PROBLEMAS DE FLUXO DE CUSTO MÍNIMO EM REDES DE CUSTO CÔNCAVO

MÁRCIA APARECIDA GOMES

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO

CAMPINAS - SÃO PAULO
BRASIL

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

COORDENAÇÃO DOS CURSOS DE PÓS-GRADUAÇÃO

AUTORIZAÇÃO PARA QUE A UNICAMP POSSA FORNECER, A PREÇO DE CUSTO, CÓPIAS DA TESE A INTERESSADOS

Nome do **Aluno:** MÁRCIA APARECIDA GOMES
Nº de **Identificação:** 785222
Endereço para Correspondência: DMECC - UNICAMP - CAIXA POSTAL 1170
Curso: OTIMIZAÇÃO E PESQUISA OPERACIONAL
Nome do Orientador: PAULO MORELATO FRANÇA
Título da Dissertação ou Tese: PROBLEMAS DE FLUXO DE CUSTO MÍNIMO EM REDES DE CUSTO CÔNCAVO.

Data proposta para a Defesa: 14/08/81

(O Aluno deverá assinar um dos 3 ítens abaixo)

1) Autorizo a Universidade Estadual de Campinas a partir desta data, a fornecer, a preço de custo, cópias de minha Dissertação ou Tese a interessados.

31/07/81 — Data

Márcia A. Gomes — assinatura do aluno

2) Autorizo a Universidade Estadual de Campinas, a fornecer, a partir de dois anos após esta data, a preço de custo, cópias de minha Dissertação ou Tese a interessados.

__/__/__ — Data

_____ — assinatura do aluno

3) Solicito que a Universidade Estadual de Campinas me consulte, dois anos após esta data, quanto à minha autorização para o fornecimento de cópias de minha Dissertação ou Tese, a preço de custo, a interessados.

__/__/__ — Data

_____ — assinatura do aluno

DE ACORDO

Orientador

PROBLEMAS DE FLUXO DE CUSTO MÍNIMO EM REDES DE CUSTO CÔNCAVO

MÁRCIA APARECIDA GOMES

Orientador:
Prof. Dr. PAULO MORELATO FRANÇA

Dissertação apresentada no Instituto de Matemática, Estatística e Ciência da Computação, como requisito parcial para obtenção do título de Mestre em Matemática Aplicada.

Agosto/1981

Aos meus pais.

AGRADECIMENTOS

Agradeço aos meus pais, aos meus irmãos e ao Sérgio, pelo amor, carinho, compreensão e apoio que me dedicaram neste período.

Aos amigos, que partilharam comigo os momentos bons e os momentos difíceis, pelo companheirismo e solidariedade.

Ao França, pelo incentivo e orientação deste trabalho.

Enfim, agradeço a todos que direta ou indiretamente me auxiliaram nesta pesquisa.

INTRODUÇÃO

Apresenta-se, neste trabalho, um estudo sobre os problemas de fluxo de custo mínimo numa rede quando os custos sobre os arcos da rede são côncavos.

Este tipo de problema é de difícil solução para problemas de médio e grande porte, devido a existência de muitos ótimos locais.

É feito um levantamento dos principais algorítmos que resolvem tais problemas, procurando relacioná-los com os algorítmos específicos para resolver os problemas de transporte com custo fixo ("fixed-charge") e os problemas de localização.

Desenvolve-se também um algorítmo do tipo separação e avaliação ("BRANCH AND BOUND") que é usado para resolver um problema real de escoamento de safra de milho da região centro-brasileira, considerando a armazenagem do produto.

CAPÍTULO I

PROBLEMAS DE FLUXO DE CUSTO MÍNIMO COM FUNÇÃO CUSTO CÔNCAVA: MODELOS E TÉCNICAS

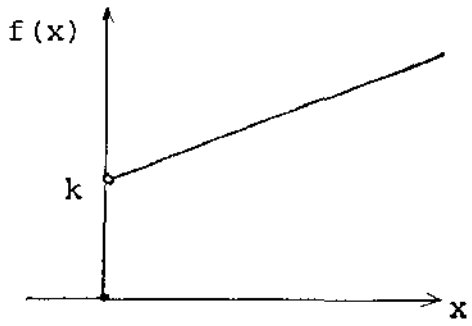
1.1) INTRODUÇÃO:

O objetivo deste capítulo é fornecer uma visão dos problemas de fluxo de custo mínimo, onde as funções custo são côncavas.

O interesse em se estudar tais problemas deriva do fato de que as suposições de funções custo lineares ou convexas, nem sempre estão próximas da realidade para alguns problemas práticos, como por exemplo, os problemas onde se distingue um custo fixo e um variável em suas estruturas de custo, ou aqueles onde se observam fenômenos de economias de escala, ou seja os custos unitários diminuem à medida em que se aumenta o nível de atividade.

As seguintes funções côncavas refletem esta situação:

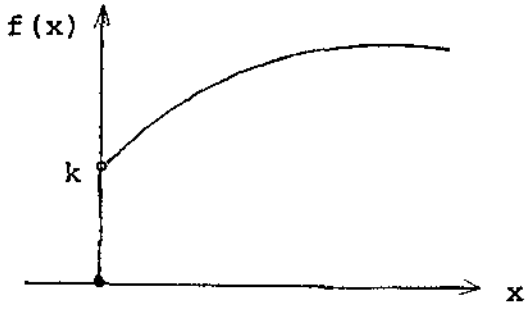
a) Função linear com custo fixo - "fixed-charge"



$$f(x) = \begin{cases} c.x + k & x > 0 \\ 0 & x = 0 \end{cases}$$

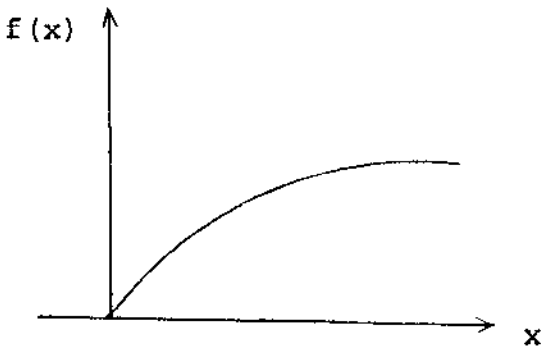
b) Função estritamente côncava:

b.1) com custo fixo



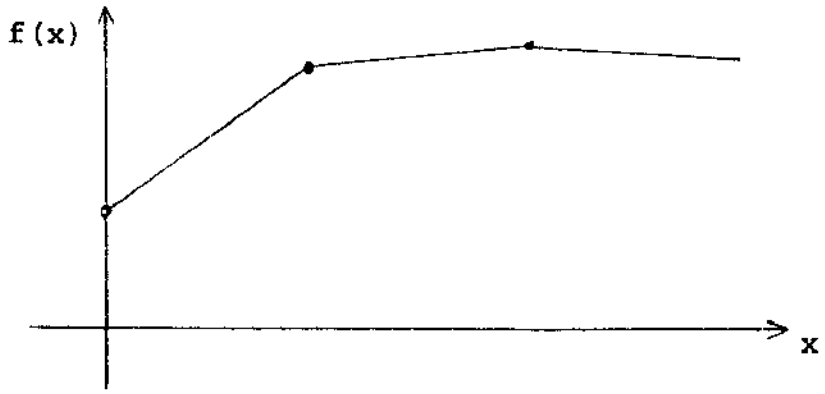
$$f(x) = \begin{cases} g(x) + k & x > 0 \\ 0 & x = 0 \end{cases}$$

b.2) sem custo fixo

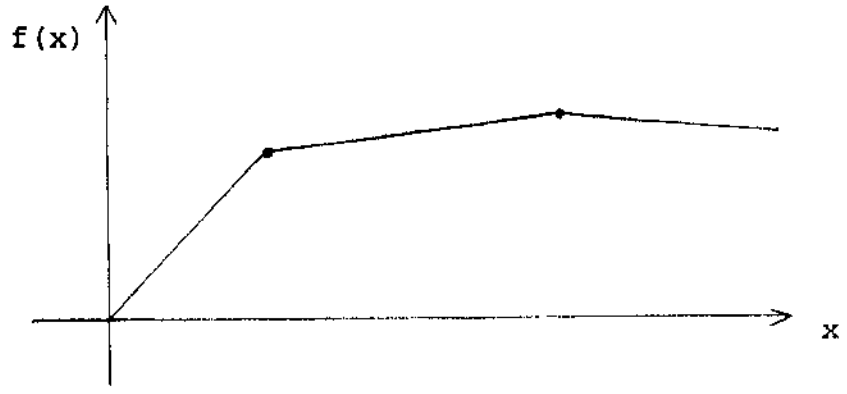


c) Função linear por partes

c.1) com custo fixo



c.2) sem custo fixo



Esta troca de funções custo convexas para côncavas afeta as características da solução de um problema de fluxo bem como os procedimentos para se obter a solução ótima.

Para se ter uma idéia das dificuldades introduzidas por estas funções, basta dizer que os algorítmos existentes para a re-

solução de problemas com funções convexas podem obter a solução ótima para redes com milhares de arcos em pouco tempo de execução, enquanto que a presença de pouco mais que 100 arcos com custo côncavo pode resultar em custos proibitivos de computação devidos ao tempo dispendido nesta execução.

Esta dificuldade de resolução decorre do fato dos problemas em questão serem não-convexos (minimizar uma função côncava sobre um conjunto convexo) admitindo portanto muitos mínimos locais.

Um procedimento ótimo de resolução deverá de algum modo localizar o mínimo global sobre todos os mínimos locais existentes.

As técnicas propostas são geralmente de natureza combinatória, e se fundamentam na estrutura especial de cada problema analisado. As experiências computacionais revelam que estas técnicas são aplicáveis somente a problemas de pequeno porte.

Alguns modelos e técnicas de resolução serão apresentados com o intuito de se mostrar um panorama geral da área de problemas de fluxo de custo mínimo com custo côncavo.

Esta área esteve, inicialmente, limitada à análise de problemas de localização e problemas de transporte com custo fixo,que são problemas particulares de fluxo com custo côncavo.

Dado o interesse despertado por estes problemas, houve um grande desenvolvimento na área específica de localização que pode ser aproveitada para o estudo de problemas mais gerais com custos côncavos.

Atualmente, é vasta a literatura em programação côncava, sendo que alguns algorítmos propostos para problemas gerais são facilmente adaptados aos problemas de fluxo.

1.2) PROBLEMAS DE LOCALIZAÇÃO E TRANSPORTE:

O estudo deste tipo especial de problema se justifica da do que a sua metodologia de solução é aplicável a problemas mais gerais de fluxo de custo mínimo com custo côncavo.

O problema de localização consiste em determinar:

a) Um subconjunto de locais, entre todos locais possíveis, onde deverão ser instaladas determinadas facilidades (fábricas ou armazéns).

b) Quais as quantidades que deverão ser transportadas destas facilidades aos centros consumidores de forma a atender as demandas nestes centros a um custo mínimo.

O que coloca estes problemas na classe dos problemas de fluxo de custo mínimo côncavo é a forma especial da função objetivo.

Esta função envolve os custos de transporte e os custos sobre as facilidades. Em geral, os primeiros são lineares, já as funções custo sobre as facilidades refletem economias de escala, sendo que o fator variável destas funções engloba os custos de produção ou armazenagem, custos de operação e manutenção das facilidades consideradas, etc., e o custo fixo para se construir a facilidade.

Na formulação do problema, poderão haver ou não restrições de capacidade sobre as facilidades, o que separa os problemas de localização em duas categorias, uma para os problemas capacitados e a outra para os não capacitados.

Os algorítmos utilizados para a resolução de tais proble

mas são de vários tipos: heurísticos, de separação e avaliação, de programação mista e os de decomposição.

Inicialmente serão colocados alguns trabalhos onde técnicas heurísticas foram utilizadas.

1.2.1) ALGORÍTMOS HEURÍSTICOS:

Os algorítmos heurísticos contentam-se em obter boas soluções para os problemas em que são aplicados, dado que para estes casos, fatores como a capacidade de memória do computador e o tempo de execução estão sendo priorizados.

As vantagens do emprego destas técnicas são as seguintes, como colocam Kuhen e Hamburger [18] .

1) Simplicidade computacional que resulta numa redução substancial no tempo de execução permitindo o tratamento de problemas de grande porte.

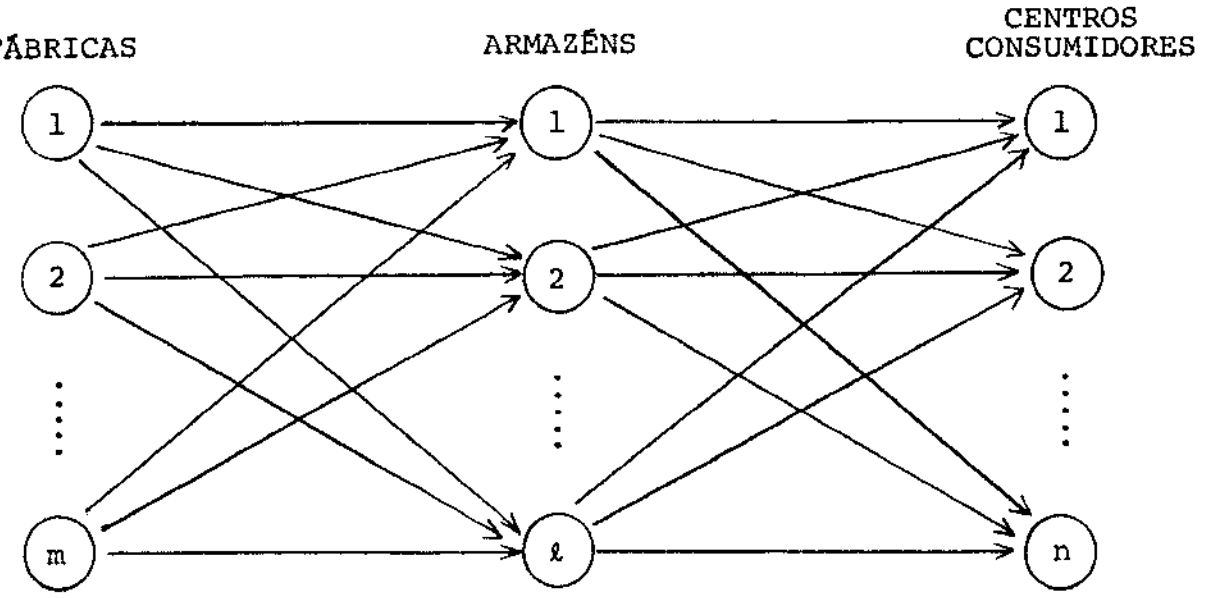
2) Flexibilidade com relação às funções custo, eliminando a necessidade de suposições restritivas.

Nos primeiros trabalhos publicados sobre problemas de localização estas técnicas foram amplamente empregadas; atualmente, dada a existência de eficientes algorítmos exatos, justifica-se o emprego destas técnicas somente a problemas em que a obtenção da solução ótima seja impraticável.

O trabalho sobre problemas de localização de armazéns apresentado por Baumol e Wolfe [5] em 1957, parece ser o pioneiro desta categoria.

No problema por eles proposto, alguns armazéns deverão

ser alugados entre os vários armazéns públicos disponíveis, de forma que estes sejam centros intermediários entre as fábricas e os centros consumidores, como mostra o esquema abaixo:



Baumol e Wolfe não consideram relevantes as limitações de capacidade dos armazéns, dado que o espaço requerido por uma fábrica consiste em uma pequena parcela desta capacidade.

Propõem uma técnica heurística para obter um ótimo local, que requer a resolução de um problema de transporte a cada iteração.

FORMULAÇÃO DO PROBLEMA:

$$\text{Min: } Z = \sum_{ijk} (c_{ij}+d_{jk}) \cdot x_{ijk} + \sum_{j} W_j \left(\sum_{ik} x_{ijk}\right)^{q} + \sum_{j} F_j \cdot y_j \qquad 0 < q < 1$$

$$\text{s.a:} \begin{cases} \sum_{jk} x_{ijk} = S_i & i = 1 \ldots m \\ \sum_{ij} x_{ijk} = D_k & k = 1 \ldots n \\ x_{ijk} \geq 0 & \forall\ i = 1 \ldots m \\ & \quad j = 1 \ldots \ell \\ & \quad k = 1 \ldots n \\ y_j = 0, 1 & j = 1 \ldots \ell \end{cases}$$

onde:

x_{ijk}: quantidade transportada da fábrica i, para o centro consumidor k, via armazém j.

S_i: quantidade disponível na fábrica i.

D_k: quantidade requerida no centro consumidor k.

c_{ij}: custo de transporte, unitário, da fábrica i para o armazém j.

d_{jk}: custo de transporte, unitário, do armazém j para o centro consumidor k.

$W_j(.)$: função custo de armazenagem.

$$y_j = 1 \quad \text{se} \quad \sum_{ik} x_{ijk} > 0$$

$$y_j = 0 \quad \text{se} \quad \sum_{ik} x_{ijk} = 0$$

F_j: custo fixo incorrido ao se alugar o armazém j.

Portanto as funções custo de transporte são lineares e as funções custo de armazenagem são côncavas envolvendo ou não custos fixos.

O algorítmo proposto é o seguinte:

INICIALIZAÇÃO:

Para toda fábrica i, $i = 1,\ldots,m$, e centro de consumo k, $k = 1,\ldots,n$, determinar o menor custo de transporte:

$$c_{ik}^{0} = \min_{j} (c_{ij} + d_{jk})$$

Com estes custos, resolver um problema de transporte das fábricas para os centros de consumo.

ITERAÇÃO r:

Seja $\{x_{ik}^{r-1}\}$ a solução obtida na iteração $r-1$.

Seja $\sum_{i,k} x_{ijk}^{r-1}$ o fluxo total no armazém j na iteração r-1.

a) Calcular os custos marginais de armazenagem:

Custo total de armazenagem no armazém j, é dado por:

$$W_j \cdot (\sum_{i,k} x_{ijk})^{q} \quad \text{com} \quad 0 < q < 1$$

O custo marginal será:

$$\frac{d\, W_j \, (\sum_{i,k} x_{ijk})^{q}}{d \sum_{i,k} x_{ijk}} = q \cdot W_j \cdot (\sum_{i,k} i_{jk})^{q-1}$$

b) Obter os novos custos para o problema de transporte:

Para cada fábrica i, $i = 1,\ldots,n$, e cada centro de consumo k, $k = 1,\ldots,n$, este custo é dado por:

$$c_{ik}^{r} = \min_{j} (c_{ij} + d_{jk} + q \cdot W_j \cdot (\sum_{i,k} x_{ijk}^{r-1})^{q-1})$$

c) Resolver o novo problema de transporte.

TESTE DE OTIMALIDADE:

Se $\sum_{i,k} x_{ijk}^{r-1} = \sum_{i,k} x_{ijk}^{r}$, a solução da iteração r é ótima.

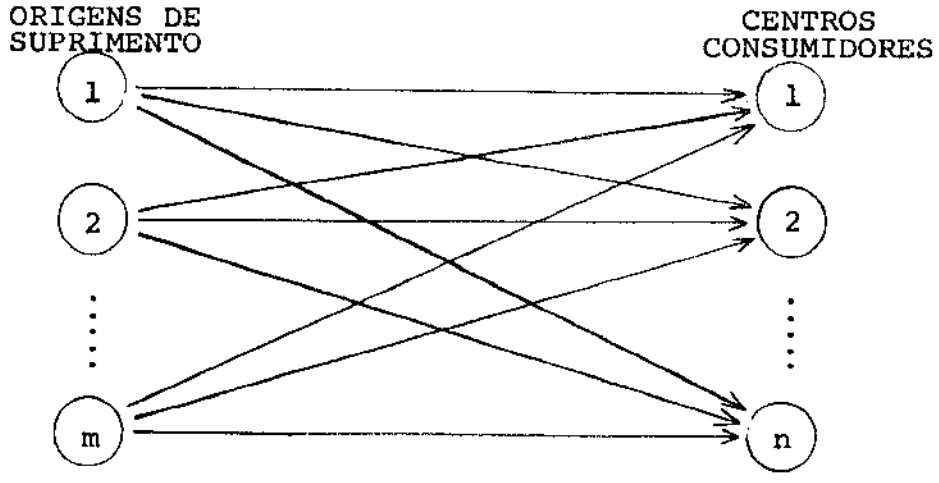
Caso contrário, executar a próxima iteração.

Um problema exemplo envolvendo duas fábricas, cinco armazéns e oito centros consumidores, foi resolvido à mão. Os custos de armazenagem são dados pela raiz quadrada do volume armazenado e não envolvem custos fixos.

A solução obtida não é satisfatória, pois soluções melhores foram obtidas, posteriormente, por outras técnicas heurísticas.

Uma última observação a ser feita é com relação aos custos fixos, que parecem ser esquecidos pelo algorítmo, dado que em nenhum passo eles são considerados, e que o cálculo das derivadas e o algorítmo de transporte não são bem designados para trabalhar com funções envolvendo elementos fixos.

O problema proposto por Manne [20], consiste em instalar origens de suprimento (tais como fábricas, armazéns) que deverão atender certos centros consumidores. Manne não considera centros intermediários e o problema pode ser assim esquematizado:



A formulação do problema é a seguinte:

$$\begin{cases} \text{Min: } Z = \sum_{ij} b_{ij} \cdot x_{ij} + \sum_{i} F_i \cdot y_i \\ \\ \text{s.a: } \begin{cases} \sum_{i} x_{ij} = D_j \quad j = 1,\ldots,n \\ \\ \text{Se } \begin{cases} y_i = 0 \\ y_i = 1 \end{cases} \quad \text{então} \quad \begin{cases} x_{ij} = 0 \\ x_{ij} \geq 0 \end{cases} \quad i = 1\ldots m \\ \\ x_{ij} \geq 0 \qquad y_i = 0 \text{ ou } 1 \end{cases} \end{cases}$$

sendo:

x_{ij}: quantidade transportada da origem i ao centro consumidor j.

y_i = 1 se uma fábrica (ou armazém) for instalada no local i.

0 caso contrário.

b_{ij}: $c_{ij} + t_i$, onde:

c_{ij}: custo unitário de transporte da fábrica i para o centro consumidor j.

t_i: custo unitário de produção na fábrica i.

F_i: custo fixo que incorre ao se instalar uma fábrica no local i.

D_j: demanda a ser suprida no centro j.

m: número de locais possíveis para a instalação das fábricas.

n: número de centros consumidores.

Dada a simplicidade da formulação (problema é não capaci

tado e não existem centros intermediários), observa-se que o problema resultante da fixação de um vetor de localização $Y = (y_1 \ldots y_n)$, pode ser resolvido por inspeção.

O algorítmo consta dos seguintes passos:

1) INICIALIZAÇÃO:

Determinar um vetor de localização central Y^0, arbitrando os valores das variáveis y_i. $(Y^0 = (y_i \ldots y_m))$.

2) ITERAÇÃO k:

Seja Y^k o vetor localização central nesta iteração. Determinar os vetores $Y^{k1}, \ldots, Y^{kn}$, sendo que

$$y_j^{ki} = y_j^k \qquad \forall\ j \neq i \qquad i = 1 \ldots n$$

Para cada vetor y^{ki} , determinar os valores

$$b_{ij}^* = \min_i \{b_{ij} \mid y_i = 1\}$$

para $j = 1 \ldots n$, de forma que o vetor solução X é obtido, fazendo

$$x_{ij} = \begin{cases} D_j & \text{para } i = i^* \\ 0 & \text{caso contrário.} \end{cases}$$

Seja $Z(Y^{ki})$ o valor da função objetivo quando o vetor Y^{ki} é fixado.

3) TESTE DE OTIMALIDADE:

Se $Z(Y^{ki}) < Z(Y^k)$ para algum i, faça $Y^{k+1} = Y^{ki}$ e retornar ao passo 2; caso contrário, a solução obtida ao se fixar Y^k será considerada a solução para o problema.

É interessante aplicar este algorítmo a problemas com mui

tos locais possíveis, de forma que o número total de vetores localização Y gerados pelo algorítmo será pequeno em relação ao número total, 2^m, de vetores de localização possíveis.

O algorítmo foi testado, computacionalmente, na resolução de problemas envolvendo 6, 8 ou 10 locais em potencial para a instalação das fábricas, de modo que uma solução exata poderia ser obtida pela enumeração dos vetores Y.

Dada a possibilidade de obter esta solução exata, os valores Z_{OT} e Z_A puderam ser comparados, onde, Z_{OT} é o valor da função objetivo na solução ótima exata e Z_A, o valor desta função na solução obtida pelo algorítmo.

Para cada problema, a porcentagem $\bar{X} = \frac{Z_A - Z_{OT}}{Z_{OT}}$ foi calculada.

Para se determinar o vetor Y^0 inicial, três estratégias foram comparadas:

A: a melhor localização possível para a instalação de uma única fábrica.

B: localiza as fábricas em todos os locais possíveis.

C: escolhe a solução de menor custo, entre as soluções obtidas por A e B.

De acordo com os testes realizados com problemas com 8 locais possíveis, conclui-se que o algorítmo produz soluções satisfatórias, sendo que a maior porcentagem de erro foi de 13%, obtida com a estratégia B. A estratégia C revelou-se como a mais eficiente no tocante à proximidade da solução obtida com a solução ótima, sendo que o custo extra de computação incorrido ao se determinar o

vetor Y^0 inicial através desta estratégia, é compensado pela pequena margem de erro obtida.

E, concluiu-se ainda, (o que já era esperado), que a porcentagem de erro $\bar{X}$, aumenta significativamente com o aumento dos custos fixos. (Nos testes realizados, os custos fixos eram iguais em todos os locais possíveis.)

Dentre os algorítmos heurísticos destaca-se o proposto por Kuhen e Hamburger [18], em 1963.

Na formulação do problema são considerados vários produtos. Os armazéns a serem implantados são capacitados e constituem centros intermediários entre as origens dos produtos e os centros de consumo. As funções custo de transporte são lineares e as funções custo de armazenagem são do tipo linear com custo fixo, ou estritamente côncava com ou sem custo fixo.

O programa heurístico proposto consta de duas partes:

a) Programa principal, que inicia com apenas um armazém e prossegue adicionando novos armazéns, um a um, até que a adição de qualquer outro armazém prejudique o custo total do sistema.

b) Subrotina que é executada com o objetivo de melhorar a solução obtida no programa principal.

A experiência computacional do algorítmo foi realizada com 12 problemas sendo cada um constituído por uma fábrica, 50 centros de consumo e dentre estes, 24 locais em potencial foram considerados para a instalação dos armazéns.

Cada problema considera somente um produto e o mesmo custo fixo para qualquer local em potencial.

O tempo de execução dispendido para obter uma solução para os 12 problemas, no programa principal, totalizou 72 minutos num IBM650 (em média 6 minutos por problema).

Testes posteriormente realizados com algorítmos exatos, revelam a eficiência da técnica proposta por Kuhen e Hamburger dada a proximidade da solução obtida por esta técnica com a solução ótima obtida por aqueles algorítmos.

Em alguns casos a solução ótima foi atingida, porém o algorítmo não está estruturado para reconhecer a solução ótima quando esta é obtida.

No problema formulado por Feldman et al. [10] a função custo de implantação de uma facilidade tem a forma de uma função estritamente côncava, contínua.

A técnica heurística proposta, ao contrário à de Kuhen e Hamburger, instala os armazéns em todos os locais possíveis e elimina os armazéns um a um, até que a eliminação de qualquer armazém ativo torne a solução infactível ou não produza uma redução no custo total do sistema.

1.2.2) ALGORÍTMOS DE SEPARAÇÃO E AVALIAÇÃO:

O algorítmo de separação e avaliação proposto em 1965 por Efroymson e Ray [7] foi o primeiro algorítmo exato colocado para a resolução de um problema de localização.

A formulação do problema é a seguinte:

$$\begin{cases} \text{Min: } Z = \sum_{ij} c_{ij} \cdot x_{ij} + \sum_{i} F_i \cdot y_i \\ \\ \text{s.a: } \begin{cases} \sum_{i \in N_j} x_{ij} = 1 & j = 1,\ldots,n \\ \\ \sum_{j \in P_i} x_{ij} \leq n_i \cdot y_i & i = 1,\ldots,m \\ \\ \sum_{j \in P_i} x_{ij} \geq 0 & i = 1,\ldots,m \\ \\ y_i = 0\ ,1 \end{cases} \end{cases}$$

sendo:

x_{ij}: fração da demanda do centro j suprida pela fábrica i.

y_i = 1 se a fábrica i for instalada.

0 caso contrário.

c_{ij}: $t_{ij} \cdot D_j$, onde:

t_{ij}: custo unitário de transporte da fábrica i para o centro consumidor j.

D_j: demanda total no centro consumidor j.

F_i: custo fixo que incorre ao se intalar uma fábrica no local i.

P_i = conjunto dos índices do centro de consumo atendidos pela fábrica i.

n_i = número de centros de consumo em P_i .

N_j = conjunto dos índices das fábricas que podem suprir o centro de consumo j.

m = número de locais possíveis para as fábricas.

n = número de centros de consumo.

A idéia básica do algorítmo é resolver uma sequência de problemas lineares que são obtidos através da fixação das variáveis y_i.

Inicialmente, resolve-se um problema linear, obtido ao se relaxar as restrições de integralidade, impondo somente que $0 \leq y_i \leq 1$ para $i = 1,\ldots,m$.

Iteração r: seja Z_k o valor da função objetivo obtido no nó k da árvore gerada pelo algorítmo.

Escolher

$$\bar{Z} = \min_k \{Z_k \mid k = 1,\ldots,r-1\}$$

ou seja, $\bar{Z}$ é o menor valor obtido para a função objetivo até a iteração r-1.

Seja ℓ o nó da árvore tal que $Z^{\ell} = \bar{Z}$.

Se todas as variáveis y_i forem inteiras no nó ℓ, a solução deste nó será a solução ótima do problema. Caso contrário, escolher y_s, tal que $0 < y_s < 1$, sendo que ao conjunto P_s pertença o centro consumidor de maior demanda.

Ramificar o nó ℓ em dois nós, sendo que em um deles fixa-se y_s em zero e no outro, em um.

Resolver os dois problemas de transporte resultantes desta ramificação.

Efetuar iteração r+1 .

Os problemas de transporte gerados pelo algorítmo são de

resolução quase que imediata, dado que o problema, inicialmente proposto, é não capacitado.

Efroymson e Ray, provam em seu artigo que a solução ótima para tais problemas será:

$$x_{ij} = \begin{cases} 1 \quad \text{se} \quad c_{ij} + \dfrac{g_i}{n_i} = \min\limits_{k \in K_1 \cup K_2} \left[c_{kj} + \dfrac{g_k}{n_k} \right] \\ 0 \quad \text{caso contrário} \end{cases}$$

$$y_i = \frac{1}{n_i} \cdot \sum_{j \in P_i} x_{ij} \qquad i \in K_2$$

sendo:

K_0 = conjunto dos índices onde $y_i = 0$

K_1 = conjunto dos índices onde $y_i = 1$

K_2 = conjunto dos índices onde y_i não

está especificado e

$$g_k = \begin{cases} F_k & k \in K_2 \\ 0 & k \in K_1 \end{cases}$$

O algorítmo permite também o tratamento de problemas com as seguintes funções custo de instalação das fábricas: linear com custo fixo e linear por partes (com dois segmentos lineares) com ou sem custo fixo.

Dado que a principal dificuldade com o algorítmo é a capacidade de memória do computador necessária para armazenar as informações de cada nó da árvore gerada pelo algorítmo, Efroymson e Ray

incorporam ao algorítmo: algumas simplificações, efetuadas em cada nó gerado, para diminuir o número de ramificações procedentes deste nó; eliminam de futuras considerações aqueles nós, cujas soluções forneçam à função objetivo um valor superior àquele obtido em alguma solução factível de algum nó; se um nó k é obtido, tal que sua solução seja factível e $Z^k - \bar{Z} \leq \varepsilon$, então, esta solução será aceita como solução ótima, onde $\bar{Z}$ é o menor valor obtido para a função objetivo até esta iteração e ε é um valor pré-estabelecido.

Um teste computacional deste algorítmo na resoluçaõ de um problema com m=50 e n=200, consumiu, aproximadamente, 10 minutos de CPU num IBM 7094.

Khumawala [17], em 1972, aperfeiçoa este algorítmo.

Adiciona regras para ramificação de um determinado nó da árvore, ou seja, estabelece regras formais para a escolha da fábrica livre que deverá ser fixada como aberta ou fechada.

Propõe também algumas melhorias ao programa computacional do algorítmo, tais como, eliminação de nós com soluções infactíveis, ou de nós já ramificados, com o objetivo de diminuir a capacidade de memória necessária para a execução deste algorítmo.

O algorítmo aperfeiçoado encontrou soluções ótimas para problemas com m=25 e n=50, num tempo menor que 17 segundos (CDC 6500).

Kaufman, et al. [16], adotam uma formulação semelhante à do problema de Efroymson e Ray, porém, consideram armazéns como centros intermediários entre as fábricas e os centros consumidores.

O problema consiste em localizar simultaneamente fábricas e armazéns.

O algorítmo proposto é do tipo separação e avaliação e é

uma extensão daquele proposto por Efroymson e Ray.

A experiência computacional na resolução de problemas com 10 locais possíveis para as fábricas, de 15 a 40 locais possíveis para os armazéns e 50 áreas de demanda, revelou que o tempo máximo de execução requerido foi de 382 segundos de CPU (6 minutos e 37 segundos) num CDC 6500.

Dentro da categoria dos problemas de localização capacitados, temos o proposto por Marks [22].

Em seu modelo os armazéns a serem instalados constituem pontos intermediários entre as fábricas e os centros de consumo, e possuem capacidade limitada.

A formulação do problema é a seguinte:

$$
\left[\begin{array}{ll}
\text{Min:} & \sum_i F_i \cdot y_i + \sum_i \sum_j \bar{c}_{ij} \cdot \bar{x}_{ij} + \sum_i \sum_k \bar{\bar{c}}_{ki} \cdot \bar{\bar{x}}_{ki} \\
\\
\text{s.a:} & \left[\begin{array}{ll}
\sum_{i=1}^{m} \bar{\bar{x}}_{ki} \leq S_k & k = 1,2,\ldots,p \\
\\
\sum_{j=1}^{n} x_{ij} = \sum_{k=1}^{p} \bar{\bar{x}}_{ki} & i = 1,2,\ldots,m \\
\\
\sum_{k=1}^{p} \bar{\bar{x}}_{ki} \leq Q_i \cdot y_i & i = 1,2,\ldots,m \\
\\
D_j^{\ell} \leq \sum_{i=1}^{m} \bar{x}_{ij} \leq D_j^{u} & j = 1,2,\ldots,n \\
\\
\bar{x}_{ij},\ \bar{\bar{x}}_{ki} \geq 0 \quad \text{e inteiros} & \\
\\
y_i = 0,\ 1 &
\end{array}\right.
\end{array}\right.
$$

onde:

y_i = 1, se um armazém for instalado no local i.

0, caso contrário.

$\bar{x}_{ij}$ = fluxo do armazém i para o centro consumidor j.

$\bar{\bar{x}}_{ki}$ = fluxo da fábrica k para o armazém i.

$\bar{c}_{ij} = c_{ij}+R_j$, custo unitário associado à transferência do produto do armazém i para o centro consumidor j onde:

c_{ij}: custo unitário de transporte.

R_j: custo unitário associado ao uso do centro consumidor j.

$\bar{\bar{c}}_{ki} = c'_{ki} + T_k + V_i$, custo unitário associado à transferência de produto da fábrica k ao armazém i, sendo:

c'_{ki} = custo unitário de transporte.

T_k = custo unitário associado ao uso da fábrica k.

V_i = custo unitário associado ao uso do armazém i.

F_i = custo fixo que incorre ao se estabelecer o armazém i.

S_k = suprimento total na fábrica k.

D_j^{ℓ} = limite inferior sobre a demanda no centro consumidor j.

D_j^{u} = limite superior sobre a demanda no centro consumidor j.

Q_i = capacidade do armazém i.

m = número de locais para instalação dos armazéns.

n = número de centros consumidores.

p = número de fábricas.

Nota-se, pela formulação, que a função custo de instalação é linear com custo fixo.

O algorítmo proposto é do tipo separação e avaliação e resolve uma sequência finita de problemas de fluxo de custo mínimo usando um algorítmo "out of Kilter".

Inicialmente, supõe que todos os armazéns são instalados e determina uma função linear aproximada para cada função custo de instalação do armazém i.

Esta função, na formulação do problema, será:

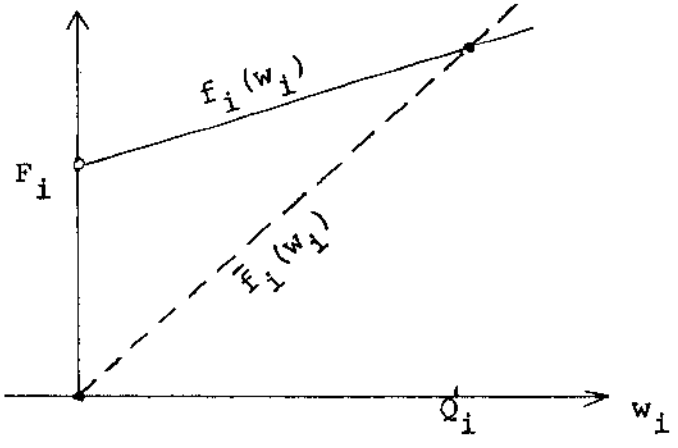
$$f_i(w_i) = V_i \cdot w_i + F_i$$

onde $w_i = \sum_{k=1}^{p} x_{ki}$.

Sua função linear aproximada será:

$$\bar{f}_i(w_i) = \left(V_i + \frac{F_i}{Q_i}\right) \cdot w_i$$

Graficamente, tem-se



A função linear aproximada é uma subestimação da função custo original de instalação do armazém i.

Calculados estes custos, resolve-se um problema de fluxo

de custo mínimo usando o algorítmo "out of Kilter".

Se a solução deste problema for tal que o fluxo através de cada armazém for zero ou Q_i, a solução ótima foi obtida.

Caso contrário, o processo de ramificação deverá ser efetuado. Escolhe-se um armazém cujo fluxo não seja um dos limitantes, em um novo nó este armazém será excluído do conjunto solução, zerando-se sua capacidade, e no outro, este armazém será fixado como aberto e sua função custo original será considerada.

A experiência computacional com o algorítmo revelou-se - eficiente, e, muitos problemas relativamente grandes foram resolvidos em tempos razoáveis de execução.

É interessante observar que as restrições de capacidade, não considerada por outros autores servem de guia neste processo de resolução.

Graciano Sá [23], também propõe um algorítmo do tipo separação e avaliação para a resolução de um problema de localização capacitado.

O problema por ele proposto é o seguinte:

$$\left[\begin{array}{lll} \text{Min:} & W(X,Y) = \sum_i \sum_j c_{ij} \cdot x_{ij} + \sum_{i \in P} F_i \cdot y_i & \\ \text{s.a:} & \left[\begin{array}{ll} \sum_j x_{ij} \leq k_i \cdot y_i & i \in I \\ \sum_i x_{ij} = D_j & j \in J \\ x_{ij} \geq 0 & \forall\ i,\ j \\ y_i = 0,1 & i \in P \\ y_i = 1 & i \in I-P \end{array}\right. & \end{array}\right.$$

onde:

x_{ij}: quantidade transportada da fábrica i para o centro consumidor j.

y_i : $i \in P = 1$ se uma fábrica for instalada no local i.

0 caso contrário.

c_{ij}: custo de transporte da fábrica i para o centro j.

F_i: custo fixo incorrido ao se instalar uma fábrica no local i.

k_i: capacidade de produção da fábrica i.

D_j: demanda no centro j.

I: conjunto dos locais onde as fábricas já estão, ou deverão ser instaladas.

P: subconjunto de I; constituído dos locais possíveis para a instalação das fábricas.

Sá propõe um algorítmo do tipo separação e avaliação, que inicialmente considera todas as fábricas instaladas. Os problemas resultantes de uma certa localização são resolvidos pelo algorítmo "out of Kilter".

Da experiência computacional conclui-se que o algorítmo deve ser aplicado a problemas onde o número de variáveis inteiras esteja em torno de 25.

Para a resolução de problemas em que este número é excedido, Sá propõe um algorítmo aproximado para obter boas soluções.

Nos testes realizados com os dois algorítmos para a reso

lução dos mesmos problemas, verificou-se que as soluções obtidas pelo algorítmo aproximado foram frequentemente ótimas ou bem próximas da solução ótima, num tempo de execução bem menor que o requerido pelo algorítmo exato.

Para citar um exemplo, para um problema com 24 variáveis inteiras, o algorítmo aproximado obteve a solução ótima em 5 minutos e 38 segundos, enquanto que o algorítmo exato usou 15 minutos (IBM 360/65).

Um eficiente algorítmo do tipo separação e avaliação para a resolução de um problema de localização capacitado, foi proposto por Akinc e Khumawala [2].

Este algorítmo é uma extensão, para o problema capacitado, do algorítmo proposto por Efroymson e Ray, no que se refere à fixação de alguns armazéns permanecendo os outros livres, e a extensão desta solução para se obter soluções factíveis melhores para o problema.

Os problemas propostos por Kuhen e Hamburger [18], foram usados para testes computacionais com o algorítmo, e, dado que estes mesmos problemas foram testados por Sá e Elwein, os tempos de execução foram comparados, revelando-se o algorítmo de Akinc e Khumawala bem mais eficiente; sendo que obteve soluções ótimas para problemas que os algorítmos exatos de Sá e de Elwein não o conseguiram, por excesso da capacidade de memória requerida ou de tempo de execução.

O problema de localização proposto por Soland [24] é mais geral, no sentido de que tanto as funções custo de instalação das

fábricas quanto as funções custo de transporte envolvem economias de escala.

A formulação é a seguinte:

$$\left[\begin{array}{l} \text{Min: } Z = \sum_i f_i(y_i) + \sum_{ij} t_{ij}(x_{ij}) \\ \\ \text{s.a: } \left[\begin{array}{ll} \sum_i x_{ij} = r_j & j = 1,\ldots,n \\ \\ \sum_j x_{ij} = y_i \leq a_i & i = 1,\ldots,m \\ \\ x_{ij} \geq 0 & \\ \\ y_i \geq 0 & \end{array}\right. \end{array}\right.$$

onde

x_{ij}: quantidade anual transportada da fábrica i para a cidade j.

$y_i = \sum_j x_{ij}$: produção anual na fábrica i.

$f_i(y_i)$: custo de construção e produção na fábrica i.

a_i: capacidade de produção na fábrica i.

$t_{ij}(x_{ij})$: custo de transporte da fábrica i para a cidade j.

O algorítmo proposto é do tipo separação e avaliação e consiste na resolução de uma sequência de problemas lineares de transporte resultantes das subestimações lineares efetuadas sobre as funções custo.

Este algorítmo é uma versão simplificada do algorítmo pro

posto por Falk e Soland [8], para problemas de programação não - convexa, e estes dois trabalhos serão descritos com mais detalhes no segundo capítulo dado que constituem a base para o desenvolvimento computacional desta pesquisa.

1.2.3) ALGORÍTMOS DE DECOMPOSIÇÃO:

Entre os algorítmos que fazem uso de técnicas de decomposição, podemos citar o de Balinski [4], como o pioneiro desta categoria de algorítmos para a resolução de um problema capacitado.

A formulação do seu problema é a seguinte:

$$\begin{cases} \text{Min:} & Z = \sum_i \sum_j c_{ij} \cdot x_{ij} + \sum_i F_i \cdot y_i \\ \text{s.a:} & \begin{cases} \sum_i x_{ij} = 1 & j = 1,\ldots,n \\ x_{ij} \leq y_i & \forall\ i,j \\ x_{ij} \geq 0 & \forall\ i,j \\ y_i = 0,1 & i = 1,\ldots,m \end{cases} \end{cases}$$

Esta formulação serviu de base para a formulação do problema proposto por Efroymson e Ray, sendo que o significado das variáveis e constantes é o mesmo.

O algorítmo proposto utiliza a decomposição de Benders , que consiste em separar a parte inteira da parte real do problema e resolver os problemas resultantes separadamente, mas dirigidos por um esquema coordenador.

Geoffrion e Graves [13] relatam uma bem sucedida aplica-

ção da decomposição de Benders para resolver problemas de localização de armazéns considerando multiprodutos, onde o esquema de decomposição separa a parte inteira da parte real e esta por sua vez é constituída de vários problemas de transporte independentes, um para cada produto considerado.

Experiências computacionais envolvendo 14 fábricas, 17 produtos diferentes, 45 locais possíveis de instalação e 121 centros consumidores foram realizados, sendo que o tempo de execução foi em média igual a 1 minuto num IBM 360/91.

1.3) PROBLEMAS DE TRANSPORTE COM CUSTO FIXO

Num modelo de transporte com custo fixo, as restrições são semelhantes às de um problema de transporte, porém, os custos de transporte são lineares com custo fixo.

A formulação para tais problemas é a seguinte:

$$\left[\begin{array}{lll} \text{Min:} & \sum_i \sum_j \left[c_{ij} \cdot x_{ij} + F_{ij} \cdot y_{ij}\right] & \\ \text{s.a:} & \left[\begin{array}{ll} \sum_i x_{ij} \geq D_j & j = 1,2,\ldots,n \\ \sum_j x_{ij} \leq S_i & i = 1,2,\ldots,m \\ x_{ij} \leq m_{ij} \cdot y_{ij} & \forall\ i,\ j \\ x_{ij} \geq 0 & \forall\ i,\ j \\ y_{ij} = 0,1 & \forall\ i,\ j \end{array}\right. & \end{array}\right.$$

onde:

x_{ij}: quantidade transportada da fábrica i para o centro consumidor j.

y_{ij}: 1 se rota de $i \rightarrow j$ for aberta.
0 caso contrário.

c_{ij}: custo de transporte da fábrica i para o centro consumidor j.

F_{ij}: custo fixo ao se implantar uma rota de $i \rightarrow j$.

D_j: demanda no centro j.

S_i: suprimento na fábrica i.

m_{ij}: $\min (D_j, S_i)$.

Balinski, em 1961, propôs um algorítmo para obter soluções aproximadas para tais problemas.

Um algorítmo exato foi proposto por Murty [21], em 1968.

Seu procedimento se baseia no fato de que a solução ótima ocorre num ponto extremo do conjunto solução.

Murty supõe que o valor do problema sem os custos fixos, Z_{min}, é finito.

Neste caso, se torna possível colocar os pontos extremos em ordem crescente do valor $\sum_i \sum_j c_{ij}.x_{ij}$.

Seja Z_k o custo variável do k-ésimo ponto extremo, onde:

$$Z_k \geq Z_{k-1} \geq \ldots \geq Z_1$$

Seja F_k o custo fixo associado ao k-ésimo ponto extremo , e F_{min} um limite inferior sobre o custo fixo total, então, $F_k \geq F_{min}$.

Seja $T_k = \min_{i \leq k} \{Z_i + F_i\}$ a melhor solução obtida até a -

iteração k.

Murty mostra que uma condição suficiente para a otimalidade é $Z_r \geq T_r - F_{min}$.

O procedimento consiste em examinar pontos extremos em ordem crescente de Z até que o limite superior Z_{max} é excedido, onde $Z_{max} = T_r - F_{min}$.

Em testes computacionais realizados verificou-se que o algorítmo é eficiente em problemas onde os custos fixos são pequenos quando comparados aos custos variáveis.

Gray [14], em 1968, propõe um algorítmo, para obter a solução exata do problema em questão, que consiste na decomposição do problema num problema mestre inteiro e numa série de subproblemas de transporte.

Com o objetivo de diminuir o número de subproblemas gerados, limites superiores e inferiores sobre o custo fixo total são estabelecidos, de forma a se eliminar de consideração as localizações que ultrapassem estes limites. Também serão eliminadas as localizações que não satisfaçam as condições de factibilidade.

A experiência computacional revelou que o algorítmo é mais eficiente em problemas onde os custos fixos são grandes quando comparados aos custos variáveis.

Gray [14], propõe um algorítmo combinado, que inicia com uma solução factível e aplica o algorítmo de Murty, até exceder o tempo de execução estabelecido para esta fase, ou até obter um certo número de pontos extremos. Neste ponto, o algorítmo de Murty garante que todos os pontos extremos com custo variável menores ou iguais a Z_k, foram examinados (onde k é atual iteração), de forma

que um menor valor para o limite superior sobre o custo fixo total pode ser obtido, e portanto o número de subproblemas de transporte gerados pelo algorítmo de Gray, será menor, diminuindo sensivelmente o tempo de execução necessário para se obter a solução ótima.

Em alguns problemas esta solução poderá ser atingida na primeira fase, durante a aplicação do algorítmo de Murty.

1.4) PROBLEMAS GERAIS DE FLUXO DE CUSTO MÍNIMO COM CUSTO CÔNCAVO:

Dentro da área geral de problemas de fluxo de custo mínimo com custo côncavo, tem-se o trabalho de Zangwill [28], como um dos pioneiros.

Dado que a solução ótima para estes problemas ocorre em pontos extremos do conjunto-solução, Zangwill desenvolve alguns teoremas para caracterizar tais pontos.

Alguns algorítmos são então desenvolvidos para obter as soluções para determinados problemas especiais, como: redes com um único nó origem e um nó destino, redes com um nó origem e vários nós destino e redes com vários nós origem e um nó destino. Em todos os casos apenas um produto é considerado.

Um teorema é ainda estabelecido de forma que redes com multiprodutos e com uma única origem e um único destino poderão ser reduzidos ao caso de um produto.

Outro trabalho dentro desta área, é desenvolvido por Florian e Robillard [11].O problema, por eles tratado envolve um único produto, as restrições são as de conservação de fluxo, este fluxo é limitado sobre cada arco, e a função objetivo tem a seguinte for

ma:

$$F(x) = \begin{cases} \sum_{(i,j) \in A} f_{ij}(x_{ij}) & x_{ij} \in [0,\ c_{ij}] \\ 0 & x_{ij} = 0 \end{cases}$$

onde:

A: conjunto de arcos da rede.

x_{ij}: fluxo ao longo do arco (i,j).

c_{ij}: limite superior sobre o fluxo do arco (i,j).

$f_{ij}(.)$: função custo côncava sobre o arco (i,j).

Estabelecem inicialmente a equivalência entre um problema de fluxo numa rede geral e um problema de fluxo numa rede bipartida, não-capacitada, reduzindo desta forma o problema inicialmente proposto a um de transporte com função custo côncava.

A solução ótima é obtida por enumeração implícita do conjunto de fluxos extremos.

Propõem um procedimento de redução para a construção de fluxos extremos e limites superior e inferior são estabelecidos,no desenvolvimento do algorítmo, para evitar a enumeração de todos os fluxos extremos do conjunto-solução.

Inicialmente, são obtidos custos lineares sobre cada arco através da subestimação linear das respectivas funções custo côncavas, sendo $\tilde{F}(x)$ a função objetivo linear.

Um problema de transporte é resolvido e seja x* a sua solução.

Seja $\tilde{F}(x^*)$ o valor da função objetivo original nesta so

lução, tem-se então que:

$$\bar{F}(x^*) \leq F^* \leq \tilde{F}(x^*)$$

onde F^* é o valor ótimo da função objetivo a ser atingido.

O processo de redução tem início, e consiste na ramificação do nó analisado e na obtenção de um fluxo extremo parcial. O fluxo complementar a este é obtido, resolvendo-se um problema de transporte reduzido, com os custos lineares, de forma que novos limitantes poderão ser estabelecidos.

Se em algum nó da árvore gerada por este procedimento, o limitante inferior obtido, for maior que o melhor valor para a função objetivo conhecido, este nó será eliminado de considerações futuras, e, desta forma, reduz-se o número de ramificações necessárias para se atingir a solução ótima.

Testes computacionais não foram realizados, e portanto nada se pode afirmar a respeito da eficiência deste algorítmo com relação ao tempo de execução e à capacidade de memória requerida.

Um único exemplo numérico foi resolvido à mão; a rede original consta de 4 nós e 5 arcos e as funções custo são do tipo linear com custo fixo. A solução ótima é obtida no 8º nó analisado.

1.5) PROBLEMAS DE PROGRAMAÇÃO CÔNCAVA:

O problema geral de programação côncava, que consiste em obter o mínimo global de uma função côncava sobre um determinado conjunto de soluções factíveis, foi inicialmente estudado por Tuy [26], em 1964. Em sua formulação as restrições são lineares.

O interesse nesta área vem se desenvolvendo devido a uma série de razões, entre elas está o grande número de problemas prá-

ticos que surgem na área de Pesquisa Operacional e Matemática Econômica que, como já foi visto, estão mais próximos do real, quando as funções custo envolvidas refletem economias de escala. Uma outra razão, é o fato de poder se reduzir problemas de programação inteira 0-1 a problemas de programação côncava.

Os primeiros métodos propostos para problemas em que o conjunto solução é um poliedro convexo, usam planos de corte e outros ainda combinam esta aproximação com subestimações lineares das funções côncavas. Contudo, a convergência não foi bem estabelecida para tais métodos.

Um algorítmo eficiente do ponto de vista computacional e também com relação à convergência foi proposto por Falk e Hoffman [9], em 1976.

O problema tratado envolve somente restrições lineares, e o método proposto utiliza técnicas de relaxação do conjunto solução e de cálculo de envelopes convexos sobre os poliedros obtidos, de forma que o algorítmo resolve uma sequência de subproblemas convexos.

O algorítmo proposto por Tuy [26], foi aperfeiçoado posteriormente por Thoai e Tuy [25], que descrevem uma classe de algorítmos do tipo separação e avaliação para a resolução de problemas desta área, com restrições lineares.

O problema tratado por Falk e Soland [8] envolve restrições não lineares, porém a função objetivo deve ser separável em cada variável. O algorítmo proposto é do tipo separação e avaliação e resolve uma sequência de problemas convexos.

CAPÍTULO II

UM ALGORÍTMO PARA O PROBLEMA DE FLUXO EM REDES COM CUSTO CÔNCAVO

2.1) INTRODUÇÃO:

Neste capítulo serão colocados:

- Formulação do problema, objeto de estudo desta pesquisa;
- Uma versão relaxada do algorítmo proposto por Falk e Soland [8] para a resolução de um problema geral de programação côncava;
- O algorítmo de Soland, que é muito semelhante à versão relaxada do algorítmo de Falk e Soland.

2.2) FORMULAÇÃO DO PROBLEMA:

Seja uma rede com n nós pertencentes ao conjunto $N = \{1,\ldots,n\}$ e m arcos pertencentes ao conjunto $A = \{(i,j) \in NxN\}$.

O conjunto N é particionado em 3 conjuntos disjuntos:

R: conjunto dos nós origem;

I: conjunto dos nós intermediários;

D: conjunto dos nós destino.

O problema de fluxo de custo mínimo pode ser assim formulado:

$$\text{PROBLEMA P} \quad \begin{cases} \text{Min: } F(X) = \sum\limits_{(i,j) \in A} f_{ij}(x_{ij}) \\ \text{s.a: } \begin{cases} \sum\limits_{j=1}^{n} (x_{ij} - x_{ji}) = r_i & i \in R \\ \qquad\qquad\quad = 0 & i \in I \\ \qquad\qquad\quad = -d_i & i \in D \end{cases} \\ \quad \ell_{ij} \leq x_{ij} \leq L_{ij} \qquad \forall\ (i,j) \in A \end{cases}$$

onde:

x_{ij}: fluxo sobre o arco (i,j)

$f_{ij}(.)$: função custo sobre o arco (i,j); estas funções são côncavas e supõe-se que $f_{ij}(0) = 0$

ℓ_{ij}: limite inferior sobre o fluxo ao longo do arco (i,j)

L_{ij}: limite superior sobre o fluxo ao longo do arco (i,j).

2.3) ALGORÍTMO PARA RESOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO CÔNCAVA-SEPARÁVEL:

2.3.1) INTRODUÇÃO:

Este algorítmo foi proposto em 1969, por Falk e Soland [8], para a resolução do problema:

$$\text{PROBLEMA PS} \quad \begin{cases} \text{Min: } \varphi(x) = \sum_{i=1}^{n} \varphi_i(x_i) \\ \\ \text{s.a: } x \in G \cap C \end{cases}$$

onde:

G = conjunto das restrições globais

C = $\{x \;/\; \ell \leq x \leq L\}$ conjunto das restrições individuais.

Supõe-se que:

G é um conjunto fechado;

$G \cap C$ é um conjunto não vazio e

$\varphi_i(x_i)$ é semicontínua inferior sobre $C_i = [\ell_i, L_i]$ e possivelmente não convexa.

A separabilidade é requerida apenas para função objetivo.

2.3.2) O ALGORÍTMO:

O algorítmo geral, proposto para a resolução do problema PS, é do tipo separação e avaliação ("BRANCH AND BOUND").

No processo de separação, o conjunto C é particionado em retângulos cada vez menores e os envelopes convexos das funções $\varphi_i(x_i)$ são tomados sobre os respectivos subretângulos.

No processo de avaliação, obtém-se limitantes inferiores e superiores para o valor ótimo da função objetivo $\varphi(x)$.

Em cada iteração k do algorítmo, efetua-se o processo de separação, de forma que uma função convexa $\psi^k(x)$ é obtida.

Para se obter os limitantes, resolve-se o subproblema PS^k que consiste na minimização de $\psi^k(x)$ sobre $G \cap C$ e, seja x^k a solução ótima de PS^k, que é factível para o problema PS.

Desta forma, o algorítmo consiste basicamente na resolução de uma série de subproblemas convexos cujas soluções, formam uma sequência $\{x^k\}$ de pontos factíveis.

Se esta sequência for finita, seu último elemento será a solução ótima para PS. Se a sequência for infinita, existirá um ponto de acumulação $\bar{x}$, desde que $G \cap C$ é compacto e, sob certas hipóteses este ponto será a solução ótima do problema PS.

É importante ressaltar que, em geral, este algorítmo não converge num número finito de passos e sua convergência é provada sob diferentes condições.

Falk e Soland propõem ainda um algorítmo relaxado, que se origina deste algorítmo geral, e, que será aqui detalhado porque o método de Soland [24], utilizado na resolução do problema P é semelhante a esta versão.

2.3.3) VERSÃO RELAXADA DO ALGORÍTMO GERAL:

Inicialização:

Determina-se o envelope convexo $\psi_i(x_i)$ de cada função $\varphi_i(x_i)$ sobre todo o intervalo $C_i = [\ell_i, L_i]$.

O problema PS^1 será :

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

Seja x^1 a solução de PS^1 .

Desde que $\psi^1(x)$ é o envelope convexo de $\varphi(x)$ sobre todo conjunto C, temos que:

$$\psi^1(x) < \varphi(x) \quad \text{para } \forall\, x \in G \cap C$$

Então, se $\psi^1(x^1) = \varphi(x^1)$, x^1 será a solução ótima de - PS.

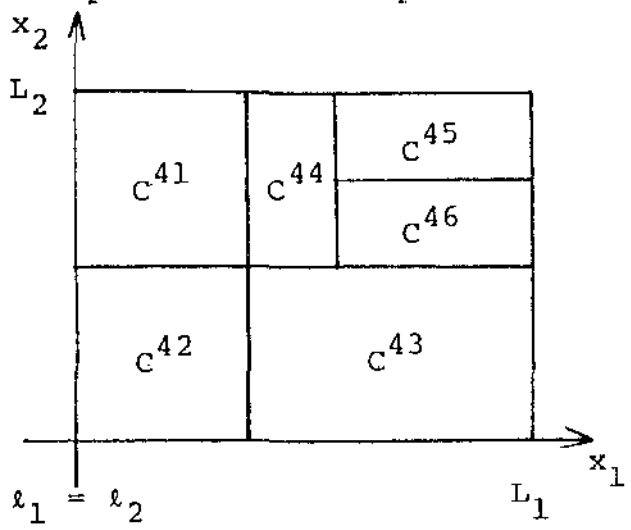
Caso contrário, uma partição do conjunto C será determinada (processo de separação ou ramificação), os novos envelopes convexos serão determinados sobre estas partições e uma nova iteração será efetuada.

Iteração k do algorítmo:

Supor que na iteração k, o conjunto C esteja particionado em p_k subconjuntos retangulares: $C^{k_1}, \ldots, C^{k_{P_k}}$, onde $C^{k_j} = \{x \,/\, \ell^{k_j} \leq x \leq L^{k_j}\}$, $j = 1, \ldots, p_k$.

Tem-se, desta forma $C^k = \bigcup_{j=1}^{p_k} C^{k_j}$.

Exemplo em $\mathbb{R}^2$. Supor $k = 4$ e $p_4 = 6$.



$$C^4 = \bigcup_{j=1}^{6} C^{4j}$$

Um ponto $x^{k_j} \in C^{k_j}$ $(j = 1,\ldots,p_k)$ pode ser determinado, resolvendo-se os problemas PS^{k_j}.

PS^{k_j}

Min: $\psi^{k_j}(x) = \sum_{i=1}^{n} \psi_i^{k_j}(x_i)$

s.a: $x \in G \cap C^{k_j}$

onde $\psi^{k_j}(x)$ é o envelope convexo de $\varphi(x)$ tomado sobre C^{k_j}.

Seja $\mu^{k_j} = \psi^{k_j}(x^{k_j})$.

Seja $\mu^{k_\ell} = \min \{\mu^{k_j} / j = 1,\ldots,p_k\}$, então, x^{k_ℓ} é a solução ótima, x^k, desta iteração.

Se tivermos $\varphi(x^{k_\ell}) = \mu^{k_\ell}$, o ponto x^{k_ℓ} será a solução do problema PS, pois x^{k_ℓ} é factível para o Problema P e

$$\mu^{k_\ell} \leq \psi^{k_j}(x) \leq \varphi(x) \quad \text{para } \forall\, x \in G \cap C^{k_j}, \quad j = 1,\ldots,p_{k_j}$$

Se $\mu^{k_\ell} < \varphi(x^{k_\ell})$ o conjunto

$$C^{k_\ell} = \{x \;/\; \ell^{k_\ell} \leq x \leq L^{k_\ell}\}$$

deverá ser particionado, e para este processo, uma das duas regras abaixo deverá ser escolhida:

a) Regra fraca de Partição:

Escolher i que maximize a diferença

$$\varphi_i(x_i^{k_\ell}) - \psi_i^{k_\ell}(x_i^{k_\ell}) \qquad i = 1,\ldots,n$$

e dividir o intervalo correspondente em $[\ell_i^{k_\ell}, L_i^{k_\ell}]$ em $[\ell_i^{k_\ell}, x_i^{k_\ell}]$ e $[x_i^{k_\ell}, L_i^{k_\ell}]$.

b) Regra forte de Partição:

Para cada i, $i = 1,\ldots,n$ que verificar a relação:

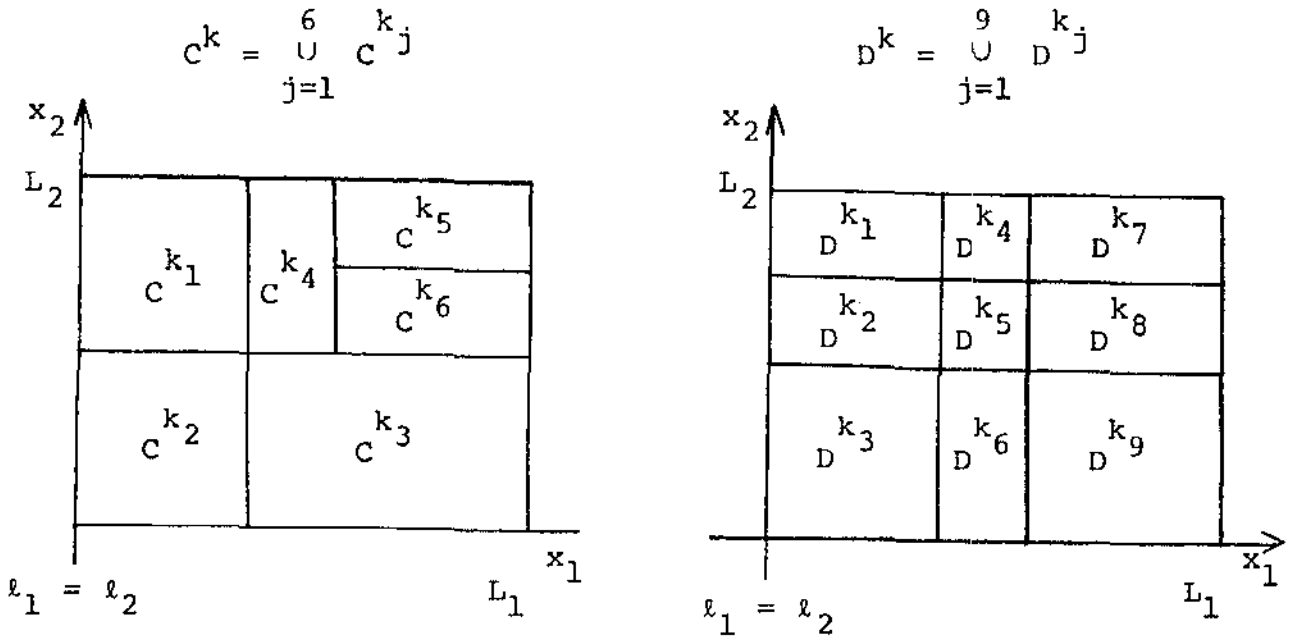
$$\varphi_i(x_i^{k_\ell}) - \psi_i^{k_\ell}(x_i^{k_\ell}) > 0 \;,$$

dividir o intervalo correspondente $[\ell_i^{k_\ell}, L_i^{k_\ell}]$ em $[\ell_i^{k_\ell}, x_i^{k_\ell}]$, $[x_i^{k_\ell}, L_i^{k_\ell}]$.

Efetuada a partição de C^{k_ℓ}, tem início uma nova iteração do algorítmo.

No algorítmo geral, requer-se que o ponto x^k, solução ótima da iteração k, pertença a um subretângulo da partição estendida D^k, de C^k .

Obtém-se D^k, estendendo-se, sobre todo o conjunto C^k, todos os hiperplanos que determinam os retângulos C^{k_j}, como mostra a figura abaixo:



A versão do algorítmo geral, apresentada, é denominada algorítmo relaxado porque o requerimento acima é dispensado, o que o torna mais simples de se executar, porêm, somente teoremas fracos de convergência são estabelecidos.

Estes teoremas, demonstrados por Falk e Soland, são:

TEOREMA 1:

Se: φ é semicontínua inferior, G é fechado e se a regra forte de partição for usada para gerar a sequência $\{x^k\}$, então existe um ponto limite x^+ de $\{x^k\}$ que é solução ótima do problema PS.

E, o limite da sequência $\{\psi^k(y^k)\}$ é $\varphi(x^+)$ onde $\{y^k\}$ é uma subsequência convergente de $\{x^k\}$.

TEOREMA 2:

Se: φ é contínua, G é fechado e se a regra fraca de partição for usada para gerar a sequência $\{x^k\}$, então, existe um

ponto limite x^+ de $\{x^k\}$ que é a solução do problema P.

E, o limite da sequência $\{\psi^k(y^k)\}$ é $\varphi(x^+)$ onde $\{y^k\}$ é uma sequência convergente de $\{x^k\}$.

TEOREMA 3:

Se: φ é côncava, G um poliedro linear e se a regra fraca de partição for usada para gerar a sequência $\{x^k\}$ então a sequência $\{x^k\}$ é finita e termina num ponto solução para o problema PS .

Os teoremas 1 e 2 são "fracos", desde que não indicam - qual subsequência convergente de $\{x^k\}$ fornece o ótimo global.Contudo, uma subsequência convergente $\{y^k\}$ de $\{x^k\}$ fornece um ótimo global via os números $\{\psi^k(y^k)\}$ desde que a sequência $\{\psi^k(x^k)\}$ é monotônica não decrescente.

Quanto ao teorema 3, é idêntico ao estabelecido, para estes problemas, pelo algorítmo geral, porém, parece possível que esta versão possa gerar pontos x^k que não sejam pontos extremos de $G \cap C$.

2.4) ALGORÍTMO DE SOLAND:

O objetivo inicial do trabalho de Soland [24], é a resolução de um problema de localização, no entanto, o algorítmo proposto resolve um problema mais geral: minimizar uma função côncava separável sobre um poliedro linear, assim formulado:

$$\text{PROBLEMA PS} \quad \begin{cases} \text{Min:} & F(x) = \sum_{i=1}^{n} f_i(x_i) \\ \text{s.a:} & x \in G \cap C \end{cases}$$

onde:

$G = \{x / Ax = b\}$

$C = \{x / \ell \leq x \leq L\}$

$f_i(x_i)$: função côncava sobre o intervalo $[\ell_i, L_i]$.

Este algorítmo é semelhante à versão relaxada do algorítmo proposto por Falk e Soland [8], porque resolve uma sequência finita de problemas convexos, porém, o conjunto-solução $G \cap C$ é o mesmo para todos estes problemas, o que torna este método importante para resolução de problemas em que o conjunto das restrições tenha uma estrutura especial, como os problemas de localização, de fluxos, etc.

2.4.1) FASES DO ALGORÍTMO:

Dado que o algorítmo é do tipo separação e avaliação, cada nó gerado pelo problema envolve as fases:

- uma partição do conjunto C
- cálculo dos envelopes convexos das funções $f_i(x_i)$ sobre os novos subretângulos obtidos no processo de partição
- resolução de um problema convexo
- cálculo dos limites inferior e superior
- teste de otimalidade.

Abaixo, cada uma destas fases será descrita com detalhes. A notação utilizada, é a seguinte:

N^0, N^1, N^2,... : nós da árvore gerada pelo algorítmo;

$LF(N^k)$: limite inferior do valor ótimo de $F(x)$ sobre $G \cap C^k$, no nó N^k ;

$LS(N^k)$: limite superior do valor ótimo da função objetivo, sobre $G \cap C$, no nó N^k;

$F(N^k)$: valor da função objetivo no nó N^k .

2.4.1.1) CÁLCULO DOS ENVELOPES CONVEXOS:

Supor que no nó N^k o conjunto C^k considerado, seja: $C^k = \{x \ / \ \ell^k \leq x \leq L^k\}$ onde $C^k \subset C$.

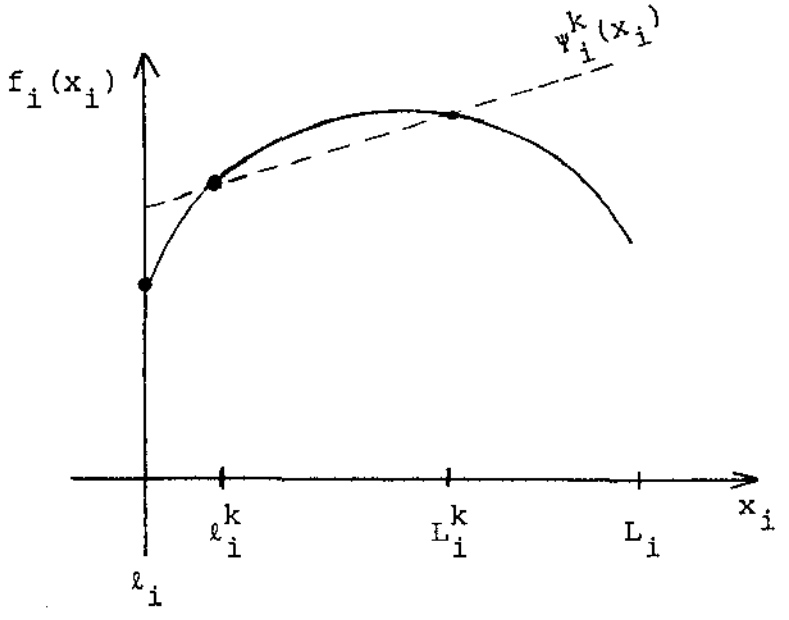
O envelope convexo $\psi_i^k(x_i)$ da função $f_i(x_i)$, é a função linear definida por:

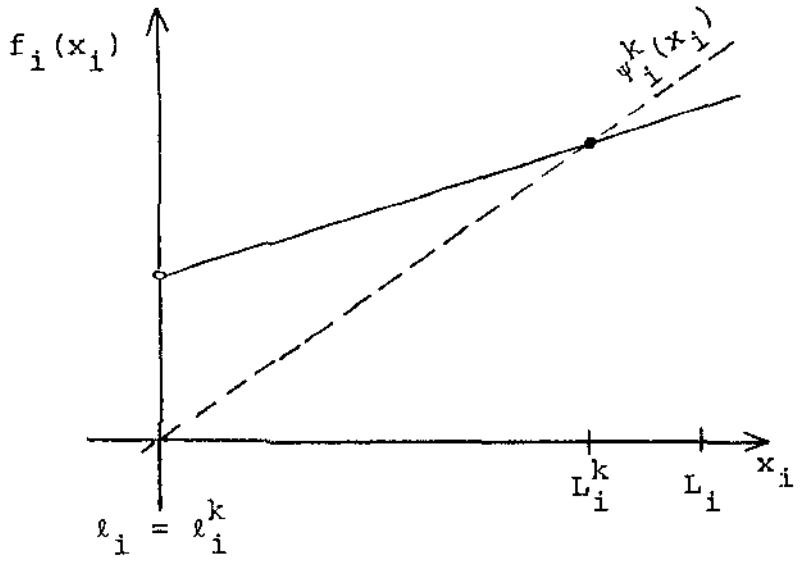
$$\psi_i^k(\ell_i^k) = f_i(\ell_i^k)$$

e

$$\psi_i^k(\ell_i^k) = f_i(L_i^k)$$

como mostram as figuras abaixo:





Desta forma, tem-se que:

$$\psi_i^k(x_i) \leq f_i(x_i) \qquad x_i \in [\ell_i^k, L_i^k]$$

e

$$\psi_i^k(x_i) > f_i(x_i) \qquad x_i \notin (\ell_i^k, L_i^k)$$

2.4.1.2) CÁLCULO DOS LIMITES INFERIORES:

Seja $\psi^k(x) = \sum_{i=1}^{n} \psi_i^k(x_i) \qquad x \in C^k$.

Então :

Min: $\psi^k(x)$ $x \in G \cap C^k$	$\leq$	Min: $F(x)$ $x \in G \cap C^k$

mas ,

Min: $\psi^k(x)$ $x \in G \cap C$	$\leq$	Min: $\psi^k(x)$ $x \in G \cap C^k$

portanto:

Min: $\psi^k(x)$ $x \in G \cap C$	$\leq$	Min: $F(x)$ $x \in G \cap C^k$

Portanto, para se obter um limite inferior $LF(N^k)$, basta resolver o problema PS^k :

$$\begin{cases} \text{Min:} & \psi^k(x) \\ & x \in G \cap C \end{cases}$$

Sua solução ótima x^k é factível para o problema PS e o valor $\psi^k(x^k)$ é um limite inferior sobre o valor ótimo de $F(x)$ sobre $G \cap C^k$.

2.4.1.3) CÁLCULO DOS LIMITES SUPERIORES:

Desde que em todos os problemas PS^k, o conjunto de restrições considerado é $G \cap C$, a solução ótima, x^k, de cada um destes problemas é factível para o problema PS, e desta forma, o algorítmo gera uma sequência $\{x^k\}$ de pontos factíveis.

Seja $LS(N^k) = F(x^h)$, onde:

$$F(x^h) = \min_j F(x^j) \qquad j = 1,\ldots,k$$

Então, $LS(N^k)$, será o limite superior sobre o valor ótimo da função objetivo sobre $G \cap C$ na iteração k, e x^h é a melhor solução conhecida até esta iteração.

2.4.1.4) TESTE DE OTIMALIDADE:

Entre todos os nós intermediários (nós ainda não ramificados) escolher o que possuir o menor limite inferior.

Seja N^t este nó.

Se $LF(N^t) > LS(N^k)$, então, $LS(N^k)$ será o valor ótimo para a função objetivo, e x^h a solução ótima para PS .

Se $LF(N^t) = LS(N^k) = F(N^t)$, a solução obtida em N^t é a solução ótima para o problema e $F(N^t)$ é o valor mínimo para a função objetivo.

Se não ocorrer nenhuma destas situações, o processo de ramificação deve se repetir.

2.4.1.5) PROCESSO DE RAMIFICAÇÃO:

Seja N^r o nó escolhido para ser ramificado, na iteração k .

Cabe aqui, colocar que o nó N^r , não precisa ser necessariamente o nó intermediário com menor limite inferior, outros processos de escolha poderão ser adotados.

O nó N^r será ramificado em dois novos nós: N^{k+1} e N^{k+2}, através da regra fraca de partição colocada por Falk e Soland [8]:

Seja x^r a solução obtida em N^r .

Escolher i que maximize a diferença: $f_i(x_1^r) - \psi_i^r(x_i^r)$, $i = 1,\ldots,m$ e dividir o intervalo correspondente em: $[\ell_i^r, x_i^r]$ e $[x_i^r, L_i^r]$.

Os envelopes convexos da função $f_i(.)$ são tomados sobre os novos intervalos, permanecendo os outros custos lineares os mesmos que aqueles do nó N^r .

Neste processo de ramificação pode não parecer óbvio que a solução ótima x^r do nó N^r pertença ao intervalo $[\ell^r, L^r]$, pois o conjunto solução considerado na resolução de PS^r foi $G \cap C$ e não $G \cap C^r$.

Na verdade, nada garante que $x_i^r \in [\ell_i^r, L_i^r]$, para qualquer i, contudo, desde que N^r foi escolhido para ser ramificado,

tem-se que :

$$LF(N^r) \leq LS(N^k)$$

e

$$LF(N^r) = \psi^r(x^r) < F(x^r)$$

e, então, para algum i, tem-se que :

$$\psi_i^r(x_i^r) < f_i(x_i^r) \quad .$$

Portanto, embora x^r possa não pertencer ao retângulo C^r, uma de suas componentes x_i^r deverá estar no intervalo (ℓ_i^r, L_i^r) .

2.4.2) CONVERGÊNCIA DO ALGORÍTMO:

A convergência deste algorítmo é estabelecida de uma maneira mais simples que a do algorítmo relaxado de Falk e Soland.

Devido à validade dos limitantes inferior e superior e - desde que em todos os subproblemas PS^k, o conjunto-solução considerado é $G \cap C$, de forma que todos os pontos extremos são considerados, tem-se que, se o algorítmo terminar no estágio s, a melhor solução obtida até este estágio será a solução ótima para o problema.

Deve-se provar, portanto, que o número de iterações realizadas, para a resolução de um problema, é finito.

TEOREMA: O algorítmo obtém uma solução ótima após um número finito de iterações.

DEMONSTRAÇÃO: Inicialmente, é importante ressaltar que: o conjunto-solução $G \cap C$ é um poliedro linear e possui um número finito de pontos extremos; desde que os problemas gerados pelo algorítmo

são lineares, sobre $G \cap C$, a resolução de cada um desses subproblemas fornece um ponto extremo deste conjunto-solução.

Deve-se provar que o número de subproblemas possíveis é finito, o que resulta na demonstração de que somente um número finito de possíveis conjuntos C^k podem ser obtidos no desenvolvimento do algorítmo.

Seja E o conjunto dos pontos extremos de $G \cap C$;

seja E_i, $i = 1,\ldots,n$, o conjunto das projeções de E sobre o i-ésimo eixo coordenado;

seja $\bar{E}_i = E_i \cup \{\ell_i\} \cup \{L_i\}$ e,

seja $\bar{E} = \bar{E}_1 \times \bar{E}_2 \times \ldots \times \bar{E}_n$.

Assim definido, $\bar{E}$ é um conjunto finito de pontos do conjunto C .

O algorítmo é tal que cada retângulo C^k é definido por um conjunto de 2^n pontos de $\bar{E}$.

Como $\bar{E}$ é finito, somente um número finito de conjuntos C^k poderão ser obtidos no desenvolvimento do algorítmo.

2.4.3) TESTES COMPUTACIONAIS:

Estes testes computacionais foram realizados através da execução de um programa escrito em FORTRAN IV num CDC 6600.

Este programa difere do algorítmo nos seguintes pontos:

a) O nó intermediário a ser ramificado é escolhido entre os dois últimos nós obtidos pelo algorítmo;

b) O intervalo $[\ell_i^r, L_i^r]$ pode ser dividido em um ponto diferente de x_i^r .

O programa foi aplicado na resolução de alguns problemas de localização resolvidos por Graciano Sá [23], e anteriormente - propostos e resolvidos por Kuhen e Hamburger [18] .

A tabela abaixo mostra o número de iterações efetuadas e os tempos de execução dispendidos na resolução dos problemas acima citados.

Nestes problemas, as funções custo de localização são lineares com custo fixo (exceto uma, que é linear) e os custos de - transporte são lineares.

Na tabela, P indica o número de referência do problema no artigo de Sá [23]; m,o número de locais possíveis; n, o número de centros consumidores; r, o número de funções custo côncavas; ρ, o parâmetro de tolerância utilizado por Soland; NS, o número de nós gerados pelo algorítmo de Soland; TS, o tempo de execução (em segundos) utilizado pelo seu algorítmo e TG o tempo de execução (em segundos) utilizado pelo algorítmo de Sá.

P	m	n	r	ρ	NS	TS	TG
4.1	16	50	15	0.01	241	38.7	90.6
4.2	16	50	15	0.01	19	5.8	108
4.3	16	50	15	0.01	13	4.2	96
5.2	16	50	15	0.01	2279	271.3	457.2
5.2	16	50	15	0.01	753	124.2	457.2
1.1	25	50	24	0.05	871	223.6	900
6	25	50	24	0.05	645	139.6	900

Cabe aqui ressaltar, que o algorítmo de Soland quando aplicado a problemas em que os limitantes não são obtigatórios, se torna mais eficiente, pois neste caso, os problemas gerados poderão ser resolvidos por inspeção, o mesmo não ocorrendo com o algorítmo de Sá, especialmente colocado para problemas capacitados.

Este fato, pode ser verificado na resolução do problema 5.2. Na primeira execução, os limitantes foram considerados por Soland e, verificando que estes limitantes não eram obrigatórios , Soland resolveu o mesmo problema, sem considerá-los nos subproblemas gerados. A solução ótima foi a mesma, porém, foi obtida num tempo de execução bem menor.

Os problemas 4.1 e 1.1 também são resolvidos como não-capacitados.

CAPÍTULO III

RESULTADOS COMPUTACIONAIS: UMA APLICAÇÃO AO PROBLEMA DE ESCOAMENTO DA SAFRA DE MILHO NA REGIÃO CENTRO-BRASILEIRA

O algorítmo proposto por Soland será aplicado à resolução do problema P, proposto no Capítulo II.

Um programa para este algorítmo foi escrito em FORTRAN IV e os testes computacionais foram realizados no sistema DEC-10 instalado na UNICAMP, Campinas-SP.

Os subproblemas gerados em cada iteração do algorítmo são resolvidos por uma subrotina que resolve um problema de fluxo de custo mínimo (linear) pelo algorítmo "OUT OF KILTER".

Ao programa principal cabe a execução das outras fases do método:

a) Linearização das funções côncavas;

b) Cálculo dos limitantes inferior e superior para o valor ótimo da função objetivo;

c) Teste de otimalidade;

d) Escolha de um nó intermediário para ser ramificado.

Para maior clareza e entendimento do que será exposto, serão colocados abaixo a nomenclatura utilizada neste capítulo e o teste de otimalidade programado.

Nomenclatura:

SLIM: Limite superior para o valor ótimo da função objetivo, até a última iteração efetuada; é o melhor valor obtido para a função objetivo até esta iteração (SLIM tem o mesmo signifi

cado que $LS(N^k)$ colocado no Capítulo II);

FLIM: Limite inferior para o valor ótimo da função objetivo; é o menor valor para a função linearizada, $\psi^k(x)$, entre todos os nós gerados mas não ramificados;

EPS: Precisão requerida.

Teste de otimalidade:

A resolução de um problema termina quando, após a ramificação de um nó, se verificar que:

$$\frac{SLIM - FLIM}{SLIM} \leq EPS$$

Se a relação acima for satisfeita a solução que forneceu o valor SLIM para a função objetivo será a solução ótima para o problema.

A principal dificuldade encontrada na implementação computacional foi relativa aos requerimentos de memória decorrentes da necessidade de se armazenar os seguintes dados dos nós intermediários: custos obtidos da linearização das funções côncavas e limitantes para a linearização.

No programa, quatro matrizes são utilizadas para o armazenamento destes dados.

É evidente que grandes economias de memória podem ser feitas no desenvolvimento do algorítmo dado que não há necessidade de se armazenar dados de nós que não serão ramificados. Nesta categoria estão os seguintes nós:

a) Nós já ramificados;

b) Nós não ramificados que possuem:

b.1) valor da função linearizada maior que SLIM : $LF(N^i) > SLIM$;

b.2) valor da função linearizada igual ao valor da função objetivo: $LF(N^i) = F(N^i)$.

Serão denominados nós intermediários, os nós gerados pelo algorítmo que não satisfaçam as condições acima colocadas. Portanto, os nós intermediários são os nós candidatos a ser ramificados.

Serão colocadas abaixo as economias de memória efetuadas no programa.

Seja uma iteração k do algorítmo na qual o nó N^r foi escolhido para ser ramificado. Dois novos nós serão gerados: N^{k+1} e N^{k+2}.

Seja N^t o nó ramificado na iteração anterior. Supor que os dados deste nó, relativos à linearização das funções côncavas, estejam armazenados nas colunas j das matrizes de dados.

Os dados do nó N^{k+1} serão armazenados nestas colunas j das matrizes de dados, pois os dados do nó N^t não serão mais necessários no desenvolvimento do algorítmo.

Quanto ao nó N^{k+2}, antes de se alocar seus dados numa coluna ainda não preenchida, verifica-se se existe algum nó N^s que satisfaça a condição b1 ou b2.

Se existir, alocam-se os dados do nó N^{k+1} nas colunas das matrizes preenchidas com os dados de N^s, caso contrário, uma nova coluna de cada matriz será preenchida.

Exemplo: supor que o nó 1, (N^1), foi ramificado, geran

do os nós 2 e 3 (N^2 e N^3), sendo que os dados destes nós ocupam as colunas 1, 2 e 3 das matrizes de dados. Supor que N^3 foi escolhido para ser ramificado, de forma que serão obtidos os nós N^4 e N^5 .

Os dados de N^4 serão armazenados nas colunas número 1 das matrizes de dados. Este nó passa a ter o número 4 para efeitos de contagem de quantos nós foram gerados pelo algorítmo, mas no desenvolvimento do algorítmo será considerado como o nó número 1 (N^1) .

Com relação ao nó 5, existem várias possibilidades:

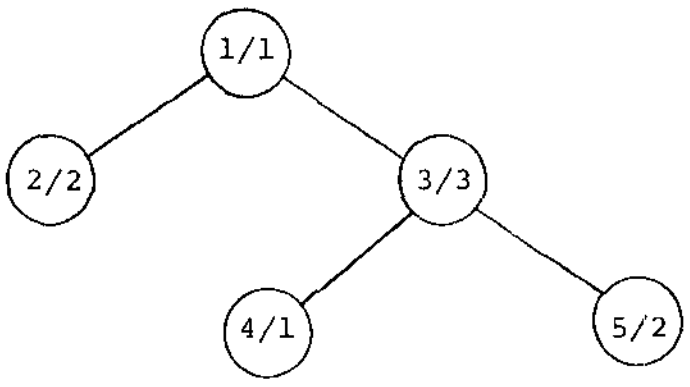
I) se $LF(N^2) > SLIM$ ou se $LF(N^2) = F(N^2)$ seus dados serão armazenados nas colunas número 2;

II) se $LF(N^1) > SLIM$ ou se $LF(N^1) = F(N^1)$ seus dados serão alocados nas colunas número 1 (aqui, N^1 refere-se ao nó número 4 obtido pelo algorítmo);

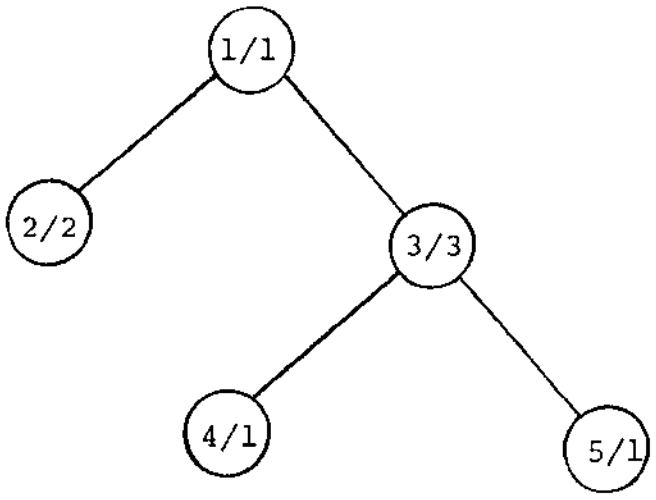
III) se as possibilidades (a) ou (b) não se verificam, os dados do nó 5 serão alocados nas colunas número 4 das matrizes de dados.

A árvore gerada em cada caso, seria:

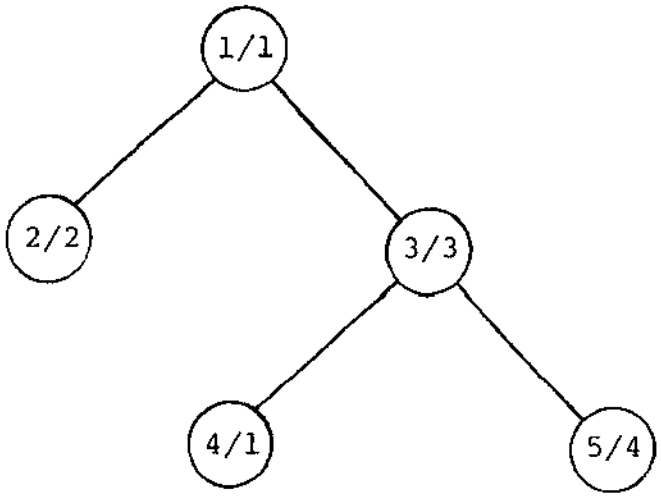
I)



II)



III)



Os números i/j representam:

i: número do nó para efeitos de contagem.

j: número do nó no desenvolvimento do algorítmo.

Outra dificuldade encontrada foi com relação ao armazenamento ou não dos fluxos ótimos obtidos em cada nó.

Com relação ao tempo de execução este armazenamento é importante, pois, quando um nó é ramificado, N^r, apenas um arco terá seu custo alterado, consequentemente, se o fluxo ótimo deste nó

fosse utilizado como solução inicial para os problemas de fluxo dos nós N^{k+1} e N^{k+2}, o algorítmo "out of Kilter" obteria o fluxo ótimo destes problemas com um mínimo esforço computacional.

Contudo, em problemas de grande porte, este armazenamento torna-se inviável dada a quantidade de memória requerida.

A forma como é obtida uma solução inicial para os problemas de fluxo de cada nó: N^{k+1} e N^{k+2}, depende de como é feita a escolha do nó que será ramificado.

Dois processos foram programados para a escolha do nó intermediário a ser ramificado:

PROCESSO Nº 1 - PR1 - ramifica-se, entre todos os nós intermediários, o nó que forneceu o valor FLIM para a função linearizada.

PROCESSO Nº 2 - PR2 - ramifica-se, entre os dois últimos nós gerados, aquele que possuir o menor valor para a função linearizada. Caso estes nós não pertençam à categoria dos nós intermediários, aplica-se o processo número 1 de ramificação.

No processo número 1, não existe a possibilidade de se estimar com antecedência qual nó será ramificado, e dada a inviabilidade de se armazenar o fluxo ótimo obtido em cada nó intermediário, a melhor opção encontrada foi a de se utilizar o fluxo ótimo obtido no último nó gerado pelo algorítmo como solução inicial para o novo problema.

Este fluxo pode não estar próximo do fluxo ótimo deste problema, mas pelo menos é factível, visto que o conjunto-solução é o mesmo em todos os problemas PS^k.

Já no processo número 2, o nó a ser ramificado será um dos dois últimos nós obtidos pelo algorítmo (desde que pelo menos um deles seja um nó intermediário) e neste caso, o armazenamento dos fluxos ótimos obtidos nestes nós é recomendável, pois não requer muita memória e resulta em tempos de execução por iteração - bem menores que em PR1.

À primeira vista, pode parecer que PR2 requer mais memória que PR1.

Contudo, os processos de economia de memória já descritos, são mais eficientes em PR2, que em PR1, pois, desde que no processo 2 o desenvolvimento da árvore "branch and bound" é realizado por ramos, uma boa solução para o problema original é obtida mais rapidamente que em PR1, possibilitando a eliminação de muitos nós não ramificados de considerações futuras.

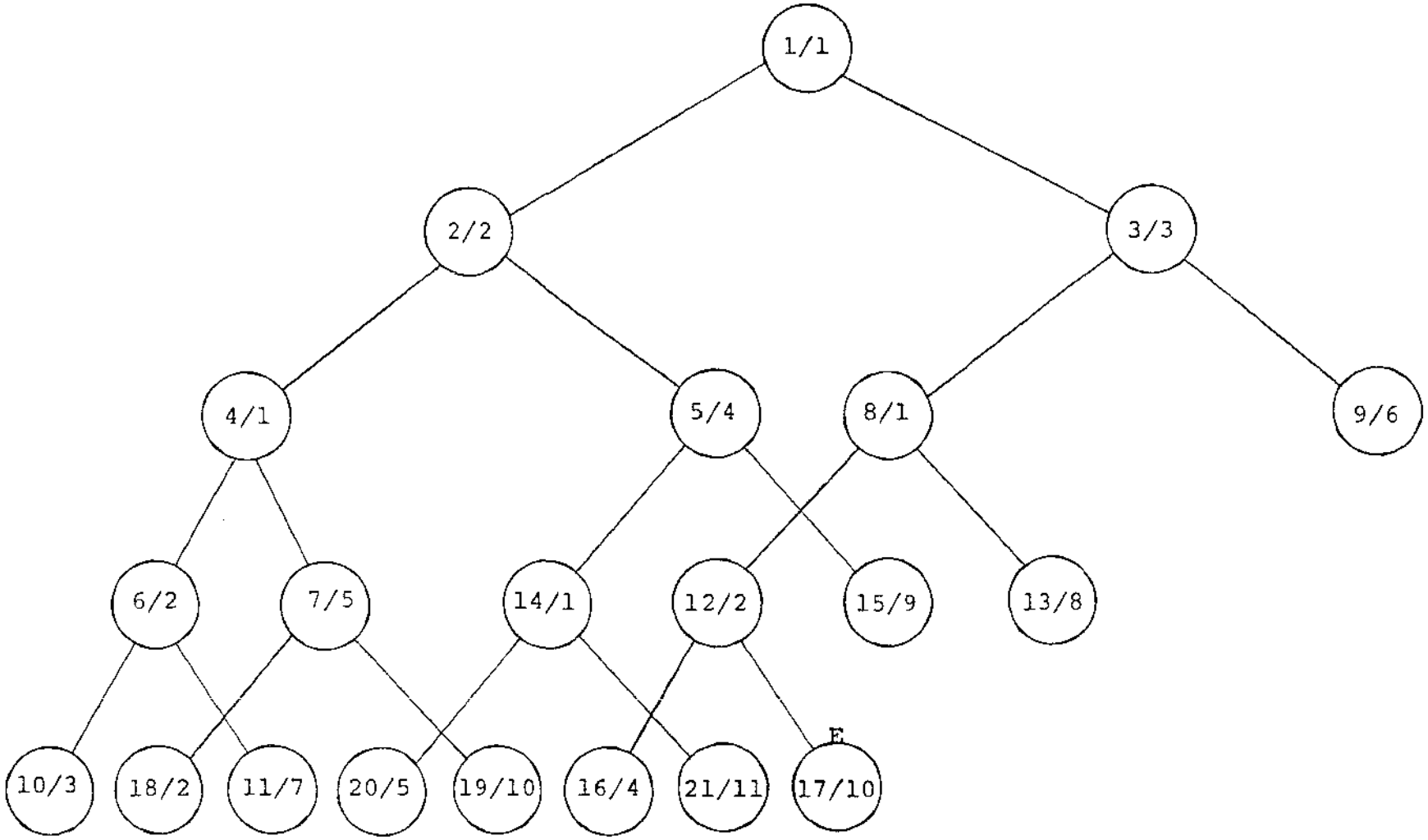
A justificativa é que a função linearizada é bem mais refinada ao longo de um ramo bem desenvolvido (PR2), que ao longo de vários ramos curtos (PR1), da árvore "branch and bound".

Para se ter uma idéia mais precisa das árvores geradas, serão colocadas as árvores geradas na resolução do problema proposto por Baumol e Wolfe [5].

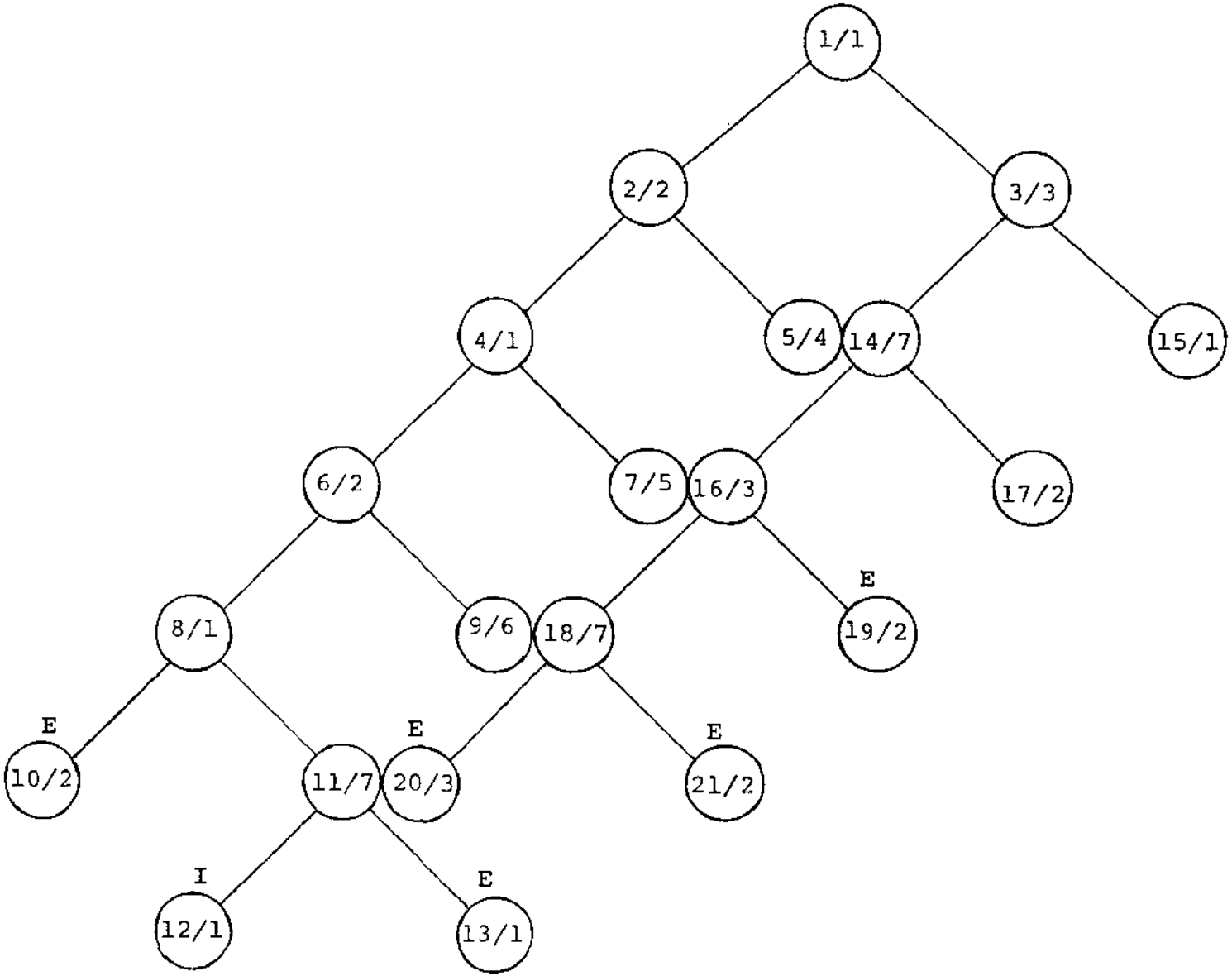
Nesta resolução, PR1 executou 33 iterações e PR2, 35 iterações.

Porém, na 12a. iteração PR2 obteve uma boa solução para o problema, eliminando muitos nós não ramificados de considerações futuras, o que proporcionou um aproveitamento melhor das matrizes de dados dos nós.

O processo 2 preencheu 7 colunas das matrizes enquanto



ÁRVORE GERADA PELO PROCESSO 1



ÁRVORE GERADA PELO PROCESSO 2

que o processo 1 preencheu 12 colunas.

As árvores foram esquematizadas até a iteração 21, dado que a colocação de muitos nós tornaria o esquema confuso.

É importante notar as diferenças entre os melhores valores para a função objetivo (SLIM) obtidos até o nó 21, pelos processos 1 e 2.

No processo 1, este valor é: SLIM = 2115,94 e foi obtido no nó 21; no processo 2, o valor para SLIM é 2069,08 e foi obtido no nó 12.

Nos esquemas serão assinalados com E, os nós eliminados de considerações futuras porque possuem limite inferior maior que SLIM; e serão assinalados com I, os nós que não serão mais ramificados porque possuem valor para a função objetivo igual ao valor para a função linearizada.

Resta ainda comentar que à medida em que se impõe precisão mais rigorosa as diferenças de execução entre os dois processos se tornam menos significativas.

Justifica-se este fato, observando-se a relação considerada no teste de otimalidade:

$$\frac{SLIM - FLIM}{SLIM} < EPS \quad \Rightarrow \quad FLIM > (1 - EPS) \cdot SLIM$$

Portanto, quanto menor o valor de EPS mais próximo de SLIM deverá estar FLIM, e neste caso, exige-se que a execução termine quando os valores de SLIM e FLIM forem praticamente iguais de maneira que tanto em PR1 como em PR2 quase todas as ramificações

possíveis são efetuadas o que implica em pouca diferença entre os dois processos, com relação a número de iterações e tempo de execução.

Com precisões menos rigorosas o processo 2 tende a efetuar um número maior de iterações que o processo 1.

Isto porque no PR2 o limite superior, SLIM, decresce mais rapidamente que no PR1, contudo, o limite inferior FLIM não cresce simultaneamente, o que resulta em novas ramificações até se atingir a precisão requerida.

Já no PR1, como é escolhido o nó com menor valor para a função linearizada, FLIM, para ser ramificado, tem-se que este valor aumenta após cada ramificação e desde que a precisão requerida não seja rigorosa, o teste de otimalidade se verificará mais rapidamente que no PR2.

É importante ressaltar que nos testes efetuados as soluções ótimas obtidas pelo PR2 foram iguais ou melhores que as obtidas pelo PR1.

Este programa foi aplicado para a resolução de alguns problemas, entre eles, os propostos por Florian e Robillard [11] e por Baumol e Wolfe [5].

A tabela I abaixo mostra os resultados obtidos.

O problema 1, (P1), se refere a uma rede com 4 nós e 5 arcos com funções custo estritamente côncavas com custo fixo.

O problema 2, (P2), trata-se de uma rede de transporte com dois nós origem e 4 nós destino, totalizando 8 arcos. As funções custo consideradas são estritamente côncavas com custo fixo.

Uma rede geral, com 6 nós e 10 arcos com funções custo estritamente côncavas com custo fixo, é considerada no problema 3 (P3).

A rede do problema 4, (P4), é também de transporte, envolvendo 4 nós origem e 6 nós destino, totalizando 24 arcos. Vários testes foram efetuados com esta rede, variando as funções custo.

O problema 5, P5, trata-se da rede proposta por Florian e Robillard [11], com 4 nós e 5 arcos e funções custo lineares com custo fixo.

Finalmente, no problema 6, (P6), considera-se a rede de transporte e armazenagem proposta por Baumol e Wolfe [5]. Este problema envolve 2 fábricas, 5 locais para instalação dos armazêns e 8 centros consumidores, fornecendo uma rede com 20 nós e 55 arcos. As funções custo de transporte são lineares e as de armazenagem são estritamente côncavas, sem custo fixo.

Notação utilizada na tabela:

P: refere-se ao problema testado;

m: número de nós da rede;

NCF: número de arcos com custo estritamente côncavo com custo fixo;

NC: número de arcos com custo estritamente côncavo sem custo fixo;

ND: número de arcos com custo linear com custo fixo;

NL: número de arcos com custo linear;

PR1: processo número 1 de ramificação;

PR2: processo número 2 de ramificação;

IT: número de nós gerados pelo algorítmo;

ICOL: número de colunas preenchidas das matrizes de dados;

T: tempo total de execução em segundos;

EPS: parâmetro de tolerância.

P	m	NCF	NC	ND	NL	IT		ICOL		T		EPS
						PR1	PR2	PR1	PR2	PR1	PR2	
P1	4	5	0	0	0	3	7	3	4	0.282	0.323	$1x10^{-2}$
						9	9	4	4	0.331	0.306	$1x10^{-3}$
						9	9	4	4	0.383	0.376	$1x10^{-4}$
P2	6	8	0	0	0	87	95	25	14	2.772	2.903	$1x10^{-2}$
						103	103	25	14	3.068	2.986	$1x10^{-3}$
						103	103	25	14	3.189	3.393	$1x10^{-4}$
P3	6	8	0	0	0	11	15	5	5	0.665	0.712	$1x10^{-2}$
						19	19	5	5	0.801	0.669	$1x10^{-3}$
						19	19	5	5	0.678	0.725	$1x10^{-4}$
P4	10	24	0	0	0	525	1145	247	243	85.493	148.264	$5x10^{-2}$
	10	0	0	24	0	287	361	71	49	38.65	46.247	$1x10^{-2}$
						427	427	71	49	61.41	56.43	$1x10^{-4}$
	10	8	0	6	10	3	13	3	7	0.932	1.977	$1x10^{-2}$
						15	13	9	7	2.516	1.79	$1x10^{-3}$
						49	57	13	10	7.084	7.123	$1x10^{-4}$
P5	4	0	0	5	0	15	15	5	4	0.412	0.398	$1x10^{-2}$
P6	20	0	5	0	50	29	33	12	7	15.699	16.124	$1x10^{-2}$
						33	35	12	7	16.497	16.408	$1x10^{-3}$
						33	35	12	7	17.653	15.806	$1x10^{-4}$

TABELA I

Este programa foi aplicado à resolução de um problema - de escoamento de safra de milho em Goiás, no ano de 1977.

Os dados foram obtidos de um estudo realizado pela EMBRAPA e estão detalhados em Veloso [27].

O escoamento desta safra de milho é realizado em dois períodos, sendo que no primeiro período o produto é colhido, transportado para os centros de armazenagem e parte desta colheita é enviada aos centros de demanda . A safra restante ficará armazenada e somente será demandada no segundo período.

Neste problema consideram-se:

a) 25 cidades goianas como centróides das zonas de produção;

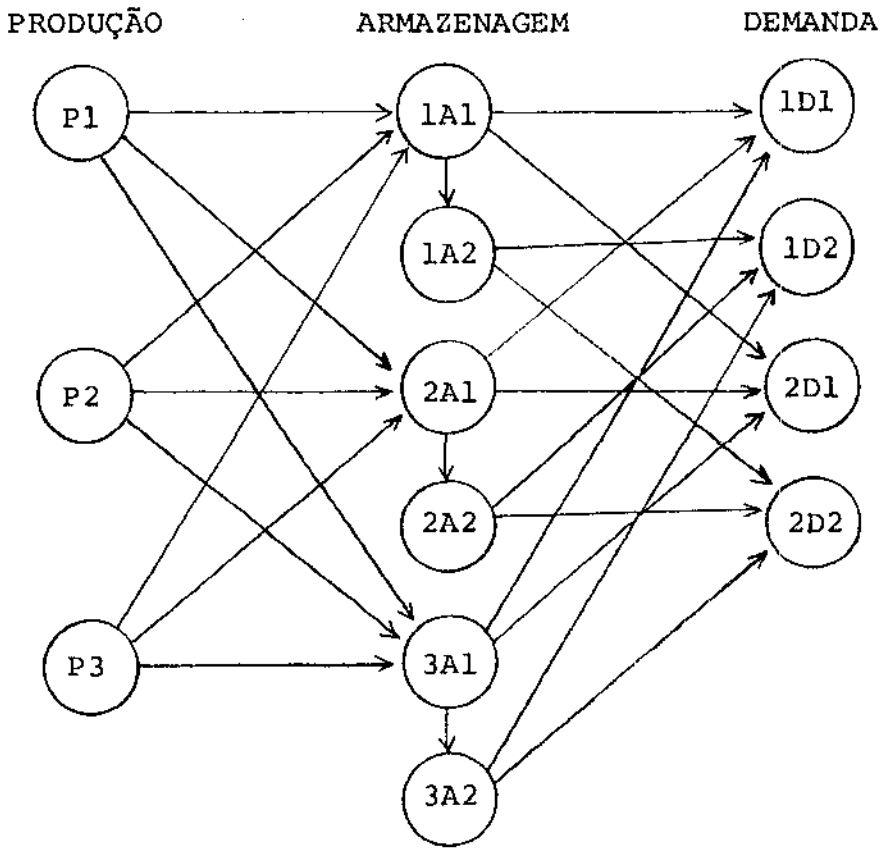
b) 22 cidades onde estão situados os armazéns;

c) 8 cidades goianas como centróides das zonas de demanda e a cidade de São Paulo e o Porto de Santos, que representam a demanda fora do Estado de Goiás.

Na tabela II serão colocadas: produção total, demanda total nos períodos I e II e capacidade de armazenagem. A unidade utilizada é tonelada.

Este problema resultou numa rede com 89 nós e 1012 arcos.

A rede abaixo, com 3 centros de produção, 3 locais para armazenagem e 2 centros de demanda, dá uma idéia da rede de escoamento estudada.



onde:

P_i: centro de produção i

iA_j: local para armazenagem i no período j

iD_j: centro de demanda i no período j.

As funções custo de transporte foram consideradas lineares.

Vários testes foram efetuados, variando-se o tipo da função custo de armazenagem e a precisão EPS.

No problema PCF as funções custo são estritamente côncavas com custo fixo e, no problema PLF estas funções são do tipo linear com custo fixo.

CIDADE	PRODUÇÃO	ARMAZENAGEM	DEMANDA	
			PERÍODO I	PERÍODO II
GOIÂNIA	-	29215	9523	30477
INHUMAS	14198	26013	457	1543
ITABERAÍ	9255	5319	-	-
ANÁPOLIS	2715	10146	4285	13715
JARAGUÁ	38336	28978	457	1543
CORUMBÁ	3289	-	-	-
PIRACANJUÍBA	2389	4951	-	-
LEOPOLDO	2747	-	-	-
PIRES	7102	4940	457	1543
PALMEIRAS	55588	56974	-	-
FIRMINÓPOLIS	23492	29952	-	-
IPORÁ	8854	20248	-	-
GOIÁS	16818	2473	457	1543
FAZENDA NOVA	10144	1428	-	-
CERES	33555	20411	457	1543
CRISTALINA	2342	-	-	-
CATALÃO	12959	6534	-	-
ITUMBIARA	172280	475123	7143	22857
RIO VERDE	174110	151531	-	-
URUAÇU	14459	-	-	-
NIQUELÂNDIA	8704	17264	-	-
JATAÍ	8002	41515	-	-
MINEIROS	2379	13750	-	-
GURUPI	10309	7028	-	-
DIANÓPOLIS	4098	67938	-	-
ARAGUAIANA	8379	2581	-	-
SÃO PAULO	-	-	110875	332628
SANTOS	-	-	25000	80000

TABELA II

Na tabela III, abaixo, a nomenclatura é a mesma da Tabela II, apenas o tempo de execução (T) é dado em minutos.

	IT		ICOL		T		EPS
	PR1	PR2	PR1	PR2	PR1	PR2	
PCF	3	29	3	14	3,35	23,1	1×10^{-2}
PCF	9	29	6	14	6,58	24,2	5×10^{-3}
PCF	15	-	9	-	12,5	-	3×10^{-3}
PCF	59	-	31	-	48,76	-	$1,5 \times 10^{-3}$
PLF	7	25	5	13	5,21	19,0	1×10^{-3}
PLF	99	-	51	-	72,30	-	4×10^{-4}

TABELA III

Pode-se observar que nem todos os testes foram resolvidos pelos dois processos.

Isto porque no PCF, a solução ótima obtida pelo PR2 com EPS = 0.005 foi melhor que a solução ótima obtida pelo PR1 em todos os testes, inclusive com EPS = 0.0015.

O mesmo ocorreu na resolução do problema PLF.

A conclusão final, com relação aos dois processos, é que o PR2 é superior ao PR1, pois, obtém soluções melhores que PR1, com precisões menos rigorosas e com tempo de execução e requerimentos menores.

Finalmente, o que permitiu a aplicação deste programa à

resolução deste problema, foram:

a) não armazenamento dos fluxos ótimos obtidos em cada nó;

b) os processos de economias de memória já descritos arteriormente.

A observação dos itens (a) e (b) permitiu reduzir em - 87% os requerimentos de memória necessários para a resolução do referido problema.

Os testes efetuados com este problema comprovam a possibilidade de se resolver problemas de grande porte com um número razoável de arcos com custo côncavo.

APÊNDICE

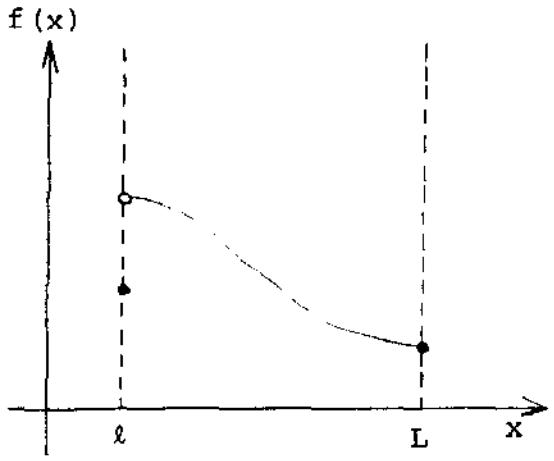
FUNÇÃO SEMICONTÍNUA INFERIOR:

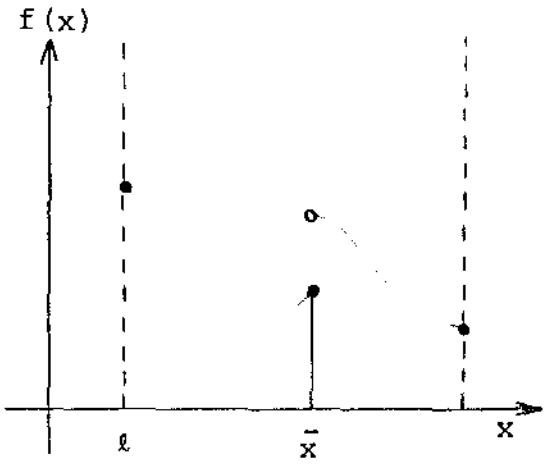
Seja C um subconjunto não vazio de $\mathbb{R}^n$.

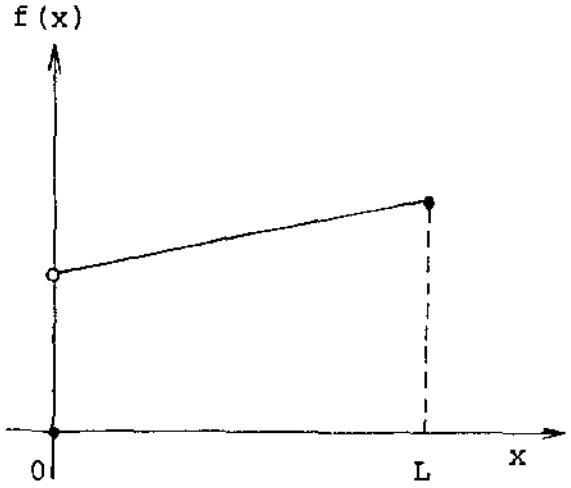
A função $f : C \to \mathbb{R}$ é semicontínua inferior em $\bar{x} \in C$, se para cada $\varepsilon > 0$ existir um $\delta > 0$, tal que se $\|x-\bar{x}\| < \delta$, $x \in C$, então $f(\bar{x})-f(x) < \varepsilon$.

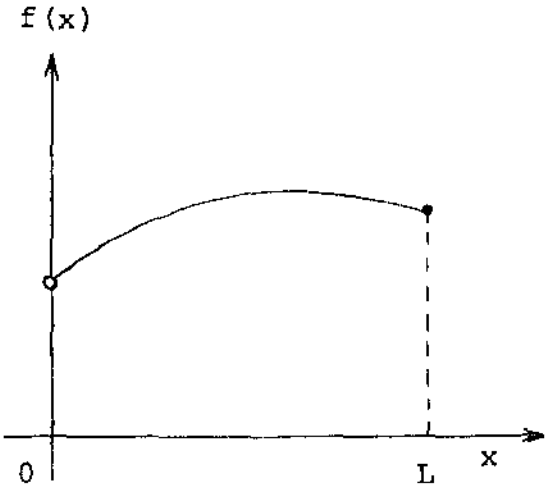
A função f será semicontínua inferior sobre C, se for semicontínua inferior em cada $x \in C$.

Exemplos:









ENVELOPES CONVEXOS:

Seja $C \subset \mathbb{R}^n$ um conjunto não vazio.

Seja f uma função definida sobre C.

Intuitivamente o envelope convexo da função f, tomado sobre C, é a "primeira" função convexa que se pode obter "abaixo" da função f.

Definição: Seja

$$P(f) = \{(x,\alpha) / x \in C, \quad \alpha \in \mathbb{R} \quad e \quad f(x) \leq \alpha\}$$

Este conjunto representa todos os pontos em $\mathbb{R}^{n+1}$, que estão sobre ou acima do gráfico da funçaõ f(x).

Seja $\langle P(f) \rangle$ o envoltório convexo do conjunto P(f). Pela definição de envoltório convexo, $\langle P(f) \rangle$ é o menor conjunto convexo que contém P(f).

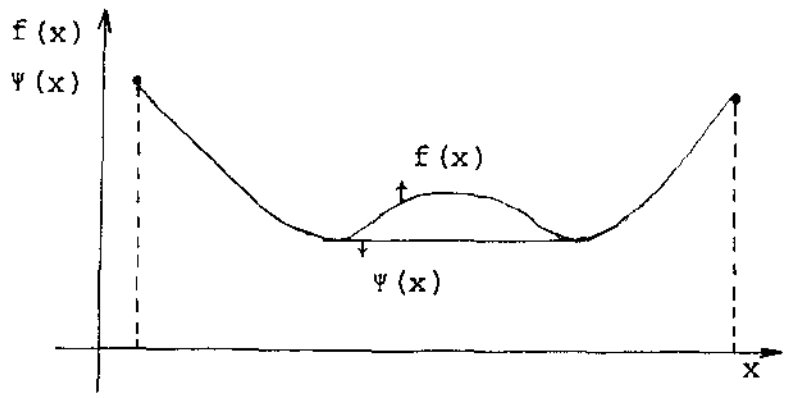
A função $\Psi(x)$ definida por:

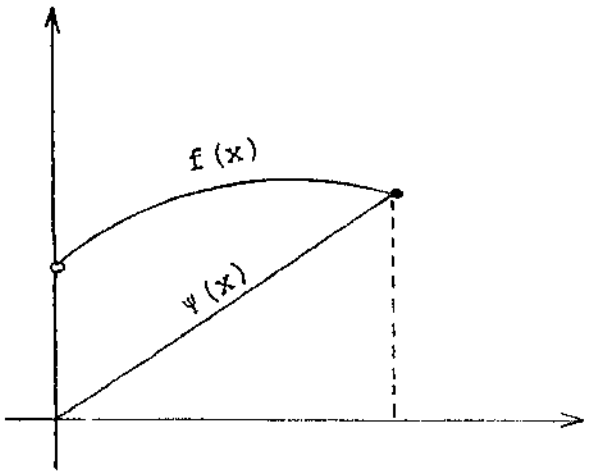
$$\Psi(x) = \inf \{\alpha / (x,\alpha) \in \langle P(f) \rangle\}$$

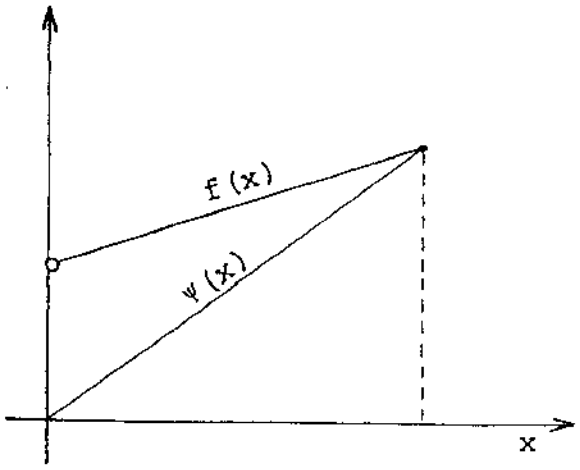
é o envelope convexo da função f(x) sobre o conjunto C.

$\Psi(x)$ é convexa e contínua sobre C.

Exemplos:




f (x)
ψ (x)


f (x)
ψ (x)
x

REFERÊNCIAS BIBLIOGRÁFICAS

[1] AGIN, N.: *Optimun Seeking with Branch and Bound.* Management Science - vol. 13, nº 4 - B176 - B185, (1966).

[2] AKINC, V. e KHUMAWALA, B.M.: *An Efficient Branch and Bound Algorithm for the Capacited Warehouse Location Problem.* - Management Science - vol. 23, nº 6, 585-594, (1977).

[3] AUTHIÉ, G.: *Otimização em Grafos.* Publicação Interna FEC / UNICAMP, 1976.

[4] BALISNKI, M.L.: *On Finding Integer Solutions to Linear Programs.* Mathematica, Princeton, N.J., (1964).

[5] BAUMOL, W.J. e WOLFE, P.: *A Warehouse Location Problems.* Operations Research, 6, 252-263, (1958).

[6] BAZARAA, S.M. e JARVIS, J.J.: *Linear Programming and Network Flows.* John Wiley & Sons, (1977).

[7] EFROYMSON, M.A. e RAY, T.L.: *A Branch and Bound Algorithm for Plant Location.* Operations Research, 14, 361-368, (1966).

[8] FALK, J.E. e SOLAND, R.N.: *An Algorithm for Separable Nonconvex Programming Problems.* Management Science, 15, 550-569, (1969).

[9] FALK, J.E. e HOFFMAN, K.R.: *A Sucessive Underestimation Method for Concave Minimization Problems.* Mathematics of Operations Research, vol. 1, nº 3, 251-259, (1976).

[10] FELDMAN ET AL.: *Warehouse Location under Continuous Economies of Scale.* Management Science, vol. 12, (1966).

[11] FLORIAN, M. e ROBILLARD, P.: *An Implicit Enumeration Algorithm for the Concave Cost Network Flow Problem.* Management Science, vol. 18, nº 3, 184-193, (1971).

[12] FRANÇA, P.M.: *Problemas de Localização: Solução por Decomposição.* Tese de Doutorado, FEC-UNICAMP, (1979).

[13] GEOFFRION, A.M. e GRAVES, G.W.: *Multicommodity Distribution System Design by Benders Decomposition.* Management Science, vol. 20, 822-844, (1974).

[14] GRAY, P.: *Exact Solution of the Fixed-Charge Transportation Problem.* Operations Research, 19, 1529-1538, (1971).

[15] JENSEN, P.A. e BARNES, J.W.: *Network Flow Programming.* New York, John Willey & Sons, (1980).

[16] KAUFMAN ET AL.: *A Plant and Warehouse Location Problem.* Operations Research Quaterly, vol. 28, 3, 547-554, (1977).

[17] KHUMAWALA, B.M.: *An Efficient Branch and Bound Algorithm for the Warehouses.* Management Science, vol. 18, nº 12, B718-B731, (1972).

[18] KUHEN, A.A. e HAMBURGER, M.J.: *A Heuristic Program for Locating Warehouses.* Management Science, 9, 643-666, (1963).

[19] LAWER, E.L. e WOOD, D.E.: *Branch and Bounds Methods: A Survey.* Operations Research, 14, 699-719, (1966).

[20] MANNE, A.S.: *Plant Location and Economies of Scale - Decentralization and Computations.* Management Science, 11, - 213-235, (1964).

[21] MURTHY, K.G.: *Solving the Fixed Charge Problem by Ranking the Extreme Points.* Operations Research, 16, 268-279 , (1968).

[22] REVELLE, C.; MARKS, D. e LIEBMAN, J.C.: *An Analysis of Private and Public Sector Location Models.* Management Science, vol. 16, nº 11, 692-707, (1970).

[23] SÁ, G.: *Branch and Bound and Aproximate Solutions to the*

Capacited Plant Location Problem. Operations Research , 17, 1005-1016, (1969).

[24] SOLAND, R.M.: *Optimal Facility Location with Concave Costs*. Operations Research, 373-382, (1971).

[25] THOAI, N. e TUY, H.: *Convergent Algorithms for Minimizing a Concave Function*. X International Symposium on Mathematical Programming, Montreal, 1979.

[26] TUY, H.: *Concave Programming Under Linear Constraints*. Soviet Mathematics, 5, 1437-1440, (1964).

[27] VELOSO, R.F.: *Análise Logística da Distribuição Física de Grãos no Estado de Goiás*. Tese de Mestrado, COPPE-UFRJ , (1979).

[28] ZANGWILL, W.I.: *Minimum Concave Cost Flows in Certain Networks*. Management Science, 14, 429-450, (1968).