dígito necessariamente será alocado na décima coluna do cartão.

- Tipo do problema:

Variável inteira (TP) que assumirá o valor 1 se o problema for de maximização e 2 se for de minimização. Deverá necessariamente ser alocado na décima quinta coluna do cartão.

- Primeiro sub-bloco:

Este sub-bloco será constituído dos coeficientes da função objetivo, sendo que cada coeficiente ocupará um campo de cinco colunas do cartão. O último dígito de cada coeficiente necessariamente será alocado na última posição de seu respectivo campo. Serão utilizados tantos cartões quantos forem necessários.

- Segundo sub-bloco:

Para cada restrição corresponderá um cartão contendo o tipo da restrição e o recurso correspondente.

Tipo da restrição: variável inteira (TR), que poderá assumir os seguintes valores:

(1) - se a restrição for maior ou igual

(2) - se a restrição for menor ou igual.

Esta variável deverá necessariamente ser alocada na quinta coluna do cartão.

Valor do recurso: variável inteira (B) que ocupará um campo de cinco colunas. O último dígito será alocado na décima coluna.

- Terceiro sub-bloco:

Este sub-bloco será constituído dos coeficientes das restrições. Cada coeficiente ocupará um campo de cinco colunas do cartão. O último dígito de cada coeficiente deverá ocupar a última posição do campo. Serão utilizados tantos cartões

quantos forem necessários para cada restrição. Mudando a restrição, deverá ser utilizado um novo cartão.