SPANNING TREES WITH FIXED CHARGES

Working Paper No. 372

Ram Mohan V. Rachamadugu
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan

Revised January 1984

372

Spanning Trees with Fixed Charges

Ram Mohan V. Rachamadugu
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan

Revised January 1984

Spanning Trees with Fixed Charges

Abstract

This paper analyzes a class of problems in network design that consists of opening plants and choosing links such that there exists a path from each customer to at least one of the plants that is open, either directly or through other customer nodes. The objective is to minimize the total costs of opening the plants and the chosen links. A two stage procedure is devised. The first stage reduces the customers to a set of "clusters," considerably reducing the problem size. The second stage is a branch-and-bound code which exploits the special structure of the problem and uses an efficient minimum spanning tree algorithm (Prim's version) to solve the sub-problems at each note in the branch-and-bound code. Computational results are encouraging: for most problems of size up to 25 plants and 130 customers, an optimum solution was found in much less than one minute of CPU time on the DEC-20 system. Extensions and applications are discussed.

1.0 Introduction and Summary

This paper addresses a special class of plant location problems. The problem to be considered is as follows: A set of potential plant locations are given. There is a fixed cost F_i associated with the opening of plant i, the cost depending upon the plant location. There is a set of customers with known locations. Also given are the costs of linking any two customers or a customer and a plant. It is required that each customer must have a path to at least one open plant. The objective is to minimize the total costs. Some of the practical problems which can be modeled on these lines are mentioned below.

Consider the problem of connecting terminals on a wiring board with minimum total wire length [10]. The problem is obviously one of finding minimum cost spanning tree. There exist efficient algorithms for solving the problem [7, 8, 12]. However, we consider the following modification of the wiring problem--suppose that the wiring board already consists of some potential "sites" which are already connected, but there are fixed costs associated with establishing lead contacts at these points. We wish to minimize the total costs associated with establishing lead contacts and connecting the terminals on the board to some lead contact or the other that has been established.

As another example, consider electrical energy distribution. In an urban context, the sites where we can locate sub-stations are limited in number. All potential customers will have to be linked to some sub-station. There are capital costs associated with opening sub-stations and link costs. We assume that the marginal cost of capacity of sub-station is constant and the differences in distributional losses between various configurations is not significant. In such a case, the problem reduces to one of the above kind.

As a final example, consider the problem of concentrators (plants) location in a distributed data processing network with one data processing facility. We wish to connect the data terminals to any one of the concentrations that we locate. The objective is to minimize the total cost of connecting the concentrators to data processing facility and costs of locating a path from each remote terminal to any one of the concentrators that have been located.

A two-stage procedure is devised to solve the problem. The first stage reduces the customers in the original problem to a set of "clusters," each of which can be treated as a pseudo-customer for computational purposes. It is established that asymptotically the number of "clusters" equals the number of plant locations under consideration. The path from one customer to the other in a given "cluster" is unique and independent of which plants are kept open in an optimum solution.

The second stage of the procedure is a branch-and-bound code which exploits the structure of the problem. At any node the sub-problem is reduced to a minimum spanning tree problem for which there are efficient algorithms available [7, 8, 12]. A binary branching procedure is followed. We branch by forcing open or forcing closed an additional plant. This is done in a "greedy" way in the sense that whichever additional plant gives maximum savings is branched on. Also a pruning rule is developed.

In §3.0 the relation of this problem to the classical Steiner problem in the plane is discussed. Computational results are encouraging; for problems of the order of 25 plants and 130 customers, in most cases, optimum solution was obtained in less than one minute of CPU time on the DEC-20 system. Results are discussed in detail in §5.0. Scope for further work and possible extensions of this paper are discussed in §6.0.

2.0 Problem Statement

Let $i \in I$ represent the set of potential plant locations

$$|I| = m$$

$j \in J$ represent the set of customers

$$|j| = n$$

F_i fixed cost of opening the plant at i

d_{ij} $(= d_{ji})$ cost of the link between i and j

Since establishing a link between two plant sites has no meaning, we set $d_{ij} = \infty \ \forall \ i,j \in I$.

$$y_i = \begin{cases} 1 & \text{if plant } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

S_i Set of all customers with a path to plant i

$C(S_i)$ represents the cost of minimum cost spanning tree for $S_i \cup \{i\}$

$$(y_i = 0 \Longrightarrow S_i = \emptyset \text{ and } C(S_i) = 0)$$

The problem can be stated as follows (P)

$$\min \sum_{i=1}^{m} C(S_i) + \sum_{i=1}^{m} F_i y_i \qquad \text{\_\_\_\_(1)}$$

$$\text{s.t.} \bigcup_{i \in I} S_i = J \qquad \text{\_\_\_\_(2)}$$

$$|S_i| \leq n \, Y_i \quad \forall \ i \in I \qquad \text{\_\_\_\_(3)}$$

$$y_i = 0 \text{ or } 1$$

In an optimum solution, S_i are disjoint sets since any solution in which a customer is linked to more than one plant can be improved upon by removing at least one link. Thus, in an optimum solution, each customer will be assigned to only one plant. A beneficial result of such a solution in the electrical energy distribution problem cited in the introduction is that the configurations will be radial, thus eliminating the need for switching.

3.0 Some Comments on Problem Formulation

In the problem as stated in (P), it has been implicitly assumed that the feasible solution set consists of only trees which can branch at customer locations. However, in reality, such is not necessarily the case. As an illustration, consider the three points shown in figure 1. The MST shown in figure 1 is longer than the tree shown in figure 2. Node S in figure is known as Steiner Point and the tree in figure 2 is known as Steiner Tree. Though the Steiner Tree problem has been the subject of considerable investigation [2, 4[, 11], the problem becomes complex for $n \geq 3$.

Though, in reality, our objective would be to find a solution that minimizes the sum of fixed charges associated with opening the plants and the cost of minimum Steiner Trees that span the plants and customers, because of the complexity of Steiner Tree problem, we confine our attention to the problem (P). However, we note that the optimum solution obtained for (P) in fact becomes upper bound for the Steiner Tree problem. It has been pointed out by Gilbert and Pollak [6] that in the Euclidean plane, the ratio of the lengths of minimum Steiner Tree to MST cannot be less than $\sqrt{3}/2$. This result provides us an idea of the extent to which the solution obtained for our problem deviates in the worst case from the optimum solution for minimum cost Steiner Tree problem.

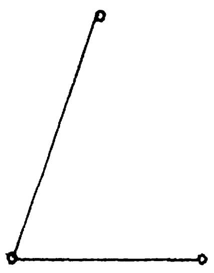


Figure 1

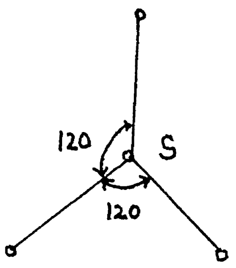


Figure 2

4.0 A Solution Procedure

The solution procedure presented here consists of two stages. At the first stage, the problem data is preprocessed so that the number of customers is reduced to a smaller number of "clusters" which can be treated as "pseudo-customers" for computational purposes. This has the following implications--

(i) The path from one customer to the other within the same cluster is unique and is independent of which plants are kept open in an optimum solution.

(ii) Since the number of clusters of customers is generally much less than number of customers, the computational time required to solve the minimum spanning tree problem at each node in the second stage of the solution procedure is considerably reduced.

In the second stage, we use a branch-and-bound method on the processed data from stage I to determine which plants are to be kept open in an optimum solution.

4.1: Stage I

An intuitive explanation of the stage I clustering procedure is as follows: any customer whose closest neighbor is not a plant site will not be linked to a plant directly in an optimum solution. Further, once we decide on which plants are to be kept open the solution to the problem is obtained using the minimum spanning tree algorithm. The minimum spanning tree algorithm version of Prim [7, 13] can be initialized at any node and if we were to initialize at a customer location whose closest neighbor is not a plant, then the link with the least cost and with this customer node as a vertex will always be in an optimum solution. However, as is clear from the argument, this is independent of which plants are kept open. Let the vertices of such a link be a and b. Link (a,b) will always be included in an optimum solution. Now, we can collapse vertices a and b into one cluster (or sub-tree), say c. We grow the cluster till the closest neighbor to a cluster is a plant site. The procedure is applied to all customer nodes.

A description of the Pre-processor, along with the proof of the procedure and some asymptotic results is given below.

Let F represent a spanning forest of J. Initially F consists of m components (clusters), with each customer being a component denoted by j, j=1 to m. Let T_j represent all the vertices (customers) in j.

Define

$$\begin{aligned} d_{jq} &= \min_{\mu \in T_j} \min_{s \in T_q} d_{\mu s} \quad \text{if } j, q \in F \\ &= \min_{\mu \in T_j} d_{\mu q} \quad \text{if } j \in F,\ q \in I \end{aligned}$$

The procedure is as follows:

```
Begin
  0   Set up the spanning forest F with each customer $j \in J$ as a
          component,
  1   REVK $\leftarrow$ n
  2   $\ell \leftarrow 1$
  3   Find minimum $d_{\ell q}$. If $q \not\in I$, go to step 6
  4   $\ell \leftarrow \ell + 1$
  5   If $\ell \leq n$ go to 3. Otherwise go to 9.
  6   Merge $T_\ell$ and $T_q$ i.e., assign all customers in $T_q$ to $T_\ell$
  7   $n \leftarrow n - 1$
  8   If $n <$ REVK, go to 4
```

```
 9    If $n < REVK$ go to 11
10    Stop
11    $REVK \leftarrow n$, go to 2.
End
```

Let j be a component (cluster) in F. We determine q^* such that

$$d_{jq^*} = \min d_{jq} \qquad \begin{array}{c} T_q \in F/T_j \\ a \in I \end{array}$$

If $q^* \in F$, let $d_{jq^*} = d_{\ell k}$ $\ell \in T_j$, $k \in T_{q^*}$. It is shown in [3] that the link (ℓ,K) is a link in the minimum spanning tree. The proof idea is as follows: Suppose at some stage we have components which are part of an optimum solution. To arrive at an optimum solution, all components must be connected. Consider any component, T_j. This must be linked to some other component. Suppose (ℓ,k) is not in optimum solution. This implies that there is an alternative path from T_j to T_{q^*} consisting of some link other than (ℓ,k) in the optimum solution. However, this contradicts optimality since such a solution can be improved by including (ℓ,k) and eliminating the other link incident at one of the vertices in T_q. We note that this is true regardless of which T_j in F is under consideration. Also, note that whenever the closest node or cluster to any T_j in F is a potential plant location side, we stop "growing" T_j. Otherwise we merge components T_j and T_{q^*}. Thus the components so formed will always be a part of an optimum solution, regardless of which plants are kept open in an optimum solution.

This procedure generally reduces customers to a small number in problems where the number of customers is large, as is likely to be the case in practical problems. This reduces the computation time for the branch-and-bound code.

An interesting result in problems where customers and plants are randomly located follows:

<u>Proposition</u>: When the number of customers is very large, and the plants and customers are randomly located, the number of "clusters" asymptotically reduces to the number of plants.

<u>Proof</u>: Let m be the number of plants and n be the number of customers. Since plants and customers are randomly located, probability that a customer has a plant as closest neighbor is

$$\frac{m}{m + n - 1}$$

$\therefore$ Expected number of customers having plants as the closest neighbors

$$E = \frac{mn}{m + n - 1} = \frac{m}{1 + \frac{m-1}{n}}$$

$$\lim_{n \to \infty} E = m$$

Additionally, this implies that, asymptotically, as n becomes large in problems where plants and customers are randomly located, not more than m customers are directly linked to any one plant in an optimum solution.

We apply the branch-and-bound code developed in stage II after the data has been pre-processed through stage I.

4.2 <u>Stage II - Branch and Bound Methodology</u>

We note that y_i's are strategic variables in the problem, i.e., once we find which plants are open, the problem of determining which links are to be established is easily solved using minimum cost spanning tree algorithm, as will be explained later in this section.

A branch-and-bound code is developed to solve the problem using depth-first strategy in order to minimize the storage requirements. Lower bounds at various nodes on the branch-and-bound tree are obtained using the Minimum Spanning Tree algorithm. Consider any node K on the branch-and-bound tree. Let

K_0 represent the set plants forced "closed" i.e., $y_i = 0 \; \forall \; i \in K_0$

K_1 represent the set of plants forced "open" i.e., $y_i = 1 \; \forall \; i \in K_1$

K_2 represent the set of plants which are free to be open or closed $K_2 = I/(K_0 \cup K_1)$

LB_K represent the lower bound on the value of the problem at node K

UB_K represent the upper bound on the value of the problem at node K

LB_0 represent the value of the problem with $F_i = 0 \; \forall \; i \in I$

<u>Lower Bound Evaluation</u>

$$LB_K: \; L_K + \sum_{i \in K_1} F_i$$

L_K represents the cost of minimum cost spanning tree with all plants in $K_1 \cup K_2$ being kept open at zero cost and plants in K_0 being closed. This can easily be done using efficient minimum cost spanning tree algorithms. Considerable amount of flexibility exists in the choice of the algorithm.

Two of the widely publicized algorithms are by Prim [7, 13] and Kruskal [8]. Prim's algorithm basically starts at any node and gradually builds the tree by successively including a node in the tree that is closest to the tree and not already included in it. Kruskal's algorithm, on the other hand, starts with a completely disconnected graph. It orders links in ascending order and then starting from the top of the list, gradually adds links such that no closed circuit is formed and n - 1 links are added in the tree (n is the number of nodes in the graph). From computational effort point of view, Prim's algorithm has a worst case bound of $O(n^2)$ and Kruskal's algorithm has a worst case bound of $O(n^3)$. As pointed out by Bradley [1], for direct implementation, Prim's algorithm is better unless the network is relatively sparse. As can be seen, when the network is dense, considerable effort is expended in sorting the links in Kruskal's algorithm. However, it has been reported [1] that with the use of new sorting techniques, Kruskal's algorithm gives good results for both sparse and dense networks.

For our purpose, we have used the Prim's algorithm as structured by Whitney [7]. The choice was made primarily because of the simplicity of the algorithm and its relatively better performance over Kruskal's algorithm for dense networks. The problem data was modified by creating a super node 0 such that

$$d_{oi} = 0 \quad \forall \quad i \in I$$
$$d_{oj} = \infty \quad \forall \quad j \in J$$
$$d_{ij} = \infty \quad \forall \quad i \in K_o, \ j \in J.$$

After modifying the data at each node in the above mentioned fashion, the minimum cost spanning tree algorithm is directly applied.

Upper Bound Evaluation

An obvious upper bound for the solution at node K is

$$U\,B_K = L_K + \sum_{i \in K_1 \cup K_2} F_i$$

However, we can improve the upper bound using the following procedure:

Consider any plant $i \in K_2$ and all the customers who are directly linked to i in the solution LB_k. Evaluate the possibility of closing the plant i and assigning all customers directly linked to plant i to the plants in K_1. If the resulting increase in costs of the links is less than F_i, then plant i is closed and UB_K is improved. This process can be repeated for all nodes $i \in K_2$.

A modification of the above procedure would be to consider linking up customers directly connected to plant $i \in K_2$ to either the plants in K_1 or customers who are directly connected to plants in K_1. This may improve the tightness of the bound, but would likely be computationally more expensive. [In the code that has been developed, the former procedure has been used. It seems to yield fairly good results. After utilizing the above mentioned procedures in one form or the other, we know which plants are to be kept open in the solution for the upper bound. Let K_3 represent the set of these plants. $K_3 \supseteq K_1$.]

Branching

Consider any plant $i \in K_3/K_1$. Estimate the savings realisable by closing plant i by reassigning all customers directly linked to i to the alternative cheapest plant in K_3/i. For all plants in K_2/K_3, which are already closed in the Upper Bound solution for node K, the saving due to closing is treated as 0. Branch on the plant that offers maximum savings.

If maximum savings is non-negative (say, for plant q), branch with $(K+1)_0 = K_0 \cup q$ and $(K+1)_1 = K_1$ (i.e., plant q is "closed" at node K + 1) and $K_0 = K_0$, $K_1 = K_1 \cup q$. Similar extensions apply when maximum savings is negative.

Pruning

Let $\overline{UB}$ represent currently available best upper bound. We can prune the tree at any node K is

$$LB_0 + \sum_{i \in K_1} F_i \geq \overline{UB}$$

Needless to say, the node K is fathomed if $LB_K = UB_K$.

5.0 Computational Study

A computer code was written in FORTRAN IV for the above mentioned branch-and-bound procedure and tested out on randomly generated problems on the DEC-20 system.

The problem generator essentially locates a random number of plants and customers in a random way on a square of side length 200 units. The link costs were made proportional to the Euclidean distance between the nodes plus a random value imposed on it which was, in turn, a random variable with uniform distribution (mean: 0, variance: Euclidean distance^2/3). For each of the above sets of data, three problems were created, each with fixed charges for plants being generated randomly using uniform distributions (0, 150), (0, 300) and (0, 600). The number of links in the problems were chosen randomly from a uniform distribution (0, $m+n_{C_2}$).

Table 1 provides the computational results which are encouraging. Except in one case, the optimum solution was found in less than one minute of CPU time for problems of the order of 25 plants and 130 customers. Obviously, as we decrease the fixed charges, the computation time for the branch-and-bound code increases. It is to be noted that the computation time at the clustering stage is independent of the fixed charges. However, the time for the branch-and-bound code depends on the number of plants and number of "clusters."

No.	Number of Plants	Number of Customers	Ratio of Numbers of "Clusters" to Number of Plants	CPU Time for Pre-Processing (in Secs)	Maximum fixed charge = 150		Maximum fixed charge = 300		Maximum fixed charge = 600	
					Total CPU Time (Secs)	Proportion of Nodes Searched in Branch-and-Bound Tree (%)	Total CPU Time (Secs)	Proportion of Nodes Searched in Branch-and-Bound Tree (%)	Total CPU Time (Secs)	Proportion of Nodes Searched in Branch-and-Bound Tree (%)
1	4.	83.	1.50000	0.203000	0.0260	25.0000	0.0280	25.0000	0.0260	25.0000
2	4.	96.	0.750000	0.264000	0.0210	28.1300	0.0200	28.1300	0.02200	28.1300
3	5.	85.	1.20000	0.188000	0.0500	26.5600	0.0490	26.5600	0.0470	23.4400
4	7.	88.	0.28571	0.235000	0.0490	6.2500	0.0480	6.2500	0.0510	6.2500
5	7.	125.	2.28571	0.545000	0.2170	8.9800	0.1830	6.6400	0.1420	5.8600
6	8.	127.	1.50000	0.523000	0.3190	8.2000	0.2040	5.0800	0.1820	4.4900
7	10.	117.	1.90000	0.504000	2.3740	8.6900	0.6380	2.2900	0.4360	1.5600
8	10.	130.	1.60000	0.583000	0.5630	2.1500	0.3430	1.4200	0.3320	1.3700
9	11.	82.	1.09091	0.237000	4.6540	12.2800	3.2180	8.1800	2.0730	5.3000
10	11.	119.	1.18182	0.470000	0.8210	2.0300	0.3980	1.0000	0.3020	0.7300
11	13.	121.	1.30769	0.533000	0.7540	0.3100	0.5140	0.1900	0.4400	0.1600
12	14.	111.	1.14286	0.446000	1.1060	0.2200	0.5030	0.1000	0.4640	0.0900
13	14.	124.	1.64286	0.688000	3.5610	0.5200	1.0740	0.1500	1.0580	0.1500
14	16.	113.	0.87500	0.499000	2.1820	0.1100	1.0890	0.0500	1.0310	0.0500
15	17.	103.	0.88235	0.434000	5.0550	0.1200	1.2110	0.0300	0.8430	0.0200
16	17.	116.	1.11765	0.569000	10.8960	0.2000	5.7260	0.1100	2.9790	0.0500
17	19.	105.	1.73684	0.641000	**		**		74.7550	0.1800
18	20.	108.	0.95000	0.519000	3.9090	<0.0100	2.3510	<0.0100	1.2340	<0.0100
19	21.	110.	1.14286	0.664000	8.7970	<0.0100	3.1840	<0.0100	1.5730	<0.0100
20	22.	97.	0.95455	0.433000	3.9570	<0.0100	2.6120	<0.0100	1.5250	<0.0100
21	23.	99.	1.04348	0.612000	4.6890	<0.0100	2.6870	<0.0100	2.3900	<0.0100
22	24.	102.	1.20833	0.710000	13.4100	<0.0100	9.0720	<0.0100	8.4780	<0.0100
23	25.	89.	0.84000	0.455000	45.9210	<0.0100	21.4400	<0.0100	7.5490	<0.0100
24	26.	91.	1.11538	0.604000	**		36.0300	<0.0100	12.6600	<0.0100
25	27.	94.	0.85185	0.540000	**		**		**	

Table - 1

** indicates job aborted after 2 minutes of CPU time.

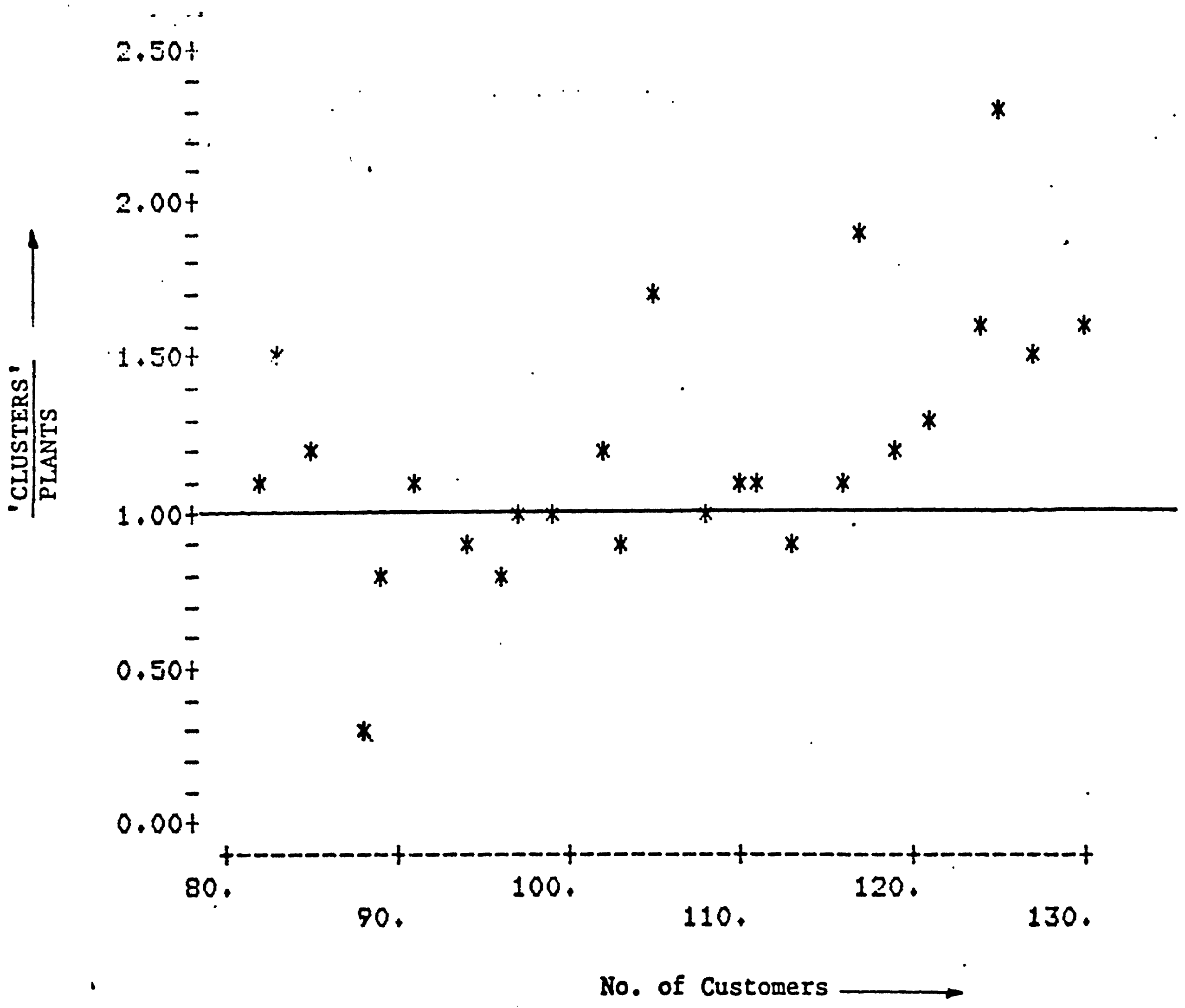
Figure 3 provides scatter plot for the ratio of number of clusters to number of plants against the number of customers. As has been proved earlier, this ratio should asymptotically approach 1.0. In the case of test problems, the average of this ratio was 1.129. We also note from Table 1 that the proportion of nodes generated on the branch-and-bound tree decreases as the number of plants increases. This is a necessary feature for the branch-and-bound code to be effective.

Table 2 provides the regression results for the computation times for the range of problems that have been tested. At the clustering stage of computation time for the range of problems tested seems to be of the order of $(\text{plants})^{0.40}$ $(\text{customers})^{1.82}$. As noted earlier, the computation time for the branch-and-bound code depends on the number of "clusters" and the number of customers. However, asymptotically, the number of "clusters" will approach the number of "plants." This implies high degree of correlation between the number of "clusters" and the number of plants. (0.835 for test problems). So, for prediction purposes, we may as well take only the number of plants as the independent variable. These observations are evident from Table 2.

6.0 Conclusion

Preliminary results indicated in §5.0 seem to be encouraging. We have used the basic version of Prim's algorithm in its simplest form [7] for estimating the minimum cost spanning tree. However, for problems of much larger size than the ones we have tested (say, of the order of 600 customers or so) we could use more sophisticated versions of the algorithms available [1] using recent developments in sorting and storing techniques.

In electric power distribution problems, the aim would be to achieve overall optimization--minimizing the sum of capital and operational costs.



2.50
2.00
1.50
1.00
0.50
0.00
'CLUSTERS'/PLANTS
80.
90.
100.
110.
120.
130.
No. of Customers

Figure 3

Table - 2

Results of Regression Analysis

for Pre-Processor

$$\log(\text{CPU Time}) = \underset{(.853)}{-10.374} + \underset{(.0478)}{0.4092}\,\log(\text{no. of plants}) + \underset{(.186)}{1.825}\,\log(\text{no. of customers})$$

$R^2 = 0.916$

0.908, adjusted for degrees of freedom

For Branch and Bound Code

MAXF	REGRESSION EQUATION	R^2	R^2, adjusted for degrees of freedom
150	$\log(\text{CPU Time}) = \underset{(.71)}{-8.725} + \underset{(.457)}{2.769}\,\log(\text{no. of plants}) + \underset{(.382)}{0.781}\,\log(\text{no. of clusters})$	0.897	0.886
300	$\log(\text{CPU Time}) = \underset{(.667)}{-8.194} + \underset{(.429)}{2.591}\,\log(\text{no. of plants}) + \underset{(.359)}{0.531}\,\log(\text{no. of clusters})$	0.884	0.872
600	$\log(\text{CPU Time}) = \underset{(.564)}{-7.709} + \underset{(.363)}{2.325}\,\log(\text{no. of plants}) + \underset{(.303)}{.496}\,\log(\text{no. of clusters})$	0.912	0.908

Note: Bracketed figures indicate standard deviation of the coefficients.

Number of observations: 22

In such cases, our procedure can be used to arrive at "good" solutions, though not necessarily an optimum solution. The procedure would be as follows: Disregard distribution or operational costs and arrive at the configuration which minimizes the total capital costs by the procedure in this paper. Having decided which "plants" (sub-stations) are to be kept open, use the procedure suggested by Gabow [14] for listing the spanning trees in the increasing order of link costs. We can proceed down the list and calculate operational costs using network based codes and stop when the sum of operational costs and capital costs is minimum. This procedure may not be applicable when the demands vary considerably among customers.

In the problem we have addressed, no constraints were imposed regarding the number of plants that can be opened. It is possible to consider an extension of this work to such a problem. In this case, the Pre-processor still remains effective and the branch-and-bound code can suitably be modified. We note that even in this case, in an optimum solution, no customer will have a path to more than one plant. Further, the computation times for branch-and-bound code will reduce since we do not have to search beyond the kth level in the branch-and-bound tree where k is the maximum number of plants that can be opened. More difficult extensions are the case where the plants have bounds on capacities and where plants have bounds on capacities, it is clear that in the optimum solution a customer could have paths to more than one plant. This is shown in an example in Figure I in the Appendix. In the case where plant capacities exhibit economies of scale, it is clear that in the optimum solution no customer will have a path to more than one plant. However, the clustering procedure will not be valid. An example for this case is shown in Figure II in the Appendix. These cases merit further investigation.

References

1. Bradley, G. H., "Survey of Deterministic Networks," AIIE Transactions, Vol. 7, No. 3, pp. 222-234, September 1975.

2. Chang, S. K., "The Generation of Minimal Trees with Steiner Topology," Journal of ACM, Vol. 19, 1972, page 699.

3. Christofides, N., Graph Theory: An Algorithmic Approach, Academic Press, New York 1975, pp. 135-137.

4. Cockayne, E. J., "On the Efficiency of the Algorithm for Steiner Minimal Trees," Journal of SIAM on Applied Mathematics, Vol. 18, p. 150.

5. Gabow, H. N., "Two Algorithms for Generating Weights Spring Trees in Order," SIAM J. on Computing, Vol. 6, No. 1, pp. 139-150, March 1977.

6. Gilbert, E. N., and Pollak, H. O., "Steiner Minimal Trees," SIAM Journal on Applied Mathmatics, Vol. 16, 1968, pp. 1-29.

7. Kevin, V. and Whitney, M., "Algorithm 422: Minimal Spanning Tree [11]," Communications of the ACM, April 1972, Vol. 15, No. 4, pp. 273-274.

8. Kruskal, J. B., "On the Shortest Spanning Subtree of a Graph and Travelling Salesman Problem," Proceedings of American Mathematical Society, Vol. 7, pp. 48-50, 1956.

9. Lawler, E. L., "Combinatorial Optimization: Networks and Malroids," pp. 291-296, Hold, Rinehart and Winston Inc., 1976.

10. Loberman, H. and Weinberger A., "A Formal Procedure for Connecting Terminals with a Minimum of Total Wire Length," Journal of ACM, Vol. 4, October 1957, pp. 428-437.

11. Melzak, Z. A., "On the Problem of Steiner," Canadian Mathematical Bulletin, Vol. 4, p. 335.

12. Prim, R. C., "Shortest Connection Networks and Some Generalizations," Bell System Technical Journal, Vol. 36, Nov. 57, pp. 1389-1401.

APPENDIX

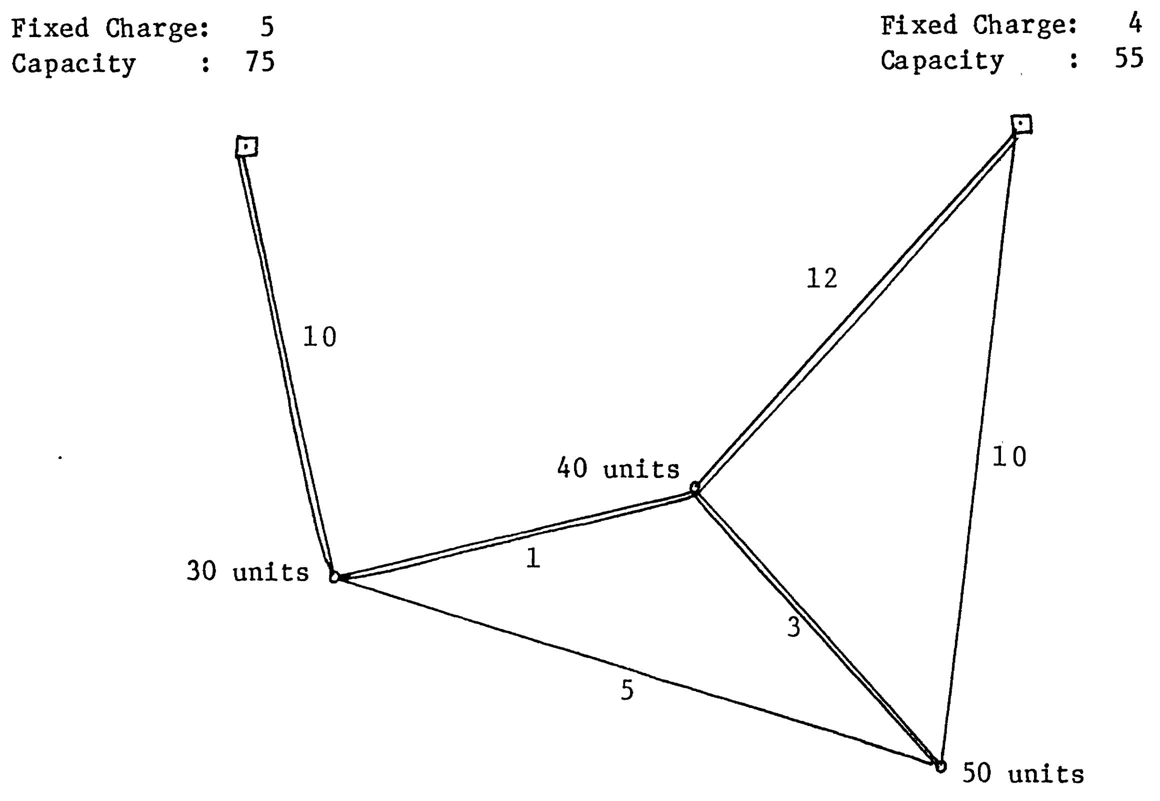


FIGURE I

Example for the case where plants have capacity limits and customers have paths to more than one plant in the optimum solution.

Double lines indicate the optimum configuration and all customers have paths to both plants A and B.

cont'd

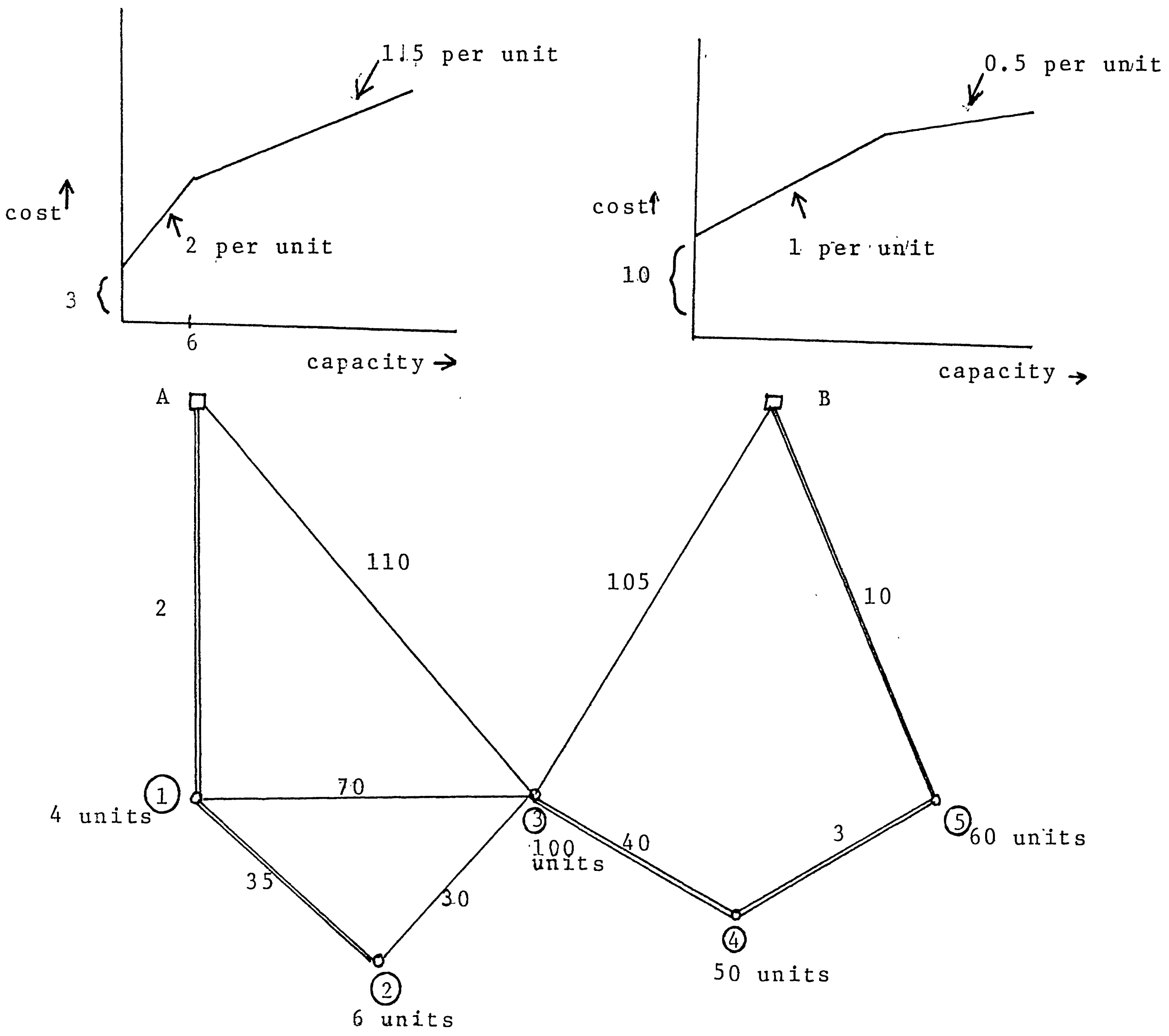


FIGURE II

Example for the case where plants have Economies of Scale

As per clustering stage, 1, 2, and 3 form a cluster and 4 and 5 form a cluster. However, in the optimum solution, 1 and 2 form a cluster and 3, 4, and 5 form a cluster. Double lines indicate the optimum solution.