Optimal Multicast Loop Algorithm for Multimedia Traffic Distribution

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

[1] Department of Technology Education,
Korea National University of Education
San 7 Darakri, Chungbuk, 363-791, Korea
lyj@knue.ac.kr
[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract. We have presented an optimal algorithm for minimal cost loop problem (MCLP), which consists of finding a set of minimum cost loops rooted at a source node. In the MCLP, the objective function is to minimize the total link cost. The proposed algorithm is composed of two phases: in the first phase, it generates feasible paths to satisfy the traffic capacity constraint, and finds the exact solution through matching in the second phase. In addition, we have derived several properties of the proposed algorithm. Performance evaluation shows that the proposed algorithm has good efficiency for small network with light traffic. Our proposed algorithm can be applied to find multicast loops for real-time multimedia traffic distribution.

1 Introduction

Current computer networks consist of backbone networks that serve as the major highways to transfer large volumes of communication traffic, and local networks that feed traffic between the backbone node and end-user nodes connected to the backbone. Several researchers have proposed algorithms for discovering the topology of the Internet backbone [1-2]. A local area network (LAN) connected to a backbone node, can be regarded as an end-user node. A local network, therefore, consists of a backbone node and several end-user nodes hanging off a LAN.

In a local network, the total traffic volume of end-user nodes that can be served by a port of the backbone node is limited. So, the local network consists of a backbone node (source node) and several trees or loops that cover all end-user nodes to satisfy the constraints on traffic volume. Topology discovery is required to find all the trees or loops in a local network that satisfies the constraints.

Issues related to topology discovery for local network has been classified into two problems in the literature: capacitated minimum spanning tree problem (CMSTP) [3,4] and minimum cost loop problem (MCLP) [5,6]. The CMSTP finds the best way

in the least cost aspect to link end-user nodes to a source node. It determines a set of minimal spanning trees with a capacity constraint. In the MCLP, end-user nodes are linked together by a loop that is connected to a port in the backbone node. The links connecting end-user nodes have a finite capacity and can handle a restricted amount of traffic, thus limit the number of end-user nodes that can be served by a single loop. The objective of the design is to form a collection of loops that serve all user nodes with a minimal connection cost.

The objective of this paper is to formulate the MCLP and to develop an exact algorithm to solve the problem. We propose a dynamic programming-based exact algorithm. Our proposed algorithm solves the MCLP in two phases: In the first phase, the algorithm uses dynamic programming to generate feasible solutions to satisfy the traffic capacity constraint. In the second phase, it finds exact solution by applying the matching procedure to the set of the capacitated loops found in the first phase.

Our performance evaluation results demonstrate that the time complexity of our proposed exact algorithm is large when the number of nodes in the local network is large. Since the MCLP is NP-hard [7], it is difficult to find the exact solution for large network in short computing time. We, therefore, suggest using our exact algorithm for small local networks (less than thirty nodes), and heuristic methods can be used for large local networks.

The main contributions of this paper are: (i) formulation of the MCLP, (ii) proposing and evaluating its exact solution, and (iii) determining the threshold in choosing between heuristic methods and the exact algorithm depending on the size of the local network.

The suggested algorithm can be applied to the design of synchronous optical network (SONET) and the finding of multicast loops rooted at a source node in order to transfer the message to end-user nodes in the local network.

The rest of the paper is organized as follows. Section 2 presents the mathematical formulation of the MCLP. Section 3 describes our proposed dynamic programming based algorithm for the MCLP. Section 4 evaluates the computational complexity and execution time of our proposed algorithm. Finally, concluding remarks are given in section 5.

2 Formulation of the MCLP

The MCLP (see Fig. 1) is concerned with finding a set of minimal cost loops that are used form the multicast paths for real time traffic, such as multimedia. The end-user nodes are linked together by a loop connected to a port in the backbone node, where the links connecting the end-user nodes have finite capacity, i.e. they can handle limited amount of traffic (Q). This translates to restricting the number of end-user nodes served by a single loop. The solution of the MCLP results in a collection of loops that serve all end-user nodes with a minimal connection cost.

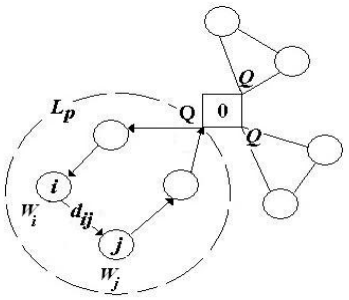

Fig. 1 MCLP

We now consider the modeling of the MCLP. Assume that there is graph, G=(V, E), where V={0,1,2,..,n}, traffic requirement at node is W_i ($i \in V$-{0}) and link cost between node i and node j is d_{ij} ($j \in V$-{0} and (i,j) $\in E$). Q represents maximum traffic served by single loop. Index 0 represents the source node and can be viewed as an object such as the backbone router with several ports. End-user nodes originate traffics and can be regarded as hosts or switching hubs. The problem formulation for the MCLP is described by Eq. (1). The objective of the MCLP is to find a collection of least-cost loops rooted at node 0. L_p is the p^{th} loop (p=1,2,..,*lcnt*: where *lcnt* is the number of loops in the solution) whose two of nodes are connected the source node. A particular case occurs when each W_i is equal to one. In that case, the constraint means that no more than Q nodes can belong to any loop of the solution. With this constraint, we can confine the fault to the loop only in which it occurred. x_{ij} represents link between node i and node j (i,j: i=1,2..,n; j=1,2…,n). If link (i,j) is included in any loop (L_p) of the solution, then x_{ij} is set to 1. m represents the number of links in the solution.

$$
\begin{aligned}
&\textit{Minimize} \quad \sum_{i,j} d_{ij} x_{ij} \\
&S.T. \\
&\qquad \sum_{i \in L_p} W_i x_{ij} \le Q \\
&\qquad \sum_{i,j} x_{ij} = m \\
&\qquad x_{ij} = 0 \quad or \quad 1
\end{aligned}
\tag{1}
$$

3 Solution of the MCLP

Having formulated the MCLP in the previous section, we now develop solution for the problem. Our solution consists of two phases: feasible path generation phase and matching phase as described below.

3.1 Feasible Path Generation Phase

To generate the feasible paths for the MCLP using dynamic programming, we define stage variable and state variables: Stage variable, k (k=1,2…), is the number of nodes to form a path rooted at the source node to any arbitrary node j. State variables, j and S are the node index to be connected and the set of node indexes included in the path in order to connect node j, respectively. Then, using the principle of optimality, the recurrence relation can be obtained as shown in Eq. (2).

$$\begin{aligned} &If \sum_{q\in S} W_q + W_j \le Q, \\ &\quad f_k(j,S) = \underset{q\in S, q\neq j}{Min} [f_{k-1}(q, S-\{q\}) + d_{qj}] \\ &\qquad (k=1,2\cdots; S\subseteq N_j) \\ &else \\ &\quad f_k(j,S) = \infty \end{aligned} \tag{2}$$

In Eq. (2), $f_k(j,S)$ represents the least cost to connect node j with the paths of which the number of nodes included in S is k to connect node j. $f_k(j,S)$ is set to infinity when the sum of traffics at end-user nodes exceeds Q.

Since boundary condition represents the cost to connect from the source node to node j directly without intermediate nodes, it is defined by Eq. (3).

$$f_0(j,-) = d_{0j} \tag{3}$$

To obtain a feasible solution, we compute $f_1(j,S)$ for all (j,S) satisfying $\sum_{q\in S} W_q + W_j \le Q$ by using f_0. Then, $f_2(j,S)$ is computed using f_1. By repeating this procedure, we reach the phase where $\sum_{q\in S} W_q + W_j > Q$, for all (j,S).

Since $f_k(j,S)$ are infinity for all (j,S) at such a phase, we set L=k. This means that the loop cannot be extended any further. So, paths obtained at the previous stage k (0,1,2,..,L-1) are feasible solutions.

3.2 Matching Phase

At stage k of the feasible path generation phase, $f_k(j,S)$ represents the cost of the path composed of the same elements as $j \cup S$, but the order of elements included in the set is different.

Among $f_k(j,S)$, the minimum cost, $f'_k(P_m)$ is computed. That is, the cost for P_m at stage k, $f'_k(P_m)$ is as follows:

$$f'_0(P_m) = f_0(j,-) + d_{j0}, \qquad k = 0 \tag{4}$$
$$f'_k(P_m) = \text{Min}[f_k(j,S) + d_{j0}], \qquad k = 1,2,..,L\text{-}1$$
$$\forall\ (j,S) \text{ such that } P_m - \{j \cap S\} = \phi$$

The value of $f'_k(P_m)$ from Eq. (4) represents the cost of the loop including the connection cost to the source node, which is composed of the same node indexes of different order. Node set of the optimal policy, R_m corresponds to $f'_k(P_m)$. R_m is the set of node indexes included in the loop rooted from the source node and represents the node sequence of loop by adding the source node to both end-side indexes. Of course, node 0 is not included in R_m.

Finally, since n nodes have to be included in any loop (R_m) rooted at the source node without duplicate inclusion in the optimal solution, the optimal solution can be obtained by substituting $f(R_m)$ for $f'_k(P_m)$ from Eq. (4). There can be several collections which have the element, R_m, to satisfy the split condition of set N'. Thus, the global optimal value, F, is the least value among the cost, $f(R_m)$, corresponding to R_m, and can be expressed by Eq. (5).

$$F = \text{Min}\ [\sum f(R_m)\ ], \tag{5}$$
$$\forall\ R_m \text{ such that } \cup R_m = N' \text{ and } R_i \cap R_j = \phi\ \ (i \neq j)$$

3.3 Optimal MCLP Algorithm

From the above model, the optimal MCLP algorithm is described as the following.

OPTIMAL MCLP ALGORITHM

Feasible path generation Phase

1. **for** $k = 0$ and **for all** j such that $j \notin N'$ **do**
2. $\quad f_0(j,-) \leftarrow d_{0j}$
3. **end for**
4. **while** $f_k(j,S) \neq \infty,\ \forall\ (j,S)$ **do**
5. $\quad$ **if** $\sum_{q \in S} W_q + W_j \leq Q$ **then**
6. $\quad\quad$ **for all** k such that $k = 1, 2,..,$ S and $S \subseteq N_j$ **do**
7. $\quad\quad\quad f_k(j,S) \leftarrow \underset{q \in S, q \neq j}{Min}\ [f_{k-1}(q, S-\{q\}) + d_{qj}]$
8. $\quad\quad$ **end for**
9. $\quad$ **else** $f_k(j, S) \leftarrow \infty$
10. $\quad$ **end if**
11. **end while**
12. L $\leftarrow k$

Matching Phase

13. **for** $k = 0$ and **for all** j such that $j \notin N'$ **do**
14. $\quad f'_0(P_j) \leftarrow f_0(j,-) + d_{j0}$
15. **end for**

16. **for all** k such that k=1,2,..,L-1 **do**
17. $m \leftarrow 1$
18. **for all** j such that $j < \underset{q \in S}{\text{Min}}\{ q \}$ **do**
19. $P_m \leftarrow \{j\} \notin S$
20. $m \leftarrow m$+1
21. **end for**
22. **end for**
23. **for all** k such that k=1,2,..,L-1 **do**
24. $m \leftarrow 1$
25. **for all** (j, S) such that P_m - $\{j \notin S\} = \varnothing$ **do**
26. $f'_k(P_m) \leftarrow \text{Min}\ [f_k(j,S) + d_{j0}]$
27. $m \leftarrow m$+1
28. **end for**
29. **end for**
30. **for all** m **do**
31. **for all** R_m such that $\cup R_m = N'$ and $R_i \cap R_j = \varnothing$ ($i \neq j$) **do**
32. find $f(R_m)$
33. **end for**
34. **end for**
35. **for all** m **do**
36. $F \leftarrow \text{Min}\ [f(R_m)]$
37. **end for**
38. find the set of optimal loops(R) corresponding to F.

4 Performance Evaluation

Having formulated the MCLP and its solution in Sections 2 and 3, in this section, we evaluate the performance of the algorithm in terms of its computational complexity.

4.1 Properties of MCLP algorithm

We present the following Lemmas in order to show the properties of the proposed algorithm (referred hereafter simply MCLP algorithm).

Lemma 1. Feasible path generation phase ends in the finite stages.
Proof) The finish time of feasible path generation phase depends on the relationship between $\Sigma^n_{i=1} W_i$ and Q. In the worst case ($\Sigma^n_{i=1} W_i \approx Q$), we might generate maximum feasible paths. In such a case, k is close to n. The maximum of k (= L) can be nearly same as n-1, but more than or equal to n. If L is equal to n, we will find single loop. This is against our assumption ($\Sigma^n_{i=1} W_i > Q$). Thus, we can finish the feasible path generation phase in at most n-1 stages.
Lemma 2. The number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively.

Proof) In each stage (k=1,2..,L), we have to add ${}_{n-1}C_k$ for n nodes, and compare ${}_{n-1}C_k$ for the previous stage and n nodes. Therefore, the number of additions $= n\sum_{k=1}^{L} k \cdot {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L} \frac{(n-2)!}{(k-1)!(n-1-k)!} = n(n-1)\sum_{k=1}^{L} {}_{n-2}C_{k-1} = n(n\text{-}1)2^L$ and the number of comparisons $= n\sum_{k=1}^{L}(k-1) \cdot {}_{n-1}C_k$ = the number of additions – $n\sum_{k=1}^{L} {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L} k \cdot {}_{n-2}C_{k-1} - n\sum_{k=1}^{L} {}_{n-1}C_k \approx n(n\text{-}1)2^L - n(2^L\text{-}1) = n(n\text{-}2)2^L$.

Lemma 3. The upper bound for the number of additions and comparisons in the matching phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively.

Proof) In the matching phase, we need not consider the sequence of nodes unlike in the feasible path generation phase. The amount of computations is decreased in most cases. However, since there might exist the case that the amount of computation is same according to the traffic capacity constraint ($\sum_{i=1}^{n} W_i \leq Q$) regardless of the consideration for node sequence, we can state that the number of additions and comparisons in the matching phase is same as those in the feasible path generation phase in the worst case. Thus, upper bound for the number of additions and comparisons in the matching phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively.

Lemma 4. MCLP algorithm produces the optimal solution.

Proof) In the feasible path generation phase, we enumerate the feasible paths by using the optimality principle of dynamic programming. So, there can be no other feasible paths except our solutions. In the matching phase, we first find partitions of which unions compose of the node set, $N' = \{1,2,..,n\}$. Next, we generate loops composed of the above partitions. These loops are found by adding the index of the source node to indexes of both end nodes included in the partition and are sub-optimal solutions. Then, we select the least cost solution among sub-optimal solutions. Therefore, it is natural for the selected solution to be the global optimal solution.

4.2 Numerical Experiments

We consider the amount of computation in the stage variable (k). It is maximal when the traffic requirement at each node is one (W_i =1, $\forall i$) and the maximum traffic per single loop is Q. First, for any stage (k), $f_k(j,S)$ must be computed for $k \times {}_nC_{n-1}$ different (j,S) pairs. Since such computation requires k additions and k-1 comparisons, where k =1 to L, the number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively by Lemma 2. In addition, the upper bound for number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively by Lemma 3.

Fig. 1 shows the mean real execution time of MCLP algorithm for three different cases (heavy traffc- Q=1/2$\sum_{i=1}^{n} W_i$, medium traffc- Q=1/3$\sum_{i=1}^{n} W_i$, and light traffic- Q=1/4$\sum_{i=1}^{n} W_i$). For each case, ten problems were randomly generated and executed on workstation. MCLP algorithm shows the best efficiency when the sum of the traffics is much less than Q. That is, when the number of nodes is 10, mean execution times for light, medium, and heavy traffic cases are 0.3, 0.3, and 0.2 seconds respectively (since results for 10 nodes are very small, they are not shown in Fig. 1). On the other

hand, when the number of nodes is 30, mean execution times for light, medium, and heavy traffic cases are 60, 134 and 200 seconds respectively. As the number of nodes becomes large, memory access time increases sharply. The reason is why $k \times {}_nC_{n-1}$ storage spaces are required in each stage k to store $f_k(j,S)$. However, the current main memory cannot maintain all the results from the previous computation. If we can use the huge main memory, we will reduce the computation time. In addition, the reason for the less execution time in the light traffics is that since L value in MCLP algorithm becomes small in light traffic case, the amount of computations for MCLP algorithm also becomes small. To summarize, the proposed MCLP algorithm is affected by the traffic volume and Q, and is effective in the case when the number of nodes is less than thirty and the traffic volume is light. Since the execution time increases sharply as the number of nodes is more than thirty, it is desirable to use the heuristic method for the network with large nodes or heavy traffic.

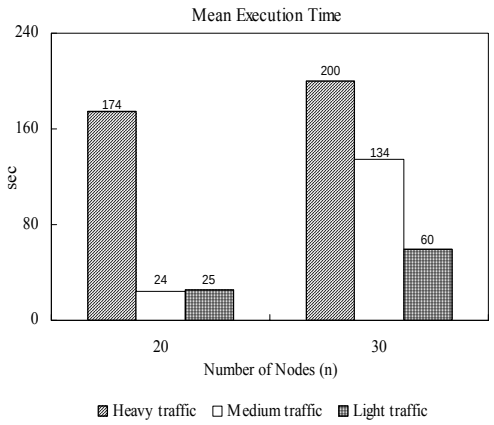

Fig. 1 Mean execution time

5 Conclusions

In this paper, we have presented the problem formulation and an optimal algorithm for the MCLP. The proposed algorithm minimizes the total cost to discover the optimal loops rooted at the source node. It consists of generating the feasible paths using dynamic programming and finding the exact solution by matching procedure. Several properties about the performance of our algorithm were derived, and through experiments, it was shown that the proposed algorithm is effective for small local network of which total traffic volume is relatively smaller than the maximum traffic to be served by a port of the source node. Our proposed algorithm can be used to discover the multicast loops for real-time multimedia traffic. Future work consists of developing a heuristic algorithm applicable to local networks with large number of nodes.

References

1. Breibart, Y., Garofalakis, M., Martin, C., Seshadri, S., and Silberschatz, A.: Topology Discovery in Heterogeneous IP Networks. INFOCOM. (2000) 265-274.
2. Lin, H., Lai, H., and Lai, S.: Automatic Link Layer Topology Discovery of IP Networks. IEEE ICC'99. (1999) 1034-1038.
3. Lee, Y. and Atiquzzaman, M.: Least Cost Multicast Spanning Tree Algorithm for Local Computer Network. IEEE ICCNMC'05. (2005) 268-275.
4. Lee, Y. and Atiquzzaman, M.: Least Cost Heuristic for the Delay-Constrained Capacitated Minimum Spanning Tree Problem. Computer Communications. Vol. 28. (2005) 1371-1379.
5. Gavish, B.: Topological Design of Telecommunication Networks-Local Access Design Methods. Annals of Operations Research, Vol. 33. (1991) 17-71.
6. Lee, Y.: Minimal Cost Heuristic Algorithm for Delay Constrained Loop Network", International Journal of Computer Systems Science &Engineering. Vol. 19, No. 4, CRL Publishing. (2004) 209-219.
7. Lenster, J. and Kan, R.: Complexity of Vehicle Routing and Scheduling Problems. Networks, Vol. 11. (1981) 221-227.