Least Cost Multicast Spanning Tree Algorithm for Local Computer Network

Yong-Jin Lee[1] and M. Atiquzzaman[2]

[1] Department of Computer Science, Woosong University,
17-2 Jayang-Dong, Dong-Ku, Taejon 300-718, Korea
yjlee@woosong.ac.kr

[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract. This study deals with the topology discovery for the capacitated minimum spanning tree network. The problem is composed of finding the best way to link nodes to a source node and, in graph-theoretical terms, it is to determine a minimal spanning tree with a capacity constraint. In this paper, a heuristic algorithm with two phases is presented. Computational complexity analysis and simulation confirm that our algorithm produces better results than the previous other algorithms in short running time. The algorithm can be applied to find the least cost multicast trees in the local computer network.

1 Introduction

Topology discovery problem [1,2] for local computer network is classified into capacitated minimum spanning tree (CMST) problem and minimal cost loop problem [3]. The CMST problem finds the best way to link end user nodes to a backbone node. It determines a set of minimal spanning trees with a capacity constraint. In the CMST problem, end user nodes are linked together by a tree that is connected to a port in the backbone node. Since the links connecting end user nodes have a finite capacity and can handle a restricted amount of traffic, the CMST problem limits the number of end user nodes that can be served by a single tree. The objective of the problem is to form a collection of trees that serve all user nodes with a minimal connection cost.

Two types of methods have been presented for the CMST problem - exact methods and heuristics. The exact methods are ineffective for instances with more than thirty nodes. Usually, for larger problems, optimal solutions can not be obtained in a reasonable amount of computing time. The reason is why CMST problem is NP-complete [4]. Therefore, heuristic methods [5,6,7] have been developed in order to obtain approximate solutions to the problem within an acceptable computing time. Especially, algorithm [5] is one of the most effective heuristics presented in the literature for performance evaluation.

In this paper, new heuristic algorithm that is composed of two phases is presented. This paper is organized as follows. The next section describes the modeling and algorithm for the CMST problem. Section 3 discusses the performance evaluation and section 4 concludes the paper.

2 Modeling and Algorithm

The CMST problem is represented in Fig. 1. Eq. (1) is the formulation for the CMST problem.

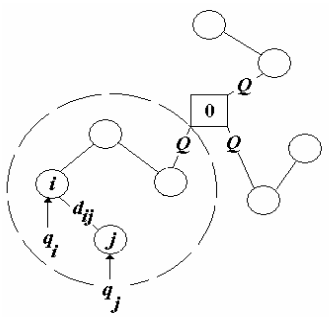

Fig. 1. CMST problem

The objective of the CMST problem is to find a collection of the least-cost spanning trees rooted at the source node. n represents the number of nodes. d_{ij} and q_i are distance between node pair (i, j) and traffic requirement at node i (i=1,..,n) respectively. Q shows the maximum traffic to be handled in a single tree and T_k is the k^{th} tree which has no any cycles.

$$\begin{aligned} &\textit{Minimize} \ \sum_{i,j} d_{ij}\, x_{ij} \\ &S.T. \\ &\sum_{i,j \in T_k} q_i\, x_{ij} \le Q, \quad \forall k \\ &\sum_{i,j} x_{ij} = n \\ &x_{ij} = 0 \ or \ 1 \end{aligned} \tag{1}$$

A particular case occurs when each q_i is equal to one. At the time, the constraint means that no more than Q nodes can belong to any tree of the solution. In this paper, we present a heuristic that consists of two phases for the CMST problem. In the first phase, using the information of trees obtained by the EW solution(we will call algorithm [5] as EW solution), which is one of the most effective heuristics and used as a benchmark for performance evaluation, we improve the solution by exchanging nodes between trees based on the suggested heuristic rules to save the total linking cost. In the second phase, using the information obtained in the previous phase, we transfer nodes to other tree in order to improve solutions.

EW solution performs the following procedure: It first compute $g_{ij} = d_{ij} - C_{ij}$ for each node pair (i,j). d_{ij} and C_i represent cost of link (i,j) and the minimum cost between a source node and node set of tree containing node i respectively. At the initialization, it sets $C_i = d_{i0}$. Then, it finds the node pair (i,j) with the minimum negative g_{ij} (we do not consider node pair's with the positive g_{ij} value). If all g_{ij}'s are positive, algorithm is terminated. Next, it check whether the connecting node i and j satisfies the traffic capacity constraint and forms a cycle together. If no, it sets $g_{ij} = \infty$ and repeats the above check procedure. Otherwise, it connects node i and j and delete the link connecting a source node and tree with the higher cost between C_i and C_j. Since new tree's formation affects C_i in EW solution, g_{ij} values have to be recomputed. When the number of nodes except the source node is n, the EW solution provides the near optimum solution with a memory complexity of $O(n^2)$ and a time complexity of $O(n^2 log\ n)$ for the CMST problem.

We will improve the EW solution by the simple heuristic rules based on the node exchange and transfer between two different trees. Starting from the trees obtained by EW solution, we first exchange nodes between different trees based on the trade-off heuristic rules (ks_{ij}). It is assumed that node i is included in $node(inx1)$, node j is included in $node(inx2)$, and $inx1$ is not equal to $inx2$. $inx1$ and $inx2$ represent indices of trees including node i and node j respectively. In addition, $node(inx1)$ and $node(inx2)$ represent sets of nodes included in tree $inx1$ and $inx2$ respectively. Exchange heuristic rule, ks_{ij} is defined as $C_{inx1} + C_{inx2} - d_{ij}$. C_{inx1} is the least cost from nodes included in tree $inx1$ to the root (source node). That is, C_{inx1} = Min $\{d_{m0}\}$ for $m \in node(inx1)$, $j \in node(inx1)$. Also, C_{inx2} = Min $\{d_{m0}\}$ for $m \in node(inx2)$, $j \in node(inx2)$. If $inx1$ is equal to $inx2$, both node i and node j are included in the same tree, trade-off value is set to $-\infty$. Since the sum of node traffic must be less than Q, Both $\sum_{m \in node(inx1)} q_m + q_j - q_i \leq Q$ and $\sum_{m \in node(inx2)} q_m + q_i - q_j \leq Q$ must be satisfied. Otherwise, ks_{ij} is set to $-\infty$.

An initial topology is obtained by applying EW solution. For each node pair (i, j) in different trees, heuristic rules (ks_{ij}'s) are calculated and ks_{ij}'s with negative value are discarded. From node pair (i, j) with the maximum positive value of ks_{ij}, by exchanging node i for node j, two new node sets are obtained. The network cost by applying the existing unconstrained minimum spanning tree algorithm [8] to two new sets of nodes is obtained. If the computed cost is less than the pervious cost, the algorithm is repeated after re-computing heuristic rules (ks_{ij}'s). Otherwise the previous ks_{ij}'s are used. If all ks_{ij}'s are negative and it is impossible to extend trees further, we terminate the algorithm.

Node transfer procedure is described as the follows: we improve solutions by transferring nodes from one tree to another tree based on node transfer heuristic rule (ps_{ij}). We first evaluate that the sum of traffics in every tree is equal to Q. If so, the algorithm is terminated. Otherwise, the node pair (i, j) with the minimum negative value of ps_{ij} is found. By transferring node j to the tree including node i, the solution is computed. If $inx1$ is equal to $inx2$ or the sum of traffic is greater than Q, node j can not be transferred to the tree $inx1$. In this case, ps_{ij} is set to ∞. Otherwise, transfer heuristic rule, ps_{ij} is defined as $d_{ij} - d_{max}$. Here, d_{max} = Max $\{C_{inx1}, C_{inx2}\}$.

If each trade-off heuristic rule (ps_{ij}) is positive for all node pair (i, j), and no change in each node set is occurred, we terminate the algorithm. From the above modeling for the CMST problem, we now present the following procedure of the proposed algorithm. In the algorithm, step 2 and step 3 perform node exchange and transfer respectively.

```
Algorithm: Least-Cost Multicast Spanning Tree
Variable: {$TEMP_{cost}$: network cost computed in each step of the algorithm
           $EW_{cost}$: network cost computed by EW solution
           $NEW_{cost}$: current least network cost
           $lcnt$: the number of trees included in any topology }
Step 1: Execute the EW solution and find the initial topology.
Step 2: A. Perform the node exchange between two different trees in the initial topology.
(1) set $TEMP_{cost} = EW_{cost}$. (or set $TEMP_{cost} = NEW_{cost}$ obtained in Step 3)
(2) For each node pair $(i, j)$ in different trees $(i < j,\ \forall\ (i, j))$, compute $ks_{ij}$.
    if $(ks_{ij} < 0),\ \forall\ (i, j)$, goto B.
(3) while $(ks_{ij} > 0)$ {
    1) For node pair $(i, j)$ with the maximum positive $ks_{ij}$,
       exchange node $i$ for node $j$ and create $node(inx1)$ and $node(inx2)$.
    2) For $node(inx1)$ and $node(inx2)$, by applying unconstrained MST algorithm,
       compute $TEMP_{cost}$.
    3) if $(TEMP_{cost} \geq NEW_{cost})$, exchange node $j$ for node $i$. set $ks_{ij} = -\infty$ and repeat (3).
       else set $NEW_{cost} = TEMP_{cost}$. set $ks_{ij} = -\infty$ and go to (2).
    } ;
B. If it is impossible to extend for all trees, algorithm is terminated.
   Otherwise, proceed to step 3
Step 3: A. Perform the node transfer between two different trees obtained in Step 2.
(1) For all $p$, $(p=1,2,..,lcnt)$, if ( $\sum_{i \in p} W_i == Q$), algorithm is terminated.
    else set $NEW_{cost} = TEMP_{cost}$.
(2) For each node pair $(i, j)$ in different trees $(i < j,\ \forall\ (i, j))$, compute $ps_{ij}$.
    if $(ps_{ij} \geq 0),\ \forall\ (i, j)$, goto B.
(3) while $(ps_{ij} < 0)$ {
    1) For node pair $(i, j)$ with the minimum negative $ps_{ij}$,
       transfer node $j$ to $node(inx1)$ and create new $node(inx1)$ and $node(inx2)$.
    2) For $node(inx1)$ and $node(inx2)$, by applying unconstrained MST algorithm,
       compute $TEMP_{cost}$.
    3) if $(TEMP_{cost} \geq NEW_{cost})$, transfer node $j$ to $node(inx2)$. set $ps_{ij} = \infty$ and repeat (3).
       else set $NEW_{cost} = TEMP_{cost}$. set $ps_{ij} = \infty$ and go to (2).
    };
B. If any change in the node set is occurred, goto Step 2. Otherwise, algorithm is terminated.
```

3 Performance Evaluation

3.1 Property of the Proposed Algorithm

We present the following lemmas in order to show the performance measure of the proposed algorithm.

Lemma 1. Memory complexity of the proposed algorithm is *$O(n^2)$.*

Proof. d_{ij}, ks_{ij}, and ps_{ij} (i=1,..,n; j=1,..,n) used in step 2 ~ step 3 of the proposed algorithm are two-dimensional array memory. Thus, memory complexity of step 2 and 3 is $O(n^2)$, respectively. Memory complexity of EW solution executed in step 1 of the proposed algorithm is $O(n^2)$. As a result, total memory complexity is *$O(n^2)$.*

Lemma 2. Time complexity of the proposed algorithm is $O(n^2 \log n)$ for sparse graph and $O(Qn^2 \log n)$ for complete graph when the maximum number of nodes to be included in a tree is limited to Q.

Proof. Assuming that q_i=1, $\forall i$, Q represents the maximum number of nodes to be included in a tree. For any graph, $G = (n, a)$, the range of Q is between 2 and n-1. In the Step 2 of the proposed algorithm, trade-offs heuristic rules (ks_{ij}) are computed for each node pair (i, j) in different trees. At the worst case, the maximum number of ks_{ij}'s to be computed is $1/2(n\text{-}Q)(n+Q\text{-}1)$ for Q=2,..,n-1. In the same manner, the maximum number of ks_{ij}'s to be computed in the Step 3 is $1/2(n\text{-}Q)(n+Q\text{-}1)$ for Q=2,..,n-1. Time complexity of minimum spanning tree algorithm is shown to be $O(E \log Q)$ [8]. E is the number of edges corresponding to Q. Since the proposed algorithm uses minimum spanning tree algorithm for two node sets obtained by exchanging node i for node j in the Step 2 or transferring node j to the tree including node i in Step 3, time complexity of the computation for minimum spanning tree is $2O$ (E log Q). In the worst case, let us assume that MST algorithms are used maximum number of ks_{ij} (or ps_{ij}) times and EW solution, Step 2 and Step 3 are executed altogether. Time complexity of EW solution is known to be $O(n^2 log\ n)$. Now, let the execution time of EW solution be $\mathrm{T_{EW}}$, that of Step 2 be $\mathrm{T_{NEA}}$, and that of Step 3 be $\mathrm{T_{NCA}}$. Then, for sparse graph ($E = Q$), T_{NEA} = MAX $_{Q=2}{}^{n-1}$ T_Q = MAX $_{Q=2}{}^{n-1}[1/2(n\text{-}Q)$ $(n+Q\text{-}1)$ $O(E\ log\ Q)] = O(n^2 log\ Q)$. In the same manner, $T_{NCA} = O(n^2 log\ Q)$. Therefore, total execution time = $T_{EW} + T_{NEA} + T_{NCA} = O[\mathrm{MAX}\ (n^2 log\ n,\ n^2 log\ Q)] = O(n^2 log\ n)$. For complete graph ($E = 1/2Q(Q+1)$), $T_{NEA} = \mathrm{MAX}_{Q=2}{}^{n-1}\ T_Q$ = MAX $_{Q=2}{}^{n-1}$ $[1/2(n\text{-}Q)(n+Q\text{-}1)O(E\ log\ Q)] = O(Qn^2 log\ n)$. In the same manner, $T_{NCA} = O(Qn^2 log\ n)$. Hence, total execution time = $T_{EW} + T_{NEA} + T_{NCA} = O[\mathrm{MAX}\ (n^2 log\ n,\ Qn^2 log\ n)] = O(Qn^2 log\ n)$.

Lemma 3. All elements of trade-off matrix in the algorithm are become negative in finite steps.

Proof. Assume that ps_{ij}'s are positive for some i,j. For node pair(i,j) with the positive ks_{ij}, our algorithm set ks_{ij} to -∞ after exchanging node i for node j. At the worst case, if all node pair(i,j) are exchanged each other, all ks_{ij} are set to -∞. Since trade-off matrix has finite elements, all elements of trade-off matrix are become negative in finite steps.

Lemma 4. The proposed algorithm can improve EW solution.

Proof. Let the solution by the proposed algorithm be NEW_{cost}, the EW solution be EW_{cost}. Also, assume that the number of trees by EW solution is $lcnt$, the set of nodes corresponding to trees j (j=1,2..,$lcnt$) is R_j and the corresponding cost is $C(R_j)$. Then EW_{cost} is $\sum_{j=1}{}^{lcnt} C(R_j)$. In this case, $\cap_{j=1}{}^{lcnt} R_j = null$ and $C(R_j)$ is the MST cost corresponding to R_j. in the step 2, NEW_{cost} is replaced by $\mathrm{Ew_{cost}}$. And only in the case that the cost ($TEMP_{cost}$) obtained in step 2 is less than EW_{cost}, $TEMP_{cost}$ is replaced by NEW_{cost}, so, $TEMP_{cost} = NEW_{cost} < EW_{cost}$. Now, one of cases which $TEMP_{cost}$ is less than EW_{cost} is considered. Let two sets of nodes changed after changing nodes in step 2 be R_{sub1}, R_{sub2} and the corresponding sets of nodes obtained by EW solution R'_{sub1}, R'_{sub2}. If cardinalities of R'_{sub1}, R'_{sub2} are $|R'_{sub1}| = |R'_{sub2}| = Q$, at the same time, $|R_{sub1}| = |R_{sub2}| = Q$ where Q is the maximum number of nodes. Assume that link costs, $d_{i1,i2} < d_{i2,i3} <,.. < d_{ik\text{-}2,ik\text{-}1} < d_{ik\text{-}1,ik} = d_{ik\text{-}1,jk\text{-}1} < d_{ik,jk\text{-}2} < d_{ik,jk\text{-}1} < d_{ik,jk} < d_{j1,j2} <.. < d_{jk\text{-}1,jk} <$ other link cost(d_{ij}), and in EW solution, $g_{i1,i2} < g_{\mathrm{i2,i3}} <,..< g_{ik\text{-}1,ik} = g_{ik\text{-}1,jk\text{-}1} < g_{ik,jk\text{-}2} < g_{ik,jk\text{-}1} < g_{ik,jk} < g_{j1,j2} <..< g_{jk\text{-}1,jk} < 0$ (other g_{ij}'s > 0) are obtained. Then, $R'_{sub1} = \{i1,i2,,ik\}$, the corresponding tree is (0-$i1$-$i2$-..-ik) and $R'_{sub2} = \{j1,j2,,jk\}$, the corresponding tree is

(0-$j1$-$j2$-..-jk). Therefore, $C(R'_{sub1}) = d_{0,i1} + d_{i1,i2} +,.. + d_{ik-2,ik-1} + d_{ik-1,ik}$ and $C(R'_{sub2}) = d_{0,j1} + d_{j1,j2} + ,..+ d_{jk-2,jk-1} + d_{jk-1,jk}$. But, edges ($ik$-1, jk-1), (ik, jk-2), (ik, jk) with the less g value are excluded in the solution because $|R_{sub1}|$ is equal to Q. By assumption, $d_{ik,jk-2} + d_{ik,jk} < d_{jk-2,jk-1} + d_{jk-1,jk}$ and $d_{ik-1,ik} = d_{ik-1,jk-1}$. If exchanging node($ik$) for node($jk$-1), $R_{sub1} = \{i1,i2,..,ik-1,jk-1\}$ and $R_{sub2} = \{j1,j2,..,jk-2,ik,jk\}$. Applying MST algorithm to the above two sets, since $C(R_{sub1})$, $C(R_{sub2})$ are the minimum cost trees, $C(R_{sub1})$ becomes $d_{0,i1} + d_{i1,i2} +,.. + d_{ik-2,ik-1} + d_{ik-1,jk-1}$ and $C(R_{sub2})$ becomes $d_{0,j1} + d_{j1,j2} +,..,+ d_{ik,jk-2} + d_{ik,jk}$. Thus, since $C(R_{sub1}) = C(R'_{sub1})$ and $C(R_{sub2}) < C(R'_{sub2})$, $C(R_{sub1}) + C(R_{sub2}) < C(R'_{sub1}) + C(R'_{sub2})$. Total cost, $TEMP_{cost} = \sum_{j=1,j\neq sub1,sub2}^{lcnt} C(R_j) + C(R_{sub1}) + C(R_{sub2}) < \sum_{j=1,j\neq sub1,sub2} C(R_j) + C(R'_{sub1}) + C(R'_{sub2}) = EW_{cost}$. Hence, there exists the case which $TEMP_{cost}$ is less than EW_{cost}.

Table 1 represents the comparison of several algorithms. In the time complexity of algorithm[7], the practical range of S is from n/Q to $nlog(n/Q)$. Algorithm[6] represents the results when every traffic requirement is one. If the traffic requirements are different or Q is not the power of 2, the results are inferior to that of EW solution. For the complete graph, since the time complexity of algorithm[6] is $O(Qn^3)$, the computing time is increased more sharply than the proposed algorithm with the time complexity of $O(Qn^2 log\ n)$ as Q is increased.

Table 1. Comparison of memory and time complexity

algorithm	memory complexity	time complexity
EW solution	$O(n^2)$	$O(n^2\ log\ n)$
Algorithm[7]	$O(n^2)$	$O(S^3 nlog\ n)$
Algorithm[6]	$O(n^2)$	$O(n^3)$
Proposed algorithm	$O(n^2)$	$O(n^2 log\ n)$

3.2 Simulation

In order to evaluate the proposed algorithm, we carried out the computational experiments on IBM-PC. The coordinates of nodes were randomly generated in a square grid of dimensions 100 by 100. The savings rate is defined as Eq. (2).

$$\text{Savings Rate} = (A - B) / A \times 100 \tag{2}$$

In Eq. (2), A is the cost by EW solution and B is the cost by the proposed algorithm suggested in this paper.

We assumed that the traffic requirements have Poisson distribution. Let exponential random number be X, the average traffic rate of network be λ, the following expression is obtained.

$$X = \frac{1}{\lambda} ln\ (\frac{1}{1-U}) \tag{3}$$

In Eq. (3), U is uniformly distributed random variable between 0 and 1. Using Eq. (3), n Poisson random number are generated and used as the traffic requirements. 20, 30, 40, and 50 as the value of λ and 30, 50, 70 as the number of nodes (n) are used respectively. Maximum traffic handled in a single tree is used between the maximum value among the Poisson random number by the simulation and *n*/4. Fig. 2 shows the mean savings rate. Increasing of λ and *n* do not affect the results of savings rate. Thus, to derive the expression describing the relation between λ or *n* and solution exactly is difficult.

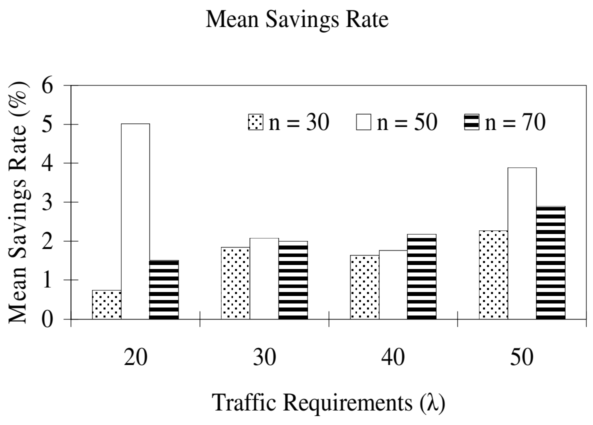

Fig. 2. Mean Savings Rate

As shown in Fig. 2, our proposed algorithm improves the EW solution up to 5 %. Mean savings rate of algorithm[7] is 1.9 % and that of algorithm[6] ranges from 1 % to 5 %. Thus, it is known that no algorithm produces the best result. The reason is because the solution is affected by the generated location of nodes. However, since the time complexity of the proposed algorithm is the least as shown in Table 1, we can state that the proposed algorithm produces reasonable improvements over the EW solution in the short running time in comparison with other heuristics.

4 Conclusions

In this paper, we present new heuristic algorithm and its computational property for the capacitated minimum spanning tree (CMST) problem. The proposed algorithm can be applied to find the least cost multicast trees and topology discovery in the local computer network. It improves solutions by exchanging or transferring nodes between trees based on the suggested heuristic rules. It has the small memory and time complexities and produces good improvements over the benchmark solution in comparison with other existing heuristics. Simulation results show that the proposed algorithm does not limit the type of traffic requirements, has not the fluctuation of solution, and is more efficient when the traffic volume of network is light. Future work includes the more efficient algorithm considering a mean delay constraint.

References

1. Bjerano, Y., Breitbart, M. and Rastogi, R.: Physical topology discovery for large multi-subnet networks. INFOCOM. (2003) 342-352.
2. Huffaker, B., Plummer, D., and Claffy, K.: Topology discovery by active probing. Applications and the Internet (SAINT) Workshops. (2002) 90-96.
3. Lee, Y.: Minimal cost heuristic algorithm for delay constrained loop network. International Journal of Computer Systems Science & Engineering, Vol. 19. CRL Publishing. (2004) 209-219.
4. Papadimitriou, C.H.: The complexity of the capacitated tree problem. Networks, Vol. 8. (1978) 217-230.
5. Esau, L.R. and Williams, K.: On teleprocessing system design, part II. IBM syst. J., Vol. 5. (1966) 142-147.
6. Gavish, B. and Altinkemer, K.: Parallel savings heuristics for the topological design of local access tree networks. Proceedings IEEE-INFOCOM '86. (1986) 139-139.
7. Kershenbaum, K., Boorstyn, R. and Oppenheim, R.: Second-order greedy algorithms for centralized teleprocessing network design. IEEE Trans. on Comm., Vol. 28. (1980) 1835-1838.
8. Sedgewick, R.: Algorithms. Addison-Wesley. (1989) 452-461.