Muhammad Atif Tahir, Asif Jamshed, Habib-ur Rehman* and Yassine Daadaa

Tabu Search for Low-Cost Dynamic Multicast Tree Generation with Quality of Service Guarantees

Abstract: In a communication network with a source node, a multicast tree is defined as a tree rooted at the source node and all its leaves being recipients of the multicast originating at the source. The tree or bandwidth cost is normally measured by its utilization of tree links along with the quality of service (QoS) measures such as delay constraint and end-to-end delay. However, if nodes are allowed to join or leave the multicast group at any time during the lifetime of the multicast connection, then the problem is known as dynamic multicast routing problem. In this article, we combine a greedy approach with static multicast routing using Tabu Search to find a low-cost dynamic multicast tree with desirable QoS parameters. The proposed algorithm is then compared with several static multicast routing algorithms. The simulation results show that, on a large number of events, i.e., where nodes are leaving or joining, the proposed algorithm is able to find multicast trees of lower cost and more desirable QoS properties.

Keywords: Dynamic multicast routing, quality of service (QoS), evolutionary computing, heuristics, graph theory and algorithms, network optimization, real-time data traffic, WCCAIS2014.

2010 Mathematics Subject Classification: 68T20, 94C15.

DOI 10.1515/jisys-2014-0043
Received February 25, 2014; previously published online January 12, 2015.

1 Introduction

Many web servers on the Internet generate data that may be destined for multiple receivers. Depending on the number of sources, this kind of data flow may be categorized as one-to-many or many-to-many communication. There are three different models of transmission to cater for this kind of traffic: unicasting, broadcasting, and multicasting. In unicasting, each recipient is sent a separate copy of the data, whereas in broadcasting, the source floods the network with the requested data so that it reaches every node even if the actual number of recipients in the network is small. In multicasting, however, the source generates data that is addressed to all the recipients only, and this data propagates through the network, with copies being made at intermediate nodes only if and when required. Some examples of one-to-many and many-to-many communication on the Internet include teleconferencing, augmented reality environments, Internet television and radio, etc. IP multicast protocols and associated underlying hardware technologies handle the requirements of group communications without hogging the network bandwidth unnecessarily.

***Corresponding author: Habib-ur Rehman,** College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, KSA, e-mail: habibr@ccis.imamu.edu.sa
Muhammad Atif Tahir: School of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK
Asif Jamshed: College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, KSA; and Predictify.Me Inc., Raleigh, NC 27601, USA
Yassine Daadaa: College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, KSA

Routing plays a major role in the efficiency of multicast communications. It is the process of determining a viable path for data transfer from the source to a destination node in a packet-switched communication network. In a multicast communication setting, this involves finding paths to all the nodes in the destination set. As mentioned earlier, it is desirable to find routes in the network that use as few resources as possible. At the same time, certain minimum quality of service (QoS) requirements need to be met that give guarantees on maximum delay, number of hops (Steiner nodes), jitter (delay variation), and other metrics. The solution to this routing problem usually requires the computation of a tree spanning the multicast group (source and the destination nodes). This scenario where the multicast group remains intact throughout the multicast session is the classical static multicast routing problem. If, however, this multicast group can change during the multicast session, then the problem is known as the dynamic multicast routing problem. This may occur if, for example, additional destination nodes decide to join the group or if some of them terminate the connection. This problem is known as the dynamic Steiner tree problem [12] in terms of graph theory.

We start with a source node and a sequence $R = (r_1, r_2, ..., r_m)$ of requests. Each r_i represents a request to be either added to or deleted from the multicast group as a destination node. As each r_i is received, an updated multicast tree T_i is constructed by the dynamic multicast routing algorithm on S_i, where S_i is the set of nodes in the multicast group after the arrival of the request. Given R, the starting tree T_0 containing the source node, and a few other nodes, the dynamic multicast routing problem consists of generating a sequence of least-cost multicast trees $T = (T_1, T_2, ..., T_m)$ such that each T_i spans the nodes in S_i for $1 \leq i \leq m$ [8, 12, 21]. If there are some QoS parameters such as delay constraints, then it is known as the delay-constrained minimum-cost multicast-routing problem with dynamic membership. In graph theoretical terms, we can define the problem as follows. Let $G = (V, E)$ be a graph, and for each edge e, let two weights $cost(e)$ and $delay(e)$ be defined. A special node s is identified as the source and a dynamic set of destination nodes D is defined according to the sequence of requests $R = (r_1, r_2, ..., r_m)$. Then, the multiobjective minimum Steiner tree problem with dynamic membership finds a sequence of minimum-cost Steiner trees on D after each r_i that satisfy the given QoS parameters that may include source bandwidth requirement, maximum end-to-end delay, and delay variation. In this article, we present a Tabu Search (TS) algorithm to find a low-cost dynamic multicast tree with desirable QoS parameters. The basic idea is to first generate a static multicast tree using a fuzzy-based TS algorithm and then use a greedy approach to modify the calculated tree for each addition or deletion request r_i that is encountered during the multicast connection.

The rest of the article is organized as follows. Section 2 looks at some proposed strategies followed by the description of our method in Section 3. Simulation results and comparison with other reported heuristics are presented in Sections 4 and 5. Section 6 concludes the article.

2 Related Work

In this section, we briefly review some of the existing approaches to the multicast routing problem. Kim et al. [9] suggest algorithms to solve the static QoS multicast routing problem. Their main contribution is the formulation of a new "cost" parameter: weighted parameter for multicast trees (WPMT). WPMT takes into account the overall effect of costs and delays in each particular network, and this parameter can be tuned to give heavier weight to either the cost or delay. For each particular network with given costs and delays, new WPMT weights are calculated for each link. Finally, a minimum Steiner tree is constructed on the source and destination nodes using the approximation algorithm introduced in Reference [19]. The algorithms are tested on sets of randomly generated networks and are shown to perform better than the currently well-known approaches to construct static multicast trees for the problem.

The idea of recalculating the multicast tree for each new request and identifying frequently used nodes and links is presented in Reference [12]. If these identified nodes and links can be incorporated into the multicast tree, we can end up with a low-cost solution. The authors propose a non-rearrangeable virtual trunk dynamic multicast (VTDM) algorithm that produces a tree (not necessarily spanning), called the virtual

trunk (VT), of the underlying network. The algorithm associates a positive real number with each node of the network, and for every additional request new nodes are attached to the VT by calculating the least-cost path connecting the nodes to the VT, whereas deletions are handled with a greedy approach.

A niche flapping dynamic tree algorithm for dynamic routing is proposed recently by Qingmei and Lilin [15]. The proposed solution rests on the idea that the source of a node keeps the path and information about every new multicast node. Although many solutions [5, 11, 14, 15] have been proposed for the dynamic multicast routing problem, only a few approaches can handle multiple constraints such as the number of hops and variation in delay (jitter). Therefore, in this article, we compare the performance of our system with delay-constrained static multicast routing algorithms such as KPP [10], Bounded Shortest Multicast Algorithm (BSMA) [26], and constrained adaptive ordering (CAO) [22]. KPP was introduced by Kompella, Pasquale, and Polyzos in Reference [10], wherein they also proved the NP-completeness of the problem. Their algorithm solves the delay-constrained shortest path problem $k(k+1)/2$ number of times, where k is the size of the multicast group, i.e., the set of source and destination nodes. BSMA is another multicast algorithm that is considered rather robust in terms of tree cost [25, 26]. BSMA tries to reduce the cost of the tree by using the Kth shortest path algorithm to find lower-cost edges and replacing the edges in the tree iteratively until further cost reduction is not possible. CAO [22] is another multicast algorithm that connects one multicast group member at a time to the source. It conducts a breadth-first search and always returns a constrained multicast tree, if one exists. The performance of a restricted class of dynamic multicast algorithms based on the delay constraint has been compared in Reference [4].

In Reference [24], a static QoS multicast routing algorithm based on the principle of ant colonies is proposed. The simulation results show that their algorithm produces results similar to the k-minimum Steiner tree algorithm. However, this algorithm requires full knowledge of the network and is not suited for dynamic multicast routing. An efficient genetic algorithm based on hill climbing and an artificial immune system is proposed in Reference [13] to solve the static multicast routing problem with bandwidth and end-to-end delay constraints. It is different from previous genetic algorithm-based studies in the sense that the clonal selection is used to select the chromosomes for the new generation instead of the classical selection methods such as tournament selection. In addition, the hill climbing algorithm is used to provide an alternative partial route from the mutation node to the destination node during mutation. However, again, this algorithm is only suitable for static multicast routing algorithms.

In Reference [1], TS with long-term memory is proposed for a low-cost multicast routing algorithm with bandwidth-delay-constraint. Instead of short-term memory, an extended candidate list is being used to improve the quality of search. This algorithm requires prior knowledge of the full network and is thus not suited for dynamic multicast routing. A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems are proposed by Xu and Qu [23]. Both TS and variable neighborhood search are combined to intensify the search in the hybrid scatter search algorithm. Qu et al. [16] present the first investigation on applying the Particle Swarm Optimization (PSO) algorithm to both the Steiner tree problem and the delay-constrained multicast routing problem. Searching of the optimized solution is improved by using a path replacement operator. However, the TS algorithms in References [1, 23] and the PSO in Reference [16] require prior knowledge of the network, and thus are not suited for dynamic multicast routing. Several other algorithms are also proposed in References [2, 3, 20] for static and dynamic multicast routing algorithms.

3 Dynamic Multicast Routing Using Fuzzy-Based TS and Greedy Algorithm

In this section, we will discuss the proposed dynamic multicast routing method. In a dynamic multicast connection, nodes are allowed to join or leave the multicast group dynamically during the lifetime of the multicast connection. A good static multicast routing algorithm can generally produce better results than a dynamic one [21]. This is because a dynamic multicast routing algorithm usually adds or removes a node

without disrupting the connections to existing members, and thus does not take the multicast group as a whole into consideration. Static algorithm, on the other hand, reconstructs the whole multicast tree. Therefore, a static routing algorithm that produces near-optimal results can serve as a starting point for dynamic multicast routing algorithms and can reconstruct a multicast tree on demand during the lifetime of dynamic multicast. Figure 1 shows the proposed system for dynamic multicast tree generation.

In the proposed approach, a multicast tree is first generated using a fuzzy-based multiobjective TS algorithm [18, 25]. Then, the multicast tree is modified for each connection request (either addition or deletion) by the greedy approach described in Sections 3.3 and 3.2. If the network topology is significantly changed, then the whole multicast tree is reconstructed using TS. We can say that the network is significantly changed if 50% of the multicast members have either joined or left. The components in the proposed system are described below.

3.1 Static Multicast Tree Generation Using Fuzzy Multiobjective TS

As discussed, in the proposed algorithm, a static multicast tree is first generated using TS in combination with Dijkstra's algorithm. This section gives a brief overview of TS. TS was introduced by Glover [6, 7]. It starts from an initial solution. The cost of the initial solution is evaluated using some cost function such as the fuzzy multiobjective function used in this article. TS then examines a set of acceptable neighboring solutions and

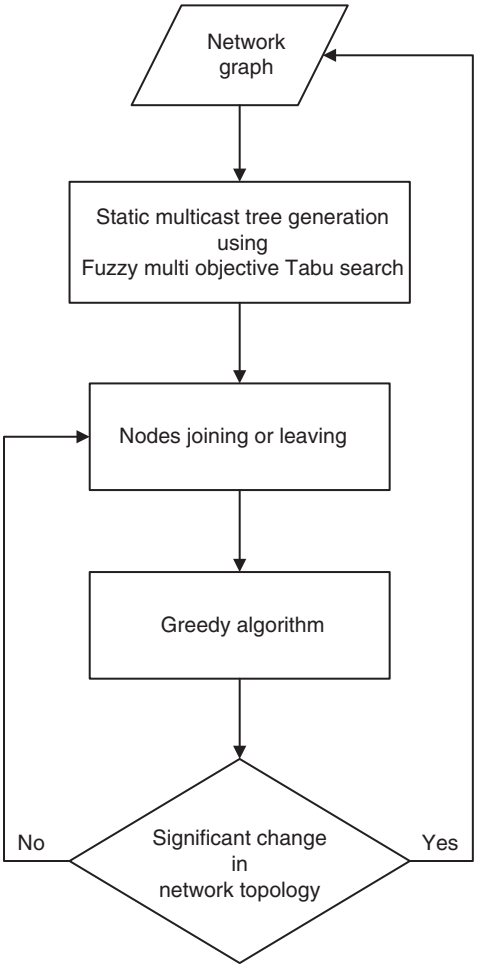

Figure 1. Proposed System for Dynamic Multicast Tree Generation.

moves to the best admissible neighbor. This is different from the greedy algorithm in the sense that TS will accept the best neighbor even if this causes the objective function to deteriorate. This will help in escaping from the local optima and provides a global search character. To avoid cycling, solutions that were recently explored are declared forbidden or "tabu" for a number of iterations. The tabu list stores the characterization of the moves that lead to those solutions. The tabu status of a solution is overridden when certain aspiration criteria are satisfied. Algorithm 1 shows the main steps for TS.

Algorithm 1: General TS Pseudocode.

```
input: Initial Solution
output: Final Solution S
While (true) do
  Create Candidate List of Solutions;
  Evaluate Solutions;
  Choose Best Admissible Solution S;
If (Stopping Conditions Are Satisfied) then
  Output S and exit
else
  Update Tabu and Aspiration Conditions;
end
end
```

As discussed earlier, multicast tree is first generated using the fuzzy-based multiobjective TS algorithm proposed in References [18, 25]. This algorithm is briefly discussed here. In this article, fuzzy rules are used on the basis of linguistic variables to evaluate the quality of a multicast tree. Four linguistic variables – number of Steiner nodes, maximum end-to-end delay, delay variance, and bandwidth cost – are defined to obtain a fuzzy logic definition of the objective function. The fuzzy subset of good multicast trees is defined by the following fuzzy logic rule:

> "**IF** a tree has *small number of Steiner nodes* AND *low maximum end-end delay* AND *low Delay Variation* AND *low cost*, **THEN** it is a good tree."

According to the and-like/or-like ordered-weighted-averaging logic, the above rule evaluates to the following:

$$\mu(x) = \beta \times \min(\mu_i(x)) + (1-\beta) \times \frac{1}{4}\sum_{i=1}^{4}\mu_i(x), \tag{1}$$

where $\mu(x)$ is the membership value for solution x in the fuzzy set *good tree* and β is a constant in the range [0,1]. Here, μ_i represents the membership values of solution x in the fuzzy sets *small number of Steiner nodes, low end-to-end delay, low delay variation,* and *low bandwidth cost*, respectively. The solution that results in the maximum value for equation (1) is reported as the best solution found by the TS algorithm.

As shown in Algorithm 1, the TS algorithm starts with an initial practical solution. In this study, this solution is initiated by the minimum-cost Steiner tree. This tree is generated using Dijkstra's shortest-path algorithm starting from the source. The cost of this tree is calculated from equation (1). For generating neighbors, it is important to first reduce the size of the solution search space. This is achieved by constructing Steiner trees from each multicast destination. The path in these trees can be used to generate neighbors from the initial solution. From the initial solution, an edge is randomly removed from the Steiner tree and replaced by different edges of Steiner trees generated from other multicast generation. This results in new Steiner trees. The number of Steiner trees depends on the size of neighbors. Each new Steiner tree is then evaluated using the fuzzy multiobjective cost function, and the one with the best cost is selected for the next iteration.

Regarding the tabu list, if an edge is deleted at iteration i, then reintroducing the same edge in an inserted operation is tabu. The tabu list size is set to 15. As is common in TS, a tabu status of a move can be overridden

if implementing it results in a better cost. This means, in this case, a tabu edge will be inserted even if it is in the tabu list. Although the choice of a maximum number of iterations should be a function of network size and the depth of the multicast tree, in this article, we have used a fixed number of iterations (maximum 500) as a stopping criterion. Future research is needed to investigate the number of iterations based on the complexity of the multicast tree.

3.2 Nodes Leaving

Let us assume that node n in the tree issues a leave request to end its participation in the multicast session. If node n is not an end-destination node in the existing tree T_i, then no action will be taken. The new tree T_{i+1} will be the same as T_i, i.e., $T_{i+1}=T_i$, with the only difference that node n will stop forwarding the multicast packets to its users. If n is an end-destination node of $T_{m'}$, then to avoid wasting bandwidth, tree T_m has to be pruned to exclude node n and related Steiner nodes and links used in the tree T_m to avoid forwarding packets to n.

3.3 Nodes Joining

There are two situations when a new node n intends to join tree T_i:
- The node n that wants to join the tree is not in tree T_i.
- The node n that wants to join the tree is a Steiner node of tree T_i.

If the node n is a Steiner node of the existing multicast tree T_m, then node n can be used without any change to forward multicast packets to its user, in addition to forwarding them to downstream nodes.

If the node n is not in the tree T_m, a shortest path is first found to all nodes (Steiner nodes, destination nodes, and source node) in the multicast tree from the new member node n. Now, we will treat every shortest path as a new added edge in the multicast tree. The result is in the number of multicast trees. Every multicast tree is evaluated by using a fuzzy membership function. The tree with the best membership function is considered as a new multicast tree T_{i+1}. Thus, T_i to new tree T_{i+1} involves only the establishment of a new path and does not affect any of the paths from source to destination nodes already in the multicast tree T_i.

4 Experiment Setup

To study the performance of the proposed dynamic multicast routing algorithm, full duplex Asynchronous Transfer Mode (ATM) networks with homogeneous link capacities of 155 Mbps are used in the experiments. ATM is a high-speed networking standard designed to support both voice and data communications. ATM networks permit the applications to specify their own QoS requirements. A random generator [17] (based on Waxman's generator [21] with some modifications) is used to create links interconnecting the nodes. In this random generator, edges are initiated between pairs of nodes (u, v) with a probability that depends on the distance between them. The edge probability is given by

$$P(u,v) = \beta \exp\frac{-d(u,v)}{L\alpha}, \tag{2}$$

where $d(u, v)$ is the distance from node u to v, L is the maximum distance between two nodes, and α and β are parameters in the range (0, 1). Larger values of β result in graphs with higher edge densities, whereas small values of α increases the density of short edges relative to longer ones.

In our experiments, we used random networks with an average node degree of 4, which is close to the average node degree of the current Internet. The values for the parameters α and β are 0.15 and 2.2,

respectively. In addition, the sequence of random requests is generated for adding and removing nodes. A simple probability model is used to determine whether the request is adding a node to the multicast group or removing the node from the multicast group. The probability for adding a node to the multicast group is determined by the probability function [10].

$$Prob(add) = \frac{\gamma(N-M)}{\gamma(N-M)+(1-\gamma)M},$$

where M is the current number of nodes in the multicast group, N is the number of nodes in the network, and γ is a parameter of real number in the range (0, 1]. The parameter γ determines the size of the multicast group in equilibrium. Note that

- $Prob(add) = \frac{1}{2}$ if $M = \gamma N$,
- $Prob(add) > \frac{1}{2}$ if $M < \gamma N$, and
- $Prob(add) < \frac{1}{2}$ if $M > \gamma N$.

The probability that a request will be a delete-request is given by $Prob(del) = 1 - Prob(add)$. If the request is a node addition, a node is randomly chosen from the nodes that are not in the multicast group. It is then added to the multicast group. If the request is a node removal, a node excluding the source node is randomly chosen from the multicast group. It is then removed from the multicast group.

The above proposed technique is then applied to 50- and 30-node networks with a group size of 15 and 12 members, respectively. A sequence of 12 events is generated using a fixed value for $\gamma = M/N$, for which $Prob(add) = 1/2$. We are comparing our approach with other static multicast routing algorithms. Instead of using the cost of the multicast tree as the performance measure directly, a performance measure called efficiency [10] is used and described as follows. Given a multicast group, let C_A denote the cost of the multicast tree constructed by algorithm A and let C_B denote the cost of the multicast tree constructed by the competitive algorithm B. Define the efficiency of algorithm A with respect to algorithm B as the ratio C_A/C_B. Note that the lower the mean efficiency is, the better the performance of the proposed algorithm, as cost is calculated in terms of low bandwidth, low delay variation, and small number of Steiner nodes.

5 Results and Discussion

Table 1 shows the simulation results obtained for each event in a 50-node network with a group size of 15. An event is either leaving or joining with a probability of 1/2. It is observed that by using the proposed technique, better performance has been achieved for almost all events when compared with the static multicast routing algorithms KPP, BSMA, and CAO, although these static algorithms reconstruct the whole tree. Overall, the average bandwidth cost is 0.78%, 4.4%, and 13.6% lower, and the average number of Steiner nodes is 21%, 29%, and 36% fewer when compared with KPP, BSMA, and CAO, respectively, for a 50-node network with a group size of 15. This improvement is due to the fact that the proposed approach encapsulates several objective functions using fuzzy logic, whereas a static multicast tree only optimizes bandwidth cost under some delay constraint. In addition, a static multicast tree can also suffer from local minima due to the NP-hard nature of the problem. However, there is performance drop in delay variance when compared with KPP, BSMA, and CAO. This is mainly due to events #8 and #9 in which there is a sudden increase in delay variance. Similar performance has been observed for 30- and 70-node networks with a group size of 12 (Tables 2 and 3).

It should be clear that performance gain may only last for some events. If the network topology is significantly changed, then the whole multicast tree needs to be reconstructed using TS. This can be depicted from

Table 1. Comparison of Dynamic Greedy Algorithm, BSMA, KPP, and CAO.

		Proposed			KPP			BSMA			CAO		
E	**D**	**C**	**V**	**S**	**C**	**V**	**S**	**C**	**V**	**S**	**C**	**V**	**S**
1	37	1342.5	80.3	5	1416.0	90	9	1584	84.53	9	1417.5	83.5	8
2	37	1245.0	85.05	5	1390.5	85.56	8	1486.5	88.81	8	1324.5	92.14	7
3	39	1450.5	93.96	6	1465.5	96.66	8	1563.0	83.55	9	1878.0	103.8	12
4	39	1408.5	99.89	6	1423.5	102.8	8	1521	88.14	9	1836.0	110.4	12
5	39	1450.5	93.96	6	1465.5	96.66	8	1563.0	83.55	9	1878.0	103.8	12
6	39	1363.5	98.15	6	1420.5	93.83	8	1476	93.83	9	1630.5	103.8	10
7	39	1363.5	97.81	5	1420.5	89.05	7	1498.5	106.43	8	1630.5	98.8	9
8	39	1348.5	102.83	5	1405.5	94.02	7	1498.5	102.0	9	1615.5	100.1	9
9	39	1684.5	107.28	8	1450.5	96.83	7	1585.5	97.64	9	1863.0	99.98	11
10	39	1684.5	92.29	9	1551.0	87.62	9	1585.5	86.91	10	1863.0	79.01	12
11	39	1684.5	87.53	8	1720.5	78.6	10	1585.5	63.25	10	1863.0	77.78	11
12	39	1684.5	92.18	9	1720.5	83.23	10	1585.5	64.14	11	1699.5	76.85	10
Average		**1362.4**	94.3	**6.5**	1373.1	91.2	8.3	1425.6	**86.4**	9.2	1576.85	94.2	10.3

E, event; D, maximum end-to-end delay in ms; C, cost in Mbps; V, delay variance in μs; S, number of Steiner nodes. Group size=15; network size=50.

Table 3 for a 70-node network. A significant change in bandwidth cost is observed after event #8, and our proposed approach never performs well when compared with static approaches after that event. Thus, it is improvement to find a static multicast tree using TS after that event.

Tables 4, 5, and 6 show the efficiency of the proposed algorithm when compared with KPP, BSMA, and CAO calculated from the average values in Tables 1, 2, and 3, respectively. As clear from the tables, the efficiency is large when compared with static multicast routing algorithms although they generate tree from scratch, whereas the proposed dynamic greedy method involves only in the establishment of a new path. Our proposed dynamic multicast routing technique performs better when bandwidth cost and Steiner nodes are used as objective functions. The results show the usefulness of the proposed approach for dynamic multicast tree generation, and we present our findings related to this problem. The proposed greedy approach was run on 30-, 50-, and 70-node networks.

Table 2. Comparison of Dynamic Greedy Algorithm, BSMA, KPP, and CAO.

		Proposed			KPP			BSMA			CAO		
E	**D**	**C**	**V**	**S**	**C**	**V**	**S**	**C**	**V**	**S**	**C**	**V**	**S**
1	27	942.0	45.71	4	994.5	33.31	6	997.5	17.05	5	1072.5	21.83	6
2	27	888.0	25.4	4	955.5	17.76	6	945.0	23.44	6	973.5	23.82	6
3	27	807	18.01	4	807	15.2	4	1056	23.795	7	1020	27.2	6
4	27	807	16.05	3	807	14.25	3	919.5	19.1	6	996	26.49	6
5	27	883.5	14.3	4	883.5	12.97	4	1060.5	24.94	6	1060.5	23.91	6
6	27	910.5	14.3	4	849	13.53	3	940.5	17.82	5	916.5	21.44	6
7	28	910.5	12.74	3	862.5	13.04	3	951	22.06	6	906	19.46	6
8	28	891	13.67	3	858	11.74	3	951	19.9	7	906	20.15	7
9	28	1036.5	16.86	3	975	14.8	3	951	18.56	6	906	18.872	6
10	28	1036.5	14.03	4	1168.5	18.5	6	793.5	18.92	6	825	18.9	6
11	28	1176	19.31	5	1168.5	17.6	5	793.5	41.08	5	825	38.8	5
12	28	1281	21.36	5	1218	18.26	4	784.5	42.52	6	816	39.15	6
Average		964.1	19.3	**3.8**	962.3	**16.8**	4.2	**928.6**	24.1	5.9	935.3	25.0	6.0

E, event; D, maximum end-to-end delay in ms; C, cost in Mbps; V, delay variance in μs; S, number of Steiner nodes. Group size=12; network size=30.

Table 3. Comparison of Dynamic Greedy Algorithm, BSMA, KPP, and CAO.

		Proposed			KPP			BSMA			CAO		
E	D	C	V	S	C	V	S	C	V	S	C	V	S
1	29	1750.5	43.86	12	2020.5	35.7	15	1816.5	36.92	13	2049	37.52	14
2	29	1750.5	36.52	13	1917	38.7	15	1816.5	39.56	14	1945	35.56	14
3	29	1750.5	33.61	12	2265	35.5	17	2038.5	32.64	15	2202	32.56	15
4	29	1695	35.17	12	2070	38.20	16	1941	39.12	14	2112	35.00	15
5	33	1914	45.47	15	2020.5	40.14	15	1804.5	40.11	15	2023.5	64.92	14
6	33	1830	49.46	15	1936.5	41.46	15	1720.5	42.20	15	1846.5	65.40	14
7	33	1830	45.93	14	2115	50.95	16	1770	46.37	14	1846.5	60.28	13
8	33	1830	47.80	15	2115	55.80	17	1824	49.68	17	1846.5	66.25	14
9	33	2124	55.61	17	2115	52.77	16	1824	46.15	16	1846.5	61.64	13
10	33	2124	53.71	18	1957.5	40.25	19	1737	37.15	16	1695	47.94	14
11	47	2382	93.51	19	1512	120.41	13	1423.5	96.67	13	1354.5	130.85	12
12	47	2332.5	105.7	18	1428	125.4	13	1339.5	101.33	13	1270.5	137.54	12
Average		1942.8	53.9	15	1956	56.3	15.9	1754.6	50.7	14.6	1836.5	64.6	13.7

E, event; D, maximum end-to-end delay in ms; C, cost in Mbps; V, delay variance in μs; S, number of Steiner nodes. Group size=12; network size=70.

Table 4. Efficiency of the Proposed Method.

	Efficiency with KPP	Efficiency with BSMA	Efficiency with CAO
Bandwidth cost	1.002	1.038	1.031
Delay variance	1.153	0.801	1.001
Steiner nodes	0.920	0.772	0.639

Network nodes=30; group size=12.

Table 5. Efficiency of the Proposed Method.

	Efficiency with KPP	Efficiency with BSMA	Efficiency with CAO
Bandwidth cost	0.992	0.956	0.864
Delay variance	1.033	1.091	1.001
Steiner nodes	0.788	0.709	0.634

Network nodes=50; group size=15.

Table 6. Efficiency of the Proposed Method.

	Efficiency with KPP	Efficiency with BSMA	Efficiency with CAO
Bandwidth cost	0.993	1.115	0.955
Delay variance	0.957	1.111	0.784
Steiner nodes	0.963	1.069	1.067

Network nodes=70; group size=12.

5.1 Effect of Fuzzy Function during Node Joining

In our approach, when a new node is not in the tree, a shortest path is first calculated from the new member node to all existing members. This results in a number of multicast trees. As discussed earlier, each multicast tree is then evaluated by using the fuzzy membership function. Figure 2 shows the effect of fuzzy membership

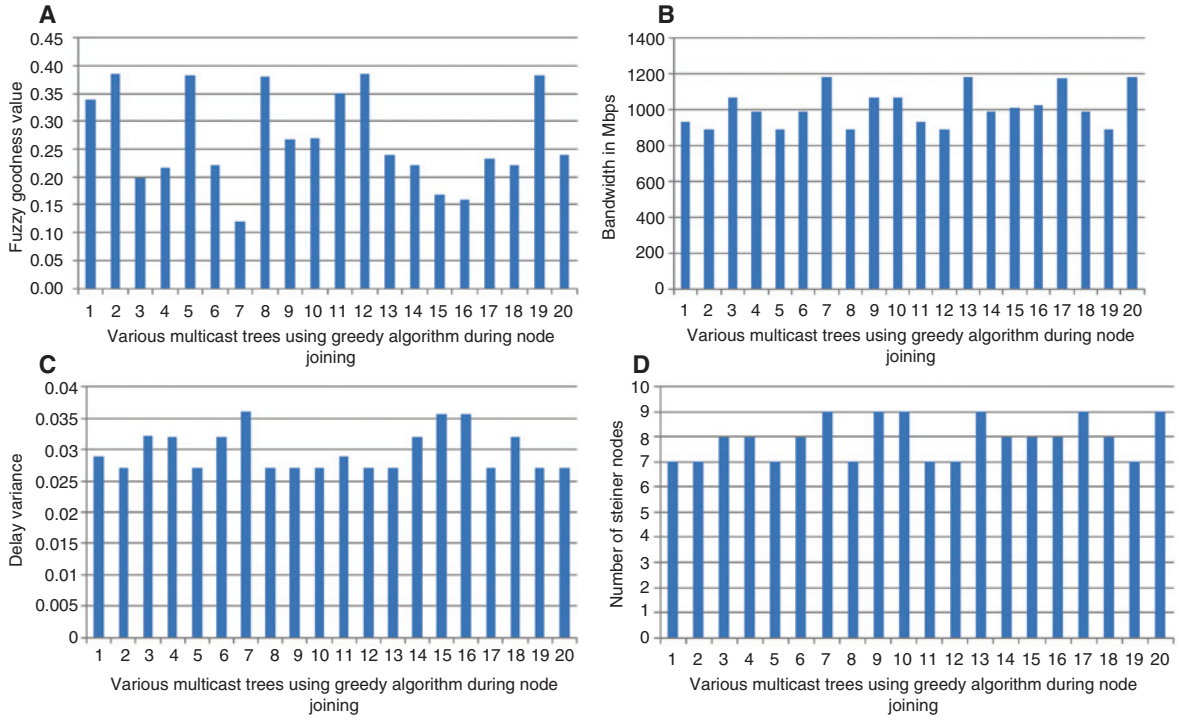

Figure 2. Effect of Fuzzy Membership Function on Various Multicast Trees Generated Using Greedy Algorithm When a Node Joins.

function on various multicast trees generated using the greedy algorithm. It is clear from the graph that this fuzzy membership function provides a convenient method for combining conflicting objectives, including minimizing bandwidth, delay variance, and the number of Steiner nodes. Basically, three objective functions are combined using the fuzzy member function in Figure 2A. The higher the value of the fuzzy membership function, the better is the solution. It is indicated in Figure 2A that the best membership values are found for multicast trees 2, 5, 8, 12, and 19. All these multicast trees will give low bandwidth, low delay variance, and a minimum number of Steiner nodes (see Figure 2B–D).

6 Conclusion

In this article, we have combined the static multicast tree using TS and the greedy algorithm to find a low-cost dynamic multicast tree with desirable QoS parameters. Our proposed algorithm involves only the establishment of a new path and is able to find a multicast tree of good cost for a small number of events and comparable with static multicast routing algorithms that reconstruct the whole tree for every event. Future research is aimed at exploring dynamic multicast routing methods that can also consider sudden failures of links.

Bibliography

[1] M. Armaghan and A. T. Haghighat, QoS multicast routing algorithms based on tabu search with elite candidate list, in: *International Conference on Application of Information and Communication Technologies, 2009, AICT 2009,* pp. 1–5, IEEE, 2009.

[2] M. L. P. Bueno and G. Oliveira, A dynamic multiobjective evolutionary algorithm for multicast routing problem, in: *IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), 2013*, pp. 344–350, IEEE, 2013.

[3] H. Cheng and S. Yang, Hyper-mutation based genetic algorithms for dynamic multicast routing problem in mobile ad hoc networks, in: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom),* pp. 1586–1592, IEEE, 2012.

[4] Y.-f. Dai and N.-f. Li, Comparison and analysis of several non-rearranged dynamic multicast routing algorithms, in: *Third International Conference on Photonics and Image in Agriculture Engineering*, 2013.
[5] H. Fujinokia and K. J. Christensen, A routing algorithm for dynamic multicast trees with end-to-end path length control, *Comput. Commun.* **23** (2000), 101–114.
[6] F. Glover, Tabu search I, *ORSA J. Comput.* **1** (1989), 190–206.
[7] F. Glover, Tabu search II, *ORSA J. Comput.* **2** (1990), 4–32.
[8] S. Hong, H. Lee and B. H. Park, An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership, in: *IEEE INFOCOM*, pp. 1433–1440, 1998.
[9] M. Kim, H. Choo, M. W. Mutka, H.-J. Lim and K. Park, On QoS multicast routing algorithms using k-minimum Steiner trees, *Inform. Sci.* **238** (2013), 190–204.
[10] V. P. Kompella, J. C. Pasquale and G. C. Polyzos, Multicast routing for multimedia communication, *IEEE/ACM Trans. Network.* **1** (1993), 286–292.
[11] H.-Y. Lin and T.-C. Chiang, Efficient key agreements in dynamic multicast height balanced tree for secure multicast communications in ad hoc networks, *EURASIP J. Wireless Commun. Network.* **2011** (2011), 382701.
[12] H. Lin and S. Lai, VTDM – a dynamic multicast routing problem, in: *IEEE INFOCOM*, pp. 1426–1432, 1998.
[13] T. M. Mahmoud, A. I. El Nashar and M. Eman, An efficient genetic algorithm based clonal selection and hill climbing for solving QoS multicast routing problem, *Int. J. Comput. Sci. Issues (IJCSI)* **11** (2014).
[14] G. Manimaran and S. Raghavan, A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees, *IEEE/ACM Transact. Network.* **7** (1999), 514–529.
[15] L. Qingmei and H. Lilin, Research on the routing algorithm based greedy multicast algorithm, in: *International Conference on Computer Applications and System Modeling*, 2010.
[16] R. Qu, Y. Xu, J. P. Castro and D. Landa-Silva, Particle swarm optimization for the Steiner tree in graph and delay-constrained multicast routing problems, *J. Heuristics* **19** (2013), 317–342.
[17] H. F. Salama, D. S. Reeves and Y. Viniotis, Evaluation of multicast routing algorithms for real-time communication on high speed networks, *IEEE J. Sel. Area. Commun.* **15** (1997), 332–346.
[18] M. A. Tahir, H. Youssef, A. Almulhem and S. M. Sait, Fuzzy based multiobjective multicast routing using tabu search, in: *International Conference on Internet Computing*, 2002.
[19] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Mathematica Japonica* **24** (1980), 573–577.
[20] X. Wang, J. Cao, H. Cheng and M. Huang, QoS multicast routing for multimedia group communications using intelligent computational methods, *Comput. Commun.* **29** (2006), 2217–2229.
[21] B. M. Waxman, Routing of multipoint connections, *IEEE J. Sel. Area. Commun.* **6** (1988), 1617–1622.
[22] R. Widyono, The design and evaluation of routing algorithms for real-time channels, Tech Report ICSI TR-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.
[23] Y. Xu and R. Qu, A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems, *Appl. Intell.* **36** (2012), 229–241.
[24] P.-Y. Yin, R.-I. Chang, C.-C. Chao and Y.-T. Chu, Niched ant colony optimization with colony guides for QoS multicast routing, *J. Network. Comput. Appl.* **40** (2014), 61–72.
[25] H. Youssef, A. Almulhem, Sadiq M. Sait and M. A. Tahir, QoS-driven multicast tree generation using tabu search, *Comput. Commun.* **25** (2002), 1140–1149.
[26] Q. Zhu, M. Parsa and J. Garcia, A source-based algorithm for delay-constrained minimum-cost multicasting, in: *IEEE INFOCOM 95*, pp. 353–360, 1995.