DEFINIÇÃO TOPOLÓGICA DE REDES DE COMUNICAÇÃO DE DADOS

Walter Rocha Palma

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

Aprovada por:

Prof. Pierre-Jean Lavelle
(Presidente)

Prof. Nelson Maculan Filho

Profa. Sueli Mendes dos Santos

RIO DE JANEIRO, RJ - BRASIL
DEZEMBRO DE 1979

PALMA, WALTER ROCHA

Definição Topológica de Redes de Comunicação de Dados (Rio de Janeiro) 1979.

X, 171p. 29,7cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1979)

Tese - Univ. Fed. Rio de Janeiro. Fac. Engenharia

1. Topologia de Redes de Comunicação de Dados

I. COPPE/UFRJ II. Título (série)

Aos meus pais e a minha noiva

AGRADECIMENTOS

- Aos meus professores orientadores Pierre-Jean Lavelle e Gerhard Schwarz;

- Aos demais professores e colegas da COPPE/UFRJ;

- A minha noiva pela ajuda na datilografia e desenhos deste trabalho;

- A minha irmã, Solange e meu cunhado, Carlos pela hospedagem durante minha estadia no Rio de Janeiro.

- A amiga Suely, pela datilografia da versão final deste trabalho.

RESUMO

Ao se planejar uma Rede de Comunicação de Dados é fundamental uma boa definição topológica devido a sua influência direta no custo total da rede. Quando a rede é pequena esta tarefa pode ser feita apenas pelo homem, porém, quando o número de pontos é grande, o problema torna-se tão complexo que é impossível ao homem chegar a uma boa solução sozinho, devido a grande variedade de soluções a serem analisadas. A opção que se tem é implementar um algoritmo num computador para que este resolva o problema independente ou com a interação do homem, que pode resultar em excelentes soluções.

Este trabalho se propõe a fazer uma cobertura, sobre os algoritmos e estudos de topologia de Redes de Comunicação de Dados e oferecer um novo algoritmo para definição topológica de redes, com vantagens sobre os demais, quanto a flexibilidade e versatilidade de soluções.

A apresentação dos algoritmos e estudos existentes foi dividida em classes de acordo com o tipo de rede abordada.

A apresentação do novo algoritmo (WRP) foi feita de forma simples e objetiva, visando facilitar a compreensão e utilização pelos interessados.

ABSTRACT

For planning a Data Communication Network it is fundamental to have a good topological definition, because of its direct influence in the total cost of the network. When the network is small this task can be made by man, but when the number of points is large, the problem becomes so complex that it is impossible for man to find a good solution, because of the great variety of solutions that have to be analyzed. The option we have is to implement an algorithm in a computer and let it solve the problem with or without man's interaction, and that can offer excellent solutions.

The goal of this work is to cover the algorithms and topological studies of Data Communication Networks and to propose a new algorithm for topological definition of networks with advantages, in flexibility and versatility of solutions, over the others.

The presentation of the algorithms and known studies, were divided in classes, according to the type of the mentioned networks.

The presentation of the new algorithm (WRP) was made in a simple and objective way, in order to facilitate the understanding and utilization to whom are interested in this subject.

ÍNDICE

Páginas

Páginas

CAPÍTULO I

INTRODUÇÃO

Ao se pensar em uma Rede de Comunicação de Dados a ser implantada torna-se fundamental o seu planejamento cuidadoso, nos seus diversos níveis, para assegurar um funcionamento compensador.

Na fase de planejamento da rede, a definição topológica, que consiste basicamente na determinação da melhor configuração de linhas para conexão dos seus pontos (terminais, concentradores, etc), torna-se relevante devido a sua influência direta no custo total da rede. Este trabalho concentra-se justamente no estudo da Topologia de Redes de Comunicação de Dados.

A depender do tipo de rede desejada serão necessárias diferentes tarefas para a sua definição topológica, podendo envolver:

- determinação do número, tipo e localização dos comutadores ("switches").
- determinação do número, tipo e localização dos concentradores.
- determinação do desenho ("lay-out") e do tipo das linhas para conexão dos terminais aos concentradores e estes aos comutadores, ou a um computador central.

Pesquisas desta natureza são importantes, devido a tendência da criação e evolução de Redes de Comunicação de Da-

dos (RCD) não só em nosso país como em diversos outros.

O trabalho está dividido em duas partes: na primeira, consta um estudo comparativo dos diversos trabalhos existentes na literatura, com a análise dos algorítmos expostos, enfocando as vantagens e desvantagens de cada um, de acordo com os objetivos propostos; na segunda parte, encontra-se o desenvolvimento de um algorítmo para definição topológica de RCD que apresente vantagens sobre os demais.

No capítulo II, será apresentada a Revisão da Literatura, por meio do estudo comparativo dos trabalhos existentes sobre topologia de redes.

No capítulo III, será apresentado o algorítmo desenvolvido, de uma maneira suscinta que facilite a compreensão.

No capítulo IV serão expostos os resultados e as discussões sobre o algorítmo desenvolvido.

Finalmente, as conclusões sobre o trabalho serão fornecidas no capítulo V.

CAPÍTULO II

REVISÃO DA LITERATURA

(ESTUDO COMPARATIVO DOS TRABALHOS EXISTENTES)

II.1. Introdução

Uma boa definição topológica de uma rede de comunicação de dados é de fundamental importância no planejamento da mesma, visto que influi diretamente no seu custo. Evidentemente se ela tiver poucos pontos a serem conectados a rede de menor custo pode ser definida diretamente, sem a ajuda de um algorítmo apropriado, o que se torna impossível quando o número de pontos é grande.

Existem muitos artigos e algorítmos na literatura que abordam este assunto. Como existem diversos tipos de rede, vários estudos e algorítmos foram desenvolvidos para cada um dos diferentes tipos e podem ser divididos em classes, de acordo com o tipo da rede.

Neste capítulo, será apresentada esta divisão em classes, com a análise do que foi publicado de importante em cada uma delas.Dentro de cada classe os estudos serão expostos em ordem mais ou menos cronológica, focalizando as interrrelações e evolução das idéias básicas, seguido das vantagens e desvantagens dos algorítmos apresentados.

II.2. Divisão em Classes por Tipo de Rede

II.2.1. Redes Centralizadas

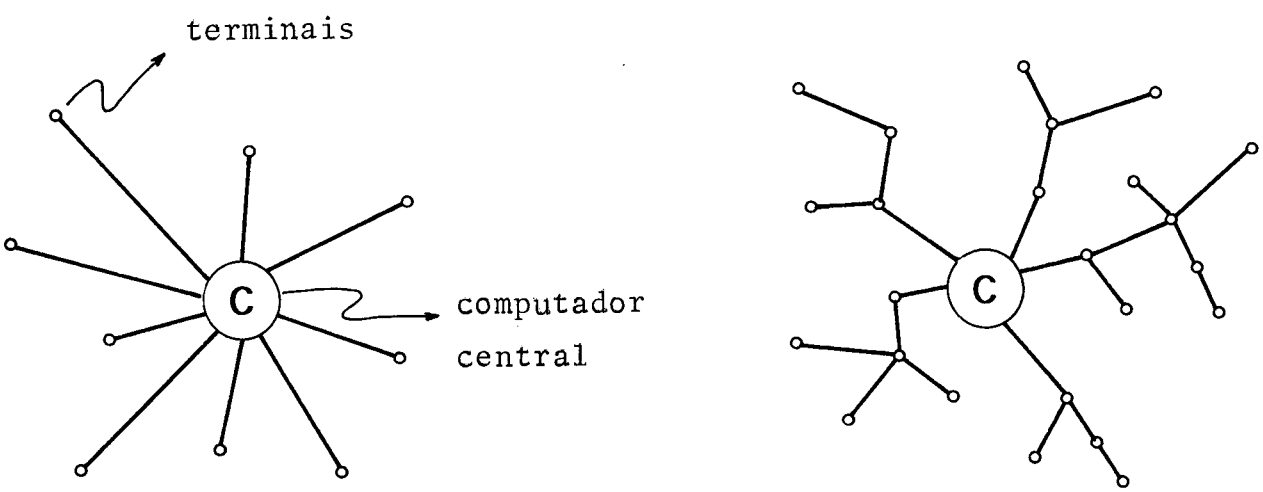
Este tipo de rede é caracterizado pelos vários ter-

minais dispersos geográficamente e conectados a um ou mais computadores centrais que se encarregam de todo o processamento e que centralizam todos os recursos.

O problema neste tipo de rede é encontrar a rede de menor custo para ligação dos terminais ao computador central, cuja localização é conhecida a priori.

II.2.1.1. Redes Centralizadas com Apenas um Nível de Ligação

Neste tipo de rede os terminais são ligados ao computador central sem a utilização de concentradores por meio de linhas ponto-a-ponto ou multiponto (figura 1).



centralizada c/linhas ponto-a-ponto. Configuração = ESTRELA =

centralizada c/linhas multiponto. Configuração = ÁRVORE = (C é a raiz)

FIGURA 1 : Exemplos de redes ESTRELA e ÁRVORE

No caso da configuração ESTRELA não existem alternativas estratégicas para ligação dos terminais ao computador central, visto que cada terminal é ligado diretamente ao computador central por meio de uma linha ponto-a-ponto. Já no caso da configuração ÁRVORE, existem várias alternativas para a ligação dos terminais ao computador central, resultando numa ÁRVORE diferente para cada alternativa. Cada ÁRVORE é formada por um conjunto de linhas diferentes, tendo portanto um custo total de "comunicação" também diferente, visto que as tarifas das linhas de comunicação são geralmente proporcionais à distância das linhas.

Vejamos uma definição simples de Árvore Geradora Mínima ("Minimal Spanning Tree" - MST):

Dado um conjunto de pontos {T}, cujos elementos t_i, i=1,2,...,n possuem coordenadas (x_i,y_i) chama-se Árvore Geradora Mínima (MST) aquela que conecta todos os pontos do conjunto T de maneira que, a soma de todos os elos é mínima.

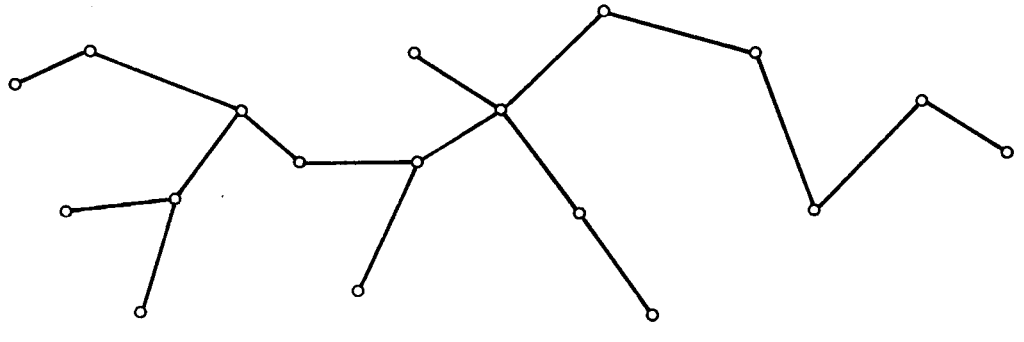


FIGURA 2 : MST - Árvore Geradora Mínima

Vale lembrar que, por ser uma árvore, a MST não forma ciclo, isto é, não existe um conjunto de conexões partindo de um ponto i e voltando a este mesmo ponto i (figura 2).

Na configuração ÁRVORE o problema é encontrar aquela que forneça o custo total de comunicação mínimo. É fácil verificar que a MST seria a solução para este problema, devido às características de proporcionalidade entre o custo das linhas de comunicação e a distância entre os pontos desta linha. Existem no entanto, certas restrições relativas às linhas multiponto que não são previstas na construção da MST, quais sejam:

- Cada linha multiponto só pode suportar um determinado número de terminais conectados a ela.

- Cada linha multiponto só suporta uma determinada quantidade de tráfego, em termos de número de bits/seg. O tráfego total de cada ramo da linha é o somatório dos trafegos gerados em cada terminal que está conectado a este ramo.

Existem diversos algorítmos que se propõem a encontrar a ÁRVORE de custo mínimo, ou próximo do mínimo, que conecta terminais a um computador central por meio de linhas multiponto. Dentre eles serão apresentados os seguintes:

- KRUSKAL - J. B. Kruskal, 1956 /SCHWM76,cap 9/
- PRIM'S - R. C. Prim, 1957 /SCHWM76, cap 9/
- ESAU-WILLIAMS - L.R. Esau e K.C. Williams, 1966 /SCHWM76, cap 9/
- UNIFICADO - A. Kershenbaum e W. Chou, 1974 /SCHWM76, cap 9/

- MARTIN - J. Martin, 1972 /MARTJ72, cap 40/
- WHITNEY'S - V. K. M. Whitney, 1970 /ELIAD74/
- SETORIAL - R. L. Sharma e M. T. El-Bardei, 1970 /ELIAD74/
- NEWCND - /ELIAD74/
- SOGA - /KARNM76/

Seguem-se as apresentações dos algorítmos citados:

Os quatro primeiros algorítmos que serão apresentados usam estratégias heurísticas para resolver o problema definido da seguinte maneira:

- Dado um conjunto de pontos correspondente aos terminais e um ponto representando o computador central, determinar a rede centralizada em configuração ÁRVORE, de custo mínimo, para ligar todos os pontos ao computador central, sujeito à restrição que o tráfego total numa linha multiponto não pode exceder a um valor pré-determinado, em bits/seg, que é o mesmo para todas as linhas.

São dados inicialmente também o tráfego gerado em cada terminal e a função de custo de ligação de dois pontos quaisquer. Para simplificar, será usada a distância como sendo o custo da ligação entre os pontos considerados.

Notação utilizada: $C = [c_{ij}]$

Matriz de custos:

c_{ij} = custo da ligação $i \rightarrow j$ (no caso, equivalente a distância)

$c_{ii} = \emptyset$ e $c_{ji} = c_{ij}$

a_i = tráfego gerado em cada terminal (considerado estável)

1) Algorítmo KRUSKAL /SCHWM76/

A filosofia básica deste algorítmo é simplesmente escolher a cada passo a ligação de custo mínimo atual. Se a ligação satisfaz a restrição de capacidade dos elos ("link") e não forma um "ciclo" na rede, então ela será considerada, caso contrário, será ignorada. Prossegue desta maneira até não haver mais ligações a serem consideradas, quando a rede estara formada.

Obs.: não há restrição de capacidade na ligação de um ponto ao computador central.

Algorítmo:

0) Inicialize a matriz de custos;

1) Escolha na matriz de custos a ligação de menor custo; Se for ∞ , então pare.

2) A ligação $i \rightarrow j$ forma ciclo?

2.1) Sim: faça o $c_{ij} \leftarrow \infty$; vá para 2;

2.2) Não: satisfaz a restrição de capacidade?

2.2.1) Sim: considere a ligação $i \rightarrow j$; faça $c_{ij} \leftarrow \infty$ atualize a restrição de capacidade; vá para 2;

2.2.2) Não: faça $c_{ij} \leftarrow \infty$; vá para 2;

A grande desvantagem deste algorítmo é a necessidade de prever a formação de ciclos. Observa-se também que a rede resultante não é, geralmente, a mais próxima da ótima.

2) Algorítmo PRIM'S /SCHWM76/

A filosofia básica deste algorítmo é ligar inicialmente o ponto mais próximo do computador central a ele. Depois, escolhe a cada passo a ligação de menor custo ao último ponto ligado, até que a ligação não satisfaça mais a restrição de capacidade, momento em que o novo ponto é ligado ao computador central, dando origem a uma nova linha.

Este algorítmo utiliza pesos associados aos pontos (w_i's) , para construir uma matriz de balanceamento, T, baseada nos pesos e nos custos.

Algorítmo:

0) Inicialize o vetor $W = [w_i]$:

$w_1 \leftarrow 0$, $w_i \leftarrow -\infty$, para $i \neq 1$;

Inicialize a matriz de custos, $C = [c_{ij}]$

Inicialize a matriz de balanceamento, $T = [t_{ij}]$:

$t_{ij} \leftarrow c_{ij} - w_i$, para todo i,j;

Obs: como as escolhas das ligações a serem efetuadas são feitas na matriz de balanceamento, esta inicialização faz com que inicialmente só sejam consideradas ligações entre terminais e o computador central.

1) Escolha o mínimo t_{ij};

Se for ∞ , então pare.

2) Satisfaz a restrição de capacidade?

2.1) Sim: considere a ligação $i \rightarrow j$;

faça $w_j \leftarrow \emptyset$; atualize a restrição de capacidade;

atualize a matriz de balanceamento T; vá para 2;

2.2) Não: faça min $t_{ij} \leftarrow \infty$; vá para 2;

Este algorítmo apresenta a vantagem de não precisar prever a formação de "ciclo", mas, geralmente, não encontra ainda a solução mais próxima da ótima.

3) Algorítmo ESAU-WILLIAMS /SCHWM76/

A filosofia básica deste algorítmo é escolher as ligações de menor custo, a partir dos pontos mais afastados do computador central, formando várias linhas simultanemanete. Prossegue desta maneira no sentido das folhas para a raiz (considerando a rede como uma árvore) até que se esgotem as possibilidades de serem feitas mais ligações sem violar a restrição de capacidade. Os pontos que sobrarem sem ter um ponto ao qual estejam ligado são conectados ao computador central.

Como o PRIM'S, este algorítmo utiliza uma matriz de balanceamento, que no entanto é construida de maneira diferente.

Algorítmo:

0) Inicialize a matriz de custos; Inicialize a matriz de balanceamento, $T = [t_{ij}]$: $t_{ij} \leftarrow c_{ij} - c_{il}$, para todo i,j;

   Obs.: como a escolha da ligação será feita na matriz de balanceamento, esta inicialização faz com que sejam escolhidos os pontos mais afastados antes dos menos afastados do centro.

1) Escolha o mínimo t_{ij};

2) $t_{ij} > 0$?

2.1) Sim: Ligue os pontos que não estão ligados, ao computador central; pare.

3) A restrição de capacidade é satisfeita?

3.1) Sim: Considere a ligação $i \leftarrow j$; Atualize a matriz balanceamento; Atualize a restrição de capacidade; Vá para 2;

3.2) Não: Faça $T_{ij} \leftarrow \infty$; Vá para 2;

Este algorítmo apresenta a vantagem de não precisar prever a formação de "ciclo" e encontra a solução mais próxima da ótima, deste grupo de algorítmos.

4) Algorítmo UNIFICADO /SCHWM76/

Kershenbaum e Chou observaram que os algorítmos anteriores diferiam na filosofia da escolha da ligação, e isto dependia diretamente dos pesos (w_i's) associados a cada ponto no início e a respectiva atualização a cada passo, mas que conservam a mesma estrutura básica. Eles então desenvolveram um algorítmo que possui uma estrutura básica semelhante aos anteriores e que é capaz de simula-los a depender da inicialização dos w_i's , que são utilizados na construção da matriz de balanceamento, e a da sua atualização a cada passo do algorítmo.

As diferenças relativas aos w_i's são mostradas nesta tabela:

ALGORÍTMO	INICIALIZAÇÃO	ATUALIZAÇÃO
KRUSKAL	$w_i \leftarrow \emptyset$, p/i=1,...,N	
PRIM'S	$w_i \leftarrow \emptyset$	
	$w_i \leftarrow -\infty$,p/i=2,...,N	$w_j \leftarrow \emptyset$
ESAU-WILLIAMS	$w_i \leftarrow C_{i1}$,p/i=1,...,N	$w_i \leftarrow w_j$

Algorítmo:

0) Inicialize a matriz de custos; Inicialize os pesos w_i's (de acordo com a tabela); Inicialize a matriz de balanceamento (ver tabela);

1) Escolha o mínimo t_{ij};

2) $t_{ij} = \infty$?

2.1) Sim: Pare.

3) A restrição de capacidade é satisfeita?

3.1) Sim: Considere a ligação $i \rightarrow j$; Atualize os w_i's (ver tabela); Atualize a matriz de balanceamento; Vá para 2;

3.2) Não: Faça $t_{ij} \leftarrow \infty$; Vá para 2;

Kreshenbaum e Chou mostram uma implementação prática eficiente do algorítmo UNIFICADO, que reduz a ordem do algorítmo de $N^2 \log N$ para $K_1 N \log N + K_2 N$, onde N é o número de pontos.

5) Algorítmo de MARTIN /MARTJ72/

A idéia básica deste algorítmo é fundamentada na de Esau-Williams, com a diferença de saturar uma linha por vez e ligá-la então ao computador central, ao invés de construir várias linhas simultaneamente, como faz o de Esau-Williams.

Algorítmo:

1) Obtenha a estimativa de carga de tráfego para cada localidade;

2) Obtenha o número de terminais em cada localidade;

3) Obtenha a carga máxima permitida na linha e o tempo de resposta máximo;

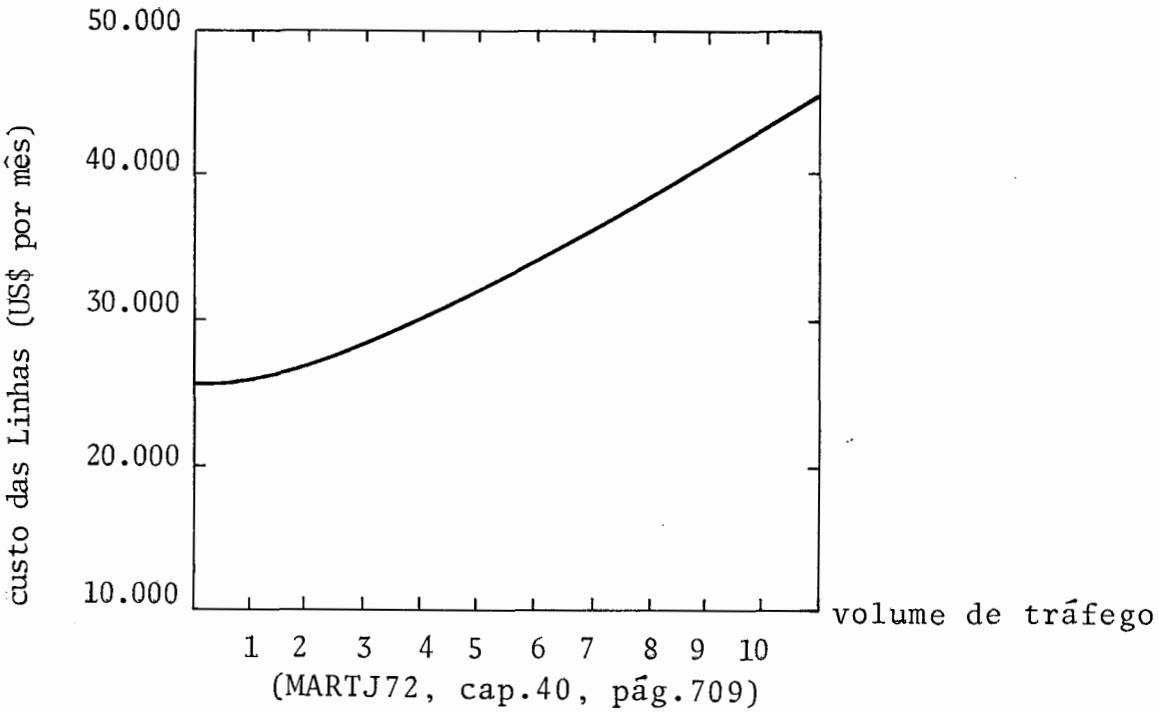
4) A partir das localidades mais afastadas do computador central, ligue os terminais a uma linha multiponto, até saturá-la, por carga máxima ou por número máximo de terminais por linha. Esta linha saturada é ligada ao computador central. Prossegue desta maneira até todos os terminais serem ligados.

Como o próprio Martin reconhece, este algorítmo é aplicável apenas para redes simples, que possam inclusive ser construídas sem auxílio de um computador, seguindo-se manualmente este algoritmo.

Para redes complexas ele recomenda o algoritmo de Esau-Williams, que oferece um resultado bem próximo do ótimo, com a vantagem de requerer pouco tempo de computação (MARTJ72).

Martin observa que multiplicando a carga de tráfego, a alteração resultante na configuração e no custo é suave, conforme mostra o gráfico.

GRÁFICO 1



(MARTJ72, cap.40, pág.709)

6) Algorítmo WHITNEY'S /ELIAD74/

Este algorítmo é uma variante do de Esau-Williams. Inicialmente ele constrói uma árvore do tipo ESTRELA, ligando todos os terminais ao computador central, diretamente. A partir daí, o algorítmo, iterativamente, seleciona um elo ("link") a ser removido e substituido por outro, que restaura a conectividade da árvore, maximiza a redução do custo da linha e não viola as restrições de capacidade da linha. Prossegue desta maneira até não poder mais remover elos.

Embora este algorítmo maximize o ganho nos custos a cada iteração, ele não considera um grande número de topologias, possivelmente boas.

7) Algorítmo SETORIAL /ELIAD74/ (de R.L.Sherma e M.T.E-Bardai)

A filosofia básica deste algorítmo é particionar o conjunto do N terminais em subconjuntos viáveis , pela divisão da área ao redor do centro (computador central) em setores. Cada setor contem tantos terminais quantos possíveis em uma linha, sem violar as restrições da linha. O algorítmo obtem uma MST para cada partição.

Geralmente, este procedimento é repetido para diferentes divisões iniciais e a melhor solução é escolhida.

A desvantagem da estratégia deste algorítmo é que nem sempre os subconjuntos ótimos podem ser separados por linhas radiais .

8) Algorítmo NEWCND /ELIAD74/

Definição formal do problema:

Dado:

1) (n-1) terminais, onde cada terminal i está localizado nas coordenadas (x_i, y_i), i=2,...,n e o computador central em (x_1, y_1);

2) A função de custo de ligação entre dois pontos, em termos da distância entre os mesmos;

3) Tráfego gerado em cada terminal, δ_1, i=2,3,...,n;

4) Carga máxima por linha, λ_{max} e/ou número máximo de terminais por linha, m_{max}.

Determine:

A rede, em forma de ÁRVORE, mais econômica para ligar os terminais ao centro (computador central), sujeito as seguintes restrições:

- Tráfego máximo numa linha não deve exceder ao máximo permitido e/ou número máximo de terminais por linha não deve exceder o máximo permitido.

Obs.: A topologia que satisfaz as restrições é dita viável.

As restrições podem ser formalizadas da seguinte maneira:

1) $\max_w \lambda_w \leq \lambda_{max}$

sendo, $\lambda_w = \sum_{i \in \{N_w - \{CPU\}\}} \delta_i$

onde:

w - é um segmento, ou seja, uma linha multiponto ligada ao computador central.

N_w - é o conjunto de pontos da linha multiponto.

λ_w - é o tráfego total no segmento w.

λ_{max} - é o tráfego máximo permitido em uma linha.

CPU - é o computador central.

2) $\max_w m_w \leq m_{max}$

onde:

m_w - é o número de terminais no segmento w

m_{max} - é o número máximo de terminais permitidos em uma linha.

Vejamos dois teoremas, que apresentam propriedades de uma rede ótima, nos quais o algorítmo NEWCND baseia-se.

Sendo: N = conjunto de nós (terminais e computador central), onde 1 é o centro; T = topologia ótima, em ÁRVORE, contendo os "s" segmentos; N_w = conjunto dos nós do segmento w.

Obs.: O conjunto N de todos os nós é a união dos "s" segmentos, todos eles tendo o nó central em comum.

Teorema 1: Um segmento $w \leq s$ na árvore ótima restrita T é uma MST irrestrita sobre N_w. Sendo assim, a topologia sobre o segmento w pode ser substituida por uma topologia MST sobre N_w, sem destruir a topologia ótima.

Teorema 2: Se existe uma MST irrestrita sobre N que inclui os elos centrais $(j_1,1),\ldots,(j_k,1)$, então existe

uma solução ótima para o problema restrito que também inclui $(j_1,1),\ldots,(j_k,1)$.

Vejamos agora um método heurístico para resolver o problema, que foi formulado no algorítmo 8, que possui a seguinte filosofia:

- Constroi inicialmente a MST para conexão dos terminais ao computador central. A partir daí, a cada iteração, corta um elo que esteja violando as restrições e substitui por um novo elo, de maneira a satisfazer as restrições e causar o incremento mínimo no custo da rede.

Notação:

p_i = o nó próximo de i no caminho de i para o centro.

SET 1 = O conjunto de todos os nós contidos nos segmentos que satisfazem as restrições.

SET 2 = O conjunto completar do SET 1, ou seja, SET 2 = N - SET 1

i* - é o ponto pertencente ao SET 2, cujo elo (i^*, p_{i^*}) pode ser eliminado e substituído por outro, sem violar as restrições e causando o menor incremento de custo.

j* - é o ponto pertencente ao SET 1, cujo elo (i^*,j^*) será o substituto do elo (i^*,p_{i^*}), de maneira que não viole as restrições e cause o menor incremento de custo.

Descrição do método heurístico /ELIAD74/

1) Constroe a MST como rede inicial. Se a MST satisfaz as restrições, então a MST é a rede ótima e o método encerra. De outra forma prossegue com o passo 2.

2) Encontra iterativamente o elo (i^*,p_{i^*}), $i^* \in$ SET 2, deleta este elo e substitui pelo elo (i^*,j^*), $j^* \in$ SET 1. O critério para a escolha dos nós i^* e j^* é tal que a deleção do elo (i^*,p_{i^*}) e sua substituição pelo elo (i^*,j^*) possa causar o menor incremento no custo e possa reduzir as ligações que violam as restrições.

3) Quando a rede não mais violar as restrições, é feita uma tentativa de redução de custo por meio de remanejamento das ligações, se o remanejamento não violar as restrições. Após este último passo a topologia final é definida e o método é encerrado.

Um algorítmo que utilize este método converge rapidamente para a solução final, mas, visto que não considera um grande número de topologias, a solução encontrada pode ser longe da ótima.

O algorítmo NEWCND /ELIAD74/ é baseado neste método heurístico, tendo porém a seguinte modificação, para encontrar uma solução final melhor:

- Admitir a deleção do elo (i^*,p_{i^*}) e sua substituição pelo (i^*,j^*) para produzir o tráfego no segmento contendo o nó j^*, maior ou igual ao tráfego do segmento contendo o nó i^*, desde que a conexão não tenha sido considerada antes e não

acarrete um incremento no custo da linha da rede.

Segundo os estudos apresentados em/ELIAD74/,o algorítmo NEWCND consegue ser melhor que os algorítmos existentes até então, em termos da solução final encontrada e tempo de computação requerido.

9) Algorítmo SOGA /KARNM76/

Notação utilizada na descrição dos procedimentos básicos e do algorítmo:

- Matriz de custos, C, onde c_{ij} é o custo mensal da transmissão entre os pontos i,j.

- N pontos, representando os terminais e o computador central.

- Aglomerado ("cluster") é a subrede, menos o elo que liga o terminal ao centro e o próprio centro, isto é, uma linha multiponto e os respectivos terminais. Dois aglomerados podem ser acoplados por meio do elo de menor custo entre eles.

- "Vizinho próximo" de um aglomerado é o aglomerado que pode ser acoplado a este, com o menor custo em relação aos demais. Quando o acoplamento satisfaz as restrições de capacidade das linhas, o vizinho é dito "Vizinho próximo viável".

Filosofia básica:

O algorítmo SOGA ("Second-Order Greedy Algorithms") usa um procedimento baseado no algorítmo de Esau-Williams para fazer vários mapeamentos do espaço das matrizes de custos

dentro do espaço das redes viáveis. O procedimento pode fornecer um conjunto de soluções distintas por meio de modificações sucessivas na matriz de custos, impostas por um outro procedimento. A modificação que fornecer a melhor solução passa então a ser permanentemente inscrita na matriz de custos e o processo se repete.

O algorítmo SOGA utiliza três procedimentos básicos, que são os seguintes:

a) Procedimento MEW (algorítmo de Esau-Williams modificado)

Este procedimento tem como entrada a tabela de tráfego (TRT), a matriz de custos, C, e os nós classificados e renumerados em ordens decrescente de custo do elo que liga o nó (terminal) ao centro. O centro é o nó N.

TRT é um vetor com K elementos, onde o elemento i contém o tráfego máximo numa linha multiponto contendo i terminais.

$R = c(i,N) - c(i,j)$ é a redução de custo obtida pela junção do aglomerado i ao aglomerado $j (i < j)$.

Um passo do MEW varre a parte superior à diagonal principal de C e guarda o maior valor de R viável encontrado, em termos dos índices dos dois aglomerados. O custo da rede é reduzido de R e os aglomerados são acoplados.

O acoplamento do aglomerado i ao aglomerado j requer uma modificação na parte superior à diagonal principal da j-ésima linha e coluna da matriz C, da seguinte maneira:

$$c(j,k) \leftarrow \min[c(j,k), c(i,k)] \quad , \quad j < k < N$$

$$c(k,j) \leftarrow \min[c(k,j), c(k,i)] \quad , \quad 1 \leq k < i$$

$$c(k,j) \leftarrow \min[c(k,j), c(i,k)] \quad , \quad 1 < k < j$$

Estas atualizações fornecem os custos da junção (acoplamento) do novo aglomerado j a outros aglomerados.

Quando este procedimento é chamado (ele é utilizado como subrotina) são inicializadas as seguintes variáveis e vetores: MCOST, representando o custo inicial da rede e três vetores, PV, TV, CV, contendo cada um N-1 elementos. PV identifica os terminais com aglomerados. O i-ésimo elemento é o índice do aglomerado contendo o terminal i. Quando o aglomerado j é aclopado ao aglomerado k, todos os elementos de PV que possui valor j recebe o valor k. TV é o vetor que guarda o tráfego de cada aglomerado. Quando o aglomerado j é aclopado ao aglomerado k, $TV(k) \leftarrow TV(k) + TV(j)$. O vetor CV guarda o número de terminais em cada aglomerado. Quando o aglomerado j é acoplado ao aglomerado k, $CV(k) \leftarrow CV(j) + CV(k)$ e $CV(j) \leftarrow 0$, para que j não seja futuramente considerado. Para que o aclopamento seja viável, é necessário que:

$$CV(j) > \emptyset \quad , \quad CV(k) > \emptyset \quad \text{e}$$

$$TV(j) + TV(k) \leq TRT(CV(j) + CV(k))$$

O procedimento MEW pode começar com uma rede ESTRELA ou com uma rede ÁRVORE intermediária gerada numa solução anterior. A forma da rede é implícita no estado inicial de C, PV, TV, CV e MCSOT.

O MEW termina quando, após uma varrida, não for

encontrada uma redução no custo.

O MEW passa informações, sobre as sucessivas junções de aglomerados para o SOGA, através de uma matriz JS, com dimensões N-2 por 2. A matriz JS é inicializada com zeros pelo MEW, que vai armazenando em cada linha os pares de índices dos aglomerados acoplados.

b) Procedimento INHIBIT /KARNM76/

Este procedimento força a subrotina MEW a gerar, sucessivamente, um conjunto de soluções distintas. A técnica consiste em fazer inibições sucessivas das funções de pares de aglomerados anotados na matriz JS. Aquela inibição que fornecer a solução de menor custo via a subrotina MEW será feita permanente e o procedimento é reiterado. O procedimento INHIBIT termina quando não puder mais encontrar redução de custo.

e) Procedimento JOIN /KARNM76/

Quando do cálculo da matriz C, uma matriz NA é formada, possuindo as dimensões N por 2. NA(i,1) contém o índice do nó ao qual o nó i pode ser conectado com o menor custo. NA(i,2) contém o índice do nó indexado mais alto, ao qual o nó i pode ser conectado com o menor custo. Visto que NA(i,2) > i pode-se dizer que NA(i,2) é mais próximo do centro que i.

Existe um vetor B, de dimensão N-1, cujo elemento i é usado para guardar o vizinho mais próximo viável do aglomerado i. O vetor B é inicializado com NA(*,2).

Quando o nó i e seu vizinho mais próximo não são

juntados,o procedimento JOIN junta-os temporariamente e chama a subrotina MEW. Isto gera uma nova solução. Se este custo é menor, então o custo i e NA(i,1) são guardados pelo JOIN.

O JOIN vai juntando sucessivamente NA(i,1) e NA(i,2) para i, que ainda não tenha sido juntado, buscando a melhor solução.

O procedimento JOIN é menos poderoso que o INHIBIT, porém gasta menos tempo, visto que, muitos dos vizinhos mais próximos já foram juntados.

Algorítmo SOGA /KARNM76/

1) Obtenha a localização e o tráfego de cada terminal e a localização do centro.
2) Calcule o custo de conexão dos terminais ao centro.
3) Classifique os pontos em ordem decrescente de custo de conexão ao centro e reindexe eles nesta ordem, guardando os índices originais.
4) Calcule a matriz de custos e a matriz dos vizinhos.
5) Obtenha a tabela de tráfego (restrições de tráfego).
6) Forme cópias(duplicatas) dos vetores usados para inicializar PV, TV, CV e B no MEW e a duplicata da matriz de custos.
7) Chame a subrotina MEW e guarde os resultados, ou seja, os valores do custo, vetor PV e JS.
8) Execute outro laço do SOGA. Isto necessitará da aplicação dos procedimentos JOIN e INHIBIT em sucessão,ou INHIBIT somente, como desejado.
9) Se foi encontrada uma melhoria na solução, faça a no-

va inibição ou acoplamento permanente, restaure as estruturas (vetores e matrizes), chame a subrotina MEW e volte ao passo 8.

10) Usando o vetor PV que corresponde a melhor solução obtida do SOGA, aplique um algorítmo de MST para cada aglomerado, guardando os ramos usados e conecte o nó correspondente ao índice do aglomerado (nó mais próximo do centro) ao centro.

11) Imprima os resultados desejados, fazendo a recuperação dos índices originais dos nós.

Em/KARNM76/, existem tabelas que mostram a eficiência deste algorítmo em relação a vários conjuntos de nós.

Nota-se que este algorítmo tem a vantagem de considerar um bom número de topologias diferentes, antes de decidir pela melhor solução.

Observações sobre os algorítmos

Algumas observações importantes podem ser feitas a respeito dos algorítmos do ítem II.2.1.1, quais sejam:

- Nenhum deles prevê a possibilidade de ligações pré-existentes na rede a ser definida, com exceção do MEW.

- Eles não prevêem a possibilidade de serem reutilizados para adição de novos nós e elos à rede definida.

- Os algorítmos fazem certas simplificações que,de certa forma, fogem ao caso prático, como por exemplo: tráfego estável, ou seja, que não varia com o tempo e/ou com variança alta; usar a distância entre dois pontos como se fosse o custo entre eles; considerar a capacidade igual para todos os elos.

II.2.1.2. Redes Centralizadas Com Dois ou Mais Níveis de Ligação

Nesse tipo de rede existem 2 ou mais níveis de concentração, ou seja, os terminais são ligados a concentradores que por sua vez podem estar ligados a outros concentradores, ou comutadores ("switches"), até chegar ao último nível onde serão ligados ao computador central. Existem dois tipos de redes derivadas deste tipo: Multicentro multiestrela e Multicentro multiponto.

II.2.1.2.1. Redes Multicentro-Multiestrela

Neste esquema os terminais são ligados por uma rede de estrela, ponto a ponto aos concentradores e estes ao computador central numa rede estrela, ponto-a-ponto também.

EXEMPLO:

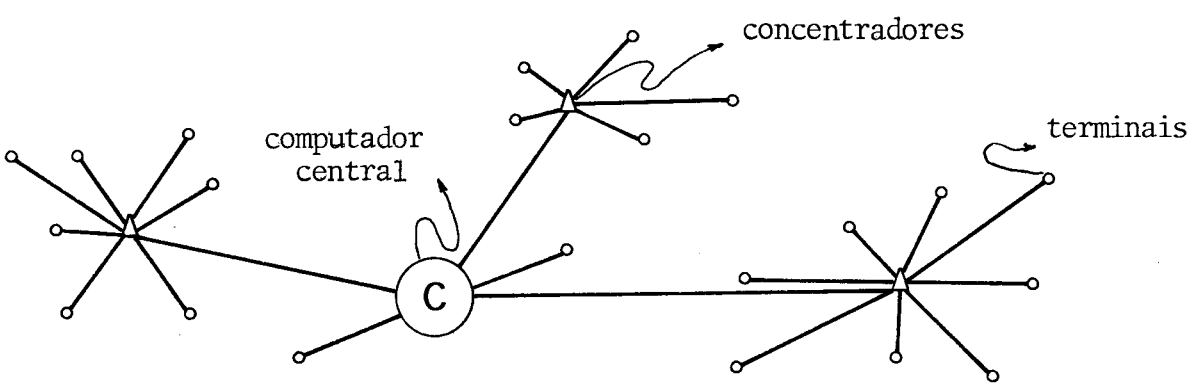


FIGURA 3 : Rede com configuração MULTICENTRO-MULTIESTRELA

O uso de concentradores e multiplexadores é uma opção para a redução do custo da rede. Surgem, porém, as seguintes questões relativas ao uso de concentradores: Quantos e quais escolher?; Onde colocar cada um deles?; Quais os terminais que devem ser conectados a cada um?.

Vejamos então, alguns algorítmos que definem este tipo de rede:

- Algorítmo de MARTIN /MARTJ72, cap.42/
- Algorítmo ADD /SCHWM76, cap.10/
- Algorítmo DROP /SCHW76, cap.10/

* * *

1) Algorítmo de MARTIN /MARTJ72/

Antes de descrevermos o algorítmo, é bom observarmos as seguintes considerações: Martin apresenta as seguintes regras para o cálculo do custo das conexões:

1 - (Custo do concentrador + custo da linha de alta velocidade do concentrador para o computador central) x $\left\lceil \frac{M}{N_L} \right\rceil$ onde:

M = número de terminais

N_L = número de linhas de baixa velocidade(terminal a concentrador) que um concentrador pode lidar.

2 - (Custo do concentrador + custo da linha de alta velocidade do concentrador ao computador central) x $\left\lfloor \frac{M}{N_L} \right\rfloor$ + (custo da linha de baixa velocidade do local do concentrador para o

computador central) x $(M-N_L \left\lfloor \frac{M}{N_L} \right\rfloor)$

Obs.: $\left\lceil \frac{M}{N} \right\rceil$ significa que o resultado da divisão deve ser aproximado para o maior inteiro, mais próximo de $\frac{M}{N}$.

$\left\lfloor \frac{M}{N} \right\rfloor$ significa que o resultado da divisão deve ser aproximado para o menor inteiro mais próximo de $\frac{M}{N}$.

Martin observa que a localização exata do concentrador não influi muito na diminuição do custo da rede, podendo variar de 1 a 10% do custo total da mesma, a depender do deslocamento em relação ao local "ideal". Por isso, uma boa estratégia é colocar concentradores em locais coincidentes com os dos terminais, onde haja grande densidade de tráfego.

O algorítmo de MARTIN se propõe a encontrar a melhor localização para os concentrados e quais os terminais que devem ser ligados a cada concentrador. É dividido em duas partes:

PARTE 1:

1) Para cada tipo de concentrador disponível, faça o seguinte:
   1.1) Considere a lista de todas as localizações dos terminais;
   1.2) Para cada localização da lista, faça o seguinte:
      1.2.1) Assuma que o concentrador está localizado nesta localização;

1.2.2) Avalie o potencial de custo ganho na conexão de cada terminal ao computador central, via este concentrador;

1.2.3) Selecione os terminais com ganho máximo, até saturar o concentrador;

1.2.4) Adicione o total ganho (se houver);

1.2.5) Recorde a localização do concentrador com maior ganho;

1.3) O ganho máximo é > Ø?

1.3.1) Sim: Deleta as localizações dos terminais conectados a este concentrador, da lista; Vá para 1.2

1.3.2) Não: Recorde o conjunto dos locais dos concentradores; Vá para PARTE 2.

PARTE 2:

1) Pegue os concentradores e suas localizações estabelecidas na PARTE 1 e faça o seguinte, começando com a lista de todas as localizações dos terminais:

1.1) Pegue na lista das localizações aquela mais distante do centro;

1.2) Conecte ela ao melhor concentrador não saturado ou diretamente ao computador central, a depender de qual das duas ligações seja a de menor custo;

1.3) Remova esta localidade da lista;

1.4) Foi a última localidade da lista?

1.4.1) Sim: Imprima os resultados; repita isto para cada tipo de concentrador, para que

possam ser comparados os custos das soluções;

1.4.2) Não: Vá para 1.1;

Este algorítmo fornece várias soluções, cada uma delas porém, só utiliza um único tipo de concentrador. Nos casos práticos, geralmente são usados vários tipos de concentradores, como observa o próprio Martin.

* * *

Uma boa definição do problema da conexão de terminais a concentradores e estes ao computador central é a apresentada em /SCHWM76/, como sendo de programação linear inteira, sob a seguinte formulação:

Dado:

$\{T_1, T_2, \ldots, T_n\}$ conjunto dos terminais,

$\{S_1, S_2, \ldots, S_m\}$ conjunto dos concentradores

$S_\emptyset$ computador central

f_j = custo fixo do concentrador j

e = número máximo de terminais por concetrador

Obs.: $f_\emptyset = \emptyset$ e $e_\emptyset \geq n$

c_{ij} = custo da conexão do terminal i ao concentrador j, para i=1,2,...,n ; j=1,2,...,n.

$$x_{ij} = \begin{cases} 1 \text{ se } T_i \text{ está conectado a } S_j \\ \emptyset \text{ caso contrário} \end{cases}$$

$$y_j = \begin{cases} 1 \text{ se } \sum_{i=1}^{n} x_{ij} > \emptyset \\ \emptyset \text{ caso contrário} \end{cases}$$

Determinar:

y_j's e x_{ij}'s de maneira a minimizar a função de custo total:

$$Z = \sum_{j=\emptyset}^{m} y_j f_j + \sum_{i=1}^{n} \sum_{j=\emptyset}^{m} x_{ij} c_{ij}$$

Sujeito às restrições

$$\sum_{j=\emptyset}^{m} x_{ij} = 1, \quad p/i = 1,2,\ldots,n$$

$$\sum_{i=1}^{n} x_{ij} \leq e y_j, \quad p/j = 1,2,\ldots,m$$

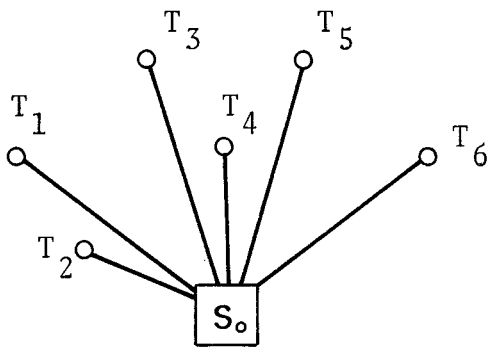
$$\text{com } y_j \in \{0,1\} \text{ e } x_{ij} \geq 0$$

A seguir veremos dois algorítmos que resolvem o problema supra definido.

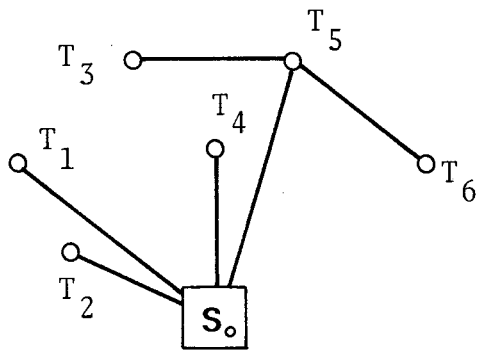
2) Algorítmo ADD /SCHWM76/

A filosofia básica deste algorítmo é a seguinte: Conecta inicialmente todos os terminais ao computador central, sem utilizar concentradores. Depois, os terminais são desligados do computador central e ligados ao concentrador mais próximo, que satisfaça as restrições. É escolhido um terminal a cada iteração, de maneira a produzir o maior decremento na função de custo, Z. Prossegue desta maneira até não poder mais ser feita redução na função de custo.

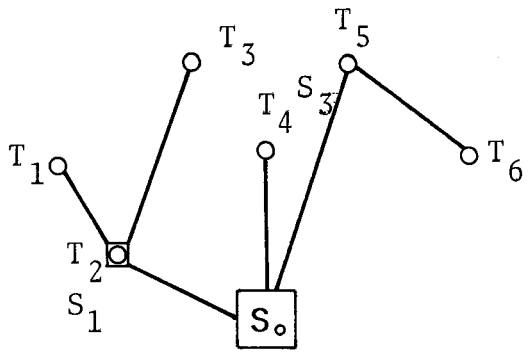
A figura 4 mostra o comportamento do algorítmo.



Inicialização



1ª. iteração



2ª. iteração - solução final

FIGURA 4 : Comportamento do ADD

3) Algorítmo DROP /SCHWM76/

O algorítmo DROP começa com todos os terminais ligados a concentradores. A partir daí, ele verifica a cada iteração se, desligando um determinado terminal do concentrador e ligando ao computador central, o valor de Z (função de custo) diminui. Se sim, o terminal é ligado ao computador central; se não, ele continua ligado ao concentrador. O terminal escolhido em cada iteração é aquele que possa causar a maior diminuição do valor de Z. Prossegue desta maneira até que todos os terminais tenham sido considerados.

O DROP tem o mesmo objetivo do ADD e geralmente encontra o mesmo resultado.

Vejamos o comportamento do DROP na figura 5.

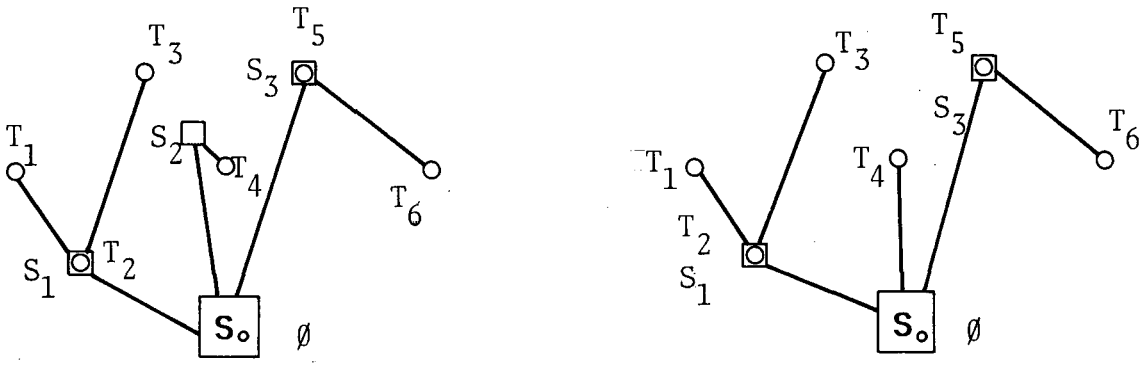


FIGURA 5 : Comportamento do DROP

* * *

Comentários importantes a respeito dos algorítmos ADD e DROP foram feitos em /BOORR77/, quais sejam:

- Originalmente o programa possui 2^m diferentes combinações para locais dos concentradores. Usando um número e concentradores, o número de combinações é $m!/(m-e)!$

- O algorítmo ADD consta de aproximadamente mp aplicações do algorítmo de conexão de terminais ao concentrador, onde p é o número de concentradores encontrados pelo algorítmo.

- Um algorítmo ótimo que investigue todas as combinações possíveis, necessitaria de m^p implementações do algorítmo de conexão de terminais a concentrador.

- O algorítmo ADD não é ótimo, porém pode melhorar bastante com algumas modificações.

- Pode-se fazer uma variação do algorítmo ADD para

torná-lo bastante rápido, mas que encontrará uma solução mais distante da ótima que a versão original.

- As observações feitas para o ADD são válidas também para o algorítmo DROP.

Observações sobre os algorítmos

Sobre os algorítmos do ítem II.2.1.2.1, pode-se observar que:

- Todos eles utilizam um único tipo de concentrador, enquanto que, nos casos práticos são usados geralmente vários tipos de concentradores combinados.

- Nenhum deles considera a possibilidade de ligar os concentradores por meio de linhas multiponto, o que já é viável hoje em dia desde que não acarrete num tempo de atraso muito grande. Esta possibilidade pode resultar em redes com um custo menor em certos casos.

II.2.1.2.2. Redes Multicentro Multiponto

Neste tipo de rede os terminais são ligados aos concentradores por meio de linhas multiponto e estes podem ser ligados a outros concentradores ou ao computador central por meio de linhas ponto-a-ponto ou multiponto.

EXEMPLO:

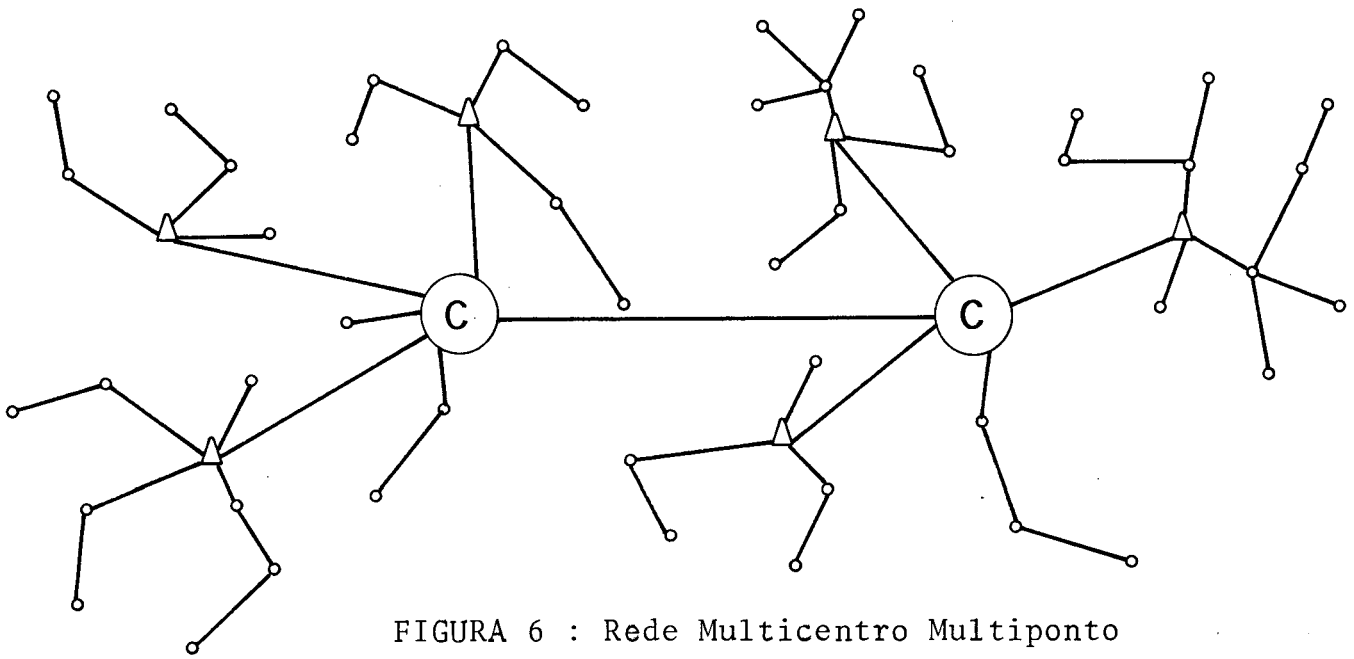


FIGURA 6 : Rede Multicentro Multiponto

Na definição desse tipo de rede, podemos destacar os seguintes algorítmos:

- Algorítmo de MARTIN /MARTJ72, cap.43/
- Algorítmo COM /GREGP77/
- Algorítmo NEWCLUSTER /DYSAH78/

1) Algorítmo de MARTIN /MARTJ72/

O objetivo do algorítmo é estabelecer o desenho da rede multiponto de menor custo, baseado no algorítmo citado no ítem II.2.1.2.1-1 (pág.26), com a modificação na parte de ligação dos terminais aos concentradores.

Martin caracteriza o trabalho do planejamento de uma rede como sendo dividido nas seguintes fases:

- Seleção dos concentradores.
- Determinação da melhor localização para cada concentrador.

- Estabelecimento dos desenhos das redes multiponto de custo mínimo para ligar os terminais aos concentradores.

O algorítmo de MARTIN está dividido em duas partes:

PARTE 1:

Aplicação do algorítmo citado no ítem II.2.1.2.1-1(pág.27/28), para seleção e localização dos concentradores.

PARTE 2:

Usa o algorítmo do ítem II.2.1.1-5(pág.12) para conectar os terminais aos concentradores.

2) Algorítmo COM /GREGP77/

Inicialmente são necessárias algumas considerações sobre convenções feitas pelo COM.

Mc Gregor e D. Shen chamam de "facilidade de acesso" os pontos considerados geralmente como concentradores.

O problema é formulado como de determinação dos locais das facilidades de acesso genéricos, GAF's ("generic access facility"), para obter a mais econômica ligação dos nós (usuários) aos pontos de conexão dos recursos, RESCOP("resource connection point").

Os nós são conectados por linhas multiponto com restrições de número máximo de nós por linha. Os GAF's são limitados por número máximo de nós e possuem o custo proporcional a este número.

O algorítmo possui 4 passos básicos, quais sejam:

1) Simplificação do problema para ponto-a-ponto pela substituição dos aglomerados de nós ("clusters") por nós centro de massa simples, COM ("center-of-mass").

2) Partição do conjunto reduzido de nós COM pela aplicação do algorítmo ADD, resultando em um dos nós COM selecionado para GAF.

3) Seleção de um dos nós originais para ser o local real para GAF em cada partição, verificando qual o nó mais próximo do nó COM selecionado para GAF anteriormente.

4) Aplicação de um algorítmo para achar o melhor desenho das linhas multiponto em cada partição, com o nó real selecionado para GAF sendo o centro.

Definição formal do problema /GREGP77/

Dado:

1) A - conjunto dos nós i, p/i=2,...,N (cidades)

2) Nó 1 - RESCOP

3) H - conjunto das possíveis cidades GAF

4) w_{max} - capacidade da linha (nós/linha)

5) c_{max} - capacidade dos GAF's (nós/GAF)

6) $d_i(j)$ - custo da conexão da cidade j para cidade i

7) taxa - custo do GAF

Determinar:

1) A rede de custo mínimo para conectar os terminais aos GAF's e estes ao RESCOP.

Restrições:

1) Número de nós por linha $\leq w_{max}$

2) Número de nós por GAF $\leq c_{max}$

Obs.: - H e A podem ou não serem disjuntos, a depender da aplicação.

- O custo da conexão do GAF do nó i para o RESCOP é igual ao custo da conexão do nó i para o RESCOP.

Algorítmo COM /GREGP77/

A) Simplificação por aglomeração:

Trata-se do agrupamento dos nós em aglomerados (clusters) que passam a ser considerados como um único nó, localizado no centro de massa do aglomerado. O problema fica então reduzido a um problema de rede ponto-a-ponto. Os aglomerados são limitados em número de nós que podem compartilhar uma linha (número máximo de nós $\leq w_{max}$).

A formação dos aglomerados é feita de dois em dois nós, que são substituídos por um novo nó, localizado no centro de massa dos anteriores, com peso igual a soma dos dois, desde que não violem a restrição de capacidade da linha. Os nós que não puderem ser agrupados são ignorados. O processo se repete até não haver mais nós a considerar.

B) Particionamento:

Aplica o algorítmo ADD para examinar o benefício ganho em colocar um GAF em cada um dos nós restantes, nós

COM's. O benefício é estimado em termos de ganho de custo ao ligar um COM a outro COM, ao invés de ligá-lo ao centro, respeitando a restrição de capacidade dos GAF's:

$$r_i = \sum_{j \in B_i} s_{ij} * w_j - d_{1(i)} * \gamma$$

Onde:

r_i = estimativa do benefício, relativa ao nó i

B_i = conjunto de nós associados com o GAF no nó i

s_{ij} = ganho da conexão do nó j ao GAF i, ao invés do RESCOP

w_j = peso do nó j

$d_{1(i)}$ = custo da conexão do GAF no nó i ao RESCOP

γ = parâmetro usado para enfatizar a proximidade do GAF ao RESCOP.

O COM que tiver maior estimativa heurística de ganho, será selecionado para GAF.

C) Otimização Local:

Os k nós mais próximos de cada COM selecionado para GAF, são avaliados heurísticamente para que seja selecionado um local real para cada GAF. São selecionados os dois nós que produzem maior economia. Dessa forma, os GAF's ficam localizados em locais de nós originais.

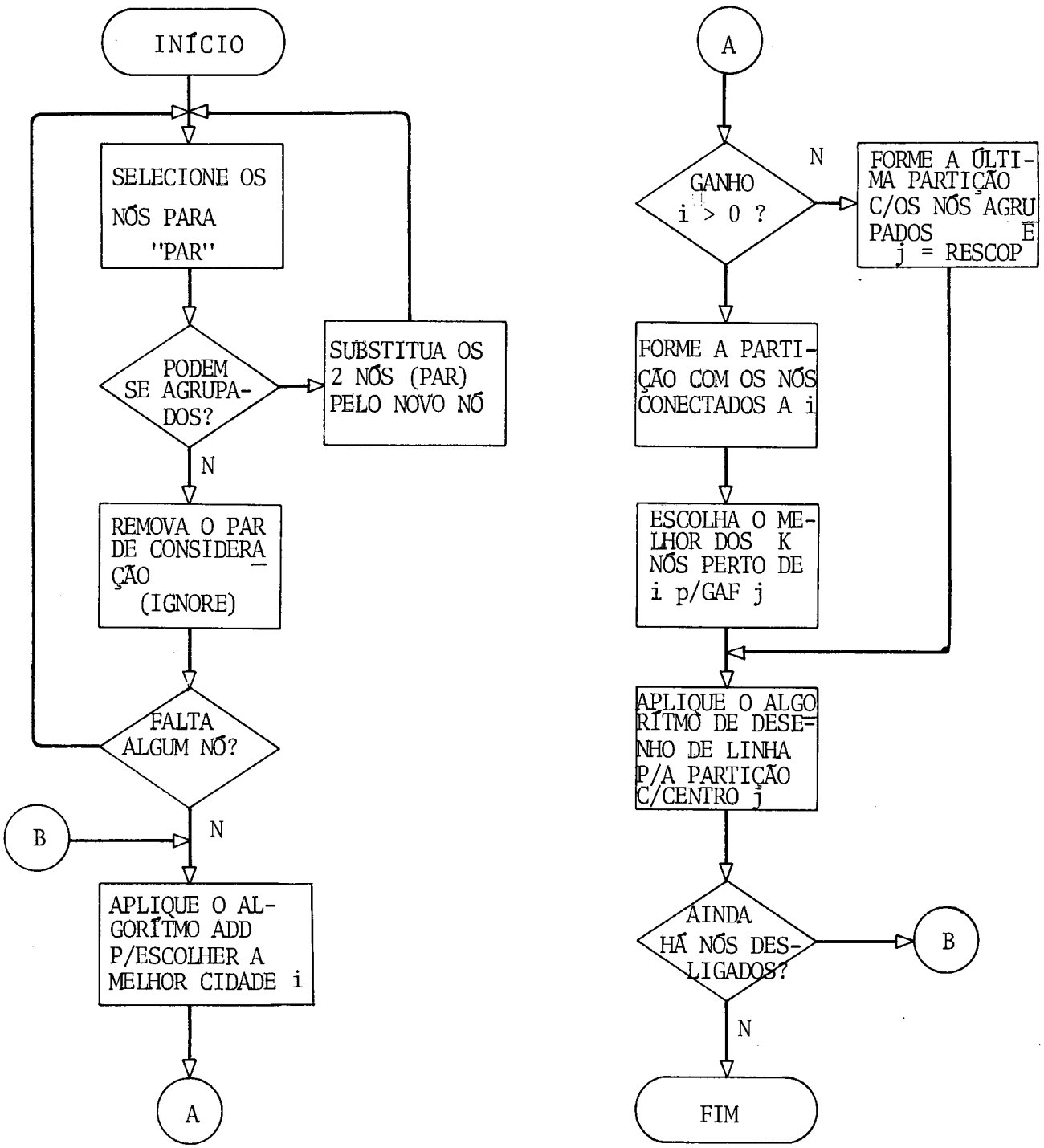
D) Desenho das linhas multiponto:

Os nós originais são restaurados e cada partição fica formada pelos nós e o GAF. Um algorítmo de desenho de

linha é aplicado em cada partição, tendo o GAF dela como centro. Os nós que não forem ligados a nenhum GAF são diretamente ligados ao RESCOP.

O fluxograma a seguir, resume o algorítmo.

Fluxograma do algorítmo COM /GREGP77,fig.1/



Mc Gregor e D.Shen apresentam alguns resultados de análises comparativas entre o algorítmo COM e ATD (Woo e Tang, 1973) para diversos números de nós com diversas restrições, os quais mostram que o COM encontra melhores soluções, gasta menos tempo de computação e, este tempo cresce muito mais lentamente com o número de nós que o ATD.

3) Algorítmo NEWCLUSTER /DYSAH78/

Trata-se de um algorítmo heurístico para definição topológica de redes de teleprocessamento, centralizadas com configuração Multicentro-Multiponto, com dois ou mais níveis de ligação:terminais a concentradores e concentradores ao computador central.

Dysart e Georganas definem o problema a ser resolvido da seguinte maneira:

Dado:

i) Conjunto dos terminais, localização e tráfego de cada um deles.

ii) Localização do computador central - UCP

iii) Função de custo para as linhas de alta e baixa velocidade e custo fixo do concentrador.

iv) Restrições:

a) Número máximo de terminais por linha, N_{max}

b) Tráfego máximo por linha, τ_{max}

c) Capacidade máxima do concentrador, T_{max}

d) Um concentrador qualquer só pode estar localizado num local coincidente com o de um terminal.

Encontrar:

A rede de custo mínimo que envolve a determinação da melhor localização para os concentradores, o número de concentradores e o desenho da rede de custo mínimo.

A filosofia básica do algorítmo é a seguinte: Inicialmente divide os terminais em várias regiões utilizando um algorítmo de agrupamento ("clustering"), que fornece os pontos que são candidatos a concentradores, os quais são os mais próximos dos demais em cada partição. Cada aglomerado ("cluster") é dividido em regiões menores chamadas "super-nós", sendo cada "super-nó" representante de uma linha multiponto, satisfazendo as restrições impostas à linha. Cada super-nó é ligado a um concentrador ou a um super-nó que esteja ligado a um concentrador. É feita uma otimização local que elimina os concentradores anti-econômicos e liga os super-nós, que estavam conectados a ele, a outros concentradores. Quando já não for mais econômico eliminar concentradores é feita a partição final da rede, considerando o conjunto de concentradores restante e seus respectivos locais. É feita então a otimização de cada partição e os concentradores são então conectados a UCP.

Os passos básicos do algorítmo são os seguintes:

1) Determine os K terminais mais próximos de cada um dos N terminais. Faça uma lista dos K terminais mais próximos para cada um dos terminais. Os terminais que tiverem maior frequência nas listas serão os candidatos a concentradores.

2) Partição da rede:

   a) Associe os terminais aos concentradores mais próximos deles.

   b) Modifique as subredes de concentradores que possuem suas restrições violadas, através de eliminação de terminais.

   c) Religue os terminais eliminados no passo b a outro concentrador que satisfaça as restrições.

3) Para cada partição:

   a) Aplique um algorítmo de desenho de linha multiponto (Esau-Williams, no caso) para ligar os terminais ao computador.

   b) Sendo cada linha multiponto considerada um super-nó, determine os parâmetros destes super-nós (número de terminais, tráfego total, custo de realocação, super-nó mais próximo, etc), dos concentradores (número de super-nós, tráfego total, conjunto dos super-nós, custo de realocação, custo fixo do concentrador, etc) e o ganho econômico dos concentradores, ou seja, a redução de custo na sua eliminação, com a consequente religação de seus super-nós a outros concentradores.

4) Determine que concentrador, C_{j*} , possui o maior ganho econômico.

   $G_{j*} = MAX[G_j]$, onde G_j é o conjunto dos ganhos econômicos dos concentradores.

   Se $G_{j*} \leq \emptyset$, vá para 8; de outra forma, vá para 5;

5) Se C_{j*} foi o último concentrador analisado, então vá para 7; de outra forma, vá para 6;

6) Verifique se é viável a religação dos super-nós de C_{j*}. Se alguma modificação ocorre, altere o ganho de G_{j*} e vá para 4; de outra forma, vá para 7;

7) Atualização da rede:

   a) Corte C_{j*} e decremente o custo por G_{j*} unidades;

   b) Religue os super-nós de C_{j*} aos respectivos vizinhos mais próximos viáveis;

   c) Altere os parâmetros dos super-nós e dos concentrados afetados pela atualização;

   d) Ajuste o ganho econômico dos concentradores afetados pela atualização;

   e) Vá para 4;

8) Registre a topologia resultante e o custo total;

9) Faça o custo igual a zero e repita os passos 2 e 3 (a & b)

10) Escolha qual das duas topologias tem o menor custo, imprima-a como resposta e termine.

Dysart e Geoganas mostram a análise de desempenho do algorítmo NEWCLUSTER, com vários tamanhos de redes (em termos de número de nós) gerados randômicamente, com várias restrições diferentes. Eles comparam os resultados do NEWCLUSTER com os resultados dos algorítmos ADD e DROP e concluem que o custo final da rede definida pelos algorítmos ADD e DROP são 3% e 9% maiores que o do NEWCLUSTER, respectivamente, e que o

tempo de execução do NEWCLUSTER cresce muito mais lentamente com o número de nós que os outros dois.

* * *

Observações sobre os algorítmos

As observações feitas para os algorítmos do ítem II.2.1.2.1(pág.33) são válidas também para os algorítmos do item II.2.1.2.2.(pág.33/34).

II.2.2. Redes Distribuidas

As principais características das redes distribuidas são as seguintes:

- Os recursos são distribuidos entre os nós da rede, ao invés de ficarem centralizados num único ponto (computador central).

- As mensagens possuem vários caminhos alternativos para atingir o destino.

- A rede é dividida, basicamente, em dois níveis hierárquicos, a rede de alto nível e a rede de acesso local, que por sua vez podem se dividir em outros níveis hierárquicos.

A figura 7 ilustra uma configuração típica.

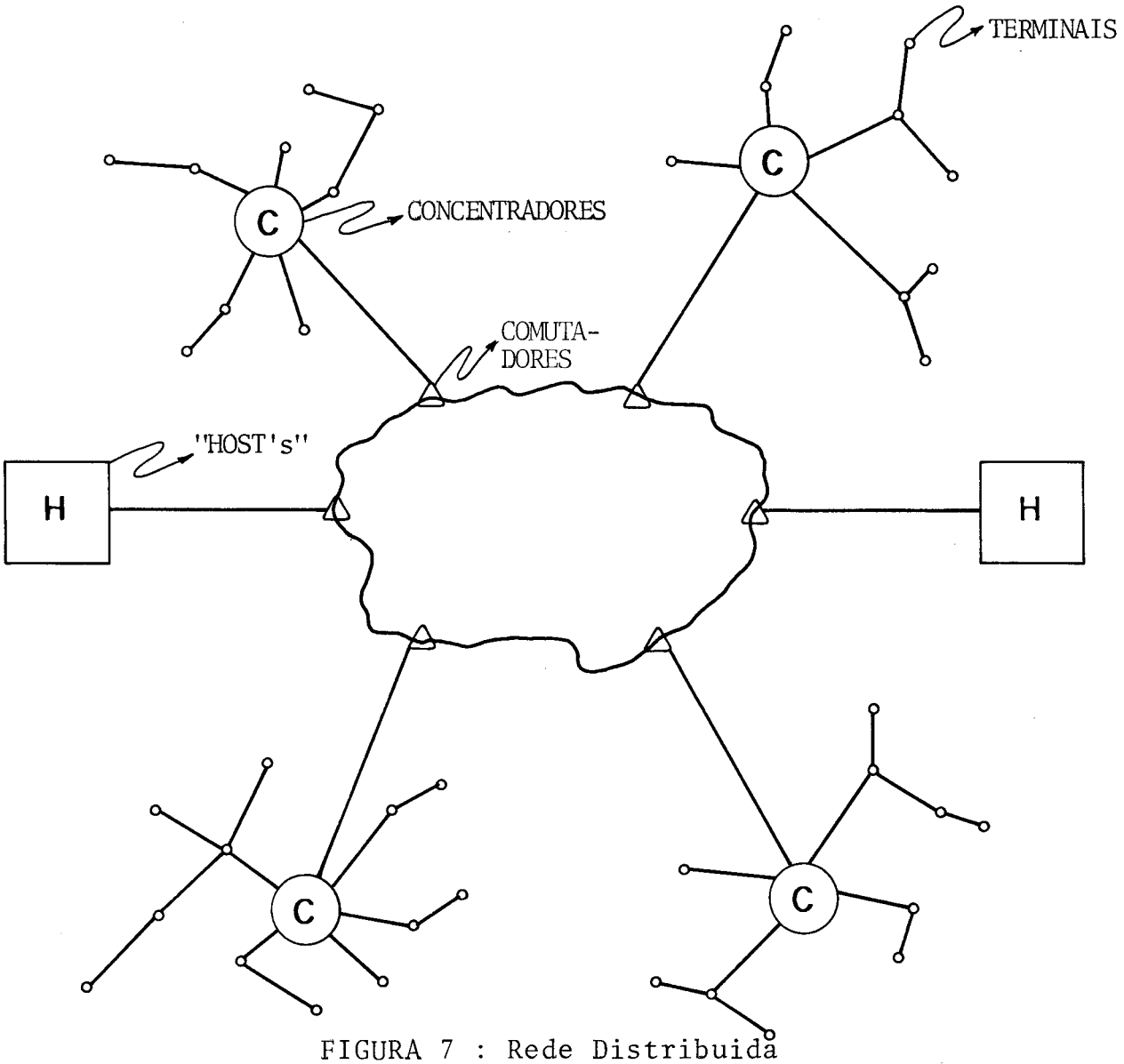


FIGURA 7 : Rede Distribuida

Geralmente, os computadores anfitriões ("host") armazenam os recursos de maior porte e resolvem os grandes problemas das redes de acesso local. Os anfitriões e a subrede de comunicação formam a rede de hierarquia mais alta, enquanto os concentradores e terminais, com as linhas multiponto (ou ponto-a-ponto) formam a rede de hierarquia mais baixa (rede de acesso local).

É comum também as redes de acesso local possuir tráfego centralizado e a rede de nível mais alto possuir tráfego distribuido.

Neste tipo de rede o problema topológico consiste em: determinar a rede de acesso local de custo mínimo, a rede de alto nível de custo mínimo e a localização dos pontos de conexão entre as duas redes.

Fica bem fora da realidade um tratamento exclusivamente topológico numa rede distribuida, visando apenas minimizar os custos, sem levar em consideração outros parâmetros importantes para o bom funcionamento da rede, quais sejam: tempo de resposta, fluxo médio de dados na rede ("throughput") e a confiabilidade da rede /BOORR77/.

Os trabalhos a serem considerados neste tipo de rede serão os seguintes:

- Algorítmo NAF /CHOUW78/
- Algorítmo CUT SATURATION (M. Gerla, H. Frank, W. Chou, 1974)/SCHWM76, cap.10/
- O estudo feito em /BOORR77/
- O estudo feito em /GERLM77/
- Algorítmo COG /SHARR77/

1) Algorítmo NAF /CHOUW78/

Este algorítmo encontra o número e a melhor localização para os NAF's , ("Network Access Facilities"), Facilidades de Acesso da rede que podem ser concentradores, multiplexadores, comutadores, etc., os quais fazem o aclopamento entre a rede de alto nível e a rede de baixo nível (rede de acesso local). A rede de alto nível é também chamada de "espinha dorsal", tendo grande influência no custo total de rede.

Trata-se de um algorítmo heurístico que leva em conta outras técnicas heurísticas já existentes em outros algorítmos (ADD,DROP, etc).

O NAF possui os seguintes passos básicos:

1) Seleciona o conjunto de cidades candidatas a localização dos NAF's, entre as cidades onde estão localizados os terminais. Se o conjunto de terminais for grande ele é reduzido a um conjunto menor pela aplicação da técnica de aglomeramento que substitui um conjunto de terminais que estejam próximos entre si por um único terminal localizado no centro de massa do aglomerado.
2) Define a rede de acesso local por meio da aplicação de uma função no conjunto de terminais e NAF's, a qual conecta subconjuntos de terminais a cada NAF, em forma de ÁRVORES, pela aplicação do algorítmo UNIFICADO (item II.2.1.1-4 pág.11) para cada NAF e os terminais próximos dele.
3) Definição da rede "espinha dorsal" pela aplicação de uma função que conecta os NAF's e as UCP's.
4) Avalia a localização dos NAF's aplicando diversas técnicas, ADD, DROP, EXCHANGE e MERGE, escolhendo a rede que fornecer o menor custo resultante da aplicação das técnicas acima citadas. Esta rede escolhida possuirá as localizações dos NAF's definitivas e os desenhos em forma de ÁRVORE para conectar os terminais aos NAF's.

A principal vantagem deste algorítmo é que ele considera várias técnicas de localização dos NAF's, podendo optar pela melhor, a depender do caso específico da rede em questão, enquanto os outros seguem apenas uma técnica, podendo encontrar uma solução distante da ótima num caso patológico. Pode-se concluir portanto que o algorítmo NAF é mais flexível que os existentes até então e tende a encontrar melhores soluções, embora seja mais demorado em termos de tempo de processamento, devido ao fato dele considerar várias técnicas antes de optar pela melhor definição.

2) Algorítmo CUT SATURATION (M. Gerla, H. Frank, W. Chou, 1974) /SCHWM76, cap.10/

Nesta versão, o algorítmo parte das seguintes condições iniciais:

i) Capacidade dos elos conhecida é a mesma para todos eles.

ii) Tráfego médio gerado por cada par de nós é conhecido e o mesmo para todos eles.

iii) Localizações dos nós conhecidas.

Imposições:

i) tempo de atraso máximo permitido

ii) confiabilidade da rede

iii) capacidade máxima dos elos.

A filosofia do algorítmo é encontrar iterativamente a rede distribuida de menor custo, para um determinado número de bits por seg., gerados em média por um par de nós qualquer da rede ("throughput") sujeito às restrições.

O algorítmo possui os seguintes passos básicos:

(Obs.: é interessante o acompanhamento simultâneo do fluxograma apresentado na página 51).

1. Rota: Utilizando um algorítmo de rota ("Routing") o algorítmo encontra os fluxos ótimos dos elos para uma definição de rede dada, que minimiza o atraso médio.

2. Determinação do conjunto saturado: Quando os fluxos ótimos são encontrados, os elos são ordenados de acordo com a utilização. Os elos são então removidos um a um, na ordem de utilização. O conjunto mínimo, que desconecta a rede se um de seus elos é removido e que possui a capacidade dos elos esgotada, é chamado conjunto saturado ("Satureted cutset").

3. Adicionamento : Adiciona os elos de custo mínimo que desviará o tráfego do conjunto saturado. Para desviar o tráfego eficientemente os nós de um lado são ligados a nós do outro lado que estejam relativamente longe. Como a ligação entre nós distantes são caras, o compromisso de "distância 2" é usado: os nós pertencentes aos dois últimos elos removidos do conjunto de nós são selecionados como candidatos a possível ligação.

4. Deleção : Neste passo o algorítmo elimina iterativamente elos que estejam aumentando o custo da rede. A cada iteração um elo é removido, sendo escolhido o elo que é mais dispendioso e menos custoso, de acordo com o seguinte critério:

Maximize: $E_i = D_i \times \frac{C_i - f_i}{C_i}$, onde:

D_i é o custo do elo i

C_i é a capacidade

f_i é o fluxo

5. Perturbação: Estando o número médio de bits por segundo satisfeito, o algorítmo rearruma os elos, usando os passos 3 e 4, de maneira a reduzir o custo. A escolha do passo 3 ou 4 é feita a depender de:

Se $R < R_{min}$ então vá para 3

Se $R > R_{max}$ então vá para 4

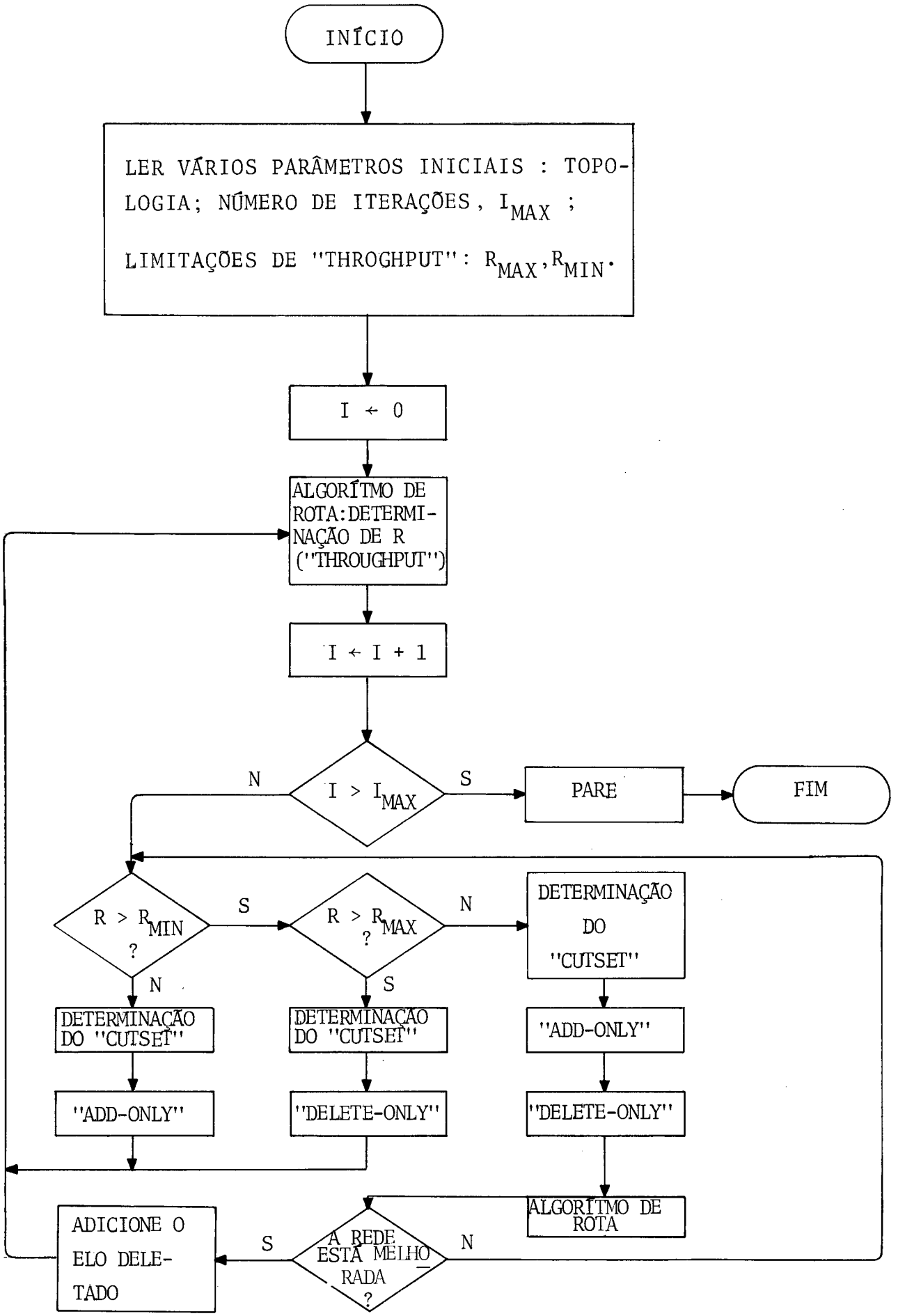
Sendo R o número médio de bits por segundo fluindo na rede.

O fluxograma pode ser visto na página 51.

A complexidade C do algorítmo, é proporcional ao quadrado do número de nós:

$C = \delta(NA)^2$, onde NA é o número de nós.

Fluxograma /SCHWM76, fig.10-8/



3) O ESTUDO DE BOORR 77

No estudo apresentado em /BOORR77/, a Arquitetura da rede (número de níveis e tipo de acesso em cada nível) é citada como o primeiro problema a ser atacado no projeto de uma grande rede. Segue-se então, uma explanação sobre as alternativas para implementação de tais tipos de redes:

- A. Alternativas de estratégias da rede

Eles assumem que a rede de alto nível usa a técnica de chaveamento de pacotes ("packet switching"), por ser comprovado que esta técnica é a mais indicada em termos de custo e confiabilidade, para comunicação distribuida de dados.

Com a concepção de chaveamento de pacotes, surgem várias alternativas de arquitetura para a rede, das quais eles citam as seguintes:

1) Alternativas de linha

As alternativas são a de usar elos terrestres somente ou elos combinados, terrestres e via satélite. Nos elos via satélite, existem as opções de ponto-a-ponto, multiplexação em tempo ou em frequência. A avaliação de elos terrestres pode ser feita sobre uma grande variedade de serviços oferecidos: multiponto, ponto-a-ponto, multiplexadores, etc.

2) Alternativas de nós

O problema aqui é a escolha de equipamentos que tenham a capacidade de suportar o fluxo de dados da rede. Existem alternativas para se juntar vários nós tradicionais em "super-nós" ou a implementação de comutadores com multiprocessadores,que é mais "estética" que a de aglomeramento ("super-nó"), mas tem limitações próprias, quais sejam, custo de de-

senvolvimento de "software" novo, integração a uma rede contendo nós tradicionais e pouca redundância.

3) Alternativas de estruturas topológicas

Para redes de alto nível pequenas ou médias (até 20 ou 30 nós) a estrutura típica é homogênea com "software" e "hardware" idênticos em todos os nós. Para redes grandes, a alternativa de usar dois níveis hierarquicos pode reduzir o custo.

Estando selecionada a arquitetura da rede, o problema a seguir é a determinação da topologia que minimiza o custo das linhas e dos nós, com a arquitetura escolhida. Para o caso de vários níveis o problema pode ser dividido em 4 subproblemas:

a) Aglomeração preliminar das localizações dos usuários
b) Seleção das localizações dos nós
c) Definição da rede de alto nível
d) Definição da rede de acesso local.

- B. Técnicas de definição da rede

Segundo Boorstyn e Frank, as técnicas de definição da rede são geralmente baseadas na aplicação de uma família de procedimentos de otimização, chamada "trocas de nós ou ramos" ("node or branch exchanges"). A filosofia básica do método é a seguinte: a partir de uma definição inicial feita pelo homem ou por um computador que satisfaz as restrições de capacidade, o método faz trocas na definição inicial, até obter uma nova definição da rede que satisfaça as restrições e tenha o custo

mínimo.

Boorstyne e Frank apresentam o algorítmo "CUT - SATURATION" (discutido no ítem II.2.2-2) como um algorítmo eficiente para definição da rede, uma vez que ele combina informações fundamentais da rede, desempenho ("throughput"),por exemplo, com o método de troca de ramos, para encontrar a definição de custo mínimo que atenda as necessidades da rede.

Vale salientar o comentário que eles fazem a respeito da eficiência do algóritmo "CUT - SATURATION", onde afirmam ser este mais eficiente tanto no resultado final encontrado da definição da rede, quanto no tempo computacional requerido, em relação ao algorítmo de troca de ramos normal.

Para completar a definição da rede eles abordam o problema da localização dos comutadores ("switches") que fazem a ligação da rede de acesso local com a rede de alto nível.

Boorstyn e Frank definem o problema da seguinte maneira:

Dados os locais dos usuários, tráfego requerido de cada ponto a outro ponto, características dos comutadores e alternativas de tráfegos de linhas, minimizar o custo total de comunicação,sujeito às restrições de tráfego requerido,atraso, e confiabilidade.

Eles então apresentam uma versão modificada do algorítmo ADD para determinar a melhor localização dos comutadores. Nesta versão, cada comutador é considerado como composto por vários módulos, cada um com uma fração da capacidade

total. Durante uma iteração do algorítmo, um módulo pode ser adicionado à rede. O esquema básico do algorítmo é o seguinte:

Entrada:

$\Phi_{\emptyset}$ - localizações de usuários ainda não considerados.

$\Phi(Q)$ - localizações de usuários considerados para comutadores.

π - comutadores selecionados.

Processo:

i) Inicialize $\Phi(Q)$ e π para Φ; inicialize $\Phi_{\emptyset}$ para o conjunto de todos os nós; inicialize o custo do comutador para cada candidato.

ii) Calcule o centro de massa $C_{\emptyset}$ dos nós em $\Phi_{\emptyset}$ e calcule o custo ganho de linhas para cada candidato.

iii) Calcule o ganho total para cada candidato e encontre o melhor candidato, Q*, com o melhor ganho, S*, e o conjunto Φ^* de localizações de usuários em $\Phi_{\emptyset}$ ligados a Q*.

iv) Se S* é positivo, vá para v); de outra forma ligue os nós em $\Phi_{\emptyset}$ a Q*, adicione Q a π e pare.

v) Adicione Φ^* a $\Phi(Q^*)$ e delete Φ^* de $\Phi_{\emptyset}$; adicione Q* a π e modifique o custo do comutador de Q*; vá para ii).

Terminando, eles comentam algumas situações em que existem ligações via satélite para algumas conexões da rede.

4) O ESTUDO DE GERLM77

M. Gerla e L.Kleinrock apresentam um resumo dos modelos, análises e projetos de redes distribuidas de comunicação de de dados. Eles dividem o problema do projeto de redes

em vários subproblemas menores e discutem as soluções desses subproblemas, sugerindo depois um método heurístico para solucionar o problema geral de definição topológica.

A principal preocupação de uma definição topológica deve ser, segundo eles, o custo da configuração e o uso de canais de comunicação.

Eles alertam que o projeto de uma rede distribuida com a filosofia de pacotes comutados ("packet switched") é muito mais complexo que o de uma rede centralizada, devido ao fato de existirem vários caminhos alternativos na rede para cada pacote e a necessidade das análises da relação entre o tempo de atraso e utilização das linhas e "buffers".

O estudo de Gerla e Kleinrock está dividido nas seguintes partes:

1. Definição do problema:

   Dado:
   - Localização dos nós;
   - Tráfego requerido nos horários de "peak";

   Minimize:
   - Custo total das linhas;

   Sobre as variáveis do problema:
   - Topologia;
   - Capacidade dos canais;
   - Política de encaminhamento ("routing policy");

   Sujeito a:
   - Restrição de capacidade dos elos;
   - Restrição do atraso médio do pacote ("packet");
   - Restrição de confiabilidade;

A partir da definição do problema geral eles definem subproblemas e discutem a relação entre a medida de performance, parâmetro de entrada, variáveis de decisão e restrições que aparecem no problema geral. Os três subproblemas são: definição da capacidade, definição do fluxo, definição da capacidade e do fluxo, os quais são derivados do problema geral por meio de fixação de algumas variáveis de decisão.

2. Análises e modelamento:

Nesta parte do estudo, Gerla e Kleinrock fazem algumas considerações sobre a rede de pacotes comutados, a qual eles modelam como um sistema de filas, definindo modelos matemáticos para:

- Tempo de atraso
- Custo das comunicações

Apresentam também uma análise mais detalhada sobre:

- Tráfego requerido
- Política de encaminhamento
- Fluxo nos elos
- Capacidade dos elos
- Confiabilidade

Os modelos surgirão neste trabalho, oportunamente.

3. Problema de definição da capacidade dos elos:

O problema consiste em determinar a capacidade de cada elo c_i, $i=1,2,\ldots,b$, sendo b o número de elos, de forma a minimizar o custo total de comunicação, D, sendo:

$$D = \sum_{i=1}^{b} d_i(c_i)$$

onde : $d_i(c_i)$ é o custo da capacidade

c_i para o elo i

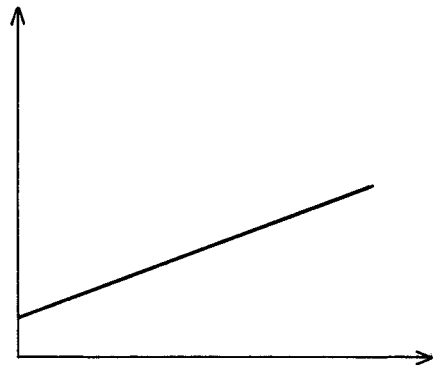
Trata-se da determinação da capacidade de cada elo de maneira a satisfazer as necessidades de tráfego de cada par de pontos, com o menor custo possível. A capacidade de cada elo tem que ser suficiente para atender ao fluxo determinado pela política de encaminhamento dos pacotes e não causar um atraso médio do pacote, maior que um valor pré-estabelecido, T_{max}.

A técnica de definição e solução do problema da capacidade dos elos depende da função de custo da capacidade ($d_i(C_i)$). Dentre as possíveis funções, Gerla e Kleirock consideram:

- Custo linear
- Custo discreto
- Custo côncavao

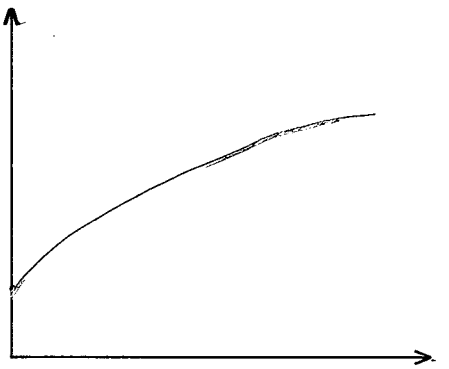
Os três gráficos que seguem ilustram estes tipos de funções.

GRÁFICO 2

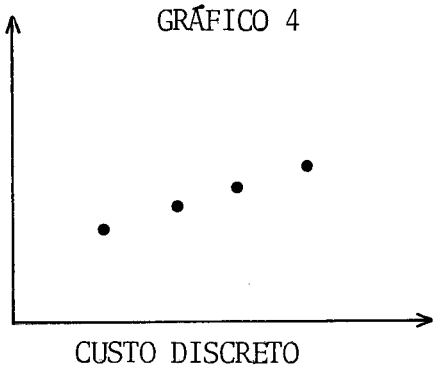


CUSTO LINEAR

GRÁFICO 3



CUSTO CÔNCAVO



4. Problema de rota:

Gerla e Kleinrock definem o problema de política de rota fixa que minimize o tempo médio de atraso.

O problema consiste em determinar o fluxo em cada elo, f_i, i=1,2,...,b , de forma a minimizar o tempo de atraso médio. O fluxo em cada elo tem que ser suficiente para atender as necessidades de tráfego entre os pontos que ele liga e, ao mesmo tempo, não pode ultrapassar a capacidade estabelecida para o elo.

Eles analisam dois algorítmos de rota: Algorítmo de caminho mínimo e o Algorítmo de derivação de fluxo, fazendo uma comparação entre os mesmos.

5. Problema de definição da capacidade e do fluxo:

O problema agora consiste em determinar tanto a capacidade de cada elo como também o fluxo sobre cada elo, de forma a minimizar o custo total de comunicação. O fluxo e a capacidade têm que ser determinados de tal forma que a capacidade de cada elo seja suficiente para atender ao seu fluxo e o atraso médio do pacote, em função do fluxo e da capacidade, e não exceda a um valor pré-estabelecido.

Este problema requer a otimização simultânea da

política de caminhamento (rota) e da capacidade das linhas.

O modelo pode variar de acordo com a função de custo-capacidade, e eles analisam o modelo para os casos de custo linear, custo côncavo e custo discreto. Para cada caso eles apresentam um ou mais algorítmos.

6. Definição Topológica:

Dado: Matriz de tráfego requerido, R

Minimize: $D(A,C) = \sum_{i \in A} d_i(C_i)$

onde: o conjunto dos elos, A, especifica a topologia

Tal que:

a) o fluxo na rede, f, satisfaça a matriz de tráfego requerido, R

b) $f \leq C$

c) $T = \frac{1}{\gamma} \sum_{i \in A} \frac{f_i}{C_i - f_i} \leq T_{max}$

d) O conjunto A tem que corresponder a uma topologia conectada

onde:

- f é o vetor dos fluxos dos elos da rede
- C é o vetor das capacidades dos elos da rede
- T é o atraso médio dos pacotes
- T_{max} é o atraso médio máximo permitido
- γ é a taxa de pacotes fluindo na rede

- f_i é o fluxo no elo i
- c_i é a capacidade do elo i

Gerla e Kleinrock fazem a análise do algorítmo BRANCH EXCHANGE - BXC (H. Frank e W. Chou, 1970), que tem a seguinte filosofia:

- Inicia com uma topologia arbitrária e pesquisa a cada passo a transformação local adequada (a transformação local, geralmente chamada de ramificação e troca ("branch exchange"), consiste na eliminação de um ou mais elos antigos e insersão de um ou mais elos).

- Os passos básicos do algorítmo são os seguintes:

1. Transformação local. Adiciona um novo elo e elimina um elo antigo, de maneira a manter a conectividade da rede.
2. Define capacidade e fluxos para a nova topologia. O custo e o "throughput" são avaliados. Se houver melhoria na relação custo - "throughput", então a nova topologia será aceita. Caso contrário será rejeitada.
3. Se todas as transformações locais já tiverem sido exploradas, então pare. Caso contrário vá para o passo 1.

Gerla e Kleinrock alertam para o fato do algorítmo BXC requerer uma exploração exaustiva de todas as trocas de topologias locais, o que acarreta na consumação de muito tempo computacional, quando aplicado para mais de 20 ou 30

nós.

Eles citam então o algorítmo CUT-SATURATION (que já foi considerado neste artigo no ítem II.2.2-2 pág.48) que é uma extensão do algorítmo BXC, porém, ao invés de examinar todas as possíveis trocas de topologia, ele seleciona apenas aquelas trocas que acarretam uma melhoria de "throughput" e custo.

Eles comentam ainda que a selção do algorítmo a propriado depende da estrutura de custo-capacidade, da presença de restrições adicionais à topologia, do grau de iteração do homem e do balanceamento entre o custo e o grau de precisão requerido pela aplicação particular.

7. Aplicações

Gerla e Kleinrock apresentam vários resultados obtidos da aplicação dos algorítmos citados, por meio de alguns gráficos que demonstram o comportamento dos algorítmos em relação a diversos fatores e efetividade. Mostram também uma tabela comparativa do tempo computacional exigido por cada algorítmo. Os testes foram feitos na rede ARPANET. Na opinião deles, o CUT-SATURATION é o algorítmo que produz os resultados mais eficientes.

5) Algorítmo COG /SHARR77/

Este algorítmo trabalha com os seguintes tipos de rede:

a) Multi-centro, multi-estrela (MCMS)
("Multi-center, multi-star")

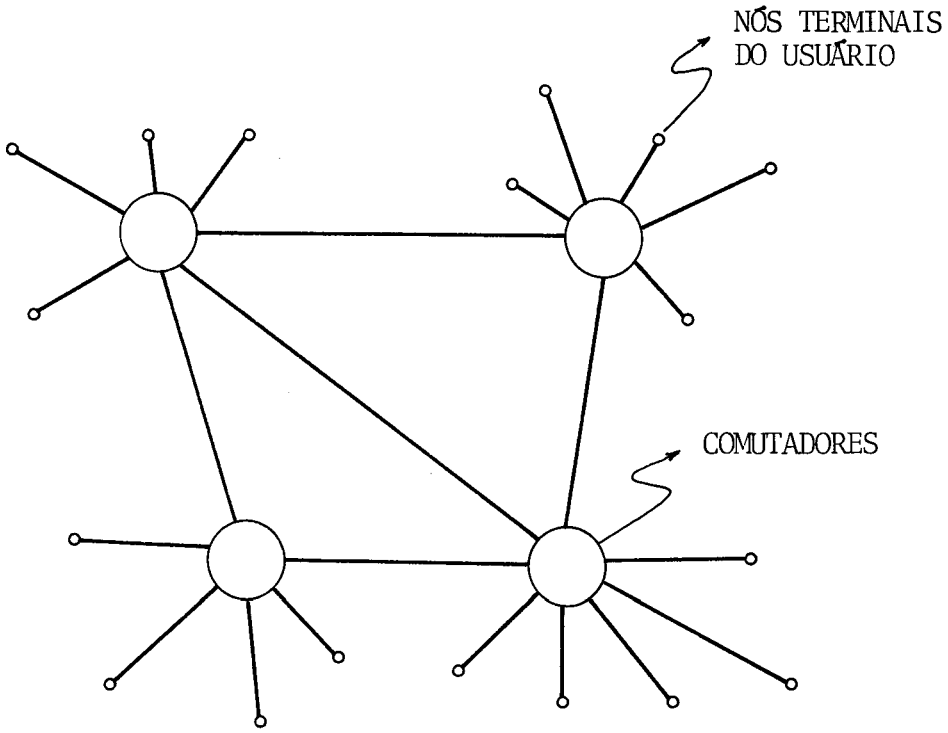


FIGURA 8 : Rede Multi-centro, Multi-estrela

b) Multi-centro, multi-ponto (MCMD)

("Multi-center, multi-drop")

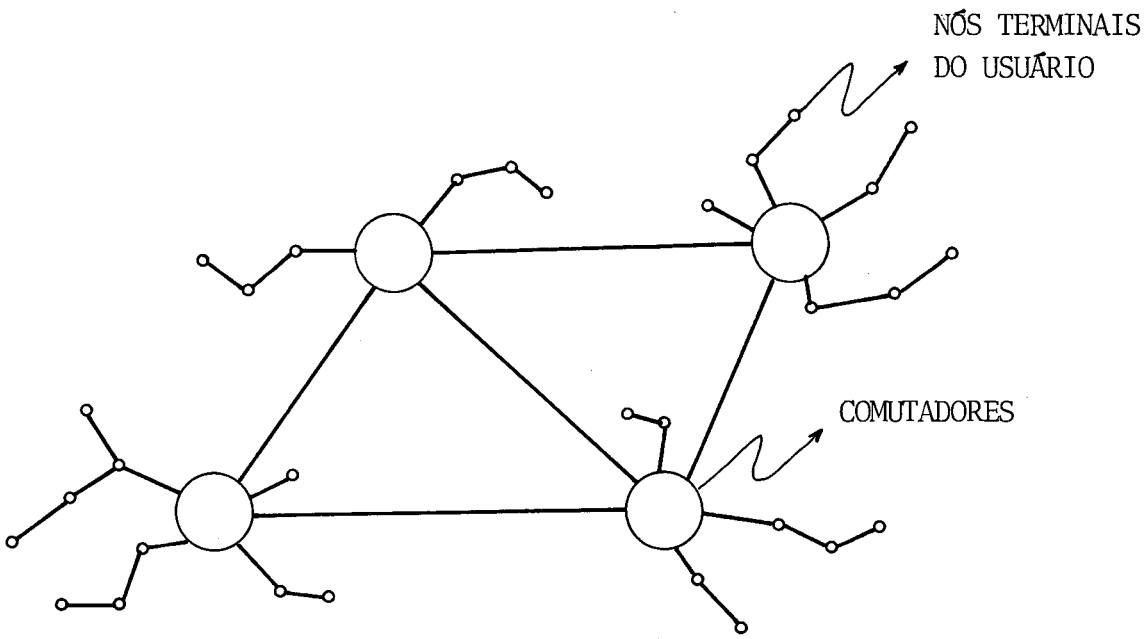


FIGURA 9 : Rede Multi-centro,Multi-ponto

Segundo Sharma, o projeto de uma rede envolve, geralmente:

a) Aquisição de dados de entrada.

b) Plano de tarifas.

c) Algorítmo para localização dos centros comutadores.

d) Algorítmo para conectar as localizações dos usuários aos centros comutadores.

e) Algorítmo para definir a conexão do tronco da rede de alto nível aos centros comutadores.

f) Iteração homem-máquina para traços adicionais no desenho da rede e redefinições.

Sobre a aquisição de dados de entrada, Sharma cita:

- Coordenadas verticais e horizontais de cada localidade
- Tráfego de entrada e/ou saída por cada terminal, ou tráfego entre todos os pares de nós
- Tempo de resposta requerido
- Taxa de crescimento em termos de locais, terminais ou intensidade de tráfego
- Restrições em relação a seleção de terminais de dados, concentradores ou multiplexadores
- Restrições na seleção dos centros comutadores
- Restrições na seleção das tarifas de comunicação
- Limitações de custos.

O plano de tarifas depende da região inerente a rede.

Sharma apresenta o algorítmo COG ("center-of-gravity") /SHARR77/ para determinar a localização dos centros comutadores.

A filosofia do algorítmo é a seguinte:

Escolhe o ponto centro de gravidade da rede, em relação a uma certa função de peso (geralmente constante). Divi

de a rede em duas regiões, uma ao lado este do centro de gravidade (COG_1) e outro ao lado oeste do COG_1, em termos das coordenadas verticais e horizontais. Encontra o COG para cada uma das novas regiões, COG_2 e COG_3, que podem ser considerados como os locais ótimos para a localização de dois centros. Decompõe as duas regiões em 4 regiões, ao norte do COG_2 e ao sul do COG_2, ao norte do COG_3 e ao sul do COG_3. Obtenha os 4 novos centros de gravidade, que serão as localizações ótimas para 4 centros. Repetindo-se este processo, obtém-se a localização ótima para 2^n centros, n=0,1,2,... .

O esquema a seguir ilustra o método:

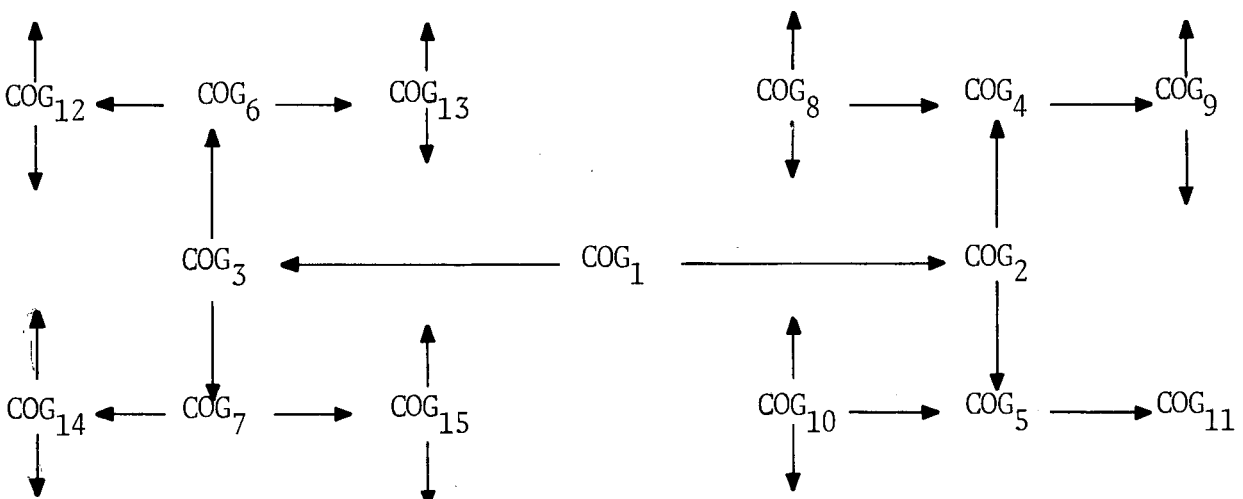


FIGURA 10: Esquema do algorítmo COG

Considerando as localizações dos nós como candidatos a locais dos centros, o critério de escolha dos COG's é o seguinte:

$$\text{MIN}\left[\sum_{j=1}^{N} D_{ij}\ W_j\right] \equiv COG_1$$

Onde:

$1 \leq i \leq N$, sendo N o número de terminais

$1 \leq j \leq N$

D_{ij} = distância i,j = $\sqrt{0,1\left[(V_i-V_j)^2 + (H_i-H_j)^2\right]}$

W_j = peso do j-ésimo nó em termos de número de circuitos, ou intensidade de tráfego, ou número de terminais, etc.

V_i = coordenada vertical do ponto i, para i=1,2,...,N

H_i = coordenada horizontal do ponto i, para i=1,...,N

Obs.: O fato do algorítmo achar um número de locais sempre múltiplo de 2 é uma desvantagem marcante.

Para escolher que locais ligar a cada centro, no caso da topologia MCMS, simplesmente usa-se o critério de ligar cada localidade ao centro mais próximo. No caso da MCMD, Sharma escolhe o seguinte procedimento:

1. Conecte as localidade aos comutadores no mesmo critério usado na rede MCMS.
2. Usa o algorítmo de Sharma, ou de Esau-Williams, ou de Martin para definir o desenho das linhas que conectam as localidades (ou terminais) aos centros.

CAPÍTULO III

DESENVOLVIMENTO DE UM ALGORÍTMO PARA DEFINIÇÃO TOPOLÓGICA DE REDES DE COMUNICAÇÃO DE DADOS (WRP)

III.1. Introdução

Baseado no estudo de vários algorítmos de topologia de redes de comunicação de dados com várias filosofias diferentes para os diversos tipos de redes, será apresentado neste capítulo um algorítmo para definição topológica de redes de comunicação de dados que possui certas vantagens quanto à flexibilidade e versatilidade de soluções.

O algorítmo procura tomar todas as decisões que influenciam no custo total da rede, procurando minimizá-lo, partindo de uma quantidade de informações iniciais tão elástica quanto se deseje.

A apresentação do algorítmo será dividida em diversos sub-ítens, para facilitar a compreensão, os quais se encarregaram de informar as características da rede gerada,os dados iniciais, os objetivos do algorítmo, a filosofia básica, as estruturas com que ele trabalha, os passos básicos, etc.

III.2. Características da Rede Gerada pelo Algorítmo

As características da rede gerada pelo algorítmo são as seguintes:

- Os terminais ficam ligados aos concentradores por meio de linhas multiponto, podendo ter uma ou mais linhas ligadas a um concentrador e um ou mais terminais pertencentes

a uma linha multiponto.

- Os concentradores ficam ligados a outros concentradores (ou comutadores), em tantos níveis quantos desejados, por meio de linhas multiponto, tendo cada linha tantos concentradores quantos se queira.

- Os terminais podem também serem ligados aos concentradores por linhas ponto-a-ponto, como também os concentradores podem ser ligados a outros concentradores (ou comutadores) por linhas ponto-a-ponto.

- A solução final pode ser uma configuração para redes distribuidas ou centralizadas.

A figura a seguir ilustra um possível resultado do algorítmo:

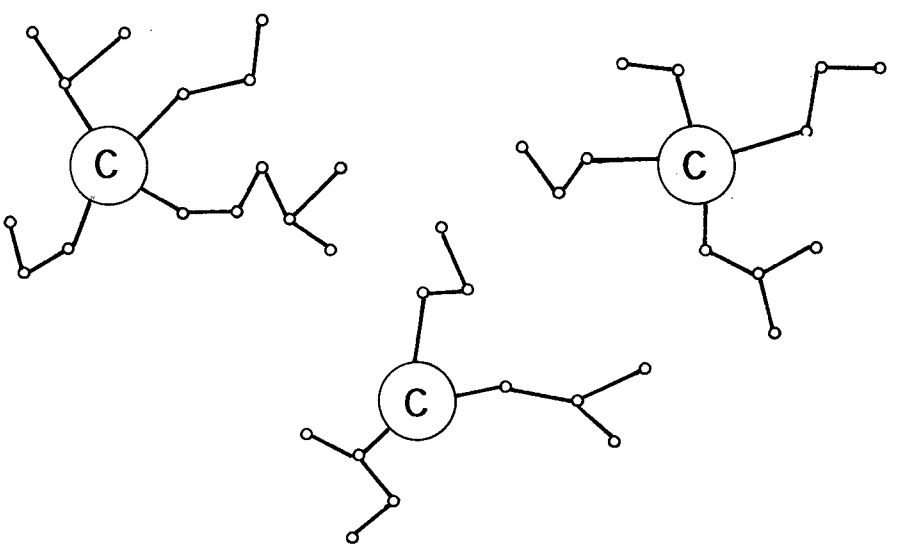


FIGURA 11 : Possível resultado do algorítmo

III.3. Objetivos do Algorítmo:

O algorítmo se propõe a encontrar a rede de menor custo, por meio de:

- Escolha do número, tipo (em termos de capacidade) e localização dos concentradores.

- Escolha dos subconjuntos de terminais que vão ser conectados a cada concentrador.

- Escolha dos desenhos ("lay-out") das sub-redes de linhas multiponto para conectar cada subconjunto de terminais aos seus respectivos concentradores.

III.4. Dados iniciais

O usuário deve fornecer os seguintes dados iniciais para o algorítmo:

1 - Número de terminais: NT

2 - Número de linhas da tabela de informações: IT

Obs.: Esta tabela será vista mais tarde.

3 - Dimensões da tabela das faixas de custo das linhas de comunicação: IF1MAX e JF1MAX

4 - Fator a ser utilizado da fórmula do cálculo do liminte de custo para uma ligação: FATOR

Obs.: Esta fórmula será vista mais adiante (pág.91).

5 - Variável que indica se existem inibições de ligações ou não: INIBE:

$$\text{INIBE} = \begin{cases} 1, \text{ Se existem inibições} \\ 0, \text{ caso contrário} \end{cases}$$

6 - Variável que indica se a função de custos será a distância entre os dois pontos ou não: CDIST:

$$\text{CDIST} = \begin{cases} 1, \text{ se a função de custos for a distância entre os dois pontos} \\ 0, \text{ caso contrário} \end{cases}$$

7 - Coordenadas dos terminais:

X_i, i=1,..., NT

Y_i, i=1,..., NT

8 - Tráfego gerado em cada terminal: $TRAFE_i$, i=1,...,NT

9 - Terminal ao qual este está ligado:

$$SINT_i = \begin{cases} j, & \text{se o terminal i está ligado ao j} \\ 0, & \text{se o terminal i não está ligado.} \end{cases}$$

para i=1,2,...,NT.

10 - Valores relativos às faixas de custos das linhas de comunicação: $FAIXA1_{i,j,1}$ → custo de instalação

$FAIXA1_{i,j,2}$ → custo mensal.

Para i=1,2,...,IF1MAX ; j=1,2,...,JF1MAX.

11 - Valores relativos aos degraus tarifáricos para as faixas de custos:

$FAIX1_i$, para i=1,2,...,JF1MAX.

A tabela a seguir mostra um exemplo da tabela de custos das linhas, em sua forma original.

TABELA DE CUSTOS DAS LINHAS DE COMUNICAÇÃO

(em Cr$)

DEGRAUS TARIFÁRIOS	VELOCIDADE EM BPS		
	1.200	2.400	4.800
2	1,792,00	2.690,00	5.380,00
3	3.217,00	4.830,00	9.660,00
4	4.416,00	6.630,00	13.260,00
5	5.901,00	8.860,00	17.720,00
6	8.258,00	12.400,00	24.800,00
7	8.844,00	13.280,00	26.560,00
8	9.437,00	14.170,00	28.340,00
9	9.917,00	14.890,00	29.790,00
10	10.723,00	16.100,00	32.200,00

DEGRAU TARIFÁRIO	DISTÂNCIA GEODÉSICA EM KM		
2	Até		50
3	50	a	100
4	100	a	200
5	200	a	300
6	300	a	500
7	500	a	700
8	700	a	1.000
9	1.000	a	1.500
10	Acima	de	1.500

12 - Tabela informativa das restrições e características dos equipamentos, por meio dos vetores que informam:

- Número máximo de linhas por concentrador:

  $VLIMAX_i$, i=1,...,IT

- O tráfego máximo por concentrador:

  $VTRCONMAX_i$, i=1,...,IT

- Percentual de tráfego liberado por concentrador:

  $VPCENT_i$, i=1,...,IT

- Custo de instalação dos concentradores:

  $VCUSTCON1_i$, i=1,...,IT

- Custo mensal dos concentradores:

  $VCUSTCON2_i$, i=1,...,IT

- Número máximo de terminais p/linha multiponto:

  $VTEMAX_i$, i=1,...,IT

- Tráfego máximo por linha multiponto:

  $VTRLIMAX_i$, i=1,..., IT

- Faixa da tabela dos custos das linhas de comunicação:

  $VIF1_i$, i=1,...,IT

Vejamos um exemplo de uma tabela típica de informações e restrições:

TABELA DE INFORMAÇÕES E RESTRIÇÕES

(em Cr$)

	VLIMAX	VTRCONMAX	VPCENT	VCUSTCON1	VCUSTCON2	VTEMAX	VTRLIMAX	VIF1
1	4	800	80%	80.000,00	8.000,00	4	100	1
2	5	1000	90%	90.000,00	9.000,00	6	150	2
3	7	1500	90%	120.000,00	15.000,00	10	220	2
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
IT								

13 - Indicação das inibições (quando as mesmas existirem) por meio de pares:

(i,j), indicando que o ponto i não pode ser ligado ao ponto j(automáticamente não será permitida também a ligação do ponto j ao ponto i.

Obs.: após a indicação do último par de inibições deve existir um par (0,0), que indica fim das inibições.

III.5. Filosofia Básica do Algorítmo

De uma maneira geral, o algorítmo constroi uma rede com as características já citadas, respeitando as restrições impostas pela tabela de informações e restrições. Para cada linha da tabela ele gera uma solução diferente dando todas as informações sobre a arquitetura e o custo total da re-

de gerada.

O algorítmo pode ser dividido em alguns passos básicos correspondentes a módulos, sendo alguns de definição da rede e obrigatórios enquanto outros de otimização e opcionais. Os passos básicos do algorítmo são os seguintes:

1) Obtenção de informações iniciais:

Inicialmente o algorítmo obtém informações iniciais que dimensionam o problema em termos de número de terminais, tráfego, capacidade das linhas e concentradores, e informam quanto às tabelas de custos das linhas de comunicações.

2) Inicializações:

O algorítmo inicializa a matriz de custos, a matriz de estados, as estruturas relativas a terminais, linhas e concentradores e as demais estruturas que auxiliam o algorítmo na gerência de espaço e manutenção das listas. Calcula o limite de custos.

3) Construção das linhas multipontos iniciais:

O algorítmo ordena os terminais de acordo com o grau de dispersividade em relação ao conjunto de terminais. A partir dos terminais mais dispersos, ou seja, aqueles que estão mais afastados dos demais, o algorítmo liga um terminal a cada iteração, ao terminal mais próximo dele (será usado o termo próximo no sentido de menos custoso) que satisfaça as restrições de capacidade das linhas. Quando uma ligação não satisfaz as restrições o algorítmo pega o terminal seguinte mais próximo. Quando há empate na escolha dos terminais para onde vai ser feita uma ligação, o algorítmo escolhe o que ti-

ver a menor distância.

Ao terminar esta fase existirá um conjunto de linhas multiponto que satisfazem as restrições e possuem um terminal que não emite ligações para nenhum outro, embora receba ligações. Estes terminais serão chamados de "fins de linha".

4) Seleção dos concentradores:

Nesta fase o algorítmo seleciona dentre os "fins de linha" aqueles mais apropriados para concentradores. Para tanto, constroi inicialmente um vetor dos "fins de linha" e classifica-o de acordo com o grau de dispersividade em relação ao conjunto de fins de linha. Depois ele faz uma estimativa do número de concentradores necessário e escolhe-os no vetor dos "fins de linha", baseado no seguinte critério:

Escolhe o "fim de linha" da primeira posição do vetor. Escolhe os demais em posições seguintes do vetor, deixando um espaço de tantas posições quanto seja o valor da variável DEGRAU, sendo:

DEGRAU = NF/NC , onde:

NF = número de "fins de linha"

NC = número de concentradores

5) Ligação das linhas multiponto aos concentradores:

Trabalhando apenas com o conjunto de fins de linha o algorítmo liga cada linha multiponto ao concentrador mais próximo, começando pelos fins de linha menos dispersos. A ligação é feita entre o fim de linha e o concentrador, contanto que sejam satisfeitas as restrições relativas aos concentradores.

6) Particionamento da rede:

Neste ponto o algorítmo particiona o conjunto de terminais, sendo cada partição composta por um concentrador e todos os terminais pertencentes as linhas multiponto que estão ligadas ao cocentrador.

Para cada partição e algorítmo define uma sub-rede em forma de ÁRVORE, tendo o concentrador como centro(raiz da árvore). O procedimento para definição das sub-redes é o seguinte: calcula o custo de ligação de cada terminal ao concentrador e classifica os terminais de acordo com estes custos. A partir daí, ele constroi as linhas multiponto definitivas a partir dos terminais mais dispersos (mais custosos) ligando a cada passo um terminal ao mais próximo de si, verificando antes se satisfaz as restrições e se é menos custoso ligar a um terminal que ao cocentrador. Ao fim desta fase, existirão linhas multipontos viáveis (que satisfazem as restrições) tendo cada uma um terminal fim de linha. Cada linha multiponto é então ligada ao concentrador por meio do "fim de linha". As definições das sub-redes são feitas iterativamente, sendo definida uma a cada iteração. A final deste passo existirá um conjunto de sub-redes definidas, tendo cada uma um concentrador e as linhas multiponto definitivas ligadas a este.

7) Otimização das ligações das linhas aos concentradores:

Após a definição final das linhas multiponto o algorítmo tenta otimizar a ligação de linhas a concentradores da seguinte maneira: constroi a princípio um vetor dos "fins de linha" e classifica este vetor de acordo com o custo de liga-

ção do "fim de linha" ao seu concentrador. Depois, a partir dos "fins de linha" mais dispersos do seu concentrador o algorítmo verifica se existe um outro concentrador ao qual este "fim de linha" possa ser ligado sem violar as restrições e cujo custo de ligação ao novo concentrador seja menor que o custo da ligação ao concentrador que ele esteja ligado na ocasião do teste. Em caso afirmativo a linha multiponto é desligada do concentrador original e ligada ao novo concentrador e as atualizações necessárias são processadas.

8) Otimização do número de concentradores:

Neste passo o algorítmo tenta reduzir o número de concentradores, desde que resulte numa redução de custos. Para tanto, ele constroi inicialmente um vetor contendo todos os concentradores e classifica-o de acordo com o número de linhas que cada um possui. Então, a partir dos concentradores que possuem o menor número de linhas até terminar o vetor de concentradores, ou encontrar um concentrador que possua 70% da capacidade de linhas, o algorítmo verifica se é vantagem eliminar um concentrador e distribuir suas linhas entre outros concentradores não saturados, da maneira mais econômica possível. Será considerada vantajosa a eliminação de um concentrador se o acréscimo de custo das linhas de comunicação provocado pela religação das linhas aos outros concentradores for menor que o custo do concentrador a ser eliminado. Em caso afirmativo o concentrador é eliminado e suas linhas são distribuídas entre os outros concentradores da maneira mais econômica possível, sendo feitas as atualizações necessárias.

9) Otimização dos tipos dos concentradores:

Este é o passo em que o algorítmo tenta misturar vários tipos de concentradores na tentativa de reduzir o custo total da rede. Para conseguir isto o algorítmo constroi a priori um vetor contendo os concentradores atuais. Depois, constroi uma matriz de distâncias entre os concentradores e, a partir desta matriz ele entra num laço iterativo onde escolhe a cada passo o par de concentradores mais próximos e tenta substituí-lo por um único concentrador de outro tipo, contanto que: esteja caracterizado na tabela de informações e restrições, seja capaz de suportar toda a carga dos outros dois concentradores e o seu custo somado com o acréscimo de custos de religação das linhas seja menor que a soma dos custos dos dois concentradores.

A pesquisa de novos concentradores é sempre feita a partir da linha seguinte à linha atual da tabela de informações e restrições, que o algorítmo esteja trabalhando.

III.6. Estruturas de Dados Usadas pelo Algorítmo

O algorítmo utiliza as seguintes estruturas para manipulação dos dados:

1 - Tabela de informações e restrições:

Esta estrutura já foi descrita no ítem III.4-12(pág.72). Ela é armazenada em um conjunto de vetores de dimensão IT, sendo cada vetor correspondente a uma coluna da tabela.

2 - Tabela de custos das linhas de comunicação:

Esta estrutura também já foi descrita no ítem III.4-10 (pág.70). Ela é armazenada em uma matriz de nome FAIXA1,com 3 dimensões: cada linha representa uma velocidade diferente;cada coluna representa um limite de distância para uma determinada tarifa;a zona 1(um) contém a tarifa para a implantação da linha e a zona 2(dois) a tarifa mensal. O esquema a seguir ilustra esta matriz:

		DISTÂNCIA EM KM																	
		50		100		200		300		500		700		1000		1500		+ ∞	
		1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
FAIXAS DE VELOCIDADE	1																		
	2																		
	.																		
	.																		
	.																		
	NF																		

3 - Matriz de custos:

É a matriz de nome MCUST que armazena o custo de uma linha de comunicação partindo do ponto i ao ponto j, conforme mostra o esquema :

	1	2	3	...	NT
1					
2					
3					
. .					
NT					

4 - Matriz de estados:

Possui o nome de MEST e tem como objetivo informar se o ponto i pode ou não ser ligado ao ponto j. A estrutura é idêntica a MCSUT, como mostra o esquema:

	1	2	3	...	NT
1					
2					
3					
. .					
NT					

Se MEST(i,j) = 0, então o ponto i pode ser ligado ao j.

Se MEST(i,j) ≠ 0, então o ponto i não pode ser ligado ao j.

Se MEST(i,j) = 1, então a ligação de i para j está inibida.

5 - Estrutura dos terminais:

É composta por um conjunto de vetores de dimensão NT, contendo as informações necessárias, como pode ser visto no esquema:

	SINT	X	Y	TRAFE	LINHA	EMITE	SOMA	VAUX
1								
2								
. .								
NT								

Os vetores X, Y, TRAFE e SINT já foram explicados nos ítens III.4-7, III.4-8 e III.4-9 (pág.70), respectivamente,pois tratam-se de dados inicias para o algorítmo.

O elemento LINHA (i) contém o índice da linha a que pertence o terminal i.

O elemento EMITE (i) contém o índice do terminal para o qual o terminal i emite uma ligação.

O elemento SOMA (i) contém a soma dos custos de ligação do terminal i para todos os outros terminais.

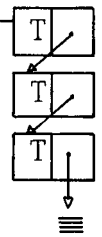
O vetor VAUX contém os índices dos terminais classificados de acordo com o grau de dispersividade.

6 - Estrutura das linhas:

É também composta por um conjunto de vetores de

dimensão NL com funções específicas:

	TFIM	TR-ACUM	TE-ACUM	CUST-ACUM	CON	LSTE
1						
2						
. .						
NL						



NL = número de linhas

O elemento TFIM (i) contém o índice do terminal "fim de linha" da linha i.

O elemento TR-ACUM (i) contém o total de tráfego acumulado na linha i, ou seja, o somatório dos tráfegos gerados por todos os terminais pertencentes à linha i.

O elemento TE-ACUM (i) possui a quantidade de terminais pertencentes à linha i.

O elemento CUST-ACUM (i) armazena o somatório dos custos de comunicação da linha i.

O elemento CON (i) contém o índice do concentrador ao qual a linha i está ligada.

O elemento LSTE (i) aponta para uma lista onde estão armazenados os índices dos terminais pertencentes à linha i. Esta lista possui nós da forma | T | PROXT | , onde T contém o índice do terminal e PROXT o próximo elemento da lista.

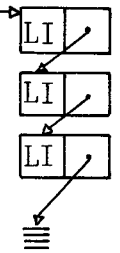
Obs.: O número de linhas, NL, é calculado pelo al

gorítmo.

7 - Estrutura dos concentradores:

Composta por um conjunto de vetores de dimensão NC, conforme ilustra o esquema:

	TCN	TR-CON-ACUM	TR-LIBER	LI-ACUM	CUST-CON-ACUM	TIPO	LISL
1							
2							
. .							
NC							



O elemento TCN (i) possui o índice do terminal escolhido par local do concentrador i.

O elemento TR-CON-ACUM (i) contém o somatório do tráfego das linhas que estão ligadas ao concentrador i.

O elemento TR-LIBER (i) possui o tráfego liberado pelo concentrador i.

O elemento LI-ACUM (i) armazena a quantidade de linhas acumuladas no concentrador i.

O elemento CUST-CON-ACUM (i) contém o somatório dos custos de comunicação das linhas ligadas ao concentrador i.

O elemento TIPO (i) possui o tipo do concentrador i, ou seja, aponta para a linha da tabela de informações e

restrições que caracteriza o concentrador i.

O elemento LISL (i) contém um apontador para a lista onde estão armazenadas as linhas que pertencem ao concentrador i. Esta lista possui nós da forma [LI | PROXL] , onde LI armazena o índice da linha e PROXL aponta para o próximo elemento da lista.

III.7. Descrição das Procedures do Programa

A seguir, apresentaremos a descrição das procedures e módulos do algorítmo. São dadas as seguintes informações: Nome da procedure ou módulo, funções, parâmetros, principais variáveis e estruturas, procedures utilizadas. No canto superior direito de cada bloco aparece o número do "BEGIN" correspondente a listagem da compilação do programa da implementação do algorítmo, que é apresentada no Anexo A.

1) PROCEDURE DISTÂNCIA B5

FUNÇÃO:

Calcular a distância entre dois pontos

PARÂMETROS:

XX : Índice do primeiro ponto

YY : Índice do segunto ponto

2) PROCEDURE CALCUSTO B6

FUNÇÃO:

Fazer o cálculo dos elementos da matriz de custos, MCUST

PARÂMETROS:

X : vetor contendo índices dos primeiros pontos;

Y : vetor contendo índices dos segundos pontos;

FAIXA1: Matriz das faixas de custos;

CDIST: Variável que indica se será usada a distância para custo entre os pontos;

FAIX1: Vetor dos degraus tarifáricos.

PROCEDURES UTILIZADAS:

DISTÂNCIA

3) PROCEDURE ORDENE B12

FUNÇÃO:

Ordenar um vetor (VET1) de acordo com os valores de outro vetor (VET2).

PARÂMETROS:

VET1 : Vetor a ser ordenado;

VET2 : Vetor dos valores a serem considerados para a ordenação;

DIM : Dimensão dos vetores.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

FIM : Variável que indica quando a ordenação já terminou

4) PROCEDURE INICIE B15

FUNÇÃO:

Inicializar uma lista de espaço disponível

PARÂMETROS:

LISTA : Lista a ser inicializada;

DIM : Dimensão da lista;

DISP : Ponteiro da lista

5) PROCEDURE PEGUE B16

FUNÇÃO:

Pegar um nó na lista de espaço disponível

PARÂMETROS:

PROX : Vetor dos ponteiros para ligação da lista de espaço disponível;

DISP : Apontador do espaço disponível atual;

TX : Apontador do nó obtido na lista de espaço disponível.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

LAMB : Indicador de fim de lista

6) PROCEDURE DEVOLVA B19

FUNÇÃO:

Devolver um nó à lista de espaço disponível

PARÂMETROS:

PROX : Vetor dos ponteiros para ligação da lista de espaço disponível ;

DISP : Apontador do espaço disponível atual;

TX : Nó a ser devolvido .

7) PROCEDURE ATEST B20

FUNÇÃO:

Atualizar uma cela na matriz de estados com o valor -1, respeitando as marcas de inibição e pré-fixação.

PARÂMETROS:

MEST : Matriz de estados,

IEST : Índice de linha da matriz de estados,

JEST : Índice de coluna da matriz de estados.

8) PROCEDURE INICIESTA B21

FUNÇÃO:

Inicializar a matriz de estado

PARÂMETROS:

MEST : Matriz de estados;

SINT : Vetor que registra as ligações pré-fixadas;

INIBE: Variável que indica se existe ou não inibição;

PASSEI: Variável que indica se as inibições já foram registradas;

I,J : Índices das celas a serem inibidas;

NT : Número de terminais.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

ACABOU : Variável que controla laços.

9) PROCEDURE GERELIN B27

FUNÇÃO:

Gerar uma linha para um terminal dado.

PARÂMETROS:

TØ : Terminal para o qual a linha será gerada;

ESTRUTURA DAS LINHAS;

L : apontador da linha gerada.

10) PROCEDURE DEVOLVELIN B29

FUNÇÃO:

Devolver uma linha à estrutura das linhas não utilizadas.

PARÂMETROS:

LIN : Linha a ser devolvida;

ESTRUTURA DAS LINHAS.

11) PROCEDURE IMPRE1 B30

FUNÇÃO:

Imprimir cabeçalhos, parâmetros iniciais, a tabela de restrições e informações iniciais, tabela das faixas de custos.

PARÂMETROS:

NT : Número de terminais;

IT : Dimensão da tabela de informações e restrições;

IF1MAX: Número de linhas da tabela das faixas de custos;

JF1MAX: Número de colunas da tabela das faixas de custos;

TABELA DE INFORMAÇÕES E RESTRIÇÕES;

TABELA DAS FAIXAS DE CUSTOS.

12) PROCEDURE IMPRE2 ..... B33

FUNÇÃO:

Imprimir a estrutura dos terminais e das linhas e a variável de controle do algorítmo.

PARÂMETROS:

CONTR1 : Variável que controla a execução do algorítmo;

ESTRUTURA DOS TERMINAIS;

ESTRUTURA DAS LINHAS.

13) PROCEDURE LIGAÇÃO B36

FUNÇÕES:

Fazer a ligação entre dois pontos, respeitando as restrições; atualizar a matriz de estados, atualizar as estruturas dos terminais e das linhas.

PARÂMETROS:

MEST : Matriz de estados

ESTRUTURA DAS LINHAS;

ESTRUTURA DOS TERMINAIS;

MCUST : Matriz de custos;

TEMAX : Número máximo de terminais por linha;

LIGOU : Variável que indica se a ligação foi efetivada;

CUSTLIM: Limite de custos;

TRMAX : Tráfego máximo por linha.

PROCEDURES UTILIZADAS:

GERELIM;

DEVOLVELIN;

PEGUE;

ATEST.

14) PROCEDURE RETIRE B57

FUNÇÃO:

Retirar uma determinada linha de um concentrador.

PARÂMETROS:

CX : Concentrador de onde a linha será retirada;

LX : Linha a ser retirada;

PX : Ponteiro da linha que foi retirada;

ESTRUTURA DOS CONCENTRADORES.

15) PROCEDURE IMPRE3 ..... B59

FUNÇÕES:

Imprimir a estrutura dos concentradores, imprimir o custo total das ligações; imprimir o custo total dos concentradores; imprimir o custo total da rede.

PARÂMETROS:

ESTRUTURA DOS CONCENTRADORES;

TOTCUSTLIN : Custo total das linhas;

TOTCUSTCON : Custo total dos concentradores;

TOTCUST : Custo total da rede.

III.8. Descrição dos Módulos do Programa

1) Módulo do algorítmo WRP B1

FUNÇÃO:

Definir a topologia de uma Rede de Comunicação de Dados que possua o menor custo possível. A rede será distribuida ou centralizada, com linhas multiponto ou ponto-a-ponto, a depender das informações fornecidas pela tabela de restrições e informações iniciais.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

NT : Número de terminais;

IT : Dimensão da tabela de restrições e informações;

IF1MAX: Número de linhas da tabela das faixas de custos;

JF1MAX: Número de colunas da tabela das faixas de custo;

FATOR : Parâmetro para a fórmula do cálculo do limite de custo.

Fórmula: $CUSTLIM = CUSTMED + \frac{CUSTMED}{FATOR}$

onde: CUSTLIM é o limite de custo para a ligação entre dois pontos

CUSTMED é o custo médio das ligações

INIBE : variável que indica se tem inibições;

CDIST : variável que indica se a distância entre dois pontos será considerada como custo ou não;

CONTR : variável que controla a execução do programa;

I,J : variáveis ponteiros das estruturas.

SUB-MÓDULOS UTILIZADOS:

Todos os sub-módulos.

PROCEDURES UTILIZADAS:

Todas as procedures.

2) Módulo definidor de estruturas e executor de WRP B2

FUNÇÕES:

- Definir a estrutura da tabela de restrições e informações;
- Definir a estrutura dos terminais;
- Definir a estrutura da tabela das faixas de custos;
- Definir a matriz de estados;
- Definir a variável que controla a marcação das inibições;
- Preencher todas as estruturas definidas;
- Executar os procedimentos do algorítmo, gerando uma rede para cada linha da tabela de restrição e informações.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

Aquelas definidas pelo módulo

SUB-MÓDULOS UTILIZADOS:

3

PROCEDURES UTILIZADAS:

Todas as procedures.

3) Módulo definidor de estruturas para os terminais e sub-executor do WRP B3

FUNÇÕES:

- Definir a estrutura dos terminais;
- Definir o vetor VAUX, que armazena os índices dos terminais;
- Definir a matriz de custos;
- Definir as variáveis: LIMAX, TEMAX, IF1, TRCONMAX, TRLIMAX, TRLIBER, que armazenam os valores dos respectivos vetores, de acordo com a variável CONTR.

- Definir variáveis de uso geral;
- Fazer a estimativa do número de linhas necessárias.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- Aquelas definidas pelo módulo;
- NL : Número de linhas necessárias;
- NT : Número de terminais.

SUB-MÓDULOS UTILIZADOS:

4

PROCEDURES UTILIZADAS:

Todas procedures.

4) Módulo definidor da estrutura das linhas, das procedures e sub-executor do WRP B4

FUNÇÕES:

- Definir a estrutura das linhas;
- Definir as procedures 1 a 13;
- Fazer as inicializações, chamando as procedures CALCUSTO, INICIESTA e INICIE;
- Encher o vetor VEAUX;
- Calcular o vetor SOMA;
- Calcular o custo médio e o limite de custos;
- Construir as linhas multiponto iniciais;
- Calcular a dimenção do vetor dos fins de linha.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

SOMA : Vetor que armazena a soma deos custos das ligações de cada terminal a todos os outros;

VEAUX: Vetor auxiliar que armazena os índices de todos os terminais;

SINT : Vetor que contém as ligações pré-fixadas ;

MINIC : Variável que contém o custo mínimo;

MEST : Matriz de estados;

MCUST : Matriz de custos;

DIST1 e DIST2: Variáveis que armazenam distâncias entre dois pontos;

LIGOU : Variável que indica se a ligação foi efetuada;

TEMAX : Número máximo de terminais por linha;

NF : Número de terminais fins-de-linha.

SUB-MÓDULOS UTILIZADOS:

5.

PROCEDURES UTILIZADAS

CALCUSTO; INICIESTA; INICIE; ORDENE;

LIGAÇÃO; DISTÂNCIA; ATEST; IMPRE1.

5) Módulo construtor do vetor dos fins-de-linha e sub-executor do WRP B54

FUNÇÕES:

- Definir e construir o vetor VFIM;
- Definir e construir o vetor SOMAF;
- Fazer a estimativa do número de concentradores necessários;
- Ordenar o VFIM de acordo co o SOMAF.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- VFIM : Vetor que armazena os terminais fins de linha;
- SOMAF : Vetor que contém a soma dos custos das ligações de cada fim-de-linha a todos os outros;

- TFIM : Vetor que armazena os terminais que são os fins-de-linha de cada linha multiponto;
- NF : Número de fins-de-linha;
- NL : Número de linhas;
- LIMAX: Número máximo de linhas por concentrador;
- NC : Número de concentradores.

SUB-MÓDULOS UTILIZADOS:

6;10;11.

PROCEDURES UTILIZADAS:

ORDENE.

6) Módulo seletor dos concentradores e sub-executor do WRP B56

FUNÇÕES:

- Definir a estrutura dos concentradores;
- Definir as procedures: RETIRE, IMPRE3
- Selecionar os concentradores;
- Fazer as ligações dos fins-de-linha aos concentradores.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- Estrutura dos concentradores;
- DEGRAU : Variável que serve de degrau; quando da pesquisa do vetor VFIM para escolha dos concentradores;
- NF : Número de fins-de-linha;
- VFIM : Vetor que armazena os fins-de-linha;
- LI : Vetor que compõe a lista das linhas.

SUB-MÓDULOS UTILIZADOS:

7;8;9.

PROCEDURES UTILIZADAS:

INICIE; PEGUE.

7) Módulo particionador da rede B71

FUNÇÕES:

- Particionar a rede construindo uma partição para cada concentrador composto do concentrador e os terminais de suas linhas;
- Construir o conjunto de linhas multiponto para cada partição e liga-las ao centro (concentrador).

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- NC : Número de concentradores;
- NP : Número de terminais da partição;
- PART: Vetor que contém os terminais da partição;
- SOMAP: Vetor que armazena o custo da ligação de cada terminal ao centro;
- CENTRO: Terminal que é centro da paritição
- Estrutura dos terminais;
- Estrutura das linhas;
- Estrutura dos concentradores;
- LIGADO: Vetor que indica se a linha está, ou não, ligada ao centro;
- MCUST : Matriz de custos;
- MEST : Matriz de estados.

PROCEDURES UTILIZADAS:

INICIESTA ; DEVOLVA; DEVOLVELIN; ORDENE; LIGAÇÃO; DISTÂNCIA; GERELIN; ATEST; PEGUE.

8) Módulo preparador para otimizações B88

FUNÇÕES:

- Construir o novo vetor VFIM;
- Construir o novo vetor SOMAF;
- Ordenar o vetor VFIM de acordo com o SOMAF.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- NF : Número de fins-de-linha;
- VFIM : Vetor dos fins-de-linha atuais;
- SOMAF: Vetor que contém o custo da ligação de cada fim-de-linha ao centro;
- MCUST: Matriz de custos;

PROCEDURES UTILIZADAS:

ORDENE.

9) Módulo otimizados das ligações dos fins-de-linha aos concentradores B90

FUNÇÃO:

Otimizar as ligações dos fins-de-linha aos concentradores, por meio de troca de linhas entre os concentradores, quando isto ocasionar redução de custos.

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

CUST1 : Custo do fim-de-linha ao concentrador atual;

CUST2 : Custo do fim-de-linha ao concentrador não saturado mais próximo dele;

NF : Número de fins-de-linha;

LIMAX : Número máximo de linhas por concentrador;

TRCONMAX: Tráfego máximo por concentrador;

MCUST : Matriz de custos;

CØ : Concentrador selecionado para receber uma linha que será trocada.

PROCEDURES UTILIZADAS:

RETIRE.

10) Módulo otimizador do número de concentradores B94

FUNÇÃO:

Otimizar o número de concentradores, por meio da eliminação de concentradores anti-econômicos, com a religação de suas linhas aos outros concentradores não saturados mais próximos

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- NC1 : Novo número de concentradores;
- LIMITE: Varíavel que aponta para a posição do vetor dos concentradores, até onde deve ser tentata a eliminação de concentradores;
- L170 : Variável que contém a quantidade de linhas, a partir da qual os concentradores que possuirem uma quantidade de linhas igual ou superior,não serão cogitados para eliminação;

- FOLGA : Contém a folga atual, em termos de saturação dos concentradores;
- VAUX1 : Vetor auxiliar que armazena os concentradores;
- VAUX2 : Vetor auxiliar que armazena o número de linhas acumuladas em cada concentrador;
- ACABOU: Variável que indica se já esgotou a tentativa de eliminação de concentradores;
- NC : Número de concentradores;
- LIMAX : Número máximo de linhas por concentradores;
- JØS : Vetor que armazena os melhores concentradores para receber as religações;
- PJØS : Ponteiro do vetor JØS
- Estrutura das linhas;
- Estrutura dos concentradores;
- NL1 : Variável que guarda o número de linhas do concentrador que está sendo considerado;
- ACABE : Variável que controla a pesquisa da tentativa de eliminação de cada concentrador;
- CUSTAD: Custo adicional das religações das linhas;

PROCEDURES UTILIZADAS:

ORDENE; IMPRE2; IMPRE3.

11) Módulo otimizador dos tipos dos concentradores B105

FUNÇÃO:

Otimizar os tipos de concentradores, por meio de substituição de pares de concentradores, por um concentrador maior, desde que suporte a carga (linhas e tráfego) dos dois concentradores e haja redução de custos

PRINCIPAIS VARIÁVEIS E ESTRUTURAS:

- VCON : Vetor que armazena os concentradores;
- CONTR1 : Variável que aponta para as próximas linhas da tabela de restrições e informações;
- Estrutura dos terminais;
- LIMAX2 : Número máximo de linhas para o concentrador, com características apontadas por CONTR1;
- TRCONMX2 : Idem anterior, para tráfego máximo por concentrador;
- CUST12 : Custo de instalação do concentrador, com características apontadas por CONTR1;
- CUST22 : Idem anterior, para custo mensal;
- MDIST : Matriz das distâncias entre os concentradores;
- ACABOU : Variável que controla a busca do par de concentradores mais próximos;

- CUSTAD : Custo adicional com o novo concentrador;
- CUSTAD1 : Diferença entre o custo com os dois concentradores antigos e o novo concentrador;
- MCUST : Matriz de custos;
- MEST : Matriz de estados;

PROCEDURES UTILIZADAS:

DISTÂNCIA, IMPRE2; IMPRE3.

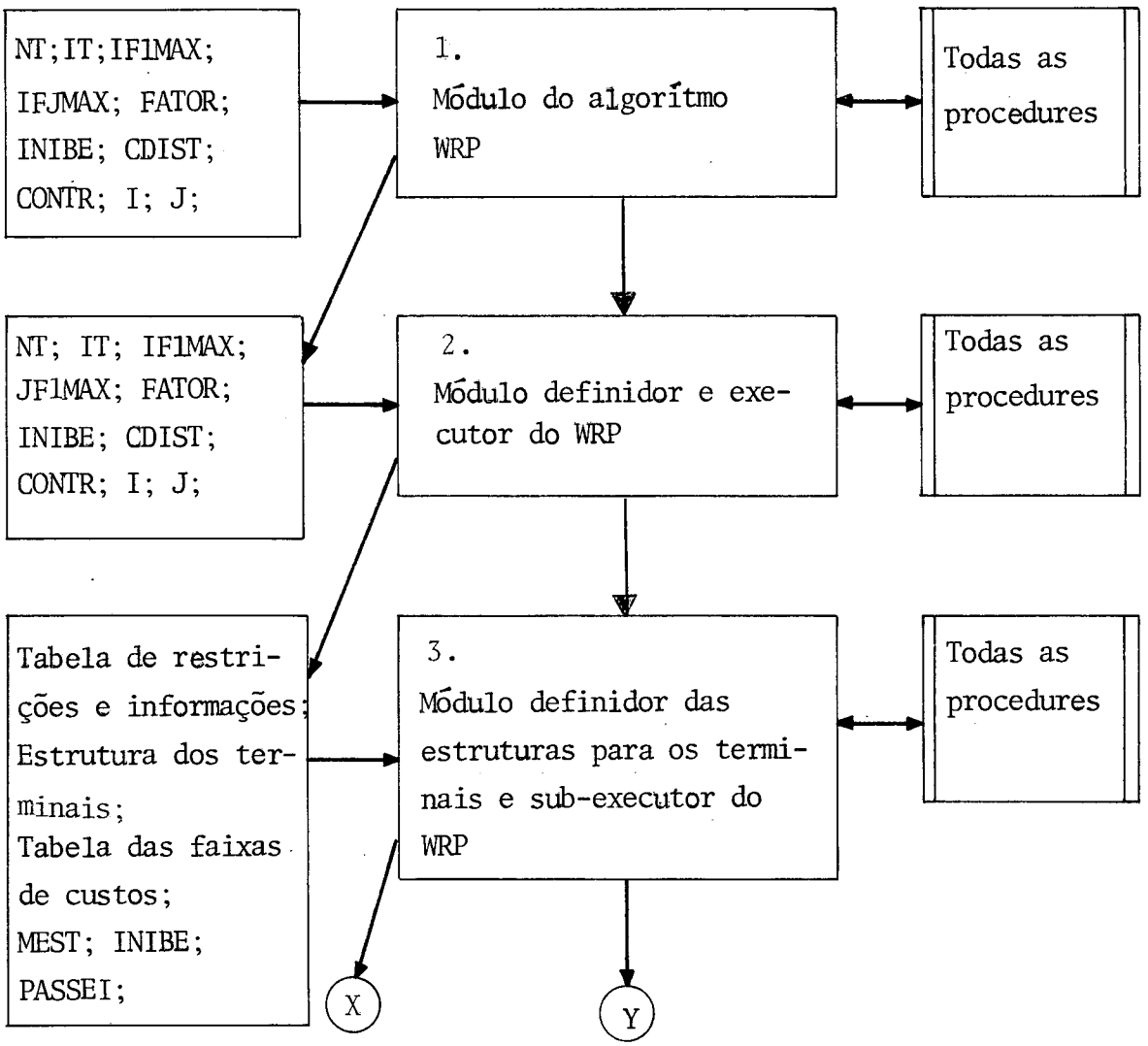
III.9. Diagrama Funcional do Algorítmo

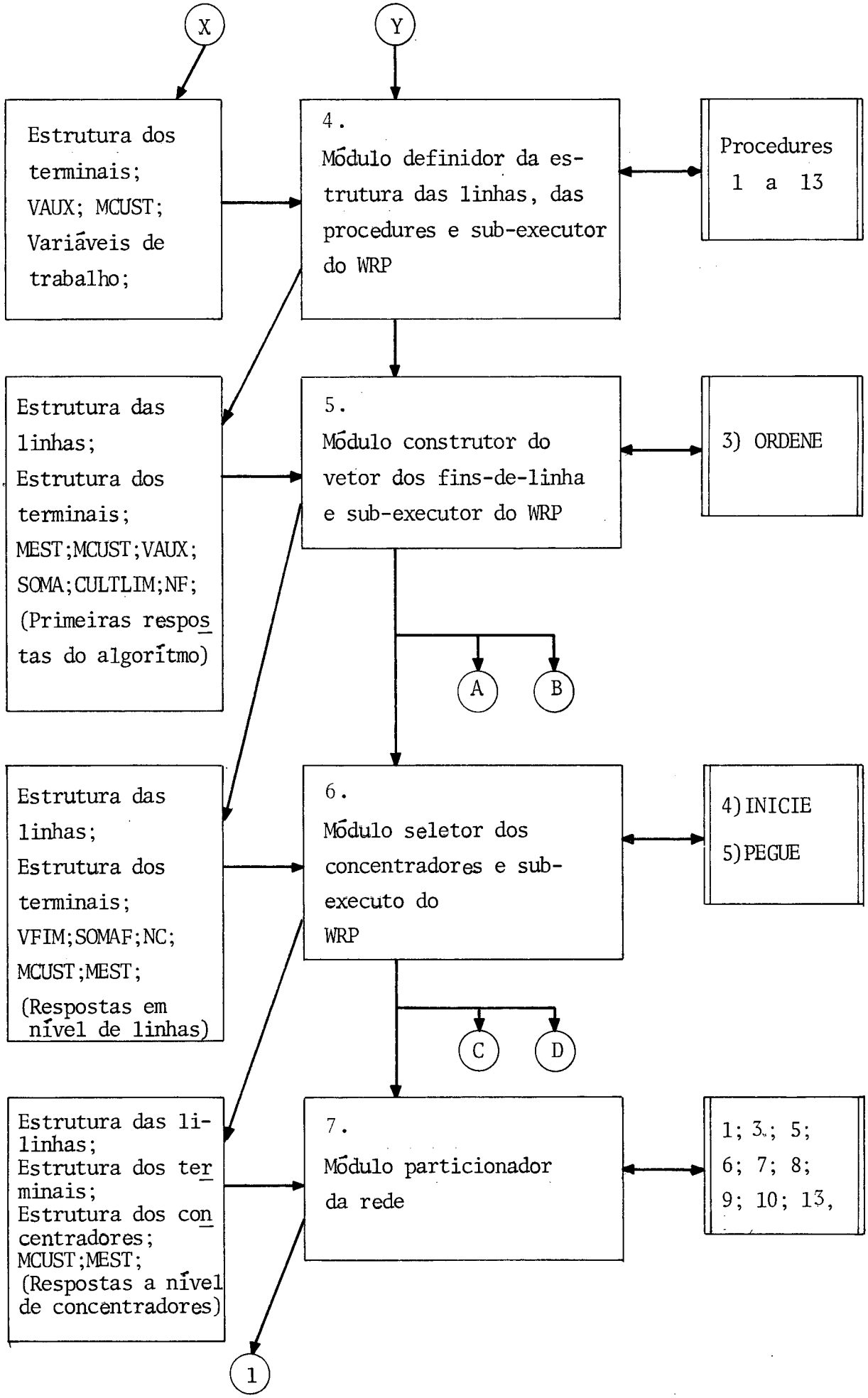
Será apresentado a seguir um diagrama funcional do algorítmo WRP, mostrando a comunicação de dados entre os módulos e as procedures que cada módulo utiliza.

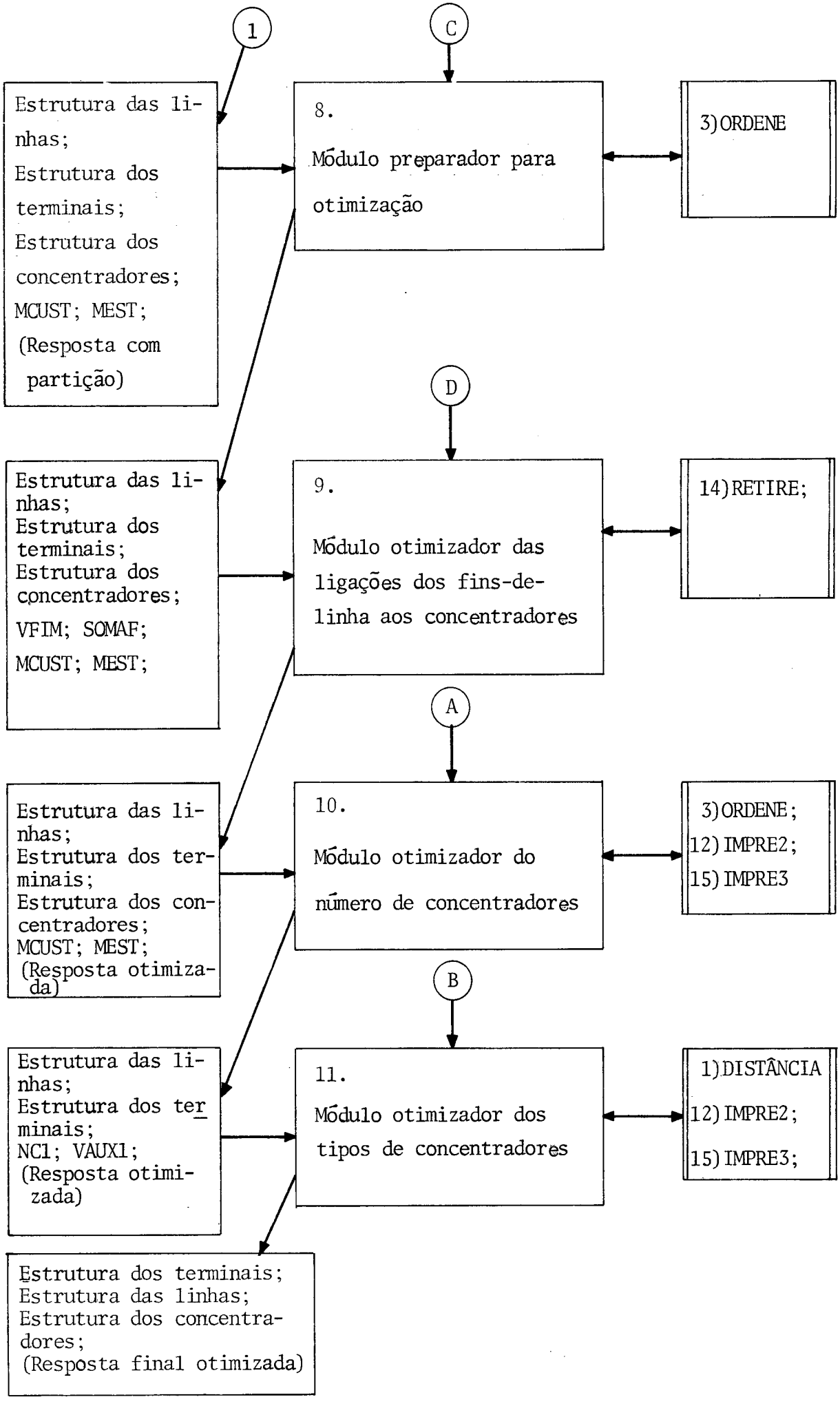
O diagrama está organizado da seguinte maneira:

- No centro, retângulos representando os módulos;
- A esquerda, retângulos representando os dados que entram e saem em cada módulo;
- A direita, as procedures requisitadas por cada módulo.

DIAGRAMA FUNCIONAL:




X
Y
Estrutura dos terminais; VAUX; MCUST; Variáveis de trabalho;
4. Módulo definidor da estrutura das linhas, das procedures e sub-executor do WRP
Procedures 1 a 13
Estrutura das linhas; Estrutura dos terminais; MEST;MCUST;VAUX; SOMA;CULTLIM;NF; (Primeiras respostas do algorítmo)
5. Módulo construtor do vetor dos fins-de-linha e sub-executor do WRP
3) ORDENE
A
B
Estrutura das linhas; Estrutura dos terminais; VFIM;SOMAF;NC; MCUST;MEST; (Respostas em nível de linhas)
6. Módulo seletor dos concentradores e sub-executo do WRP
4) INICIE
5) PEGUE
C
D
Estrutura das li-linhas; Estrutura dos terminais; Estrutura dos concentradores; MCUST;MEST; (Respostas a nível de concentradores)
7. Módulo particionador da rede
1; 3.; 5; 6; 7; 8; 9; 10; 13,
1


1
C
Estrutura das linhas;
Estrutura dos terminais;
Estrutura dos concentradores;
MCUST; MEST;
(Resposta com partição)
8.
Módulo preparador para otimização
3) ORDENE
D
Estrutura das linhas;
Estrutura dos terminais;
Estrutura dos concentradores;
VFIM; SOMAF;
MCUST; MEST;
9.
Módulo otimizador das ligações dos fins-de-linha aos concentradores
14) RETIRE;
A
Estrutura das linhas;
Estrutura dos terminais;
Estrutura dos concentradores;
MCUST; MEST;
(Resposta otimizada)
10.
Módulo otimizador do número de concentradores
3) ORDENE;
12) IMPRE2;
15) IMPRE3
B
Estrutura das linhas;
Estrutura dos terminais;
NC1; VAUX1;
(Resposta otimizada)
11.
Módulo otimizador dos tipos de concentradores
1) DISTÂNCIA
12) IMPRE2;
15) IMPRE3;
Estrutura dos terminais;
Estrutura das linhas;
Estrutura dos concentradores;
(Resposta final otimizada)

III.10 . Análise da Complexidade do Algorítmo

Será apresentada a seguir a análise da complexidade do algorítmo, de uma maneira bem simples dividida nas procedures e blocos que o compõe. Não serão levadas em consideração as partes irrelevantes, em termos de tempo computacional.

1- Passemos então a análise das procedures.

a) Procedure 2 - CALCUSTO

Para cada terminal i, são visitados todos os outros terminais, a partir de i + 1. Assim o número de visitas é:

$$(NT - 1) + (NT - 2) + \ldots + (NT - NT + 1) =$$

$$= 1 + 2 + \ldots + (NT - 1) =$$

$$= \frac{1 + (NT - 1)(NT - 1)}{2} = \frac{(NT)^2 - NT}{2}$$

onde NT = número de terminais

b) Procedure 3 - ORDENE

No pior caso, ou seja, o vetor completamente desordenado, a procedure necessitará do seguinte número de visitas:

$$(NT - 1) + (NT - 2) + \ldots + 1 =$$

$$= 1 + 2 + \ldots + (NT - 1) =$$

$$= \frac{1 + (NT - 1)(NT - 1)}{2} = \frac{(NT)^2 - NT}{2}$$

c) Procedure 4 - INICIE

No pior caso, que é o da inicialização da lista dos terminais, a complexidade é da ordem de:

$$(NT - 1) + 1 = NT$$

d) Procedure 8 - INICIESTA

Como para cada terminal todos os outros são visitados, o número de visitas é:

$(NT)^2$

As outras procedures são irrelevantes em termos de tempo computacional.

2 - Passemos agora à análise dos módulos. Quando das chamadas das procedures, serão considerados os argumentos para o cálculo da complexidade, ao invés dos parâmetros das procedures.

A) Módulo 2.

Tempo gasto na leitura dos vetores que armazenam dados sobre os terminais:

$4(NT)$

B) Módulo 4.

Tempo gasto nas inicializações:

$$(NT)^2 + (NT)^2 + NT + NT + NT = 2(NT)^2 + 3(NT)$$

Tempo gasto na construção das linhas:

$$\frac{(NT)^2 - NT}{2} + (NT)^2$$

C) Módulo 5.

$$NL + NL + (NL)^2 + \frac{(NL)^2 - NL}{2}$$

onde: NL = número de linhas.

d) Módulo 6.

$NL + NL + NC(NL) = 2(NL) + NC(NL)$

onde NC = número de concentradores

e) Módulo 7.

Tempo gasto nas reinicilizações:

$(NT)^2 + NT +$

Tempo gasto na contrução de partições:

$$+ \sum_{i=1}^{NC} \left(\frac{(NP_i)^2 - NP}{2} + (NP_i)^2 + NP_i\right)$$

onde NP_i é o número de terminais da partição i

f) Módulo 8.

Tempo gasto na construção do vetor dos fins-de-linha e sua ordenação:

$$NL + \frac{(NL)^2 - NL}{2}$$

g) Módulo 9.

$NL(NC)$

h) Módulo 10.

Tempo gasto na construção do vetor auxiliar e sua ordenação:

$$NC + \frac{(NC)^2 - NC}{2}$$

Tempo gasto na pesquisa das linhas de cada concentrador (no pior caso):

$$\sum_{i=1}^{NC} NL_i(NC)$$

onde: NL_i é o número de linhas conectadas ao concentrador i

i) Módulo 11.

Tempo gasto na construção do vetor dos concentradores e na construção da matriz das distâncias entre os concentradores:

$$NC + (NC)^2 +$$

Tempo gasto na pesquisa dos pares mais próximos, candidatos a substituição:

$$+ (NC)^2 +$$

Tempo gasto na avaliação de custos e religação (no pior caso):

$$+ 2 \sum_{i=1}^{NC} NL_i(NC)$$

onde: NL_i = número de linhas do concentrador i.

3 - Para simplificar, consideramos agora somente os termos em que aparece NT, pois estes têm influência muito grande no tempo computacional em relação aos demais.

Sendo assim, a complexidade do algorítmo fica da ordem de:

$$4(NT) + 2(NT)^2 + 3(NT) + \frac{1}{2}(NT)^2 + (NT)^2 - \frac{1}{2}(NT) +$$

$$+ (NT)^2 + (NT) =$$

$$= \frac{9}{2}(NT)^2 + \frac{15}{2}(NT) =$$

$= 4,5(NT)^2 + 7,5(NT)$

Dessa forma, verificamos que o tempo de execução do algorítmo é aproximadamente da ordem de $(NT)^2$, considerando apenas os termos mais relevantes, do ponto de vista do tempo de execução.

III.11. Implementação

A implementação do algorítmo foi feita em ALGOL do DEC10 da UFBA - Universidade Federal da Bahia. Para facilitar sua implementação em outros computadores tivemos o cuidado de não usarmos características particulares do ALGOL do DEC10, com exceção das instruções de entrada e saída, por não termos opção. Como tinhamos a facilidade de trabalhar com um terminal, que ficava conectado ao computador utilizado o serviço tipo "time-sharing", optamos pelo arquivo de entrada em disco, para termos a flexibilidade de criá-lo diretamente pelo terminal, usando o editor de textos, ou a partir de cartão, usando um utilitário para transferí-lo para o disco. Adotamos também a técnica de assinalarmos o disco como impressora, sendo portanto os resultados gravados no disco e impressos a seguir por meio de um utilitário.

Esta técnica de utilizarmos o disco para entrada e saída tem a vantagem de reduzir o tempo total de processamento.

O anexo A apresenta uma listagem da compilação do programa.

Para implementar o programa em outro computador é bem provável que seja necessário a alteração das instruções de entrada e saída, pois no ALGOL do DEC10 essas instruções são bem peculiares. As instruções de entrada estão concentradas nos módulos: 1 , 2 e na procedure INICIESTA (para inibições). As saídas estão concentradas nas procedures: IMPRE1, IMPRE2, IMPRE3, PEGUE e GERELIN.

O programa fornece as seguintes informações na saída:

- Parâmetros iniciais: NT, IT, IF1MAX, JF1MAX, FATOR, INIBE e CDIST;
- Tabela das informações e restrições;
- Tabela dos custos das linhas;
- Variável CONTR (Informa a linha da tabela de informações e restrições, correspondente a resposta);
- Estrutura dos terminais;
- Estrutura das linhas;
- Estrutura dos concentradores;
- Custo total das linhas;
- Custo total dos concentradores;
- Custo total da rede.

Os terminais escolhidos para localização dos concentradores são indicados no vetor TCN, na estrutura dos concentradores. Para construir o desenho da rede, basta seguir as ligações indicadas na estrutura dos terminais que partem do índice do terminal (coluna mais a esquerda) e terminam no terminal indicado no vetor EMITE.

Para cada linha da tabela de restrições e informações são geradas duas respostas, uma com apenas um tipo de concentrador e a outra considerando mais de um tipo de concentrador.

III.12. Tempo de Execução

Para uma rede composta de 118 terminais e uma tabela de restrições e informações formada por 3 linhas (o que significa na geração de 3 redes em cada execução) o tempo total de execução variou entre 1 a 1,5 minutos. Vale ressaltar que a CPU não ficava totalmente disponível para o programa, visto que o DEC10 trabalha com multiprogramação.

CAPÍTULO IV

RESULTADOS E DISCUSSÃO

Testamos o algorítmo para diversos tipos de rede, com diferentes características, das quais selecionamos um conjunto de soluções para apresentarmos neste capítulo.

O conjunto dos 118 pontos iniciais, numerados de 1 a 118, representativos das localizações dos terminais, são mostrados no Anexo B.1.

Para a formação da primeira rede, que está apresentada no Anexo B.2, o algorítmo partiu da seguintes condições:

- Custo da ligação = a distância entre os dois pontos;
- No máximo 5 linhas por concentrador,
- No máximo 10 terminais por linha;
- Tráfego máximo numa linha = 2.000 bits/seg.;
- Tráfego máximo num concentrador = 40.000 bits/seg.;
- FATOR = 16(veja pág.91)
- Sem inibições;
- Sem pré-fixações;
- Custo de instalação de cada concentrador = 50.000,00;
- Custo mensal de cada concentrador = 5.000,00;

A rede gerada ficou composta por 3 concentradores, tendo, cada concentrador 4 linhas multiponto, apresentando um custo total de 165.442,00.

Vale frisar que quando aparecem ligações que to-

cam no círculo representativo do terminal escolhido para concentrador, isto significa que elas fazem parte da mesma linha e não de linhas diferentes, como poderia pensar-se. Por exemplo, observe no Anexo B.2, no concentrador localizado no terminal 39 (os concentradores são representados por quadrados) cujas ligações 52-39 e 20-39 pertencem a mesma linha.

Para formar a rede apresentada no Anexo B.3,o algorítmo partiu das seguintes condições:

- Custo da ligação = distância entre dois pontos;
- Número ilimitado de linhas por concentrador;
- No máximo 12 terminais por linha;
- Tráfego máximo numa linha = 2.000 bits/seg.;
- Tráfego máximo no concentrador = 40.000 bits/seg.;
- FATOR = 16;
- Sem inibições;
- Sem pré-fixação;
- Custo de instalação do concentrador = 80.000,00;
- Custo mensal do concentrador = 8.000,00;

Como o concentrador não foi limitado o algorítmo gerou uma rede centralizada, com o custo total = 88.510,00. O custo ficou muito menor que a rede do Anexo B.2, devido ao alto peso do custo dos concentradores em relação as linhas, visto que a distância está sendo considerada como custo de ligação.

Para a rede apresentada no Anexo B.4, as condições foram as mesmas da rede do Anexo B.2, com exceção de que foram feitas inibições para as ligações 89-52 e 92-54. O custo total

desta rede foi de 165.444,00, ou seja, 2,00 a mais que o custo da rede sem inibições.

A rede do Anexo B.5 foi formada a partir das mesmas condições da rede do Anexo B.4 (inclusive com as inibições) tendo porém duas ligações pré-fixadas: 36-39 e 37-40. O custo total desta rede foi de 165.510,00.

Para as redes seguintes, o algorítmo utilizou uma tabela de custos das linhas,que representa o custo, em função da distância e da capacidade, a qual será mostrada a seguir:

TABELA DOS CUSTOS DAS LINHAS DE COMUNICAÇÃO

(em Cr$)

DEGRAUS TARIFÁRICOS	VELOCIDADE EM BIST/SEG.			
	2.000		4.000	
	CUSTO INSTALAÇÃO	CUSTO MENSAL	CUSTO INSTALAÇÃO	CUSTO MENSAL
2	1.000,00	1.700,00	2.000,00	2.600,00
3	1.000,00	3.200,00	2.000,00	4.800,00
4	1.000,00	4.400,00	2.000,00	6.300,00
5	1.000,00	5.900,00	2.000,00	8.800,00
6	1.000,00	8.200,00	2.000,00	12.400,00
7	1.000,00	8.800,00	2.000,00	13.200,00
8	1.000,00	9.400,00	2.000,00	14.100,00
9	1.000,00	9.900,00	2.000,00	14.800,00
10	1.000,00	10.000,00	2.000,00	16.100,00

DEGRAUS TARIFÁRICOS	DISTÂNCIA EM KM
2	Até 1
3	1 a 2
4	2 a 3
5	3 a 4
6	4 a 6
7	6 a 10
8	10 a 20
9	20 a 50
10	Acima de 50

Obs.: Nas figuras que aparecem nos anexos cada Km é representado por um Cm (centímetro).

Para a rede apresentada no Anexo B.6, o algorítmo partiu das seguintes condições:

- Custo das ligações, de acordo com a tabela dos custos das linhas de comunicação.
- No máximo 5 linhas por concentrador
- No máximo 8 terminais por linha
- Tráfego máximo num concentrador = 40.000 bits/seg.;
- Tráfego máximo numa linha = 2.000 bits/seg.;
- FATOR = 256;
- Sem inibições;
- Sem pré-fixações;
- Custo de instalação do concentrador = 50.000,00;

- Custo mensal do concentrador = 5.000,00.

Esta rede apresentou um custo total = 890.700,00., porém, como o algorítmo fornece também uma resposta onde existe vários tipos de concentradores (desde que estejam caracterizados na tabela de restrições e informações) o Anexo B.7 mostra esta mesma rede com a utilização de um concentrador (o localizado no terminal 75) que suporta 15 linhas por concentrador, resultando num custo total da rede de 862.300,00 , menor que o custo da rede que utiliza apenas um tipo de concentrador.

Para formar a rede apresentada no Anexo B.8 as condições iniciais foram as mesmas da rede do Anexo B.6, salvo a seguinte diferença:

- No máximo 12 terminais por linha

Para linha desse tipo, o custo total da rede baixou para 817.200,00.

Quando o algorítmo considerou a mistura de tipos diferentes de concentradores, o custo total ficou em 815.800,00, cuja rede equivalente aparece no Anexo B.9. Os concentradores que ficavam nos terminais 99 e 68 foram substituídos por um concentrador maior, com capacidade para suportar 10 linhas, que ficou localizado no terminal 99.

A rede mostrada no Anexo B.10 foi construída a partir das mesmas condições da rede do Anexo B.6, com a seguinte exceção:

- No máximo 1 terminal por linha.

Esta restrição obriga o algorítmo a gerar uma rede multicentro-multiestrela, ou seja, vários concentradores com os terminais ligados a eles por meio de linhas ponto-a-ponto. O custo total desta rede foi de 1.930.199,00. Este alto custo foi devido ao grande número de concentradores utilizados. Observe que o algorítmo usou dois tipos de concentradores: uns com capacidade para 5 linhas e outros com capacidade para 10 linhas.

A rede do Anexo B.11 é do tipo centralizada e, para formá-la o algorítmo partiu das mesmas condições iniciais que a rede do Anexo B.6, com as seguintes diferenças.

- O concentrador é capaz de suportar qualquer quantidade de linhas.
- No máximo 12 terminais por linha.
- Custo de instalação de concentrador = 90.000,00.
- Custo mensal do concentrador = 9.000,00.

A capacidade ilimitada do concentrador faz com que o algorítmo defina uma rede centralizada. O custo total da rede foi de 812.300,00.

O Anexo C mostra um gráfico onde aparece o custo total de várias redes em função de vários tipos de linhas, para vários tipos de concentradores.

Dessa forma, o algorítmo pode oferecer uma série de opções ao usuário, de acordo com seus recursos que ele possua e permite ainda a flexibilidade de fazer inibições com certas ligações indesejáveis, por razões políticas, de segurança ou qualquer outra, além de permitir pré-fixação de liga

ções, o que torna o algorítmo disponível para fazer ampliações de redes já existentes.

Comparando com outros algorítmo podemos concluir que, embora o WRP não seja o mais rápido ou o mais econômico em utilização de memória, certamente ele é o mais flexível e versátil sobre os seguintes aspectos:

- Permite que se tenha ligações já existentes;
- Permite que se faça inibições de ligações indesejadas;
- Fornece duas opções para o usuário em cada rede: uma em que utiliza um só tipo de concentrador e outra em que mistura vários tipos de concentradores;
- Oferece uma variedade de opções de configurações de redes tão grande quanto se queira;
- Decide quanto ao número, localização e tipo dos concentradores.

CAPÍTULO V

CONCLUSÕES

Na primeira parte deste trabalho apresentamos um estudo comparativo abrangendo uma série de algorítmos e estudos sobre Topologia de Redes de Comunicações de Dados, tendo o cuidado de classificar os diversos tipos de redes para então apresentar o que existe de importante na literatura para resolver os problemas topológicos de cada tipo de rede.

Na segunda parte apresentamos o algorítmo que desenvolvemos para definir Topologias de Redes de Comunicação de Dados (WRP), mostrando sua filosofia, os dados iniciais que necessita, a estrutura de dados para armazenamento interno, a resposta fornecida, o diagrama funcional e as demais informações que possibilitem sua compreensão e utilização. Destacamos ainda as vantagens do algorítmo desenvolvido, quanto à flexibilidade e versatilidade de soluções.

Acreditamos termos prestado uma modesta contribuição para o campo de pesquisas sobre redes aqui no Brasil e esperamos que o algorítmo WRP venha a ser utilizado como uma "ferramenta" para os técnicos envolvidos no trabalho de definicação topológica de redes em nosso país.

BIBLIOGRAFIA

(1) BOORR77 — BOORSTYN, R.R., H. Frank; Large Scale Network Topological Otimization; IEEE - Transactions on Communications, vol.COM-35, nº 10, Oct.77, pags.29-47;

(2) BUHRR78 — BUHR, R.J. , C.M. Woodside; The Optimal Number of Nodes in a Star-Configured Distributed Computer System; IEEE - Transactions on Systems, Man, and Cybernetics, vol.SMC-8, nº 1, Jan.78, pags.76-79;

(3) CHOUW78 — CHOU, W., F. Ferrante, M. Galagandhar, M. Gerke; An Algorithm for Optimal Locating Network Access Facilities; ICC/78; Toronto, Canadá, june 1978, pags. 24.5.1-24.5.8;

(4) DIRIH77 — DIRIILTEN, H., R. W. Donaldson; Topological Design of Distributed Data Communication Networks Using Linear Regression Clustering; IEEE - Transaction on Communications, vol.COM-25, nº 10, Oct.77, pags.83-92;

(5) DYSAH78 DYSART, H.G., N.D. Georgans; NEWCLUSTER: An Algorithm for the Topological Design of Two-Level Multipoint Teleprocessing Networks; IEEE - Transactions on Communications, vol.COM-26, nº 1, Jan.78, pags.55-74;

(6) ELIAD74 ELIAS, D., M.J. Ferguson; Topological Design of Multipoint Teleprocessing Networks; IEEE - Transaction on Communications, vol.COM-22, nº 11, Nov.74, pags.1753-62;

(7) GERLM77 GERLA, M., L. Kleinrock; On the Topological Design of Distributed Computer Networks; IEEE - Transactions on Communications, vol.COM-25, nº 1, Jan.77, pags.48-60;

(8) GREGP77 McGREGOR, P.V., Diana Shen; Network Design: An Algorithm for the Access Facility Location Problem; IEEE - Transactions on Communications, vol.COM-25, nº 1, Jan.77, pags.61-72;

(9) KARNE76 KARNAUGH, M.; A New Class of Algorithms of Multipoint Networks Optimization; IEEE - Transactions on Communications, vol.COM-24, nº 5, May 76, pags.500-506;

(10) KERSA74 KERSHENBAUM, A., W. CHOU;
A Unified Algorithm for Designing Multidrop Teleprocessing Networks;
IEEE - Transactions on Communications, vol.COM-22, nº 11, Nov.74, pags.62-71;

(11) MARTJ72 MARTIN, James;
System Analysis for Data Transmission, (Cap.39-43, pags.677-773);
Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, 910 pags.;

(12) SCHWG78 SCHWARZ, Gerhard;
Redes de Computadores - Uma Análise Qualitativa; Publicação da COPPE/UFRJ, PDD-11/78, 1978, 115 págs.

(13) SCHWM76 SCHWARTZ, Mischa;
Computer Communication Network-Design and Analysis;
(Cap.2-4,9-10, pags.13-84,171-211);
Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976, 372 pags.;

(14) SHARR77 SHARMA, Roshan L.;
Design of Economical Distributed, Switched Communication Networks;
EUROCON/77,Veneza,Itália,Mag.1977, pags.3.4.2.1-3.4.2.6;

(15) TAROL77 TAROUCO, Liane M.R.;
Redes de Comunicação de Dados,
(Cap.3, pags.40-88);
Livros Técnicos e Científicos Editora
S.A., Rio de Janeiro - RJ, 1977,
176 pags.;

ANEXO A

LISTAGEM DA COMPILAÇÃO DO PROGRAMA IMPLEMENTADOR DO ALGORÍTMO WRP

```
000000                  1 !-------------------------------------------------------------------------------;
000000                  2 !  PROGRAMA DE IMPLEMENTACAO DO ALGORITMO WRP PARA DEFINICAO TOPOLO-           ;
000000                  3 !  GICA DE REDES DE COMUNICACAO DE DADOS                                       ;
000000                  4 !  WALTER ROCHA PALMA - 1979                                                  ;
000000                  5 !-------------------------------------------------------------------------------;
000000                  6 ! DECLARACAO DE VARIAVEIS GLOBAIS QUE DIMENSIONAM AS ESTRUTURAS;
000004   B1             7 BEGIN
START OF BLOCK 1
000005                  8 INTEGER NT,IT,IF1MAX,JF1MAX,I,J,CONTR,FATOR,INIBE,CDIST;
000005                  9 INPUT(10,"DSK");
000011                 10 SELECTINPUT(10);
000014                 11 OPENFILE(10,"WRPTOP.CDR");
000020                 12 OUTPUT(11,"LPT");
000024                 13 SELECTOUTPUT(11);
000027                 14 OPENFILE(11,"WRPTOP.SAI");
000027                 15 !-------------------------------------------------------------------------------;
000027                 16 ! LEITURA DOS PARAMETROS INICIAIS                                              ;
000027                 17 ! DECLARACAO DA TABELA INICIAL E DAS FAIXAS DE CUSTOS                         ;
000027                 18 !-------------------------------------------------------------------------------;
000033                 19 READ(NT,IT,IF1MAX,JF1MAX,FATOR,INIBE,CDIST);
000044   B2            20 BEGIN
START OF BLOCK 2
000065                 21 INTEGER ARRAY VLIMAX[1:IT],VTEMAX[1:IT],VIF1[1:IT],
000121                 22                VPCENT[1:IT],VTRCONMAX[1:IT],VTRLIMAX[1:IT],X[1:NT],
000155                 23                Y[1:NT],TRAF[1:NT],SINT[1:NT],FAIX1[1:JF1MAX];
000170                 24 REAL ARRAY FAIXA1[1:IF1MAX,1:JF1MAX,1:2],
000206                 25             VCUSTCON1[1:IT],VCUSTCON2[1:IT];
000206                 26 !-------------------------------------------------------------------------------;
000206                 27 ! DECLARACAO DA MATRIS DE ESTADOS E DA BOOLEANA QUE CONTROLA AS MARCAS          ;
000206                 28 ! DE INIBICAO                                                                 ;
000206                 29 !-------------------------------------------------------------------------------;
000217                 30 INTEGER ARRAY MEST[1:NT,1:NT];
000224                 31 BOOLEAN PASSEI;
000224                 32 !-------------------------------------------------------------------------------;
000224                 33 ! ENCHER A TABELA INICIAL DE RESTRICOES, CUSTOS E INFORMACOES SOBRE            ;
000224                 34 ! OS TERMINAIS                                                                ;
000224                 35 !-------------------------------------------------------------------------------;
000226                 36 FOR I:=1 STEP 1 UNTIL IT DO
000257                 37     READ(VLIMAX[I],VTRCONMAX[I],VPCENT[I],VCUSTCON1[I],VCUSTCON2[I],
000303                 38          VTEMAX[I],VTRLIMAX[I],VIF1[I]);
000320                 39 FOR I:=1 STEP 1 UNTIL NT DO
000351                 40     READ(X[I],Y[I],TRAF[I],SINT[I]);
```

```
000351                41 !-----------------------------------------------------------------------;
000351                42 ! LEITURA DAS FAIXAS DE CUSTOS DAS LINHAS DE COMUNICACAO               ;
000351                43 !-----------------------------------------------------------------------;
000374                44 FOR I:=1 STEP 1 UNTIL JF1MAX DO READ (FAIX1[I]);
000402                45 FOR I:=1 STEP 1 UNTIL IF1MAX DO
000410                46     FOR J:=1 STEP 1 UNTIL JF1MAX DO
000433                47         READ(FAIXA1[I,J,1],FAIXA1[I,J,2]);
000433                48 !-----------------------------------------------------------------------;
000433                49 ! CONTROLE SUPERIOR DO ALGORITMO.                                      ;
000433                50 ! EXECUTA TODOS OS PROCEDIMENTOS PARA CADA CONJUNTO DE PARAMETROS      ;
000433                51 ! FORMADO POR UMA LINHA DA TABELA INICIAL                            ;
000433                52 !-----------------------------------------------------------------------;
000443                53 FOR CONTR := 1 STEP 1 UNTIL IT DO
000447   B3           54 BEGIN
000447                55 !-----------------------------------------------------------------------;
000447                56 ! DECLARACAO DAS ESTRUTURAS RELATIVAS AOS TERMINAIS;
000447                57 ! DECLARACAO DE VARIAVEIS GERAIS DO PROGRAMA;
000447                58 !-----------------------------------------------------------------------;
START OF BLOCK 3
000506                59 INTEGER ARRAY LINHA[1:NT],EMITE[1:NT],T[1:NT],PROXT[1:NT],VAUX[1:NT];
000526                60 REAL ARRAY MCUST[1:NT,1:NT],SOMA[1:NT];
000533                61 INTEGER LIMAX,TEMAX,IF1,IF2,TRCONMAX,TRLIMAX,I,J,IO,JO,NL,L,AUX1,AUX2,
000533                62         DISPL,DISPT,NF,NC,LO,PCENT,DIST1,DIST2,LAMB,TRLIBER;
000533                63 REAL INFINITO,MINIC,CUSTCON1,CUSTCON2,TOTCUSTLIN,TOTCUSTCON,TOTCUST,
000533                64      CUSTMED,CUSTLIM;
000533                65 BOOLEAN ACABOU,LIGOU;
000533                66 NF := 0;
000534                67 LAMB := 0;
000535                68 INFINITO := 1.7&38;
000545                69 TRCONMAX := VTRCONMAX[CONTR]; CUSTCON1 := VCUSTCON1[CONTR];
000561                70 CUSTCON2 := VCUSTCON2[CONTR]; PCENT := VPCENT[CONTR];
000575                71 TRLIMAX := VTRLIMAX[CONTR]; LIMAX := VLIMAX[CONTR];
000611                72 TEMAX := VTEMAX[CONTR]; IF1 := VIF1[CONTR];
000611                73 !-----------------------------------------------------------------------;
000611                74 ! ESTIMATIVA DO NUMERO DE LINHAS MULTIPONTO NECESSARIAS                ;
000611                75 !-----------------------------------------------------------------------;
000617                76 NL := ENTIER(NT/2 + 2);
000633                77 IF TEMAX = 1 THEN NL := NT;
000635   B4           78 BEGIN
START OF BLOCK 4
000640                79 REAL ARRAY CUSTACUM[1:NL];
000703                80 INTEGER ARRAY TFIM[1:NL],TEACUM[1:NL],CONC[1:NL],LSTE[1:NL],TRACUM[1:NL],
000721                81                LI[1:NL],PROXL[1:NL];
000721                82 !-----------------------------------------------------------------------;
```

```
000733                  83   INTEGER PROCEDURE DISTANCIA(XX,YY);
000737                  84            INTEGER XX,YY;
000741    B5            85   BEGIN
START OF BLOCK 5
000742                  86   REAL AUX1;
000742                  87   AUX1 := (X[XX] - X[YY])^2  +  (Y[XX] - Y[YY])^2;
001004                  88   DISTANCIA := SQRT(AUX1);
001010         E5       89   END;
END BLOCK 5, CONT 4
001010                  90 !-------------------------------------------------------------------------------------------------;
001014                  91   PROCEDURE CALCUSTO;
001024    B6            92   BEGIN
001026                  93   FOR I:=1 STEP 1 UNTIL NT DO
001032    B7            94        BEGIN
001035                  95        FOR J:=I+1 STEP 1 UNTIL NT DO
001041    B8            96             BEGIN
001041                  97             AUX1 := DISTANCIA(I,J);
001046                  98             IF CDIST # 1
001051    B9            99             THEN BEGIN
001053                 100                  FOR L:=1 STEP 1 UNTIL JF1MAX DO
001057                 101                       IF FAIX1[L] >= AUX1
001066  B10            102                       THEN BEGIN
001066                 103                            MCUST[I,J] := FAIXA1[IF1,L,1] + FAIXA1[IF1,L,2];
001112                 104                            MCUST[J,I] := MCUST[I,J];
001126                 105                            L := JF1MAX + 1;
001131         E10     106                            END;
001132          E9     107                  END
001133  B11            108             ELSE BEGIN
001133                 109                  MCUST[I,J] := AUX1;
001144                 110                  MCUST[J,I] := AUX1;
001155         E11     111                  END;
001155          E8     112             END;
001156                 113        MCUST[I,I] := 0;
001164          E7     114        END;
001165          E6     115   END;
001165                 116 !-------------------------------------------------------------------------------------------------;
```

```
001171                  117   PROCEDURE ORDENE(VET1,VET2,DIM);
001175            3      118             VALUE DIM; INTEGER DIM;
001175                   119             INTEGER ARRAY VET1;  REAL ARRAY VET2;
001200  B12              120   BEGIN
START OF BLOCK 6
001201                   121   BOOLEAN FIM;
001201                   122   INTEGER I,AUX1;
001201                   123   REAL AUX2;
001201                   124   WHILE NOT FIM DO
001203  B13              125         BEGIN
001203                   126         FIM := TRUE;
001206                   127         FOR I:=1 STEP 1 UNTIL DIM-1 DO
001214                   128             IF VET2[I] > VET2[I+1]
001231  B14              129             THEN BEGIN
001254                   130                  AUX1:=VET1[I]; VET1[I]:=VET1[I+1]; VET1[I+1]:=AUX1;
001307                   131                  AUX2:=VET2[I]; VET2[I]:=VET2[I+1]; VET2[I+1]:=AUX2;
001317                   132                  FIM := FALSE;
001320          E14      133                  END;
001321                   134         DIM := DIM - 1;
001324          E13      135         END;
001325          E12      136   END;
END BLOCK 6, CONT 4
001325                   137 !-------------------------------------------------------------------------------;
001335                   138   PROCEDURE INICIE(LISTA,DIM,DISP);
001341                   139             INTEGER DIM,DISP;  INTEGER ARRAY LISTA;
001344  B15              140   BEGIN
001347                   141   FOR DISP := 1 STEP 1 UNTIL DIM-1 DO
001370                   142             LISTA[DISP] := DISP + 1;
001403                   143   LISTA[DIM] := LAMB;
001412                   144   DISP := 1;
001415          E15      145   END;
001415                   146 !-------------------------------------------------------------------------------;
```

```
001421                  147   PROCEDURE PEGUE(PROX,DISP,TX);
001425                  148             INTEGER DISP,TX;  INTEGER ARRAY PROX;
001430  B16             149   BEGIN
001430                  150   IF DISP = LAMB
001433  B17             151   THEN BEGIN
001433                  152        WRITE("[C]********** ESTOURO DE LISTA ********** MENS 1")
001433                  153        !STOP;
001433        E17       154        END
001437  B18             155   ELSE BEGIN
001437                  156        TX := DISP;
001445                  157        DISP := PROX[DISP];
001460                  158        PROX[TX] := LAMB;
001467        E18       159        END;
001467        E16       160   END;
001467                  161 !-------------------------------------------------------------------------------!
001474                  162   PROCEDURE DEVOLVA(PROX,DISP,TX);
001500                  163             INTEGER DISP,TX;  INTEGER ARRAY PROX;
001503  B19             164   BEGIN
001503                  165   PROX[TX] := DISP;
001514                  166   DISP := TX;
001522        E19       167   END;
001522                  168 !-------------------------------------------------------------------------------;
001526                  169   PROCEDURE ATEST(IEST,JEST);
001532                  170             INTEGER IEST,JEST;
001534  B20             171   BEGIN
001546                  172   IF MEST[IEST,JEST] = 0 THEN MEST[IEST,JEST] := -1;
001556        E20       173   END;
001556                  174 !-------------------------------------------------------------------------------;
```

```
001557                  175   PROCEDURE INICIESTA;
001567  B21             176   BEGIN
START OF BLOCK 7
001570                  177   INTEGER I,J;
001572                  178   FOR I:=1 STEP 1 UNTIL NT DO
001576  B22             179       BEGIN
001600                  180       FOR J:=1 STEP 1 UNTIL NT DO
001604                  181           IF MEST[I,J] < 0
001614                  182           THEN MEST[I,J] := 0;
001623                  183       MEST[I,I] := -1;
001631                  184       IF SINT[I] # 0
001642                  185       THEN FOR J := 1 STEP 1 UNTIL NT DO
001655, B23             186                IF SINT[I] # J THEN BEGIN
001655                  187                                    MEST[I,J] := 2;
001664                  188                                    MEST[J,I] := 2;
001673        E23       189                                    END
001674                  190                                ELSE MEST[I,J] := 0;
001703        E22       191       END;
001704                  193   IF (INIBE = 1) AND (NOT PASSEI)
                        001713  B24          194 THEN BEGIN
START OF BLOCK 8
001714                  195           INTEGER I,J;
001714                  196           ACABOU := FALSE;
001715                  197           PASSEI := TRUE;
001716                  198           WHILE NOT ACABOU DO
001720  B25             199                 BEGIN
001720                  200                 READ(I,J);
001727                  201                 IF I = 0  THEN ACABOU := TRUE
001731  B26             202                           ELSE BEGIN
001731                  203                                MEST[I,J] := 1;
001740                  204                                MEST[J,I] := 1;
001747        E26       205                                END;
001747        E25       206                 END;
001750        E24       207           END;
END BLOCK 8, CONT 7
001753        E21       208   END;
END BLOCK 7, CONT 4
001753                  209 !-------------------------------------------------------------------------------;
```

```
001763                  210   PROCEDURE GERELIN(TO)§
001767                  211             INTEGER TO§
001770  B27             212   BEGIN
START OF BLOCK 9
001771                  213   BOOLEAN ACABOU§
001771                  214   INTEGER P§
001771                  215   P := 1§
002000                  216   WHILE NOT ACABOU DO IF NL < P THEN ACABOU := TRUE
002011                  217                                 ELSE IF TFIMCP] = 0 THEN ACABOU := TRUE
002013                  218                                                    ELSE P := P + 1§
002017                  219   IF P > NL
002022  B28             220   THEN BEGIN
002022                  221        WRITE("CC]********** ESTOURO NA LISTA DE LINHAS ********** MENS 2")§
002022                  222        ! STOP§
002025        E28       223        END
002026                  224   ELSE L := P§
002063                  225   TFIMCL] := TO§  TRACUMCL] := TRAFCTO]§  TEACUMCL] := 1§ CONCL] := 0§
002102                  226   LSTECL] := 0§  CUSTACUMCL] := 0§  LINHACTO] := L§
002111        E27       227   END§
END BLOCK 9, CONT 4
002111                  228 !-------------------------------------------------------------------------------------------------------------------------------§
002121                  229   PROCEDURE DEVOLVELIN(LIN)§
002125                  230             INTEGER LIN§
002126  B29             231   BEGIN
002142                  232   TFIMCLIN] := 0§  TRACUMCLIN] := 0§  TEACUMCLIN] := 0§
002164                  233   CUSTACUMCLIN] := 0§  CONCLIN] := 0§  LSTECLIN] := 0§
002172        E29       234   END§
002172                  235 !-------------------------------------------------------------------------------------------------------------------------------§
```

```
002173                  236    PROCEDURE IMPRE1;
002203   B30            237    BEGIN
START OF BLOCK 10
002204                  238    INTEGER I,J,L;
002204                  239    WRITE("[P2C39S]*  S A I D A    D O     A L G O R I T M O     W R P  *")
002222                  240    WRITE("[2C9S]NT=");PRINT(NT,3,0); WRITE("[4S]IT=");PRINT(IT,3,0);
002237                  241         WRITE("[4S]IF1MAX=");PRINT(IF1MAX,3,0); WRITE("[4S]JF1MAX=");
002252                  242         PRINT(JF1MAX,3,0); WRITE("[4S]FATOR=");PRINT(FATOR,4,0);
002262                  243         WRITE("[4S]INIBE=");PRINT(INIBE,2,0);
002272                  244         WRITE("[4S]CDIST=");PRINT(CDIST,2,0);
002277                  245    WRITE("[2C39S]TABELA INICIAL DE RESTRICOES E INFORMARCOES[2C9S]");
002302                  246    WRITE("LI-CON-MAX  TR-CON-MAX  % LIBER     CUSTO-1         CUSTO-2");
002305                  247    WRITE("[5S]TE-LI-MAX  TR-LI-MAX       IF1");
002312                  248    FOR I:=1 STEP 1 UNTIL IT DO
002316   B31            249         BEGIN
002334                  250         WRITE("[C]");PRINT(I,4,0);PRINT(VLIMAX[I],6,0);
002367                  251         PRINT(VTRCONMAX[I],13,0);PRINT(VPCENT[I],10,0);WRITE(" %");
002413                  252         PRINT(VCUSTCON1[I],10,2);PRINT(VCUSTCON2[I],10,2);
002441                  253         PRINT(VTEMAX[I],10,0);PRINT(VTRLIMAX[I],12,0);
002454                  254         PRINT(VIF1[I],8,0);
002461        E31       255         END;
002462                  256    WRITE("[2C39S]TABELA DOS CUSTOS DAS LINHAS[2C]");
002503                  257    FOR I:=1 STEP 1 UNTIL (JF1MAX - 1) DO PRINT(FAIX1[I],12,0);
002511                  258    WRITE("[2S]INFINITO");
002516                  259    FOR I:=1 STEP 1 UNTIL IF1MAX DO
002522   B32            260         BEGIN
002525                  261         WRITE("[2C]");PRINT(I,4,0);
002550                  262         FOR J:=1 STEP 1 UNTIL JF1MAX DO PRINT(FAIXA1[I,J,1],8,2);
002556                  263         WRITE("[C5S]");
002577                  264         FOR J:=1 STEP 1 UNTIL JF1MAX DO PRINT(FAIXA1[I,J,2],8,2);
002605        E32       265         END;
002606        E30       266    END;
END BLOCK 10, CONT 4
002606                  267 !-------------------------------------------------------------------------------------------------------------------;
```

```
002612                  268    PROCEDURE IMPRE2;
002622  B33             269    BEGIN
START OF BLOCK 11
002623                  270    INTEGER I;
002626                  271    WRITE("[3C9S]CONTR=");  PRINT(CONTR,2,0);
002633                  272    WRITE("[2C9S]ESTRUTURA DOS TERMINAIS[2C9S]SINT      X         Y");
002636                  273    WRITE("[6S]TRAFE LINHA EMITE[11S]SOMA[7S]VAUX[9S]T[5S]PROXT");
002643                  274    FOR I:=1 STEP 1 UNTIL NT DO
002647  B34             275        BEGIN
002700                  276        WRITE("[C]");  PRINT(I,4,0);  PRINT(SINT[I],4,0);  PRINT(X[I],8,0);
002741                  277        PRINT(Y[I],8,0);  PRINT(TRAF[I],8,0);  PRINT(LINHA[I],5,0);
003002                  278        PRINT(EMITE[I],6,0);  PRINT(SOMA[I],14,2);  PRINT(VAUX[I],9,0);
003030                  279        PRINT(T[I],8,0);  PRINT(PROXT[I],8,0);
003035        E34       280        END;
003036                  281    WRITE("[3C9S]ESTRUTURA DAS LINHAS[2C9S]TFIM      TR-ACUM   TE-ACUM");
003041                  282    WRITE("[7S]CUST-ACUM    CON   LSTE    LI       PROXL");
003046                  283    FOR I:=1 STEP 1 UNTIL NL DO
003052  B35             284        BEGIN
003070                  285        WRITE("[C]");  PRINT(I,4,0);  PRINT(TFIM[I],5,0);
003116                  286        PRINT(TRACUM[I],11,0);  PRINT(TEACUM[I],6,0);
003157                  287        PRINT(CUSTACUM[I],14,2);  PRINT(CON[I],5,0);  PRINT(LSTE[I],5,0);
003205                  288        PRINT(LI[I],7,0);  PRINT(PROXL[I],8,0);
003212        E35       289        END;
003213        E33       290    END;
END BLOCK 11, CONT 4
003213                  291  !-------------------------------------------------------------------------------;
```

```
003217                  292   PROCEDURE LIGACAO;
003227  B36             293   BEGIN
START OF BLOCK 12
003230                  294   INTEGER TEACI,TRACI,TEACJ,TRACJ;
003230                  295   LIGOU := FALSE;
003237                  296   MESTCIO,JOJ := -1;  MESTCJO,IOJ := -1;
003254  B37             297   IF LINHACIOJ = 0 THEN BEGIN
003254                  298                         TRACI := TRAFCIOJ;
003262                  299                         TEACI := 1;
003264        E37       300                         END
003265  B38             301                   ELSE BEGIN
003265                  302                         TRACI := TRACUMCLINHACIOJJ;
003277                  303                         TEACI := TEACUMCLINHACIOJJ;
003311        E38       304                         END;
003320  B39             305   IF LINHACJOJ = 0 THEN BEGIN
003320                  306                         TRACJ := TRAFCJOJ;
003326                  307                         TEACJ := 1;
003330        E39       308                         END
003331  B40             309                   ELSE BEGIN
003331                  310                         TRACJ := TRACUMCLINHACJOJJ;
003343                  311                         TEACJ := TEACUMCLINHACJOJJ;
003355        E40       312                         END;
003355                  313   IF (TRLIMAX >= (TRACI + TRACJ)) AND (TEMAX >= (TEACI + TEACJ))
003355                  314      AND (MCUSTCIO,JOJ <= CUSTLIM)
003403                  315      THEN LIGOU := TRUE;
```

```
003404                  316    IF LIGOU
003406  B41             317    THEN BEGIN
START OF BLOCK 13
003407                  318         INTEGER TO,P,Q;
003416                  319         IF LINHAC JO] = 0 THEN GERELIN(JO)
003422                  320                           ELSE L := LINHAC JO];
003430                  321         TRACUMC L] := TRACUMC L] + TRACI;
003444                  322         TEACUMC L] := TEACUMC L] + TEACI;
003460                  323         IF LINHAC IO] = 0
003467                  324         THEN CUSTACUMC L] := CUSTACUMC L] + MCUSTC IO,JO]
003510                  325         ELSE CUSTACUMC L] := CUSTACUMC L]+CUSTACUMC LINHAC IO]]+MCUSTC IO,JO];
003541                  326         PEGUE(PROXT,DISPT,TO);
003546                  327         TC TO] := IO;
003554                  328         P := LSTC C L];
003562                  329         IF P = 0
003565                  330         THEN LSTC C L] := TO
003574  B42             331         ELSE BEGIN
003603                  332              WHILE PROXTC P] # LAMB DO P := PROXTC P];
003612                  333              PROXTC P] := TO;
003620        E42       334              END;
003620                  335         IF LINHAC IO] # 0
003627  B43             336         THEN BEGIN
003627                  337              Q := LINHAC IO];
003635                  338              P := LSTC C Q];
003646  B44             339              WHILE P # LAMB DO BEGIN
003646                  340                                 LINHAC TC P]] := L;
003666                  341                                 ATEST(JO,TC P]);
003672                  342                                 P := PROXTC P];
003700        E44       343                                 END;
003701                  344              PROXTC TO] := LSTC C Q];
003713                  345              DEVOLVELIN(Q);
003716        E43       346              END;
003716                  347         LINHAC IO] := L;
003724                  348         EMITC C IO] := JO;
003732        E41       349         END;
END BLOCK 13, CONT 12
003735        E36       350    END;
END BLOCK 12, CONT 4
```

```
003735                 351
003735                 352 !------------------------------------------------------------------------------;
003735                 353 !                    PROGRAMA  PRINCIPAL                                       ;
003735                 354 !------------------------------------------------------------------------------;
003735                 355 ! INICIALIZACOES;
003741                 356 CALCUSTO;
003743                 357 INICIESTA;
003745                 358 INICIE(PROXT,NT,DISPT);
003754                 359 FOR I:=1 STEP 1 UNTIL NT DO
003760  B45            360     BEGIN
003760                 361     VAUX[I] := I;
003770                 362     FOR J:=1 STEP 1 UNTIL NT DO
003774                 363         SOMA[I] := SOMA[I] + MCUST[I,J];
004015       E45       364     END;
004024                 365 FOR I:=1 STEP 1 UNTIL NT DO CUSTMED := CUSTMED + SOMA[I];
004034                 366 CUSTMED := CUSTMED/(NT * NT - NT);
004044                 367 CUSTLIM := CUSTMED + CUSTMED/FATOR;
004052                 368 IMPRE1;
004052                 369 !------------------------------------------------------------------------------;
004052                 370 !  CONSTRUCAO DAS LINHAS MULTIPONTO INICIAIS                                    ;
004052                 371 !------------------------------------------------------------------------------;
004054                 372 ORDENE(VAUX,SOMA,NT);
004063                 373 FOR I:=1 STEP 1 UNTIL NT DO
004067  B46            374     BEGIN
004067                 375     IO := VAUX[I];
004104  B47            376     IF SINT[IO] # 0 THEN BEGIN
004104                 377                          JO := SINT[IO];
004112                 378                          LIGACAO;
004114       E47       379                          END;
004114       E46       380     END;
```

```
004117                    381 FOR I:=NT STEP -1 UNTIL 1 DO
004123  B48               382     BEGIN
004123                    383     IO := VAUX[I];
004123                    384  L1:
004141                    385     MINIC := INFINITO;
004145                    386     FOR J:=1 STEP 1 UNTIL NT DO
004151                    387     IF MEST[IO,VAUX[J]] = 0
004151                    388     THEN
004165  B49               389     BEGIN
004165                    390     IF MCUST[IO,VAUX[J]] < MINIC
004201  B50               391     THEN BEGIN
004201                    392          JO := VAUX[J];
004207                    393          MINIC := MCUST[IO,JO];
004216        E50         394          END
004217                    395     ELSE IF MCUST[IO,VAUX[J]] = MINIC
004233  B51               396          THEN BEGIN
004233                    397               DIST1 := DISTANCIA(IO,JO);
004246                    398               DIST2 := DISTANCIA(IO,VAUX[J]);
004256  B52               399               IF DIST2 < DIST1 THEN BEGIN
004256                    400                                     JO := VAUX[J];
004264                    401                                     MINIC := MCUST[IO,JO];
004273        E52         402                                     END;
004273        E51         403               END;
004273        E49         404     END;
004277                    405     IF MINIC # INFINITO THEN LIGACAO;
004310                    406     IF (MINIC # INFINITO) AND (NOT LIGOU) THEN GO TO L1;
004327                    407     IF (MINIC = INFINITO) AND (LINHA[IO] = 0) THEN GERELIN(IO);
004332                    408     IF (MINIC # INFINITO) AND LIGOU
004340  B53               409     THEN BEGIN
004340                    410          IF (TEACUM[LINHA[JO]] = TEMAX)
004373                    411          THEN FOR J:=1 STEP 1 UNTIL NT DO ATEST(TFIM[LINHA[JO]],J);
004400        E53         412          END;
004400        E48         413     END;
```

```
004400                414 !-------------------------------------------------------------------------------;
004400                415 !  CONSTRUCAO DO VETOR DOS FINS DE LINHA                                        !
004400                416 !-------------------------------------------------------------------------------;
004401                417 NF := 0;
004404                418 FOR I:=1 STEP 1 UNTIL NL DO
004417                419     IF TFIM[I] # 0 THEN NF := NF + 1;
004423  B54           420 BEGIN
START OF BLOCK 14
004426                421 REAL ARRAY SOMAF[1:NF];
004435                422 INTEGER ARRAY VFIM[1:NF];
004442                423 INTEGER P;
004442                424 P := 0;
004445                425 FOR I:=1 STEP 1 UNTIL NL DO
004475  B55  E55      426     IF TFIM[I] # 0 THEN BEGIN P := P+1; VFIM[P] := TFIM[I]; END;
004475                427 !-------------------------------------------------------------------------------;
004475                428 !  ESTIMATIVA DO NUMERO DE CONCENTRADORES                                       !
004475                429 !-------------------------------------------------------------------------------;
004502                430 IF NF REM LIMAX = 0 THEN AUX1 := NF/LIMAX
004513                431                     ELSE AUX1 := ENTIER(NF/LIMAX + 1);
004531                432 FOR I:=1 STEP 1 UNTIL NF DO
004535                433     AUX2 := AUX2 + TRACUM[LINHA[VFIM[I]]];
004561                434 IF AUX2 REM TRCONMAX = 0 THEN AUX2 := AUX2/TRCONMAX
004572                435                          ELSE AUX2 := ENTIER(AUX2/TRCONMAX+1);
004614                436 IF AUX1 > AUX2 THEN NC := AUX1  ELSE NC := AUX2;
004614                437 !-------------------------------------------------------------------------------;
004614                438 !  CALCULO DO SOMATORIO DOS CUSTOS DOS FIM DE LINHA                             !
004614                439 !-------------------------------------------------------------------------------;
004620                440 FOR I:=1 STEP 1 UNTIL NF DO
004626                441     FOR J:=1 STEP 1 UNTIL NF DO
004632                442         SOMAF[I] := SOMAF[I] + MCUST[VFIM[I],VFIM[J]];
004632                443 !-------------------------------------------------------------------------------;
004632                444 !  SELECAO DOS CONCENTRADORES                                                   !
004632                445 !-------------------------------------------------------------------------------;
004664                446 ORDENE(VFIM,SOMAF,NF);
004671  B56           447 BEGIN
START OF BLOCK 15
004721                448 INTEGER ARRAY TCN[1:NC],TRCONACUM[1:NC],LIACUM[1:NC],LISL[1:NC],
004730                449                 TIPO[1:NC];
004737                450 REAL ARRAY CUSTCONACUM[1:NC];
004744                451 INTEGER DEGRAU;
004744                452 !-------------------------------------------------------------------------------;
```

```
004751                  453    PROCEDURE RETIRE(CX,LX,PX);
004755                  454    INTEGER CX,LX,PX;
004760  B57             455    BEGIN
START OF BLOCK 16
004761                  456    INTEGER PX1;
004761                  457    TRCONACUM[CX] := TRCONACUM[CX] - TRACUM[LX];
005006                  458    LIACUM[CX] := LIACUM[CX] - 1;
005025                  459    CUSTCONACUM[CX] := CUSTCONACUM[CX] -
005025                  460                       (CUSTACUM[LX]+MCUST[TFIM[LX],TCN[CX]]);
005074                  461    PX := LISL[CX];
005107                  462    PX1 := PX;
005143  B58  E58        463    WHILE LX # LI[PX] DO BEGIN PX1 := PX; PX := PROXL[PX]; END;
005147                  464    IF PX1 = PX THEN LISL[CX] := PROXL[LX]
005166                  465                ELSE PROXL[PX1] := PROXL[LX];
005203                  466    CON[LX] := 0;
005211       E57        467    END;
END BLOCK 16, CONT 15
005211                  468 !-------------------------------------------------------------------------------------------------------------------;
```

```
005215                469    PROCEDURE IMPRE3;
005225  B59           470    BEGIN
START OF BLOCK 17
005226                471    INTEGER I;
005226                472    WRITE("[2C9S]ESTRUTURAS DOS CONCENTRADORES[2C6S]");
005231                473    WRITE("    TCN    TR-CON-ACUM    TR-LIBER  LI-ACUM  CUST-CON-ACUM    TIPO
SL");
005236                474    FOR I:=1 STEP 1 UNTIL NC DO
005242  B60           475        BEGIN
005242                476        TRLIBER := TRCONACUM[I] * VPCENT[CONTR] / 100;
005277                477        WRITE("[C]"); PRINT(I,4,0); PRINT(TCN[I],5,0);
005332                478        PRINT(TRCONACUM[I],11,0); PRINT(TRLIBER,11,0); PRINT(LIACUM[I],7,0);
005360                479        PRINT(CUSTCONACUM[I],12,2); PRINT(TIPO[I],7,0);
005373                480        PRINT(LISL[I],6,0);
005400        E60     481        END;
005403                482    TOTCUST := 0; TOTCUSTLIN := 0; TOTCUSTCON := 0;
005406                483    FOR I:=1 STEP 1 UNTIL NC DO
005412                484        IF TCN[I] # 0
005412                485        THEN
005421  B61           486        BEGIN
005421                487        TOTCUSTLIN := TOTCUSTLIN + CUSTCONACUM[I];
005430                488        TOTCUSTCON := TOTCUSTCON + VCUSTCON1[TIPO[I]] +VCUSTCON2[TIPO[I]];
005455        E61     489        END;
005456                490    TOTCUST := TOTCUSTLIN + TOTCUSTCON;
005464                491    WRITE("[2C9S]CUSTO DE LINHAS[11S]"); PRINT(TOTCUSTLIN,10,2);
005474                492    WRITE("[2C9S]CUSTO DE CONCENTRADORES[3S]"); PRINT(TOTCUSTCON,10,2);
005504                493    WRITE("[2C9S]CUSTO TOTAL[13S]"); PRINT(TOTCUST,12,2);
005511        E59     494    END;
END BLOCK 17, CONT 15
005511                495 !-------------------------------------------------------------------------------;
```

```
005525                 496 DEGRAU := NF/NC;   P := 2;
005531                 497 FOR I:= 1 STEP DEGRAU UNTIL (NF-(DEGRAU+1)) DO
005544  B62            498     BEGIN
005544                 499     AUX1 := VFIMCI + DEGRAU];
005557                 500     FOR J := (I+DEGRAU) STEP -1 UNTIL (P+1) DO
005565                 501         VFIMCJ] := VFIMCJ-1];
005602                 502     VFIMCP] := AUX1;
005610                 503     P := P + 1;
005613         E62     504     END;
005614                 505 INICIE(PROXL,NL,DISPL);
005623                 506 FOR I:=1 STEP 1 UNTIL NC DO
005627  B63            507     BEGIN
005641                 508     TCNCI] := VFIMCI]; TRCONACUMCI] := TRACUMCLINHACVFIMCI]]];
005671                 509     LIACUMCI] := 1;    CUSTCONACUMCI] := CUSTACUMCLINHACVFIMCI]]];
005721                 510     TIPOCI] := CONTR;  CONCLINHACVFIMCI]] := I;
005737                 511     PEGUE(PROXL,DISPL,LO);
005762                 512     LICLO] := LINHACVFIMCI]];  LISLCI] := LO;
005770         E63     513     END;
005770                 514 !-------------------------------------------------------------------------------;
005770                 515 !  LIGACAO DAS LINHAS AOS CONCENTRADORES                                        !
005770                 516 !-------------------------------------------------------------------------------;
005774                 517 FOR I:=(NC+1) STEP 1 UNTIL NF DO
006000  B64            518     BEGIN
006006                 519     IO := VFIMCI];  MINIC := INFINITO;
006012                 520     FOR J:=1 STEP 1 UNTIL NC DO
006016                 521         IF (MESTCIO,TCNCJ]] # 1) AND (LIACUMCJ] < LIMAX)
006016                 522         AND  ((TRCONACUMCJ] + TRACUMCLINHACIO]]) <= TRCONMAX)
006070  B65            523         THEN BEGIN
006070                 524              IF MCUSTCIO,TCNCJ]] < MINIC
006104  B66            525                THEN BEGIN
006104                 526                     JO := J;
006106                 527                     MINIC := MCUSTCIO,TCNCJ]];
006121         E66     528                     END
006122                 529                ELSE IF MCUSTCIO,TCNCJ]] = MINIC
006136  B67            530                     THEN BEGIN
006136                 531                          DIST1 := DISTANCIA(IO,JO);
006151                 532                          DIST2 := DISTANCIA(IO,TCNCJ]);
006161  B68            533                          IF DIST2 < DIST1 THEN BEGIN
006161                 534                                                JO := J;
006163                 535                                                MINIC := MCUSTCIO,TCNCJ]];
006176         E68     536                                                END;
006176         E67     537                          END;
006176         E65     538              END;
```

```
006177                  539      IF MINIC = INFINITO
006202  B69             540      THEN BEGIN
006202                  541           WRITE("C2C9SJ******** LINHA SEM LIGAR ******** MENS 3")§
006202                  542           ! STOP§
006205        E69       543           END
006206  B70             544      ELSE BEGIN
006206                  545           CONCLINHACIOJJ := JO§
006220                  546           P := LISLCJOJ§
006235                  547           WHILE PROXLCPJ # LAMB DO P := PROXLCPJ§
006244                  548           PEGUE(PROXL,DISPL,LO)§
006271                  549           LICLOJ := LINHACIOJ§   PROXLCPJ := LO§  EMITECIOJ := TCNCJOJ§
006303                  550           TRCONACUMCJOJ := TRCONACUMCJOJ + TRACUMCLINHACIOJJ§
006326                  551           LIACUMCJOJ := LIACUMCJOJ + 1§
006342                  552           CUSTCONACUMCJOJ := CUSTCONACUMCJOJ + CUSTACUMCLINHACIOJJ +
006342                  553                              MCUSTCIO,TCNCJOJJ§
006377        E70       554           END§
006377        E64       555      END§
006377                  556 !-------------------------------------------------------------------------------§
006377                  557 !  PARTICIONAMENTO DA REDE E DESENHO FINAL DAS LINHAS MULTIPONTO                 §
006377                  558 !-------------------------------------------------------------------------------§
006400                  559 INICIESTA§
006404                  560 FOR I:=1 STEP 1 UNTIL NC DO
006410  B71             561      BEGIN
START OF BLOCK 18
006411                  562      INTEGER NP§
006411                  563      NP := 0§
006412                  564      P := LISLCIJ§
006420                  565      WHILE PROXLCPJ # LAMB DO
006427  B72             566           BEGIN
006427                  567           NP := NP + TEACUMCLICPJJ§
006442                  568           P := PROXLCPJ§
006450        E72       569           END§
006451                  570      NP := NP + TEACUMCLICPJJ§
```

```
006451                  571 !-------------------------------------------------------------------------------;
006451                  572 !  DIMENSIONAMENTO DA PARTICAO E CONSTRUCAO DA MESMA                            ;
006451    //            573 !-------------------------------------------------------------------------------;
006464  B73             574     BEGIN
START OF BLOCK 19
006467                  575     INTEGER ARRAY PART[1:NP];
006476                  576     REAL ARRAY SOMAP[1:NP];
006503                  577     INTEGER P1,P2,PP,PX,P1X,J1,CENTRO;
006505                  578     BOOLEAN ARRAY LIGADO[1:NL];
006512                  579     PP := 0;
006513                  580     P := LISL[I];
006521                  581     WHILE P # LAMB DO
006524  B74             582           BEGIN
006534                  583           P1 := LI[P];  PX := P;  PP := PP + 1;
006551                  584           PART[PP] := TFIM[P1];  LINHACTFIM[P1]] := 0;
006573                  585           EMITEC[TFIM[P1]] := 0;  P1 := LSTE[P1];
006601                  586           WHILE P1 # LAMB DO
006604  B75             587                 BEGIN
006611                  588                 PP := PP + 1;  P1X := P1;  PART[PP] := TE[P1];
006645                  589                 LINHACT[P1]] := 0;  EMITEC[TE[P1]] := 0;  P1 := PROXTE[P1];
006660                  590                 TE[P1X] := 0;  DEVOLVA(PROXT,DISPT,P1X);
006665        E75       591                 END;
006674                  592           DEVOLVELIN(LI[P]);
006705                  593           P := PROXL[P];  LI[PX] := 0;
006712                  594           DEVOLVA(PROXL,DISPL,PX);
006717        E74       595           END;
006733                  596     CENTRO := TC[I];  LISL[I] := 0;  LIACUM[I] := 0;
006745                  597     TRCONACUM[I] := 0;  CUSTCONACUM[I] := 0;
006745                  598 !-------------------------------------------------------------------------------;
006745                  599 !  CONSTRUCAO DAS LINHAS MULTIPONTO PARA AS PARTICOES                           ;
006745                  600 !-------------------------------------------------------------------------------;
006754                  601     FOR PP:=1 STEP 1 UNTIL NP DO
006760                  602           SOMAP[PP] := MCUST[PART[PP],CENTRO];
007000                  603     ORDENE(PART,SOMAP,NP);
007007                  604     FOR PP:=1 STEP 1 UNTIL NP DO
007013  B76             605         BEGIN
007013                  606         IO := PART[PP];
007030  B77             607         IF SINT[IO] # 0 THEN BEGIN
007030                  608                              JO := SINT[IO];
007036                  609                              LIGACAO;
007040        E77       610                              END;
007040                  611
007040        E76       612         END;
```

```
007043                  613      FOR PP:=NP STEP -1 UNTIL 1 DO
007047  B78             614          BEGIN
007047                  615          IO := PART[PP];
007047                  616       L2:
007065                  617          MINIC := INFINITO;
007071                  618          FOR J:=1 STEP 1 UNTIL NP DO
007075  B79             619              BEGIN
007075                  620              J1 := PART[J];
007103                  621              IF (MESTI[IO,J1] = 0) AND (MCUSTI[IO,J1] <= MCUSTI[IO,CENTRO])
007134  B80             622              THEN BEGIN
007134                  623                   IF MCUSTI[IO,J1] < MINIC
007155  B81  E81        624                   THEN BEGIN JO := J1; MINIC := MCUSTI[IO,JO]; END
007156                  625                   ELSE IF MCUSTI[IO,J1] = MINIC
007166  B82             626                        THEN BEGIN
007166                  627                             DIST1 := DISTANCIA(IO,JO);
007173                  628                             DIST2 := DISTANCIA(IO,J1);
007200                  629                             IF DIST2 < DIST1
007214  B83  E83        630                             THEN BEGIN JO:=J1; MINIC:=MCUSTI[IO,JO]; END;
007214       E82        631                             END;
007214       E80        632                   END;
007214       E79        633              END;
007220                  634          IF MINIC # INFINITO THEN LIGACAO;
007231                  635          IF (MINIC # INFINITO) AND (NOT LIGOU) THEN GO TO L2;
007250                  636          IF (MINIC = INFINITO) AND (LINHA[IO] = 0) THEN GERELIN(IO);
007253                  637          IF (MINIC # INFINITO) AND LIGOU
007261  B84             638          THEN BEGIN
007261                  639               IF TEACUM[LINHA[JO]] = TEMAX
007276                  640               THEN FOR J:=1 STEP 1 UNTIL NP DO
007302  B85             641                         BEGIN
007302                  642                         J1 := PART[J];
007322                  643                         ATEST(TFIM[LINHA[JO]],J1);
007326       E85        644                         END;
007327       E84        645               END;
007327       E78        646          END;
```

```
007327                 647 !-------------------------------------------------------------------------------;
007327                 648 !  LIGACAO DOS FINS DE LINHA DA PARTICAO AO CENTRO                               ;
007327                 649 !-------------------------------------------------------------------------------;
007344                 650     L := LINHA[CENTRO];  CONC[L] := I;  LIACUM[I] := 1;
007364                 651     TRCONACUM[I] := TRACUM[L];  CUSTCONACUM[I] := CUSTACUM[L];
007376                 652     LIGADO[L] := TRUE;
007403                 653     PEGUE(PROXL,DISPL,LO);
007416                 654     LI[LO] := L;  LISL[I] := LO;
007426                 655     FOR PP:=1 STEP 1 UNTIL NP DO
007432  B86            656         BEGIN
007440                 657         J1 := PART[PP];  L := LINHAC[J1];
007446                 658         IF (EMITEC[J1] = 0) AND (NOT LIGADO[L])
007466  B87            659         THEN BEGIN
007466                 660              CONC[L] := I;
007474                 661              PEGUE(PROXL,DISPL,LO);
007507                 662              LI[LO] := L;  P := LISL[I];
007524                 663              WHILE PROXL[P] # LAMB DO P := PROXL[P];
007541                 664              PROXL[P] := LO;  LIACUM[I] := LIACUM[I] + 1;
007555                 665              TRCONACUM[I] := TRCONACUM[I] + TRACUM[L];
007574                 666              CUSTCONACUM[I] := CUSTCONACUM[I] + CUSTACUM[L] +
007574                 667                                MCUST[TFIM[L],CENTRO];
007637                 668              EMITEC[TFIM[L]] := CENTRO;  LIGADO[L] := TRUE;
007644        E87      669              END;
007644        E86      670         END;
007645        E73      671     END;
END BLOCK 19, CONT 18
007646        E71      672     END;
END BLOCK 18, CONT 15
007646                 673 !-------------------------------------------------------------------------------;
007646                 674 !  CONSTRUCAO DO VETOR DOS FINS DE LINHA ATUAIS                                 ;
007646                 675 !-------------------------------------------------------------------------------;
007652                 676 NF := 0;
007655                 677 FOR I:=1 STEP 1 UNTIL NL DO
007670                 678     IF TFIM[I] #0 THEN NF := NF + 1;
```

```
007674  B88             679 BEGIN
START OF BLOCK 20
007677                  680 INTEGER ARRAY VFIM1[1:NF];
007706                  681 REAL ARRAY SOMAF1[1:NF];
007713                  682 P := 0;
007716                  683 FOR I:=1 STEP 1 UNTIL NL DO
007722                  684     IF TFIM[I] # 0
007731  B89             685     THEN BEGIN
007731                  686          P := P + 1;
007734                  687          VFIM1[P] := TFIM[I];
007746                  688          SOMAF1[P] := MCUST[TFIM[I],TCN[CON[I]]];
007775       E89        689          END;
007776                  690 NF := P;
007776                  691 !-------------------------------------------------------------------------------;
007776                  692 !  OTIMIZACAO DAS LIGACOES DAS LINHAS AOS CONCENTRADORES                        ;
007776                  693 !-------------------------------------------------------------------------------;
010000                  694 ORDENE(VFIM1,SOMAF1,NF);
010007                  695 FOR I:=NF STEP -1 UNTIL 1 DO
010013  B90             696     BEGIN
START OF BLOCK 21
010014                  697     REAL CUST1,CUST2;
010014                  698     INTEGER CO,I1,PX;
010022                  699     CUST1 := SOMAF1[I];  CUST2 := INFINITO;
010032                  700     I1 := VFIM1[I];  L := LINHA[I1];
010042                  701     FOR J:=1 STEP 1 UNTIL NC DO
010046                  702         IF (LIACUM[J] < LIMAX)
010046                  703         AND (TRCONACUM[J] + TRACUM[L] <= TRCONMAX)
010075  B91             704         THEN BEGIN
010075                  705              IF MCUST[I1,TCN[J]] < CUST2
010126  B92  E92        706              THEN BEGIN CO := J;  CUST2 := MCUST[I1,TCN[J]];  END;
010126       E91        707              END;
010127                  708     IF CUST2 < CUST1
010132  B93             709     THEN BEGIN
010132                  710          TRCONACUM[CO] := TRCONACUM[CO] + TRACUM[L];
010151                  711          LIACUM[CO] := LIACUM[CO] + 1;
010165                  712          CUSTCONACUM[CO] := CUSTCONACUM[CO] + CUSTACUM[L] +
010165                  713                             MCUST[I1,TCN[CO]];
010216                  714          P := LISL[CO];
010233                  715          WHILE PROXL[P] # LAMB DO P := PROXL[P];
010250                  716          RETIRE(CON[L],L,PX);
010263                  717          PROXL[P] := PX;  PROXL[PX] := LAMB;
010277                  718          CON[L] := CO;  EMITE[I1] := TCN[CO];
010311       E93        719          END;
010311       E90        720     END;
END BLOCK 21, CONT 20
```

```
010311                721 !-----------------------------------------------------------------------;
010311                722 !  OTIMIZACAO DO NUMERO DE CONCENTRADORES                               ;
010311                723 !-----------------------------------------------------------------------;
010315  B94           724 BEGIN
START OF BLOCK 22
010316                725 INTEGER NC1,LIMITE,CONTR1,I1,P1,LI70,FOLGA;
010320                726 INTEGER ARRAY VAUX1[1:NC];
010327                727 REAL ARRAY VAUX2[1:NC];
010336                728 FOR I:=1 STEP 1 UNTIL NC DO
010342  B95           729     BEGIN
010350                730     VAUX1[I] := I;  VAUX2[I] := LIACUM[I];
010364        E95     731     END;
010365                732 ORDENE(VAUX1,VAUX2,NC);
010401                733 LI70 := LIMAX * .7;  FOLGA := 0;  ACABOU := FALSE;  NC1 := NC;
010403                734 I := 1;
010405                735 WHILE NOT ACABOU DO
010407  B96           736       BEGIN
010407                737       I1 := VAUX1[I];
010415                738       IF LIACUM[I1] < LI70
010424  B97           739       THEN BEGIN
010426                740            LIMITE := I;  FOLGA := FOLGA + (LIMAX - LIACUM[I1]);
010436        E97     741            END
010437                742       ELSE ACABOU := TRUE;
010443                743       IF NC = I THEN ACABOU := TRUE
010445                744                 ELSE I := I + 1;
010450        E96     745       END;
010452                746 ACABOU := FALSE;  I := 1;
```

```
010454                  747 WHILE NOT ACABOU DO
010456  B98             748       BEGIN
010456                  749       IF LIACUM[VAUX1[I]] > (FOLGA - (LIMAX - LIACUM[VAUX1[I]]))
010504                  750       THEN ACABOU := TRUE
010506  B99             751       ELSE BEGIN
START OF BLOCK 23
010507                  752            INTEGER NL1,PJOS;
010511                  753            INTEGER ARRAY JOS[1:LIMAX];
010516                  754            REAL CUSTAD;
010516                  755            BOOLEAN ACABE;
010532                  756            I1 := VAUX1[I];  P := LISL[I1];  NL1 := LIACUM[I1];
010540                  757            WHILE NOT ACABE DO
010542 B100             758                  BEGIN
010550                  759                  LO := LI[P];  MINIC := INFINITO;
010554                  760                  FOR J:=1 STEP 1 UNTIL NC DO
010560                  761                      IF (TCN[J] # TCN[I1]) AND
010560                  762                         (MCUST[TFIM[LO],TCN[J]] < MINIC) AND
010560                  763                         (LIACUM[J] < LIMAX) AND
010560                  764                         (TRCONACUM[J] + TRACUM[LO] < TRCONMAX) AND
010560                  765                         (MEST[TFIM[LO],TCN[J]] # 1)
010670 B101             766                      THEN BEGIN
010672                  767                           JO := J;  MINIC := MCUST[TFIM[LO],TCN[J]];
010711       E101       768                           END;
010712                  769                  IF MINIC = INFINITO
010715                  770                  THEN ACABE := TRUE
010717 B102             771                  ELSE BEGIN
010717                  772                       CUSTAD := CUSTAD + (MCUST[TFIM[LO],TCN[JO]] -
010717                  773                                           MCUST[TFIM[LO],TCN[I1]]);
010756                  774                       NL1 := NL1 - 1;
010761                  775                       PJOS := PJOS + 1;
010764                  776                       JOS[PJOS] := JO;
010775                  777                       IF NL1 = 0 THEN ACABE := TRUE;
010776                  778                       P := PROXL[P];
011004       E102       779                       END;
011004       E100       780                  END;
```

```
011004                781 !-------------------------------------------------------------------------------;
011004                782 !  VERIFICAR SE O CONCENTRADOR PODE SER ELIMINADO E SE A ELIMINACO      ;
011004                783 !  OCASIONARA REDUCAO NOS CUSTOS                                        ;
011004                784 !-------------------------------------------------------------------------------;
011005                785              IF (NL1 = 0) AND (CUSTAD < (CUSTCON1 + CUSTCON2))
011020 B103           786              THEN BEGIN
011020                787                   PJOS := 0;
011021                788                   P := LISLCI1];
011027                789                   WHILE P # LAMB DO
011032 B104           790                         BEGIN
START OF BLOCK 24
011033                791                         INTEGER P2;
011033                792                         PJOS := PJOS + 1;
011036                793                         JO := JOSCPJOS];
011044                794                         TRCONACUMCJO] := TRCONACUMCJO] + TRACUMCLICP]];
011067                795                         LIACUMCJO] := LIACUMCJO] + 1;
011103                796                         CUSTCONACUMCJO] := CUSTCONACUMCJO] +
011103                797                                            CUSTACUMCLICP]] +
011103                798                                            MCUSTCTFIMCLICP]],TCNCJO]];
011150                799                         CONCLICP]] := JO;
011162                800                         EMITECTFIMCLICP]]] := TCNCJO];
011204                801                         P1 := LISLCJO];
011221                802                         WHILE PROXLCP1] # 0 DO P1 := PROXLCP1];
011230                803                         P2 := PROXLCP];
011236                804                         PROXLCP1] := P;
011244                805                         PROXLCP] := LAMB;
011252                806                         P := P2;
011254      E104      807                         END;
END BLOCK 24, CONT 23
011270                808                   NC1 := NC1 - 1;  TCNCI1] := 0;  TRCONACUMCI1] := 0;
011307                809                   LIACUMCI1] := 0;  CUSTCONACUMCI1] := 0;  LISLCI1] := LAMB;
011315                810                   TIPOCI1] := 0;
011322      E103      811                   END;
011325                812              IF I = LIMITE THEN ACABOU := TRUE
011327                813                             ELSE I := I + 1;
011332       E99      814              END;
END BLOCK 23, CONT 22
011333       E98      815         END;
011334                816 IMPRE2;
011336                817 IMPRE3;
011336
```

```
                    818 !-------------------------------------------------------------------------------;
011336              819 !  OTIMIZACAO DOS TIPOS DOS CONCENTRADORES                                      ;
011336              820 !-------------------------------------------------------------------------------;
011340 B105         821 BEGIN
START OF BLOCK 25
011341              822 INTEGER CONTR1,I01,J01;
011341              823 CONTR1 := CONTR + 1;
011344              824 WHILE CONTR1 <= IT DO
011347 B106         825 BEGIN
START OF BLOCK 26
011352              826 INTEGER ARRAY VCONC1:NC1];
011357              827 P := 0;
011362              828 FOR I:=1 STEP 1 UNTIL NC DO
011406 B107 E107    829     IF TCNC[] # 0 THEN BEGIN P := P + 1; VCONC[P] := I; END;
011407 B108         830       BEGIN
START OF BLOCK 27
011410              831       INTEGER LIMAX2,TRCONMAX2,P1,L1,J1;
011410              832       REAL CUST12,CUST22;
011414              833       REAL ARRAY MDISTC1:NC1,1:NC1];
011423              834       FOR I:=1 STEP 1 UNTIL NC1 DO
011427 B109         835           BEGIN
011427              836           I1 := VCONC[I];
011437              837           FOR J:=1 STEP 1 UNTIL NC1 DO
011443 B110         838               BEGIN
011443              839               J1 := VCONC[J];
011464              840               MDISTC[I,J] := DISTANCIA(TCNC[I1],TCNC[J1]);
011501              841               MDISTC[J,I] := MDISTC[I,J];
011515      E110    842               END;
011516              843           MDISTC[I,I] := INFINITO;
011525      E109    844           END;
011526              845       ACABOU := FALSE;
011527              846       WHILE NOT ACABOU DO
011531 B111         847             BEGIN
011531              848             MINIC := INFINITO;
011535              849             FOR I:=1 STEP 1 UNTIL NC1 DO
011543              850                 FOR J:=1 STEP 1 UNTIL NC1 DO
011547              851                     IF MDISTC[I,J] < MINIC
011557 B112         852                     THEN BEGIN
011557              853                          MINIC := MDISTC[I,J];
011570              854                          IO := I;  JO := J;
011572      E112    855                          END;
```

```
011574                  856
011577                  857
011601  B113            858
011601                  859
011610                  860
011625                  861
011633                  862
011641                  863
011647                  864
011655                  865
011663                  866
011663                  867
011717  B114            868
START OF BLOCK 28
011720                  869
011720                  870
011720                  871
011737  B115  E115      872
011744  B116  E116      873
011744                  874
011752                  875
011755  B117            876
011755                  877
011755                  878
011755                  879
012024                  880
012032        E117      881
012033                  882
012033                  883
012033                  884
012033                  885
012033                  886
```

```
IF (MINIC = INFINITO)
THEN ACABOU := TRUE
ELSE BEGIN
     MDISTCIO,JO] := INFINITO;
     MDISTCJO,IO] := INFINITO;
     IO1 := VCONCIO];  JO1 := VCONCJO];
     LIMAX2 := VLIMAXCCONTR1];
     TRCONMAX2 := VTRCONMAXCCONTR1];
     CUST12 := VCUSTCON1CCONTR1];
     CUST22 := VCUSTCON2CCONTR1];
     IF (LIACUMCIO1] + LIACUMCJO1] <= LIMAX2) AND
        (TRCONACUMCIO1] + TRCONACUMCJO1] <= TRCONMAX2)
     THEN BEGIN

          INTEGER IC,IC1;
          REAL CUSTAD,CUSTAD1;
          IF LIACUMCIO1] >= LIACUMCJO1]
          THEN BEGIN  IC:=IO1;  IC1:=JO1;  END
          ELSE BEGIN  IC:=JO1;  IC1:=IO1;  END;
          P := LISLCIC1];
          WHILE P # LAMB DO
                BEGIN
                CUSTAD := CUSTAD +
                          (MCUSTCTFIMCLICP]],TCNCIC]] -
                           MCUSTCTFIMCLICP]],TCNCIC1]]);
                P := PROXLCP];
                END;
          CUSTAD1 := CUSTAD - ((VCUSTCON1CTIPOCIC]] +
                               VCUSTCON2CTIPOCIC]] +
                               VCUSTCON1CTIPOCIC1]] +
                               VCUSTCON2CTIPOCIC1]] -
                              (CUST12 +CUST22));
```

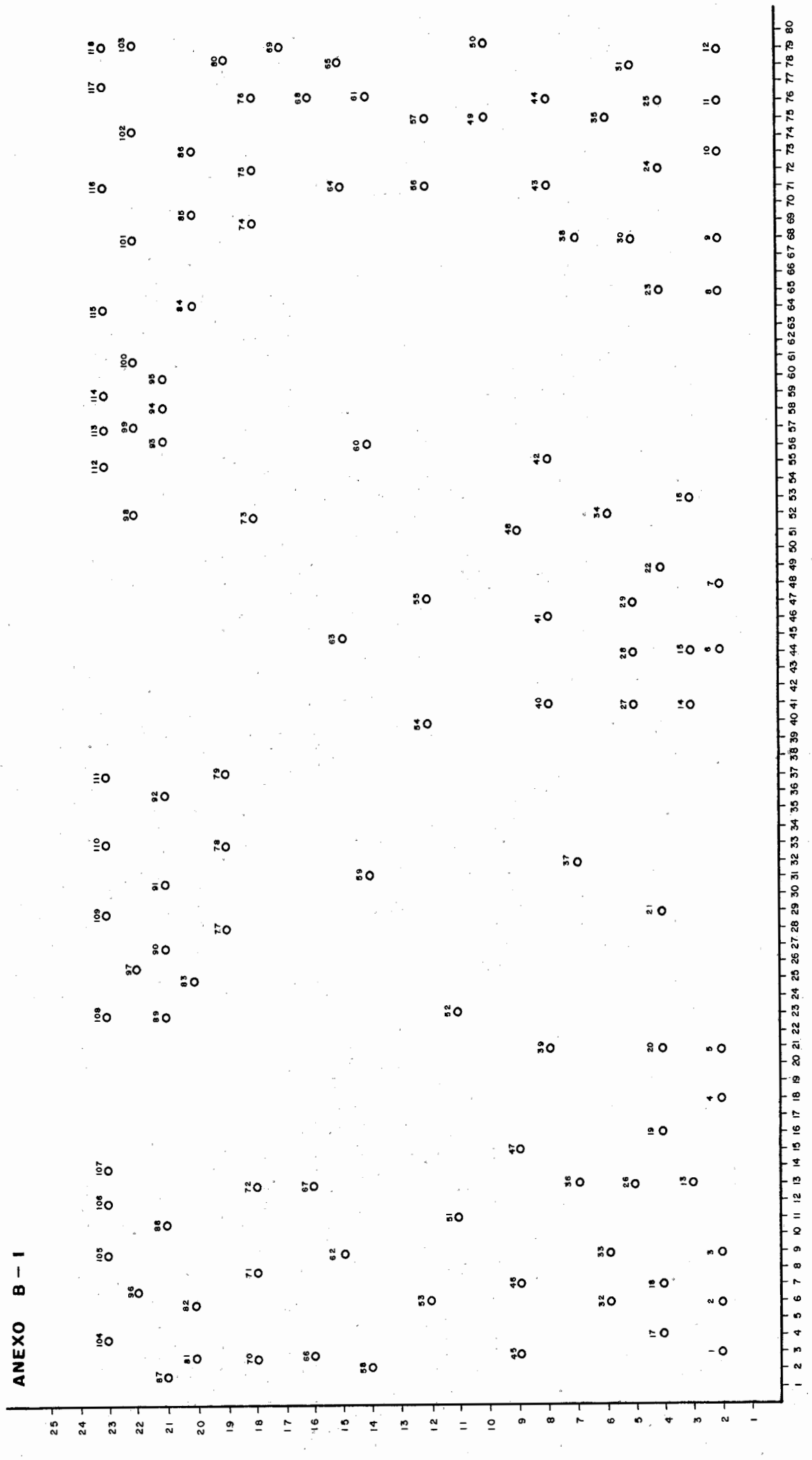
```
012105                  887
012111 B118             888
012111                  889
012126                  890
012135                  891
012143                  892
012151                  893
012154 B119             894
012154                  895
012162                  896
012170                  897
012170                  898
012225                  899
012243                  900
012251       E119       901
012254                  902
012260 B120             903
012260                  904
012263 B121             905
012263                  906
012272                  907
012301       E121       908
012302 B122             909
012302                  910
012311                  911
012320       E122       912
012320       E120       913
012321                  914
012340                  915
012365                  916
012375                  917
012410                  918
012422                  919
012427       E118       920
012427       E114       921
END BLOCK 28, CONT 27
012432       E113       922                    END;
012432       E111       923                END;
012433       E108       924            END;
END BLOCK 27, CONT 26
012434                  925 CONTR1 := CONTR1 + 1;
012437       E106       926 END;
END BLOCK 26, CONT 25
```

```
IF CUSTAD1 < 0
THEN BEGIN
     P := LISL[IC];
     WHILE PROXL[P] # LAMB DO P := PROXL[P];
     P1 := LISL[IC1];
     PROXL[P] := P1;
     WHILE P1 # LAMB DO
           BEGIN
           L1 := LIC[P1];
           CONL[L1] := IC;
           CUSTCONACUM[IC] := CUSTCONACUM[IC] +
             MCUSTCTF[IM[L1],TCN[IC]]+CUSTACUM[L1];
           EMITECTF[IM[L1]] := TCN[IC];
           P1 := PROXL[P1];
           END;
     FOR I:=1 STEP 1 UNTIL NC1 DO
         BEGIN
         IF IO1 = IC1
         THEN BEGIN
              MDIST[I,IO] := INFINITO;
              MDIST[IO,I] := INFINITO;
              END
         ELSE BEGIN
              MDIST[I,JO] := INFINITO;
              MDIST[JO,I] := INFINITO;
              END;
         END;
     TRCONACUM[IC]:=TRCONACUM[IC]+TRCONACUM[IC1];
     LIACUM[IC] := LIACUM[IC] + LIACUM[IC1];
     TIPO[IC] := CONTR1;  NC1 := NC1 - 1;
     TCN[IC1] := 0;  LISL[IC1] := LAMB;
     TRCONACUM[IC1] := 0;  LIACUM[IC1] := 0;
     TIPO[IC1] := 0;  CUSTCONACUM[IC1] := 0;
     END;
END;
```

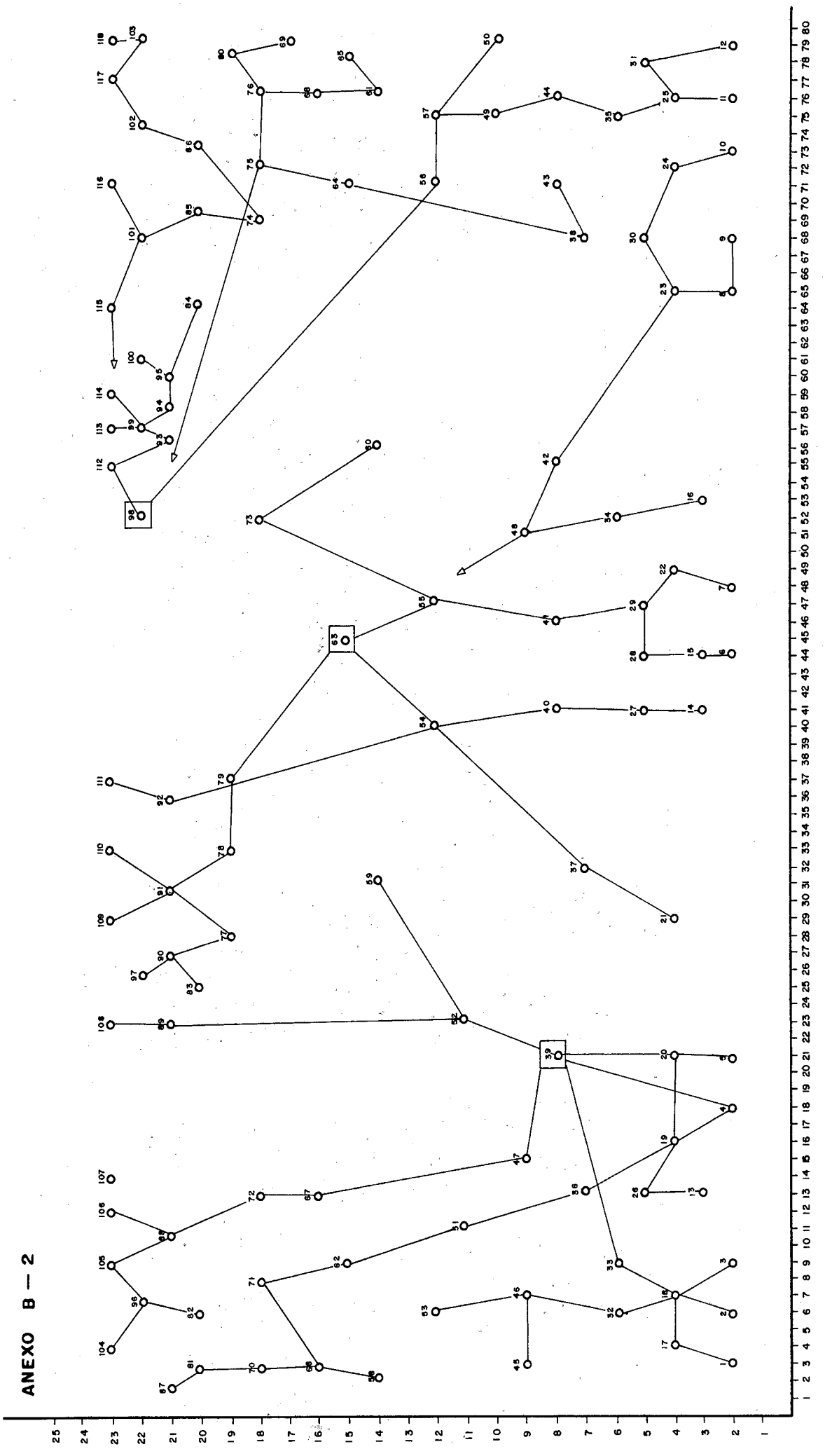
```
012441                  927 IMPRE2;
012443                  928 IMPRE3;
012445       E105       929 END;
END BLOCK 25, CONT 22
012450         E94      930 END;
END BLOCK 22, CONT 20
012451         E88      931 END;
END BLOCK 20, CONT 15
012452         E56      932 END;
END BLOCK 15, CONT 14
012453         E54      933 END;
END BLOCK 14, CONT 4
012454          E4      934 END;
END BLOCK 4, CONT 3
012455          E3      935 END;
END BLOCK 3, CONT 2
012457          E2      936 END;
END BLOCK 2, CONT 1
012460          E1      937 END;
END BLOCK 1, CONT 0

No errors
```

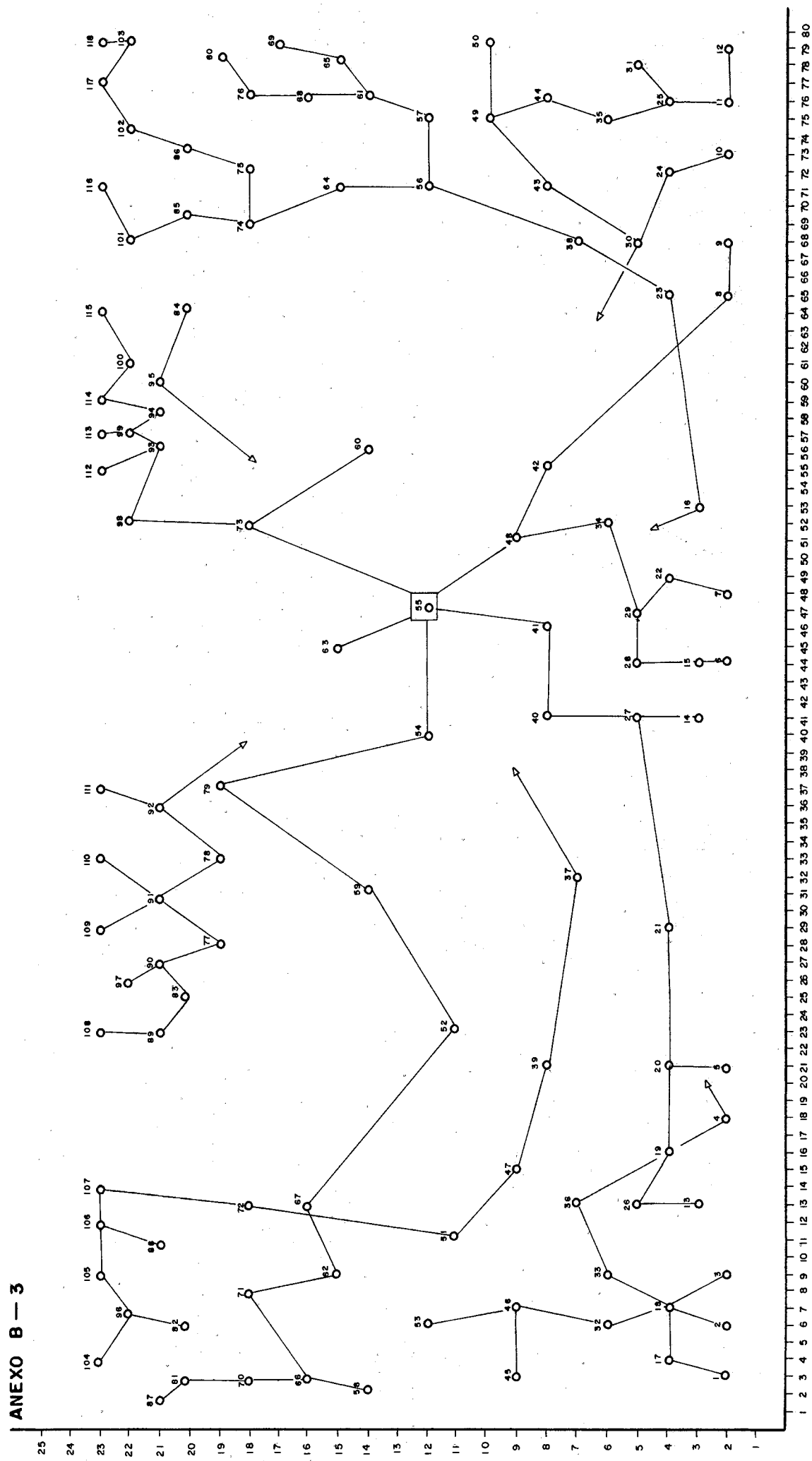
ANEXO B

CONFIGURAÇÃO DAS REDES GERADAS PELO ALGORÍTMO WRP

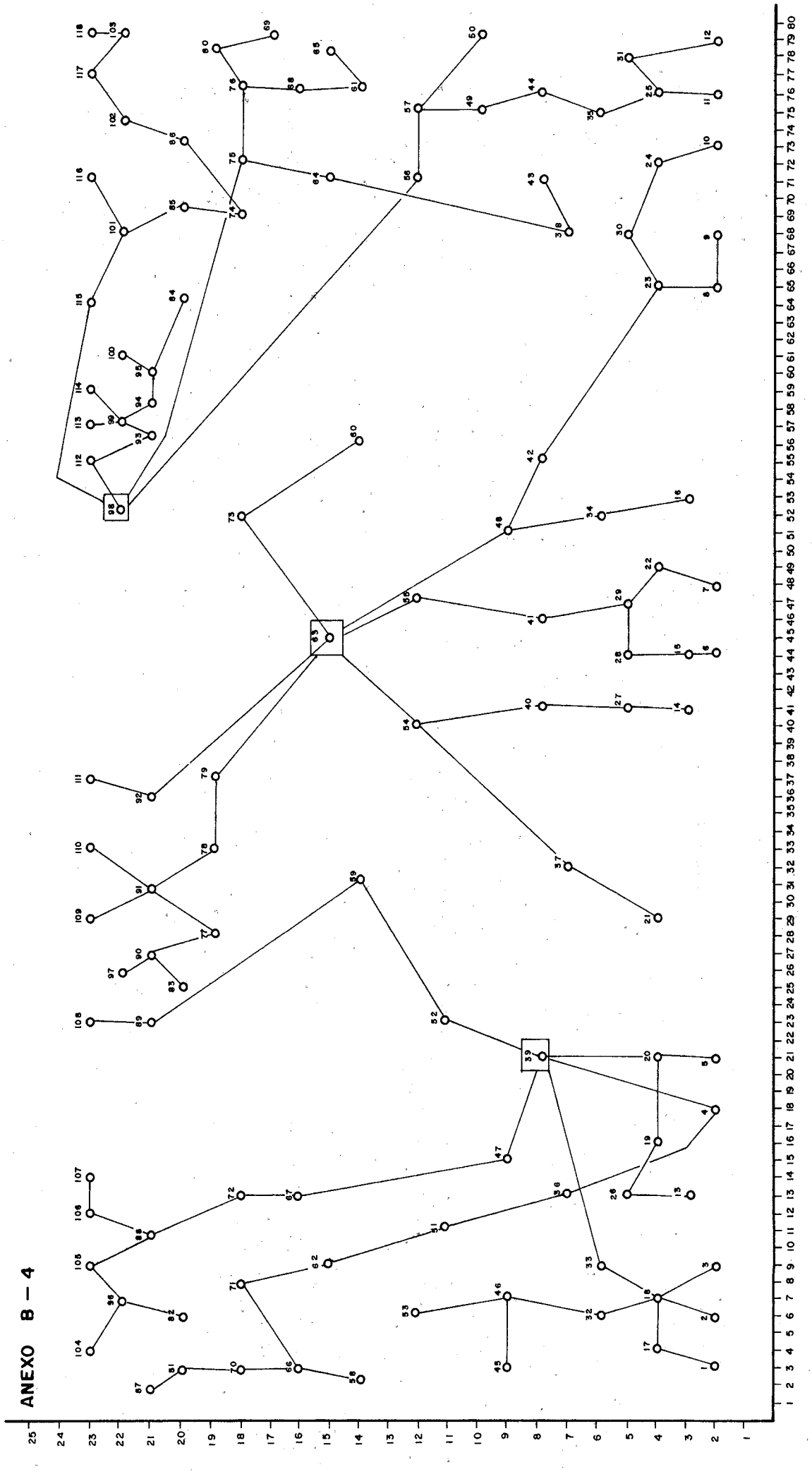


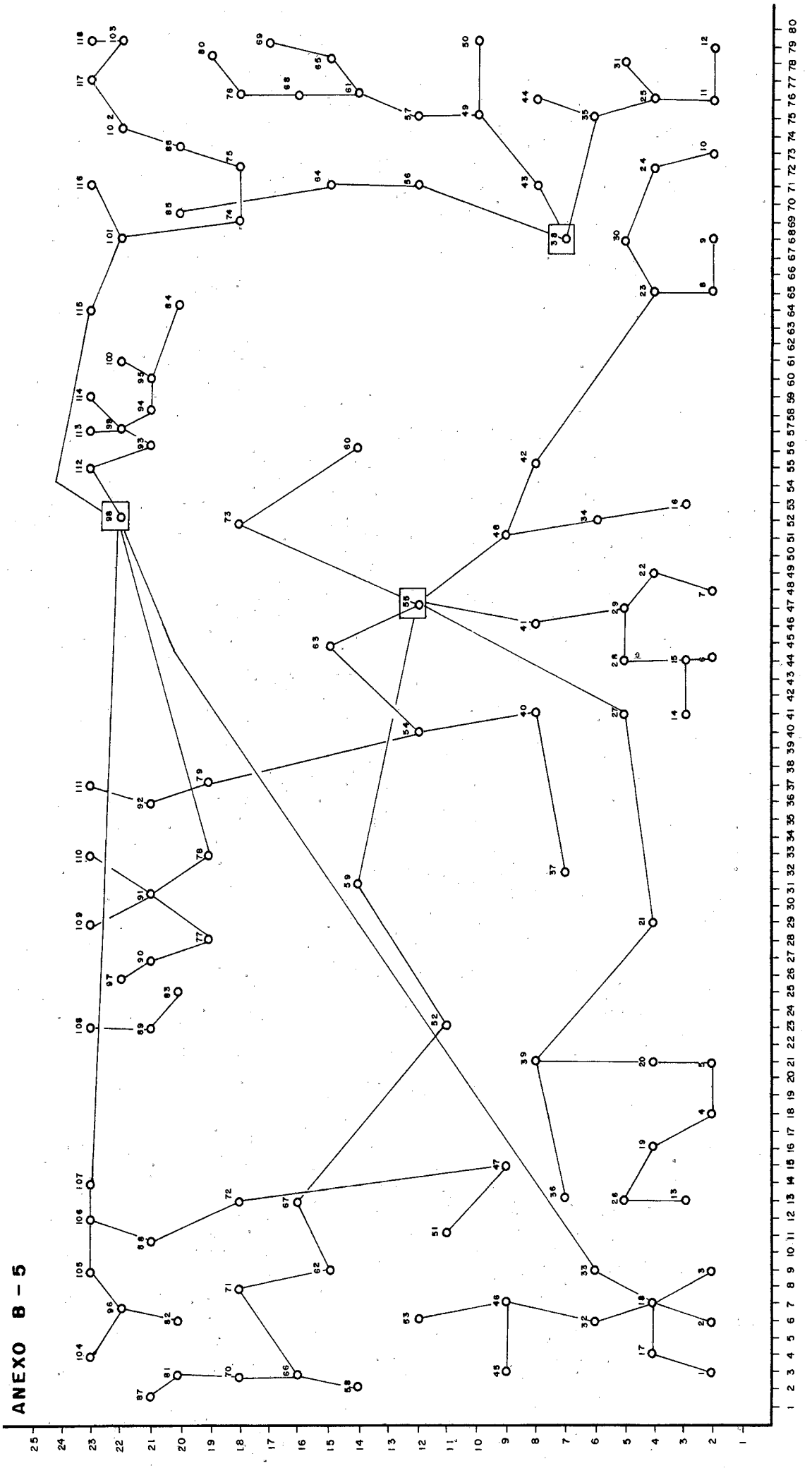
ANEXO B – I


ANEXO B — 2

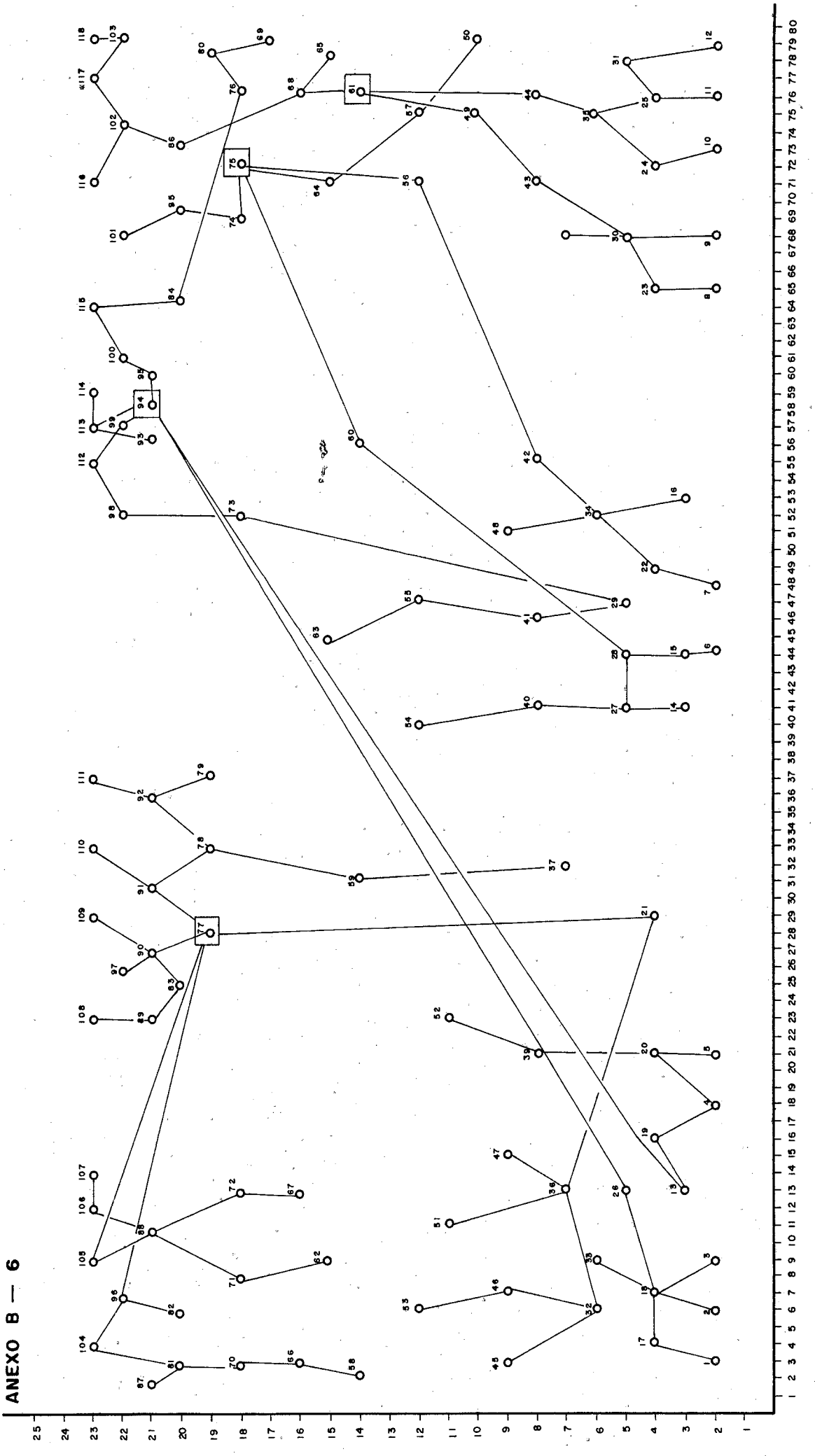


ANEXO B — 3

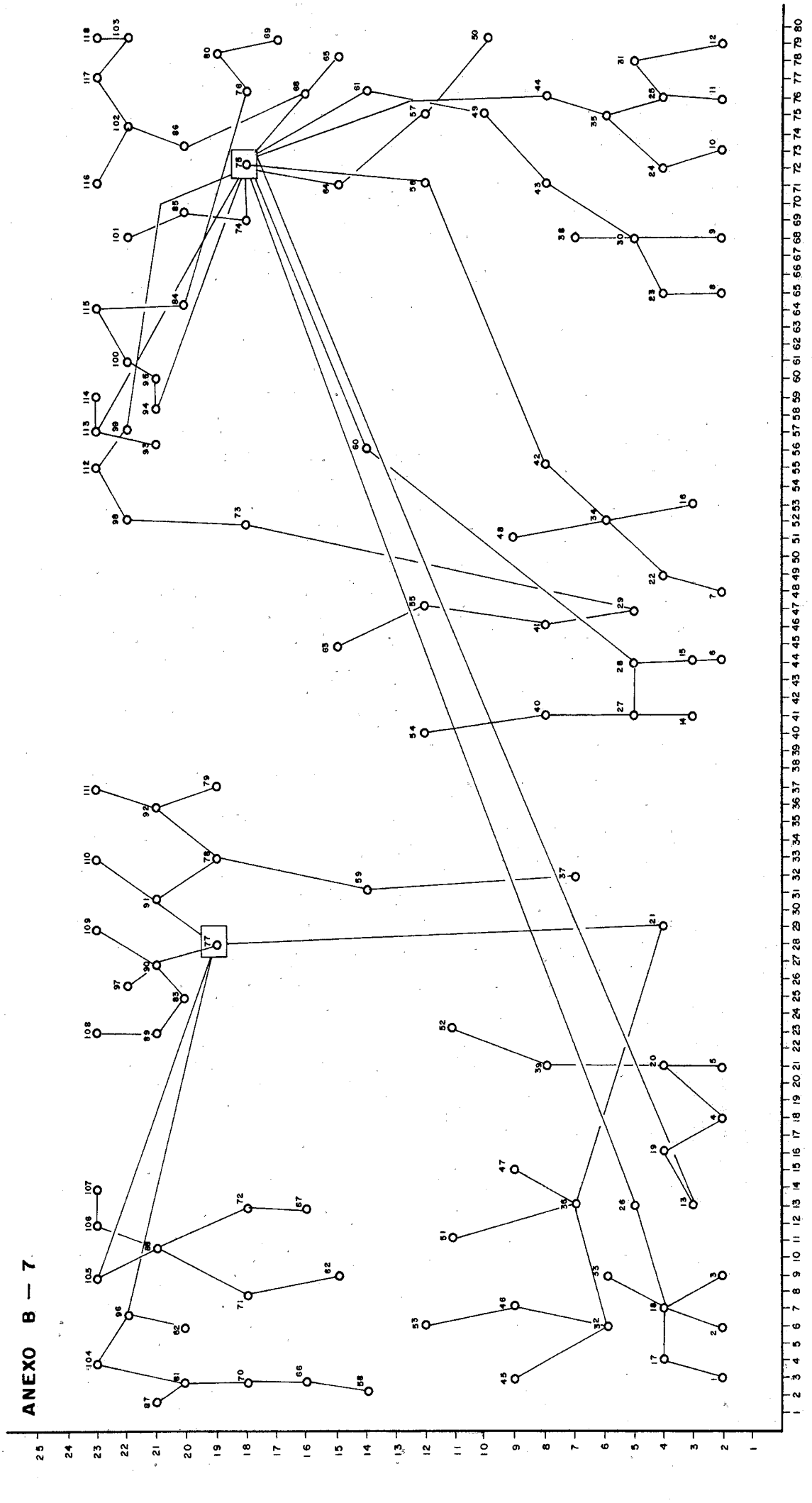

ANEXO B — 4

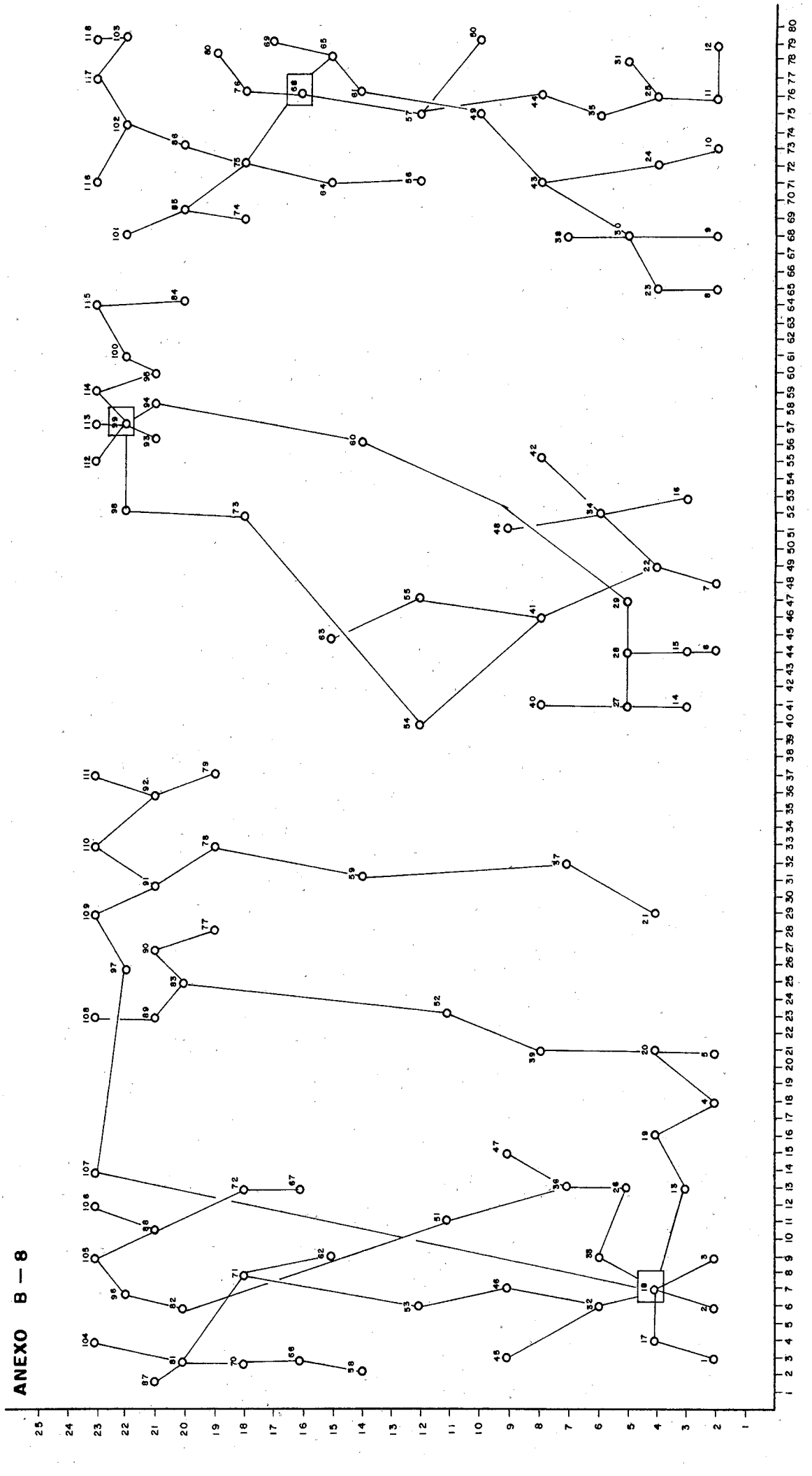


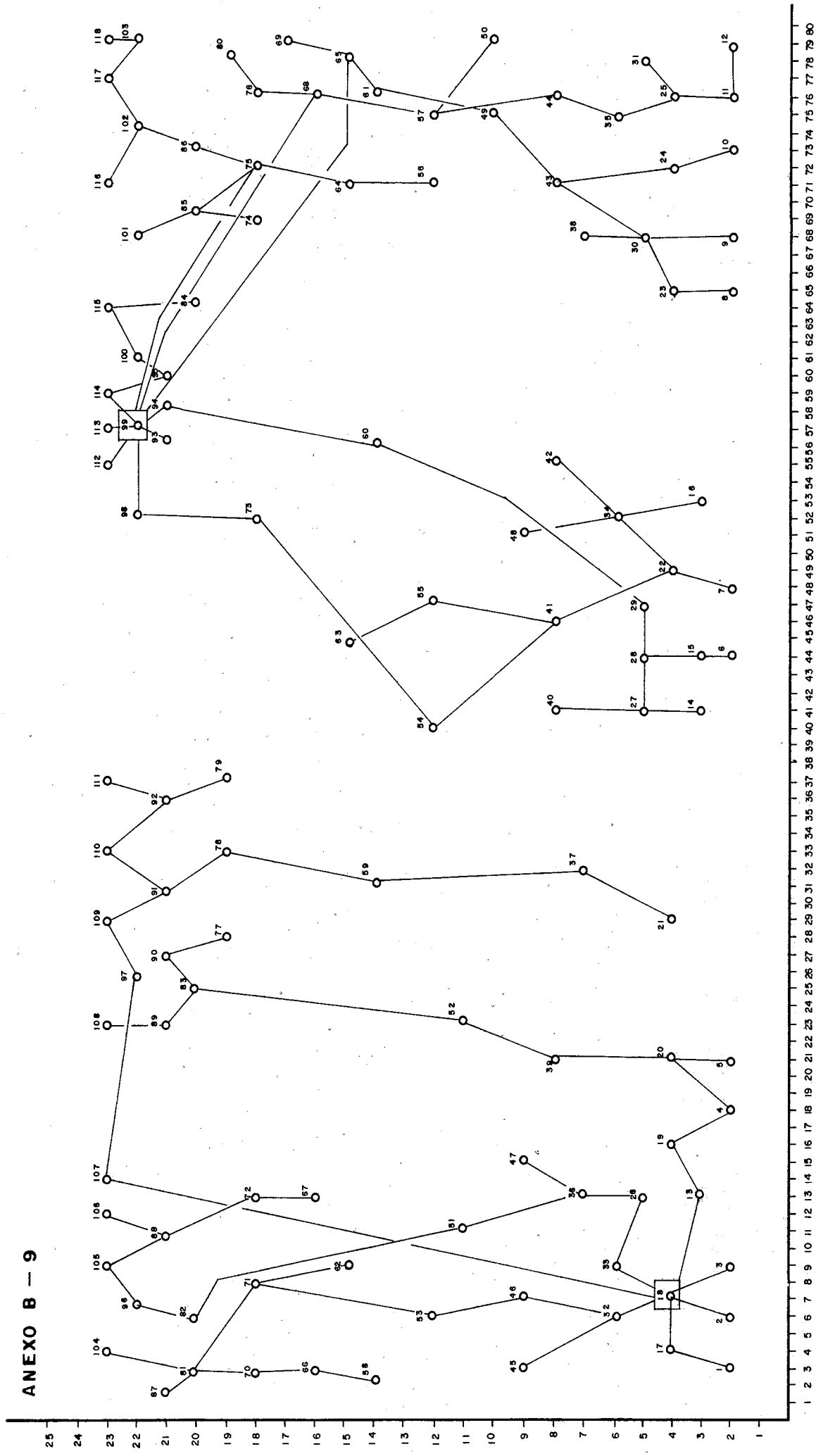
ANEXO B – 5



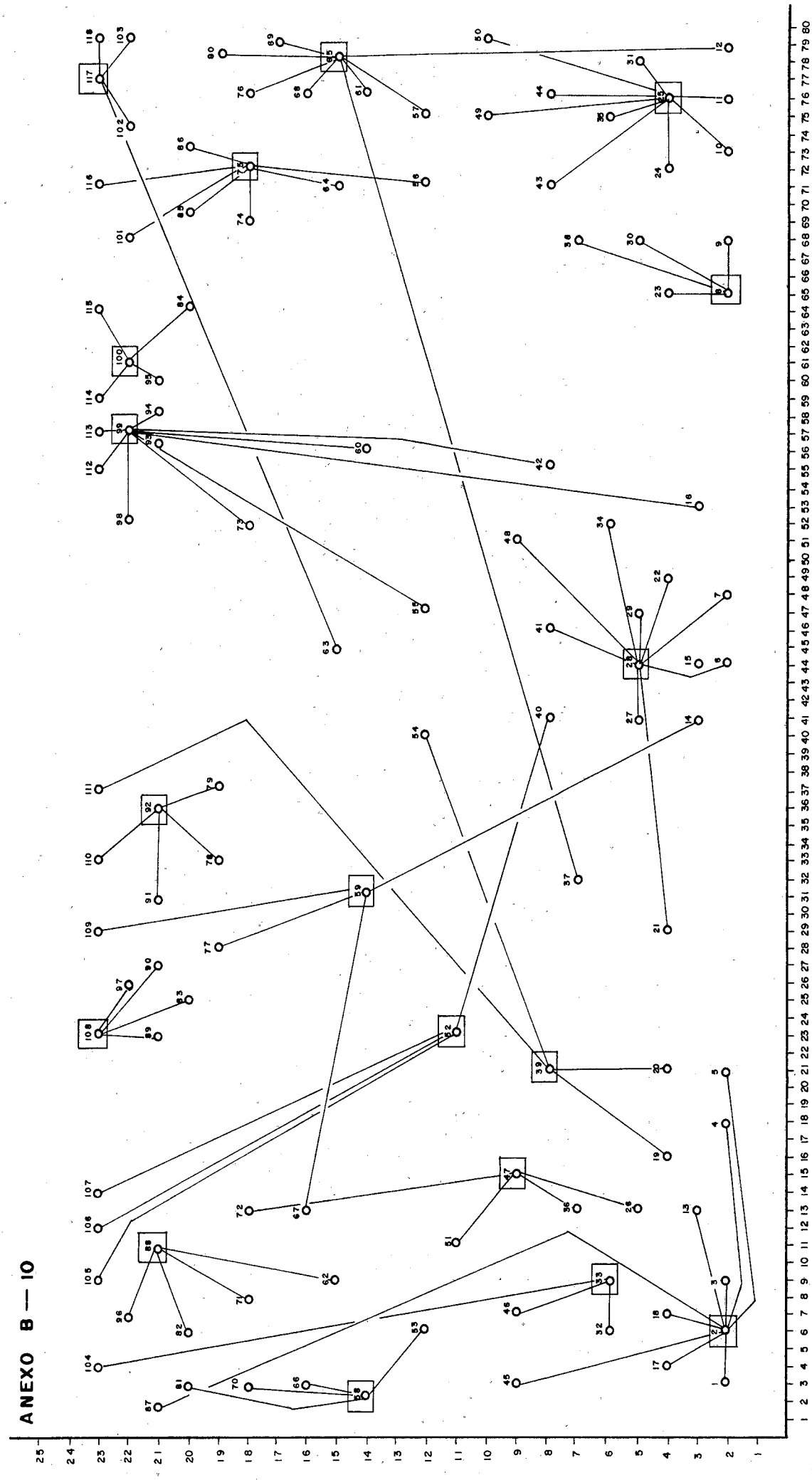
ANEXO B — 6


ANEXO B — 7

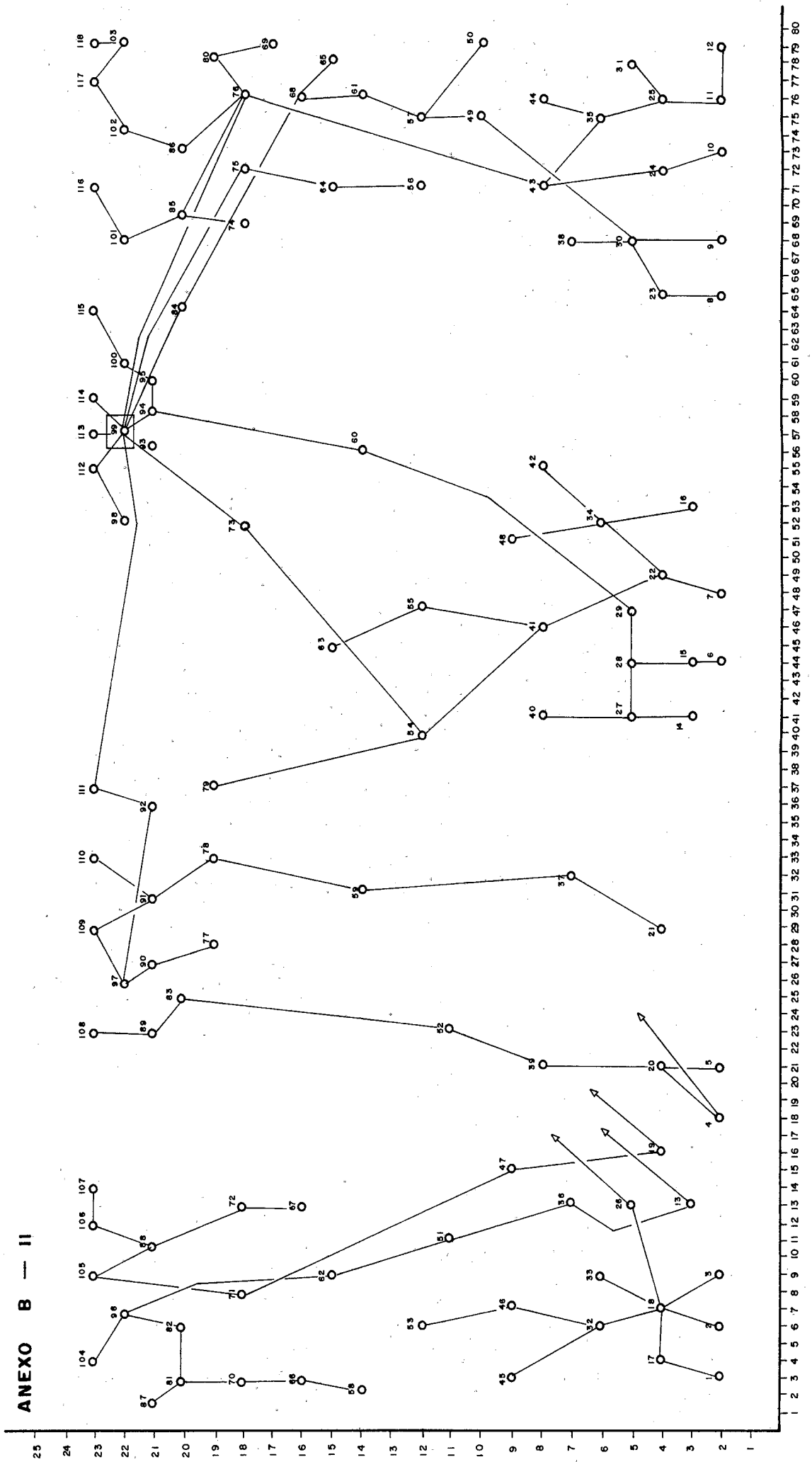

ANEXO B – 8



ANEXO B — 9



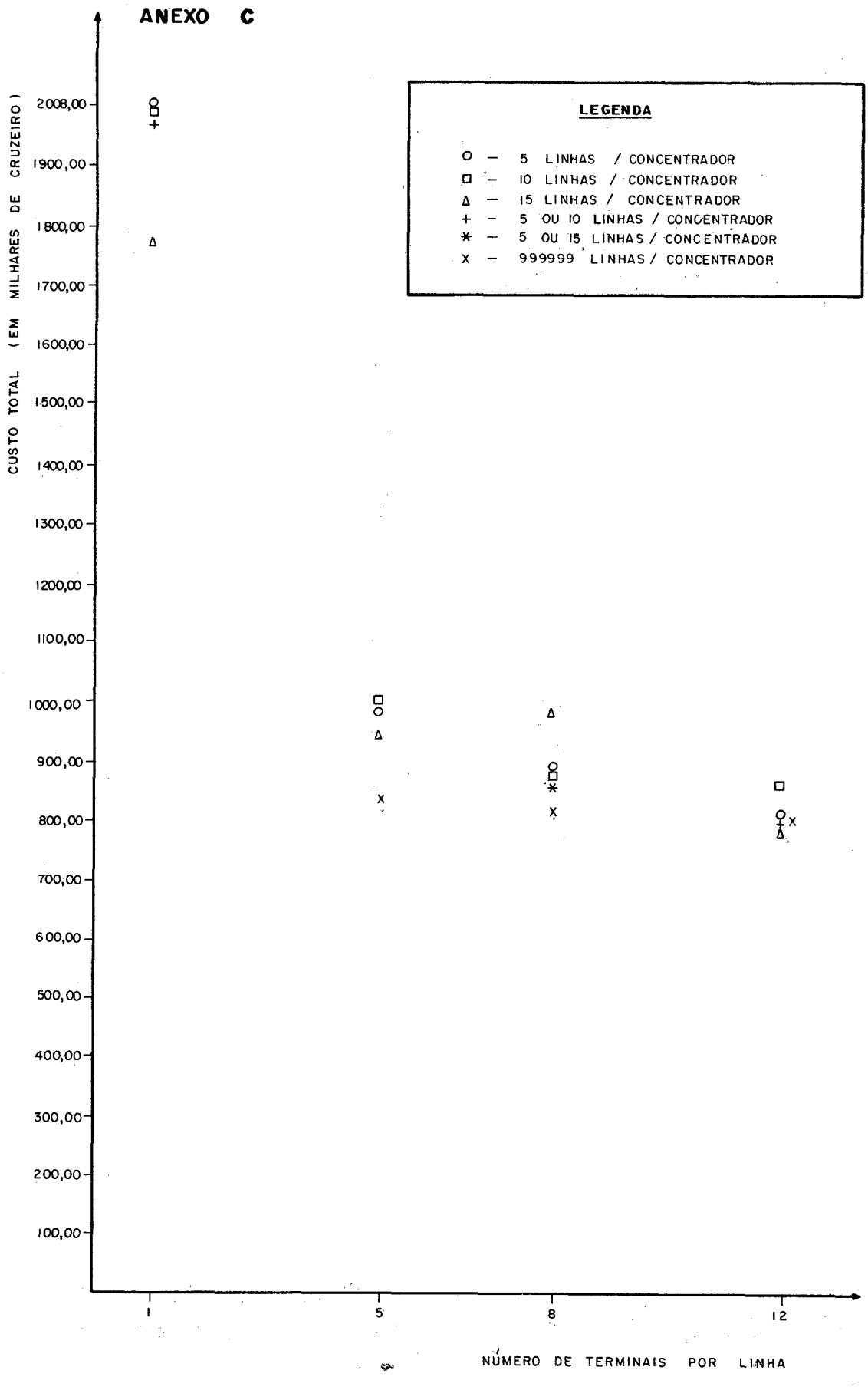
ANEXO B — 10


ANEXO B — II

ANEXO C

GRÁFICO DO CUSTO TOTAL DAS REDES EM FUNÇÃO DE DIVERSOS TIPOS DE LINHAS, PARA DIFERENTES TIPOS DE CONCENTRADORES

ANEXO C


LEGENDA
o — 5 LINHAS / CONCENTRADOR
□ — 10 LINHAS / CONCENTRADOR
Δ — 15 LINHAS / CONCENTRADOR
+ — 5 OU 10 LINHAS / CONCENTRADOR
* — 5 OU 15 LINHAS / CONCENTRADOR
x — 999999 LINHAS / CONCENTRADOR
CUSTO TOTAL (EM MILHARES DE CRUZEIRO)
2008,00
1900,00
1800,00
1700,00
1600,00
1500,00
1400,00
1300,00
1200,00
1100,00
1000,00
900,00
800,00
700,00
600,00
500,00
400,00
300,00
200,00
100,00
1
5
8
12
NÚMERO DE TERMINAIS POR LINHA

ANEXO D

ÍNDICE POR ASSUNTO