UM ALGORITMO BRANCH AND BOUND PARA RESOLUÇÃO DE PROBLEMAS DE LOCALIZAÇÃO CAPACITADOS

Ronaldo Rust

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

Aprovada por:

Claudio Thomas Bornstein
(Presidente)

Nelio Domingues Pizzolato

Ronaldo Cesar Marinho Persiano

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 1983

RUST, RONALDO

Um Algoritmo Branch and Bound para Resolução de Problemas de Localização Capacitados - (Rio de Janeiro) 1983

X , 204 p. 29,7 cm (COPPE-UFRJ, M. Sc., Engenharia de Sistemas e Computação).

Tese - Universidade Federal do Rio de Janeiro. Faculdade de Engenharia

1. Problemas de Localização

I. COPPE/UFRJ

II. Título (Série)

DEDICATÓRIA

Dedico esse trabalho em primeiro lugar à minha mulher Maria Vitoria e ao meu filho Daniel, por terem me incentivado e me apoiado em todas as etapas do mestrado e terem, cada um à sua maneira, tentado compreender a minha ausência nos períodos de aula, de estudo e de confecção da tese. Sem seu amor e carinho eu talvez não tivesse conseguido concluir o trabalho, sendo que o Daniel com suas frases simples e seu maravilhoso raciocínio de criança, me fez refletir sobre a nossa tendência de complicar muitas vezes problemas simples, fazendo com que eu tentasse simplificar ao máximo a solução dos problemas aqui encontrados.

Presto uma homenagem póstuma ao meu pai que infelizmente não pode ver a conclusão da tese, mas que teve uma participação importante nela, já que me deu a infraestrutura moral e intelectual para que eu pudesse executar essa empreitada.

À minha mãe de quem eu herdei a vitalidade e que com seu carinho materno apoiou toda a minha formação, também presto uma homenagem.

Agradeço à Daisy pelo excelente trabalho de datilografia apresentado e também aos profissionais do Núcleo de Computação Eletrônica da UFRJ pela grande ajuda prestada.

Finalmente agradeço ao Professor Claudio Thomas Bornstein por ter com sua capacidade, dedicação e seriedade profissional me ajudado a concluir a bom termo esse trabalho.

RESUMO

Neste trabalho apresentamos um algoritmo baseado no método "Branch and Bound" para obtenção da solução ótima exata de problemas de localização capacitados, com restrições lineares e funções de custo côncavas ou do tipo escada. As funções côncavas normalmente aparecem em problemas onde temos a presença de economias de escala. As do tipo escada, quando temos por exemplo fábricas ou armazéns constituídos de módulos, onde cada módulo tem custo fixo.

Desenvolvemos a parte teórica, onde mostramos a filosofia do método Branch and Bound e a sua aplicação a problemas de localização capacitados com funções côncavas ou do tipo escada. Em seguida, na parte prática do trabalho, detalhamos o software que implementamos no computador Burroughs B-6700 da UFRJ e apresentamos os resultados dos testes realizados, testes estes baseados em dados reais referentes à localização de armazéns para estocagem de arroz.

ABSTRACT

We present an algorithm based on the Branch and Bound method, to obtain the exact optimal solution of capacitated location problems, with linear restrictions and concave or staircase cost functions. Concave cost functions normaly apear in problems involving economies of scale. Staircase functions are present in problems were the facilities are constructed as modules, were each module has fixed costs.

We formulate the principles of the Branch and Bound method. Then we examine the application of this method to the capacitated plant location problem with concave or staircase cost functions. We also develop the software which was implemented on a Burroughs B-6700 and we present the results of tests based on a real warehouse location problem.

ÍNDICE

Pág.

Pág.

Pág.

CAPÍTULO 1

INTRODUÇÃO

Dentre os inúmeros problemas que podem ser encontrados na prática, quando lidamos com sistemas de distribuição, destaca-se um, que é o da localização de fábricas, armazéns, silos, etc... onde os custos de produção, armazenamento e distribuição são funções côncavas das quantidades fabricadas, armazenadas e distribuídas, devido à presença de economias de escala. Em alguns casos também, as funções de custo podem ser do tipo escada, o que normalmente ocorre quando temos por exemplo armazéns constituídos de módulos, onde cada módulo tem custo fixo.

O objetivo desse trabalho é o desenvolvimento de um algoritmo baseado no método Branch and Bound, que nos leve a uma solução ótima exata de um problema de localização com limitações de capacidade na rede, restrições lineares e funções de custo côncavas, ou do tipo escada.

Objetivamos também o desenvolvimento de um software que possa ser usado com facilidade por universidades, empresas públicas ou privadas ou por qualquer pessoa que se interesse em resolver problemas dessa natureza.

A metodologia desenvolvida nessa dissertação para minimizações de funções de custo côncavas, com restrições lineares, se baseia em trabalho de Soland [22], tendo sido desenvol-

vida uma extensão do método para tratamento de problemas com funções do tipo escada.

Para tornar mais clara a aplicação do método aos nossos problemas, fazemos uma apresentação formal do método Branch and Bound, visando um entendimento da sua filosofia básica.

Apresentamos então a parte teórica do Branch and Bound e sua aplicação aos problemas citados anteriormente, envolvendo cálculo dos limites, regras de separação do problema e provas de convergência do algoritmo. Detalhamos o software desenvolvido, de modo a torná-lo o mais aberto possível e de fácil utilização. Mostramos os resultados computacionais alcançados, nos testes com dados reais, obtidos a partir de um trabalho da CIBRAZEM. Dentre os testes realizados um deles foi confrontado com resultados anteriores obtidos por recursos heurísticos, tendo sido constatado que o nosso programa, além de alcançar a solução ótima, o fez num tempo de processamento inferior.

A organização da nossa dissertação é então a seguinte:

No Capítulo 2 apresentamos a bibliografia usada na nossa pesquisa; no Capítulo 3 descrevemos um problema de localização capacitado como motivação para o nosso trabalho e fazemos a modelagem correspondente; o Capítulo 4 formaliza o método

Branch and Bound, sendo que, para uma melhor compreensão, apresentamos um exemplo de uma aplicação à programação inteira; o Capítulo 5 refere-se à aplicação do método a problemas de localização com funções côncavas; no Capítulo 6 é apresentada a adaptação do algoritmo para problemas de localização com funções do tipo escada; no Capítulo 7 detalhamos o algoritmo computacional desenvolvido, dando uma descrição detalhada das rotinas e variáveis de programa, bem como a forma de se entrar com os dados e a apresentação dos relatórios impressos; no Capítulo 8 fazemos uma análise dos testes realizados; finalmente no Capítulo 9, apresentamos as conclusões a que chegamos.

Ao final da dissertação apresentamos três anexos a saber:

Anexo I, contendo a listagem do programa fonte desenvolvido; Anexo II, constando de formulários para entrada de dados, facilitando a transcrição de dados que serão processados pelo programa e Anexo III que é um teste completo do problema 8.4.1 apresentado no Capítulo 8.

CAPÍTULO 2

REVISÃO DA LITERATURA

Fazendo uma revisão da literatura relacionada com o assunto, constatamos que já em 1958 Baumol e Wolfe |[3]|, propuseram uma técnica heurística para resolução de problemas de localização de armazéns, com funções de custo côncavas, O método envolve a resolução de um problema de transporte a cada iteração.

Após este artigo vários outros foram editados, dos quais citaremos em ordem cronológica alguns que se destacam.

Em 1966 Lawler e Wood |[13]| escreveram um artigo citando diversas aplicações do método Branch and Bound, incluindo problemas com funções não-convexas e sugerindo o uso de envoltórias convexas das funções em substituição às funções originais. Como veremos, esta é a base do nosso método proposto para o caso das funções côncavas.

Em 1969, Graciano Sá |[20]|, propõe um algoritmo Branch and Bound (exato) para resolução de um problema de localização capacitado com funções côncavas; para resolução de problemas grandes ele propõe um algoritmo não exato. Ainda no mesmo ano, dois artigos merecem ser citados: No primeiro Falk e Soland |[6]| propõem um algoritmo Branch and Bound para resolução

de problemas de localização capacitados com funções semi-contínuas inferiormente, possivelmente não convexas. Resolvem então uma série de problemas cada qual com função objetiva convexa, fazendo sucessivas partições do conjunto de soluções viáveis até a obtenção da solução final do problema original. No segundo artigo, Jones e Soland |[11]|, apresentam um algoritmo também baseado no Branch and Bound para obtenção do ótimo global de problemas onde a parte separável da função objetivo se constitui na soma de funções contínuas por partes a uma variável. O algoritmo tem a característica de gerar uma solução inicial viável boa, gerando a cada passo uma nova solução viável que é comparada com a melhor obtida até então.

Em 1970 Rech e Barton |[18]| apresentam um algoritmo para a resolução de problemas de localização capacitados onde as funções de custo podem ser lineares por partes, não convexas. São feitas aproximações das funções através de funções lineares por partes convexas. A solução é obtida através do Branch and Bound fazendo-se desdobramento do problema original sucessivamente até a obtenção da solução ótima.

Outro artigo foi publicado por Soland |[23]| em 1971 dando solução análoga para problemas com funções lineares por parte convexas.

Estes dois artigos são particularmente interessantes, pois apresentam soluções para problemas semelhantes aos nossos onde as funções de custo são do tipo "escada".

Falk e Horowitz |[5]| e Soland |[22]| apresentaram respectivamente em 1972 e 1974 algoritmos para resolução de problemas de localização capacitados com funções objetivas côncavas. Ambos usaram métodos Branch and Bound para obtenção da solução. Particularmente o artigo de Soland |[22]| serviu como base para todo o nosso trabalho.

Mais recentemente foram publicados trabalhos importantes de Johanshahlou |[10]| (1978) com problemas de localização não capacitados e funções côncavas, Florian e Robillard |[7]| (1978) com problemas de localização capacitados e funções côncavas onde uma solução ótima é encontrada pela enumeração implícita do conjunto de fluxos extremos na rede, Gallo, Sandi e Sodini |[8]| (1980), resolvendo um problema análogo, também usando o Branch and Bound no qual o esquema de enumeração é baseado na caracterização do conjunto de soluções viáveis e em 1982 com Gomes |[9]| apresentando um trabalho de tese para resolução de problemas de localização de armazéns onde o custo de armazenagem pode ser expresso por funções côncavas, usando também um algoritmo baseado no Branch and Bound.

CAPÍTULO 3

MOTIVAÇÃO

Considere-se um conjunto de armazéns já existentes ou a construir em um determinado número de cidades. Um certo número de centros produtores fabricam um determinado produto e deseja-se que toda a produção seja encaminhada aos armazéns que por sua vez a distribuirão aos centros consumidores, sendo que toda a demanda deve ser satisfeita. O período T a ser considerado é tal que todas as variações sazonais estão nele contidas, repetindo-se então no período seguinte.

Sabe-se que os custos de transporte são funções da quantidade distribuida e que os custos de armazenagem são função da quantidade estocada, Como esta última pode ser expressa, através do índice de rotação, como uma função do fluxo através dos armazéns, podemos então considerar os custos de armazenagem como uma função do fluxo.

Vamos considerar ainda que a oferta é igual a demanda dentro do período, sendo que essa condição pode sempre ser obtida, introduzindo-se centros produtores, armazéns e centros de consumo fictícios no nosso modelo.

Para tornarmos o nosso problema mais abrangente, vamos considerar que tanto os armazéns como as vias de escoamento tenham limites inferiores e superiores de capacidade.

O problema anterior pode então ser formulado matematicamente da seguinte forma:

Modelo 3.1

$$\min. \sum_{i=1}^{q} \sum_{j=1}^{m} f_{ij}(x_{ij}) + \sum_{j=1}^{m} g_j(y_j) + \sum_{j=1}^{m} \sum_{k=1}^{n} h_{jk}(z_{jk})$$

s.a.

$$\sum_{j=1}^{m} x_{ij} = p_i \qquad i = 1, \ldots, q$$

$$\sum_{j=1}^{m} z_{jk} = d_k \qquad k = 1, \ldots, n$$

$$\sum_{i=1}^{q} x_{ij} - y_j = 0, \quad \sum_{k=1}^{n} z_{jk} - y_j = 0 \qquad j = 1, \ldots, m$$

$$\ell^x_{ij} \leq x_{ij} \leq u^x_{ij} \qquad i = 1 \ldots q$$

$$\ell^z_{jk} \leq z_{jk} \leq u^z_{jk} \qquad j = 1 \ldots m \quad k = 1 \ldots n$$

$$\ell^y_j \leq y_j \leq u^y_j$$

onde

x_{ij} = fluxo do centro de produção i ao armazém j

z_{jk} = fluxo do armazém j ao centro de consumo k

y_j = fluxo no armazém j no período T

$f_{ij}(x_{ij})$ = custo da distribuição de x_{ij} unidades do produto do centro de produção i ao armazém j no período T

$g_j(y_j)$ = custo da construção, manutenção e armazenagem do armazém j no período T

$h_{jk}(z_{jk})$ = custo da distribuição de z_{jk} unidades do produto, do armazém j ao centro de consumo k

p_i = produção do centro i

d_k = demanda do centro k

ℓ^x_{ij}, ℓ^y_j, ℓ^z_{jk} = limites inferiores para as variáveis x_{ij}, y_j e z_{jk}, respectivamente

u^x_{ij}, u^y_j, u^z_{jk} = limites superiores para as variáveis x_{ij}, y_j e z_{jk}, respectivamente

q, m, n = número de centros produtores, locais candidatos para armazéns e centros consumidores, respectivamente

CAPÍTULO 4

O "BRANCH AND BOUND"

4.1 - INTRODUÇÃO

Desde 1960, quando Land e Doig [12] apresentaram um algoritmo computacional denominado "Branch and Bound", para a resolução de problemas de programação inteira mixta, uma grande variedade de algoritmos baseados nessa técnica vêm sendo desenvolvidos e aperfeiçoados em diversas aplicações da Pesquisa Operacional, como por exemplo: em programação inteira e mista e seus casos particulares, no problema do caixeiro viajante, nos problemas de localização e atribuição, como também em problemas de programação não linear e outros.

Como cada autor normalmente aborda o "Branch and Bound" sob um ponto de vista restrito à sua aplicação particular, foram feitas várias tentativas de se dar uma descrição geral do método, destacando-se Egon Balas [2], Norman Agin [1], Mitten [16], B. Roy [19], Lawler [13] e outros.

Neste capítulo objetivamos fazer um resumo do método, visando a um entendimento da filosofia básica nele contida, tentando na medida do possível evitar a generalização ou a particularização excessivas, ambas prejudiciais ao nosso intento.

4.2 - DEFINIÇÕES BÁSICAS

Vamos considerar o seguinte problema geral P:

$$\begin{cases} \text{Min } f(x) \\ \text{s.a. } x \in S \end{cases}$$

onde S é um conjunto compacto.

Vamos assumir que o problema não possa ser equacionado por um método direto de resolução de problemas de programação matemática, tais como: o simplex, o out-of-kilter, o algoritmo de transportes, etc... Mais particularmente podemos considerar que a função objetivo seja não-linear, ou que S ou parte dele seja um conjunto discreto, que são os casos mais comuns para aplicação do método.

Então:

a) Chamaremos de Problema Relaxado ou Simplificado ao problema P^0 obtido a partir do problema P e com as seguintes características gerais:

- P^0 deve ser uma "extensão" do problema original, de modo a conter todas as soluções viáveis de P.

- P^0 deve ser de resolução mais fácil do que P.

b) A partir do problema P^0 geramos novos problemas p^1, p^2...que conterão subconjuntos do conjunto de soluções viáveis de p^0. Estes problemas darão por sua vez origem a novos problemas até que o algoritmo chegue à solução ótima. Tal processo recebe o nome de separação (branching) observando-se o seguinte:

- Ao longo do processo não se deve eliminar nenhuma solução viável do problema original que tenha condições de ser solução ótima de P.

- A solução dos novos problemas deverá nos aproximar da solução do problema P.

c) Cada problema P^j é avaliado em termos de seu potencial, isto é, suas possibilidades de fornecer a solução ótima do problema dado P. Tal etapa é denominada de Avaliação (bounding).

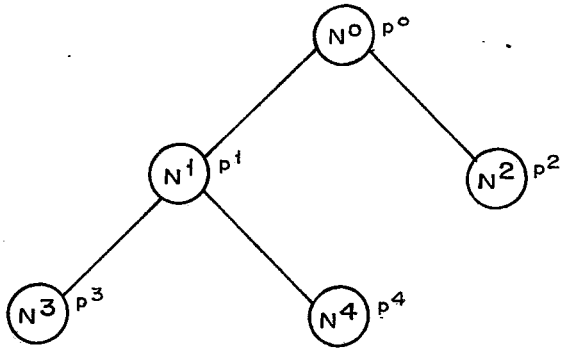
Embora existam diversas formas de se fazer a Avaliação, normalmente, para um problema de minimização, procede-se da seguinte maneira:

- Calculo de um Limite Superior LS para a função objetivo do problema dado P, correspondente a uma solução viável de P. Na prática todo $x^i \in S$ fornece um limite superior $f(x^i)$.

- Cálculo de Limites Inferiores LI (P^j) para a função objetivo de cada problema P^j. Na medida que LI (P^j) > LS, o pro-

blema P^j poderá ser eliminado, não precisando ser separado, pois certamente não contribuirá para encontrar a solução ótima de P.

d) O processo de separações e avaliações sucessivas pode ser visualizado como uma árvore, onde a raiz (nó N^0) representa o problema P^0 e os demas nos N^j representam os problemas P^j gerados sucessivamente a partir de P^0.



Assim,

Ramificação de um nó N^j, consideraremos como sendo a geração de dois novos nós a partir de N^j, em função da separação do problema P^j. Na realidade, Nós e Problemas se confundem, sendo que em alguns casos usaremos por comodidade, N^j ou P^j para representar o problema P^j ou o nó N^j respectivamente.

e) Denominamos de nós abertos aqueles que ainda não foram ramificados, (como vemos são as folhas da árvore correspondendo a problemas ainda não separados). Os demais serão considerados nós fechados.

Daremos o nome de A^k ao conjunto de nós abertos em uma determinada etapa k do algoritmo. Computacionalmente falando, é

necessário mantermos a cada etapa k uma lista de nós abertos, pois entre esses escolheremos o próximo nó a ser ramificado.

f) A escolha do próximo nó a ser ramificado pode ser feito de várias maneiras, dentre as quais podemos citar como exemplos:

- O critério LIFO (Last-in-first-out), onde o último a entrar na lista de abertos é o primeiro a ser ramificado. Esse critério é conveniente quando queremos aproveitar a inversa da base, B^{-1}, do Tableaux Simplex do último estágio.

- O critério de se escolher o nó N^j, tal que a solução de P^j seja a melhor da lista de abertos. Por exemplo: esse critério é o preferido quando o processo de branching vai sucessivamente aproximando a função objetivo dos problemas P^j da função objetivo f(x) de P. (Usaremos esse critério no algoritmo para resolução dos nossos problemas de localização).

4.3 - O "BRANCH AND BOUND" APLICADO À PROGRAMAÇÃO LINEAR INTEIRA

Seja o problema de programação linear inteira:

$$P \equiv \begin{cases} \text{Min } f(x) \\ \text{s.a. } x \in S \\ x_i \text{ inteiro} \\ x_i \geq 0 \end{cases}$$

O método "Branch and Bound", denominado em francês, com muita propriedade, de "Methode de Séparation et Évaluation Progressive" (S.E.P.), aplicado ao problema P, pode ser representado pelo seguinte algoritmo: (Maculan |15|, Salkin |21| ou Taha |24|).

Passo 0

0.1 K = 0

0.2 Resolva o problema P^0 definido por: $\left\{\begin{array}{l} \text{Min } f(x) \\ \text{s.a. } x \in S \\ x_i \geq 0 \end{array}\right\} \equiv x \in S^0$

(repare que relaxamos a condição de x_i ser inteiro)

0.3 Se o problema tiver solução, coloque P^0 na lista de nós abertos.

Passo 1

1.1 Se não há nó aberto, pare ! → o problema P não tem solução

1.2 Selecione da lista de abertos o nó N^{k*} tal que:
$f(x^{k*}) \leq f(x^i) \ \forall \ N^i \in A^k$

1.3 Se x^{k*}, solução ótima de P^{k*}, tiver todas as componentes inteiras, pare ! $\rightarrow$ x^{k*} é a solução ótima de P

Passo 2

2.1 Faça k = k+1;

2.2 Faça a separação do problema P^{k*}, isto é, defina os problemas

$$P^{2k-1}\left\{\begin{array}{l}\text{Min } f(x)\\ \text{s.a.}\\ \quad x \in S^{k*}\\ \quad x_g \leq \lfloor x_g^{k*} \rfloor\end{array}\right\} \equiv x \in S^{2k-1}$$

$$P^{2k}\left\{\begin{array}{l}\text{Min } f(x)\\ \text{s.a.}\\ \quad x \in S^{k*}\\ \quad x_g \geq \lfloor x_g^{k*} \rfloor + 1\end{array}\right\} \equiv x \in S^{2k}$$

onde x_g^{k*} é uma componente não inteira de x^{k*} e $\lfloor x_g^{k*} \rfloor$ é o maior inteiro menor ou igual a x_g^{k*}. O que fizemos foi meramente "forçar" x_g a assumir um valor inteiro imediatamente inferior ou superior a ele na solução dos problemas P^{2k-1} e P^{2k}

Passo 3

3.1 Faça a avaliação dos novos problemas gerados, isto é:

Calcule as soluções de P^{2k-1} e P^{2k}

3.2 Guarde na lista de nós abertos A^k, os problemas que tiverem solução

3.3 Vá para o passo 1

4.4 - EXEMPLO EXPLICATIVO

Daremos um exemplo simples de uma aplicação à programação linear interna, acompanhando passo a passo o desenvolvimento do algoritmo até a obtenção da solução ótima.

Seja o problema: (Maculan |15|)

$$P \equiv \left\{ \begin{array}{l} \text{Min } f(x) = 4x_1 + 5x_2 \\ \text{s.a. } x_1 + 4x_2 \geq 5 \\ \quad\quad 3x_1 + 2x_2 \geq 7 \\ \quad\quad x_1 \geq 0, \text{ inteiro} \\ \quad\quad x_2 \geq 0, \text{ inteiro} \end{array} \right\} \; x \in S$$

Vamos então definir o problema simplificado

$$P^0 \equiv \left\{ \begin{array}{l} \text{Min } f(x) = 4x_1 + 5x_2 \\ \text{s.a.} \quad x_1 + 4x_2 \geq 5 \\ \qquad\quad 3x_1 + 2x_2 \geq 7 \\ \qquad\quad x_1 \, , \, x_2 \geq 0 \end{array} \right\} \; x \in S^0$$

Este problema como vemos é uma simplificação do problema P, podendo ser resolvido pelo método Simplex, contendo todas as soluções viáveis de P, podendo eventualmente nos fornecer uma solução que seja também solução ótima de P.

Vamos então aplicar o algoritmo ao nosso problema. Para melhor visualização da evolução do algoritmo, substituimos a lista de abertos pela árvore gerada, onde como já vimos as folhas representam os nós abertos.

Iteração 0

Passo 0

0.1 k = 0

0.2 Solução de P^0 $\left\{ \begin{array}{l} f(x^0) = 112/10 \\ x^0 \quad = (18/10, \; 8/10) \end{array} \right.$

03. Árvore gerada: (N^0) $f(x^0) = 112/10$

$x^0 = (18/10, \; 8/10)$

Passo 1

1.1 $A^0 \neq \phi$

1.2 $N^{k*} = N^0$

1.3 As componentes de x^0 não são inteiras $\rightarrow$ x^0 não é solução de P

Passo 2

2.1 K = 1

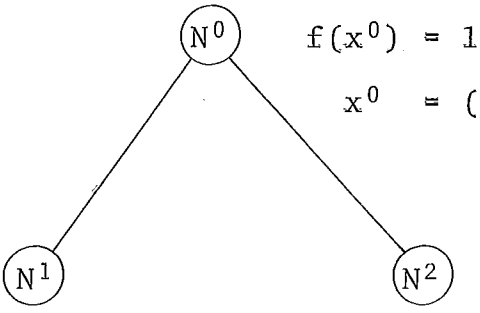
2.2 Variável escolhida: $x_1^0 = 18/10$

$$S^1 \equiv \begin{cases} x_1 + 4x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 7 \\ x_1 , x_2 \geq 0 \\ x_1 \leq 1 \end{cases} \qquad S^2 \equiv \begin{cases} x_1 + 4x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 7 \\ x_1, x_2 \geq 0 \\ x_1 \geq 2 \end{cases}$$

Passo 3

3.1 Solução de $p^1 \begin{cases} f(x^1) = 14 \\ x^1 = (1,2) \end{cases}$ solução de $p^2 \begin{cases} f(x^2) = 47/4 \\ x^2 = (2, 3/4) \end{cases}$

3.2 Árvore gerada:



$f(x^0) = 112/10$

$x^0 = (18/10,\ 8/10)$

$f(x^1) = 14$

$x^1 = (1,2)$

$f(x^2) = 47/4$

$x^2 = (2,\ 3/4)$

Iteração 1

Passo 1

1.1 $A^1 \neq \phi$

1.2 $N^{k*} = N^2$

1.3 x^2 tem componentes não inteiras $\rightarrow$ x^2 não é solução de P.

Passo 2

2.1 K = 2

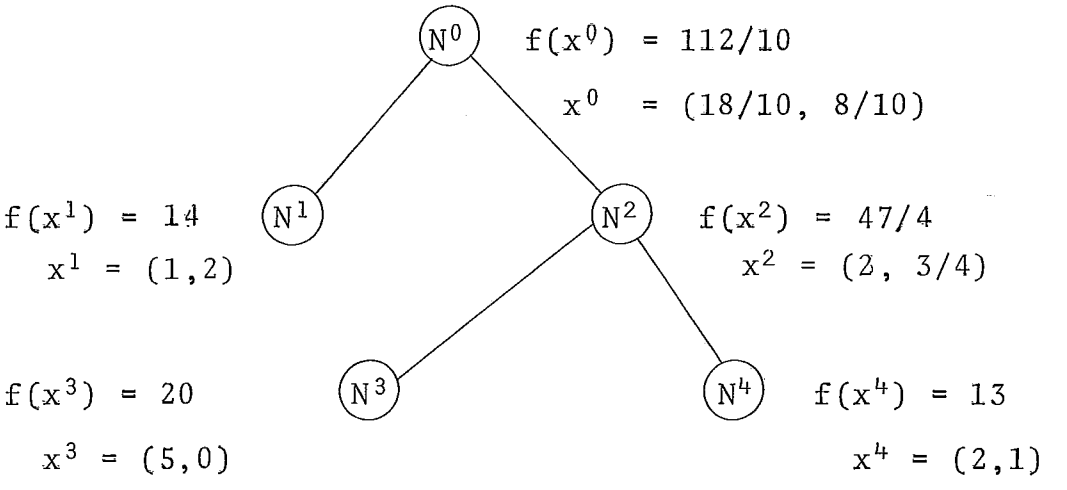
2.2 Variável escolhida: $x_2^2 = 3/4$

$$S^3 \equiv \begin{cases} x_1 + 4x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 7 \\ x_1 , x_2 \geq 0 \\ x_1 \geq 2 \\ x_2 \leq 0 \end{cases} \qquad S^4 \equiv \begin{cases} x_1 + 4x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 7 \\ x_1 , x_2 \geq 0 \\ x_1 \geq 2 \\ x_2 \geq 1 \end{cases}$$

Passo 3

3.1 Solução de $P^3 \begin{cases} f(x^3) = 20 \\ x^3 = (5,0) \end{cases}$ solução de $P^4 \begin{cases} f(x^4) = 13 \\ x^4 = (2,1) \end{cases}$

3.2 Árvore gerada:



Iteração 2

Passo 1

1.1 $A^2 \neq \phi$

1.2 $N^{k*} = N^4$

1.3 Como vemos x^4 tem as componentes inteiras. Então pare! → Solução ótima $\begin{cases} f(\overline{x}) = 13 \\ \overline{x} = (2, 1) \end{cases}$

4.5 - CONCLUSÃO

O método pode ser aplicado a uma grande gama de problemas de programação linear, não linear, dinâmica ou outros. Para cada aplicação terão que ser feitas as adptações necessárias, tomando-se o cuidado de definir os diversos procedimentos de forma a garantir a convergência do algoritmo e também diminuir ao máximo o número de iterações e o trabalho computacional gasto em cada iteração.

Egon Balas | 2 |, faz uma prova da convergência do algoritmo em condições bastante particulares. De uma maneira geral teremos que fazer uma prova de convergência específica para cada caso, de acordo com a definição dada aos problemas P^j, as regras de separação e avaliação, as regras de parada, a escolha do nó a ser ramificado, etc...

É fácil verificar também que em problemas de grande porte a tendência natural é a arborescência se estender muito, implicando na resolução de uma quantidade grande de problemas P^j e na ocupação de um espaço muito grande em memória para se guardar os nós abertos. É esse então o "termômetro" que normalmente indica a conveniência ou não de se aplicar o método a

um determinado tipo de problema.

CAPÍTULO 5

APLICAÇÃO DO B&B A UMA REDE DE FLUXO CAPACITADA COM F.O. CÔNCAVA

5.1 - INTRODUÇÃO

Seja a generalização do nosso modelo 3.1:

$$\text{Problema P} \begin{cases} \text{Min. } f(x) = \sum_{i=1}^{n} f_i(x_i) \\ \text{s.a.} \\ \qquad x \in G \equiv \{x \mid Ax = b\} \\ \qquad x \in C \equiv \{x \mid \ell \leq x \leq u\} \end{cases}$$

onde:

$x = (x_1, x_2, \ldots, x_n)$ → vetor que representa os fluxos ao longo dos n arcos

G → restrições de atendimento à oferta, demanda, coerência entre os fluxos, condições particulares, etc...

C → restrições relativas ao problema capacitado, isto é, impondo limites superiores e inferiores para o fluxo, (o conjunto C será por nós denominado de retângulo);

f_i → custo ao longo do arco i. As funções f_i são côncavas sobre o intervalo $[\ell_i, u_i]$

Para o problema acima, f assume o seu valor mínimo em um ponto extremo ou vértice do politopo linear G ∩ C. Isso pode ser garantido pelo Teorema 5.2.1 que veremos a seguir.

Soland [22] propõe um algoritmo, que será apresentado a seguir, baseado no Branch and Bound para resolver o problema P. O algoritmo resulta em uma sequência finita de programas lineares, todos com o mesmo espaço de restrições G ∩ C. O fato de não se modificar o espaço de soluções viáveis é como veremos muito vantajoso já que cada novo problema será uma simples parametrização do problema anterior.

5.2 - DEFINIÇÕES, LEMAS E TEOREMAS BÁSICOS

Em primeiro lugar vamos dar algumas definições e um lema necessários à demonstração do Teorema 5.2.1.

Definição 5.2.1

Uma função f definida sobre um conjunto convexo S é dita côncava se, para todo x_1 e x_2 ε S e todo α ε [0, 1], temos:

$$f(\alpha x_1 + (1 - \alpha)\, x_2) \geq \alpha\, f(x_1) + (1 - \alpha)\, f(x_2)$$

Definição 5.2.2

H é dito um hiperplano de suporte para um conjunto

convexo S, quando um de seus semi-espaços fechados contém S e além disso contém um ponto da fronteira de S.

Definição 5.2.3

Um ponto $x \in S$ convexo é dito ponto extremo desse conjunto S se não existem dois pontos distintos x_1 e $x_2 \in S$ tal que $x = \alpha x_1 + (1 - \alpha) x_2$ para algum $\alpha \in (0, 1)$.

Por exemplo:

Em E^2 os pontos extremos de um quadrado são os seus vértices.

Lema 5.2.1

Seja S um conjunto convexo, H um hiperplano de suporte para S e T a interseção de H e S. Então, todo ponto extremo de T é um ponto extremo de S.

Prova: Seja $x_o \in T$. Então cabe demonstrar que:

x_o é ponto extremo de T $\rightarrow$ x_o é ponto extremo de S.

Para isso usaremos a proposição equivalente:

x_o não é ponto extremo de S $\rightarrow$ x_o não é ponto extremo de T.

Suponhamos então que $x_0 \in T$ não seja um ponto extremo de S.

Então $x_0 = \alpha x_1 + (1 - \alpha)\, x_2$ para algum x_1 e $x_2 \in S$, $x_1 \neq x_2$ e $\alpha \in (0, 1)$.

Seja H definido por: $H = \{x: ax = s\}$ de modo que o subespaço $ax \geq s$ contém S.

Então: $ax_1 \geq s$ e $ax_2 \geq s$.

Seja então: $ax_1 = s + \theta_1$ e $ax_2 = s + \theta_2$ com $\theta_1 \geq 0$ e $\theta_2 \geq 0$ (1)

Logo: $ax_0 = a(\alpha x_1 + (1-\alpha)x_2) = \alpha ax_1 + (1-\alpha)\, ax_2 = s$ (2)

Substituindo (1) em (2) temos:

$$\alpha(s + \theta_1) + (1-\alpha)\,(s + \theta_2) = \alpha s + \alpha\theta_1 + s + \theta_2 - \alpha s - \alpha\theta_2 = s$$

daí: $\alpha\theta_1 + (1-\alpha)\,\theta_2 = 0$

Como $\alpha > 0$, $(1-\alpha) > 0$, $\theta_1 \geq 0$ e $\theta_2 \geq 0$ segue:

$$\theta_1 = \theta_2 = 0$$

Logo: $ax_1 = s$ e $ax_2 = s$

e portanto x_1 e $x_2 \in H$ e consequentemente x_1 e $x_2 \in T$ e x_0 não é ponto extremo de T. ▒

Podemos agora enunciar e demonstrar o teorema que garante que a solução ótima do problema P se encontra em um dos vértices do politopo linear $G \cap C$ que é compacto já que o conjun

to C é fechado e limitado.

Teorema 5.2.1: (Luenberger |14|)

Seja f uma função côncava definida no conjunto compacto S. Se f tem um mínimo sobre S, este se encontra sobre um ponto extremo (vértice) de S.

Prova:

Suponhamos que f atinja o seu mínimo em $x^* \in S$.

a) Vamos mostrar primeiro que este mínimo se encontra em um ponto da fronteira de S.

Se x^* já é um ponto da fronteira então não há nada a provar. Vamos então assumir que x^* não é um ponto da fronteira.

Seja L uma reta passando pelo ponto x^*.

A interseção dessa reta com S é um intervalo dessa reta L, tendo por pontos extremos y_1 e y_2 que são pontos da fronteira de S, e tais que $x^* = \alpha y_1 + (1-\alpha)y_2$ para um dado $\alpha \in (0, 1)$. Pela concavidade de f temos:

$$f(x^*) = f(\alpha y_1 + (1-\alpha)\, y_2) \geq \alpha f(y_1) + (1-\alpha)\, f(y_2) \geq$$
$$\geq \min \{f(y_1), f(y_2)\}$$

então $f(y_1)$ ou $f(y_2)$ tem que ser menor ou igual a $f(x^*)$. Portanto se x^* é um ponto de mínimo, também o será y_1 ou y_2 que são pontos da fronteira de S.

b) Vamos agora mostrar que o mínimo, se encontra em um ponto extremo de S, onde S é definido em um espaço de dimensão n.

Se x^* já é um ponto extremo de S então não há nada a provar. Vamos assumir então que x^* não é ponto extremo de S.

Consideremos então a interseção de S com um hiperplano de suporte H contendo x^*. Essa interseção T_1, tem dimensão menor ou igual a n-1 e o mínimo global de f sobre T_1 é igual a $f(x^*)$ e de acordo com (a) tem que se encontrar em um ponto x_1 da fronteira de T_1.

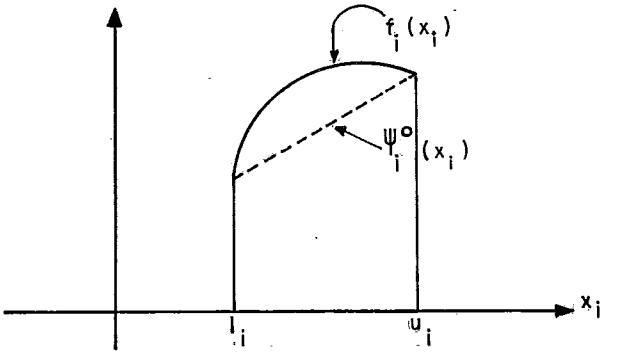
Se esse ponto é um ponto extremo de T_1, pelo Lema 5.2.1, também será um ponto extremo de S e o Teorema está provado. Caso contrário, nós tomaremos T_2, de dimensão menor ou igual a n-2, interseção de T_1 com um hiperplano de suporte contendo x_1. Esse processo pode continuar no máximo n vezes quando um conjunto T_n de dimensão zero, consistindo de um único ponto é obtido. Esse ponto extremo de T_n é portanto também um ponto extremo de S. ▒

5.3 - O ALGORITMO PROPOSTO

Seja o problema P, proposto no item 5.1.

O algoritmo se desenvolve baseado no seguinte raciocínio:

Inicialmente nós linearizamos todas as funções $f_i(x_i)$ da seguinte forma:

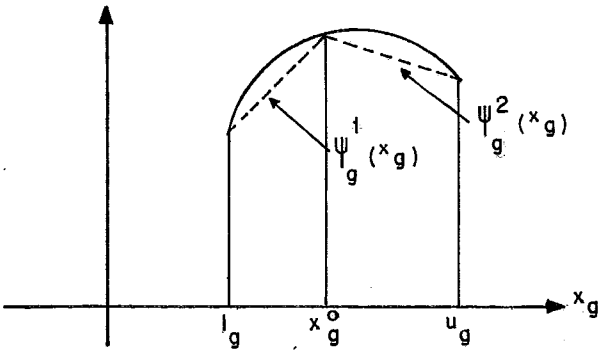


Então como vemos $\psi_i^0(x_i) \leq f_i(x_i)\ \forall x_i \in [\ell_i, u_i]$ (1)

Resolvemos então o problema linear $P^0 \equiv \begin{cases} \text{Min } \psi^0(x_o) = \sum_{i=1}^{n} \psi_i^0(x_i) \\ \text{s.a. } Ax = b \\ \ell \leq x \leq u \end{cases}$

Denominando de x^0 a solução do problema linear P^0, podemos verificar facilmente baseados em (1) que: $\psi^0(x^0) \leq f(x^0)$

Escolhemos então um dos arcos onde $x_i^0 \neq \ell_i$ e $x_i^0 \neq u_i$, digamos o arco g, e definimos para ele duas novas funções lineares:



Podemos então definir a partir de P^0 dois novos problemas substituindo a função $\psi_g^0(x_g)$ pelas funções $\psi_g^1(x_g)$ e $\psi_g^2(x_g)$ respectivamente, desejando com isso "aproximar" as funções $\psi^j(x)$ da função $f(x)$ de modo a obtermos uma solução "melhor" para o problema linearizado. Esse processo de "separação" do problema em dois constitui o "branching" do algoritmo e o cálculo da solução de cada problema linear $\psi^j(x)$, constitui a "avaliação" ou "bounding", pois como vemos, achamos para cada um deles um limite inferior ("lower bound").

Repetimos então os processos de separação e avaliação até que a solução ótima seja encontrada (a menos de um valor ε).

O grande trunfo que temos é o fato de que cada novo problema linear se resume na parametrização da função objetivo do problema anterior, já que usamos um artifício para continuarmos com o mesmo conjunto de restrições a cada iteração.

Daremos agora algumas definições necessárias ao entendimento do algoritmo:

a) Separar um problema P^j corresponde a particionar o subretângulo $C^j \subset C$ em dois subretângulos $C^{j(1)}$ e $C^{j(2)}$ onde $C^{j(1)} \cup C^{j(2)} = C^j$, definindo duas novas funções objetivo conforme vimos anteriormente.

b) A avaliação para um problema P^j consiste em calcular o valor ótimo para a função objetivo linearizada de P^j que nos fornecerá como veremos um limite inferior para f(x) no subretângulo C^j que chamaremos de LI (N^j).

c) Consideramos que:

N^0, N^1, N^2 ... são os nós da árvore gerada pelo algoritmo onde N^0 é o nó inicial. Correspondem aos problemas P^0, P^1, P^2 ... obtidos pela divisão do retângulo C em subretângulos C^1, C^2 ... de acordo com o que foi especificado em a)

Os nós N^{2k-1} e N^{2k} são criados no estágio k do algoritmo.

d) LS_f^j é o melhor valor encontrado para f(x) considerando-se os nós N^0, N^1, N^2 ... N^j, isto é, se x^0, x^1, x^2 ... x^j forem as soluções ótimas encontradas para os problemas lineares correspondentes a estes nós, então:

$$LS_f^j = \min\{f(x^0), f(x^1), f(x^2), \ldots, f(x^j)\}$$

Podemos considerá-lo com sendo um limite superior para o valor ótimo da função objetivo do problema P, já que $x^0, x^1, \ldots, x^j \in G \cap C$.

LS_x^j é a solução viável que corresponde a LS_f^j, isto é $LS_f^j = f(LS_x^j)$.

e) A^k é o conjunto de nós abertos na etapa k.

Vamos então elaborar o algoritmo:

Passo 0

k = 0;

Calcule $LI(N^0)$, LS_f^0, LS_x^0

Se não tiver solução → infactível

Coloque o problema P^0 na lista de abertos caso ele tenha solução

Passo 1

Selecione da lista de abertos, (retirando da lista), um nó N^{k*} tal que:

$$LI(N^{k*}) = \min_{N^i \in A^k} \{LI(N^i)\}$$

Então:

Se $LI(N^{k*}) \geq LS_f^{2k} - \varepsilon \rightarrow$ Pare ! LS_x^{2k} é a solução ótima.

Passo 2 (separação)

$k = k + 1$

Parta do nó N^{k*} para criar os nós N^{2k-1} e N^{2k}, de tal maneira que $C^{2k-1} \cup C^{2k} = C^{k*}$

Passo 3 (Avaliação)

Calcule $LI(N^{2k-1})$ e $LI(N^{2k})$ e atualize LS_f^{2k} e LS_x^{2k}, se necessário, colocando os problemas na lista de abertos.

Vá para o passo 1.

5.4 - DETALHAMENTO DO ALGORITMO

Passo 0

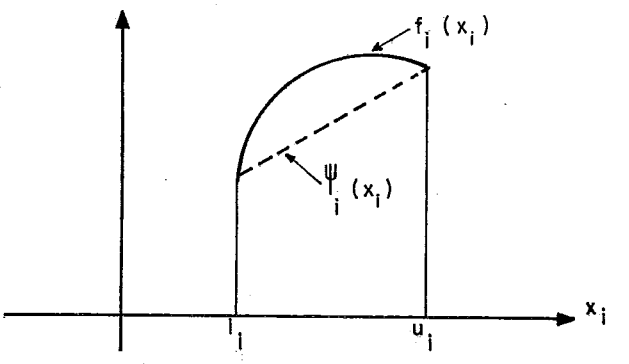
É o passo de inicialização do algoritmo.

Nele é feita a primeira avaliação do problema P, isto é, é calculado um limite inferior para a solução do problema P que é o $LI(N^0)$.

Além disso são calculados os valores LS_f^0 e LS_x^0.

Esses cálculos são feitos da seguinte forma:

Linearizamos a função objetivo do problema P para podermos usar técnicas de programação linear para resolvê-lo. Essa linearização pode ser vista pelo gráfico:



$\psi_i(x_i)$ corresponde a linearização de $f_i(x_i)$, sendo que, como vemos:

$$\begin{cases} \psi_i(\ell_i) = f_i(\ell_i) \\ \psi_i(u_i) = f_i(u_i) \end{cases}$$

Resolvido o problema linearizado, encontramos uma solução ótima x^0, que no entanto é somente uma solução viável do problema P dado.

Como $f_i(x_i)$ é côncava p/i=1,2...n, temos pela definição de função côncava no intervalo $[\ell_i, u_i]$

$$\forall\ \alpha\ \varepsilon [0, 1] \rightarrow f_i(\alpha(\ell_i) + (1-\alpha)\ (u_i)) \geq \alpha\ f_i(\ell_i) + (1-\alpha) f_i(u_i) \qquad (1)$$

Para a reta definida em $[\ell_i, u_i]$ temos:

$$\forall\ \alpha\ \varepsilon\ [0,\ 1] \rightarrow \psi_i(\alpha(\ell_i) + (1-\alpha)(u_i)) = \alpha\ \psi_i(\ell_i) + (1-\alpha)\ \psi_i(u_i) \tag{2}$$

Como pela construção da nossa reta $\begin{cases} \psi_i(\ell_i) = f_i(\ell_i) \\ \psi_i(u_i) = f_i(u_i) \end{cases}$ (3)

Temos de (1), (2) e (3)

$$f_i(\alpha(\ell_i) + (1-\alpha)(u_i)) \geq \psi_i(\alpha(\ell_i) + (1-\alpha)u_i)\ \forall\ \alpha\ \varepsilon\ [0,\ 1]$$

Então concluimos que:

$$f_i(x_i) \geq \psi_i(x_i)\ \forall\ x_i\ \varepsilon\ [\ell_i,\ u_i] \quad i = 1,\ 2,\ \ldots,\ n$$

O que podemos constatar pelo gráfico das funções.

Logo:

$$\sum_{u=1}^{n} \psi_i(x_i) = \psi(x) \leq f(x) = \sum_{i=1}^{n} f_i(x_i) \text{ para } x\ \varepsilon\ C \tag{4}$$

Ora, como x^0 é a solução ótima do problema linearizado:

$$\psi(x^0) \leq \psi(x) \quad \forall x\ \varepsilon\ G \cap C \tag{5}$$

De (4) e (5):

$$\psi(x^0) \leq f(x) \quad \forall x\ \varepsilon\ G \cap C$$

Temos então um limite inferior para o problema P, que é:

$$LI(N^0) = \psi(x^0) = \sum \psi_i(x_i^0)$$

Além disso como $x^0 \in G \cap C$, $f(x^0)$ fornece um limite superior para o valor ótimo (mínimo) da função objetivo em N^0.

Temos então:

$$\begin{cases} LS_f^0 = f(x^0) \\ LS_x^0 = x^0 \end{cases}$$

Passo 1

Nesse passo escolhemos o nó (problema) a ser ramificado (separado), bem como verificamos se já é possível identificar a solução ótima.

$LI(N^{k*}) = \underset{N^i \in A^k}{\text{Min}} \; LI(N^i)$ nos dá um limite inferior para o valor ótimo da função objetivo do problema P. Isso pode ser garantido pelo fato de ao fazermos a ramificação (passo 2) nós dividimos o problema P^k em dois (duas folhas) calculando para cada um o valor LI (N^j) de forma a não eliminar nenhuma solução viável do espaço $G \cap C$.

Como até a etapa k nós geramos N^0, N^1, ... , N^{2k} e LS_f^{2k} é por definição a melhor solução (menor valor) encontrada para f(x) nesta etapa, então, se tivermos:

$$LI(N^{k*}) \geq LS_f^{2k} - \varepsilon$$

nós estamos de posse da solução ótima $\bar{x} = LS_x^{2k}$, a menos de uma tolerância ε.

Fica então provada a otimalidade do algoritmo, ou seja, o fato da regra de parada implicar na determinação da solução ótima do problema.

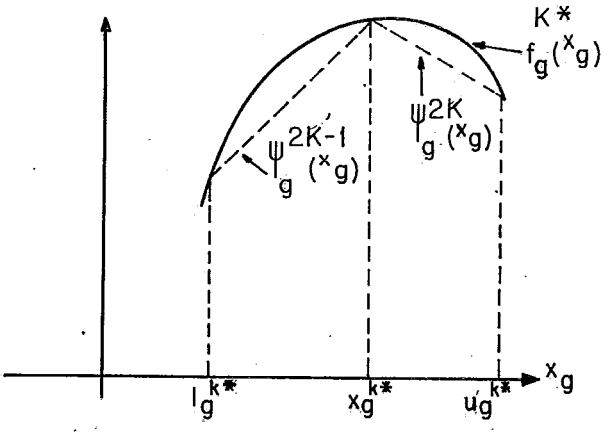
Passo 2

Essa é a etapa de ramificação do algoritmo.

Primeiro fazemos k = k + 1, contando mais um ciclo.

A ramificação é feita, utilizando-se o nó N^{k*} que como vimos corresponde ao menor limite inferior das folhas.

A ramificação é feita então em um determinado arco g como mostra a figura:



Precisamos então saber qual vai ser o arco g que vai originar a ramificação do problema. Essa escolha é feita de tal maneira que:

$$f_g(x_g^{k*}) - \psi_g^{k*}(x_g^{k*}) = \max_{i=1\ldots n} \{f_i(x_i^{k*}) - \psi_i^{k*}(x_i^{k*})\}$$

Ramificamos o nó N^{k*} em dois outros nós, N^{2k-1} e N^{2k}, tais que:

$$N^{2k-1} \rightarrow \ell_g^{k*} \leq x_g \leq x_g^{k*}$$

e

$$N^{2k} \rightarrow x_g^{k*} \leq x_g \leq u_g^{k*}$$

Para os demais arcos manteremos os mesmos intervalos do nó N^{k*}. Isto é, chamando de C^{2k-1} e C^{2k} respectivamente os intervalos correspondentes aos nós N^{2k-1} e N^{2k}, temos:

$$C^{2k-1}\begin{cases} \ell_g^{2k-1} = \ell_g^{k*} \leq x_g \leq x_g^{k*} = u_g^{2k-1} \\ \ell_i^{2k-1} = \ell_i^{k*} \leq x_i \leq u_i^{k*} = u_i^{2k-1} \quad \forall_i \neq g \end{cases}$$

$$C^{2k}\begin{cases} \ell_g^{2k} = x_g^{k*} \leq x_g \leq u_g^{k*} = u_g^{2k} \\ \ell_i^{2k} = \ell_i^{k*} \leq x_i \leq u_i^{k*} = u_i^{2k} \quad \forall_i \neq g \end{cases}$$

Poderíamos supor então que de posse de dois novos conjuntos de restrições $G \cap C^{2k-1}$ e $G \cap C^{2k}$ para os nós N^{2k-1} e N^{2k} passaríamos ao passo 3 para fazer a avaliação. Na verdade, como veremos, manteremos para todos os problemas o mesmo espaço de soluções viáveis $G \cap C$.

A ramificação então só servirá para determinar as funções objetivo $\psi_g^{2k-1}(x_g)$ e $\psi_g^{2k}(x_g)$ para resolução dos problemas P^{2k-1} e P^{2k} linearizados.

Justificaremos os procedimentos acima mais adiante.

Passo 3

Esse é o passo de avaliação do algoritmo.

A avaliação é feita analogamente ao procedimento do passo 0 com a seguinte ressalva:

Digamos que queiramos achar $LI(N^j)$ (no nosso caso $j = 2k - 1$ ou $j = 2k$).

Então:

$LI(N^j) = Min\{\psi^j(x) \mid x \in G \cap C^j\}$ seria um limite inferior para $Min\{f(x) \mid x \in G \cap C^j\}$

Porém $C^j \subseteq C$ e então $G \cap C^j \subseteq G \cap C$.

Logo:

$$Min\{\psi^j(x) \mid x \in G \cap C\} \leq Min\{\psi^j(x) \mid x \in G \cap C^j\} \leq Min\{f(x) \mid x \in G \cap C^j\}$$

Daí nós poderemos usar o conjunto de restrições $G \cap C$ para cálculo de $LI(N^j)$ e o problema linearizado será:

$$\begin{cases} \text{Min } \psi^j(x) = \sum_{i=1}^{n} \psi_i^j(x_i) \\ \\ \text{s.a. } x \in G \cap C \end{cases}$$

Evidentemente, podemos assim obter um limite inferior "pior", isto é, menor do que se usassemos $G \cap C^j$. A vantagem no entanto é que o conjunto $G \cap C$ independe do nó N^j considerado e portanto teremos o mesmo conjunto de soluções viáveis para todos os nós. Isso vai permitir que para a resolução de um problema nós partamos da solução do problema anterior. Podemos então obter uma redução substancial no número de iterações para

a resolução dos problemas lineares.

O fato de usarmos o conjunto de restrições $G \cap C$ pode ocasionar que a solução do problema P^j linear, caia fora do intervalo $[\ell^j, u^j]$. Digamos que em alguma iteração posterior k do algoritmo o nó N^j seja o de menor limite inferior, isto é, que no passo 1 tenhamos $N^{k*} = N^j$. É óbvio que para fazermos a ramificação num arco g do nó, é necessário que $x_g^{k*} \in (\ell_g^{k*}, u_g^{k*})$, o que como vimos não está garantido para todos os arcos.

Vamos então enunciar o seguinte lema:

Lema 5.4.1

Suponhamos que em uma certa etapa k o nó N^{k*} é tal que $LI(N^{k*}) < LS_f^{2k}$.

Então para o arco g tal que: $f_g(x_g^{k*}) - \psi_g^{k*}(x_g^{k*}) =$

$$= \max_i \{f_i(x_i^{k*}) - \psi_i^{k*}(x_i^{k*})\}$$

temos: $x_g^{k*} \in (\ell_g^{k*}, u_g^{k*})$

Prova

Como a função é côncava e pela construção de $\psi_i(x_i)$ temos:

$$\begin{cases} \psi_i^{k^*}(x_i) \leq f_i(x_i) & \text{para } x_i \in [\ell_i^{k^*},\ u_i^{k^*}] \\ \psi_i^{k^*}(x_i) \geq f_i(x_i) & \text{para } x_i \notin (\ell_i^{k^*},\ u_i^{k^*}) \end{cases}$$

mas:

$$\psi_i^{k^*}(x_i) < f_i(x_i) \text{ somente se } x_i \in (\ell_i^{k^*},\ u_i^{k^*})$$

Então basta demonstrar que existe

$$f_i(x_i^{k^*}) - \psi_i^{k^*}(x_i^{k^*}) > 0 \text{ para algum } i.$$

Mas pela escolha do nó N^{k^*} (no passo 1), temos que

$$\psi^{k^*}(x^{k^*}) = LI(N^{k^*}) < LS_f^{2k} = \min_{j=0,1\ldots k^*,\ldots 2k} \{f(x^j)\} \leq f(x^{k^*})$$

Então:

$$\psi^{k^*}(x^{k^*}) < f(x^{k^*})$$

Daí vemos que existe ao menos um arco i tal que $f_i(x_i^{k^*}) - \psi_i^{k^*}(x_i^{k^*}) > 0$.

Então:

$$f_g(x_g^{k^*}) - \psi_g^{k^*}(x_g^{k^*}) = \max_{i=1\ldots n} \{f_i(x_i^{k^*}) - \psi_i^{k^*}(x_i^{k^*})\} > 0$$

e isso só é possível se $x_g^{k^*} \in (\ell_g^{k^*},\ u_g^{k^*})$ ▩

Garantimos então que com a escolha apropriada do arco g a ser ramificado no passo 2, x_g^{k*} não estará fora do intervalo (ℓ_g^{k*}, u_g^{k*}) e então sempre será possível fazer a ramificação.

Resta então no passo 3 calcularmos LS_f^{2k} e LS_x^{2k}.

Pela definição de LS_f^k sabemos que:

$$LS_f^k = \min_{j=0,1\ldots 2k} \{f(x^j)\}$$

que corresponde à solução LS_x^k, isto é:

$$LS_f^k = f(LS_x^k)$$

Chamamos atenção para o fato de que pela própria definição de LS_f^k nós só precisamos calcular LS_f^{2k} e LS_x^{2k} e não LS_f^{2k-1} e LS_x^{2k-1}. Isto é óbvio, levando-se em consideração que primeiro calculamos a solução de P^{2k-1} e em seguida P^{2k}.

A convergência do algoritmo Branch and Bound aplicado a funções côncavas pode ser provada. Aconselhamos o leitor interessado na demonstração da convergência a consultar Soland [22].

CAPÍTULO 6

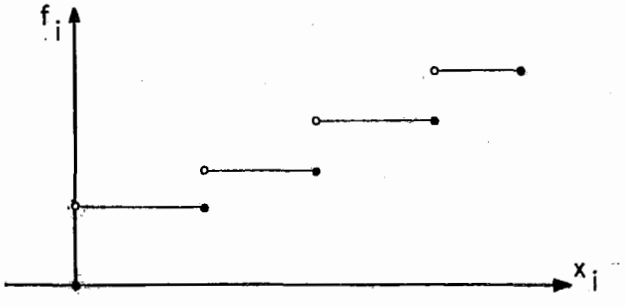
APLICAÇÃO DO B&B A UMA REDE DE FLUXO CAPACITADA COM F.O. DO TIPO "ESCADA"

6.1 - INTRODUÇÃO

Consideremos a mesma generalização proposta no item 5.1.

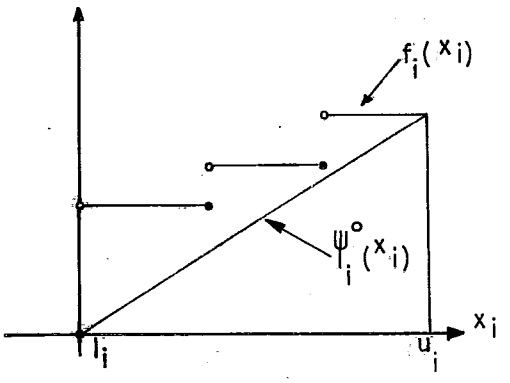
$$\text{Problema P} \begin{cases} \text{Min } f(x) = \sum_{i=1}^{n} f_i(x_i) \\ \text{s.a.} \\ \quad x \in G \equiv \{x \mid Ax = b\} \\ \quad x \in C \equiv \{x \mid \ell \leq x \leq u\} \end{cases}$$

sendo que consideraremos f_i como sendo funções do tipo "escada". Isto ocorre quando temos armazéns modulares e as funções assumem o formato:



Cada patamar se refere a um módulo do armazém. Poderemos ter um caso analogo para os custos de transporte, se considerarmos o custo da aquisição de uma frota de caminhões por exemplo.

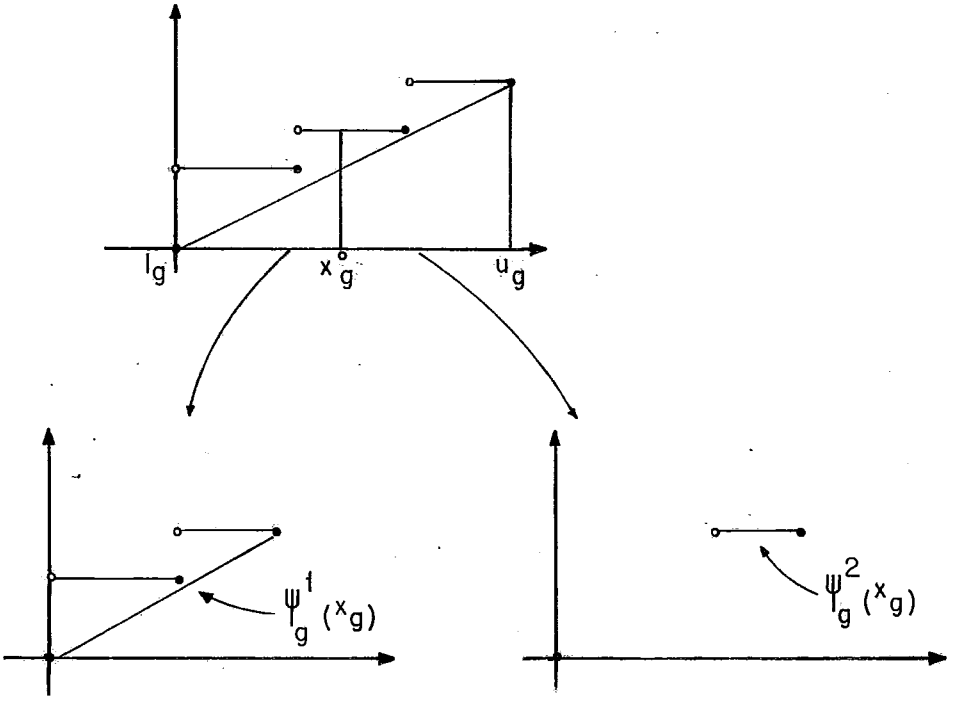
A linearização das funções $f_i(x_i)$ será feita da seguinte forma:



Resolvemos então o problema linear $P^0 \equiv$

$$\begin{cases} \text{Min } \psi^0(x) = \sum_{i=1}^{n} \psi_i^0(x_i) \\ \text{s.a. } Ax = b \\ \ell \leq x \leq u \end{cases}$$

Denominando de x^0 a solução do problema linear P^0, com algum componente $\ell_g < x_g^0 < u_g$, podemos fazer a separação do problema em dois:



Veremos mais adiante uma definição mais precisa das funções $f_i(x_i)$, visando basicamente garantir que em cada iteração j, $\psi_i^j(x_i) \leq f_i(x_i)\ \forall x_i \in [\ell_i^j, u_i^j]$. Além disso veremos também com precisão como é feita a divisão do retângulo C.

No Capítulo 5, trabalhamos com funções $f_i(x_i)$ côncavas o que nos permitiu garantir que a solução ótima do problema se encontrava num vértice de $G \cap C$. Além disso trabalhamos a cada iteração com $x \in G \cap C$ ao invés de $x \in G \cap C^k$. Assim, tinhamos o mesmo conjunto de restrições para todos os problemas, variando somente as linearizações feitas para a função-objetivo, implicando em um estudo de análise de sensibilidade ou pós-otimização. De uma maneira geral podemos dizer que no problema aqui examinado, relaxamos a condição de concavidade de $f_i(x_i)$, exigindo meramente a condição mais fraca de $f_i(x_i)$ ser côncava por partes ou mais particularmente $f_i(x_i)$ ser linear por partes. Neste caso não podemos mais garantir que a solução ótima seja vértice de $G \cap C$. Também em cada problema P^k precisamos trabalhar com o conjunto C^k, o que geometricamente equivale à criação de novos vértices. Temos portanto agora para cada problema não só a alteração dos coeficientes das variáveis da função objetivo, mas também alteramos os termos independentes das restrições. Isto no entanto não nos causa maiores problemas pelo fato de trabalharmos em cada problema P^k com um algoritmo primal-dual como é o caso do out-of-kilter.

6.2 - ADAPTAÇÃO DO ALGORITMO PARA FUNÇÕES ESCADA

Seja a função escada $f_i(x_i)$ definida da seguinte maneira para $\ell_i \leq x_i \leq u_i$

$$f_i(x_i)\begin{cases} k_i^1 & x_i = \lambda_i^1 = \ell_i \\ k_i^2 & \lambda_i^1 < x_i \leq \lambda_i^2 \\ \vdots & \\ k_i^t & \lambda_i^{t-1} < x_i \leq \lambda_i^t = u_i \end{cases} \qquad (1)$$

tal que:

$$\frac{k_i^j - k_i^{j-1}}{\lambda_i^j - \lambda_i^{j-1}} \geq \frac{k_i^{j+1} - k_i^j}{\lambda_i^{j+1} - \lambda_i^j} \qquad (2)$$

isto é, o quociente da "altura" pela "largura" do degrau ou patamar i, decresce a medida que aumenta i.

Esta condição faz com que a função escada tenha um "aclive" decrescente, lhe dando uma "aparência de concavidade". Podemos verificar que unindo os pontos $f(\lambda_i^1)$, $f(\lambda_i^2)$,...,$f(\lambda_i^t)$ com segmentos de reta, obteremos uma função côncava.

Para efeito de simplificar o nosso raciocínio, assumiremos que todas as funções $f_i(x_i)$ tem o mesmo número t de patamares, uma generalização que não nos trará maiores problemas.

Fazendo a linearização das funções $f_i(x_i)$ através das

funções $\psi_i(x_i)$ tais que $\psi_i(\ell_i) = f_i(\ell_i)$ e $\psi_i(u_i) = f_i(u_i)$ é fácil verificar que $\psi_i(x_i) \leq f_i(x_i)$ $\forall x_i \in [\ell_i, u_i]$. Teremos então as etapas de Inicialização e de Avaliação análogas ao algoritmo para funções côncavas. Resta precisarmos um pouco mais a etapa de separação (Branching) e mostrarmos como calcular $\psi_i^k(x_i)$.

Como já vimos no Capítulo 5, de acordo com o passo 2 do algoritmo, quando um problema N^{k*} é escolhido para ser ramificado, temos $LI(N^{k*}) = \psi^{k*}(x^{k*}) < LS_f^{2k} = \min(f(x^1), f(x^2) \ldots, f(x^{k*}), \ldots f(x^{2k})$. Logo temos $\psi^{k*}(x^{k*}) < f(x^{k*})$, o que nos permite garantir que existe algum valor i tal que $f_i(x_i^{k*}) - \psi_i^{k*}(x_i^{k*}) > 0$.

Seja $f_g(x_g^{k*}) - \psi_g^{k*}(x_g^{k*}) = \max_{i=1\ldots n} (f_i(x_i^{k*}) - \psi_i^{k*}(x_i^{k*}))$. Então o intervalo $[\ell_g^{k*}, u_g^{k*}]$ será subdividido em dois intervalos $[\ell_g^{2k-1}, u_g^{2k-1}]$ e $[\ell_g^{2k}, u_g^{2k}]$ onde $\ell_g^{2k-1} = \ell_g^{k*}$, $u_g^{2k} = u_g^{k*}$ e $\ell_g^{k*} \leq u_g^{2k-1} = \ell_g^{2k} \leq u_g^{k*}$.

Na prática, como veremos, u_g^{2k-1} difinirá de ℓ_g^{2k} de um valor ε correspondente à precisão do computador, o que nos garante não termos eliminado nenhuma solução viável, de maneira que $C^{2k-1} \cup C^{2k} = C^{k*}$, pois, para os demais valores $i \neq g$, manteremos $\ell_i^{2k-1} = \ell_i^{2k} = \ell_i^{k*}$ e $u_i^{2k-1} = u_i^{2k} = u_i^{k*}$.

É importante ressaltar que nos problemas com função côncava as subdivisões dos intervalos para efeito de restrições não precisam ser efetivamente realizados. Agora porém, isto terá que ser feito, de modo que um dos problemas P^{2k-1} ou P^{2k}

gerados poderá não ter solução por ter o espaço de soluções viáveis vazio, mas certamente um deles pelo menos terá solução, pois não eliminamos nenhuma solução viável do problema P^{k*}.

A subdivisão do intervalo $[\ell_g^{k*}, u_g^{k*}]$, pode ser feita de várias maneiras, cabendo ressaltar que:

1) Deve permitir uma fácil definição das funções $\psi_g^{2k-1}(x_g)$ e $\psi_g^{2k}(x_g)$

2) Deve ser tal que geremos um número mínimo de subdivisões, isto é, um número mínimo de problemas.

3) Deve ser tal que geremos sempre problemas distintos, o que é importante para garantir a convergência do algoritmo.

Veremos a seguir como fizemos a subdivisão do intervalo $[\ell_g^{k*}, u_g^{k*}]$. Os três casos (a), (b) e (c), discriminam o número de patamares contidos no intervalo.

Para cada caso, apresentamos exemplos ilustrativos, indicando com uma linha tracejada as linearizações efetuadas. Nos casos (a) e (b) não indicamos x_g^{k*}, pois a maneira de realizar a subdivisão independe de onde cai x_g^{k*}. Para os exemplos sempre estamos supondo que $k_g^1 = 0$ para $x_g = \lambda_g^1 = \ell_g = 0$.

Seja $\ell_g^{k*} = \lambda_g^r + e\ \varepsilon \leq \lambda_g^{q-1} + e\ \varepsilon \leq x_g^{k*} \leq \lambda_g^q \leq \lambda_g^s = u_g^{k*}$, onde $e = \min(1, r-1)$ e ε é um valor muito pequeno cor-

respondente a precisão do computador. Através de ε podemos gerar intervalos fechados e evitar a superposição dos intervalos. Evidentemente temos $1 \leq r < q \leq s \leq t$ onde t é o número de patamares. Através de q podemos determinar qual o patamar de $f_g(x_g)$ que corresponde a x_g^{k*}.

Temos os seguintes casos:

(a) $s - r = 1$

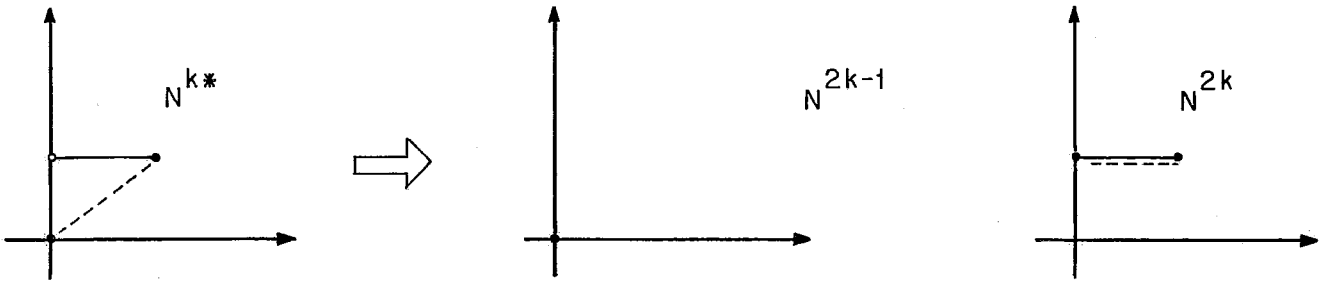
Para este caso, basta considerarmos $r = 1$. Se tivessemos $r > 1$, estariamos de posse de um patamar somente onde não nos interessa subdividir o arco.

Então:

$$N^{2k-1} \begin{cases} x_g = \lambda_g^1 \\ \psi_g^{2k-1}(\lambda_g^1) = k_g^1 \end{cases} \tag{3}$$

$$N^{2k} \begin{cases} \lambda_g^1 + \varepsilon \leq x_g \leq \lambda_g^2 \\ \psi_g^{2k}(\lambda_g^1 + \varepsilon) = \psi_g^{2k}(\lambda_g^2) = k_g^2 \end{cases} \tag{4}$$

Exemplo:



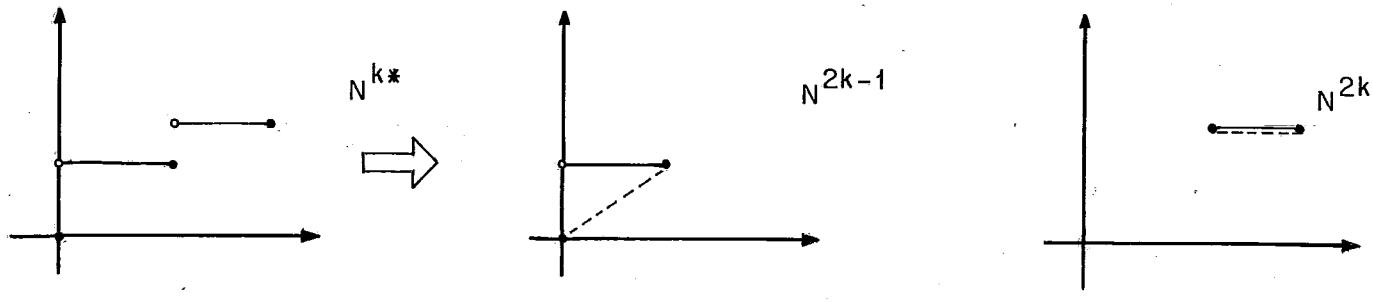
(b) $s - r = 2$

$$N^{2k-1}\begin{cases} \lambda_g^r + e\ \varepsilon \leq x_g \leq \lambda_g^{r+1} \\ \psi_g^{2k-1}\ (\lambda_g^r + e\ \varepsilon) = k_g^{r+e} \\ \psi_g^{2k-1}\ (\lambda_g^{r+1}) = k_g^{r+1} \end{cases} \tag{5}$$

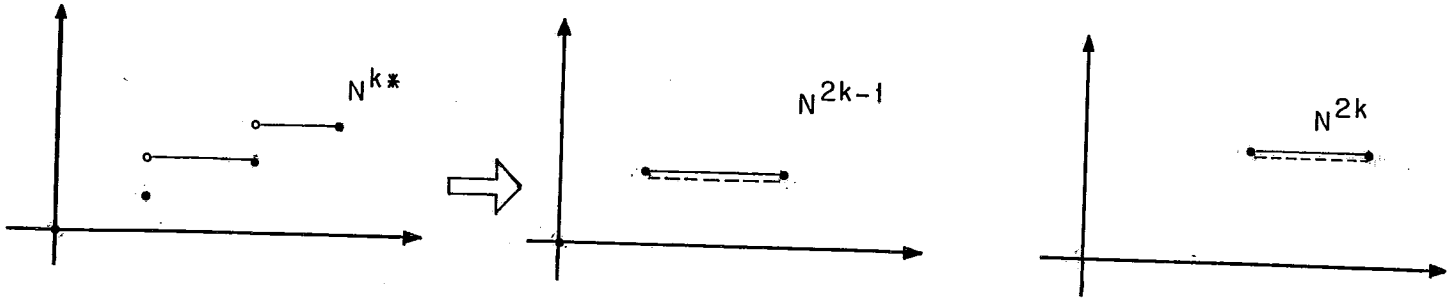
$$N^{2k}\begin{cases} \lambda_g^{s-1} + \varepsilon \leq x_g \leq \lambda_g^s \\ \psi_g^{2k}\ (\lambda_g^{s-1} + \varepsilon) = \psi_g^{2k}(\lambda_g^s) = k_g^s \end{cases} \tag{6}$$

Exemplo:

1. Com $r = 1$



2. com $r \neq 1$



(c) $s - r > 2$

Calcule $h = \min(q-r, r-1) \cdot \max(2-q+r, 0)$ assim como $i = \min(s-q, 1)$ e $j = \min(s-q, 2)$.

A subdivisão de $[\ell_g^{k*}, u_g^{k*}]$ dá origem aos seguintes nós:

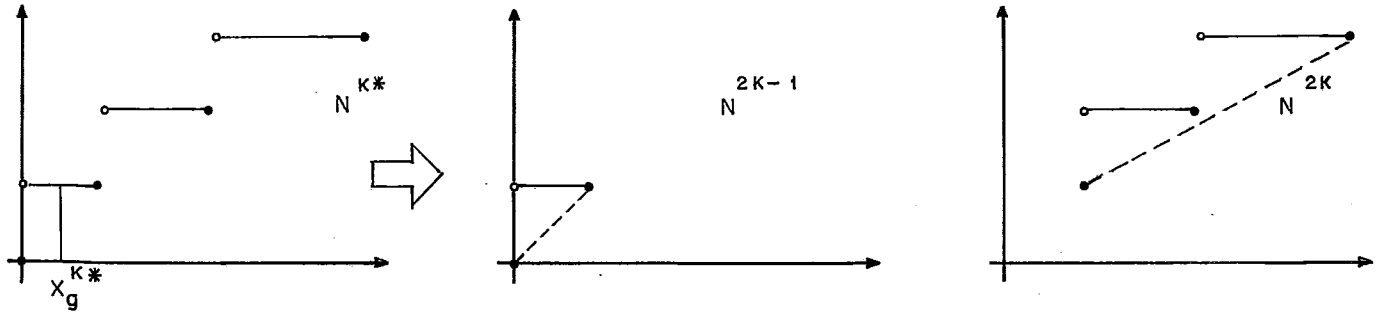
$$N^{2k-1} \begin{cases} \lambda_g^r + e\,\varepsilon \leq x_g \leq \lambda_g^{q+i-1} \\ \psi_g^{2k-1}(\lambda_g^r + e\,\varepsilon) = k_g^{r+h} \\ \psi_g^{2k-1}(\lambda_g^{q+i-1}) = k_g^{q+i-1} \end{cases} \qquad (7)$$

$$N^{2k}\begin{cases} \lambda_g^{q+i-1} + \varepsilon \leq x_g \leq \lambda_g^{s} \\ \psi_g^{2k}\ (\lambda_g^{q+i-1} + \varepsilon) = k_g^{q+2i-j} \\ \psi_g^{2k}\ (\lambda_g^{s}) = k_g^{s} \end{cases} \qquad (8)$$

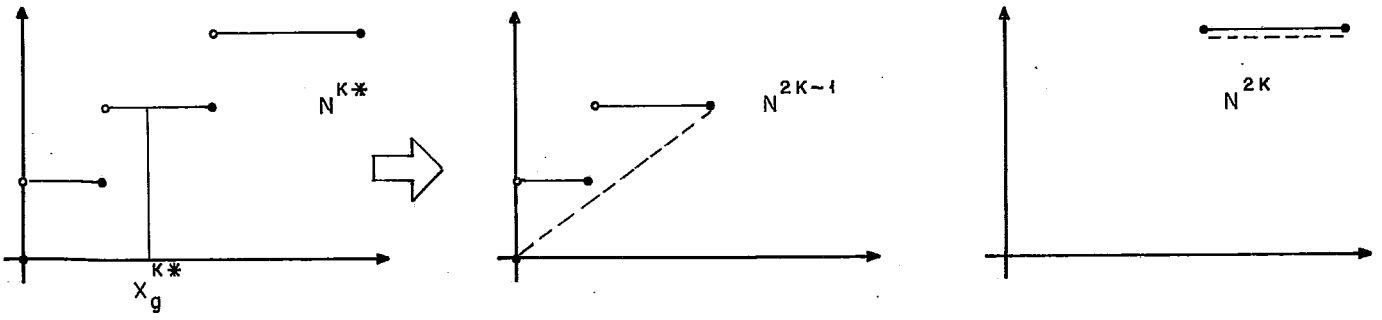
Exemplo:

1. Com r = 1

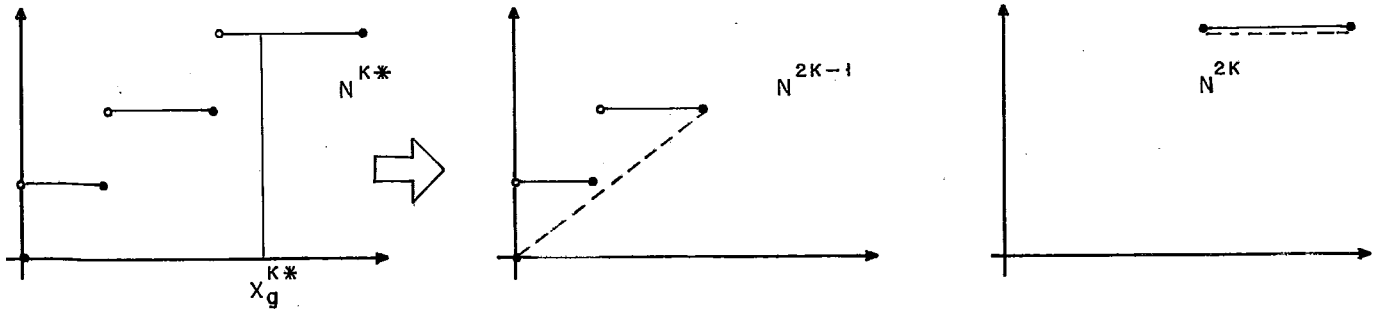
1.1



1.2

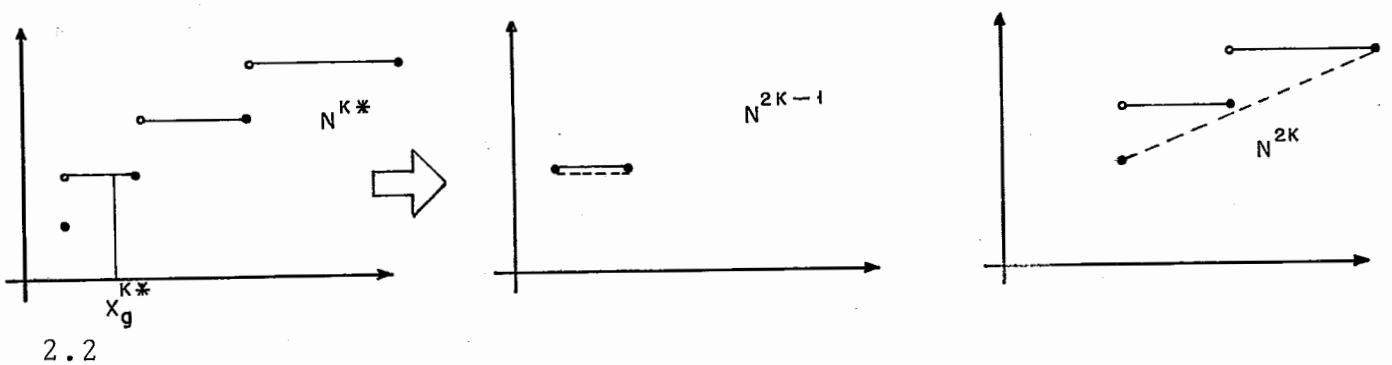


1.3

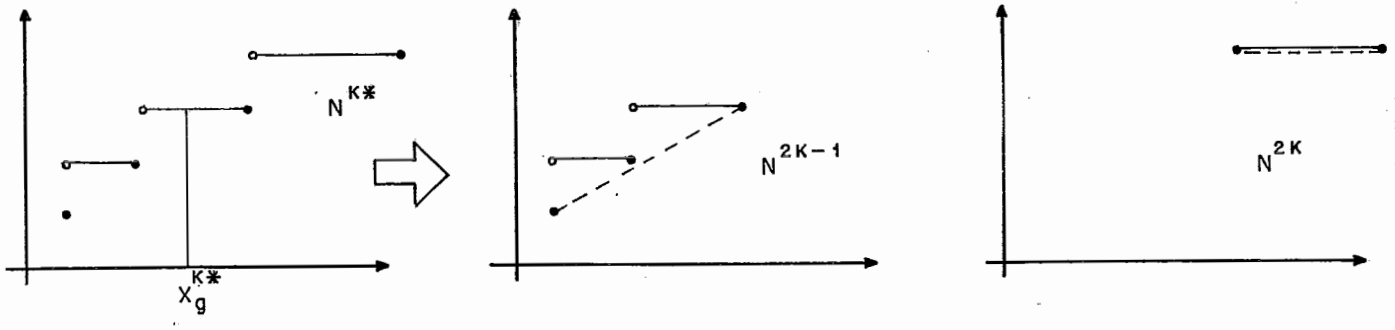


2. Com $r \neq 1$

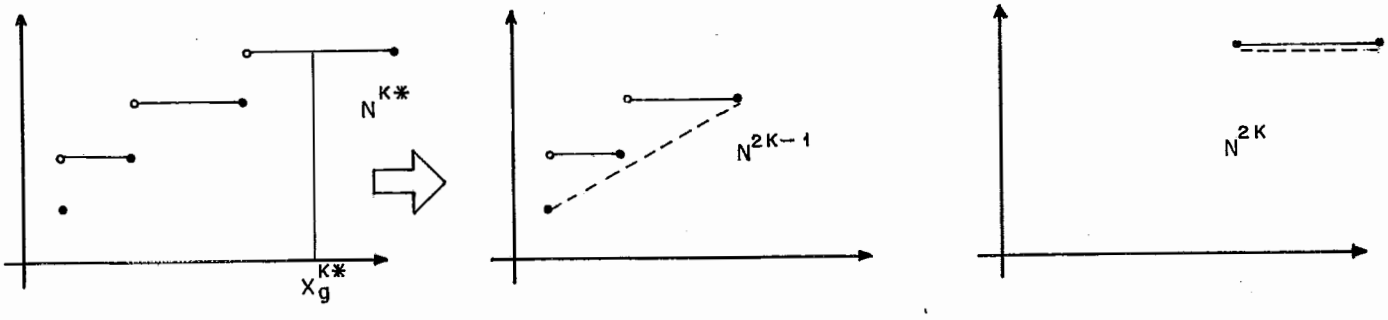
2.1



2.2



2.3



Como vimos no início do item 6.1, dado um nó N^k, o que nos garante a obtenção do limite inferior $LI(N^k) = \psi^k(x^k)$ é o fato de $\forall i$ termos $\psi_i^k(x_i) \leq f_i(x_i)$ para $x_i \in [\ell_i^k, u_i^k]$. Agora, uma vez tendo definido precisamente a função $\psi_i^k(x_i)$, este fato pode ser demonstrado. É o que faremos através do Teorema a seguir.

Teorema 6.2.1

Dadas as funções $f_i(x_i)$, definidas segundo (1) e (2) e dado um problema N^k e funções $\psi_i^k(x_i)$ definidos segundo as regras (a), (b) e (c) temos $\psi_i^k(x_i) \leq f_i(x_i)$ para $\ell_i^k \leq x_i \leq u_i^k$, $\forall i$.

Demonstração

Seja um problema N^k e seja o intervalo $[\ell_i^k , u_i^k]$ onde:

$$\ell_i^k = \lambda_i^r + e\ \varepsilon \leq x_i \leq \lambda_i^s = u_i^k \tag{9}$$

Pelo tipo de subdivisão feita em (a), (b) e (c) vemos que sempre que $s-r = 1$ e $r \neq 1$, ou então, $s=r=1$, temos $\psi_i^k(x_i) = f_i(x_i)$, estando portanto demonstrado o teorema.

Vejamos agora os casos em que $s-r \geq 2$ ou então $s-r = 1$ e $r = 1$. Para estes casos temos, generalizando e considerando $\varepsilon = 0$ (na prática podemos fazer ε tão pequeno quanto se queira) $\psi_i^k(\lambda_i^r) = k_i^r$ e $\psi_i^k(\lambda_i^s) = k_i^s$. Utilizando a equação da reta que passa por dois pontos, e considerando sem perda da generalidade um valor $\hat{x}_i$ tal que:

$$\lambda_i^r \leq \lambda_i^{q-1} < \hat{x}_i \leq \lambda_i^q \leq \lambda_i^s \tag{10}$$

temos que:

$$\frac{\psi_i^k(\hat{x}_i) - k_i^r}{\hat{x}_i - \lambda_i^r} = \frac{k_i^s - k_i^r}{\lambda_i^s - \lambda_i^r} \tag{11}$$

o que nos permite determinar $\psi_i^k(\hat{x}_i)$. Evidentemente de (10) e (1) segue $f_i(\hat{x}_i) = k_i^q$.

Por outro lado, de acordo com (2) temos:

$$\frac{k_i^{r+1} - k_i^r}{\lambda_i^{r+1} - \lambda_i^r} \geq \frac{k_i^{r+2} - k_i^{r+1}}{\lambda_i^{r+2} - \lambda_i^{r+1}} \tag{12}$$

Sabemos que se $\frac{A}{B} \geq \frac{C}{D}$ então temos $\frac{A}{B} \geq \frac{A+C}{B+D}$. Basta ver que $\frac{A+C}{B+D} \leq \frac{A+\frac{AD}{B}}{B+D} = \frac{A}{B}$. Aplicando este raciocínio a (12)

$$\frac{k_i^{r+1} - k_i^r}{\lambda_i^{r+1} - \lambda_i^r} \geq \frac{k_i^{r+2} - k_i^r}{\lambda_i^{r+2} - \lambda_i^r}$$

Generalizando, podemos escrever:

$$\frac{k_i^q - k_i^r}{\lambda_i^q - \lambda_i^r} \geq \frac{k_i^s - k_i^r}{\lambda_i^s - \lambda_i^r} \tag{13}$$

para $r < q \leq s$. Aplicando (13) em (11) e lembrando que $\hat{x}_i \leq \lambda_i^q$ temos:

$$\frac{\psi_i^k(\hat{x}_i) - k_i^r}{\lambda_i^q - \lambda_i^r} \leq \frac{\psi_i^k(\hat{x}_i) - k_i^r}{\hat{x}_i - \lambda_i^r} = \frac{k_i^s - k_i^r}{\lambda_i^s - \lambda_i^r} \leq \frac{k_i^q - k_i^r}{\lambda_i^q - \lambda_i^r}$$

de onde segue $\psi_i^k(\hat{x}_i) \leq k_i^q = f_i(\hat{x}_i)$. ▩

6.3 - CONVERGÊNCIA E COMPLEXIDADE DO ALGORITMO

Seja $f_i(x_i)$ definido segundo (1). Imaginando que o ponto $f_i(x_i) = k_i^1$ é um patamar de "largura" nula, podemos dizer que $f_i(x_i)$ é constituído de t patamares. Denominemos $f_i(x_i)=k_i^j$ de patamar j da função $f_i(x_i)$.

Dado um problema N^p, a cada intervalo $\ell_i^p = \lambda_i^r + \varepsilon \leq x_i \leq \lambda_i^s = u_i^p$ podemos associar um conjunto $F_i^p = \{r+1, r+2, \ldots, s\}$ constituido pelos patamares correspondentes a este intervalo.

Sinteticamente podemos então dizer que, o que o algoritmo faz é, partindo de um problema N^0, onde $F_i^0 = \{1, 2, \ldots, t\}$, subdividir os diversos intervalos, de maneira que, numa etapa k do algoritmo, se for separado o problema N^p nos subproblemas N^{2k} e N^{2k-1}, através da subdivisão do intervalo $[\ell_i^p, u_i^p]$, temos $F_i^p = F_i^{2k} \cup F_i^{2k-1}$. O máximo de subdivisões do intervalo $[\ell_i^0, u_i^0]$ que pode ser feito, é tal que tenhamos, no final conjuntos F_i^j com um único patamar, isto é, $|F_i^j| = 1$. O máximo número de subdivisões feitas para um intervalo $[\ell_i^0, u_i^0]$, que corresponde ao máximo número de problemas gerados através da subdivisão deste intervalo, é ilustrado sem perda de generalidade, pela Figura 6.3.1 onde constam os conjuntos F_i^j:

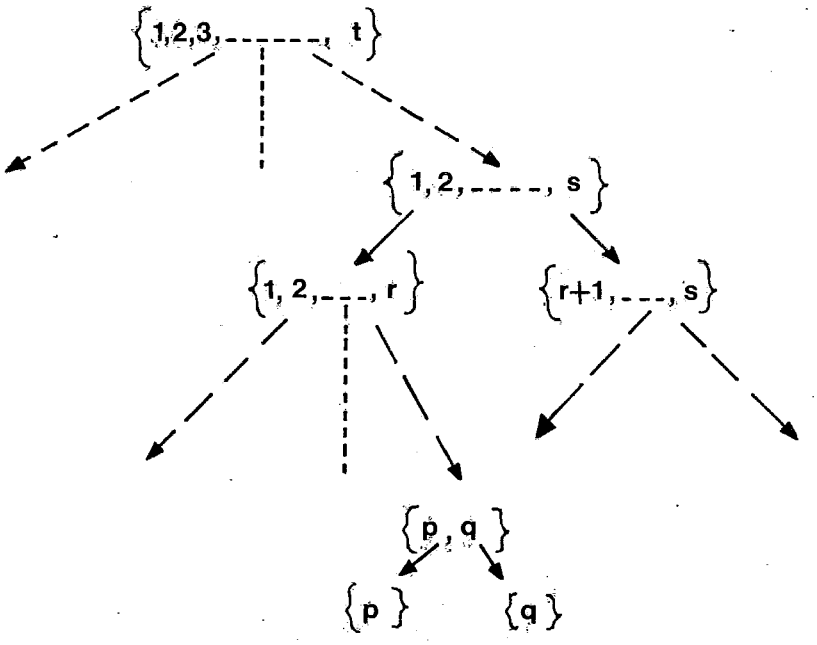


Figura 6.3.1

Por indução finita é possível mostrar que se $|F_i^j| = k$, isto é, F_i^j contém k elementos (existem k patamares), então o máximo número de subdivisões dá origem a (2k-1) problemas. O passo inicial da indução é para $|F_i^j| = 2$ (caso do conjunto {p, q} da Figura 6.3.1). Neste caso damos origem a três problemas. Seja agora $|F_i^j| = s$, e suponhamos verdadeira a hipótese para conjuntos que contenham $g < s$ elementos. Suponhamos agora, sem perda de generalidade, o conjunto F_i^j subdividido em dois subconjuntos respectivamente de r e s-r elementos. Pela hipótese da indução cada um destes subproblemas dá origem respectivamente a (2r-1) e (2s-2r-1) problemas. Num total temos portanto 2s-2 problemas. Levando ainda em conta F_i^j que evidentemente está associado ao problema N^j temos um total de 2s-1 problemas, o que prova a hipótese.

Assim, o intervalo $[\ell_i^0, u_i^0]$ associado ao problema N^0, para o qual $f_i(x_i)$ tem t patamares, dá origem, feita a subdivisão do intervalo segundo as regras (a), (b) e (c) no máximo a (2t-1) problemas distintos. Supondo n variáveis $x_1, x_2, \ldots, x_n$ as quais estão associadas funções escada $f_i(x_i)$ com t patamares temos um total de, no máximo, $(2t-1)^n$ problemas distintos.

Assim, fica provada a convergência do algoritmo em um máximo de $(2t-1)^n$ resoluções, pois, ao gerarmos os $(2t-1)^n$ problemas obrigatoriamente o algoritmo pára. Basta lembrar que no final do processo teríamos para todos os problemas N^k não separados (todos os problemas intermediários) $f_i(x_i) = \psi_i^k(x_i)$, o que garante o atendimento à regra de parada.

O estudo da convergência do algoritmo, permite também fazer uma afirmativa sobre a sua complexidade com base na consideração do pior caso possível. A complexidade é da ordem $(2t-1)^n$ onde n são o número de variáveis as quais estão associadas funções escada com t patamares. É fácil ver, que este número é bastante elevado. Para n = 5 e t = 3 temos 3.125 problemas. Fazendo n = 10 e t = 3 temos $9,7 \times 10^6$ problemas gerados.

Cabe no entanto lembrar que estes números referem-se ao pior caso possível, nada permitindo dizer sobre os tempos realmente gastos pelo algoritmo (também o Simplex, numa análise do pior caso possível, tem um comportamento muito ruim; na prática, no entanto, ele tem demonstrado grande eficiência).

Na prática dificilmente o algoritmo gerará todos os problemas possíveis, isto é, fará todas as subdivisões dos intervalos, principalmente se consideramos a tolerância ε do algoritmo. Isto ainda é acentuado pelo fato dos custos de transporte num problema de localização real poderem ser assumidos como lineares, sendo somente os custos de armazenagem expressos por funções escada. Assim, o erro relativo à linearização destas últimas é pequeno considerando-se a massa dos custos de transporte.

CAPÍTULO 7

O PROGRAMA

7.1 - INTRODUÇÃO

Para resolver os problemas apresentados anteriormente, desenvolvemos um programa em ALGOL (vide Anexo I), fazendo uma série de testes para verificar a sua confiabilidade e eficiência.

Gostaríamos de ressaltar que tivemos e continuaremos tendo a preocupação de tornar o programa o mais aberto possível, ficando o nosso intento frustado se o mesmo for usado como um "pacote" ou "caixa preta", podento tal atitude levar a erros e imperfeições. Além disso colocamos dessa forma o nosso trabalho ao dispor dos pesquisadores, que poderão com precisão e segurança fazer modificações nas rotinas, nas variáveis e na entrada e saída de dados, caso sintam necessidade.

Uma grande preocupação nossa foi também a de colocar comentários nos pontos chave do programa de modo a que o seu entendimento ficasse mais claro.

Fazemos então a seguir, uma descrição do programa dando detalhes de linguagem, equipamento usado, rotinas, formato de variáveis, etc... Mostramos com detalhes como é feita a entrada de dados e como são os relatórios impressos. Nos capítulos se

guintes falamos sobre os testes efetuados e a performance.

É importante frizar também, que o programa foi desenvolvido para resolver os problemas apresentados nos Capítulos 5 e 6, sendo que certamente outros problemas que se constituiam numa combinação dos dois ou que sigam as mesmas regras de separação e avaliação, poderão ser resolvidos, com nenhuma ou quase nenhuma modificação.

A única limitação apresentada no programa é quanto ao número de iterações que não pode ultrapassar 8000, isto devido a alguns arrays que depois desse número de iterações ultrapassam o tamanho máximo permitido pelo Algol.

7.2 - DESCRIÇÃO GERAL DO PROGRAMA

7.2.1 - Principais Características

O programa foi escrito em Algol, tendo sido compilado, testado e implementado no computador B-6700 do Núcleo de Computação Eletrônica da UFRJ.

Todas as variáveis foram definidas com precisão simples.

Para as listas mais longas, a estrutura de dados usada foi a de listas linkadas. A principal razão se deve ao fato de que com este tipo de lista consegue-se um bom aproveitamen

to do espaço da memória. Um fato marcante do nosso algoritmo é o crescimento da árvore gerada pelo Branch and Bound, criando-se novos nós na etapa de branching, à medida em que se vai fechando nós. Estes nós fechados, tem seu espaço reocupado graças ao tipo de listas que usamos.

O programa tem ao todo cinco versões que representam a evolução na sua confecção. Em cada etapa dessa evolução procuramos dar ênfase a um determinado detalhe e cada versão então tem uma característica marcante como veremos a seguir:

A versão 1 é a versão básica onde tão somente tivemos a preocupação de resolver problemas com funções côncavas. Não tivemos a preocupação de economizar tempo ou memória, de fazer comentários no programa fonte, etc...

Na versão 2 tivemos a preocupação de diminuir o tempo de execução, usando comandos potentes do Algol, como o Listlookup, otimizando as rotinas, utilizando mais variáveis globais, variáveis reais, arrays de somente uma dimensão etc... o que nos reduziu o tempo de execução para aproximadamente 1/5 do tempo usado pela versão 1.

Na versão 3 tivemos a preocupação de poder resolver problemas maiores, de grande porte, e para isso tivemos que guardar determinadas listas em memória auxiliar (disco), não se tendo em consequência nenhuma limitação com relação ao número de variáveis e de restrições do problema a ser resolvido.

A versão 4 tem como característica principal a resolução de problemas com funções do tipo "escada" e a aceitação de Índices de Rotação para os armazéns.

A versão 5 é a versão final, onde nós tivemos a preocupação de melhorar o aspecto das saídas impressas, racionalizar a entrada de dados, colocar comentários e melhorar a apresentação do programa fonte.

O número da versão aparece impresso nos cabeçalhos dos relatórios, devendo ser modificado na rotina de impressão, caso se façam modificações na versão 5.

7.2.2 - Blocos e Procedures

A estrutura dos blocos e procedures usados no programa é a seguinte:

1. Bloco MESTRE
   - 1.1 Rotina de Leitura do Cartão Mestre
   - 1.2 Bloco Principal
     - 1.2.1 Procedure LER
       - 1.2.1.1 Procedure CRIA
       - 1.2.1.2 Rotina de Leitura dos Dados
     - 1.2.2 Procedure KILTER
       - 1.2.2.1 Procedure STACALC
       - 1.2.2.2 Procedure ROTULAT
       - 1.2.2.3 Rotina Corpo do Out-of-Kilter

1.2.3 Procedure CUSTO

1.2.4 Procedure CALCLIM

1.2.5 Procedure OBTER

1.2.6 Procedure GUARDAR

1.2.7 Procedure INICIALIZAR

1.2.8 Procedure SEPARAR

1.2.9 Procedure AVALIAR

1.2.10Procedure IMPRIMIR

1.2.11Rotina Principal

Na realidade bloco e procedure tem aqui significado quase idêntico e só fizemos a distinção para que fosse fácil a sua localização dentro do programa. As procedures são encabeçadas pelo verbo PROCEDURE e seu nome e na maioria das vezes começam em uma nova página. Os dois blocos são fáceis de localizar pois o bloco Mestre é por assim dizer o próprio programa e o bloco Principal tem um comentário destacado, identificando-o.

Daremos a seguir uma descrição de cada bloco e procedure, ressaltando que não se pretende que a partir dessa descrição se entenda o funcionamento do programa, mas tão somente, que a pessoa que for se aprofundar nos detalhes de procedures, comandos, etc. tenha uma ajuda no seu trabalho.

1. BLOCO MESTRE

As variáveis definidas no início do bloco servem somente para guardar dados lidos do cartão mestre, os valores

calculados a partir destes dados e as horas de início e término que nos interessam para cálculo dos tempos de execução.

A seguir temos a rotina de leitura do cartão Mestre, cuja finalidade é ler o cartão Mestre, criticando os seus dados. Caso haja alguma inconsistência imprimimos uma mensagem de erro, não executando o Bloco Principal.

2. BLOCO PRINCIPAL

Uma vez lidos os parâmetros de execução do cartão Mestre, que nos dão uma série de dados para dimensionamento dos arrays, podemos executar o Bloco Principal.

No início do Bloco são definidas todas as variáveis e arrays comuns às procedures internas.

Aparecem depois todas as suas procedures internas que serão descritas adiante.

Por último vem a Rotina Principal que executa as procedures LER que lê os cartões de dados INICIALIZAR que constitui o Passo 0 do nosso algoritmo, o Passo 1 verificando se a solução que temos é ótima e as procedures SEPARAR e AVALIAR que correspondem respectivamente aos Passos 2 e 3 do algoritmo. Além disso executa a procedure IMPRIMIR, que ao final, imprime os relatórios com a solução do problema. Essa rotina como vemos, constitui-se na espinha dorsal do programa.

3. PROCEDURE LER

Tem como finalidade básica ler e criticar os cartões de dados (cartões tipo 0, 1, 2 e 3). Tem interna a ela a Procedure CRIA que cria as listas ARCO e NPO importantes para o OUT-OF-KILTER e a rotina de leitura de dados. Nessa rotina, os cartões são lidos e criticados, sendo que a quantidade de cartões 0, 1, 2 e 3 é determinada pelos parâmetros do cartão MESTRE. Havendo alguma inconsistência nos cartões, é impressa uma mensagem de advertência, cancelando-se o programa.

4. PROCEDURE CRIA

Interna à Procedure LER, ela tem a finalidade como já vimos, de criar as listas ARCO e NPO usadas no OUT-OF-KILTER.

5. PROCEDURE KILTER

Constitui-se no algoritmo OUT-OF-KILTER usado para resolver nossos problemas de programação linear. Fizemos uma adaptação da rotina usada por Monterosso |[17]| no seu programa.

Tem duas procedures internas STACALC e ROTULAT e um corpo principal no qual são chamadas essas procedures para cálculo do estado dos arcos e a sua colocação em KILTER.

6. PROCEDURE STACALC

Interior à procedure Kilter, tem a finalidade de verificar o estado do arco K. (Como o arco, está posicionado na figura de Kilter).

7. PROCEDURE ROTULAT

Também interior a procedure Kilter faz a rotulação dos nós da rede. Tem por finalidade colocar os arcos em Kilter.

8. PROCEDURE CUSTO

Calcula o valor da função objetivo $f_i(x_i)$ num ponto x_i dado.

9. PROCEDURE CALCLIM

Tem por finalidade principal calcular os valores de LI, LSF e LSX do problema.

Além disso descobre qual o "pior" arco, isto é, qual o que tem maior diferença entre $\psi_i(x_i^{k*})$ e $f_i(x_i^{k*})$ e nos dá o fluxo nesse arco.

10. PROCEDURE OBTER

Obtém os valores das variáveis primais e duais do

"primeiro" nó, da lista de nós abertos. Estes valores podem estar armazenados em memória ou em disco, isto sendo verificado através do valor do indexador INAB que aponta para o "início" da lista de nós abertos.

Nós guardamos os valores das variáveis primais e duais para cada folha da árvore para que, ao ramificar a folha, possamos entregar à rotina KILTER numa solução inicial primal-dual viável do problema anterior. Como normalmente só tiramos um arco do estado KILTER, nós levamos bem menos tempo para resolver o nosso problema do que se entrarmos com uma solução inicial qualquer. Isso ficou demonstrado na prática onde se pode verificar a enorme diferença de tempo entre a primeira chamada da procedure KILTER quando entramos com uma solução inicial X = 0 e W = 0 e as demais chamadas.

11. PROCEDURE GUARDAR

Guarda os valores das variáveis primais X_i e duais W_i na lista de nós abertos. Esta lista tem seus 65000 primeiros campos na memória e os demais armazenados em disco. Isto se deve a uma limitação do computador que não armazena arrays que ocupem mais do que 65536 palavras. Então se o indexador I (aponta para o nó que estamos querendo inserir), contém um valor que nos faça cair fora das 65000 palavras, armazenamos os dados do nó em disco, caso contrário armazenamos os dados do nó em memória.

12. PROCEDURE INICIALIZAR

Constitui o Passo 0 do nosso algoritmo; nela inicializamos variáveis, calculamos os coeficientes angular e linear de cada função côncava linearizada, calculamos o problema P^0 linearizado, e inicializamos as listas de nós abertos e nós fechados.

13. PROCEDURE SEPARAR

Corresponde ao Passo 2 do algoritmo. Nela fechamos o "menor" nó da arborencência e abrimos dois novos nós. Fisicamente um desses novos nós vai ocupar o mesmo espaço na lista, do nó que foi fechado. Isto pode ser feito graças a estrutura de listas linkadas que usamos, já que precisamos manter a lista em ordem, isto é, o primeiro nó da lista nós queremos que seja o "menor".

A parte mais interessante da procedure se refere à subida na arborecência partindo da folha que vai ser fechada até a raiz da árvore, para a obtenção dos dados da folha. Isto se deve ao fato de não guardarmos na lista de nós abertos todos os dados das folhas, só os que mudaram ao se fechar o nó. Maiores detalhes da construção das listas de nós abertos e fechados serão vistos no item 7.2.3.

14. PROCEDURE AVALIAR

Se refere ao Passo 3 do algoritmo. Nela calculamos os dois novos problemas gerados e incluimos os dados na lista de nós abertos.

15. PROCEDURE IMPRIMIR

Tem a finalidade de imprimir os resultados do problema. São impressos o Resumo do Processamento, os dados dos Centros Produtores e Centros Consumidores e os dados de Transporte e Armazenagem com os respectivos custos. Maiores detalhes de cada relatório nós veremos no item 7.4.

7.2.3 - Variáveis e Listas

Todas as nossas variáveis foram definidas com precisão simples.

Não usamos arrays com mais do que uma dimensão. As matrizes foram guardadas em listas com uma dimensão e a localização dos itens das matrizes foi sempre calculada usando-se os comandos DEFINE. Isto se deve ao fato de que o ALGOL não é eficiente no manuseio de matrizes. Todas as variáveis e arrays (listas) do programa tem um comentário ao lado, dando a sua descrição, que acreditamos ser suficiente para o entendimento de seus significados. Daremos porém explicações mais detalhadas a respeito de alguns arrays especiais:

1. PRIPAT, PAT, NUMFUN e PRIPATAR

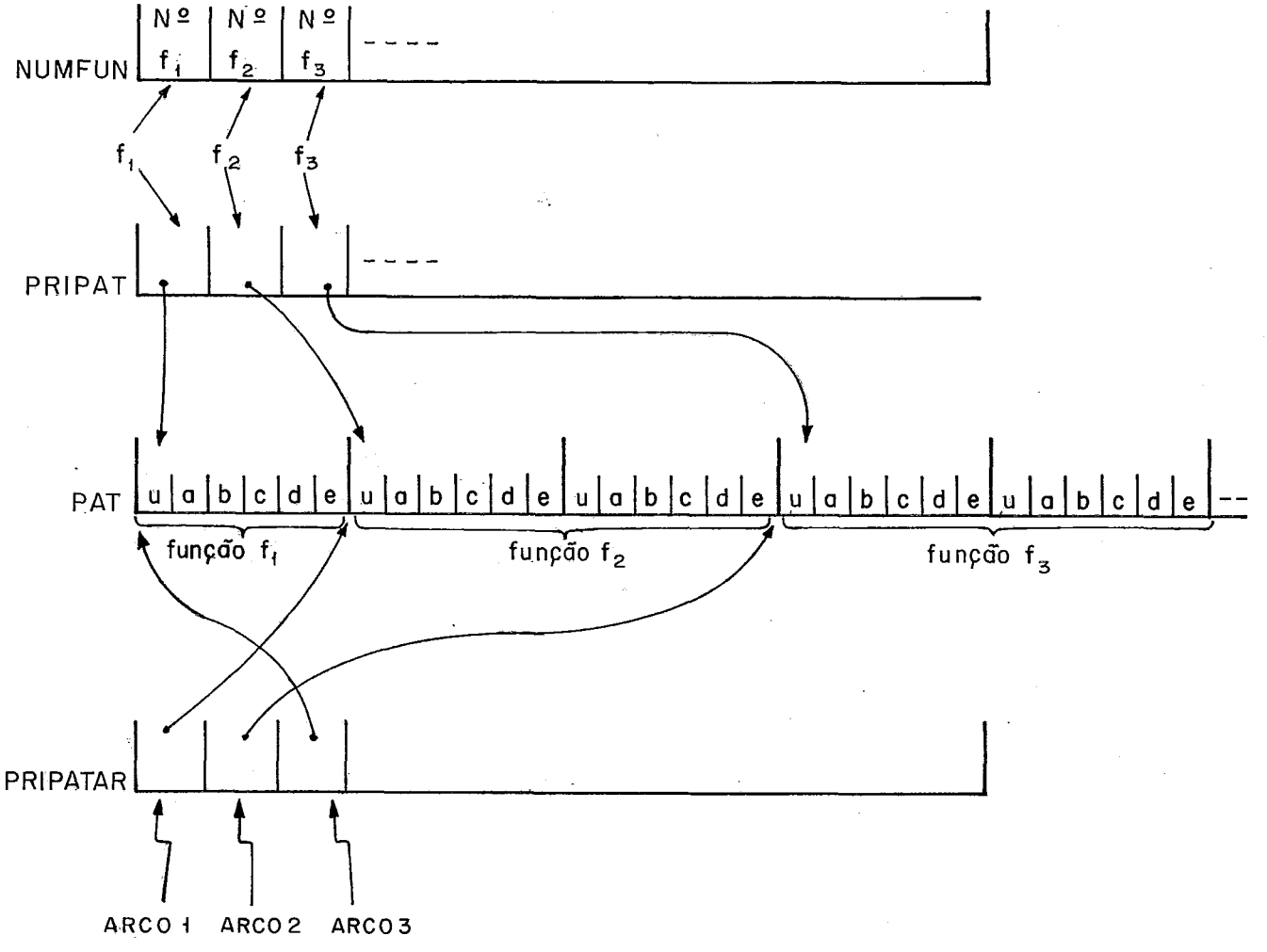
PRIPAT é um array que contém os ponteiros para o primeiro patamar de cada função no array PAT.

PAT contém os dados de todos os patamares de todas as funções. Para cada patamar contém os valores de U (limite superior do patamar)e os valores a, b, c, d, e, considerando-se que todas as funções devem ter a forma $a + bx^c + dx^e$. (Ver maiores detalhes no item 7.3 - Entrada dos Dados).

NUMFUN contém o número das funções patamar; cada função deve ser associada a um número para facilitar a sua identificação nos cartões de dados e nos relatórios de saída.

PRIPATAR contém os ponteiros para o primeiro patamar de cada arco no array PAT.

Esquematicamente podemos representar PRIPAT, PAT, NUNFUM e PRIPATAR da seguinte forma:



2. XAB e WAB

XAB contém o valor do fluxo nos nós abertos (folhas).

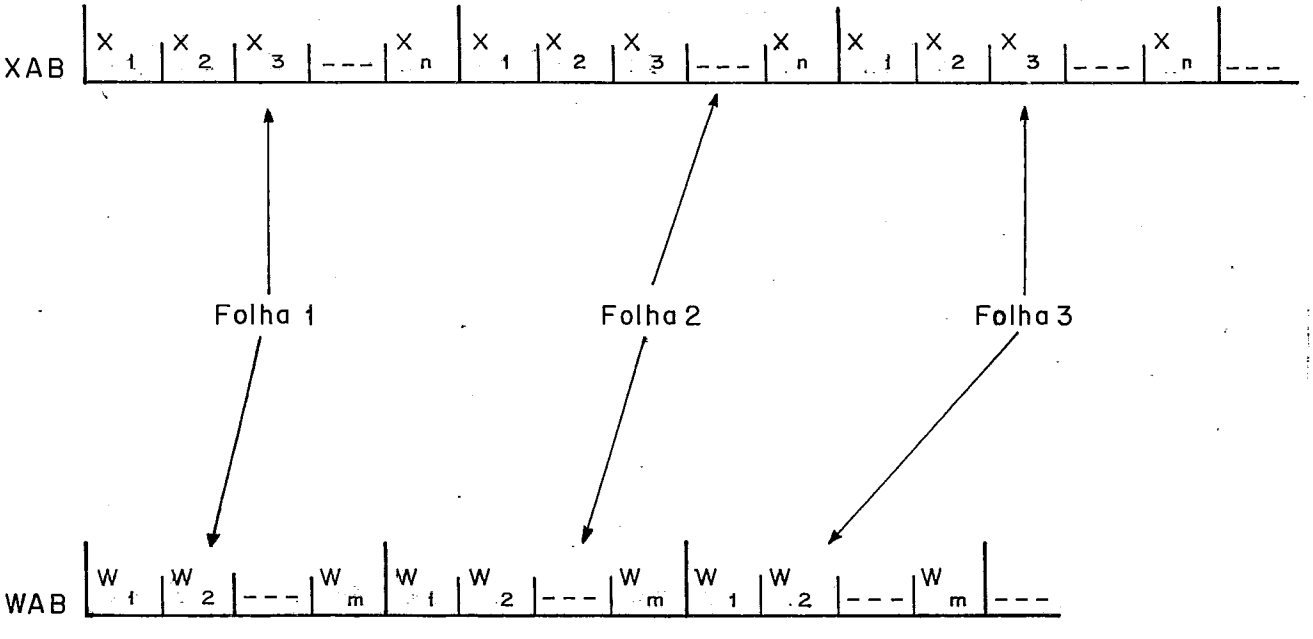
WAB contém o valor das variáveis duais dos nós abertos (folhas).

A existência dos dois arrays se deve ao fato de que no Out-Of-Kilter nós já entramos com uma solução inicial. Em

experiências práticas vimos que, entrando com uma solução inicial qualquer do tipo $\overline{X}=0$ e $\overline{W}=0$, nós levamos muito tempo para chegarmos à solução. Como no nosso caso nós só tiramos um arco de Kilter, quando separamos o problema em dois, o tempo gasto fica muito menor pois basta que o algoritmo Out-Of-Kilter coloque esse arco em Kilter.

Outro detalhe importante é que como a tendência é nós ficarmos com um número extremamente grande de folhas na nossa arborescência, número esse diretamente proporcional ao número de iterações. Os dois arrays podem não ser suficientes para armazenar os valores de X e W de todas as folhas. Então, quando excedemos as 65000 palavras previstas, gravamos o resto em disco num arquivo de acesso direto.

Esquematicamente podemos representar os dois arrays da seguinte forma:



3. AB, FE E LISTAB

AB contém dados dos nós abertos (folhas).

Cada "segmento" da lista contém os valores:

PIOR - pior arco (já resolvido o novo problema)

XPI - fluxo no pior arco (já resolvido o novo problema)

ARC - arco que foi dividido (gerando dois "novos arcos")

L - limite inferior de fluxo no novo arco

U - Limite superior de fluxo no novo arco

C - coeficiente linear de $\psi_i(x)$ do novo arco

B - coeficiente angular de $\psi_i(x)$ do novo arco

PONT P/PAI - ponteiro para o "segmento" de FE que é na arborescência o pai da folha

FE Contém dados dos nós fechados.

Cada "segmento" da lista contém os valores:

ARCO - limite inferior de fluxo no arco

U - limite superior de fluxo no arco

C - coeficiente linear de $\psi_i(x)$ do arco

B - coeficiente angular de $\psi_i(x)$ do arco

PONT P/PAI - ponteiro para o "segmento" de FE que é o pai do nó na arborescência

Os campos de FE correspondem aos campos ARC, L, U, C, B e PONT P/PAI de AB. Quando se fecha um nó, só se transfere os dados de uma lista para outra, sendo que o espaço correspondente da lista AB é reaproveitado para guardar uma nova folha.

LISTAB é uma lista diretamente relacionada com AB (a posição relativa dos "segmentos" é a mesma) contendo os valores LI (limite inferior p/a f.0.) e o ponteiro para o próximo nó aberto.

Cada "segmento" do LISTAB contém 2 campos e está contido em uma única palavra.

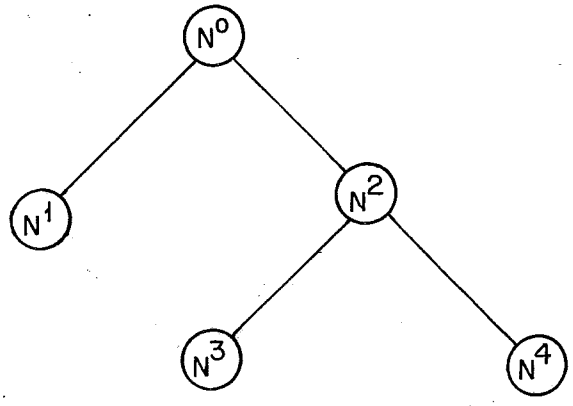
VALAB - Contém o valor de LI (bits: 47:28)

PROXAB - Contem o ponteiro para o próximo nó aberto (bits:19:20)

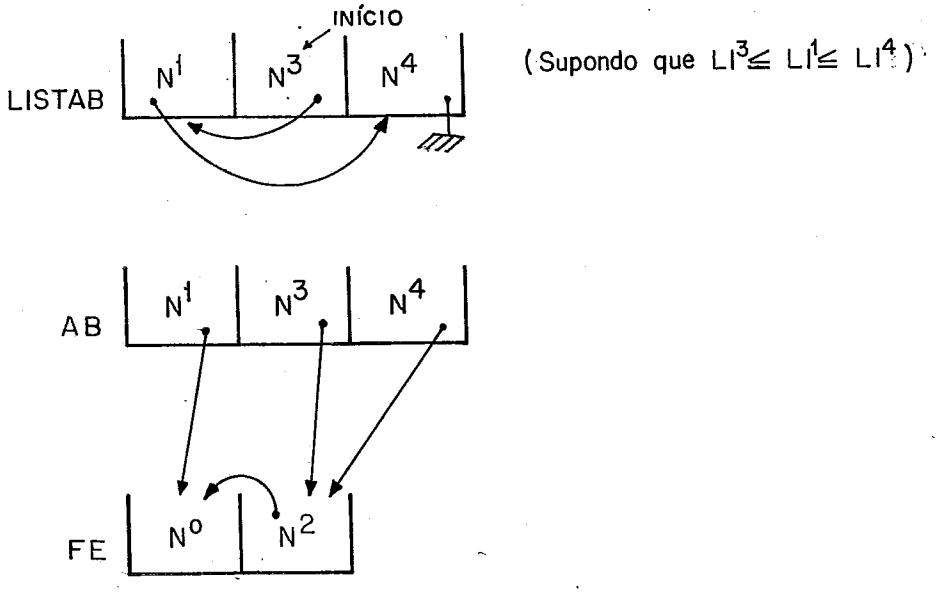
A classificação da lista é mantida em ordem ascendente pelos valores de LI.

A razão de se ter colocado a lista neste formato é que se pôde usar o comando LISTLOOKUP, próprio para pesquisa e inserção em listas linkadas ordenadas.

Digamos então que em uma dada iteração nós tenhamos gerado a seguinte arborescência:



O esquema das listas AB, FE e LISTAB após a geração da folha N^4 estaria assim:

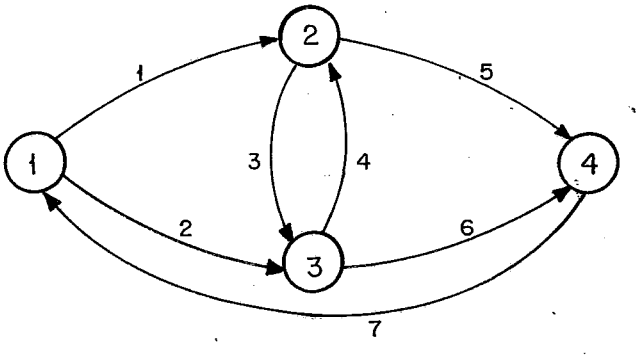


Essa estrutura nos permite subir na arborescência a partir de um nó N^k qualquer obtendo os valores mais recentes de cada arco modificado nas etapas de separação. Não é necessário então que se guarde todos os dados de todas as folhas, o que representa enorme economia de espaço em memória.

Seguindo o nosso exemplo, se nós fossemos agora, proceder a mais uma etapa de separação, o nó escolhido seria o N^3 (início da lista de abertos LISTAB), fecharíamos esse nó, partindo de um problema cujos dados seriam compostos dos nós N^3, N^2 e N^0. N^0 logicamente representa o nosso problema original P^0.

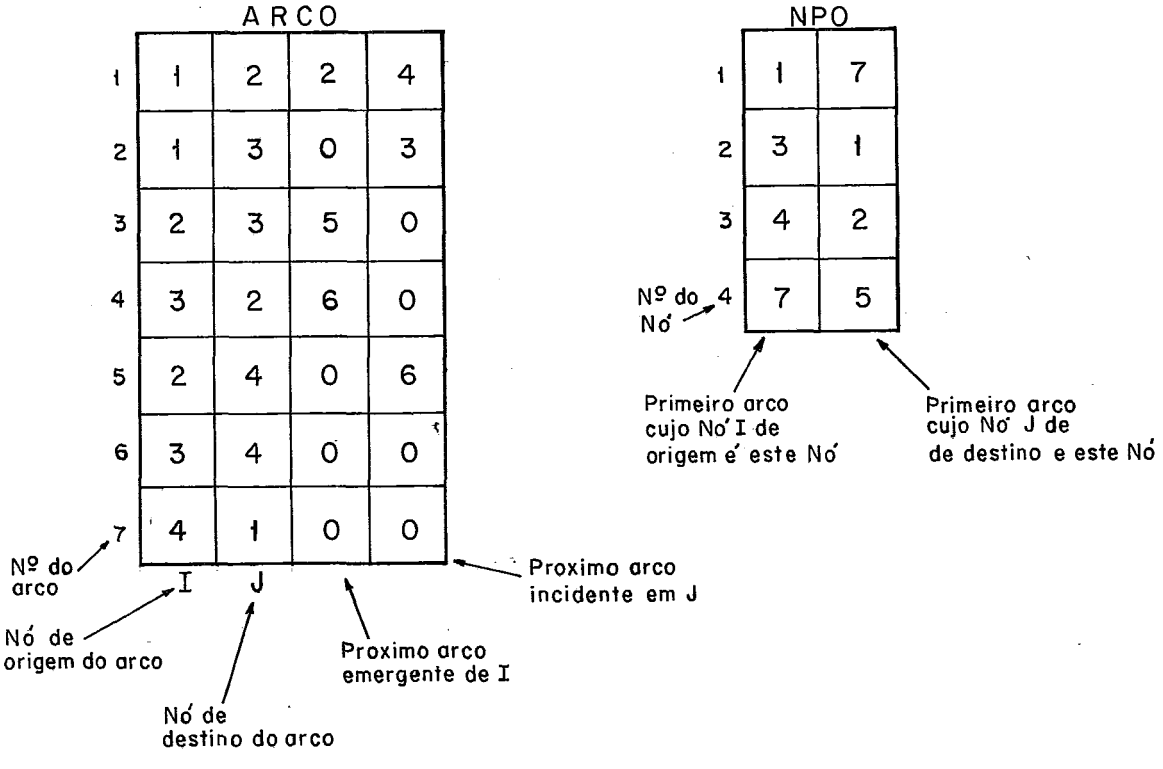
4. ARCO E NPO

As listas ARCO e NPO tem a finalidade de nos permitir no out-of-kilter ir percorrendo os arcos da rede para rotular os nós. Podemos com as listas, dado um nó, sabermos quais são os arcos que nele incidem e quais as que dele emergem. Daremos um exemplo elucidativo que a nosso ver é a melhor forma de se entender as duas listas. Seja a rede:



Colocamos nos arcos uma numeração de 1 a 7 que corresponde a ordem de entrada deles na lista ARCO. (Conforme vão sendo lidos os cartões).

Então ficamos com as listas:



Assim, se queremos varrer todos os arcos emergentes do nó 3 por exemplo fazemos o seguinte:

Em NPO vemos que o primeiro arco cujo nó de origem I é o nó 3 é o arco 4. Vamos então à lista ARCO e visitamos o arco 4, feito isso vemos que o próximo arco emergente de 3 é o arco 6, visitamos então o arco 6. Vemos que o próximo arco emergente de 3 é o arco 0, isto é, não há mais arco a ser visitado.

A nossa varredura visitou então os arcos 4 e 6 que são como queríamos os arcos emergentes do nó 3. O mesmo poderia ser feito para os arcos incidentes em um determinado nó, por exemplo, o nó 4, quando então visitaríamos o primeiro (ver em NPO) que é o 5 e em seguida (ver em ARCO) o nó 6.

7.3 - ENTRADA DOS DADOS

7.3.1 - Introdução

A entrada de dados poderá ser feita utilizando-se os cartões do tipo M, 0, 1, 2 e 3 descritos nos itens a seguir.

Os cartões devem vir na seguinte ordem: primeiro deverá vir o cartão tipo M (Mestre), em seguida os cartões tipo 0 (Descrição das funções), Tipo 1 (Centros Produtores), Tipo 2 (Armazéns e Transporte) e finalmente os cartões Tipo 3 (Centros Consumidores).

Nós projetamos para cada tipo de Cartão um formulário de entrada. Os formulários (vide anexo II) podem ser reproduzidos e visam facilitar a informação dos dados que deverão ser digitados.

7.3.2 - Cartão tipo M (Mestre)

A finalidade desse cartão é fornecer os dados básicos para a execução do programa. Deve ser o primeiro da massa de cartões de dados.

Seus campos são:

Col. 1: TIPO

Digitar M

Col. 2: LISTA

Se refere a opção de listagem:

0 Resumo do processamento

1 Resultado do problema (contendo dados de produção, transporte, armazenagem e consumo)

2 Listagem dos cartões de dados

4 Listagem passo a passo (contendo a cada iteração o conteúdo das principais variáveis para análise do progresso do algoritmo).

A combinação dessas listagens pode ser obtida através da informação da soma dos valores. Por exemplo para se obter o resumo do processamento, o resultado do problema e a listagem passo a passo, o valor informado deverá ser 5 (0 + 1 + 4). Aliás, o resumo do processamento como se verifica sempre será impresso.

Maiores detalhes sobre as listagens serão vistos no item 7.4 - Relatórios.

Col. 3 a 6 - PROBLEMA

Informar um número que identifique o problema apresentado. Esse número será impresso nos cabeçalhos do resumo do processamento e do resultado do problema.

Col. 7 a 10 - NÚMERO DE NÓS

Se refere ao número de nós da rede. A esses nós serão automaticamente acrescentados dois nós artificiais que serão ligados por um arco artificial de retorno, permitindo a resolução dos problemas linearizados pelo Out-Of-Kilter.

Col. 11 a 14 - NÚMERO DE CENTROS PRODUTORES

Informar o número de centros produtores da rede.

Col. 15 a 18 - NÚMERO DE ARCOS

Informar o número de arcos da rede.

Devemos lembrar que como o problema é de transbordo, os armazéns e meios de transporte devem constituir arcos capacitados, tendo os nós intermediários da rede produção e consumo nulos.

Aos arcos informados, será automaticamente acrescentado um arco de retorno para que se possa resolver os proble-

mas lineares pelo Out-of-Kilter.

Col. 19 a 22 - NÚMERO DE CENTROS CONSUMIDORES

Informar o número de centros consumidores da rede.

Col. 23 a 26 - NÚMERO DE ARCOS COM FUNÇÕES NÃO-LINEARES

Colocar o número de arcos da rede cujas funções de custo são côncavas, do tipo escada e/ou outras não-lineares.

Col. 27 a 36 - TOLERÂNCIA (com duas casas decimais)

Informar a tolerância que se deseja. A tolerância é obtida através da diferença entre os valores de $F(x)$ e $\psi(x)$.

O valor da tolerância pode ser dado em termos absolutos ou em valor percentual.

No segundo caso é obtida pela fórmula:

$$\frac{F(x) - \psi(x)}{F(x)} \times 100$$

Col.37 - PERCENTUAL

Colocar o símbolo % caso se tenha informado a tolerância com um valor em percentual.

Col. 38 a 41 - NÚMERO DE FUNÇÕES NÃO-LINEARES

Colocar o número de funções de custo não lineares. (Definidas nos cartões tipo 0).

7.3.3 - Cartão Tipo Ø (Funções)

Para cada função não-linear é necessário um grupo de cartões tipo 0.

O primeiro deverá ter:

Col. 1 - TIPO

Digitar 0.

Col. 2 a 6 - NÚMERO DA FUNÇÃO

Digitar um número que identifique a função. Qualquer referência futura a essa função será feita através desse número.

Col. 7 a 8 - NÚMERO DE PATAMARES

Indicar o número de "patamares" da função. Se a função for côncava, informar 00.

Col. 9 a 80 - Não digitar nada nessas colunas.

Os demais cartões são descritores dos "patamares" da função. Cada "patamar" deve ser descrito por uma função do tipo $a + b\ x^c + d\ x^e$ que nos casos praticos que vimos, serve para representar genericamente os tipos de funções lineares e côncavas. Vale ressaltar que para funções côncavas só teremos um cartão do que seria um "patamar" côncavo.

Os campos de cada cartão então são:

Col. 1 - TIPO

Digitar 0

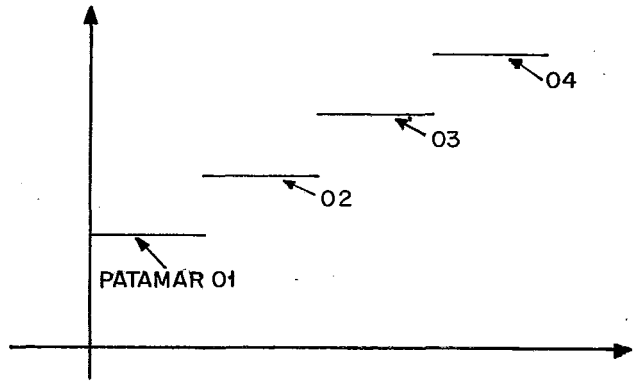
Col. 2 a 6 - NÚMERO DA FUNÇÃO

Deve ser idêntico ao do primeiro cartão da função.

Col. 7 a 8 - NÚMERO DO PATAMAR

Deve conter o número do patamar. Este número deve ser sequencial unitário a partir de 01 de tal maneira que se numere os patamares da esquerda para a direita (ver figura) e que o número do último patamar coincida com o número de patamares informado no primeiro cartão da função. Para funções côncavas deve conter 00.

Exemplo:



Col. 11 a 20 - LIMITE SUPERIOR DO PATAMAR (com duas casas decimais)

É o limite superior de fluxo para cada patamar.Não informamos o limite inferior, pois estamos considerando que as funções são "Lower-Semicontinous", o que nos indica que os patamares são fechados à direita. Além disso nós assumimos que $f(0) = 0$ para todas as funções não lineares, logo, L do primeiro patamar é $\cong 0$.

Col. 21 a 70 - PARÂMETROS DO PATAMAR (cada campo com 10 posições, tendo duas casas decimais)

Informar os parâmetros a, b, c, d e e da função $a + b\ x^c + d\ x^e$ que descreve o patamar. Se os valores de c e e forem nulos ou não forem informados, serão transformados em 1.

7.3.4 - Cartão Tipo 1 (Produção)

A finalidade dos cartões Tipo 1 é fornecer os dados dos centros produtores. Para cada centro um cartão. Os cartões Tipo 1 devem vir após os cartões tipo 0. O número de cartões tipo 1 tem que ser igual ao informado no campo Número de Centros Produtores (Cols. 11 a 14) do Cartão Mestre.

Os campos dos cartões são:

Col. 1 - TIPO

Digitar 1.

Col. 2 a 6 - NÚMERO DO CARTÃO

Colocar um número que identifique o cartão.

Col. 7 a 10 - NÓ PRODUTOR

Colocar o número do nó produtor.

Col. 11 a 20 - VALOR DA PRODUÇÃO (com duas casas decimais)

Informar a quantidade produzida pelo centro produtor.

Col. 21 a 32 - NOME DO CENTRO PRODUTOR

Informar o nome do centro produtor.

7.3.5 - Cartão Tipo 2 (Transporte/Armazenagem)

A finalidade dos cartões tipo 2 é fornecer os dados dos armazéns e meios de transporte da rede. Para cada arco da rede devemos ter um cartão.

Estes cartões devem vir em seguida aos cartões tipo 1. O número de cartões do tipo 2, deve ser igual ao informado no campo Número de arcos (Cols. 15 a 18) do Cartão Mestre.

Importante: Os arcos com funções não-lineares (côncavas, "escada" ou outras) devem vir na frente dos demais. O número desses cartões deve ser igual ao informado no campo Número de arcos com funções não lineares (Cols. 38 a 41) do Cartão Mestre.

Os campos dos cartões tipo 2 são:

Col. 1 - TIPO

Digitar 2

Col. 2 a 6 - NÚMERO DO CARTÃO

Digitar um número que identifique o cartão

Col.7 a 10 - NÓ DE ORIGEM

Informar o número do nó de origem do arco

Col. 11 a 14 - NÓ DE DESTINO

Informar o número do nó de destino do arco

Col. 15 - TIPO DO ARCO

Informar:

0 - se o arco se refere a um meio de transporte

1 - se o arco se refere a um armazém

Col. 16 - TIPO DE FUNÇÃO

Informar:

0 - se a função custo do arco é linear

1 - se a função custo do arco é não-linear (côncava, "escada", ou outras)

Col. 21 a 30 - LIMITE INFERIOR (com duas casas decimais)

Informar o limite inferior de fluxo no arco. Para arco com funções não-lineares não admitimos $L \neq 0$.

Col. 31 a 40 - LIMITE SUPERIOR (com duas casas decimais)

Informar o limite superior de fluxo no arco.

Col. 41 a 50 - CUSTO UNITÁRIO (com duas casas decimais)

Caso a função seja linear, informar o custo por unidade de fluxo no arco.

Col. 51 a 55 - NÚMERO DA FUNÇÃO

Caso a função seja não linear, informar o número da função custo. Este número deve corresponder a um dos cartões tipo 0.

Col. 56 a 67 - NOME DO ARCO

Informar o nome do armazém ou meio de transporte a que se refere o arco.

Col. 68 a 72 - ÍNDICE DE ROTAÇÃO (com duas casas decimais)

Informar o índice de rotação do armazém. Caso não seja informado, ou caso sejo nulo, transformamos o índice no valor 1.

7.3.6 - Cartão Tipo 3 (Consumo)

A finalidade dos cartões tipo 3 é fornecer os dados dos centros consumidores da rede. Para cada centro consumidor temos um cartão. Os cartões tipo 3 devem vir após os cartões tipo 2.

O número de cartões tipo 3 deve ser igual ao informado no campo número de centros consumidores (Cols. 19 a 22) do cartão Mestre. Os campos dos cartões são:

Col. 1 - TIPO

Digitar 3

Col. 2 a 6 - NÚMERO DO CARTÃO

Informar um número que identifique o cartão

Col. 7 a 10 - NÓ CONSUMIDOR

Colocar o número do nó consumidor

Col. 11 a 20 - VALOR DO CONSUMO (com duas casas decimais)

Informar a quantidade consumida no centro consumidor

Col. 21 a 32 - NOME DO CENTRO CONSUMIDOR

Informar o nome do centro consumidor

7.4 - RELATÓRIOS

7.4.1 - Introdução

O programa imprime os seguintes relatórios:

1. Listagem dos cartões de dados - É um "espelho"dos cartões de dados de entrada

2. Listagem passo a passo - Contendo a cada iteração o conteúdo das principais variáveis para análise do progresso do algoritmo.

3. Resumo do processamento - Contendo dados resumidos sobre o resultado do processamento.

4. Resultado do problema - Contendo dados de produção, transporte, armazenagem e consumo.

O resumo do processamento, sempre sairá impresso ao final da execução do programa. Os demais relatórios são opcionais, podendo ser selecionados pelo campo Lista do Cartão Mestre (ver item 7.3.2). Todos os relatórios tem um cabeçalho somente no início, pois achamos uma sofisticação desnecessária imprimir cabeçalho a cada página dos relatórios.

Dois campos aparecem em todos os cabeçalhos, com exceção da listagem dos cartões de dados, que são VERSÃO e PROBLEMA.

O campo VERSÃO indica a versão do programa. No nosso caso é igual a 5 pois como já vimos no item 7.2.1, pela evolução do programa nós estamos na 5ª. versão. Este valor só deve ser alterado quando da criação de uma nova versão do programa e para tando teremos que alterar o comando VERSÃO: = 5 na rotina IMPRIMIR.

O campo PROBLEMA identifica o problema resolvido. Este dado é copiado do Cartão Mestre.

Nos itens a seguir daremos uma descrição do conteúdo dos relatórios, sendo que um teste completo com todos os relatórios impressos pode ser visto no Anexo II.

Vale ressaltar que as mensagens impressas devido a algum erro detectado na leitura dos cartões de dados, são auto-explicativas, não sendo portanto descritas neste capítulo.

7.4.2 - Listagem dos Cartões de Dados

Como vemos no exemplo 7.4.1, contém um cabeçalho indicando as colunas de 1 a 80 e em seguida em cada linha o "espelho" de cada cartão de dados de entrada.

Chamamos atenção para o fato de que os campos são editados na saída, de modo que se por exemplo, um campo com 5 inteiros e 2 decimais foi digitado com o valor 0123, na listagem sairá com 123,00.

7.4.3 - Listagem Passo a Passo

Uma ilustração pode ser vista no exemplo 7.4.2. A cada iteração do algoritmo é impresso um conjunto de 2 linhas.

O cabeçalho impresso no início indica a posição dos campos K, NO, LS, LI, ERRO, TEMPO, PIOR, L (PIOR) e CUSTO em L(PIOR), X(PIOR) e CUSTO em X(PIOR), U(PIOR) e CUSTO em U(PIOR) e PTO. DIVISÃO e CUSTO no PTO. DIVISÃO, campos esses cuja descrição daremos a seguir:

1. K

   Indica a iteração do algoritmo

2. NO

   Indica o nó da lista de abertos que está sendo separado.

3. LS

   Contém o valor do limite superior da f.o.

4. LI

   Contém o valor do limite inferior da f.o.

5. ERRO

   Nos dá o erro máximo cometido até a iteração obtida pela fórmula

$$\frac{LS - LI}{LS}$$

   O erro, dependendo da opção no cartão Mestre pode ser dado em valor percentual. (É então multiplicado por 100)

6. TEMPO

Tempo de CPU em segundos, gasto até a iteração (não é contado o tempo de leitura dos cartões de dados)

7. PIOR

Indica o pior arco. É o arco que vai ser dividido em dois na iteração. O número que aparece nesse campo corresponde ao n-ésimo arco não-linear descrito através dos cartões tipo 2 na entrada de dados.

8. L(PIOR) e CUSTO em L(PIOR)

Correspondem respectivamente ao limite inferior de fluxo no arco a ser dividido e o custo neste ponto.

9. X(PIOR) e CUSTO em X(PIOR)

Se referem ao fluxo no arco a ser dividido e ao custo no ponto X(PIOR)

10. U(PIOR) e CUSTO em U(PIOR)

Se referem ao limite superior de fluxo no arco a ser dividido e ao custo neste ponto

11. PTO. DIVISÃO e CUSTO em PTO. DIVISÃO

Correspondem ao ponto de divisão do arco e ao custo neste ponto. É bom lembrar que para funções côncavas o ponto de divisão é o próprio X(PIOR). Para funções patamar os dois pontos podem ser distintos.

7.4.4 - Resumo do Processamento

Um resumo do processamento pode ser visto no exemplo 7.4.3.

São impressas as principais características como o número de nós da rede, número de centros produtores, número de arcos e número de centros consumidores, dados estes obtidos do cartão Mestre. Além disso são impressos: número de iterações, valor da função objetivo, tolerância (dada por cartão Mestre), erro máximo cometido em valor absoluto e um percentual, tempo total gasto, tempo do out-of-kilter e tempo do algoritmo.

Com relação aos tempos medidos, merece ser ressaltado o seguinte:

- Os tempos são todos em segundos

- Os tempos foram medidos com o comando TIME (2) do Algol que nos dá o tempo de CPU

- O tempo de out-of-Kilter se refere a soma dos tempos gastos nas diversas chamadas da procedure KILTER no programa

Separamos o tempo da primeira chamada das demais pois ela normalmente leva mais tempo que as demais, pois partimos da solução inicial X=0; como vimos, nas demais chamadas já entramos com uma solução inicial onde só uma parte dos arcos estão fora de KILTER.

- O tempo do algoritmo se refere ao algoritmo Branch and Bound sem considerar o tempo gasto com o out-of-kilter.

Importante: Caso o número de iterações necessário para atingir a tolerância desejada seja de tal ordem que não seja mais possível armazenar os dados das folhas da arborescência, nós encerramos o programa imprimindo ao final do resumo do processamento a mensagem:

Atenção: Não foi possível atingir a tolerância desejada

7.4.5 - Resultado do Problema

O resultado do problema é impresso em quatro partes: Centros Produtores, Transporte, Armazenagem e Centros Consumidores.

7.4.5.1 - Centros Produtores

Como mostra o exemplo 7.4.4, contém os valores da produção dos nós da rede. Os campos impressos correspondem aos informados nos cartões tipo 1 (Produção), além do total da produção. À esquerda de cada linha é impresso um número de ordem.

7.4.5.2 - Transporte

Como pode ser visto no exemplo 7.4.5, são impressos: um número de ordem, o nome do meio de transporte, o arco(re

presentado pelo nó de origem e nó de destino), o limite inferior e superior de fluxo no arco, o custo unitário (se a função for linear), o número da função (se a função for não linear), o índice de rotação, a capacidade estatíca do arco, o fluxo no arco e o custo referente ao fluxo.

Além disso são impressos ao final o total de fluxo e o custo de transporte.

7.4.5.3 - Armazenagem

Como vemos no exemplo 7.4.6, é um relatório análogo ao do Transporte, sendo que se refere aos dados dos armazéns.

7.4.5.4 - Centros Consumidores

No exemplo 7.4.7 pode ser verificado que é um relatório análogo ao dos Centros Produtores, sendo que se refere aos dados dos Centros Consumidores informados nos cartões tipo 3 (consumo).

```
1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

M7    6    8    2    8    2    2        0.00%    2
0  1003 3
0  1003 1          1.50         3.00         0.00         0.00         0.00         0.00
0  1003 2          3.50         5.00         0.00         0.00         0.00         0.00
0  1003 3          6.00         7.00         0.00         0.00         0.00         0.00
0  1004 3
0  1004 1          0.50         7.00         0.00         0.00         0.00         0.00
0  1004 2          2.00         9.00         0.00         0.00         0.00         0.00
0  1004 3          4.00        11.00         0.00         0.00         0.00         0.00
1     1    1       3.00   PRODUTOR 1
1     2    2       5.00   PRODUTOR 2
2     3    3    511              0.00         6.00         0.00 1003 ARMAZEM 3/5 0.00
2     4    4    611              0.00         4.00         0.00 1004 ARMAZEM 4/6 0.00
2     5    1    300              1.00         9.00         1.00    0 ESTRADA 1/3 0.00
2     6    2    300              0.00         6.00         1.00    0 ESTRADA 2/3 0.00
2     7    2    400              0.00         5.00         2.00    0 ESTRADA 2/4 0.00
2     8    5    700              0.00         8.00         3.00    0 ESTRADA 5/7 0.00
2     9    6    700              0.00         6.00         1.00    0 ESTRADA 6/7 0.00
2    10    6    800              0.00         5.00         2.00    0 ESTRADA 6/8 0.00
3    11    7       7.00CONSUMIDOR 7
3    12    8       1.00CONSUMIDOR 8
```

Exemplo 7.4.1 - Listagem dos cartões de dados

K	NO	LS	LI	ERRO	TEMPO	PIOR	L(PIOR) CUSTO	X(PIOR) CUSTO	U(PIOR) CUSTO	PTO DIVISAO CUSTO
1	1	47.00	44.00	6.38%	0.40	2	0.00	2.00	4.00	2.00
							0.00	9.00	11.00	9.00
2	2	47.00	45.00	4.26%	0.48	1	0.00	4.00	6.00	3.50
							0.00	7.00	7.00	5.00

Exemplo 7.4.2 - Listagem passo a passo

```
PROBLEMA DE LOCALIZACAO COM FUNCOES DE CUSTO NAO LINEARES

AUTOR: RONALDO RUST      ORIENTADOR: CLAUDIO THOMAS BORNSTEIN


VERSAO  5  PROBLEMA    6    **  RESUMO DO PROCESSAMENTO  **

CARACTERISTICAS:
   8 NOS      2 PRODUTORES      8 ARCOS      2 CONSUMIDORES
NO. DE ITERACOES.................              2
VALOR DA FUNCAO OBJETIVO........           47.00
TOLERANCIA.......................            0.00 %
ERRO MAXIMO COMETIDO............            0.00 =>  0.00 %
TEMPO TOTAL GASTO...............       0.5500 SEG
OUT-OF-KILTER...................       0.3667 SEG
      PRIMEIRA CHAMADA..........       0.2833 SEG
      MEDIA DAS DEMAIS CHAMADAS.       0.0208 SEG/CHAMADA
ALGORITMO.......................       0.1833 SEG
      INICIALIZACAO.............       0.1167 SEG
      MEDIA DAS ITERACOES.......       0.0333 SEG/ITERACAO
```

Exemplo 7.4.3 - Resumo do processamento

VERSAO 5 PROBLEMA 6 ** CENTROS PRODUTORES **

-NO-	-- NOME --	-- PRODUCAO --
1	PRODUTOR 1	3.00
2	PRODUTOR 2	5.00
	**** TOTAL	8.00

Exemplo 7.4.4 - Resultado do problema - Centros produtores

VERSAO 5 PROBLEMA 6 ** T R A N S P O R T E **

	-- NOME --	--ARCO--	LIM. INF.	LIM. SUP.	CUSTO UNIT.	FUNC	I.ROT	CAPACIDADE	--FLUXO--	-- CUSTO --
1	ESTRADA 1/3	1- 3	1.00	9.00	1.00	0	0.00	3.00	3.00	3.00
2	ESTRADA 2/3	2- 3	0.00	6.00	1.00	0	0.00	3.00	3.00	3.00
3	ESTRADA 2/4	2- 4	0.00	5.00	2.00	0	0.00	2.00	2.00	4.00
4	ESTRADA 5/7	5- 7	0.00	8.00	3.00	0	0.00	6.00	6.00	18.00
5	ESTRADA 6/7	6- 7	0.00	6.00	1.00	0	0.00	1.00	1.00	1.00
6	ESTRADA 6/8	6- 8	0.00	5.00	2.00	0	0.00	1.00	1.00	2.00

**** TOTAL DO FLUXO 16.00

**** CUSTO DE TRANSPORTE 31.00

Exemplo 7.4.5 - Resultado do problema - Transporte

VERSAO 5 PROBLEMA 6 ** A R M A Z E N A G E M **

	-- NOME --	--ARCO--	LIM. INF.	LIM. SUP.	CUSTO UNIT.	FUNC	I.ROT	CAPACIDADE	--FLUXO--	-- CUSTO --
1	ARMAZEM 3/5	3- 5	0.00	6.00	0.00	1003	1.00	6.00	6.00	7.00
2	ARMAZEM 4/6	4- 6	0.00	4.00	0.00	1004	1.00	2.00	2.00	9.00
						**** TOTAL DO FLUXO			8.00	
					**** CUSTO DE ARMAZENAGEM					16.00

Exemplo 7.4.6 - Resultado do problema - Armazenagem

VERSAO 5 PROBLEMA 6 ** CENTROS CONSUMIDORES **

	-NO-	-- NOME --	-- CONSUMO --
1	7	CONSUMIDOR 7	7.00
2	8	CONSUMIDOR 8	1.00
		**** TOTAL	8.00

Exemplo 7.4.7 - Resultado do problema - Centros consumidores

CAPÍTULO 8

TESTES

8.1 - INTRODUÇÃO

Foram realizados dois tipos de testes no programa: o primeiro deles, cujos resultados não achamos necessário apresentar neste trabalho, foi feito com um grupo de pequenos problemas com os resultados de todos os passos facilmente calculáveis sem o auxílio de computador e que foram usados somente para verificação de possíveis erros de lógica e de codificação; o segundo tipo de teste foi realizado com problemas maiores, apresentados neste Capítulo, cujos resultados servem para medidas de performance, comportamento em casos reais e para comparações com outros trabalhos.

Apresentaremos então três grupos de testes a saber: testes com funções côncavas, testes com funções tipo escada e testes comparativos.

Fizemos então como veremos, testes para verificar o comportamento do algoritmo com diferentes funções e com menor ou maior número de funções não lineares usando basicamente dados dos trabalhos da CIBRAZEM [4] e MONTEROSSO [17].

Todas as funções de custo foram expressas em termos da capacidade estática de armazenagem v. Esta capacidade é

obtida dividindo-se o fluxo anual do armazém pelo índice de rotação, fornecido pela CIBRAZEM.

Como resultado de cada problema, damos o valor encontrado para a função objetivo F(x) do problema (na realidade é um limite superior para o valor ótimo de F(x)),o valor de LI (limite inferior para o valor ótimo de F(x)), o erro cometido, o tempo de CPU e o número de iterações, além disso,apresentamos um gráfico Iteração x Erro através do qual se pode ter uma visualização da convergência do algoritmo, verificando a cada iteração o erro aproximado cometido e permitindo uma projeção para que se tenha idéia do número de iterações necessárias para obtenção de uma determinada tolerância.

Todos os testes com excessão do 8.4.1 foram executados no tempo de 150 segundos.

Um teste completo pode ser visto no Anexo III onde apresentamos todos os relatórios do problema 8.4.1.

Vale lembrar que em todos os problemas assumimos que o valor da função de custo de cada armazém g(v) na origem é igual a zero, isto é, g(0) = 0.

8.2 - TESTES COM FUNÇÕES CÔNCAVAS

Usamos para estes testes os dados da rede 1 do trabalho de MONTEROSSO |[17]|.

Trata-se basicamente de um problema de localização de armazéns para estocagem de arroz nas micro-regiões 34 e 35 do Maranhão com dados obtidos da CIBRAZEM |[4]|.

Os custos de transporte foram inferidos por MONTEROSSO |[17]|, de dados da Tabela Nacional de Fretes, emitida pelo Sindicato de Transporte de Carga do Rio de Janeiro.

Como centros produtores, foram considerados 9 municípios da micro-região 34 e 14 municípios da micro-região 35. Como oferta em cada município, foi usada a projeção para 1976 dos saldos comercializáveis, obtida a partir de uma regressão linear com dados dos últimos anos.

A produção total oferecida é de 317487 toneladas/ano, que deve fluir através dos armazéns, para 5 centros consumidores fixados em São Luiz (35%), Timon (30%), Candido Mendes (20%), Porto Franco (10%) e Imperatriz (5%). Esta distribuição de demanda foi arbitrada, por não haver informação precisa disponível.

Foram escolhidos 13 locais considerados potencialmente atrativos para a localização de armazéns. São eles: Bom Jardim, Lago da Pedra, Monção, Pindaré Mirim, Santa Inês, Santa Luzia, Vitorino Freire, Bacabal, Ipixuna, Olho d'água das Cunhas, Pedreiras, Poção de Pedras e Santo Antonio dos Lopes. A capacidade máxima de cada armazém é de 60.000 toneladas.

Os índices de rotação dos armazéns são 1,1 para a micro-região 34 e 1,07 para a micro-região 35.

A rede de transportes utilizada é constituída de 243 arcos.

Os três testes a seguir foram feitos para que se pudesse verificar o comportamento do programa com funções de custo diferentes.

Problema 8.2.1

Função de custo: $g_1(v) = 1439230,40$, $0 \leq v \leq 60000$

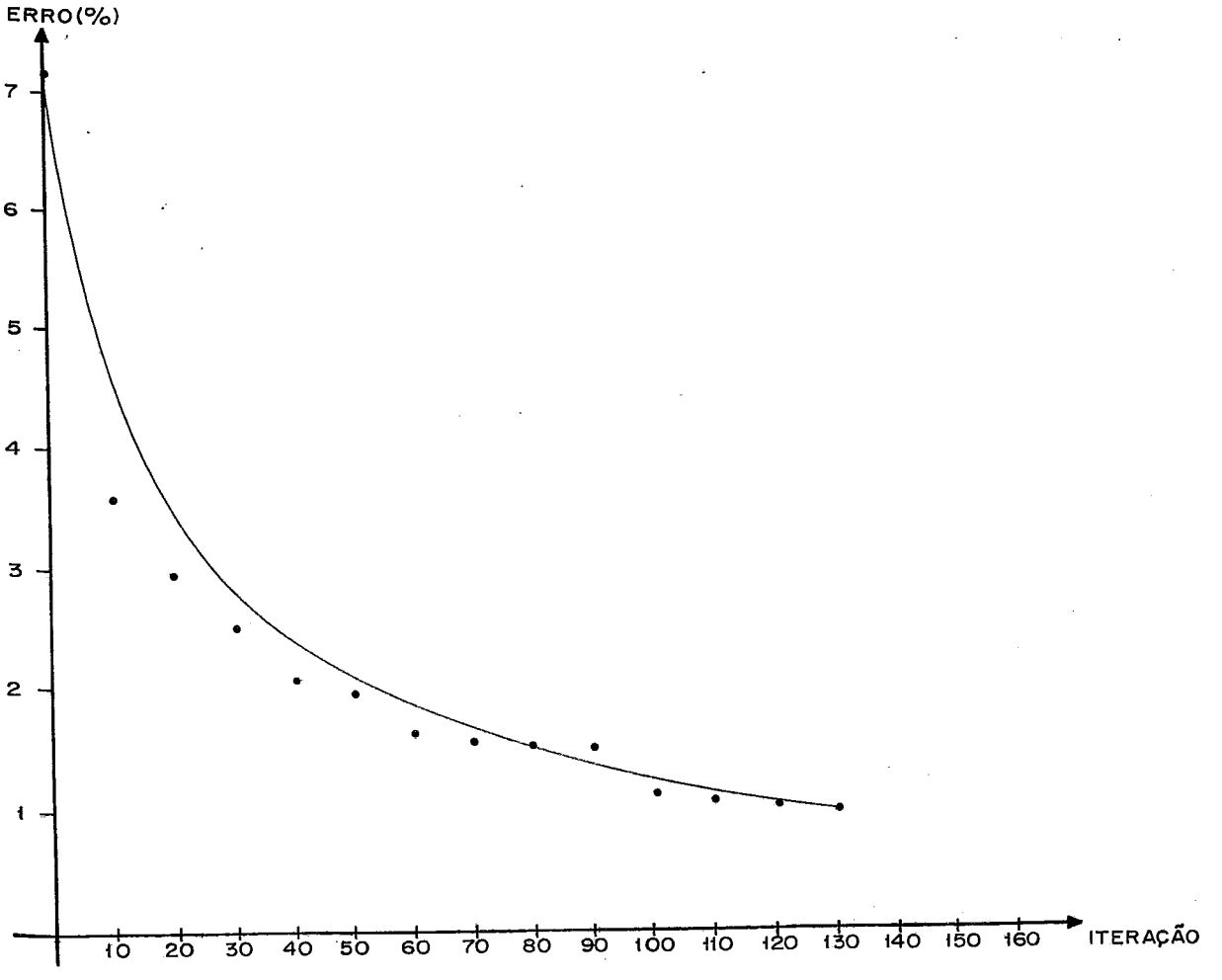
Resultado:

F(x) = 136282166,94

LI = 134914103,00

Erro = 1,00%

Número de iterações = 131



Problema 8.2.2

Função de custo: $g_2(v) = 400000 + 50000 \sqrt{v}, \quad 0 < v \leq 60000$

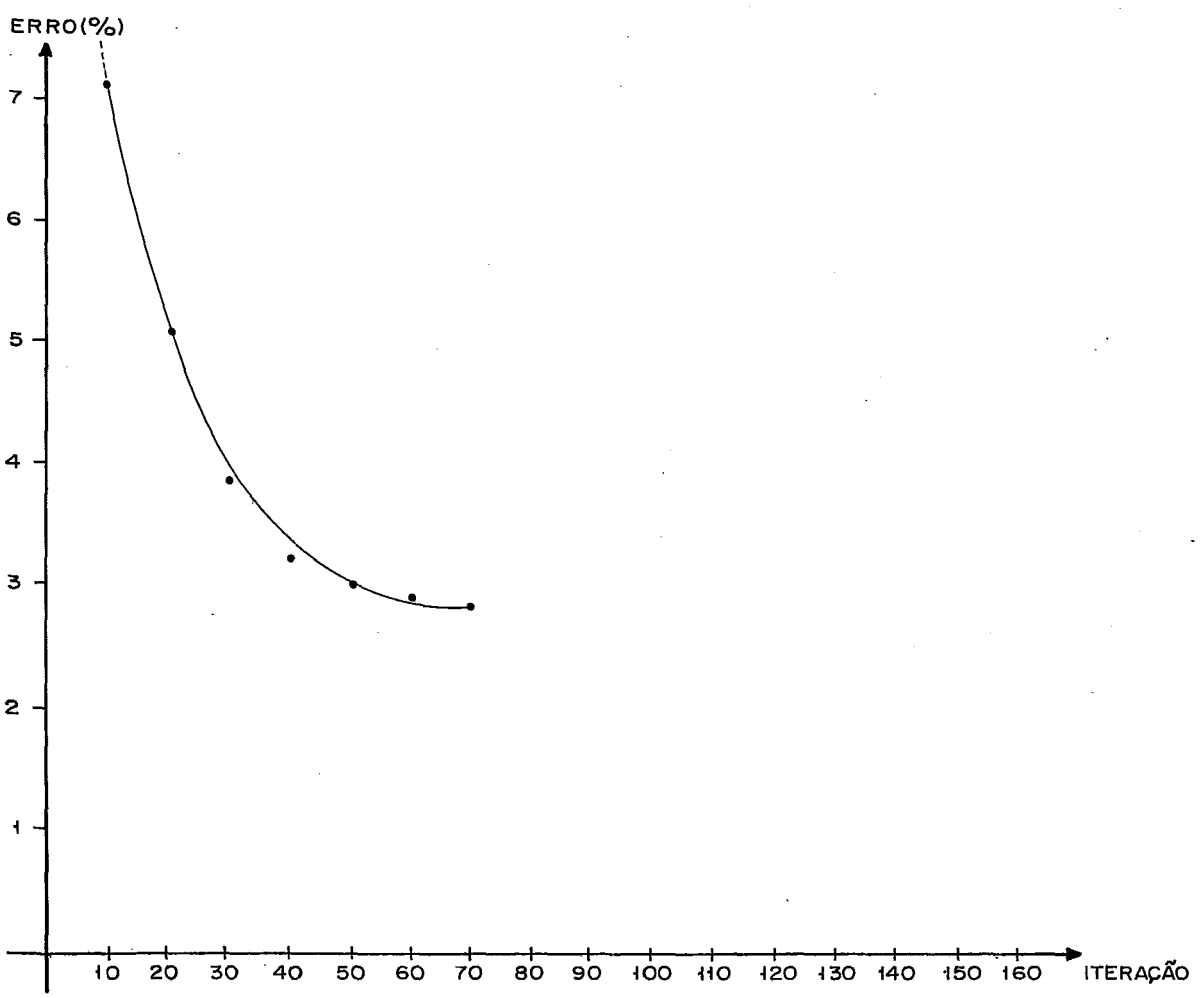
Resultado:

F(x) = 196833359,81

LI = 191303068,00

Erro = 2,81%

Número de iterações = 66



Problema 8.2.3

Função de custo: $g_3(v) = 400000 + 6000 \sqrt{v}$, $0 < v \leq 60000$

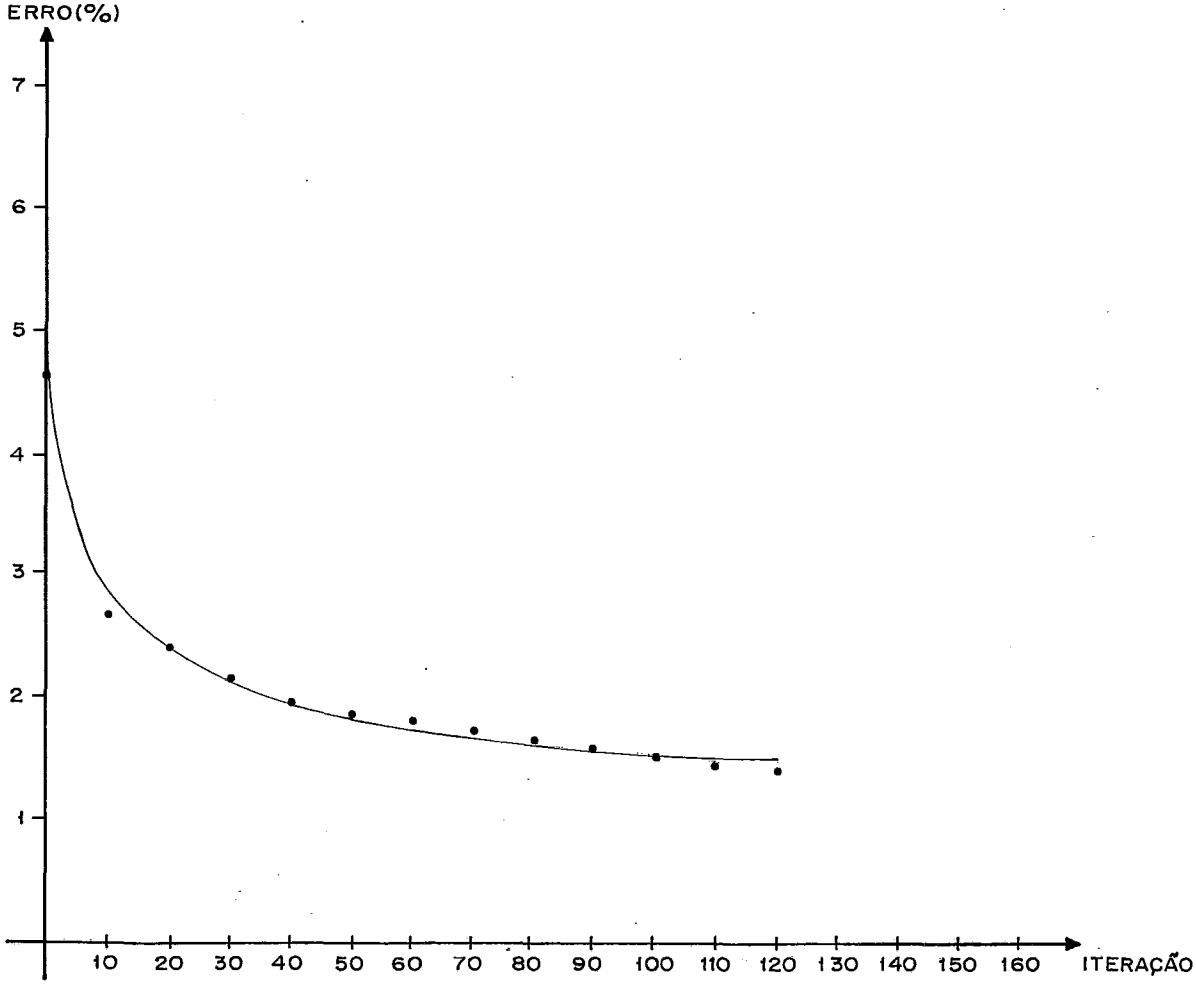
Resultado:

F(x) = 137786880,97

LI = 135902985,00

Erro = 1,37%

Número de iterações = 126



Nos nossos testes usamos em primeiro lugar uma função extremamente simples que é a função linear $g_1(v)$. Como era de se esperar o algoritmo se comportou extremamente bem, como pode ser visto pelo gráfico.

No problema 8.2.2 usamos uma função côncava do tipo $a + b\sqrt{v}$ onde o parâmetro b era bastante elevado, tentando com isso criar uma situação bem desfavorável. O resultado como se pode verificar foi bem pior do que o do problema anterior.

No problema 8.2.3, usamos uma função côncava do mesmo tipo, com o parâmetro b bem menor. Essa função é bastante semelhante às usadas por Monterosso [17] nos testes com dados reais. Obtivemos um comportamento intermediário em relação aos dois testes anteriores.

Apesar do que foi visto com as três funções usadas, seria dificílimo chegarmos a alguma conclusão definitiva sobre o comportamento do algoritmo, baseados nas características das funções côncavas usadas.

Apresentamos a seguir mais dois problemas, nos quais usamos os dados do problema 8.2.3, transformando respectivamente os custos dos 13 e 26 primeiros arcos de transporte de fixos para côncavos, usando a função $g_3(v)$. Tentamos com isso manter o tanto quanto possível as condições do problema 8.2.3, com a finalidade de verificarmos o comportamento do algoritmo com o aumento do número de arcos com funções côncavas.

Problema 8.2.4

Custo de transporte dos 13 primeiros arcos côncavos e custos de armazenagem côncavos (total: 26 arcos com custos côncavos)

Função de custo: $g_3(v) = 400000 + 6000\sqrt{v}$, $0 < v \leq 60000$

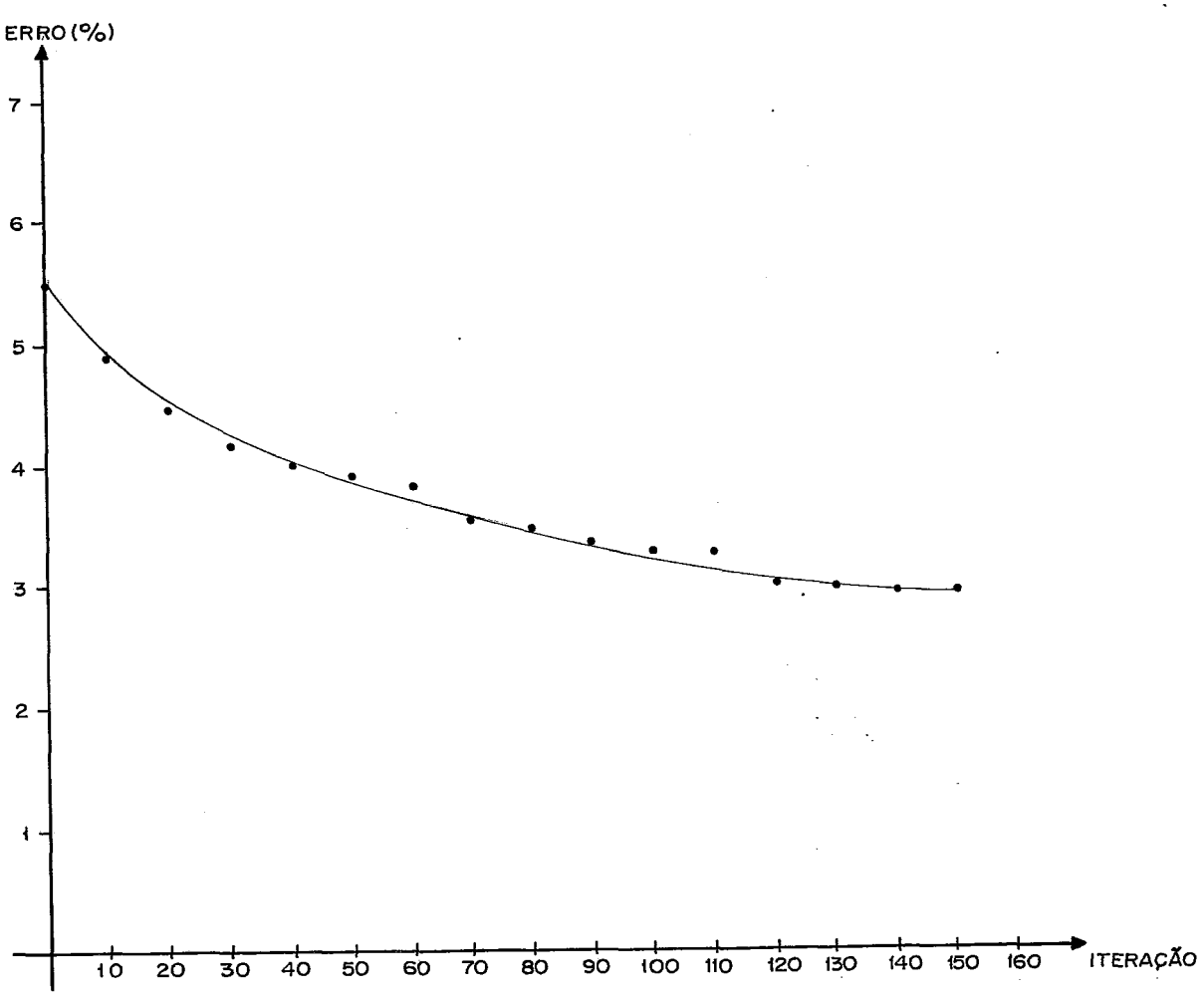
Resultado:

F(x) = 137371892,60

LI = 133340809,00

Erro = 2,93%

Número de iterações = 151



Problema 8.2.5

Custo de transporte dos 26 primeiros arcos côncavos e custos de armazenagem côncavos (total: 39 arcos com funções côncavas).

Função do custo: $g_3(v) = 400000 + 6000\sqrt{v}$, $0 < v \leq 60000$

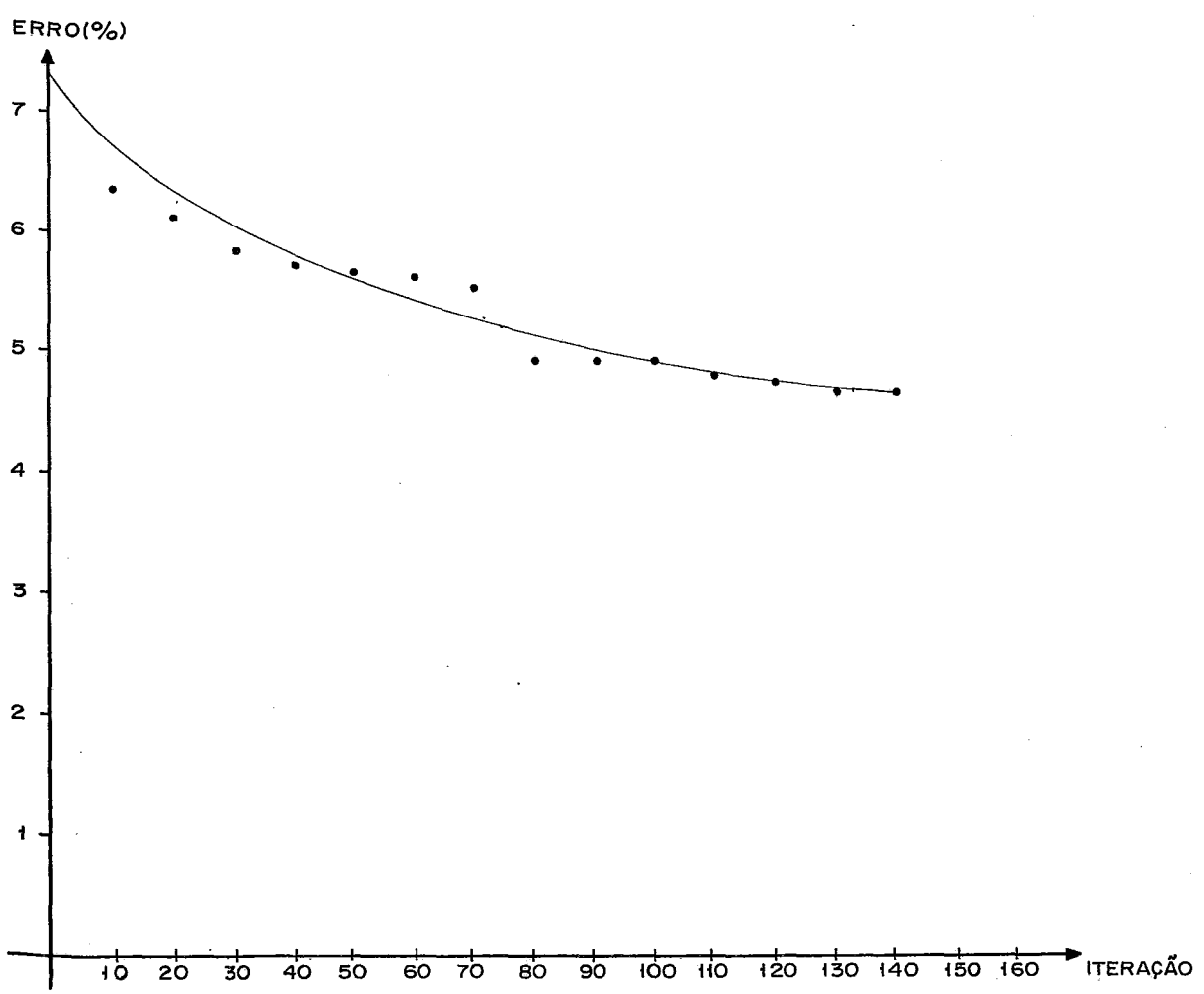
Resultado:

F(x) = 135383556,43

LI = 129042900,00

Erro = 4,68%

Número de iterações = 139

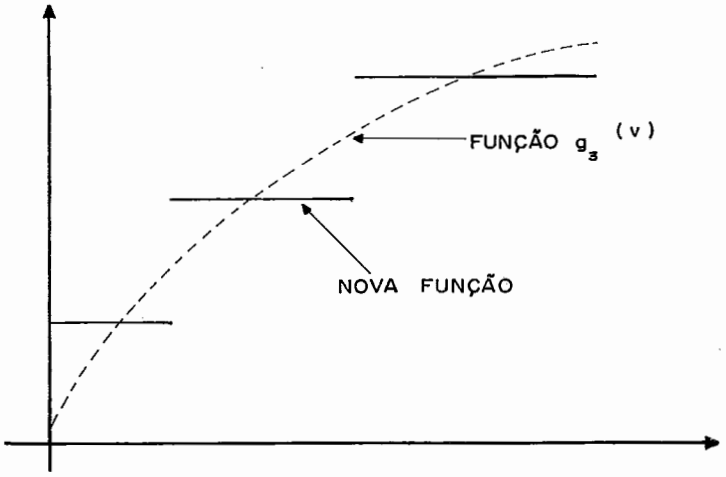


Como era de se esperar, comparados os resultados dos problemas 8.2.3, 8.2.4 e 8.2.5, a medida em que o número de arcos com funções de custo côncavas cresce piora a velocidade de convergência do algoritmo, isto é, a taxa de decréscimo médio do erro por iteração é menor. Não se pode dizer que a relação seja direta. Nós duplicamos e triplicamos o número de arcos com funções de custo côncavas em relação ao problema 8.2.3 e a velocidade não decresceu na mesma proporção.

8.3 - TESTES COM FUNÇÕES DO TIPO ESCADA

Para que tivessemos dados para comparações usamos como base o problema 8.2.3.

Transformamos a função $g_3(v)$ nas funções $g_4(v)$, $g_5(v)$ e $g_6(v)$ com 1,3 e 10 patamares respectivamente. A transformação foi feita como mostra a figura abaixo



Apesar da aproximação não ser muito boa, achamos que o comportamento dos testes deveria se assemelhar ao do problema 8.2.3.

Pelo tipo de divisão que fazemos no caso das funções escada é de se esperar que para funções com maior número de patamares a convergência piore.

Apresentamos então a seguir os resultados dos testes com as três novas funções.

Problema 8.3.1

Função de custo $g_4(v) = 1439230,40$, $0 < v \leq 60000$

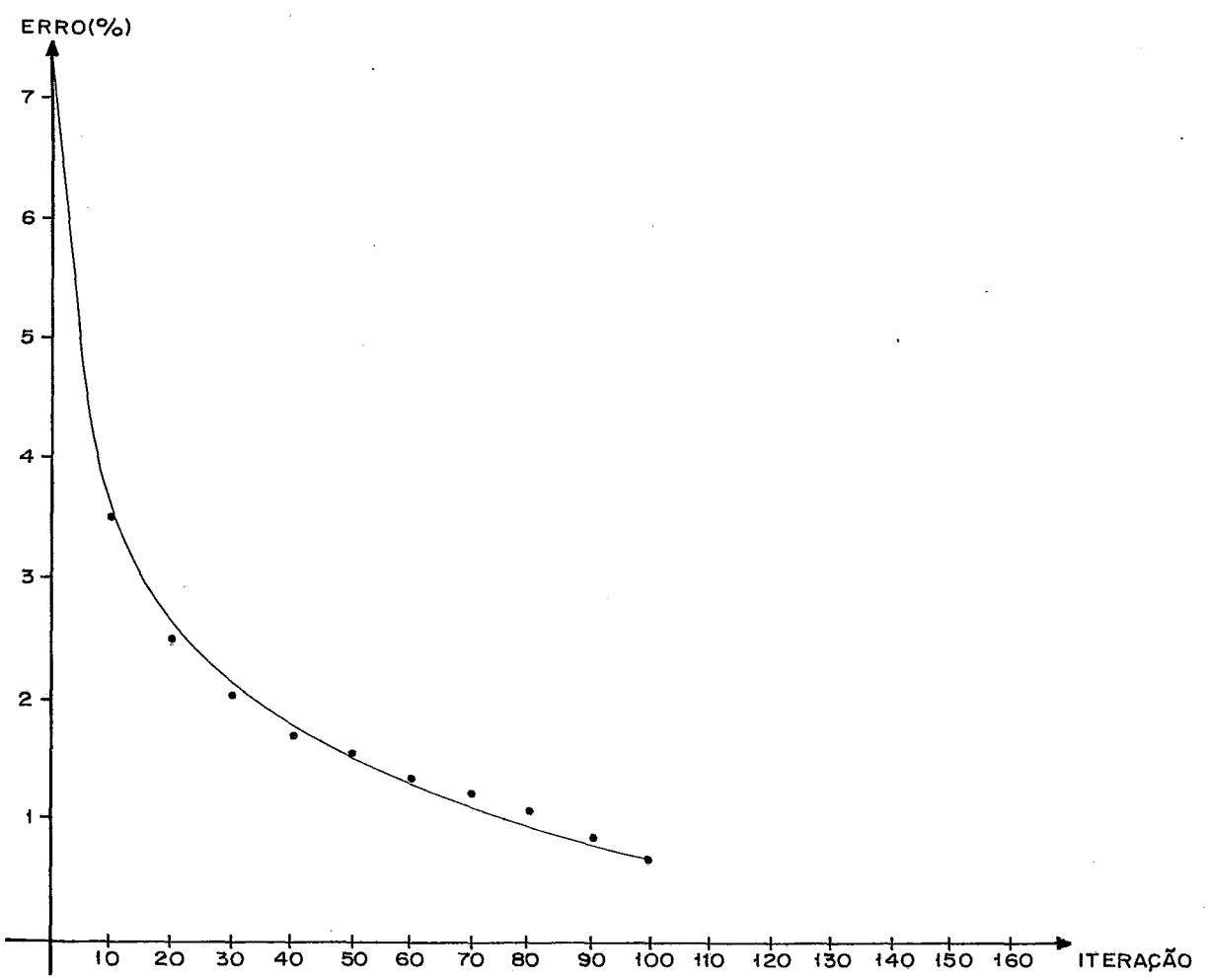
Resultado: (note que $g_4(v) = g_1(v)$)

F(x) = 136282166,94

LI = 135371674,00

Erro = 0,67%

Número de iterações = 101



Problema 8.3.2

$$\text{Função de custo } g_5(v) = \begin{cases} 824264{,}00 & 0 < v \leq 10000 \\ 1248528{,}10 & 10000 < v \leq 30000 \\ 1672792{,}20 & 30000 < v \leq 60000 \end{cases}$$

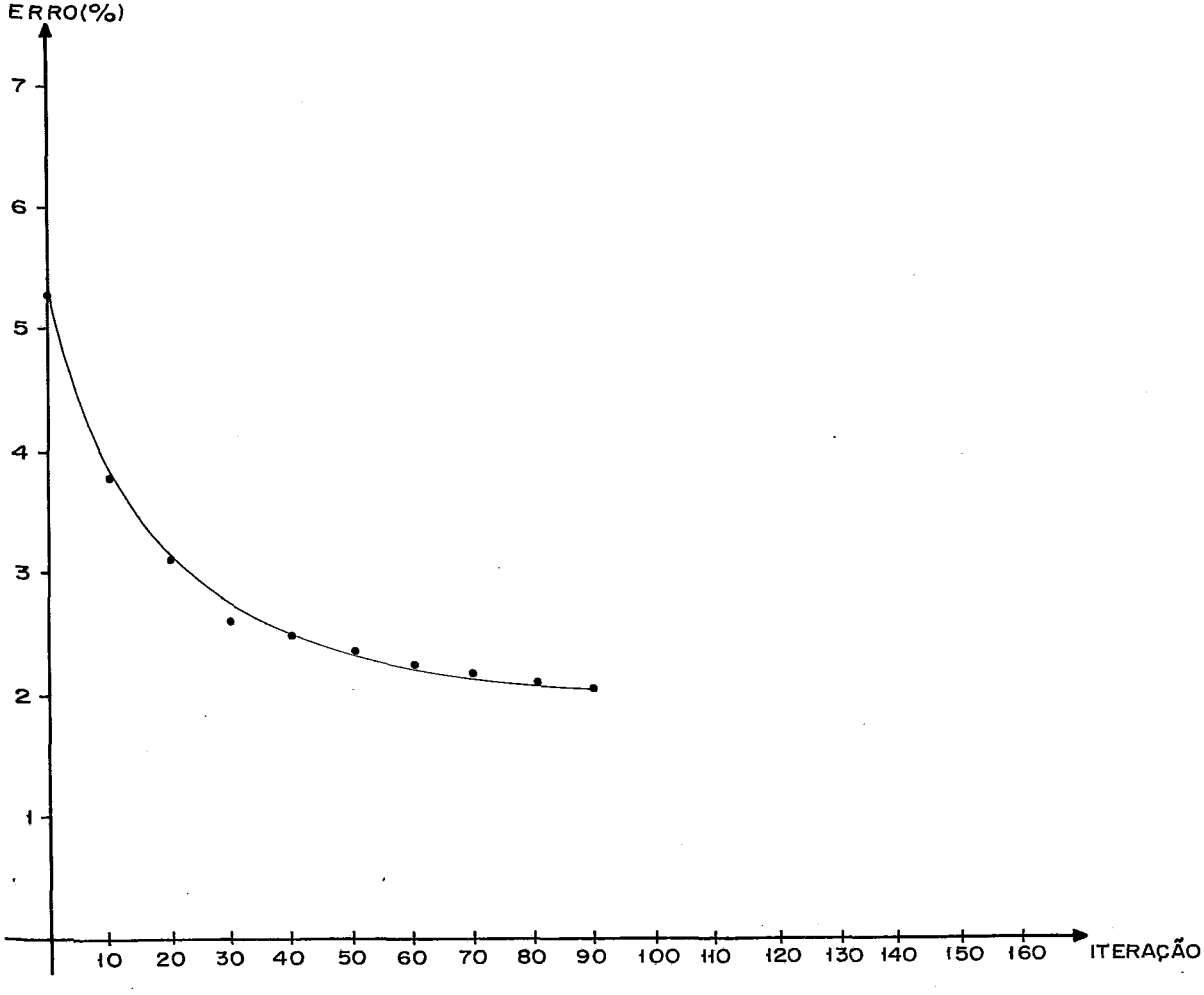
Resultado:

F(x) = 137736144,15

LI = 134925707,00

Erro = 2,04%

Número de iterações = 97



Problema 8.3.3

Função de custo $g_6(v)$ =

728633,53	$0 < v \leq 6000$
969209,97	$6000 < v \leq 12000$
1134846,90	$12000 < v \leq 18000$
1269482,60	$18000 < v \leq 24000$
1385900,60	$24000 < v \leq 30000$
1489954,10	$30000 < v \leq 36000$
1584905,00	$36000 < v \leq 42000$
1672792,20	$42000 < v \leq 48000$
1754990,70	$48000 < v \leq 54000$
1832480,30	$54000 < v \leq 60000$

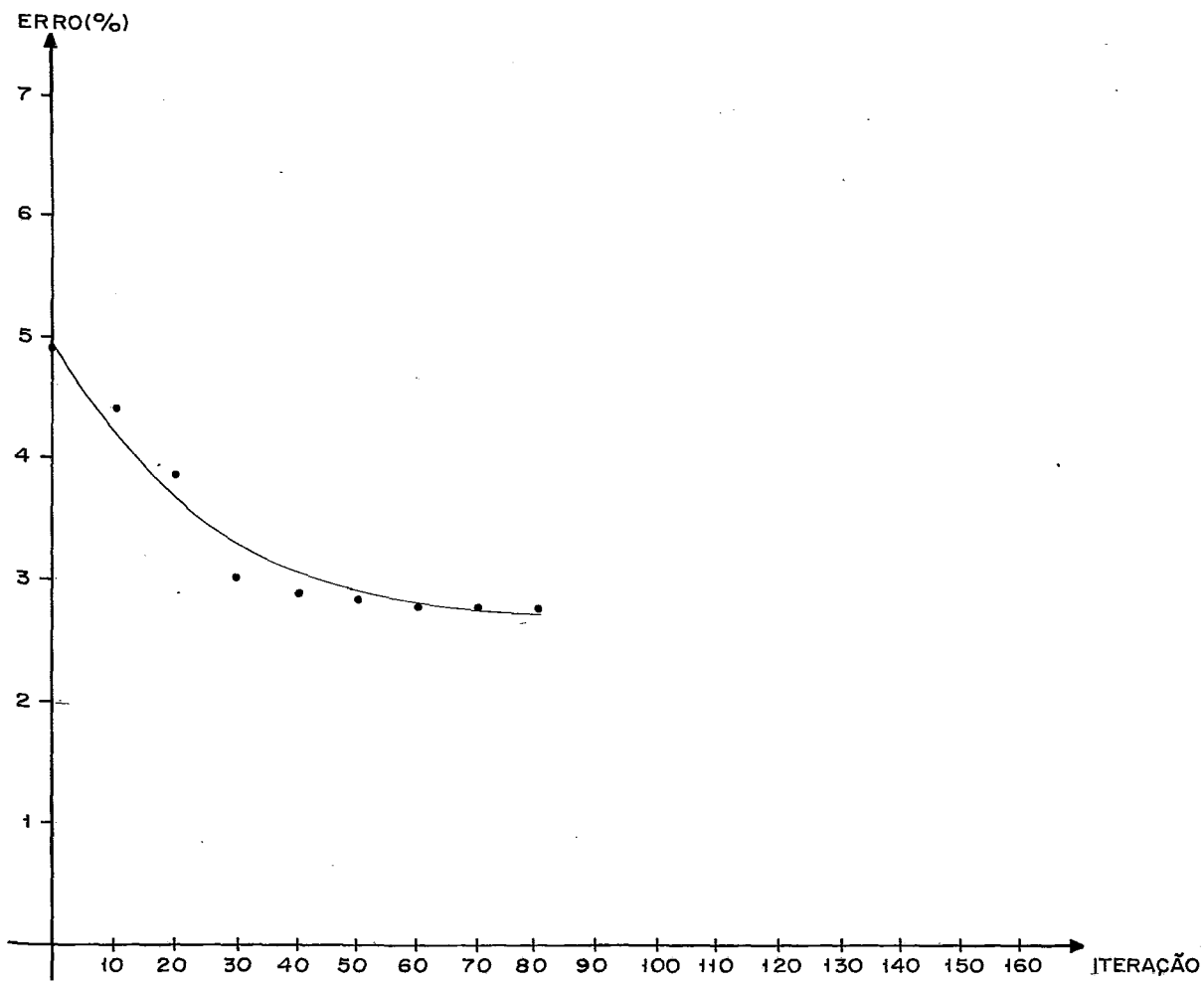
Resultado:

F(x) = 137917375,40

LI = 134028578,00

Erro = 2,82%

Número de iterações = 80



Como vemos, apesar de mais uma vez não podermos chegar a conclusões definitivas, o aumento do número de patamares da função escada, tende a piorar a convergência do algoritmo. Isso fica bastante ressaltado quando se compara o problema 8.3.1 com o problema 8.3.2 quando aumentamos o número de patamares de 1 para 3. Já não ficou tão evidenciado na comparação entre os problemas 8.3.2 e 8.3.3 quando aumentamos o número de patamares de 3 para 10. Neste segundo caso, poderia se esperar uma piora considerável no comportamento do algoritmo, o que não aconteceu. Acreditamos que a justificativa possa estar no fato de que, da forma como construimos as funções escada, à medida em que aumentamos consideravelmente o número de patamares, as aproximações lineares ficaram melhores, isto é, ficaram menos "afastadas" da função escada.

Um fato que deve ser ressaltado nos nossos testes é que apesar de $g_4(v)$ ser igual a $g_1(v)$, a convergência para o problema 8.3.1 ficou bem melhor do que para o problema 8.2.1, mostrando que numa função escada com 1 patamar, deve ser tratada efetivamente como uma função escada e não como uma função côncava.

Analogamente ao que foi feito no item 8.2, resolvemos mais dois problemas onde aumentamos o número de arcos com funções escada para 26 e 39 respectivamente usando desta vez como função de custo a $g_5(v)$. Os resultados foram os que se seguem:

Problema 8.3.4

Custo de transporte dos 13 primeiros arcos e custos de armazenagem expressos por funções do tipo escada (total, 26 arcos com funções do tipo escada)

Função de custo $g_5(v)$ =
$$\begin{cases} 824264,00 & 0 < v \leq 10000 \\ 1248528,10 & 10000 < v \leq 30000 \\ 1672792,20 & 30000 < v \leq 60000 \end{cases}$$

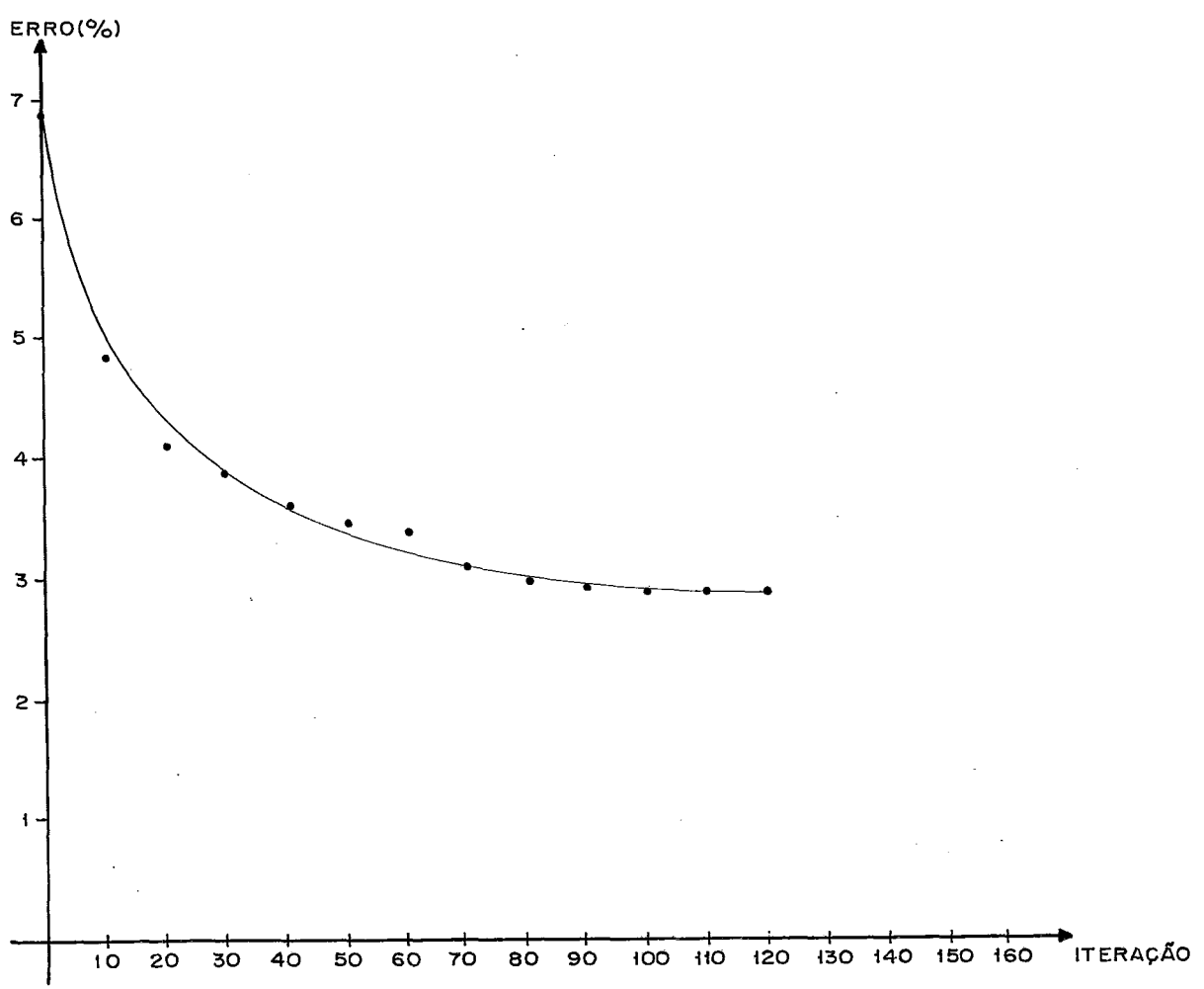
Resultado:

F(x) = 136902758,23

LI = 133009649,00

Erro = 2,84%

Número de iterações = 120



Problema 8.3.5

Custo de transporte dos 26 primeiros arcos e custo de armazenagem expressos por funções do tipo escada (total: 39 arcos com funções do tipo escada)

$$\text{Função de custo } g_5(v) = \begin{cases} 824264{,}00 & 0 < v \leq 10000 \\ 1248528{,}10 & 10000 < v \leq 30000 \\ 1672792{,}20 & 30000 < v \leq 60000 \end{cases}$$

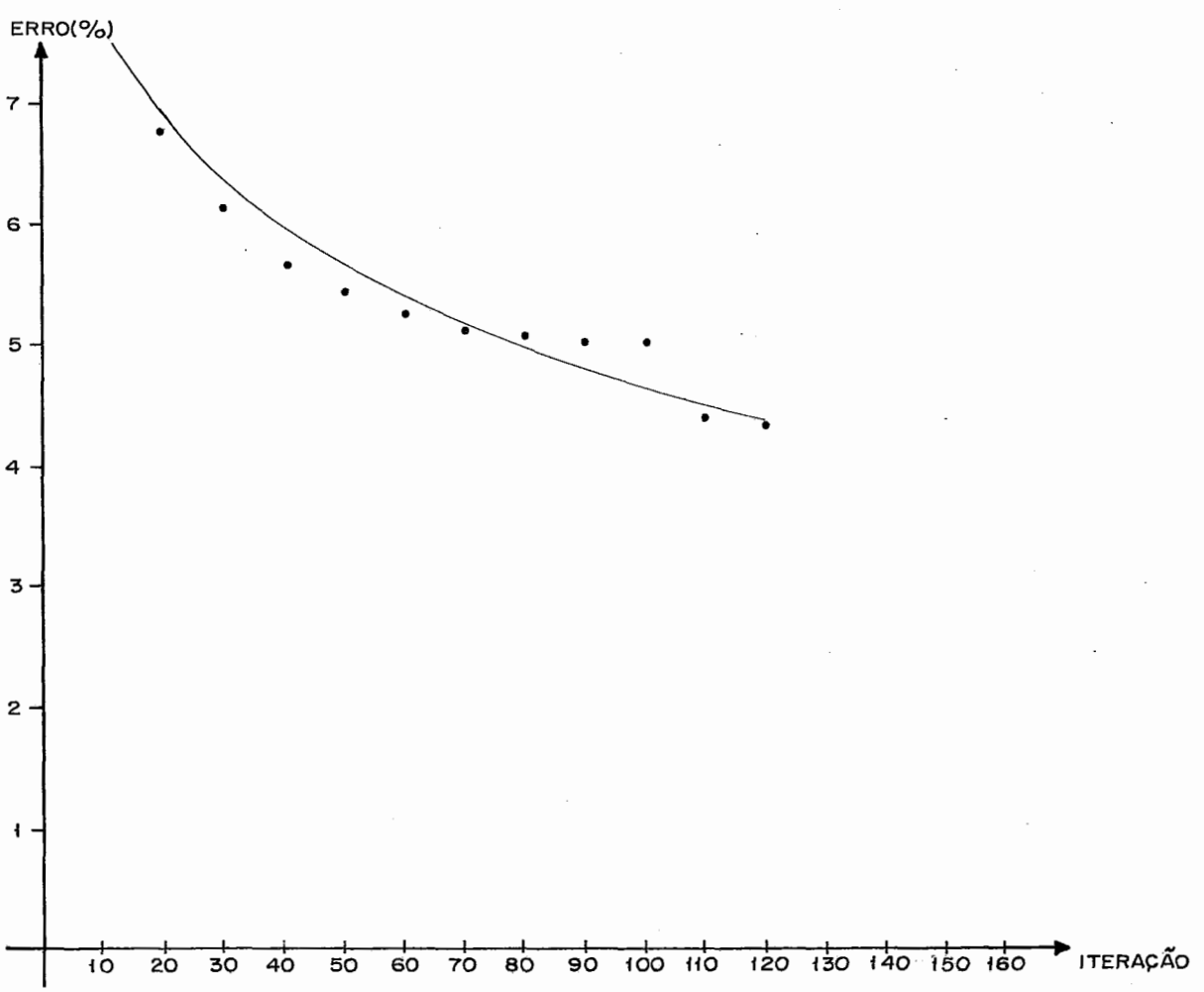
Resultado:

F(x) = 134674687,67

LI = 128847556,00

Erro = 4,33%

Número de iterações = 124



Comparados os resultados dos problemas 8.3.2, 8.3.4 e 8.3.5, verificamos analogamente ao que vimos nos testes com funções côncavas, que, à medida em que aumentamos o número de arcos com funções de custo do tipo escada, piora a velocidade de convergência do algoritmo.

8.4 - TESTE COMPARATIVO

Fizemos um teste comparativo com um dos problemas resolvidos por Monterosso |17| e cujo resultado completo é apresentado em anexo ao seu trabalho. Trata-se da mesma rede que utilizamos nos nossos testes, com função côncava $g(v) = 215250 + 6566{,}25\sqrt{v}$. Apresentamos a seguir o resumo do processamento analisando os resultados e no anexo III o resultado completo do teste por nós realizado.

Problema 8.4.1

Função $g_7(v) = 215250 + 6566,25 \sqrt{v}$

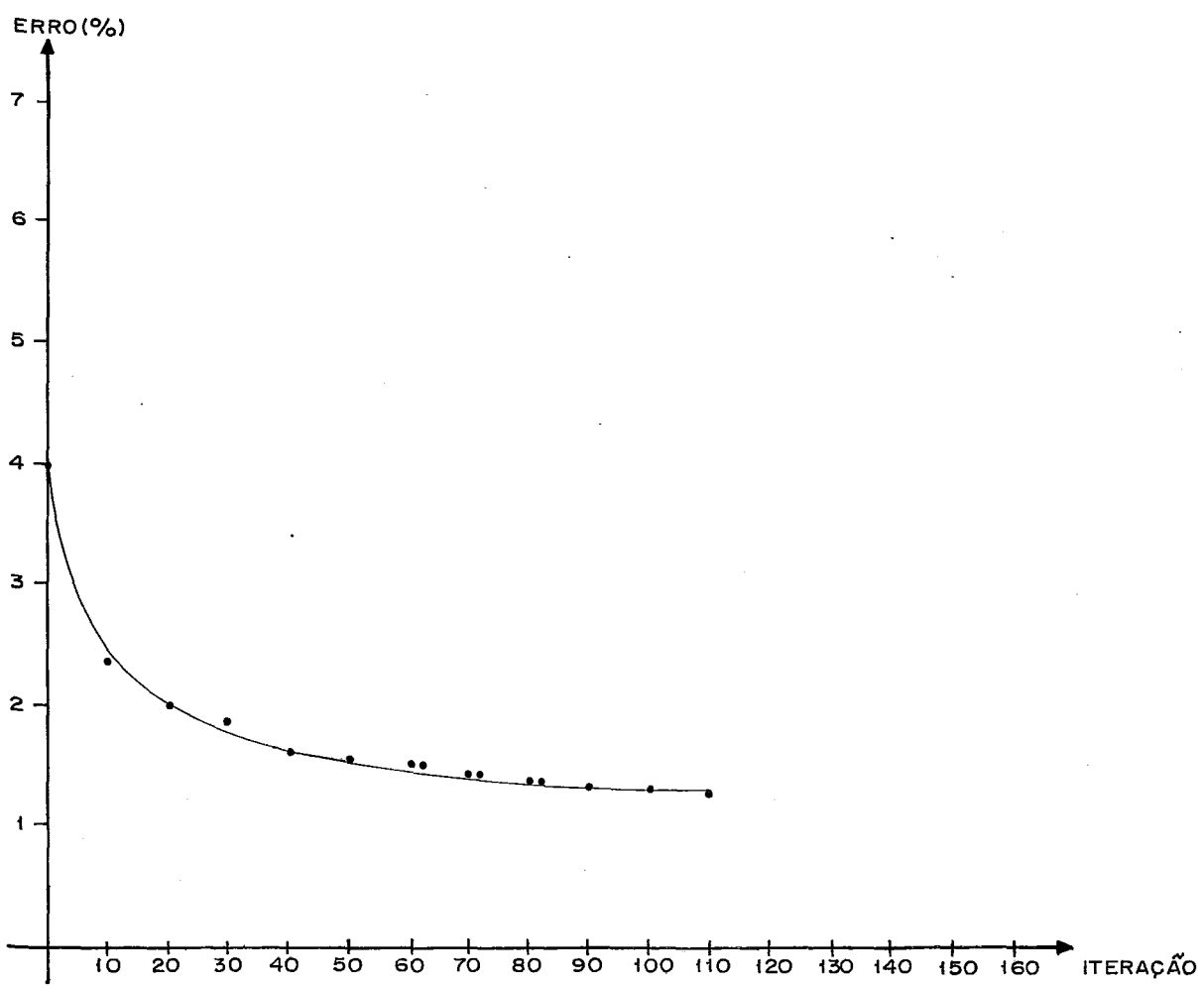
Resultado:

F(x) = 136874908,95

LI = 135089656,00

Erro = 1,30%

Número de iterações = 110



Comparando os resultados com os de Monterosso |[17]| podemos verificar que o nosso algoritmo obteve um valor melhor para F(x) com um erro máximo de 1,30%, com menor tempo de CPU.

A configuração final de armazenagem encontrada foi quase a mesma. No nosso resultado foi acrescentado um armazém em Pedreiras, tendo sido diminuido o fluxo em Vitorino Freire, Bacabal e Santo Antonio dos Lopes. Houve também um pequeno acréscimo no fluxo em Lago da Pedra.

8.5 - COMENTÁRIOS ADICIONAIS

Além do que já foi dito nos itens anteriores, gostaríamos ainda de ressaltar o seguinte:

1. Existe uma tendência muito forte de obtermos performances piores quando o número de arcos com funções de custo não lineares aumenta.

2. O tipo de função de custo usada influi no comportamento do algoritmo, embora não se possa afirmar a priori que tipos de função dão melhores ou piores resultados.

3. Foi importantíssimo termos escolhido um algoritmo para a resolução dos problemas lineares com uma performance boa. Como pode ser visto no resumo do processamento do teste 8.4.1 (vide anexo III), nós chamamos 221 vezes o algoritmo

OUT-OF-KILTER, gastando com isso cerca de 90% do tempo total.

Interessante também é a diferença de tempo entre a primeira chamada do OUT-OF-KILTER com cerca de 12 segundos e a média das demais chamadas que foi de aproximadamente 0.5 segundos, comprovando o que tinhamos previsto na parte teórica deste trabalho.

4. Em alguns testes o tempo médio por iteração foi bem menor do que em outros o que pode ter sido ocasionado por condições particulares destes testes que pioraram o tempo médio utilizado pelo OUT-OF-KILTER.

5. Comparando os testes com funções do tipo escada com aqueles feitos com funções côncavas vemos que a performance é em alguns casos até melhor para as funções tipo escada.

6. Os testes com funções do tipo escada foram realizados com um número de iterações bem menor do que se poderia supor, já que teoricamente o número de iterações poderia no pior caso ser bem maior.

Apesar de não termos feito testes comparativos, isto nos dá a certeza de termos desenvolvido um bom algoritmo para a resolução de problemas com funções do tipo escada.

CAPÍTULO 9

CONCLUSÕES FINAIS

O objetivo básico do nosso trabalho foi o desenvolvimento de um algoritmo baseado no método Branch and Bound, que nos levasse a uma solução ótima exata de um problema de localização, com limitações de capacidade na rede, restrições lineares e funções de custo côncavas ou do tipo escada, além de desenvolvermos um software implementado em computador de fácil manuseio.

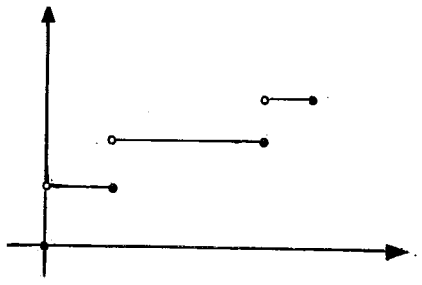
Para atingirmos esse onjetivo, uma das nossas preocupações foi a de apresentar em primeiro lugar um resumo do Branch and Bound, visando um entendimento da filosofia básica nele contida. Apresentamos como ilustração um exemplo simples de uma aplicação à programação linear inteira, acompanhando passo a passo o desenvolvimento do algoritmo até a obtenção da solução ótima. Apresentamos em seguida o Branch and Bound em aplicações a redes de fluxo capacitadas com função objetiva côncava. Nesta parte, fizemos um detalhamento do trabalho de Soland [22], apresentando definições, lemas e teoremas básicos e detalhando os passos do algoritmo proposto. Fizemos em seguida uma adaptação do algoritmo a problemas de localização capacitados com função objetivo do tipo escada. Esta foi uma parte inovadora do nosso trabalho onde, além da apresentação detalhada do algoritmo, fizemos a prova de otimalidade e convergência, além de um estudo da complexidade do algoritmo. Apresentamos então o programa desenvolvido, dando detalhes de linguagem, equipamento usado, roti-

nas, formato das variáveis etc... Mostramos como se faz a entrada de dados e como são os relatórios impressos. Podemos dizer que o programa, da forma como foi desenvolvido e está apresentado, constitui-se num "pacote aberto", possibilitando em caso de necessidade, que seja modificado, e que novos desenvolvimentos possam ser feitos, utilizando-se o seu "fonte". Fizemos com o programa vários testes com o intuito de verificar o comportamento do algoritmo com diferentes funções e com menor ou maior número de funções não lineares, usando basicamente dados dos trabalhos da CIBRAZEN |[4]| e Monterosso |[17]|.

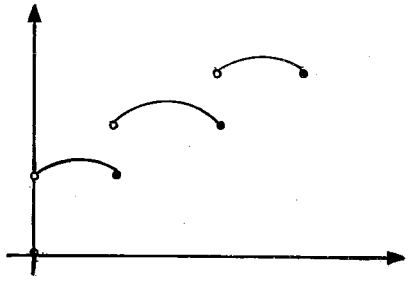
Apesar da utilização do Branch and Bound muitas vezes constituir-se num desafio, havendo dificuldade de se resolver problemas maiores, já que o tempo ou a memória gastos podem tornar a tarefa inviável, conseguimos mostrar que, usando-se tolerâncias muitas das vezes perfeitamente aceitáveis, consegue-se resultados excelentes e acima de tudo confiáveis. Comparados os tempos dos nossos testes da seção 8.1 e da seção 8.4 com os de Monterosso |[17]|, verificamos que estamos obtendo resultados com tempos de resposta bastante bons.

No nosso trabalho usamos o algoritmo para tratamento de funções côncavas com base para os problemas com funções do tipo escada.

Achamos que a evolução natural desse trabalho é em primeiro lugar o tratamento de funções tipo escada com os degraus não obrigatoriamente decrescentes, como mostra a Figura:



Mais interessante ainda e perfeitamente aceitável seria evoluir-se para funções côncavas por partes como mostra a figura:



Estes tipos de funções poderiam ser aplicados quando se tratasse da construção de silos onde cada um tivesse um custo inicial e tivessemos que considerar economia de escala para a construção individual de cada um deles.

De qualquer forma, seja qual for o caminho adotado a partir daqui, consideramos que apesar da quantidade grande de trabalhos publicados a respeito do assunto ele não está esgotado, havendo ainda um vasto campo aberto para estudos.

BIBLIOGRAFIA

1. AGIN, NORMAN - Optimum Seeking With Branch & Bound Principle. Mg. Sc., 13/4: B176-B185, 1966.

2. BALAS, EGON - A Note on the Branch and Bound Principle. Op. Res., 16/2: 442-445, 1968.

3. BAUMOL, WILLIAM J. & WOLFE, PHILIP - A Warehouse Location Problem. Op. Res., 6: 252-263, 1958.

4. _____. Alternativa de Solução para o Sistema de Armazenagem e Meio Ambiente no Estado do Maranhão. CIBRAZEM, 1976.

5. FALK, J. E. & HOROWITZ, J. L. - Critical path Problems with Concave Cost-Time Curves. Mg. Sc., 17/11: 446-455 , 1972.

6. FALK, J. E. & SOLAND, R. M. - An Algorithm for Separable Nonconvex Programming Problems. Mg. Sc., 15/9: 550-569, 1979.

7. FLORIAN, M. & ROBILLARD, P. - An Implicit Enumeration Algorithm for the Concave Cost Network Problem. Mg. Sc., 13/3: 184-193, 1971.

8. GALLO, G.; SANDI, C. & SODINI, C. - An Algorithm for the

Min Concave Cost Flow Problem. European Journal of Op. Res., 4: 248-255, 1980.

9. GOMES, M. A. - Problemas de Fluxo de Custo Mínimo em Redes de Custo Concavo. Tese de M. Sc., São Paulo, UNICAMP; 1981.

10. JOHANSHAHLOU, G. R. - Plant Location Problem. Iran, Tehran, University for Teacher Education, 1978.

11. JONES, A. P. & SOLAND, R. M. - A Branch and Bound Algorithm for Multi-Level Fixed-Charge Problems. Mg. Sc., 16/1: 67-76, 1969.

12. LAND, A. & DOIG, A. - An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28/3: 497-520, 1960.

13. LAWLER, E. L. & WOOD, D. E. - Branch and Bound Methods: a Survey. Op. Res., 14/4: 699-719, 1966.

14. LUENBERGER, D. G. - Introduction to Linear and Nonlinear Programming. EEUU, Addison-Wesley, 1973.

15. MACULAN, Fº. N. - Programação Linear Inteira. Rio, COPPE, PDD 17/78.

16. MITTEN, L. G. - Branch and Bound Methods: General Formula-

tion and Properties. Op. Res., 18/1: 24-34, 1970.

17. MONTEROSSO, C. D. B. - Um Método Heurístico para a Localização e Dimensionamento de Armazéns em Sistemas de Grande Porte Considerando Economias de Escala. Tese M. Sc., Rio, COPPE/UFRJ, 1977.

18. RECH, P. & BARTON, L. G. - A Non-Convex Transportation Algorithm. Application of Mathematical Programming Technics. EEUU, E.M.L. Beale, 1970.

19. ROY, B. - Procedures d'Exploration par Separation et evaluation Progressive. Revue Française d'Informatique et de Recherche Operationelle, 5: 61-90, 1969.

20. SÁ, G. - Branch and Bound and Approximate Solutions to the Capacitated Plant-Location Problem. Op. Res., 17/6:1005-1016, 1969.

21. SALKIN, H. M. - Integer Programming, Londres, Addison-Wesley, 1975.

22. SOLAND, R. M. - Optimal Facility Location With Concave Costs. Op. Res., 22/2: 373-382, 1974.

23. SOLAND, R. M. - An Algorithm for Separable Piecewise Convex Programming Problems. EEUU, Center of Cybernetics Studies - University of Texas, 1971.

24. TAHA, H. - Integer Programming: Theory, Applications, and Computations, New York, Academic Press, 1975.

ANEXO I

LISTAGEM DO PROGRAMA FONTE

```
BEGIN

%  VERSAO 14/03/83

FILE CARTAO (KIND=READER);

FILE IMP (KIND=PRINTER);


%
%               DADOS LIDOS DO CARTAO MESTRE
%
INTEGER
                LISTA,          % IND. SE LISTA OU NAO OS CARTOES DE DADOS
                CONF,           % CONFIGURACAO
                N,              % NUMERO DE NOS
                NP,             % NUMERO DE CENTROS PRODUTORES
                A,              % NUMERO DE ARCOS
                NC,             % NUMERO DE CENTROS CONSUMIDORES
                NF,             % NO. DE FUNCOES NAO LINEARES
                AF;             % NO. DE ARCOS C/ FUN NAO LINARES

REAL            TOLERA;         % TOLERANCIA

ALPHA           PERC,           % PERCENTUAL
                TIPC;           % TIPO DE CARTAO
%
%               VALORES CALCULADOS A PARTIR DO MESTRE
%
INTEGER         NK,             % QUANT. NOS P/ OUT-OF-KILTER
                AK,             % QUANT. ARCOS P/ OUT-OF-KILTER
                AP1,            % LIMITES PARA OS
                AP2,            %    ARCOS ARTIFICIAIS DE
                AC1,            %    PRODUCAO E CONSUMO
                AC2,            %    NO USO DO OUT-OF-KILTER
                TAM,            % TAMANHO DO REGISTRO DE DISCO
                NAB,            % NO. DE ELEMENTOS DA LISTA DE ABERTOS
                NFE;            % NO. DE ELEMENTOS DA LISTA DE FECHADOS
%
%               CAMPOS AUXILIARES
%
REAL            HORAIN,         % HORA DE INICIO
                HORAFIM1,       % HORA DE FIM DA INICIALIZACAO
                HORAFIM;        % HORA DE TERMINO
```

```
%
%                LEITURA DO CARTAO MESTRE
%
READ(CARTAO,<A1,I1,6I4,F10.2,A1,I4>,TIPC,LISTA,CONF,N,NP,A,NC,AF,
          TOLERA,PERC,NF);
IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
THEN
     BEGIN
     WRITE (IMP[SKIP 1 ];
     WRITE (IMP,<"1...5...10...15...20...25...30...35...40...45",
          "...50...55...60...65...70...75...80">);
     WRITE (IMP,</,A1,I1,6I4,F10.2,A1,I4>,TIPC,LISTA,CONF,N,NP,A,NC,
          AF,TOLERA,PERC,NF);
     END;

NK:=N+2;
AK:=A+NP+NC+1;
AP1:=A+1;
AP2:=A+NP;
AC1:=A+NP+1;
AC2:=A+NP+NC;
NAB:=8002;                    % 8000 ITERACOES + 1 + FOLGA
NFE:=8001;                    % 8000 ITERACOES + FOLGA
TAM:=A+NK;                    % TAMANHO DO REGISTRO
IF TIPC ¬= "M" OR LISTA > 7 OR N < (NP+NC) OR
     (PERC = "%" AND TOLERA > 100.00)
     OR AF > A
THEN
     WRITE (IMP,<//,"CARTAO MESTRE INVALIDO OU ERRO NA LEITURA">)
ELSE
     BEGIN
```

```
%-------------------------------------------------%
%          BLOCO PRINCIPAL COM TODAS AS PROCEDURES  %
%               INTERNAS DO PROGRAMA                %
%-------------------------------------------------%

LABEL           FIM;

FILE DISCO(KIND=PACK,TITLE="XX.",FILETYPE=7,MYUSE=OUT,
     MAXRECSIZE=TAM,BLOCKSIZE=TAM*30);



%
%          CAMPOS COMUNS A TODAS AS PROCEDURES
%

INTEGER         K,                      % CONTA OS PASSOS DO ALGORITMO
                VERSAO,                 % VERSAO DO PROGRAMA
                CHALIN,                 % NO. DE CHAMADAS P/ PROG LINEAR
                LI,                     % VALOR DA F.O. LINEAR
                PIOR,                   % INDICA QUAL O PIOR ARCO
                VAZIO,                  % INDICA SE O OUT-OF-KILTER
                                        %    ACHOU SOLUCAO
                INDICE,                 % ARCO COM CUSTO DIFERENTE
                FIMAB,FIMFE,            % INDICAM O PRIMEIRO ITEM
                                        %    DISPONIVEL DAS LISTAS DE
                                        %    ABERTOS E FECHADOS
                INAB,                   % INDICA O INICIO/LISTA ABERTOS
                AB1,AB2;                % INDICE DOS ITENS DA LISTA DE
                                        % ABERTOS GERADOS NA SEPARACAO

REAL            LSF,                    % LIMITE SUPERIOR P/ F.O.
                TOLRE,                  % TOLERANCIA CALCULADA
                XPI,                    % FLUXO NO PIOR ARCO
                TLIN,                   % TEMPO DO PROG. LINEAR
                TLIN1,                  % TEMPO DA 1A. CHAMADA DO P LIN
                TOTPRO,TOTCON;          % TOTAIS DE PRODUCAO E CONSUMO

BOOLEAN         ULTRAPASSOU,            % INDICA SE O LIMITE DE UMA DAS
                                        %    LISTAS FOI ULTRAPASSADO
                GRAVOU;                 % INDICA SE GRAVOU DISCO

ALPHA ARRAY     % LIDOS DOS CARTOES 1 2 E 3

                DEFNOMA[0:A*2],         % NOME DO ARCO
                DEFNOMP[0:NP*2],        % NOME DO CENTRO-PRODUTOR
                DEFNOMC[0:NC*2];        % NOME DO CENTRO CONSUMIDOR

REAL ARRAY      % LIDOS DOS CARTOES 1 2 E 3


                DEFARCO[0:AK*4],        % NOS INICIAL, FINAL, INDICES
                TARC[0:A],              % TIPO DE ARCO
                NFUN[0:A],              % NO./TIPO DA FUNCAO CUSTO
                L[0:AK],                % LIMITE INFERIOR DE FLUXO
                U[0:AK],                % LIMITE SUPERIOR DE FLUXO
                C[0:AK],                % CUSTO ATUAL (CORRESP. AO COEF.
                                        % ANGULAR ATUAL DAS F. LINEARES)
                IROT[0:AF],             % INDICES DE ROTACAO

                % REGISTRO DO DISCO

                REG[0:TAM-1],           % GUARDA X E W NO DISCO
```

```
                % CAMPOS AUXILIARES

                PRIPAT[0:NF+1],      % PONT P/ PRIM PATAM.EM DEFPAT
                DEFPAT[0:65000],     % U,A,B,C,D,E DE CADA PATAMAR
                NUMFUN[0:NF],        % NO. DAS FUNCOES
                PRIPATAR[0:AF],      % PONT. P/ PRIM PAT DE CADA ARCO
                TIPOF[0:NF],         % TIPO DAS FUNCOES
                TIPOFA[0:AF],        % TIPO DAS FUNCOES DOS ARCOS
                DEFNPO[0:NK*2],      % INDICES AUXILIARES
                X[0:AK],             % FLUXO CALCULADO
                LSX[0:A],            % FLUXO CORRESP. A LSF
                DEFXAB[0:65000],     % FLUXO NOS ABERTOS
                DEFWAB[0:65000],     % VAR. DUAIS DOS ABERTOS
                DEFAB[0:NAB*8],      % LISTA DE ABERTOS- PIOR
                                     % XPI, ARC, L,U,C,B,PONT P/PAI
                                     % (FECHADO)
                DEFFE[0:NFE*6],      % LISTA DE FECHADOS: ARCO, L, U,
                                     % C, B, PONT.P/PAI
                SALVAL[0:AF],        % SALVO L
                SALVAU[0:AF],        % SALVO U
                ALFA[0:A],           % COEF. ANGULAR DAS F. LINEARES
                BETA[0:A],           % COEF. LINEAR DAS F. LINEARES
                B[0:A];              % COEF. LINEAR ATUAL
LONG ARRAY      LISTAB[0:NAB+2];     % LISTA DE ABERTOS - LI + LINK

                %
                %  ATENCAO . . . DEFINES PARA OS ARRAYS DE MAIS
                %                DE UMA DIMENSAO

DEFINE          NOMA[I,J] = DEFNOMA[(I-1)*2+J]#,
                NOMP[I,J] = DEFNOMP[(I-1)*2+J]#,
                NOMC[I,J] = DEFNOMC[(I-1)*2+J]#,
                ARCO[I,J] = DEFARCO[(I-1)*4+J]#,
                NPO [I,J] = DEFNPO [(I-1)*2+J]#,
                XAB [I,J] = DEFXAB [(I-1)*A+J]#,
                WAB [I,J] = DEFWAB[(I-1)*NK+J]#,
                RX[I]     = REG[I-1]#,
                W[I]      = REG[A+I-1]#,  % ATENCAO: E W MESMO
                AB  [I,J] = DEFAB[(I-1)*8+(J-1)]#,
                VALAB[I]  = LISTAB[I].[47:28]#,
                PROXAB[I] = LISTAB[I].[19:20]#,
                PAT[I,J]  = DEFPAT[(I-1)*6+J]#,
                FE  [I,J] = DEFFE  [(I-1)*6+J]#;
```

```
PROCEDURE LER;
%
%                LEITURA DOS CARTOES DE DADOS
%                ----------------------------
%

BEGIN

      LABEL             ERRO1,FIMLER;


      INTEGER           TCART,NCART,          % TIPO E NUMERO DO CARTAO
                        TFUN,                 % TIPO DE FUNCAO
                        NPAT,                 % NO. DE PATAMARES
                        CPAT,                 % NO. DO PATAMAR
                        NFUNC,NFUNP,          % NUM DAS FUNCOES
                        I,J,K;                % INDEXADORES

      REAL
                        UANT,                 % U ANTERIOR
                        INDICE;               % INDICE DE ROTACAO

      REAL ARRAY
                        LI[0:NK],LJ[0:NK];    % INDICES AUXILIARES


PROCEDURE CRIA;                 % INTERNA A PROCEDURE LER
%
%                       CRIA AS LISTAS ARCO E NPO
%
BEGIN
      INTEGER           II,JJ,M,N;            % INDEXADORES

II:=ARCO[I,1];
JJ:=ARCO[I,2];
IF NPO[II,1] = 0
THEN
      NPO[II,1]:=I;
IF NPO[JJ,2] = 0
THEN
      NPO[JJ,2]:=I;
M:=LI[II];
IF M ¬= 0
THEN
      ARCO[M,3]:=I;
N:=LJ[JJ];
IF N ¬= 0
THEN
      ARCO[N,4]:=I;
LI[II]:=I;
LJ[JJ]:=I;
END;                    % FIM DA PROCEDURE CRIA

%
%           LEITURA DOS CARTOES TIPO 0 (FUNCAO / PATAMAR)
%
K:=1;
FOR I:=1 STEP 1 UNTIL NF
DO
      BEGIN
      READ (CARTAO,<I1,I5,I2>,TCART,NFUNC,NPAT) [ERRO1:ERRO1:ERRO1];
      IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
      THEN
            WRITE (IMP,<I1,I5,I2>,TCART,NFUNC,NPAT);
```

```
NUMFUN[I]:=NFUNC;
IF NPAT = 0
THEN
     BEGIN
     NPAT:=1;
     TIPOF[I]:=1;
     END;
PRIPAT[I]:=K;
UANT:=0;
FOR J:=1 STEP 1 UNTIL NPAT
DO
     BEGIN
     READ (CARTAO,<I1,I5,I2,X2,6F10.2>,TCART,NFUNP,CPAT,
          PAT[K,1],PAT[K,2],PAT[K,3],PAT[K,4],PAT[K,5],
          PAT[K,6]) [ERRO1:ERRO1:ERRO1];
     IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
     THEN
          WRITE (IMP,<I1,I5,I2,X2,6F10.2>,TCART,NFUNP,CPAT,
               PAT[K,1],PAT[K,2],PAT[K,3],PAT[K,4],PAT[K,5],
               PAT[K,6]);
     IF TCART ¬= 0
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,
               "   TIPO NAO E 0">,TCART,NFUNP);
          GO TO FIM
          END;
     IF NFUNC  ¬= NFUNP
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"-",I2,
               "   NO. DA FUNCAO NAO E IGUAL AO DO ",
               "CARTAO ANTERIOR">,TCART,NFUNP,CPAT);
          GO TO FIM
          END;
     IF CPAT ¬= J AND TIPOF[I] = 0
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"-",I2,
               "   NO. DO CARTAO PATAMAR NAO E SEQUENCIAL ",
               "UNITARIO">,TCART,NFUNP,CPAT);
          GO TO FIM
          END;
     IF PAT[K,1] = 0
          OR PAT[K,1] <= UANT
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"-",I2,
               "   LIMITE SUPERIOR DO PATAMAR INVALIDO">,
               TCART,NFUNP,CPAT);
          GO TO FIM
          END;
     IF PAT[K,4] = 0
     THEN
          PAT[K,4]:=1;
     IF PAT[K,6] = 0
     THEN
          PAT[K,6]:=1;
     UANT:=PAT[K,1];
     K:=K+1
     END;
END;
PRIPAT[I]:=K;      % CUIDADO: ESTA CORRETO

     LEITURA DOS CARTOES TIPO 1 (PRODUCAO)
```

```
J:=1;
FOR I:=AP1 STEP 1 UNTIL AP2
DO
      BEGIN
      READ (CARTAO, <I1,I5,I4,F10.2,2A6>,TCART,NCART,ARCO[I,2],L[I],
           NOMP[J,1],NOMP[J,2]) [ERRO1:ERRO1:ERRO1];
      IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
      THEN
           WRITE (IMP, <I1,I5,I4,F10.2,2A6>,TCART,NCART,ARCO[I,2],L[I],
                NOMP[J,1],NOMP[J,2]);
      IF TCART ¬= 1
      THEN
           BEGIN
           WRITE (IMP,</,"ERRO NO CARTAO ",I1,"-",I5,"   TIPO NAO E 1">,
                TCART,NCART);
           GO TO FIM
           END;
      IF ARCO[I,2] = 0 OR  ARCO[I,2] > N
      THEN
           BEGIN
           WRITE (IMP,</,"ERRO NO CARTAO ",I1,"-",I5,"   NO DE PRODUCA",
                "O INVALIDO">,TCART,NCART);
           GO TO FIM
           END;
      ARCO[I,1]:=N+1;
      U[I]:=L[I];
      TOTPRO:=TOTPRO+L[I];
      CRIA;
      J:=J+1;
      END;
%
%          LEITURA DOS CARTOES TIPO 2 (ARCOS)
%
FOR I:=1 STEP 1 UNTIL A
DO
      BEGIN
      READ (CARTAO,<I1,I5,2I4,2I1,X4,3F10.2,I5,2A6,F5.2>,TCART,NCART,
           ARCO[I,1],ARCO[I,2],TARC[I],TFUN,L[I],U[I],C[I],
           NFUN[I],NOMA[I,1],NOMA[I,2],INDICE) [ERRO1:ERRO1:ERRO1];
      IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
      THEN
           WRITE (IMP,<I1,I5,2I4,2I1,X4,3F10.2,I5,2A6,F5.2>,TCART,
                NCART,ARCO[I,1],ARCO[I,2],TARC[I],TFUN,L[I],U[I],C[I],
                NFUN[I],NOMA[I,1],NOMA[I,2],INDICE);
      IF TCART ¬= 2
      THEN
           BEGIN
           WRITE (IMP,</,"ERRO NO CARTAO ",I1,"-",I5,"   TIPO NAO E 2">,
                TCART,NCART);
           GO TO FIM
           END;

      IF ARCO[I,1] = 0 OR ARCO[I,2] = 0 OR ARCO[I,1] > N OR
           ARCO[I,2] > N
      THEN
           BEGIN
           WRITE (IMP,</,"ERRO NO CARTAO ",I1,"-",I5,"   ORIGEM/DESTIN",
                "O INVALIDOS">,TCART,NCART);

           GO TO FIM
           END;
      IF (I <= AF AND TFUN = 0) OR (I > AF AND TFUN = 1)
      THEN
           BEGIN
           WRITE(IMP,</,"ERRO NO CARTAO ",I1,"-",I5,
             "   ERRO NO TIPO DE FUNCAO OU NO MESTRE (NO. DE ARCOS NAO ",
```

```
          "LINEARES)">,TCART,NCART);
          GO TO FIM;
          END;
     IF TARC[I] > 1 OR TFUN > 1 OR (TFUN = 0 AND (C[I] = 0 OR
          INDICE ¬= 0 OR
          NFUN[I] ¬= 0)) OR (TFUN = 1 AND C[I] ¬= 0)
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"   PARAMETRO",
               ", PENALIDADE, N.FUNCAO OU IND. ROT INVALIDO">,
               TCART,NCART);
          GO TO FIM
          END;
     IF L[I] > U[I] OR (I <= AF AND L[I] ¬= 0)
     THEN
          BEGIN
          WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"   LIMITES",
               " INVALIDOS">,TCART,NCART);
          GO TO FIM
          END;
     IF INDICE = 0  % ATENCAO   E AQUI MESMO
     THEN
          INDICE:=1.0;
     IF I <= AF
     THEN
          BEGIN
          J:=0;
          DO
               J:=J+1
          UNTIL J = NF OR NFUN[I] = NUMFUN[J];
          IF NFUN[I] NEQ NUMFUN[J]
          THEN
               BEGIN
               WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,
                    "   A FUNCAO ",I5," NAO FOI DEFINIDA">,TCART,NCART
                    ,NFUN[I]);
               GO TO FIM
               END;
          K:=PRIPAT[J+1] - 1;
          IF PAT[K,1] ¬= U[I]
          THEN
               BEGIN
               WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"   LIMITE ",
                    "SUPERIOR INVALIDO">,TCART,NCART);
               GO TO FIM
               END;
          TIPOFA[I]:=TIPOF[J];
          PRIPATAR[I]:=PRIPAT[J];
          IROT[I]:=INDICE;
          L[I]:=L[I]*INDICE;
          U[I]:=U[I]*INDICE;
          SALVAL[I]:=L[I];
          SALVAU[I]:=U[I];
          END;
     ALFA[I]:=C[I];
     CRIA;
     END;
%
%          LEITURA DOS CARTOES TIPO 3 (CONSUMO)
%
J:=1;
FOR  I:=AC1 STEP 1 UNTIL AC2
DO
     BEGIN
     READ (CARTAO,<I1,I5,I4,F10.2,2A6>,TCART,NCART,ARCO[I,1],L[I],
          NOMC[J,1],NOMC[J,2]) [ERRO1:ERRO1:ERRO1];
```

```
      IF LISTA = 2 OR LISTA = 3 OR LISTA = 6 OR LISTA = 7
      THEN
           WRITE (IMP,<I1,I5,I4,F10.2,2A6>,TCART,NCART,ARCO[I,1],L[I],
                NOMC[J,1],NOMC[J,2]);
      IF TCART ¬= 3
      THEN
           BEGIN
           WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"   TIPO NAO E 3">,
                TCART,NCART);
           GO TO FIM
           END;
      IF ARCO[I,1] = 0 OR ARCO [I,1] > N
      THEN
           BEGIN
           WRITE (IMP,<//,"ERRO NO CARTAO ",I1,"-",I5,"  NO DE CONSUMO",
                " INVALIDO">,TCART,NCART);
           GO TO FIM
           END;
      ARCO[I,2]:=N+2;
      U[I]:=L[I];
      TOTCON:=TOTCON+L[I];
      CRIA;
      J:=J+1;
      END;
%
%          TESTE DE TOTAIS DE PRODUCAO E CONSUMO
%
IF  TOTPRO ¬= TOTCON
THEN
      BEGIN
      WRITE (IMP,<//,"ERRO NOS CARTOES TIPO 1 E 3 - TOTAL DA PRODUCAO ",
           F15,2," DIFERENTE DO TOTAL DE CONSUMO ",F15,2>,TOTPRO,TOTCON);

      GO TO FIM
      END;
%
%          CRIACAO DO ARCO DE RETORNO P/ O OUT-OF-KILTER
%
ARCO[AK,1]:=N+2;
ARCO[AK,2]:=N+1;
L[AK]:=TOTPRO;
U[AK]:=TOTPRO;
I:=AK;
CRIA;
GO  TO FIMLER;
%
%          IMPRESSAO DO ERRO DE LEITURA
%
ERRO1:
WRITE (IMP,<//,"CARTAO ",I1,"-",I5,"     ERRO DE LEITURA OU FALTAM ",
           "CARTOES TIPO 1,2 OU 3">,TCART,NCART);
GO  TO FIM;
FIMLER:
END;           % FIM DA PROCEDURE LER
```

```
PROCEDURE KILTER;
%
%                    OUT-OF-KILTER
%                    -------------
%

BEGIN

      INTEGER        K,KT;                 % INDEXADOR P/ ARCOS


      REAL           HIN,HFIM,             % HORA DE INICIO E TERMINO
                     QBARRA;               % AUXILIAR P/ QBAR[K]

      REAL ARRAY
                     ESTADO [0:AK],        % ESTADO DO ARCO (1 A 9)
                     QBAR[0:AK];           % QBAR= C[K] + Z[K] = C[K] +
                                           % W[I] - W[J] (EMBORA ESTRANHO,
                                           %      DEVE SER DEIXADO ASSIM)
      LABEL RETORNO;

PROCEDURE STACALC;
%
%          CALCULO DO ESTADO DO ARCO K (INTERNA A OUT-OF-KILTER)
%

BEGIN

      INTEGER        I,J;                  % INDEXADORES



I:=ARCO[K,1];
J:=ARCO[K,2];
QBARRA:=QBAR[K]:=C[K]+W[I]-W[J];           % ATENCAO]]- ESTA CORRETO
IF QBARRA > 0
THEN
      BEGIN
      IF X[K] = L[K]
      THEN
           ESTADO[K]:=1                    % L - EM KILTER
      ELSE
           BEGIN
           IF X[K] < L[K]
           THEN
                ESTADO[K]:=4               % L1 - OUT-OF-KILTER
           ELSE
                ESTADO [K]:=7              % L2 - OUT-OF-KILTER
           END
      END
ELSE
      BEGIN
      IF QBARRA = 0
      THEN
           BEGIN
           IF L[K] <= X[K] AND X[K] <= U[K]
           THEN
                ESTADO [K]:=2              % B - EM KILTER
           ELSE
                BEGIN
                IF X[K] < L[K]
                THEN
                     ESTADO [K]:=5         % B1 - OUT-OF-KILTER
```

```
                ELSE
                    ESTADO[K]:=8          % B2 - OUT-OF-KILTER
                END
            END
        ELSE
            BEGIN
            IF X[K] = U[K]
            THEN
                ESTADO[K]:=3              % K - EM KILTER
            ELSE
                BEGIN
                IF X[K] < U[K]
                THEN
                    ESTADO[K]:=6          % K1 - OUT-OF-KILTER
                ELSE
                    ESTADO[K]:=9          % K2 - OUT-OF-KILTER
                END;
            END;
        END;
END;        % FIM DA PROCEDURE STACALC


PROCEDURE ROTULAT;
%
%           ROTINA DE ROTULACAO (INTERIOR A OUT-OF-KILTER)
%

BEGIN

        INTEGER         ICIC,KM,P,J,D,Y,Z,  % INDEXADORES

                        M,N,                %
                        FONTE,META;         % INICIO E FIM DOS CICLOS

        REAL            DELTA,              % VALOR A SOMAR NAS VAR. DUAIS
                        PSI;                % VALOR A SOMAR NAS VAR. PRIMAIS

        REAL ARRAY
                        ROTULO[0:NK],       % ROTULO DO NO NK
                        ROTULADOS[0:NK];    % INDICA QUEM FOI ROTULADO
%
%           TESTE DO ESTADO PARA SABER SE AUMENTA OU
%             DIMINUI O FLUXO NO NO KT

IF ESTADO [KT] < 7              % O FLUXO DEVE AUMENTAR NO ARCO KT
THEN
        BEGIN
        META:=ARCO[KT,1];
        FONTE:=ARCO[KT,2];
        ROTULO[FONTE]:=KT;      % ROTULACAO DO
        ROTULADOS[1]:=FONTE;    %   PRIMEIRO NO
        END
ELSE                            % O FLUXO DEVE DIMINUIR NO ARCO KT
        BEGIN
        META:=ARCO[KT,2];
        FONTE:=ARCO[KT,1];
        ROTULO[FONTE]:=-KT;     % ROTULACAO DO
        ROTULADOS[1]:=FONTE;    %   PRIMEIRO NO
        END;
%
%           ROTULACAO DOS DEMAIS NOS, ATE QUE META SEJA ROTULADO
%
J:=1;
ICIC:=1;
%
%           LOOPING DE VARREDURA
```

```
%
WHILE (ROTULADOS[ICIC] ¬= META AND ROTULADOS [J] ¬= 0)
DO
      BEGIN          % PESQUISA ARCOS EMERGENTES
      Y:=ROTULADOS[J];
      K:=NPO[Y,1];
      WHILE (K ¬=0 AND ROTULADOS[ICIC] ¬= META)
      DO
            BEGIN
            D:=ARCO[K,2];
            IF ROTULO[D] = 0
            THEN
                  BEGIN
                  IF ESTADO[K] = 4 OR (QBAR[K] <= 0 AND X[K] < U[K])
                  THEN
                        BEGIN
                        ICIC:=ICIC+1;
                        ROTULO[D]:=K;
                        ROTULADOS[ICIC]:=D;
                        END;
                  END;
            K:=ARCO[K,3];
            END;
      Z:=ROTULADOS[J];          % PESQUISA ARCOS INCIDENTES
      K:=NPO[Z,2];
      WHILE (K ¬= 0 AND ROTULADOS[ICIC] ¬= META)
      DO
            BEGIN
            D:=ARCO[K,1];
            IF ROTULO[D] = 0
            THEN
                  BEGIN
                  IF ESTADO[K] = 9 OR (QBAR[K] >= 0 AND X[K] > L[K])
                  THEN
                        BEGIN
                        ICIC:=ICIC+1;
                        ROTULO[D]:=-K;
                        ROTULADOS[ICIC]:=D;
                        END;
                  END;
            K:=ARCO[K,4];
            END;
      J:=J+1;
      END;
%
%           TESTE PARA SABER SE VOU OU NAO PARA A FASE DUAL
%
IF ROTULADOS[J] = 0
THEN
      BEGIN          % OCORREU NON-BREAKTHROUGH (DUAL)
      DELTA:=1@50;
      %
      %     TESTO OS ARCOS, SE VAO DE X PARA X-BARRA OU VICE-VERSA
      %
      FOR K:=1 STEP 1 UNTIL AK
      DO
            BEGIN
            QBARRA:=QBAR[K];
            M:=ARCO[K,1];
            N:=ARCO[K,2];
            IF ROTULO[M] ¬=0 AND ROTULO[N] = 0 AND QBARRA > 0 AND
                  X[K] <= U[K]
            THEN
                  BEGIN
                  IF QBARRA < DELTA
                  THEN
```

```
                DELTA:=QBARRA
            END;
        IF ROTULO[M] = 0 AND ROTULO[N] ¬= 0 AND QBARRA < 0 AND
            X[K] >= L[K]
        THEN
            BEGIN
            IF -QBARRA < DELTA
            THEN
                DELTA:=-QBARRA;
            END;
        END;
    %
    %     TESTO SE PROBLEMA E IMPOSSIVEL
    %
    IF DELTA < 1@50
    THEN
        BEGIN
        FOR K:=1 STEP 1 UNTIL NK        % SOMO DELTA AS VARIAVEIS DUAIS
        DO
            BEGIN
            IF ROTULO[K] = 0            % ATENCAO!! - EMBORA NAO PARECA,
                                        %   ESTA CORRETO (ESTA RELACIO-
                                        %   NADO COM O CALCULO DE QBAR)
            THEN
                W[K]:=W[K]+DELTA;
            END;
        FOR K:=1 STEP 1 UNTIL AK        % RECALCULO O ESTADO DE TODOS
        DO                              % OS ARCOS
            STACALC;
        END
    ELSE
        BEGIN
        VAZIO:=1;               % CONJ. SOL. VIAVEIS VAZIO
        GO TO RETORNO
        END
    END
ELSE
    BEGIN               % OCORREU BREAKTHROUGH (PRIMAL)
    P:=FONTE;
    PSI:=1@50;
    DO                          % FACO O CICLO PROCURANDO O MENOR PSI
        BEGIN
        K:=ROTULO[P];
        IF K < 0
        THEN
            BEGIN
            K:=-K;
            IF QBAR[K] < 0
            THEN
                PSI:=MIN(PSI,X[K]-U[K])
            ELSE
                PSI:=MIN(PSI,X[K]-L[K]);
            P:=ARCO[K,2]
            END
        ELSE
            BEGIN
            IF QBAR[K] > 0
            THEN
                PSI:=MIN(PSI,L[K]-X[K])
            ELSE
                PSI:=MIN(PSI,U[K]-X[K]);
            P:=ARCO[K,1]
            END
        END
    UNTIL P = FONTE;
    DO                          % FACO O CICLO INCREM. OU DECREM. PSI
```

```
            BEGIN
            K:=ROTULO[P];
            IF K < 0
            THEN
                  BEGIN
                  K:=-K;
                  X[K]:=X[K]-PSI;
                  P:=ARCO[K,2];
                  END
            ELSE
                  BEGIN
                  X[K]:=X[K]+PSI;
                  P:=ARCO[K,1];
                  END;
            STACALC;              % FACO O CALCULO DO ESTADO DO ARCO K
            END
      UNTIL P = FONTE;
      END;
END;        % FIM DA PROCEDURE ROTULAT


%
%                     CORPO DO OUT-OF-KILT)R
%
HIN:=TIME(2);
VAZIO:=0;
IF INDICE = 0         % PRIMEIRA CHAMADA DA OUT-OF-KILTER
THEN
      BEGIN
      FOR K:=1 STEP 1 UNTIL AK            % CALCULO DO ESTADO
      DO                                  % DE TODAS OS ARCOS
            STACALC;
      FOR KT:=1 STEP 1 UNTIL AK
      DO
            BEGIN
            WHILE ESTADO[KT] > 5
            DO
                  ROTULAT
            END
      END
ELSE                  % DEMAIS CHAMADAS DA OUT-OF-KILTER COM INDICE
      BEGIN           %   INDICANDO O ARCO COM CUSTO DIFERENTE
      K:=KT:=INDICE;
      STACALC;
      IF ESTADO[KT] > 5
      THEN
            BEGIN
            FOR K:=1 STEP 1 UNTIL AK      % CALCULO DO ESTADO
            DO                            % DE TODOS OS ARCOS
                  STACALC;
            WHILE ESTADO[KT] > 5
            DO
                  ROTULAT;
            END;
      END;
RETORNO:
HFIM:=TIME(2);
IF TLIN1 = 0
THEN
      TLIN1:=HFIM-HIN
ELSE
      BEGIN
      TLIN:=TLIN+(HFIM-HIN);
      CHALIN:=CHALIN+1;
      END;
END;        % FIM DO OUT-OF-KILTER
```

```
REAL PROCEDURE CUSTO(XI,I);
%
%               CALCULO DO VALOR DE F[I] NO PONTO X[I]
%               --------------------------------------
%
      VALUE           XI;                   % ATENCAO: P/ IND. ROTACAO

      INTEGER
                      I;                    % NUM. DA FUNCAO
      REAL
                      XI;                   % PTO. X[I]
BEGIN

      INTEGER
                      J;                    % INDEXADOR


XI:=XI/IROT[I];       % ATENCAO: DIVIDO PELO INDICE DE ROTACAO
IF XI = 0
THEN
      CUSTO:=0
ELSE
      BEGIN
      J:=PRIPATAR[I] - 1;
      DO
           J:=J+1
      UNTIL XI <= PAT[J,1];
      CUSTO:= PAT[J,2] + (PAT[J,3] * XI ** PAT[J,4]) +
                                          (PAT[J,5] * XI ** PAT[J,6]);
      END;
END;                  % FIM DA PROCEDURE CUSTO
```

```
PROCEDURE CALCLIM;
%
%                CALCULO DE LI, LSF, LSX, PIOR ARCO E FLUXO NO PIOR
%                ---------------------------------------------------
%


BEGIN

      INTEGER          I;                      % INDEXADOR


      REAL             DIFUN,                  % DIFERENCA ENTRE F CONC. E LIN.
                       LS,                     % VALOR DA F.O. CONCAVA
                       ZI,                     % VALOR DA F.O. LINEAR
                       LSA,LIA,LDA;            % AUXILIARES NO CALCULO DE DIFUN

%
%           CALCULO DE LS, LI E PIOR
%
FOR  I:=1 STEP 1 UNTIL A
DO
      BEGIN
      LIA:=C[I]*X[I]+B[I];
      IF I > AF
      THEN
            LSA:=LIA
      ELSE
            BEGIN
            LSA:=CUSTO(X[I],I);
            LDA:=LSA-LIA;
            IF LDA > DIFUN
            THEN
                  BEGIN
                  DIFUN:=LDA;
                  PIOR:=I;
                  XPI:=X[I];
                  END;
            END;
      ZI:=ZI+LIA;
      LS:=LS+LSA;
      END;
LI:=ZI;
%
%           CALCULO DE LSF E LSX
%
IF  LS < LSF
THEN
      BEGIN
      LSF:=LS;
      REPLACE POINTER(LSX) BY POINTER(X)  FOR A+1 WORDS;
      END;
END;            % FIM DA PROCEDURE CALCLIM
```

```
PROCEDURE OBTER;

%
%                    OBTEM OS VALORES DA LISTA DE ABERTOS OU DO DISCO
%
BEGIN
IF  INAB*A > 65000 OR INAB*NK > 65000
THEN
      BEGIN
      READ(DISCO[INAB],TAM,REG[*]);
      REPLACE POINTER(X[1]) BY POINTER (RX[1]) FOR A WORDS;
      END
ELSE
      BEGIN
      REPLACE POINTER(X[1])  BY POINTER (XAB[INAB,1]) FOR A  WORDS;
      REPLACE POINTER(W[1])  BY POINTER (WAB[INAB,1]) FOR NK WORDS;
      END;
END;                  % FIM DA PROCEDURE OBTER
```

```
PROCEDURE GUARDAR(I);

%
%                    GUARDA OS VALORES NA LISTA DE ABERTOS OU NO DISCO
%

INTEGER         I;

BEGIN
IF I*A > 65000 OR I*NK > 65000
THEN
      BEGIN
      IF NOT GRAVOU
      THEN
            BEGIN
            WRITE(DISCO,TAM,REG[*]);
            LOCK(DISCO);
            DISCO.MYUSE:=VALUE(IO);
            GRAVOU:=TRUE;
            END;
      REPLACE POINTER (RX[1]) BY POINTER (X[1]) FOR A  WORDS;
      WRITE(DISCO[I],TAM,REG[*]);
      END
ELSE
      BEGIN
      REPLACE POINTER(XAB[I,1]) BY POINTER(X[1])  FOR A  WORDS;
      REPLACE POINTER(WAB[I,1]) BY POINTER(W[1])  FOR NK WORDS;
      END;
END;                    % FIM DA PROCEDURE GUARDAR
```

```
PROCEDURE INICIALIZAR;
%
%               PASSO  0  DO ALGORITMO  E  INIC. DE VARIAVEIS E LISTAS
%               ----------------------------------------------------
%


BEGIN

     INTEGER        I,J;                  % INDEXADORES

%
%          INICIALIZACAO DAS VARIAVEIS COMUNS A TODAS OS PROCEDURES
%

LSF := 10@60;

%
%          CALCULO DOS COEFICIENTES ANGULAR E LINEAR DE CADA FUNCAO
%               CONCAVA LINEARIZADA  F := ALFA * X + BETA
%

FOR I:=1 STEP 1 UNTIL AF
DO
          BEGIN
          C[I]:=(CUSTO(U[I],I)-CUSTO(L[I],I)) /
               (U[I]-L[I]);
          B[I]:=CUSTO(U[I],I)-(C[I]*U[I]);
          ALFA[I]:=C[I];
          BETA[I]:=B[I];
          END;
%
%          CALCULO DO PROBLEMA LINEARIZADO
%
KILTER;
IF VAZIO = 1
THEN
     BEGIN
     WRITE (IMP,<"**** PROBLEMA INFACTIVEL - CONJUNTO DE SOLUCOES",
          " VIAVEIS VAZIO">);
     GO TO FIM
     END;
%
%          CALCULO DE LI, LSF, LSX E PIOR ARCO
%
PIOR:=1;            % SE NAO HA PIOR (PROB. LIN) ASSUMO QUE E O PRIMEIRO
XPI:=X[1];
CALCLIM;
%
%          CRIACAO DAS LISTAS DE ABERTOS E FECHADOS
%
INAB:=1;
VALAB[1]:=LI;
PROXAB[1]:=(NAB+1);
AB[1,2]:=PIOR;
AB[1,3]:=XPI;
AB[1,4]:=PIOR;      % INICIALIZO COM O PIOR ARCO POR CONVENIENCIA
AB[1,5]:=L[PIOR];
AB[1,6]:=U[PIOR];
AB[1,7]:=C[PIOR];
AB[1,8]:=B[PIOR];
VALAB[NAB+1]:=4"FFFFFFF";
PROXAB[NAB+1]:=(NAB+2);
GUARDAR(1);
```

```
FIMAB:=2;
FIMFE:=1;
IF LISTA > 3
THEN
      BEGIN
      WRITE (IMP(SKIP 1 ));
      WRITE (IMP,<"  K    NO",X10,"LS",X11,"LI",X8,"ERRO     TEMPO ",
           "PIOR  L(PIOR)",X5,"X(PIOR)",X5,"U(PIOR)",X3,"PTO DIVISAO">);
      WRITE (IMP,<X61,"CUSTO",X7,"CUSTO",X7,"CUSTO",X7,"CUSTO">);
      END;
END;       % FIM DA PROCEDURE INICIALIZAR
```

```
PROCEDURE SEPARAR;
%
%                PASSO  2  DO ALGORITMO
%                ----------------------
%

BEGIN

      INTEGER          I,J,                    % INDEXADORES

                       ARCOPI,                 % ARCO PIOR
                       IP;                     % ARCO PIOR


      REAL             LP,                     % L DO PIOR ARCO
                       UP,                     % U DO PIOR ARCO
                       XP,                     % FLUXO NO PIOR ARCO
                       PD,                     % PTO P/ DIVISAO DO ARCO
                       PE,                     % PD + EPSLON
                       LE,                     % LP + EPSLON
                       UPA,                    % U DO PATAMAR ANTERIOR
                       IPR,                    % PONTEIRO P/ PRIMEIRO PATAMAR
                       IUL,                    % PONTEIRO P/ ULTIMO PATAMAR
                       IXP,                    % PONT P/ PATAMAR DE XP
                       TEMPO,                  % TEMPO GASTO
                       CPE,CLE,                % CUSTOS EM PE E LE
                       CXP,CUP,CLP,CPD;        % CUSTOS EM XP, UP, LP E PD

%
%          TESTE DE ESTOURO DA LISTA DE FECHADOS
%
IF  FIMFE > (NFE-1)
THEN
      ULTRAPASSOU := TRUE;
%
%          FECHO O ARCO
%
REPLACE POINTER(FE[FIMFE,1]) BY POINTER (AB[INAB,4]) FOR 6 WORDS;
%
%          INICIALIZACAO DE C[I]
%
REPLACE POINTER(C) BY 10@60 FOR AF+1 WORDS;
%
%          SUBIDA NA ARBORESCENCIA ATE A RAIZ PEGANDO OS VALORES
%               MAIS RECENTES DE C[I] E B[I], ALEM DISSO, OBTENDO
%               L[I] U[I] E X[I] DO PIOR ARCO.
%
I:=FIMFE;
IP:=0;
ARCOPI:=AB[INAB,2];
WHILE I ¬= 0
DO
      BEGIN
      J:=FE[I,1];
      IF C[J] = 10@60
      THEN
           BEGIN
           IF ARCOPI = J
           THEN
                BEGIN
                IP:=ARCOPI;
                LP:=FE[I,2];
                UP:=FE[I,3];
                XP:=AB[INAB,3];
```

```
                END;
            L[J]:=FE[I,2];
            U[J]:=FE[I,3];
            C[J]:=FE[I,4];
            B[J]:=FE[I,5];
            END;
      I:=FE[I,6];
      END;
IF  IP = 0
THEN
      BEGIN
      IP:=ARCOPI;
      LP:=SALVAL[IP];
      UP:=SALVAU[IP];
      XP:=AB[INAB,3];
      END;
%
%           OBTENCAO DOS C[I] E B[I] QUE NAO FORAM ALTERADOS
%
I:= AF+1;
WHILE I:=(MASKSEARCH(10@60,4"FFFFFFFFFFFF",C[I-1])) > 0
DO
      BEGIN
      L[I]:=SALVAL[I];
      U[I]:=SALVAU[I];
      C[I]:=ALFA[I];
      B[I]:=BETA[I];
      END;
%
%           TESTE DE ESTOURO DA LISTA DE ABERTOS
%
IF  FIMAB > (NAB-1)
THEN
      ULTRAPASSOU := TRUE;
%
%              DIVISAO DOS PATAMARES
%
%
LE:=LP;
PD:=PE:=XP;
IF  TIPOFA[IP] = 0
THEN
      BEGIN           % A FUNCAO TEM PATAMARES
      UPA:=-1;        % ATENCAO : NAO MEXER
      IUL:=PRIPATAR[IP] - 1;
      DO
            BEGIN
            IF IUL NEQ PRIPATAR[IP] - 1
            THEN
                  UPA:=(PAT[IUL,1]*IROT[IP]);     % ACHO U DO PAT. ANTERIOR
            IUL:=IUL + 1;                         % SOMO 1 AO INDEXADOR
            IF XP <= (PAT[IUL,1]*IROT[IP])
            THEN
                  IXP:=IUL;
            IF LP > UPA AND LP <= (PAT[IUL,1]*IROT[IP])
            THEN
                  IPR:=IUL;
            END
      UNTIL UP > UPA AND UP <= (PAT[IUL,1]*IROT[IP]);
      IF LP > 0 AND IPR = IUL
      THEN
            PD:=PE:=XP            % E O UNICO PATAMAR
      ELSE
            BEGIN
            IF IPR = IUL
            THEN
```

```
                BEGIN                   % SÓ TEM 1 PATAMAR
                PD:=0;
                PE:=@-3
                END
          ELSE
                BEGIN
                IF (IUL - IPR) = 1 OR IXP = IPR
                THEN
                     BEGIN
                     IF LP NEQ 0                % TEM 2 PATAMARES OU XP
                     THEN                       % CAIU NO PRIMEIRO PATAMAR
                          LE:=LP + @-3;
                     END;
                IF (IUL - IXP) <= 1
                THEN
                     BEGIN
                     PD:=(PAT[IUL-1,1]*IROT[IP]); % XP CAIU NO ULTIMO OU
                     PE:=PD + @-3               % PENULTIMO PATAMAR
                     END
                ELSE
                     PD:=PE:=(PAT[IXP,1]*IROT[IP]);
                END;
          END;
     END;
CPD:=CUSTO(PD,IP);
CLP:=CUSTO(LP,IP);
CUP:=CUSTO(UP,IP);
CPE:=CUSTO(PE,IP);
CLE:=CUSTO(LE,IP);
%
%          IMPRESSAO DOS DADOS DA ITERACAO K
%
IF LISTA > 3
THEN
     BEGIN
     TEMPO:=TIME(2);
     TEMPO:=(TEMPO-HORAIN)/60;
     CXP:=CUSTO(XP,IP);
     WRITE(IMP,<I4,I5,2F13.2,F11.2,A1,F7.2,I5,F11.2,3F12.2>,
                K,INAB,LSF,LI,TOLRE,PERC,TEMPO,IP,LP,XP,UP,PD);
     WRITE (IMP,<X58,4F12.2>,CLP,CXP,CUP,CPD);
     END;
%
%          INCLUSAO DOS DADOS DO PRIMEIRO PROBLEMA GERADO
%               NO LUGAR DO QUE FOI FECHADO
%
AB[INAB,4]:=IP;
AB[INAB,5]:=LE;
AB[INAB,6]:=PD;
IF PD = LE
THEN
     AB[INAB,7]:=0
ELSE
     AB[INAB,7]:=(CPD-CLE) / (PD-LE);
AB[INAB,8]:=CPD-(AB[INAB,7]*PD);
AB[INAB,9]:=FIMFE;
%
%          INCLUSAO DO SEGUNDO PROBLEMA GERADO NO FIM DA LISTA DE ABERTOS
%
AB[FIMAB,4]:=IP;
AB[FIMAB,5]:=PE;
AB[FIMAB,6]:=UP;
AB[FIMAB,7]:=(CUP-CPE) / (UP-PE);
AB[FIMAB,8]:=CUP-(AB[FIMAB,7]*UP);
AB[FIMAB,9]:=FIMFE;
END;            % FIM DA PROCEDURE SEPARAR
```

```
PROCEDURE AVALIAR;
%
%               PASSO  3  DO ALGORITMO
%               ----------------------
%

BEGIN

      INTEGER         I,J;                  % INDEXADORES


PROXAB[0]:=PROXAB[INAB];
I:=INDICE:=AB[INAB,4];
%
%               CALCULO DO SEGUNDO PROBLEMA GERADO ( POR CONVENIENCIA
%                    FIZ ESSE PRIMEIRO )
%
L[I]:=AB[FIMAB,5];
U[I]:=AB[FIMAB,6];
C[I]:=AB[FIMAB,7];
B[I]:=AB[FIMAB,8];
OBTER;
KILTER;
IF  VAZIO = 1
THEN
      LI:=4"FFFFFFE"
ELSE
      CALCLIM; %CALCULO DE LI, LSF, LSX E PIOR PARA O SEG. PROBLEMA
%
%          INCLUSAO DOS DADOS NA LISTA DE ABERTOS
%
VALAB[FIMAB]:=LI;
AB[FIMAB,2]:=PIOR;
AB[FIMAB,3]:=XPI;
%
%               COLOCACAO DE FIMAB EM ORDEM
%
J:=LISTLOOKUP(LI,LISTAB,0);
PROXAB[FIMAB]:=PROXAB[J];
PROXAB[J]:=FIMAB;
GUARDAR(FIMAB);
%
%          CALCULO DO PRIMEIRO PROBLEMA GERADO
%
L[I]:=AB[INAB,5];
U[I]:=AB[INAB,6];
C[I]:=AB[INAB,7];
B[I]:=AB[INAB,8];
OBTER;
KILTER;
IF  VAZIO = 1
THEN
      LI:=4"FFFFFFE"
ELSE
      CALCLIM; %CALC. LI, LSF, LSX, PIOR E XPI PARA O PRIM. PROBLEMA
%
%               INCLUSAO DOS DADOS NA LISTA DE ABERTOS
%
VALAB[INAB]:=LI;
AB[INAB,2]:=PIOR;
AB[INAB,3]:=XPI;
%
%               COLOCACAO DE INAB EM ORDEM
%
```

```
J:=LISTLOOKUP(LI,LISTAB,0);
PROXAB[INAB]:=PROXAB[J];
PROXAB[J]:=INAB;
GUARDAR(INAB);
INAB:=PROXAB[0];
FIMAB:=FIMAB+1;
FIMFE:=FIMFE+1;
END;                    % FIM DA PROCEDURE AVALIAR
```

```
PROCEDURE IMPRIMIR;
%
%               IMPRESSAO DOS RESULTADOS
%
%

BEGIN

      INTEGER         I,J;                    % INDEXADORES


      REAL            TOTF,                   % TOTAL DO FLUXO
                      TOTC,                   % TOTAL DO CUSTO
                      TOL1, TOL2,             % TOLERANCIAS
                      TLINT,                  % TEMPO TOTAL DO ALG LINEAR
                      MEDIA,                  % MEDIA DE TEMPO
                      TAL,                    % TEMPO DO ALGORITMO
                      TAL1,                   % TEMPO DA PROC INICIALIZAR
                      TEMPO,                  % TEMPO DE PROCESSAMENTO
                      AL,                     % AUX. PARA ALFA[I]
                      IR,                     % AUX. P/ I. ROT.
                      CAPA,                   % CAPACIDADE ESTATICA
                      CUS;                    % CUSTO DO ARCO

      FORMAT FA01("PROBLEMA DE LOCALIZACAO COM FUNCOES DE CUSTO",
           " NAO LINEARES");
      FORMAT FA02(//,"AUTOR: RONALDO RUST",
           "       ORIENTADOR: CLAUDIO THOMAS BORNSTEIN");
      FORMAT FB01(///,"VERSAO ",I2,"  PROBLEMA ",I4,X4,"**  RESUMO DO",
           " PROCESSAMENTO  **");
      FORMAT FB02(//,"CARACTERISTICAS:");
      FORMAT FB03(/,I4," NOS   ",I4," PRODUTORES   ",I4," ARCOS   ",
           I4," CONSUMIDORES");
      FORMAT FB52(/,"NO. DE ITERACOES",16(".."),X10,I7);
      FORMAT FB54(/,"VALOR DA FUNCAO OBJETIVO",8(".."),F17.2);
      FORMAT FB56(/,"TOLERANCIA",22(".."),X10,F7.2,X1,A1);

      FORMAT FB57(/,"ERRO MAXIMO COMETIDO",12(".."),F17.2," =>",
           F6.2," %");
      FORMAT FB59(/,"TEMPO TOTAL GASTO",15(".."),X8,F9.4," SEG");
      FORMAT FB60(/,"OUT-OF-KILTER",19(".."),X8,F9.4," SEG");
      FORMAT FB61(/,"      PRIMEIRA CHAMADA.........",X8,F9.4,
           " SEG");
      FORMAT FB62(/,"      MEDIA DAS DEMAIS CHAMADAS.",X8,F9.4,
           " SEG/CHAMADA");
      FORMAT FB63(/,"ALGORITMO....",19(".."),X8,F9.4," SEG");
      FORMAT FB64(/,"      INICIALIZACAO.............",X8,F9.4,
           " SEG");
      FORMAT FB65(/,"      MEDIA DAS ITERACOES.......",X8,F9.4,
           " SEG/ITERACAO");
      FORMAT FB19(//,"ATENCAO:  **  NAO FOI POSSIVEL ATINGIR A TOLERAN",
           "CIA DESEJADA");
      FORMAT FB26("VERSAO ",I2,"  PROBLEMA ",I4,X2,
           "** CENTROS PRODUTORES **");
      FORMAT FB28(//,X20,"-NO-   -- NOME --    -- PRODUCAO --");
      FORMAT FB39("VERSAO ",I2,"  PROBLEMA ",I4,X2,
           "** CENTROS CONSUMIDORES **");
      FORMAT FB41(//,X20,"-NO-   -- NOME --    -- CONSUMO --");
      FORMAT FBDE(/,I5,X15,I4,X2,2A6,X5,F10.2);
      FORMAT FBTO(//,X28,"**** TOTAL ",F14.2);
      FORMAT FC01("VERSAO ",I2,"  PROBLEMA ",I4,X20,
           "**  T R A N S P O R T E  **");

      FORMAT FC02("VERSAO ",I2,"  PROBLEMA ",I4,X19,
```

```
          "**  A R M A Z E N A G E M",
          "  **");
     FORMAT FC03(//,X8,"-- NOME --    --ARCO--  LIM. INF.  LIM. SUP.",
          " CUSTO UNIT. FUNC I.ROT CAPACIDADE  --FLUXO--  -- CUSTO --");
     FORMAT FCDE(/,I5,X2,2A6,I5,"-",I4,F11.2,F11.2,F11.2,I6,F6.2,F11.2,
          F11.2,F13.2);
     FORMAT FC09(//,X60,"**** TOTAL DO FLUXO   ",F14.2);
     FORMAT FC11(/,X55,"**** CUSTO DE TRANSPORTE",X12,F18.2);
     FORMAT FC12(/,X54,"**** CUSTO DE ARMAZENAGEM",X12,F18.2);

VERSAO:=5;
WRITE (IMP[SKIP 1 ]);
%
%          RESUMO DO PROCESSAMENTO
%
WRITE (IMP,FA01);
WRITE (IMP,FA02);
WRITE  (IMP,FB01,VERSAO,CONF);
WRITE (IMP,FB02);
WRITE  (IMP,FB03,N,NP,A,NC);
WRITE (IMP,FB52,K);
WRITE (IMP,FB54,LSF);
WRITE (IMP,FB56,TOLERA,PERC);
TOL1:=LSF-LI;
TOL2:=(TOL1/LSF)*100;
WRITE (IMP,FB57,TOL1,TOL2);
HORAIN:=HORAIN/60; HORAFIM:=HORAFIM/60; HORAFIM1:=HORAFIM1/60;
TLIN:=TLIN/60; TLIN1:=TLIN1/60;
TEMPO:=HORAFIM-HORAIN;
WRITE (IMP,FB59,TEMPO);
TLINT:=TLIN+TLIN1;
WRITE (IMP,FB60,TLINT);
WRITE (IMP,FB61,TLIN1);
IF CHALIN ^= 0
THEN
     MEDIA:=TLIN/CHALIN;
WRITE (IMP,FB62,MEDIA);
TAL:=TEMPO-TLINT;
WRITE (IMP,FB63,TAL);
TAL1:=HORAFIM1-HORAIN-TLIN1;
WRITE (IMP,FB64,TAL1);
IF K ^= 0
THEN
     MEDIA:=(TAL-TAL1)/K
ELSE
     MEDIA:=0;
WRITE (IMP,FB65,MEDIA);
IF ULTRAPASSOU
THEN
     WRITE (IMP,FB19);
IF LISTA = 1 OR LISTA = 3 OR LISTA = 5 OR LISTA = 7
THEN
     BEGIN
%
%               CENTROS PRODUTORES
%
     WRITE (IMP[SKIP 1 ]);
     WRITE (IMP,FB26,VERSAO,CONF);
     WRITE (IMP,FB28);
     WRITE (IMP[SPACE(1)]);
     J:=AP1;
     FOR  I:=1 STEP 1 UNTIL NP
     DO
          BEGIN
          WRITE(IMP,FBDE,I,ARCO[J,2],NOMP[I,1],NOMP[I,2],L[J]);
          J:=J+1;
```

```
      END;
   WRITE (IMP,FBTO,TOTPRO);
%
%                    TRANSPORTE
%
   REPLACE POINTER(L[1]) BY POINTER(SALVAL[1]) FOR AF WORDS;
   REPLACE POINTER(U[1]) BY POINTER(SALVAU[1]) FOR AF WORDS;

   WRITE (IMP[SKIP 1 ]);
   WRITE (IMP,FC01,VERSAO,CONF);
   WRITE (IMP,FC03);
   WRITE (IMP[SPACE(1)]);
   J:=1;
   CUS:=0;
   TOTF:=0;
   TOTC:=0;
   FOR  I:=1 STEP 1 UNTIL A
   DO
        BEGIN
        IF TARC[I] = 0
        THEN
             BEGIN
             IF NFUN[I] = 0
             THEN
                  BEGIN
                  CUS:=LSX[I]*ALFA[I];
                  IR:=0;
                  CAPA:=LSX[I];
                  AL:=ALFA[I]
                  END
             ELSE
                  BEGIN
                  IR:=IROT[I];
                  AL:=0;
                  CUS:=CUSTO(LSX[I],I);
                  L[I]:=L[I]/IROT[I];
                  U[I]:=U[I]/IROT[I];
                  CAPA:=LSX[I]/IROT[I];  % ATENCAO: E AQUI MESMO
                  END;
             WRITE(IMP,FCDE,J,NOMA[I,1],NOMA[I,2],ARCO[I,1],ARCO[I,2],
                  L[I],U[I],AL,NFUN[I],IR,CAPA,LSX[I],CUS);
             TOTF:=TOTF+LSX[I];
             TOTC:=TOTC+CUS;
             J:=J+1;
             END;
        END;
   WRITE (IMP,FC09,TOTF);
   WRITE (IMP,FC11,TOTC);
%
%                    ARMAZENAGEM
%
   WRITE (IMP[SKIP 1 ]);
   WRITE (IMP,FC02,VERSAO,CONF);
   WRITE (IMP,FC03);
   WRITE (IMP[SPACE(1)]);
   J:=1;
   CUS:=0;
   TOTF:=0;
   TOTC:=0;
   FOR  I:=1 STEP 1 UNTIL A
   DO
        BEGIN
        IF TARC[I] = 1
        THEN
             BEGIN
             IF NFUN[I] = 0
```

```
                THEN
                     BEGIN
                     CUS:=LSX[I]*ALFA[I];
                     IR:=0;
                     CAPA:=LSX[I];
                     AL:=ALFA[I]
                     END
                ELSE
                     BEGIN
                     IR:=IROT[I];
                     AL:=0;
                     CUS:=CUSTO(LSX[I],I);
                     L[I]:=L[I]/IROT[I];
                     U[I]:=U[I]/IROT[I];
                     CAPA:=LSX[I]/IROT[I];  % ATENCAO: E AQUI MESMO
                     END;
                WRITE(IMP,FCDE,J,NOMA[I,1],NOMA[I,2],ARCO[I,1],ARCO[I,2],
                     L[I],U[I],AL,NFUN[I],IR,CAPA,LSX[I],CUS);
                TOTF:=TOTF+LSX[I];
                TOTC:=TOTC+CUS;
                J:=J+1;
                END;
          END;
     WRITE (IMP,FC09,TOTF);
     WRITE (IMP,FC12,TOTC);
%
%                    CENTROS CONSUMIDORES
%
     WRITE (IMP[SKIP 1]);
     WRITE (IMP,FB39,VERSAO,CONF);
     WRITE (IMP,FB41);
     WRITE (IMP[SPACE(1)]);
     J:=AC1;
     FOR  I:=1 STEP 1 UNTIL NC
     DO
          BEGIN
          WRITE(IMP,FBDE,I,ARCO[J,1],NOMC[I,1],NOMC[I,2],L[J]);
          J:=J+1;
          END;
     WRITE (IMP,FBTO,TOTCON);
     END;
END;            % FIM DA PROCEDURE IMPRIMIR
```

```
      LER;
      HORAIN:=TIME(2);
      INICIALIZAR;
      LI:=VALAB[INAB];
      IF PERC = "%"
      THEN
            BEGIN
            IF LSF ¬= 0
            THEN
                  TOLRE:=(LSF-LI)*100/LSF
            ELSE
                  TOLRE:=0
            END
      ELSE
            TOLRE:=LSF-LI;
      HORAFIM1:=TIME(2);
      %
      % TESTE REFERENTE AO PASSO 1 DO ALGORITMO (LEMBRAR QUE O PRIMEIRO
      % DA LISTA DE ABERTOS E O MENOR). ALEM DISSO TESTA ESTOURO DE LISTA
      %
      WHILE TOLRE > TOLERA AND NOT ULTRAPASSOU
      DO
            BEGIN
            K:=K+1;    %  CONTA MAIS UMA ITERACAO DO ALGORITMO
            SEPARAR;
            AVALIAR;
            LI:=VALAB[INAB];
            IF PERC = "%"
            THEN
                  BEGIN
                  IF LSF ¬= 0
                  THEN
                        TOLRE:=(LSF-LI)*100/LSF
                  ELSE
                        TOLRE:=0
                  END
            ELSE
                  TOLRE:=LSF-LI;
            END;
      HORAFIM:=TIME(2);
      IMPRIMIR;
FIM:
      END;



      %%-------------------------------------------------%%
      %%              FIM DO BLOCO PRINCIPAL             %%
      %%                                                 %%
      %%-------------------------------------------------%%


END.
```

ANEXO II

FORMULÁRIOS PARA ENTRADA DE DADOS

PROBLEMA DE LOCALIZAÇÃO

RESPONSAVEL	DATA	FOLHA
	___/___/___	___ DE ___

MESTRE

TIPO	LISTA	PROBLEMA	Nº DE NÓS	Nº DE CENTROS PRODUT.	Nº DE ARCOS	Nº DE CENTROS CONSUM.	Nº DE ARCOS COM FUN. NÃO LIN.	TOLERÂNCIA (2 DECIMAIS)	%	Nº DE FUNÇÕES NÃO LIN.
M										
1	2	3 4 5 6	7 8 9 10	11 12 13 14	15 16 17 18	19 20 21 22	23 24 25 26	27 28 29 30 31 32 33 34 35 36 37		38 39 40 41

DESCRIÇÃO DO PROBLEMA

FUNÇÕES

TIPO	Nº DA FUNÇÃO	Nº PAT.	LIMITE SUPERIOR DO PATAMAR (2 DECIMAIS)	PARAMETROS DO PATAMAR $a+bx^c+dx^e$ (2 DECIMAIS)				
				a	b	c	d	e
O								
O								
O							

PROBLEMA DE LOCALIZAÇÃO

PRODUÇÃO / CONSUMO

TIPO	Nº DO CARTÃO	NÓ PRODUT./ CONSUM.	VALOR DA PRODUÇÃO/CONSUMO (2 DECIMAIS)	NOME DO CENTRO PRODUTOR/CONSUMIDOR

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

RESPONSAVEL	DATA ___/___/___	FOLHA ___ DE ___

PRODUÇÃO / CONSUMO (CONTINUAÇÃO)

TIPO	Nº DO CARTÃO	NÓ PRODUT./ CONSUM.	VALOR DA PRODUÇÃO/CONSUMO (2 DECIMAIS)	NOME DO CENTRO PRODUTOR / CONSUMIDOR

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

PROBLEMA DE LOCALIZAÇÃO

RESPONSAVEL	DATA	FOLHA
	___/___/___	____ DE ____

TRANSPORTE / ARMAZENAGEM

TIPO	Nº DO CARTÃO	NÓ DE ORIGEM	NÓ DE DESTINO	TIP. ARC.	TIP. FUN.			LIMITE INFERIOR (2 DECIMAIS)	LIMITE SUPERIOR (2 DECIMAIS)	CUSTO UNITÁRIO (2 DECIMAIS)	Nº DA FUNÇÃO	NOME DO ARMAZEM OU MEIO DE TRANSPORTE	INDICE DE ROTAÇÃO (2 DECIMAIS)
2													
2													
2													

ANEXO III

TESTE COMPLETO DO PROBLEMA 8.4.1

ANEXO III.1 - CARTÕES DE DADOS

```
1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...8(

M7 841  54  23 243   5  13      1,30%   1
0  103 0
0  103 0     60000,00 215250,00   6566,25       0,50       0,00       0,00
1    1   2    9902,00ALTAMIRA
1    2   3   16114,00BACABINHA
1    3   4   21782,00BOM JARDIM
1    4   5   38552,00LAGO PEDRA
1    5   6   39459,00MONCAO
1    6   7   10799,00PINDARE MIR.
1    7   8    9399,00SANTA INES
1    8   9   51128,00SANTA LUZIA
1    9  10   14687,00VIT. FREIRE
1   10  11   10646,00BACABAL
1   11  12    7377,00ESPERANTOP.
1   12  13    3305,00IGARAPE GRDE
1   13  14   18463,00IPIXUNA
1   14  15   10946,00JOSELANDIA
1   15  16    9551,00LAGO JUNCO
1   16  17    2947,00LAGO VERDE
1   17  18    2219,00LIMA CAMPOS
1   18  19    3963,00OLHO D'AGUA
1   19  20    1998,00PEDREIRAS
1   20  21    8882,00PIO XII
1   21  22    6011,00POCAO PEDRA
1   22  23   11268,00STO A. LOPES
1   23  24    8089,00SAO MATEUS
2  194  25  3811          0,00  60000,00       0,00  103BOM JARDIM    1,10
2  195  26  3911          0,00  60000,00       0,00  103LAGO PEDRA    1,10
2  196  27  4011          0,00  60000,00       0,00  103MONCAO        1,10
2  197  28  4111          0,00  60000,00       0,00  103PINDARE MIR.  1,10
2  198  29  4211          0,00  60000,00       0,00  103SANTA INES    1,10
2  199  30  4311          0,00  60000,00       0,00  103SANTA LUZIA   1,10
2  200  31  4411          0,00  60000,00       0,00  103VIT. FREIRE   1,10
2  201  32  4511          0,00  60000,00       0,00  103BACABAL       1,07
2  202  33  4611          0,00  60000,00       0,00  103IPIXUNA       1,07
2  203  34  4711          0,00  60000,00       0,00  103OLHO D'AGUA   1,07
2  204  35  4811          0,00  60000,00       0,00  103PEDREIRAS     1,07
2  205  36  4911          0,00  60000,00       0,00  103POCAO PEDRAS  1,07
2  206  37  5011          0,00  60000,00       0,00  103STO A. LOPES  1,07
2   24   2  2500          0,00999999999,99   222,75     0              0,00
2   25   2  2600          0,00999999999,99   113,05     0              0,00
2   26   2  2800          0,00999999999,99   222,75     0              0,00
2   27   2  2900          0,00999999999,99   198,00     0              0,00
2   28   2  3000          0,00999999999,99   145,20     0              0,00
2   29   2  3100          0,00999999999,99   129,05     0              0,00
2   30   2  3200          0,00999999999,99   198,00     0              0,00
2   31   2  3300          0,00999999999,99   232,65     0              0,00
2   32   2  3400          0,00999999999,99   156,00     0              0,00
2   33   3  2600          0,00999999999,99   129,05     0              0,00
2   34   3  2700          0,00999999999,99   198,00     0              0,00
2   35   3  2800          0,00999999999,99   174,00     0              0,00
2   36   3  2900          0,00999999999,99   150,00     0              0,00
2   37   3  3000          0,00999999999,99   205,20     0              0,00
2   38   3  3100          0,00999999999,99    37,95     0              0,00
2   39   3  3200          0,00999999999,99   144,00     0              0,00
2   40   3  3300          0,00999999999,99   204,60     0              0,00
2   41   3  3400          0,00999999999,99   108,00     0              0,00
2   42   4  2500          0,00999999999,99    90,00     0              0,00
2   43   4  2700          0,00999999999,99   205,20     0              0,00
2   44   4  2800          0,00999999999,99   148,50     0              0,00
2   45   4  2900          0,00999999999,99   231,00     0              0,00
2   46   4  3000          0,00999999999,99   180,00     0              0,00
2   47   4  3400          0,00999999999,99   267,30     0              0,00
```

2	48	5	2500	0.009999999.99	237.60	0	0.00
2	49	5	2600	0.009999999.99	54.00	0	0.00
2	50	5	2900	0.009999999.99	171.00	0	0.00
2	51	5	3000	0.009999999.99	163.35	0	0.00
2	52	5	3100	0.009999999.99	201.96	0	0.00
2	53	5	3400	0.009999999.99	181.50	0	0.00
2	54	5	3500	0.009999999.99	198.00	0	0.00
2	55	5	3600	0.009999999.99	188.10	0	0.00
2	56	6	2500	0.009999999.99	70.00	0	0.00
2	57	6	2700	0.009999999.99	60.00	0	0.00
2	58	6	2800	0.009999999.99	62.54	0	0.00
2	59	6	2900	0.009999999.99	63.00	0	0.00
2	60	6	3100	0.009999999.99	191.40	0	0.00
2	61	6	3400	0.009999999.99	178.20	0	0.00
2	62	7	2500	0.009999999.99	110.40	0	0.00
2	63	7	2600	0.009999999.99	238.00	0	0.00
2	64	7	2700	0.009999999.99	145.00	0	0.00
2	65	7	2800	0.009999999.99	67.00	0	0.00
2	66	7	2900	0.009999999.99	68.30	0	0.00
2	67	7	3000	0.009999999.99	114.00	0	0.00
2	68	7	3100	0.009999999.99	189.00	0	0.00
2	69	7	3400	0.009999999.99	151.00	0	0.00
2	70	8	2500	0.009999999.99	113.85	0	0.00
2	71	8	2600	0.009999999.99	214.00	0	0.00
2	72	8	2700	0.009999999.99	132.00	0	0.00
2	73	8	2800	0.009999999.99	69.00	0	0.00
2	74	8	2900	0.009999999.99	54.00	0	0.00
2	75	8	3000	0.009999999.99	134.00	0	0.00
2	76	8	3100	0.009999999.99	160.00	0	0.00
2	77	8	3300	0.009999999.99	225.00	0	0.00
2	78	8	3400	0.009999999.99	132.00	0	0.00
2	79	9	2500	0.009999999.99	264.00	0	0.00
2	80	9	2600	0.009999999.99	202.50	0	0.00
2	81	9	2800	0.009999999.99	252.00	0	0.00
2	82	9	3000	0.009999999.99	80.00	0	0.00
2	83	9	3100	0.009999999.99	193.00	0	0.00
2	84	9	3400	0.009999999.99	217.80	0	0.00
2	85	10	2500	0.009999999.99	109.60	0	0.00
2	86	10	2600	0.009999999.99	112.20	0	0.00
2	87	10	2700	0.009999999.99	204.60	0	0.00
2	88	10	2800	0.009999999.99	132.00	0	0.00
2	89	10	2900	0.009999999.99	132.00	0	0.00
2	90	10	3100	0.009999999.99	32.00	0	0.00
2	91	10	3200	0.009999999.99	166.50	0	0.00
2	92	10	3300	0.009999999.99	173.25	0	0.00
2	93	10	3400	0.009999999.99	110.40	0	0.00
2	94	11	2600	0.009999999.99	189.00	0	0.00
2	95	11	2700	0.009999999.99	202.50	0	0.00
2	96	11	2800	0.009999999.99	175.00	0	0.00
2	97	11	2900	0.009999999.99	162.00	0	0.00
2	98	11	3100	0.009999999.99	144.00	0	0.00
2	99	11	3200	0.009999999.99	45.00	0	0.00
2	100	11	3300	0.009999999.99	66.00	0	0.00
2	101	11	3400	0.009999999.99	120.00	0	0.00
2	102	11	3500	0.009999999.99	112.20	0	0.00
2	103	11	3600	0.009999999.99	184.80	0	0.00
2	104	12	2600	0.009999999.99	227.70	0	0.00
2	105	12	3300	0.009999999.99	178.20	0	0.00
2	106	12	3500	0.009999999.99	178.20	0	0.00
2	107	12	3600	0.009999999.99	60.72	0	0.00
2	108	12	3700	0.009999999.99	132.00	0	0.00
2	109	13	2600	0.009999999.99	198.00	0	0.00
2	110	13	3200	0.009999999.99	165.00	0	0.00
2	111	13	3300	0.009999999.99	138.60	0	0.00
2	112	13	3500	0.009999999.99	91.08	0	0.00
2	113	13	3600	0.009999999.99	75.90	0	0.00

2	114	13	3700	0.009999999.99	191.40	0	0.00
2	115	14	2800	0.009999999.99	227.70	0	0.00
2	116	14	2900	0.009999999.99	195.00	0	0.00
2	117	14	3200	0.009999999.99	48.00	0	0.00
2	118	14	3300	0.009999999.99	41.00	0	0.00
2	119	14	3400	0.009999999.99	162.00	0	0.00
2	120	14	3500	0.009999999.99	68.31	0	0.00
2	121	14	3600	0.009999999.99	204.60	0	0.00
2	122	15	3200	0.009999999.99	207.90	0	0.00
2	123	15	3300	0.009999999.99	214.50	0	0.00
2	124	15	3500	0.009999999.99	178.20	0	0.00
2	125	15	3600	0.009999999.99	106.26	0	0.00
2	126	15	3700	0.009999999.99	132.00	0	0.00
2	127	16	2600	0.009999999.99	125.40	0	0.00
2	128	16	3100	0.009999999.99	183.15	0	0.00
2	129	16	3200	0.009999999.99	12.10	0	0.00
2	130	16	3300	0.009999999.99	112.20	0	0.00
2	131	16	3400	0.009999999.99	191.40	0	0.00
2	132	16	3500	0.009999999.99	158.40	0	0.00
2	133	16	3600	0.009999999.99	132.00	0	0.00
2	134	16	3700	0.009999999.99	188.10	0	0.00
2	135	17	2600	0.009999999.99	186.00	0	0.00
2	136	17	2700	0.009999999.99	183.15	0	0.00
2	137	17	2800	0.009999999.99	167.40	0	0.00
2	138	17	2900	0.009999999.99	173.40	0	0.00
2	139	17	3000	0.009999999.99	227.70	0	0.00
2	140	17	3100	0.009999999.99	165.00	0	0.00
2	141	17	3200	0.009999999.99	121.44	0	0.00
2	142	17	3300	0.009999999.99	145.20	0	0.00
2	143	17	3400	0.009999999.99	110.40	0	0.00
2	144	17	3500	0.009999999.99	187.00	0	0.00
2	145	17	3600	0.009999999.99	227.70	0	0.00
2	146	18	3200	0.009999999.99	157.50	0	0.00
2	147	18	3300	0.009999999.99	132.00	0	0.00
2	148	18	3500	0.009999999.99	15.49	0	0.00
2	149	18	3600	0.009999999.99	198.00	0	0.00
2	150	18	3700	0.009999999.99	214.50	0	0.00
2	151	19	2500	0.009999999.99	198.00	0	0.00
2	152	19	2600	0.009999999.99	156.00	0	0.00
2	153	19	2700	0.009999999.99	178.20	0	0.00
2	154	19	2800	0.009999999.99	126.00	0	0.00
2	155	19	2900	0.009999999.99	114.00	0	0.00
2	156	19	3100	0.009999999.99	75.90	0	0.00
2	157	19	3200	0.009999999.99	126.00	0	0.00
2	158	19	3300	0.009999999.99	174.20	0	0.00
2	159	19	3400	0.009999999.99	18.00	0	0.00
2	160	19	3500	0.009999999.99	178.20	0	0.00
2	161	20	3200	0.009999999.99	176.00	0	0.00
2	162	20	3300	0.009999999.99	102.00	0	0.00
2	163	20	3400	0.009999999.99	183.15	0	0.00
2	164	20	3500	0.009999999.99	41.00	0	0.00
2	165	20	3600	0.009999999.99	145.20	0	0.00
2	166	20	3700	0.009999999.99	132.00	0	0.00
2	167	21	2500	0.009999999.99	213.18	0	0.00
2	168	21	2600	0.009999999.99	186.00	0	0.00
2	169	21	2700	0.009999999.99	132.00	0	0.00
2	170	21	2800	0.009999999.99	96.60	0	0.00
2	171	21	2900	0.009999999.99	75.90	0	0.00
2	172	21	3100	0.009999999.99	82.10	0	0.00
2	173	21	3200	0.009999999.99	156.00	0	0.00
2	174	21	3300	0.009999999.99	190.00	0	0.00
2	175	21	3400	0.009999999.99	41.30	0	0.00
2	176	21	3500	0.009999999.99	198.00	0	0.00
2	177	22	2600	0.009999999.99	188.10	0	0.00
2	178	22	3200	0.009999999.99	185.00	0	0.00
2	179	22	3300	0.009999999.99	198.00	0	0.00

2	180	22	3500	0,009999999,99	138,60	0	0,00
2	181	22	3600	0,009999999,99	45,00	0	0,00
2	182	22	3700	0,009999999,99	118,80	0	0,00
2	183	23	3200	0,009999999,99	268,95	0	0,00
2	184	23	3300	0,009999999,99	237,60	0	0,00
2	185	23	3500	0,009999999,99	132,00	0	0,00
2	186	23	3600	0,009999999,99	118,80	0	0,00
2	187	23	3700	0,009999999,99	41,00	0	0,00
2	188	24	2500	0,009999999,99	316,80	0	0,00
2	189	24	2600	0,009999999,99	244,80	0	0,00
2	190	24	3200	0,009999999,99	156,00	0	0,00
2	191	24	3300	0,009999999,99	182,40	0	0,00
2	192	24	3400	0,009999999,99	198,00	0	0,00
2	193	24	3500	0,009999999,99	189,00	0	0,00
2	207	38	5100	0,009999999,99	417,60	0	0,00
2	208	38	5300	0,009999999,99	288,00	0	0,00
2	209	38	5400	0,009999999,99	440,00	0	0,00
2	210	38	100	0,009999999,99	324,00	0	0,00
2	211	39	5100	0,009999999,99	309,00	0	0,00
2	212	39	5200	0,009999999,99	430,00	0	0,00
2	213	39	5300	0,009999999,99	366,00	0	0,00
2	214	39	5400	0,009999999,99	417,00	0	0,00
2	215	39	100	0,009999999,99	426,00	0	0,00
2	216	40	5100	0,009999999,99	475,20	0	0,00
2	217	40	5300	0,009999999,99	306,00	0	0,00
2	218	40	5400	0,009999999,99	423,00	0	0,00
2	219	40	100	0,009999999,99	312,00	0	0,00
2	220	41	5100	0,009999999,99	372,00	0	0,00
2	221	41	5300	0,009999999,99	280,80	0	0,00
2	222	41	5400	0,009999999,99	402,00	0	0,00
2	223	41	100	0,009999999,99	348,00	0	0,00
2	224	42	5100	0,009999999,99	372,00	0	0,00
2	225	42	5300	0,009999999,99	270,00	0	0,00
2	226	42	5400	0,009999999,99	390,00	0	0,00
2	227	42	100	0,009999999,99	338,40	0	0,00
2	228	43	5100	0,009999999,99	297,00	0	0,00
2	229	43	5200	0,009999999,99	495,00	0	0,00
2	230	43	5300	0,009999999,99	300,00	0	0,00
2	231	43	5400	0,009999999,99	474,00	0	0,00
2	232	43	100	0,009999999,99	363,00	0	0,00
2	233	44	5100	0,009999999,99	350,00	0	0,00
2	234	44	5200	0,009999999,99	510,00	0	0,00
2	235	44	5300	0,009999999,99	315,00	0	0,00
2	236	44	5400	0,009999999,99	363,00	0	0,00
2	237	44	100	0,009999999,99	366,00	0	0,00
2	238	45	5100	0,009999999,99	400,00	0	0,00
2	239	45	5200	0,009999999,99	545,00	0	0,00
2	240	45	5300	0,009999999,99	333,00	0	0,00
2	241	45	5400	0,009999999,99	313,50	0	0,00
2	242	45	100	0,009999999,99	300,00	0	0,00
2	243	46	5100	0,009999999,99	420,00	0	0,00
2	244	46	5200	0,009999999,99	544,50	0	0,00
2	245	46	5300	0,009999999,99	354,00	0	0,00
2	246	46	5400	0,009999999,99	316,00	0	0,00
2	247	46	100	0,009999999,99	321,00	0	0,00
2	248	47	5100	0,009999999,99	354,00	0	0,00
2	249	47	5200	0,009999999,99	554,00	0	0,00
2	250	47	5300	0,009999999,99	342,00	0	0,00
2	251	47	5400	0,009999999,99	333,00	0	0,00
2	252	47	100	0,009999999,99	342,00	0	0,00
2	253	48	5100	0,009999999,99	447,00	0	0,00
2	254	48	5200	0,009999999,99	491,40	0	0,00
2	255	48	5300	0,009999999,99	375,00	0	0,00
2	256	48	5400	0,009999999,99	310,20	0	0,00
2	257	48	100	0,009999999,99	330,00	0	0,00
2	258	49	5100	0,009999999,99	580,00	0	0,00

```
2   259   49   5200          0,009999999,99     400,00    0          0,00
2   260   49   5300          0,009999999,99     423,00    0          0,00
2   261   49   5400          0,009999999,99     369,60    0          0,00
2   262   49    100          0,009999999,99     372,00    0          0,00
2   263   50   5200          0,009999999,99     460,00    0          0,00
2   264   50   5300          0,009999999,99     510,00    0          0,00
2   265   50   5400          0,009999999,99     284,40    0          0,00
2   266   50    100          0,009999999,99     417,00    0          0,00
3   267   51   15874,00IMPERATRIZ
3   268   52   31748,00PORTO FRANCO
3   269   53   63497,00CAND, MENDES
3   270   54   95248,00TIMON
3   271    1  111120,00SAO LUIZ
```

ANEXO III.2 - LISTAGEM PASSO A PASSO

K	NO	LS	LI	ERRO	TEMPO	PIOR	L(PIOR) CUSTO	X(PIOR) CUSTO	U(PIOR) CUSTO	PTO DIVISAO CUSTO
1	1	138193729,44	132668479,00	4,00%	11,15	12	0,00	9070,00	64200,00	9070,00
							0,00	819795,74	1823646,20	819795,74
2	1	136914682,83	132709224,00	3,07%	12,27	11	0,00	7522,00	64200,00	7522,00
							0,00	765794,20	1823646,20	765794,20
3	1	136914682,83	133178595,00	2,73%	13,30	5	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
4	2	136914682,83	133230635,00	2,69%	13,92	9	0,00	7646,00	64200,00	7646,00
							0,00	770313,50	1823646,20	770313,50
5	3	136914682,83	133261351,00	2,67%	14,48	5	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
6	2	136914682,83	133265042,00	2,67%	15,17	11	0,00	7522,00	64200,00	7522,00
							0,00	765794,20	1823646,20	765794,20
7	1	136914682,83	133523679,00	2,48%	16,17	4	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
8	3	136914682,83	133606435,00	2,42%	16,87	4	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
9	2	136914682,83	133658951,00	2,38%	17,58	5	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
10	1	136914682,83	133725518,00	2,33%	18,30	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
11	4	136914682,83	133725518,00	2,33%	19,03	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
12	5	136914682,83	133783758,00	2,29%	19,73	11	0,00	7522,00	64200,00	7522,00
							0,00	765794,20	1823646,20	765794,20
13	3	136914682,83	133808275,00	2,27%	20,80	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
14	6	136914682,83	133808275,00	2,27%	21,55	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
15	7	136914682,83	133817168,00	2,26%	22,38	5	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
16	2	136914682,83	134004036,00	2,15%	23,28	4	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
17	8	136914682,83	134070603,00	2,08%	23,97	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
18	9	136914682,83	134153359,00	2,02%	24,47	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
19	5	136914682,83	134161323,00	2,01%	25,15	5	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
20	7	136914682,83	134162253,00	2,01%	25,83	4	0,00	20198,00	66000,00	20198,00
							0,00	1105014,87	1823646,20	1105014,87
21	2	136914682,83	134205875,00	1,98%	26,63	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70
22	10	136914682,83	134205875,00	1,98%	27,35	1	0,00	21782,00	66000,00	21782,00
							0,00	1139245,70	1823646,20	1139245,70

23	11	136914682.83	134261784.00	1.94%	28.02	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
24	4	136914682.83	134262906.00	1.94%	29.13	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
25	12	136914682.83	134262906.00	1.94%	30.15	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
26	1	136914682.83	134262906.00	1.94%	31.33	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
27	13	136914682.83	134335885.00	1.88%	32.27	5	0.00	20198.00	66000.00	20198.00
							0.00	1105014.87	1823646.20	1105014.87
28	14	136914682.83	134344540.00	1.88%	33.03	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
29	6	136914682.83	134345662.00	1.88%	34.08	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
30	15	136874908.95	134345662.00	1.85%	35.23	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
31	3	136874908.95	134345662.00	1.85%	36.27	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
32	7	136874908.95	134364092.00	1.83%	37.20	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
33	16	136874908.95	134364092.00	1.83%	38.07	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
34	5	136874908.95	134506408.00	1.73%	38.73	4	0.00	20198.00	66000.00	20198.00
							0.00	1105014.87	1823646.20	1105014.87
35	17	136874908.95	134550960.00	1.70%	39.42	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
36	11	136874908.95	134595590.00	1.67%	39.98	9	0.00	10951.00	64200.00	10951.00
							0.00	879531.89	1823646.20	879531.89
37	1	136874908.95	134605209.00	1.66%	40.95	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
38	12	136874908.95	134605209.00	1.66%	42.18	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
39	4	136874908.95	134605209.00	1.66%	43.68	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
40	8	136874908.95	134607990.00	1.66%	45.18	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
41	18	136874908.95	134607990.00	1.66%	46.07	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
42	11	136874908.95	134670456.00	1.61%	47.08	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58
43	4	136874908.95	134670535.00	1.61%	48.10	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58
44	12	136874908.95	134670535.00	1.61%	49.18	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58
45	1	136874908.95	134670535.00	1.61%	50.40	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58

46	14	136874908.95	134678346.00	1.60%	51.63	9	0.00	7646.00	64200.00	7646.00
							0.00	770313.50	1823646.20	770313.50
47	13	136874908.95	134680970.00	1.60%	52.22	4	0.00	20198.00	66000.00	20198.00
							0.00	1105014.87	1823646.20	1105014.87
48	14	136874908.95	134686843.00	1.60%	53.00	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
49	3	136874908.95	134687965.00	1.60%	53.88	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
50	15	136874908.95	134687965.00	1.60%	54.55	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
51	6	136874908.95	134687965.00	1.60%	55.22	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
52	9	136874908.95	134690747.00	1.60%	55.92	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
53	19	136874908.95	134690747.00	1.60%	57.03	10	0.00	21323.00	64200.00	21323.00
							0.00	1142185.85	1823646.20	1142185.85
54	22	136874908.95	134695260.00	1.59%	58.05	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
55	5	136874908.95	134708247.00	1.58%	59.13	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
56	20	136874908.95	134708247.00	1.58%	59.85	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
57	21	136874908.95	134709177.00	1.58%	60.53	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
58	10	136874908.95	134743262.00	1.56%	61.05	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
59	23	136874908.95	134743262.00	1.56%	62.15	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
60	2	136874908.95	134743262.00	1.56%	63.18	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
61	27	136874908.95	134797971.00	1.52%	64.17	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
62	26	136874908.95	134797971.00	1.52%	65.45	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
63	25	136874908.95	134797971.00	1.52%	67.07	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
64	24	136874908.95	134797971.00	1.52%	68.57	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
65	25	136874908.95	134851143.00	1.48%	70.30	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
66	26	136874908.95	134851143.00	1.48%	70.80	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
67	33	136874908.95	134853478.00	1.48%	71.33	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
68	24	136874908.95	134858367.00	1.47%	72.45	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62

69	27	136874908.95	134858367.00	1.47%	73.15	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
70	32	136874908.95	134880728.00	1.46%	73.85	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
71	31	136874908.95	134880728.00	1.46%	74.80	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
72	30	136874908.95	134880728.00	1.46%	75.70	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
73	29	136874908.95	134880728.00	1.46%	76.42	7	0.00	30801.00	66000.00	30801.00
							0.00	1314011.62	1823646.20	1314011.62
74	13	136874908.95	134882809.00	1.46%	77.32	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
75	28	136874908.95	134882809.00	1.46%	78.30	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
76	16	136874908.95	134901479.00	1.44%	79.27	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
77	34	136874908.95	134901479.00	1.44%	80.42	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
78	7	136874908.95	134901479.00	1.44%	81.55	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
79	18	136874908.95	134950294.00	1.41%	82.93	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
80	8	136874908.95	134950294.00	1.41%	84.32	9	0.00	3305.00	64200.00	3305.00
							0.00	580181.42	1823646.20	580181.42
81	43	136874908.95	135005262.00	1.37%	85.68	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
82	46	136874908.95	135006384.00	1.37%	86.77	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
83	45	136874908.95	135006384.00	1.37%	87.67	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
84	44	136874908.95	135006384.00	1.37%	88.40	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
85	22	136874908.95	135014897.00	1.36%	89.17	13	0.00	31835.00	64200.00	31835.00
							0.00	1347853.64	1823646.20	1347853.64
86	8	136874908.95	135015620.00	1.36%	90.38	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58
87	18	136874908.95	135015620.00	1.36%	91.38	13	0.00	37600.00	64200.00	37600.00
							0.00	1446140.58	1823646.20	1446140.58
88	19	136874908.95	135033050.00	1.35%	92.98	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
89	9	136874908.95	135033050.00	1.35%	94.15	13	0.00	35602.00	64200.00	35602.00
							0.00	1412990.47	1823646.20	1412990.47
90	40	136874908.95	135035984.00	1.34%	94.88	7	0.00	31633.00	66000.00	31633.00
							0.00	1328752.67	1823646.20	1328752.67
91	39	136874908.95	135035984.00	1.34%	95.88	7	0.00	31633.00	66000.00	31633.00
							0.00	1328752.67	1823646.20	1328752.67

92	11	136874908.95	135044838.00	1.34%	97.18	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
93	1	136874908.95	135044917.00	1.34%	99.03	3	0.00	39459.00	66000.00	39459.00
							0.00	1458888.36	1823646.20	1458888.36
94	12	136874908.95	135044917.00	1.34%	99.98	3	0.00	39459.00	66000.00	39459.00
							0.00	1458888.36	1823646.20	1458888.36
95	4	136874908.95	135044917.00	1.34%	100.97	3	0.00	39459.00	66000.00	39459.00
							0.00	1458888.36	1823646.20	1458888.36
96	35	136874908.95	135053332.00	1.33%	101.85	1	0.00	21782.00	66000.00	21782.00
							0.00	1139245.70	1823646.20	1139245.70
97	38	136874908.95	135056073.00	1.33%	102.40	7	0.00	31633.00	66000.00	31633.00
							0.00	1328752.67	1823646.20	1328752.67
98	2	136874908.95	135062899.00	1.32%	103.65	13	0.00	31835.00	64200.00	31835.00
							0.00	1347853.64	1823646.20	1347853.64
99	23	136874908.95	135062899.00	1.32%	105.18	13	0.00	31835.00	64200.00	31835.00
							0.00	1347853.64	1823646.20	1347853.64
100	10	136874908.95	135062899.00	1.32%	106.78	13	0.00	31835.00	64200.00	31835.00
							0.00	1347853.64	1823646.20	1347853.64
101	17	136874908.95	135088347.00	1.31%	108.32	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
102	36	136874908.95	135088347.00	1.31%	109.17	10	0.00	22747.00	64200.00	22747.00
							0.00	1172637.14	1823646.20	1172637.14
103	14	136874908.95	135088534.00	1.31%	110.45	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
104	49	136874908.95	135088534.00	1.31%	111.53	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
105	6	136874908.95	135089656.00	1.30%	112.48	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
106	52	136874908.95	135089656.00	1.30%	113.80	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
107	15	136874908.95	135089656.00	1.30%	114.75	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
108	51	136874908.95	135089656.00	1.30%	116.20	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
109	3	136874908.95	135089656.00	1.30%	117.20	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56
110	50	136874908.95	135089656.00	1.30%	118.13	7	0.00	39279.00	66000.00	39279.00
							0.00	1456048.56	1823646.20	1456048.56

ANEXO III.3 - RESUMO DO PROCESSAMENTO

PROBLEMA DE LOCALIZACAO COM FUNCOES DE CUSTO NAO LINEARES

AUTOR: RONALDO RUST ORIENTADOR: CLAUDIO THOMAS BORNSTEIN

VERSAO 5 PROBLEMA 841 ** RESUMO DO PROCESSAMENTO **

CARACTERISTICAS:

54 NOS	23 PRODUTORES	243 ARCOS	5 CONSUMIDORES

NO. DE ITERACOES.....	110
VALOR DA FUNCAO OBJETIVO.....	136874908.95
TOLERANCIA.....	1.50 %
ERRO MAXIMO COMETIDO.....	1731852.95 => 1.27 %
TEMPO TOTAL GASTO.....	119.3500 SEG
OUT-OF-KILTER.....	107.3833 SEG
PRIMEIRA CHAMADA.....	11.0500 SEG
MEDIA DAS DEMAIS CHAMADAS.	0.4379 SEG/CHAMADA
ALGORITMO.....	11.9667 SEG
INICIALIZACAO.....	0.0833 SEG
MEDIA DAS ITERACOES.....	0.1080 SEG/ITERACAO

ANEXO III.4 - CENTROS PRODUTORES

VERSAO 5 PROBLEMA 841 ** CENTROS PRODUTORES **

	-NO-	-- NOME --	-- PRODUCAO --
1	2	ALTAMIRA	9902.00
2	3	BACABINHA	16114.00
3	4	BOM JARDIM	21782.00
4	5	LAGO PEDRA	38552.00
5	6	MONCAO	39459.00
6	7	PINDARE MIR.	10799.00
7	8	SANTA INES	9399.00
8	9	SANTA LUZIA	51128.00
9	10	VIT. FREIRE	14687.00
10	11	BACABAL	10646.00
11	12	ESPERANTOP.	7377.00
12	13	IGARAPE GRDE	3305.00
13	14	IPIXUNA	18463.00
14	15	JOSELANDIA	10946.00
15	16	LAGO JUNCO	9551.00
16	17	LAGO VERDE	2947.00
17	18	LIMA CAMPOS	2219.00
18	19	OLHO D'AGUA	3963.00
19	20	PEDREIRAS	1998.00
20	21	PIO XII	8882.00
21	22	POCAO PEDRA	6011.00
22	23	STO A. LOPES	11268.00
23	24	SAO MATEUS	8089.00
		**** TOTAL	317487.00

ANEXO III.5 - TRANSPORTE

VERSAO 5 PROBLEMA 841 ** T R A N S P O R T E **

	-- NOME --	--ARCO--	LIM. INF.	LIM. SUP.	CUSTO UNIT.	FUNC	I.ROT	CAPACIDADE	--FLUXO--	-- CUSTO --
1		2- 25	0.00	9999999.99	222.75	0	0.00	0.00	0.00	0.00
2		2- 26	0.00	9999999.99	113.05	0	0.00	1424.00	1424.00	160985.20
3		2- 28	0.00	9999999.99	222.75	0	0.00	0.00	0.00	0.00
4		2- 29	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
5		2- 30	0.00	9999999.99	145.20	0	0.00	0.00	0.00	0.00
6		2- 31	0.00	9999999.99	129.03	0	0.00	8478.00	8478.00	1093916.34
7		2- 32	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
8		2- 33	0.00	9999999.99	232.65	0	0.00	0.00	0.00	0.00
9		2- 34	0.00	9999999.99	156.00	0	0.00	0.00	0.00	0.00
10		3- 26	0.00	9999999.99	129.03	0	0.00	0.00	0.00	0.00
11		3- 27	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
12		3- 28	0.00	9999999.99	174.00	0	0.00	0.00	0.00	0.00
13		3- 29	0.00	9999999.99	150.00	0	0.00	0.00	0.00	0.00
14		3- 30	0.00	9999999.99	205.20	0	0.00	0.00	0.00	0.00
15		3- 31	0.00	9999999.99	37.95	0	0.00	16114.00	16114.00	611526.30
16		3- 32	0.00	9999999.99	144.00	0	0.00	0.00	0.00	0.00
17		3- 33	0.00	9999999.99	204.60	0	0.00	0.00	0.00	0.00
18		3- 34	0.00	9999999.99	108.00	0	0.00	0.00	0.00	0.00
19		4- 25	0.00	9999999.99	90.00	0	0.00	21782.00	21782.00	1960380.00
20		4- 27	0.00	9999999.99	205.20	0	0.00	0.00	0.00	0.00

21	4-	28	0.00	9999999.99	148.50	0	0.00	0.00	0.00	0.00
22	4-	29	0.00	9999999.99	231.00	0	0.00	0.00	0.00	0.00
23	4-	30	0.00	9999999.99	180.00	0	0.00	0.00	0.00	0.00
24	4-	34	0.00	9999999.99	267.30	0	0.00	0.00	0.00	0.00
25	5-	25	0.00	9999999.99	237.60	0	0.00	0.00	0.00	0.00
26	5-	26	0.00	9999999.99	54.00	0	0.00	38552.00	38552.00	2081808.00
27	5-	29	0.00	9999999.99	171.00	0	0.00	0.00	0.00	0.00
28	5-	30	0.00	9999999.99	163.35	0	0.00	0.00	0.00	0.00
29	5-	31	0.00	9999999.99	201.96	0	0.00	0.00	0.00	0.00
30	5-	34	0.00	9999999.99	181.50	0	0.00	0.00	0.00	0.00
31	5-	35	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
32	5-	36	0.00	9999999.99	188.10	0	0.00	0.00	0.00	0.00
33	6-	25	0.00	9999999.99	70.00	0	0.00	0.00	0.00	0.00
34	6-	27	0.00	9999999.99	60.00	0	0.00	39459.00	39459.00	2367540.00
35	6-	28	0.00	9999999.99	62.54	0	0.00	0.00	0.00	0.00
36	6-	29	0.00	9999999.99	63.00	0	0.00	0.00	0.00	0.00
37	6-	31	0.00	9999999.99	191.40	0	0.00	0.00	0.00	0.00
38	6-	34	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
39	7-	25	0.00	9999999.99	110.40	0	0.00	0.00	0.00	0.00
40	7-	26	0.00	9999999.99	238.00	0	0.00	0.00	0.00	0.00
41	7-	27	0.00	9999999.99	145.00	0	0.00	0.00	0.00	0.00
42	7-	28	0.00	9999999.99	67.00	0	0.00	0.00	0.00	0.00
43	7-	29	0.00	9999999.99	68.30	0	0.00	10799.00	10799.00	737571.70

44	7-	30	0.00	9999999.99	114.00	0	0.00	0.00	0.00	0.00
45	7-	31	0.00	9999999.99	189.00	0	0.00	0.00	0.00	0.00
46	7-	34	0.00	9999999.99	151.00	0	0.00	0.00	0.00	0.00
47	8-	25	0.00	9999999.99	113.85	0	0.00	0.00	0.00	0.00
48	8-	26	0.00	9999999.99	214.00	0	0.00	0.00	0.00	0.00
49	8-	27	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
50	8-	28	0.00	9999999.99	69.00	0	0.00	0.00	0.00	0.00
51	8-	29	0.00	9999999.99	54.00	0	0.00	9399.00	9399.00	507546.00
52	8-	30	0.00	9999999.99	134.00	0	0.00	0.00	0.00	0.00
53	8-	31	0.00	9999999.99	160.00	0	0.00	0.00	0.00	0.00
54	8-	33	0.00	9999999.99	225.00	0	0.00	0.00	0.00	0.00
55	8-	34	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
56	9-	25	0.00	9999999.99	264.00	0	0.00	0.00	0.00	0.00
57	9-	26	0.00	9999999.99	202.50	0	0.00	0.00	0.00	0.00
58	9-	28	0.00	9999999.99	252.00	0	0.00	0.00	0.00	0.00
59	9-	30	0.00	9999999.99	80.00	0	0.00	51128.00	51128.00	4090240.00
60	9-	31	0.00	9999999.99	193.00	0	0.00	0.00	0.00	0.00
61	9-	34	0.00	9999999.99	217.80	0	0.00	0.00	0.00	0.00
62	10-	25	0.00	9999999.99	109.60	0	0.00	0.00	0.00	0.00
63	10-	26	0.00	9999999.99	112.20	0	0.00	0.00	0.00	0.00
64	10-	27	0.00	9999999.99	204.60	0	0.00	0.00	0.00	0.00
65	10-	28	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
66	10-	29	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00

67		10-	31	0.00	9999999.99	32.00	0	0.00	14687.00	14687.00	469984.00
68		10-	32	0.00	9999999.99	166.50	0	0.00	0.00	0.00	0.00
69		10-	33	0.00	9999999.99	173.25	0	0.00	0.00	0.00	0.00
70		10-	34	0.00	9999999.99	110.40	0	0.00	0.00	0.00	0.00
71		11-	26	0.00	9999999.99	189.00	0	0.00	0.00	0.00	0.00
72		11-	27	0.00	9999999.99	202.50	0	0.00	0.00	0.00	0.00
73		11-	28	0.00	9999999.99	175.00	0	0.00	0.00	0.00	0.00
74		11-	29	0.00	9999999.99	162.00	0	0.00	0.00	0.00	0.00
75		11-	31	0.00	9999999.99	144.00	0	0.00	0.00	0.00	0.00
76		11-	32	0.00	9999999.99	45.00	0	0.00	10646.00	10646.00	479070.00
77		11-	33	0.00	9999999.99	66.00	0	0.00	0.00	0.00	0.00
78		11-	34	0.00	9999999.99	120.00	0	0.00	0.00	0.00	0.00
79		11-	35	0.00	9999999.99	112.20	0	0.00	0.00	0.00	0.00
80		11-	36	0.00	9999999.99	184.80	0	0.00	0.00	0.00	0.00
81		12-	26	0.00	9999999.99	227.70	0	0.00	0.00	0.00	0.00
82		12-	33	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
83		12-	35	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
84		12-	36	0.00	9999999.99	60.72	0	0.00	0.00	0.00	0.00
85		12-	37	0.00	9999999.99	132.00	0	0.00	7377.00	7377.00	973764.00
86		13-	26	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
87		13-	32	0.00	9999999.99	165.00	0	0.00	0.00	0.00	0.00
88		13-	33	0.00	9999999.99	138.60	0	0.00	0.00	0.00	0.00
89		13-	35	0.00	9999999.99	91.08	0	0.00	3305.00	3305.00	301019.40

90	13-	36	0.00	9999999.99	75.90	0	0.00	0.00	0.00	0.00
91	13-	37	0.00	9999999.99	191.40	0	0.00	0.00	0.00	0.00
92	14-	28	0.00	9999999.99	227.70	0	0.00	0.00	0.00	0.00
93	14-	29	0.00	9999999.99	195.00	0	0.00	0.00	0.00	0.00
94	14-	32	0.00	9999999.99	48.00	0	0.00	18463.00	18463.00	886224.00
95	14-	33	0.00	9999999.99	41.00	0	0.00	0.00	0.00	0.00
96	14-	34	0.00	9999999.99	162.00	0	0.00	0.00	0.00	0.00
97	14-	35	0.00	9999999.99	68.31	0	0.00	0.00	0.00	0.00
98	14-	36	0.00	9999999.99	204.60	0	0.00	0.00	0.00	0.00
99	15-	32	0.00	9999999.99	207.90	0	0.00	0.00	0.00	0.00
100	15-	33	0.00	9999999.99	214.50	0	0.00	0.00	0.00	0.00
101	15-	35	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
102	15-	36	0.00	9999999.99	106.26	0	0.00	0.00	0.00	0.00
103	15-	37	0.00	9999999.99	132.00	0	0.00	10946.00	10946.00	1444872.00
104	16-	26	0.00	9999999.99	125.40	0	0.00	0.00	0.00	0.00
105	16-	31	0.00	9999999.99	183.15	0	0.00	0.00	0.00	0.00
106	16-	32	0.00	9999999.99	12.10	0	0.00	9551.00	9551.00	115567.10
107	16-	33	0.00	9999999.99	112.20	0	0.00	0.00	0.00	0.00
108	16-	34	0.00	9999999.99	191.40	0	0.00	0.00	0.00	0.00
109	16-	35	0.00	9999999.99	158.40	0	0.00	0.00	0.00	0.00
110	16-	36	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
111	16-	37	0.00	9999999.99	188.10	0	0.00	0.00	0.00	0.00
112	17-	26	0.00	9999999.99	186.00	0	0.00	0.00	0.00	0.00

113	17-	27	0.00	9999999.99	183.15	0	0.00	0.00	0.00	0.00
114	17-	28	0.00	9999999.99	167.40	0	0.00	0.00	0.00	0.00
115	17-	29	0.00	9999999.99	173.40	0	0.00	0.00	0.00	0.00
116	17-	30	0.00	9999999.99	227.70	0	0.00	0.00	0.00	0.00
117	17-	31	0.00	9999999.99	165.00	0	0.00	0.00	0.00	0.00
118	17-	32	0.00	9999999.99	121.44	0	0.00	2947.00	2947.00	357883.68
119	17-	33	0.00	9999999.99	145.20	0	0.00	0.00	0.00	0.00
120	17-	34	0.00	9999999.99	110.40	0	0.00	0.00	0.00	0.00
121	17-	35	0.00	9999999.99	187.00	0	0.00	0.00	0.00	0.00
122	17-	36	0.00	9999999.99	227.70	0	0.00	0.00	0.00	0.00
123	18-	32	0.00	9999999.99	157.50	0	0.00	0.00	0.00	0.00
124	18-	33	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
125	18-	35	0.00	9999999.99	13.49	0	0.00	2219.00	2219.00	29934.31
126	18-	36	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
127	18-	37	0.00	9999999.99	214.50	0	0.00	0.00	0.00	0.00
128	19-	25	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
129	19-	26	0.00	9999999.99	156.00	0	0.00	0.00	0.00	0.00
130	19-	27	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
131	19-	28	0.00	9999999.99	126.00	0	0.00	0.00	0.00	0.00
132	19-	29	0.00	9999999.99	114.00	0	0.00	0.00	0.00	0.00
133	19-	31	0.00	9999999.99	75.90	0	0.00	0.00	0.00	0.00
134	19-	32	0.00	9999999.99	126.00	0	0.00	0.00	0.00	0.00
135	19-	33	0.00	9999999.99	174.20	0	0.00	0.00	0.00	0.00

136	19-	34	0.00	9999999.99	18.00	0	0.00	3963.00	3963.00	71334.00
137	19-	35	0.00	9999999.99	178.20	0	0.00	0.00	0.00	0.00
138	20-	32	0.00	9999999.99	176.00	0	0.00	0.00	0.00	0.00
139	20-	33	0.00	9999999.99	102.00	0	0.00	0.00	0.00	0.00
140	20-	34	0.00	9999999.99	183.15	0	0.00	0.00	0.00	0.00
141	20-	35	0.00	9999999.99	41.00	0	0.00	1998.00	1998.00	81918.00
142	20-	36	0.00	9999999.99	145.20	0	0.00	0.00	0.00	0.00
143	20-	37	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
144	21-	25	0.00	9999999.99	213.18	0	0.00	0.00	0.00	0.00
145	21-	26	0.00	9999999.99	186.00	0	0.00	0.00	0.00	0.00
146	21-	27	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
147	21-	28	0.00	9999999.99	96.60	0	0.00	0.00	0.00	0.00
148	21-	29	0.00	9999999.99	75.90	0	0.00	0.00	0.00	0.00
149	21-	31	0.00	9999999.99	82.10	0	0.00	0.00	0.00	0.00
150	21-	32	0.00	9999999.99	156.00	0	0.00	0.00	0.00	0.00
151	21-	33	0.00	9999999.99	190.00	0	0.00	0.00	0.00	0.00
152	21-	34	0.00	9999999.99	41.30	0	0.00	8882.00	8882.00	366826.60
153	21-	35	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
154	22-	26	0.00	9999999.99	188.10	0	0.00	0.00	0.00	0.00
155	22-	32	0.00	9999999.99	185.00	0	0.00	0.00	0.00	0.00
156	22-	33	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
157	22-	35	0.00	9999999.99	138.60	0	0.00	0.00	0.00	0.00
158	22-	36	0.00	9999999.99	45.00	0	0.00	0.00	0.00	0.00

159	22-	37	0.00	9999999.99	118.80	0	0.00	6011.00	6011.00	714106.80
160	23-	32	0.00	9999999.99	268.95	0	0.00	0.00	0.00	0.00
161	23-	33	0.00	9999999.99	237.60	0	0.00	0.00	0.00	0.00
162	23-	35	0.00	9999999.99	132.00	0	0.00	0.00	0.00	0.00
163	23-	36	0.00	9999999.99	118.80	0	0.00	0.00	0.00	0.00
164	23-	37	0.00	9999999.99	41.00	0	0.00	11268.00	11268.00	461988.00
165	24-	25	0.00	9999999.99	316.80	0	0.00	0.00	0.00	0.00
166	24-	26	0.00	9999999.99	244.80	0	0.00	0.00	0.00	0.00
167	24-	32	0.00	9999999.99	156.00	0	0.00	8089.00	8089.00	1261884.00
168	24-	33	0.00	9999999.99	182.40	0	0.00	0.00	0.00	0.00
169	24-	34	0.00	9999999.99	198.00	0	0.00	0.00	0.00	0.00
170	24-	35	0.00	9999999.99	189.00	0	0.00	0.00	0.00	0.00
171	38-	51	0.00	9999999.99	417.60	0	0.00	0.00	0.00	0.00
172	38-	53	0.00	9999999.99	288.00	0	0.00	0.00	0.00	0.00
173	38-	54	0.00	9999999.99	440.00	0	0.00	0.00	0.00	0.00
174	38-	1	0.00	9999999.99	324.00	0	0.00	21782.00	21782.00	7057368.00
175	39-	51	0.00	9999999.99	309.00	0	0.00	8228.00	8228.00	2542452.00
176	39-	52	0.00	9999999.99	430.00	0	0.00	31748.00	31748.00	13651640.00
177	39-	53	0.00	9999999.99	366.00	0	0.00	0.00	0.00	0.00
178	39-	54	0.00	9999999.99	417.00	0	0.00	0.00	0.00	0.00
179	39-	1	0.00	9999999.99	426.00	0	0.00	0.00	0.00	0.00
180	40-	51	0.00	9999999.99	475.20	0	0.00	0.00	0.00	0.00
181	40-	53	0.00	9999999.99	306.00	0	0.00	0.00	0.00	0.00

182	40-	54	0.00	9999999.99	423.00	0	0.00	0.00	0.00	0.00
183	40-	1	0.00	9999999.99	312.00	0	0.00	39459.00	39459.00	12311208.00
184	41-	51	0.00	9999999.99	372.00	0	0.00	0.00	0.00	0.00
185	41-	53	0.00	9999999.99	280.80	0	0.00	0.00	0.00	0.00
186	41-	54	0.00	9999999.99	402.00	0	0.00	0.00	0.00	0.00
187	41-	1	0.00	9999999.99	348.00	0	0.00	0.00	0.00	0.00
188	42-	51	0.00	9999999.99	372.00	0	0.00	0.00	0.00	0.00
189	42-	53	0.00	9999999.99	270.00	0	0.00	20198.00	20198.00	5453460.00
190	42-	54	0.00	9999999.99	390.00	0	0.00	0.00	0.00	0.00
191	42-	1	0.00	9999999.99	338.40	0	0.00	0.00	0.00	0.00
192	43-	51	0.00	9999999.99	297.00	0	0.00	7646.00	7646.00	2270862.00
193	43-	52	0.00	9999999.99	495.00	0	0.00	0.00	0.00	0.00
194	43-	53	0.00	9999999.99	300.00	0	0.00	43299.00	43299.00	12989700.00
195	43-	54	0.00	9999999.99	474.00	0	0.00	0.00	0.00	0.00
196	43-	1	0.00	9999999.99	363.00	0	0.00	183.00	183.00	66429.00
197	44-	51	0.00	9999999.99	330.00	0	0.00	0.00	0.00	0.00
198	44-	52	0.00	9999999.99	510.00	0	0.00	0.00	0.00	0.00
199	44-	53	0.00	9999999.99	315.00	0	0.00	0.00	0.00	0.00
200	44-	54	0.00	9999999.99	363.00	0	0.00	39279.00	39279.00	14258277.00
201	44-	1	0.00	9999999.99	366.00	0	0.00	0.00	0.00	0.00
202	45-	51	0.00	9999999.99	400.00	0	0.00	0.00	0.00	0.00
203	45-	52	0.00	9999999.99	545.00	0	0.00	0.00	0.00	0.00
204	45-	53	0.00	9999999.99	333.00	0	0.00	0.00	0.00	0.00

205	45-	54	0.00	9999999.99	313.50	0	0.00	0.00	0.00	0.00
206	45-	1	0.00	9999999.99	300.00	0	0.00	49696.00	49696.00	14908800.00
207	46-	51	0.00	9999999.99	420.00	0	0.00	0.00	0.00	0.00
208	46-	52	0.00	9999999.99	544.50	0	0.00	0.00	0.00	0.00
209	46-	53	0.00	9999999.99	354.00	0	0.00	0.00	0.00	0.00
210	46-	54	0.00	9999999.99	316.00	0	0.00	0.00	0.00	0.00
211	46-	1	0.00	9999999.99	321.00	0	0.00	0.00	0.00	0.00
212	47-	51	0.00	9999999.99	354.00	0	0.00	0.00	0.00	0.00
213	47-	52	0.00	9999999.99	534.00	0	0.00	0.00	0.00	0.00
214	47-	53	0.00	9999999.99	342.00	0	0.00	0.00	0.00	0.00
215	47-	54	0.00	9999999.99	333.00	0	0.00	12845.00	12845.00	4277385.00
216	47-	1	0.00	9999999.99	342.00	0	0.00	0.00	0.00	0.00
217	48-	51	0.00	9999999.99	447.00	0	0.00	0.00	0.00	0.00
218	48-	52	0.00	9999999.99	491.40	0	0.00	0.00	0.00	0.00
219	48-	53	0.00	9999999.99	375.00	0	0.00	0.00	0.00	0.00
220	48-	54	0.00	9999999.99	310.20	0	0.00	7522.00	7522.00	2333324.40
221	48-	1	0.00	9999999.99	330.00	0	0.00	0.00	0.00	0.00
222	49-	51	0.00	9999999.99	580.00	0	0.00	0.00	0.00	0.00
223	49-	52	0.00	9999999.99	400.00	0	0.00	0.00	0.00	0.00
224	49-	53	0.00	9999999.99	423.00	0	0.00	0.00	0.00	0.00
225	49-	54	0.00	9999999.99	369.60	0	0.00	0.00	0.00	0.00
226	49-	1	0.00	9999999.99	372.00	0	0.00	0.00	0.00	0.00
227	50-	52	0.00	9999999.99	460.00	0	0.00	0.00	0.00	0.00

228	50-	53	0.00	9999999.99	510.00	0	0.00	0.00	0.00	0.00
229	50-	54	0.00	9999999.99	284.40	0	0.00	35602.00	35602.00	10125208.80
230	50-	1	0.00	9999999.99	417.00	0	0.00	0.00	0.00	0.00

**** TOTAL DO FLUXO 634974.00

**** CUSTO DE TRANSPORTE 123874001.63

ANEXO III.6 - ARMAZENAGEM

VERSAO 5 PROBLEMA 841 ** A R M A Z E N A G E M **

	-- NOME --	--ARCO--	LIM. INF.	LIM. SUP.	CUSTO UNIT.	FUNC	I.ROT	CAPACIDADE	--FLUXO--	-- CUSTO --
1	BOM JARDIM	25- 38	0.00	60000.00	0.00	103	1.10	19801.82	21782.00	1139245.70
2	LAGO PEDRA	26- 39	0.00	60000.00	0.00	103	1.10	36341.82	39976.00	1467009.05
3	MONCAO	27- 40	0.00	60000.00	0.00	103	1.10	35871.82	39459.00	1458888.36
4	PINDARE MIR.	28- 41	0.00	60000.00	0.00	103	1.10	0.00	0.00	0.00
5	SANTA INES	29- 42	0.00	60000.00	0.00	103	1.10	18561.82	20198.00	1105014.87
6	SANTA LUZIA	30- 43	0.00	60000.00	0.00	103	1.10	46480.00	51128.00	1630882.33
7	VIT. FREIRE	31- 44	0.00	60000.00	0.00	103	1.10	35708.18	39279.00	1456048.56
8	BACABAL	32- 45	0.00	60000.00	0.00	103	1.07	46444.86	49696.00	1630347.10
9	IPIXUNA	33- 46	0.00	60000.00	0.00	103	1.07	0.00	0.00	0.00
10	OLHO D'AGUA	34- 47	0.00	60000.00	0.00	103	1.07	12004.67	12845.00	934686.68
11	PEDREIRAS	35- 48	0.00	60000.00	0.00	103	1.07	7029.91	7522.00	765794.20
12	POCAO PEDRAS	36- 49	0.00	60000.00	0.00	103	1.07	0.00	0.00	0.00
13	STO A. LOPES	37- 50	0.00	60000.00	0.00	103	1.07	33272.90	35602.00	1412990.47

**** TOTAL DO FLUXO 317487.00

**** CUSTO DE ARMAZENAGEM 13000907.32

ANEXO III.7 - CENTROS CONSUMIDORES

VERSAO 5 PROBLEMA 841 ** CENTROS CONSUMIDORES **

	-NO-	-- NOME --	-- CONSUMO --
1	51	IMPERATRIZ	15874.00
2	52	PORTO FRANCO	31748.00
3	53	CAND. MENDES	63497.00
4	54	TIMON	95248.00
5	1	SAO LUIZ	111120.00
		**** TOTAL	317487.00