SÔBRE A PROGRAMAÇÃO LINEAR INTEIRA

EDSON WALMIR CAZARINI

Dissertação apresentada ao INSTITUTO DE CIÊNCIAS MATEMÁTICAS DE SÃO CARLOS DA UNIVERSIDADE DE SÃO PAULO para obtenção do grau de MESTRE EM "CIÊNCIAS DE COMPUTAÇÃO E ESTATÍSTICA"

Orientador : Dr. ODELAR LEITE LINHARES

SÃO CARLOS
outubro/76

SUMÁRIO

SUMÁRIO

RESUMO

Faz-se um apanhado geral sôbre a programação linear inteira.

Apresenta-se no capítulo, Preliminéres Matemáticos, resultados matemáticos necessários ao desenvolvimento dos demais capítulos, compreendendo alguns tópicos como: Fundamentos Matemáticos, abrangendo definições, propriedades, aplicações e exemplos sôbre matrizes, vetores, espaços vetoriais e conjuntos convexos. Introduz-se a programação linear com definições e propriedades das soluções, geração de uma solução possível e o método simplex, apresentado nas formas primitiva e revisada, considerando-se ainda a utilização de variáveis artificiais na base. Apresenta-se também alguns conceitos e aplicações da dualidade em programação linear, finalizando com a apresentação do problema de transporte e algumas de suas variações.

A programação inteira, sua definição, propriedades, modêlo matemático do problema geral, e em particular o da programação linear inteira, aplicações como o exemplo do problema da mochila e do caixeiro viajante com seus respectivos modêlos matemáticos, bem como considerações sôbre a solução de um problema de programação linear inteira, são assuntos tratados em Introdução à Programação Inteira.

Os algorítmos de GOMORY, baseados no método simplex dual e primal assim como êsse algorítmo dual, adaptado para resolver problemas de programação linear inteira onde as soluções são limitadas superiormente, são mostrados juntamente com os respectivos programas, escritos em linguagem FORTRAN IV, no capítulo Métodos Numéricos e Programas.

Em Conclusões e Análise Crítica, faz-se críticas sôbre o desempenho dos programas apresentados para o computador com a análise dos resultados obtidos, dos testes de parada e tempos de execução dos programas, gerados através de execuções com sistemas obtidos aleatoriamente.

É apresentado também, bibliografia com o objetivo de conduzir o leitor às fontes de tratamento do problema da programação linear inteira. Ressalta-se que este trabalho de modo algum esgota o assunto que, a cada dia recebe novas contribuições de especialistas de todo o mundo.

ABSTRACT

The present work makes a general survey the integer programming problems.

Chapter "Preliminâres Matemáticos" covers the theoretical results required for the next chapters understanding. It develops theoretical aspects about matrix, vector and vector spaces, and gives an introduction to linear programming, duality, simplex method in the usual and revised form.

Integer programming, definitions, propriety and mathematical models are introduces in "Introdução à Programação Inteira".

The Gomory algorithms, based on the simplex, dual and primal, methods, and correspondents programs are showed in the chapter: "Métodos Numéricos e Programas".

Finally, in "Conclusões e Análise Crítica" it is analyzed the performance of integer programming algorithms' programs, through convergence and processing times tests, using random systems.

AGRADECIMENTOS

Ao Professor Odeler Leite Linhares , pela orientação.

Ao Centro de Processamento de Dados da Escola de Engenharia de São Carlos, da Universidade de São Paulo, pelo apôio computacional.

A todos que direta ou indiretamente colaboraram, em especial à Fundação de Amparo à Pesquisa do Estado de São Paulo.

CAPÍTULO 0 - INTRODUÇÃO

A programação linear inteira vem sendo objeto de intensa pesquisa, dado o grande número de aplicações práticas que oferece. É por exemplo utilizada em problemas de distribuição ótima de cargas dentro de aeronaves, escolha de rotas mais econômicas em transportes em geral, minimização de custos de transporte ferroviário, com a utilização mínima de vagões e percurso mínimo da carga transportada, etc..

O problema porém, não é de solução tão simples, por meio de computadores, pelo fato de essas máquinas trabalharem com um número finito de dígitos na representação decimal de números reais e com limitado intervalo de valores numéricos inteiros.

Por outro lado, pode ocorrer que, em alguns métodos, o espaço de memória utilizado com resultados intermediários aumente em cada iteração, esgotando-se, em pouco tempo a memória de trabalho, disponível do computador tendo-se que lançar mão de memórias auxiliares o que prejudica o tempo de processamento.

Embora antigo, o problema da programação linear inteira, mereceu a atenção mais cuidadosa dos estudiosos somente a partir da década de quarenta e com o advento dos computadores digitais, poderosos e com extraordinária velocidade de cálculo e alta confiabilidade.

I.1- Fundamentos Matemáticos

O entendimento dos conceitos utilizados em programação linear e em particular a programação inteira requerem alguns elementos básicos da matemática que define-se a seguir.

I.1.1- Matrizes

Entende-se por matriz um arranjo retangular de n x m números, dispostos em m linhas e n colunas conforme a seguinte forma:

$$\begin{matrix} a_{11} & a_{12} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & \cdots\cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots\cdots & a_{mn} \end{matrix}$$

Normalmente esse arranjo é colocado entre parênteses e recebe o nome de Matriz A, neste exemplo, que usa-se também representar por (a_{ij}).

A matriz A recebe o nome de MATRIZ QUADRADA se o número de colunas for igual ao número de linhas (m = n), e se diz de ordem n.

Um VETOR COLUNA é uma matriz com somente uma coluna e um VETOR LINHA, uma matriz com apenas uma linha.

MATRIZ DIAGONAL é uma matriz quadrada cujos elementos são todos iguais a zero, com exceção daqueles situados na diagonal principal.

MATRIZ IDENTIDADE é uma matriz diagonal em que os elementos da diagonal são iguais a 1 (um). Essa matriz é normalmente representada por I_n (ou I simplésmente), onde n é a ordem da matriz.

MATRIZ TRANSPOSTA, A^t de uma matriz A é definida trocando-se as linhas pelas colunas da matriz A, como por exemplo:

$$\begin{pmatrix} a_{11} & a_{21} & \cdots\cdots & a_{m1} \\ a_{12} & a_{22} & \cdots\cdots & a_{m2} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ a_{1n} & a_{2n} & \cdots\cdots & a_{mn} \end{pmatrix}$$

Duas matrizes são IGUAIS se, e somente se, são de dimensões iguais e seus elementos correspondentes, são iguais.

Uma matriz quadrada é dita TRIANGULAR se todos os seus elementos $a_{ij} = 0$ para $i > j$ (SUPERIOR) ou todos os elementos $a_{ij} = 0$ para $i < j$ (INFERIOR).

Uma matriz A é dita SIMÉTRICA se for igual a sua transposta , ou seja, $a_{ij} = a_{ji}$ para todo i e j.

MATRIZ NULA é a matriz em que todos os seus elementos são iguais a zero.

Dado um escalar α qualquer, e uma matriz A o PRODUTO DO ESCALAR PELA MATRIZ será dado por :

$$\alpha A = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \cdots\cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots\cdots & \alpha a_{2n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots\cdots & \alpha a_{mn} \end{pmatrix} = (\alpha a_{ij})$$

A SOMA de duas matrizes de mesma dimensão $\underline{m \times n}$ será dado por outra matriz de mesma dimensão, definida a partir da ADIÇÃO dos elementos correspondentes como por exemplo:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & \cdots\cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots\cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots\cdots & b_{1n} \\ b_{21} & b_{22} & \cdots\cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots\cdots & b_{mn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots\cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots\cdots & a_{2n} + b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots\cdots & a_{mn} + b_{mn} \end{pmatrix}$$

O PRODUTO de duas matrizes A e B ($C = A.B$) se define somente quando o número de colunas de A for igual ao número de linhas de B e a matriz resultante será obtida através da soma dos produtos das linhas de A pelas colunas de B. Se A for de dimensão $\underline{m \times \ell}$ e B for de dimensão $\underline{\ell \times n}$ a matriz produto será de dimensão $\underline{m \times n}$ e com valores definidos da seguinte forma:

$$c_{ij} = \sum_{k=1}^{\ell} a_{ik} \cdot b_{kj} \qquad \text{para:} \begin{cases} i = 1,2,\ldots,m \\ j = 1,2,\ldots,n \end{cases}$$

Propriedades de ADIÇÃO e MULTIPLICAÇÃO por ESCALAR.

a) $(A + B) + C = A + (B + C)$

b) $A + B = B + A$

c) $(\alpha + \beta) \times A = \alpha A + \beta A$

d) $\alpha (A + B) = \alpha A + \alpha B$

e) $A + 0 = A$

Onde, A, B, C são matrizes de dimensão $m \times n$ e α e β são escaleres.

Propriedades de MULTIPLICAÇÃO de matrizes.

a) $(A \times B) \times C = A \times (B \times C)$

b) $(A + B) \times C = A \times C + B \times C$

c) $C \times (A + B) = C \times A + C \times B$

d) $\alpha \times (A \times B) = (\alpha A) \times B = (A) \times (\alpha B)$

e) $A \times I = I \times A = A$

f) $(A \times B)^t = B^t \times A^t$

Onde A, B e C são matrizes com dimensões competiveis com a definição de produto de matrizes e α é um escalar.

Cebe ressaltar que a propriedade comutativa não é válida para o produto de matrizes, ou seja: $A \times B \neq B \times A$.

Matriz NÃO SINGULAR é uma matriz quadrada cujo determinante é diferente de ZERO.

Uma matriz B é dita INVERSA de uma matriz A quadrada, se $A \times B = B \times A = I$. Toda matriz A quadrada, NÃO SINGULAR tem uma e uma só matriz inversa correspondente.

Noteção: A matriz inversa de A é representada por A^{-1}.

I.1.2- Vetores e espaços vetoriais.

A título didático apresenta-se a definição, operações e propriedades dos vetores utilizando o espaço Euclideano bi dimensional (plano).

Em um plano os pontos podem ser representados por peres ordenados de números $P = (p_1 , p_2)$. Refere-se a P como sendo um ponto de coordenadas (p_1 , p_2) em relação à origem , fixa de coordenadas (0 , 0) ou simplesmente como um VETOR $\vec{p}$.

Graficamente pode-se visualizar a noção de vetor:

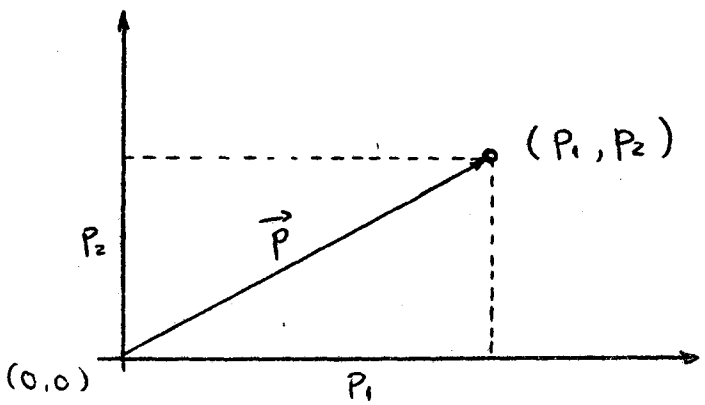


fig 1.1

Seja E_2 o espaço Euclideano bi-dimensional.

Algumas propriedades importantes dos vetores em E_2

a) MULTIPLICAÇÃO DE VETORES POR ESCALARES: A todo par α e $\vec{u}$, onde α é um escalar e $\vec{u}$ é um vetor, tem-se um vetor $\vec{v}$ correspondente ao produto escalar de α por $\vec{u}$ representados por $\vec{v} = \alpha\vec{u} = (\alpha u_1 , \alpha u_2)$ e tal que:

$\alpha > 1$ (escalar)

$\beta < 1$ (escalar)

$\vec{u}$ e $\vec{v}$ (vetores)

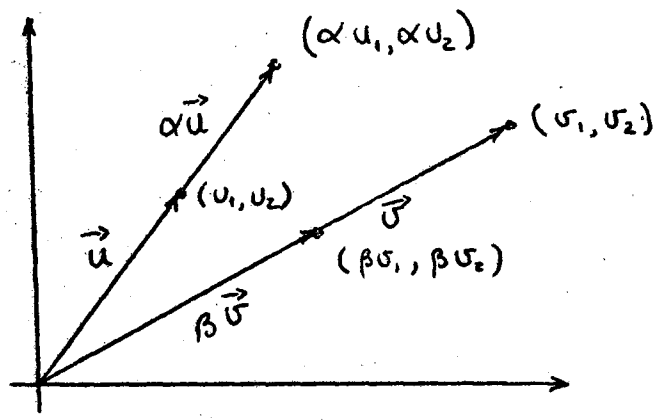


fig 1.2

a.1- $\alpha (\beta \vec{u}) = (\alpha\beta) \times \vec{u}$

a.2- $\alpha (\vec{u} + \vec{v}) = \alpha \vec{u} + \alpha \vec{v}$

a.3- $(\alpha + \beta) \times \vec{u} = \alpha \vec{u} + \beta \vec{u}$

a.4- $1 \times \vec{u} = \vec{u}$

a.5- $0 \times \vec{u} = \vec{0}$

b) SOMA DE VETORES: A todo par $\vec{u}$ e $\vec{v}$ de vetores em E_2 corresponde um vetor $\vec{w}$, chamado SOMA de $\vec{u}$ e $\vec{v}$ que se representa por $\vec{w} = \vec{u} + \vec{v} = (u_1 + v_1 , u_2 + v_2)$ e tal que:

b.1- $\vec{u} + \vec{v} = \vec{v} + \vec{u}$

b.2- $(\vec{u} + \vec{v}) + \vec{y} = \vec{u} + (\vec{v} + \vec{y})$

b.3- $\vec{u} + \vec{0} = \vec{u}$ para qualquer $\vec{u}$ em E_2.

b.4- $\vec{u} + (- \vec{u}) = \vec{0}$ A cada vetor $\vec{u}$ em E_2 corresponde um único vetor chamado inverso de $\vec{u}$, designado por $- \vec{u}$.

c) PRODUTO ESCALAR DE VETORES: A todo par de vetores $\vec{u}$ e $\vec{v}$, em E_2 corresponde um NÚMERO REAL Chamado PRODUTO ESCALAR de $\vec{u}$ com $\vec{v}$ definido como: $(\vec{u} , \vec{v}) = u_1 . v_1 + u_2 . v_2$ e tal que:

c.1- $(\vec{u} , \vec{v}) = (\vec{v} , \vec{u})$

c.2- $((\alpha \vec{u} + \beta \vec{v}) , \vec{y}) = \alpha (\vec{u} , \vec{y}) + \beta (\vec{v} , \vec{y})$

c.3- $(\vec{u} , \vec{u}) \geq 0$ e $(\vec{u} , \vec{u}) = 0$ se e somente se, $\vec{u} = \vec{0}$.

para todos os escalares α e β, e vetores $\vec{u}$, $\vec{v}$ e $\vec{y}$ em E_2

d) COMPRIMENTO (módulo) de um vetor: A todo vetor $\vec{u}$ de E_2, correponde um número real chamado de comprimento ou módulo de $\vec{u}$, definido como:

$$\| \vec{u} \| = + \sqrt{ u_1^2 + u_2^2 }$$ e tal que:

d.1- $\| \vec{u} \| \geq 0$ e $\| \vec{u} \| = 0$ se e somente se $\vec{u} = \vec{0}$

d.2- $\| \alpha \vec{u} \| = |\alpha| \cdot \| \vec{u} \|$

d.3- $\| \vec{u} + \vec{v} \| \leq \| \vec{u} \| + \| \vec{v} \|$

d.4 $\| \vec{u} \| = + \sqrt{(\vec{u} , \vec{u})}$

e) DISTÂNCIA entre dois vetores: A todo par de vetores $\vec{u}$ e $\vec{v}$ em E_2, corresponde um número real, chamado distância entre $\vec{u}$ e $\vec{v}$, o que se representa por $d (\vec{u} , \vec{v})$ tal que:

$$d (\vec{u} , \vec{v}) = \| \vec{u} - \vec{v} \| = + \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

f) Um conjunto de vetores $\vec{u}_1 , \vec{u}_2 , \ldots , \vec{u}_n$ se denomina , LINEARMENTE INDEPENDENTE se, para todos os escalares $\alpha_1 , \alpha_2 \ldots , \alpha_n$:

$\alpha_1 \vec{u}_1 + \alpha_2 \vec{u}_2 + \ldots + \alpha_n \vec{u}_n = \vec{0}$ implica que:

$\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$

Caso contrário o conjunto de vetores se denomina, LINEARMENTE DEPENDENTE.

g) ESPAÇO EUCLIDEANO: Entende-se por espaço euclideano de dimensão n , E_n , um conjunto de n vetores LINEARMENTE INDEPENDENTES em que se verifice ainde as propriedades enunciadas acima (de a a c). Qualquer conjunto de $n + 1$ vetores dess se espaço são LINEARMENTE DEPENDENTES.

BASE para o espaço E_n é um conjunto de n vetores LINEARMENTE INDEPENDENTES. Qualquer vetor de E_n poderá ser escrito como COMBINAÇÃO LINEAR dos vetores da base.

h) HIPERPLANO: Uma condição linear, definida em E_2, como sendo $x_1 \cdot \vec{u}_1 + x_2 \cdot \vec{u}_2 = a$, onde x_1, x_2 e a são constantes, representa uma linha reta. Se o mesmo for definido em E_3 teremos um plano e se definirmos no espaço E_n teremos uma figura chamada de HIPERPLANO. Um hiperplano $H(\vec{x}, a)$ definido em E_n é o conjunto de todos os vetores $\vec{u}$ tal que $(\vec{x}, \vec{u}) = a$ para $\vec{x}$ diferente de zero e um dado número real a.

Um hiperplano divide o espaço em dois semi-espeços representados por:

$$H^+(\vec{x}, a) = (\vec{u} / (\vec{x}, \vec{u}) \geqslant a)$$

$$H^-(\vec{x}, a) = (\vec{u} / (\vec{x}, \vec{u}) \leqslant a)$$

I.1.3- Conjuntos Convexos.

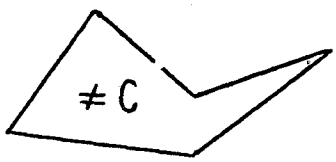
Uma COMBINAÇÃO CONVEXA dos vetores $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ é um vetor $\vec{u}$ em que:

$$\vec{u} = \alpha_1 \cdot \vec{u}_1 + \alpha_2 \cdot \vec{u}_2 + \ldots + \alpha_n \cdot \vec{u}_n$$

onde os ESCALARES α_i são maiores ou iguais a zero, e ainda, $\sum_{i=1}^{n} \alpha_i = 1$. Um sub-conjunto C é CONVEXO, contido em E_n, se e somente se, para todos os pares de vetores $\vec{u}_1$ e $\vec{u}_2$ contidos em C, qualquer combinação convexa $\vec{u} = \alpha_1 \cdot \vec{u}_1 + \alpha_2 \cdot \vec{u}_2$ pertence tambem a C.

Como exemplos clássicos de conjuntos convexos pode se sitar: o espaço, um circulo, um quadrado, um cubo, etc.

Como exemplo de conjunto não convexo pode-se citar uma circunferência.



I.2- O problema geral da programação linear.

O problema geral da programação linear consiste em encontrar um vetor $\vec{x} = (x_1, x_2, \ldots, x_n)$ que minimize a forma linear:

$$c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \qquad (1.1)$$

Sujeito às seguintes restrições lineares:

$$\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\ \ldots\ldots \cdot \ldots\ldots \cdot \ldots \cdot \ldots\ldots &\quad .. \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n &= b_m \end{aligned} \qquad (1.2)$$

$$x_j \geqslant 0 \qquad (\; j = 1, 2, \ldots, n \;) \qquad (1.3)$$

onde a_{ij} , b_i e c_j são constantes conhecidas e $m < n$.

Toma-se o cuidado de gerantir que os $b_i \geqslant 0$, uma vez que serão utilizados como primeira solução possivel para o processo iterativo que será desenvolvido. Se o problema original

contiver algum $b_i < 0$, multiplica-se a equação por (-1).

Os problemas de programação linear se referem ao uso eficiente ou distribuição ótima de recursos limitados, para se alcançar os objetivos desejados.

Caso se deseja maximizar uma forme linear (função objetivo), utiliza-se a definição anterior, apenas minimizando o negativo de forma linear desejada.

A caracteristica principal desses problemas são um grande número de soluções que satisfazem as restrições impostas pelo problema. Desse elenco de soluções toma-se aquela que minimizar a função objetivo.

Segue uma série de definições de têrmos próprios , que serão constantemente utilizados.

FORMA LINEAR: Seja E um espaço vetorial real. uma função f de E em R ($f : E \longrightarrow R$), será dita forma linear, se:

a) $f (\vec{x} + \vec{y}) = f (\vec{x}) + f (\vec{y})$

b) $f (\alpha \vec{x}) = \alpha f (\vec{x})$

SOLUÇÃO POSSIVEL: É um vetor $\vec{x}$ que satisfaz as restrições impostas pelo problema de programação linear (1.2) e (1.3).

MATRIZ BÁSICA: É uma matriz m x m , não singular formada por m colunas da matriz das restrições.

SOLUÇÃO BÁSICA: É o vetor (único) onde os elementos relativos às (n-m) colunas, da matriz das restrições, não pertencentes a matriz básica são iguais a zero e os demais elementos são obtidos, resolvendo-se o sistema não singular da matriz básica.

SOLUÇÃO POSSIVEL BÁSICA: É uma solução básica, onde todos os seus elementos têm valores não negativos.

SOLUÇÃO POSSIVEL BÁSICA NÃO DEGENERADA: É uma solução possivel básica com exatamente $\underline{m}$ elementos x_i positivos.

SOLUÇÃO ÓTIMA: É uma solução possivel que minimiza a função objetivo (1.1) .

I.2.1- Propriedades da Solução.

Apresenta-se a seguir uma série de teoremas que mostrarão o caminho para obtenção da solução ótima.

TEOREMA 1: O conjunto de tôdas as soluções positives , possiveis, de um problema de programação linear é um conjunto convexo.

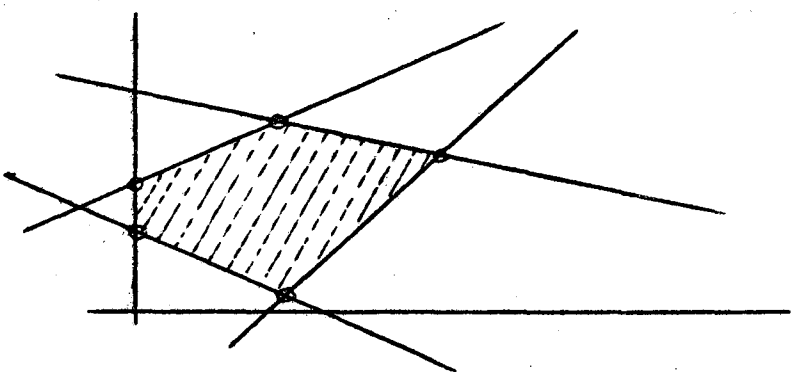


fig 1.4

TEOREMA 2: A função objetivo apresenta seu mínimo em um ponto extremo do conjunto convexo formado pelas soluções possiveis a um probleme de programação linear. Se o mínimo aparece em mais de um ponto extremo então será mínimo para toda combinação convexa desses dois pontos.

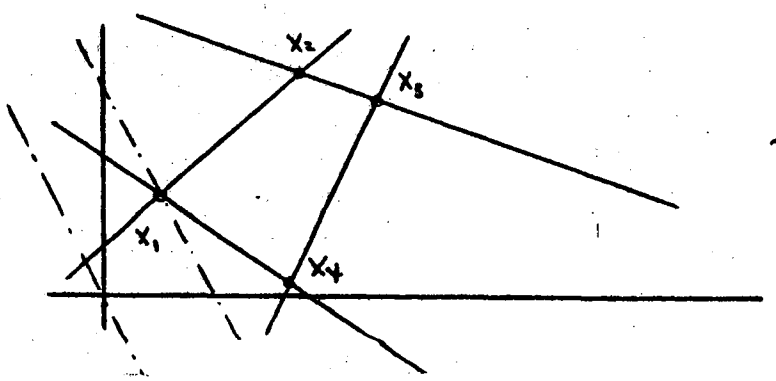


fig 1.5

TEOREMA 3: Se encontrarmos um conjunto de $k \leq m$ vetores $\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_k$ linearmente independentes e tal que:

$$x_1.\vec{A}_1 + x_2.\vec{A}_2 + \ldots + x_k.\vec{A}_k = \vec{A}_o$$

e todos os $x_i \geq 0$, então o ponto $\vec{x} = (x_1, x_2, \ldots , x_k, 0$ $0, \ldots , 0)$ é um ponto extremo do conjunto convexo formado pelas soluções possíveis. Neste caso $\vec{x}$ é um vetor n-dimensional em que os últimos (n-k) elementos são iguais a zero.

TEOREMA 4: Se $\vec{x} = (x_1, x_2, \ldots , x_n)$ é um ponto extremo de K (conjunto convexo), então os vetores associados às x_i positivas, formam um conjunto linearmente independente. Disto se observa que pelo menos m das x_i são positivas.

TEOREMA 5: O vetor $\vec{x} = (x_1, x_2, \ldots , x_n)$ é um ponto extremo do conjunto convexo K , se e somente se, os x_i positivos são coeficientes de vetores linearmente independentes $\vec{A}_j$.

$$\sum_{j=1}^{n} x_j.\vec{A}_j = \vec{A}_o$$

Tendo-se em vista as suposições e os teoremas anteriores, que estão demonstrados em [4] , temos o seguinte resumo:

1.- Existe um ponto de K (conjunto convexo) em que a função objetivo tem o seu mínimo.

2.- Cada solução possivel básica corresponde a um ponto extremo de K.

3.- Cada ponto extremo de K , tem associado a êle m vetores linearmente independentes de de um conjunto de n vetores.

Pode-se concluir que necessitamos investigar somente as soluções de ponto extremo, e portanto apenas as soluções geradas, possiveis, por m vetores linearmente independentes. Existe no máximo $\binom{n}{m}$ soluções possiveis para o probleme (combinação de n vetores, m linearmente independentes).

Para n e m grandes é praticamente impossivel obter-se tôdas as soluções possiveis, e para cada uma, obter o valor correspondente da função objetivo, escolhendo-se àquela que minimize a função objetivo.

Foi desenvolvido um algorítmo que seleciona, em uma forma ordenada, um sub-conjunto entre as soluções possiveis , que convergem para a solução ótima.

Este processo, obtem uma solução possivel e verifica se é ótima. Se não for, o procedimento encontra um outro ponto extremo, vizinho, cujo valor da objetivo é menor ou igual ao valor correspondente à precedente. em um número finito de passos o procedimento termina, encontrando a solução ótima. É possivel saber tambem quando o problema não tem solução ou a solução é indeterminada.

I.2.2- Geração de uma Solução de Ponto Extremo.

Seja:

$$x_1.\vec{A}_1 + x_2.\vec{A}_2 + \ldots + x_m.\vec{A}_m + \ldots + x_n.\vec{A}_n = \vec{A}_o$$

Supondo que se conheça uma solução de ponto extremo, em termos de m vetores $\vec{A}_j$.

Supondo esses vetores linearmente independentes, os m primeiros:

$$\vec{x} = (x_1 , x_2 , \ldots , x_m , 0 , \ldots , 0)$$

vetor solução de ponto extremo.

$$x_1.\vec{A}_1 + x_2.\vec{A}_2 + \ldots + x_m.\vec{A}_m = \vec{A}_o \qquad (1.4)$$

onde $x_i \geqslant 0$.

O problema consiste agora em determinar, por um processo eficiente, uma nova solução de ponto extremo.

Sendo os vetores $\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_m$, linearmente independentes, sabemos que formam uma base no espaço m-dimensional. Podemos então exprimir os n vetores como combinação linear dos vetores da base.

$$\vec{A}_j = \sum_{i=1}^{m} X_{ij}.\vec{A}_i \qquad (j = 1,2,\ldots,n)$$

Tomando-se o vetor $\vec{A}_{m+1}$, tem-se pelo menos um $X_{i,m+1} > 0$ na expressão:

$$X_{1,m+1}.\vec{A}_1 + X_{2,m+1}\vec{A}_2 + \ldots + X_{m,m+1}.\vec{A}_m = \vec{A}_{m+1} \qquad (1.5)$$

Tomando-se Θ , qualquer, multiplicando (1.5) por e subtraindo de (1.4) tem-se:

$$(x_1 - \Theta . X_{1,m+1}).\vec{A}_1 + (x_2 - \Theta . X_{2,m+1}).\vec{A}_2 + \ldots\ldots\ldots\ldots\ldots$$

$$\ldots + (x_m - \Theta . X_{m,m+1}).\vec{A}_m + \Theta . \vec{A}_{m+1} = \vec{A}_o \qquad (1.6)$$

O vetor $\vec{y} = (x_1 - \Theta . X_{1,m+1} , x_2 - \Theta . X_{2,m+1} , \ldots , x_m - \Theta . X_{m,m+1} , \Theta)$ é uma solução do problema.

Para $\vec{y} \neq \vec{x} \Rightarrow \Theta > 0$

Tem-se que encontrar um valor real para Θ que faça $\vec{y} \geqslant \vec{0}$, ou seja:

$$x_i - \Theta . X_{i,m+1} \geqslant 0 \quad \text{para todo} \quad X_{i,m+1} > 0 \qquad (1.7)$$

$$\frac{x_i}{X_{i,m+1}} \geqslant \Theta$$

ou, $$0 < \Theta \leqslant \min_i \frac{x_i}{X_{i,m+1}}$$

O valor de Θ obtido desta forma vai proporcionar uma solução possivel para (1.6).

Para se ter uma solução de ponto extremo não se permite que o vetor $\vec{y}$ tenha <u>m+1</u> elementos positivos. Deve- se então, forçar que pelo menos um elemento de $\vec{y}$, se torme igual a zero.

Para isso tome-se:

$$\Theta = \Theta_o = \min_i \frac{x_i}{X_{i,m+1}} \quad \text{para} \quad X_{i,m+1} > 0.$$

O elemento de $\vec{y}$ para o qual se obtem esse mínimo se reduzirá a zero. Se esse elemento for o primeiro teremos:

$$\Theta = \Theta_o = \frac{x_1}{x_{1,m+1}} \qquad \text{e}$$

$$x'_2.\vec{A}_2 + x'_3.\vec{A}_3 + \ldots + x'_m.\vec{A}_m + x'_{m+1}.\vec{A}_{m+1} = \vec{A}_o$$

onde:

$$\begin{cases} x'_i = x_i - \Theta_o.x_{i,m+1} & (i = 2,3,\ldots,m) \\ x'_{m+1} = \Theta_o \end{cases}$$

Se todos os $x_{i,m+1} \leqslant 0$, não seria possivel selecionar $\Theta > 0$ que eliminaria um dos vetores $\vec{A}_1$, $\vec{A}_2$, ... $\vec{A}_m$ de solução possivel de ponto extremo. Para esse caso obteremos, para qualquer $\Theta > 0$, uma solução possivel mas não de ponto extremo, associada aos vetores $\vec{A}_1$, $\vec{A}_2$, ... , $\vec{A}_m$, $\vec{A}_{m+1}$. Isso indica que o problema não tem solução mínima finita.

I.2.3- O Método Simplex.

Seja $\vec{x} = (x_1, x_2, \ldots, x_m)$ e o conjunto de vetores associados, linearmente independentes $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_m$.

Temos então:

$$x_1.\vec{A}_1 + x_2.\vec{A}_2 + \ldots + x_m.\vec{A}_m = \vec{A}_o \qquad (1.8)$$

$$x_1.c_1 + x_2.c_2 + \ldots + x_m.c_m = z_o \qquad (1.9)$$

$$x_i \geqslant 0$$

c_i - são coeficientes de custo da função objetivo e z_o o valor correspondente da função objetivo para um $\vec{x}$ dado.

Sendo $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_m$ linearmente independente, pode-se exprimir qualquer vetor $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_n$ em função de $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_m$.

$$X_{1j}.\vec{A}_1 + X_{2j}.\vec{A}_2 + \ldots + X_{mj}.\vec{A}_m = \vec{A}_j \qquad (1.10)$$

$$(j = 1,2,\ldots,n)$$

e objetivo correspondente será:

$$X_{1j}.c_1 + X_{2j}.c_2 + \ldots + X_{mj}.c_m = z_j \qquad (1.11)$$

c_i - são os coeficientes correspondentes a $\vec{A}_i$.

Multiplicando-se (1.10) e (1.11) por Θ e subtraindo de (1.8) e (1.9) respectivamente teremos:

$$(x_1 - \Theta.X_{1j}).\vec{A}_1 + \ldots + (x_m - \Theta.X_{mj}).\vec{A}_m + \Theta.\vec{A}_j = \vec{A}_o \qquad (1.12)$$

$$(x_1 - \Theta.X_{1j}).c_1 + \ldots + (x_m - \Theta.X_{mj}).c_m + \Theta.c_j = z_o - \Theta.(z_j - c_j) \qquad (1.13)$$

foi somado em ambos os lados da equação o valor $\Theta.c_j$.

Se todos os coeficientes de $\vec{A}_1, \ldots, \vec{A}_m, \vec{A}_j$ em (1.12) forem não negativos, temos então uma nova solução possivel em que o valor da função objetivo será: $z = z_o - \Theta.(z_j - c_j)$. Como $x_1, x_2, \ldots, x_m$ são positivos, temos $\Theta > 0$ tal que os coeficientes de (1.12) sejam positivos.

Como por hipótese $(z_j - c_j) > 0$, para a coluna j, temos:

$$z = z_o - \Theta.(z_j - c_j) < z_o \qquad \text{para } \Theta > 0.$$

Se pelo menos um $X_{ij} > 0$ em (1.10) para ($i = 1, 2, \ldots, m$) o maior valor de Θ para que os coeficientes de (1.12) sejam não negativos será:

$$\Theta_o = \min_i \frac{x_i}{X_{ij}} > 0 \qquad \text{para} \quad X_{ij} > 0.$$

Substituindo-se Θ_o em (1.12) e (1.13) o coeficiente correspondente a i se anulará. Temos então uma nova base constituida de A_j e $\underline{m-1}$ vetores da base original.

Se nessa nova base ainda encontrarmos $z_j - c_j > 0$ e um correspondente $X_{ij} > 0$ pode obter-se uma nova solução que levará a uma função objetivo ainda menor. Esse processo continua até que todos os valores de $z_j - c_j \leqslant 0$ ou $z_j - c_j > 0$ e todos os valores de $X_{ij} \leqslant 0$.

Se todos os valores obtidos para $z_j - c_j \leqslant 0$, o processo termina e a solução obtida nesta iteração será a solução ótima, marcando assim o fim do processamento.

TEOREMA 1.- Se para qualquer j (fixo), a condição $z_j - c_j > 0$ se verifica, então pode-se construir um conjunto de soluções possiveis tal que $z < z_o$ para qualquer membro do conjunto de soluções possiveis, onde o limite inferior de z pode ser finito ou infinito. (z é o valor da F.O. para uma solução particular do conjunto de soluções possiveis).

CASO 1.- Se o limite INFERIOR é FINITO, pode-se construir uma nova solução possivel com exatamente m variaveis positivas e cujo valor da Função Objetivo é menor que o valor para a solução precedente.

CASO 2.- Se o limite INFERIOR é INFINITO, pode-se construir uma nova solução possivel com m+1 variaveis positivas cujo valor da Função Objetivo pode ser arbitrariamente pequeno.

TEOREMA 2.- Se para qualquer solução básica possivel $\vec{x} = (x_1, \ldots, x_m)$ as condições $z_j - c_j \leqslant 0$ se verificam para $j = 1,2,\ldots,n$ então (1.4) e (1.5) constituem uma solução ótima (mínima).

As provas desses teoremas podem ser encontrades em [4] às páginas 74 e 75.

Para facilidade do cálculo manual desse procedimento, aconselha-se trabalhar em forma de tabela:

$$(\vec{A}_o \mid \vec{A}_1 \vec{A}_2 \ldots \vec{A}_m \mid \vec{A}_{m+1} \ldots \vec{A}_n)$$

fazendo $B = \vec{A}_1 . \vec{A}_2 . \ldots . \vec{A}_m$ (base admissivel)

teremos: $\vec{x} = B^{-1} . \vec{A}_o$ e $\vec{X}_j = B^{-1} . \vec{A}_j$ onde;

$\vec{x} = (x_1, x_2, \ldots, x_m)$ e $\vec{X}_j = (X_{1j}, X_{2j}, \ldots, X_{mj})$

$x_i \geqslant 0$

Agrupando-se os vetores e multiplicando-se por B^{-1} teremos:

$$(\vec{x} \mid I_m \mid \vec{X}_{m+1} \ldots \vec{X}_n)$$

i	base	$\vec{c}$	$\vec{A}_o$	$c_1 \ldots c_k \ldots c_m$	$c_{m+1} \ldots c_n$
				$\vec{A}_1 \ldots \vec{A}_k \ldots \vec{A}_m$	$\vec{A}_{m+1} \ldots \vec{A}_n$
1	A_1	c_1	x_1	$1 \ldots 0 \ldots 0$	$X_{1,m+1} \ldots X_{1,n}$
2	A_2	c_2	x_2	$0 \ldots 0 \ldots 0$	$X_{2,m+1} \ldots X_{2,n}$
.	.	.	.	$. \ldots . \ldots .$	$. \ldots .$
k	A_k	c_k	x_k	$0 \ldots 1 \ldots 0$	$X_{k,m+1} \ldots X_{k,n}$
.	.	.	.	$. \ldots . \ldots .$	$. \ldots .$
m	AA_m	c_m	x_m	$0 \ldots 0 \ldots 1$	$X_{m,m+1} \ldots X_{m,n}$
m+1			z	$0 \ldots 0 \ldots 0$	$z_{m+1} \ldots z_n$ $- c_{m+1} \quad - c_n$

tab. 1.1 - Tabela Simplex.

Inicialmente:

$$x_i = b_i \quad , \quad X_{ij} = a_{ij} \quad , \quad z_o = \sum_{i=1}^{m} c_i . x_i$$

$$z_j = \sum_{i=1}^{m} c_i . X_{ij} \qquad (j = 1 , 2 , \ldots , n)$$

Se todos os $z_j - c_j \leqslant 0 \Rightarrow \vec{x} = \vec{b}$ é solução ótima.

Deve ser introduzida na base qualquer vetor que tenha o correspondente $z_j - c_j > 0$, teoricamente.

Supoem-se que o número de iterações será menor , se se tomar o $\max_j \Theta_o . (z_j - c_j)$ (Dantzig).

O critério mais usado é o de introduzir o vetor que tem o $\max_j (z_j - c_j) = (z_k - c_k)$. No caso de empate introduz-se o de menor índice j .

Para se determinar a solução que vai seir da base usa-se $\Theta_o = \min_i (x_i / X_{ik})$ para $X_{ik} > 0$.

RESUMO: Após construida a tebela inicial o procedimento constitui-se dos sequintes passos:

PASSO 1.- Verificar os velores de $z_j - c_j$ para determinar se se obteve a solução ÓTIMA (mínima). Se todos os $z_j - c_j \leqslant 0$, esteremos com a solução ÓTIMA.

PASSO 2.- Seleção do vetor que deve entrar na base. Vetor cujo $z_j - c_j > 0$, seja máximo (espera-se que o nº de iterações seja mínimo).

PASSO 3.- Seleção do vetor que vai ser eliminado da base. Escolher o vetor correspondente ao $\Theta_o = \min_i (x_i / X_{ik})$, para $X_{ik} > 0$ onde k corresponde ao vetor selecionado no passo 2. Se todos os $X_{ik} \leqslant 0$, então a solução é ILIMITADA.

PASSO 4.- Transformação da tebela pelo processo de eliminação de GAUSS, completa, para se obter a nova solução e elementos associados. Cada uma dessas iterações produz uma nova solução básica, que pelos teoremas 1 e 2, obteremos uma solução MÍNIMA ou determinamos que ela é ILIMITADA.

I.2.4- Base Artificiel.

Até este ponto, foi suposto que o probleme continha sempre uma matriz identidade que pudesse ser utilizada como base pera a obtenção da solução possivel inicial. Supondo que o sisteme não apresente essa facilidade lança-se mão do seguinte artifício.

Seja o seguinte probleme de programação linear:

Minimizar: $z = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n$

Sujeito à:

$$\begin{aligned}
a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\
a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\
\ldots\ldots \quad \ldots\ldots \quad \ldots \quad \ldots\ldots \quad & \ldots \\
a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n &= b_m
\end{aligned}$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n)$$

Usando-se o artifício da base artificiel o problema fica alterado para o seguinte:

Minimizar:

$$z = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n + w \cdot x_{n+1} + \ldots + w \cdot x_{n+m}$$

Sujeito à:

$$\begin{aligned}
a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n + x_{n+1} &= b_1 \\
a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n + x_{n+2} &= b_2 \\
\ldots\ldots \quad \ldots\ldots \quad \ldots \quad \ldots\ldots \quad \ldots & \quad \ldots \\
a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n + x_{n+m} &= b_m
\end{aligned}$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n, n+1, \ldots, n+m)$$

O vetores $\vec{A}_{n+1}$, ... , $\vec{A}_{n+m}$ formam uma base artificial para o sistema aumentado.

Se existir pelo menos uma solução possivel para o problema original, essa solução tambem será possivel para o sistema aumentado. O Método Simplex então, fornecerá uma solução mínima para o sisteme, forçando o não aparecimento de valor positivo para as variaveis artificiais x_{n+i} . Se o problema original não tiver solução possivel ótima, então a solução ótima do problema aumentado conterá pelo menos uma $x_{n+i} > 0$.

Para o problema aumentado a primeira solução possivel será:

$$\vec{x} = (x_{n+1} , x_{n+2} , \ldots , x_{n+m}) = (b_1 , b_2 , \ldots , b_m)$$

com o valor da função objetivo : $z_o = w. \sum_{i=1}^{m} b_i$

Visto que são vetores artificiais na base, cada $z_j - c_j$ será uma função linear de w.

Para a primeira solução teremos:

$$z_j - c_j = w. \sum_{i=1}^{m} x_{ij} - c_j$$

para cada $z_j - c_j$ existirá um coeficiente $\underline{w}$ e um coeficiente independente de $\underline{w}$.

Para esse caso coloca-se na tabela do processo Simplex, duas linhas adicionais, correspondentes a esses fatores, que serão a $(m+1)^{ésima}$ e $(m+2)^{ésima}$ linhas.

A nova tabela do processo Simplex será:

i	base	c	A_o	c_1	...	c_n	w	...	w	...	w
				A_1	...	A_n	A_{n+1}	...	A_{n+k}	...	A_{n+m}
1	A_{n+1}	w	x_{n+1}	x_{11}	...	X_{1n}	1	...	0	...	0
2	A_{n+2}	w	x_{n+2}	X_{21}	...	X_{2n}	0	...	0	...	0
.	....	.	....	...	...	...	.	...	.	...	.
.	....	.	....	...	...	...	.	...	.	...	.
k	A_{n+k}	w	x_{n+k}	X_{k1}	...	X_{kn}	0	...	1	...	0
.	....	.	....	...	...	...	.	...	.	...	.
.	....	.	....	...	...	...	.	...	.	...	.
m	A_{n+m}	w	x_{n+m}	X_{m1}	...	X_{mn}	0	...	0	...	1
m+1			0	$-c_1$	...	$-c_n$	0	...	0	...	0
m+2			x_{n+i}	X_{i1}	...	X_{in}	0	...	0	...	0

tab. 1.2 - Tabela Simplex com base Artificial.

O procedimento será o mesmo que no processo da tabela enterior com uma única diferença. O vetor a ser introduzido na base, será selecionado pelo maior elemento positivo da linha (m+2). Esta linha deverá sofrer tambem a transformação usual de eliminação. Um vetor das varieveis artificiais eliminado da base nunca deverá voltar à base.

Usa-se a linha (m+2) para selecionar os vetores que entrarão na base até que:

1.- Todos os vetores artificiais sejam eliminados da base.

2.- Não exista $(m+2)^{ésimo}$ elemento positivo.

A primeira alternativa implica em que todos os elementos da linha (m+2) sejam iguais a zero e a correspondente base seja possivel para o problema original. Em seguida, aplica-se o algorítmo simplex, visto anteriormente, até que a solução ótima seja obtida. Na segunda alternativa, se o elemento (m+2 , 0) , parte artificial de valor correspondente na função objetivo, for maior que zero, então o problema original terá uma solução possivel degenerada que conterá pelo menos um vetor artificial na base

Se as condições apresentadas anteriormente forem satisfeitas e se a solução obtida não for ótima, o processo continua pelo simplex normal, utilizando-se a $(m+1)^{ésima}$ linha da tabela.

Sempre que um problema contiver um vetor unitário, este deverá fazer parte da base inicial para que seja reduzido o número de iterações.

I.2.5- Método Simplex Revisado.

1.2.5.1- Forma geral da inversa.

O elemento principal que permite passar de uma solução básica para outra é o conhecimento explícito da representação dos vetores que não se encontram na base em termos da base atual. Dado essa observação pode-se dizer o seguinte:

1.- Calcular os elementos $z_j - c_j$ para determinar o vetor que deverá ser introduzido na base, ou para determinar se a solução atual é ótima.

2.- Determinar que vetor deverá ser eliminado da base.

3.- Transformar a base e obter-se a nova solução.

Como já se viu anteriormente, dado uma base B, de vetores m-dimensionais ($\vec{A}_1$, $\vec{A}_2$, ... , $\vec{A}_m$), e combinação linear que exprime qualquer vetor $\vec{A}_j$ em termos de B é determinado por:

(1.14) $\vec{X}_j = B^{-1}.\vec{A}_j$ onde $\vec{X}_j = (X_{1j} , X_{2j} , \ldots , X_{mj})$ é um vetor coluna. Temos então:

$\vec{A}_j = X_{1j}.\vec{A}_1 + X_{2j}.\vec{A}_2 + \ldots + X_{mj}.\vec{A}_m$ que é a combinação linear desejada.

Seja o problema de programação linear seguinte:

Minimizar: $z = \vec{c}.\vec{x}$

Sujeito à: $A.\vec{x} = \vec{b}$ e $\vec{x} \geqslant \vec{0}$

Seja B correspondente aos m-primeitos vetores de A.

$$B.\vec{x}_o = \vec{b}$$

$$\vec{x}_o \geqslant \vec{0}$$ onde,

$\vec{x}_o = (x_1 , x_2 , \ldots , x_m)$ é uma solução possivel básica.

$$\vec{x}_o = B^{-1}.\vec{b} \qquad (1.15)$$

A combinação linear de todos os vetores de A em termos de B pode ser determinada mediante $\vec{x}_j = B^{-1}.\vec{A}_j$ para $j = 1, 2, \ldots, n$.

Definimos $z_j = c_1.X_{1j} + \ldots + c_m.X_{mj}$ (1.16) para $j = 1, 2, \ldots, n$ onde c_i são coeficientes de custo.

Por (1.14) temos que (1.16) pode ser escrito na seguinte forma:

$z_j = (\vec{c}_o , \vec{X}_j) = \vec{c}_o . B^{-1} . \vec{A}_j$ ($j = 1, 2, \ldots, n$) onde $\vec{c}_o = (c_1 , c_2 , \ldots , c_m)$ é um vetor linha.

Portanto, dado $\vec{c}_o . B^{-1}$ para uma possivel base B podemos calcular o correspondente z_j atravéś de:

$$\vec{x}_j = B^{-1} . \vec{A}_j \quad ; \quad \vec{x}_o = B^{-1} . \vec{b} \quad \text{e} \quad \vec{c}_o . B^{-1} .$$

Nota-se que para se obter uma solução possivel de outra já conhecida, é necessário para cada base B ter-se:

B^{-1} , A , $\vec{b}$ e $\vec{c}$ (dados originais).

Baseado nesse princípio, desenvolveu-se o Método Simplex Revisado.

A diferença principal entre o Simplex Original e o Revisado é que no primeiro transformamos todos os elementos da tabela por meio das fórmulas de eliminação de Gauss, e no segundo transformamos sàmente os elementos da matriz inversa. Este segundo método é mais indicado para o uso em computadores porque

1.- Utiliza-se os coeficientes originais, oque permite executar as multiplicações com apenas os elementos diferentes de Zero, reduzindo-se os cálculos em matrizes esparsas, com muitos zeros em seu interior. Os elementos originais podem ser ainda, armazenados na forma compacta (zeros não se armazenam) resultando em uma economia de memória e consequentemente aumento na capacidade de resolução.

2.- A quantidade de informações novas a se registrar é reduzida, visto que necessitamos apenas a inversa da base atual e do vetor solução.

OBTENÇÃO DA NOVA INVERSA: Seja B formada pelos vetores $(\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_\ell , \ldots , \vec{A}_m)$ que difere da nova base apenas pela substituição do vetor $\vec{A}_\ell$ pelo vetor $\vec{A}_k$.

Seja $\bar{B} = (\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_k , \ldots , \vec{A}_m)$ a nova base. Teremos então:

$$B^{-1} . B = B^{-1} . (\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_\ell , \ldots , \vec{A}_m) = I$$

$$B^{-1} . \bar{B} = B^{-1} . (\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_k , \ldots , \vec{A}_m) =$$

$$= \begin{pmatrix} 1 & 0 & \ldots & x_{1k} & \ldots & 0 \\ 0 & 1 & \ldots & x_{2k} & \ldots & 0 \\ . & . & \ldots & \ldots & \ldots & . \\ . & . & \ldots & \ldots & \ldots & . \\ 0 & 0 & \ldots & x_{mk} & \ldots & 1 \end{pmatrix}$$

Seja:

$\bar{b}_{ij} = (\bar{B}^{-1})_{ij}$ e $b_{ij} = (B^{-1})_{ij}$ teremos:

$$\begin{cases} \bar{b}_{ij} = b_{ij} - \dfrac{b_{\ell j}}{x_{\ell k}} \cdot x_{ik} & \text{para} \quad i \neq \ell \\ \\ \bar{b}_{ij} = \dfrac{b_{\ell j}}{x_{\ell k}} \end{cases}$$

PROCEDIMENTO SIMPLEX REVISADO : Seja o seguinte probleme de programação linear.

Maximizar x_{n+m+1}

Sujeito à:

$$x_j \geqslant 0 \quad (j = 1, 2, \ldots, n)$$

$$b_i \geqslant 0 \quad (i = 1, 2, \ldots, m)$$

$$x_{n+m+1} = -c_1 \cdot x_1 - c_2 \cdot x_2 - \ldots - c_n \cdot x_n$$

$$\begin{aligned}
a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\
a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\
\ldots\ldots \quad \ldots\ldots \quad \ldots \quad \ldots\ldots &\quad .. \\
a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n &= b_m \\
c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n + x_{n+m+1} &= 0
\end{aligned}$$

O Probleme EQUIVALENTE será o seguinte:

Minimizar $c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n$

Sujeito à

$$\begin{aligned}
a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\
a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\
\ldots\ldots \quad \ldots\ldots \quad \ldots \quad \ldots\ldots &\quad .. \\
a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n &= b_m
\end{aligned}$$

$$x_j \geqslant 0 \quad (j = 1, 2, \ldots, n)$$

Como no caso anterior, nesse processo tambem necessita-se de uma solução básica inicial. Para facilitar os calculos da Fase I coloca-se uma equeção redundante definida como

$$a_{m+2,1} \cdot x_1 + a_{m+2,2} \cdot x_2 + \ldots + a_{m+2,n} \cdot x_n + x_{m+n+2} = b_{m+2}$$

onde,

$$a_{m+2,j} = -\sum_{i=1}^{m} a_{ij} \qquad (j = 1,2, \ldots, n)$$

$$b_{m+2} = -\sum_{i=1}^{m} b_i$$

Se fizermos $c_j = a_{m+1,j}$, podemos escrever o seguinte problema:

Maximizar x_{n+m+1}

Sujeito à:

$$\begin{aligned}
a_{1,1} \cdot x_1 + \ldots + a_{1,n} \cdot x_n + x_{n+1} &= b_1 \\
\ldots\ldots\ldots \quad \ldots \quad \ldots\ldots\ldots \quad \ldots \quad &\ldots \\
a_{m,1} \cdot x_1 + \ldots + a_{m,n} \cdot x_n + x_{n+m} &= b_m \\
a_{m+1,1} \cdot x_1 + \ldots + a_{m+1,n} \cdot x_n + x_{n+m+1} &= 0 \\
a_{m+2,1} \cdot x_1 + \ldots + a_{m+2,n} \cdot x_n + x_{n+m+2} &= b_{m+2}
\end{aligned}$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n+m)$$

Algorítmo Simplex Revisado:

Passo 0.- Monta-se a seguinte matriz com os coeficientes,

$$\bar{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,k} & \cdots & a_{m,n} \\ a_{m+1,1} & a_{m+1,2} & \cdots & a_{m+1,k} & \cdots & a_{m+1,n} \\ a_{m+2,1} & a_{m+2,2} & \cdots & a_{m+2,k} & \cdots & a_{m+2,n} \end{pmatrix}$$

$\bar{A}_j$ = vetor coluna j de $\bar{A}$.

Nossa base inicial é a matriz identidade. Sua inversa será tambem a identidade e a informação se registra na seguinte matriz:

$$\left(\begin{array}{ccccccc|cc} 1 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 0 & \cdots & 1 & 0 & 0 \\ \hline 0 & 0 & \cdots & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 1 \end{array}\right)$$

FASE I - Variaveis artificiais na base com valores positivos.

Passo I.1 - Se $x_{m+n+2} < 0$, calcular, para $j = 1,2,\ldots, n$:

$$\delta_j = U_{m+2} \cdot \bar{A}_j = u_{m+2,1} \cdot a_{1j} + \ldots + u_{m+2,m+2} \cdot a_{m+2,j}$$

Vá para I.2 .

Se $x_{m+n+2} = 0$, Vá para o passo II.1 .

Passo I.2 - Se todos os $\delta_j \geq 0$, x_{n+m+2} é MÁXIMO e portanto não existe solução para o problema original.
Se pelo menos um $\delta_j < 0$, então a variavel a ser introduzida na base será $\vec{x}_k$, correspondente a $\delta_k = \min_j \delta_j$.

Passo I.3 - Calcular:

$$x_{ik} = U_i \cdot \bar{A}_k = u_{i1} \cdot a_{1k} + u_{i2} \cdot a_{2k} + \ldots + u_{i,m+2} \cdot a_{m+2,k}$$

para $i = 1,2,\ldots,m+2$. A variavel $\vec{x}_\ell$ deve ser eliminada correspondente a : $\theta_o = \min_i \frac{x_i}{X_{ik}} = \frac{x_\ell}{X_{\ell k}}$.
($i = 1,2,\ldots,m$).
Usa-se apenas as $X_{ik} > 0$.

Passo I.4 - Os novos valores das variaveis na solução básica se obtem com as fórmulas:

$$x'_i = x_i - \frac{x_\ell}{X_{\ell k}} \cdot X_{ik} \quad \text{para} \quad i \neq k$$

$$x'_k = \frac{x_\ell}{X_{\ell k}}$$

e os novos valores de U são transformados com:

$$u'_{ij} = u_{ij} - \frac{u_{\ell j}}{x_{\ell k}} \cdot x_{ik} \qquad \text{para} \quad i \neq \ell$$

$$u'_{\ell j} = \frac{u_{\ell j}}{x_{\ell k}}$$

Vá para o Passo I.1 .

FASE II - Nenhuma variavel artificial com valores positivos.

Passo II.1- Calcular:

$$\gamma_j = U_{m+1} \cdot \bar{A}_j = u_{m+1,1} \cdot a_{1j} + \cdots + u_{m+1,m+2} \cdot a_{m+2,j}$$

$$(\quad j = 1,2,\ldots,n \;)$$

Passo II.2- Se todos os $\gamma_j \geq 0$ então x_{n+m+1} é máximo e a solução correspondente é ÓTIMA. O valor negativo de x_{n+m+1} é o valor verdadeiro da função objetivo a ser minimizada. Se pelo menos um $\gamma_j < 0$, então : $\gamma_k = \min_j \gamma_j$. A variavel $\vec{x}_k$ deve ser introduzida na base.

Passo II.3- Calcular:

$$x_{ik} = U_i \cdot \bar{A}_k = u_{i1} \cdot a_{ik} + \cdots + u_{i,m+2} \cdot a_{m+2,k}$$

para $i = i,2,\ldots,m+2$. A variavel $\vec{x}_\ell$ que deve ser eliminada é indicada por:

$$\theta_o = \min_i \frac{x_i}{x_{ik}} = \frac{x_\ell}{x_{\ell k}} \qquad i = 1,2,\ldots,m$$

para $x_{ik} > 0$. Se todas as $x_{ik} < 0$ - Solução ILIMITADA.

Passo II.4- Obtem-se os novos valores:

$$x_i' = x_i - \frac{x_\ell}{x_{\ell k}} \cdot x_{ik} \qquad i \neq k$$

$$x_k' = \frac{x_\ell}{x_{\ell k}}$$

$$u_{ij}' = u_{ij} - \frac{u_{\ell j}}{x_{\ell k}} \cdot x_{ik} \qquad i \neq \ell$$

$$u_{\ell j}' = \frac{u_{\ell j}}{x_{\ell k}}$$

Repete-se até encontrar a solução OTIMA

(vá para o Passo II.1).

A seguir é apresentado a programação em linguagem FORTRAN dêsse algorítmo. A programação se compõem de três sub-programas e um programa principal. O programe principal recebe o nome de SIMPLX e os sub-programas têm as finalidades descritas a abaixo:

SIMPR - Subrotina, para determinar a solução ótima através do método Simplex Revisado. Está previsto um tempo máximo de 10 minutos, após o que a solução é interrompida.

PVMK - Função Real, programada para determinar o produto escalar do vetor v pela coluna k da matriz A.

PMMK - Subrotina, para determinar o produto da matriz U pela coluna k da matriz A .

Os CARTÕES DE DADOS para o programa SIMPLX deverão ser apresentados da seguinte forma:

1º Cartão: Ordem e tipo do problema

Colunas	Descrição
1 a 5	Número de linhas (equações)
6 a 10	Número de colunas (incógnitas)
11 a 22	Tipo do problema (MAXIMO ou MINIMO)

2º Cartão: Termos independentes (limites)

Os valores devem ser perfurados da coluna 1 a 80 ocupando cada número, o máximo de 5 colunas (colocando-se o ponto decimal). Cada cartão conterá o máximo de 16 números e poderá ser utilizado quantos cartões forem necessários.

3º Cartão: Coeficientes da função objetivo

Idem ao ítem anterior.

4º Cartão: Matriz dos coeficientes, por linha.

São fornecidos da mesma forma que o ítem anterior sendo que os valores devem ser perfurados por linha, iniciando - se um novo cartão a cada nova linha.

É apresentado a seguir a resolução de um exemplo, onde são fornecidos os cartões de dados correspondente, e a solução obtida pelo computador.

Segue tambem a listagem dos programas e sub - programas auxiliares.

Listagem dos dados fornecida pelo programa :

```
        TERMOS INDEPENDENTES (LIMITES)

 0.3000000E+01  0.1000000E+01  0.3000000E+01  0.2000000E+01  0.1100000E+02


        COEFICIENTES DA FUNCAO OBJETIVO

-0.4000000E+01  0.5000000E+01 -0.3000000E+01  0.7000000E+01  0.1000000E+02 -0.9000000E+01  0.4000000E+01 -0.3000000E+01
-0.9000000E+01 -0.1000000E+02


        RESTRICOES POR LINHA    5 X   10

 0.5000000E+01  0.5000000E+01 -0.6000000E+01  0.7000000E+01  0.4000000E+01 -0.3000000E+01  0.1400000E+02 -0.1500000E+02
 0.2000000E+01  0.1000000E+01

 0.6000000E+01 -0.7000000E+01  0.8000000E+01 -0.6000000E+01 -0.2000000E+01  0.4000000E+01  0.2000000E+01  0.7000000E+02
 0.4000000E+01 -0.1000000E+02

 0.1500000E+02 -0.7000000E+01  0.8000000E+01  0.8000000E+01  0.4000000E+01 -0.3000000E+01  0.1100000E+02  0.1300000E+02
 0.9000000E+01  0.0000000E+00

 0.1200000E+02 -0.1400000E+02  0.1200000E+02 -0.1200000E+02  0.4000000E+01 -0.3000000E+01 -0.4000000E+01  0.1200000E+02
-0.1300000E+02  0.1100000E+02

 0.2000000E+02  0.4000000E+02  0.4000000E+01  0.3000000E+01 -0.4000000E+01  0.1300000E+02  0.1400000E+02 -0.3500000E+02
 0.4200000E+02  0.3800000E+02


        SOLUCAO V.ARTIFICIAL

        NUMERO DE ITERACOES=   211

        PROBLEMA DE MINIMO

        FUNCAO OBJETIVO=  0.3817822E+01

        TEMPO DE EXECUCAO =  35.853 SEGUNDOS

        VETOR SOLUCAO

        X(  5)=  0.1072494E+01
        X(  1)=  0.9762556E-01
        X(  7)=  0.1037749E-01
        X(  2)=  0.7875810E-01
        X(  6)=  0.7728406E+00
```

Os cartões de dados serão os seguintes:

```
 5   10MINIMO
 3.   1.   3.   2.  11.
-4.   5.  -3.   7. -10.  -9.   4.  -3.  -9. -10.
 5.   5.  -6.   7.   4.  -3.  14. -15.   2.   1.
 6.  -7.   8.  -6.  -2.   4.   2.  20.   4. -10.
15.  -7.   8.   8.   4.  -3.  11.  13.   9.   0.
12. -14.  12. -12.   4.  -3.  -4  12. -13.  11.
20.  40.   4.   3.  -4.  13.  14. -35.  42.  38.
```

```
        IDENT SIMPLX
        FILE  2=DSIMPL,UNIT=READER
        FILE  5=PSIMPL,UNIT=PRINTER
        SIZE  ALPHA=12
FLAG  141** THIS CONTROL CARD WILL BE IGNORED IN THE NEXT RELEASE
(0001)        DIMENSION A(2000),U(2000),X(50),C(50),IND(50)
(0002)        ALPHA ICON, MAX/"MAXIMO      "/,COND(5)
FLAG  142** THIS TYPE STATEMENT WILL NOT BE AVAILABLE NEXT RELEASE
(0003)        DATA COND/"OTIMA       ","IMPOSSIVEL-1","IMPOSSIVEL-2",
             A"V.ARTIFICIAL","INTERROMPIDA"
(0004)        ID(I,J,N)=(J-1)*N+I
(0005)        PREC=1.E-7
        C         M= NUMERO DE COLUNAS
        C         N= NUMERO DE LINHAS
        C         ICON= TIPO DE PROBLEMA ( MAXIMO OU MINIMO )
(0006)  11111 READ(2,10000,END=99999)N,M,ICON
(0007)  10000 FORMAT(2I5,T11,A12)
        C         LEITURA DOS TERMOS INDEPENDENTES
(0008)        READ(2,20000)(X(I),I=1,N)
        C         LEITURA DOS COEFICIENTES DE CUSTO
(0009)        READ(2,20000)(C(I),I=1,M)
(0010)  20000 FORMAT(16F5.0)
(0011)        NM=N*M
(0012)        DO 100 I=1,N
(0013)  100   READ(2,20000)(A(J),J=I,NM,N)
        C         LEITURA DAS RESTRICOES POR LINHA (1/CARTAO)
(0014)        ICOD=2
(0015)        IF(ICON.NE.MAX)GO TO 200
(0017)        ICOD=1
        C         MAX = ICOD=1 , MIN= ICOD=2
        C         IMPRESSAO DOS DADOS LIDOS
(0018)  200   WRITE(5,30000)(X(I),I=1,N)
(0019)  30000 FORMAT(1H1,10X,'TERMOS INDEPENDENTES (LIMITES)',//(1X,8E15.7))
(0020)        WRITE(5,40000)(C(I),I=1,M)
(0021)  40000 FORMAT(////11X,'COEFICIENTES DA FUNCAO OBJETIVO',//(1X,8E15.7))
(0022)        WRITE(5,50000)N,M
(0023)  50000 FORMAT(////11X,'RESTRICOES POR LINHAS  I5,' X',I5,' / ')
(0024)        DO 300 I=1,N
(0025)  300   WRITE(5,60000)(A(J),J=I,NM,N )
(0026)  60000 FORMAT(/(1X,8E15.7))
        C         CHAMADA PARA CALCULO PELO SIMPLEX REVISADO
(0027)        CALL SIMPR(A,C,X,N,M,PREC,ICOD,NI,IR,Z,IND,U,TEMPO)
(0028)        WRITE(5,70000)COND(IR),NI,ICON,Z,TEMPO
(0029)  70000 FORMAT(////11X,'SOLUCAO ',A12//11X,'NUMERO DE ITERACOES=',I6
             1//11X,'PROBLEMA DE ',A12 //11X,'FUNCAO OBJETIVO=',E15.7 //
             211X,'TEMPO DE EXECUCAO =',F8.3,' SEGUNDOS',///11X,'VETOR SOLUCAO',/)
(0030)        WRITE(5,80000)(IND(I),X(I),I=1,N)
(0031)  80000 FORMAT(11X,'X(',I3,')=',E15.7 )
(0032)        GO TO 11111
(0033)  99999 STOP
(0034)        END


 5/12/77  10:11 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER ASR 5.6        2 FLAGS     0 ERRORS
ELAPSED TIME    44 SECS      49 CARDS AT   66 C.P.M.


COMMON =      0  DATA =  50732  TEMPORARIES =     36
CODE =   1994     DIGITS
```

```
        IDENT SIMPR
(0001)        SUBROUTINE SIMPR(A,C,X,N,M,PREC,ICOD,NI,IR,Z,IND,U,TEMPO)
(0002)        DIMENSION A(1),U(1),X(1),C(1),IND(1)
        C         SUBROTINA PARA METODO SIMPLEX REVISADO
        C                   IR=1 , SOLUCAO  OTIMA
        C                   IR=2 , SOLUCAO  IMPOSSIVEL
        C                   IR=3 , SOLUCAO  PIVO INFINIT -IMPOSSIVEL
        C                   IR=4 , SOLUCAO  V.ARTIFICIAL
        C                   IR=5 , SOLUCAO  INTERROMPIDA
(0003)        IMP=600.
(0004)        CALL TIME(0.,TEMP1)
(0005)        IK=1
(0006)        IR=1
(0007)        JS=N*M
(0008)        IY=JS+M
(0009)        IC=N*N
(0010)        IS=IC+M
(0011)        N2=IY+N+2
(0012)        N1=N2-1
(0013)        NF=IS+N
(0014)        IS1=IS+1
(0015)        IC1=IC+1
(0016)        IY1=IY+1
(0017)        I=0.
(0018)        Z=0.
(0019)        NI=0
(0020)        DO 100 I=1,NF
(0021)  100   U(I)=0.
(0022)        DO 110 I=1,N
(0023)        K=(I-1)*N+I
(0024)        U(K)=1.
(0025)        I=I+X(I)
(0026)  110   IND(I)=I-M
(0027)        I=-I
(0028)        DO 130 J=1,M
(0029)        K=(J-1)*N+1
(0030)        II=JS+J
(0031)        A(II)=A(K)
(0032)        DO 120 I=2,N
(0033)        K=K+1
(0034)  120   A(II)=A(II)+A(K)
(0035)  130   A(II)=-A(II)
(0036)        GO TO(140,22222),ICOD
(0037)  140   DO 150 J=1,M
(0038)  150   C(J)=-C(J)
(0039)  22222 NI=NI+1
(0040)        GO TO(170,190),IK
(0041)  170   IF(ABS(I).LE.PREC) GO TO 180
(0043)        IK=1
(0044)        GO TO 190
(0045)  180   IK=2
(0046)  190   K=0
(0047)        A(N2)=1.E98
(0048)        DO 230 J=1,M
(0049)        GO TO(200,210),IK
(0050)  200   JJ=JS+J
```

```
(0051)          XY=PVMK(U(IS1),A,J,N)+A(JJ)
(0052)          GO TO 220
(0053)    210   XY=PVMK(U(IC1),A,J,N)+C(J)
(0054)    220   IF(A(N2)+LE+XY) GO TO 230
(0056)          A(N2)=XY
(0057)          K=J
(0058)    230   CONTINUE
          C           SE K MENOR OU IGUAL A ZERO  IR=2  = FIM
(0059)          IF(K+GT+0) GO TO 270
(0061)          IR=2
(0062)    99999 GO TO(260,250),ICOD
(0063)    250   Z=-Z
(0064)    260   RETURN
(0065)    270   IF(A(N2)+PREC)300,280,280
(0066)    280   GO TO(290,99999),IK
(0067)    290   IR=4
(0068)          GO TO 99999
(0069)    300   CALL PMMK(U,A,K,N,A(IY1))
(0070)          A(N1)=PVMK(U(IC1),A,K,N)+C(K)
(0071)          PIVO=1+E9R
(0072)          L=0
(0073)          DO 310 I=1,N
(0074)          II=IY+1
(0075)          IF(A(II)+LE+0+) GO TO 310
(0077)          P=Y(I)/A(JI)
(0078)          IF(PIVO+LF+P) GO TO 310
(0080)          L=I
(0081)          PIVO=P
(0082)    310   CONTINUE
(0083)          IF(L+GT+0) GO TO 320
(0085)          IR=3
(0086)          GO TO 99999
(0087)    320   IND(L)=K
(0088)          II=IY+L
(0089)          PIVO=A(II)
(0090)          X(L)=X(L)/PIVO
(0091)          DO 330 J=1,N
(0092)          LJ=(J-1)*N+L
(0093)    330   U(LJ)=U(LJ)/PIVO
(0094)          DO 350 I=1,N
(0095)          LI=(I-1)*N+L
(0096)          II=IC+I
(0097)          I2=IS+I
(0098)          I3=IY+I
(0099)          U(I1)=U(I1)-U(LI)*A(N1)
(0100)          U(I2)=U(I2)-U(LI)*A(N2)
(0101)          IF(L+EQ+I) GO TO 350
(0103)          DO 340 J=1,N
(0104)          JA=(J-1)*N
(0105)          IJ=JA+I
(0106)          LJ=JA+L
(0107)    340   U(IJ)=U(IJ)-U(LJ)*A(I3)
(0108)          X(I)=X(I)-X(L)*A(I3)
(0109)    350   CONTINUE
(0110)          Z=Z-X(L)*A(N1)
(0111)          T=T-X(L)*A(N2)
(0112)          CALL TIME(TEMP1,TEMPO)
(0113)          TEMPO=TEMPO/1000
(0114)          IF(TEMPO+LE+TMP) GO TO 22222
(0116)          IR=5
(0117)          GO TO 99999
(0118)          END
```

```
 5/12/77  10:12 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER: ASR 5.6        0 FLAGS     0 ERRORS
ELAPSED TIME     19 SECS       118 CARDS AT  372 C.P.M.
COMMON =       0  DATA =     306  TEMPORARIES =      62
CODE =    8532     DIGITS
```

```
          IDENT PMMK
(0001)          SUBROUTINE  PMMK (U,A,K,N,Y)
(0002)          DIMENSION  U(1),A(1),Y(1)
          C           SUBROUTINA PARA PRODUTO
          C           PRODUTO  DA MATRIZ U PELA COLUNA  K  DE  A
(0003)          L=(K-1)*N
(0004)          DO 100 I=1,N
(0005)          Y(I)=0.
(0006)          DO 100 J=1,N
(0007)          IJ=(J-1)*N+I
(0008)          LJ=L+J
(0009)    100   Y(I)=Y(I)+U(IJ)*A(LJ)
(0010)          RETURN
(0011)          END
```

```
 5/12/77  10:12 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER: ASR 5.6        0 FLAGS     0 ERRORS
ELAPSED TIME      5 SECS       14 CARDS AT  168 C.P.M.
COMMON =       0  DATA =      48  TEMPORARIES =      50
CODE =     726     DIGITS
```

```
          IDENT PVMK
(0001)          FUNCTION  PVMK (V,A,K,N)
          C           FUNCAO    PARA O PRODUTO ESCALAR
          C           PRODUTO DO VETOR V  PELA COLUNA  R  DA MATRIZ  A '
(0002)          DIMENSION  V(1),A(1)
(0003)          L=(K-1)*N
(0004)          PVMK=0
(0005)          DO 100 I=1,N
(0006)          L=L+1
(0007)    100   PVMK=PVMK+V(I)*A(L)
(0008)          RETURN
(0009)          END
```

```
 5/12/77  10:12 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER: ASR 5.6        0 FLAGS     0 ERRORS
ELAPSED TIME      3 SECS       12 CARDS AT  240 C.P.M.
COMMON =       0  DATA =      24  TEMPORARIES =      50
CODE =     412     DIGITS
```

1.2.5.2- Forma do Produto de Inversa.

Seja:

$$B = (\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_\ell , \ldots , \vec{A}_m)$$

$$\bar{B} = (\vec{A}_1 , \vec{A}_2 , \ldots , \vec{A}_k , \ldots , \vec{A}_m)$$

$$\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1\ell} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2\ell} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{\ell 1} & b_{\ell 2} & \cdots & b_{\ell\ell} & \cdots & b_{\ell m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{m\ell} & \cdots & b_{mm} \end{pmatrix} = B^{-1}$$

$$B^{-1} . \vec{A}_k = \vec{x}_k = \begin{pmatrix} x_{1k} \\ \cdots \\ x_{\ell k} \\ \cdots \\ x_{mk} \end{pmatrix}$$

$$\begin{pmatrix} \bar{b}_{11} & \bar{b}_{12} & \cdots & \bar{b}_{1\ell} & \cdots & \bar{b}_{1m} \\ \bar{b}_{21} & \bar{b}_{22} & \cdots & \bar{b}_{2\ell} & \cdots & \bar{b}_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \bar{b}_{\ell 1} & \bar{b}_{\ell 2} & \cdots & \bar{b}_{\ell\ell} & \cdots & \bar{b}_{\ell m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \bar{b}_{m1} & \bar{b}_{m2} & \cdots & \bar{b}_{m\ell} & \cdots & \bar{b}_{mm} \end{pmatrix} = \bar{B}^{-1}$$

onde:

$$\bar{b}_{ij} = b_{ij} - \frac{x_{ik}}{x_{\ell k}} \cdot b_{\ell j} \qquad (i \neq \ell)$$

$$\bar{b}_{\ell j} = b_{\ell j} \cdot \frac{1}{x_{\ell k}}$$

Formemos a matriz elementar $\underline{m \times m}$:

$$\begin{pmatrix} 1 & 0 & \dots & -\frac{x_{1k}}{x_{\ell k}} & \dots & 0 \\ 0 & 1 & \dots & -\frac{x_{2k}}{x_{\ell k}} & \dots & 0 \\ . & . & .... & \dots & \dots & . \\ 0 & 0 & \dots & +\frac{1}{x_{\ell k}} & \dots & 0 \\ . & . & \dots & \dots & \dots & . \\ 0 & 0 & \dots & -\frac{x_{mk}}{x_{\ell k}} & \dots & 1 \end{pmatrix} = E_{\ell}$$

O produto dessa matriz por B^{-1} resultará a matriz $\bar{B}^{-1}$

$E^{\ell} . B^{-1} = \bar{B}^{-1}$, equivale a aplicar as fórmulas do método anteiror.

Se fizermos:

$$y_{i\ell} = -\frac{x_{ik}}{x_{\ell k}} \quad (i \neq \ell) \quad \text{e} \quad y_{\ell\ell} = \frac{1}{x_{\ell k}}$$

poderemos armazenar na memória do computador apenas os valores

$$E^{\ell} = (\ell \; ; \; y_{1\ell} \; , \; y_{2\ell} \; , \; \cdots \; , \; y_{\ell\ell} \; , \; \cdots \; , \; y_{m\ell} \;)$$

Inicia-se o processo com B^{o} igual a identidade.

$$B_{o}^{-1} = I = E_{o}$$

A inversa da base seguinte será: $B_{1}^{-1} = E_{1}^{\ell} . E_{o}$

A inversa da p^{esima} base poderá ser obtida por:

$$E_{p}^{\ell} . E_{p-1}^{\ell} . \cdots . E_{1}^{\ell} . E_{o}^{\ell} = B_{p}^{-1}$$

Em cada estágio teremos um ℓ correspondente a coluna da metriz elementar. Aplica-se então o procedimento Simplex Revisado, utilizando essa técnica, obtendo-se uma melhor performance no processamento.

I.3- Dualidade em Programação Linear

A cada problema de programação linear, chamado PRIMAL, existe um outro associado a este chamado DUAL. A dualidade entre os dois problemas pode ser mostrada pela seguinte tabela:

PRIMAL	DUAL
minimize $z = \sum_{i=1}^{n} c_i . x_i$	maximize $z = \sum_{i=1}^{m} \pi_i . b_i$
sujeito à :	sujeito à:
$\sum_{j=1}^{n} a_{ij} . x_j = b_i \quad i \in E$	$\sum_{i=1}^{m} \pi_i . a_{ij} = c_j \quad j \in \bar{P}$
$\sum_{j=1}^{n} a_{ij} . x_j \geq b_i \quad i \in \bar{E}$	$\sum_{i=1}^{m} \pi_i . a_{ij} \leq c_j \quad j \in P$
$x_i \geq 0 \quad , \quad i \in P$	$\pi_i \geq 0 \quad , \quad i \in \bar{E}$
x_i qualquer, $i \in \bar{P}$	π_i qualquer, $i \in E$
$E \cup \bar{E} = (1,2,\ldots,m)$	Nota: π é um vetor linha
$P \cup \bar{P} = (1,2,\ldots,n)$	

tab. 1.3

Temos a seguinte tabela, para a determinação do DUAL (ou primal) baseando-se no PRIMAL (ou dual) respectivamente:

PRIMAL	DUAL
objetivo: $\vec{c}^t.\vec{x} \rightarrow$ mín.	objetivo: $\vec{\pi}\,\vec{b} \rightarrow$ máx.
variavel: $x_j \geq 0$	restrição: $\vec{\pi}\,P_j \leq c_j$ (ineq.)
variavel: x_j qualquer	restrição: $\vec{\pi}\,P_j = c_j$ (ig.)
restrição: $A_i.\vec{x} = b_i$ (ig.)	variavel: π_i qualquer
restrição: $A_i.x \geq b_i$ (in.)	variavel: $\pi_i \geq 0$
matriz coeficientes: A	matriz coeficientes: A^t
termos independentes: $\vec{b}$	termos independentes: $\vec{c}$
coeficientes custo: $\vec{c}$	coeficientes custo: $\vec{b}$

tab. 1.4

Nota: Se tomermos o DUAL do probleme DUAL, obteremos o probleme PRIMAL.

Atrevés de Tabela 1, pode-se construir duas cles ses de problemas PRIMAL-DUAL.

1-) Problemas Simétricos: ($\mathcal{E} = \bar{P} = \emptyset$)

PRIMAL	DUAL
minimize: $\vec{c}^t.\vec{x}$	maximize: $\vec{\pi}\,\vec{b}$
sujeito à:	sujeito à:
$A.\vec{x} \geq \vec{b}$	$A^t.\vec{\pi}^t \leq c$
$\vec{x} \geq 0$	$\vec{\pi} \geq 0$

tab 1.5

2-) Problemas Assimétricos: ($\bar{E} = \bar{P} = \phi$)

PRIMAL	DUAL
minimize: $\vec{c}^t.\vec{x}$	meximize: $\vec{\pi}.\vec{b}$
sujeito à:	sujeito à:
$A.\vec{x} = \vec{b}$	$A^t.\vec{\pi}^t \leqslant \vec{c}$
$\vec{x} \geqslant 0$	$\vec{\pi}$ quelquer

tab 1.6

Quelquer probleme primal pode ser transformado de uma desses formas para outra aplicando-se as seguintes regras:

1.- Transformar uma variavel quelquer na diferença de duas varieveis não negativas.

2.- Transformar uma igualdade em duas desigualdades opostas.

3.- Transformar uma desigualdade em igualdade utilizando-se varieveis de folga.

TEOREMA 1.- Se $\bar{x}$ e $\bar{\pi}$ são soluções possiveis para o primal e o duel, respectivamente, então:

$$\bar{z} = \bar{c}^t.\bar{x} \geqslant \bar{\pi}\,\bar{b} = \bar{v}$$

(prova- 5 , pag.47)

TEOREMA 2.- Se ambos os problemes (primal e o duel) têm soluções possiveis então ambos têm soluções ótimas e

$$\min z = \max v$$

Interpretação geométrica desses dois teoremas:

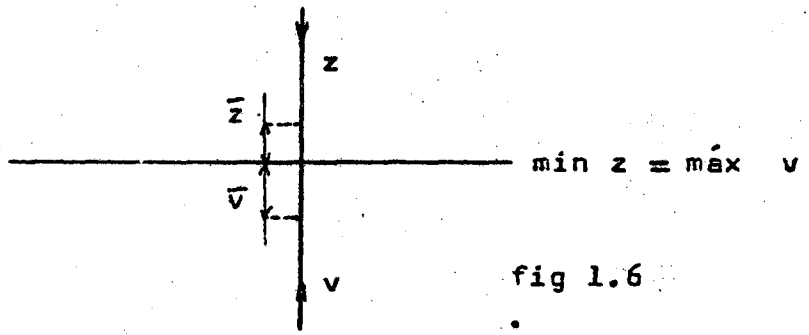


fig 1.6

As dues objetivos caminham pera o mesmo valor ótimo, mas por diferentes direções.

(prova- 5 , pag. 48)

APLICAÇÕES: Se tivermos que resolver o seguinte probleme:

$$\text{minimizar} \quad z = c_1 \cdot x_1 + c_2 \cdot x_2$$

sujeito è:

$$\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 &\geqslant b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 &\geqslant b_2 \\ \ldots \quad \ldots \quad &\quad \ldots \\ \ldots \quad \ldots \quad &\quad \ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 &\geqslant b_m \end{aligned}$$

pare $m = 20$.

Teremos um sistema de 20 (vinte) inequações com 2 (dues) incógnites. Transformando em igualdades utilizando es variaveis de folge, teremos uma matriz dos coeficientes de 20 linhes por 22 colunes. Pelo método Simplex Revisado teriamos uma matriz básica de 20 linhes por 20 colunes.

O probleme dual equivalente seria dado por:

$$\text{maximizar:} \quad v = b_1 \cdot \bar{\pi}_1 + b_2 \cdot \pi_2 + \ldots + b_{20} \cdot \pi_{20}$$

sujeito à:

$$e_{11}\cdot\pi_1 + e_{21}\cdot\pi_2 + \ldots + e_{20,1}\cdot\pi_{20} \leq c_1$$

$$e_{12}\cdot\pi_1 + e_{22}\cdot\pi_2 + \ldots + e_{20,2}\cdot\pi_{20} \leq c_2$$

Para esse problema teremos no método Simplex Revisado uma matriz básica de 2 (duas) linhas por 2 (duas) colunas, oque traz grandes vantagens na computação.

Os multiplicadores simplex (π) teriam neste caso a interpretação de coeficientes de custo.

COROLARIO: Se um dos problemas, primal ou dual, tiver solução ótima, o outro também a terá e o valor ótimo dos objetivos serão iguais.

(prova- 5 , pag. 49)

TEOREMA 3.- Se um dos problemas, primal ou dual, tiver uma solução ilimitada então o outro problema terá solução impossivel.

(prova- 5 , pag. 50)

Esses teoremas podem ser resumidos na seguinte tabela:

PRIMAL \ DUAL	POSSIVEL	IMPOSSIVEL
POSSIVEL	mín $\vec{c}^t.\vec{x}$ = máx $\vec{\pi}\vec{b}$	$\vec{c}^t.\vec{x} \longrightarrow -\infty$
IMPOSSÍVEL	$\vec{\pi}\vec{b} \longrightarrow +\infty$	possivel

tab 1.7

TEOREMA 4.- Um par $(\vec{x}_o ; \vec{\pi}_o)$ com $\vec{x}_o$ uma solução possivel para o primal e $\vec{\pi}_o$ para o dual, tem $\vec{x}_o$ e $\vec{\pi}_o$ como soluções ótimas do primal e dual, respectivamente, se e somente se,

$$(\vec{c}^t - \vec{\pi}_o . A) . \vec{x}_o = 0$$

(prova- 5 , pag. 50)

I.4- O Problema do Transporte.

Um produto deve ser embarcado nas quantidades a_1, a_2, ..., a_m, respectivamente, desde cada um dos m pontos de origem e devem ser recebidos em quantidades b_1, b_2, ..., b_n respectivamente por cada um dos n destinos.

O custo do transporte de cada unidade de origem i até o destino j é c_{ij} e é conhecido para todas as combinações (i , j). O problema é determinar as quantidades x_{ij}, que devem ser transportadas através de todas as rotas , com o objetivo de minimizar o custo de transporte.

Uma restrição imposta é que a quantidade total a ser transportada deve ser igual a quantidade a ser recebida, isto é :

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = A$$

O custo para transportar x_{ij} unidades será dado por $c_{ij} \cdot x_{ij}$. Como não tem sentido transporte negativo, tem-se a restrição $x_{ij} \geqslant 0$. Teremos a seguinte definição matemática para o problema:

Determinar x_{ij} que minimize o custo total,

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \qquad (1.17)$$

Sujeito às restrições:

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1, 2, \ldots, m) \qquad (1.18)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1, 2, \ldots, n) \qquad (1.19)$$

$$x_{ij} \geqslant 0 \qquad (1.20)$$

Para a consistência do sistema teremos que:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = A$$

O sistema de (1.17) a (1.20) é um problema de programação linear com $\underline{m+n}$ equeções com $\underline{m.n}$ varieveis.

TEOREMA 1.- O problema de transporte tem uma Solução Possivel.

TEOREMA 2.- Existe uma Solução Possivel com quanto muito $\underline{m+n-1}$ x_{ij} positivas.

TEOREMA 3.- Supondo que a_i e b_j são números inteiros não negativos, então qualquer solução básica possivel, (solução de ponto extremo) tem valores inteiros.

TEOREMA 4.- Sempre existe uma Solução Possivel , Mímima finita.

As provas correspondentes aos teoremas apresentados acima estão em (4).

VARIAÇÕES CLÁSSICAS DO PROBLEMA DE TRANSPORTES :

a) Problema de designação de pessoal.

Seja uma companhia que necessita preencher $\underline{n}$ postos que demanda diferentes habilidades e treinamento. Dispoem - se de $\underline{n}$ candidatos que podem ser empregados com salários identicos. A companhia faz uma estimativa, tendo como referência os serviços prestados por cada candidato, dando para cada um deles, uma certa nota.

Deseja-se designar os candidatos aos postos vagos e que o valor total (dos pontos) para a companhia seja máximo. Existem para esse problema n! designações possíveis.

Neste caso restringe-se os valores de x_{ij} a um e zero. Se:

$x_{ij} = 0 \rightarrow$ o candidato i não obtem o posto j

$x_{ij} = 1 \rightarrow$ o candidato i obtem o posto j.

A formulação matemática será a seguinte:

Sendo c_{ij} a contribuição que o candidato i daria ao posto j se fosse designado.

$$\text{maximizar} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij}$$

$$\text{sujeito à:} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,2,\ldots,n)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,2,\ldots,n)$$

$$x_{ij} \geq 0$$

Este problema é idêntico ao problema de transporte, bastando para isso minimizar o negativo da função objetivo.

Uma restrição imposta a esse tipo de problema é que o número de candidatos deve ser igual ao número de cargos oferecidos.

b) O probleme da concorrência

Em uma concorrência pública, as firmas concorrentes epresentam suas propostas onde se estabelece:

1- Preço por unidade do artigo ou artigos

2- As quantidades máximas e mínimas de cada artigo que podem ser produzidas(ou fornecidas) ao preço estebelecido.

3- Qualquer outra condição que se deseja impor.

Deverá ser contratado aquele que ofereça o custo total mínimo. Supondo que existam m concorrentes distintos, com possibilidades de satisfazer as necessidades de n armazens de depósito. O concorrente i deseja fornecer uma quantidade que não excede a_i e o deposito j requer a quantidade b_j.

Custa c_{ij}, comprar e transportar uma unidade, do concorrente i para o deposito j. Seja x_{ij} a quantidade comprada do concorrente i e transportada para o depósito j.

Teremos o seguinte modêlo matemático:

$$\sum_{j=1}^{n} x_{ij} \leqslant a_i \qquad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1,2,\ldots,n)$$

$$\text{minimizar} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij}$$

Este é um problema do mesmo tipo do problema de transporte. Nota-se que : $\sum_i a_i \neq \sum_j b_j$. Existe mais oferta que o valor do contrato total, isto é : $\sum_i a_i \geqslant \sum_j b_j$.

Se colocarmos um depósito fictício que absorva a quantidade excedente, igual a diferença, $b_{n+1} = \sum_i a_i - \sum_j b_j$. Atribuindo custo zero para $c_{i,n+1}$ para todo i formamos um problema compatível com o problema de transporte.

Este problema foi simplificado visto que podem ocorrer algumas cláusulas no contrato que não permitam a utilização dos resultados obtidos, como por exemplo:

- um concorrente não admite frações da quantia especificada,
- os sub-lotes têm outros custos, etc..

CAPITULO II - INTRODUÇÃO À PROGRAMAÇÃO INTEIRA

2.1- Otimização Inteira.

Qualquer problema decisório que tenha um objetivo a ser maximizado (ou minimizado) em que as variaveis de decisão e serem quantificadas devam assumir valores não fracionários, ou discretos, pode ser classificado como problema de otimização inteira. O conjunto das restrições impostas ao problema poderão ser inequações ou equações, lineares ou não-lineares, bem como sua equação objetivo. Pode-se ainda classificar os problemas de otimização inteira em lineares e não-lineares, conforme a característica de suas equações ou inequeções. Um programa é dito de otimização linear setodo o conjunto de restrições e função objetivo forem estritamente lineares. Em outros casos são chamados de não-lineares.

Em geral as técnicas e desenvolvimentos envolvidos na resolução desses problemas estão concentrados na resolução de problemas lineares, por serem relativamente mais fáceis de serem resolvidos.

Os problemas de Otimização Inteira, da mesma maneira em que foi definido acima não é nenhuma novidade matemática, mas sua aplicação operacional começou a ser reconhecida somente a partir da década de quarenta.

A importância da Otimização Inteira na resolução de problemas práticos pode ser notada através do seu grande desenvolvimento, na área da pesquisa operacional, em especiel para os problemas de otimização linear.

Vários modêlos de casos próticos estão sendo propostos por pesquisadores e técnicos especialistas nesse área e vão se aprimorando à cada dia, as técnicas envolvidas na resolução desses problemas que iniciou com a apresentação, por Gomory, da primeira técnica finita de programação inteira, para resolução desses problemas lineares, em 1.958 .

2.2- Modêlo Matemático do Problema de Otimização Linear

Um probleme de otimização inteira pode ser definido em geral como:

MINIMIZAR (ou MAXIMIZAR)

$$z = g_o (x_1 , x_2 , ... , x_n)$$

SUJEITO À :

$$g_i (x_1 , x_2 , ... , x_n) \left\{ \begin{matrix} \leqslant \\ = \\ \geqslant \end{matrix} \right\} b_i$$

para (i = 1, 2, ... , m) e $x_j \geqslant 0$ para $j \in N$

x_j inteiro para $j \in I \subseteq N$

onde $N \equiv \{ 1, 2, ..., n \}$

Se $I \equiv N$, todas as variaveis x_j deverão ser inteiras, formando um conjunto de problemas chamados de PROGRAMAÇÃO INTEIRA PURA.

Se $I \subset N$, onde apenas algumas varieveis x_j deverão ser inteiras, define-se outro conjunto de problemas chamados de PROGRAMAÇÃO INTEIRA MISTA.

2.3- Programação Linear Inteira.

O problema de programação linear inteira é definido como um modêlo matemático em que se deseja determinar um conjunto de varieveis não-negatives, em geral inteiras, que satisfaça um conjunto de restrições lineares e minimize (maximize) uma função linear chamada **Objetivo.**

Utilizando-se a notação matricial teremos por exemplo :

$$\text{Minimizar} \quad \vec{c} \cdot \vec{x} + \vec{d} \cdot \vec{y} = z$$

$$\text{Sujeito à} \quad A \cdot \vec{x} + D \cdot \vec{y} \leqslant \vec{b}$$

onde : $\vec{x} \geqslant \vec{0}$; $\vec{y} \geqslant \vec{0}$ e $\vec{x}$ inteiro, representando :

$\vec{c}$ - vetor custo de dimensão n

$\vec{d}$ - vetor custo de dimensão n'

A - matriz das restrições de dimensão $m \times n$

D - matriz das restrições de dimensão $m \times n'$

$\vec{b}$ - vetor coluna (termos independentes) de dimensão m

$\vec{x}$ - vetor solução de dimensão n

$\vec{y}$ - vetor solução de dimensão m'

Quando se tem $n' = 0$, o problema será de programação inteira pura. Se $n = 0$ tem-se um problema de programação linear contínua (não-inteira) e em geral ($n \neq 0$ e $n' \neq 0$), programação inteira mista.

Se em um problema de programação inteira, pura ou mista, os valores das varieveis inteiras forem restritos aos valores zero e um, estamos diante de uma classe de problemas normalmente chamados de Programação Inteira Zero-Um.

Outra classe bastante importante de programação inteira são os conhecidos como problemas de "mochila" (Knepseck) onde tem-se apenas uma única função como restrição. O nome desse problema vem do fato de se ter que carregar em uma "mochila", um agrupamento de n ítens pesando cada um a_j e tendo um valor relativo c_j (importância por exemplo). O problema consiste em encher a mochila, atingindo no máximo um pêso b especificado, tal que a soma dos valôres relativos a cada ítem seja maximizada.

O modêlo matemático para esse problema pode ser escrito da seguinte forma:

Determinar x_j ($j = 1, 2, \ldots, n$) tal que:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j . x_j$$

SUJEITO às restrições seguintes:

$$\sum_{j=1}^{n} a_j . x_j \leq b$$

$$x_j \geq 0 \quad \text{todos inteiros}$$

e ($j = 1,2, \ldots, n$)

Onde os coeficientes a_j , c_j e o número b são conhecidos e inteiros. Os velores de a_j ($j = 1, 2, .., n$) e b são também positivos.

2.4- Aplicações da Programação Inteira

Existem praticamente duas classes de aplicações da programação inteira. A primeira constitui dos problemas de transporte e variações, que sob determinadas condições, sua estrutura obriga a uma solução inteira, através de métodos desen -

volvidos para programação contínua, o que dispensa a utilização de métodos menos eficientes próprios para programação inteira.

A segunda classe constitui-se de uma série interminável de problemas em que é necessário a utilização de métodos próprios. Mostremos a seguir alguns exemplos clássicos.

EXEMPLO Nº 1- (Problema da mochila). Necessita se transportar uma série de ítens onde o pêso total é limitado. Cada ítem tem seu valor e o objetivo é transporter o máximo valor possivel, dentro do limite de pêso especificado.

O modêlo matemático desse problema pode ser o seguinte: Seja e_j o pêso do ítem j e c_j o seu valor (normalmente especificado pelo interessado no transporte). Seja b o limite de pêso a ser transportado e x_j uma variavel binária indicando para cada ítem uma das alternativas. Resultado 1 (um) indica a ação de transporter e 0 (zero) não transporter.

$$\text{MAXIMIZAR} \quad z = \sum_{j=1}^{n} c_j . x_j$$

$$\text{SUJEITO À:} \quad \sum_{j=1}^{n} e_j . x_j \leqslant b$$

$$\text{e} \quad x_j = (0 , 1)$$

para ($j = 1, 2, \ldots , n$), onde n é o número de ítens, candidátos ao transporte.

Neste caso simples temos um problema de uma única dimensão. Poderíamos, entretanto, adicionar mais uma restrição para o problema limitando por exemplo o volume máximo a ser transportado. Teriamos com isso mais uma restrição alem da restrição de pêso. Esse tipo de problema é bastante comum na área de transportes aéreos onde o fator pêso-volume é de grande im

portância. Posso otimizar o transporte de valores dentro das limitações impostas pelas aeronaves.

EXEMPLO Nº 2- (Problema do caixeiro viajante). O caixeiro viajante deve visitar n cidades, passando por todas elas apenas uma vez e terminar sua viagem na cidade de origem. A distância entre as cidades i e j poderão ser chamadas de d_{ij} e é conhecido que essa distância pode depender da direção tomada, ou seja, o valor de d_{ij} não é necessariamente igual ao valor de d_{ji} . O problema consiste em determinar uma rota para a viagem do interessado em que a distância total percorrida seja a mínima possível e seu início e final ocorram na mesma cidade de origem.

Supondo uma visita a n cidades. A uma cidade j pode-se chegar de n outras cidades e de uma cidade i posso partir para outras n cidades.

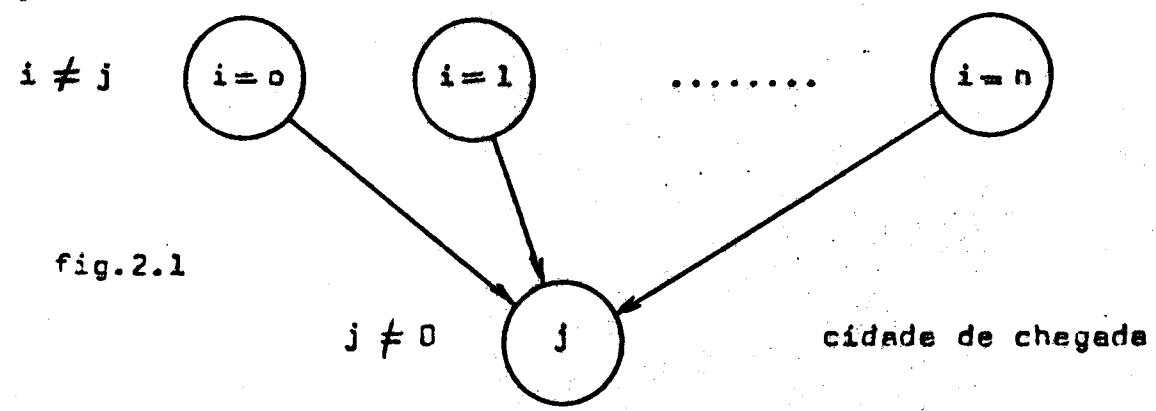


fig.2.1

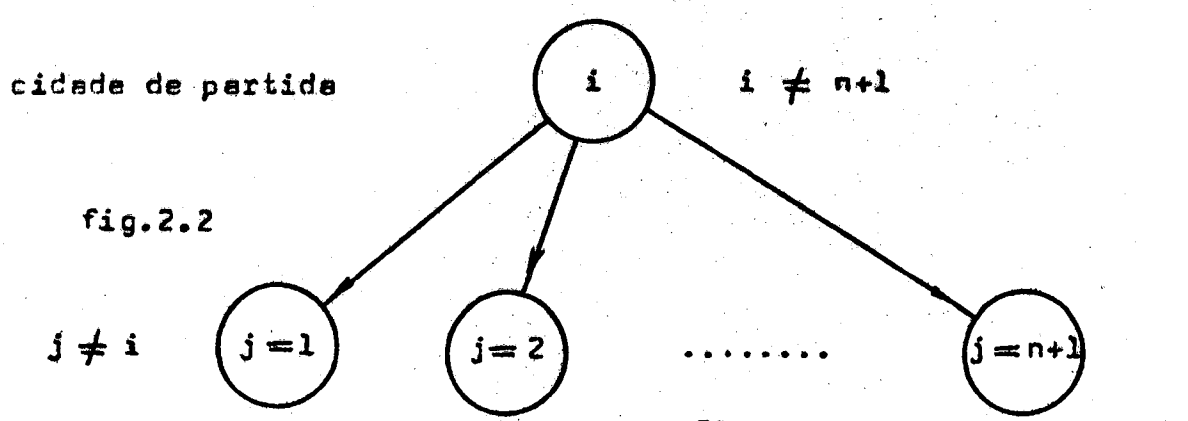


fig.2.2

A cidade 0 (zero) corresponde à cidade origem e a cidade $n+1$ corresponde a cidade de chegada que coincide com a origem.

Pode-se dizer que o viajante parte da cidade zero e chega na cidade $n+1$.

Seja x_{ij} uma variavel binária que corresponde a passagem ou não do viajante por aquele ramo, ou seja, se $x_{ij} = 0$ o caixeiro viajante não se transladará da cidade i para a cidade j e se $x_{ij} = 1$, este irá de i para j.

Para garantir que cada cidade será visitada apenas uma vez tem-se as seguintes restrições:

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad (\ j = 1,\ 2,\ \ldots,\ n+1,\ i \neq j\ )$$

$$\sum_{j=1}^{n+1} x_{ij} = 1 \qquad (\ i = 1,\ 2,\ \ldots,\ n\ ,\ i \neq j\ )$$

Essas restrições, no entanto, não elimina a possibilidade de ciclos ("loops") entre as cidades como por exemplo

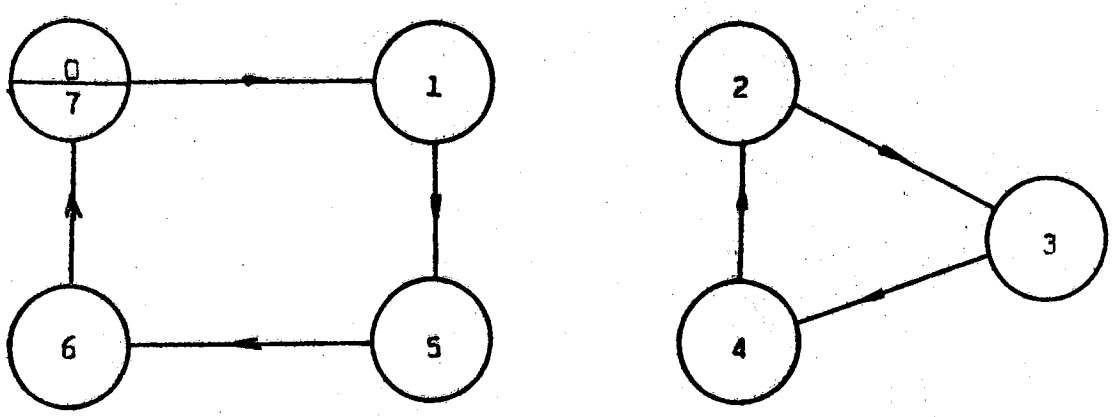


fig. 2.3

$x_{01} = x_{15} = x_{56} = x_{67} = 1$

$x_{23} = x_{34} = x_{42} = 1$

Os demais $x_{ij} = 0$, onde $n = 6$.

Uma maneira de eliminar a possibilidade desses ciclos seria adicionar-se mais uma restrição:

$$\alpha_i - \alpha_j + (n+1)\, x_{ij} \leqslant n \qquad \begin{cases} i = 0, 1, \ldots, n \\ j = 1, 2, \ldots, n+1 \\ i \neq j \end{cases}$$

onde α_i é um número real associado a cada cidade i.

Essa equação deve ser satisfeita quando não há ciclos na trajetória. Podemos definir $\alpha_0 = 0$, $\alpha_{n+1} = n+1$ e $\alpha_i = k$ se a cidade i é a $k^{\text{ésima}}$ cidade visitada. Desde que $1 \leqslant \alpha_i \leqslant n+1$ ($i = 1, 2, \ldots, n+1$) a diferença $\alpha_i - \alpha_j$ será sempre $\leqslant n$ para todo (i, j).

A função objetivo a ser minimizada será a de distância total percorrida, ou seja:

$$\sum_{i=0}^{n} \sum_{j=1}^{n+1} d_{ij} \cdot x_{ij} \qquad (i \neq j)$$

O modêlo matemático correspondente será: Determinar as variáveis x_{ij} e valores arbitrários α_i e α_j reais tal que;

MINIMIZE $$\sum_{i=0}^{n} \sum_{j=1}^{n+1} d_{ij} \cdot x_{ij} \qquad (i \neq j)$$

SUJEITO À :

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad (j=1, 2, \ldots , n+1; \quad i \neq j)$$

$$\sum_{j=1}^{n+1} x_{ij} = 1 \qquad (i=0, 1, \ldots , n ; \quad i \neq j)$$

$$\alpha_i - \alpha_j + (n+1) . x_{ij} \leqslant n \qquad \begin{cases} i = 0, 1, \ldots, n \\ j = 1, 2, \ldots, n+1 \\ i \neq j \end{cases}$$

e

$$x_{ij} = 0 \text{ ou } 1$$

onde $x_{0,n+1} = 0$ e $x_{ij} = 0$ para $i = j$.

A formuleção originel desse problema eparece em Tucker (peç. 12 - Salkin) [1].

2.5- Solução para os problemas de Programeção Inteira.

2.5.1- Exemplo de solução gráfica.

Seje o seguinte problema:

Meximizer: $2x - y = z$

Sujeito à: $5x + 7y \leqslant 45$

$-2x + y \leqslant 1$

$2x - 5y \leqslant 5$

x e $y \geqslant 0$ (inteiros)

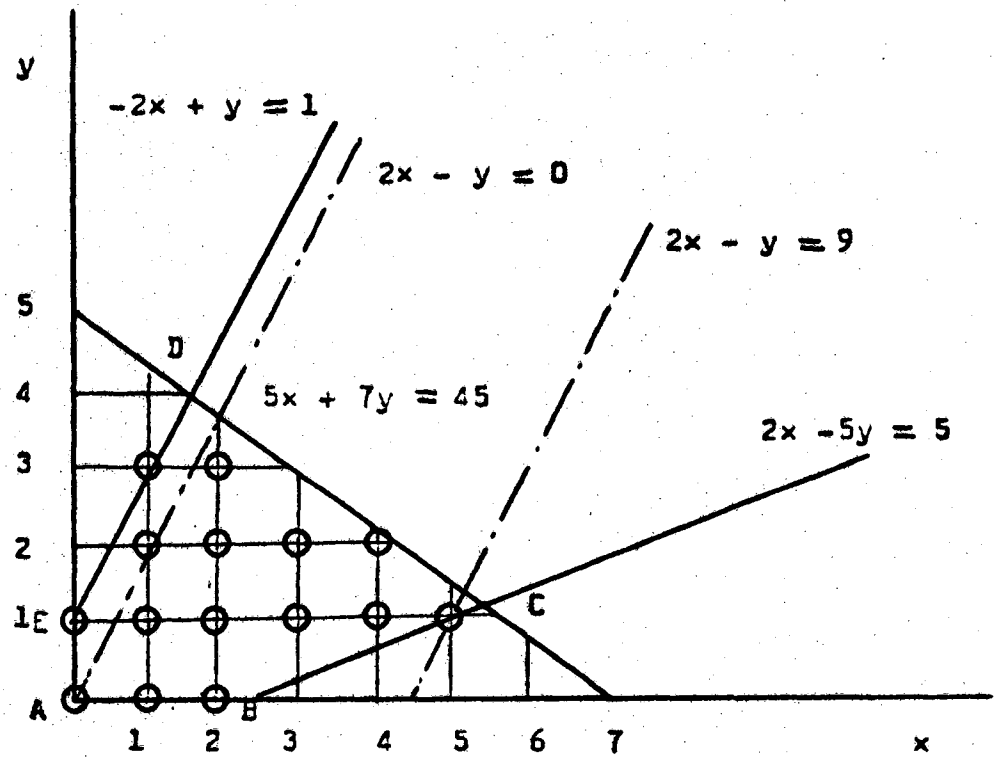


fig. 2.4

Soluções possíveis:

(1 , 2) ; (1 , 3) ; (2 , 3) ; (2 , 2) ; (3 , 2)
(1 , 1) ; (0 , 0) ; (1 , 0) ; (2 , 0) ; (0 , 1)
(2 , 1) ; (3 , 1) ; (4 , 1) ; (5 , 1) ; (4 , 2)

O poliedro convexo ABCDE contém o conjunto de soluções possíveis, que será o sub-conjunto formados pelos pontos discretos, contido no poliedro.

Desse conjunto, verifica-se facilmente que o ponto (5 , 1) maximize o objetivo. O valor da função objetivo é obtido substituindo-se: 2x5 - 1 = 9.

No caso de progremeção inteira, nem sempre o ponto ótimo pertence a um ponto extremo do poliedro convexo obtido etrevés do conjunto de restrições impostes pelo probleme, como era certo em resoluções de problemes contínuos.

2.5.2- Solução por aproximação

Atualmente existe um grande número de pesquisadores empenhados em desenvolver métodos de arredondamento partindo se da solução obtida por métodos continuos. Esse processo consiste em resolver o problema através dos métodos normais para soluções fracionérias e, se a solução obtida for inteira, o processo termina. Se for fracionária devemos prosseguir até obter a solução inteira.

O processo de arredondamento normalmente utilizado (para mais ou para menos), ocasiona êrros que dependem de ordem de grandeza dos números envolvidos, como por exemplo, o êrro cometido ao se arredondar 15,3 para 15 é maior do que o arredondamento de 150,3 para 150. Esse problema deve ser muito bem analizado.

Exemplo: Seja o seguinte problema;

MAXIMIZAR $\quad 21\ x_1 + 11\ x_2$

Sujeito à: $\quad 7\ x_1 + 4\ x_2 \leqslant 13$

x_1 e $x_2 \geqslant 0$ e inteiros

A solução será, $x_1 = 0$ e $x_2 = 3$. Dado a simplicidade pode-se obtê-la por meio de vários métodos.

A solução fracionária seria $x_1 = 13/7$ e $x_2 = 0$ Arredondando esses valores teríamos $x_1 = 2$ e $x_2 = 0$, oque cause uma solução impossivel, pois, ao substituirmos na restrição teríemos : $7 \times 2 + 4 \times 0 = 14 \leqslant 13$ (impossivel).

Se arredondermos para menos teremos $x_1 = 1$ e $x_2 = 0$ que é uma solução possivel mas, longe do ponto ótimo (21).

CAPÍTULO III - MÉTODOS NUMÉRICOS E PROGRAMAS

3.1- Desenvolvimento Teórico

Estamos interessados em determiner os valores de $x_j \geqslant 0$, e inteiros ($j = 1,2,\ldots,n$) que minimize a função objetivo z onde:

$$\sum_{j=1}^{n} c_j . x_j = z \qquad | (3.1)$$

Sujeito às seguintes restrições:

$$\sum_{j=1}^{n} e_{ij} . x_j \geqslant b_i \qquad | (3.2) \qquad (i = 1,2,\ldots,m)$$

e são conhecidas as constantes inteiras; e_{ij} , b_i , e c_j .

O método desenvolvido consiste em transformer as equações | (3.1 e 3.2) na forma implicite e desenvolver um processo iterativo que converge para a solução ótima em um número finito de passos.

$$x_j = d_j + \sum_{k=1}^{n} d_{ik} . y_k \qquad | (3.3) \qquad (j = 1,2,\ldots,n)$$

A transformação inicial é obtida tomando-se as equeções | (3.1 e 3.2) na forma paremétrica. Os valores de d_j e d_{jk} são definidos pelo processo desenvolvido.

$$z = \sum_{j=1}^{n} c_j \cdot y_j$$

$$x_j = y_j \geq 0 \qquad |(3.4) \qquad (j = 1,2,\ldots,n)$$

$$x_{n+1} = -b_i + \sum_{j=1}^{n} a_{ij} \cdot y_j \geq 0 \qquad (i = 1,2,\ldots,m)$$

As varieveis x_{n+1} são varieveis de folga.

Eliminando-se x_j de |(3.1 e 3.2) usando |(3.3) obtem-se o seguinte probleme equivalente:

Determiner $y_j \geq 0$ $(j = 1,2,3,\ldots,n)$ que minimize z, onde:

$$z = \bar{z}_o + \sum_{j=1}^{n} \bar{c}_j \cdot x_j$$

$$x_j = d_j + \sum_{k=1}^{n} d_{jk} \cdot y_k \geq 0 \qquad |(3.5) \quad (j = 1,2,\ldots,n)$$

$$x_{n+i} = -\bar{b}_i + \sum_{j=1}^{n} \bar{a}_{ij} \cdot y_j \geq 0 \qquad (i = 1,2,\ldots,m)$$

As constantes $\bar{z}_o$, $\bar{c}_j$, $\bar{b}_i$ e $\bar{a}_{ij}$ são resultados da transformação com:

$$\bar{z}_o = \sum_{j=1}^{n} c_j \cdot d_j$$

$$\bar{c}_j = \sum_{k=1}^{n} c_k \cdot d_{kj} \qquad (j = 1,2,\ldots,n)$$

$$\bar{b}_j = b_i - \sum_{j=1}^{n} e_{ij} \cdot d_j \qquad (i = 1,2,\ldots,m) \qquad |(3.6)$$

$$\bar{e}_{ij} = \sum_{k=1}^{n} e_{ik} \cdot d_{kj} \qquad (i = 1,2,\ldots,m)$$
$$(j = 1,2,\ldots,n)$$

TEOREMA - Se as constantes $\bar{c}_j \geqslant 0$, $d_j \geqslant 0$ e $\bar{b}_i \leqslant 0$ para todo i e j então a solução MINIMA para o problema |(3.5) é dada por $z = \bar{z}_o$, $y_j = 0$, para $j = 1,2,\ldots,n$.

(prova- 3 , pag. 49)

Determinação da solução ótima:

Considerando $c_j \geqslant 0$ ($j = 1,2,\ldots,n$), podemos escrever |(3.5) como,

$$\vec{x} = \vec{\beta} + \sum_{j=1}^{n} \vec{\alpha}_j \cdot y_j \qquad |(3.7)$$ onde;

$\vec{x}$ é um vetor coluna cujos componentes são z , x_1 , x_2 , ... , x_{n+m} ; $\vec{\beta}$ é tambem um vetor coluna com componentes $\bar{z}_o$, d_1 , d_2 , ... , d_n , $-\bar{b}_1$, $-\bar{b}_2$, ... , $-\bar{b}_m$ e $\vec{\alpha}_j$ são vetores colunas com os componentes $\bar{c}_j$, d_{1j} , d_{2j} , ... , d_{nj} , $\bar{e}_{1j}$, $\bar{e}_{2j}$,

$\ldots$, $\bar{e}_{mj}$.

Inicialmente podemos escrever o problema (3.4) na forma dada por (3.7) onde os componentes de $\vec{\beta}$ serão : 0 , 0 , $\ldots$, 0 , $-b_1$, $-b_2$, $\ldots$, $-b_m$ e os componentes de $\vec{\alpha}_j$ serão: c_j , $d_{ij} = 0$, para $i \neq j$ e $d_{ij} = 1$ para $i = j$ e $\bar{e}_{ij} = e_{ij}$.

Para obtermos um algoritmo finito usaremos os conceitos de ordenação lexicográfica sôbre o que faremos as seguintes observações:

a) Um vetor $\vec{x}$ é dito lexicograficamente maior que zero (ou lexicopositivo) se $\vec{x}$ tem pelo menos um componente diferente de zero e o primeiro deles for maior que zero (positivo) .

   Notação : $\vec{x} \succ \vec{0}$

b) Um vetor $\vec{x}$ é menor que $\vec{y}$, lexicograficamente se o vetor diferença $\vec{y}-\vec{x}$ for lexicopositivo.

   Notação : $\vec{x} \prec \vec{y}$

3.2- Algorítmo Dual

Esse algorítmo se baseia no método simplex dual, de programação linear contínua, e foi desenvolvido po Gomory com propriedades diferentes de convergência apresentedes por Greemberg.

Passo 1.- Monter e tabela com as colunas $\vec{\beta}$, $\vec{\alpha}_1$, $\vec{\alpha}_2$, ... , $\vec{\alpha}_n$. A cada iteração, $\vec{\beta}$ conterá: $\bar{z}_o$, d_1 , d_2 , ... , d_n , $-\bar{b}_1$, $-\bar{b}_2$, ... , $-\bar{b}_m$; e as colunas $\vec{\alpha}_j$: $\bar{c}_j$, d_{1j} , d_{2j} , ... , d_{nj} , $\bar{a}_{1j}$, $\bar{a}_{2j}$, ... , $\bar{a}_{mj}$

Inicialmente: $\bar{z}_o = 0$; $d_j = 0$; $\bar{b}_i = b_i$; $\bar{c}_j = c_j \geq 0$ $d_{ii} = 1$, $d_{ij} = 0$ ($i \neq j$) e $\bar{a}_{ij} = a_{ij}$ para $i = 1,2,...,m$ e $j = 1,2,...,n$.

Passo 2.- Se $d_j \geq 0$ para $j = 1,2,...,n$ e $\bar{b}_i \leq 0$ para $i = 1,2,...,m$ então $z = \bar{z}_o$ é a solução ótima mínima e $x_j = d_j$ para $j = 1,2,...,n$. FIM.

Senão, selecionar o menor componente da coluna $\vec{\beta}$, não considerando a função objetivo. Definir J^+ igual ao conjunto de índices j em que $a_j > 0$. Se todos os $a_j \leq 0$, da linha selecionada, o problema não tem solução. FIM.

Passo 3.- Determinar o índice s tal que $\vec{\alpha}_s = \min \vec{\alpha}_j$, para $j \in J^+$, lexicograficamente. Pesquisar o MAIOR INTEIRO μ_j que faz $\vec{\alpha}_j - \mu_j \vec{\alpha}_s > 0$ para $j \in J^+$ e $j \neq s$. Faça $\mu_s = 1$. Se $\vec{\alpha}_j$ e $\vec{\alpha}_s$ iniciem com um número diferentes de zeros $\mu_j = \infty$.

Tomar: $\lambda = \max (a_j / \mu_j)$, para $j \in J^+$.

Passo 4.- Calcular $q = \{b_o/\lambda\}$. $p_j = \{a_j/\lambda\}$ e os novos valores da tabela serão $\vec{\beta}' = \vec{\beta} + q\,\vec{\alpha}_s$ e $\vec{\alpha}'_j = \vec{\alpha}_j - p_j\,\vec{\alpha}_s$ para $j \neq s$.

Faça $\vec{\beta} = \vec{\beta}'$ e $\vec{\alpha}_j = \vec{\alpha}'_j$. Volte para o passo 2.

Obs.- $\{a\}$ = menor inteiro que contém a .

A seguir é apresentado a programação dêsse algorítmo, em linguagem FORTRAN. O programa está dimensionado para resolver sistemas de até 49 colunas e a soma do número de colunas com o número de linhas não deve ultrapassar 49. O programa utiliza 11 digitos decimais para os números inteiros.

Os CARTÕES DE DADOS (arquivo CARTAO), devem ser apresentados da seguinte forma:

1º Cartão : Identificação

Colunas	Descrição
1 a 5	Número de colunas do sistema (incógnitas)
6 a 10	Número de linhas (inequações)
11 a 20	Nome (sigla) para o problema

2º Cartão : Função Objetivo

Os valores devem ser perfurados da coluna 1 à 80 ocupando cada número um máximo de 5 colunas (ajustados à direita). Cada cartão conterá um máximo de 16 números e se poderá utilizar quantos cartões forem necessários.

3º Cartão : Termos Independentes (limites inferiores)

Idem ao ítem anterior.

4º Cartão : Matriz dos coeficientes

São fornecidos da mesma forma que o ítem anterior sendo que os valores devem ser perfurados por linha, iniciando - se um novo cartão a cada linha.

Aplicação do algorítmo programado, na resolução de um problema exemplo.

Nome do Probleme: EX.DU.1010

Seja o seguinte probleme de programação linear inteira

Minimizar:

$$50\,x_1 + 3\,x_2 + 9\,x_3 + 12\,x_4 + 4\,x_5 + 2\,x_6 + x_7 + 3\,x_9 + 15\,x_{10}$$

Sujeito à:

$$\begin{pmatrix} 1 & 0 & -2 & 1 & 4 & -9 & 4 & 3 & 2 & 4 \\ 3 & 4 & 1 & 9 & 4 & 7 & -15 & 9 & 3 & 5 \\ 4 & -5 & 8 & -4 & 7 & -1 & 3 & 4 & 1 & 7 \\ 0 & 2 & 1 & 4 & 9 & 5 & 13 & 1 & 4 & 9 \\ 1 & 0 & 1 & 2 & 1 & 0 & 4 & -3 & 8 & 3 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \geqslant \begin{pmatrix} 20 \\ 30 \\ 2 \\ 90 \\ -8 \end{pmatrix}$$

Utilizando-se o programa teremos os seguintes cartões de dados:

.....

```
10    5EX.DU.0101
50    3    9    12    4    2    1    0    3    15
20    30    2    90    -8
 1    0    -2    1    4    -9    4    3    2    4
 3    4    1    9    4    7    -15    9    3    5
 4    -5    8    -4    7    -1    3    4    1    7
 0    2    1    4    9    5    13    1    4    9
 1    0    1    2    1    0    4    -3    8    3
```

```
        IDENT DUAL
        FILE  2=CARTAO,UNIT=READER
        FILE  5=IMPRSS,UNIT=PRINTER
        SIZE  ALPHA=5,INTEGER=10
FLAG  141** THIS CONTROL CARD WILL BE IGNORED IN THE NEXT RELEASE
(0001)        INTEGER A(50,50),LB(2),S,P,Q
(0002)        ALPHA NOME(2)
FLAG  142** THIS TYPE STATEMENT WILL NOT BE AVAILABLE NEXT RELEASE
C         ALGORITMO DUAL PARA PROGRAMACAO INTEIRA
C         MINIMO
C         RESTRICOES =(MAIOR=IGUAL)
C.............PASSO UM
C
(0003)        CALL DATE(IM,ID,IA)
(0004)  11111 READ(2,10000,END=99999)N,M,NOME
(0005)        I2=0
(0006)        IT=0
C            N= NUMERO DE COLUNAS
C            M= NUMERO DE LINHAS
(0007)        ML=N+M+1
(0 8)         N1=N+1
(0 9)         J1=N1+1
C            LEITURA DA FUNCAO OBJETIVO
(0010)        READ(2,10000)(A(1,J),J=2,N1)
C            CONSTRUCAO DA IDENTIDADE E DADOS INICIAIS
(0011)        DO 11110 J=1,N1
(0012)        DO 11100 I=2,N1
(0013)        A(I,J)=0
(0014)  11100 CONTINUE
(0015)        A(J,J)=1
(0016)  11110 CONTINUE
(0017)        A(1,1)=0
C            LEITURA DOS TERMOS INDEPENDENTES
(0018)        READ(2,10000)(A(I,1),I=J1,ML)
C            LEITURA DA MATRIZ DOS COEFICIENTES ( UMA LINHA POR CARTAO)
(0019)        DO 11120 I=J1,ML
(0020)        A(I,1)=-A(I,1)
(0021)        READ(2,10000)(A(I,J),J=2,N1)
(0022)  11120 CONTINUE
(0023)        WRITE(5,60000)NOME ,ID,IM,IA
C
C.............PASSO 2.
C
(0024)        CALL TIME(0.,TI)
(0025)  22222 DO 22200 I=2,ML
(0026)        IF(A(I,1))22210,22200, 22200
(0027)  22200 CONTINUE
(0028)        WRITE(5,30000)A(1,1),IT,T2
(0029)        WRITE(5,30000)(I,A(I+1,1),I=1,N)
(0030)        GO TO 11111
C            DETERMINACAO DO MINIMO
(0031)  22210 MIN=A(2,1)
(0032)        L=2
(0033)        DO 22230 I=3,ML
(0034)        IF(MIN-A(I,1))22230,22230,22220
(0035)  22220 MIN=A(I,1)
(0036)        L=I
(0037)  22230 CONTINUE
C            VERIFICA NAO SOLUCAO
(0038)        DO 22240 J=2,N1
(0039)        IF(A(L,J))22240,22240,33333
(0040)  22240 CONTINUE
(0041)  22250 WRITE(5,20000)IT,T2
(0042)        GO TO 11111
C
C.............PASSO 3.
C
(0043)  33333 K=1
(0044)        IX=J
(0045)        DO 33370 I=1,ML
(0046)        MIN=A(I,IX)
(0047)      . S=IX
(0048)        DO 33360 J=IX,N1
(0049)        IF(A(L,J))33360,33300,33300
(0050)  33300 GO TO(33310,33330),K
(0051)  33310 IF(A(I,J))33320,33350,33320
(0052)  33320 K=2
(0053)  33330 IF(MIN-A(I,J))33360,33340,33340
(0054)  33340 MIN=A(I,J)
(0055)  33350 S=J
(0056)  33360 CONTINUE
(0057)        GO TO(33370,33380),K
(0058)  33370 CONTINUE
(0059)        WRITE(IMP,20000)IT,T2
(0060)        GO TO 11111
(0061)  33380 LIN=I
(0062)        XLM=0
(0063)        IF(A(LIN,S))33390,33470,33390
(0064)  33390 DO 33460 J=2,N1
(0065)        IF(J-S)33400,33460,33400
(0066)  33400 IF(A(L,J))33460,33460,33410
(0067)  33410 MI=A(LIN,J)/A(LIN,S)
(0068)        IF(A(LIN,J)-MI*A(LIN,S))33420,33420,33430
(0069)  33420 MI=MI-1
(0070)  33430 IF(MI)33440,33460,33440
(0071)  33440 XLB=FLOAT(A(L,J))/MI
(0072)        IF(XLM-XLB)33450,33460,33460
(0073)  33450 XLM=XLB
(0074)        LB(1)=A(L,J)
(0075)        LB(2)=MI
(0076)  33460 CONTINUE
(0077)        IF(XLM-A(L,S))33470,33480,33480
(0078)  33470 XLM=A(L,S)
(0079)        LB(1)=A(L,S)
(0080)        LB(2)=1
(0081)  33480 IF(XLM)22250,22250,44444
C
C.............PASSO 4.
C
(0082)  44444 LX=-A(L,1)
(0083)  44400 Q=LX*LB(2)/LB(1)
(0084)        IF(LX-Q*LB(1)/LB(2))44420,44420,44410
```

```
(0085)   44410 Q=Q+1
(0086)   44420 DO 44470 J=2,NL
(0087)         IF(J-S)44430,44470,44430
(0088)   44430 P=A(L,J)*LB(2)/LB(1)
(0089)         IF(A(L,J)-P*LB(1)/LB(2))44450,44450,44440
(0090)   44440 P=P-1
(0091)   44450 DO 44460 I=1,NL
(0092)         A(I,J)=A(I,J)-P*A(I,S)
(0093)   44460 CONTINUE
(0094)   44470 CONTINUE
(0095)         DO 44480 I=1,NL
(0096)         A(I,1)=A(I,1)+Q*A(I,S)
(0097)   44480 CONTINUE
(0098)         IT=IT+1
(0099)         CALL TIME(T1,T2)
(0100)         T2=T2/1000.
(0101)         IF(T2-600.)22222,22222,44490
(0102)   44490 WRITE(5,40000) A(1,1),IT,T2
(0103)         WRITE(5,50000)(I,A(I+1,1),I=1,N)
(0104)         GO TO 11111
(0105)   10000 FORMAT(16G5)
(0106)   20000 FORMAT(//10X8SOLUCAO IMPOSSIVEL8//10X8IT=8I10 //
               A10X8TEMPO DE EXECUCAO=8F10.3)
(0107)   30000 FORMAT(//10X8SOLUCAO OTIMA8//10X8F.O.(MINIMO)=8I10//
               A10X8N-ITERACOES=8I10//10X8TEMPO DE EXECUCAO=8F10.3//)
(0108)   40000 FORMAT(///10X8EXECUCAO INTERROMPIDA8//10X8F.O.=8I10//
               A10X8N-ITERACOES=8I10//10X8TEMPO DE EXECUCAO=8F10.3//)
(0109)   50000 FORMAT(  10X8X(8I38)=8,I10 )
(0110)   60000 FORMAT(1H1/10X'PA5'10X'I28/8I28/8I2)
(0111)   99999 STOP
(0112)         END

 5/12/77  10:08 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER: ASR 5.6        2 FLAGS     0 ERRORS
ELAPSED TIME     44 SECS      141 CARDS AT  192 C.P.M.
COMMON =      0  DATA =  31280  TEMPORARIES =    104
CODE =   7272     DIGITS
```

Solução do Exemplo Apresentado:

```
EX.DU.0101                    27/10/76

SOLUCAO OTIMA

F.O.(MINIMO)=                  12

N-TERACOES=                    5

TEMPO DE EXECUCAO=          1.403 S


X(  1)=                        0
X(  2)=                        0
X(  3)=                        0
X(  4)=                        0
X(  5)=                        0
X(  6)=                        0
X(  7)=                        6
X(  8)=                       14
X(  9)=                        2
X( 10)=                        0
```

3.3- Método PRIMAL para Programação Inteira

Esse método é baseado no algorítmo Simplex para programação linear contínua e foi apresentado por Gomory. Esse método apresenta as seguintes vantagens em relação ao método anteriormente apresentado:

a) Uma solução possivel, obtida por qualquer processo poderá ser utilizada como solução inicial para o algoritmo.

b) Uma solução possivel poderá ser avaliada a qualquer iteração.

Define-se inicialmente o seguinte problema, na sua forma canônica: Determinar x_j ($j = 1,2,\ldots,n,n+1,\ldots,n+m$), inteiro que minimize z onde:

$$-z + \sum_{j=1}^{n} c_j . x_j = 0$$

$$x_{n+i} + \sum_{j=1}^{n} a_{ij} . x_j = b_i \qquad (i = 1,2,\ldots,m) \tag{3.8}$$

$$x_j \geq 0 \qquad (j = 1,2,\ldots,n,n+1,\ldots,n+m)$$

onde a_{ij} , b_i , e c_j são constantes inteiras conhecidas

Para que a solução exista é necessário que $b_i \geq 0$ pois, a solução possivel inicial é $x_{n+i} = b_i$.

Para determinar a solução ótima devemos tomar a seguinte transformação:

$$x_j = d_j - \sum_{k=1}^{n} d_{jk} . y_k \qquad (j = 1,2,\ldots,n) \tag{3.9}$$

Determina-se as constantes d_j e d_{jk} em um processo iterativo durante a determinação da solução do problema. A transformação inicial é estabelecida escrevendo-se a equação (3.8) da seguinte forma:

$$-z + \sum_{j=1}^{n} c_j \cdot y_j = 0$$

$$x_j - y_j = 0 \qquad (j = 1,2,\ldots,n)$$

$$x_{n+i} + \sum_{j=1}^{n} e_{ij} \cdot y_j = b_i \qquad (i = 1,2,\ldots,m)$$

Eliminando-se x_j de (3.8) usando (3.9), obtem se o problema equivalente: Determinar $y_j \geq 0$, inteiro para $j = 1,2,\ldots,n$ que minimize z, onde:

$$-z + \sum_{j=1}^{n} \bar{c}_j \cdot y_j = -\bar{z}_o$$

(3.10)

$$x_j + \sum_{k=1}^{n} d_{jk} \cdot y_k = d_j \qquad (j = 1,2,\ldots,m)$$

$$x_{n+i} + \sum_{j=1}^{n} \bar{e}_{ij} \cdot y_j = \bar{b}_i \qquad (i = 1,2,\ldots,m)$$

As constantes $\bar{z}_o$, $\bar{c}_j$, $\bar{e}_{ij}$ e $\bar{b}_i$ são desenvolvidas pela transformação. Se em um dado momento as constantes acima forem; $\bar{c}_j \geq 0$, $\bar{b}_i \geq 0$, e $\bar{d}_j \geq 0$ para todo i e j, então $z = \bar{z}_o$ e $x_j = d_j$ para $j = 1,2,\ldots,n$ e $x_{n+1} = b_i$ para $i = 1,2,\ldots,m$ serão a solução ótima para o problema.

A equação (3.10) pode ser escrita na forma vetorial como a seguinte:

$$\vec{x} + \sum_{j=1}^{n} \vec{\alpha}_j \cdot y_j = \vec{\beta} \qquad (3.11)$$

onde os componentes dos vetores $\vec{x}$, $\vec{\alpha}_j$ e $\vec{\beta}$ são:

$$\vec{x} \rightarrow -z \, , \, x_1 \, , \, x_2 \, , \, \ldots \, , \, x_{n+m}$$

$$\vec{\alpha}_j \rightarrow c_j \, , \, d_{1j} \, , \, d_{2j} \, , \, \ldots \, , \, d_{nj} \, , \, e_{1j} \, , \, e_{2j} \, , \, \ldots \, , \, e_{mj}$$

$$\vec{\beta} \rightarrow -z_o \, , \, d_1 \, , \, d_2 \, , \, \ldots \, , \, d_n \, , \, b_1 \, , \, b_2 \, , \, \ldots \, , \, b_m$$

Para iniciar o processo utiliza-se os seguintes dados iniciais: $\bar{c}_j = c_j$, $d_{ij} = 0$, para $i \neq j$, $d_{jj} = -1$ $\bar{z}_o = 0$, $d_j = 0$ e $\bar{b}_i = b_i$.

3.4- Algorítmo Primel

Passo 1.- Montar a tabela inicial contendo as colunas $\vec{\alpha}_1 \, , \vec{\alpha}_2 \, \ldots \, , \vec{\alpha}_n \, , \, \vec{\beta}$. Após cada iteração $\vec{\alpha}_j$ conterá os componentes $\bar{c}_j \, , \, d_{1j} \, , \, d_{2j} \, , \, \ldots \, , \, d_{nj} \, , \, \bar{e}_{1j} \, , \, \ldots \, , \, \bar{e}_{mj}$ e a coluna $\vec{\beta}$, $-\bar{z}_o \, , \, d_1 \, , \, d_2 \, , \, \ldots \, , \, d_n \, , \, \bar{b}_1 \, , \, \bar{b}_2 \, , \, \ldots \, , \, \bar{b}_m$.

Inicialmente : $\bar{c}_j = c_j$, $d_{jj} = -1$, $d_{ij} = 0$ ($j \neq i$) $\bar{e}_{ij} = e_{ij}$, $\bar{z}_o = 0$, $d_j = 0$, $\bar{b}_i = b_i$.

Passo 2.- Se $\bar{c}_j \geq 0$ ($j = 1,2,\ldots,n$) a solução ótima será: $z = \bar{z}_o$; $x_j = d_j$ ($j = 1,2,\ldots,n$) e $x_{n+i} = \bar{b}_i$, ($i = 1,2,\ldots,m$) . FIM

Senão , selecionar o índice s em que $\bar{c}_s = \min \bar{c}_j$ para $\bar{c}_j < 0$.

Passo 3.- Se todo $d_{js} \leq 0$ e $\bar{a}_{is} \leq 0$, a solução será ILIMITADA. FIM.

Senão : Determiner $\theta = \min (\bar{b}_i / \bar{a}_{is} ; d_j / d_{js})$ para $\bar{a}_{is} > 0$ e $d_{js} > 0$. Selecionar uma linha em que $\left[d_j / d_{js} \right] \leq \theta$ ou $\left[\bar{b}_i / \bar{a}_{ij} \right] \leq \theta$, onde .... $d_{js} > 0$ e $\bar{a}_{is} > 0$. Seja $a_1, a_2, \ldots, a_n, b_o$ a linha selecionada.

Passo 4.- Calcular $p_j = \left[a_j / a_s \right]$ para $j = 1,2,\ldots,n$ e $q = \left[b_o / a_s \right]$. Determinar os novos valores da tabela:

$$\vec{\alpha}'_j = \vec{\alpha}_j - p_j \vec{\alpha}_s \qquad (j \neq s)$$

$$\vec{\beta}' = \vec{\beta} - q \vec{\alpha}_s$$

$$\vec{\alpha}'_s = -\vec{\alpha}_s$$

Faça $\vec{\alpha}_j = \vec{\alpha}'_j$ ($j = 1,2,\ldots,n$) e $\vec{\beta} = \vec{\beta}'$.

Vá para 2.

Obs.
$\left[a \right]$ = maior inteiro contido

A seguir é apresentado a programação dêsse algorítmo, em linguagem FORTRAN. O programa está dimensionado para resolver sistemas de até 49 colunas e a soma do número de linhas com o número de colunas não deve ultrapassar 49. O programa utiliza 11 digitos decimais para os números inteiros.

Os CARTÕES DE DADOS (arquivo CARTAO), devem ser apresentados da seguinte forma:

1º Cartão : Identificação

Colunas	Descrição
1 a 5	Número de colunas (incógnitas)
6 a 10	Número de linhas (retrições)
11 a 20	Nome do problema

2º Cartão : Função Objetivo

Os valores são lidos a cada 5 colunas, no máximo 16 números por cartão. No caso de se ter um número maior de dados utiliza-se quantos cartões forem necessários.

3º Cartão : Termos independentes (limites superiores)

O mesmo formato que o ítem anterior.

4º Cartão : Matriz dos Coeficientes

São fornecidos da mesma forma que os valores do ítem anterior, sendo que os valores da matriz devem ser perfurados por linha, iniciando-se um novo cartão a cada linha.

Apliceção do algorítmo programado, ne resolução de um problema exemplo.

Nome do problema: EX.PRIMAL.

Problema: Determinar $x_1 \geqslant 0$, $x_2 \geqslant 0$, $x_3 \geqslant 0$, inteiros que minimize z.

$$z = -2\,x_1 - 3\,x_2 + x_3$$

Sujeito à:

$$4\,x_1 - x_2 - 3\,x_3 \leqslant 5$$
$$-2\,x_1 + 2\,x_2 + 3\,x_3 \leqslant 7$$

Os cartões de dados deverão ser perfurados da seguinte forma:

```
.....................................................................
    3    2EX.PRIMAL.
   -2   -3    1
    5    7
    4   -1   -3
   -2    2    3
```

Solução obtida pelo programa:

```
 EX.PRIMAL.          29/10/76

 SOLUCAO OTIMA

 F.O.(MINIMO)=        -20

 N-ITERACOES=          3

 TEMPO DE EXECUCAO=    0.134

 X(  1)=          3
 X(  2)=          5
 X(  3)=          1
```

```
        IDENT PRIMAL
        FILE  2=CARTAO,UNIT=READER
        FILE  5=IMPRSS,UNIT=PRINTER
        SIZE  ALPHA=5,INTEGER=10
FLAG  101** THIS CONTROL CARD WILL BE IGNORED IN THE NEXT RELEASE
(0001)        INTEGER A(50,50),S,P
(0002)        ALPHA NOME(2)
FLAG  102** THIS TYPE STATEMENT WILL NOT BE AVAILABLE NEXT RELEASE
        C          ALGORITMO PRIMAL PARA PROGRAMACAO INTEIRA
        C          MINIMO
        C          RESTRICOES   ( MENOR=IGUAL )
        C
        C.............PASSO UM
        C
(0003)        CALL DATE(IM,ID,IA)
(0004)    11111 READ(2,10000,END=99999)N,M,NOME
(0005)        IT=0
(0006)        T2=0.
        C          N=NUMERO DE COLUNAS
        C          M=NUMERO DE LINHAS
(0007)        ML=M+M+1
(0008)        N1=N+1
(0009)        J1=N1+1
        C          LEITURA DA FUNCAO OBJETIVO
(0010)        READ(2,10000)(A(1,J),J=2,N1)
        C          CONSTRUCAO DA IDENTIDADE E DADOS INICIAIS
(0011)        DO 11110 J=1,N1
(0012)        DO 11100 I=2,ML
(0013)        A(I,J)=0
(0014)    11100 CONTINUE
(0015)        A(J,J)=-1
(0016)    11110 CONTINUE
(0017)        A(1,1)=0
        C          LEITURA DOS TERMOS INDEPENDENTES
(0018)        READ(2,10000)(A(I,1),I=J1,ML)
        C          LEITURA DA MATRIZ DOS COEFICIENTES ( UMA LINHA POR CARTAO )
(0019)        DO 11120 I=J1,ML
(0020)        READ(2,10000)(A(I,J),J=2,N1)
(0021)    11120 CONTINUE
(0022)        WRITE(5,20000)NOME,ID,IM,IA
(0023)        CALL TIME(0.,T1)
        C
        C.............PASSO 2
        C
(0024)    22222 S=2
(0025)        MIN=A(1,S)
(0026)        DO 22210 J=3,N1
(0027)        IF(A(1,J)-MIN)22200,22210,22210
(0028)    22200 MIN=A(1,J)
(0029)        S=J
(0030)    22210 CONTINUE
(0031)        IF(MIN)33333,22220,22220
        C          SOLUCAO OTIMA
(0032)    22220 A(1,1)=-A(1,1)
(0033)        WRITE(5,40000)A(1,1),IT,T2
(0034)        WRITE(5,50000)(I,A(I+1,1),I=1,N)
(0035)        GO TO 11111
        C.............PASSO 3.
(0036)    33333 L=0
(0037)        TETA=1.E90
(0038)        DO 33320 I=2,ML
(0039)        IF(A(I,S))33320,33320,33300
(0040)    33300 TE=FLOAT(A(I,1))/A(I,S)
(0041)        IF(TE-TETA)33310,33320,33320
(0042)    33310 TETA=TE
(0043)        L=I
(0044)    33320 CONTINUE
(0045)        IF(L)33330,33330,44444
        C          SOLUCAO ILIMITADA
(0046)    33330 WRITE(5,30000)IT,T2
(0047)        GO TO 11111
        C.............PASSO 4.
(0048)    44444 DO 44420 J=1,N1
(0049)        IF(J-S)44400,44420,44400
(0050)    44400 P=A(L,J)/A(L,S)
(0051)        IF(A(L,J)-P*A(L,S))44401,44402,44402
(0052)    44401 P=P-1
(0053)    44402 CONTINUE
(0054)        DO 44410 I=1,ML
(0055)        A(I,J)=A(I,J)-P*A(I,S)
(0056)    44410 CONTINUE
(0057)    44420 CONTINUE
(0058)        DO 44430 I=1,ML
(0059)        A(I,S)=-A(I,S)
(0060)    44430 CONTINUE
(0061)        IT=IT+1
(0062)        CALL TIME(T1,T2)
(0063)        T2=T2/1000.
(0064)        IF(T2-600.)22222,22222,44440
(0065)    44440 A(1,1)=-A(1,1)
(0066)        WRITE(5,60000)A(1,1),IT,T2
(0067)        WRITE(5,50000)(I,A(I+1,1),I=1,N)
(0068)        GO TO 11111
(0069)    10000 FORMAT(10F5)
(0070)    20000 FORMAT(1H1,10X,2A5,10X,I2,'/',I2,'/',I2)
(0071)    30000 FORMAT(///10X,'SOLUCAO ILIMITADA, ITERACOES=',I5//
           A10X,'TEMPO DE EXECUCAO=',F10.3//)
(0072)    40000 FORMAT(//10X,'SOLUCAO OTIMA',//10X,F.0,'=(MINIMO)=',I10//
           A10X,'N-ITERACOES=',I10//10X,'TEMPO DE EXECUCAO=',F10.3//)
(0073)    50000 FORMAT(  10X,'X(',I3,')=',I10)
(0074)    60000 FORMAT('///10X,'EXECUCAO INTERROMPIDA',//10X,F.0,'=',I10//
           A10X,'N-ITERACOES=',I10//10X,'TEMPO DE EXECUCAO=',F10.3//)
(0075)    99999 STOP
(0076)        END


 5/12/77  10:13 A.M.      XFORTN COMPILER (75/140)
RELEASE NUMBER: ASR 5.6        2 FLAGS     0 ERRORS
ELAPSED TIME     41 SECS      102 CARDS AT  149 C.P.M.
COMMON =       0  DATA =   31184  TEMPORARIES =    116
CODE =   458A      OCTAL
```

3.4- Problemas com variaveis limitadas

Esses problemas aparecem de forma idêntices aos apresentados anteriormente, apenas introduz-se uma restrição às variaveis, como por exemplo os problemas que necessitam de soluções iguais a zero e um.

O modêlo matemático será:

Minimizar z onde;

$$\sum_{j=1}^{n} c_j . x_j = z$$

$$\sum_{j=1}^{n} a_{ij} . x_j \geqslant b_i \qquad (i = 1,2,\ldots,m)$$

$$0 \leqslant x_j \leqslant u_j \quad (\text{inteiros}) \quad (j = 1,2,\ldots,n)$$

São conhecidos, as constantes inteires:

a_{ij} , b_i , c_j e u_j .

O algoritmo apresentado se baseia no algoritmo dual, adaptado para manusear os limites superiores de solução.

ALGORÍTMO:

Passo 1.- Montar a tabela com as colunas $\vec{\beta}$, $\vec{\alpha}_1$, ... , $\vec{\alpha}_n$, $\vec{u}$. A cada iteração, β conterá $\bar{z}_o$, d_1 , ... , d_n $-\bar{b}_1$, $-\bar{b}_2$, ... , $-\bar{b}_m$; e as colunas α_j : $\bar{c}_j$, d_{1j}, d_{2j}, ... , d_{nj} , $\bar{a}_{1j}$, $\bar{a}_{2j}$, ... , $\bar{a}_{mj}$. A coluna $\vec{u}$, conterá os valores de u_1 , u_2, ... , u_n .

Inicialmente: $\bar{z}_o = 0$; $d_j = 0$; $\bar{b}_i = b_i$; $\bar{c}_j = c_j$, para $c_j \geq 0$; $d_{ii} = 1$, $d_{ij} = 0$ ($i \neq j$) e $\bar{a}_{ij} = a_{ij}$ para todo i e j .

Passo 2.- a) Se $0 \leq d_j \leq u_j$ para $j = 1,2,\ldots,n$ e $\bar{b}_i \leq 0$ para $i = 1,2,\ldots,m$. FIM (a solução mínima é $z = \bar{z}_o$. $x_j = d_j$ para $j = 1,2,\ldots,n$).

Senão, vá para 2 (b).

b) Se $d_j \leq u_j$ para $j = 1,2,\ldots,n$, vá para 2(c).

Senão, selecionar a linha em que o componente d_j , de $\vec{\beta}$ seja o máximo de $d_j - u_j > 0$. Tome $b_o = d_j - u_j$ e $a_k = - d_{jk}$ para $k = 1,2,\ldots,n$ como linha selecionada. Vá para 2 (d).

c) Selecionar a linha em que o correspondente de β seja mínimo. Seja $-b_o$, a_1, a_2, ..., a_n, a linha selecioneda. Vá para 2 (d) .

d) Definir J^+ como o conjunto dos índices j onde , $a_j > 0$. Se todo $a_j \leq 0$, o problema não tem solução. FIM

Senão, vá para o passo 3.

Passo 3.- Determinar o índice s tal que $\alpha_s = \min_{j \in J^+} \alpha_j$, Lexicograficamente.

Pesquiser o meior inteiro μ_j que faz $\alpha_j - \mu_j . \alpha_s > 0$, para $j \in J^+$, $j \neq s$. Faça $\mu_s = 1$.

Se α_j e α_s inicia com números diferentes de zeros, faça $\mu_j = \infty$.

Tomar $\lambda = \max_{j \in J^+} \dfrac{a_j}{\mu_j}$

Passo 4.- Calcular $q = \left\{ \frac{b_o}{\lambda} \right\}$, $p_j = \left\{ \frac{a_j}{\lambda} \right\}$ e os novos valores

para a tabela que serão:

$$\vec{\beta}' = \vec{\beta} + q.\vec{\alpha}_s \quad \text{e} \quad \vec{\alpha}'_j = \vec{\alpha}_j - p_j.\vec{\alpha}_s \quad \text{para}$$

($j \neq s$).

Faça $\vec{\beta} = \vec{\beta}'$ e $\vec{\alpha}_j = \vec{\alpha}'_j$. Vá para o passo 2.

A seguir é apresentado a programação dêsse algo - rítmo, em linguagem FORTRAN. O programa utiliza uma precisão de 11 digitos decimais para os números inteiros.

Os CARTÕES DE DADOS devem ser apresentados da seguinte forma:

1º Cartão : Identificação

Colunas	Descrição
1 a 5	Número de colunas (incógnitas)
6 a 10	número de linhas (inequações)
11 a 20	Nome para o problema

2º Cartão : Função Objetivo

Os valores são lidos a cada 5 colunas, no máxi- mo 16 números por cartão. No caso de se ter um número maior de dados utiliza-se quantos cartões forem necessarios.

3º Cartão : Termos Independentes

Idem ao ítem anterior.

4º Cartão : Limites das variaveis

Idem ao ítem anterior.

5º Cartão : Matriz dos coeficientes

São fornecidos da mesma forma que o ítem anterior sendo que os valores devem ser perfurados por linha, iniciando - se um novo cartão a cada inicio de linha.

Aplicação do algorítmo programado, na resolução de um problema exemplo.

Nome do problema: EX.DUAL2.01

Determinar os valores não negativos de $x_1 \leqslant 1$, $x_2 \leqslant 1$ $x_3 \leqslant 1$, inteiros que minimize z .

$$z = 3x_1 + 2x_2 + 5x_3 \quad ,$$

Sujeito à:

$$3x_1 + 5x_2 + 4x_3 \geqslant 3$$
$$3x_1 + 2x_2 + 2x_3 \geqslant 3$$
$$2x_1 + 2x_2 + 7x_3 \geqslant 4$$

A solução obtida pelo probleme será a seguinte:

```
EX.DUAL2.01      29/10/76

        FUNCAO OBJETIVO

    3    2    5

        TERMOS INDEPENDENTES

    3    3    4

        LIMITES SUPERIORES PARA A SOLUCAO

    1    1    1

        MATRIZ DOS COEFICIENTES POR LINHA  3 X  3


        SOLUCAO OTIMA
        OBJETIVO=          5 (MINIMO)
        ITERACOES=         3
        TEMPO DE EXECUCAO=     0.582 SEGUNDOS
        SOLUCAO

        X(  1)=  1
        X(  2)=  1
        X(  3)=  0
```

```
      IDENT DUAL2
      FILE  2=DADOS2,UNIT=READER
      FILE  5=IMPRU2,UNIT=PRINTER
      SIZE  INTEGER=11,ALPHA=12
FLAG  141** THIS CONTROL CARD WILL BE IGNORED IN THE NEXT RELEASE
(0001)      INTEGER X(50),AUX(50),P(50),EME(50),A(2500),LR(2),S,Q
(0002)      ALPHA COND(3)/"OTIMA      ","IMPOSSIVEL  ","INTERROMPIDA"/,NOME
FLAG  142** THIS TYPE STATEMENT WILL NOT BE AVAILABLE NEXT RELEASE
(0003)      IND(I,J,M)=(J-1)*M+I
      C          A L G O R I T M O   D U A L   ( COM LIMITE )
(0004)      TMP=300.
      C**********PASSO 1.
      C           N=NUMERO DE COLUNAS
      C           M=NUMERO DE LINHAS
      C           NOME=NOME DO EXEMPLO
(0005)      CALL DATE(IM,ID,IA)
(0006)  11111 READ(2,90000,END=99999)M,N,NOME
(0007)  90000 FORMAT(2I4,T11,A12)
(0008)      WRITE(5,70000)NOME,ID,IM,IA
(0009)  70000 FORMAT(1H1,A12,5X,2(I2.0,1),I2)
(0010)      ML=N+M+1
(0011)      NML=ML*N
(0012)      IT=0
      C          LEITURA DA FUNCAO OBJETIVO
(0013)      READ(2,10000)(AUX(I),I=1,N)
(0014)  10000 FORMAT(20I4)
(0015)      WRITE(5,50000)
(0016)  50000 FORMAT(//,10X,'FUNCAO OBJETIVO',//)
(0017)      WRITE(5,30000)(AUX(I),I=1,N)
(0018)      N1=N+1,
(0019)      DO 110 J=1,N
(0020)      I1=IND(1,J,ML)
(0021)      A(I1)=AUX(J)
(0022)      DO 100 I=1,N
(0023)      I1=I+1
(0024)      IJ=IND(I1,J,ML)
(0025)      A(IJ)=0
(0026)  100   CONTINUE
(0027)      X(J)=0
(0028)      I1=IND(J+1,J,ML)
(0029)      A(I1)=1
(0030)  110   CONTINUE
      C          LEITURA DOS LIMITES
(0031)      READ(2,10000)(AUX(I),I=1,M)
(0032)      WRITE(5,40000)
(0033)  40000 FORMAT(//,10X,'TERMOS INDEPENDENTES',//)
(0034)      WRITE(5,30000)(AUX(I),I=1,M)
(0035)      DO 120 I=1,M
(0036)      II=N1+I
(0037)      X(II)=-AUX(I)
(0038)  120   CONTINUE
      C          LEITURA DOS LIMITES SUPERIORES DAS SOLUCOES
(0039)      READ(2,10000)(EME(I),I=1,N)
(0040)      WRITE(5,60000)
(0041)  60000 FORMAT(//,10X,'LIMITES SUPERIORES PARA A SOLUCAO',//)
(0042)      WRITE(5,30000)(EME(I),I=1,N)
(0043)  30000 FORMAT(1X,24I5)
      C          LEITURA DA MATRIZ DOS COEFICIENTES POR COLUNA
(0044)      N2=N1+1
(0045)      M3=M-1
(0046)      DO 130 J=1,N
(0047)      M1=IND(N2,J,ML)
(0048)      M2=M3+M1
(0049)      READ(2,10000)(A(I),I=M1,M2)
(0050)  130   CONTINUE
(0051)      WRITE(5,80000)M,N
(0052)  80000 FORMAT(//,10X,'MATRIZ DOS COEFICIENTES POR LINHAS',I3,' X ',I3,//)
      C.........PASSO 2.
(0053)      CALL TIME(0.,TEMP)
FLAG  042** REAL NUMBER WAS TRUNCATED
      C-----PASSO 2(A).
(0054)  22222 DO 1200 I=2,ML
(0055)      IF(X(I))2200,1200,1200
(0056)  1200  CONTINUE
(0057)      DO 1210 I=1,N
(0058)      J=I+1
(0059)      IF(X(J)-EME(I))1210,1210,2200
(0060)  1210  CONTINUE
(0061)      IR=1
(0062)      GO TO 77777
      C-----PASSO 2(B).
(0063)  2200  DO 2210 I=1,N
(0064)      J=I+1
(0065)      IF(X(J)+GT+EME(I)) GO TO 2220
(0067)  2210  CONTINUE
(0068)      GO TO 3200
(0069)  2220  DO 2230 I=1,N
(0070)      J=I+1
(0071)      AUX(I)=X(J)-EME(I)
(0072)  2230  CONTINUE
(0073)      MAX=AUX(1)
(0074)      L=2
(0075)      DO 2240 I=2,N
(0076)      IF(MAX+GT+AUX(I)) GO TO 2240
(0078)      MAX=AUX(I)
(0079)      L=I+1
(0080)  2240  CONTINUE
(0081)      MAX=-MAX
(0082)      DO 2250 J=1,N
(0083)      LJ=IND(L,J,ML)
(0084)      AUX(J)=-A(LJ)
(0085)  2250  CONTINUE
(0086)      GO TO 4200
      C-----PASSO 2(C).
(0087)  3200  MIN=X(2)
(0088)      L=2
(0089)      DO 3210 I=3,ML
(0090)      IF(MIN+LE+X(I)) GO TO 3210
(0092)      MIN=X(I)
(0093)      L=I
(0094)  3210  CONTINUE
(0095)      MAX=MIN
```

```
(0096)          DO 3220 J=1,N
(0097)          LJ=IND(L,J,ML)
(0098)          AUX(J)=A(LJ)
(0099)    3220  CONTINUE
          C------PASSO 2(D).
(0100)    4200  DO 4210 J=1,N
(0101)          IF(AUX(J).GT.0) GO TO 33333
(0103)    4210  CONTINUE
(0104)          IR=2
(0105)          GO TO 77777
          C.........PASSO 3.
(0106)    33333 K=1
(0107)          IX=J
(0108)          DO 340 I=1,ML
(0109)          II=IND(I,IX,ML)
(0110)          MIN=A(II)
(0111)          S=IX
(0112)          DO 330 J=IX,N
(0113)          IF(AUX(J).LT.0) GO TO 330
(0115)          IJ=IND(I,J,ML)
(0116)          GO TO (300,310),K
(0117)    300   IF(A(IJ).EQ.0) GO TO 320
(0119)          K=2
(0120)    310   IF(MIN.LT.A(IJ)) GO TO 330
(0122)          MIN=A(IJ)
(0123)    320   S=J
(0124)    330   CONTINUE
(0125)          GO TO (340,350),K
(0126)    340   CONTINUE
(0127)          IR=2
(0128)          GO TO 77777
(0129)    350   LIN=I
(0130)          JS=IND(LIN,S,ML)
(0131)          XLM=0.
(0132)          IF(A(JS).EQ.0) GO TO 380
(0134)          DO 370 J=1,N
(0135)          IF(J.EQ.S) GO TO 370
(0137)          IF(AUX(J).LE.0) GO TO 370
(0139)          IJ=IND(LIN,J,ML)
(0140)          MI=A(IJ)/A(JS)
(0141)          IF(A(IJ).GT.MI*A(JS)) GO TO 360
(0143)          MI=MI-1
(0144)    360   IF(MI.LE.0) GO TO 370
(0146)          XLB=FLOAT(AUX(J))/MI
(0147)          IF(XLM.GE.XLB) GO TO 370
(0149)          XLM=XLB
(0150)          LB(1)=AUX(J)
(0151)          LB(2)=MI
(0152)    370   CONTINUE
(0153)          IF(XLM.GE.AUX(S)) GO TO 390
(0155)    380   XLM=AUX(S)
(0156)          LB(1)=AUX(S)
(0157)          LB(2)=1
(0158)    390   IF(XLM.GT.0) GO TO 44444
(0160)          IR=2
(0161)          GO TO 77777
          C.........PASSO 4.
(0162)    44444 LX=-MAX
(0163)          Q=LX*LB(2)/LB(1)
(0164)          IF(LX.LE.0.LB(1)/LB(2)) GO TO 400
(0166)          G=G+1
(0167)    400   DO 410 J=1,N
(0168)          P(J)=AUX(J)*LB(2)/LB(1)
(0169)          IF(AUX(J).LE.P(J)*LB(1)/LB(2)) GO TO 410
(0171)          P(J)=P(J)+1
(0172)    410   CONTINUE
(0173)          NS=IND(1,S,ML)
(0174)          IS=NS-1
(0175)          DO 420 I=1,ML
(0176)          IS=IS+1
(0177)          X(I)=X(I)+Q*A(IS)
(0178)    420.  CONTINUE
(0179)          DO 440 J=1,N
(0180)          IF(J.EQ.S) GO TO 440
(0182)          IS=NS-1
(0183)          DO 430 I=1,ML
(0184)          IJ=IND(I,J,ML)
(0185)          IS=IS+1
(0186)          A(IJ)=A(IJ)-P(J)*A(IS)
(0187)    430   CONTINUE
(0188)    440   CONTINUE
(0189)          IT=IT+1
(0190)          CALL TIME(TEMP,TEMPO)
(0191)          TEMPO=TEMPO/1000
(0192)          IF(TEMPO.LE.TMP) GO TO 22222
(0194)          IR=3
(0195)    77777 WRITE(5,20000) COND(IR),X(1),IT,TEMPO,(I,X(I+1),I=1,N)
(0196)    20000 FORMAT(//10X8SOLUCAO BA12/10XBUBJETIVO=BI108 (MININO)8/
                A10XBITERACOES=BI10/ 10XBTEMPO DE EXECUCAO=BF10.3B SEGUNDOS8//
                B10X8SOLUCAOB// (10X8X(BI3)=B I3) )
(0197)          GO TO 11111
(0198)    99999 STOP
(0199)          END
```

```
 5/12/77   10210 A.M.      XFORTN COMPILER (75/140).
RELEASE NUMBER= ASR 5.6       3 FLAGS     0 ERRORS
ELAPSED TIME     51 SECS      202 CARDS AT  237 C.P.M.
COMMON =       0  DATA =  34008  TEMPORARIES =     152
CODE =    8524     DIGITS
```

CAPÍTULO IV - CONCLUSÕES E ANÁLISE CRÍTICA

4.1- Metodologia

Foram montados aleatoriamente, sistemas de equações lineares, utilizando-se números pseudo-aleatórios gerados , através de função ALEAT (usando a mesma técnica utilizada pela subrotina RANDU, fornecida pela IBM para o computador IBM-1130). Esses sistemas foram desenvolvidos por dois métodos apresentados no capítulo 3.

Os programas foram escritos em linguagem FORTRAN-IV e executados em um computador B-3500, com a instrução de interromper a execução após 150 segundos para cada sistema, independentemente do número de itereções observado. O tempo utilizado para tal, é o tempo fornecido pelo relógio da máquina e não o tempo de processador. Os sistemas que tiveram sua execução interrompida são considerados rejeitados pelo método, para efeito de observações posteriores.

Foi utilizada uma precisão de 11 (onze) digitos decimais para os números inteiros durante o processamento.

4.2- Resultados Obtidos

Os resultados obtidos pelo algorítmo PRIMAL não foram satisfatórios, havendo um índice de rejeição muito elevado chegando a ser da ordem de aproximadamente 100 % .

Foi observado, verificação feita a alguns sistemas, que a solução era obtida após um pequeno número de iterações, mas o indicativo de solução ótima não era verificado levando o programa a um ciclo infinito.

Para o algorítmo DUAL, os resultados já se apre - senteram de forma satisfatória. Foram gerados para esse método, 1037 sistemas distribuidos de seguinte forma:

200	sistemas	com	5	equações	e	10	incógnitas
200	"	"	5	"	"	15	"
200	"	"	10	"	"	15	"
200	"	"	10	"	"	20	"
237	"	"	15	"	"	20	"

O resumo dos resultados obtidos são apresentados na tabela 1. A tabela 2 , apresenta um quadro de porcentagens para comparações.

4.3- Análise dos Resultados

Analisando-se as tabelas 1 e 2 observa-se:

a) Quanto às restrições:

O índice de rejeição verificado foi de 1,06%. Esse número será aceitavel dado a dificuldade de manipuleção de números inteiros, no computador e adaptações feitas, no método desenvolvido para aplicações em programação com números frecionários.

b) Quanto ao tipo de solução obtida.

Os sistemas foram gerados aleatoriamente o que não nos surpreende o aparecimento de soluções ótimas e soluções impossiveis. É interessante notar que o número de soluções impossiveis aumenta com o número de equações do sistema. Isso decorre do fato de se restringir o conjunto de soluções possiveis com o aumento de hiperplanos no conjunto formado pelas restrições.

No caso da programação contínua, as soluções possiveis estão contidas no conjunto convexo definido através das restrições do problema e a solução possivel ótima será um ponto extremo desse conjunto. As soluções possiveis para o problema de programação linear inteira se restringem ao conjunto de soluções inteiras contidas no politopo. Ao se aumentar o número de hiperplanos, mantendo-se a mesma ordem de grandeza dos termos independentes das inequações, diminui-se o volume do politopo e consequentemente, o número de soluções possiveis do seu interior, resultando um aumento das soluções impossiveis.

Nota-se uma média de 66,36 % de soluções ótimas obtidas nos sistemas de 5 restrições; 77,66 % nos sistemas com 10 , e 4,23 para os de 15 restrições.

c) Quanto ao número de iterações

Observa-se pela tabela o número de iterações não obedece nenhuma tendência a se caracterizar em função do tamanho do sistema. Apesar de apresentar uma média de 9,96 para soluções impossiveis e 8,74 para soluções ótimas (diferença de 14 %) determina-se que esses valores não são significativos, após uma análise mais detalhada.

Pode-se sugerir que o número de iterações depende das relações entre os valores (o que se observa para qualquer método iterativo), e não da ordem da matriz do sistema.

d) Quanto ao tempo de processamento

O tempo de processamento apresentado se refere ao tempo médio por iteração para as várias ordens dos sistemas testados. Conclui-se que êsse tempo depende do número de equações e do número de incógnitas.

Esse tempo tem efeito apenas comparativo uma vez que é obtido através do relógio do computador e não do processador, como já foi dito anteriormente.

TABELA 1 - Número Médio de Iterações e Tempo de Processamento

Sistemas DUAL	Sistemas Processados / Tempo	Nº de sistemas rejeitados	Solução ótima		Solução impossivel		Total	
			Nº Sistemas	Nº Iter.	Nº Sistemas	Nº Iter.	Nº Sistemas	Nº Iter.
5 X 10	200 / 0,385 s	3	112	6	85	6	197	6
5 X 15	200 / 0,675 s	1	151	7	48	9	199	7,5
10 X 15	200 / 1,334 s	4	39	13	157	9	196	10
10 X 20	200 / 1,205 s	2	49	15,5	149	13,5	198	14
15 X 20	237 / 1,600 s	1	10	16	226	10	236	10
TOTAL	1037	11	361	8,74	665	9,96	1026	9,52

Tabela 2.- Porcentagens em relação ao total

Sistema Dual	Total	Rejeições	sol. ótima	sol. imp.	total
5x10	19.286	0.289	10.800	8.197	18.997
5x15	19.286	0.096	14.561	4.629	19.190
10x15	19.286	0.386	3.760	15.140	18.900
10x20	19.286	0.193	4.725	14.368	19.093
15x20	22.856	0.096	0.964	21.794	22.758
Total	100.00	1.060	34.810	64.128	98.938

NOTAS BIBLIOGRÁFICAS

[1]SALKIN, H. M. - Integer Programming. Massachusetts, Addison Wesley Publishing, c-1975.

[2]TAHA, Handy A. - Integer Programming. (Theory, Applications and Computations), New York, Academic Press, c-1975.

[3]GREENBERG, Herold - Integer Programming. New York, Academic Press, c-1971.

[4]GASS, Saul I. - Programacion Lineal (Metodos y Aplicecione) Mexico, Compañía Editorial Continental, c-1958.

[5]LASDON, Leon S. - Optimization Theory for Large Systems. New York, The Macmillan Company, c-1970.

BIBLIOGRAFIA

AOKI, Masanao. - Introduction to Optimization Techniques (Fundamentals and Applications of Nonlinear Programming). New York, The Macmillen Company, c-1971.

CHURCHMAN, C. West & ACKOFF, Russel L. & ARNOFF, E. Leonard. - Introduction to Operations Reseerch. New York, John-Wiley , c-1957.

DANTZIG, George B. - Linear Procramming and Extensions. Princeton, Princeton University Press, c-1963.

GASS, Saul I. - Programacion Lineal (Metodos y Aplicaciones)., Mexico, Compañía Editorial Continental, c-1958.

GREENBERG, Harold - Integer Programming., New York, Academic Press, c-1971.

HADLEY, G. - Linear Programming. - Massachusetts, Addison - Wesley Publishing, c-1962.

KAUFMANN, A. & FAURE, R. - Introdução à Investigeção Operacional. trad. Dra. Maria de Fátima Fontes de Souze, Lisboa , Livros Horizonte, c-1966.

LASDON, Leon S. - Optimization Theory for Large System. New York, The Macmillan Company, c-1970.

SALKIN, H. M. - Integer Programming., Massachusetts, Addison - Wesley Publishing, c-1975.

SIMONNARD, Michel - Linear Programming., trad. Willien S. Jewell, Englewood Cliffs, Prentice-Hall, c-1957.

TAHA, Hendy A. - Integer Programming., (Theory, Applications, and Computations), New York, Academic Press, c-1975.

WAGNER, Harvey M. - Principles of Operations Reseerch (With Applications to Managerial Decisions), Englewood Cliffs , Prentice-Hall, c-1962.