MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO
AUTHORIZATION FOR PUBLICATION

PALAVRAS CHAVES/KEY WORDS
AUTORES/KEYWORDS:
P-MEDIANAS
RELAXAÇÃO LAGRANGEANA
"BRANCH AND BOUND"

AUTORIZADA POR/AUTHORIZED BY
Ralf Gielow
Pres.Cons.Pós-Graduação

AUTOR RESPONSÁVEL
RESPONSIBLE AUTHOR
Hilcéa Ferreira Marengoni

DISTRIBUIÇÃO/DISTRIBUTION
[ ] INTERNA / INTERNAL
[X] EXTERNA / EXTERNAL
[ ] RESTRITA / RESTRICTED

REVISADA POR / REVISED BY
Luiz Antonio N. Lorena

CDU/UDC
519.688

DATA / DATE
Setembro 1989

PUBLICAÇÃO Nº
PUBLICATION NO
INPE-4924-TDL/384

ORIGEM
ORIGIN
PG/CPD

PROJETO
PROJECT
FRH/ANS

Nº DE PAG.
NO OF PAGES
109

ULTIMA PAG.
LAST PAGE
A.24

VERSÃO
VERSION

Nº DE MAPAS
NO OF MAPS

TÍTULO/TITLE

UM ALGORITMO EXATO PARA O PROBLEMA DAS P-MEDIANAS

AUTORES/AUTHORSHIP

Hilcéa Ferreira Marengoni

RESUMO - NOTAS / ABSTRACT - NOTES

Este trabalho descreve o estudo de alguns métodos exatos e heurísticos para resolver o problema das p-medianas. Em particular enfoca um algoritmo exato baseado em uma formulação de programação inteira do problema. Um algoritmo do tipo "branch and bound" é utilizado e os limitantes são obtidos através da relaxação lagrangeana do problema usando um método de otimização por subgradientes. Uma estrutura de dados explorando a estrutura matricial do problema é desenvolvida e a implementação é feita em microcomputadores tipo IBM-PC.

OBSERVAÇÕES / REMARKS

Dissertação de Mestrado em Análise de Sistemas e Aplicações, aprovada em 02 de junho de 1989.

INSTITUTO DE PESQUISAS ESPACIAIS

DISSERTAÇÃO

UM ALGORITMO EXATO PARA O PROBLEMA DAS P-MEDIANAS

SUBMETIDA POR

Hilcéa Ferreira Marengoni

Em cumprimento parcial dos requisitos exigidos para obtenção do título de Mestre em Análise de Sistemas e Aplicações

1989

Aprovada pela Banca Examinadora
em cumprimento a requisito exigido
para a obtenção do Título de Mestre
em Análise de Sistemas e Aplicações

Dr. Paulo Renato de Morais — Presidente

Dr. Luiz Antonio Nogueira Lorena — Orientador

Dr. Edson Luiz França Senne — Orientador

Dr. Edgard Dias Batista Jr. — Membro da Banca
- convidado -

Candidata: Hilcéa Ferreira Marengoni

São José dos Campos, 02 de junho de 1989

Ao Maurício que, com apoio, carinho e incentivo, muito contribuiu para a realização deste trabalho.

AGRADECIMENTOS

Ao INPE, Instituto de Pesquisas Espaciais, pela oportunidade de concretização deste trabalho.

Ao pesquisador Acioli Antônio de Olivo, Msc. pelo incentivo, apoio e primeira orientação.

Aos Drs. Luiz Antônio Nogueira Lorena e Edson Luiz França Senne pela orientação e principalmente pela atenção que sempre me dispensaram.

A todos que direta ou indiretamente colaboraram na realização deste trabalho.

RESUMO

Este trabalho descreve o estudo de alguns métodos exatos e heurísticos para resolver o problema das p-medianas. Em particular enfoca um algoritmo exato baseado em uma formulação de programação inteira do problema. Um algoritmo do tipo "branch and bound" é utilizado e os limitantes são obtidos através da relaxação lagrangeana do problema usando um método de otimização por subgradientes. Uma estrutura de dados explorando a estrutura matricial do problema é desenvolvida e a implementação é feita em micro-computadores tipo IBM-PC.

ABSTRACT

This work reports a study of a few heuristic and exact methods for solving the p-median problem. It particularly covers an exact algorithm based on an integer programming formulation of the problem. A "branch and bound" algorithm is presented and the bounds are obtained by solving the lagrangean relaxation and using the subgradient optimization method. A data structure that takes into account the matricial character of the problem is developed and the algorithm is implemented to run on a IBM-PC micro-computer.

SUMÁRIO

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

A determinação das p-medianas de uma rede é um problema clássico de localização. O seu objetivo é achar p nós - chamados de medianas da rede - que minimizem a soma das distâncias entre cada nó e sua mediana mais próxima.

Um grande número de algoritmos (exatos e heurísticos) tem sido desenvolvido para a solução do problema das p-medianas (PPM). Alguns autores tentaram resolver o problema através de procedimentos heurísticos; houve aqueles que usaram algoritmos do tipo "branch and bound" . Mais recentemente, surgiram abordagens baseadas em Programação Linear Inteira usando relaxação lagrangeana combinada com um método de otimização por subgradientes. Beasley(1985) combinando um método de subgradientes e um procedimento de busca em árvores ("branch and bound"), conseguiu resolver problemas com até 900 nós graças ao emprego do super computador de processamento vetorial CRAY-1S.

O objetivo deste trabalho é fazer um estudo do problema e dos métodos (heurísticos e exatos) existentes para sua resolução. Espera-se obter um algoritmo para resolução exata do problema e desenvolver uma estrutura de dados que explore a estrutura matricial do problema, para implementação em micro-computadores do tipo IBM-PC. Portanto, não se tem a pretensão de resolver problemas da dimensão daqueles resolvidos por Beasley(1985).

Inicialmente, no segundo capítulo, faz-se a definição formal do problema e são citadas algumas de suas aplicações mais comuns. Posteriormente, é feito um relato detalhado sobre algumas abordagens importantes na consolidação do PPM.

No Capítulo 3, apresenta-se o algoritmo usado para resolver o problema.

No Capítulo 4 descrevem-se a implementação do algoritmo e os testes realizados, sendo apresentados, no Capítulo 5, os comentários finais.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

2.1 - DEFINIÇÃO DO PROBLEMA

Uma questão que normalmente aparece nos sistemas de distribuição industrial é a localização de armazéns, fábricas ou outras facilidades numa rede, comumente referido como Problema de Localização de Armazéns (PLA). O problema que vai ser tratado, conhecido na literatura como das P-MEDIANAS (PPM), é uma versão do PLA que consiste em localizar p facilidades (medianas) em uma rede tal que seja minimizada a soma das distâncias entre cada nó e sua mediana mais próxima.

O PPM pode ser formulado como o seguinte problema (P) de programação inteira zero-um:

$$\text{MIN } Z = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} dij \; xij \right) \tag{2.1}$$

$$\text{Sujeito à:} \quad \sum_{i=1}^{n} xij = 1, \qquad j = 1,2,\ldots,n; \tag{2.2}$$

$$\sum_{i=1}^{n} xii = p \; ; \tag{2.3}$$

$$xij \leq xii \; ; \tag{2.4}$$

$$xij \in \{0,1\} \qquad i,j = 1,2,\ldots,n \tag{2.5}$$

onde (dij) é qualquer matriz não negativa com dii = 0 para todo i; (xij) é uma matriz de alocações onde xij = 1 se o nó j está alocado ao nó i e xij = 0 caso contrário; xii = 1 significa que o nó i é mediano (uma facilidade está localizada nele), xii = 0 caso contrário; p é o número de medianas (facilidades) a serem localizadas na rede e n é o número total de nós da rede. Considera-se V como o conjunto de todos os nós (qualquer nó pode ser uma mediana).

As restrições (2.2) e (2.4) garantem que cada nó j é alocado somente a um único nó i que, por sua vez, precisa ser uma mediana, isto é, xii = 1. A restrição (2.3) determina o número exato de medianas que devem ser localizadadas (p) e a (2.5) impõe a integralidade.

Os problemas de localização de facilidades em uma rede são variados e usualmente complexos. Dentre os critérios de otimização utilizados, um é o de minimizar a soma total das distâncias percorridas.

Em estatística, um problema de agrupamento ("Clustering") de observações em grupos homogêneos possui uma formulação idêntica ao problema de localização de medianas em uma rede. Kusiak et al. (1985) desenvolveram três formulações distintas usando programação inteira e cobrindo variações do problema tais como: fixação do número de "clusters" e restrição quanto ao número de elementos dentro de cada "cluster".

O problema descrito que incorpora a restrição quanto ao número de "clusters" possui formulação idêntica a (P), onde n é o número de elementos, p é o número requerido de "clusters", dij é a distância do elemento i ao elemento j ($dij \geq 0$ para qualquer $i \neq j$, i,j=1,2,...,n e dii=0 para qualquer que seja i=1,2,...,n), xij=1 se o i-ésimo elemento pertence ao j-ésimo "cluster" e xij=0 caso contrário. A função objetivo é minimizar a soma total das distâncias de cada "cluster" para o seu mediano. Kusiak et al. (1985) desenvolveram

técnicas heurísticas, empregando Lagrangeano e autovetores, para resolver o problema.

Um outro problema importante que ocorre freqüentemente em estatística envolve a determinação de limites para estratos a serem usados em amostragem estratificada. Mulvey (1983) descreve um método prático para estratificação de uma população de observações baseado em análise ótima de "clusters". O objetivo da estratificação é construir uma partição tal que observações dentro de um estrato sejam homogêneas. O problema é definido como um modelo de otimização determinístico com variáveis inteiras cuja formulação é bastante semelhante a do PPM.

2.2 - HISTÓRICO

A seguir, serão apresentados, de forma sintética, alguns algoritmos importantes na consolidação do PPM. A maior parte das provas será omitida pois elas escapam do escopo deste trabalho. Entretanto, as referências apropriadas estão indicadas, de tal forma que os interessados podem examinar em detalhes qualquer dos algoritmos apresentados. Em alguns algoritmos apresentam-se exemplos numéricos para ilustrar os passos do algoritmo (é importante ressaltar que nestes casos o objetivo é apenas ilustrar, portanto não se pretende chegar à solução ótima).

As abordagens na resolução do PPM serão classificadas, neste trabalho, em três categorias: (1)método de enumeração, (2)métodos heurísticos e (3)métodos exatos e programação matemática baseada no primal e dual. Para localização de uma única facilidade, ou quando a rede é pequena, os métodos de enumeração fornecem um procedimento manual para ilustrar o conceito de localização de p-medianas e partição do conjunto de nós em p subconjuntos. Procedimentos heurísticos iterativos baseiam-se em métodos intuitivos de tentativa e erro e não podem garantir uma solução ótima, mas podem ser aplicados a qualquer estrutura geral de redes, fornecendo boas soluções (próximas do ótimo). Os métodos exatos englobam procedimentos

do tipo "branch and bound" e as abordagens de programação matemática são baseadas em uma formulação de programação inteira do PPM. Estas últimas têm atraído muita atenção e têm provado sucesso razoável para redes gerais. O PPM pode ser formulado como um problema de programação inteira e resolvido através de algoritmos baseados no primal e dual do PPM.

2.2.1 - MÉTODO DE ENUMERAÇÃO

O problema mediano mais simples em uma rede é o da determinação de uma única facilidade de tal forma que seja minimizada a média das distâncias.

Hakimi (1964) foi o primeiro a examinar este problema preocupando-se basicamente com duas questões: encontrar a localização ótima dos centros de ligação em uma rede de comunicação e localizar o melhor lugar para construir uma estação policial em um sistema de estradas. O problema da rede de comunicação pode ser descrito como um sistema de interconexão de telefones onde existe, normalmente, um centro de ligação S que recebe todo o tráfico de mensagens e o distribui para seu destino. Um modelo para este sistema pode ser considerado como um grafo finito ponderado G cujos pesos (números não-negativos) estão ligados aos vértices e ramificações da rede. O peso wi que é ligado ao ramo bi representa o tamanho (ou custo por unidade) daquele elemento. O peso hi ligado ao vértice vi de G representa o número de linhas que devem ser conectadas entre o vértice vi e o centro de ligação S para manipular o fluxo de informação que tanto se origina como termina em vi. O problema é encontrar a localização exata do centro de ligação S tal que, o tamanho total das linhas seja mínimo.

O problema de localização ótima da estação policial é de certa forma similar. Hakimi (1964) considerou um número de comunidades de diferentes tamanhos que estão interconectadas por um sistema de estradas. O objetivo é encontrar a localização da estação policial P tal que a distância máxima até P seja mínima.

Da discussão acima fica claro que os conceitos teóricos usuais de grafos para centros e medianas não podem ser usados.

Hakimi (1964) definiu, então, um grafo G e um ponto x em G que pode estar sobre algum ramo de G e ser ou não um vértice de G. A distância entre quaisquer dois pontos x e y em G, representada por d(x,y), é o tamanho de um caminho mais curto em G entre os pontos x e y, onde o tamanho de um caminho é a soma das distâncias multiplicadas pelos pesos dos ramos do caminho.

Definiu-se um ponto x0, de um grafo ponderado G, como sendo um centro absoluto de G, se para todo ponto x em G:

$$\text{MAX}\ \ hi\ d(vi,x0) \leq \text{MAX}\ \ hid(vi,x), \quad \forall x \tag{2.6}$$

Similarmente, definiu-se um ponto y0 em G como sendo mediana absoluta de G, se para todo ponto y em G:

$$\sum_{i=1}^{n} (hi\ d(vi,y0)) \leq \sum_{i=1}^{n} (hi\ d(vi,y)), \quad \forall y \tag{2.7}$$

A mediana absoluta do grafo deve ser identificada com a localização ótima do centro de ligação na rede de comunicação, e o centro absoluto deve ser identificado com a localização ótima da estação policial no sistema de estradas.

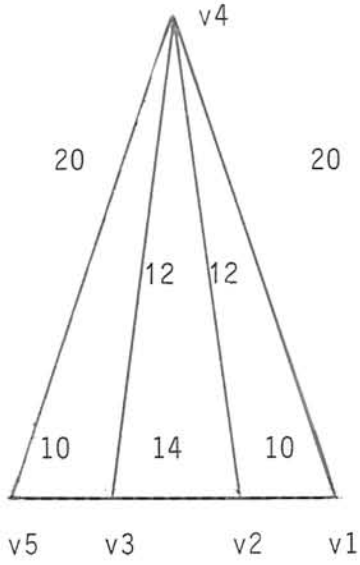
Finalmente Hakimi (1964) demonstrou um teorema que é de grande valia para estudos posteriores do PPM:

"Uma mediana absoluta de um grafo é sempre um vértice do grafo"

Então, concluiu-se que, encontrar a localização ótima em um centro de ligação de uma rede de comunicação pode se limitar a encontrar o vértice mediano do grafo correspondente. E para o problema

da localização da estação policial deve-se encontrar o centro absoluto do grafo correspondente.

Para ilustração, seja o grafo abaixo:



A matriz de distâncias deste grafo é:

$$D = \begin{bmatrix} 0 & 10 & 24 & 20 & 34 \\ 10 & 0 & 14 & 12 & 24 \\ 24 & 14 & 0 & 12 & 10 \\ 20 & 12 & 12 & 0 & 20 \\ 34 & 24 & 10 & 20 & 0 \end{bmatrix}$$

É facilmente determinado que v4 é o centro absoluto do grafo, pois:

$$MAX\ (d(vi,vj)) = MAX\ (d(v1,vj)) = 34 = d(v1,v5)$$

$$MAX\ (d(v2,vj)) = 24 = d(v2,v5)$$

MAX (d(v3,vj)) = 24 = d(v3,v1)

MAX (d(v4,vj)) = 20 = d(v4,v1) = d(v4,v5)

MAX (d(vi,vj)) = 34 = d(v5,v1)

MIN [MAX (d(vi,vj))] = d(v4,v1) = d(v4,v5)

Também tem-se que v2 e v3 são duas chances possíveis de medianas no grafo, pois:

$$\sum_{i=1}^{n} d(v1,vi) = 88$$

$$\sum_{i=1}^{n} d(v2,vi) = 60$$

$$\sum_{i=1}^{n} d(v3,vi) = 60$$

$$\sum_{i=1}^{n} d(v4,vi) = 64$$

$$\sum_{i=1}^{n} d(v5,vi) = 88$$

Em uma publicação seguinte Hakimi (1965) generalizou o conceito de mediana, em um grafo ponderado, para multi-medianas e apresentou um procedimento simples de enumeração. O problema tratado pode ser descrito assim: em uma rede de comunicação N, como em um sistema de interconexão de telefones, existe usualmente um número de centros de ligação S1,S2,S3, ... , Sp; todo fluxo de mensagens que passa dentro da rede deve chegar até um centro de ligação para então ser processado e enviado à seu destino; assume-se que o custo para manter as ligações é desprezível. Este problema pode ser modelado através de um grafo ponderado G cujos pesos wi(tamanho ou custo por

unidade de cada ramo) e hi(número de linhas que devem ser conectadas entre o vértice vi e o centro de ligação para permitir o fluxo de informações) estão ligados aos ramos e nós de G respectivamente.

Assumiu-se que qualquer centro de ligação pode estar em qualquer lugar sobre o ramo de G, isto é, a localização do centro de ligação não é necessariamente limitada aos vértices de G. O problema é encontrar a distribuição dos centros de ligação S1,S2,...,Sp tal que o tamanho total das linhas seja mínimo (distribuição ótima dos centros de ligação).

Seja um grafo ponderado G. Um ponto x em G é um ponto sobre o ramo bi de G que pode ou não ser um vértice de G. A distância entre quaisquer dois pontos x e y em G, denotada por d(x,y), é o tamanho do caminho mais curto em G entre os pontos x e y, onde o tamanho do caminho é a soma dos pesos dos ramos do caminho. Seja Xp o conjunto de p pontos x1,x2,...,xp em G e seja

$$d(vi,Xp) = MIN\ [d(vi,x1),d(vi,x2),...,d(vi,xp)]$$

O conjunto de pontos Xp* é uma mediana de G para todo Xp em G se:

$$\sum_{i=1}^{n} (hi\ d(vi,Xp^*)) \leq \sum_{i=1}^{n} (hi\ d(vi,Xp))$$

onde hi é o peso ligado ao vértice vi em G.

Claramente o conjunto de pontos Xp* pode ser identificado como localização ótima dos centros de ligação numa rede de comunicação. Hakimi (1965) não apresenta um procedimento para encontrar Xp*. Mas, por outro lado, demonstra um teorema que é o principal passo nesta direção:

"Existe um subconjunto Vp* de V contendo p vértices tal que para todo conjunto de p pontos X em G:

$$\sum_{i=1}^{n} (hi\ d(vi,X)) \quad \geq \quad \sum_{i=1}^{n} (hi\ d(vi,Vp^*))"$$

O teorema acima prova que para encontrar as p-medianas deve-se enumerar todos os subconjuntos de V contendo p vértices e examinar as somas. Esta técnica de enumeração parece razoável para n não muito grande, já que o número de subconjuntos possíveis é Cn.p, onde:

$$Cn.p = \frac{n!}{p!\ (n-p)!}$$

2.2.2 - MÉTODOS HEURÍSTICOS

Vários procedimentos heurísticos foram bem sucedidos na resolução do PPM. Alguns deles são usados para gerar soluções iniciais boas e/ou regras de decisão na escolha de nós, que ajudem na obtenção de convergência mais rápida de algoritmos usando programação matemática.

Um dos primeiros procedimentos heurísticos para o PPM foi desenvolvido por Maranzana (1964). Ele considerou o problema de localização ótima de pontos de abastecimento (estabelecimentos comerciais e fábricas para servir fregueses distribuidos em uma rede de cidades) com relação aos custos de transporte. Se os custos de transporte são uniformes e lineares com relação à distância, o custo total de transporte é proporcional a soma das distâncias entre os pontos de abastecimento até as cidades servidas, cada uma ponderada pelo volume de remessa. Sempre é considerado o caminho mais curto de um nó para outro (assume-se que existe algum caminho conectando todos os pares de nós da rede).

Um nó no qual um ponto de abastecimento está localizado é chamado fonte e um nó que possui uma demanda a ser satisfeita é referenciado como destino.

Em termos matemáticos tem-se o seguinte problema:

Dados um conjunto P de n pontos p1,p2,...,pn, um conjunto de pesos associados w1,w2,...,wn e uma matriz de distâncias não-negativa, n-dimensional e simétrica [dij], quer-se encontrar p fontes, (px1,px2,...,pxp) e uma partição associada de P em p subconjuntos de destinos (Px1,...Pxp) servidos respectivamente pelas p fontes, tais que:

$$\sum_{i=1}^{n} \sum_{Pj \in Pxj} Dxi,j \; wj \qquad (2.8)$$

seja mínima, onde Dij é a distância mínima entre pi e pj .

O custo do transporte é proporcional à soma (2.8).

Segue-se uma rápida descrição do método iterativo proposto por Maranzana que pode ser aplicado a uma seleção inicial de p pontos de abastecimento para produzir, sucessivamente, seleções melhores.

O algoritmo começa com uma seleção arbitrária de p fontes (px1,...,pxp) e partição da rede em subconjuntos ((Px1,...Pxp) onde Pxi = { pk; Dk,xi $\leq$ Dk,xj p/todo j }) a serem servidos por estas fontes. Isto é acompanhado pela associação de cada ponto com sua fonte mais próxima. Depois, o centro de gravidade de cada conjunto na partição é determinado e as fontes originais sao substituídas por estes pontos. Este processo é repetido até que as fontes não mudem mais. Considera-se pj um centro de gravidade de Q $\subseteq$ P se:

$$\sum_{pk \in Q} Dj,k\ wk < \sum_{pk \in Q} Di,k\ wk \qquad \text{para todo } i$$

Segue o seguinte exemplo. Seja a matriz de distâncias para um problema com n = 6 e p = 2. Suponha-se que os pesos já estão devidamente associados na matriz:

$$D = \begin{bmatrix} 0 & 79 & 95 & 75 & 66 & 64 \\ 79 & 0 & 49 & 21 & 86 & 52 \\ 95 & 49 & 0 & 58 & 6 & 5 \\ 75 & 21 & 58 & 0 & 83 & 64 \\ 66 & 86 & 6 & 83 & 0 & 38 \\ 64 & 52 & 5 & 64 & 38 & 0 \end{bmatrix} \qquad (2.9)$$

Sejam inicialmente os dois fontes p1 e p2. Associados à p1 e p2 determinam-se as partições correspondentes:

Px1 = {p1, p5}
Px2 = {p2, p3, p4, p6}

Determinam-se os centros de gravidade:

Para Px1 --> o centro de gravidade é p5
Para Px2 --> o centro de gravidade é p3

Consideram-se agora os fontes p3 e p5 e associados à eles determinam-se:

Px5 = {p1, p5 }
Px3 = {p2, p3, p4, p5 }

Os centros de gravidade são p5 e p3. Atingiu-se a convergência: as medianas são os nós 3 e 5. Maranzana (1964) sugere que se aplique o algoritmo para diferentes valores iniciais para garantir que p3 e p5 são os melhores nós que podem ser encontrados (otimalidade).

Embora o algoritmo possa falhar em convergir para uma solução ótima, Maranzana (1964) afirma que repetidas aplicações com valores iniciais alternativos asseguram uma boa solução. Poucos testes computacionais foram desenvolvidos por Maranzana (1964), embora ele indique que seu algoritmo apresentou resultados satisfatórios em problemas amostrais.

Teitz e Bart (1968) apresentaram bons resultados com a utilização de uma heurística de substituição de vértices. É uma abordagem alternativa que se concentra na definição formal dos vértices medianos generalizados e sua matriz de distâncias associada.

Teitz e Bart (1968) consideraram uma rede com n nós e para cada um deles uma dada demanda hi. Para cada par de nós i,j da rede, a distância mínima di,j é calculada. Cada vértice vi no grafo G é ponderado por hi e cada arco i-j pela distância mínima di,j. Seja D a matriz simétrica n X n de distâncias (dij) entre os pares vi,vj para um grafo com pesos iguais nos vértices. O vértice mediano vk será aquele cuja soma dos elementos na coluna correspondente em D seja minimizada:

$$dj = \sum_{i=1}^{n} dij \quad (j = 1,2,\ldots,n) \tag{2.10}$$

Então, vk é mediana se, somente se:

$$dk = MIN\ (di,d2,\ldots,dn) \tag{2.11}$$

Se o grafo G tem diferentes pesos nos vértices, é necessário redefinir a matriz das distâncias de G:

$$R = [H.\ D] = [rij] = [hi.dij]$$

A generalização dos vértices medianos seguem logicamente de (2.10) e (2.11) na matriz R.

Considere-se a definição dos vértices medianos generalizados:

$$rm = \sum_{i=1}^{n} (rik)$$

$$rm^* = MIN\ [r1,r2,\ldots,r(Cn,p)]$$

$$rm^* \leq rm \quad (m=1,\ldots,Cn,p)$$

Para cada conjunto possível de vértices fontes Vpm, pode-se construir uma submatriz Rpm de R juntando-se as p colunas correspondentes aos vértices vj em Vpm. A fonte da qual qualquer vértice destino vi é servido é definido como vk tal que:

$$rik \leq rij \qquad \text{para todo } vk, vj \in Vpm$$

Esta é a i-ésima linha mínima em Rpm. A distância total rm para o m-ésimo conjunto de fontes será a soma destas linhas mínimas.

Vários tipos de efeitos resultantes na distância ponderada total podem ocorrer caso troque-se um vértice vj no conjunto de fontes por outro vb.

Se rij não for a i-ésima linha mínima de Rpm, então, nenhuma mudança ocorreria na contribuição de r na i-ésima linha . Se rij for a i-ésima linha mínima, então, sua substituição por rib pode produzir vários resultados dependentes de:

$$rib \leq rij \qquad (a)$$

ou $$rij \leq rib \leq riS \qquad (b)$$

ou $$rij \leq riS \leq rib \qquad (c)$$

onde riS é a distância ponderada de vi a vs para a qual

$$rij \leq riS \leq riS^* \qquad \text{p/ } vS^* \in Vpm \quad \text{e} \quad vS^* \neq vj, vS$$

Em outras palavras, riS é a segunda menor i-ésima linha em Rpm.

No caso (a) a contribuição de r na i-ésima linha da substituição de vb por vj é agora incrementada por:

$$i \Delta bj = rij - rib$$
$$i \Delta bj \leq 0$$

No caso (b) a contribuição de r na i-ésima linha é incrementada por:

$$i \Delta bj = rij - rib$$
$$i \Delta bj \geq 0$$

No caso (c) o incremento comparável é:

$$i \Delta bj = rij - ris$$
$$i \Delta bj \geq 0$$

Se vai ser pior substituir o vértice vb pelo vj depende do efeito de rede do incremento somado sobre todas as linhas:

$$\Delta bj = \sum_{i=1}^{n} i \Delta bj$$

Substituir vb por vj reduz a distância total somente se:

$$\Delta bj < 0$$

Estas observações sugerem uma abordagem para encontrar a mediana generalizada que envolve um processo iterativo de substituição de um único vértice, seguindo os seguintes passos:

PASSO 1: Seleciona-se algum subconjunto inicial de vértices fontes; por conveniência, sejam v1, v2, ... ,vj, ..., vp.

PASSO 2: Para cada vértice vj encontra-se seu P1j associado (de vértices destinos) como no método da partição (Maranzana, 1964):

$$Pij = \{vi \;/\; rij < rik \text{ , qq seja } vk \in V1\}$$

e, computa-se a distância total para o sistema r1, assim:

$$r1 = \sum_{vj \in V1} \left(\sum_{V1 \in P1} rij \right)$$

PASSO 3: Seleciona-se algum vértice, vb, fora do conjunto de fontes.

PASSO 4: Para cada vértice vj em V1 substitua vb e calcule Δbj.

PASSO 5: Encontra-se o vértice vk em V1 tal que:

$$\Delta bk < 0$$

$$\Delta bk = MIN \quad \Delta bj \qquad j=1,\ldots,p$$

PASSO 6: Se existe um vértice que satisfaça às condições do PASSO 5, substitui-se vb por vk no conjunto de fontes, chama-se o novo conjunto de V2 e calcula-se r2. Se nenhum vértice satisfaz às condições, mantem-se o conjunto de fontes V1 e segue-se para o PASSO 7.

PASSO 7: Seleciona-se outro vértice que não esteja em V1 nem em V2 e não experimentado anteriormente, e repetem-se os PASSOS 4 até 6.

PASSO 8: Quando todos os vértices do complemento de V1 tiverem sido experimentados, define-se o subconjunto de fontes resultante Vt como um novo V1 e repetem-se os PASSOS 2 até 7. Chama-se cada repetição desta de ciclo completo.

PASSO 9: Quando um ciclo completo (PASSOS 2 à 8) resultar em nenhuma redução em r, o procedimento termina. O Vt final é uma estimativa dos vértices p-medianos de G.

Seja a matriz de distâncias (2.9) definida anteriormente. Neste caso consideram-se os pesos hi=1 (então R = [HD] = [D]). Segue um esboço da primeira parte do primeiro ciclo do algoritmo proposto.

V = {v1,...,v6}
Seja V1 = {v1, v2}
Para v1 --> P11 = {v5}
Para v2 --> P12 = {v3, v4, v6}
r1 = r15 + r23 + r24 + r26 = 188

Seja vb = v3

Para v1 --> V1 = {v2, v3} e $\Delta b1 = \sum_{i=1}^{n} i \Delta b1$

1 Δ b1 = r11 - r15 = 0 - 64 = -64 (caso c)
2 Δ b1 = 79 - 49 = 30 (caso a)
3 Δ b1 = 95 - 0 = 95 (caso a)
4 Δ b1 = 75 - 58 = 17 (caso a)
5 Δ b1 = 66 - 6 = 60 (caso a)
6 Δ b1 = 66 - 5 = 59 (caso a)

Δb1 = 64 - 30 - 95 - 17 - 60 - 59 = -197

Para v2 --> V1 = {v1, v3}

1 Δ b2 = 79 - 95 = -16 (caso b)
2 Δ b2 = 0 - 21 = -21 (caso c)
3 Δ b2 = 49 - 0 = 49 (caso a)
4 Δ b2 = 59 - 21 = 37 (caso a)
5 Δ b2 = 86 - 6 = 80 (caso a)
6 Δ b2 = 52 - 5 = 47 (caso a)

$$\Delta b2 = 16 + 21 - 49 - 37 - 80 - 47 = -176$$

Como v1 em V1 satisfaz as condições:

$$\Delta b1 < 0$$
$$\Delta b1 = MIN \ (\Delta b1, \ \Delta b2).$$

Substitui-se o vértice v3 pelo vértice v1: V2 = {v2, v3} e r2 = 111. Agora seleciona-se outro vértice que não esteja contido nem em V1 nem em V2 (e não experimentado anteriormente) e repetem-se os PASSOS 4 a 6, e assim por diante.

À primeira vista este procedimento parece trabalhoso em comparação com o método de Maranzana (1964). Entretanto, os passos são simples e o tempo de processamento é razoável. Em nenhum dos casos analisados por Teitz e Bart (1968) houve mais que 4 ciclos até encontrar as p-medianas. Como no Método de Maranzana (1964), não há segurança de que Vt e rt sejam valores ótimos globais.

2.2.3 - MÉTODOS EXATOS E PROGRAMAÇÃO MATEMÁTICA

Dentre os métodos exatos encaixam-se os procedimentos do tipo "branch and bound". Jarvinen et al. (1972) desenvolveram um algoritmo deste tipo para resolver o PPM. Sejam:

- Vp o conjunto de p pontos {v1,...,vp} no grafo G
- d(Vp,vj) = min {d(v1,vj),d(v2,vj),...,d(vp,vj) }
- d(vi,vj) o tamanho do menor caminho entre vi e vj, vi ∈ Vp

- a matriz de distâncias D (n X n)
- o conjunto VOp é uma p-mediana de G se, para todo Vkp em G:

$$\sum_{j=1}^{n} d(VOp,vj) \leq \sum_{j=1}^{n} d(Vkp,vj)$$

- Dij = d(vi,vj)
- Para o conjunto de p pontos VOp:

$$\sum_{j=1}^{n} d(VOp,vj) = \sum_{j=1}^{n} (\min Dij) = S$$

Eles observaram que (1) na soma S existe somente um elemento de cada coluna de D, (2) todos os termos em S estão localizados em somente p linhas de D, e (3) a soma S consiste de pelo menos p termos nulos que correspondem aos Dii na diagonal principal. O algoritmo apresentado por eles baseia-se principalmente na observação (2) e objetiva escolher n-p vértices que não pertençam a VOp. A regra de partição adotada é a seguinte: tira-se primeiro um vértice do conjunto e depois 2,3,4,...,n-p. Para calcular o limitante inferior da soma S, assume-se que r vértices vK1,vK2,...,vKr foram tirados do conjunto ($1 \leq r \leq n-p$). Então tem-se n-r vértices dos quais p vértices devem ser escolhidos. Conseqüentemente, tem-se dois conjuntos de vértices e os correspondentes conjuntos de índices:

Vr = {vK1,vK2,...,vKr} e Vn-r = {vJ1,vJ2,...,vJ(n-r)}
K = {K1,K2,K3,...,Kr} e J = {J1,J2,...,J(n-r)}

Para as colunas da matriz D, que correspondem ao conjunto K, calcula-se para a coluna $k \in K$:

sk = MIN Dik para $i \in J$

E para outra coluna $j \in J$:

$sJ = MIN\ Dij \qquad$ para $i \in J,\ i \neq j$

O limite inferior da soma S é:

$LB(K) = SK + SJ$, onde:

$$Sk = \sum_{k \in K} sk \qquad e$$

SJ é a soma dos (n-p-r) menores sj ($j \in J$).

O algoritmo possui os seguintes passos:

PASSO 1: Seja r = 1, S = ∞. Gera-se o conjunto de índices K = {1},{2},...,{n}. Segue-se para o PASSO 3.

PASSO 2: Seja K' = {K1,K2,...,Kr}. Seja r = r + 1. Inativa-se o limitante inferior LB(K') e gera-se um novo conjunto de índices K onde todos os índices do conjunto consistem dos índices de K' e de um novo índice.

PASSO 3: Calcula-se LB(K) para todos os índices.

PASSO 4: Se r < n - p escolhe-se o menor LB(K), chama-se este conjunto de K'e segue-se para o PASSO 2.

PASSO 5: Se r = n - p tem-se uma solução viável para o problema. Seja LB(K') o menor LB(K) do novo conjunto de índices. Se LB(K') < S, então faz-se S = LB(K'), colocam-se as p-medianas correspondentes na memória e destiva-se o limitante inferior cujo LB(K) $\geq$ S. Se após isto não existirem mais conjuntos ativos, o algoritmo termina e a soma mínima é S. Caso contrário, escolhe-se o menor LB(K) do conjunto de limitantes inferiores ativos, chama-se este conjunto de K' e vai-se para o PASSO 2.

Seguem os cálculos indicados nos passos do algoritmo. A matriz de distâncias é a definida em (2.9).

r = 1 n - r = 5 n - r - p = 3
V1 = {v1} e V5 = {v2, v3, v4, v5, v6}
LB(1) = 64 + 5 + 6 + 5 = 80
LB(2) = 21 + 51 + 6 + 5 = 83
LB(3) = 5 + 21 + 21 + 38 = 85
LB(4) = 21 + 5 + 6 + 5 = 37
LB(5) = 6 + 5 + 5 + 21 = 37
LB(6) = 5 + 6 + 6 + 21 = 38

Como r < n - p , continua-se. O menor LB(K) é LB(4) = 37 = LB(5)
K' = {4} r = 2
K = {4, 1} LB(K) = 21 + 5 + 5 = 31
n - r - p = 2
K = {4, 2} LB(K) = 21 + 5 + 5 = 31
K = {4, 3} LB(K) = 5 + 38 + 38 = 81
K = {4, 5} LB(K) = 6 + 5 + 5 = 16
K = {4, 6} LB(K) = 5 + 6 + 6 = 17
r = 2 n - p = 4 e K' = {4,5}

E assim por diante, são calculados os LB(K) até que r = n - p ...

Jarvinen et al. (1972) fornecem também uma heurística para obter uma solução inicial para o método de substituição proposto

por Teitz e Bart (1968). Eles afirmaram que este procedimento melhorou a execução do método de substituição. Segundo Handler e Mirchandani (1979) isto não é surpreendente, já que, a combinação de heurísticas é eficiente na resolução do PPM. Os procedimentos heurísticos são igualmente importantes quando usados em conjunto com alguns algoritmos exatos, pois além de fornecerem soluções iniciais viáveis boas, uma solução heurística também fornece um limitante superior do valor ótimo da função objetivo, ajudando na convergência mais rápida dos algoritmos exatos.

Revelle e Swain (1970) modificando a formulação do problema de localização de armazéns (PLA), apresentaram uma formulação de programação inteira para resolução do PPM, a mesma adotada neste trabalho (descrita na Seção 2.1).

Os algoritmos descritos a seguir são baseados em uma formulação dual do problema de programação matemática (P). Relaxação lagrangeana é uma técnica, baseada no dual, que tem provado sucesso na resolução de problemas de programação inteira. Tendo em vista que esta técnica é amplamente empregada, optou-se por fazer uma breve exposição teórica a respeito.

Muitos pesquisadores consideram a Relaxação Lagrangeana vantajosa para resolução de problemas com estruturas especiais, já que, relaxar um conjunto de restrições pode produzir um problema lagrangeano de fácil resolução, cujo valor ótimo é um limitante inferior (para o problema de minimização) do valor ótimo do problema original.

Segundo Fisher (1981) o nascimento da relaxação lagrangeana, como ela existe hoje, ocorreu em 1970 quando Held e Karp utilizaram um procedimento de relaxação para a solução aproximada de um problema de programação linear relacionado ao problema do caixeiro-viajante. Logo após o método foi aplicado em problemas de "scheduling" e problemas de programação inteira em geral.

Esta técnica foi amplamente discutida por Geoffrion (1974) e Held et al. (1974).

Surgem três questões quando se usa a abordagem lagrangeana: (1) quais restrições devem ser relaxadas, (2) como calcular bons multiplicadores e (3) como deduzir uma solução viável para o problema original dada a solução do problema relaxado.

Para Fisher (1985), a relaxação deve ser feita de modo a facilitar o problema e, ele sugere que, para calcular bons multiplicadores, existe um procedimento de propósito geral chamado método de subgradientes (uma variação do método de gradientes) que vem sendo utilizado em grande escala nos últimos anos e com bastante sucesso. Uma primeira abordagem deste método foi usada por Poljak (1967). Subseqüentemente, Held et al. (1974) apresentaram a formulação teórica do método e resultados computacionais aplicados aos problemas de atribuição e caixeiro-viajante. Maiores detalhes serão apresentados a seguir, no decorrer da apresentação do algoritmo.

Segundo Handler e Merchandani (1979), primeiro Diehr (1972) e mais tarde Narula et al. (1977) e Cornuejols et al. (1977), aplicaram especificamente esta técnica (relaxação lagrangeana em conjunto com um método de otimização por subgradientes) para resolver o PPM.

A seguir, descrevem-se os passos na resolução do problema (P) descrito na Seção 2.1, relaxando a restrição (2.2):

Para qualquer vetor $\lambda = (\lambda 1, \lambda 2, \ldots, \lambda n) \geq 0$, formar o seguinte problema lagrangeano (PL):

$$Zd(\lambda) = MIN \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} dij\ xij \right) - \sum_{j=1}^{n} \lambda j \left(\sum_{i=1}^{n} xij - 1 \right) \right) \qquad (2.12)$$

Sujeito à: $xii \leq xij$ (2.13)

$$\sum_{i=1}^{n} xii = p \qquad (2.14)$$

$$xij \in \{0,1\}, \quad i,j = 1,2,\ldots,n \qquad (2.15)$$

cuja função objetivo pode ser reescrita como:

$$Zd(\lambda) = MIN \sum_{j=1}^{n} \left(\sum_{i=1}^{n} (dij - \lambda j)\ xij \right) + \sum_{j=1}^{n} \lambda j \qquad (2.16)$$

Tal problema para um dado λ pode ser resolvido por inspeção, ignorando momentaneamente a restrição (2.14) e decompondo-o sobre o índice i, obtendo n subproblemas da seguinte forma:

$$\text{Minimizar} \sum_{j=1}^{n} (dij - \lambda j)\ xij$$

Sujeito à $xij \leq xii \quad i,j=1,2,\ldots,n$

$xij \in \{0,1\} \quad i,j=1,2,\ldots,n$

Cada subproblema é resolvido por inspeção. Seja:

$$\alpha i \ \phi = \sum_{j=1}^{n} [MIN(0, dij - \lambda j)]$$

e seja I o conjunto de índices dos p menores αi's (aqui a restrição (2.14) é tratada implicitamente). Então, uma solução $\{ xij^* \}$ para o problema lagrangeano é:

$$xii^* = \begin{cases} 1 \text{ se } i \in I \\ 0 \quad \text{caso contrário} \end{cases}$$

e para todo $i \neq j$:

$$xij^* = \begin{cases} 1 \text{ se } i \in I \text{ e } dij - \lambda j \quad 0 \\ 0, \text{ caso contrário} \end{cases}$$

A solução $\{xij^*\}$ não é necessariamente viável para o PPM original, uma vez que a condição (2.2) pode não estar sendo atendida. Todavia, o conjunto I identifica o conjunto de p nós que pode sempre oferecer uma possível solução viável e logo um limitante superior. Se $\lambda \geq 0$, então $Zd(\lambda)$ é menor ou igual a Z (o valor ótimo da função objetivo do PPM original).

Assim o problema dual (D) é encontrar o melhor limitante inferior para Z, ou seja:

(D) Maximizar $Zd(\lambda)$ $(\lambda \geq 0)$ (2.17)

Para obter um bom vetor λ, sugere-se o uso de um método de subgradientes que é uma técnica de atualização progressiva dos multiplicadores de Lagrange, de maneira sistemática, com o objetivo de maximizar o valor do problema dual lagrangeano.

Assume-se que o problema lagrangeano tenha sido resolvido para um dado λ e seja:

$$Sj = \sum_{i=1}^{n} xij^* - 1$$

Logo, Sj indica o número de medidas em excesso para as quais o vértice vj está alocado na solução dual $\{xij^*\}$. Observa-se que $S=(S1,S2,\ldots, Sn)$ é um subgradiente de Zd em λ e pode ser usado como direção a fim de alterá-lo. Assim:

Se $Sj < 0$, λj é diminuído
Se $Sj > 0$, λj é aumentado
Se $Sj = 0$, λj permanece inalterado

Para se determinar o "tamanho do passo" (tk), sugere-se as regras dadas por Held et al. (1974):

Seja a seqüência: $\lambda jk+1 = \lambda jk + tkSjk$

Pode-se assegurar a convergência de $Zd(\lambda)$ ao seu valor máximo sob as seguintes condições (Poljak, 1967):

$$\text{Se } k \dashrightarrow \infty \text{ , } tk \dashrightarrow 0 \text{ e } \sum_{i=1}^{n} ti \dashrightarrow \infty$$

O "tamanho do passo" usado mais comumente é (Held e Karp, 1970):

$$tk = \frac{\pi k \ (Z^* - Zd(\lambda k))}{\|Sk\|^2}$$

onde πk é um escalar que satisfaz a $0 \leq \pi k \leq 2$, Z^* o valor da função objetivo em seu melhor valor viável para (P) e $\|Sk\|^2$ é qualquer norma (neste caso, norma Euclideana) de S.

Freqüentemente a seqüência πk é determinada começando com $\pi 0 = 2$ e reduzindo πk por um fator 2 sempre que $Zd(\lambda)$ falhar em crescer em um número específico de iterações. O valor Z^*, inicialmente, pode ser calculado usando qualquer heurística para o PPM (Teitz e Bart, por exemplo) e depois atualizado usando as soluções que são obtidas nas iterações em que a solução do problema lagrangeano devolve uma solução viável para o problema original (P). A menos que se obtenha um λk para o qual $Zd(\lambda) = Z^*$, não existe maneira de provar a otimalidade do método dos subgradientes. Para resolver esta dificuldade, o método é usualmente encerrado depois de atingir um limite específico de iterações.

Narula et al. (1977) utilizam a metodologia acima definindo o problema (P) como definido na Seção 2.1 (equações (2.1)-(2.5)), e os problemas (PL - equações (2.12)-(2.15)) e (D - equação (2.17)).

O algoritmo (chamado Algoritmo 1, para futuras referências) para resolver o problema (D), ou seja, maximizar $Zd(\lambda)$ pode ser estabelecido como segue:

PASSO 1: Escolher um $\lambda 0$ inicial. Uma boa escolha é $\lambda j0 = MIN\{dij\}$, para $i \neq j$. Faz-se k=0.

PASSO 2: Para o λk escolhido, o valor de $Zd(\lambda k)$ é obtido como se segue: para cada i, calcula-se

$$\alpha i0 = \sum_{j=1}^{n} (MIN(0, dij - \lambda j))$$

Selecionam-se os p menores αi e seja I o conjunto de tais índices. Calcula-se

$$Zd(\lambda k) = \sum_{i \in I} \alpha i + \sum_{j=1}^{n} \lambda jk$$

Para obter o valor do correspondente Z*, calcula-se para cada j, Djk = MIN {dij}, $i \in I$. A soma dos n-p maiores Djk dá o valor de Z* .

PASSO 3: A j-ésima componente do subgradiente de $Zd(\lambda k)$ em λk é

$$Sj = \sum_{i=1}^{n} xij^* - 1, \text{ onde}$$

$$xij^* = \begin{cases} 1 \text{ se } i \in I \text{ e } (dij - \lambda j) < 0 \\ 0, \text{ caso contrário.} \end{cases}$$

Termina a busca se todo Sj = 0 ou o valor de $Zd(\lambda k)$ igualar ao valor de Z. Caso contrário, calcula-se o "tamanho do passo" tk. Faz-se $\lambda k+1 = \lambda k + tkSk$ e k=k+1. Vai-se para o PASSO 2.

Nesse trabalho (Narula et al., 1977), os autores apresentam resultados comparativos com outros algoritmos para problemas com n=10 (p=2, 4, 6, 8), n=20 (p=2, 4, ..., 18), n=30 (p=2, 4, ..., 28). A técnica utilizada apresentou excelentes resultados relativos.

Christofides e Beasley (1982) também utilizaram relaxação lagrangeana, tentando dois limites para o PPM. Para um destes limites a relaxação foi relativa às mesmas restrições consideradas por Narula et al. (1977), só que, neste caso foram incluídos testes de penalidades baseados nos limites obtidos. No segundo limite estudado a relaxação foi relativa às restrições:

$$\sum_{j=1}^{n} aij\ xij \leq ni\ xii \quad i,j=1,\ldots,n \quad i \neq j$$

que substituiram as restrições $xij \leq xii$ da formulação anterior. Neste caso considerou-se $aij \geq 0$ para todo i e j e $aij = 0$ se e somente se j não pode ser alocado a i ($dij = \infty$).

Também neste caso foram incluídos testes de penalidade sobre o limite. Um procedimento de busca em árvore é descrito e são apresentados os resultados computacionais para problemas com número arbitrário de medianas num total de até 200 nós.

O trabalho de Christofides e Beasley (1982) mostrou que a relaxação conforme considerada por Narula et al. (1977) é mais eficiente. Os testes de penalidade incluídos serviram para a redução do problema, isto é, fixando algumas variáveis xij no seu valor ótimo zero ou um reduzindo a dimensão do problema.

Em uma publicação posterior, Beasley (1985) mostrou ser possível melhorar seu algoritmo, resolvendo problemas com até 900 nós, com o uso de um super-computador com capacidade de processamento vetorial.

Raggi e Galvão (1982) trataram o PPM com custos fixos, isto é, um problema de localização de recursos em redes no qual os custos fixos de instalação variam ao longo da rede. Eles apresentaram um algoritmo com base na relaxação lagrangeana da formulação de programação zero-um do problema. O uso deste método para problemas de custo fixo é uma extensão dos métodos usados por Narula et al. (1977) e Christofides e Beasley (1982). O algoritmo foi testado em vários problemas gerados aleatoriamente, relativos a redes de até 100 vértices. Para todos os problemas testados a solução ótima foi obtida.

Olivo e Senne (1988) apresentaram a implementação, em micro-computador do tipo IBM-PC, de um método de subgradientes associado à relaxação lagrangeana e concluíram que este pode ser considerado uma excelente heurística na obtenção de bons limitantes. Eles sugerem o uso destes limitantes em um algoritmo do tipo "branch and bound" para solucionar, de modo exato, problemas de grande porte.

Neste trabalho segue-se a sugestão de Olivo e Senne (1988): aproveita-se a abordagem de Christofides e Beasley (1982), incorporando-se a busca em árvore e implementando-se o algoritmo em micro-computador do tipo IBM-PC.

CAPÍTULO 3

ALGORITMO PROPOSTO

3.1 - INTRODUÇÃO

No capítulo anterior, descreveu-se o problema das p-medianas e suas aplicações. Foram destacados alguns dos algoritmos mais importantes na resolução do PPM. Além disso, foram fornecidos alguns resultados teóricos importantes (relaxação lagrangeana e otimização por subgradientes), que serão úteis na formulação do algoritmo utilizado neste trabalho.

Neste trabalho enfoca-se a resolução do PPM de uma forma clássica: aplica-se uma relaxação lagrangeana ao problema de modo que permita a resolução analítica do problema lagrangeano e, em seguida, busca-se o ótimo da função dual usando um método de subgradientes. Se uma solução ótima primal não for encontrada em um determinado número de iterações lança-se mão de um esquema de "branch and bound" (procedimento de busca em árvore).

Neste capítulo, inicialmente, é apresentado o algoritmo enfocando a relaxação lagrangeana e um método de subgradientes. A seguir, após a apresentação de uma visão geral do método "branch and bound", descreve-se a sua inclusão no algoritmo.

3.2 - ALGORITMO DE SUBGRADIENTES

Como já foi descrito anteriormente, o PPM pode ser formulado como um problema de programação linear inteira zero-um (problema (P) - Seção 2.1). A maior dificuldade encontrada na resolução de (P) reside na determinação do valor das variáveis xii, que define a localização das medianas na rede. Supondo-se que xii* representem os valores de xii na solução ótima de (P), a determinação das demais variáveis xij, $i \neq j$ - chamada de problema da alocação dos nós às medianas - pode ser realizada sem dificuldade a partir de xii*.

Se Q = {i: xii* = 1} é o conjunto das medianas da solução ótima, então os xij* estão definidos, para cada j ∈ V-Q, pela seguinte regra:

$$xij^* = \begin{cases} 1 \text{ para um nó } i \quad Q \text{ tal que } dij = MIN\ dik\ (k \in Q) \\ \\ 0 \quad \text{caso contrário.} \end{cases}$$

O problema lagrangeano é o (PL) descrito no capítulo anterior, equações (2.12)-(2.15). A constante αi, também definida anteriormente, pode ser considerada como o "custo" de localizar uma mediana no nó i. Assim, a resolução do problema lagrangeano reduz-se a determinar o conjunto I dos nós correspondentes aos p menores αi, $i \in V$ e fazer xii = 1 se $i \in I$ e xii = 0, caso contrário.

O problema dual correspondente a (P) é o problema (D) definido pela equação (2.17). Para resolver (D) utiliza-se um método de subgradientes, já descrito anteriormente.

A seguir, baseado no Algoritmo 1 descrito no Capítulo 2, formaliza-se o algoritmo de subgradientes usado. Determina-se um valor inicial para LS (o limitante superior no problema). Isto pode ser feito usando qualquer heurística para o PPM (Teitz e Bart, 1968 por exemplo). Neste trabalho, por economia de tempo e recursos computacionais, optou-se por inicializar LS com um valor grande (maior número inteiro permitido pela linguagem de programação utilizada). Logo na primeira iteração este valor é atualizado pelo valor da solução viável do primal. Seja Zd_max o valor do limitante inferior máximo. Seja Zd_art uma heurística usada quando não há melhora no LS, naquela iteração, servindo, assim, para acelerar a convergência do algoritmo. Este valor não pode superar LS e quando isso acontece Zd_art é feito igual a Zd_max.

ALGORITMO DE SUBGRADIENTES:

Dados λ e LS.

PASSO 1: Faz-se Zd_art $\leftarrow$ 0 e Zd $\leftarrow$ 0.

PASSO 2: Enquanto (a) $Sj \neq 0$, para algum j, ou

(b) LS - Zd_max ≥ 0.05, ou

(c) $\pi > 0.005$,

Resolvem-se os PASSOS 3 a 6.

PASSO 3: Resolve-se (PL) obtendo:

$$xij^*,$$

$$Zd(\lambda) \leftarrow \sum_{i \in I} \alpha i + \sum_{j=1}^{n} \lambda j$$

$$Z^* \leftarrow \sum_{j=1}^{n} (\text{MIN}_{i \in I} \; dij)$$

PASSO 4: Calculam-se os subgradientes:

$$Sj \leftarrow \sum_{i=1}^{n} xij^* - 1, \quad j=1,\ldots,n$$

PASSO 5: Atualizam-se os limitantes:

Se $Zd(\lambda) >$ Zd_max, então Zd_max $\leftarrow Zd(\lambda)$;

Se $Z^* <$ LS, então LS $\leftarrow Z^*$ e Zd_art $\leftarrow$ Zd_max

Caso contrário, faz-se:

$$Zd\_art \leftarrow Zd\_art + 0.2(LS - Zd\_art);$$

Se $Zd\_art > LS$, então $Zd\_art \leftarrow Zd\_max$;

PASSO 6: Determina-se o novo "tamanho do passo":

$$t \leftarrow \frac{\pi\ (LS - Zd - art)}{\|S\|^2}$$

e atualizam-se os multiplicadores de Lagrange:

$$\lambda j \leftarrow MAX\ \{0,\ \lambda j + tSj\}\ ,\quad j = 1,\ldots,n$$

Para escolher o valor de π, segue-se a abordagem de Held et al. (1974) fazendo $\pi = 2$ para $2n$ iterações. Depois π é dividido ao meio a cada 5 iterações seguidas em que não houver melhora no Zd_max.

Como se pode observar executa-se este algoritmo de subgradientes até que uma ou mais das condições de saída abaixo sejam satisfeitas:

a- Todos os subgradientes sejam nulos
b- A solução viável mínima (LS) e o limitante inferior máximo Zd_max coincidem (dentro de uma precisão preestabelecida, 5% neste caso), ou seja, LS corresponde a uma solução ótima.
c- π alcance o valor 0.005.

Estes valores, 5% e 0.005, são fixados empiricamente.

Segundo Narula et al. (1977), caso o algoritmo não forneça uma solução ótima para (D), o método de "branch and bound" pode ser usado para encontrar uma solução ótima inteira. O limitante de cada nó seria obtido maximizando $Zd(\lambda)$ usando o algoritmo proposto, com um conjunto de variáveis fixadas em zero ou um.

Beasley (1985) segue este mesmo procedimento, implementando-o e refinando-o. Basicamente as melhoras no algoritmo envolvem:

(1) O valor inicial de π e a regra para divisão de seu valor: $\pi = 2$ inicialmente e é dividido ao meio se Zd_max parar de crescer em trinta iterações consecutivas do subgradiente.

(2) Dentro da árvore de busca, inicialmente $\pi = 1$ e é dividido ao meio a cada cinco iterações seguidas em que não houver melhora de Zd_max.

(3) Na resolução de grandes problemas, Beasley (1985) concluiu ser vantajoso abandonar a árvore de busca depois de um certo estágio e recomeçar o problema. Se em qualquer estágio da árvore, encontra-se uma solução viável melhor, ou seja LS (LS_inicial e que ainda não é ótima pois LS $\neq$ Zd_max, então recomeça-se o problema da "linha de saída", ou seja, com um novo nó inicial da árvore.

Além disso, Beasley (1985) sugeriu uma heurística para geração do LS inicial. Ele concluiu que estas mudanças geraram um "gap" de dualidade menor no nó inicial da árvore em menos iterações do subgradiente, além de gerarem (quase sempre) menos nós na árvore de busca.

3.3 - ALGORITMO DE BRANCH AND BOUND

Coloca-se a seguir o procedimento geral para se resolver problemas de programação inteira pela técnica "branch and bound". Esta técnica pode ser considerada como um procedimento de enumeração implícita, onde apenas uma fração das soluções viáveis é examinada.

A idéia básica desta técnica é a seguinte. Suponha-se o problema de minimização de uma função objetivo cujo limitante superior do valor ótimo esteja disponível. O primeiro passo é a partição do conjunto de todas as soluções viáveis em vários subconjuntos, e, para cada um, obtem-se um limitante inferior do valor da função objetivo dentro do subconjunto. Aqueles subconjuntos, cujo limitante inferior exceder o limitante superior corrente da função objetivo, são excluídos de futuras considerações. Um subconjunto que é excluído por isso ou por outras razões legítimas é chamado "esgotado". Em um dos subconjuntos restantes, naquele com o menor limitante inferior, é feita uma partição em vários subconjuntos. Seus limitantes inferiores são obtidos e usados como anteriormente para excluir alguns dos subconjuntos de consideraçoes futuras. De todos os subconjuntos que ficarem, outro é selecionado para a próxima partição e assim por diante. Este processo é repetido até que uma solução viável seja encontrada tal que o valor correspondente da função objetivo não seja maior que o limitante inferior de qualquer subconjunto. Esta solução viável deve ser ótima já que nenhum dos subconjuntos apresentou uma melhor.

É importante que se defina uma regra de ramificação para selecionar um dos subconjuntos que ficarem (aqueles nem esgotados nem divididos). Cada novo subconjunto é excluido de considerações futuras (é esgotado) se: (1) o limitante inferior calculado exceder o limitante superior em questão, (2) o subconjunto não possui soluções viáveis e (3) a melhor solução viável no subconjunto foi identificada (se esta solução não exceder o limitante superior, armazena-se o seu valor como solução incumbente), e reaplica-se o teste de esgotamento

(1) para todos os subconjuntos restantes. Deve-se parar quando não existirem subconjuntos inesgotados, e então, a solução incumbente é ótima.

Para o PPM, onde todas as variáveis (xij) estão restritas a dois valores (0 ou 1), constroi-se uma árvore binária, passo a passo, cujo percurso é feito em profundidade. Fixam-se nós como medianas e não medianas baseado na seguinte estratégia para ramificação: fixa-se a variável xii para i correspondente a: α i = MIN (α j) para j $\in$ I. Em qualquer momento da árvore, uma variável é selecionada e dois subproblemas são gerados - um no qual aquela variável (xii) é igualada a zero, e outro no qual a variável é igualada a um. Como o percurso da árvore é feito em profundidade, primeiramente examinam-se os nós, fixando-os em um (nós da esquerda). Os testes de esgotamento são feitos. Sempre que um nó é esgotado volta-se para o anterior e ramifica-se considerando aquele último como não mediana (xii = 0, nós da direita).

O limitante considerado (Zd_max) é o obtido pela relaxação lagrangeana. O procedimento dos subgradientes é executado até trinta iterações para cada nó da árvore ou até que uma de suas condições de saída ocorra. Adotam-se as sugestões (2) e (3) de Beasley (1985).

Tem-se então o seguinte:

- Após executar inicialmente o Algoritmo de Subgradientes, se Sj = 0, para todos os j, ou Zd_max - LS < 0.05, então a solução ótima é encontrada para o primal e dual. Do contrário, será preciso construir a árvore de busca.

- Dentro da árvore, se Sj = 0, para todos os j, ou Zd_max > LS, então deve-se esgotar o nó atual e, como a busca é em profundidade não se pode concluir otimalidade no primal. Deve-se, portanto, continuar a expandir a árvore até esgotá-la completamente.

Segue a descrição do Algoritmo de "Branch and Bound" utilizado.

ALGORITMO DE "BRANCH AND BOUND":

Dada a raiz da árvore com Zd_max e LS_inicial obtidos do Algoritmo de Subgradientes:

PASSO 1: Seja CONTE ← 0; no_arvore ← raiz;

PASSO 2: Enquanto (a) todos os subgradientes não são nulos,
(b) LS - Zd_max > 0.05 e
(c) CONTE > p

Faz-se:

2.1) no_arvore ← filho_esquerdo(no_arvore);

2.2) Executa-se o Algoritmo de Subgradientes (até 30 iterações);

2.3) Se a solução ótima não foi encontrada, mas LS (LS_inicial, então recomeça-se a construção da árvore com uma nova raiz (Zd_max e LS), a partir do PASSO 1.

PASSO 3: Esgota-se no_arvore.

PASSO 4: Se no_arvore = raiz, termina-se. Caso contrário, faz-se:
no_pai ← pai(no_arvore) e CONTE ← CONTE - 1;

Se no_arvore = filho_direito(no_pai), então faz-se
no_arvore ← no_pai e volta-se ao PASSO 4.

Caso contrário, faz-se: no_arvore ← filho_direito(no_arvore)
e volta-se ao PASSO 2.

Segue um exemplo (Figura 3.1) com n = 20 e p = 3. No início, até começar a busca na árvore, houve 203 iterações e obteve-se LS = 293 e Zd_max = 274,02. Na Figura 3.1 o número x (-x) dentro do quadrado representando o nó, implica que o vértice x foi escolhido para ser (não ser) uma mediana. As letras próximas aos nós indicam a ordem do percorrimento da árvore (A primeiro, B segundo, etc). Como não houve melhora no LS (permaneceu 293), a árvore não foi recomeçada. A melhor solução encontrada foi a da raiz (nós 13,1,3).

Segue um outro exemplo (Figura 3.2) com n = 20 e p = 3, em que houve melhora do LS. No início, ate começar a busca na árvore, ocorreram 136 iterações e obteve-se LS = 353 e Zd_max = 312.95. Após 140 iterações, quando fixou-se o nó 2 como não mediana, houve uma melhora no LS. Recomeçou-se a árvore de busca, a partir deste ponto. Após esgotada toda a árvore, a melhor solução encontrada foi a da segunda raiz (nós 17,2,16) onde LS = 336.

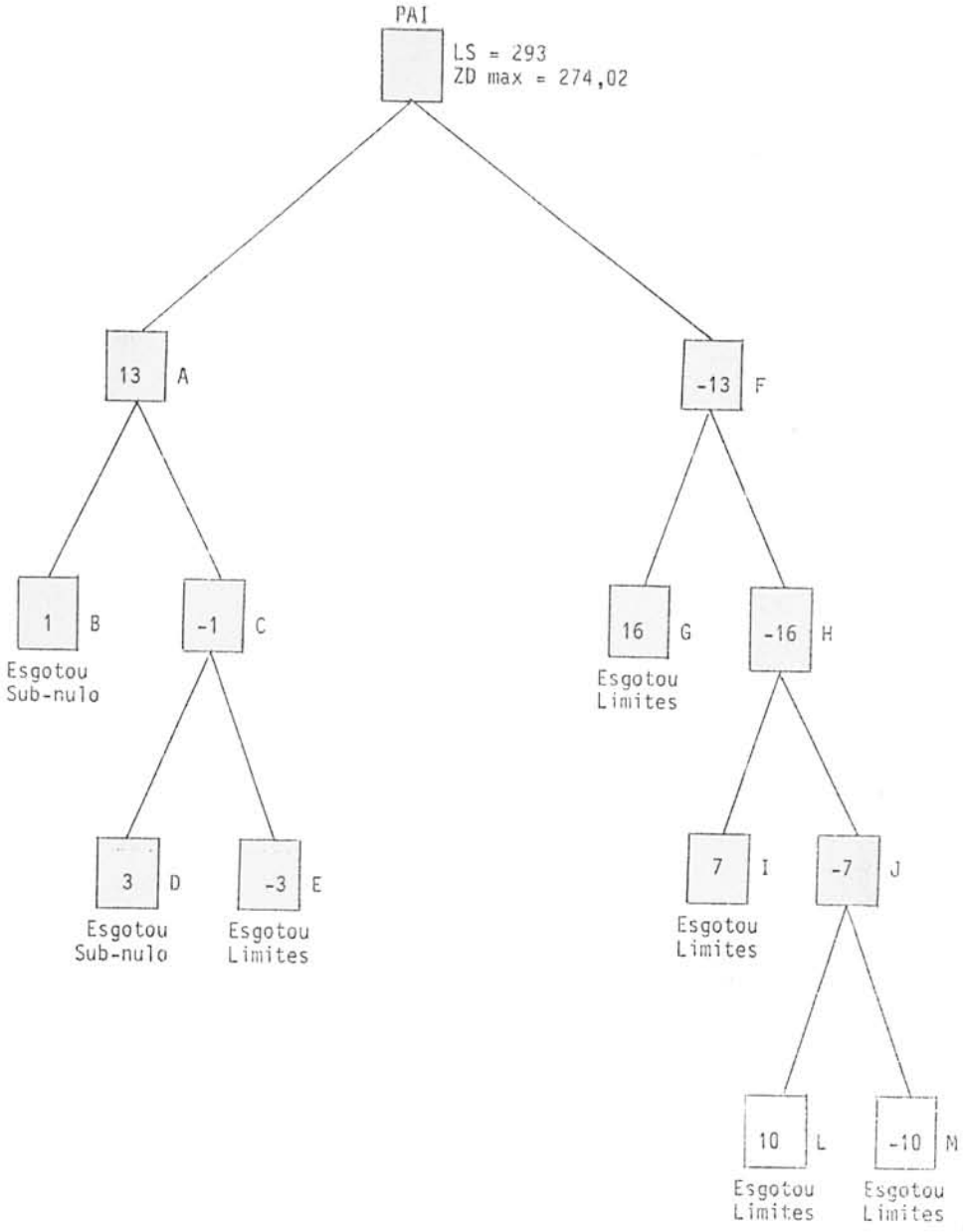


Fig. 3.1 - Exemplo com n = 20 e p = 3 (sem recomeço).

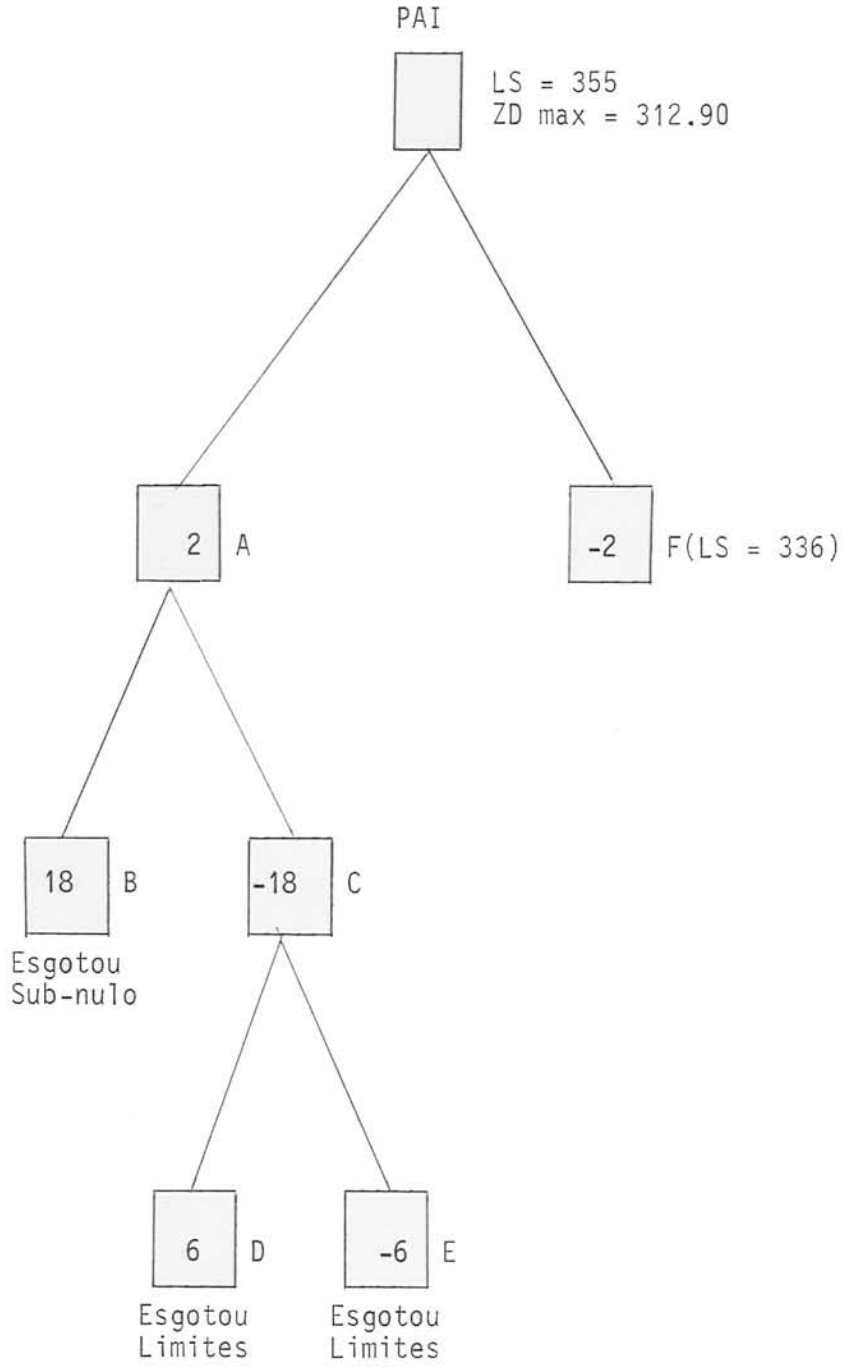


Fig. 3.2 - Exemplo com n = 30 e p = 3 (com um recomeço).

(continua)

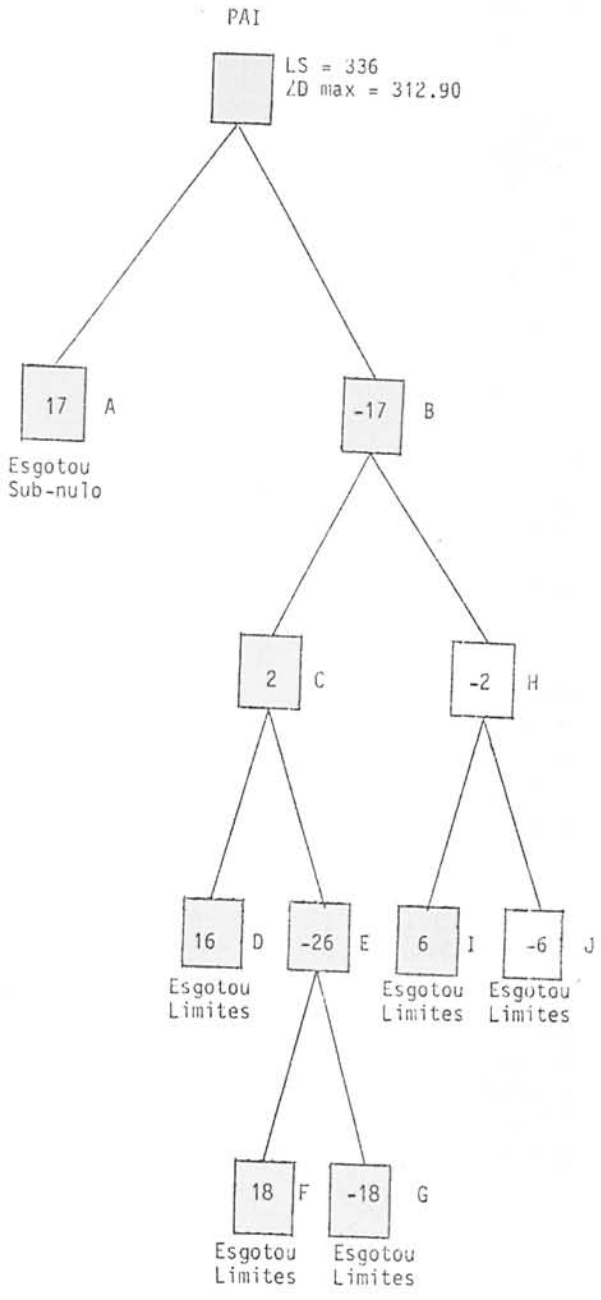


Fig. 3.2 - Conclusão.

CAPÍTULO 4

IMPLEMENTAÇÃO E TESTES

4.1 - INTRODUÇÃO

O algoritmo proposto foi implementado em TURBO-PASCAL (versão 3.0) usando um microcomputador tipo IBM-PC-AT. O programa pode ainda ser facilmente adaptado para uso em computadores de grande porte. A linguagem escolhida para programação, Pascal, é adequada quando se quer usar estruturas especiais de dados como listas e árvores (caso deste trabalho).

Uma consideração importante em qualquer implementação é sua eficiência cuja avaliação está relacionada ao tempo de execução do código final e a quantidade de espaço (memória) envolvido na execução. Estes dois fatores, espaço e tempo, encontram-se relacionados (nos sistemas que não implementam multiprocessamento) de tal forma que a redução de um deles implica na ampliação do outro e vice-versa.

Como um dos objetivos deste trabalho é a implementação de um algoritmo para o PPM em micro-computadores, a eficiência deste programa está basicamente relacionada com o espaco. Procurou-se economizar ao máximo o número de variáveis e matrizes (principalmente os tamanhos destas) de modo que o programa pudesse ser executado para um número total de nós (n) maior possível. Assim, obteve-se um programa que consegue suportar até 205 nós.

Um exemplo de otimização de espaço é o armazenamento da matriz de distâncias do problema. Como ela é simétrica, pode-se armazenar apenas a parte acima da diagonal principal em um vetor (d(k)), onde k é a posição que cada elemento desta parte da matriz ocupa em d. Cria-se um outro vetor (e) formado por ponteiros (tantos quantas forem as linhas da matriz de distâncias, menos um) que apontam, de acordo com sua ordem, para o início de cada linha (relativa à sua ordem). Por exemplo, e(1) aponta para o primeiro elemento em d da primeira linha da matriz de distâncias, e(2) aponta para o primeiro elemento em d da segunda linha da matriz de distâncias e assim sucessivamente. Para determinar o elemento que ocupa a posição i,j da matriz de distâncias, utiliza-se a seguinte expressão:

SE i (j ENTÃO k(i,j) = e(i) + j - i - 1 e dist(i,j) = d(k(i,j))
CASO CONTRÁRIO
  SE i) j ENTÃO k(i,j) = e(j) + i - j - 1 e dist(i,j) = d(k(i,j))
    CASO CONTRÁRIO dist(i,j) = 0
onde: e(i) é a posição no vetor d onde começa a linha i.

Um exemplo é ilustrado a seguir. Seja a matriz de distâncias (para n=4):

$$dist = \begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix}$$

O vetor d e os respectivos k serão:

d: (a12 a13 a14 a23 a24 a34)
k: 1 2 3 4 5 6
e(1) aponta para k=1
e(2) aponta para k=4
e(3) aponta para k=6

Para saber o elemento que ocupa a posição i=3 e j=2 executa-se:

k(3,2) = e(2) + 3 - 2 - 1 = 4 + 3 - 2 - 1 = 4
logo, d32 = d(4) = a23

A listagem do programa encontra-se no Apêndice A deste trabalho.

4.2 - DESCRIÇÃO DOS TESTES

O algoritmo foi testado com problemas gerados aleatoriamente com dij no intervalo [5,100]. O tamanho das redes (n) variou de 15 a 200 e o número de medianas de 5 a n/3 (a literatura indica que para n/3 medianas o problema torna-se mais difícil). Devido ao grande número de testes e tempo de execução de cada um, fixou-se o número de iterações em 1000 (em alguns casos, 2000).

Foram realizados cinco tipos de testes e cada um deles possui uma entrada e seus objetivos específicos, a saber:

TESTE 1

ENTRADA - Para cada n = 25, 50, 75, 100 e 125 gerou-se a matriz de distâncias e realizaram-se os testes com número de medianas igual a 5, 10 e n/3.

OBJETIVOS - Tempo total de cada teste e comparação entre os resultados obtidos antes de entrar pela primeira vez na árvore e após sair do programa (solução ótima ou 1000 iterações).

TESTE 2

ENTRADA - Para n = 25 foram geradas cinco matrizes diferentes e para cada matriz realizaram-se os testes com número de medianas igual a 5, 8 e 10.

OBJETIVOS - Observação da variação dos resultados e comparação entre os tempos de execução.

TESTE 3

ENTRADA - Para n = 15 foram geradas cinco matrizes diferentes e para cada matriz realizaram-se os testes com número de medianas igual a 5. As iterações foram limitadas em 1000 e no último caso em 2000.

OBJETIVOS - Semelhante ao anterior, mas tentando comparar a solução encontrada antes do "branch and bound" e depois, quando a árvore fica completamente esgotada. O número de iterações foi aumentado com o objetivo de mostrar que, neste caso, a solução ótima foi encontrada.

TESTE 4

ENTRADA - Para n = 200 e número de medianas igual a 5.

OBJETIVOS - Teste da capacidade de processamento do programa, tempo de execução e diferença entre os resultados antes de entrar pela primeira vez na árvore e após sair do programa (solução ótima ou 1000 iterações).

TESTE 5

ENTRADA - Para n = 50 foram realizados quatro testes para p variando em torno de n/3 (p = 15, 16, 17, 18).

OBJETIVOS - Verificação da performance do programa para p = n/3.

4.3 - APRESENTAÇÃO DOS RESULTADOS

Os tempos indicados excluem o tempo necessário para gerar a matriz de distâncias ou lê-la, caso se considere uma já existente. Os resultados dos testes encontram-se nas Tabelas 4.1, 4.2, 4.3, 4.4 e 4.5 que contêm:

- Número de nós (n).
- Número de medianas (p).
- Limitante superior (LS) antes e depois de entrar no "branch and "bound.
- Limitante inferior (Zd_max).
- Tipo de solução que está classificada em: Ótima, Não Ótima e Incumbente. Dentro da Rotina de Subgradiente tem-se:
  - . Solução Ótima implicando em término do programa;
  - . Solução Não Ótima implicando em continuação do programa com a Rotina de Branch and Bound.

  Dentro da Rotina de Subgradiente com Branch and Bound tem-se:
  - . Solução Ótima implicando em término do programa;
  - . Solução Incumbente implicando em alteração (diminuição) do limitante superior (LS) proporcionando uma solução viável melhor;
  - . Solução Não Ótima quando não há alteração no LS.
- Número de iterações na Rotina de Subgradiente e número de iterações total (incluindo as duas rotinas).
- "Gap" de Dualidade inicial e final calculado pela equação:

$$\text{"gap"} = \frac{LS - Zd\_max}{LS} \times 100\%$$

- Tempo total (incluindo as duas rotinas).
- Número de recomeços da árvore de busca.

As Tabelas 4.1 e 4.2 apresentam os resultados do Teste 1 e 2, respectivamente. Pode-se notar uma variação enorme dos resultados. Para 25 e 5, por exemplo, em duas repetições ocorreu a

solução ótima no nó inicial e em três houve entrada no "branch and bound". Os resultados de tempo podem ter alguma correlação se considerar-se, apenas, quando o algoritmo entra na árvore e quando não entra, separadamente.

A Tabela 4.3 apresenta alguns casos, nos testes mostrados, em que o algoritmo entra no "branch and bound" e sai com uma solução ótima. Vale comparar os dois últimos exemplos, pois, quando o número de iterações foi aumentado para 2000 (último exemplo), o algoritmo convergiu para o ótimo. Portanto, nota-se que, se não houvesse limitação no número de iterações (penúltimo exemplo), a solução ótima seria encontrada (que poderia ser a própria solução incumbente).

A Tabela 4.4 mostra um caso de maior exigência do sistema onde n = 200. Como houve limitação de iterações em 1000 não houve convergência para o ótimo. Pode-se notar, no entanto, que a solução incumbente encontrada é melhor do que a solução no início da árvore.

A Tabela 4.5 apresenta o estudo de p = n/3 e sua vizinhança. Em nenhum dos casos o algoritmo convergiu para o ótimo. Como esse comportamento verificou-se para outros valores de p, os resultados deste teste não são conclusivos.

Observou-se, durante a execução dos testes, que à medida em que p cresce em relação a n, há uma diminuição da convergência de Zd_max em relação à LS.

Seguem as tabelas 4.1, 4.2, 4.3, 4.4 e 4.5.

TABELA 4.1

TESTE 1

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
25	5	232	141	N. OTIMA	91	39,22	228	1	INCUMBENT	1000	445	38,16
	10	141	95	OTIMA	31	32,62					13	
	8	169	113	OTIMA	23	33,14					10	
50	5	603	531	N. OTIMA	186	11,94	603		N.OTIMA	1000	1294	11,94
	10	368	289	N. OTIMA	264	21,47	368		N.OTIMA	1000	1330	21,47
	17	284	177	N. OTIMA	141	37,68	261	5	INCUMBENT	1000	1465	32,18
75	5	943	781	N. OTIMA	242	17,18	943		N OTIMA	1000	2627	17,18
	10	562	477	N. OTIMA	296	15,12	542	1	INCUMBENT	1000	2670	11,99
	25	333	259	N. OTIMA	191	22,22	333		N OTIMA	1000	3410	22,22

(continua)

Tabela 4.1 - Conclusão.

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
100	5	1388	1132	N. OTIMA	242	18,44	1388		N OTIMA	1000	4247	18,44
	10	877	699	N. OTIMA	241	20,30	857	1	INCUMBENT	1000	4262	18,44
	33	430	347	N. OTIMA	241	19,30	430		N OTIMA	1000	6173	19,30
125	5	1811	1462	N. OTIMA	358	19,24	1811		N OTIMA	1000	7057	19,24
	10	1156	890	N. OTIMA	292	23,01	1106	4	INCUMBENT	1000	6693	19,53
	42	482	415	N. OTIMA	291	13,90	482		N OTIMA	1000	10677	13,90

TABELA 4.2

TESTE 2

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
25	5	240	152	N. OTIMA	93	36,67	205	2	INCUMBENT	1000	446	25,85
	10	180	81	N. OTIMA	91	55,00	129	6	INCUMBENT	1000	466	37,21
	8	166	99	N. OTIMA	91	40,36	161	1	INCUMBENT	1000	474	38,51
25	5	306	258	OTIMA	139	15,65					121	
	10	214	112	OTIMA	34	47,66					36	
	8	223	134	OTIMA	44	39,91					42	
25	5	216	130	OTIMA	17	39,81					18	
	10	143	87	OTIMA	28	39,16					22	
	8	178	103	OTIMA	31	42,13					19	
25	5	251	213	N. OTIMA	104	15,14	247	1	INCUMBENT	1000	441	13,77
	10	191	98	N. OTIMA	91	48,69	164	2	INCUMBENT	1000	521	40,24
	8	228	120	N. OTIMA	91	47,37	192	4	INCUMBENT	1000	475	37,50
25	5	191	155	N. OTIMA	97	18,45	191		OTIMA	619	260	18,45
	10	146	87	OTIMA	40	40,41					23	
	8	170	101	N. OTIMA	91	40,49	164		INCUMBENT	1000	474	38,41

TABELA 4.3

TESTE 3

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
15	5	131	61	N. OTIMA	71	53,44	104	4	OTIMA	885	206	41,35
	5	144	71	N. OTIMA	71	50,69	124	2	OTIMA	250	56	42,74
	5	144	90	OTIMA	27	37,50					6	
	5	173	89	N. OTIMA	71	48,55	164	1	INCUMBENT	1000	239	45,73
	5	143	77	N. OTIMA	71	46,15	143		OTIMA	1298	323	46,15

TABELA 4.4

TESTE 4

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
200	5	3150	2329	N. OTIMA	441	26,06	3056	5	INCUMBENT	1000	16481	23,79

TABELA 4.5

TESTE 5

NUM. NOS	NUM. MED.	ROTINA DE SUBGRADIENTE					ROTINA DE SUBGRADIENTE COM BRANCH AND BOUND					
		LS	ZD MAX	SOLUCAO	N.ITER	GAP INICIAL (%)	LS	RECOMECOS	SOLUCAO	N.ITER	TEMPO (seg)	GAP FINAL (%)
50	15	303	189	N. OTIMA	141	37,62	293	1	INCUMBENT	1000	1424	35,49
	16	270	183	N. OTIMA	141	32,22	270		N.OTIMA	1000	1437	32,22
	17	284	177	N. OTIMA	141	37,68	261	5	INCUMBENT	1000	1465	32,18
	18	272	171	N. OTIMA	141	37,13	267	1	INCUMBENT	1000	1482	35,96

CAPÍTULO 5

COMENTÁRIOS FINAIS

O objetivo deste trabalho foi estudar o problema das p-medianas e os métodos (heurísticos e exatos) existentes para sua resolução. Desenvolveu-se um algoritmo para resolução exata do problema, utilizando uma formulação de programação inteira e um método de subgradientes associado à relaxação lagrangeana. Este método comprovadamente obtem bons limitantes que usados em um algoritmo do tipo "branch and bound" tornam possível solucionar de modo exato problemas de grande porte.

De acordo com os resultados computacionais apresentados pode-se concluir que, apesar de não se ter esgotado a árvore na maioria dos casos, houve uma redução nos "gaps" de dualidade quando o "branch and bound" foi executado. Então, mesmo sem garantir uma solucão ótima pode-se garantir, nestes casos, uma melhor solução. Em uma situação prática convem deixar o algoritmo executar até o fim, sem limitar o número de iterações, para avaliar realmente suas vantagens.

Quando as matrizes são diferentes, mesmo que para um mesmo n e p, nada se pode inferir, ou seja, não existe um comportamento padrão (o que era de se esperar já que as matrizes são geradas aleatoriamente). Em alguns casos o ótimo é alcançado sem entrar na árvore; em outros, dentro da árvore, o ótimo pode ou não ser alcançado dependendo do número de iterações.

Há que se avaliar o problema quando n se aproxima de 200 pois neste caso o tempo de processamento é elevado.

Neste algoritmo não se usou nenhuma heurística para o cálculo do limitante superior inicial. A literatura indica que para um bom LS inicial, o "gap" de dualidade é menor no início da árvore e portanto há menos nós na árvore.

REFERÊNCIAS BIBLIOGRÁFICAS

BEASLEY, J.E. A note on solving large P-MEDIAN problems. **European Journal of Operational Research, 21**:270-273, 1985.

BOFFEY, T.B.; KARKAZIS, J. p-Medians and Multi-Medians. **Journal of the Operational Research Society, 35**(1):57-64, 1984.

CHRISTOFIDES, N.; BEASLEY, J.E. A tree search algorithm for the p-median problem. **European Journal of Operational Research, 10**(2):196-204, 1982.

CORNUEJOLS, G.; FISHER, M.L.; NEMHAUSER, G.L. Location of bank accounts to optimize float : an analytic study of exact and approximate algorithms. **Management Science 23**:789-810, 1977.

DIEHR, G. An algorithm for the p-median problem. Los Angeles, CA. Western Man. Sci. Institute, University of California, 1972. (**Working paper 191**).

EVERETT III, H. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. **Operations Research, 11**:399-417, 1963.

EFROYMSON, M.A.; RAY, T.L. A branch-bound algorithm for plant location. **Operations Research, 14**:361-368, 1966.

FISHER, M.L. The lagrangean relaxation method for solving integer programming problems. **Management Science, 27**(1):1-18, 1981.

FISHER, M.L. An application oriented guide to lagrangean relaxation. **Interfaces, 15**(2):10-21, 1985.

GARFINKEL, R.S.; NEEBE, A.W.; RAO, M.R. An Algorithm for the m-median plant location problem. **Transportation Science, 8**:217-236, 1974.

GEOFFRION, A.M. Lagrangian relaxation and its uses in integer programming. **Mathematical Programming Study,** 2:82-114, 1974.

HAKIMI, S.L. Optimum locations of switching centers and the absolute centers and medians of a graph. **Operations Research, 12**:450-459, 1964.

HAKIMI, S.L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. **Operations Research, 13**:462-475, 1965.

HANDLER, G.Y.; MIRCHANDANI, P.B. Location on networks theory and algorithms. Cambridge, MA. The MIT Press, 1979.

HELD, M.; KARP, R.M. The traveling salesman problem and minimum spanning trees. **Operations Research**, 18:1138-1162, 1970.

HELD, M.; WOLFE, P.; CROWDER, H.P. Validation of subgradient optimization. **Mathematical Programming,** 6(1):62-88, 1974.

JARVINEN, P.; RAJALA, J.; SINERVO, H. A branch and bound algorithm for seeking the p-median. **Operations Research, 20**:173-178, 1972.

KHUMAWALA, M.B.; NEEBE, A.W.; DANNENBRING, D.G. A note on El-Shaieb's new algorithm for location sources among destinations. **Management Science, 21**(2):230-233, 1974.

KUSIAK, A.; VANNELLI, A.; KUMAR, K.R. Clustering analysis: models and algorithms. In School on Combinatorial Optimization, Rio de Janeiro, 8-19 jul. 1985.

MARANZANA, F.E. On the location of supply points to minimize transport costs. **Operational Research Quarterly, 15**:261-270, 1964.

MULVEY, J.M. Multivariate stratified sampling by optimization. **Management Science, 29(6)**:715-724, 1983.

NARULA, S.C.; OGBU, U.I.; SAMUELSSON, H.M. An algorithm for the p-median problem. **Operations Research, 25**:709-713, 1977.

NEEBE, A.W. A branch and bound algorithm for the p-median transportation problem. **Journal of the Operational Research Society, 29**(10):989-995, 1978.

OLIVO, A.A.; SENNE E.L.F. Implementação de um método de subgradientes para o problema das p-medianas. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL (CNMAC), 11., Ouro Preto, ago. 1988. **Resumo dos Trabalhos.** [Rio de Janeiro], SBMAC, 1988. p. 8-10.

POLJAK, B.T. A general method for solving extremum problems. **Soviet Mathematics Dokl., 8**(3):593-597, 1967.

RAGGI, L.A.; GALVÃO, R.D. Um algoritmo para o problema das p-medianas com custos fixos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL (SOBRAPO), 15. Rio de Janeiro, 8-11 nov. 1982. **Anais.** [Rio de Janeiro], SOBRAPO, 1982. p. 320-334.

REVELLE, C.; SWAIN, R.W. Central facilities location. **Geographical Analysis, 2:**30-42, 1970.

TEITZ, M.B.; BART, P. Heuristic methods for estimating the vertex median of a weighted graph. **Operations Research, 16**:955-961, 1968.

TENENBAUM, A.M.; AUGENSTEIN, M.J. **Data structures using Pascal.** Englewood Clifes, NJ. Prentice-Hall, 1981.

APÊNDICE A

LISTAGEM DO PROGRAMA

```
{$R+,U+}
program k_mediana;

const
  dim_n = 205;          (*   dim_n = n                  *)
  dim_p = 9384;         (*   dim_p = (n - p).p          *)
  dim_d = 20910;        (*   dim_d = (n.n - n) div 2    *)
type
  nopt   = ^tipono;
  tipono = record
             pl,
             pm       : real;
             valor    : integer;
             ant,
             filhoe,
             filhod   : nopt;
           end;
var
  ch        : char;
  i,
  j,
  k,
  n,
  p,
  ls,
  val,
  d_min,
  conte,
  ls_incumb,
  ls_inicial,
  status,
  tam_arv,
```

```
   rec,
   iter_inicial,
   itertot   :  integer;
   fim,
   arvore,
   documenta :  boolean;
   zd,
   tempo,
   zd_max_inicial,
   zd_max    :  real;
   d         :  array [0..dim_d] of integer;
   pos       :  array [1..dim_p] of integer;
   e,
   posmed,
   posicao   :  array [1..dim_n] of integer;
   sub,
   lambda    :  array [1..dim_n] of real;
   pp,
   pq,
   pai       :  nopt;
   arq       :  text;

function RELOGIO : real;

(*  Retorna o tempo atual em segundos  *)
(*  ---------------------------------  *)
var
  regs : record
           AX,BX,CX,DX,BP,SI,DI,DS,ES,flags : integer;
         end;
  hor,
  min,
  seg  : real;
begin
  with regs do
    begin
```

```
      AX := $2C00;
      msdos (regs);
      hor := hi(CX);
      min := lo(CX);
      seg := hi(DX);
      relogio := 3600*hor + 60*min + seg;
    end;
end;

function ENDER (i,j : integer) : integer;
begin
  if i < j then
    ender := e[i] + j - i - 1
  else
  if i > j then
    ender := e[j] + i - j - 1
  else
    ender := 0;
end;

procedure ESPERA;
var
  x : char;
begin
  if ch = 'v' then
    begin
      write('Aperte uma tecla... ');
      readln(x);
    end;
end;
```

```
function MAKETREE (VAL : integer) : NOPT;
var
  pp : nopt;
begin
  new(pp);
  tam_arv := tam_arv + 1;
  pp^.valor := val;
  pp^.ant := nil;
  pp^.filhoe := nil;
  pp^.filhod := nil;
  maketree := pp;
end;

procedure CRIAFILHOE (VAR PP : NOPT; VAL : INTEGER);
var
  pq : nopt;
begin
  if pp = nil then
    writeln(arq,'** Criacao invalida - No = NIL')
  else
    if pp^.filhoe <> nil then
      writeln (arq,'** Criacao invalida - Ja existe filho a esquerda')
    else
      begin
        pq := maketree(val);
        pq^.ant := pp;
        pp^.filhoe := pq;
        pp := pq;
      end;
end;   (* CRIAFILHOE *)
```

```
procedure CRIAFILHOD (VAR PP : NOPT ; VAL : INTEGER);
var
  pq : nopt;
begin
  if pp = nil then
    writeln(arq,'** Criacao invalida - No = NIL')
  else
    if pp^.filhod <> nil then
      writeln (arq,'** Criacao invalida - Ja existe filho a direita')
    else
      begin
        pq := maketree(val);
        pq^.ant := pp;
        pp^.filhod := pq;
        pp := pq;
      end;
END;   (* CRIAFILHOD *)

procedure TITULO;
begin
  clrscr;
  textcolor(0);
  textbackground(7);
  writeln(' ':30,' PROGRAMA K-MEDIANA ',' ':30);
  textcolor(7);
  textbackground(0);
  writeln;
end;

procedure ATUALIZA_LIMITES(pp : nopt);
begin
  pp^.pl := zd_max;
  pp^.pm := zd;
end;
```

```
procedure INICIALIZA_ARVORE;
begin
  if ls < ls_incumb then
    begin
      TITULO;
      rec := rec + 1;
    end;
  ls_incumb := ls;
  zd_max := zd_max_inicial;
  for i := 1 to p do
    posmed[i] := pos[i];
  pai := MAKETREE(0);
  ATUALIZA_LIMITES(pai);
  pp := pai;
  arvore := true;
  conte := 0;
end;

procedure MOSTRA_ARVORE;
var
  pq : nopt;
begin
  pq := pp;
  write('*** Arvore: ');
  while pq <> pai do
    begin
      write(pq^.valor,'   ');
      pq := pq^.ant;
    end;
  writeln;
end;
```

```
procedure QUICKSORT (L,R : INTEGER);
var
  x,w   : real;
  i,j,k : integer;
begin   (* QuickSort *)
  i:=l;
  j:=r;
  x:=sub[(l+r) div 2];
  repeat
    while sub[i] < x do i:=i+1;
    while x < sub[j] do j:=j-1;
    if i <= j then
      begin
        w:=sub[i];
        sub[i]:=sub[j];
        sub[j]:=w;
        k:=pos[i];
        pos[i]:=pos[j];
        pos[j]:=k;
        i:=i+1;
        j:=j-1;
      end
  until i > j;
  if l < j then QUICKSORT (L,J);
  if i < r then QUICKSORT (I,R);
end;    (* QUICKSORT *)
```

```
function ACHA_POSICAO(v : integer) : integer;
var
  i     : integer;
  achei : boolean;
begin
  i := 1;
  achei := false;
  while not achei do
    if pos[i] = v then
      achei := true
    else
      i := i + 1;
  acha_posicao := i;
end;

procedure SUBGRADIENTE;
var
  i,j,k,l,z,
  iter,
  np,
  pass          : integer;
  viavel        : boolean;
  pi,
  teta,
  zd_art,
  subgrad       : real;
  med           : array [1..dim_n] of integer;
begin
  pass := 0;
  iter := 0;
  if arvore then
    pi := 1
  else
    pi := 2;
  if not documenta then
    TITULO;
```

```
  repeat
    iter := iter + 1;
    pass := pass + 1;
    if documenta then
      writeln(arq,'----':10,' Iteracao ',iter,' ---- ','Pass ',
              pass,' ----')
    else
      begin
        gotoxy(5,3);
        clreol;
        write('*** Iteracao ',iter,'   Pass ',pass);
        if arvore then
          write('   No ',tam_arv,'   Recomecos : ',rec);
      end;

(*   Resolver problema lagrangeano   *)
(*   ----------------------------    *)
    d[0] := 0;
    for i := 1 to n do
      begin
        sub[i] := 0;
        pos[i] := i;
        med[i] := 9999;
        for j := 1 to n do
          begin
            k := ENDER(i,j);
            if d[k] < lambda[j] then
              sub[i] := sub[i] + d[k] - lambda[j];
          end;
      end;
    if documenta then
      begin

        (*  Custos = D(i,j)  *)
        (*  ---------------  *)
        writeln(arq,'*** custos:');
```

```
    for i := 1 to n do
      writeln(arq,' ':6,'c[',i,'] = ',sub[i]:1:3);
  end;

(*  Ordena a matriz de custos e pos(i)  *)
(*  ----------------------------------  *)
QUICKSORT(1,n);

if arvore then
  if not documenta then
    begin
      gotoxy(5,14);
      MOSTRA_ARVORE;
    end;

(*  Acertar o conjunto de medianas  *)
(*  ------------------------------  *)
if arvore then
  begin

    (*  Retira os filhos direitos  *)
    (*  -------------------------  *)
    pq := pp;
    while pq <>pai do
      begin
        val := pq^.valor;
        if val < 0 then
          begin
            val := - val;
            j := acha_posicao(val);
            subgrad := sub[j];
            for i := j+1 to n do
              begin
                pos[i-1] := pos[i];
                sub[i-1] := sub[i];
              end;
```

```
            pos[n] := val;
            sub[n] := subgrad;
            if documenta then
              begin
                for i := 1 to p do
                  write(arq,' ',pos[i],'  ');
                writeln(arq);
                ESPERA;
              end;
          end;
        pq := pq^.ant;
      end;

    (*  Inclui os filhos esquerdos  *)
    (*  --------------------------  *)
    pq := pp;
    np := p;
    while pq <>pai do
      begin
        val := pq^.valor;
        if val > 0 then
          begin
            j := acha_posicao(val);
            subgrad := sub[j];
            pos[j] := pos[np];
            sub[j] := sub[np];
            pos[np] := val;
            sub[np] := subgrad;
            np := np - 1;
          end;
        pq := pq^.ant;
      end;
end;
```

```
(*  Calculo do Dual --> ZD  *)
(*  ----------------------  *)
zd := 0;
for i := 1 to p do
  begin
    med[pos[i]] := -1;
    zd := zd + sub[i];
  end;
for i := 1 to n do
  zd := zd + lambda[i];
if documenta then
  writeln(arq,'*** zd = ',zd:1:3);

(*  Atualiza o limite inferior --> ZD_MAX  *)
(*  ------------------------------------  *)
if zd > zd_max then
  begin
    pass := 0;
    zd_max := zd;
    zd_art := zd_max;
  end;

(*  Calcular subgradientes para verificar viabilidade  *)
(*  e calculo do primal                                *)
(*  -------------------------------------------------  *)

viavel := true;
d[0] := 9999;
z := 0;
```

```
for i := 1 to n do
  begin
    sub[i] := 1;
    if med[i] = -1 then
      sub[i] := 0
    else
      begin
        for j := 1 to p do
          begin
            k := ENDER(i,pos[j]);
            if d[k] < lambda[i] then
              sub[i] := sub[i] - 1;
            if d[k] < med[i] then
              med[i] := d[k];
          end;
        z := z + med[i];
      end;
    if sub[i] <> 0 then viavel := false;
  end;

if documenta then
  begin
    writeln(arq,'*** subgradientes:');
    for i := 1 to n do
      writeln(arq,' ':6,'sub[',i,'] = ',sub[i]:1:3);
    writeln(arq,'*** z = ',z);
  end;

(*  Atualizacao do limitante superior --> LS  *)
(*  ----------------------------------------  *)
if z   ls then
  begin
    ls := z;
    zd_art := zd_max;
  end
else
```

```
   (*  Nao houve melhora no ls  *)
   (*  -----------------------  *)
   zd_art := zd_art + 0.2*(ls - zd_art);

(*  zd_art nao pode superar ls  *)
(*  --------------------------  *)
if zd_art > ls then
  zd_art := zd_max;
if not documenta then
  begin
    gotoxy(5,6);
    clreol;
    writeln('*** limite superior: ',ls);
  end
else
  writeln(arq,'*** limite superior: ',ls);
if not documenta then
  begin
    gotoxy(5,8);
    clreol;
    writeln('*** zd: ',zd:1:4,'    zd_max: ',zd_max:1:4);
  end
else
  writeln(arq,'*** zd: ',zd:1:4,'    zd_max: ',zd_max:1:4);

(*  pi = 2 para 2*n iteracoes.                    *)
(*  Depois e' dividido ao meio a cada 5 iteracoes  *)
(*  ---------------------------------------------  *)
if not viavel then
  begin
    if (iter > 2*n) or (arvore) then
      if pass > 4 then
        begin
          pi := pi/2;
```

```
      if documenta then
        writeln(arq,'*** pi corrigido para ',pi,
                    ' apos ',pass,' iteracoes.');
      pass := 0;
    end;
if not documenta then
  begin
    gotoxy(5,10);
    clreol;
    writeln('*** pi: ',pi:1:6);
  end
else
  writeln(arq,'*** pi: ',pi:1:6);
teta := 0;
for i := 1 to n do
  teta := teta + sub[i] * sub[i];
if not arvore then
  teta := pi * (ls - zd_art) / teta
else
  teta := pi * (ls - zd) / teta;
if documenta then
  writeln(arq,'*** teta = ',teta);
if not documenta then
  begin
    gotoxy(5,12);
    clreol;
    writeln('*** teta: ',teta:1:6);
  end;
for i := 1 to n do
  begin
    lambda[i] := lambda[i] + teta * sub[i];
    if lambda[i] < 0 then
      lambda[i] := 0;
  end;
```

```
      if documenta then
        begin
          writeln(arq,'*** lambda corrigido:');
          for i := 1 to n do
            writeln(arq,' ':6,'lambda[',i,'] = ',lambda[i]:1:5);
        end;
    end;

  (*  Caso tenha entrado na arvore,                    *)
  (*  so vai executar 30 iteracoes do subgradiente.    *)
  (*  ---------------------------------------------    *)
  if arvore then
    if iter > 29 then
      pi := 0.004;
  itertot := itertot + 1;
until (viavel) or (pi < 0.005) or (ls - zd_max <= 0.05) or
      (ls < ls_incumb) or (itertot > 999);
if viavel then
  status := 0
else
if (zd_max <= ls) and (ls - zd_max <= 0.05) then
  status := 1
else
if ls< ls_incumb then
  status := 2
else
  status := 3;
```

```
    if documenta then
      begin
        writeln(arq,'*** Numero de nos: ',n);
        writeln(arq,'*** Numero de medianas: ',p);
      end
    else
      begin
        gotoxy(5,14);
        write(arq,'*** Status da solucao: ');
        if not arvore then
          case status of
            0: writeln(arq,'OTIMA - SUBGRADIENTE NULO');
            1: writeln(arq,'OTIMA');
            3: writeln(arq,'NAO OTIMA');
          end
        else
          case status of
            0: writeln(arq,'SUBGRADIENTE NULO - NO ESGOTADO');
            1: writeln(arq,'NO ESGOTADO - POR LIMITES');
            2: writeln(arq,'RECOMECA A ARVORE');
            3: writeln(arq,'NAO OTIMA');
          end;
        writeln(arq,'*** Valor da solucao, ls = ',ls,
                    '    z = ',z,'    zd = ',zd:1:2,
                    '    (dual: ',zd_max:1:2,')');
        writeln(arq,'*** Numero de iteracoes: ',iter);
        writeln(arq,'*** Alocacao das medianas: ');
        write(arq,pos[1]);
        for i := 2 to p do
          write(arq,', ',pos[i]);
        writeln(arq);
      end;
    if ((not arvore) and (status < 2)) or (itertot > 999) then
      fim := true;
  end;    (* SUBGRADIENTE *)
```

```
procedure CALCULA_ALFA (var d_min : integer);
const
  infinito = 1.0e+30;
var
  pq       : nopt;
  alfa     : array [1..dim_n] of real;
  alfa_min : real;
begin
  if documenta then
    begin
      gotoxy(1,16);
      for i := 1 to p do
        write(arq,'POS[',i,'] = ',pos[i],'  ');
      ESPERA;
    end;
  for i := 1 to p do
    begin
      alfa[pos[i]] := 0;
      for j := 1 to n do
        begin
          k := ENDER(pos[i],j);
          if d[k] < lambda[j] then
            alfa[pos[i]] := alfa[pos[i]] + d[k] - lambda[j];
        end;
    end;
  pq := pp;
  while pq <> pai do
    begin
      if pq^.valor > 0 then
          alfa[pq^.valor] := infinito;
      pq := pq^.ant;
    end;
  alfa_min := alfa[pos[1]];
  d_min := pos[1];
```

```
  for i := 2 to p do
    begin
      if alfa[pos[i]] < alfa_min then
        begin
          alfa_min := alfa[pos[i]];
          d_min := pos[i];
        end;
    end;
END;    (* Calcula_Alfa *)

(*************************************************************************
                              PROGRAMA PRINCIPAL
*************************************************************************)

begin
  TITULO;
  gotoxy(1,10);
  write('    Numero de medianas ..... ');
  readln(p);
  write('    (L)er ou (g)erar os dados? ..... ');
  readln(ch);
  if ch = 'l' then
    begin
      assign(arq,'a:dados25.med');
      reset(arq);
      readln(arq,n,k);
      for i := 1 to n-1 do
        read(arq,e[i]);
      readln(arq);
      for i := 1 to k do
        read(arq,d[i]);
    end
  else
    begin
      write('    Numero de nos ..... ');
      readln(n);
```

```
      write('    Gravar os dados em arquivo? (s/n): ');
      readln(ch);
      randomize;
      k := 0;
      for i := 1 to n-1 do
        begin
          e[i] := k + 1;
          for j := i+1 to n do
            begin
              k := k + 1;
              d[k] := random(96)+5;
            end;
        end;
      if ch = 's' then
        begin
          assign(arq,'a:dados25.med');
          rewrite(arq);
          writeln(arq,n,' ',k);
          for i := 1 to n-1 do
            write(arq,e[i],' ');
          writeln(arq);
          for i := 1 to k do
            write(arq,d[i],' ');
          close(arq);
        end;
    end;
  write('    Documentacao completa? (s/n) ..... ');
  readln(ch);
  documenta := ch = 's';
  write('    (V)ideo ou (i)mpressora? ..... ');
  readln(ch);
  assign(arq,'con');
  if ch = 'i' then
    assign(arq,'prn');
  rewrite(arq);
```

```
if documenta then
  begin
    writeln(arq,'*** distancias:');
    d[0] := 0;
    for i := 1 to n do
      begin
        for j := 1 to n do
          begin
            k := ENDER(i,j);
            write(arq,d[k]:4);
          end;
        writeln(arq);
      end;
    ESPERA;
  end;
d[0] := 9999;
for i := 1 to n do
  begin
    d_min := 9999;
    for j := 1 to n do
      begin
        k := ENDER(i,j);
        if d[k] < d_min then
          d_min := d[k];
      end;
    lambda[i] := d_min;
  end;
ls := 32767;          (*  Maior inteiro = infinito  *)
rec := 0;             (*  Numero de recomecos na arvore  *)
ls_incumb := 0;
itertot := 0;
tam_arv := 0;
zd_max := 0;
fim := false;
arvore := false;
```

```
(*  CHAMADA DA PROCEDURE PARA CALCULO DO SUBGRADIENTE  *)
(*  ------------------------------------------------  *)
tempo := RELOGIO;
SUBGRADIENTE;

clrscr;
writeln(arq,'** LIMITE SUPERIOR   = ',LS);
writeln(arq,'** ZD_MAX            = ',ZD_MAX:1:2);
writeln(arq,'** TOTAL ITERACOES   = ',ITERTOT);
writeln(arq,'** ALOCACAO DAS MEDIANAS:');
for i := 1 to p do
  write(arq,'   ',pos[i]);
writeln(arq);
iter_inicial := itertot;
ls_inicial := ls;
zd_max_inicial := zd_max;
ls_incumb := 0;
INICIALIZA_ARVORE;

while (not fim) do
  begin
    if conte < p then
      begin
        CALCULA_ALFA(val);
        CRIAFILHOE(PP,val);
        conte := conte + 1;
        SUBGRADIENTE;
        if status = 2 then
          INICIALIZA_ARVORE
        else
          ATUALIZA_LIMITES(pp);
      end;
```

```
    while ((status < 2) or (zd_max >ls) or (conte = p))
            and (not fim) do
      begin
        (*  verifica se vem de um filho direito  *)
        (*  ----------------------------------  *)
        val := pp^.valor;
        if val >0 then
          conte := conte - 1;
        pp := pp^.ant;
        zd_max := pp^.pl;
        zd := pp^.pm;
        while (pp <>pai) and (pp^.filhod <>nil) do
          begin
            val := pp^.valor;
            if val > 0 then
              conte := conte - 1;
            pp := pp^.ant;
            zd_max := pp^.pl;
            zd := pp^.pm;
          end;
        if pp = pai then
          if pp^.filhod <>NIL then
            fim := true;
        if not fim then
          begin
            CRIAFILHOD(PP,-val);
            SUBGRADIENTE;
            if status = 2 then
              INICIALIZA_ARVORE
            else
              ATUALIZA_LIMITES(pp);
          end;
      end;
end;
```

```
  TITULO;
  writeln(arq,'** LIMITE SUPERIOR INICIAL  = ',ls_inicial);
  writeln(arq,'** LIM.SUP.INCUMBENTE       = ',ls_incumb);
  writeln(arq,'** LIMITE INFERIOR INICIAL  = ',trunc(zd_max_inicial));
  writeln(arq,'** ITERACOES NO INICIO      = ',iter_inicial);
  writeln(arq,'** TOTAL ITERACOES          = ',itertot);
  writeln(arq,'** TAMANHO DA ARVORE        = ',tam_arv);
  writeln(arq,'** ALOCACAO DAS MEDIANAS:');
  for i := 1 to p do
    write(arq,'   ',posmed[i]);
  writeln(arq);
  tempo := RELOGIO - tempo;
  writeln(arq,'** TEMPO                    = ',tempo:1:2,'
              segundos.');
  writeln(arq,'** NUMERO DE RECOMECOS      = ',rec);

end.
```