A Heuristic Algorithm for the Multi-constrained Multicast Tree

Wen-Lin Yang

Department of Information Technology
National Pingtung Institute of Commerce
No.51, Ming-Sheng East Road, Pingtung City,Taiwan
wly@npic.edu.tw

Abstract. Multicasting is an important communication mechanism for implementing real-time multimedia applications, which usually require the underlying network to provide a number of quality-of-service (QoS) guarantees to users. In this paper, we study the problem concerning how to find a feasible multicast tree in which all the multicast paths satisfy the given set of QoS constraints. This problem is referred as the multi-constrained multicast tree (MCMT) problem. Based on tabu-search technique, a heuristic algorithm named MCMTS for the MCMT problem is proposed in this study. According to the experimental results, the probability of finding a feasible multicast tree by our MCMTS method is more than 99% if one exists. Furthermore, the MCMTS is also shown to be a simple and highly efficient method. As a result, it would be a practical approach for developing multicast routing protocol.

1 Introduction

For multimedia applications involved in real-time communications, they usually require the underlying network to provide a number of quality-of-service (QoS) guarantees to users. For a network with a set of QoS constraints, a routing problem studied in this paper is to find a multicast tree such that any path between a pair of source and destination nodes satisfies the given set of QoS constraints simultaneously. This routing problem is also referred as the multi-constrained multicast tree (MCMT) problem.

In the past, the MCMT problem did not receive much attention. Most of previous research on finding a multicast tree considers only one or two QoS constraints, like delay and jitter [8,9,10,11]. As far we know, only two methods have been published [6,13] recently for the MCMT problem. In paper [6], an algorithm named MAMCRA is proposed for the MCMT problem. Their algorithm begins with finding a set of shortest paths from the source node to all destinations. By removing the overlap of paths, a multicast tree is then obtained. However, the problem finding a path between any two nodes is a NP-complete problem [3], and it is referred as the multi-constrained path (MCP) problem in literature. A number of heuristic algorithms have been proposed for it [1,4,5,15,16].

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 78–89, 2003.

In paper [13], a heuristic based on genetic algorithm called MCMGA is proposed. The goal of this method is not just to find a feasible multicast tree, but also to minimize the cost of the tree. The first step of MCMGA is to generate k shortest paths for each pair of source and destination. Then, a large of number of multicast trees must be maintained for generic operations. Since k could be a large number and many multicast trees must be kept for every generation, the MCMGA is not just too complex but also very inefficient for large-scale networks with a large set of destinations.

For networks with tight QoS constraints, the number of feasible multicast trees could be very few, especially when the number of QoS constraints increases. Hence, a routing algorithm for finding a least-cost multicast tree would be too expensive to be practical for routing protocol implementation. In this paper we are only concerned about how to find a feasible multicast tree efficiently. The resulting multicast tree is not necessary to be a least-cost tree. It has been noticed that all multicast routing protocols are developed based on simple multicast routing algorithm [7], where simplicity and ease of implementation should be the most important criterion when evaluating multicast routing algorithms. Hence, the goal of this study is to develop a simple and efficient algorithm for the MCMT problem.

Our heuristic algorithm proposed in this paper is named the multi-constrained multicast tree algorithm based on Tabu-search technique [2] (MCMTS). The MCMTS method begins with finding a good spanning tree in which a multicast tree is embedded. Since the resulting multicast tree may be infeasible, some multicast paths may violate the QoS constraints. A path replacement procedure guided by tabu-search strategy is then used to improve those infeasible multicast paths. Our MCMTS algorithm is represented in section 3. The results of numerical simulations are given in section 4.

2 The Multi-constrained Multicast Tree Problem

Consider a network that is modeled by a directed graph $G(V, E)$, where V is a set of nodes and E is a set of links. Each link $(u, v) \in E$ is associated with k positive additive QoS parameters, such as delay, delay jitter, data loss rate, etc. The value of QoS parameter i is represented by $w_i(u, v)$, where $i = 0, 1, ..., (k-1)$. Since the network could be asymmetric, $w_i(u, v)$ may be not equal to $w_i(v, u)$ for some QoS parameter i. For a path P and a QoS parameter i, we use the path constraint $W_i(P)$ to represent the summation of $w_i(u, v)$ on each link (u, v) on the path P. That is, $W_i(P) = \sum_{(u,v)\in P} w_i(u, v)$. In this paper, we also use the notation $(S \to j)$ to represent the path from S node to node j.

Let D be a subset of V ($D \subset V$ and $|D| = m$), and denote a group of destination nodes of a multicast tree. Given a set of nodes D which does not contain the source node S, and k QoS constraints C_i, $0 \leq i \leq (k-1)$, the goal of our MCMT problem is to find a multicast tree T such that the following conditions are hold:

Assume P_j is the path from source S to any node j in the destination group D, $W_i(P_j) \leq C_i$, for all i and j, where $0 \leq i \leq (k-1)$, and $0 \leq j \leq (m-1)$.

3 The MCMTS Algorithm for the MCMT Problem

A simple and efficient heuristic algorithm named MCMTS based on tabu search strategy is developed in this section. The basic idea of our MCMTS method can be outlined as follows:

(a) First of all, a 'good' spanning tree T rooted at the source node must be determined. The spanning tree covers all the nodes in the given network, and a multicast tree M that covers all the nodes in the set D is embedded in T. A method for determining 'good' spanning tree is presented in section 3.1.
(b) Based on the multicast tree M obtained in step (a), for each node u in the set D, all the QoS path constraints are checked. If M is a feasible solution, the procedure stops. Otherwise, go to step (c).
(c) For any destination node u, whose path from source violates QoS constraints, an efficient tabu-search based procedure is applied to modify the path from source to u. The procedure stops after a given number of iterations. This method is presented in section 3.2.
(d) If the multicast tree M obtained from step (c) is still infeasible, we collect all the destination nodes whose paths do not satisfy the given QoS constraints. Let this collection be Γ.
(e) For any node u in Γ, a fix procedure is used to find a feasible path from source to u. The fix procedure is an optimal or heuristic algorithm for solving the multi-constrained path (MCP) problem. Since the simulation results presented in section 4 show that Γ is a small set, the running time spent on these MCP problems is small.

In general, our MCMTS algorithm contains two parts: a modified Prim's algorithm for determining a good spanning tree, and a tabu-search based procedure for improving the multicast tree embedded in a spanning tree.

3.1 The Modified Prim's Method

The well-known Prim's algorithm designed for finding the minimum spanning tree is modified in this section to determine a 'good' spanning tree T for the given network. A multicast tree M covers all the destinations is embedded in T. The modified Prim's method is presented in Fig. 1. During the construction of spanning tree T, an attribute vector $Y(u)$ is computed and saved for any new node u added into T. $Y(u)$ is defined as follows:

$Y(u) = [\alpha_0, ..., \alpha_{k-1}]$, $\alpha_i = W_i(P)/C_i$, $W_i(P) = \sum_{(x,y)\in P} w_i(x,y)$,

$0 \leq i \leq (k-1)$, where P is the path from source node s to node u and $P \in T$.

Hence, for a link $(u,v) \in T$, assume $Y(u) = [\alpha_0, ..., \alpha_{k-1}]$ and $Y(v) =$

$[\beta_0, ..., \beta_{k-1}]$, then $\beta_i = \alpha_i + w_i(u,v)/C_i$, $0 \le i \le (k-1)$. If a path $(s \to u)$ is feasible, then every element of the attribute vector $Y(u)$ must be equal to or less than one. For any node u with attribute vector $Y(u)$, a cost value $\Psi(u)$ is defined as follows: $\Psi(u) = \sum_{i=0}^{k-1} \alpha_i$. In each iteration of the 'While' loop in Fig. 1, only one node is selected to add into a partially built spanning tree T. The selection strategy is that the node with the smallest cost value Ψ has the highest priority to be selected.

An example of 4-node network is given in Fig. 2 to illustrate the modified Prim's method. In Fig. 2, c_0 and c_1 are two given QoS constraints. When $T = \phi$ and $F = s$, there are two nodes, 'a' and 'b', can be added into set F. For node 'a', the cost value $\Psi(a)$ is equal to 0.5 since the attribute vector $Y(a) = [1/4, 1/4]$. For node 'b', the cost value $\Psi(b)$ is equal to 1.5 because of $Y(b) = [1, 1/2]$. Hence,link (s,a) has higher priority than link (s,b) to be selected to add into T, which is shown in Fig. 2(b). The desired 'good' spanning tree T is given in Fig. 2(c).

```
1. Given a network G(V,E) and a source s;
2. Let F = {s}, T = φ, and Y(s) = [0,..,0];
3. Let Q = {v|(u,v) ∈ E, u ∈ F and v ∉ F};
4. While(F ≠ V){
5.   For each node v ∈ Q {
6.     For each node u ∈ F, if the link (u,v) ∈ E{
7.       Compute attribute vector Y(v) based on Y(u) and w_i(u,v),0 ≤ i ≤
   (k − 1);
8.       Let Y(v) = [β_0,..,β_{k−1}], and a cost value Ψ(v) = Σ_{i=0}^{k−1} β_i;
9.       Keep the link (u,v) with the smallest value of Ψ;
10.    }}
11.    T = T ∪ (u,v), F = F ∪ {v};
12. }; Output the spanning tree T;
```

Fig. 1. The modified Prim's algorithm

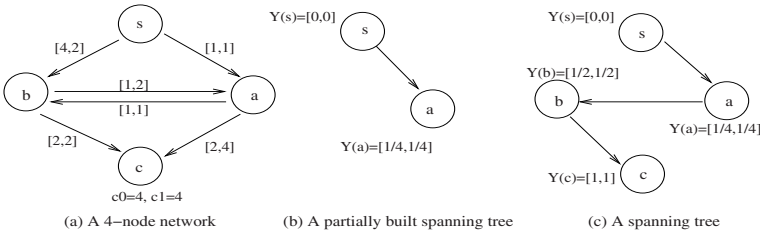

Fig. 2. An example for generating spanning tree.

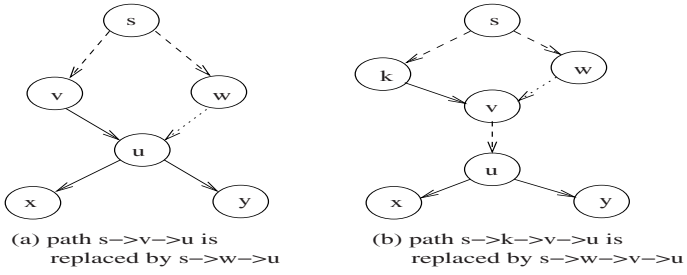

Fig. 3. The path rebuilding process.

3.2 The Tabu-Search Based Procedure

In the multicast tree M embedded in the spanning tree T found by the modified Prim's algorithm, some paths from source to destinations may not satisfy the given QoS path constraints. These infeasible paths are then improved by our tabu-search based procedure presented in Fig. 4.

Consider an infeasible path $p = (s \rightarrow v \rightarrow u)$ shown in Fig. 3(a) where s is the source and u is a destination, the goal of our tabu-search based procedure is to find a new path $\bar{p}$ that can reach destination u. Since $\bar{p}$ may be infeasible, the decision about whether p is replaced by $\bar{p}$ is based on the equation (A) given in the following: at node u, assume that

(a) $Y(u) = [\alpha_0, .., \alpha_{k-1}]$, $\Psi = \sum_{i=0}^{k-1} \alpha_i$ for path p;
(b) $Y'(u) = [\beta_0, .., \beta_{k-1}]$, $\Psi' = \sum_{i=0}^{k-1} \alpha_i$ for path $\bar{p}$;
(c) $\Psi' < \Psi + \theta$, $0 \leq \theta < k$, ... (A)

When θ is zero and the equation (A) is hold, the path $\bar{p}$ is said to be better than path p, and p can be replaced by $\bar{p}$. However, this condition may be too strict for our tabu-search based procedure. To avoid the searching path trapped in a local optimal, our heuristic procedure may accept a new path $\bar{p}$ worse than path p by setting θ value greater than zero. Hence, a path $\bar{p}$ can replace a path p even though Ψ' is greater than Ψ. In our simulations presented in section 4, the value of θ is set to be 0.5.

In Fig. 3(a), the new path $\bar{p} = (s \rightarrow w \rightarrow u)$ satisfied equation (A) is found at the destination u. Note that a link (w, u) must satisfy the following conditions: $(w, u) \in E$, $(w, u) \notin T$ and w cannot be in the sub-tree rooted at node u to avoid creating a cycle. However, in Fig. 3(b), no acceptable path can be found at the destination u. The ancestor nodes of u are then tested sequentially in order to construct a desired new path $\bar{p}$. In Fig. 3(b), for example, a new path is built based on node v, which is an ancestor of the destination u. In this paper, node u in Fig. 3(a) and node v in Fig. 3(b) are all referenced as a "branch point" on the upward path from u to s. In Fig. 3(b), if a new path $\bar{p}$ based on a "branch point" v is constructed successfully, the attribute vectors of v and its downstream nodes must be recomputed. For any destination node u, if any element of the attribute vector $Y'(u)$ is greater than one, the new path $\bar{p}$ is still infeasible.

The above path replacement procedure is implemented in an iteration loop, which is developed based on the tabu-search strategy [2], and is shown at lines $5 \sim 23$ in Fig. 4. A circular queue is maintained in our procedure and served as a tabu-list for memorizing the links having been considered recently for constructing a new path (see line 12 in Fig. 4). For example, two links (w, u) and (w, v), which are in Fig. 3(a) and Fig. 3(b) respectively, should be stored in the tabu-list after a new path is built. Since only links not in the tabu-list are considered for rebuilding, our procedure can avoid revisiting some links that have been visited recently, and has better chance to explore some unvisited solution space.

```
1. Given a network G(V,E), a source node s and a set of destination
   nodes D, D ⊂ V;
2. Find a 'good' spanning tree T based on the modified Prim's algorithm
   given in Fig. 1. A multicast tree M is in T;
3. Let linked list R = {u|p = (s → u), u ∈ D, p violates at least one QoS
   constraints.};
4. Initialize a circular queue Q to serve as a tabu-list; I = 0;
5. While(R ≠ φ and I < ITERATIONS) {
6.   Let u be the first element of R; R = R − {u};
7.   Assume the infeasible path be p = (s → u); Let v = u ;
8.   While(v ≠ s){
9.    Assume w = v → parent_node;
10.   Z = {w'|w ≠ w', (w', v) ∈ E, (w', v) ∉ T};
11.   Find w' ∈ Z such that (w', v) ∉ Q and w' is not in the sub-tree
   rooted at v {
12.     Q = Q ∪ (w', v);Build a new path p̄ = (s → w' → v → u);
13.     Compute Y'(v) and Ψ'(v) based on the new path p̄;
14.     If(Ψ'(v) < Ψ(v) + θ){
15.      T = T − (w, v); T = T ∪ (w', v);
16.      Update Y for v and its downstream nodes in M;
17.      R=R − {u|u ∈ D, u is in the subtree rooted at v; the path p from
   s to u satisfies all QoS constraints};
18.      goto next;
19.     } }
20.   v = v → parent_node;
21.  }
22. next:  I = I + 1;
23. }
24. If(R ≠ φ){call a fix procedure to find feasible paths from s to each
   node in R;}
25. Output the multicast M embedded in T;
```

Fig. 4. The MCMTS algorithm

3.3 The Optimal Algorithm and Fix Procedures

Recall that the multi-constrained path (MCP) problem is concerned about how to find a feasible path between two given nodes, so that a set of QoS constraints can be satisfied simultaneously. Hence, for the MCMT problem of a network with m destinations, a multi-constrained multicast tree can be obtained by finding a multi-constrained path for each destination. As a result, the MCMT problem can be solved optimally if the corresponding MCP problem is solved optimally.

For a multicast tree determined by our heuristic procedure presented in Fig. 4, it is possible to have a small set of destinations whose multicast paths are infeasible. At line 24 in Fig. 4, a fix procedure is then called to find a set of multi-constrained paths for those destinations. The fix procedures used in this study can be the optimal or heuristic algorithm [15,16] developed for the MCP problem. In fact, based on the simulations conducted in section 4, the set of paths that must be determined by the fix procedure is very small. That is the reason why our MCMTS procedure is very efficient when it is compared to the optimal algorithm for the MCMT problem.

3.4 Time Complexity

The time complexity of the MCMTS algorithm is determined by the following terms: $O(n^2)$ for the modified Prim's procedure and $O(m * n * ITERATIONS)$ for the loop, where $O(n)$ is required at lines 16-17 in Fig. 4. The time complexity of line 24 in Fig. 4 is $O(d^h)$ if the heuristic algorithm proposed in paper [15] is used as the fix procedure, where d is the maximum number of node-degree and h is the maximum number of hops between source and any destination. Hence, the time complexity of our MCMTS is $O(n^2 + m * n * ITERATIONS) + O(d^h) * \delta$ where $\delta \ll m$. Obviously, the executing efficiency of our MCMTS method is determined by the value of δ.

4 Experimental Results

In this section, we have several sets of experiments for comparing executing performance and solution quality between the optimal algorithm and the MCMTS heuristic proposed in this paper for solving the MCMT problem. The optimal algorithm is extended from the optimal algorithm proposed for the MCP problem [14]. Two network topologies, random graph and mesh, are simulated in this study. All the simulations are done with the following experimental parameters: P4 2.0 GHz CPU, 512MB RAM, Linux OS, and programs are developed by C++. For all benchmarks, the values of QoS parameters assigned on each link in the network are randomly selected from the range $0 \sim 100$. In this study, each data is measured based on 1000 runs. For each run, a network configuration with different QoS assignments on each link is randomly generated. As for the values of QoS path constraints, they are assigned in such a way that feasible multicast trees can only exist in around 90% of 1000 network configurations constructed for the simulations. All networks considered in this study are asymmetric.

Table 1. Simulation results for random graphs with fix procedures

∗ Results for 100-node random graphs. Two QoS constraints: $C_0 = C_1 = 320$.

Fix procedure	#destinations	#tabu_tree /cpu(secs)	#optimal_tree /cpu(secs)	run_time_ratio	solution quality
BB_optimal	50	846/4.95	846/170.47	2.90%	100.0%
	40	870/5.46	870/180.47	3.03%	100.0%
	30	883/4.31	883/100.63	4.28%	100.0%
	20	911/3.6	911/71.81	5.01%	100.0%
	10	958/3.1	958/34.11	9.09%	100.0%
	Average			4.86%	100.0%
TS_heuristic	50	830/7.39	834/182.3	4.05%	99.52%
	40	843/7.1	849/141.77	5.01%	99.29%
	30	897/4.08	899/98.48	4.14%	99.78%
	20	902/4.48	904/71.76	6.24%	99.78%
	10	943/3.15	944/36.15	8.71%	99.89%
	Average			5.63%	99.65%

Table 2. The simulation results for 8x8 meshes with fix procedures

∗ Results for 64-node meshes. Two QoS constraints: $C_0 = C_1 = 560$.

Fix procedure	#destinations	#tabu_tree /cpu(secs)	#optimal_tree /cpu(secs)	run_time_ratio	solution quality
BB_optimal	32	880/110.2	880/987.5	11.16%	100.0%
	25	905/129.6	905/902	14.37%	100.0%
	19	937/62.2	937/610.5	10.19%	100.0%
	12	963/55.7	963/482.2	11.55%	100.0%
	6	972/33.1	972/191.2	17.30%	100.0%
	Average			12.91%	100.0%
TS_heuristic	32	899/11.2	911/1062.3	1.05%	98.68%
	25	904/6.8	912/829.7	0.82%	99.12%
	19	918/8.1	927/577.7	1.4%	99.03%
	12	959/5.4	966/418	1.28%	99.28%
	6	975/4.0	977/161.7	2.49%	99.80%
	Average			1.41%	99.18%

A random graph generator [7], which was modified from Waxman's graph generator [12], was used to create links interconnecting the nodes. In order to imitate real networks more closely, the average degree of nodes in each network is set to be four [7].

4.1 With Fix Procedures

Two fix procedures are used in this study. One is the branch-and-bound based optimal algorithm proposed in paper [14], another one is the heuristic algorithm proposed in paper [15]. They are referred as BB_optimal and TS_heuristic in

Table 3. The simulation results on three QoS constraints

Networks	$C_0/C_1/C_2$	Fix_procedure	#destinations	run_time_ratio	solution quality
100-node random graph	320/320/320	BB_optimal	50	5.33%	100.0%
			40	5.02%	100.0%
			30	7.99%	100.0%
			20	9.17%	100.0%
			10	15.41%	100.0%
			Average	8.58%	100.0%
		TS_heuristic	50	16.07%	98.09%
			40	26.37%	97.74%
			30	16.00%	98.91%
			20	21.29%	99.14%
			10	31.91%	99.32%
			Average	22.32%	98.64%
8x8 mesh	560/560/560	BB_optimal	32	22.91%	100.0%
			25	19.84%	100.0%
			19	27.79%	100.0%
			12	16.71%	100.0%
			6	18.98%	100.0%
			Average	19.84%	100.0%
		TS_heuristic	32	3.70%	96.26%
			25	3.96%	95.59%
			19	3.93%	97.73%
			12	4.07%	98.11%
			6	7.21%	98.98%
			Average	4.57%	97.33%

Table 4. The simulation results for random graphs without fix procedures

$*$ The number of nodes is 100. At most 3 QoS constraints: C_i.

$C_0/C_1/C_2$	#destinations	run_time_ratio	solution quality
320/320/-	50	2.0%	96.2%
	40	2.5%	96.0%
	30	2.7%	96.2%
	20	4.0%	98.0%
	10	7.9%	99.2%
	Average	3.8%	97.1%
320/320/320	50	3.7%	88.6%
	40	4.3%	92.5%
	30	5.4%	94.2%
	20	8.2%	93.3%
	10	15.4%	97.2%
	Average	7.4%	93.2%

Table 5. The simulation results for 8x8 meshes without fix procedures

∗ The number of nodes is 64. At most 3 QoS constraints: C_i.

$C_0/C_1/C_2$	#destinations	run_time_ratio	solution quality
560/560/-	32	0.74%	90.1%
	25	0.44%	93.4%
	19	0.82%	93.3%
	12	0.76%	95.3%
	6	1.11%	97.8%
	Average	0.78%	94.0%
560/560/560	32	0.92%	71.5%
	25	0.82%	76.9%
	19	1.14%	81.9%
	12	1.74%	86.4%
	6	2.24%	93.3%
	Average	1.37%	82.0%

Table 1 and 2 respectively. For performance comparisons, several terms shown in Table 1 $\sim$ 5 are defined as follows:

run_time_ratio= CPU time required by the MCMTS procedure in 1000 runs / CPU time required by the optimal procedure in 1000 runs.

solution quality= the number of feasible trees found by the MCMTS procedure in 1000 runs / the number of feasible trees found by the optimal procedure in 1000 runs.

#tree_tabu= the number of feasible trees found by the MCMTS algorithm.

#tree_optimal= the number of feasible trees found by the optimal algorithm. In Table 1, the *run_time_ratio* is around 5% on average when the BB_optimal and TS_heuristic are used as the fix procedures. While in Table 2, the *run_time_ratio* is 12.9% when the BB_optimal is used as the fix procedure, and it is only 1.4% when the TS_heuristic is used as the fix procedure. These data show that the number of destinations, whose multicast paths cannot be determined by our MCMTS procedure and are then found by fix procedures, is very small. As a result, our heuristic can perform very efficient. For example, in Table 1, it takes around 0.0074 (7.39/1000) seconds to solve a 100-node random network in which 50% of nodes are destinations.

We also notice that in Table 2 the executing speed is greatly improved when the fix procedure is implemented with the TS_heuristic instead of the BB_optimal. This is because that the average number of hops between any two nodes in an 8x8 mesh is large enough to slow down the executing speed of the BB_optimal, since its time complexity is $O(d^h)$ where h is the number of hops between a pair of source and destination and d is the maximum degree of nodes in the network. As for the performance, the solution quality of our MCMTS procedure with a fix procedure based on TS_heuristic is more than 99% on average in Table 1 and 2. That is, compared to the optimal algorithm, the probability of finding a feasible multicast tree by our heuristic procedure is very high. As

shown in Table 3, when three QoS constraints are considered, the solution quality of our MCMTS procedure is slightly decreased to around 98.6% for random graphs and 97.3% for meshes respectively.

4.2 Without Fix Procedures

In this section, we have two sets of simulations to show the performance of our MCMTS procedure when it is executed without implementing any fix procedure. For random graphs in Table 4, the average solution quality is around 97.1% when two QoS constraints are required. However, it decreases to 93.2% when three QoS constraints are required. The same phenomenon is also hold for meshes in Table 5. For the same values of QoS constraints, a network becomes tight when the number of QoS constraints increases. Obviously, for tight networks, it is not easy for our MCMTS algorithm with no fix procedure to find an acceptable path for replacement. As a result, its performance degrades. Although the probability of finding a feasible multicast tree decreases, this heuristic method is still very efficient, and is a practical approach for developing routing protocols, especially when the executing speed is the major concern for the underlying network.

5 Conclusions

In this paper, we have presented a heuristic algorithm called MCMTS for the MCMT problem. The experimental results show that the probability of finding a feasible multicast tree for a given network based on our heuristic is more than 99% if one exists. Furthermore, the MCMTS is also shown to be a simple and highly efficient method. As a result, it would be a practical approach for developing multicast routing protocol.

Acknowledgments. This work was supported by the National Science Council, Taiwan, R.O.C., NSC-91-2213-E-251-001.

References

1. Shigang Chen, Klara Nahrstedt, “On Finding Multi-constrained Paths,” The Proceedings of the ICC’98 Conference, IEEE, pp. 874–979, 1998.
2. F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.
3. J.M. Jaffe, “Algorithms for finding paths with multiple constraints,” Networks, vol. 14, pp. 95–116, 1984.
4. Turgay Korkmaz, Marwan Krunz, “A Randomized Algorithm for Finding a Path Subject to Multiple QoS Constraints,” The Proceedings of the IEEE Global Telecommunications Conference, IEEE, pp. 1694–1698, 1999.
5. Turgay Korkmaz, Marwan Krunz, Spyros Tragoudas, “An efficient algorithm for finding a path subject to two additive constraints,” Computer Communications, vol. 25, pp. 225–238, 2002.

6. Fernando Kuipers, Piet Van Mieghem, "MAMCRA: a constrained-based multicast routing algorithm," Computer Communications, vol. 25, pp. 802–811, 2002.
7. Hussein Salama, "Multicast Routing for Real-Time Communication on High-Speed Networks," Ph.D. dissertation, Dept. of Electrical and Computer Engineering, North Carolina State University, 1996.
8. H.F. Salama, D.S. Reeves, and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks," IEEE Journal on Selected Areas in Communications, 15(3), pp. 332–345, April 1997.
9. G.N. Rouskas, I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," IEEE Journal on Selected Areas in Communications, 15(3), pp. 346–356, April 1997.
10. S. Verma, R.K. Pankaj, A. Leon-Garica, "QoS based multicast routing algorithms for real time applications," Performance Evaluation, vol. 34, pp. 273–294, 1998.
11. Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE Journal on Selected Areas in Communications, 14(7), pp. 1228–1234, 1996.
12. B.M. Waxman, "Routing of multicast connections," IEEE Journal on Selected Areas in Communications, 6(9), pp. 1617–1622, 1988.
13. Jan-Jiin Wu, Ren-Hung Hwang, Hseueh-I Lu, "Multicast routing with multiple QoS constraints in ATM networks," Information Sciences, vol. 124, pp. 29–57, 2000.
14. Wen-Lin Yang, "Solving the MCOP Problem Optimally," The Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN 2002), IEEE, pp. 100–109, 2002.
15. Wen-Lin Yang, "Exact and Heuristic Algorithms for Multi-Constrained Path Selection Problem," Advances in Multimedia Information Processing-PCM2002, Lecture Notes in Computer Science, Vol. 2532, Springer-Verlag, pp. 952–959, 2002.
16. Xin Yuan, Xingming Liu, "Heuristic Algorithms for Multi-Constrained Quality of Service Routing," The Proceedings of the IEEE INFOCOM Conference, IEEE, pp. 844–853, 2001.