ELSEVIER

Computer Communications 25 (2002) 1140–1149

computer communications

www.elsevier.com/locate/comcom

# QoS-driven multicast tree generation using Tabu search

Habib Youssef[a,1], Abdulaziz Al-Mulhem[b,*], Sadiq M. Sait[b], Muhammad Atif Tahir[b]

[a]*Department des Sciences de Informatique, Faculte des Sciences de Monastir, Monastir 5019, Tunisia*
[b]*Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia*

Received 1 August 2001; revised 30 October 2001; accepted 28 November 2001

**Abstract**

Many multimedia communication applications require a source to transmit messages to multiple destinations subject to Quality-of-Service (QoS) delay constraint. The problem to be solved is to find a minimum cost multicast tree where each source to destination path is constrained by a delay bound. This problem has been proven to be NP-complete. In this paper, we present a Tabu Search (TS) algorithm to construct a minimum cost delay bounded multicast tree. The proposed algorithm is then compared with many existing multicast algorithms. Results show that on almost all test cases, TS algorithm exhibits more intelligent search of the solution subspace and is able to find better solutions than other reported multicast algorithms. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords*: QoS routing; Multicast tree; Tabu Search; Optimization; Shortest path

## 1. Introduction

The Internet is widely recognized to deliver connectivity to the data world. The monetary investment in the Internet is enormous. Many Internet applications involve one-to-many or many-to-many (multipoint) communications, where one or more sources send data to multiple receivers. It is possible to provide transmissions to multiple receivers in three different ways: *unicast*, where a separate copy of the data is delivered to each recipient; *broadcast*, where a data packet is forwarded to all portions of the network even if only a few of the destinations are intended recipients; and *multicast*, where a single packet is addressed to all intended recipients and the network replicates packet only as needed. Internet applications that use unicast include E-mail and Web browsers. Potential applications of multicasting are in the business world, to help to increase the ability of organizations to communicate and collaborate, leveraging more value from network investment. Examples of multicasting are the transmission of corporate messages to employees, video and audio conferencing for remote meetings and teleconferencing, transmission over networks of live TV or radio news and entertainment programs, and many more. These new applications are compelling the need for advances in traffic handling to overcome bottlenecks [1]. Figs. 1 and 2 show the difference between unicast and multicast data flows [1].

To support multicast communication efficiently, one of the key issues that needs to be addressed is routing. Routing primarily refers to the determination of a set of paths to be used for carrying the messages from source nodes to destination nodes. In multicasting, the routing function mainly finds the best route from source to all destinations in a multicast group. It is important that the routes used for such communications consume a minimal amount of resource. In order to use network resources as little as possible while meeting the network service requirements, the highly recommended solution involves the generation of a multicast tree spanning the source and destination nodes.

Minimum Steiner tree (MST) [2,3] algorithms attempt to minimize the total cost of the multicast tree. Total cost of the multicast tree is generally defined as the sum of costs of all edges in the multicast tree. The cost is usually measured as the bandwidth consumed by the tree. The minimum Steiner tree problem is known to be NP-complete [3].

With the advent of the real-time interactive applications, minimizing delay of the multicast tree is also an important objective along with minimizing cost. The delay corresponds to the time required to deliver a packet from the source to any member of the multicast group. These two goals can be used in the multicast routing algorithm to determine what constitutes a good tree. However, the cost and delay measures individually are insufficient to

* Corresponding author. Tel.: +966-3-860-3588; fax: +966-3-860-3059.
*E-mail addresses:* habibyoussef@yahoo.com (H. Youssef), almulhem@ccse.kfupm.edu.sa (A. Al-Mulhem), sadiq@ccse.kfupm.edu.sa (S.M. Sait), atif@ccse.kfupm.edu.sa (M.A. Tahir).
[1] Tel./fax: +216-3-274373.

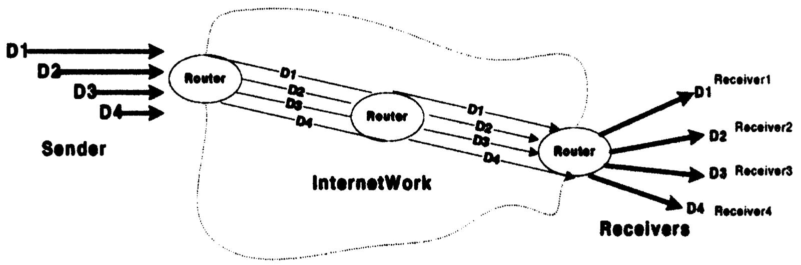

Fig. 1. Unicast dataflow.

characterize a good multicast routing tree for interactive multimedia communication. For example, when the optimization objective is only to minimize the total cost of the tree, we will have a minimum cost tree. Although total cost as a measure of bandwidth efficiency is certainly an important parameter, it is not sufficient to characterize the quality of the tree as perceived by interactive multimedia and real-time applications. This is because, networks supporting real-time traffic need to provide certain quality of service guarantees in terms of the end-to-end delay along the individual paths from sources to each destination node. Therefore, both cost optimization and delay optimization goals are important for the multicast routing tree construction. The problem of minimizing tree cost under the constrained that all path delays are within a user-specified delay bound is referred to as the *delay constraint multicast routing problem.*

The rest of the paper is organized as follows. Section 2 defines the delay constrained multicast routing problem. Section 3 surveys some recently proposed heuristics. Section 4 describes our Tabu-Based QoS Driven Multicast Tree design followed by time complexity of the algorithm. Simulation results and comparison with other reported heuristics are presented in Section 6. Section 7 concludes the paper.

## 2. Problem formulation

The communication network is modeled as a graph $G = (V, E)$, where $V$ is a set of nodes and $E$ is a set of edges (links). A link represents a physical connection between two nodes. The cost and delay of an edge $e$ are denoted by cost($e$) and delay($e$), respectively, where cost($e$) and delay($e$) are positive real numbers. A directed network model is assumed, i.e. the links $e = (u, v)$ and $e = (v, u)$ cannot be used interchangeably.

1. *Link cost function.* The link cost is a function of the amount of traffic traversing the link $e$ and the expected buffer space needed for that traffic [2,4].
2. *Link delay function.* The delay of a link is the sum of the queuing delay, transmission delay, and propagation delay of the link. Let $D$ be the set of destination nodes, then, for each path from a source $s$ to any destination $d \in D$, $D \subseteq V$, the delay of the $s$–$d$ path is defined as the sum of link delays along the path, i.e.

$$\text{delay}(\Pi_{s,d}) = \sum_{e \in \Pi_{s,d}} \text{delay}(e) \qquad d \in D$$

   The maximum end-to-end delay of a multicast tree is the maximum delay from the source to any multicast group member, i.e.

$$\text{Max\_delay}(s, D) = \max_{d \in D} \Big( \sum_{e \in \Pi_{s,d}} \text{delay}(e) \Big) \qquad d \in D$$

3. *Delay bound function.* An upper bound $U$ is assigned to the delay along every path from a source to any destination $d \in D$, $D \subseteq V$. The delay bound for each destination node can be different since each communication link in the network can have different delay constraints as specified by the multicast application.

The delay constrained multicast routing problem [5–7] is to determine a multicast tree connecting the source node $s$ to every destination $d \in D$ node such that the cost of this tree

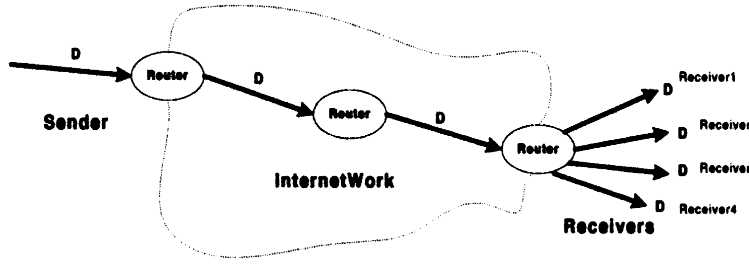

Fig. 2. Multicast dataflow.

```
Ω       : Set of feasible solution
S       : Current Solution
S*      : Best admissible solution
Cost    : Objective function
N(S)    : Neighborhood of solution S
V*      : Sample of neighborhood solutions
T       : Tabu list
AL      : Aspiration Level

        Begin
1.      Start with an initial feasible solution $S \in \Omega$.
2.      Initialize tabu lists and aspiration level.
3.      For fixed number of iterations Do
4.              Generate neighbor solutions $V^* \subset N(S)$.
5.              Find best $S^* \in V^*$.
6.              If move $S$ to $S^*$ is not in T Then
7.                  Accept move and update best solution.
8.                  Update tabu list and aspiration level.
9.                  Increment iteration number.
10.             Else
11.                 If $Cost(S^*) < AL$ Then
12.                     Accept move and update best solution.
13.                     Update tabu list and aspiration level.
14.                     Increment iteration number.
15.                 End If
16.             End If
17.     End For
        End
```

Fig. 3. Algorithmic description of a short-term Tabu search [8].

is minimal while the total delay from the source node to any destination node is not greater than $U$. Mathematically, the problem is to find a tree $T = (V_T, E_T)$ where $V_T \subseteq V$ and $E_T \subseteq E$ such that the total cost of this tree $\sum_{e \in E_T} \text{cost}(e)$ is minimized subject to the following two constraints:

1. $\{s\} \cup D \subseteq V_T$;
2. $\sum_{e \in \Pi_{s,d}} \text{delay}(e) \leq U \forall d \in D$, where $\Pi_{s,d}$ is the set of edges constituting the path from source node $s$ to destination node $d$ in the tree $T$.

## 3. Related work

In this section, we briefly discuss some recently proposed delay constrained Steiner tree heuristics. A good survey for multicast routing algorithms is provided by Salama [2].

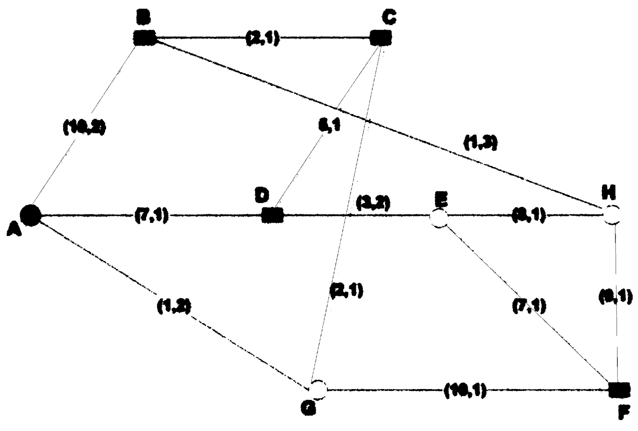

Fig. 4. An example network.

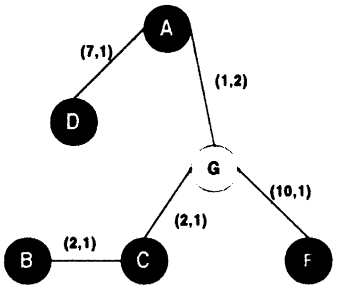

Fig. 5. Sink tree for source A, cost = 22 with $U = 5$.

Multicast routing algorithms can be classified into two categories. The first category is the shortest path algorithms, which minimize the cost of each path from the source node to a multicast group member node. The other category is the minimum Steiner tree algorithms. Their objective is to minimize the total cost of the multicast tree. The problem is proved to be NP-complete [3]. KMP is an efficient minimum Steiner tree heuristic for undirected networks [3]. The first heuristic for delay constrained minimum Steiner tree problem was given by Kompella [5] and is referred to KPP. The heuristic is dominated by the computation of a constrained closure graph which takes time $O(U|V|^3)$. When the link delays and $U$ takes noninteger values, KPP multiplies out fractional values to get integers. Following this approach, KPP is guaranteed to construct a constrained tree if one exists.

The bounded shortest multicast algorithm, BSMA is another delay constrained minimum Steiner tree algorithm that is considered the best in terms of tree cost [6]. BSMA iteratively replaces the edges in the tree until the tree cost cannot be further reduced. BSMA used $K$th shortest path algorithm to find lower cost edges. The time complexity for BSMA is $O(K|V|^3 \log|V|)$, where $K$ may be very large in case of large, densely connected networks, and it may be difficult to achieve acceptable running times.

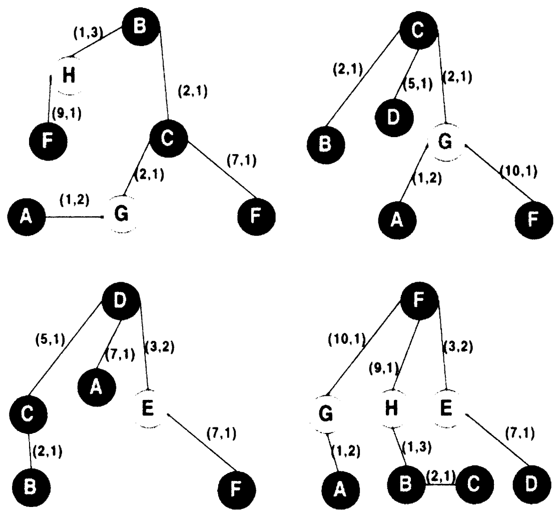

Fig. 6. Sink trees for destinations $D = \{B, C, D, F\}$.

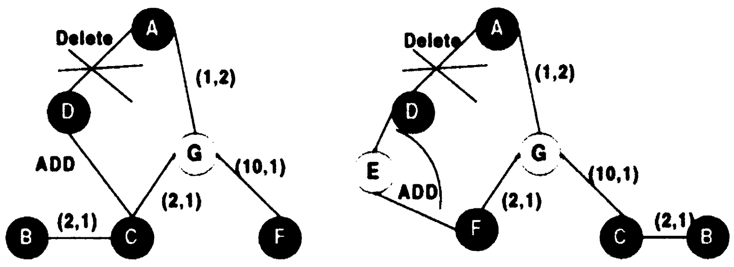

Fig. 7. Two possible neighbors from current solution.

CAO is another heuristics proposed by Widyono [7]. In CAO, the constrained Bellman–Ford algorithm is used to connect one group member at a time to the source. After each run of the constrained Bellman–Ford algorithm, the unconnected member with the minimum cost constrained path to the source is chosen and is added to the existing subtree. The cost of links in the existing subtree is set to zero. CAO is always capable of constructing a constrained multicast tree, if one exists, because of the nature of the breadth-first search conducted.

## 4. Tabu-based QoS driven multicast tree design

The number of possible multicast trees in a computer network of a moderate size is extremely large. Further, because of the multiobjective nature of the problem and the various cost parameters, it is not clear what constitutes the best tree. Modern iterative heuristics such as Tabu Search (TS) have been found effective in tackling this category of problems which have an exponential and noisy search space with numerous local optima [8]. These iterative algorithms are heuristic search methods, which perform a nondeterministic but intelligent walk through the search space.

### 4.1. Overview of Tabu Search

TS was introduced by Fred Glover [9,10] as a general iterative metaheuristic for solving combinatorial optimization problems. TS is conceptually simple and elegant (see Fig. 3). It is a form of local neighborhood search. Each solution $S \in \Omega$ has an associated set of neighbors $N(S) \subseteq \Omega$, where $\Omega$ is the set of feasible solutions. A solution $S' \in N(S)$ can be reached from $S$ by an operation called a move to $S'$. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that were recently explored are declared forbidden or Tabu for a number of iterations. The Tabu status of a solution is overridden when certain criteria (aspiration criteria) are satisfied. To produce good results, any implementation of TS must be engineered to suite the structure of the problem at hand.

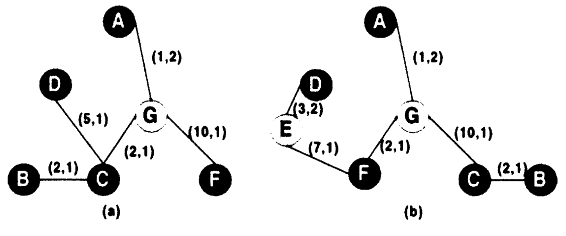

Fig. 8. Final solutions from current solution.

### 4.2. Proposed Tabu Search based algorithm

Our algorithm assumes that sufficient global information is available to the source, i.e. the source node of the network has complete information regarding all network links to construct a multicast tree.

### 4.3. Initial solution

The algorithm starts with an initial feasible solution $S \in \Omega$ built in a greedy fashion as follows: a minimum cost delay constraint Steiner tree is constructed using Dijkstra's shortest path algorithm starting from the source. This results in a set of superpaths. A superpath is defined as a set of edges whose starting and ending nodes are the root of the tree and any node in $\{s\} \cup D$. We call this tree as sink tree of source $s$. To illustrate how our algorithm works, we consider a network as shown in Fig. 4, here $s = \{A\}$ and $D = \{B, C, D, F\}$.

The sink tree for source $s$ is shown in Fig. 5. The solution encoding is path based, where a solution is encoded as an array of $k$ elements where each element is a superpath representing a branch of the multicast tree and $k = |D|$, i.e. cardinality of the set $D$, which contains the members of the multicast group.

The encoding for the initial solution corresponding to the sink tree of Fig. 5 is as follows:

| 0 | 1 | 2 | 3 | |
|---|---|---|---|---|
| $A, D$ | $A, G, F$ | $A, G, C$ | $A, G, C, B$ | Cost = 22 |

### 4.4. Neighborhood solutions

For generating neighbors, we choose a neighborhood structure based on 'delete and add' operations. To reduce the size of the solution space to be searched by Tabu, we

Table 1
Asymmetric load. Cost comparison of various heuristics. Group size = 5 and $U = 0.04$ s

| $N$ | Cost (Mbps) | | | | | Gain (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | KPP1 | KPP2 | CAO | BSMA | Tabu | KPP1 | KPP2 | CAO | BSMA |
| 100 | 1468.2 | 1498.5 | 1262.1 | 1283.4 | 1221.0 | 20.25 | 22.73 | 3.36 | 5.11 |
| 90 | 1167.3 | 1206.5 | 1125.0 | 1093.5 | 1070.4 | 9.05 | 12.75 | 5.10 | 2.16 |
| 80 | 1251.9 | 1372.8 | 1202.0 | 1191.3 | 1135.2 | 10.28 | 20.93 | 5.88 | 4.94 |
| 70 | 1244.4 | 1230.3 | 1028.7 | 1050.3 | 1007.7 | 23.49 | 22.09 | 2.08 | 4.23 |
| 60 | 1270.8 | 1242.3 | 1049.4 | 1053.6 | 997.8 | 27.36 | 24.50 | 5.17 | 5.60 |
| 50 | 984.0 | 1192.2 | 943.2 | 933.9 | 914.1 | 7.65 | 30.42 | 3.18 | 2.16 |
| 40 | 981.3 | 1048.2 | 954.9 | 923.1 | 899.3 | 9.19 | 16.56 | 6.18 | 2.64 |
| 30 | 895.5 | 996.3 | 858.9 | 878.4 | 836.3 | 7.10 | 19.16 | 2.73 | 5.06 |
| 20 | 699.6 | 732.9 | 654.0 | 641.7 | 625.8 | 11.79 | 17.12 | 4.51 | 2.54 |

construct a sink tree for each destination using Dijkstra's shortest path algorithm. The destination is the root of the tree and the remaining nodes out of $\{s\} \cup D$ are the destinations of the new sink tree. Fig. 6 shows the sink trees generated for the network of Fig. 5.

Each iteration begins by generating a set $V^*$ of neighboring solutions. In our algorithm, the set $V^*$ is dynamic, i.e. it varies from one iteration to another. At each iteration, we randomly delete one superpath from the encoding of current solution and then generate different feasible solutions by adding superpaths from one of the destinations sink trees of Fig. 6.

Among the neighbors, the one with the best cost is selected, and considered as new current solution for the next iteration. For example, the two new trees of Fig. 6 are shown in Fig. 7 and are encoded as follows:

| | 0 | 1 | 2 | 3 | Cost |
|---|---|---|---|---|---|
| 8(a) | $D, C$ | $A, G, F$ | $A, G, C$ | $A, G, C, B$ | 20 |
| 8(b) | $D, E, F$ | $A, G, F$ | $A, G, C$ | $A, G, C, B$ | 25 + penalty |

Thus, the new multicast tree of Fig. 8(a) is selected for the next iteration and considered as new current solution. As mentioned above, we randomly delete one superpath from the current solution in the next iteration. Thus, for all candidate solutions in the current iteration, it is guaranteed that at least one solution will give a multicast tree. It might happen that some of the trees violate delay constraint; in that case we assign an extra penalty by increasing its cost, so that it is less likely to be accepted in the candidate list as shown in the tree of Fig. 8(b).

## 4.5. Tabu moves

A Tabu list is maintained to prevent returning to previously visited solutions. This list contains information that to some extent forbids the search from returning to a previously visited solution. In our approach, if a superpath is deleted at iteration *i*, then reintroducing the same superpath in an add operation is Tabu. The Tabu list size is set to 7.

## 4.6. Aspiration criterion

Aspiration criterion is a device used to override the Tabu status of moves whenever appropriate. It temporarily overrides the Tabu status if the move is sufficiently good. The aspiration criterion must make sure that the reversal of a recently made move (that is, a move in the Tabu list) leads the search to an unvisited solution, generally a better one. In our approach, a Tabu superpath will be inserted if implementing and it results in a better cost.

## 4.7. Termination rule

We have used a fixed number of iterations as a stopping criterion. We experimented with different values

Table 2
Symmetric load. Cost comparison of various heuristics. Group size = 5 and $U = 0.05$ s

| $N$ | Cost (Mbps) | | | | | Gain (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | KPP1 | KPP2 | CAO | BSMA | Tabu | KPP1 | KPP2 | CAO | BSMA |
| 100 | 1012.2 | 1012.5 | 1012.5 | 942 | 942 | 6.96 | 6.96 | 6.96 | 0.00 |
| 90 | 1296.0 | 1261.5 | 1242.0 | 1242.0 | 1188.0 | 8.33 | 5.83 | 4.35 | 4.35 |
| 80 | 1426.5 | 1449.0 | 1291.5 | 1291.5 | 1291.5 | 9.46 | 10.87 | 0.00 | 0.00 |
| 70 | 1131.0 | 1155.0 | 1131.0 | 1107.0 | 1066.5 | 5.70 | 7.66 | 5.70 | 3.66 |
| 60 | 732.0 | 873.0 | 684.0 | 798.0 | 684.0 | 21.65 | 6.56 | 0.00 | 14.29 |
| 50 | 1062.0 | 1102.5 | 1077.0 | 1066.5 | 1044.0 | 1.69 | 5.31 | 3.06 | 2.11 |
| 40 | 1198.5 | 1026.0 | 1120.5 | 1093.5 | 1026.0 | 14.39 | 0.00 | 8.43 | 6.17 |
| 30 | 561.0 | 561.0 | 538.5 | 538.5 | 538.5 | 4.01 | 4.01 | 0.00 | 0.00 |
| 20 | 585.0 | 618.0 | 565.5 | 565.5 | 565.5 | 3.33 | 8.50 | 0.00 | 0.00 |

Table 3
Asymmetric load. Cost comparison of various heuristics. Network size = 100 nodes and $U = 10.05$ s

| $G$ | Cost (Mbps) | | | | | Gain (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | KPP1 | KPP2 | CAO | BSMA | Tabu | KPP1 | KPP2 | CAO | BSMA |
| 5 | 729.0 | 726.0 | 697.5 | 879.0 | 697.5 | 4.5 | 4.1 | 0 | 26.0 |
| 10 | 1348.5 | 1686.0 | 1278.0 | 1408.5 | 1248.0 | 8.05 | 35.1 | 2.4 | 12.9 |
| 15 | 2268.0 | 2890.5 | 2263.5 | 2053.5 | 2043 | 11.01 | 41.5 | 10.8 | 0.5 |
| 20 | 2536.5 | 2829.0 | 2479.5 | 2253.0 | 2148.0 | 18.09 | 31.7 | 15.4 | 4.8 |
| 25 | 3088.5 | 3063.0 | 2818.5 | 2677.5 | 2623.5 | 17.72 | 16.75 | 7.4 | 2.0 |

of iterations, and found that for all the test cases, the TS algorithm converges within a maximum of 500 iterations.

## 5. Time complexity

The most expensive step of our heuristic is the initial step, where Dijkstra's shortest path algorithm is used for generating sink trees for the source and the destinations. The worst time complexity of this step is $O(M|V|^2)$ where $M$ is the number of multicast members including the source and $|V|$ is the number of nodes in the network. In TS, one iteration costs $O(M)$. Thus, for $k$ iterations, the cost becomes $O(Mk)$. Thus the expected time complexity of proposed TS algorithm is $O(Mk + M|V|^2)$. The term $Mk$ is usually much smaller than $M|V|^2$. In comparison with other algorithms with respect to time complexity, our TS algorithm compares favorably with KPP and BSMA which have time complexities of $O(U|V|^3)$ and $O(K|V|^3 \log|V|)$, respectively. The time complexity of CAO is not known.

## 6. Simulation results and discussion

The TS algorithm described in this paper has been tested on several randomly generated networks. A random generator [2] (based on Waxman's generator [3] with some modifications) was used to create links interconnecting the nodes. A multicast routing simulator MCRSIM [11] developed at North Carolina State University was used to generate random graphs as described in Ref. [2]. In this model, $n$ nodes are randomly distributed over a rectangular coordinate grid. Each node is placed at a location with integer coordinates. The Euclidean metric is then used to determine the distance between each pair of nodes. On the other hand, edges are introduced between pairs of nodes $u$, $v$ with a probability that depends on the distance between them. The edge probability is given by

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

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

The link delay function delay($e$) is defined as the propagation delay of the link. Queuing and transmission delays are negligible [2]. The propagation speed through the links is taken to be two third the speed of light. The link cost function cost($e$) is defined as the current total bandwidth reserved on the link [4]. MCRSIM already implements many existing Steiner tree heuristics, including BSMA [6], CAO [7], KPP [1]. The average degree of a node is set to 4 and the capacity of each link is 155 Mbps.

The TS based algorithm was run on 20, 30, 40, 50, 60, 70, 80, 90, and 100 nodes random graphs with $\alpha = 0.15$ and $\beta = 2.2$. There are two sets of experiments. In the first set of experiments, TS is compared with KPP1, KPP2, CAO and BSMA for both symmetric and asymmetric networks. Table 1 shows the tree cost for varying network sizes with $U = 0.04$ s for asymmetric network. Table 2 shows the tree cost for varying network sizes with $U = 0.05$ s for symmetric network. Table 3 shows the comparison of Tabu with other algorithms for different group size in a network of

Table 4
Asymmetric load. Cost comparison of various heuristics. Group size = 10 and network size = 100 nodes

| $U$ | Cost (Mbps) | | | | | Gain (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | KPP1 | KPP2 | CAO | BSMA | Tabu | KPP1 | KPP2 | CAO | BSMA |
| 0.04 | 1468.2 | 1498.5 | 1262.1 | 1283.4 | 1221.0 | 20.2 | 22.7 | 3.7 | 5.1 |
| 0.045 | 1452.9 | 1362.6 | 1217.4 | 1232.7 | 1175.4 | 23.6 | 15.9 | 3.6 | 4.8 |
| 0.05 | 1377.3 | 1373.1 | 1182.9 | 1232.4 | 1145.1 | 20.3 | 19.9 | 3.3 | 7.6 |
| 0.055 | 1252.8 | 1350.9 | 1157.4 | 1170.9 | 1139.7 | 9.9 | 18.5 | 1.5 | 2.7 |
| 0.06 | 1262.4 | 1340.1 | 1156.5 | 1177.2 | 1131.9 | 11.5 | 18.4 | 2.2 | 4.0 |
| 0.065 | 1277.7 | 1322.4 | 1156.5 | 1143.9 | 1131.9 | 12.8 | 16.8 | 2.2 | 1.06 |
| 0.07 | 1277.7 | 1322.4 | 1138.8 | 1143.9 | 1131.0 | 12.9 | 16.9 | 0.69 | 1.14 |
| 0.075 | 1277.7 | 1322.4 | 1130.1 | 1143.9 | 1121.7 | 13.9 | 17.9 | 0.75 | 1.98 |

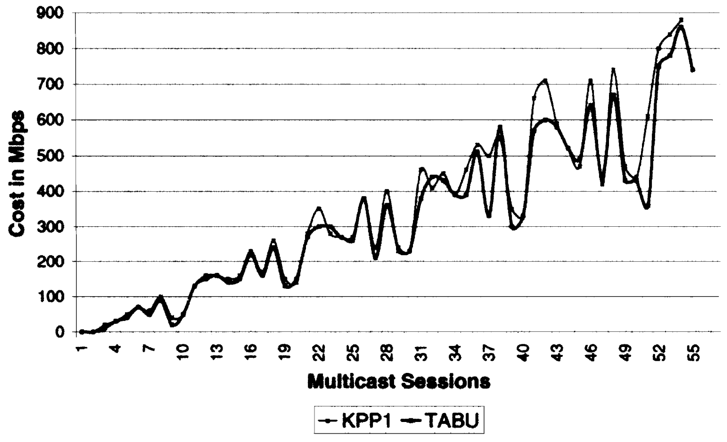

Fig. 9. Number of multicast session versus cost between KPP1 and Tabu. Network size = 20, group size = 5.

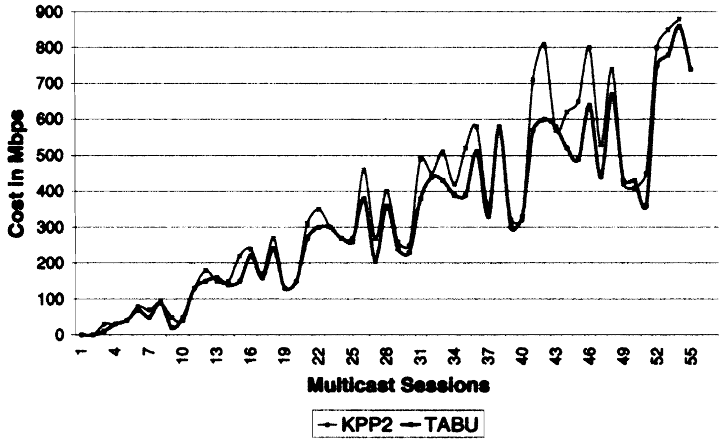

Fig. 10. Number of multicast session versus cost between KPP2 and Tabu. Network size = 20, group size = 5.

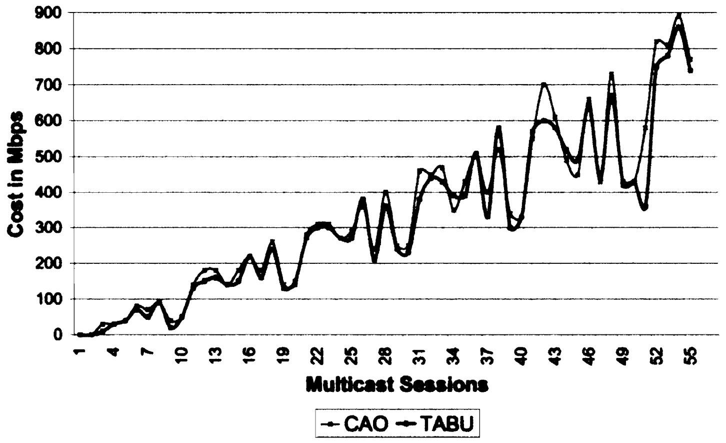

Fig. 11. Number of multicast session versus cost between CAO and Tabu. Network size = 20, group size = 5.

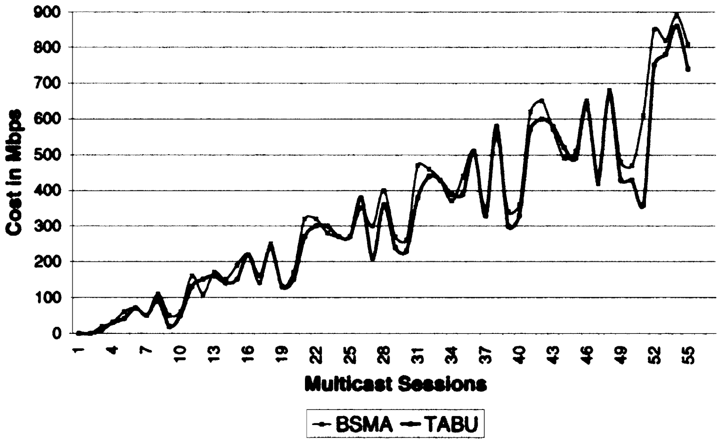

Fig. 12. Number of multicast session versus cost between BSMA and Tabu. Network size = 20, group size = 5.

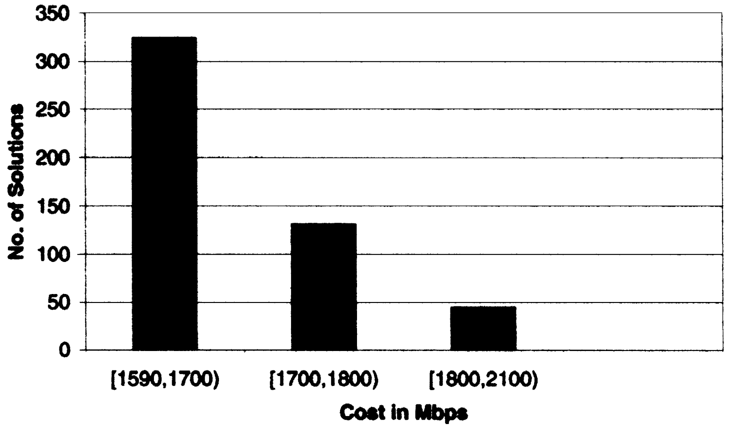

Fig. 13. Total number of solutions evaluated by Tabu search during 500 iterations for different cost ranges. Network size = 100 nodes, group size = 10 and $U = 0.05$ s.

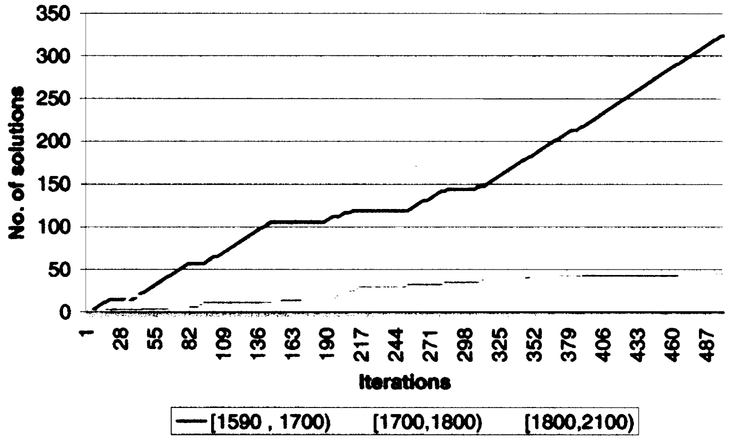

Fig. 14. Total number of solutions evaluated by Tabu search versus iteration number, for different cost ranges. Network size = 100 nodes, group size = 10 and $U = 0.05$ s.

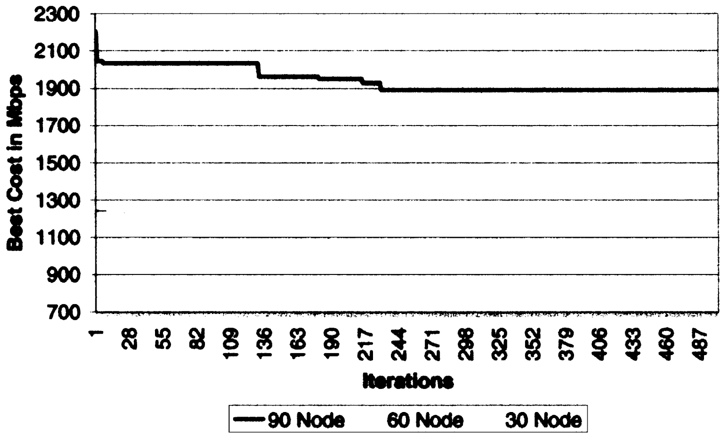

Fig. 15. Cost of best solution found by Tabu search versus iteration number for three networks. Group size = 10 and $U = 0.05$ s.

80 nodes. As the number of group members increases, TS performs better in terms of tree cost. Table 4 shows the comparison of TS algorithm with other algorithms with respect to solution cost for a network of 40 nodes and increasing values of $U$. Our proposed TS based heuristic was able to identify better trees for all test networks. TS achieves better cost than BSMA for all scenarios we tested. BSMA is proved to be within 7% of the optimal for small networks (<40 nodes) [2].

In the second set of experiments, a completely unloaded network is taken and kept adding multicast sessions and constructing the corresponding multicast trees until the cumulative tree failure rate exceeded 15%. A multicast session consisted of a random source node generating VBR video traffic with an equivalent bandwidth of 0.5 Mbps, and a multicast group of randomly chosen destination nodes. The experiment was repeated with different multicast groups [2]. Figs. 9–12 show the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The Tabu able to find multicast session of better quality in almost all cases as compared to KPP1, KPP2, CAO and BSMA.

Fig. 13 shows how well focused is TS on the good solution subspace. As it is clear from the figure, more than 50% of the multicast trees found and evaluated by TS were in the good solution subspace (bar chart highly skewed towards the left), i.e. in the cost interval (1590–1700). Fig. 14 tracks with time the total number of solutions found by the proposed TS algorithm for various cost intervals. The plot clearly indicates that as more iterations are evaluated, TS keeps converging to better solution subspaces. For example, very few solutions (<20) are found beyond 300 iterations in cost interval (1590–1700). Figs. 13 and 14 clearly indicate that TS has been well tuned to the problem addressed in this work. Fig. 15 tracks the cost of the best solution over time. As is clear, for small networks size (<60), TS converges within a maximum of 100 iterations.

## 7. Conclusion

In this paper, we have presented a TS algorithm for delay constrained multicast routing problem. The proposed TS algorithm was always able to find a multicast tree if one exists. TS is better in terms of tree cost as compared to BSMA, CAO, KPP1 and KPP2. Further, as time elapsed, TS progressively zoomed towards a better solution subspace, a desirable characteristics of approximation iterative heuristics.

## Acknowledgements

The authors acknowledge King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for supporting this research project.

## References

[1] D. Kosiur, IP Multicasting: The Complete Guide to Interactive Corporate Networks, Wiley, New York, 1998.
[2] H.F. Salama, D.S. Reeves, Y. Viniotis, Evaluation of multicast routing algorithms for real-time communication on high speed networks, IEEE Journal on Selected Areas in Communication 15 (1997) 332–346.
[3] B.M. Waxman, Routing of multipoint connections, IEEE Journal on Selected Areas in Communication 6 (9) (1988) 1617–1622.
[4] S. Rampal, D. Reeves, An evaluation of routing and admission control algorithms for multimedia traffic, Computer Communications 18 (10) (1995) 755–768.
[5] P. Kompella, C. Joseph, Multicast routing for multimedia communication, IEEE/ACM Transactions on Networking 1 (3) (1993) 286–292.
[6] Q. Zhu, M. Parsa, J. Garcia, A source-based algorithm for delay-constrained minimum-cost multicasting, Proceedings of IEEE INFOCOM 95, 1995, pp. 353–360.
[7] 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.
[8] S.M. Sait, H. Youssef, General iterative algorithms for combinatorial optimization, IEEE Computer Society (1999).

[9] F. Glover, Tabu search; a tutorial, Technical report, University of Colorado, Boulder, February 1990.

[10] F. Glover, Tabu search and adaptive memory programming-advances, applications and challenges, Technical report, College of Business, University of Colorado at Boulder, 1996.

[11] H.F. Salama, et al., MCRSIM simulator source code and users manual, Center for Advanced Computing and Communication, North Carolina State University, Raleigh, 1995, Available by anonymous ftp at ftp.csc.ncsu.edu:/pub/rtcomm.