PERGAMON

Computers & Operations Research 29 (2002) 905–924

computers & operations research

www.elsevier.com/locate/dsw

Heuristic algorithms for packing of multiple-group multicasting

Chu-Fu Wang, Chun-Teng Liang, Rong-Hong Jan*

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, 30050, Taiwan

Received 1 September 1999; received in revised form 1 July 2000

Abstract

Multicast communication is an efficient routing method for multimedia data distribution, since it can save network bandwidth during the communication session. Thus, the multicast routing problems have received much attention from many researchers. In this paper, we consider a multicast routing problem with multiple multicast sessions under a capacity limited constraint. This problem is formulated as a *tree packing problem*. We propose two heuristic algorithms, *Steiner-tree-based heuristic* (STH) algorithm and *cut-set-based heuristic* (CSH) algorithm, for solving this problem. The simulation results show that the STH algorithm can find a better approximate solution in a shorter computation time compared to CSH. In addition, if the available bandwidth for the service is just enough, the STH and CSH algorithms may fail to find a solution even if the solution exists. The simulation results also indicate that CSH has a higher probability than STH to find a solution. Thus, it is suggested that one can apply the STH algorithm first to solve the tree packing problem. In case STH fails, CSH algorithm will be used instead.

Scope and purpose

Several new applications in multimedia networks have been developed in recent years, such as video conferencing, video on demand and network TV. The routing problem for these multimedia services is primarily concerned with efficiently delivering data to multiple destinations. Multicast communication which delivers data along a tree is a preferred routing method because it can reduce network traffic and save network resources, e.g., bandwidth. Most of the current researches for the multicast routing problems only consider a single multicast session. However, several multicast sessions may occur simultaneously in a network and these multicast sessions will contend for the limited network bandwidth. This creates a new network optimization problem. In this research, we studied an optimal tree packing problem. The goal of this study is to arrange the multiple multicast trees in the network such that the bandwidth constraint is maintained and the overall transmission cost is minimized. We believe that the results are useful for improving network utilization in multimedia networks which have multicast services.

Keywords: Network optimization; Multicast routing; Steiner tree problem; Shortest path problem

* Corresponding author. Tel.: + 886-3-571-5900; fax: + 886-3-572-1490.
E-mail address: rhjan@cis.nctu.edu.tw (R.-H. Jan).

0305-0548/02/$ - see front matter
PII: S0305-0548(00)00092-7

1. Introduction

As multimedia services become more widely used through computer networks with multimedia streams consuming the high bandwidth in a network, conserving network bandwidth becomes increasingly important. Many multimedia applications, such as video on demand and video conferencing use multicast communication to conserve network bandwidth. Given a multicast communication session, the network must establish a communication tree in advance, which spans the source node and all destination nodes in this multicast session. Then, the source node can send identical data to each destination node along this tree. Since, the data can be duplicated at switching nodes (the intermediate nodes of the tree), it is not necessary for the source node to send separate copies to each destination node. Thus, a lot of network bandwidth can be conserved when multimedia transmissions are performed in this manner. Due to the network bandwidth conservation benefit, many researchers have paid a lot of attention to the multicast routing problem which has become an important research subject in computer network technology.

Usually, the objective functions that are considered in multicast routing problems are to minimize the transmission cost of the routing tree. For the unconstrained case, it is known as the Steiner tree problem. In this problem, we are given a source node s and a set of destination nodes K and minimum cost tree that contains every node in set $K \cup \{s\}$ that must be determined. In the remainder of this paper, we refer to set $K \cup \{s\}$ as the *member node set* with respect to a multicast session. For example, in Fig. 1(a), node s is the source node and nodes v_2, v_3, v_5 are destination nodes. Each link is associated with a nonnegative integer to represent its transmission cost. Fig. 1(b) shows the optimal solution for Steiner tree problem of Fig. 1(a).

The Steiner tree problem is known to be NP-complete [1]. Several solution approaches, such as, dynamic programming [2,3], branch-and-bound [4–6], enumeration approaches [7,8], linear relaxations [9,10] and Lagrangean relaxations [11,12], have been proposed for finding an exact

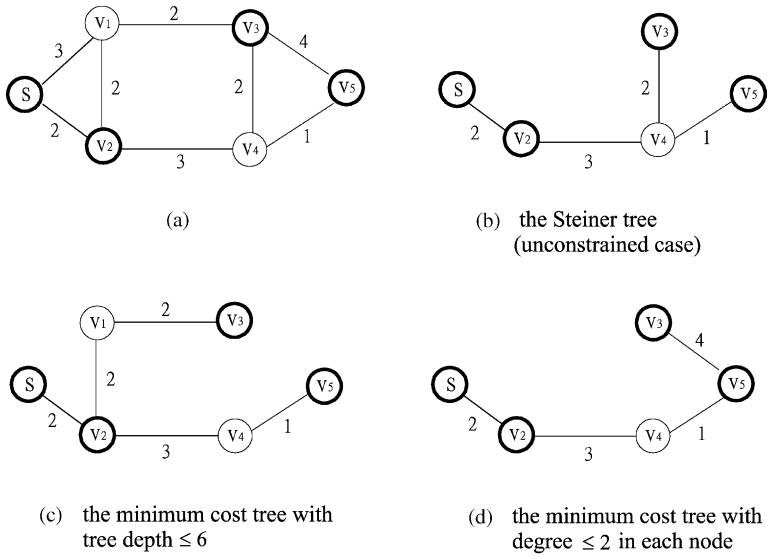

Fig. 1. An example for solving the multicast routing problem with different constraints.

solution for the Steiner tree problem. In addition, there are many heuristic algorithms [13–18] that have been presented for solving the Steiner tree problem. For detailed descriptions and more information on both exact and heuristic methods, one can refer to two survey papers, Hwang and Richards [19] and Winter [20].

Because multimedia transmission is time sensitive, the quality of service (QoS) requirements (such as, end-to-end transmission delay and the data duplicated time consumption in the intermediate nodes) have also been considered in multicast routing problems. We refer to these problems as constrained multicast routing problems. Two types of constraints, which are related to the QoS requirements, are considered in solving these routing problems. The first is to find the minimum cost tree under a bounded tree depth [21–23] (which is related to the QoS requirement of end-to-end transmission delay) (see Fig. 1(c)). Note that, for simplicity, we assume that the value of the transmission cost associated with each link is equal to its transmission delay in this example. The second type of constraint is to find the minimum cost tree under a bounded degree for each intermediate node (which is related to the QoS requirement of the duplicate time consumption in each intermediate node [24]) (see Fig. 1(d)).

However, in the real world, several multicast sessions will occur simultaneously. When several multicast sessions are set up in the same period of time, the routing involves finding a set of routing trees, one for each multicast session. If the network bandwidth is large enough, then the minimum cost tree (the Steiner tree) can be set up for each multicast session. The overall cost can also be minimized. Otherwise, these multicast trees must accommodate each other to satisfy the bandwidth constraint. Therefore, a new optimization problem is created. The new problem is more difficult than the multicast routing problem. Note that, the problems that consider multiple multicast sessions as coexisting are referred to as the *group multicast routing problem*. Until now, only a few related papers have been published [25–28]. Wang et al. [25] considered the problem of how to allocate network bandwidth to each multicast session in a VOD system, such that the number of VOD customers served is maximized and the bandwidth constraint is maintained. However, the multicast trees of this solution are all rooted at the same source node (the server site). Jia and Wang [26] proposed a group multicast routing using multiple Steiner trees. Priwan et al. [27] and Chen et al. [28] used an integer-programming approach to solve the group multicast routing problem. However, the member node sets that are considered in those papers are all identical.

In this paper, we consider a more general group multicast routing problem (the *tree packing problem*), in which the member node sets may be different in each multicast session. We propose two heuristic algorithms, Steiner-tree-based heuristic (STH) algorithm and cut-set-based heuristic (CSH) algorithm, to find an approximate solution for this problem. The remainder of this paper is organized as follows. Section 2 describes the network model and the formulation for the tree packing problem. Section 3 proposes STH and CSH algorithms for solving the tree packing problem. In Section 4, we give the simulation results for these algorithms. Finally, the concluding remarks are given in Section 5.

2. Statement of the tree packing problem

Consider an undirected network $N = (V, E, \mathbf{c}, \mathbf{b})$, where V is the set of vertices which represents communication nodes, E is the set of edges which represents communication links, $\mathbf{c}: E \to R^+$ is

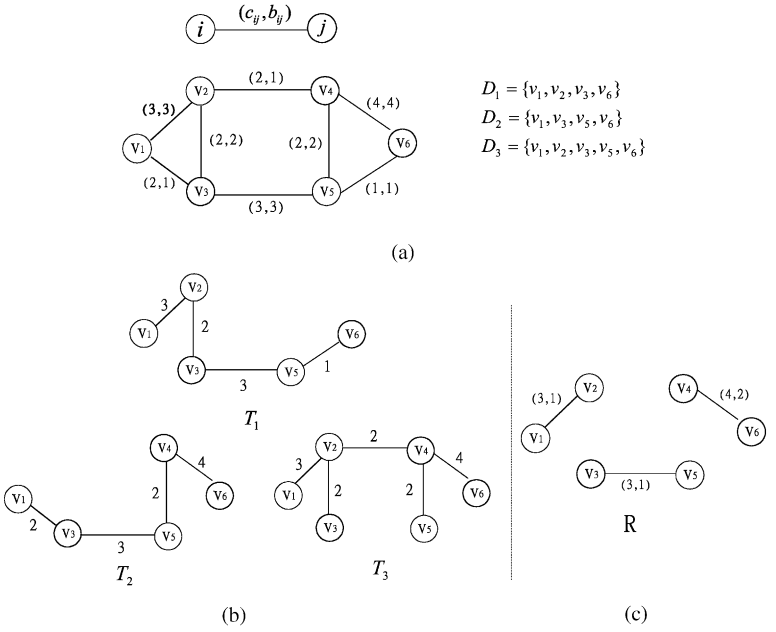

Fig. 2. (a) An instance of tree packing problem (b) a feasible solution for this tree packing problem (c) the residual network.

a cost function and $\mathbf{b}: E \to Z^+$ is a capacity function. The cost c_{ij} of a link $(i, j) \in E$ represents its communication cost. The capacity b_{ij} of a link $(i, j) \in E$ represents the amount of data that can be transmitted through this link. Note that, we denote $G_N = (V, E)$ as the underlying graph of network N. Let H be a subgraph of G_N. The cost (weight) of H with respect to the cost function $\mathbf{c}$ is defined as $\mathbf{c}(H) = \sum_{(i,j)\in E(H)} c_{ij}$. We assume that any multicast stream that flows in each link requires an unit of link capacity. Thus, the link capacity for each link is equal to the maximum number of multicast stream that can flow in it. Suppose there are k multicast sessions occurring simultaneously in the network. Sets D_1, $D_2, \ldots,$ and D_k are member node sets for each multicast session, respectively. The *tree packing problem* involves finding a set of routing trees $\{T_1, T_2, \ldots, T_k\}$ under the bandwidth constraint, such that T_i contains the member node set D_i $(1 \leqslant i \leqslant k)$, and the overall cost of these routing trees (i.e., $\sum_{i=1}^{k} \mathbf{c}(T_i)$) is minimized.

For example, an undirected network is shown in Fig. 2(a). Each link (i, j) is associated with (c_{ij}, b_{ij}), where c_{ij} represents its communication cost and b_{ij} represents the available capacity of this link. Suppose, there are three multicast sessions D_1, D_2 and D_3 occurring within a time unit. Fig. 2(b) shows a feasible solution $\{T_1, T_2, T_3\}$ for the tree packing problem in Fig. 2(a). The tree packing problem can be mathematically stated as follows:

$$\text{Minimize} \quad \sum_{m=1}^{k} \mathbf{c}(T_m)$$

$$\text{Subject to} \quad \sum_{m=1}^{k} x_{ij}^{m} \leqslant b_{ij}, \quad \forall (i, j) \in E(G_N), \tag{1}$$

$$D_m \subseteq V(T_m), \quad 1 \leqslant m \leqslant k, \tag{2}$$

$$x_{ij}^m = 0 \text{ or } 1 \quad \text{for all } i, j, m, \tag{3}$$

where

$$x_{ij}^m = \begin{cases} 1 & \text{if } (i, j) \in E(T_m), \\ 0 & \text{otherwise.} \end{cases}$$

Constraint (1) ensures that the multicast stream flow in each link does not exceed the bandwidth boundaries. Constraint (2) ensures that each multicast tree must involve every node in its member node set.

In the rest of this section, we define some notations that will be used throughout this paper. Given an undirected network $N = (V, E, \mathbf{c}, \mathbf{b})$ and a subgraph H of $G_N = (V, E)$. We use $T_D^*(G_N, \mathbf{c})$ to refer to the minimum Steiner tree in G_N that depicts set D as the member node set and a cost function $\mathbf{c}$. A *path* is a set of links $\{(i_1, i_2), (i_2, i_3), \ldots, (i_p, i_{p+1})\}$ such that links (i_{l-1}, i_l) and (i_l, i_{l+1}) $(2 \leqslant l \leqslant p)$ have a common endpoint. Let s and t be two specific nodes; s is called the source and t, the sink; set P be a subset of vertices such that $s \in P$ and $t \in \bar{P}$, where $\bar{P} = V - P$. A cut set $(P, \bar{P})$ between s and t is defined by the set of edges whose start-vertex is in P and end-vertex is in $\bar{P}$. The capacity of cut set $(P, \bar{P})$ with respect to the capacity function $\mathbf{b}$ is defined as $\mathbf{b}(P, \bar{P}) = \sum_{(i,j)\in(P,\ \bar{P})} b_{ij}$. The cut-set with minimum capacity taken all cut sets $(P,\ \bar{P})$ in N is called the *minimum cut set*, and we denote it by $(P^*, \overline{P^*})$.

The *residual network* $N - H$, after removing subgraph H from N is an undirected network, where the cost function $\mathbf{c}'$ of $N - H$ is inherited from N but the capacity function $\mathbf{b}'$ of $N - H$ is altered and defined as follows.

$$b'_{ij} = \begin{cases} b_{ij} - 1 & \text{if } (i, j) \in E(H), \\ b_{ij} & \text{otherwise.} \end{cases} \tag{4}$$

Note that the link (i, j) is deleted from $N - H$ when value b'_{ij} becomes 0. For example, Fig. 2(c) shows the residual network R after sequentially removing T_1, T_2 and T_3 from N, i.e., $R = ((N - T_1) - T_2) - T_3$.

3. The solution methods

Since the Steiner tree problem is NP-complete, there does not exist any polynomial-time algorithm to find the exact optimal solution for it. However, we can find the approximate solution for Steiner tree $T_D^*(G_N, \mathbf{c})$ using any heuristic algorithm in a reasonable time. One of the popular heuristic algorithms is the Shortest Path Heuristic (SPH) [14]. In SPH, the algorithm starts with an arbitrary multicast member $u \in D$. It then joins the next close node $v \in D - \{u\}$ to the current tree using the shortest path. The algorithm repeats this process until all member nodes have joined the tree. We denote $T_{\text{SPH}}(G_N, D, \mathbf{c})$ as the resulting tree for applying SPH on network N with member node set D and the cost function $\mathbf{c}$. In this paper, we used SPH as the underlying method for the following proposed algorithms.

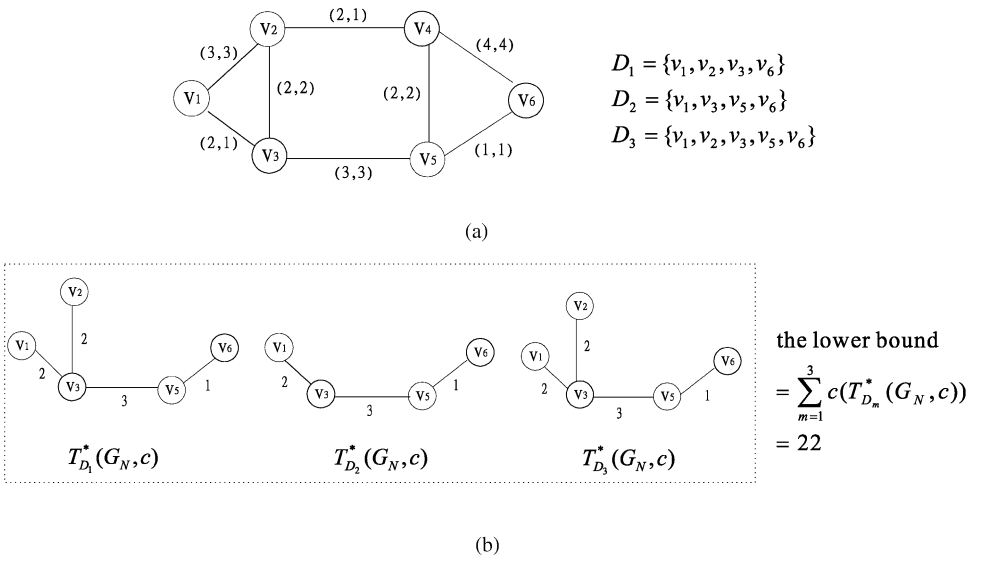

Fig. 3. The lower bound for tree packing problem.

Consider a tree packing problem, we are given an undirected network $N = (V, E, \mathbf{c}, \mathbf{b})$ and k multicast sessions $D_1, D_2, \ldots, D_k$, in which they occur simultaneously. If we ignore the bandwidth constraint and then find the optimal Steiner trees $T^*_{D_1}(G_N, \mathbf{c})$, $T^*_{D_2}(G_N, \mathbf{c}), \ldots, T^*_{D_k}(G_N, \mathbf{c})$ for sessions $D_1, D_2, \ldots, D_k$, respectively. Obviously, the overall cost for these k trees (i.e., $\sum_{m=1}^{k} \mathbf{c}(T^*_{D_m}(G_N, \mathbf{c}))$) is a lower bound for this tree packing problem. Fig. 3(b) shows that the lower bound is equal to 22 for the tree packing problem in Fig. 3(a).

3.1. A greedy algorithm

The lower bound of the tree packing problem discussed above involves ignoring the bandwidth constraint and then solving k Steiner tree problems on G_N. However, the resulting trees $\{T^*_{D_1}(G_N, \mathbf{c}), T^*_{D_2}(G_N, \mathbf{c}), \ldots, T^*_{D_k}(G_N, \mathbf{c})\}$ may be infeasible for the tree packing problem. One of the simple methods for finding a feasible solution is that we apply the Steiner tree heuristic algorithm to the residual network R each time instead of the network N. A greedy heuristic algorithm is given as follows. At first, we use SPH to find k approximate Steiner trees $T_{\mathrm{SPH}}(G_N, D_1, \mathbf{c})$, $T_{\mathrm{SPH}}(G_N, D_2, \mathbf{c}), \ldots, T_{\mathrm{SPH}}(G_N, D_k, \mathbf{c})$ in G_N for multicast sessions $D_1, D_2, \ldots, D_k$, respectively. Assume that $T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})$ has the least cost among these trees (i.e., $\mathbf{c}(T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})) = \min_{1 \leqslant j \leqslant k} \mathbf{c}(T_{\mathrm{SPH}}(G_N, D_j, \mathbf{c}))$). Then, we adopt $T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})$ as the multicast tree for session D_i and put it into the solution set. Now, remove tree $T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})$ from N and thus the resulting network $R = N - T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})$. In the second iteration, apply SPH algorithm to find $k-1$ multicast trees $T_{\mathrm{SPH}}(G_R, D_1, \mathbf{c})$, $T_{\mathrm{SPH}}(G_R, D_2, \mathbf{c}), \ldots, T_{\mathrm{SPH}}(G_R, D_{i-1}, \mathbf{c})$, $T_{\mathrm{SPH}}(G_R, D_{i+1}, \mathbf{c}), \ldots, T_{\mathrm{SPH}}(G_R, D_k, \mathbf{c})$ on the residual network R for the remaining $k-1$ sessions. Similarly, we find a multicast tree with the least cost among these $k-1$ trees and then add this tree to the solution set. The algorithm repeats this process until k multicast trees are determined. Since we solved each

```
Algorithm Greedy-heuristic;
  Input: an undirected network N = (V, E, c, b) and k multicast
         sessions D_1, D_2, ..., D_k.
  Output: a feasible solution {T_1, T_2, ..., T_k} for tree packing problem.
  begin
    R = N;
    I = {1, 2, 3, ..., k};
    for m = 1 to k do
      begin
        T_i = T_SPH(G_R, D_i, c), where
        c(T_SPH(G_R, D_i, c)) = min_{j∈I} c(T_SPH(G_R, D_j, c))
        R = R − T_i;
        I = I − {i};
      end;
  end;
```

Fig. 4. The greedy algorithm for the tree packing problem.

tree problem on the residual network, the bandwidth constraint holds and the resulting solution is feasible. The detailed algorithm is shown in Fig. 4.

3.2. The Steiner-tree-based heuristic algorithm

In this subsection, we present another heuristic algorithm for the tree packing problem. The main idea is that initially we apply the SPH method k times to graph G_N and obtain $\{T_{\mathrm{SPH}}(G_N, D_1, \mathbf{c}), T_{\mathrm{SPH}}(G_N, D_2, \mathbf{c}), \ldots, T_{\mathrm{SPH}}(G_N, D_k, \mathbf{c})\}$. For notation simplicity, we let T_i be $T_{\mathrm{SPH}}(G_N, D_i, \mathbf{c})$ $(1 \leqslant i \leqslant k)$. Note that, the solution set $\{T_1, T_2, \ldots, T_k\}$ may be infeasible. Packing these trees all together some links may violate the bandwidth constraints. We call the violating link as an *overloaded link*. That is, a link $(i,j) \in E(G_N)$ is called an *overloaded link* with respect to solution trees $\{T_1, T_2, \ldots, T_k\}$ when

$$\sum_{m=1}^{k} x_{ij}^m > b_{ij}, \quad \text{where } x_{ij}^m = \begin{cases} 1 & \text{if } (i,j) \in E(T_m), \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

For example, in Fig. 3, the overloaded links are (v_1, v_3) and (v_5, v_6) with respect to solution trees $\{T^*_{D_1}(G_N, \mathbf{c}), T^*_{D_2}(G_N, \mathbf{c}), T^*_{D_3}(G_N, \mathbf{c})\}$. We collect all of the overloaded links into a set S, and call it the *overloaded link set*. Note that, there is no overloaded link if the solution is feasible. In addition, given a set of solution trees $\{T_1, T_2, \ldots, T_k\}$, we let the residual network $R = (((G - T_1) - T_2) \cdots) - T_k$. The basic idea of our solution method is to reduce the number of overloaded links in S until set S becomes empty.

The following process can reduce the number of overloaded links. Let link (i, j) be an overloaded link that is arbitrarily chosen from set S. Note that, the bandwidth of (i, j) is overused. In order to relieve the load on link (i, j), some multicast trees $T_l \in \{T_1, T_2, \ldots, T_k\}$, which contain link (i, j), must be forced to give up link (i,j). If we remove link (i, j) from tree T_l, the graph $T_l - \{(i, j)\}$ will be divided into two subtrees T_l^1 and T_l^2 (see Fig. 5(a)). Thus, we need to re-connect the two subtrees by a path. In order to avoid increasing the number of overloaded links during the solution process, we find a shortest path between T_l^1 and T_l^2 in the residual network R and use it to connect T_l^1 and T_l^2

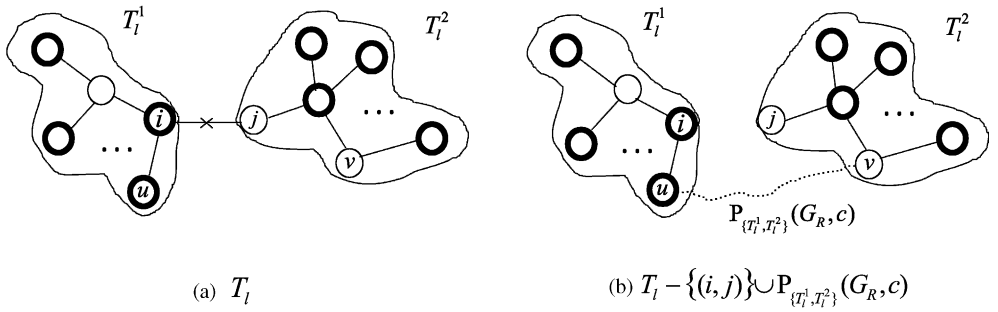

Fig. 5. The substitution process of STH.

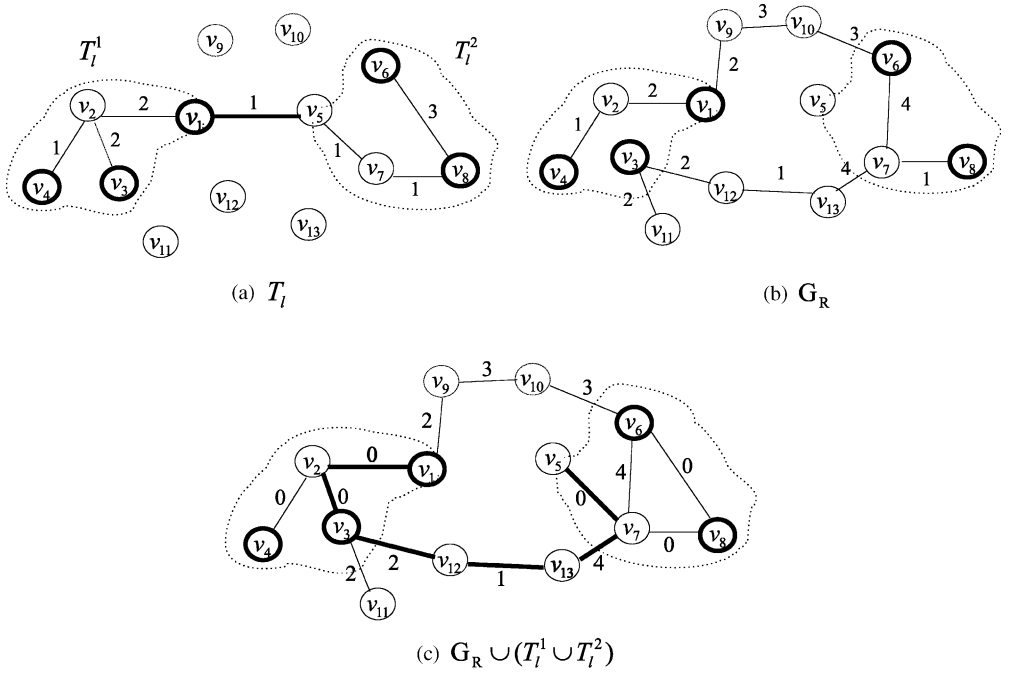

Fig. 6. An example for illustrating the finding process of $P_{\{T_l^1,T_l^2\}}(G_R, \mathbf{c})$.

(see Fig. 5(b)). The shortest path is denoted as $P_{\{T_l^1,T_l^2\}}(G_R, \mathbf{c})$ and we call the above process as a *substitution process*.

Note that, traditionally we can obtain path $P_{\{T_l^1,T_l^2\}}(G_R, \mathbf{c})$ by applying the shortest path algorithm (such as, Dijkstra's algorithm) on each node pair between T_l^1 and T_l^2 and then choose the path which costs the minimum. Thus, there are $|V(T_l^1)|\cdot|V(T_l^2)|$ Dijkstra's algorithms applied, which is computationally very expensive. Indeed, we can obtain path $P_{\{T_l^1,T_l^2\}}(G_R, \mathbf{c})$ by performing Dijkstra's algorithm only once, and the method is illustrated as follows.

Applying Dijkstra's algorithm to find the shortest path from node i to node j on graph $G_R \cup (T_l^1 \cup T_l^2)$ with a cost function $\mathbf{c}'$, is defined as follows.

$$c'_{uv} = \begin{cases} c_{uv} & \text{if } (u,v) \in E(G_R - (T_l^1 \cup T_l^2)), \\ 0 & \text{if } (u,v) \in E(T_l^1 \cup T_l^2). \end{cases} \tag{6}$$

That is, any link (u,v) in tree T_l^1 or T_l^2 has a zero cost. After the shortest path from i to j is found, we remove the subpaths that lie on $T_l^1 \cup T_l^2$. Then, the resulting path is a shortest path which connects T_l^1 and T_l^2. For example, Fig. 6(a) shows a multicast tree T_l that contains an overloaded link (v_1, v_5). The value c_{uv} associated with link (u,v) represents its communication cost, and the member node set is $\{v_1, v_3, v_4, v_6, v_8\}$, which is denoted by a heavy circle. Fig. 6(b) gives the underlying graph G_R of the residual network R. Fig. 6(c) shows the graph $G_R \cup (T_l^1 \cup T_l^2)$ and the associated cost function $\mathbf{c}'$. The heavy lines indicate the shortest path $\mathrm{P}_{\{v_1, v_5\}}(G_R \cup (T_l^1 \cup T_l^2), \mathbf{c}')$. Then, removing the subpaths that lie on $T_l^1 \cup T_l^2$, we have a shortest path $\mathrm{P}_{\{T_l^1, T_l^2\}}(G_R, \mathbf{c}) = \{(v_3, v_{12}), (v_{12}, v_{13}), (v_{13}, v_7)\}$ which connects T_l^1 and T_l^2.

In the substitution process, we choose an overloaded link (i, j) from S and a multicast tree T_l that contains (i, j). Remove link (i, j) from T_l, and then connect subtrees T_l^1, T_l^2 by path $\mathrm{P}_{\{T_l^1, T_l^2\}}(G_R, \mathbf{c})$. A new multicast tree $T'_l = T_l - \{(i, j)\} \cup \mathrm{P}_{\{T_l^1, T_l^2\}}(G_R, \mathbf{c})$ is obtained. Note that, the residual network also needs to be updated by letting $R = R - \mathrm{P}_{\{T_l^1, T_l^2\}}(G_R, \mathbf{c})$. However, the new multicast tree T'_l may be reducible. That is, some unnecessary links in T'_l can be pruned away and the resulting tree will still contain the member node set. A *recycling process* is designed to prune the unnecessary links from T'_l, and add these unnecessary links back to residual network R. For example, Fig. 7(a) shows the results of new multicast tree T'_l and the residual network R of the example in Fig. 6. Since, node v_5 is a leaf node in T'_l but is not a member node, link (v_5, v_7) is an unnecessary link in this multicast tree. Therefore, we remove link (v_5, v_7) from T'_l and then add it into the residual network R. Repeat this process until no leaf node in the multicast tree is a nonmember node. In this example, after removing link (v_5, v_7), we will find that the new multicast tree is no longer reducible. Thus, the recycling process is terminated. Fig. 7(b) shows the result of applying the recycling process in Fig. 7(a). Note that, the overloaded link (i, j) can be removed from the overloaded link set S when the new solution trees $\{T_1, T_2, \ldots, T'_l, \ldots, T_k\}$ satisfy the bandwidth constraint on this link. By continuing to perform the substitution process and recycling process we can reduce the number of overloaded links one by one until no overloaded link exists. Then a feasible solution $\{T_1, T_2, \ldots, T_k\}$ is obtained.

In order to refine this solution, we propose a *refinement process* to produce a better solution. A refinement process involves choosing any two multicast trees, say T_l and T_m in $\{T_1, T_2, \ldots, T_k\}$ and placing them into the residual network R. Next, the two new multicast trees $T'_l = T_{\mathrm{SPH}}(G_1, D_l, \mathbf{c})$ and $T'_m = T_{\mathrm{SPH}}(G_2, D_m, \mathbf{c})$, where $G_1 = G_R \cup T_l \cup T_m$ and $G_2 = G_R \cup T_l \cup T_m - T'_l$, are recomputed. If a better solution is found, (i.e., $\mathbf{c}(T'_l) + \mathbf{c}(T'_m) < \mathbf{c}(T_l) + \mathbf{c}(T_m)$), then trees T_l and T_m can be replaced by T'_l and T'_m, respectively, and a new solution $\{T_1, T_2, \ldots, T'_l, \ldots, T'_m, \ldots, T_k\}$ is obtained. Otherwise, the solution trees remain unchanged. A complete description of our proposed algorithm is shown in Fig. 8.

The complexity analysis of our proposed algorithm is shown as follows. Let $n = |V|$ be the number of nodes in N and m be the maximum size among the member node set $D_1, D_2, \ldots, D_k$

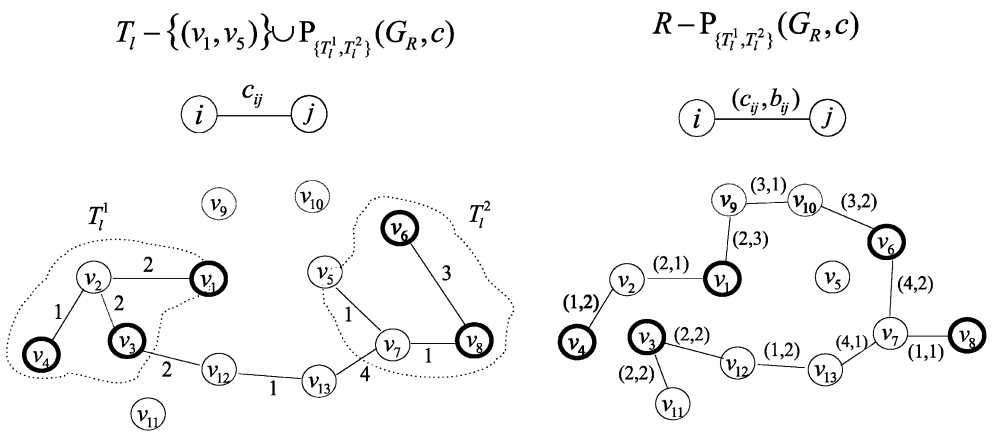

Fig. 7. An example for the recycling process.

(i.e., $m = \max\{|D_1|, |D_2|, \ldots, |D_k|\}$). Since the SPH runs in $O(m(e + n\log n))$ [14] where e is the number of links, and in the worst-case, $O(e) = O(n^2)$, thus the worst-case running time for SPH is $O(mn^2)$. Also, it is known that the worst case-running time for Dijkstra's algorithm is $O(n^2)$. In our proposed algorithm, the initial process performs SPH k times; thus, it takes $O(kmn^2)$ time. Since the substitution process only performs Dijkstra's algorithm once, it takes $O(n^2)$ time. The recycling process can be performed in $O(n^2)$ time. Finally, because the refinement process performs SPH $2 \times C_2^k$ times, the refinement process takes $O(k(k-1)mn^2) = O(k^2mn^2)$ time. Note that, in the worst case, the number of iterations that runs in the loop is equal to the size of overloaded links set S times the maximum number of multicast trees that are congested in each overloaded link (i.e., $O(|S|k) = O(n^2k)$). Therefore, summing the running time for these major processes, our proposed algorithm costs $O(kmn^2 + n^2k(n^2 + n^2) + k^2mn^2) = O(kn^4 + k^2mn^2)$. From the above discussion, we have the following lemma.

```
Algorithm STH;
  Input: an undirected network N = (V, E, c, b) and k multicast sessions
         D_1, D_2, ..., D_k.
  Output: a feasible solution {T_1, T_2, ..., T_k} for tree packing problem.
  begin
    for m = 1 to k do
        T_m = T_SPH(G_N, D_m, c);
    determine the residual network R and the overloaded link set S;
    while (S ≠ ∅) do
      begin
        /* The substitution process */
        choose an overloaded link (i, j) ∈ S and a multicast tree T_l ∈
        {T_1, T_2, ..., T_k}, such that (i, j) ∈ E(T_l);
        T_l = T_l − {(i, j)} ∪ P_{T_l^1, T_l^2}(G_R, c);
        R = R − P_{T_l^1, T_l^2}(G_R, c);
        perform the recycling process;
        if link (i, j) is no longer be overused then S = S − {(i, j)};
      end;
    /* The refinement process */
     for i = 1 to k do
      for j = i + 1 to k do
        begin
          T_i' = T_SPH(G_R ∪ T_i ∪ T_j, D_i, c);
          T_j' = T_SPH(G_R ∪ T_i ∪ T_j − T_i', D_j, c);
          if (c(T_i') + c(T_j') < c(T_i) + c(T_j)) then let T_i = T_i' and T_j = T_j';
        end;
  end;
```

Fig. 8. The STH algorithm.

Lemma 1. *The STH algorithm runs in* $O(kn^4 + k^2mn^2)$ *time.*

3.3. The cut-set based-heuristic algorithm

Note that if the network bandwidth is just sufficient, the STH may fail to find a solution even if the solution exists. This is because the main idea of STH is to find the multicast tree for each session, place these trees to the network and check the feasibility. Thus, some multicast session may be blocked. In this subsection, we will focus on how to utilize the network bandwidth carefully as well as to find the minimum cost solution. The idea of the cut-set-based algorithm is to extract a multicast tree sequentially from the network such that the residual network can be packed with as many multicast trees as possible.

We assume the member node sets are all identical (i.e., $D_1 = D_2 = \cdots = D_k = D$). Let u and v be any two nodes in the member node set. The capacity of the minimum cut-set between u and v is equal to the maximum number of paths from u to v that can be packed on the network. This implies that the minimum capacity between u and v is greater than the maximum number of multicast trees that can be packed in the network. Thus, the capacity of the minimum cut-set between any member node-pair is an upper bound of the number of multicast trees that can be successfully established in the network. Moreover, any multicast tree at least has one link (consumes at least one unit of bandwidth) in the minimum cut-set. Therefore, we find a link from the minimum cut-set as a link of the multicast tree. Intuitively, the number of multicast trees that can be established in the resulting network is maximized.

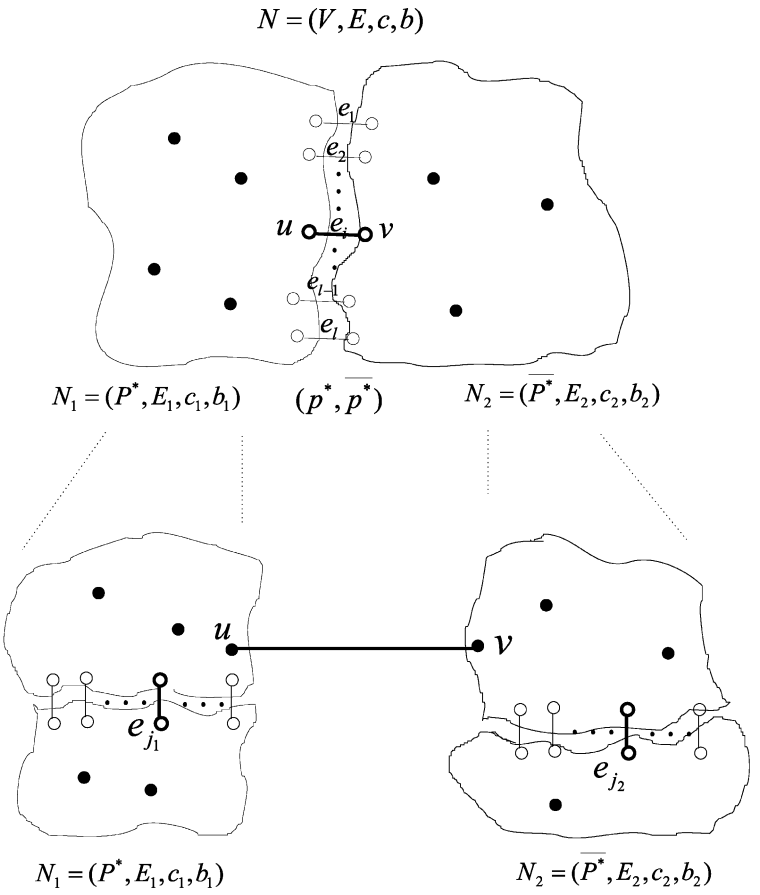

Fig. 9. The tree-extracting procedure.

Now, consider a tree packing problem. We are given an undirected network $N=(V,E,\mathbf{c},\mathbf{b})$ and k multicast sessions $D_1,D_2,\ldots,D_k$, where $D_1=D_2=\cdots=D_k=D$. The following steps can be applied to extract a tree T from the network N for the multicast session D. At first, we compute the all-pair minimum cut-set[1] with respect to the member node set D. That is, for every node pair (u,v) in set D, we find the minimum cut-set between u and v. Thus, we have $C_2^{|D|}$ minimum cut-sets. Then find a cut-set, denoted as minimum all-pair cut-set $(P^*,\overline{P^*})$, whose capacity value is the smallest among $C_2^{|D|}$ minimum cut-sets. Let minimum all-pair cut-set $(P^*,\overline{P^*})=\{e_1,e_2,\ldots,e_l\}$. We find edge $e_i=(u,v)$ such that $\mathbf{b}(e_i)=\max\{\mathbf{b}(e_1),\mathbf{b}(e_2),\ldots,\mathbf{b}(e_l)\}$ and then add e_i to the multicast tree T.

Since set $(P^*,\overline{P^*})$ is an edge cut-set, thus removing $(P^*,\overline{P^*})$ from network N will partition N into two sub-networks, $N_1=(P^*,E_1,\mathbf{c}_1,\mathbf{b}_1)$ and $N_2=(\overline{P^*},E_2,\mathbf{c}_2,\mathbf{b}_2)$ (see Fig. 9). Now, we apply the same procedure to find the edges e_{j_1} and e_{j_2} on networks N_1 and N_2 with member node sets

[1] Gomory and Hu [29] developed a very elegant method for finding all-pair minimum cut-set. Their method only solves $|D|-1$ maximum flow problems.

```
Procedure Tree-extracting(network: N, member node set: D);
begin
    If (|D| ≥ 1) then
      begin
        compute the all-pair minimum cut (P*, P̄*) with respect to set
        D;
        let N1 and N2 be the resulting subnetwork when removing
        (P*, P̄*) from N;
        let e_i = (u, v), where b(e_i) = max_{e_j ∈ (P*, P̄*)} b(e_j);
        T1 = Tree-extracting(N1, (D ∩ P*) ∪ {u});
        T2 = Tree-extracting(N2, (D ∩ P̄*) ∪ {v});
        output(e_i ∪ T1 ∪ T2);
      end;
  end;
```

Fig. 10. The tree-extracting procedure.

```
Algorithm CSH;
  Input: an undirected network N = (V, E, c, b) and k multicast
         sessions D1, D2, ..., Dk.
  Output: a feasible solution {T1, T2, ..., Tk} for tree packing problem.
  begin
    R = N;
    for m = 1 to k do
      begin
        T_m = Tree-extracting(R, D_m);
        R = R − T_m;
      end;
        Perform the refinement process;
  end;
```

Fig. 11. The CSH algorithm.

$(D \cap P^*) \cup \{u\}$ and $(D \cap \overline{P^*}) \cup \{v\}$, respectively, and add e_{j_1} and e_{j_2} to T. Note that, the reason for adding nodes u and v to the member node subsets is to ensure that the resulting tree T is connected. The procedure for constructing a multicast tree will stop when the member node subset contains only one node. In this way, a connected multicast tree T for member node set D will be eventually established and we call the above operations a *tree-extracting* procedure.

The complete operations of CSH is that, initially, we perform the tree-extracting procedure for multicast session D_1 on network N, and obtain a multicast tree T_1. Let the residual network be $R = N - T_1$. Next, performing the tree-extracting procedure for multicast session D_2 on the residual network R, obtain a multicast tree T_2. Repeat the same operations and finally a set of multicast tree $\{T_1, T_2, \ldots, T_k\}$ will be obtained. In order to improve the solution, the refinement process is performed on the set $\{T_1, T_2, \ldots, T_k\}$. The detailed description for tree-extracting procedure and CSH are shown in Fig. 10 and Fig. 11, respectively. Note that the CSH algorithm can be applied to the problem in which the member node set is not identical.

4. Simulation results

In this section, the performance of the proposed algorithms for solving the tree packing problem is described. The network topologies used in the simulations were generated by a random graph

model proposed by Waxman [30]. The generator first randomly distributed n nodes over an $L \times L$ square coordinate grid, where L is a fixed integer number. The link between any two nodes u and v is added by the probability function $P((u,v)) = \beta \exp(-d(u,v)/\alpha l)$, where $d(u,v)$ is the Cartesian distance between nodes u and v, l is the maximum possible distance between any two nodes and parameters α and β are real numbers in the range (0, 1]. Note that parameters α and β can be appropriately selected to obtain the desired characteristics in the resulting graph. In the simulation, we set $\alpha = 0.3$ and $\beta = 0.25$ to obtain a sparse graph. We only use the connected graphs for simulation. If the generated graph is not connected, then discard it.

In the simulation, we observed the performance of the proposed algorithms (the quality of the solutions and the blocking rate[2] of the algorithms). The performance of these algorithms are influenced by the following factors:

(1) number of nodes,
(2) the capacity constraint for each link,
(3) the cost for each link,
(4) the size of the member node sets,
(5) the shape of the given graph.

Hence, the following assumptions were made about this experiment to address these factors.

(1) The cost for each link was assigned to be the distance between two end nodes of that link.
(2) The capacity constraint for each link was assigned to be the link cost times a random number in $[2/n, 1]$, where n is the number of nodes.
(3) The member node sets are all equal (i.e., $D_1 = D_2 = \cdots = D_k = D$) and the number of multicast sessions is equal to the size of member node set (i.e., $k = |D|$).

In order to show that the solution found by STH is a good approximation, a comparison between the objective value X_{STH} found by STH and the optimal value X_{opt} (or lower bound $X_{\mathrm{LB}} = \sum_{m=1}^{k} \mathbf{c}(T^*_{D_m}(G_N, \mathbf{c}))$ of optimal value) is made. Note that the value of X_{opt} can be obtained by a simple exhaustive search. That is, we generate all possible connected multicast trees and check the feasibility of every possible combination of these trees, then pick up a combination with the least cost. In this way, the value of X_{opt} can be found. Similarly, the value of lower bound X_{LB} is also found by an exhaustive search.

Table 1 shows the gap ratios of the objective value X_{STH} to the optimal value X_{opt} and the objective value X_{greedy} found by the greedy algorithm to the optimal value X_{opt} for the graph with 10, 12 and 15 nodes, where the gap ratio of X_{STH} (X_{greedy}) to the optimal value X_{opt} is defined to be $(X_{\mathrm{STH}} - X_{\mathrm{opt}})/X_{\mathrm{opt}} \times 100\%$ ($(X_{\mathrm{greedy}} - X_{\mathrm{opt}})/X_{\mathrm{opt}} \times 100\%$). Fig. 12 shows the ratios of the objective value X_{STH} (and X_{greedy}) to the optimal value X_{opt} for the graphs with 15 nodes. Fig. 13 shows the ratios of X_{STH} to the lower bound X_{LB} and X_{greedy} to the lower bound X_{LB} for the graphs with 20

[2] The blocking rate is defined to be the probability that the algorithm fails to obtain a feasible solution, but the solution does exist.

Table 1
Gap ratio of optimal to size of member node set ($|D|$) for $|V| = 10$, 12 and 15

Size of V	Size of member node set D				
	Algorithms	2	3	4	5
10	greedy	$\ll 0.01\%$	0.31%	0.59%	0.76%
	STH	$\ll 0.01\%$	$\ll 0.01\%$	0.15%	0.18%
12	greedy	$\ll 0.01\%$	1.63%	2.47%	3.50%
	STH	$\ll 0.01\%$	0.14%	0.23%	0.35%
15	greedy	0.50%	2.20%	4.40%	7.60%
	STH	0.30%	0.80%	1.20%	3.80%

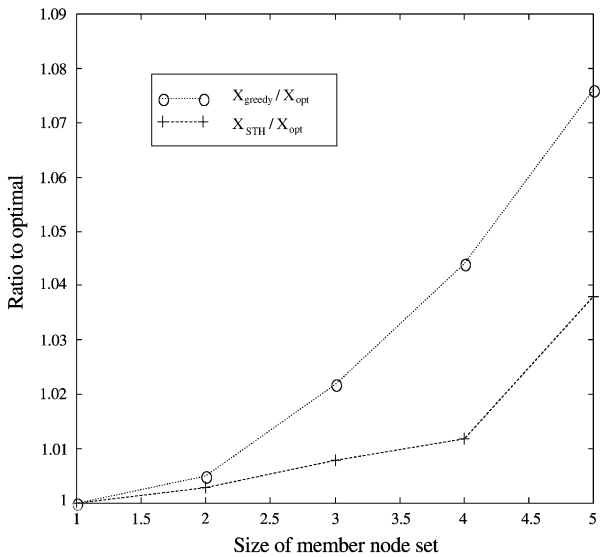

Fig. 12. Ratio of optimal to size of member node set ($|D|$) for $|V| = 15$.

nodes. From Table 1, Figs. 12 and 13 we learn that the objective value X_{STH} is close to the optimal value X_{opt} and X_{STH} is better than X_{greedy}.

For large-scale networks, the comparisons among X_{greedy}, X_{STH} and X_{CSH} are made (shown in Fig. 14) where X_{CSH} is the objective value found by CSH algorithm. The results show that the gaps among X_{greedy}, X_{CSH} and X_{STH} become larger as the number of nodes increase. The objective values X_{STH} are better than X_{greedy} or X_{CSH}.

Next, we investigate the average blocking rate of these algorithms. The blocking rate is defined to be the probability that the algorithm fails to find a feasible solution, but the solution exists. In order to ensure that all simulation instances have a feasible solution, we randomly generate $|D|$ multicast

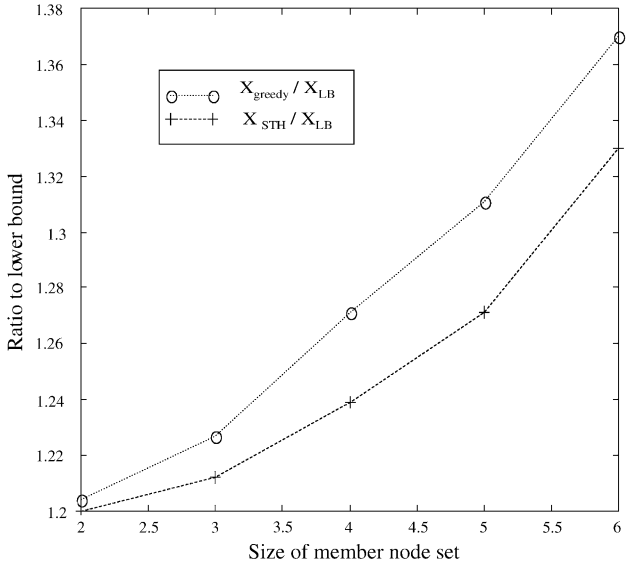

Fig. 13. Ratio of lower bound to size of member node set ($|D|$) for $|V| = 20$.

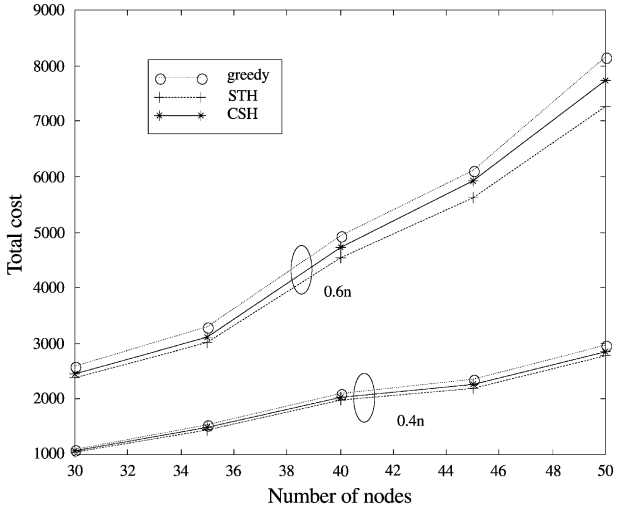

Fig. 14. Total cost vs. number of nodes (n) $|D| = 0.4n$ and $0.6n$.

trees in advance and pack these trees together to from a network. Then, we allocate additional bandwidth (say 20% or 30% of the total bandwidth) to the network. Apply the greedy, STH and CSH algorithms to these instances. The results are summarized in Figs. 15–17. From Figs. 15 and 16, CSH algorithm receives a very low blocking rate. Fig. 17 shows that CSH can always find a feasible solution if the extra bandwidth percentage is greater than 40%.

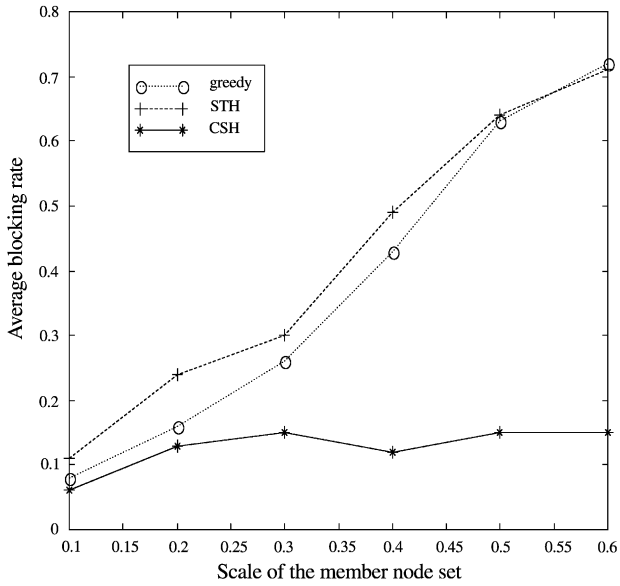

Fig. 15. Average blocking rate vs. scale of the member node set ($|D|/|V|$) for $|V| = 50$ with 20% extra bandwidth percentage.

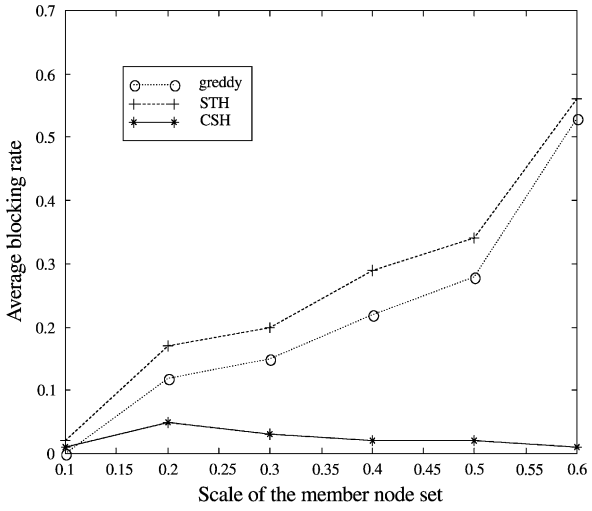

Fig. 16. Average blocking rate vs. scale of the member node set ($|D|/|V|$) for $|V| = 50$ with 30% extra bandwidth percentage.

Regarding the real execution time of these algorithms, all instances were run on a PC with Pentium II CPU of 366 MHz. Table 2 summarizes the average computation time (second) for greedy, STH, and CSH algorithms. Note that STH algorithm takes a reasonable computation time while CSH algorithm takes the longest computation time.

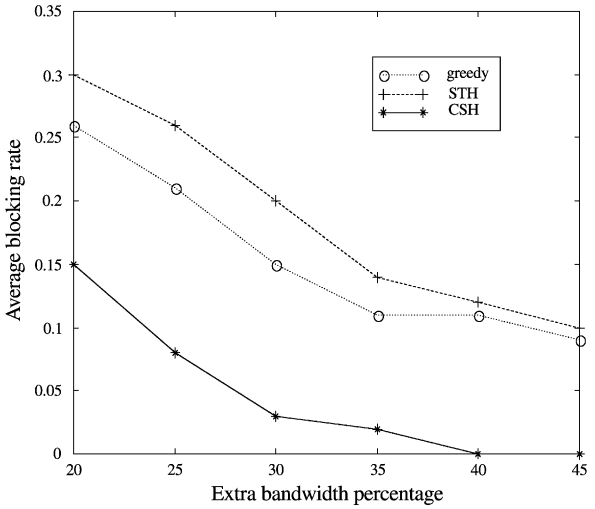

Fig. 17. Average blocking rate vs. extra bandwidth percentage for $|D| = 0.3n$ and $|V| = 50$.

Table 2
Comparison of greedy heuristic, STH and CSH on average computation time

Size of D	Number of nodes (n)						
	Algorithms	25	30	35	40	45	50
0.4n	greedy	0.1	0.2	0.4	0.6	1.0	1.6
	STH	1.0	2.9	6.9	14.8	29.8	51.3
	CSH	20.1	64.5	152.9	441.9	926.9	1941.0
0.6n	greedy	0.1	0.2	0.5	1.1	2.4	5.1
	STH	4.0	11.5	26.9	59.7	115.7	206.9
	CSH	49.9	163.7	418.8	1113.2	2285.8	4197.3

5. Concluding remarks

In this paper, we considered an optimal packing problem with multiple multicast sessions occurring simultaneously, which occurs frequently in a multimedia network. We proposed two multicast tree packing algorithms (STH and CSH) to find the approximate solution for this problem. In the experiment, it was shown that the STH performs well compared to the greedy method, and CSH. The computation time of STH is also reasonable. On the other hand, the CSH algorithm receives a lower blocking rate. Thus, it is suggested that one can apply the STH algorithm first to solve the tree packing problem. In case STH fails, CSH algorithm will be used instead. A possible future work is to design an algorithm with 0 blocking rate.

References

[1] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: W.H. Freeman and Company, 1979.
[2] Drefus SE, Wagner RA. The Steiner problem in graphs. Networks 1971;1:195–207.
[3] Levin AJ. Algorithm for the shortest connection of a group of graph vertices. Soviet Mathematics Dokl 1971;12:1477–81.
[4] Shore ML, Foulds LR, Gibbons PB. An algorithm for the Steiner problem in graphs. Networks 1982;12:323–33.
[5] Yang YY, Wing O. An algorithm for the wiring problem. Digest of the IEEE International Symposium on Electrical Networks, 1971. p. 14–5.
[6] Foulds LR, Gibbons PB. A branch and bound approach to the Steiner problem in graphs. Proceedings of the 14th Annual Conference of the Operational Research Society of New Zealand, vol. 1, 1978. p. 61–70.
[7] Hakimi SL. Steiner's problem in graphs and it's implications. Networks 1971;1:113–33.
[8] Balakrishnan A, Patel NR. Problem reduction methods and a tree generation algorithm for the Steiner network problem. Networks 1987;17:65–85.
[9] Wong RT. A dual ascent approach for Steiner tree problems on a directed graphs. Mathematical Programming 1984;28:271–87.
[10] Liu WG. A lower bound for the Steiner tree problem in directed graphs. Networks 1990;20:765–78.
[11] Beasley JE. An algorithm for the Steiner problem in graphs. Networks 1984;14:147–59.
[12] Dror M, Gavish B, Choquette J. Directed Steiner tree problem on a graph: models, relaxations and algorithms. INFOR 1990;28:266–81.
[13] Shaikh A, Lu S, Shin K. Localized multicast routing. Proceedings of Globecom'95, 1996. p. 1352–6.
[14] Takahashi H, Matsuyama A. An approximate solution for the Steiner problem in graphs. Mathematica Japonica 1980;24:573–7.
[15] Winter P, Smith JM. Path-distance heuristics for the Steiner problem in undirected networks. Algorithmica 1992;7:309–27.
[16] Mehlhorn K. A faster approximation algorithm for the Steiner problem in graphs. Information Processing Letters 1988;27:125–8.
[17] Rayward-Smith VJ, Clare A. On finding Steiner vertices. Networks 1986;16:283–94.
[18] Floren R. A note on a faster approximation algorithm for the Steiner problem in graphs. Information Processing Letters 1991;38:177–8.
[19] Hwang FK, Richards DS. Steiner tree problem. Networks 1992;22:55–89.
[20] Winter P. Steiner problem in networks: a survey. Networks 1987;17:129–67.
[21] Jia XH. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Transaction on Networking 1998;6:828–37.
[22] Zhu Q, Parsa M, Garcia-Luna-Aceves JJ. A source-based algorithm for delay-constrained minimum-cost multicasting. Proceedings of IEEE Infocom'95, 1995. p. 377–85.
[23] Chung CJ, Hong SP, Huh HS. A fast multicast routing algorithm for delay-sensitive applications, Proceedings of Globecom'97, 1997. p. 1898–902.
[24] Bauer F, Varma A. Degree-constrained multicasting in point-to-point networks. Proceedings of IEEE Infocom'95, 1995. p. 369–76.
[25] Wang CF, Lai BR, Jan RH. Optimum multicast of multimedia streams. Computer & Operations Research 1999;26:461–80.
[26] Jia XH, Wang LS. A group multicast routing algorithm by using multiple minimum Steiner trees. Computer Communications 1997;20:750–8.
[27] Priwan V, Aida H, Saito T. The multicast tree based routing for the complete broadcast multipoint-to-multipoint communications. IEICE Transactions of the Communications 1995;E78-B:720–8.
[28] Chen SW, Gunluk O, Yener, B. Optimal packing of group multicastings. Proceedings of IEEE Infocom'98, 1995. p. 369–76.
[29] Gomory RE, Hu TC. Multi-terminal network flows. SIAM Journal on Applied Mathematics 1961;9:551–6.
[30] Waxman B. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 1988;6:1617–22.

Chu-Fu Wang received the B.S. degree in Applied Mathematics from National Cheng Kong University, and the M.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan. Since 1995, he has been working toward the Ph.D. degree in Computer and Information Science at National Chiao Tung University, Taiwan. His research interests include multicast distribution, network reliability, and network optimization.

Chun-Teng Liang received the B.S. and M.S. degrees in Computer and Information Science from National Chiao Tung University, Taiwan. His research interests include computer networks and multicast distribution.

Rong-Hong Jan is a Professor in the Department of Computer and Information Science, National Chiao Tung University. He received the B.S. and M.S. degrees in Industrial Engineering, and the Ph.D. degree in Computer Science from National Tsing Hua University, Taiwan. He has been a Visiting Associate Professor in the Department of Computer Science, University of Maryland. His research interests include computer networks, distributed systems and network reliability.