INFORMATION TO USERS

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

This copy was produced from a microfiche copy of the original document. The quality of the copy is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Canadian Theses Division
Cataloguing Branch
National Library of Canada
Ottawa, Canada K1A ON4

AVIS AUX USAGERS

LA THESE A ETE MICROFILMEE TELLE QUE NOUS L'AVONS RECUE

Cette copie a été faite à partir d'une microfiche du document original. La qualité de la copie dépend grandement de la qualité de la thèse soumise pour le microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

NOTA BENE: La qualité d'impression de certaines pages peut laisser à désirer. Microfilmée telle que nous l'avons reçue.

Division des thèses canadiennes
Direction du catalogage
Bibliothèque nationale du Canada
Ottawa, Canada K1A ON4

THE TOPOLOGICAL DESIGN OF TWO-LEVEL, MULTIDROP TELEPROCESSING NETWORKS

BY

Hugh G. Dysart

Submitted to The School of Graduate Studies
in partial fulfillment of the requirements
for the degree of Master of Applied Science.

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa Canada
September 1976

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

ABSTRACT

This thesis presents NEWCLUST, a new heuristic algorithm for the design of two-level multidrop teleprocessing networks. From experimentation with randomly generated networks of various sizes, and comparisons with the Martin-Doll and modified Bahl and Tang algorithms, the NEWCLUST algorithm is shown to be a better heuristic in network cost and computer time. It is also shown that the algorithm can easily be extended to handle multiple types of concentrators and a number of CPU's.

ACKNOWLEDGEMENTS

The author expresses his thanks and gratitude to his supervisor, Dr. N.D. Georganas for his continued support and guidance in the preparation of this thesis. A special note of thanks is expressed to the members of the Working Group on Networks for allowing the author a forum in which to express his ideas; especially Dr. Jiri Soukup for discussions leading to the topic of this thesis. Last but by no means least, the deepest gratitude is expressed to Carolyn for her moral and emotional support.

CHAPTER 1

INTRODUCTION

Consider the following problem facing the network designer of a large retail business concern that has stores or outlets located throughout Canada. Suppose there is a backbone network of long haul facilities linking the company's data processing centers, DPC, which are located throughout the provinces, to the main headquarters in Sault-St. Marie (Fig. 1.1). These DPC's are to service large geographical areas which encompass a number of cities, towns and villages. Within each of these cities or towns there are a number of company stores, each one having one or more point of sale, POS, terminals such as those used for credit checking, verification, etc. The operation of the system is then to have some data entered into the POS terminal, have that terminal communicate with its regional DPC which will then do some data processing and send a message back to the terminal. The DPC may then in turn communicate with the main DPC at the headquarters, via the long haul transmission facilities.

If the immediate problem is how to connect the terminals to the regional DPC's in an economical manner, then one has a number of options available. Figure 1.2 illustrates some of the possible choices for the layout of a regional network. The network in (a) might be desirable if very fast response times are required and/or each terminal generates a large amount of traffic. If the

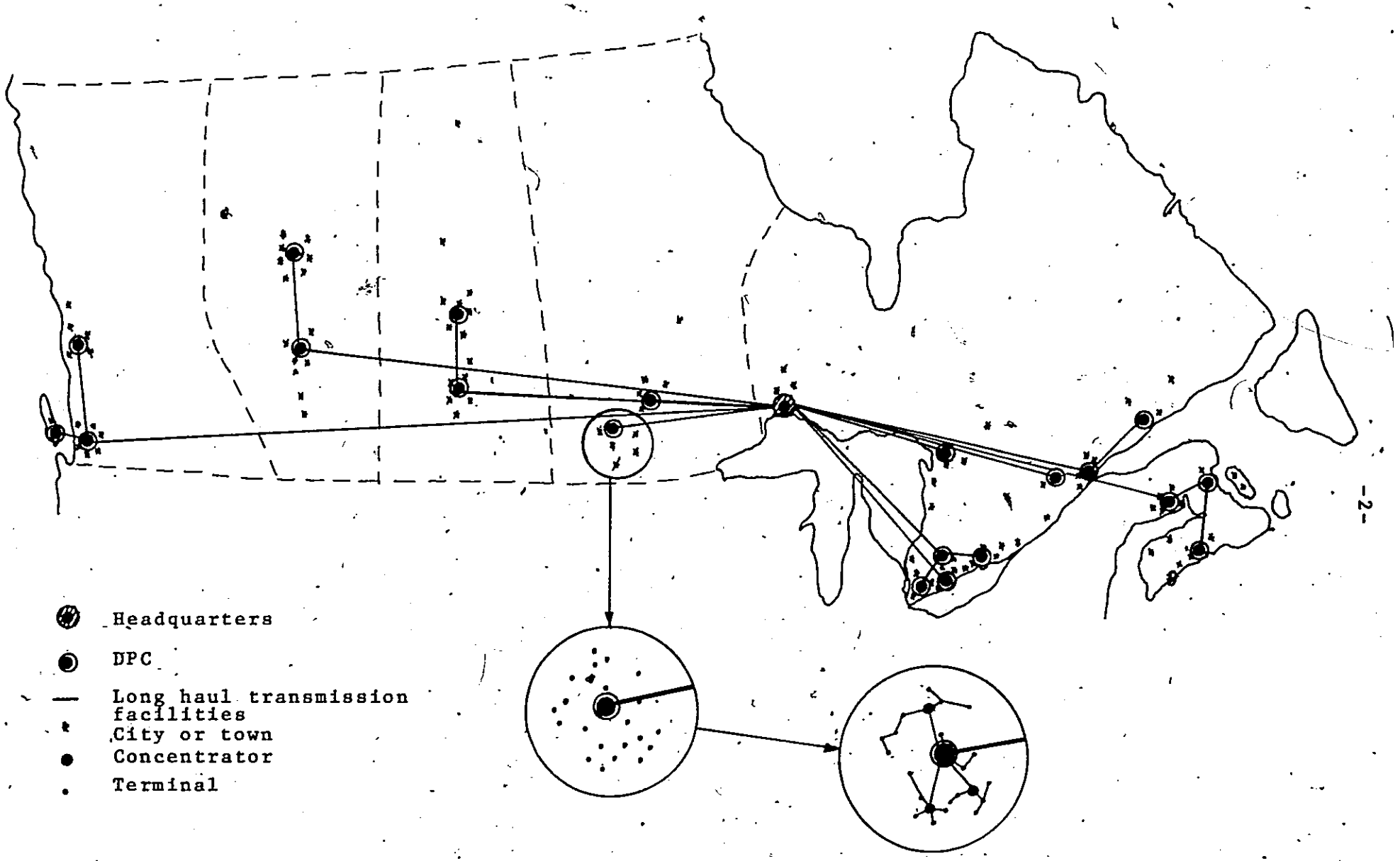
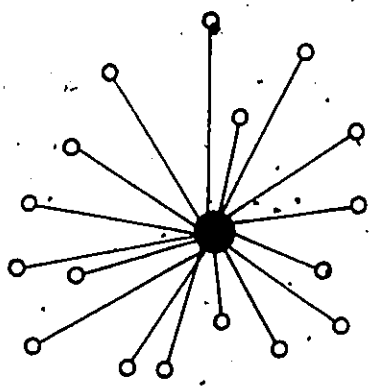
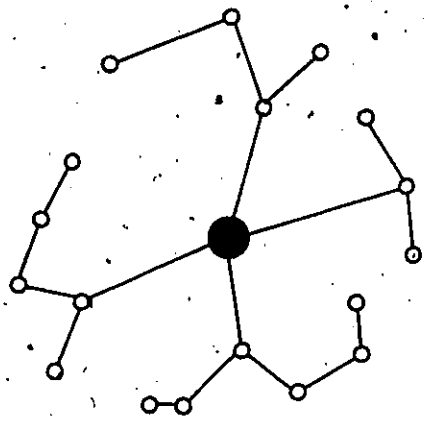


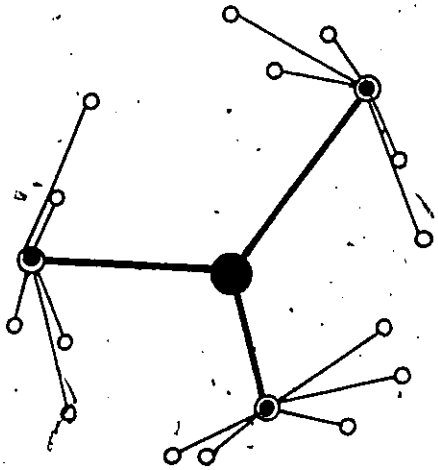
Fig. 1.1
Nation-wide POS Network



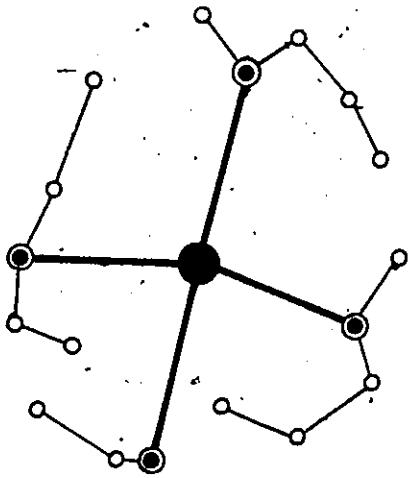
(a)
point-to-point lines



(b)
multidrop lines



(c)
Concentrators and point-to-point lines



(d)
Concentrators and multidrop lines

Fig. 1.2
Possible Network Topologies

response requirements are less restrictive the multidrop network in (b) might be a better choice since savings in line costs may be realized by having more than one terminal share a line (i.e. multidrop or multipoint line). Perhaps one might want to use concentrating devices as in (c) and (d). This might be desirable if some of the data processing can be delegated to local centers or if higher cost savings can be realized. Higher cost savings might be expected using concentrators and medium or high-speed lines because of the non-linear relationship between cost and bandwidth. That is it might be cheaper to multiplex or concentrate a number of low-speed lines onto one high-speed line, rather than connecting each of those terminals directly to the DPC via separate low-speed lines.

Now suppose that the reliability, response time and other requirements and constraints indicate that a multidrop network with concentrators, as in (d) is the best choice of topologies. The problem is then to determine the exact layout of the network, having that type of topology, such that all design constraints are met and the lowest possible cost is realized.

This is the problem that has been addressed in this thesis. That is the minimum cost design of two-level

centralized networks, where the upper level consists of low-speed multidrop lines connecting terminals to concentrators and the lower level is composed of point-to-point high-speed lines connecting concentrators to the CPU. All concentrators are assumed to have the same capacity.

We have used the generic term "concentrator" for any equipment capable of concentrating the output of a number of terminals onto a single high-speed transmission line; such as frequency and time division multiplexers and store and forward type concentrators. The term central processing unit, CPU, will be used throughout the rest of the thesis instead of DPC. This is because we are considering what would be a regional network in the example as if it were a whole network.

This type of problem has been approached from both a mathematical programming and heuristic programming points of view in the literature. Since the approach that is taken in this thesis is heuristic in nature, a discussion of heuristics and heuristic programming is in order. Heuristics may be defined as any rules of thumb, principles or devices, that aid problem solving and usually result in the reduction of the search for a solution. A heuristic program may

incorporate a number of different heuristics. By doing so the time required by the program to obtain a solution and the amount of computer storage required may be reduced. Heuristic programming can be thought of as "an approach to problem solving where the emphasis is on working towards optimum solution procedures rather than optimum solutions" [25, p.644], although optimum solutions may be obtained.

Heuristic programming differs from other types of programming (eg. mathematical), in two main ways. The first is that factors such as computer storage and time are given consideration as well as the quality of the solution produced. Thus if a heuristic program consistantly produces solutions that are within a few percent of the optimum and is a thousand times faster than the optimal procedure, then it may be in many cases a more desirable tool for problem solving. The other difference is that heuristics are justified on the basis that they have been found to be useful through experimentation and not because they arrive at any analytically derivied solution [25].

Comparisons between heuristic programs or procedures are thus generally based on computer storage and time to obtain a solution as well as the quality of the solution produced. This does, however, present some problems.

Computer storage and run times, especially for large complex algorithms or programs, depend to a large extent on how efficiently the algorithm has been coded, in what language and what machine it was run on (eg. IBM 360, IBM 370; CDC6600, etc.). Since no two people have exactly the same expertise in programming, some discrepency is bound to arise in the coding. Also if algorithms are explained only in general terms and details of how they were implemented are omitted, then one person's interpretation and implementation of the algorithm might be quite different from someone else's. Thus two identical solutions may be obtained from two different programs that have different storage and run time requirements. In general then one cannot compare results with those published in the literature even if the same language and machine have been used.

An attempt to facilitate comparisons can be made by putting a source deck of the program through an optimizing compiler and obtaining an efficient object deck of the program. This does optimize the coding of the algorithm to some extent but does not alter the program's structure. For example, if a FORTRAN source program has two DO loops in succession, both using different DO indices, I and J, then the object deck will have either I or J as the index of both the DO loops. This would reduce the storage

requirements for that section of the program by one word. However, it may be the case that the procedure accomplished by the two successive DO loops could have been accomplished more efficiently using two nested DO loops.

The only real recourse for comparison purposes is then for one to code the algorithm as efficiently as possible and base the comparison results on ratios and percentages.

In chapter two of the thesis, various approaches and algorithms for the design of the upper level, the multidrop subnetworks around the concentrators, are discussed. In chapter three, algorithms for the design of two-level networks with point-to-point lines, Fig. 1.2(c), are discussed and the existing algorithms for designing two-level multidrop networks are presented. A new heuristic algorithm for designing the latter type of two-level networks is presented in chapter 4 along with its evaluation and extentions.

CHAPTER 2

ALGORITHMS FOR THE DESIGN OF MULTIDROP NETWORKS

2.1 Introduction

A tree or multidrop network can be simply specified as any network in which there is a unique path between any terminal and the center, such that there exists some path on which there is more than one terminal or node. Figure 2.1 illustrates a tree network that has a number of multidrop lines. A multidrop line is any branch of the tree that has more than one terminal connected to it.

A minimum spanning tree (MST) is a tree, as specified above, which has the minimum cost over all spanning trees, for a particular set of nodes. Since in most network designs constraints are imposed on the problem, the MST is usually not a feasible solution, in that it violates some constraint. Thus the problem of designing a multidrop or tree network can be viewed as a constrained MST (CMST) problem. The CMST problem can be formally stated as:

Given: a) A set of terminals $\{n_i\}$ $i=1,2,\ldots N$

b) A set of loads (eg. traffic rates) $\{t_i\}$, $i=1,2,\ldots N$, associated with the nodes.

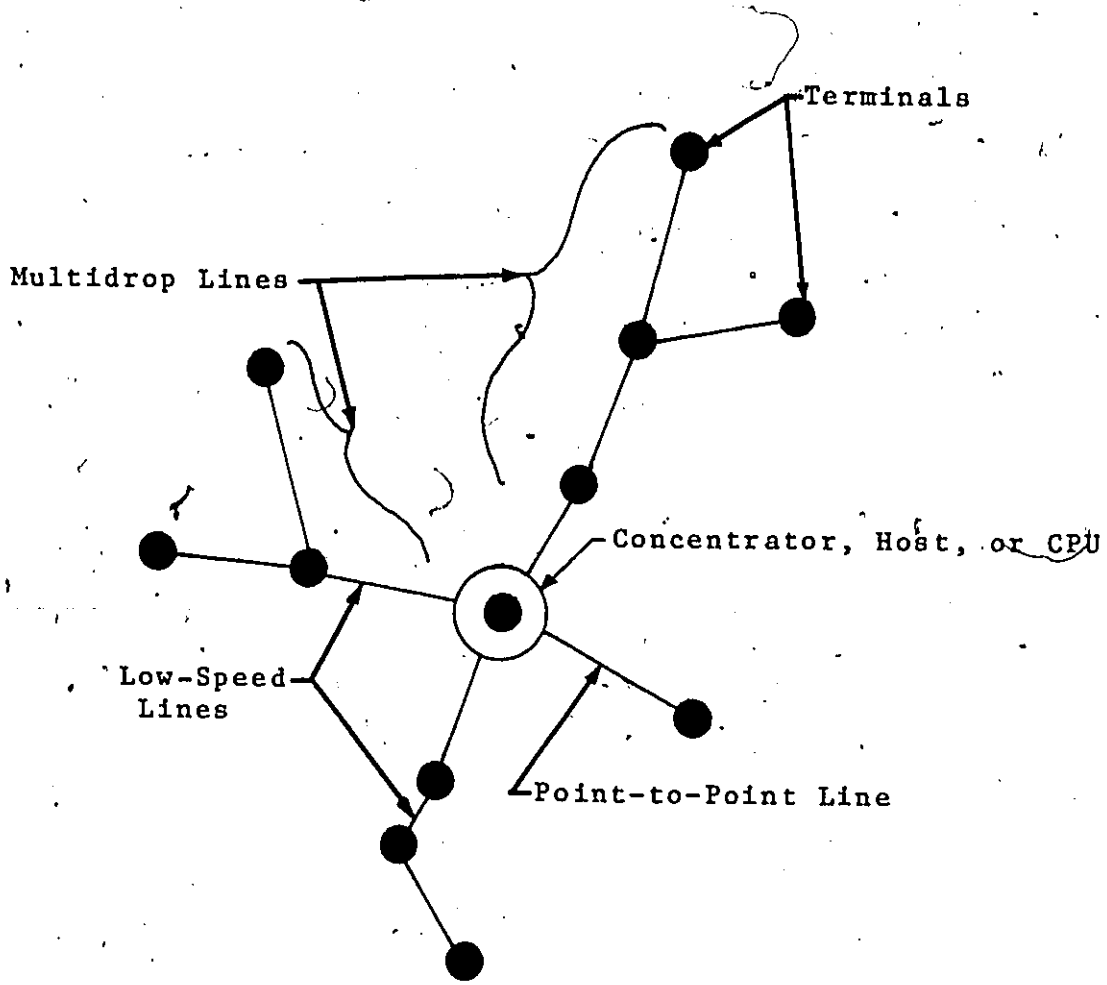


Fig. 2.1
A Tree Network

c) A cost or distance matrix.

$D = \{d_{ij}\}$, $i,j = 1,2, \ldots, N$, representing connection costs between nodes.

d) A set of constraints C

Find: The minimum cost spanning tree over the set of nodes subject to the restriction that no constraint is violated.

In general there may be more than one type of constraints. For example, there might be a capacity constraint that requires that the traffic flowing over any link be less than some maximum. A reliability constraint might also be included, which restricts the number of terminals on a multidrop line.

The problem may be formulated as an integer or dynamic programming [3,10] problem, but the solution may become unwieldly if the network size is much larger than 20 to 40 nodes or terminals. Other techniques for solving this problem can be grouped into three general categories; optimal, CMST and tree transformations.

2.2 Solution Techniques

2.2-1 Optimal

Most of the algorithms that produce globally optimal solutions make use of Branch and Bound (BB)

techniques, although in [2] a graph theoretic approach has been used.

The basic BB procedure of Chandy and Russel [3] uses the MST as the starting topology. The algorithm then proceeds to partition the set of all feasible solutions into smaller and smaller subsets, utilizing the central links of the MST in the initial partition. Each of these subsets is characterized by a set of "established" links (the ones included in all the subsets' feasible solutions) and a set of "disallowed" links (those excluded). Each subset is also characterized by a lower bound on the cost of all feasible solutions in that subset, which is the MST cost, with the disallowed link costs set to infinity. A link that is neither established nor disallowed is considered "free". These free links are used at each step to partition that subset, S, of the current network partition, which has the lowest lower bound. The subset S is partitioned into two smaller subsets S_1 and S_2 by making a free link of S established in S_1 and disallowed in S_2. New lower bounds are calculated for S_1 and S_2 and the partitioning continues until a subset with a feasible solution, whose cost is the lowest lower bound, is found.

That solution is then the optimal solution, since the lowest lower bound is also the lower bound to the optimal solution.

The methods in [4,5] take advantage of some spécial properties of the CMST problem to accelerate the convergence of the basic BB technique. However, this technique, because of its lengthy convergence time, is not of much practical use in finding solutions to networks with much more than 50 nodes.

2.2-2 CMST Methods

Both the CMST and tree transformation methods are heuristics that produce locally optimum solutions, which are in some cases within a few per cent of the optimum [3] .

The CMST type of heuristic arrives at the final solution by successively adding links to a tree, without violating the constraints, until all nodes have been brought into the tree and no further cost improvements can be realized. The difference between the various algorithms lies in the way in which terminals are brought into the tree. Kershenbaum [6] has used this fact in deriving a more general heuristic

algorithm in which a number of other algorithms are special cases.

All the algorithms of the CMST type consider each node initially as a separate component and attempt to join components together in a least costly manner, until a constrained spanning tree is found. In general each algorithm may start by joining different components and end up with different solutions. The algorithms choose different links to be added to the tree because they consider the terminals in different orders. They implicitly assign weights to terminals to decide the order in which components are to be merged. A brief description of some of these algorithms is outlined in the following sections.

2.2-2.1 Sharma - El Bardai and Martin Algorithms

The algorithm no. 2 of Sharma [7] and the one by J. Martin [8] grow feasible tree solutions starting with the farthest terminal from the center. The only difference between the two algorithms is that Sharma's forces the nodes on a given branch of the tree into a MST topology whereas Martin's does not. The algorithm grows branches of a tree by starting with

the furthest node from the center, not already in a branch, and filling up the branch with nearest feasible neighbours until no more can be added without violating the constraints. If all nodes have been brought into the tree then the algorithm terminates, if they have not, then a new branch is formed. If a tradeoff function, representing the effect of joining node i to node j, is defined as $t_{ij} = d_{ij}/w_i$, and the weight assigned to each node is its distance from the center 1 (e.g. $w_i = d_{i1}$), then choosing the minimum t_{ij}, when a new branch is formed, over all nodes not already in the tree is equivalent to choosing that node which is farthest from the center.

2.2-2.2 Esau-Williams Algorithm

Each terminal and the center are initially in separate components. A tradeoff function t_{ij} is used to determine the order in which compoents are joined. The tradeoff function is defined as the cost saving realized by connecting terminal i to j rather than connecting the component containing terminal i to the center 1 (i.e. $t_{ij} = d_{i1} - d_{ij}$) [9] . At each stage the maximum tradeoff is found ($t_{i^*j^*} = \text{MAX}\ \{t_{ij}\}$) and the link (i* j*) is brought into the tree. The new component containing i* and j* has its tradeoffs recalculated if necessary. The algorithm terminates when no further cost savings can be

realized. If any nodes are then not in the tree they are directly connected to the center. Another way of describing the algorithm is that it initially starts with a star topology (all nodes directly connected to the center) and at each iteration reduces the overall cost by replacing a central link with a less costly non-central link until no further reductions are possible. If the tradeoff function is written in the form $t_{ij} = d_{ij} - w_i$, where $w_i = d_{i1}$ and at each stage the minimum t_{ij} is found rather than the maximum, the algorithm will performn the same. However in this form one can see that the weighting factor, w_i, assigns larger weights to terminals which are far from the center, thus tending to minimize the t_{ij} and so giving preference to links emanating from those nodes.

2.2-2.3 Vogel's Approximation Method (VAM)

The tradeoff function for the VAM algorithm [10,11] between terminals i and j, is defined as the difference between the cost, b_i, of connecting terminal i to its second closest feasible neighbour and the cost, a_i, of connecting it to its closest feasible neighbour minus the cost, d_{ij}, of the direct connection between i and j. (i.e $t_{ij} = b_i - a_i - d_{ij}$).

At each stage the largest t_{ij} is found and terminal i is connected to its nearest feasible neighbour, thus forming or augmenting a segment or branch of the tree. The cost of joining two segments, components, or branches is defined as the cost of the least costly link between terminals in one segment and terminals in the other. Again we see that the tradeoff function can be rewritten in the form $t_{ij} = d_{ij} - w_i$ where $w_i = b_i - a_i$ and again the minimum t_{ij} at each stage is found.

2.2-2.4 Kershenbaums' Unified Algorithm

The unified algorithm [6,18] of Kershenbaum incorporates as special cases the Prim [12] and Kruskal [13] algorithms, which are MST algorithms. It also incorporates the Esau-Williams and VAM algorithms using the tradeoff function and respective weights as defined above. The unified algorithm is basically the Kruskal algorithm with two modifications. The first is that the ordering of the links, smallest first, is done on the basis of the tradeoff function, using the appropriate weighting rule. The other modification is that whenever two components are being considered for merging, the feasibility of the merger is checked. The advantage

of this algorithm is that several heuristics may be used in succession or simultaneously producing results that are at least as good and in some cases better than any single heuristic can produce.

2.2-3 Tree Transformation Methods

Tree transformation methods are local optimization procedures that use any tree as a starting topology and apply successive elementary tree transformations in such a way as to reduce the overall network cost while keeping within the constraints. Alternatively, they may apply transformations in such a way as to minimize the additional costs that must be incurred to transform an infeasible topology into a feasible one.

A method of this type, by Frank et al [14,15] is used in the design of tree or multidrop networks with optimum link capacities. The transformation used is that for a given tree some node i and its closest neighbour, not already connected to i, are chosen. The link (i,j) is added to the tree forming a circuit or a loop. If this loop has links (j,j_1), ..., (j_k,i), then new treees are formed by deleting each link in turn. When a link is deleted, an optimal capacity assignment algorithm is used to

determine the new tree cost. The algorithm scans nodes sequentially and considers link additions from each node to its d nearest neighbours. When a lower cost tree is found then that tree is adopted and the node scan begins again. When no further cost improvement can be found then a local optimum solution has been found. A number of different starting topologies are considered with each resulting in a local optimum. The best of these local optimums is then adopted as the final solution.

Another tree transformation type of heuristic has been proposed by Elias [4]. This algorithm uses the MST as a starting topology. If the MST is infeasible then the tree is modified by iteratively exchanging links to satisfy the constraints, while incurring the minimum additional cost. Initially the network is partitioned into two sets. SET 1 contains all the nodes on segments of the tree that satisfy the constraints. SET 2 is the complement of SET 1, or all other nodes in the network except those in SET 1. Links are then iteratively deleted between nodes in SET 1 and replaced by links between nodes in SET 1 and SET 2. The criteria for link exchanges is that they should result in the smallest additional cost and lead

to a reduction in the overall amount by which the constraints are violated. Once the tree structure satisfies the constraints, an attempt to further reduce the cost is made by attempting to feasibly recombine the tree's segments and give them a MST topology.

2.3 Comparisons

From some comparisons of some of the different heuristic algorithms [3,4,6,16] it appears that the widely accepted Esau-Williams algorithm is generally the best heuristic available for the design of multipoint networks with a single line capacity. It usually produces solutions that are within 5% of the optimum and is one of the faster heuristics. There are, however, heuristic that produce solutions that are as close or closer to the optimum [17,6] but this is usually achieved at the expense of computer time and program complexity.

Since the multidrop layout problem is embedded within the overall design problem considered and may be required to be solved a number of times, it is desirable to use a heuristic that produces fast and good solutions. We have adopted the Esau-Williams

algorithm to use in our design problem since it satisfies the above conditions and also for comparison purposes since it has been used in one of the other design algorithms we will be comparing ours to.

CHAPTER 3

THE CONCENTRATOR LOCATION PROBLEM

3.1 Introduction

The general concentrator location problem can be broken down into two major sub-problems. The first is the location problem which involves determining the number, positions and types of concentrators to be used in the network. The second is the allocation problem which involves the partitioning of the network in such a way that the line layout of the concentrator subnetworks will produce the desired minimum cost feasible topology. That topology, for centralized networks, is usually that of either a star, tree or loop. In order to solve the design problem effectively, however, it is necessary to consider the location and allocation problems simultaneously, within the context of the overal minimum cost design problem [37].

Most of the more commonly known algorithms for the design of centralized communication or teleprocessing networks are modified versions of ones that were originally developed for selecting warehouse locations. A number of mathematical programming approaches have been used to solve these warehousing problems. They cover the areas of linear, nonlinear,

mixed integer and dynamic programming [19-24] . Optimal solutions may be found in some cases using mixed integer programming in conjunction with decomposition and Branch and Bound techniques. However, experience in solving the warehouse location problems by these methods has indicated that they are rather inefficient in solving meddium to large-scale problems [20], and that one must usually resort to approximate or heuristic methods in order to reduce computer storage and run time requirements.

Most of these algorithms can be characterized as some type of one-point-move search, among the points in the solution space. For example, if there are three possible locations at A, B and C then the point (100) would represent a possible solution in which a warehouse is located only at location A, and the point (101) implies warehouses are located or open at A and C and closed at B. The solution space can then be thought of as the set of solutions corresponding to the set of these points which includes all combinations of the possible locations. The "Add" algorithms' [25] search is a directed one, starting at the point(000...0)

and moving towards (111...1). That is to say that at each move or step in the Add algorithm some plant or warehouse that is not already open is opened, if that move proves economical. In the Drop algorithm [26], the search is directed in the opposite direction starting with all warehouses open (111...1) and at each stage closing a warehouse, thus moving towards the (000...0) point. Since both these algorithms terminate their search whenever the opening or closing of another warehouse will not improve the present solution, they result in local optima. Other heuristic algorithms [27,28,30] have tried to circumvent the local optimum problem by either starting their search at some randomly chosen point or changing the direction of the search during the course of the solution process. However, their results have not indicated any drastic improvements over either the Add or Drop approaches.

3.2 Existing Algorithms

The problem of designing networks with concentrators and point-to-point lines has been approached from a number of different directions in the literature but very little literature on designing nets with concentrators and multidrop lines or loops is available.

Since this thesis is concerned only with the multidrop or tree type of topology, a brief description of some of the existing algorithms that consider the point-to-point case will be given, followed by a detailed description of those that consider multidrop lines.

3.2-1 Concentration and Point-to-Point Lines

3.2-1.1 Add Approach

The Add approach can be illustrated by the algorithms of Martin [31] & Kuehn-Hamburger [25]. There are two main differences between these two algorithms. One is that Martin initially considers all terminals in the network as possible concentrator sites whereas Kuehn and Hamburger consider only some selective subset of the terminals. The other difference is that at each iteration, Martin considers locating a concentrator from the set of all possible remaining candidate sites, whereas Kuehn and Hamburger only consider a small subset of the remaining candidate sites. The algorithm by Martin will be described here.

Initially every terminal in the network is considered as a possible concentrator site. Each site is considered in turn, independently from the others. For each site terminals are assigned to it until the capacity of the concentrator at the site is full or until no other terminals can be economically connected to it. The criterion for assigning terminals to concentrators is that those with the highest gains are connected first. The gain function is defined as the difference in cost between connecting a terminal i to the central processing unit 1 (CPU) and connecting it to a concentrator j.

i.e. $g_{ij} = c_{i1} - c_{ij}$; (only $g_{ij} \geq 0$ are considered)

where g_{ij} = gain associated with connecting terminal i to concentrator j

c_{i1} = cost of connecting terminal i to the CPU

c_{ij} = cost of connecting terminal i to the concentrator at location j

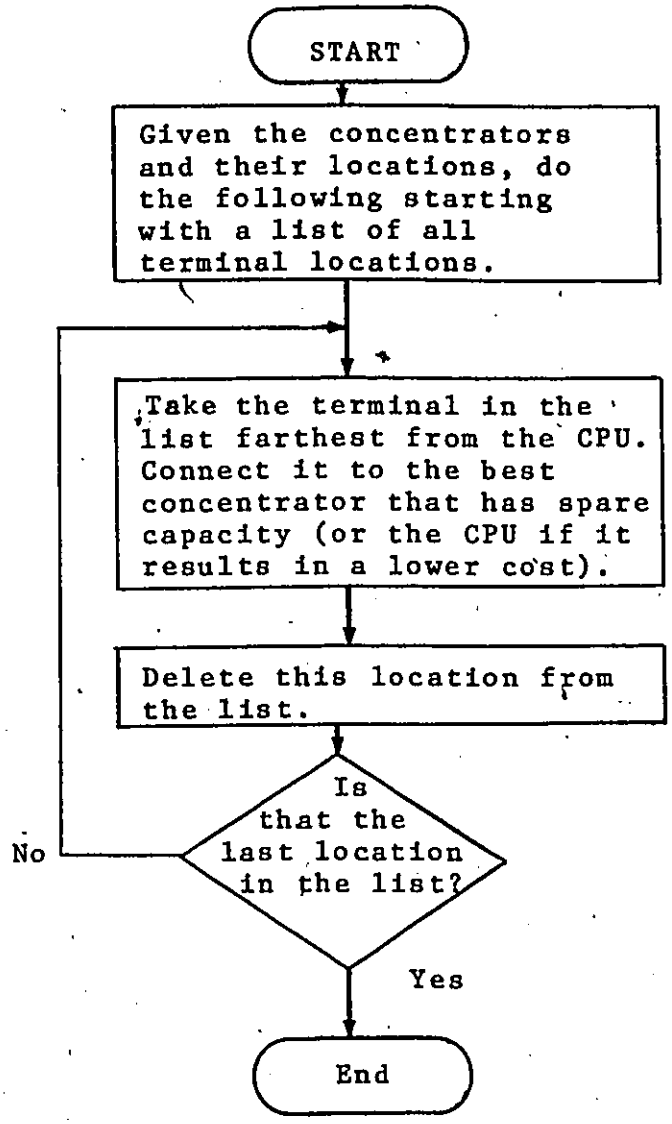


Fig. 3.1

Stage two of Martin's algorithm

The gain for each site can then be calculated as:

$$G_j = \sum_{i=1}^{N} g_{ij}\, x_{ij} - f_j$$

where g_{ij} = is as defined above

$$x_{ij} = \begin{cases} 1 \text{ if terminal } i \text{ is connected to} \\ \text{concentrator } j; \\ 0 \text{ otherwise.} \end{cases}$$

f_j = cost of locating a concentrator at location j.

Once the gain for every site has been calculated, then that site which has the highest gain has a concentrator located at it and this site and all the terminals that were connected to it are removed from further consideration. The whole process is then repeated for the remaining terminals. The algorithm terminates when it is no longer economical to locate another concentrator. The concentrators, thus located, are then used in a second stage for the terminal assignments as illustrated in Fig. 3.1.

3.2-1.2, Tree Search Algorithm

In this algorithm [32] it is assumed that a subset of the terminals is considered as the set of

possible concentrator sites. The concentrators are ranked according to their maximum savings and only a subset of them is retained for further analysis. From this subset of possible sites, a tree of solutions is built up by considering at each stage certain combinations of some number of the locations. At the end of each stage, that combination which produces the best solution is saved. When no further cost savings can be realized, then the best solution out of the tree of solutions is chosen. It appears that this algorithm is similar in a number of ways to the Add [25] algorithm except that concentrators aren't located sequentially, thus allowing for more flexibility in the search.

3.2-1.3 Drop Approach

This algorithm by Bahl and Tang [33] was used to compare our new algorithm, thus a more detailed description of the algorithm and its modifications will be given in the next section.

This algorithm is an iterative procedure which is based on the assumption that all terminals are potential concentrator sites. Gain functions

as described in 3.2-1.1 are used to assign terminals to concentrators and to determine the order in which concentrators are dropped from the network. Initially the maximum number of terminals is assigned to each concentrator, if possible, in such a way that the cost savings or gains g_{ij} are maximized. Thus a terminal may be connected to many concentrators initially. Terminals which cannot be accommodated by any concentrator are then connected to the CPU. During each iteration, a link between a terminal and concentrator is either deleted or inserted. Whenever a link is removed, the gain G_j of the associated concentrator is checked. If the gain is negative, indicating that the cost saving to the network of having that concentrator is less than the cost of locating it there, then that concentrator is dropped along with its links. The algorithm terminates when each terminal is connected to only one concentrator or the CPU. This algorithm, although it shows some cost improvement over the Add algorithm, is usually much slower.

3.2-1.4 Selective Combinatorial Optimization

This procedure [34] involves combinatorial optimization over certain, selected subsets of the network. It starts by partitioning the network into a number of subsets or clusters. Terminals that cannot be added to any cluster are considered as possible concentrator sites themselves. After the initial partitioning, the optimal solution for each cluster is found. Each cluster or partition then has a set of optimal locations. The network is then repartitioned, using these locally optimum location sets, into another set of subsets or clusters. The locally optimum solutions are then extended over larger subsets by forming pairwise unions of subsets. These larger subsets are then considered for optimization. Once this stage is completed then the network is again repartitioned into a new set of subsets based on the new set of concentrator locations. The procedure of pairwise optimization is then applied again.

It is shown in the paper that pairwise optimization is a necessary but not sufficient condition for global optimality. This algorithm also showed some improvement over the Add algorithm. The major

advantage of this algorithm is that the concentrator sites are solely determined by the algorithm, whereas, in others a possible set must be determined a priori.

3.2-1.5 Mathematical Approach

In this paper [35] the problem of determining the locations and boundaries of local exchanges (analogous to concentrators) in a telephone network is mathematically formulated as a mixed integer quadratic programming problem. The Capacitated Plant Location problem which is analogous to the concentrator location problem is shown to be a special case of the more general problem mentioned above. Only mathematical models and possible solution methods are presented.

3.2-1.6 Linear Regression Clustering (LRC)

In a recent paper [36] the problem has been attacked by clustering terminals on the basis of connection costs between terminals and then optimizing the locations, capacity and number of concentrators in each cluster.

Initially, all terminals are in one cluster P_o. A linear regression line for that cluster, which

represents the best mean-square fit to the terminals in the cluster, is calculated. This line is then used to partition the cluster into two smaller clusters as shown symbolically in Fig. 3.2(b). The partitioning is repeated until all terminals are associated with a cluster that has its total traffic requirement R_i less than or equal to the minimum available concentrator capacity C_1. Thus a tree of clusters is growing as shown in Fig. 3.2(a), where the solid nodes are clusters with capacity requirements less than or equal to C_1.

Once the tree has been derived, the final solution is determined in the following manner. Some cluster P_j (eg. P_7) is chosen whose requirement $R_j < C_m$, where C_m is the capacity of the maximum-capacity concentrator, and which is a descendant of a cluster P_i (eg. P_3) for which $R_i > C_m$. The optimal position for a concentrator in P_j and the associated minimum cost are calculated. This is also done for all of P_js sub-clusters (eg. P_{11}, P_{15}, P_{16}, P_{12}), always using concentrators whose capacities are the minimum sufficient to handle the traffic of each sub-cluster. The final design for P_j is then chosen as that combination of sub-clusters and concentrators which minimizes the total costs.

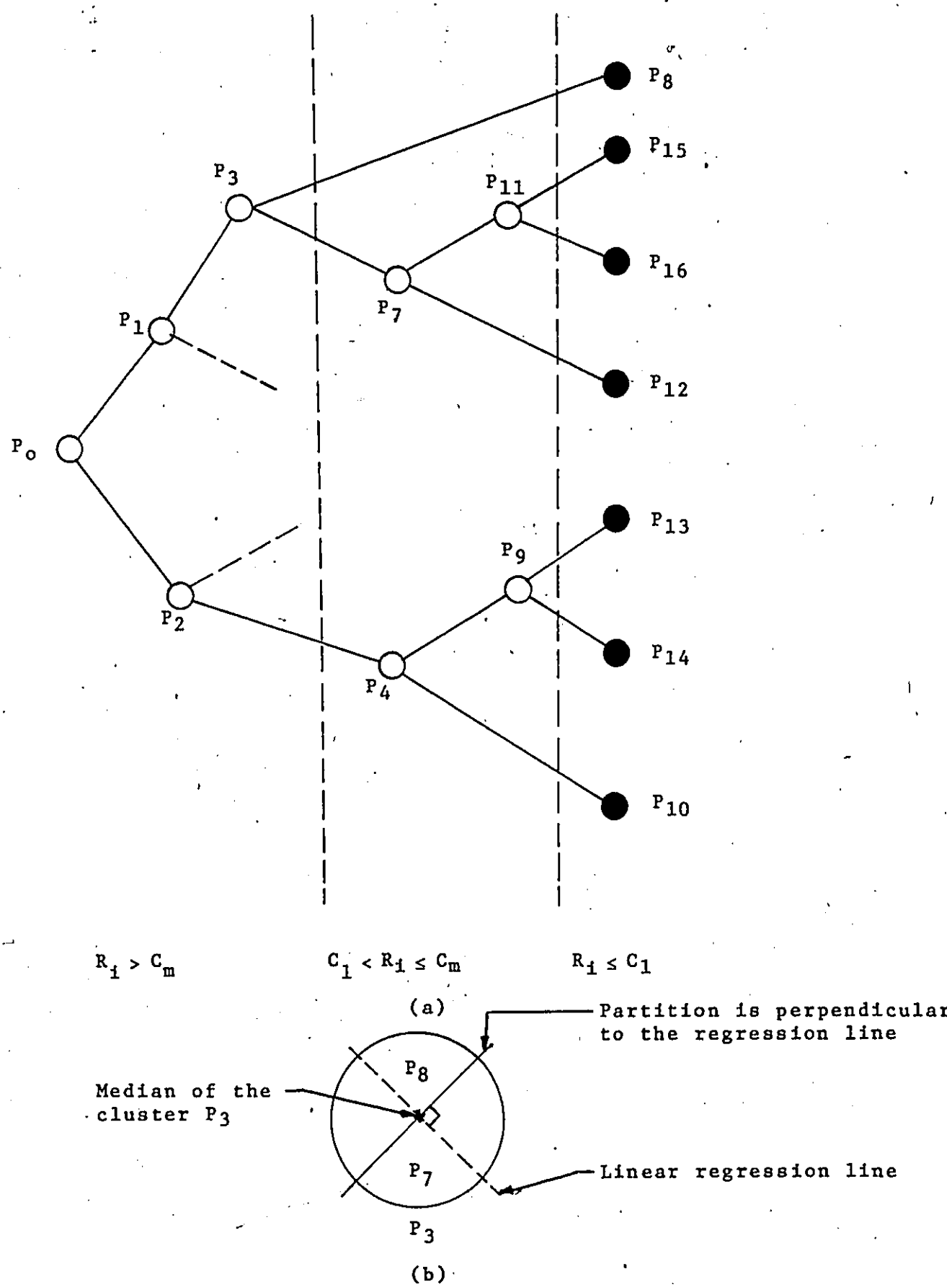


Fig. 3.2
Linear Regression Clustering Method

For example, starting with P_7 which is a descendant of P_3 there are three possible choices. One is to locate one concentrator in P_7 having the minimum required capacity greater than C_1. Another is to locate one concentrator in P_{11} with $C_1 < R_{11} < Cm$ and another one in P_{12} with capacity C_1. The third choice is to locate three concentrators each with capacity C_1 in each of P_{15}, P_{16}, and P_{12}. The whole process is then repeated with some other P_j and continued until all terminals have been associated with a concentrator. Once this stage is reached then some post-processing is done to further reduce the network cost. This algorithm shows substantial savings in both time and network cost when it is compared to the Add algorithm.

3.3 Concentration and Multidrop Lines

All of the existing algorithms [31,38] that consider locating concentrators in multidropped networks, transform the problem to the one of locating concentrators in point-to-point networks. The transformation is accomplished by estimating the multidrop line costs for the terminals and then using these costs as equivalent point to point costs in an algorithm designed for analyzing point-to-point networks. It is necessary for these algorithms to use estimated costs because they are of the Add type

and it is impossible to know à priori the actual multidrop line costs using this approach.

The algorithm by Bahl and Tang [33] is included in this section because we have used the concentrator sites that are determined by the algorithm as the final concentrator sites and then grown a multidrop network around them. This algorithm was also used for comparing our algorithm because it uses a similar Drop approach to locate concentrators.

3.3-1 Martin - Doll Add Approach

Both Martin [31] and Doll [39-41] have used an iterative approach in locating concentrators in multidrop networks. Their approach has been to locate concentrators one at a time and once the final set of concentrator sites has been determined to use a many-center assignment and multidrop optimization procedure to arrive at the final layout of the multidrop network. The basic multidrop optimization procedure used is the Esau-Williams algorithm.

The algorithm considers all terminal locations as possible concentrator sites. It is assumed that a concentrator has a fixed capacity and number of lines

that can be directly connected to it. The algorithm scans outward from a concentrator assigning terminals to the concentrators lines, considering the closest terminal first. The scan is continued until all the line capacities and thus the concentrator capacity is filled or until no other terminal can be economically connected to the concentrator. The terminals thus assigned to the concentrator have their mean distance from the CPU and concentrator calculated. These averages and the actual number of the concentrator's low-speed lines that were used are then used as a basis for determining the estimated cost saving of connecting these terminals to the CPU via the concentrator.

The estimated cost saving of using the concentrator in the network is determined by a cost density factor which can be formulated as:

DENS (J) = LIN (DCPU-DCON) - COST (J)

where

DENS (J) is the cost density factor for locating a concentrator at location J.

LIN is the actual number of concentrator J's lines that were used.

DCPU is the average low-speed line cost to connect a terminal to the CPU.

DCON is the average low-speed line cost to connect a terminal to the concentrator at location J.

COST(J) is the cost of locating a concentrator at location J. (high-speed line cost plus fixed costs).

Initially the above procedure is done for each terminal location in the list. The location that has the highest cost density factor is chosen as a concentrator site. That location and all the terminals that were connected to the concentrator at that location are removed from the list. The entire process is then repeated for the remaining locations in the list. The algorithm terminates when the largest cost density factor is negative. The final set of concentrator locations is then used in a many-center multipoint optimization procedure to determine the final network layout. A general flow chart of the algorithm is given in Fig. 3.3.

Although this method produces good solutions, it has several drawbacks. One is that the computational

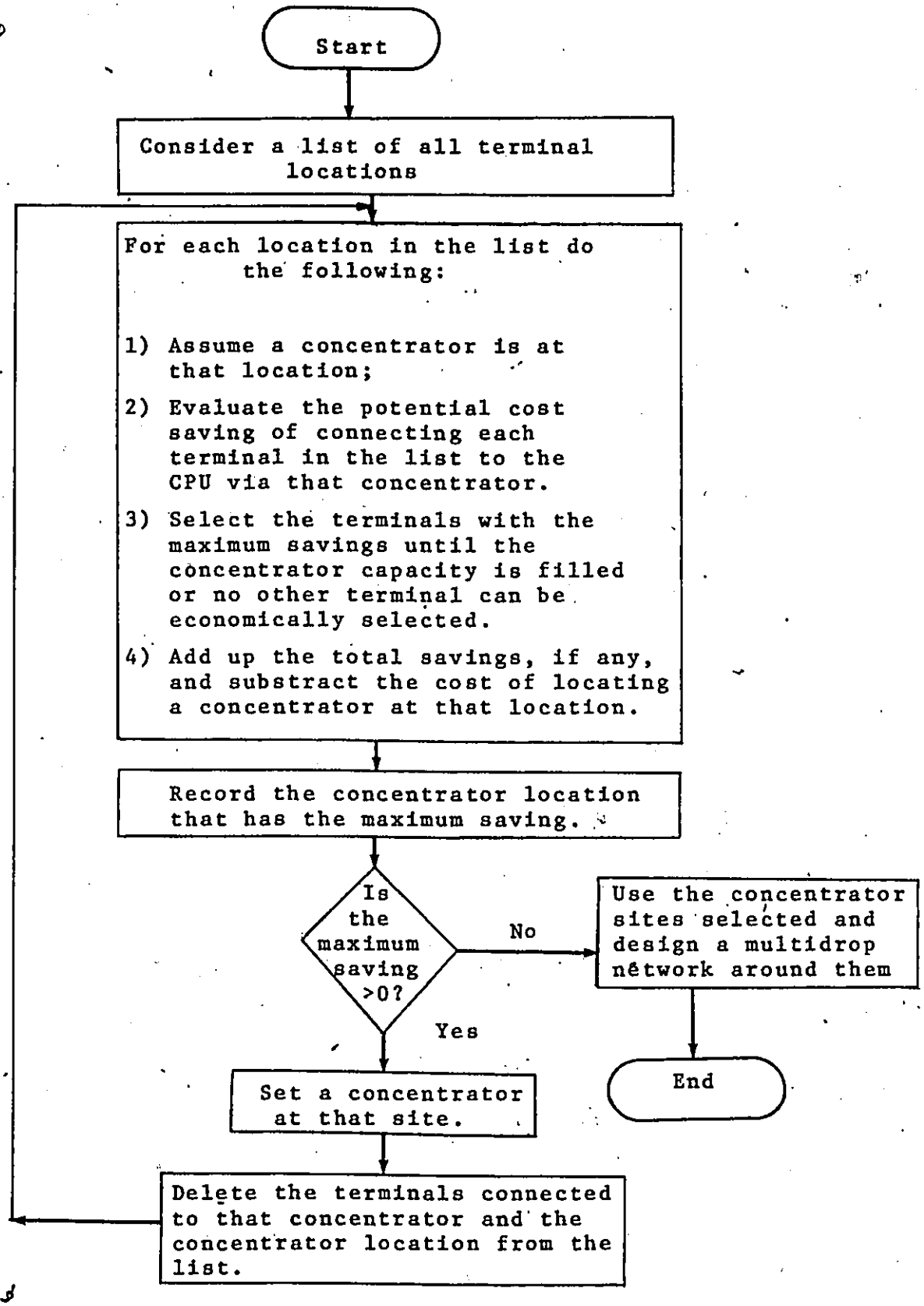


Fig. 3.3
Flow Chart for the Martin-Doll Algorithm

time may become excessive for large networks since each terminal location is considered as a possible concentrator site. Suppose, for example, in a network with N terminals, k concentrators are located, C_1, C_2 ..., C_k. Each time a concentrator is located, the terminals that were assigned to it are removed. Let $n_1, n_2, \ldots, n_k$ be the number of terminals removed when concentrators C_1, $C_2 \ldots C_k$ are located respectively. Thus in the first iteration of the algorithm, all N locations must be scanned and cost density factors for each calculated. The second iteration scans $N-n_1$, locations, the third $N-(n_1 + n_2)$ and the i^{th} $N-(\sum_{j=1}^{i-1} n_j)$ locations. Thus if N=100 and $n_1=n_2=n_3=10$, during the first four iterations, 280 scans must be made, and thus 280 cost density factors calculated.

Another drawback is the basis for determining the estimated cost savings. In the first term of the relation with DENS(J),

$$DCPU = \sum_{i=1}^{n_j} c_{io} / n_j$$

and

$$DCON = \sum_{i=1}^{n_j} c_{ij} / n_j$$

where n_j is the number of terminals connected to the concentrator at location j

C_{io} is the cost of connecting terminal i to the CPU (o).

C_{ij} is the cost of connecting terminal i to the concentrator at location j.

Thus the first term can be written as:

$$LIN(DCPU-DCON) = \frac{LIN}{n_j} \left(\sum_{i=1}^{n_j} C_{io} - \sum_{i=1}^{n_j} C_{ij}\right)$$

$$= a \sum_{i=1}^{n_j} g_{ij}$$

where $$a = \frac{LIN}{n_j}, \quad a \leq 1$$

$g_{ij} = C_{io} - C_{ij}$ and is the cost saving to the network of connecting terminal i to concentrator j rather than directly to the CPU.

These g_{ij}'s are the same as the ones used in the previous relations. Thus the estimated tree cost or multidrop network cost for a concentrator is just some fraction of the star or point-to-point cost. Even though there is some justification for this, as will be pointed out in the next section, concentrators are still located on the basis of

designing a star network.

There is also a problem that is inherent in any type of add approach. The problem is that if the initial choice of a concentrator location is a poor one, then the subsequent choices and thus the final topology may deviate a long way from the optimal solution. This has been found to be the case in a number of examples in connection with the warehouse location problem [27].

The fact that all terminal locations are considered as possible concentrator sites may be advantageous in that it may eliminate some biaising as to the choice of sites, which may occur if some subset of the terminal locations is chosen as a set of possible sites à priori. However, this may prove to be disadvantageous in other respects as mentioned earlier.

3.3-2 Woo and Tang Add Approach

The approach taken in this algorithm [38] is to first estimate the multidrop line costs and establish "equivalent star" link costs which will then be used in the Add algorithm [25] to determine

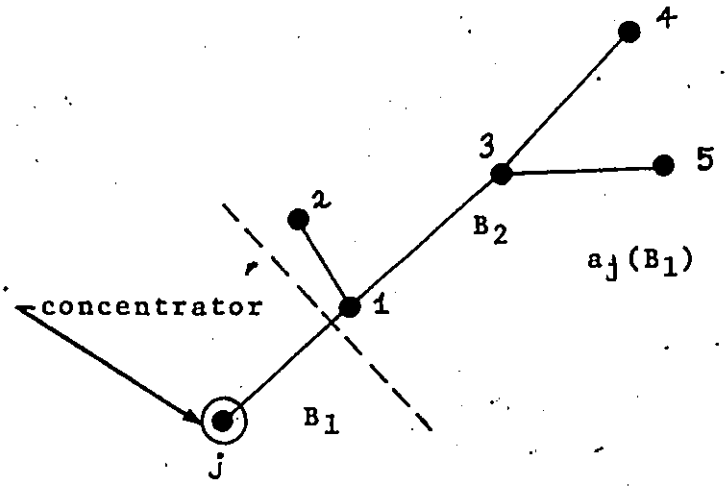
the concentrator locations. Once the concentrator locations have been determined they are used in a many-center multipoint optimization procedure to determine the final network topology. The algorithm considers a set of potential sites for concentrators.

Since the optimal multipoint network falls somewhere between a star and minimum spanning tree, MST, the authors assume that the equivalent star cost can be calculated as a linear combination of the star cost c_{ij} and the MST cost t_{ij} for the node pair (i,j), where i is a terminal and j a concentrator.

The tree costs are calculated from a minimum spanning tree and can be defined as:

$$t_{ij}=\sum_{B_k \epsilon P_i} C(B_k)/[a_j(B_k)]$$

where B is the set of branches or links in the MST, B_k is a particular branch, $a_j(B_k)$ is the set of terminals which are disconnected from the tree or concentrator j by the removal of branch B_k, $[a_j(B_k)]$ is the number of terminals disconnected, $C(B_k)$ is the cost of branch B_k and P_i is the unique path from concentrator j to terminal i. Figure 3.4 illustrates a sample calculation.



MST

$$B=\{(j,1),\ (1,2),\ (1,3),\ (3,5),\ (3,4)\}$$

$$t_{3,j}=\sum_{B_k \in P_3} C(B_k)\ /[a_j(B_k)]$$

where $P_3=\{(j,1),\ (1,3)\}=\{B_1,B_2\}$

$C(B_1)=c_{1j}$ $\quad C(B_2)=c_{1,3}$

$a_j(B_1)=\{1,2,3,4,5\}$ $\quad a_j(B_2)=\{3,4,5\}$

$[a_j(B_1)]=5$ $\quad [a_j\ (B_2)]=3$

thus

$$t_{3,j}=c_{1,j}/5+c_{1,3}/3$$

Fig. 3.4
Calculation of Tree Costs

The Add algorithm then uses these equivalent star costs in locating concentrators. Initially all candidate concentrator sites are closed and then one at a time, a concentrator is opened on the basis of maximizing the overall network cost saving. The algorithm thus involves solving a number of different terminal assignment problems for a number of different sets of concentrator sites, since only a fixed number of sites is considered for detailed analysis at each stage. The algorithm terminates when no further cost saving can be realized by opening another concentrator. A possible flow chart for the algorithm is shown in Fig. 3,5.

A network with three candidate concentrator sites A,B,C and its MST is shown in Fig. 3.6. Assume that the equivalent star costs have been calculated and the Add algorithm has chosen concentrator A as the first concentrator site. If the terminals 1 to 6 that were associated with concentrator A and are removed from further consideration, then what would the tree costs t_{iB} i=8,...,13 be, since the removal of terminals 5 and 6 disconnects these terminals, 8-13, from concentrator C. One possibility is to ignore this and continue to use the original tree costs.

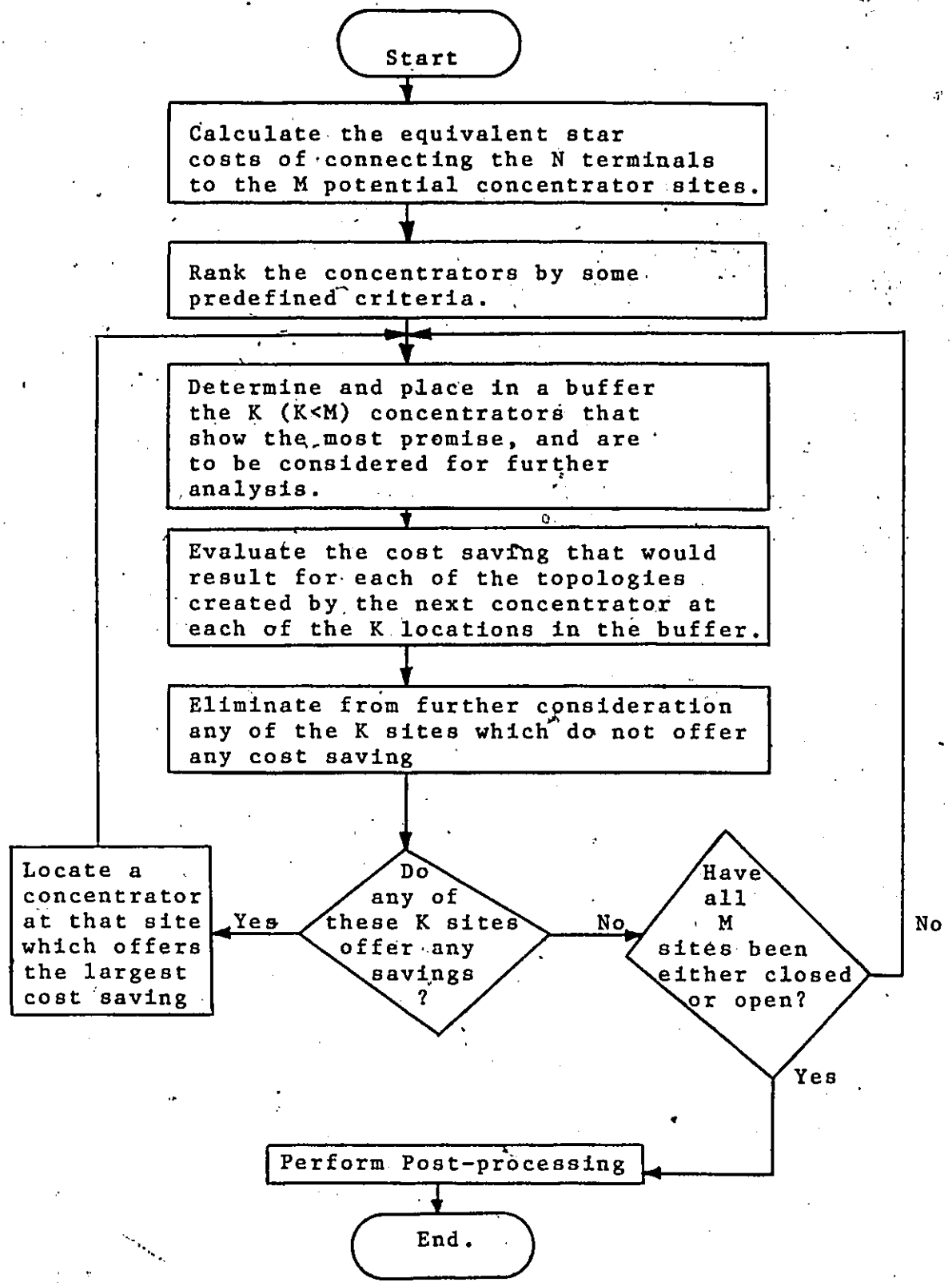


Fig. 3.5
Flow Chart for the Woo and Tang Algorithm

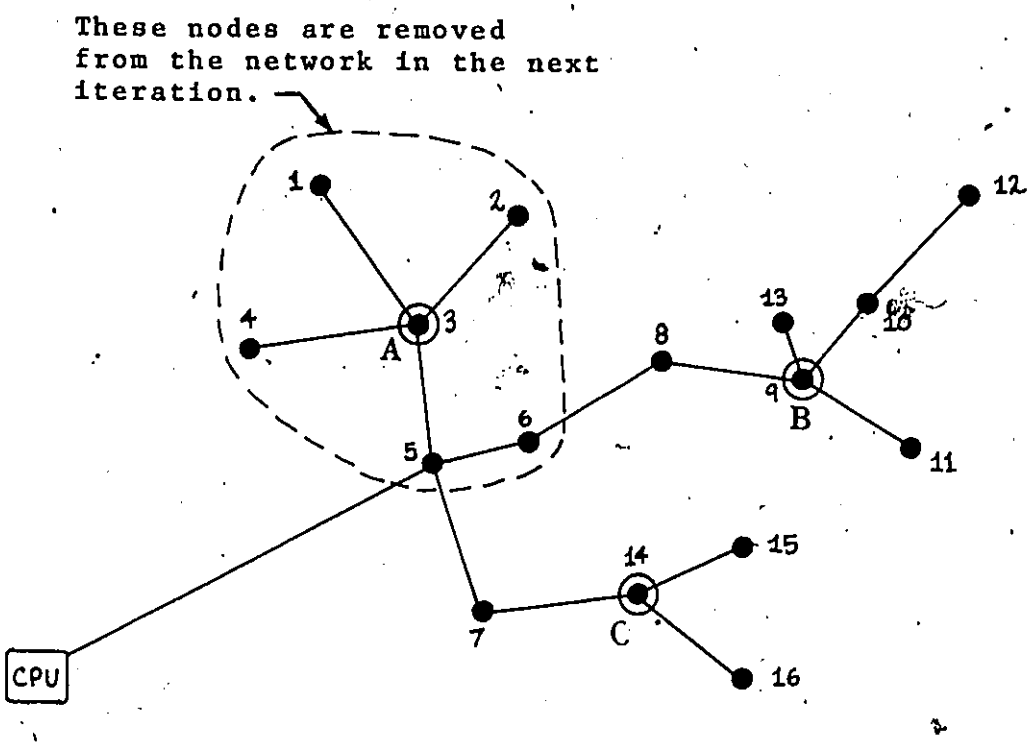


Fig. 3.6
MST and Candidate Concentrator Sites

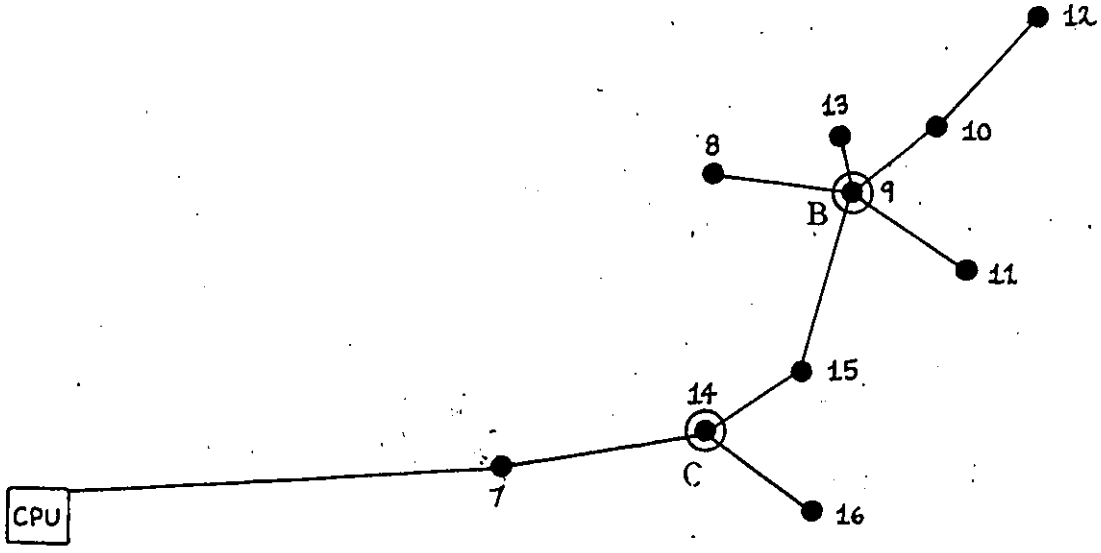


Fig. 3.7
The new MST

Another possibility would be to establish a new tree as in Fig. 3.7 and recalculate some of the tree costs. This would perhaps provide a better approximation to the actual multidrop costs. However, no mention of any detailed procedure is given in the paper. Also details as what criterion is used to rank concentrators or to evaluate the cost savings are not mentioned.

As the equivalent star costs are considered as linear combinations of the tree and star costs (e.g. $k_1 t_{ij} + k_2 c_{ij}$) and since the authors assume that in many cases the star and tree costs are linearly correlated due to constraints, they have considered estimating the costs using a single weighting factor, as $k c_{ij}$. The constants k_1, k_2, and k are complex functions of line costs, geographical distribution of the terminals and the number of terminals and traffic constraints per line, and as such must be empirically estimated or determined. This single weighting is similar to that used in the Martin-Doll algorithm. By using a single weighting factor, the approach becomes that of the standard Add approach used for point-to-point networks, with a many-center multipoint optimizing procedure used to determine the final network cost and topology. This algorithm was not used for

comparison purposes for the simple reason that not enough details were available for an accurate implementation of the algorithm.

3.3-3 Bahl and Tang's Drop Approach and Modifications

In a previous section a general description of this algorithm [33], which is a heuristic algorithm specifically designed to handle point-to-point networks, was given. A more detailed description will now be given.

The basic optimization problem is the following: given a set of terminals $\{T_1, T_2, \ldots T_N\}$ and CPU, we want to establish concentrators at a subset of possible concentrator sites $\{C_1, C_2, \ldots, C_M\}$ and assign terminals to these concentrators in such a way that the overall line and fixed costs of the network are minimized. The topology of any star network in which terminals are directly connected to a concentrator or the CPU and concentrators are directly connected to the CPU, can be completely described by an incidence matrix $[x_{ij}]$ and a concentrator state sector $[y_i]$ where:

$$x_{ij} = \begin{cases} 1 & \text{iff } T_i \text{ is connected to } C_j \\ 0 & \text{otherwise} \end{cases} \qquad \forall i = 1,2,\ldots N; \quad \forall j = 1,2,\ldots M$$

and $$y_j = \begin{cases} 1 & \text{iff } \sum x_{ij} > 0 \quad C_j \text{ is open} \\ 0 & \text{otherwise} \quad C_j \text{ is closed.} \end{cases} \qquad \forall j = 1,2,\ldots M$$

Then if we defined c_{ij} as the cost of connecting T_i to C_j and f_j as the cost of locating a concentrator at location C_j, the total cost of the network can be defined as:

$$Z = \sum_{j=1}^{M} y_j f_j + \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} c_{ij}$$

If we assume that a concentrator can handle at most E terminals and T traffic units and that each terminal can be connected to only one concentrator or the CPU, C_o, then we have the following constraints.

$$\sum_{i=1}^{N} x_{ij} \leq E \qquad \forall j$$

$$\sum_{i=1}^{N} x_{ij} t_i \leq T \qquad \forall j$$

$$\sum_{j=o}^{M} x_{ij} = 1 \qquad \forall i$$

where t_i is the number of traffic units generated at terminal T_i. Thus Z is the functional we wish to minimize subject to the above constraints.

For each potential link x_{ij} we calculate a link gain

$$g_{ij} = c_{io} - c_{ij}$$

which is the cost saving of connecting terminal T_i to C_j rather than the CPU, Co. Thus only $g_{ij} \geq 0$ are considered. The flow chart for the algorithm is given in Fig. 3.8.

To initialize the network we define for each concentrator C_j an index set of terminals I_j. To determine I_j we connect terminals to C_j in order of decreasing gain per unit traffic, g_{ij}/t_i, until the addition of another terminal violates the constraints. Initialization implies $x_{ij} = 1$ for all $i \in I_j$. If no terminal can be accommodated by any concentrator than it is connected to the CPU, i.e. $x_{io} = 1$.

For each existing link ($x_{ij} = 1$) an augmented cost $\bar{c}_{ij}$ is calculated.

$$\bar{c}_{ij} = c_{ij} + \left[\frac{t_i f_j}{\sum_{i=1}^{N} x_{ij} t_i} \right]$$

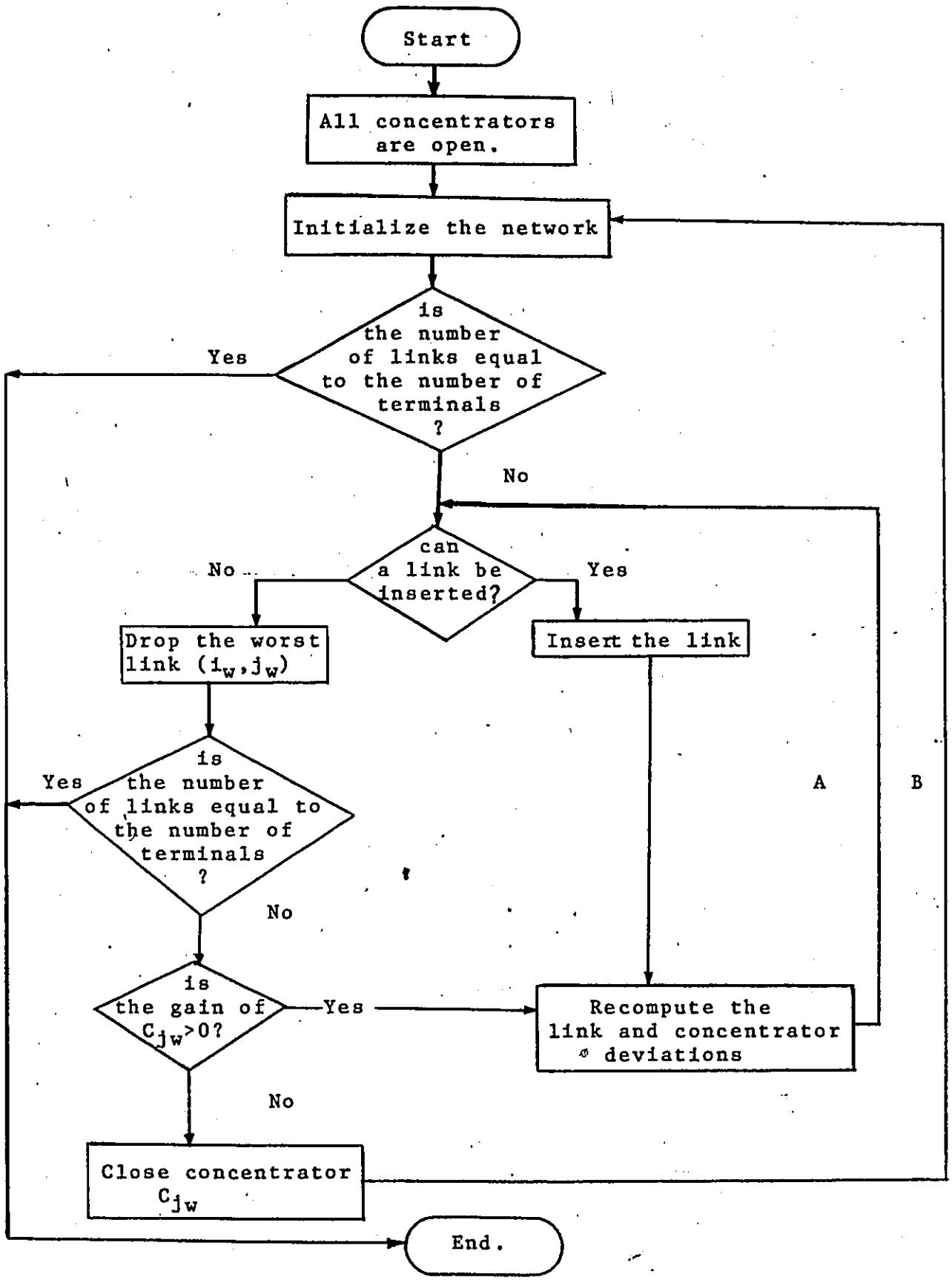


Fig. 3.8
Flow Chart for the Bahl and Tang Algorithm

c_{ij} can be thought of as the cost to t_i of using C_j if the fixed cost of C_j is shared among the terminals in proportion to the traffic generated by them. For each terminal, T_i, an average augmented cost, h_i, is also calculated:

$$h_i = \frac{\sum_{j=1}^{M} \bar{c}_{ij}}{\sum_{j=1}^{M} x_{ij}} \qquad \forall i$$

This is then used in determining the quality of the links for that terminal which is determined by the link deviation q_{ij}

$$q_{ij} = \bar{c}_{ij} - h_i$$

For each concentrator, a concentrator deviation, r_j, is obtained by summing the link deviations of the terminals in the concentrators index set.

$$r_j = \sum_{i \in I_j} q_{ij}$$

These deviations are used as the basis for dropping and adding links in the network.

In loop A of the flow chart a link is added to a concentrator if its constraints aren't full and

the addition of the link will reduce the concentrators deviation. There are two instances where this could happen:

CASE 1 If T_i is connected to the CPU and $g_{ij} \geq 0$

CASE 2 If the augmented cost of link x_{ij} is less than the average augmented cost of T_i.

$$\text{i.e.} \quad c_{ij} + \frac{t_i f_j}{\sum_{\ell=1}^{N} x_{\ell j} + t_\ell + t_i} < h_i$$

If a link cannot be inserted then the worst link (iw, jw) is dropped and then the gain of C_{jw} is checked.

If we let:

$$R = \max_j \left\{ r_j \right\}$$

$$Jw = \left\{ j \;\middle|\; r_j = R \right\}$$

$$Q = \max_{j \in Jw} \left\{ q_{ij} \;\middle|\; \sum_{j=1}^{M} x_{ij} > 1 \right\}$$

and $$(iw, jw) = \left\{ i,j \;\middle|\; j \in Jw \; ; \; q_{ij} = Q \right\}$$

then the criteria for deleting a link is that the worst link (i_w j_w) is the link with the largest link deviation among all links connected to the worst concentrator.

Each concentrator has associated with it a gain G_j.

$$G_j = \sum_{i=1}^{N} g_{ij} x_{ij} - f_j$$

The gain, G_j, is then the increase in cost that would be incurred by the network if the concentrator at C_j was closed and all of its terminals were connected to the CPU. Thus when a link is dropped the gain of the worst concentrator j_w is decreased. If the concentrator gain then becomes negative C_{jw} is closed and all of its links are dropped and removed from further consideration.

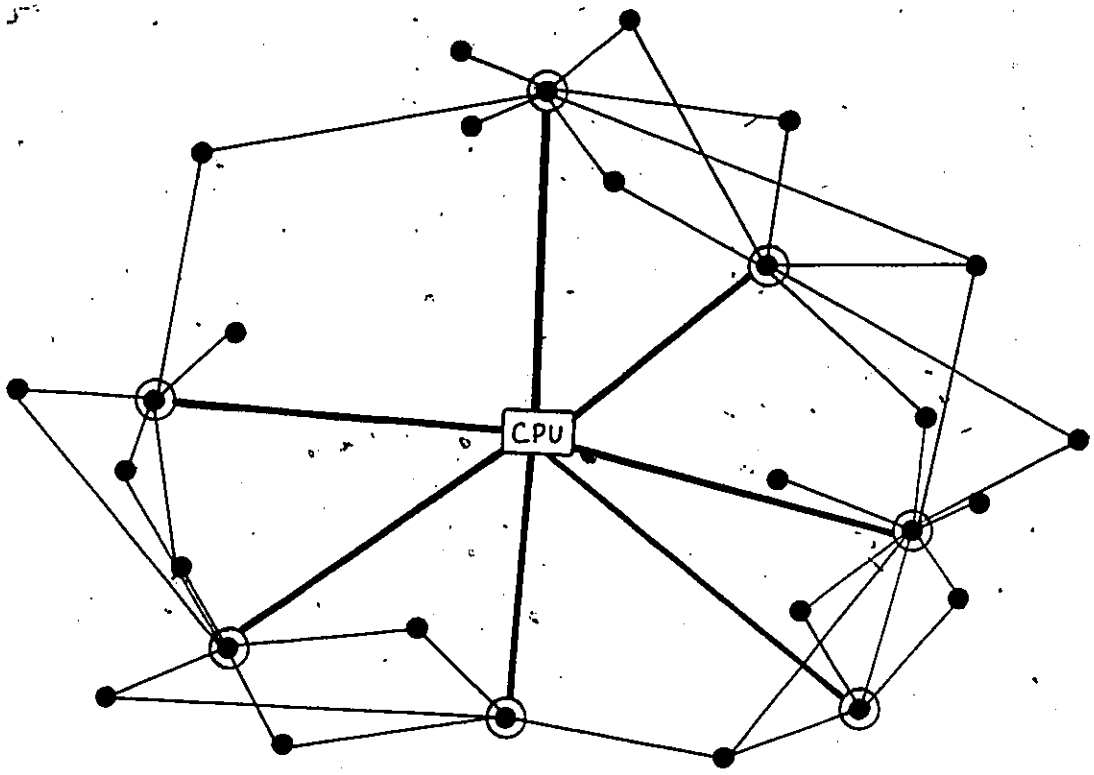
In summary then, the algorithm starts with all possible concentrators open. As each concentrator is assigned a set of terminals, a terminal may be connected to more than one concentrator thus creating link and concentrator deviations. Each time loop B is executed a concentrator and a number of links are dropped from the network and the network is then reinitialized for the remaining concentrators. Each time the inner loop A is executed, the network is modified by either the deletion or insertion of a single link in such a manner that the deviations are decreased. The algorithm terminates when the number of links equals the number

of terminals, thus each terminal is connected to one and only one concentrator. Figure 3.9 shows what a network might look like when it is first initialized and when the algorithm terminates.

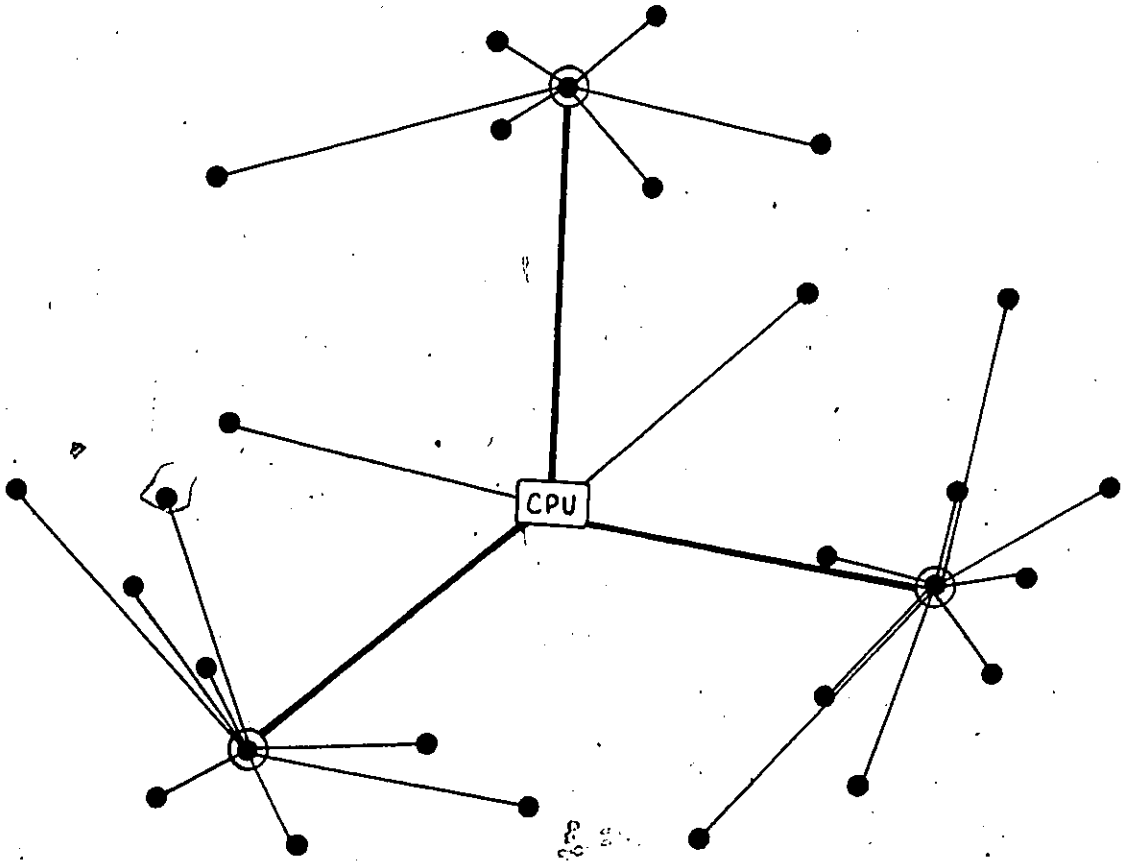
Some pre-and pro-processing techniques can be used to speed up the algorithm, reduce storage, and incrementally improve the solution. The only pre-processing we have used in the implementation of the algorithm is the permanent elimination of concentrators which have a negative gain when the terminals in their initial index set, I_j, are connected to them. Also in order to speed up the algorithm, we have not totally reinitialized the network each time a concentrator is dropped. Rather than doing that, which involves a number of timely calculations, we have only considered connecting terminals to the CPU whenever they have been stranded (i.e. $\sum_j x_{ij} = 0 \quad \forall j$) due to a concentrator being dropped. If these terminals could have been economically connected to some concentrator then they more than likely will be when some link is added.

In using this algorithm to design multidrop networks, we have considered the following approaches. Since one of the constraints involves the maximum number of terminals and thus point-to-point lines, the concentrator can handle, and the constraint for a multipoint network is more oriented towards the number of multidrop lines, then some modification as to how terminals are assigned to concentrators is necessary. Initially we allow up to E+1 terminals to be assigned to a concentrator. Thus initially some concentrators will have E+1 terminals and others will have less. Then when the algorithm considers adding links, those concentrators with more than E terminals may have more terminals assigned to them until the concentrators capacity is filled. Those concentrators with less than E terminals may then have only up to E terminals assigned to them. Initially, then many more links are added than dropped. This does tend to be slightly biased against isolated concentrators and to some extent slow down the algorithm but is a more judicious manner in allocating links.

Once the algorithm terminates, rather than optimize each concentrators final subnetwork, we have considered reintitializing the network and then optimizing each concentrator's subnetwork. This allows us to determine if any of the concentrators have become uneconomical based on the new terminal assignments. This was also done for comparison purposes since the same many-center multipoint optimization procedure was used in all the algorithms that were compared.



(a)
Initial Network



(b)
Final Network

Fig. 3.9
Initial and Final Topologies of the Drop Algorithm

CHAPTER 4

THE NEWCLUST ALGORITHM

4.1 Introduction

The new heuristic algorithm which we now propose for the design of two-level multidrop centralized networks has been labelled NEWCLUST. The algorithm uses a drop approach in determining the number and location of concentrators. The multipoint optimization procedure used was the Esau-Williams algorithm, for reasons stated earlier. However, any multipoint optimization procedure may be used instead of the Esau-Williams one, and in some cases another type may be more desirable, as will be pointed out later. This multipoint optimization algorithm may be used a number of times during the execution of the main algorithm in attempts to reduce the overall network cost.

The NEWCLUST algorithm is compared to the Martin-Doll algorithm and the modified Bahl and Tang algorithm. In the Bahl and Tang algorithm, all terminal locations may be considered as potential concentrator sites, however, for large networks, this may result in an excessive amount of time taken to obtain a solution. Some pre-processing and post-processing techniques are suggested to reduce the

number of potential-concentrator sites. This might involve, in the pre-processing stage, the placing of a grid on the network and choosing representative sites from each of the grid sections. In the post-processing stage considering each grid containing a concentrator in turn, the other locations in the grid section are then considered as possible concentrator sites.

In the Woo and Tang algorithm only a subset of the terminal locations are considered as candidate concentrator sites. However, there is no particular rationale described for the choice of this subset. It is usually assumed that the choice of the initial set of candidate concentrator sites is left to the designer's discretion. If this is the case then a logical procedure is to try a number of different subsets, which may vary in size, and choose the one whose resultant topology has the lowest cost. The choice of these subsets may be either random or partially determined from the resultant topology of previous subsets [6]. Thus one can see that if no structured procedure is used to determine the initial set of candidate concentrator sites, which is a subset in which the optimal set resides, then there is always the possibility of omitting a number of good locations.

Also, if an iterative or recursive procedure is used to determine the final set then the time may become excessive unless the design algorithm is fairly fast. It should also be kept in mind that these other algorithms use estimated costs in determining concentrator locations. Thus the "goodness" of the choice of locations is dependant to some degree on the "goodness" of the approximated costs. Another factor that must be taken into consideration is how computer storage and time of an algorithm are affected by increasing network size. It may be that storage and time increase nonlinearly, perhaps even exponentially, with respect to the number of terminals of the network.

We have tried to develop NEWCLUST such that these factors are reasonably accounted for. That is we have used a clustering-density heuristic to determine the initial candidate concentrator set. Then by using the drop approach we have attempted to home in on the best subset of these sites. Also by using the drop approach and a fairly large set, large compared to the number in the final solution, of concentrator sites, we can easily partition the network; independantly optimize small sections of the network, and use actual costs in determining the concentrator sites. Thus by

partitioning the network into smaller sections, subnetworks around concentrators, and again partitioning the nodes of these subnetworks into clusters of nodes called "super nodes", we have attempted to reduce the cardinality of the problem and thus the storage; while at the same time reducing the search and thus hopefully the time.

4.2 Determining the Initial Concentrator Sites

The reason for developing a procedure for determining an initial set of candidate concentrator sites was first to reduce the search by considering only a selective subset of the N terminal locations and second to determine a set that included all reasonably good locations.

Since the efficient use of concentration implies the clustering of terminals in a geographic or topological sense, it seemed intuitively desirable to devise a method of determing areas of high density within the network and then to assume that terminals within these areas would be good choices for concentrator sites. What we are trying to do in a topological sense is to eliminate terminals on peripheries of clusters of terminals, rather than just those on the periphery of the

network or close to the CPU. The procedure we have used for this purpose is as follows:

STEP 1: For each terminal in the network determine and list its K nearest neighbours (N.N.). K is a design parameter supplied by the user. The terminal is also included in its K-N.N. list.

STEP 2: For each terminal count the number of times that it occurs in the K.-N.N. lists.

STEP 3: Each terminal now has an integer associated with it, f_i, that represents its frequency of occurance in the K-N.N. lists. If F is the maximum frequency of occurance over all the terminals, then the frequencies will be $f_i=\ell$ $\ell=1,2,\ldots,F$. All terminals with the same frequency $f_i=\ell$ are stored in a list $S\ell$. The number of terminals in a list $S\ell$ is $x\ell$. Thus the sum of all these $x\ell$'s is equal to the number of terminals, N, of the network.

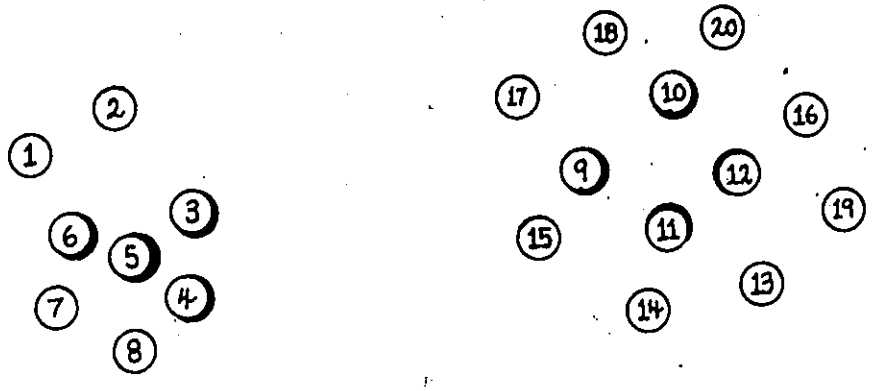
STEP 4: The weighted mean, W.M., of this discrete distribution and its integral part plus one, KM, are calculated.

STEP 5: The concentrator set, C, is initialized by the terminals in the set with the highest frequency S_F. Subsequent terminals are added to C by adding terminals from the set with the next lower frequency i.e. $S_{F-1}, S_{F-2}, \ldots$ etc., until either the addition of the terminals in the next lower set brings the total number of concentrator locations in C above $N/2$ or the sets' frequency is less than KM. This assures that no more than $N/2$ terminal locations are considered as concentrator sites. When either of the two above conditions are met the algorithm terminates.

Figure 4.1(a) illustrates the procedure for a small set of terminals and Fig. 4.1(b) shows the discrete distribution for them.

4.3 Terminal Allocation and Initialization

Once the concentrator sites have been selected terminals must be assigned to concentrators in some economical manner. One could proceed by assigning terminals to concentrators solely on the basis of cost or distance, considering the closest one first,



Number of Terminals = N = 20
Number of K - N.N. = 3

Terminal	K-NN.	f_i
1	(1,2,6,5)	3
2	(2,1,3,5)	3
3	(3,4,2,5)	4
4	(4,3,5,8)	5
5	(5,6,4,3)	8
6	(6,1,7,5)	4
7	(7,6,4,5)	3
8	(8,7,4,5)	2
9	(9,11,17,10)	4
10	(10,9,11,12)	8
11	(11,9,10,12)	7
12	(12,19,11,10)	5
13	(13,14,11,12)	2
14	(14,13,15,11)	3
15	(15,14,9,11)	3
16	(16,19,20,10)	3
17	(17,9,18,15)	3
18	(18,10,20,17)	3
19	(19,16,12,10)	2
20	(20,18,16,10)	3

ℓ	$x\ell$	$S\ell$
1	0	$\emptyset$
2	3	(8,13,19)
3	9	(1,2,7,14,15,16, 17,18,20)
4	3	(3,6,9)
5	2	(4,12)
6	0	$\emptyset$
7	1	(11)
8	2	(5,10)

Weighted Mean $= \frac{1}{N} \sum_{\ell=1}^{8} \ell x_\ell$

$KM = [W.M] + 1 = 4$

Concentrator Sites
C = (5,10,11, 4,12,3,6,9)

(a)

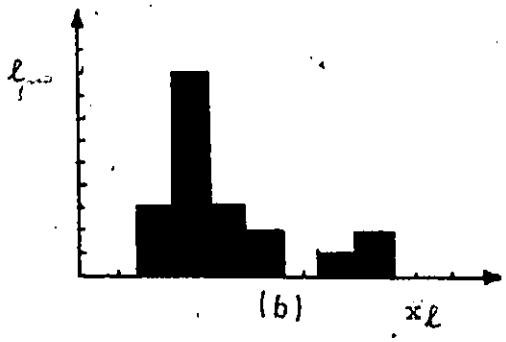
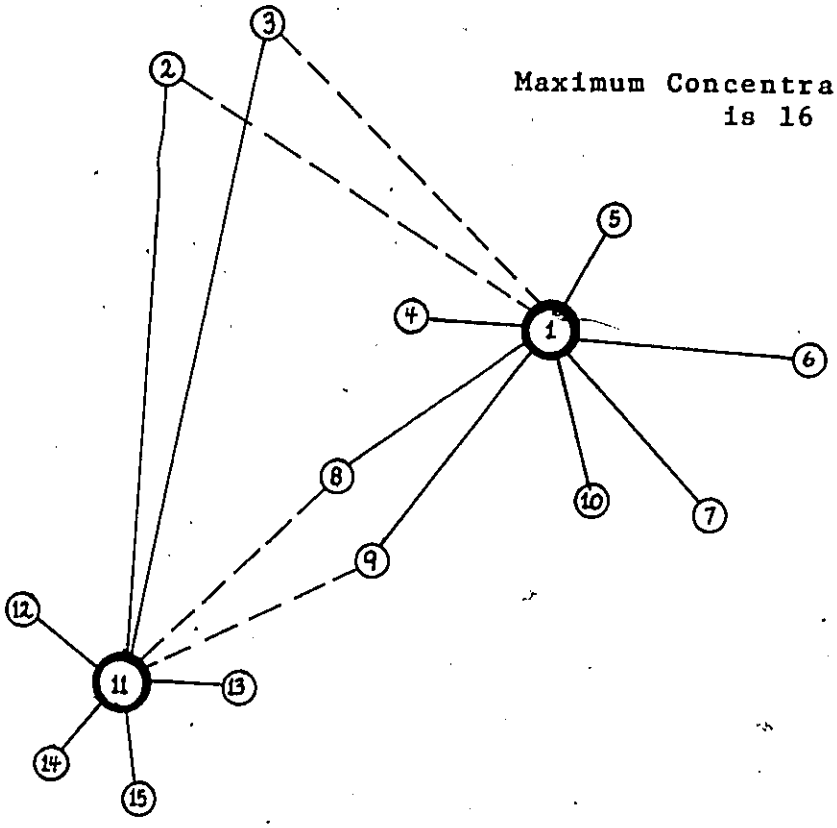


Fig. 4.1
Determining Candidate Concentrator Sites

until the concentrator capacities are filled. However, this may result in some terminals being stranded in the sense that they cannot be assigned to their closest concentrator due to capacity restrictions and are thus forced to be assigned to another concentrator which in many cases may be an unnecessarily uneconomical choice. Figure 4.2 illustrates this situation. Terminals 2 and 3 cannot be assigned to the concentrator at 1 because they would violate the capacity constraint of the concentrator (eg. maximum capacity is 16 units). They must therefore be assigned to the concentrator at 11. However, if terminals 8 and 9 had been assigned to the concentrator at 11, then 2 and 3 could be assigned to 1 resulting in a more economical terminal assignment.

In attempting to overcome the problem of stranded terminals we have first assigned terminals to their closest concentrator on the basis of the minimum cost assignment. Then if the total traffic of the terminals assigned to a concentrator exceeds the maximum capacity of the concentrator the terminal assignments to the concentrator are modified to satisfy the constraint. A number of modification methods were investigated with the result that no particular method produced totally satisfactory results



Concentrator Traffic

$TC_{11} = 9+t_5+t_6=15$ TC_1 =15 original assignment

$TC_{11} = 14$ TC_1 =16 modified assignment

Terminal i	Traffic t_i
1	2
2	2
3	4
4	3
5	1
6	3
7	3
8	2
9	3
10	2
11	2
12	1
13	1
14	3
15	2

Cost Matrix

	1	11
2	10.	15.
3	9.	16.
.	.	.
.	.	.
.	.	.
.	.	.
8	5.	6.
9	7.	7.5
.	.	.
.	.	.
.	.	.

Fig. 4.2
Terminal Assignment

over a range of different topologies. The one that appeared most promising was adopted. That procedure was based on defining a tradeoff for each terminal in the concentrator subnet. Terminals are dropped, starting with the one with the smallest tradeoff, until the concentrators capacity constraint is satisfied. The tradeoff was defined as:

$$t_i = d_{ij}^* - d_{ij}$$

where d_{ij} is the cost of connecting terminal i to concentrator j which is the concentrator it is assigned to and d_{ij}^* is the cost of connecting terminal i to its next feasible concentrator j*. Since $d_{ij}^* \geq d_{ij}$ in all cases, then choosing the minimum t_i at each stage will allow the minimum additional expected cost to be incurred.

This procedure is performed on all concentrators that have their capacity constraint violated. The terminals that have been dropped are stored and once all the concentrator capacity constraints have been satisfied they are reassigned to available concentrators or the CPU. If at the end of this assignment stage a concentrator has not been assigned any terminals, other than the one at the site, then it is dropped from the

network and its terminal is assigned to another concentrator.

Once the concentrator subnetworks have been initialized by this feasible terminal assignment, then each subnetwork is optimized by the Esau-Williams algorithm. The initial optimization was done in this serial manner rather than in a parallel manner (i.e. optimizing the network as a whole rather than by parts) in order to reduce the amount of storage required. Since the initial subnetworks are usually small, the optimization can be done quickly.

After each concentrator subnetwork is optimized into a tree network, the terminals of the tree are clustered into super nodes. This is done by defining the set of nodes on each segment (i.e. a multidrop or point-to-point line) of the tree as a super node. The terminal at the concentrator site is associated with that super node which is closest to it and can accommodate it. If no super node of the subnetwork can accommodate it, it is then considered a super node by itself.

The super nodes thus formed have a number of parameters associated with them, which are: (in the notation used in the programs).

NN(I,1) = the number of nodes in super node I

NN(I,2) = the total traffic of the nodes in super node I.

NN(I,3) = the concentrator to which super node I is presently assigned.

NN(I,4) = the node in super node I that is directly connected to the concentrator.

NON(I) = the set of nodes in super node I.

Also associated with each super nodes are a number of variables, which are:

TNN(I,1)* = the next closest feasible super node to super node I.

TNN(I,2) = the node in super node I from which the connection to * will be made.

TNN(I,3) = the node in * to which the connection from I is to be made.

TNN(I,4) = the concentrator * is assigned to.

TNN(I,5) = the concentrator of * if the terminal at the concentrator site is the closest node; 0 otherwise.

TNN1(I)= the reallocation cost for super node I (i.e. the connection cost between I and *).

At this stage each concentrator also has a number of variable associated with it which are:

NCONC(J) = the number of super nodes in the subnetwork of concentrator J.

TCONC(J) = the total traffic of all the super nodes of concentrator J.

CONC(J) = the set of super nodes assigned to concentrator J.

CST(J) = the connection cost for concentrator J, i.e. the sum of link costs connecting super nodes to J (excluding the one with the concentrators terminal).

RECOST(J)= the reallocation cost for concentrator J, i.e. the sum of the reallocation costs of its super nodes.

Once all of these variable have been determined, then the gain of each concentrator is calculated as:

$$GAIN(J) = CST(J) + F(J) - RECOST(J)$$

where F(J) is the cost of locating concentrator J.

Thus if the concentrator gain is positive then it is economical to drop it and reassign its super nodes to their respective next closest feasible neighbours which have been uniquely determined by the TNN(I,1) and TNN(I,3) variables.

If the constraints are NMAX, the maximum number of nodes per multidrop line, TMAX, the capacity of the low-speed lines, and MAXCON the capacity of a concentrator then we have the constraints:

$$NN(I,1) \leq NMAX, \forall\ I = 1,2,\ldots LL$$

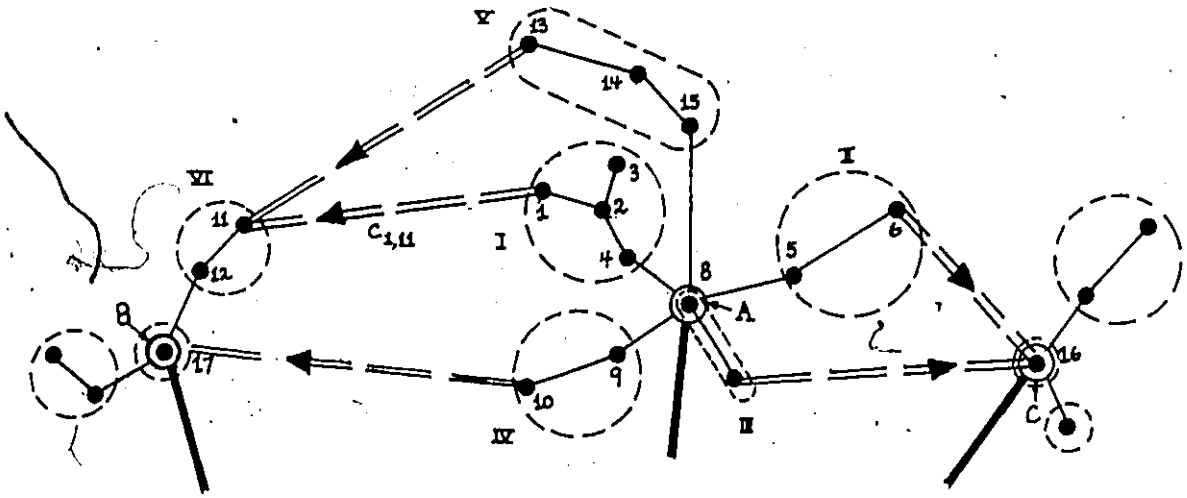
$$NN(I,2) \leq TMAX, \forall\ I$$

$$TCONC(J) \leq MAXCON, \forall\ J=1,2,\ldots M$$

where LL is the number of super nodes and M is the number of concentrators. Figure 4.4 illustrates the calculation of concentrator gain.

4.4 Locating Concentrators and Super Node Assignment

The basic assumption used in calculating the concentrator gains is that the cluster of nodes in each super node will never be broken up. This assumption is based upon using a fairly large number of concentrators initially such that each concentratoronly has a small number of terminals assigned to it. This will then



Concentrator A's Variables

$NCONC(A)=4$

$TCONC(A)=\sum_{i=I}^{V} NN(i,2)$

$CST(A)=\sum_{i} c_{i8}$ $i=4,5,9,16$

$RECOST(A)=\sum_{i=I}^{V} TNN_1(i)$

$CONC(A)=(I,II,III,IV,V)$

Super Node I's

Parameters	Variables
$NN(I,1)=4$	$TNN(I,1=VI$
$NN(I,2)=7$	$TNN(I,2)=1$
$NN(I,3)=A$	$TNN(I,3)=11$
$NN(I,4)=4$	$TNN(I,4)=B$
$NON(J)=(1,2,3,4)$	$TNN(I,5)=0$
	$TNN1(I)=C_{1,11}$

Gain of concentrator A

$$GAIN(A)= F(A)+CST(A) - RECOST(A)$$
$$= F(A)+(C_{9,8}+C_{4,8}+C_{5,8}+C_{16,8})+(C_{10,17}+C_{1,11}+C_{13,11}+C_{6,16}+C_{7,16}).$$

Traffic Vector

Node	Traffic
1	3
2	1
3	2
4	1
.	.
.	.
.	.
.	.

Fig. 4.4

Super Nodes and Concentrators

result in terminals being connected to their nearest neighbours and thus forming small clusters of nodes. As it is assumed that these super nodes are never broken up, although they may be added to, then the cost of these super nodes (i.e. the link costs connecting the nodes of the super node) will always be incurred by the network independant of which concentrator they are assigned to. The only costs that are variable in the network are then the costs of connecting super nodes to concentrators, and the costs of concentrators because they may be removed from the network. If each super node is thought of as a single entity (i.e. as just one large node or terminal) then a gain can be associated with each super node i which is connected to a concentrator j as:

$$g_{ij} = \bar{c}_{ij} - \bar{c}_{ij^*} \qquad i \in CONC(j)$$

where $\bar{c}_{ij}$ is the least costly connection between super node i and the concentrator j to which it is assigned and the cost $\bar{c}_{ij^*}$ is the least costly connection between super node i and its next closest feasible neighbour j*. This is similar to the gains used in the other algorithms but is more general in that j* may be any other concentrator rather than just the CPU,

as is the case in the other algorithms. The gain of concentrator j is then, again the notation of the other algorithms:

$$G_j = \sum_{i \in I_j} g_{ij} - f_j$$

where f_j is the cost of concentrator j and I_j is analogous to CONC(j).

Thus locating concentrators involves determining which concentrator has the largest gain, providing its super nodes reassignments are feasible, and dropping it from the network and reassigning its super nodes to their respective next closest feasible neighbours, which had previously been determined. Thus multiple link exchanges occur each time a concentrator is dropped. By choosing that concentrator which has the largest gain we are attempting to maximize the cost saving to the network, as the gain of a concentrator is the actual line and concentrator costs that will be deleted from the overall cost of the network. Concentrators are dropped one at a time from the network in this manner until it is no longer economical to drop another concentrator (i.e. the largest gain is negative).

As mentioned above, a concentrator is only dropped if its super nodes reassignments are feasible. Thus before a concentrator is dropped this is checked. The reasons for checking this feasibility are best illustrated by the example in Fig. 4.4. Suppose concentrator A is being checked for feasibility as it was the one with the largest gain. If super nodes I, IV and V are added to the subset of concentrator B it may be that the sum total of their traffic when added to that already at B may violate the capacity of the concentrator, MAXCON, at B. It may also be the case where super node VI is the next closest feasible neighbour of both super node V and I. If both V and I are merged with VI it may be that this merging may violate the constraints of the maximum number of terminals per multidrop line and/or the maximum capacity of the low-speed line. If either of these conditions are met, then it is necessary to modify the super node reassignments. Whenever a super node has its assignment changed due to this, it is necessary to recalculate its variables TNN(I,1),..., TNN(I,5), and its reallocation cost TNN1(I) and thus the gain of the concentrator it is connected to.

The feasibility check and super node reassignment is done in the following manner for a concentrator j.

a) The potential traffic that has been assigned to each of the other concentrators, from concentrator j's super nodes, is determined. Also the total potential traffic and number of terminals that each of concentrator j's super nodes may contribute to their next closest super node is determined.

b) If the total traffic that may be (potentially) assigned to any of these neighbouring concentrators, plus the traffic already there, exceeds the concentrator capacity constraint, MAXCON, then the assignment of super nodes to that concentrator is modified by:

i) Dropping that super node that is farthest away (i.e. the one with the most costly connection);

ii) Recalculating the super nodes next potential concentrator and super node;

iii) Checking if the concentrator's capacity is still violated. If it is, then we return to i). If it is not, then we move on and check the next neighbouring concentrator.

Once all the super node assignments to the concentrators are feasible, then we must check the super node to super node assignments.

c) The super node assignments are modified, if necessary, in the same way as in b). Each super node that has at least one of concentrator j's super nodes potentially assigned to it is checked. All of these super nodes are checked in turn to see if either of the constraints, NMAX or TMAX, are violated.

If any modification of super node assignments is made then the concentrator is not dropped. The concentrator with the largest gain is again found and the process is repeated unless it happens to be the same concentrator that was just checked in which case it is immediately dropped.

As noted earlier, a super node may have as its next closest feasible neighbour another super node or a

concentrator. If the concentrator of a super node is dropped and the super nodes' next closest feasible neighbour is another super node, then it is merged with that super node by the least costly link, which has previously been defined by TNN(I,2) and TNN(I,3). Whenever a super node is merged with another super node, it becomes inactive and its parameters of traffic, NN(I,2) and number of nodes, NN(I,1) and its nodes, NON(I), are added to the super node it was merged to. By making a super node inactive we simply mean that it is no longer considered to exist and is removed from further consideration.

If a super nodes concentrator is dropped and its next closest feasible neighbour happens to be a concentrator, rather than another super node, then it is connected directly to the concentrator. However, whenever this occurs there is the possibility of further reducing the network cost by reorganizing the super nodes of the concentrators subnetwork. That is although it may not be feasible and/or economical to merge the super node from another concentrator it may be possible to reoptimize the nodes of the concentrator subnetwork into other super nodes which are less costly. This is particularly true in the case where two or more super nodes

have been added to a concentrator subnetwork at the same time. For example, in Fig. 4.4 if super nodes V, I, and IV are all added to the subnetwork of concentrator B, then it may be possible to merge I to IV rather than I to VI and thus reduce the cost. This is attempted every time a super node is added directly to a concentrator by applying the Esau-Williams algorithm to the nodes of the super nodes of the concentrator subnetwork.

In summary, the following steps describe the NEWCLUST algorithm. The names that appear in brackets after each step are the subroutines that were used in the program to implement them.

STEP 1: Determine the M concentrator sites using the clustering algorithm.

STEP 2: Initialize the network variables and calculate the concentrator costs, F(j), j=1,2,..., M, and the terminal to concentrator costs DD(i,j) i=1,2,...N, j=1,2,...M.

STEP 3: Initialize the concentrator subnetworks by:

a) Assigning each terminal to its closest concentrator (AMIN).

b) Dropping terminals from concentrator subnetworks that have their concentrator capacity constraint violated. Continue this until all subnetworks satisfy the capacity constraint (CFUL).

c) Reassign terminals that were dropped in b) (REAR, AAMIN, FMI).

STEP 4: Perform preprocessing by dropping those concentrators that have not had any terminals, other than the one at the concentrator location, assigned to them. Reassign their terminals (AAMIN).

STEP 5: i) Apply the multipoint optimization algorithm to each subnetwork (SAHC, RECOM).

ii) Cluster the terminals of each concentrators subnetwork into a set of super nodes, CONC(j), and calculate each concentrators variables and connection cost, CST(j), (DECOM).

STEP 6: Determine each super nodes variables, TNN(i,1)...TNN(i,5), and reallocation cost TNN1(i), i = 1,2,..., L, where L is the number of super nodes that have been formed (CLOSET).

STEP 7: Calculate the gain of each concentrator as:

$$GAIN(j)=CST(j)+ F(j)-\sum_{i\in CONC(j)} TNN1(i) \quad j = 1,2,3,...M.$$

STEP 8: The concentrator with the largest gain GAIN(j*) is found (MAX).

$$j^*= \left\{ J \;\middle|\; GAIN(j^*)= \max_{j}\left\{GAIN(j^*)\right\};\ j=2,3,...,M \right\}$$

If $GAIN(j^*) \leq 0$ then GO TO STEP 15; else GO TO STEP 9.

STEP 9: If $j^* = j_{last}$ the last concentrator that had been checked for feasibility, then GO TO STEP 10; else check the feasibility of j*'s super node assignments (FEAS). If any change occurs GO TO STEP 8; else GO TO STEP 10.

STEP 10: Close concentrator j*, and decrease the network cost by GAIN(j*).

STEP 11: Assign j*'s super nodes to their respective closest feasible neighbours, $TNN(i,1) \in CONC(j^*)$ and update the parameters of these neighbours, and the super nodes parameters.

STEP 12: Recalculate the variables, TNN(i,1),...,TNN(i,5) and reallocation costs, TNN1(i) of these neighbours. Also recalculate the gains of their repsective concentrators.

STEP 13: If a concentrator has had at least one of j*'s super nodes directly assigned to it, then attempt to reoptimize its subnetwork in an attempt to reduce the network cost (MINSA).

STEP 14: The variables of those super nodes, excluding those of j*, who had j* as their next closest concentrator, i.e. TNN(i,4)=j* and the gains of their associated concentrators are recalculated. GO TO STEP 8.

STEP 15: Record the resultant topology and network cost.

STEP 16: Initialize the network cost to zero and repeat steps 3,4,5(i), and record that topology and network cost.

STEP 17: Choose that topology which has the lower cost as the final topology and terminate.

A general flow chart of the algorithm is given in Fig. 4.5, with the corresponding steps beside the appropriate boxes. Figure 4.6 illustrates some of the transformations a hypothetical network might go through when NEWCLUST is applied.

4.5 Evaluation of NEWCLUST

The NEWCLUST algorithm was compared with the Martin-Doll algorithm and the modified Bahl and Tang algorithm. Twenty networks were used for the comparison study. All networks and traffic were randomly generated (Power Residue Method). The size of the networks ranged from 40 to 120 nodes, in increments of 20, with four networks being generated for each size. The CPU in each of the networks was randomly located. The traffic units generated at each terminal was assumed to be proportional to the number of users and generally ranged from 1 to 8. The constraints of the maximum number of terminals per multidrop line, NMAX, the maximum capacity of a low-speed line, TMAX, and the maximum capacity of a concentrator, MAXCON, were varied from network to network and generally ranged from 4-10, 6-16, and 24-64 respectively. The same nonlinear costs functions, which were scaled down versions of Martin's [31], were used for all networks.

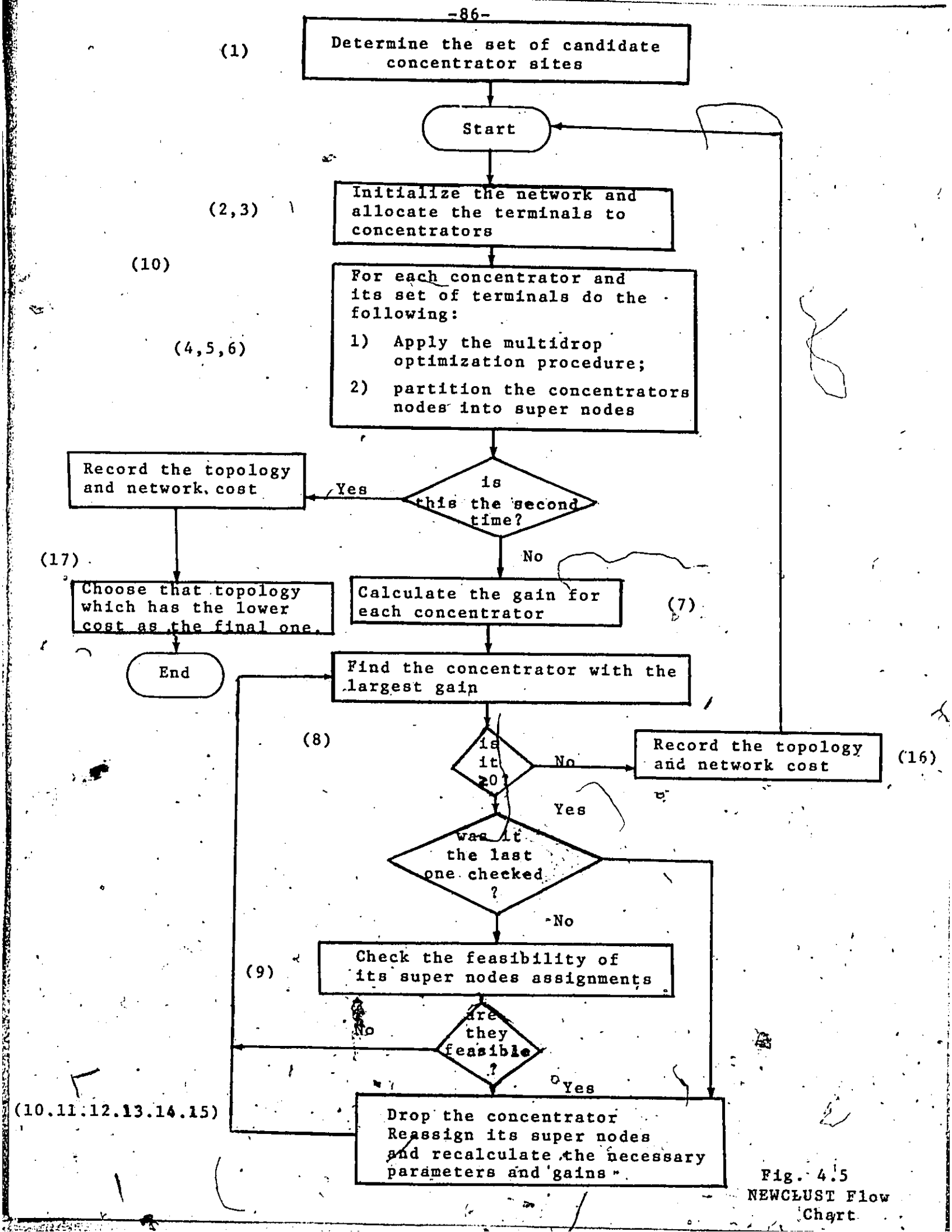
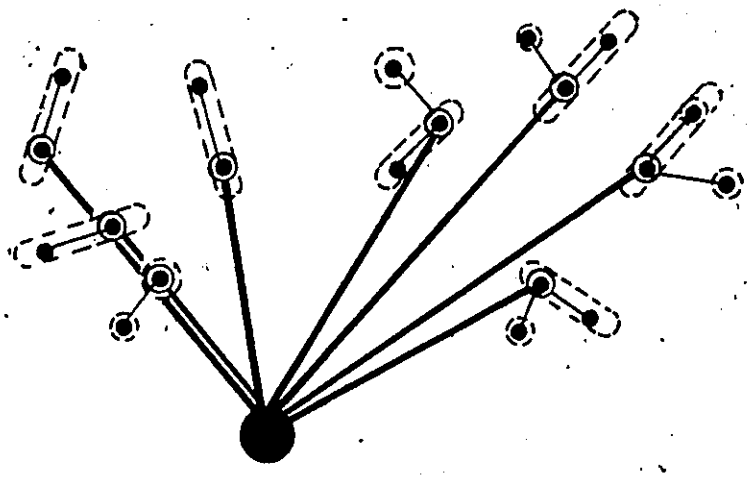
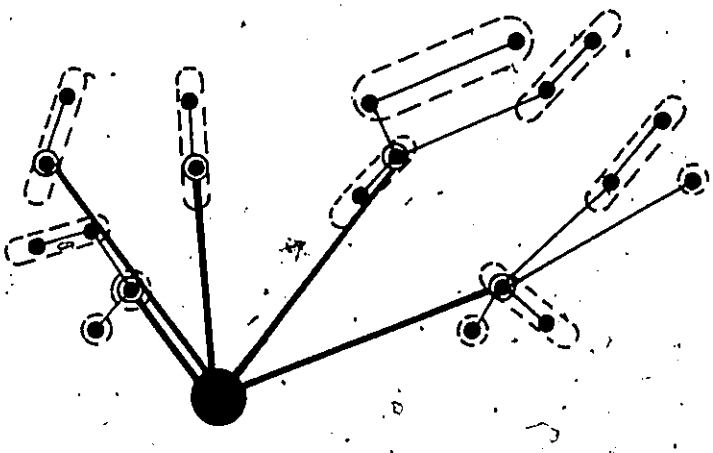


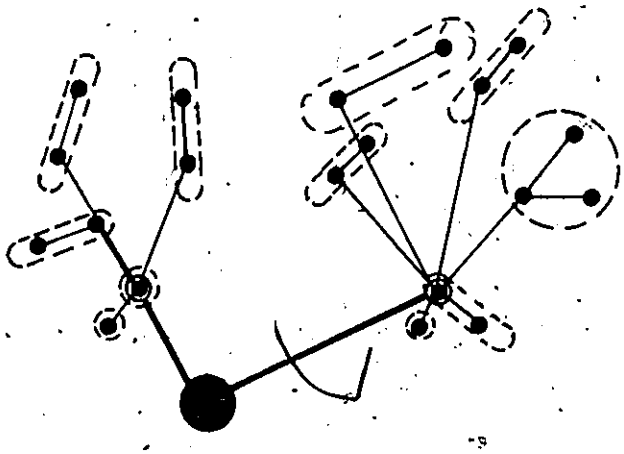
Fig. 4.5
NEWCLUST Flow Chart



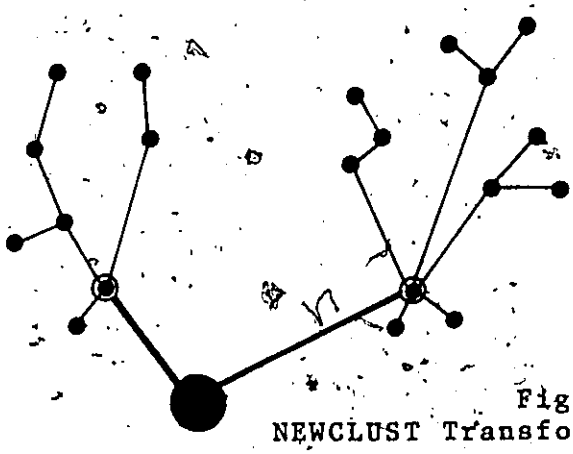
Topology after terminal allocation and super node formation.



Concentrators have been dropped and some super nodes merged.



The resulant topology when it is no longer economical to drop another concentrator.



Topology resulting from the reinitialization of the network with the two concentrators from the above network.

Fig. 4.6
NEWCLUST Transformation of a Network

Steps 2-5i) of NEWCLUST constituted the many-center multipoint optimization procedure that was used for all of the algorithms. The same set of candidate concentrator sites, determined by the clustering algorithm, were used to initialize both NEWCLUST and the Bahl and Tang algorithms. Since we were interested in comparing the times of these two algorithms, the clustering algorithm was run separately from the main algorithms. As the Martin-Doll algorithm did not use any set of candidate concentrator sites we were interested in comparing the times of the overall procedures and thus the clustering algorithm time was added to the time of the basic NEWCLUST algorithm.

Table 1 shows the time and cost comparisons. The numbers for each size of network were averaged over the four networks of that size. From this table we can see that NEWCLUST showed fairly good improvements in both time and cost over the other two algorithms. The overall average cost improvement over the Martin-Doll algorithm was 3.3% and 9.11% over the modified Bahl and Tang algorithm. The most noted improvement is in time as can be seen from Fig. 4.7. It appears that the time for the modified Bahl and Tang and the Martin-Doll algorithms increase almost

Network Size N	NEWCLUST		Average Cost and Time Improvements Over			
			Martin-Doll		Modified Bahl and Tang	
	$T_{Avg.}$	$T_{Avg.}$*	% Cost	R*	% Cost	R
40	11.83	14.43	6.01	2.42	8.12	1.13
60	17.39	23.27	3.39	4.66	7.05	2.79
80	25.99	36.38	2.42	8.24	8.00	9.16
100	34.40	52.06	0.71	9.27	11.23	8.10
120	40.45	65.62	3.63	13.15	11.15	11.81

TABLE 1

$T_{Avg.}$ =CPU sec. of NEWCLUST averaged over 4 nets

$T_{Avg.}$ =CPU sec. NEWCLUST + CPU sec. clustering algorithm, averaged over 4 nets.

R* =CPU sec. Martin-Doll/Tang*.

R =CPU sec. Bahl and Tang/$T_{Avg.}$

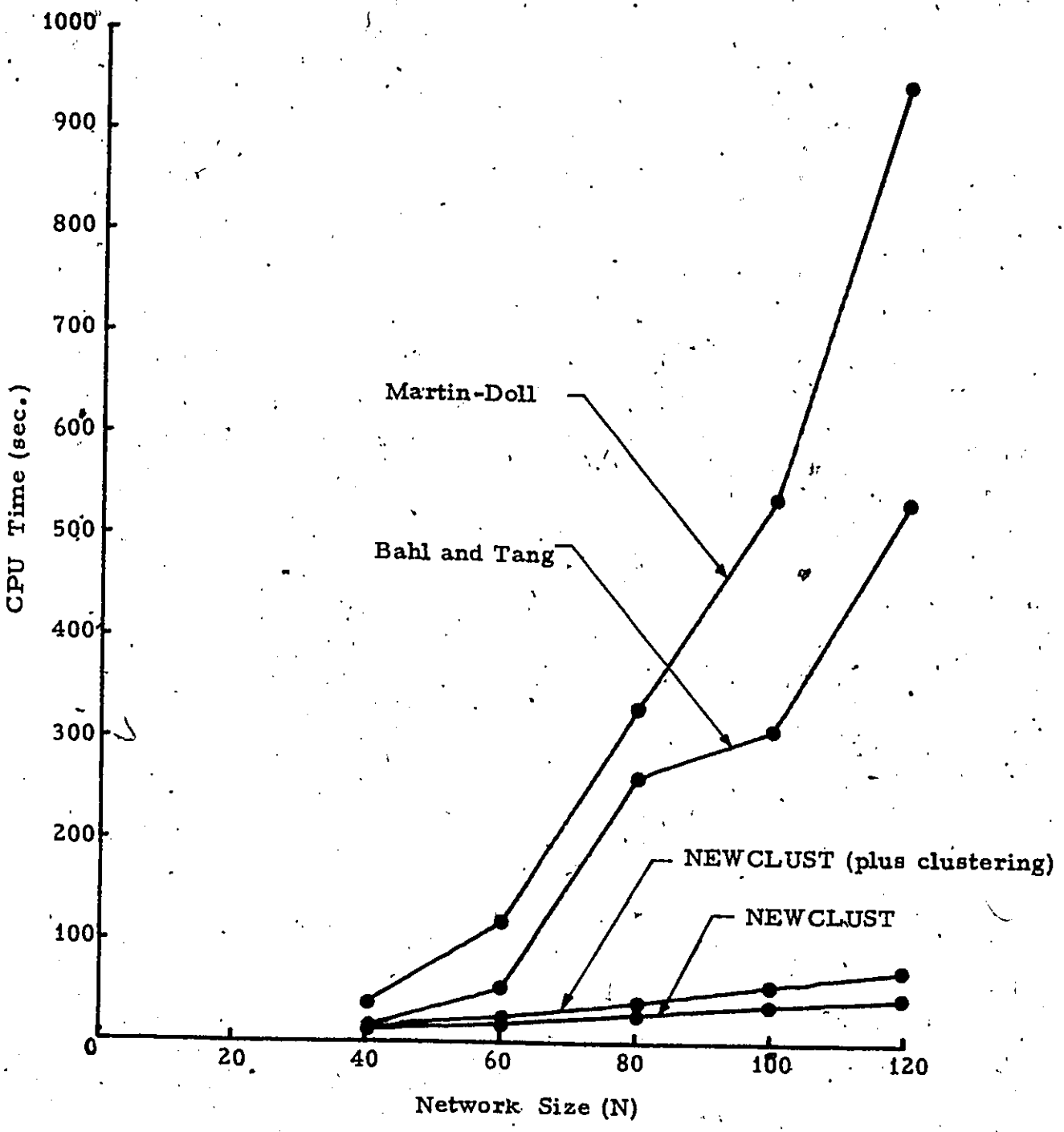


Fig. 4.7
Algorithm Times

exponentially with network size, whereas NECLUST's time increases linearly. The amount of computer storage required by the algorithm is approximately:

$k(11LL+2N+8M+MN)$;	$k\simeq 2$	NEWCLUST
$k(N+M+\frac{MN}{4})$;	$k\simeq 9$	Martin-Doll
$k(N+M+MN)$;	$k\simeq 9$	Bahl and Tang

where N= is the number of terminals in the network.

LL= is the number of expected super nodes which is in the order of N/NMAX.

M= is the number of initial candidate concentrator sites or the expected number of concentrators in the Martin-Doll case, which is in the order of $\sum_i t_i$/MAXCON

Figure 4.8 shows the different topologies and costs derived by the three algorithms for the same network of 100 terminals.

The Martin-Doll and NEWCLUST algorithms produced topologies with approximately the same number of concentrators, although NEWCLUST usually had a few more. The Bahl and Tang algorithm usually produced about twice as many concentrators as the others. This can be accounted for by the fact that the determination of the concentrators was primarily based on producing a star network. The number of concentrators could perhaps

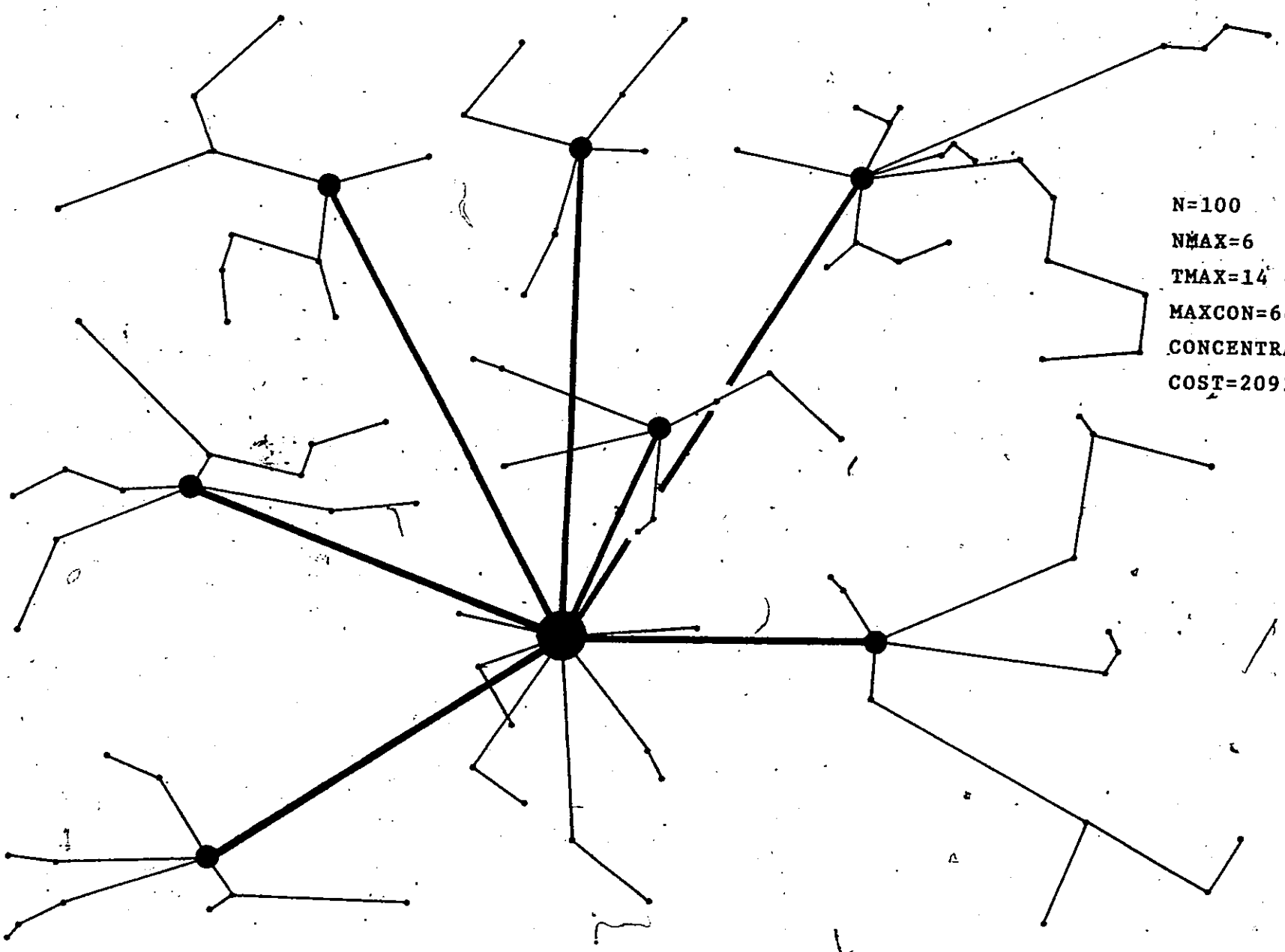


Fig. 4.8(a)
Martin-Doll

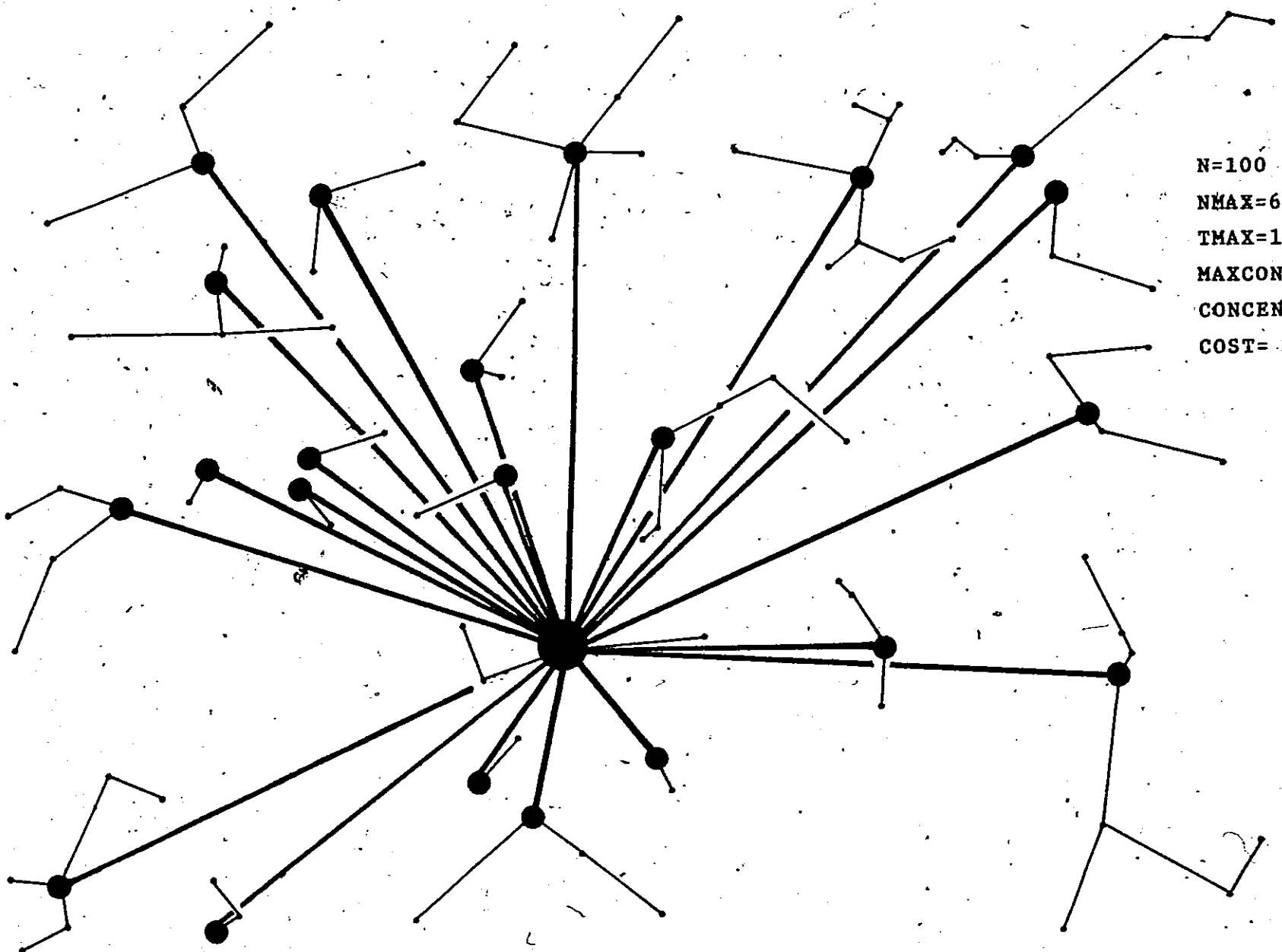


Fig. 4.8(b)
Bahl and Tang

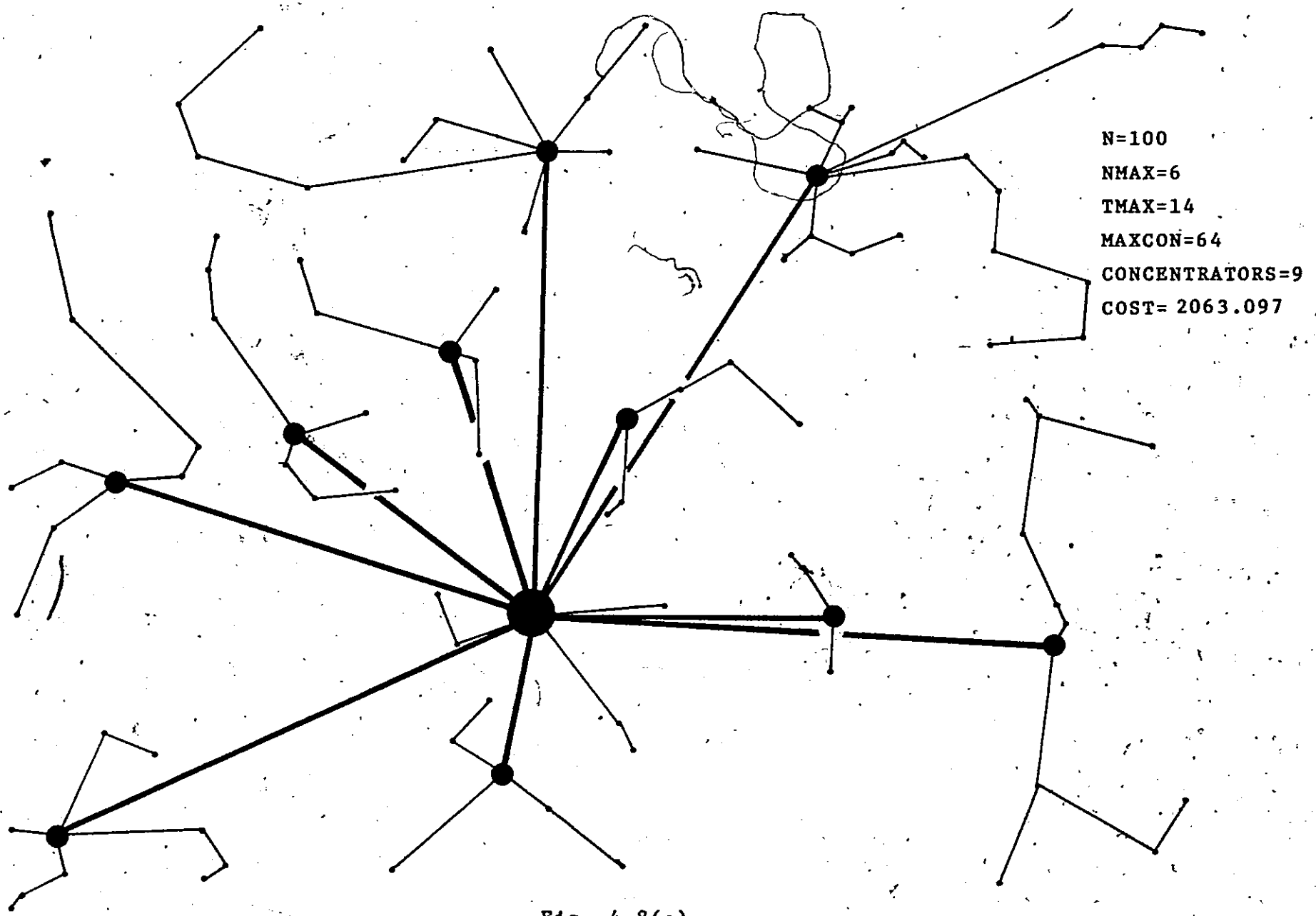


Fig. 4.8(c)
NEWCLUST

be reduced if the resultant network was used as a starting point and NEWCLUST was applied to it.

Due to the limitations on our computer-time budget we were not able to compare networks of more than 120 terminals.

4.6 Comments and Extentions

4.6-1 Multiple Concentrator Types:

Thus far we have considered the case where only one type of concentrator is used in the network. In a real design problem, however, there may be a number of different types available, C^i (i= 1,2,...K). This may be incoporated into the algorithm as follows. Assuming concentrator costs increase with capacity, we can order the concentrators by increasing capacity and cost as $C^1 < C^2 < C^3 \ldots < C^k$ and $f^1 < f^2 < f^3 \ldots < f^k$ where C^i and f^i are the capacity and fixed costs of the i^{th} type of concentrator. After the network has been initialized by assigning terminals to the concentrator sites, without considering any capacity constraints, concentrators with the minimum capacity sufficient to meet the traffic requirements are placed at each site. If any site has a traffic requirement $>C^k$ then either the terminal assignments may be modified as before or more than one concentrator located at that site. It will be necessary in this case to keep track of each

concentrators present capacity CO_j and total cost F_j.

A modification to the calculation of each concentrators gain is also necessary. Suppose a concentrator at location j_o, has the set of super nodes $\{I, II, III\}$. Also suppose these super nodes have concentrators j_1, j_2, and j_3 respectively as their next closest concentrators. These concentrators have as their present capacity and cost $[CO(1), F(1)]$, $[CO(2), F(2)]$, and $[CO(3), F(3)]$ respectively. Concentrator j_o has the present capacity $CO(j_o)$ and cost $F(j_o)$ and connection cost $CST(j_o)$. Each super node has reallocation costs and traffic requirements of $TNN1(i)$ and $NN(i,2)$ $i = I, II, III$ respectively. The gain of concentrator j_o, $GAIN(j_o)$, may then be calculated as:

$$GAIN(j_o) = CST(j_o) + F(j_o) - \sum_{i=I}^{III} TNN1(i) - \sum_{i=1}^{III} \Delta f_i$$

$$\text{where} \quad \Delta fi = \begin{cases} f^{p+k} - f^p & \text{iff} \quad CO(i) + NN(i,2) \le C^{p+k} \\ 0 & \text{if} \quad CO(i) + NN(i,2) \le C^p \end{cases}$$

where $C^{p+k} > C^p$, $f^{p+k} > f^p$ and C^p is the present maximum capacity of the concentrator j_i which has a present capacity requirement of $CO(i)$ $i = 1,2,3$. It is advisable that this modification to the gain calculation be done after the concentrator is checked for the

feasibility of its super node assignments, because more than one of its super nodes may be assigned to the same concentrator thus affecting that concentrators potential capacity requirements. For example, if super nodes I and III both had j_1 as their next closest concentrator, then the last term in the expression GAIN(j.) would only have two components rather than three as it had before. Thus the last term, which can be thought of as the incremental concentrator costs, is:

$$\sum_{i=I}^{II} \Delta f_i = \Delta f_I + \Delta f_{II}$$

where $$\Delta f_{I=} \begin{cases} f^{p+k} - f^p & \text{iff } CO(1) + NN(I,2) + NN(III,2) \leq C^{p+k} \\ 0 & \text{if } CO(1) + NN(I,2) + NN(III,2) \leq C^p \end{cases}$$

and $$\Delta f_{II=} \begin{cases} f^{\ell+k} - f^{\ell} & \text{iff } CO(2) + NN(II,2) \leq C^{\ell+k} \\ 0 & \text{if } CO(2) + NN(II,2) \leq C^{\ell} \end{cases}$$

4.6-2 Distributed CPU's

If there happens to be more than one CPU in the network and there is no capacity restrictions on the CPU's then they may be considered collectively as one super CPU. Concentrators may then be connected to that CPU in the super CPU node which provides the lowest

connection cost. Figure 4.9 illustrates a multiple CPU network in which the CPU's or hosts are interconnected. This might be applicable in either a circuit or packet switched type of computer-communication network.

4.6-3 Post-processing

Certain techniques can be used in a post-processing phase to usually incrementally improve the final solution. One technique that may reduce the network cost is to find the optimum location for a concentrator within each of the resultant concentrator subnetworks. One approach may be to consider all terminals in each subnetwork as the possible concentrator site and then apply some optimal or sub-optimal multipoint procedure a number of times considering a different terminal as the center each time. However, if there are a large number of concentrator subnets, each having a large number of terminals, this may be very time consuming. An alternative approach as illustrated in Fig. 4.10 might be more suitable. Each super node SN_i $i = 1,2,...5$ has the best location for a concentrator determined as I_i $i = 1,2,...5$ as shown in Fig. 4.10 (a). Each of these locations, I_i's, are considered in turn as concentrator locations and the connection costs between the super node of I_i and the other super nodes are calculated. In Fig. 4.10(b) a concentrator is being

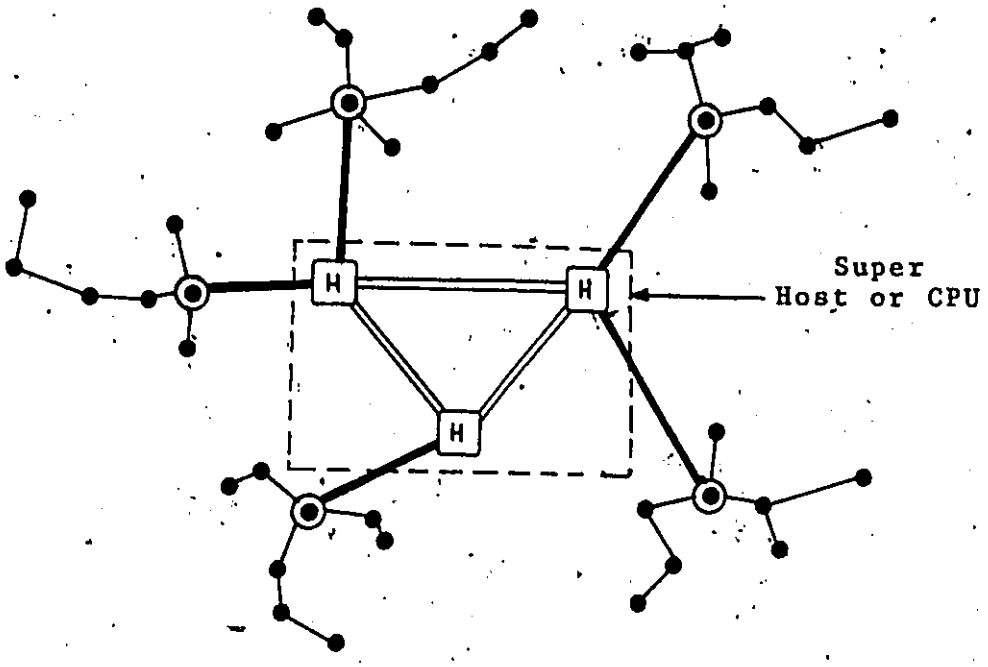
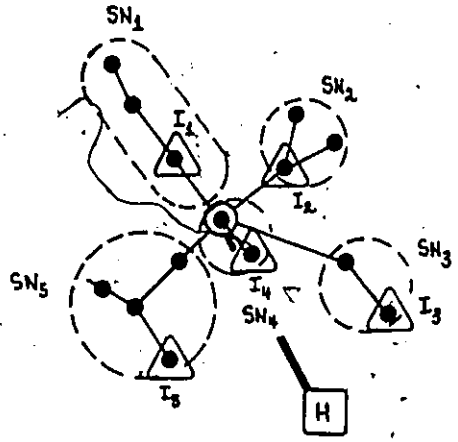
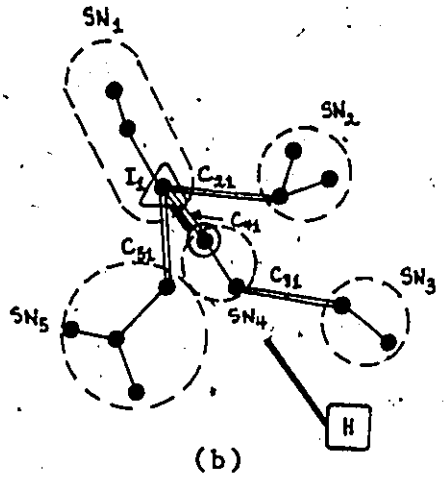


Fig. 4.9
Distributed CPU's



(a)



(b)

Cost Matrix

	I_1	I_2	I_3	I_4	I_5
SN_1	C_{11}	.	.	.	C_{15}
SN_2	.	.			.
SN_3	.		C_{33}		.
SN_4	.			.	.
SN_5	C_{51}	.	.	.	C_{55}

where $C_{ii} = f_{Ii}$ = cost of locating a concentrator at I_i

C_{ij} = the cost of the shortest feasible connection to connect the tree

(c)

Fig. 4.10
Post-processing

considered at location I_1 in SN_1 and connecting links C_j, $j=2,3..5$ are shown. If a cost matrix as in Fig. 4.10(c) is set up where the diagonal elements are the costs of locating concentrators at the I_i locations and the non-diagonal elements are the unique connection costs between super nodes, then the best location for a concentrator can be determined as:

$$I^* = \left\{ I_i \;\middle|\; \sum_{j=1}^{k} C_{jI_i} = \underset{I_i}{\text{MIN}} \left\{ \sum_{j=1}^{k} C_{jI_i} \right\} \right\}$$

where k is the number of super nodes in the concentrator subnetwork.

Another frequently used technique that may be applied after some sub-optimal multipoint procedure has been used on a subnetwork is to give each multidrop line of the subnetwork a MST topology [4]. This can be done since each multidrop line will satisfy the constraints and any topology over the set of nodes on the multidrop line will not violate the constraints. Thus the minimum cost topology, the MST, may be given to any multipoint line.

4.6-4 Other Concentrator and Multidrop Line Constraints

In many real design problems, another constraint that may be encountered is that a concentrator may only be able to handle a fixed number of low-speed lines, either multidropped or point-to-point. Usually the number of these lines increases with increased concentrator capacity. We have not considered this constraint in our design algorithm. Instead we have only considered a total capacity constraint for each concentrator. The present method of assigning terminals to concentrators and the use of the Esau-Williams algorithm cannot be adapted to handle this type of constraint because it is impossible to determine a priori how many multidrop lines will be created for any given set of terminals. If this type of constraint is introduced then it would be necessary to use a CMST type of algorithm that grows a tree outwards from a concentrator. Thus by not allowing any more than the maximum number of lines, directly connected to the concentrator to be created, one can ensure that the constraint is satisfied.

Another physical limitation that one might run into is the restriction on the number of cascaded bridges that may be allowed on one multipoint line.

That is when three or more links are incident to a terminal, the physical realization of the connection may be the bridging of two or more of the analogue telephone lines. If two or more of these bridges occur on a single multipoint line, this is called cascading of bridges. If more than a certain number of bridges are cascaded, there is then the possibility of degredation of the signal and thus the reliability is affected. The use of a CMST type of algorithm in this instance would also be desired since the number of bridges can easily be restricted as the tree is grown outward. Figure 4.11 illustrates bridging and cascading.

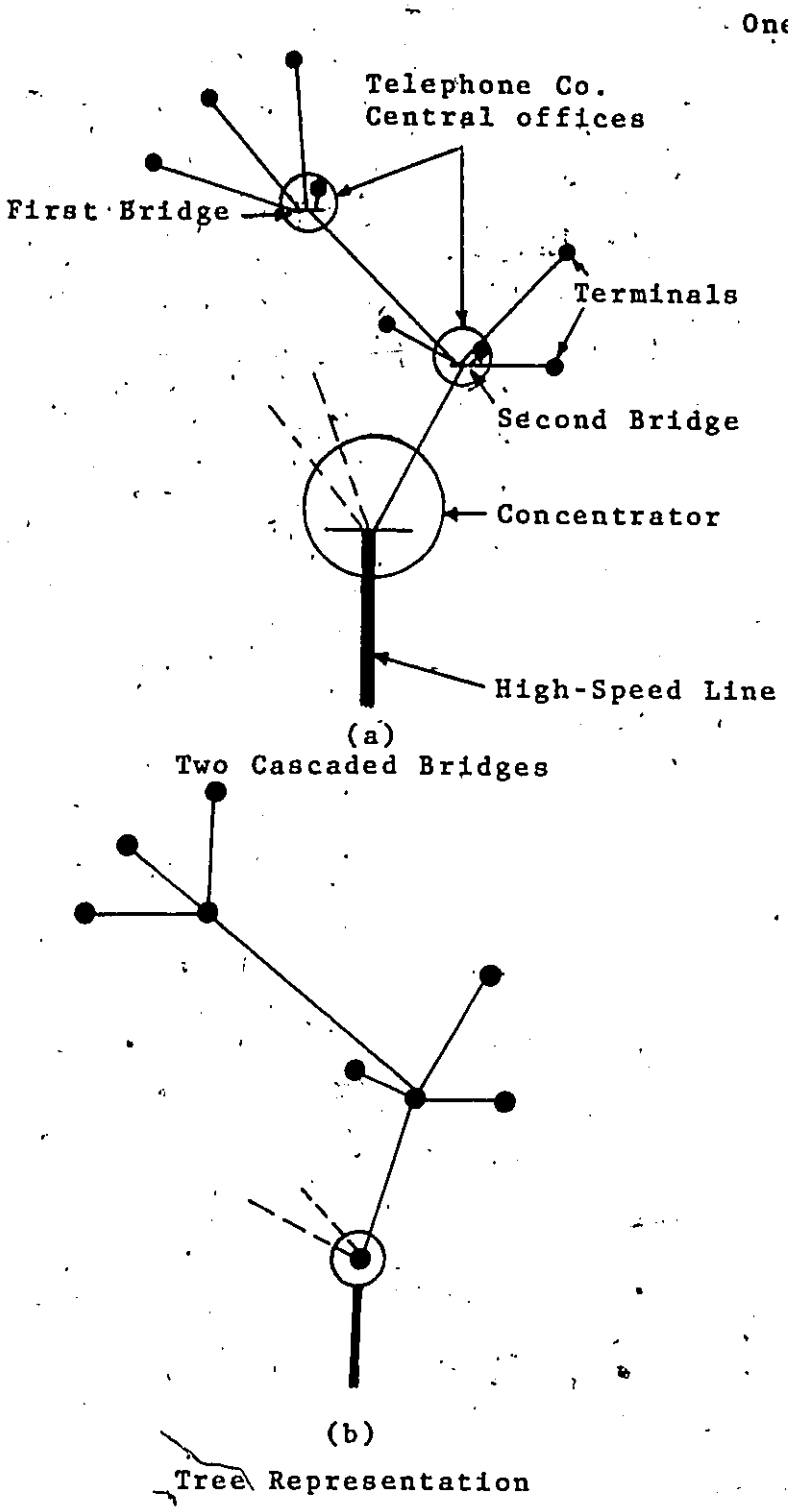


Fig. 4.11
Cascading and Bridging

CONCLUSIONS

A new heuristic algorithm, called NEWCLUST, for designing two-level multidrop networks has been presented along with some possible extentions. It has been shown to be a better heuristic algorithm than those it was compared to. The "goodness" of NEWCLUST depends on how good the choice of initial candidate concentrator sites is and how good the terminal allocation and many-center multipoint optimization procedure is. Improvements in either of these two areas should improve the quality of the solutions produced by NEWCLUST.

From the results, it appears that the reinitialization the network, once the final concentrator sites have been determined, can usually incrementally improve the cost, especially for the larger networks. This would tend to indicate that since the initial solution is a local optimum, that by reinitializing the network and iterating on the local optimum solutions, the cost may be further reduced. No experimentation, however, has been done with NEWCLUST to substantiate this.

The basic NEWCLUST algorithm appears to be the best heuristic of its type, to the author's best knowledge at least. However, as no sophisticated post-processing

techniques were implemented in any of the algorithms, further experimentation may effect the percentage improvement in cost of the NEWCLUST algorithm.

It should also be noted that in many practical solutions, the terminal population may consist of a number of different terminal types (e.g. TTY, POS, etc) as well as a number of different types of concentrators which may support only one or a varied mixture of types of terminals. If this case is to be considered, then it would be necessary to develop a more general type of algorithm that could accommodate more stringent and different concentrator constraints and also perhaps partition the network not only geographically but also by terminal types. We feel that,to this end,much more work is required in the area of the partitioning of the network, which may involve more sophisticated clustering types of procedures.

REFERENCES

[1] S. Lin, "Heuristic Programming as an aid to Network Design", Networks, Vol. 5, pp. 33-43, 1975.

[2] I. Cahit and R. Cahit, "Topological Considerations in the Design of Optimum Teleprocessing Tree Networks", IEEE National Telecommunications Conference (NTC), Vol. 1, pp. 37F 1-7, 1972.

[3] K.M. Chandy and R.A. Russel, "The Design of Multipoint Linkages in a Teleprocessing Tree Network", IEEE Trans. Computers, Vol. C-21, No. 10, pp. 1062-66, 1972.

[4] D.Z. Elias and M.J. Ferguson, "Topological Design of Multipoint Teleprocessing Tree Networks", IEEE Trans. Comm., Vol. COM-22, No. 11, pp. 1753-1762, 1974; also D. . Elias, M. Eng. thesis, McGill University, Montreal, 1973.

[5] K.M. Chandy and T. Lo, "The Capacitated Minimum Spanning Tree", Networks, Vol. 3, No. 2, pp. 173-182, 1973.

[6] A. Kershenbaum and W. Chou, "A Unified Algorithm for Designing Multidrop Teleprocessing Networks", IEEE Trans. Comm. Vol.-COM-22, No. 11, pp. 1062-1071, 1974; also, Proc. Data Comm. Conf., pp. 148-155,1973

[7] R.L. Sharma and M.T. El-Bardai, "Suboptimal Communications Network Synthesis", Proc. Int. Conf. Comm., pp. 19.11-16, 1970.

[8] J. Martin, Design of Real Time Computer Systems, Prentice-Hall Inc. Englewood Cliffs, N.Y., 1967.

[9] L.R. Esau and K.C. Williams "A Method for approximating the Optimal Network", IBM Syst. J., Vol. 5, pp. 142-147, 1966.

[10] K.M. Chandy, "Algorithms for Optimal Tree Networks", IEEE NTC, pp. 37-E1-5, 1972.

[11] N.V. Reinfield and W.R. Vogel, Mathematical Programming, Prentice-Hall Inc., Englewood Cliffs, N.Y., 1958.

[12] R.C. Prim, "Shortest Connection Networks and some Generalizations", Bell Syst. Tech. J., Vol. 36, pp. 1389-1401, 1957.

[13] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem", Proc. Am. Math. Soc., Vol. 7, pp. 48-50, 1956.

[14] H. Frank, "Optimal Design of Computer Networks", in Computer Networks, R. Rustin (ed.), Prentice-Hall Inc., Englewood Cliffs, N.Y., pp. 167-183, 1971.

[15] H. Frank and W. Chou, "Topological Optimization of Computer Networks", Proc. IEEE, Vol. 60, No. 11, pp. 1385-1397, 1972.

[16] V.K.M. Whitney, "Comparison of Network Topology Optimization Algorithms", Proc. ICCC, pp. 332-337, 1972.

[17] M. Karnaugh, "A New Class of Algorithms for Multipoint Network Optimization", IEEE Trans. Comm. Vol. COM-24, No. 5, pp. 500-505, 1976; also Proc. ICC, Vol. 3, pp. 33.7-11, 1975.

[18] A. Kershenbaum, "Computing Capacitated Minimal Spanning Trees Efficiently", Networks, Vol. 4, pp. 299-310, 1974.

[19] P.S. Davis and T.L. Ray, "A Branch and Bound Algorithm for the Capacitated Facilities Location Problem", Naval Res. Log. Quart., Vol. 16, pp. 331-344, 1969.

[20] G. Sà, "Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem", Oper. Res., Vol. 17, pp 1005-1016, 1969.

[21] M.A. Efroymson and T.L. Ray, "A Branch and Bound Algorithm for Plant Location", Oper. Res., Vol. 14, pp. 361-369, 1966.

[22] G. Zoutenduk, "Mixed Integer Programming and the Warehouse Allocation Problem", in Applications of Mathematical Programming Techniques, E.M. Beale (ed.), American Elsivier Pub. Co., New York, N.Y., pp. 203-216, 1970.

[23] E.M. Beale, "Selecting an Optimal Subset", in Integer and Nonlinear Programming, J. Abadie (ed.), North Holland Pub Co., Amsterdam, pp. 461-462, 1970.

[24] S.A. Kramer, "Warehouse Location and Allocation Problems solved by Mathematical Programming", in Integer and Nonlinear Programming, J. Abadie (ed.), North Holland Pub. Co., Amsterdam, pp. 541-549, 1970.

[25] A.K. Kuehn and M.J. Hamburger, "A Heuristic Program for Locating Warehouses", Management Sc., Vol. 9, pp. 643-666, 1963.

[26] E. Feldman, F.A. Lehrer and T.L. Ray, "Warehouse Location under Continuous Economies-of-Scale", Management Sc., Vol. 12, pp. 670-684, 1966.

[27] A.S. Manne, "Plant Location under Economies-of-Scale-Decentralization and Computation", Management Sc., Vol. 11, pp. 213-235, 1965.

[28] K. Spielburg, "Plant Location with Generalized Search Origin", Management Sc., Vol. 16, pp. 165-178, 1969.

[29] W.J. Baumol and P. Wolfe, "A Warehouse-Location Problem", Oper. Res., Vol. 6, pp. 252-263, 1968.

[30] R.C. Vergin and J.D. Rogers, "An Algorithm and Computational Procedure for Locating Economic Facilities", Management Sc., Vol. 13, pp. B-240-254, 1967.

[31] J. Martin, System Analysis for Data Transmission, Prentice-Hall Inc., Englewood Cliffs, N.Y., 1972.

[32] B.L. Deekshatulu, "Allocation of Concentrators in Teleprocessing Communication Networks", Proc. Sixth Annual Princeton Conf. Inf. Sc. and Syst., p. 132, 1972.

[33] L.R. Bahl and D.T. Tang, "Optimization of Concentrator Locations in Teleprocessing Networks", in Computer Communications Networks and Teletraffic, J. Fox (ed.), Polytechnic Press, New York, N.Y., pp.355-362, 1972.

[34] D.A. Greenberg, "A New Approach for the Optimal Placement of Concentrators in a Remote Terminal Communication Network". Proc. NTC, Vol. 1, pp. 37D1-7, 1972.

[35] M. Lewin, "Determination of Locations, Capacities and Boundaries of Local Exchanges in a Telephone Network", DATACOM 73, pp. 16E1-6, 1973.

[36] H. Dirilten and R.W. Donaldson, "Topological Design of Teleprocessing Networks using Linear Regression Clustering", to be published.

[37] H. Frank, "Computer Networks: Art to Science to Art", Networks, Vol. 5, pp. 7-32, 1975.

[38] L.S. Woo and D.T. Tang, "Optimization of Teleprocessing Networks with Concentrators", Proc. NTC, Vol. 1, pp. 37C1-5, 1972.

[39] D. Doll, "Topology and Transmission Rate considerations in the Design of Centralized Computer Communication Networks", IEEE Trans. Comm. Tech., Vol. COM-19, pp. 339-344, 1971.

[40] _____, "Multiplexing and Concentration", Proc. IEEE, Vol. 60, pp. 1313-1321, 1972.

[41] R.S. Wilkov, "On the Design of Multiple Access Computer Communication Networks", Proc. NTC, Vol. II, pp. 23F1-4, 1972.

[42] H. Dysart and N.D. Georganas, "Optimizing Concentrator Positioning in the Design of Two-Level Teleprocessing Networks", Proc. 1976 IEEE, Canadian Conf. on Comm. and Power, Montréal, Oct. 1976.

[43] H. Dysart and N.D. Georganas, "A Heuristic Approach to Locating Concentrators in Computer-Communication Networks", Eighth Biennial Symposium on Comm., Queen's University, Kingston, June 1976.

[44] P.E. Green and D.T. Tang, "Some Recent Developments in Teleprocessing System Optimization", Proc. IEEE INTERCON, Vol. 1, pp. 1-7, 1973.

[45] J. Soukup, "Multilevel Network Design: An Allocation Approach", private communication.

[46] V.G. Cerf, D.D. Cowan, R.C. Mullin and R.G. Stanton, "Topological Design Considerations in Computer Communication Networks", in Computer Communications, R.L. Grimsdale and F.F. Kuo (ed.s), (NATO Advanced Study Institutes Series, Series E: Applied Sciences, No. 4), Noordhoff Int. Pub. Co., Netherlands, pp. , 1975.

[47] H. Frank and I.T. Frisch, Communication, Transmission and Transportation Networks, Addison - Wesley Pub. Co. Inc., Reading, Mass., 1971.

[48] F.T. Boesch (ed.), Large-Scale Networks: Theory and Design, IEEE Press, New York, N.Y., 1976.

[49] P.E. Green and R.W. Lucky (ed.s), Computer Communications, IEEE Press, New York, N.Y.,1975

[50] H. Dysart and N.D. Georganas "Topological Design of Hierarchical Teleprocessing Networks", Proc. 1976 International Conference on Information Sciences and Systems, Patras-Greece, Aug. 1976.

APPENDIX A

Fortran Programs

```
C
C
C                    JAMES MARTINS ALGORITHM
C
C
C     THIS ALGORITHM IS THE ADD TYPE OF APPROACH
C     IT CONSIDERS ALL TERMINAL LOCATIONS AS POSIBLE CONCENTRATOR SITES
C     CONCENTRATORS ARE LOCATED SEQUENTIALLY ONE AT A TIME UNTIL IT IS
C     NO LONGER ECONOMICAL TO LOCATE ANOTHER ONE.ONCE THE CONCENTRATOR
C     SITES HAVE BEEN DETERMINED,THEN A MANY-CENTER MULTIPOINT OPTIMIZ-
C     ATION PROCEDURE IS USED TO DETERMINE THE LOW-SPEED LINE LAYOUT.
C
C
C     USER SUPPLIED PARAMETERS:
C
C        NTERM= NUMBER OF TERMINALS IN THE NETWORK
C        MMM= THE NETWORK IDENTIFIER NUMBER
C        FIXC= THE FIXED COST OF THE CONCENTRATOR
C        NMAX= THE MAXIMUM NUMBER OF TERMINALS PER MULTIDROP LINE
C        TMAX= THE MAXIMUM TRAFFIC PER MULTIDROP LINE
C        LMAX= THE MAXIMUM NUMBER OF LINES PER CONCENTRATOR (NOT USED)
C        MAXCON= THE CAPACITY OF A CONCENTRATOR
C
C
      REAL FIXC,XT(160,2),COST,SAVC,DENS(160),DCPU,DCON,MAX,LD,HD,
     *DO(160,20)
      INTEGER*2 NMAX,TMAX,LMAX,NTERM,C(20),T(160),NONC(20,160)
      INTEGER*2 NUM,PPX,PX,IQST,IMQ,MAXCON,ISS,XCONC(160),TLIN(20)
      INTEGER*2 TCO(20),REAS(20,2),N(20),Y(20)/20*0/,QST,IG,YZ(160)/
     *160*0/,YYZ(160)/160*0/,NA,IX,IN(40)
      LOGICAL FLAG(160)/160*F/,CONC(160),POUT(160)/160*F/
      COMMON /AREA1/DO,T/AREA2/TMAX,NMAX/AREA3/COST/AREA4/XT/AREA5/FIXC
      COMMON /AREA7/TCO/AREA8/C/AREA9/N,NONC
C
C
 75   FORMAT('1','********** MARTIN ADD **********')
      PRINT75
      READ,NTERM,MMM
      READ,FIXC,NMAX,TMAX,LMAX,MAXCON
      PRINT774,MMM
 774  FORMAT(' ','********** RANDOM NET ',I4,'  **********
     @**********')
      PRINT1,FIXC,NTERM,NMAX,TMAX,LMAX,MAXCON
 1    FORMAT(' ','FIXC=',F7.2,' NTERM=',I3,' NMAX=',I3,' TMAX=',I3,
     *' LMAX=',I3,' MAXCON=',I3)
       PRINT70
 70   FORMAT('1','NODE',5X,'X',5X,'Y',3X,'TRAFFIC')
      DO 2 I=1,NTERM
      READ,(XT(I,J),J=1,2),T(I)
       PRINT73,I,XT(I,1),XT(I,2),T(I)
  73  FORMAT(' ',I3,2X,F7.3,2X,F7.3,2X,I4)
 2    CONTINUE
      X1=XT(1,1)
      Y1=XT(1,2)
      C(1)=1
      SAVC=0.00
      PPX=1
 9000 MAX=0.00
C
```

```
C       INITIALIZE THE DENSITY FACTORS
C
        DO 11 I=1,NTERM
11         DENS(I)=0.00
        DO 21 I=1,NTERM
           IF(FLAG(I)) GO TO 21
           IF(POUT(I)) GO TO 21
        DO 31 II=1,LMAX
31         TLIN(II)=0.
        DO 41 II=1,NTERM
41         CONC(II)=.FALSE.
           NUM=0
           DCPU=0.00
           DCON=0.00
200        DMIN=10000.0
        DO 51 J=1,NTERM
           IF(FLAG(J)) GO TO 51
           DX=LD(I,J)
           IF(DX.GE.DMIN.OR.CONC(J)) GO TO 51
             DMIN=DX
             L=J
51      CONTINUE
        DO 61 M=1,LMAX
             IF(TMAX-TLIN(M).LE.T(L).OR.CONC(L)) GO TO 61
             TLIN(M)=TLIN(M)+T(L)
             CONC(L)=.TRUE.
             NUM=NUM+1
             DCPU=DCPU+LD(L,1)
             DCON=DCON+LD(I,L)
             LIN=M
             GO TO 200
61      CONTINUE
        DCPU=DCPU/NUM
        DCON=DCON/NUM
        DENS(I)=LIN*(DCPU-DCON)-HD(I)
        IF(DENS(I).LT.0.0) POUT(I)=.TRUE.
        IF(DENS(I).LE.MAX) GO TO 21
             MAX=DENS(I)
             K=I
             PX=0
        DO 7 JJ=1,NTERM
           IF(JJ.EQ.I) GO TO 7
           IF(.NOT.CONC(JJ)) GO TO 7
           PX=PX+1
           XCONC(PX)=JJ
7       CONTINUE
21      CONTINUE
        IF(MAX.EQ.0.00) GO TO 1000
        PPX=PPX+1
        C(PPX)=K
        PRINT272,PPX,K
272     FORMAT(' ','CONCENTRATOR',I4,' IS LOCATED AT NODE',I4)
        FLAG(K)=.TRUE.
        DO 8 J=1,PX
           FLAG(XCONC(J))=.TRUE.
8       CONTINUE
        SAVC=SAVC+MAX
        GO TO 9000
C
```

```
C     NOW THE CONCENTRATOR SITES HAVE BEEN DETERMINED
C     THE COST MATRIX DO(I,J) IS CALCULATED
C     DO(I,J) IS THE COST OF CONNECTING TERMINAL I TO CONCENTRATOR J
C
 1000 DO 110 I=1,NTERM
      DO 110 J=1,PPX
         JX=C(J)
         DO(I,J)=LD(I,JX)
  110 CONTINUE
      DO 211 I=1,PPX
         II=C(I)
  211    YYZ(II)=1
C
C     TERMINALS ARE ASSIGNED TO THEIR CLOSEST CONCENTRATOR
C
      DO 20 I=1,NTERM
         IF(YYZ(I).EQ.1) GO TO 20
         CALL AMIN(I,PPX,ISS)
         N(ISS)=N(ISS)+1
         NA=N(ISS)
         IX=I
         NONC(ISS,NA)=IX
   53    TCO(ISS)=TCO(ISS)+T(I)
   20 CONTINUE
C
C     COCENTRATORS ARE CHECKED TO SEE IF THEIR CAPACITY CONSTRAINT
C     HAS BEEN VIOLATED.THE TERMINAL ASSIGNMENTS OF VIOLATED CONCENT-
C     RATORS ARE MODIFIED.
C
      DO 77 I=2,PPX
         IF(TCO(I).GT.MAXCON) Y(I)=1
   77 CONTINUE
      DO 50 I=2,PPX
         IF(Y(I).EQ.0) GO TO 50
         CALL CFUL(I,MAXCON,Y,PPX)
         IF(TCO(I).LT.MAXCON) Y(I)=0
   50 CONTINUE
C
C     THOSE TERMINALS THAT WERE DROPPED WHEN THE TERMINAL ASSIGNMENTS
C     WERE MODIFIED ARE NOW REASSIGNED TO OTHER CONCENTRATORS.
C
      CALL REAR(PPX,MAXCON,Y)
C
C     EACH CONCENTRATOR SUBNET IS OPTIMIZED INTO A MULTIDROP NETWORK
C
      DO 111 I=1,PPX
         NA=N(I)
         IX=I
         IC=C(I)
         COST=COST+HD(IC)
         IF(NA.EQ.0) GO TO 111
      DO 121 J=1,NA
  121    IN(J)=NONC(I,J)
         CALL SAHC(NA,IX,IN)
  111 CONTINUE
      PRINT7000,COST
 7000 FORMAT(' ','TOTAL NETWORK COST IS=',F8.3)
      PRINT7001,PPX,SAVC
 7001 FORMAT(' ','THERE ARE',I4,2X,'CONC,RESULTING IN A SAVING OF',F8.3)
```

```
      PRINT7002
 7002 FORMAT(' ',' CONC NUM',5X,'NODE',5X,'# OF NODES',5X,'NODES')
      DO 44 J=1,PPX
      NA=N(J)
      PRINT,J,C(J),NA,(NONC(J,K),K=1,NA)
 44   CONTINUE
      STOP
      END
      REAL FUNCTION LD(II,J)
C
C     CALCULATE LOW SPEED LINE COST
C
      REAL XT(160,2),DIST(5),FAC(5),FIX(5),FIXC
      COMMON /AREA4/XT
      DATA DIST/100.,50.,25.,10.,0.0/,FIX/86.2,59.7,42.2,23.75,6.25/,
     *FAC/.35,.53,.70,1.23,1.75/
C
      D=SQRT((XT(II,1)-XT(J,1))**2+(XT(II,2)-XT(J,2))**2)
      IF(D.EQ.0.0) GO TO 3
      DO 1 I=1,5
        IF(D.LE.DIST(I)) GO TO 1
        LD=FIX(I)+(D-DIST(I))*FAC(I)
        GO TO 2
 1    CONTINUE
 3    LD=0.0
 2    RETURN
      END
      REAL FUNCTION HD(J)
C
C     CALCULATE HIGH SPEED LINE COSTS
C
      REAL XT(160,2),HDIST(5),HHFIX(5),HFAC(5),FIXC
      DATA HDIST/50.,25.,10.,2.5,0.0/,HHFIX/74.5,48.25,25.75,10.,2.5/,
     *HFAC/.75,1.05,1.5,2.1,3.0/
      COMMON /AREA4/XT/AREA5/FIXC
C
      X=XT(1,1)
      Y=XT(1,2)
      D=SQRT((XT(J,1)-X)**2+(XT(J,2)-Y)**2)
      IF(D.EQ.0.0) GO TO 3
      DO 1 I=1,5
         IF(D.LE.HDIST(I)) GO TO 1
         HD=HHFIX(I)+(D-HDIST(I))*HFAC(I)
         HD=HD+FIXC
         GO TO 2
 1    CONTINUE
 3    HD=FIXC
 2    RETURN
      END
      SUBROUTINE REAR(PPX,MAXCON,Y)
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM
C     IT REASSIGNS THE TERMINALS THAT WERE DROPPED WHEN THE CONCENTRATOR
C     TERMINAL ASSIGNMENTS WERE MODIFIED.
C
      REAL DO(160,20)
      INTEGER*2 T(160),NONC(20,160),TCO(20),DCON(20),N(20),TC(20)
      INTEGER*2 DCN(20,50),ISS,PPX,MAXCON,Y(20),IJ
      INTEGER LP,REAS(40,2)
```

```
      COMMON /AREA1/DO,T/AREA7/TCO/AREA9/N,NONC/AREA10/TC,DCON,DCN
      COMMON /AREA11/REAS,LP
C
      IF(LP.EQ.0) GO TO 900
      DO 1 I=1,PPX
         TC(I)=0
 1       DCON(I)=0
C
C     THE CONCENTRATORS TO WHICH THE DROPPED TERMINALS ARE TO BE POTENT-
C     IALLY ASSIGNED ARE DETERMINED.THE NUMBER OF TERMIALS AND THEIR
C     TOTAL TRAFFIC TO EACH CONCENTRATOR IS ALSO FOUND.
C
      DO 2 I=1,LP
         IX=REAS(I,1)
         IXC=REAS(I,2)
         DCON(IXC)=DCON(IXC)+1
         NA=DCON(IXC)
         DCN(IXC,NA)=IX
         TC(IXC)=TC(IXC)+T(IX)
 2    CONTINUE
C
C     IF A CONCENTRATOR CAN HANDLE ALL OF THE TRAFFIC OF ITS POTENTIALLY
C     ASSIGNED SET OF TERMINALS,THEN THEY ARE ASSIGNED TO IT.
C     IF IT CANNOT,THEN SOME OF THE TERMINALS ARE POTENTIALLY ASSIGNED
C     TO SOME OTHER CONCENTRATOR THAT CAN ACCOMODATE THEM.
C
      DO 3 I=1,PPX
         IF(DCON(I).EQ.0) GO TO 3
         IF(TCO(I)+TC(I).GT.MAXCON) GO TO 5
 55      TCO(I)=TCO(I)+TC(I)
         KP=N(I)
         NP=N(I)+1
         NNP=N(I)+DCON(I)
         N(I)=NNP
      DO 4 J=NP,NNP
         KKP=J-KP
         NONC(I,J)=DCN(I,KKP)
 4    CONTINUE
         GO TO 3
 5       CALL FMI(I,IJ)
         Y(I)=1
         CALL AAMIN(PPX,IJ,ISS,MAXCON,Y)
         Y(I)=0
         IF(ISS.LT.I) GO TO 7
         DCON(ISS)=DCON(ISS)+1
         TC(ISS)=TC(ISS)+T(IJ)
         DCN(ISS,DCON(ISS))=IJ
         IF(TCO(I)+TC(I).GT.MAXCON) GO TO 5
         IF(DCON(I).EQ.0) GO TO 3
         GO TO 55
 7       N(ISS)=N(ISS)+1
         TCO(ISS)=TCO(ISS)+T(IJ)
         NONC(ISS,N(ISS))=IJ
         IF(DCON(I).EQ.0) GO TO 3
         IF(TCO(I)+TC(I).GT.MAXCON) GO TO 5
         GO TO 55
 3    CONTINUE
 900  RETURN
      END
```

```
      SUBROUTINE FMI(IK,IJ)
C
C     THIS SUBROUTINE IS CALLED BY REAR.
C     IT DETERMINES WHICH TERMINAL OF THE SET OF TERMINALS POTENTIALLY
C     ASSIGNED TO A CONCENTRATOR, IS TO BE DROPPED, AND THEN DROPS IT.
C
      REAL MAX,DO(160,20)
      INTEGER*2 DCN(20,50),TC(20),DCON(20),NK,T(160),IX,IJ
      COMMON /AREA1/DO,T/AREA10/TC,DCON,DCN
C
      MAX=0.0
      NK=DCON(IK)
      DO 1 I=1,NK
         IX=DCN(IK,I)
         D=DO(IX,IK)
         IF(D.LE.MAX) GO TO 1
           MAX=D
           IJ=IX
           IJP=I
 1    CONTINUE
      TC(IK)=TC(IK)-T(IJ)
      DCN(IK,IJP)=DCN(IK,NK)
      DCON(IK)=DCON(IK)-1
      RETURN
      END
      SUBROUTINE CFUL(K,MAXCON,Y,PPX)
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT MODIFIES A CONCENTRATORS TERMINAL ASSIGNMENT SO AS TO SATISFY
C     THE CAPACITY CONSTRAINT.
C     A TRADOFF IS CALCULATED FOR EACH TERMINAL.THE TERMINALS WITH THE
C     SMALLEST TRADEOFFS ARE DROPPED FROM THE CONCENTRATOR SUBNET UNTIL
C     THE CAPACITY CONSTRAINT IS SATISFIED.
C
      REAL DO(160,20),MIN,MAX,MMAX,STO3(40)
      INTEGER*2 T(160),N(20),NONC(20,160),TCO(20),NA,NAA,MAXCON
      INTEGER*2 Y(20),Z(40),PPX,STO1(40),STO2(40),COM(20,2)
      INTEGER REAS(40,2),LP
      COMMON /AREA1/DO,T/AREA7/TCO/AREA9/N,NONC/AREA11/REAS,LP
C
      DO 33 I=1,40
 33      Z(I)=0
      NA=N(K)
      NAA=NA
C
C     THE NEAREST CONCENTRATOR FOR EACH TERMINAL ASSIGNED TO CONCENTRA-
C     TOR K IS FOUND.
C
      DO 1 I=1,NA
         IX=NONC(K,I)
         MIN=10000.
      DO 2 J=1,PPX
         IF(Y(J).EQ.1) GO TO 2
         IF(J.EQ.K) GO TO 2
         D=DO(IX,J)
         IF(D.GE.MIN) GO TO 2
           MIN=D
           IXJ=J
 2    CONTINUE
```

```
C
C     THE TERMINAL ,CLOSEST CONCENTRATOR AND TRADEOFF ARE STORED.
C
         STO1(I)=IXJ
         STO2(I)=IX
         STO3(I)=MIN-DO(IX,K)
 1     CONTINUE
 400   L=0
       MAX=10000.
C
C     THE SMALLEST TRADEOFF IS FOUND.
C
       DO 3 I=1,NA
          IF(Z(I).EQ.1) GO TO 3
          D=STO3(I)
          IF(D.GE.MAX) GO TO 3
            MAX=D
            IXP=I
 3     CONTINUE
C
C     ALL TRADEOFFS OF THE SAME VALUE ARE CONSIDERED.
C
       DO 4 I=1,NA
          IF(Z(I).EQ.1) GO TO 4
          DX=STO3(I)
          IF(DX.NE.MAX) GO TO 4
          L=L+1
          COM(L,1)=I
          COM(L,2)=DO(STO2(I),K)
 4     CONTINUE
          MMAX=COM(1,2)
          IM=1
          IF(L.EQ.1) GO TO 6
C
C     THE TERMINAL FARTHEST FROM THE CONCENTRATOR K IS DROPPED FROM
C     K'S SUBNET.
C     IF DROPPING THIS TERMINAL DOES NOT SATISFY THE CAPACITY CONSTRAINT
C     ,THEN THE OTHER TERMINALS WITH THE PRESENT MINIMUM TRADEOFF
C     ARE CONSIDERED FOR DROPPING.IF THERE ARE NO MORE TERMINALS WITH
C     THE PRESENT MINIMUM TRADEOFF, THEN A NEW MINIMUM AND THE CORRESP-
C     ONDING TERMINALS IS FOUND.
C
       DO 5 I=2,L
          D=COM(I,2)
          IF(D.LE.MMAX) GO TO 5
          MMAX=D
          IM=I
 5     CCNTINUE
 6     LP=LP+1
       IQ=COM(IM,1)
       IQQ=STO2(IQ)
       IPQ=STO1(IQ)
       REAS(LP,1)=IQQ
       REAS(LP,2)=IPQ
       TCO(K)=TCO(K)-T(IQQ)
       Z(IQ)=1
       IF(TCO(K).LE.MAXCON) GO TO 7
       GO TO 400
 7     N(K)=0
```

```
C
C     THE CONCENTRATOR'S VECTOR OF TERMINALS ASSIGNED TO IT IS RESTORED.
C
      DO 45 I=1,NA
         IF(Z(I).EQ.1) GO TO 45
         N(K)=N(K)+1
         NONC(K,N(K))=STO2(I)
 45   CONTINUE
      RETURN
      END
      SUBROUTINE AMIN(K,M,ISS)
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT FINDS THE CLOSEST CONCENTRATOR TO A TERMINAL.
C
      INTEGER*2 M,T(160),ISS
      REAL DO(160,20),MIN
      COMMON /AREA1/DO,T
C
      MIN=DO(K,1)
      ISS=1
      DO 1 I=2,M
         DX=DO(K,I)
         IF(DX.GE.MIN) GO TO 1
           MIN=DX
           ISS=I
 1    CONTINUE
      RETURN
      END
      SUBROUTINE AAMIN(PPX,IG,ISS,MAXCON,Y)
C
C     THIS SUBROUTINE IS CALLED BY REAR.
C     IT FINDS THE CLOSEST CONCENTRATOR TO A TERMINAL THAT CAN HANDLE
C     ITS TRAFFIC.
C
      INTEGER*2 PPX,IG,MAXCON,Y(20),T(160),TCO(20),C(20),ISS
      REAL DO(160,20),MIN
      COMMON /AREA1/DO,T/AREA7/TCO/AREA8/C
C
      MIN=DO(IG,1)
      ISS=1
      DO 1 I=2,PPX
         IF(Y(I).EQ.1) GO TO 1
         IF(TCO(I)+T(IG).GT.MAXCON) GO TO 1
         DX=DO(IG,I)
         IF(DX.GE.MIN) GO TO 1
           MIN=DX
           ISS=I
 1    CONTINUE
      RETURN
      END
      SUBROUTINE SAHC(KK,JK,IN)
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT IS BASED ON THE ESAU-WILLIAMS ALGORITHM
C     IT FORMS A SUBOPTIMAL CONSTRAINED MULTIDROP NETWORK AROUND
C     A CONCENTRATOR.
C
      REAL MMAX,MAX,DO(160,20),D(40,40),TB1(40),TO3(40,40),XT(160,2)
```

```
     *,TT(40),TTMAX,COST,MIN,LD,L(40)
      INTEGER*2 TS(40),Y(40),N(40),REL(40,40),X(40,40),IN(40),NI(40)
      INTEGER*2 TO1(40,40),TO2(40,40),SS(40,40),T(160),TB2(40)
      INTEGER*2 TMAX,TX,JK,OO,DE,IISEG,BX,TON,TOS,DP,P,PP,I
      INTEGER*2 SI,SJ,SJJ,SEG,NONX,LX,PX,KX,KK,NMAX,LLL,OUT(10)/10*0/
      INTEGER*2 XO(40,80),PPQ
      COMMON/AREA1/DO,T/AREA2/TMAX,NMAX/AREA3/COST/AREA4/XT/AREA6/SS,D
C
  25  FORMAT(' ','X(',I3,',',I3,')=1')
  19  FORMAT(' ','COST=',F7.2)
  20  FORMAT(' ',2X,I4,5X,I4)
  21  FORMAT(' ',2X,'TERMINAL',I4,2X,'REQUIRES',F6.2,'LINES')
  30  FORMAT(' ','SAHC NODE',5X,'ORIGINAL NET NODE')
C
C     DEFINITION OF THE VARIABLES:
C
C     TO1(I,J)= THE NODE OF SEGMENT I WHICH IS TO BE JOINED TO SEGMENT J
C     TO2(I,J)= THE NODE OF SEGMENT J WHICH IS TO RECEIVE THE LINK
C               FROM SEGMENT I
C     TO3(I,J)= THE TRADEOFF FOR SEGMENT I. THIS IMPLIES DROPPING
C               THE CENTRAL LINK FROM SEGMENT I AND ADDING A
C               NON-CENTRAL LINK BETWEEN I AND J.
C     TB1(I)= THE MAXIMUM TRADEOFF FOR THE SEGMENT I
C     TB2(I)= THE SEGMENT THAT THE MAXIMUM TRADEOFF FOR SEGMENT I IS
C             ASSOCIATED WITH
C     TS(I)= THE TOTAL TRAFFIC ON SEGMENT I
C     N(I)= THE TOTAL NUMBER OF NODES ON SEGMENT I
C           IT IS LESS THAN OR EQUAL TO NMAX.
C     IN(J)= THE VECTOR OF TERMINALS ASSIGNED TO CONCENTRATOR J.
C     REL(I,J)= THE RELATION MATRIX FOR THE SEGMENTS. IF A
C               CHANGE OCCURS IN SEGMENT I, THEN REL(I,J)=1 FOR ALL J.
C               IF A CHANGE OCCURS IN SEGMENT J THEN REL(J,I)=1
C               FOR ALL I. THIS IS DONE TO EVALUATE WHETHER OR
C               NOT IT IS NECESSARY TO RECALCULATE TRADEOFFS.
C               BEFORE ACTING ON THE EXCHANGE ASSOCIATED WITH THE
C               MAXIMUM TRADEOFF BETWEEN I AND J REL(I,J) AND REL(J,I)
C               CHECKED. IF EITHER EQUALS 1 THEN THE TRADEOFF(S) IS
C               RECALCULATED.
C     OUT(J)= THE VECTOR OF TERMINALS THAT REQUIRE ONE OR
C             MORE FULLY LOADED LINES.
C     NI(J)= THE VECTOR OF TERMINALS THAT DONNOT REQUIRE FULLY
C            LOADED LINES. IT IS A PROPER SUBSET OF IN(J).
C     D(I,J)= THE COST MATRIX. IT IS ASSUMED TO BE SYMMETRICAL.
C     SS(I,...)= THE VECTOR OF NODES THAT ARE ON SEGMENT I
C     Y(*)= THE STATE VECTOR OF THE CENTRAL LINKS.THIS IMPLIES THAT
C           WHEN Y(*)=1 THE CENTRAL LINK * IS IN THE TREE. IF Y(*)=0
C           THEN THE CENTRAL LINK * IS NOT IN THE TREE AND HAS BEEN
C           DROPPED.
C
C     DETERMINE WHICH TERMINALS REQUIRE FULLY LOADED LINES,AND
C     THE NUMBER OF LINES , AND ELIMIATE THEM FROM FURTHER
C     CONSIDERATION.
C
C

      PPQ=0
      P=0
      DO 1 I=1,KK
      IZ=I
      TT(IZ)=0.0+T(IN(IZ))
```

```
      TTMAX=0.0+TMAX
      L(IZ)=TT(IZ)/TTMAX
      IF(L(IZ).LE.1.0) GO TO 2
      JL=T(IN(IZ))/TMAX
      PPQ=PPQ+1
      OUT(PPQ)=IN(IZ)
      IF(L(IZ)-JL.EQ.0.0) GO TO 29
      L(IZ)=JL+1.0
 29   PRINT21,IN(IZ),L(IZ)
      DO 52 JT=1,M
 52      DO(IN(IZ),JT)=L(IZ)*DO(IN(IZ),JT)
         COST=COST+DO(IN(IZ),JK)
         PRINT25,IN(IZ),JK
         GO TO 1
 2       P=P+1
         NI(P)=IN(IZ)
 1    CONTINUE
      KK=P
C
C     THE VECTOR NI NOW HAS THE SET OF FEASIBLE OR NOT FULLY LOADED
C     TERMINALS OR NODES.
C
C
C     THE COST MATRIX FOR THESE TERMINALS IS NOW CALCULATED.
C
      LLL=0
      DO 3 J=1,KK
         LLL=LLL+1
         JJ=NI(J)
         D(J,J)=0.
      DO 34 I=LLL,KK
         IF(I.EQ.J) GO TO 34
         II=NI(I)
         D(I,J)=LD(II,JJ)
         D(J,I)=D(I,J)
 34   CONTINUE
 3    CONTINUE
C
C
C        INITIALIZATION
C
C
      DO 51 I=1,KK
      DO 51 J=1,KK
         X(I,J)=0
 51   CONTINUE
      DO 50 I=1,KK
         IX=I
         XO(IX,JK)=1
         Y(IX)=1
         TB1(IX)=0.0
         TB2(IX)=JK
         SS(IX,1)=IX
         N(IX)=1
         TS(IX)=T(NI(IX))
 50   CONTINUE
C
C     THE MAXIMUM TRADEOFFS FOR EACH SEGMENT ARE NOW CALCULATED.
C
```

```
      DO 6 I=1,KK
         IX=I
      DO 66 J=1,KK
         IY=J
         IF(IX.EQ.IY)GO TO 66
         REL(IX,IY)=0
         TX=TS(IX)+TS(IY)
         IF(TX.GT.TMAX) GO TO 7
         TO1(IX,IY)=IX
         TO2(IX,IY)=IY
         TO3(IX,IY)=DO(NI(IX),JK)-D(IX,IY)
         GO TO 66
7        TO1(IX,IY)=0
         TO2(IX,IY)=0
         TO3(IX,IY)=-1.0
66    CONTINUE
6     CONTINUE
      DO 8 I=1,KK
         IXX=I
         MMAX=-1.0
         ISEG=0
      DO 9 J=1,KK
         IYY=J
         IF(IXX.EQ.IYY) GO TO 9
         IF(TO3(IXX,IYY).LE.MMAX) GO TO 9
         MMAX=TO3(IXX,IYY)
         ISEG=IYY
9     CONTINUE
         TB1(IXX)=MMAX
         TB2(IXX)=ISEG
8     CONTINUE
C
C     THE MAXIMUM OF THE SEGMENT TRADEOFFS IS FOUND.
C     IF THE MAXIMUM IS NEGATIVE THEN TERMINATE.
C
100   MAX=-1.0
      SI=0
      DO 10 I=1,KK
         IP=I
         IF(Y(IP).EQ.0) GO TO 10
         IF(TB1(IP).LE.MAX) GO TO 10
         MAX=TB1(IP)
         SI=IP
10    CONTINUE
      IF(MAX.LE.0.0) GO TO 1000
C
C     THE VALIDITY AND THE FEASIBILITY OF THE TRADEOFF IS CHECKED.
C
       SJ=TB2(SI)
       IF(Y(SJ).EQ.0) GO TO 90
       IF((N(SJ)+N(SI)).GT.NMAX) GO TO 90
       IF((TS(SJ)+TS(SI)).GT.TMAX) GO TO 90
       IF(REL(SI,SJ).EQ.0) GO TO 80
       REL(SI,SJ)=0
C
C     THE TRADEOFF BETWEEN THE SEGMENTS SI AND SJ IS RECALCULATED.
C
68     CALL RECOM(SI,SJ,N(SI),N(SJ),NONX,SEG,MIN)
       TO1(SI,SJ)=NONX
```

```
      TC2(SI,SJ)=SEG
      TC3(SI,SJ)=DO(NI(SI),JK)-MIN
      TB1(SI)=TO3(SI,SJ)
      GO TO 100
 90   TO3(SI,SJ)=-1.0
C
C     CALCULATE THE NEW MAXIMUM TRADEOFF FOR SEGMENT SI.
C
 91   XMIN=9000.
      DP=N(SI)
      Y(SI)=0
      IISEG=0
      DO 73 J=1,KK
         IX=J
         IF(Y(IX).EQ.0) GO TO 73
         DE=N(IX)
         IF((DE+DP).GT.NMAX) GO TO 733
         IF((TS(IX)+TS(SI)).GT.TMAX) GO TO 733
      DO 72 I=1,DP
         OO=SS(SI,I)
      DO 74 K=1,DE
       BX=SS(IX,K)
       DDUM=D(OO,BX)
       IF(DDUM.GE.XMIN) GO TO 74
       XMIN=DDUM
       IISEG=IX
       TON=OO
       TOS=BX
 74    CONTINUE
 72    CONTINUE
       GO TO 73
 733   TO3(SI,IX)=-1.0
 73   CONTINUE
       IF(IISEG.NE.0) GO TO 63
       TB1(SI)=-1.0
       GO TO 100
 63    Y(SI)=1
       REL(SI,IISEG)=0
       TB1(SI)=DO(NI(SI),JK)-XMIN
       TB2(SI)=IISEG
       TO1(SI,IISEG)=TON
       TO2(SI,IISEG)=TOS
       TO3(SI,IISEG)=TB1(SI)
       GO TO 100
 80    IF(REL(SJ,SI).EQ.0) GO TO 82
       REL(SJ,SI)=0
       GO TO 68
C
C     THE CENTRAL LINK (SI,JK) IS DROPPED AND THE NON-CENTRAL LINK
C     (IXI,IYI) IS ADDED.
C
 82    XO(SI,JK)=0
       IXI=TO1(SI,SJ)
       IYI=TO2(SI,SJ)
       X(IXI,IYI)=1
       X(IYI,IXI)=1
       COST=COST+D(IXI,IYI)
       PRINT25,NI(IXI),NI(IYI)
       Y(SI)=0
```

```
       TB1(SI)=-1.0
C
C      THE DATA AND VECTORS OF THE SEGMENT ARE UPDATED.
C
       LX=N(SJ)+1
       PX=LX-1
       KX=PX+N(SI)
      DO 99 I=LX,KX
         PP=I-PX
         SS(SJ,I)=SS(SI,PP)
 99    CONTINUE
       N(SJ)=N(SJ)+N(SI)
       TS(SJ)=TS(SJ)+TS(SI)
       IF(N(SJ).EQ.NMAX) GO TO 79
       IF(TS(SJ).EQ.TMAX) GO TO 79
      DO 81 I=1,KK
         IF(Y(I).EQ.0) GO TO 81
         REL(SJ,I)=1
 81    CONTINUE
      GO TO 100
 79    Y(SJ)=0
      TB1(SJ)=-1.0
      GO TO 100
C
C      THE OUTPUT FOR THIS CONCENTRATOR IS GENERATED.
C
 1000 PRINT2001,PPQ
 2001 FORMAT(' ','THE NO. OF ISOLATED NODES ARE',I4,2X,'THEY ARE:')
      IF(PPQ.EQ.0) GO TO 2005
      DO 55 I=1,PPQ
 55      PRINT,OUT(I)
 2005 PRINT2002,JK
 2002 FORMAT(' ','NODES CONNECTED TO CONC.',I4)
      DO 56 J=1,KK
         IF(XO(J,JK).EQ.0) GO TO 56
         PRINT,NI(J)
         COST=COST+DO(NI(J),JK)
 56    CONTINUE
      PRINT2003
 2003 FORMAT(' ','NODE',5X,'TO NODE')
      DO 57 II=1,KK
      DO 57 JJ=1,KK
         IF(X(II,JJ).EQ.0) GO TO 57
         PRINT2004,NI(II),NI(JJ)
 2004 FORMAT(' ',I4,8X,I4)
 57    CONTINUE
      PRINT19,COST
      PRINT30
      DO 2000 I=1,KK
 2000 PRINT20,I,NI(I)
      RETURN
      END
      SUBROUTINE RECOM(IV,JV,II,JJ,NONX,SEG,MIN)
C
C      THIS SUBROUTINE IS CALLED BY SAHC.
C      IT RECALCULATES THE MINIMUM COST CONNECTION BETWEEN TWO TREE
C      SEGMENTS IV AND IJ.
C
      INTEGER*2 SEG,NONX,SS(40,40),IX,IY,IV,JV,II,JJ
```

```
      REAL MIN,DUM,XT(160,2),D(40,40)
      COMMON /AREA4/XT/AREA6/SS,D
C
      MIN=9000.
      DO 31 I=1,II
         IX=SS(IV,I)
      DO 31 J=1,JJ
         IY=SS(JV,J)
         DUM=D(IX,IY)
         IF(DUM.GE.MIN) GO TO 31
           MIN=DUM
           SEG=IY
           NONX=IX
 31   CONTINUE
      RETURN
      END
      BLOCK DATA
C
C     THIS ROUTINE INITIALIZES CERTAIN VARIABLES AND ARRAYS.
C
      REAL COST
      INTEGER*2 TCO(20),N(20),NONC(20,160)
      INTEGER REAS(40,2),LP
      COMMON /AREA11/REAS,LP
      COMMON /AREA3/COST/AREA7/TCO
      COMMON /AREA9/N,NONC
      DATA REAS,LP/80*0,0/
      DATA COST,TCO/0.0,20*0/
      DATA N/20*0/
      END
$ENTRY
```

```
C                    THE BAHL AND TANG ALGORITHM
C     THIS ALGORITHM IS BASICALLY A TWO-LEVEL STAR NETWORK ALGORITHM.
C      IT STARTS WITH A SET OF CANDIDATE CONCENTRATOR SITES AND
C      ASSIGNS TERMINALS TO THEM UNTIL THEIR CAPACITIES ARE FILLED
C      OR UNTIL IT IS NO LONGER ECONOMICAL TO DO SO.
C      SINCE THE ASSIGNMENT IS DONE ON A CONCENTRATOR BASIS RATHER
C      THAN A TERMINAL BASIS, A TERMINAL MAY BE CONNECTED TO MORE THAN
C      ONE CONCENTRATOR. THIS PROVIDES THE BASIS FOR CALCULATING
C      THE TERMINAL AND CONCENTRATOR DEVIATIONS, WHICH IN TURN
C      PROVIDE THE BASIS FOR DELETING OR INSERTING LINKS.
C      THE ALGORITHM ATTEMPTS TO DECREASE THE COSTS AT EACH
C      STAGE BY EITHER INSERTING OR DELETING A LINK (THUS DECREASING
C      THE DEVIATIONS). EACH TIME A LINK IS DROPPED THE CONCENTRATOR
C      IT WAS DROPPED FROM HAS ITS GAIN CHECKED. IF IT IS NEGATIVE THE
C      CONCENTRATOR IS DROPPED.
C      THE ALGORITHM TERMINATES WHEN EACH TERMINAL IS CONNECTED TO
C      ONE AND ONLY ONE CONCENTRATOR OR THE CPU. THIS PRODUCES A STAR
C      NETWORK, WITH TERMINALS HOMING ONTO CONCENTRATORS. WITH THIS
C      PARTITIONING OF THE NETWORK WE CAN OPTIMIZE EACH PARTITION
C      TO PRODUCE A MULTIDROP SUBNET AND THUS ARRIVE AT A TOTAL
C      MULTIDROP NETWORK.
C
C
C      USER SUPPLIED PARAMETERS:
C
C      N= THE NUMBER OF TERMINALS IN THE NETWORK.
C      MMM= THE NETWORK IDENTIFIER.
C      NMAX= THE MAXIMUM NUMBER OF TERMINALS PER MULTIDROP LINE.
C      TMAX= THE MAXIMUM TRAFFIC PER MULTIDROP LINE.
C      E= THE MAXIMUM NUMBER OF TERMINALS PER CONCENTRATOR.
C      B= THE CAPACITY OF A CONCENTRATOR.
C      M= THE NUMBER OF CANDIDATE CONCENTRATOR SITES.
C      FIXC= THE FIXED COST FOR A CONCENTRATOR.
C      XT(I,J)= THE X-Y COORDINATES.
C      C(I)= THE VECTOR OF CONCENTRATOR SITES.
C      T(I)= THE TRAFFIC VECTOR.
C
C
C      DEFINITION OF VARIABLES:
C      NC(I,J)= THE COST MATRIX.
C      G(I,J)= THE GAIN MATRIX
C      F(J)= THE CONCENTRATOR COST VECTOR.
C      TG(I,J)= THE GAIN PER UNIT TRAFFIC MATRIX.
C      H(I)= THE WEIGHTED AVERAGE AUGMENTED COST FOR TERMINAL I
C      CCT(J)= THE SUM OF THE GAINS FOR CONCENTRATOR J
C      NCCT(J)= THE GAIN OF CONCENTRATORJ
C      LC(I,J)= THE AUGMENTED COST MATRIX
C      Q(I,J)= THE LINK DEVIATION MATRIX
C      R(J)= THE CONCENTRATOR DEVIATION MATRIX
C      TCB(I)= THE SUM OF THE AUGMENTED COSTS FOR TERMINAL I
C      TSUM(J)= THE TOTAL TRAFFIC AT CONCENTRATOR J
C      NTT(I)= THE NUMBER OF LINKS FROM TERMINAL I
C      ID(I,J)= THE VECTOR OF TERMINALS ASSIGNED TO CONCENTRATOR J
C      Y(J)= THE STATUS VECTOR FOR THE CONCENTRATORS.
C      X(I,J)= THE NETWORK INCIDENCE MATRIX
C      LSUM= THE NUMBER OF NETWORK LINKS
```

```
C     TC(J)= THE NUMBER OF TERMINALS ASSIGNED TO CONCENTRATOR J
C     INSER(I,J)= THE INHIBIT MATRIX. IF LINK (I,J) HAS BEEN
C                 DROPPED INSER(I,J)=.FALSE. SO THAT IT CAN NEVER
C                 BE INSERTED AGAIN.
C
C
      REAL XT(160,2),NC(160,80),G(160,80),F(80),TG(160,80),H(160)
      REAL CCT(80),NCCT(80),LC(160,80),Q(160,80),R(80),TCB(160)
      REAL FIXC,LD,HD
      INTEGER*2 NMAX,TMAX,E,B,MMM,OUTER,INNER,CLO
      INTEGER*2 TSUM(80),T(160),NTT(160),ID(80,160),Y(80),X(160,80)
      INTEGER*2 NIN(160)/160*0/,TC(80),C(80),IN(160,80)
      INTEGER*2 CN(80),YYZ(160)/160*0/,AY(160)/160*0/,ISS,IX,IIN(40)
      INTEGER IW,W,PW,LSUM,PP,PX,PXX,N,K
      LOGICAL*1 INSER(160,80)
      COMMON /AREA1/XT/AREA2/T/AREA3/N,K/AREA4/NC,H,LC/AREA5/CCT,TC,TSUM
      COMMON /AREA6/NTT/AREA7/Y/AREA8/Q,R/AREA9/ID/AREA10/X/AREA11/
     *INSER/AREA12/G/AREA13/INNER/AREA14/TCB/AREA16/F/AREA17/C/
     *AREA18/TG,IN,CN/AREA19/LSUM/AREA21/FIXC
C
C     READ IN THE PARAMETERS
C
      READ,N,MMM
      READ,FIXC,NMAX,TMAX,E,B,M
 8410  FORMAT('1','********** BAHL & TANG ALGORITHM *********
     @**********')
      PRINT8410
      PRINT8100,MMM
 8100 FORMAT(' ','********** RANDOM NET ',I4,' **********')
      PRINT8210,FIXC,N,NMAX,TMAX,E,B,M
 8210 FORMAT('0','FIXC=',F7.2,' N=',I3,' NMAX=',I3,' TMAX=',I3,' E=',
     *I3,' B=',I3,' M=',I3)
      DO 201 I=1,N
 201     READ,(XT(I,J),J=1,2),T(I)
      PRINT8310
 8310 FORMAT(' ','INITIAL CONCENTRATOR SITES')
      READ,(C(I),I=1,M)
      PRINT,(C(I),I=1,M)
C
C     INITIALIZE THE VARIABLES
C
      K=M
      KM=M
      INNER=0
      OUTER=0
      CLO=0
      COST=0.0
      LSUM=0
C
C     CALCULATE THE COST AND GAIN MATRICES
C
      DO 107 I=1,N
         NC(I,1)=LD(I,1)
      DO 92 II=2,K
         J=C(II)
         NC(I,II)=LD(I,J)
         G(I,II)=NC(I,1)-NC(I,II)
 92   CONTINUE
 107  CONTINUE
```

```
C
C       CALCULATE THE CONCENTRATOR COSTS
C
        DO 91 I=1,K
           J=C(I)
           F(I)=HD(J)
 91     CONTINUE
C
C       CALCULATE THE GAIN PER UNIT TRAFFIC AND SORT IT IN DESCENDING
C       ORDER
C
        DO 94 J=2,K
           L=0
        DO 5 I=1,N
           IF(G(I,J).LT.0.0) GO TO 5
           L=L+1
           TG(L,J)=G(I,J)/T(I)
           IN(L,J)=I
 5      CONTINUE
           TC(J)=L
           CN(J)=L
           CALL SORT(L,J)
 94     CONTINUE
C
C
C       ASSIGN TERMINALS TO THE CONCENTRATORS
C
        DO 7 J=2,K
           IU=TC(J)
           TSUM(J)=T(C(J))
           TC(J)=1
           CCT(J)=G(C(J),J)
           ID(J,1)=C(J)
           X(C(J),J)=1
           NTT(C(J))=NTT(C(J))+1
           LSUM=LSUM+1
           LV=1
        DO 8 IH=1,IU
           IP=IN(IH,J)
           IF(IP.EQ.C(J)) GO TO 8
           II=TSUM(J)+T(IP)
           IF(II.GT.B) GO TO 7
           IF(IH.GT.E) GO TO 7
           TSUM(J)=II
           TC(J)=TC(J)+1
           CCT(J)=CCT(J)+G(IP,J)
           LV=LV+1
           LSUM=LSUM+1
           ID(J,LV)=IP
           X(IP,J)=1
           NTT(IP)=NTT(IP)+1
 8      CONTINUE
 7      CONTINUE
C
C       PRE-PROCESSING.
C       DROP CONCENTRATORS WITH NEGATIVE GAINS.
C       ASSIGN STRANDED TERMINALS TO THE CPU.
C
        DO 11  J=2,K
```

```
      NCCT(J)=CCT(J)-F(J)
      IF(NCCT(J).GT.0.0) GO TO 11
      Y(J)=1
      KM=KM-1
      PRINT4442,J,C(J)
4442  FORMAT(' ','CONC',I4,2X,'DROPPED AT NODE',I4)
      LSUM=LSUM-TC(J)
      L=TC(J)
      DO 12 I=1,L
      II=ID(J,I)
      X(II,J)=0
12    NTT(II)=NTT(II)-1
11    CONTINUE
      LL=0
      TSUM(1)=0
      DO 72 I=1,N
      IF(NTT(I).NE.0) GO TO 72
      LL=LL+1
      ID(1,LL)=I
      LSUM=LSUM+1
      NTT(I)=1
      X(I,1)=1
      TSUM(1)=TSUM(1)+T(I)
72    CONTINUE
      TC(1)=LL
C
C     CHECK IF DONE
C
      IF(KM.EQ.1) GO TO 100
      IF(LSUM.EQ.N) GO TO 100
C
C     CALCULATE THE AUGMENTED COSTS AND DEVIATIONS
C
      DO 14 I=2,N
      DO 15 J=2,K
      IF(Y(J).EQ.1) GO TO 15
      IF(X(I,J).EQ.0) GO TO 15
      LC(I,J)=NC(I,J)+((T(I)*F(J))/TSUM(J))
      TCB(I)=TCB(I)+LC(I,J)
15    CONTINUE
      H(I)=TCB(I)/NTT(I)
14    CONTINUE
      DO 108 I=2,K
      IF(Y(I).EQ.1) GO TO 108
      L=TC(I)
      R(I)=0.0
      DO 109 J=1,L
      IX=ID(I,J)
      Q(IX,I)=LC(IX,I)-H(IX)
      R(I)=R(I)+Q(IX,I)
109   CONTINUE
108   CONTINUE
C
C     CHECK EACH CONCENTRATOR TO SEE IF A LINK CAN BE INSERTED
C     IF SO THEN INSERT THE BEST LINK
C
200   DO 18 J=2,K
      IF(Y(J).EQ.1) GO TO 18
      IF(TC(J).EQ.E) GO TO 18
```

```
          CALL LADD(J,E,B)
 18     CONTINUE
C
C       DETERMINE THE WORST LINK AND DROP IT
C
 9929      CALL RMAX
           CALL QMAX(PP,PX,PXX)
           IW=PX
           W=PP
           PW=PXX
           INSER(IW,W)=.FALSE.
           X(IW,W)=0
           LSUM=LSUM-1
           TSUM(W)=TSUM(W)-T(IW)
           CCT(W)=CCT(W)-G(IW,W)
           ID(W,PW)=ID(W,TC(W))
           TC(W)=TC(W)-1
C
C       CHECK THE GAIN OF THE WORST CONCENTRATOR
C
C       IF THE GAIN IS POSITIVE THEN WE RECALCULATE DEVIATIONS
C       AND CONTINUE. IF THE GAIN IS NEGATIVE THEN WE DROP THE
C       CONCENTRATOR.
C
           GCON=CCT(W)-F(W)
           IF(GCON)22,22,23
C
C       AGAIN WE CHECK IF WE ARE DONE
C
 23        IF(LSUM.EQ.N) GO TO 100
           CALL DEV(IW,W,0)
           INNER=INNER+1
           GO TO 200
 22        OUTER=OUTER+1
           CALL CLOSE(IW,W,KM)
           IF(KM.EQ.1) GO TO 100
           IF(LSUM.EQ.N) GO TO 100
           GO TO 200
 100    PRINT559,KM
 559    FORMAT(' ',' THE NUMBER OF OPEN CONCENTRATORS IS=',I4)
        COST=0.0
        DO 30 J=1,K
           IF(Y(J).EQ.1) GO TO 30
           COST=COST+F(J)
           L=TC(J)
        DO 31 I=1,L
           IX=ID(J,I)
 31        COST=COST+NC(IX,J)
 30     CONTINUE
        PRINT51,COST
 51     FORMAT(' ','NETWORK COST FOR PT. TO PT.=',F9.3)
C
C       INITIALIZE THE NETWORK FOR THE MULTIDROP PORTION.
C
        COST=0.0
        DO 990 I=1,K
           AY(I)=0
           IF(Y(I).EQ.1) GO TO 990
           COST=COST+F(I)
```

```
         TC(I)=0
         YYZ(C(I))=1
         TSUM(I)=T(C(I))
  990  CONTINUE
C
C      ASSIGN TERMINALS TO CONCENTRATORS
C
       DO 987 I=1,N
         IF(YYZ(I).EQ.1) GO TO 987
         CALL AMIN(I,K,ISS)
         TC(ISS)=TC(ISS)+1
         TSUM(ISS)=TSUM(ISS)+T(I)
         NA=TC(ISS)
         IX=I
         ID(ISS,NA)=IX
  987  CONTINUE
C
C      CHECK THE CONCENTRATOR CAPACITY CONSTRAINTS.
C
       DO 984 I=2,K
         IF(Y(I).EQ.1) GO TO 984
         IF(TSUM(I).GT.B) AY(I)=1
  984  CONTINUE
C
C      MODIFY THE TERMINAL ASSIGNMENTS IF NECESSARY.
C
       DO 991 I=2,K
         IF(Y(I).EQ.1) GO TO 991
         IF(AY(I).EQ.0) GO TO 991
         CALL CFUL(I,B,AY,K)
         IF(TSUM(I).LT.B) AY(I)=0
  991  CONTINUE
C
C      REASSIGN THE DROPPED TERMINALS
C
         CALL REAR(K,B,AY)
C
C      CHECK IF ANY MORE CONCENTRATORS CAN BE DROPPED DUE
C      TO THE ASSIGNMENT MODIFICATIONS
C
       DO 5453 I=2,K
         IF(Y(I).EQ.1) GO TO 5453
         IF(TC(I).NE.0) GO TO 5453
         COST=COST-F(I)
         Y(I)=1
         IX=C(I)
         CALL AAMIN(K,IX,ISS,B,AY)
         TC(ISS)=TC(ISS)+1
         TSUM(ISS)=TSUM(ISS)+T(IX)
         ID(ISS,TC(ISS))=IX
 5453  CONTINUE
       DO 5553 J=1,K
         IF(Y(J).EQ.1) GO TO 5553
         PRINT1422,J,C(J),TC(J),TSUM(J)
 1422  FORMAT(' ','CONC',I3,' AT NODE',I3,' HAS ',I3,' NODES AND ',I3,'
      *UNITS OF TRAFFIC')
       PRINT1471
 1471  FORMAT(' ','TERMINAL SET')
       NA=TC(J)
```

```
      IF(NA.EQ.0) GO TO 5553
      PRINT,(ID(J,L),L=1,NA)
 5553 CONTINUE
C
C     OPTIMIZE EACH CONCENTRATOR SUBNET
C
      DO 111 I=1,K
         IF(Y(I).EQ.1) GO TO 111
         NA=TC(I)
         IF(NA.EQ.0) GO TO 111
      DO 121 J=1,NA
  121    IIN(J)=ID(I,J)
         CALL SAHC(NA,I,TMAX,NMAX,IIN,COST)
  111 CONTINUE
      PRINT7004,COST
 7004 FORMAT(' ',' NEW COST FOR MULTIDROPPED NET IS',F8.3)
      STOP
      END
C
C
      SUBROUTINE CLOSE(IW,W,KM)
      REAL NC(160,80),H(160),LC(160,80),CCT(80),Q(160,80),R(80),TCB(160)
      INTEGER*2 ID(80,160),TC(80),TSUM(80),NTT(160),Y(80),X(160,80),
     *C(80),T(160)
      INTEGER N,K,IW,W,KM,LSUM
      COMMON /AREA3/N,K/AREA4/NC,H,LC/AREA5/CCT,TC,TSUM/AREA6/NTT/
     *AREA7/Y/AREA8/Q,R/AREA9/ID/AREA10/X/AREA14/TCB/AREA17/C
      COMMON /AREA19/LSUM/AREA2/T
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT DROPS THE WORST CONCENTRATOR AND MAKES THE ADJUSTMENTS
C     TO THE DEVIATIONS.
C
      Y(W)=1
      KM=KM-1
      L=TC(W)
      LSUM=LSUM-L
      KZ=TC(W)+1
      ID(W,KZ)=IW
      DO 1 J=1,KZ
         IX=ID(W,J)
         NTT(IX)=NTT(IX)-1
         IF(IX.EQ.IW) GO TO 9
    9    X(IX,W)=0
         IF(NTT(IX).NE.0) GO TO 3
C
C     IF A NODE BECOMES STRANDED IT IS ASSIGNED TO THE CPU.
C
         NTT(IX)=1
         X(IX,1)=1
         TCB(IX)=0.0
         H(IX)=0.0
         TC(1)=TC(1)+1
         TSUM(1)=TSUM(1)+T(IX)
         LSUM=LSUM+1
         ID(1,TC(1))=IX
         GO TO 1
    3    TCB(IX)=TCB(IX)-LC(IX,W)
C
```

```
C       THE NECESSARY ADJUSTMENTS OF THE DEVIATIONS AFFECTED BY
C       THE DROPPING OF THE CONCENTRATOR LINKS ARE MADE.
C
           H(IX)=TCB(IX)/NTT(IX)
        DO 2 I=2,K
           IF(Y(I).EQ.1) GO TO 2
           IF(X(IX,I).EQ.0) GO TO 2
           R(I)=R(I)-Q(IX,I)
           Q(IX,I)=LC(IX,I)-H(IX)
           R(I)=R(I)+Q(IX,I)
 2      CONTINUE
 1      CONTINUE
        RETURN
        END
C
C
        SUBROUTINE QMAX(PP,PX,PXX)
        REAL G(160,80),Q(160,80),R(80),CCT(80),LARGE
        INTEGER*2 ID(80,160),TSUM(80),TC(80),NTT(160),X(160,80)
        INTEGER PP,PX,PXX,JW(10),PL
        INTEGER IQ(10,3)
        COMMON /AREA5/CCT,TC,TSUM/AREA6/NTT/AREA8/Q,R/AREA9/ID/AREA15/
      *PL,JW
        COMMON /AREA10/X/AREA12/G
C
C       THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C       IT DETERMINES THE LINK (CONNECTED TO A WORST CONCENTRATOR)
C       WHICH HAS THE WORST LINK DEVIATION
C
        LARGE=-10000.
C
C       THE STARTING VALUE IS THE FIRST ONE THAT SATISFIES THE
C       CRITERIA THAT THE TERMINAL HAVE MORE THAN ONE LINK AND IS
C       CONNECTED TO ONE OF THE WORST CONCENTRATORS.
C
        DO 1 I=1,PL
           JJ=JW(I)
           L=TC(JJ)
        DO 2 II=1,L
           IX=ID(JJ,II)
           IF(NTT(IX).EQ.1) GO TO 2
           IF(Q(IX,JJ).LE.LARGE) GO TO 2
             LARGE=Q(IX,JJ)
             PP=JJ
             PX=IX
             PXX=II
 2       CONTINUE
 1       CONTINUE
         IQ(1,1)=PX
         IQ(1,2)=PP
         IQ(1,3)=PXX
         IIL=1
C
C       THE SET OF WORST LINKS IS FOUND. ALL LINKS WITH THE SAME
C       WORST DEVIATION ARE CONSIDERED.
C
        DO 3 I=1,PL
           JJ=JW(I)
           L=TC(JJ)
```

```
      DO 4 II=1,L
         IX=ID(JJ,II)
         IF(NIT(IX).EQ.1) GO TO 4
         IF(IX.EQ.PX.AND.JJ.EQ.PP) GO TO 4
         IF(Q(IX,JJ).NE.LARGE) GO TO 4
         IIL=IIL+1
         IQ(IIL,1)=IX
         IQ(IIL,2)=JJ
         IQ(IIL,3)=II
 4    CONTINUE
 3    CONTINUE
      IF(IIL.EQ.1) GO TO 9
      XMIN=10000.
C
C     FROM THE SET OF WORST LINKS THE ONE WITH THE WORST GAIN
C     IS SELECTED AS THE WORST LINK.
C
      DO 5 J=1,IIL
         IX=IQ(J,1)
         IJ=IQ(J,2)
         IP=IQ(J,3)
         IF(G(IX,IJ).GE.XMIN) GO TO 5
           XMIN=G(IX,IJ)
           PP=IJ
           PX=IX
           PXX=IP
 5    CONTINUE
 9    RETURN
      END
C
C
      SUBROUTINE RMAX
      REAL R(80),Q(160,80),LARGE
      INTEGER*2 Y(80)
      INTEGER JW(10),PL,P,N,K
      COMMON /AREA3/N,K/AREA7/Y/AREA8/Q,R/AREA15/PL,JW
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT DETERMINES THE SET OF WORST CONCENTRATORS.
C
      PL=1
C
C     THE STARTING CONCENTRATOR DEVIATION IS FOUND.
C
      DO 1 I=2,K
         IF(Y(I).EQ.1) GO TO 1
         IF(R(I).EQ.0.0) GO TO 1
           P=I
           II=I
           LARGE=R(I)
           GO TO 2
 1    CONTINUE
C
C     THE WORST CONCENTRATOR DEVIATION IS FOUND
C
 2    DO 3 J=II,K
         IF(Y(J).EQ.1) GO TO 3
         IF(R(J).EQ.0.0) GO TO 3
         IF(R(J).LE.LARGE) GO TO 3
```

```
          LARGE=R(J)
          P=J
 3    CONTINUE
      JW(1)=P
C
C     ALL CONCENTRATORS WITH THE SAME DEVIATION ARE CONSIDERED.
C
      DO 4 J=2,K
         IF(Y(J).EQ.1) GO TO 4
         IF(R(J).EQ.0.0) GO TO 4
         IF(J.EQ.P) GO TO 4
         IF(R(J).NE.LARGE) GO TO 4
           PL=PL+1
           JW(PL)=J
 4    CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DEV(II,JJ,P)
      REAL F(80),H(160),LC(160,80),R(80),Q(160,80),TCB(160),NC(160,80),
     *CCT(80)
      INTEGER*2 X(160,80),T(160),TSUM(80),NTT(160),ID(80,160),TC(80),
     *Y(80)
      INTEGER N,P,K
      INTEGER LSUM
      COMMON /AREA2/T/AREA3/N,K/AREA4/NC,H,LC/AREA5/CCT,TC,TSUM/AREA6
     */NTT/AREA7/Y/AREA8/Q,R/AREA9/ID/AREA10/X/AREA14/TCB/AREA16/F
      COMMON /AREA19/LSUM
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM,
C     AND BY LADD.
C     IT RECALCULATES THE NECESSARY AUGMENTED COSTS AND
C     DEVIATIONS WHENEVER A LINK IS ADDED OR DROPPED.
C     P=1 IMPLIES ADD
C
      XX=F(JJ)/TSUM(JJ)
      IF(P.EQ.1) GO TO 3
C
C     DROPPED LINK
C
      NTT(II)=NTT(II)-1
      IF(NTT(II).EQ.0) GO TO 9
      TCB(II)=TCB(II)-LC(II,JJ)
      H(II)=TCB(II)/NTT(II)
      R(JJ)=R(JJ)-Q(II,JJ)
      Q(II,JJ)=0.0
      GO TO 4
 9    NTT(II)=1
      X(II,1)=1
      LSUM=LSUM+1
      TC(1)=TC(1)+1
      TSUM(1)=TSUM(1)+T(II)
      ID(1,TC(1))=II
      R(JJ)=R(JJ)-Q(II,JJ)
      Q(II,JJ)=0.0
      TCB(II)=0.0
      H(II)=0.0
      GO TO 7
```

```
C
C       LINK ADDED
C
 3      NTT(II)=NTT(II)+1
        LC(II,JJ)=NC(II,JJ)+T(II)*XX
        IF(NTT(II).NE.1) GO TO 444
        TCB(II)=LC(II,JJ)
        H(II)=TCB(II)
        Q(II,JJ)=0.0
        GO TO 7
 444    TCB(II)=TCB(II)+LC(II,JJ)
        H(II)=TCB(II)/NTT(II)
        Q(II,JJ)=LC(II,JJ)-H(II)
        R(JJ)=R(JJ)+Q(II,JJ)
C
C       RECALCULATE DEVIATION CHANGES AT NODE II
C
 4      DO 1 JL=2,K
           IF(Y(JL).EQ.1) GO TO 1
           IF(JL.EQ.JJ) GO TO 1
           IF(X(II,JL).EQ.0) GO TO 1
           R(JL)=R(JL)-Q(II,JL)
           IF(NTT(II).NE.1) GO TO 445
           Q(II,JL)=0.00
           H(II)=TCB(II)
           GO TO 7
 445       Q(II,JL)=LC(II,JL)-H(II)
           R(JL)=R(JL)+Q(II,JL)
 1      CONTINUE
 7      IL=TC(JJ)
        DO 2 L=1,IL
           LL=ID(JJ,L)
           IF(LL.EQ.II) GO TO 2
           IF(NTT(LL).NE.1) GO TO 22
           LC(LL,JJ)=NC(LL,JJ)+T(LL)*XX
           TCB(LL)=LC(LL,JJ)
           H(LL)=TCB(LL)
           R(JJ)=R(JJ)-Q(LL,JJ)
           Q(LL,JJ)=0.00
           GO TO 2
 22        TCB(LL)=TCB(LL)-LC(LL,JJ)
           LC(LL,JJ)=NC(LL,JJ)+T(LL)*XX
           TCB(LL)=TCB(LL)+LC(LL,JJ)
           H(LL)=TCB(LL)/NTT(LL)
C
C       RECALCULATE DEVIATION FOR OTHER NODES OF CONC JJ
C
        DO 33 JX=2,K
           IF(Y(JX).EQ.1) GO TO 33
           IF(X(LL,JX).EQ.0) GO TO 33
           R(JX)=R(JX)-Q(LL,JX)
           Q(LL,JX)=LC(LL,JX)-H(LL)
           R(JX)=R(JX)+Q(LL,JX)
 33     CONTINUE
 2      CONTINUE
        RETURN
        END
C
C
```

```
      SUBROUTINE LADD(J,E,B)
      REAL NC(160,80),H(160),LC(160,80),F(80),G(160,80),CCT(80),R(80),
     *Q(160,80),TG(160,80)
      REAL MAX
      INTEGER N,K,LSUM
      INTEGER*2 X(160,80),TSUM(80),TC(80),T(160),ID(80,160),E,B,INNER
      INTEGER*2 C(80),NTT(160),CN(80),IN(160,80),IK,IKK
      LOGICAL*1 INSER(160,80)
      COMMON /AREA2/T/AREA3/N,K/AREA5/CCT,TC,TSUM/AREA8/Q,R/AREA9/ID
     */AREA10/X/AREA11/INSER/AREA12/G/AREA13/INNER/AREA19/LSUM
      COMMON /AREA4/NC,H,LC/AREA16/F/AREA17/C
      COMMON /AREA6/NTT
      COMMON /AREA18/TG,IN,CN
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT DETERMINES IF A LINK CAN BE INSERTED AT A CONCENTRATOR J
C     IF ONE IS INSERTED THEN THE DEVIATIONS AFFECTED ARE
C     CREATED AND RECOMPUTED.
C
      LQ=CN(J)
      LR=1
C
C     FIND THE LINK WITH THE LARGEST GAIN
C
  120 MAX=1.0
      DO 12 I=LR,LQ
         IK=IN(I,J)
         IF(X(IK,J).EQ.1) GO TO 12
         IF(.NOT.INSER(IK,J)) GO TO 12
         IF(TSUM(J)+T(IK).GT.8) GO TO 12
         LRR=I+1
         MAX=0.0
         GO TO 122
   12 CONTINUE
  122 IF(MAX.EQ.1.0) GO TO 7
      IF(X(IK,1).EQ.0) GO TO 2
C
C     DROP A LINK TO THE CPU AND ADD ONE TO CONCENTRATOR J
C
      X(IK,J)=1
      X(IK,1)=0
      NTT(IK)=NTT(IK)-1
      LL=TC(1)
      DO 44 II=1,LL
         IX=ID(1,II)
         IF(IK.EQ.IX) GO TO 45
   44 CONTINUE
   45 ID(1,II)=ID(1,LL)
      TC(1)=TC(1)-1
      TSUM(1)=TSUM(1)-T(IK)
      GO TO 3
    2 IT=TSUM(J)+T(IK)
      XLC=NC(IK,J)+((T(IK)*F(J))/IT)
      IF(XLC-H(IK).GE.0.0) GO TO 121
C
C     ADD A LINK TO CONCENTRATOR J FROM TERMINAL IK
C
      X(IK,J)=1
      LSUM=LSUM+1
```

```
 3     TC(J)=TC(J)+1
       CCT(J)=CCT(J)+G(IK,J)
       TSUM(J)=TSUM(J)+T(IK)
       INNER=INNER+1
       ID(J,TC(J))=IK
       IIK=IK
C
C      RECALCULATE THE NECESSARY DEVIATIONS
C
       CALL DEV(IIK,J,1)
       IF(TC(J).EQ.E) GO TO 7
       IF(TSUM(J).EQ.B) GO TO 7
 121   IF(LRR.GT.LQ) GO TO 7
       LR=LRR
       GO TO 120
 7     RETURN
       END
C
C
       SUBROUTINE SORT(L,J)
       REAL LARGE,TG(160,80)
       INTEGER*2 ZP,P,IN(160,80),CN(80)
       COMMON /AREA18/TG,IN,CN
C
C      THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C      IT SORTS THE GAINS PER UNIT TRAFFIC IN DESCENDING ORDER
C
       DO 6 I=1,L
          CALL MAX(I,L,J,P,ZP,LARGE)
          TG(ZP,J)=TG(I,J)
          IN(ZP,J)=IN(I,J)
          TG(I,J)=LARGE
          IN(I,J)=P
 6     CONTINUE
        RETURN
        END
C
C
       SUBROUTINE MAX(I,L,J,P,ZP,LARGE)
       REAL LARGE,TG(160,80)
       INTEGER*2 ZP,P,MO,IN(160,80),CN(80)
       COMMON /AREA18/TG,IN,CN
C
C      THIS SUBROUTINE IS CALLED BY SORT
C      IT FINDS THE LARGEST TG(I,J) FOR A CONCENTRATOR J
C
        LARGE=TG(I,J)
        P=IN(I,J)
        ZP=I
        DO 6 MO=I,L
          IF(TG(MO,J).LE.LARGE) GO TO 6
          LARGE=TG(MO,J)
          P=IN(MO,J)
          ZP=MO
 6     CONTINUE
       RETURN
       END
C
C
```

```
      BLOCK DATA
      REAL TCB(160)
      INTEGER*2 NTT(160),X(160,80),Y(80)
      INTEGER REAS(40,2),LP
      LOGICAL*1 INSER(160,80)
      COMMON /AREA6/NTT/AREA7/Y/AREA10/X/AREA11/INSER/AREA14/TCB
      COMMON /AREA23/REAS,LP
C
C     THIS ROUTINE INITIALIZES SOME VARIABLES.
C
      DATA REAS,LP/80*0,0/
      DATA X,INSER,Y,TCB,NTT/12800*0,12800*T,80*0,160*0.,160*0/
      END
C
C
C     THE FOLLOWING ROUTINES USED BY THIS PROGRAM ARE THE SAME
C     AS THOSE USED IN NEWCLUST AND MARTIN-DOLL AND ARE NOT
C     LISTED HERE.
C
C                  FUNCTION LD          FUNCTION HD
C                  SAHC                 RECOM
C                  AAMIN                AMIN
C                  CFUL                 FMI
C                  REAR
C
C
$ENTRY
```

```
C
C
C                          THE NEWCLUST ALGORITHM
C
C     THIS ALGORITHM DESIGNS A MULTIDROP TWO-LEVEL NETWORK.
C     IT STARTS WITH M CANDIDATE CONCENTRATOR SITES AND PARTITIONS THE
C     NT TERMINALS OF THE NETWORK USING THESE M SITES.
C     EACH PARTITION IS OPTIMIZED TO FORM A MULTIDROP SUBNETWORK OR
C     A TREE NETWORK.
C     THE TERMINALS OF EACH SUBNETWORK ARE THEN PARTITIONED
C     INTO CLUSTERS OF TERMINALS (NODES), WHICH ARE THE NODES ON A BRANCH
C     OF THE CONCENTRATOR TREE SUBNET. THESE CLUSTERS OF NODES ARE
C     CALLED SUPER NODES.
C     COST SAVINGS CAN BE REALIZED BY MERGING AND/OR CONNECTING A
C     CONCENTRATORS' SUPER NODES TO OTHER CONCENTRATORS AND DROPPING
C     THAT CONCENTRATOR. CONCENTRATORS ARE DROPPED FROM THE NETWORK,
C     SEQUENTIALLY, UNTIL NO FURTHER COST SAVING CAN BE REALIZED.
C
C
      REAL TNN1(60),XT(160,2),FF(80),DD(160,80),CST(80),RECOST(80),
     *GAIN(80),COST,LD,HD,PMIN,COST1/0.0/
      INTEGER*2 M,NT,NMAX,TMAX,MAXCON,LMAX,IZZ,JX,ISS,NA,KM,LZQ,NLZQ,IXY
      INTEGER*2 LL,IRN,IR,IWI,JW,IVI,JSTAR,ISA,IPI,IFF,JVW
      INTEGER*2 IXP,NCC,LI,MMX,MX,IST,IIST,NXN,LO,QST,IQST,MJ,IMQ,XNX,
     *NX,JV,JLAST,IX
      INTEGER*2 C(80),Y(80)/80*0/,NNCON(80),IN(40),NN(60,6)
     *,YN(60),NON(60,10),NCONC(80),TCONC(80),CONC(80,40),T(160)
     *,LCONC(80),TNN(60,10),REAS(40)/40*0/,DEL(80)/80*0/,TDEL(80)/80*0/,
     *AY(80)/80*0/,YYZ(160)/160*0/
      LOGICAL FLAG,SEC
      COMMON /AREA1/DD,T/AREA2/XT/AREA3/C/AREA4/MAXCON,LMAX/AREA5/NNCON
      COMMON /AREA6/NON/AREA7/LCONC,NN,TCONC/AREA8/NCONC,CONC,CST,LL
      COMMON /AREA11/COST/AREA13/YN/AREA14/TNN/AREA15/GAIN/AREA16/TNN1
      COMMON /AREA18/SEC/AREA19/FIXC
C
C
C     USER DEFINED PARAMETERS:
C
C     NT= THE NUMBER OF TERMINALS OR NODES IN THE NETWORK.
C     MMM= THE NETWORK IDENTIFYING NUMBER.
C     FIXC= THE FIXED CONCENTRATOR COSTS.
C     NMAX= THE MAXIMUM NUMBER OF TERMINALS PER MULTIDROP LINE.
C     TMAX= THE MAXIMUM AMMOUNT OF TRAFFIC PER MULTIDROP LINE.
C     MAXCON= THE CAPACITY OF A CONCENTRATOR.
C     LMAX= THE MAXIMUM NUMBER OF LINES PER CONCENTRATOR ( NOT USED).
C     M= THE NUMBER OF INITIAL CANDIDATE CONCENTRATOR SITES.
C     C(*)= THE VECTOR OF THE CANDIDATE CONCENTRATOR SITES.
C
      READ,NT,MMM
      READ,FIXC,NMAX,TMAX,LMAX,MAXCON,M
      PRINT4040
 4040  FORMAT('1','****************** NEWCLUST ************************')
      PRINT4004,MMM
 4004  FORMAT(' ','****************** RANDOM NET',I3,' ***************')
      PRINT1,FIXC,NT,NMAX,TMAX,LMAX,MAXCON,M
 1    FORMAT('0','FIXC=',F7.2,' NT=',I3,' NMAX=',I3,' TMAX=',I3,'LMAX=
     *',I3,'  MAXCON=',I3,'  M=',I3)
C
C     INPUT THE TERMINAL X AND Y COORDINATES.
```

```
C
      DO 3 I=1,NT
         READ,(XT(I,J),J=1,2),T(I)
  3   CONTINUE
C
C     INPUT THE SET OF CANDIDATE CONCENTRATOR SITES.
C
      READ,(C(I),I=1,M)
      DO 188 I=1,M
         II=C(I)
 188     YYZ(II)=1
      PRINT70
 70   FORMAT(' ',/CONCENTRATOR LOCATIONS ARE AT NODES :/)
      PRINT,(C(I),I=1,M)
C
C     CALCULATE THE CONCENTRATOR COSTS AND THE TERMINAL TO CONCENTRATOR
C     COSTS.
C
      SEC=.FALSE.
      JLAST=0
      DO 9 I=1,M
         IZZ=C(I)
         FF(I)=HD(IZZ)
      DO 9 J=1,NT
         IX=J
         DO(J,I)=LD(IX,IZZ)
  9   CONTINUE
C
C     THE INITIAL ASSIGNMENT IS DONE BY ASSIGNING TERMINALS TO THEIR
C     CLOSEST CONCENTRATOR,WITHOUT CONSIDERING THE TRAFFIC.
C     IF THE CONCENTRATOR CAPACITY CONSTRAINT OF ANY CONCENTRATOR
C     BECOMES VIOLATED THEN THAT CONCENTRATOR IS FLAGGED (Y(*)=1).
C
 3222 CONC(1,1)=1
      NNCON(1)=1
      TCONC(1)=T(1)
      DO 1222 I=2,M
         NNCON(I)=0
 1222    TCONC(I)=T(C(I))
      DO 10 I=1,NT
         IF(YYZ(I).EQ.1) GO TO 10
         CALL AMIN(I,M,ISS,Y)
         NNCON(ISS)=NNCON(ISS)+1
         NA=NNCON(ISS)
         CONC(ISS,NA)=I
         TCONC(ISS)=TCONC(ISS)+T(I)
 10   CONTINUE
      DO 1042 I=1,M
         IF(Y(I).EQ.1) GO TO 1042
         IF(TCONC(I).GT.MAXCON) AY(I)=1
 1042 CONTINUE
C
C     CONCENTRATORS THAT HAVE BEEN FLAGGED HAVE THEIR TERMINAL
C     ASSIGNMENTS MODIFIED. THIS IS DONE BY DEFINING A TRADEOFF FOR
C     EACH TERMINAL IN THE SUBNET AND DROPPING THOSE WITH THE MINIMUM
C     ONES UNTIL THE CAPACITY CONSTRAINT IS SATISFIED.
C
      DO 1147 I=2,M
         IF(Y(I).EQ.1) GO TO 1147
```

```
         IF(AY(I).EQ.0) GO TO 1147
         CALL CFUL(I,Y,AY,M)
         IF(TCONC(I).LT.MAXCON) AY(I)=0
 1147 CONTINUE
C
C      THE TERMINALS THAT HAVE BEEN DROPPED FROM THE CONCENTRATORS ARE
C      NOW REASSIGNED TO OTHER CONCENTRATORS OR THE CPU.
C
       CALL REAR(M,Y,AY)
       DO 1212 J=1,M
 1212     AY(J)=0
       KM=M
C
C      KM= THE NUMBER OF CONCENTRATORS THAT ARE PRESENTLY OPEN.
C
C
C      IF NO NODES ARE ASSIGNED TO A CONCENTRATOR THEN THE CONCENTRATOR
C      IS CLOSED AND ITS NODE IS ASSIGNED TO SOME OTHER CONCENTRATOR THAT
C      CAN HANDLE IT OR TO THE CPU.
C
       DO 12 J=2,M
          IF(Y(J).EQ.1) GO TO 12
          IF(NNCON(J).NE.0) GO TO 12
          Y(J)=1
          YYZ(C(J))=0
          CALL AAMIN(M,C(J),ISS,MAXCON,Y,AY)
          NNCON(ISS)=NNCON(ISS)+1
          NA=NNCON(ISS)
          CONC(ISS,NA)=C(J)
          TCONC(ISS)=TCONC(ISS)+T(C(J))
          KM=KM-1
  12   CONTINUE
C
C      IF THERE IS ONLY ONE CONCENTRATOR LEFT (THE CPU) THEN TERMINATE.
C
       IF(KM.EQ.1) GO TO 1110
C
C      EACH PARTITION IS OPTIMIZED INTO A MULTIDROP SUBNET WORK WITH A
C      CONCENTRATOR AS THE HOMING OR SINK NODE.
C      THE TERMINALS OR NODES OF EACH SUBNET ARE THEN PARTITIONED INTO
C      SUPER NODES.
C
       DO 14 I=1,M
          IF(Y(I).EQ.1) GO TO 14
          LZQ=I
          NLZQ=NNCON(I)
          COST=COST+FF(LZQ)
          IF(NLZQ.EQ.0) GO TO 14
       DO 15 J=1,NLZQ
  15      IN(J)=CONC(LZQ,J)
          PRINT 7110,LZQ
 7110     FORMAT('0','CONCENTRATORS',I4,' SUBNETWORK')
          CALL SAHC(NLZQ,LZQ,TMAX,NMAX,IN)
       DO 1414 II=1,NLZQ
 1414       IN(II)=0
  14   CONTINUE
       IF(SEC) GO TO 3199
       DO 6000 I=2,M
          IF(Y(I).EQ.1) GO TO 6000
```

```
          IFF=I
          IPI=NCONC(IFF)
          RECOST(IFF)=0.0
C
C      THE VARIABLES FOR EACH SUPER NODE ARE CALCULATED AND THE GAIN FOR
C      EACH CONCENTRATOR IS CALCULATED.
C
       DO 6001 J=1,IPI
          ISA=CONC(IFF,J)
          CALL CLOSET(ISA,IFF,TMAX,NMAX,M,Y,PMIN)
          RECOST(IFF)=RECOST(IFF)+PMIN
 6001  CONTINUE
       GAIN(IFF)=(CST(IFF)+FF(IFF))-RECOST(IFF)
 6000  CONTINUE
C
C
C      THE CONCENTRATOR WITH THE HIGHEST GAIN IS FOUND AND CONSIDERED FOR
C      DROPPING.
C      IF IT HAS NOT BEEN CHECKED FOR FEASIBILITY IT IS CHECKED.
C      IF IT HAS BEEN CHECKED AND NO CHANGES HAVE TAKEN PLACE OR IF
C      IT HAD BEEN THE ONE LAST CHECKED THEN IT IS DROPPED.
C
C
 3300  CALL MAX(M,Y,HIGH,JW)
       IF(HIGH.LE.0) GO TO 9991
       IF(JLAST.EQ.JW) GO TO 81
       CALL FEAS(M,JW,MAXCON,TMAX,NMAX,FLAG,JLAST,Y)
       IF(.NOT.FLAG) GO TO 3300
 81    Y(JW)=1
       YYZ(C(JW))=0
C
C
C      THE CONCENTRATOR IS DROPPED AND THE GAIN IS SUBTRACTED FROM THE
C      COST. THE SUPER NODES OF THE DROPPED CONCENTRATOR ARE NOW ASSIGNED
C      TO THEIR NEW CONCENTRATOR SUBNETS.
C
C
       COST=COST-HIGH
       KM=KM-1
       NXN=NCONC(JW)
       DO 7000 IXX=1,NXN
          IR=CONC(JW,IXX)
          IF(YN(IR).EQ.1) YN(IR)=0
          IRN=TNN(IR,4)
C
C
C      THOSE CONCENTRATORS THAT HAVE A SUPER NODE DIRECTLY ASSIGNED TO
C      THEM ARE FLAGGED.
C
          IF(DEL(IRN).EQ.0) DEL(IRN)=1
          NN(IR,3)=IRN
          IF(YN(IR).EQ.3) GO TO 45
          NN(IR,4)=TNN(IR,2)
          TCONC(IRN)=TCONC(IRN)+NN(IR,2)
 45       NCONC(IRN)=NCONC(IRN)+1
          NA=NCONC(IRN)
          CONC(IRN,NA)=IR
          IF(YN(IR).EQ.3) GO TO 7000
          IF(TNN(IR,5).NE.0) GO TO 7002
```

```
          YN(IR)=3
          MMX=TNN(IR,1)
          NX=NN(MMX,1)
          NN(MMX,1)=NN(MMX,1)+NN(IR,1)
          NN(MMX,2)=NN(MMX,2)+NN(IR,2)
          MX=NN(MMX,1)
          XNX=NX+1
C
C      SUPER NODES THAT HAVE HAD A SUPER NODE MERGED TO THEM HAVE THEIR
C      PARAMETERS UPDATED AND THEIR VARIABLES RECALCULATED.
C
       DO 4078 I=XNX,MX
          II=I-NX
 4078     NON(MMX,I)=NON(IR,II)
          NN(IR,5)=TNN(IR,3)
          NN(IR,6)=TNN(IR,1)
       PRINT9075,TNN(IR,2),TNN(IR,3)
 9075  FORMAT(' ','X(',I4,',',I4,')=1 *')
       PRINT9023,IR,MMX
 9023  FORMAT(' ','NEW NODE',I3,2X,'MERGED WITH NEW NODE',I4)
          IF(IRN.EQ.1) GO TO 7000
          GAIN(IRN)=GAIN(IRN)+TNN1(MMX)
          CALL CLOSET(MMX,IRN,TMAX,NMAX,M,Y,PMIN)
          GAIN(IRN)=GAIN(IRN)-PMIN
          GO TO 7000
 7002  IF(TDEL(IRN).EQ.0) TDEL(IRN)=1
C
C      THE GAIN OF THE CONCENTRATORS OF THESE SUPER NODES IS ADJUSTED
C      DUE TO ANY POSSIBLE CHANGE IN THEIR NEXT CLOSEST CONCENTRATOR .
C
C      IF A SUPER NODE HAS BEEN CONNECTED DIRECTLY TO A CONCENTRATOR THEN
C      THE GAIN OF THE CONCENTRATOR IS ADJUSTED TO REFLECT IT AND AN
C      ATTEMPT IS MADE TO REOPTIMIZE THE CONCENTRATOR SUBNET.
C
       LCONC(IRN)=LCONC(IRN)+1
       IF(IRN.EQ.1) GO TO 7000
          GAIN(IRN)=GAIN(IRN)+TNN1(IR)
          CALL CLOSET(IR,IRN,TMAX,NMAX,M,Y,PMIN)
          GAIN(IRN)=GAIN(IRN)-PMIN
 7000  CONTINUE
       DO 7001 I=1,M
          IF(Y(I).EQ.1) GO TO 7001
          IF(TDEL(I).EQ.0) GO TO 7001
          IX=I
          CALL MINSA(IX,C(I),NMAX,TMAX,M,Y)
 7001  CONTINUE
       DO 7801 J=1,M
 7801     TDEL(J)=0
       IF(KM.EQ.1) GO TO 1110
C
C      DUE TO THE FACT THAT CONCENTRATOR JW HAS BEEN DROPPED AND SOME OF
C      THE SUPER NODES,OTHER THAN THE ONES THAT WERE CONNECTED TO JW,
C      HAD JW AS THEIR NEXT CLOSEST CONCENTRATOR,THE VARIABLES AND
C      ASSOCIATED CONCENTRATOR GAINS OF THESE SUPER NODES MUST BE
C      RECALCULATED AND ADJUSTED.
C
C
       DO 8001 LI=1,LL
          IF(TNN(LI,4).NE.JW) GO TO 8001
```

```
         IF(YN(LI).EQ.3) GO TO 8001
         JVW=NN(LI,3)
         GAIN(JVW)=GAIN(JVW)+TNNI(LI)
         CALL CLOSET(LI,JVW,TMAX,NMAX,M,Y,PMIN)
         GAIN(JVW)=GAIN(JVW)-PMIN
 8001 CONTINUE
      GO TO 3300
 1110 PRINT1111
 1111 FORMAT(' ',*NO CONCENTRATORS ARE FEASIBLE*)
C
C     THE FINAL TOPOLOGY IS OUTPUT SINCE IT IS LESS COSTLY IN TERMS OF
C     STORAGE,THAN TO RETAIN IT IN MEMORY FOR COMPARISON PURPOSES WITH
C     THE REINITIALIZED NETWORK.
C
C
 9991  PRINT8882,COST
 8882 FORMAT(' ',*THE  COST OF THE FINAL NETWORK IS=*,F8.3)
      PRINT8983,KM
 8983 FORMAT(' ',*NUMBER OF OPEN CONCENTRATORS IS*,I4)
      PRINT8883
 8883 FORMAT(' ',*THE FINAL CONCENTRATORS ARE AT THE TERMINALS:*)
      DO 2277 I=1,M
         IF(Y(I).EQ.1) GO TO 2277
         PRINT,C(I)
 2277 CONTINUE
      DO 336 I=1,M
         IF(Y(I).EQ.1) GO TO 336
         IXY=C(I)
         PRINT8884,IXY
 8884 FORMAT(' ',*THE CONCENTRATOR AT TERMINAL*,I4,2X,*HAS THE FOLLOWING
     * SUPER NODES:*)
         IXP=NCONC(I)
      DO 337 J=1,IXP
         NA=CONC(I,J)
         PRINT8886,NA
         IF(YN(NA).EQ.3) PRINT8086
 8086    FORMAT('+',40X,*MERGED SUPER NODE*)
 8886    FORMAT(' ',30X,I4)
 337  CONTINUE
         PRINT777,IXY,IXP,LCONC(I),TCONC(I)
 777     FORMAT(' ',*THE CONCENTRATOR AT TERMINAL*,I4,2X,*HAS*,I4,2X
     *,*SUPER NODES AND*,I4,2X,*LINES AND*,I4,2X,*UNITS OF TRAFFIC*)
 336  CONTINUE
      PRINT49
 49   FORMAT(' ',*EXISTING NEW NODES*)
      PRINT42
 42   FORMAT(' ',*SUPER NODE  TERMINAL   TO CONCENTRATOR*)
 43   FORMAT(' ',I7,9X,I5,12X,I5)
      DO 71 I=1,LL
         IF(YN(I).EQ.3) GO TO 71
         PRINT43,I,NN(I,3),NN(I,4)
 71   CONTINUE
      PRINT48
 48   FORMAT(' ',*MERGED SUPER NODES:*)
      PRINT47
 47   FORMAT(' ',*SUPER NODE*,5X,*MERGED TO*,5X,*FROM TERMINAL*,5X,*TO
     *TERMINAL*)
      DO 72 I=1,LL
         IF(YN(I).NE.3) GO TO 72
```

```
      PRINT46,I,NN(I,6),NN(I,4),NN(I,5)
 46   FORMAT(' ',I4,17X,I4,11X,I4,10X,I4)
 72   CONTINUE
      PRINT3089
 3089 FORMAT('1','**********************************************************
     **********************************************************')
      COST1=COST
      COST=0.0
      SEC=.TRUE.
      GO TO 3222
 3199 IF(COST1.GT.COST) GO TO 3499
      IF(COST1.EQ.COST) GO TO 3599
      PRINT3399,COST1
 3399 FORMAT(' ','THE REINITIALIZED NETWORK HAS THE LOWER COST OF',F8.3)
      GO TO 3699
 3499 PRINT3799,COST
 3799 FORMAT(' ','THE FINAL NETWORK HAS THE LOWER COST OF',F8.3)
      GO TO 3699
 3599 PRINT3899
 3899 FORMAT(' ','BOTH TOPOLOGIES HAVE THE SAME COST,THUS NEITHER ARE
     *OPTIMAL BUT ARE EQUALLY AS GOOD')
 3699 CONTINUE
      STOP
      END
C
C
      REAL FUNCTION LD(II,J)
      REAL XT(160,2),DIST(5),FIX(5),FAC(5)
      INTEGER*2 II,J
      COMMON /AREA2/XT
      DATA DIST/100.,50.,25.,10.,0.0/,FIX/86.2,59.7,42.2,23.75,6.25/,
     *FAC/.35,.53,.70,1.23,1.75/
C
C     CALCULATE LOW SPEED LINE COST
C
C
      D=SQRT((XT(II,1)-XT(J,1))**2+(XT(II,2)-XT(J,2))**2)
      IF(D.EQ.0.0) GO TO 3
      DO 1 I=1,5
         IF(D.LE.DIST(I)) GO TO 1
         LD=FIX(I)+(D-DIST(I))*FAC(I)
         GO TO 2
 1    CONTINUE
 3    LD=0.0
 2    RETURN
      END
C
C
      REAL FUNCTION HD(J)
      REAL XT(160,2),HDIST(5),HHFIX(5),HFAC(5),FIXC
      INTEGER*2 J
      COMMON /AREA2/XT/AREA19/FIXC
      DATA HDIST/50.,25.,10.,2.5,0.0/,HHFIX/74.5,48.25,25.75,10.,2.5/,
     *HFAC/.75,1.05,1.5,2.1,3.0/
C
C     CALCULATE THE HIGH-SPEED LINE COST AND THE TOTAL CONCENTRATOR COST
C
C
      X=XT(1,1)
```

```
      Y=XT(1,2)
      D=SQRT((XT(J,1)-X)**2+(XT(J,2)-Y)**2)
      IF(D.EQ.0.0) GO TO 3
      DO 1 I=1,5
         IF(D.LE.HDIST(I)) GO TO 1
         HD=HHFIX(I)+(D-HDIST(I))*HFAC(I)+FIXC
         GO TO 2
 1    CONTINUE
 3    HD=FIXC
 2    RETURN
      END
C
C
      SUBROUTINE SAHC(KK,JK,TMAX,NMAX,IN)
      REAL MMAX,MAX,DO(160,80),D(40,40),TB1(40),TO3(40,40),XT(160,2),LD
     *,L(40),TT(160),COST,MIN
      INTEGER*2 TS(40),Y(40),N(40),REL(40,40),X(40,40),IN(40),NI(40),
     *TO1(40,40),TO2(40,40),SS(40,40),T(160),TB2(40),XO(40,80)
      INTEGER*2 SI,SJ,SJJ,SEG,NONX,LX,PX,KX,KK,NMAX,LLL,OUT(10)/10*0/
      INTEGER*2 TMAX,TX,JK,OO,DE,IISEG,BX,TON,TOS,DP,P,PP,I,PPQ,II,JJ
      LOGICAL SEC
      COMMON /AREA1/DO,T/AREA2/XT/AREA9/N,TS,X/AREA10/SS,D/AREA11/COST
      COMMON /AREA12/XO/AREA18/SEC
 25   FORMAT(' ','X(',I3,',',I3,')=1')
 21   FORMAT(' ',2X,'TERMINAL',I4,2X,'REQUIRES',F6.2,'LINES')
C
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT IS BASED ON THE ESAU-WILLIAMS ALGORITHM.
C     IT FORMS A SUBOPTIMAL CONSTRAINED MULTIDROP NETWORK
C     AROUND A CONCENTRATOR.
C
C
C     DEFINITION OF PROGRAM VARIABLES:
C     TO1(I,J)= THE NODE OF SEGMENT I WHICH IS TO BE JOINED
C               TO SEGMENT J.
C     TO2(I,J)= THE NODE OF SEGMENT J WHICH IS TO RECIEVE THE
C               LINK FROM SEGMENT I.
C     TO3(I,J)= THE TRADEOFF FOR SEGMENT I. THIS IMPLIES DROPPING
C               THE CENTRAL LINK FROM SEGMENT I AND ADDING A
C               NON-CENTRAL LINK BETWEEN I AND J.
C     TB1(I)= THE MAXIMUM TRADEOFF FOR SEGMENT I.
C     TB2(I)= THE SEGMENT THAT THE MAXIMUM TRADEOFF SEGMENT
C             I IS ASSICIATED WITH.
C     TS(I)= THE TOTAL TRAFFIC ON SEGMENT I.
C     N(I)= THE TOTAL NUMBER OF NODES OR TERMINALS ON SEGMENT I.
C           IT IS LESS THAN OR EQUAL TO NMAX.
C     IN(J)= THE VECTOR OF TERMINALS ASSIGNED TO CONCENTRATOR J.
C     REL(I,J)= THE RELATION MATRIX FOR THE SEGMENTS. IF A
C               CHANGE OCCURS IN SEGMENT I, THEN REL(I,J)=1 FOR ALL J.
C               IF A CHANGE OCCURS IN SEGMENT J THEN REL(J,I)=1
C               FOR ALL I. THIS IS DONE TO EVALUATE WHETHER OR
C               NOT IT IS NECESSARY TO RECALCULATE TRADEOFFS.
C               BEFORE ACTING ON THE EXCHANGE ASSOCIATED WITH THE
C               MAXIMUM TRADEOFF BETWEEN I AND J REL(I,J) AND REL(J,I)
C               CHECKED. IF EITHER EQUALS 1 THEN THE TRADEOFF(S) IS
C               RECALCULATED.
C     OUT(J)= THE VECTOR OF TERMINALS THAT REQUIRE ONE OR
C             MORE FULLY LOADED LINES.
```

```
C      NI(J)= THE VECTOR OF TERMINALS THAT DONNOT REQUIRE FULLY
C             LOADED LINES. IT IS A PROPER SUBSET OF IN(J).
C      D(I,J)= THE COST MATRIX. IT IS ASSUMED TO BE SYMMETRICAL.
C      SS(I,...)= THE VECTOR OF TERMINALS ON SEGMENT I.
C      Y(*)= THE STATE VECTOR OF THE CENTRAL LINKS. THIS IMPLIES
C            THAT WHEN Y(*)=1 THE CENTRAL LINK FROM TERMINAL * IS
C            IN THE TREE. IF Y(*)=0 THEN THE CENTRAL LINK BETWEEN
C            TERMINAL * AND THE CONCENTRATOR HAS BEEN REPLACED
C            BY A NON-CENTRAL LINK.
C
C
C      DETERMINE WHICH TERMINALS REQUIRE FULLY LOADED LINES,AND
C      THE NUMBER OF LINES , AND ELIMIATE THEM FROM FURTHER
C      CONSIDERATION.
C
       PPQ=0
       P=0
       DO 1 I=1,KK
          IZ=I
          TT(IZ)=T(IN(IZ))
          TTMAX=0.0+TMAX
          L(IZ)=TT(IZ)/TTMAX
          IF(L(IZ).LE.1.) GO TO 2
          JL=T(IN(IZ))/TMAX
          PPQ=PPQ+1
          OUT(PPQ)=IN(IZ)
          IF(L(IZ)-JL.EQ.0.0) GO TO 29
          L(IZ)=JL+1.0
 29       PRINT21,IN(IZ),L(IZ)
       DO 52 JT=1,M
 52       DO(IN(IZ),JT)=L(IZ)*DO(IN(IZ),JT)
       COST=COST+DO(IN(IZ),JK)
       PRINT25,IN(IZ),JK
       GO TO 1
 2     P=P+1
       NI(P)=IN(IZ)
 1     CONTINUE
       KK=P
C
C      THE VECTOR NI(J) NOW HAS THE SET OF FEASIBLE OR NOT FULLY
C      LOADED TERMINALS OR NODES.
C
C
C      THE COST MATRIX FOR THESE TERMINALS IS CALCULATED.
C
       LLL=0
       DO 3 J=1,KK
          LLL=LLL+1
          JJ=NI(J)
          D(J,J)=0.
       DO 34 I=LLL,KK
          IF(I.EQ.J) GO TO 34
          II=NI(I)
          D(I,J)=LD(II,JJ)
          D(J,I)=D(I,J)
 34     CONTINUE
 3     CONTINUE
C
C
```

```
C         INITIALIZATION
C
C
        DO 51 I=1,KK
        DO 51 J=1,KK
           X(I,J)=0
 51     CONTINUE
        DO 50 I=1,KK
           IX=I
           XO(IX,JK)=1
           Y(IX)=1
           TB1(IX)=0.0
           TB2(IX)=JK
           SS(IX,1)=IX
           N(IX)=1
           TS(IX)=T(NI(IX))
 50       CONTINUE
C
C       THE MAXIMUM TRADEOFFS FOR EACH SEGMENT ARE CALCULATED.
C
        DO 6 I=1,KK
           IX=I
        DO 66 J=1,KK
           IY=J
           IF(IX.EQ.IY)GO TO 66
           REL(IX,IY)=0
           TX=TS(IX)+TS(IY)
           IF(TX.GT.TMAX) GO TO 7
           TO1(IX,IY)=IX
           TO2(IX,IY)=IY
           TO3(IX,IY)=DO(NI(IX),JK)-D(IX,IY)
           GO TO 66
 7         TO1(IX,IY)=0
           TO2(IX,IY)=0
           TO3(IX,IY)=-1.0
 66     CONTINUE
 6       CONTINUE
C
        DO 8 I=1,KK
           IXX=I
           MMAX=-1.0
           ISEG=0
        DO 9 J=1,KK
           IYY=J
           IF(IXX.EQ.IYY) GO TO 9
           IF(TO3(IXX,IYY).LE.MMAX) GO TO 9
           MMAX=TO3(IXX,IYY)
           ISEG=IYY
 9       CONTINUE
          TB1(IXX)=MMAX
          TB2(IXX)=ISEG
 8       CONTINUE
C
C       THE MAXIMUM OF THE SEGMENT TRADEOFFS IS FOUND.
C       IF THE MAXIMUM IS NEGATIVE THEN TERMINATE.
C
 100    MAX=-1.0
        SI=0
        DO 10 I=1,KK
```

```
          IP=I
          IF(Y(IP).EQ.0) GO TO 10
          IF(TB1(IP).LE.MAX) GO TO 10
          MAX=TB1(IP)
          SI=IP
 10      CONTINUE
         IF(MAX.LE.0.0) GO TO 1000
C
C        THE VALIDITY AND FEASIBILITY OF THE TRADEOFF IS CHECKED.
C
C
          SJ=TB2(SI)
          IF(Y(SJ).EQ.0) GO TO 90
          IF((N(SJ)+N(SI)).GT.NMAX) GO TO 90
          IF((TS(SJ)+TS(SI)).GT.TMAX) GO TO 90
          IF(REL(SI,SJ).EQ.0) GO TO 80
          REL(SI,SJ)=0
C
C        THE TRADEOFF BETWEEN THE SEGMENTS SI AND SJ IS RECALCULATED.
C
 68       CALL RECOM(SI,SJ,N(SI),N(SJ),NONX,SEG,MIN)
          TO1(SI,SJ)=NONX
          TO2(SI,SJ)=SEG
          TO3(SI,SJ)=DO(NI(SI),JK)-MIN
          TB1(SI)=TO3(SI,SJ)
          GO TO 100
 90       TO3(SI,SJ)=-1.0
C
C        THE NEW MAXIMUM TRADEOFF FOR SEGMENT SI IS CALCULATED.
C
  91    XMIN=9000.
        DP=N(SI)
        Y(SI)=0
        IISEG=0
        DO 73 J=1,KK
           IX=J
           IF(Y(IX).EQ.0) GO TO 73
           DE=N(IX)
           IF((DE+DP).GT.NMAX) GO TO 733
           IF((TS(IX)+TS(SI)).GT.TMAX) GO TO 733
        DO 72 I=1,DP
           OO=SS(SI,I)
        DO 74 K=1,DE
           BX=SS(IX,K)
           DDUM=D(OO,BX)
           IF(DDUM.GE.XMIN) GO TO 74
           XMIN=DDUM
           IISEG=IX
           TON=OO
           TOS=BX
 74       CONTINUE
 72       CONTINUE
          GO TO 73
 733      TO3(SI,IX)=-1.0
   73   CONTINUE
        IF(IISEG.NE.0) GO TO 773
         TB1(SI)=-1.0
         GO TO 100
 773     Y(SI)=1
```

```
      REL(SI,IISEG)=0
      TB1(SI)=DO(NI(SI),JK)-XMIN
      TB2(SI)=IISEG
      TO1(SI,IISEG)=TON
      TO2(SI,IISEG)=TOS
      TO3(SI,IISEG)=TB1(SI)
      GO TO 100
 80   IF(REL(SJ,SI).EQ.0) GO TO 82
      REL(SJ,SI)=0
      GO TO 68
C
C     THE CENTRAL LINK (SI,SJ) IS DROPPED AND THE NON-CENTRAL
C     LINK (IXI,IYI) IS ADDED.
C
 82   XO(SI,JK)=0
      IXI=TO1(SI,SJ)
      IYI=TO2(SI,SJ)
      X(IXI,IYI)=1
      X(IYI,IXI)=1
      COST=COST+D(IXI,IYI)
      PRINT25,NI(IXI),NI(IYI)
      Y(SI)=0
      TB1(SI)=-1.0
C
C     THE DATA AND VECTORS OF THE SEGMENT ARE UPDATED.
C
      LX=N(SJ)+1
      PX=LX-1
      KX=PX+N(SI)
      DO 99 I=LX,KX
         PP=I-PX
        SS(SJ,I)=SS(SI,PP)
 99     CONTINUE
        N(SJ)=N(SJ)+N(SI)
      TS(SJ)=TS(SJ)+TS(SI)
        IF(N(SJ).EQ.NMAX) GO TO 79
        IF(TS(SJ).EQ.TMAX) GO TO 79
      DO 81 I=1,KK
        IF(Y(I).EQ.0) GO TO 81
        REL(SJ,I)=1
 81   CONTINUE
      GO TO 100
      TB1(SJ)=-1.0
      GO TO 100
C
C     THE OUTPUT FOR THIS CONCENTRATOR IS GENERATED.
C
 79   Y(SJ)=0
 1000 PRINT2001,PPQ
 2001 FORMAT(' ','THE NO. OF ISOLATED NODES ARE',I4,2X,'THEY ARE:')
      IF(PPQ.EQ.0) GO TO 2005
      DO 55 I=1,PPQ
 55      PRINT,OUT(I)
 2005 PRINT2002,JK
 2002 FORMAT(' ','NODES CONNECTED TO CONC.',I4)
      DO 56 J=1,KK
         IF(XO(J,JK).EQ.0) GO TO 56
         PRINT,NI(J)
         COST=COST+DO(NI(J),JK)
```

```
 56    CONTINUE
          IF(SEC) GO TO 3001
          CALL DECOM(JK,OUT,PPQ,KK,NI,NMAX,TMAX)
 3001 RETURN
      END
C
C
      SUBROUTINE RECOM(IV,JV,II,JJ,NONX,SEG,MIN)
      REAL MIN,XT(160,2),D(40,40)
      INTEGER*2 SEG,NONX,IX,IY,IV,JV,II,JJ,
     *SS(40,40)
      COMMON /AREA2/XT/AREA10/SS,D
C
C
C     THIS SUBROUTINE IS CALLED BY SAHC
C     IT RECALCULATES THE MINIMUM COST CONNECTION BETWEEN
C     TWO TREE SEGMENTS IV AND IJ.
C
C
          MIN=9000.
          DO 31 I=1,II
          IX=SS(IV,I)
          DO 31 J=1,JJ
          IY=SS(JV,J)
          DUM=D(IX,IY)
          IF(DUM.GE.MIN) GO TO 31
            MIN=DUM
            SEG=IY
            NONX=IX
 31     CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DECOM(JK,OUT,PPQ,KK,NI,NMAX,TMAX)
      REAL DD(160,80),D(40,40),CST(80)
      INTEGER*2 NNCON(80),CONC(80,40),LCONC(80),N(40),TS(40),T(160),
     *X(40,40),NON(60,10),NCONC(80),NN(60,6),TCONC(80),SS(40,40),
     *NI(40),C(80),OUT(10),XO(40,80),YN(60)
      INTEGER*2 IT,PZ,KKP,IF,KP,NK,LL,NP,JK,PPQ,LPQ,NMAX,TMAX,IX,IQX,KK
      COMMON /AREA1/DD,T/AREA5/NNCON/AREA6/NON/AREA7/LCONC,NN,TCONC/
     *AREA8/NCONC,CONC,CST,LL
      COMMON /AREA3/C/AREA9/N,TS,X/AREA10/SS,D
      COMMON /AREA12/XO/AREA13/YN
C
C     THIS SUBROUTINE IS CALLED BY SAHC.
C     IT PARTITIONS THE NODES IN A CONCENTRATOR SUBNET INTO SUPER NODES
C     BY CONSIDERING EACH MULTIDROP OR POINT TO POINT LINE AS A CLUSTER
C     OF NODES.
C
C
C
C
C     SUPER NODE AND CONCENTRATOR PARAMETERS.
C
C     NN(LL,1)= THE NUMBER OF NODES IN THE SUPER NODE LL
C     NN(LL,2)= THE TOTAL TRAFFIC OF SUPER NODE LL.
C     NN(LL,3)= THE CONCENTRATOR SUPER NODE LL IS CONNECTED TO.
C     NN(LL,4)= THE NODE IN SUPER NODE LL THAT IS DIRECTLY CONNECTED TO
```

```
C          THE CONCENTRATOR
C       LCONC(JK)= THE NUMBER OF DIRECT LINKS TO CONCENTRATOR JK.
C       CONC(JK,*)= THE VECTOR OF SUPER NODES THAT ARE CONNECTED TO
C                   CONCENTRATOR JK.
C       NON(LL,*)= THE VECTOR OF TERMINALS IN SUPER NODE LL.
C       NNCON(JK)= THE NUMBER OF TERMINALS CONNECTED THROUGH CONCENTRATOR
C                  JK TO THE CPU.
C       NCONC(JK)= THE NUMBER OF SUPER NODES CONNECTED TO CONCENTRATOR JK.
C
C
 23    FORMAT(' ',10X,'NEW NODE',5X,'ORIGINAL NODES')
 24    FORMAT(' ','CONCENTRATORS',3X,'NEW NODES')
 26     FORMAT('0','NEW NODE',I4,'      INCIDENCE MATRIX')
 27    FORMAT(' ','X(',I3,',',I3,')=1')
 8155  FORMAT(' ','NEW NODE',I4,2X,'HAS ONLY ONE NODE',I4)
       LPQ=LL+1
       IT=NNCON(JK)
       DO 772 I=1,IT
          CONC(JK,I)=0
 772   CONTINUE
       PZ=0
       NX=0
       CST(JK)=0.
       LCONC(JK)=0
       NCONC(JK)=0
       IF(JK.EQ.1) GO TO 5555
C
C      THE TERMINAL OR NODE AT THE CONCENTRATOR IS ASSIGNED TO THE CLOSEST
C      SUPER NODE THAT CAN HANDLE IT. IF IT CANNOT BE ACCOMODATED BY ANY
C      SUPER NODE THEN IT IS CONSIDERED AS AN ISOLATED SUPER NODE.
C
       MIN=1000.
       DO 59 IL=1,KK
          IF(XO(IL,JK).EQ.0) GO TO 59
          IF((N(IL)+1).GT.NMAX) GO TO 59
          IF((TS(IL)+T(C(JK))).GT.TMAX) GO TO 59
          DX=DO(NI(IL),JK)
          IF(DX.GE.MIN) GO TO 59
            MIN=DX
            NX=IL
 59    CONTINUE
       IF(NX.NE.0) GO TO 45
       PZ=PZ+1
       LL=LL+1
       NN(LL,1)=1
       NN(LL,2)=T(C(JK))
       NN(LL,3)=JK
       NN(LL,4)=C(JK)
       NON(LL,1)=C(JK)
       YN(LL)=1
       CONC(JK,PZ)=LL
       GO TO 5555
 45    LCONC(JK)=LCONC(JK)+1
       PZ=PZ+1
       LL=LL+1
       CONC(JK,PZ)=LL
       KKP=N(NX)
       DO 75 JX=1,KKP
 75       NON(LL,JX)=SS(NX,JX)
```

```
      YN(LL)=1
      IF(KKP.EQ.1) GO TO 5421
      PRINT26,LL
      DO 5005 IG=1,KKP
         IQX=NON(LL,IG)
      DO 5006 JD=1,KK
         IF(X(IQX,JD).EQ.0) GO TO 5006
         IF(IQX.GT.JD) GO TO 5006
         PRINT27,NI(IQX),NI(JD)
 5006 CONTINUE
 5005 CONTINUE
      GO TO 5432
 5421 PRINT8155,LL,NI(NON(LL,1))
 5432 DO 77 JJX=1,KKP
 77      NON(LL,JJX)=NI(NON(LL,JJX))
      IF=KKP+1
      NON(LL,IF)=C(JK)
      NN(LL,1)=IF
      NN(LL,2)=TS(NX)+T(C(JK))
      NN(LL,3)=JK
      NN(LL,4)=NI(NX)
      PRINT27,NN(LL,4),C(JK)
C
C     EACH ISOLATED TERMINAL BECOMES AN ISOLATED SUPER NODE.
C
 5555 IF(PPQ.EQ.0) GO TO 5556
      DO 44 I=1,PPQ
         PZ=PZ+1
         LL=LL+1
         NN(LL,1)=1
         NN(LL,2)=T(OUT(I))
         NN(LL,3)=JK
         NN(LL,4)=OUT(I)
         NON(LL,1)=OUT(I)
         CONC(JK,PZ)=LL
         LCONC(JK)=LCONC(JK)+1
         CST(JK)=CST(JK)+DO(OUT(I),JK)
         YN(LL)=2
 44   CONTINUE
 5556 DO 55 I=1,KK
         IF(I.EQ.NX) GO TO 55
         IF(XO(I,JK).EQ.0) GO TO 55
         LCONC(JK)=LCONC(JK)+1
         CST(JK)=CST(JK)+DO(NI(I),JK)
         LL=LL+1
         PZ=PZ+1
         NN(LL,1)=N(I)
         NN(LL,2)=TS(I)
         NN(LL,3)=JK
         NN(LL,4)=NI(I)
         CONC( JK,PZ)=LL
         KP=N(I)
      DO 65 JJL=1,KP
 65      NON(LL,JJL)=SS(I,JJL)
      IF(NN(LL,1).EQ.1) GO TO 67
      PRINT26,LL
      DO 5001 II=1,KP
         IX=NON(LL,II)
      DO 5002 JA=1,KK
```

```
          IF(X(IX,JA).EQ.0) GO TO 5002
          IF(IX.GT.JA) GO TO 5002
          PRINT27,NI(IX),NI(JA)
 5002  CONTINUE
 5001  CONTINUE
 67    DO 66 JLR=1,KP
 66       NON(LL,JLR)=NI(NON(LL,JLR))
 55    CONTINUE
       NCONC(JK)=PZ
 999   DO 83 IE=LPQ,LL
          NK=NN(IE,1)
 47       PRINT23
          PRINT,IE,(NON(IE,J),J=1,NK)
 83    CONTINUE
       PRINT24
       PRINT,JK,(CONC(JK,L),L=1,PZ)
       RETURN
       END
C
C
       SUBROUTINE MAX(M,Y,HIGH,JW)
       REAL GAIN(80),HIGH
       INTEGER*2 M,Y(80),JW,IQ,I
       COMMON /AREA15/GAIN
C
C      THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C      IT FINDS THE MAXIMUM OF THE OPEN CONCENTRATOR GAINS.
C
C
C
C      JW=THE CONCENTRATOR WITH THE LARGEST GAIN
C      HIGH= THE VALUE OF THE LARGEST GAIN
C      Y(*)= THE STATUS VECTOR FOR THE CONCENTRATORS.
C
       HIGH=0.000
       DO 1 I=2,M
          IQ=I
          IF(Y(IQ).EQ.1) GO TO 1
          IF(GAIN(IQ).LE.HIGH) GO TO 1
          HIGH=GAIN(IQ)
          JW=IQ
 1     CONTINUE
       RETURN
       END
C
C
       SUBROUTINE CLOSET(ISA,IFF,TMAX,NMAX,M,Y,PMIN)
       REAL XT(160,80),DO(160,80),CST(80),TNN1(60),LD,PMIN
       INTEGER*2 ISA,IFF,M,MAXCON,TMAX,NMAX,LMAX,LL,JSTAR,IVI,JR,IVP,
      *NIVP,NIVI,NCN,NNCN,QPP,NA,NXX,JPX,NNT,NNO,IX
       INTEGER*2 Y(80),YN(60),T(160),C(80),TCONC(80),NCONC(80),
      *CONC(80,40),NN(60,6),NON(60,10),LCONC(80),TNN(60,5),DDCON(80,2),
      *CAPC(80)
       COMMON /AREA1/DO,T/AREA2/XT/AREA3/C/AREA4/MAXCON,LMAX/AREA6/NON
       COMMON /AREA7/LCONC,NN,TCONC/AREA8/NCONC,CONC,CST,LL/AREA13/YN
       COMMON /AREA14/TNN/AREA16/TNN1
       COMMON /AREA22/DDCON,CAPC
C
C      THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
```

```
C     IT FINDS THE CLOSEST FEASIBLE SUPER NODE FOR THE SUPER NODE ISA
C     ,EXCLUDING THOSE CONNECTED TO CONCENTRATOR IFF.
C
C
C
C     ISA= THE SUPER NODE BEING CONSIDERED.
C     IFF= THE CONCENTRATOR ISA IS CONNECTED TO.
C     JSTAR= THE NEXT CLOSEST FEASIBLE CONCENTRATOR TO SUPER NODE ISA.
C     TNN(ISA,1)= THE CLOSEST SUPER NODE TO ISA.
C     TNN(ISA,2)= THE TERMINAL IN ISA FROM WHICH THE MINIMUM DISTANCE
C     WAS CALCULATED.
C     TNN(ISA,3)= THE TERMINAL IN THE SUPER NODE CLOSEST TO ISA TO WHICH
C     THE MINIMUM DISTANCE WAS CALCULATED.
C     TNN(ISA,4)= JSTAR
C     TNN(ISA,5)=THE CONCENTRATOR TERMINAL IF IT IS THE CLOSEST TERMINAL
C               =ZERO OTHERWISE.
C     TNN1(ISA)= THE REALLOCATION COST FOR SUPER NODE ISA.
C     YN(*)= THE VECTOR DENOTING THE STATUS OF THE SUPER NODES.
C          =1 IF THE SUPER NODE HAS THE CONCNETRATORS TERMINAL.
C          =2 IF THE SUPER NODE IS AN ISOLATED SUPER NODE DUE TO
C           TRAFFIC CONSTRAINTS.
C          =3 IF THE SUPER NODE HAS BEEN MERGED TO ANOTHER SUPER NODE.
C
C     Y(*)= THE VECTOR DENOTING THE STATUS OF THE CONCENTRATORS.
C        =1  IF THE CONCENTRATOR IS CLOSED.
C        =0 IF THE CONCENTRATOR IS OPEN.
C
C
      Y(IFF)=1
      PMIN=1000.
      JSTAR=0
      NIVP=0
      IQ=1
      NA=NN(ISA,2)
      JPX=NN(ISA,1)
      DO 4 J=1,M
         IF(Y(J).EQ.1) GO TO 4
         IF(J.NE.1) GO TO 78
         IX=1
         GO TO 11
 78      IF((NA+TCONC(J)).GT.MAXCON) GO TO 4
         IF(NA+CAPC(J).GT.MAXCON) GO TO 4
         IX=CONC(J,1)
         IF(YN(IX).EQ.3) GO TO 11
         IF(NN(IX,1)+JPX.GT.NMAX) GO TO 11
         IF(DDCON(IX,1)+JPX.GT.NMAX) GO TO 11
         IF(NN(IX,2)+NA.GT.TMAX) GO TO 11
         IF(DDCON(IX,2)+NA.GT.TMAX) GO TO 11
         GO TO 12
 11      JR=C(J)
      DO 15 JL=1,JPX
         IVI=NON(ISA,JL)
         D=LD(IVI,JR)
         IF(D.GT.PMIN) GO TO 15
           PMIN=D
           NIVI=IVI
           NIVP=JR
           JSTAR=J
           NNCN=IX
```

```
15    CONTINUE
         IF(JSTAR.NE.1) IQ=2
12    NX=NCONC(J)
         IF(NX.EQ.1.AND.IQ.EQ.2) GO TO 4
      DO 3 I=IQ,NX
         NCN=CONC(J,I)
         IF(YN(NCN).EQ.3) GO TO 3
         NNT=NN(NCN,2)
         NNO=NN(NCN,1)
         IF((NA+NNT).GT.TMAX) GO TO 3
         IF(NA+DDCON(NCN,2).GT.TMAX) GO TO 3
         IF((JPX+NNO).GT.NMAX) GO TO 3
         IF(JPX+DDCON(NCN,1).GT.NMAX) GO TO 3
         NXX=NN(NCN,1)
      DO 2 JJ=1,NXX
         IVP=NON(NCN,JJ)
      DO 1 II=1,JPX
         IVI=NON(ISA,II)
         D=LD(IVI,IVP)
         IF(D.GE.PMIN) GO TO 1
           NIVI=IVI
           NIVP=IVP
           JSTAR=J
           NNCN=NCN
           PMIN=D
 1     CONTINUE
 2    CONTINUE
 3    CONTINUE
 4    CONTINUE
 9    IF(NIVP.EQ.C(JSTAR)) TNN(ISA,5)=JSTAR
         TNN(ISA,1)=NNCN
         TNN(ISA,2)=NIVI
         TNN(ISA,3)=NIVP
         TNN(ISA,4)=JSTAR
         TNN1(ISA)=PMIN
13    Y(IFF)=0
      RETURN
      END
C
C
      SUBROUTINE AMIN(K,M,ISS,Y)
      REAL DO(160,80),MIN
      INTEGER*2 T(160),Y(80),M,ISS
      COMMON /AREA1/DO,T
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT FINDS THE CLOSEST CONCENTRATOR TO A TERMINAL.
C
C
      MIN=DO(K,1)
      ISS=1
       DO 1 I=2,M
         IF(Y(I).EQ.1) GO TO 1
         DX=DO(K,I)
         IF(DX.GE.MIN) GO TO 1
           MIN=DX
           ISS=I
 1    CONTINUE
      RETURN
```

```
      END
C
C
      SUBROUTINE AAMIN(PPX,IG,ISS,MAXCON,Y,AY)
      REAL DO(160,80),MIN
      INTEGER*2 Y(80),T(160),TCONC(80),NN(60,6),LCONC(80),AY(80)
      INTEGER*2 ISS,PPX,IG,MAXCON
      COMMON /AREA1/DO,T/AREA7/LCONC,NN,TCONC
C
C     THIS SUBROUTINE IS CALLED BY REAR.
C     IT FINDS THE CLOSEST CONCENTRATOR TO A TERMINAL THAT CAN
C     HANDLE ITS TRAFFIC.
C
C
      MIN=DO(IG,1)
      ISS=1
      DO 1 I=2,PPX
         IF(Y(I).EQ.1) GO TO 1
         IF(AY(I).EQ.1) GO TO 1
         IF(TCONC(I)+T(IG).GT.MAXCON) GO TO 1
         DX=DO(IG,I)
         IF(DX.GE.MIN) GO TO 1
           MIN=DX
           ISS=I
 1    CONTINUE
      RETURN
      END
C
C
      SUBROUTINE FEAS(M,JW,MAXCON,TMAX,NMAX,FLAG,JLAST,Y)
      REAL MAX,MIN,PMIN,GAIN(80),CST(80),TNN1(60)
      INTEGER*2 NN(60,6),DDCON(80,2),CAPC(80),CONC(80,40),TCONC(80),
     *NCONC(80),NDCO(80,20),TNN(60,5),Y(80),YN(60),LCONC(80),DC(80)
      INTEGER*2 MA,JW,JLAST,MAXCON,TMAX,NMAX,IV,IPP,LL,M,IX,IY,OO,TEX
      LOGICAL FLAG
      COMMON /AREA7/LCONC,NN,TCONC/AREA8/NCONC,CONC,CST,LL/AREA13/YN
      COMMON /AREA14/TNN/AREA15/GAIN/AREA16/TNN1/AREA22/DDCON,CAPC
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT CHECKS THE FEASIBILITY OF A CONCENTRATORS' SUPER NODE
C     ASSIGNMENTS.
C     IF THEY ARE NOT FEASIBLE THEN THE NECESSARY ONES ARE CHANGED.
C
C
      FLAG=.TRUE.
      MA=NCONC(JW)
      DO 1 I=1,LL
         DC(I)=0
         DDCON(I,1)=0
         DDCON(I,2)=0
 1    CONTINUE
      DO 22 I=1,M
 22      CAPC(I)=0
      DO 2 I=1,MA
         IX=CONC(JW,I)
         IF(YN(IX).EQ.3) GO TO 2
         IY=TNN(IX,1)
         OO=TNN(IX,4)
         DDCON(IY,1)=DDCON(IY,1)+NN(IX,1)
```

```
      DDCON(IY,2)=DDCON(IY,2)+NN(IX,2)
      DC(IY)=DC(IY)+1
      NDCO(IY,DC(IY))=IX
      CAPC(00)=CAPC(00)+NN(IX,2)
2     CONTINUE
      DO 3 J=1,M
      IF(J.EQ.1) GO TO 3
      IF(Y(J).EQ.1) GO TO 3
      IF(CAPC(J).EQ.0) GO TO 3
      IF(TCONC(J)+CAPC(J).LE.MAXCON) GO TO 3
      IF(FLAG) FLAG=.FALSE.
592   MAX=0.0
      MP=NCONC(J)
      DO 572 I=1,MP
      IX=CONC(J,I)
      IF(YN(IX).EQ.3) GO TO 572
      IF(DC(IX).EQ.0) GO TO 572
      NA=DC(IX)
      DO 582 JJ=1,NA
      IYY=NDCO(IX,JJ)
      D=TNN1(IYY)
      IF(D.LE.MAX) GO TO 582
      MAX=D
      IPP=IYY
      IJP=JJ
      IPX=IX
582   CONTINUE
572   CONTINUE
      DDCON(IPX,1)=DDCON(IPX,1)-NN(IPP,1)
      DDCON(IPX,2)=DDCON(IPX,2)-NN(IPP,2)
      NDCO(IPX,IJP)=NDCO(IPX,DC(IPX))
      DC(IPX)=DC(IPX)-1
      CAPC(J)=CAPC(J)-NN(IPP,2)
      IV=YN(IPX)
      YN(IPX)=3
      Y(J)=1
      GAIN(JW)=GAIN(JW)+TNN1(IPP)
      CALL CLOSET(IPP,JW,TMAX,NMAX,M,Y,PMIN)
      GAIN(JW)=GAIN(JW)-PMIN
      YN(IPX)=IV
      Y(J)=0
      IYY=TNN(IPP,1)
      IJJ=TNN(IPP,4)
      DDCON(IYY,1)=DDCON(IYY,1)+NN(IPP,1)
      DDCON(IYY,2)=DDCON(IYY,2)+NN(IPP,2)
      DC(IYY)=DC(IYY)+1
      NDCO(IYY,DC(IYY))=IPP
      CAPC(IJJ)=CAPC(IJJ)+NN(IPP,2)
      IF(CAPC(J)+TCONC(J).GT.MAXCON) GO TO 592
3     CONTINUE
      DO 44 J=1,M
      IF(Y(J).EQ.1) GO TO 44
      IF(CAPC(J).EQ.0) GO TO 44
      MA=NCONC(J)
      DO 445 I=1,MA
      IX=CONC(J,I)
      IF(YN(IX).EQ.3) GO TO 445
      IF(DC(IX).EQ.0) GO TO 445
      IF(DDCON(IX,1)+NN(IX,1).GT.NMAX) GO TO 446
```

```
        IF(DDCON(IX,2)+NN(IX,2).GT.TMAX) GO TO 446
        GO TO 445
446   NA=DC(IX)
      MAX=0.0
      DO 447 JJ=1,NA
         IP=NDCO(IX,JJ)
         IF(TNN(IP,5).NE.0) GO TO 447
         D=TNN1(IP)
         IF(D.LE.MAX) GO TO 447
           MAX=D
           IPP=IP
           IJP=JJ
447   CONTINUE
      IF(MAX.EQ.0.0) GO TO 445
      IF(FLAG) FLAG=.FALSE.
      DDCON(IX,1)=DDCON(IX,1)-NN(IPP,1)
      DDCON(IX,2)=DDCON(IX,2)-NN(IPP,2)
      NDCO(IX,IJP)=NDCO(IX,DC(IX))
      DC(IX)=DC(IX)-1
      CAPC(J)=CAPC(J)-NN(IPP,2)
      IV=YN(IX)
      YN(IX)=3
      GAIN(JW)=GAIN(JW)+TNN1(IPP)
         CALL CLOSET(IPP,JW,TMAX,NMAX,M,Y,PMIN)
      GAIN(JW)=GAIN(JW)-PMIN
      YN(IX)=IV
      IYY=TNN(IPP,1)
      IJJ=TNN(IPP,4)
      DDCON(IYY,1)=DDCON(IYY,1)+NN(IPP,1)
      DDCON(IYY,2)=DDCON(IYY,2)+NN(IPP,2)
      DC(IYY)=DC(IYY)+1
      NDCO(IYY,DC(IYY))=IPP
      CAPC(IJJ)=CAPC(IJJ)+NN(IPP,2)
      IF(DDCON(IX,1)+NN(IX,1).GT.NMAX) GO TO 446
      IF(DDCON(IX,2)+NN(IX,2).GT.TMAX) GO TO 446
 445  CONTINUE
 44   CONTINUE
      JLAST=JW
      DO 117 I=1,LL
         DDCON(I,1)=0
 117     DDCON(I,2)=0
      DO 118 I=1,M
 118     CAPC(I)=0
      RETURN
      END
C
C
      SUBROUTINE MINSA(JK,JJK,NMAX,TMAX,M,Y)
      REAL TD(40),TD1(40,40),CST(80),GAIN(80),TNN1(60)
      REAL MINX,COST,LD,PMIN
      INTEGER*2 XX,YY,IX,IY,P,L,NIX,NIY,TIX,TIY,MN,TON,NX,XN,NA,XNX
      INTEGER*2 FN,TN,IC,IT,NAA,ISA,NMAX,TMAX,LL,JJK,JK,JST,M
      INTEGER*2 NN(60,6),NCONC(80),CONC(80,40),TD2(40,40,2),YN(60),
     *STO(40),NON(60,10),TCONC(80),LCONC(80),Y(80),NI(40)
      COMMON/AREA6/NON/AREA7/LCONC,NN,TCONC/AREA8/NCONC,CONC,CST,LL
      COMMON /AREA11/COST/AREA13/YN/AREA15/GAIN/AREA16/TNN1
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT ATTEMPTS TO REOPTIMIZE A CONCENTRATORS SUBNETWORK USING THE
```

```
C       SAHC (STEEPEST ASCENT HILL CLIMBING) APPROACH.
C       THE OPTIMIZATION IS ATTEMPTED WHENEVER AT LEAST ONE SUPER NODE
C       HAS BEEN ADDED DIRECTLY TO THE CONCENTRATOR.
C
C
C
        LP=0
        L=0
        P=0
        IR=1
        IF(JK.EQ.1) GO TO 57
        IF(NN(CONC(JK,1),1).EQ.1) IR=2
 57     NA=NCONC(JK)
        DO 55 I=1,NA
 55        STO(I)=0
C
C       DETERMINE FEASIBLE NODE EXCHANGE W.R.T. CONSTRAINTS
C
        DO 1 I=IR,NA
           IX=CONC(JK,I)
           IF(YN(IX).EQ.3) GO TO 1
           NIX=NN(IX,1)
           TIX=NN(IX,2)
           IF(NIX.EQ.NMAX) GO TO 1
           IF(TIX.EQ.TMAX) GO TO 1
        DO 2 J=IR,NA
           IF(J.EQ.I) GO TO 2
           IY=CONC(JK,J)
           IF(YN(IY).EQ.3) GO TO 2
           NIY=NN(IY,1)
           TIY=NN(IY,2)
           IF(NIY+NIX.GT.NMAX) GO TO 2
           IF(TIY+TIX.GT.TMAX) GO TO 2
           P=P+1
           NI(P)=IX
           GO TO 1
 2      CONTINUE
 1      CONTINUE
        IF(P.EQ.0) GO TO 3000
        DO 3 I=1,P
           IX=NN(NI(I),4)
 3         TD(I)=LD(IX,JJK)
C
C       CALCULATE THE INITIAL TRADEOFFS.
C
        DO 4 I=1,P
           NIX=NN(NI(I),1)
           TIX=NN(NI(I),2)
        DO 5 J=1,P
           IF(J.EQ.I) GO TO 6
           NIY=NN(NI(J),1)
           TIY=NN(NI(J),2)
           IF(NIX+NIY.GT.NMAX) GO TO 6
           IF(TIY+TIX.GT.TMAX) GO TO 6
           CALL BMIN(NI(I),NI(J),FN,TN,XMIN,NN,JJK)
           TD1(I,J)=TD(I)-XMIN
           TD2(I,J,1)=FN
           TD2(I,J,2)=TN
           GO TO 5
```

```
 6        TD1(I,J)=-1.0
 5      CONTINUE
 4      CONTINUE
C
C      FIND THE MAXIMUM OVER ALL TRADEOFFS
C
 2000  AMAX=-1.0
       DO 7 I=1,P
       DO 8 J=1,P
          IF(I.EQ.J) GO TO 8
          IF(TD1(I,J).LE.AMAX) GO TO 8
          AMAX=TD1(I,J)
          MN=NI(I)
          TON=NI(J)
          JJW=J
          IW=I
          XX=TD2(I,J,1)
          YY=TD2(I,J,2)
 8      CONTINUE
 7      CONTINUE
          IF(AMAX.LE.0) GO TO 1000
C
C      THE COST OF THE NETWORK IS DECREASED BY THE MAXIMUM TRADEOFF
C      AND THE GAIN OF THE CONCENTRATOR JK IS DECREASED BY TD(IW).
C
       COST=COST-AMAX
       LCONC(JK)=LCONC(JK)-1
          PRINT90,MN,TON
 90    FORMAT(' ','NEW NODE',I3,2X,'IS MERGED TO NODE',I3)
          PRINT91,XX,YY
 91    FORMAT(' ','BY LINK  X(',I3,',',I3,')')
       GAIN(JK)=GAIN(JK)+(TNN1(MN)-TD(IW))
       YN(MN)=3
       NN(MN,4)=XX
       NN(MN,5)=YY
       NN(MN,6)=TON
       TD1(IW,JJW)=-1.0
       TD1(JJW,IW)=-1.0
C
C      UPDATE SUPER NODE TON
C
       NX=NN(MN,1)
       XN=NN(TON,1)
       NN(TON,1)=XN+NX
       NN(TON,2)=NN(TON,2)+NN(MN,2)
       NAA=NN(TON,1)
       XNX=XN+1
       DO 9 I=XNX,NAA
          JJ=I-XN
 9        NON(TON,I)=NON(MN,JJ)
       LP=LP+1
       STO(LP)=TON
       NXN=NN(TON,1)
       TXN=NN(TON,2)
       IF(NXN.EQ.NMAX) GO TO 111
       IF(TXN.EQ.TMAX) GO TO 111
C
C      RECALCULATE OR INHIBIT THE TRADEOFFS BETWEEN THE NODE THAT HAS
C      JUST BEEN CHANGED AND THE OTHER NODES.
```

```
C
      DO 10 I=1,P
         IF(I.EQ.JJW) GO TO 10
         IX=NI(I)
         IF(YN(IX).EQ.3) GO TO 11
         IF(NXN+NN(IX,1).GT.NMAX) GO TO 11
         IF(TXN+NN(IX,2).GT.TMAX) GO TO 11
         CALL BMIN(TON,IX,FN,TN,XMIN,NN,JJK)
         TD1(JJW,I)=TD(JJW)-XMIN
         TD2(JJW,I,1)=FN
         TD2(JJW,I,2)=TN
         TD1(I,JJW)=TD(I)-XMIN
         TD2(I,JJW,1)=TN
         TD2(I,JJW,2)=FN
         GO TO 10
11       TD1(JJW,I)=-1.0
         TD1(I,JJW)=-1.0
10    CONTINUE
      GO TO 2020
111   DO 211 I=1,P
         TD1(JJW,I)=-1.0
         TD1(I,JJW)=-1.0
211   CONTINUE
2020  DO 20 I=1,P
         TD1(IW,I)=-1.0
20       TD1(I,IW)=-1.0
      GO TO 2000
1000  IF(JK.EQ.1) GO TO 4000
      IF(NN(CONC(JK,1),1).NE.1) GO TO 3000
      IC=CONC(JK,1)
      IT=NN(IC,2)
      JS=0
      MINX=10000.
      DO 13 J=1,P
         IX=NI(J)
         IF(YN(IX).EQ.3) GO TO 13
         IF(NN(IX,1)+1.GT.NMAX) GO TO 13
         IF(NN(IX,2)+IT.GT.TMAX) GO TO 13
         IF(TD(J).GE.MINX) GO TO 13
           MINX=TD(J)
           JS=J
           JST=IX
13    CONTINUE
      IF(JS.EQ.0) GO TO 3000
         YN(IC)=3
         YN(JST)=1
         NN(JST,1)=NN(JST,1)+1
         NN(JST,2)=NN(JST,2)+NN(IC,2)
         PRINT92,IC,JST
92       FORMAT(' ',*CONCENTRATOR SUPER NODE*,I3,* MERGED WITH SUPER
     *NODE*,I4)
         NAA=NN(JST,1)
         NDN(JST,NAA)=JJK
         NN(IC,4)=JJK
         NN(IC,5)=NN(JST,4)
         NN(IC,6)=JST
         PRINT91,NN(IC,4),NN(IC,5)
         GAIN(JK)=GAIN(JK)+TNN1(JST)
         CALL CLOSET(JST,JK,TMAX,NMAX,M,Y,PMIN)
```

```
         GAIN(JK)=GAIN(JK)-PMIN
      DO 14 I=1,NA
         IX=CONC(JK,I)
         IF(IX.EQ.JST)IS=I
14    CONTINUE
         CONC(JK,1)=JST
         CONC(JK,IS)=IC
3000  IF(LP.EQ.0) GO TO 4000
      DO 21 J=1,LP
         JST=STO(J)
         GAIN(JK)=GAIN(JK)+TNN1(JST)
         CALL CLOSET(JST,JK,TMAX,NMAX,M,Y,PMIN)
         GAIN(JK)=GAIN(JK)-PMIN
     *CHANGE*)
21    CONTINUE
4000  RETURN
      END


      SUBROUTINE BMIN(ISA,IX,FN,TN,XMIN,NN,JJK)
      REAL XMIN,LD
      INTEGER*2 ISA,IX,FN,TN,IZ,IW,JJK
      INTEGER*2 NON(60,10),NN(60,6),YN(60)
      COMMON /AREA6/NON/AREA13/YN
C
C     THIS SUBROUTINE IS CALLED BY MINSA.
C     IT DETERMINES THE MINIMUM DISTANCE (COST) BETWEEN TWO SUPER NODES.
C
C
C
      XMIN=10000.
      NA=NN(ISA,1)
      NX=NN(IX,1)
      IF(YN(ISA).EQ.1) NA=NA-1
      IF(YN(IX).EQ.1) NX=NX-1
      DO 1 I=1,NA
         IZ=NON(ISA,I)
      DO 2 J=1,NX
         IW=NON(IX,J)
         DX=LD(IZ,IW)
         IF(DX.GE.XMIN) GO TO 2
           XMIN=DX
           FN=IZ
           TN=IW
2     CONTINUE
1     CONTINUE
      RETURN
      END
C
C
      SUBROUTINE FMI(IK,IJ)
      REAL DD(160,80),MAX
      INTEGER*2 DCN(80,40),TC(80),DCON(80),NK,T(160),IX,IJ
      COMMON /AREA1/DD,T/AREA21/TC,DCON,DCN
C
C     THIS SUBROUTINE IS CALLED BY REAR.
C     IT DETERMINES WHICH TERMINALS ,OF THE SET OF POTENTIALLY
C     ASSIGNED TO A CONCENTRATOR,IS TO BE DROPPED, AND THEN DROPS IT.
C
```

```
C
      MAX=0.0
      NK=DCON(IK)
      DO 1 I=1,NK
         IX=DCN(IK,I)
         D=DO(IX,IK)
         IF(D.LE.MAX) GO TO 1
           MAX=D
           IJ=IX
           IJP=I
 1    CONTINUE
      TC(IK)=TC(IK)-T(IJ)
      DCN(IK,IJP)=DCN(IK,NK)
      DCON(IK)=DCON(IK)-1
      RETURN
      END
C
C
      SUBROUTINE REAR(PPX,Y,AY)
      REAL DO(160,80),CST(80)
      INTEGER*2 T(160),CONC(80,40),TCONC(80),DCON(80),NNCON(80),
     *TC(80),DCN(80,40),Y(80),AY(80),NCONC(80),LCONC(80),NN(60,6)
      INTEGER*2 ISS,PPX,MAXCON,IJ,LMAX,LL
      INTEGER REAS(40,2),LP
      COMMON /AREA1/DO,T/AREA4/MAXCON,LMAX/AREA5/NNCON/AREA7/LCONC,NN,
     *TCONC/AREA8/NCONC,CONC,CST,LL/AREA20/REAS,LP/AREA21/TC,DCON,DCN
C
C     THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C     IT REASSIGNS THE TERMINALS THAT WERE DROPPED WHEN THE
C     CONCENTRATOR TERMINAL ASSIGNMENTS WERE MODIFIED.
C
C
      IF(LP.EQ.0) GO TO 900
      DO 1 I=1,PPX
      TC(I)=0
 1    DCON(I)=0
C
C     THE CONCENTRATORS TO WHICH THE DROPPED TERMINALS ARE TO BE
C     POTENTIALLY ASSIGNED ARE DETERMINED. THE NUMBER OF TERMINALS
C     AND THEIR TOTAL TRAFFIC TO EACH CONCENTRATOR IS ALSO FOUND.
C
      DO 2 I=1,LP
         IX=REAS(I,1)
         IXC=REAS(I,2)
         DCON(IXC)=DCON(IXC)+1
         NA=DCON(IXC)
         DCN(IXC,NA)=IX
         TC(IXC)=TC(IXC)+T(IX)
 2    CONTINUE
C
C     IF A CONCENTRATOR CAN HANDLE ALL OF THE TRAFFIC OF ITS
C     POTENTIALLY ASSIGNED SET OF TERMINALS, THEN THEY ARE ASSIGNED
C     TO IT. IF IT CANNOT, THEN SOME OF THE TERMINALS ARE POTENTIALLY
C     ASSIGNED TO SOME OTHER CONCENTRATOR THAT CAN ACCOMODATE THEM.
C
      DO 3 I=1,PPX
         IF(DCON(I).EQ.0) GO TO 3
         IF(TCONC(I)+TC(I).GT.MAXCON) GO TO 5
 55      TCONC(I)=TCONC(I)+TC(I)
```

```
          KP=NNCON(I)
          NP=NNCON(I)+1
          NNP=NNCON(I)+DCON(I)
          NNCON(I)=NNP
       DO 4 J=NP,NNP
          KKP=J-KP
          CONC(I,J)=DCN(I,KKP)
 4     CONTINUE
          GO TO 3
 5        CALL FMI(I,IJ)
          Y(I)=1
          CALL AAMIN(PPX,IJ,ISS,MAXCON,Y,AY)
          Y(I)=0
          IF(ISS.LT.I) GO TO 7
          DCON(ISS)=DCON(ISS)+1
          TC(ISS)=TC(ISS)+T(IJ)
          DCN(ISS,DCON(ISS))=IJ
          IF(TCONC(I)+TC(I).GT.MAXCON) GO TO 5
          IF(DCON(I).EQ.0) GO TO 3
          GO TO 55
 7        NNCON(ISS)=NNCON(ISS)+1
          TCONC(ISS)=TCONC(ISS)+T(IJ)
          CONC(ISS,NNCON(ISS))=IJ
          IF(DCON(I).EQ.0) GO TO 3
          IF(TCONC(I)+TC(I).GT.MAXCON) GO TO 5
          GO TO 55
 3     CONTINUE
 900   RETURN
       END
C
C
       SUBROUTINE CFUL(K,Y,AY,PPX)
       REAL DO(160,80),STO3(40),CST(80),MIN,MAX,MMAX
       INTEGER*2 Y(80),Z(40),STO1(40),STO2(40),COM(20,2),LCONC(80),
      *NN(60,6),NCONC(80),T(160),NNCON(80),CONC(80,40),TCONC(80),
      *AY(80),PPX,LL,LMAX,NA,NAA,MAXCON
       INTEGER REAS(40,2),LP
       COMMON /AREA1/DO,T/AREA4/MAXCON,LMAX/AREA5/NNCON/AREA7/LCONC,NN,
      @TCONC/AREA8/NCONC,CONC,CST,LL/AREA20/REAS,LP
C
C      THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM.
C      IT MODIFIES A CONCENTRATORS TERMINAL ASSIGNMENT SO AS TO
C      SATISFY THE CAPACITY CONSTRAINT.
C      A TRADEOFF IS CALCULATED FOR EACH TERMINAL. THE TERMINALS WITH
C      THE SMALLEST TRADEOFFS ARE DROPPED FROM THE CONCENTRATOR
C      SUBNET UNTIL THE CAPACITY CONSTRAINT IS SATISFIED.
C
C
       DO 33 I=1,40
 33       Z(I)=0
       NA=NNCON(K)
       NAA=NA
C
C      THE NEAREST CONCENTRATOR FOR EACH TERMINAL ASSIGNED TO
C      CONCENTRATOR K IS FOUND.
C
       DO 1 I=1,NA
          IX=CONC(K,I)
          MIN=10000.
```

```
      DO 2 J=1,PPX
         IF(Y(J).EQ.1) GO TO 2
         IF(AY(I).EQ.1) GO TO 2
         IF(J.EQ.K) GO TO 2
         D=DO(IX,J)
         IF(D.GE.MIN) GO TO 2
           MIN=D
           IXJ=J
 2    CONTINUE
C
C     THE TERMINAL, CLOSEST CONCENTRATOR , AND TRADEOFF ARE STORED.
C
         STO1(I)=IXJ
         STO2(I)=IX
         STO3(I)=MIN-DO(IX,K)
 1    CONTINUE
 400  L=0
      MAX=10000.
C
C     THE SMALLEST TRADEOFF IS FOUND.
C
      DO 3 I=1,NA
         IF(Z(I).EQ.1) GO TO 3
         D=STO3(I)
         IF(D.GE.MAX) GO TO 3
           MAX=D
           IXP=I
 3    CONTINUE
C
C     ALL TRADEOFFS OF THE SAME VALUE ARE CONSIDERED.
C
      DO 4 I=1,NA
         IF(Z(I).EQ.1) GO TO 4
         DX=STO3(I)
         IF(DX.NE.MAX) GO TO 4
         L=L+1
         COM(L,1)=I
         COM(L,2)=DO(STO2(I),K)
 4    CONTINUE
         MMAX=COM(1,2)
         IM=1
         IF(L.EQ.1) GO TO 6
C
C     THE TERMINAL FARTHEST FROM THE CONCENTRATOR K IS DROPPED
C     IF DROPPING THIS TERMINAL DOES NOT SATISFY THE CAPACITY CONSTAINT
C     THEN THE OTHER TERMINALS WITH THE PRESENT MINIMUM TRADEOFF ARE
C     CONSIDERED FOR DROPPING. IF THERE ARE NO MORE TERMINALS WITH
C     THE PRESENT MINIMUM TRADEOFF, THEN A NEW MINIMUM AND THE
C     CORRESPONDING TERMINALS ARE FOUND.
C
      DO 5 I=2,L
         D=COM(I,2)
         IF(D.LE.MMAX) GO TO 5
         MMAX=D
         IM=I
 5    CONTINUE
 6    LP=LP+1
      IQ=COM(IM,1)
      IQQ=STO2(IQ)
```

```
      IPQ=STO1(IQ)
       REAS(LP,1)=IQQ
       REAS(LP,2)=IPQ
      TCONC(K)=TCONC(K)-T(IQQ)
      Z(IQ)=1
      IF(TCONC(K).LE.MAXCON) GO TO 7
      GO TO 400
 7    NNCON(K)=0
C
C     THE CONENTRATORS VECTOR OF TERMINALS ASSIGNED TO IT IS RESTORED.
C
      DO 45 I=1,NA
         IF(Z(I).EQ.1) GO TO 45
         NNCON(K)=NNCON(K)+1
         CONC(K,NNCON(K))=STO2(I)
 45   CONTINUE
      RETURN
      END
C
C
      BLOCK DATA
      REAL CST(80),COST
      INTEGER*2 NCONC(80),CONC(80,40),NNCON(80),LCONC(80),NN(60,6),
     *TNN(60,5),NON(60,10),YN(60),TCONC(80),LL,DDCON(80,2),CAPC(80)
      INTEGER REAS(40,2),LP
      COMMON /AREA5/NNCON/AREA6/NON/AREA7/LCONC,NN,TCONC/AREA8/NCONC,
     *CONC,CST,LL/AREA11/COST/AREA13/YN/AREA14/TNN
      COMMON /AREA20/REAS,LP/AREA22/DDCON,CAPC
C
C     THIS ROUTINE INITIALIZES THE NECESSARY DATA.
C
C
      DATA REAS,LP/80*0,0/
      DATA COST,YN,TNN/0.0,60*0,300*0/
      DATA NCONC,CONC,CST,LL/80*0,3200*0,80*0.,0/
      DATA NNCON,LCONC,TCONC,DDCON,CAPC/80*0,80*0,80*0,160*0,80*0/
      END
$ENTRY
```

```
C
C
C     THIS PROGRAM DETERMINES THE SET OF INITIAL CANDIDATE CONCENTRATOR
C     SITES. IT MAY BE WRITTEN AS A SUBROUTINE AND USED IN THE MAIN
C     ALGORITHM.
C
C
C     USER DEFINED PARAMETERS:
C
C     NT= THE NUMBER OF NODES IN THE NETWORK.
C     K= THE NUMBER OF NEAREST NEIGHBOURS TO BE CONSIDERED.
C     NR= THE NETWORK IDENTIFIER NUMBER.
C
C
      REAL XT(40,2)
      INTEGER*2 C(20,20),T(40),NT,CC(20),TC(20),IMAX
      COMMON /AREA1/XT/AREA2/C/AREA3/TC/AREA4/IMAX
C
      READ,NT,NR,K
      PRINT72,NT,NR,K
 72   FORMAT('1','NODE SET=',I3,2X,'RANDOM NET',I4,2X,'NN=',I3)
      DO 1 I=1,NT
         READ,(XT(I,J),J=1,2),T(I)
      PRINT442,I,XT(I,1),XT(I,2),T(I)
 442  FORMAT(' ',I3,2F10.2,I5)
 1    CONTINUE
      KK=1
      CC(1)=1
      CALL SITE(NT,K)
      TOT=0.0
C
C     THE WEIGHTED MEAN AND ITS INTEGRAL PART (PLUS ONE) ARE CALCULATED.
C
      DO 2 I=1,IMAX
 2       TOT=TOT+I*TC(I)
      TOT=TOT/NT
      IIT=IFIX(TOT)+1
      PRINT242,IIT
 242  FORMAT(' ','AVERAGE CLASS NUMBER IS=',I4)
      NP=NT/2
      II=IMAX+1
      ITOT=0
C
C     ALL TERMINALS ,STARTING WITH THE ONES WITH THE HIGHEST WEIGHT,
C     THAT ARE WITHIN THE BOUNDS,ARE CONSIDERED AS CANDIDATE
C     CONCENTRATOR SITES.
C
      DO 3 I=1,IMAX
         IJ=II-I
         IF(ITOT+TC(IJ).GT.NP.OR.IJ.LT.IIT) GO TO 6
         IK=TC(IJ)
         IF(IK.EQ.0) GO TO 3
      DO 4 J=1,IK
         KK=KK+1
         CC(KK)=C(IJ,J)
 4    CONTINUE
         ITOT=ITOT+IK
 3    CONTINUE
 6    PRINT20,KK
```

```
 20   FORMAT(' ','NUMBER OF CONC. SITES ARE',I4)
C
C     SINCE THE CPU IS INCLUDED IN THE CALCULATIONS IT MAY SHOW UP
C     TWICE AND IN THAT CASE THE SECOND OCCURANCE IS REMOVED.
C
      DO 21 I=2,KK
         IF(CC(I).EQ.1) GO TO 22
 21   CONTINUE
      GO TO 23
 22   CC(I)=CC(KK)
      KK=KK-1
 23   PRINT,(CC(I),I=1,KK)
      STOP
      END
      REAL FUNCTION LD(I,J)
C
C     THIS SUBROUTINE CALCULATES THE DISTANCE BETWEEN TWO POINTS.
C
      REAL XT(40,2)
      COMMON /AREA1/XT
C
         XX1=XT(I,1)
         XX2=XT(I,2)
         YY1=XT(J,1)
         YY2=XT(J,2)
         LD=SQRT((XX1-YY1)**2+(XX2-YY2)**2)
      RETURN
      END
      SUBROUTINE SITE(NT,K)
C
C
C     THIS SUBROUTINE DETERMINES THE WEIGHT OF EACH TERMINAL OR ITS
C     FREQUENCY OF OCCURANCE AND THEN GROUPS ALL TERMINALS OF THE
C     SAME WEIGHT INTO ONE CLASS WHICH IS IDENTIFIED BY THE WEIGHTING.
C
C
      REAL XT(40,2),D(40),LD
      INTEGER*2 N(40)/40*1/,NT,TC(20),IMAX,IX,IY,C(20,20)
      INTEGER*2 Y(40)/40*0/,NEBH(40,10)
      COMMON /AREA1/XT/AREA2/C/AREA3/TC/AREA4/IMAX
C
C
C     THE NEAREST NEIGHBOUR LISTS FOR EACH TERMINAL ARE DETERMINED.
C
      DO 1 I=1,NT
         L=0
      DO 2 J=1,NT
 2       D(J)=LD(I,J)
 31      CALL CMIN(I,D,Y,IX,NT)
         L=L+1
         NEBH(I,L)=IX
         Y(IX)=1
         IF(L.NE.K) GO TO 31
      DO 5 J=1,NT
 5       Y(J)=0
 1    CONTINUE
      Y(1)=1
      NSUM=0
C
```

```
C     FOR EACH TERMINAL ITS FREQUENCY OF OCCURANCE (WEIGHT) IN THE
C     NEAREST NEIGHBOUR LISTS IS DETERMINED.
C
      DO 3 I=1,NT
      DO 3 J=1,K
         IX=NEBH(I,J)
         N(IX)=N(IX)+1
 3       NSUM=NSUM+1
         IMAX=0
C
C     THE MAXIMUM FREQUENCY OF OCCURANCE (WEIGHT) IS FOUND.
C
      DO 4 I=1,NT
         IF(N(I).LE.IMAX) GO TO 4
         IMAX=N(I)
 4    CONTINUE
 42   FORMAT(' ','CLASS',I3,2X,'HAS',I3,'MEMBERS')
      DO 6 I=1,IMAX
         CALL CLASS(I,N,NT)
 6       PRINT42,I,TC(I)
      RETURN
      END
      SUBROUTINE CLASS(KP,N,NT)
C
C     THIS SUBROUTINE PUTS ALL TERMINALS OF WEIGHT KP IN THE SAME CLASS.
C
      INTEGER*2 N(40),NT,C(20,20),TC(20)
      COMMON /AREA2/C/AREA3/TC
C
      KK=0
      DO 1 I=1,NT
         IF(N(I).NE.KP) GO TO 1
         KK=KK+1
         C(KP,KK)=I
 1    CONTINUE
      TC(KP)=KK
      PRINT40,KP
 40    FORMAT('0','CLASS',I3,2X,'MEMBERS:')
      IF(KK.NE.0) GO TO 7
       PRINT41
 41   FORMAT('+',22X,'NONE')
      GO TO 8
 7    PRINT,(C(KP,I),I=1,KK)
 8    RETURN
      END.
      SUBROUTINE CMIN(I,D,Y,IX,NT)
C
C     THIS SUBROUTINE DETERMINES THE NEAREST (NOT ALREADY CONSIDERED)
C     NEIGHBOUR TO A TERMINAL OR NODE.
C
      REAL D(40)
      INTEGER*2 Y(40),IX,NT
C
      MIN=10000.
      DO 1 J=1,NT
         IF(Y(J).EQ.1) GO TO 1
         IF(J.EQ.I) GO TO 1
         IF(D(J).GE.MIN) GO TO 1
           MIN=D(J)
```

```
      * IX=J
1     CONTINUE
      RETURN
      END
$ENTRY
```

```
C
C
C
C     THIS PROGRAM GENERATES A SET OF RANDOM POINTS (X-Y COORDINATES)
C     AND A RANDOM TRAFFIC NUMBER FOR EACH POINT.
C
C
C     USER DEFINED PARAMETERS:
C
C     IN= THE INITIAL STARTING VALUE FOR THE FANDU SUBROUTINE.
C     IT MUST BE AN ODD INTEGER.
C
C     IT= THE MAXIMUM NUMBER OF TRAFFIC UNITS PER TERMINAL DESIRED.
C       =4 WILL PRODUCE RANDOM INTEGERS RANGING FROM 1 TO 4 GIVEN A
C        TRAFFIC SCALE FACTOR OF 10.0.
C
C     NT= THE NUMBER OF NODES OR TERMINALS DESIRED.
C     XNSCT= THE TRAFFIC SCALE FACTOR.
C
C     XNSC= THE X-Y SCALE FACTOR.
C         =100. WILL PRODUCE RANDOM REALS BETWEEN 0.0 AND 99.99
C
C     IP= THE NETWORK IDENTIFIER NUMBER.
C
C
      REAL XT(600,2)
      INTEGER T(600)
      READ,IN,IT,NT,XNSCT,XNSC,IP
      PRINT47,IN,IT,NT,XNSCT,XNSC
 47   FORMAT('1','START=',I4,2X,'USER=',I3,2X,'NODES=',I3,2X,'T-SCALE='
     *,F7.3,2X,'X-Y SCALE=',F7.3)
      PRINT51,IP
 51    FORMAT(' ','RANDOM NET',I5)
      IX=IN
      I=0
 32   L=0
      I=I+1
 31   CALL FANDU(IX,IY,YFL)
      IX=IY
      L=L+1
C
C     THE THIRD TIME THROUGH THE TRAFFIC NUMBER IS GENERATED.
C
      IF(L.EQ.3) GO TO 32
      XT(I,L)=YFL*XNSC
      IF(L.EQ.2) GO TO 33
      ZIZ=YFL*XNSCT+1.
      IIZ=INT(ZIZ)
C
C     IF IIZ IS GREATER THAN THE NUMBER OF MAXIMUM TRAFFIC UNITS THEN
C     IT IS SCALED DOWN.
C
      IF(IIZ.LE.IT) GO TO 35
      ZIZ=ZIZ/IT
      II=INT(ZIZ)
      IR=II*IT-1
      T(I)=IIZ-IR
      GO TO 33
 35   T(I)=IIZ
```

```
 33    IF(I.NE.NT+1) GO TO 31
       PRINT41
 41    FORMAT(' ','NODE',5X,'X',9X,'Y',5X,'TRAFFI C')
       DO 43 J=1,NT
          PRINT44,J,XT(J,1),XT(J,2),T(J)
 44       FORMAT(' ',I4,3X,F6.2,2X,F6.2,2X,I4)
          PUNCH49,XT(J,1),XT(J,2),T(J)
 49       FORMAT(F7.2,2X,F7.2,2X,I4)
 43    CONTINUE
       STOP
       END
       SUBROUTINE FANDU(IX,IY,YFL)
C
C      THIS SUBROUTINE GENERATES A RANDOM INTEGER IY AND A RANDOM REAL
C      NUMBER BETWEEN 0.0 AND .999
C      THE METHOD USED IS THE POWER RESIDUE METHOD AND THE PROGRAM IS
C      ESSENTIALLY THE SAME AS THE RANDOM NUMBER GENERATOR IN
C      THE IBM SSP LIBRARY.
C
       IY=IX*65539
       IF(IY)5,6,6
 5     IY=IY+2147483647+1
 6     YFL=IY
       YFL= YFL*.4656613E-9
       RETURN
       END
$ENTRY
```

VITAE

NAME: Hugh G. Dysart

DATE AND PLACE OF BIRTH : Ottawa, Ontario, Canada, 1952

EDUCATION: B. Sc. (Electrical Engineering), May 1974.
Queen's University, Kingston, Ontario