Improved Approximation Algorithms for the Capacitated Multicast Routing Problem

Zhipeng Cai[1] *, Guohui Lin[1] **, and Guoliang Xue[2] ***

[1] Department of Computing Science, University of Alberta.
Edmonton, Alberta T6G 2E8, Canada.
[2] Department of Computer Science and Engineering, Arizona State University.
Tempe, Arizona 85287-5406, USA.

Abstract. For the Capacitated Multicast Routing Problem, we considered two models which are the Multicast k-Path Routing and the Multicast k-Tree Routing. We presented two improved approximation algorithms for them, which have worst case performance ratios of 3 and $(2+\rho)$ (ρ is the best approximation ratio for the Steiner Tree problem, and is about 1.55), respectively, thus improving upon the previous best ones having performance ratios of 4 and $(2.4+\rho)$, respectively. The designing techniques developed in the paper could be applicable to other similar networking problems.

Keywords: Capacitated Multicast Routing, Approximation Algorithm, Steiner Minimum Tree, Tree Partitioning.

1 Introduction

Multicast consists of concurrently sending the same information from a single source node to multiple destination nodes. Multicast service plays a more and more important role in computer or communication networks supporting multimedia applications [5,6,10]. It is well known that multicast can be easily implemented on local area networks (LANs) since nodes connected to a LAN usually communicate over a broadcast network. It is also known that implementing multicast in wide area networks (WANs) is quite challenging as nodes connected to a WAN communicate via a switched/routed network [2,11].

In order to perform multicast communication in WANs, the source node and all destination nodes must be interconnected by a tree. The problem of multicast routing in WANs is thus treated as finding a multicast tree in a network that spans the source and all destination nodes. Its goal is to minimize the network cost of the multicast tree, which is usually defined as the sum of the weights of all edges in the tree.

* Supported in part by NSERC. Email: zhipeng@cs.ualberta.ca.
** To whom correspondence should be addressed. Supported by NSERC, CFI, and a Startup Grant from the University of Alberta. Email: ghlin@cs.ualberta.ca.
*** Supported in part by ARO grant DAAD19-00-1-0377 and DOE grant DE-FG02-00ER45828. Email: xue@asu.edu.

In this paper, we study the *Capacitated Multicast Routing Problem* in which at a time the data is sent out of the course, at most a limited amount of destination nodes can be assigned to receive copies. Depending on whether or not the switches or routers in the underlying network have the broadcasting ability, two models were considered. In one model, switches are assumed to have the broadcasting ability and the model is called the *multi-tree model* [4]. Multi-tree model has its origin in WDM optical networks with limited light-splitting capabilities. Under this model, we were interested in finding a set of trees such that each tree spans the source node and a limited number of destination nodes which are assigned to receive data and every destination node must be designated to receive data in one of the trees. Compared with the traditional incapacitated multicast routing model — called the *Steiner Tree* problem, that doesn't have any constraint on the number of receivers in the routing tree, this simpler model makes multicast easier and more efficient to implement, at the expense of increasing the routing tree cost. Specifically, when the number of destination nodes in a tree is limited to k, we call it the *Multicast k-Tree Routing* (kMTR) problem, which is formally defined in the following.

We model the underlying communication network using an edge-weighted (simple, undirected) graph $G(s, V, D, E)$, where s is the source node, D is the set of n destination nodes, and V is the set of all nodes in the network: $s \cup D \subseteq V$. E is the set of edges (representing direct links) and $w(e) \geq 0$ is the weight (routing cost) of edge $e \in E$. The additive edge weight function $w(\cdot)$ generalizes to subgraphs of G in the natural way. Let T be a subgraph of G, the weight (or cost) of T, denoted by $w(T)$, is the sum of the edge weights among all edges in T. Let k be a given positive integer. The multicast k-tree routing (kMTR) problem asks for a partition of D into disjoint sets $D_1, D_2, \ldots, D_\ell$ each with cardinality no more than k and a Steiner tree T_i spanning the source node s and the nodes in D_i for $i = 1, 2, \ldots, \ell$, so that $\sum_{i=1}^{\ell} w(T_i)$ is minimized.

In the other model, switches have no broadcasting ability and the model is called the *multi-path routing model* [3,4]. Multi-path model could be viewed as a generalization of one-to-one connection, but a restricted version of the multi-tree routing model. It was proposed for wavelength routed optical networks, although the basic idea is also applicable for general packet switching network [9]. Under the multi-path model, data is sent from the source node to a destination node in a light path. During the data transmission along the path, if an intermediate node itself is a destination node, then the data is stored (dropped) and a copy of the data is forwarded to its adjacent neighbor in the path. In each path some destination nodes are designated where the data is stored (dropped). Accordingly, *multi-path routing* is to find a set of such paths so that every destination node is designated to received the data in a path. Compared with the multi-tree routing model, this even simpler model makes multicast even easier and even more efficient to implement, but again at the expense of increasing the routing tree cost. The parametric variant considered in [3] is the *Multicast k-Path Routing* (kMPR) problem, where every path can be designated with at most k destination nodes.

1.1 State-of-the-Art

For the kMTR problem, the cases where $k = 1, 2$ reduce to the kMPR problem [3]. So they both can be solved efficiently. The general case of kMTR is more difficult than the kMPR problem and thus is NP-hard [3,2] too. In fact, 3MTR has been shown to be NP-hard [7]. The best known approximation algorithm for kMPR has a worst case performance ratio of 4 [3]; The best known approximation algorithm for kMTR has a worst case performance ratio of $(2.4 + \rho)$ [7], where ρ is the approximation ratio for the Steiner Tree problem.

1.2 Our Contributions

We developed an averaging technique for designing better approximation algorithms for both kMPR and kMTR problems for $k \geq 3$. The averaging technique is presented in the next section. Basing on it, we gave a 3-approximation algorithm for kMPR. We presented a technique for partitioning trees in Section 3, and we applied the averaging technique one step further together with this tree partitioning technique to design a $(2 + \rho)$-approximation algorithm for kMTR. It is worth mentioning that ρ is the best approximation ratio for the Steiner Tree problem, and ρ is about 1.55 [8,1] at the writing of this paper. In more detail, we first applied the current best approximation algorithm for the Steiner Tree problem. Then we partitioned the obtained Steiner tree to get a number of subtrees each spanning at most k destination nodes, without increasing the total cost. After proving that the sum of the shortest paths from source s to all of the subtrees is smaller than twice of the cost of an optimal multicast k-routing tree, we got a $(2 + \rho)$-approximation algorithm for kMTR, which improves the previous best approximation ratio $(2.4 + \rho)$.

2 A 3-Approximation Algorithm for kMPR

In the kMPR problem, the underlying communication network can be simplified by short-cutting non-destination nodes out and thus can be assumed as an edge-weighted complete graph $G(s, D)$ where s is the source node and $D = \{d_1, d_2, \ldots, d_n\}$ is the destination node set. The edge weight function naturally satisfies the triangle inequality. The goal is to find a least cost *k-path routing*, which is a set of paths rooting at s and spanning all destination nodes, and every path contains at most k destination nodes.

Let $P_1^*, P_2^*, \ldots, P_m^*$ be the set of paths in an optimal k-path routing. Let $c(P_i^*)$ denote the cost of path P_i^*, which is the sum of the weights of edges on the path. Let $R^* = \sum_{i=1}^{m} c(P_i^*)$ be the cost of the routing tree.

The 4-approximation algorithm provided in [3] is to construct a minimum spanning tree T on $s \cup D$, then duplicate the edges in T to produce a Hamiltonian cycle C via suitable short-cutting, and then partition the cycle C into segments each containing exactly k distinct destinations (the last segment might contain less than k distinct destinations). Every segment is connected to the source s

via a shortest path from s. Since the cost of a minimum spanning tree T is at most R^* (note that $P_1^*, P_2^*, \ldots, P_{m-1}^*$ and P_m^* themselves form a spanning tree), the cost the cycle C is no more than $2R^*$. It is shown that the total cost of shortest-paths added in order to connect segments to the source s is at most R^*. However, since for every segment the shortest path connecting from the source s to it could destinate at an internal node on the path, in order to produce feasible routings the algorithm uses two copies of the added path to make two paths. Therefore, the cost of the resultant k-path routing could be as large as $4R^*$. In fact, the following example shows that the ratio 4 is asymptotically tight. In this example, the optimal k-path routing is $P_1^*, P_2^*, \ldots, P_m^*$, where $P_1^* = s\text{-}d_{mk-1}\text{-}d_{mk}\text{-}d_1\text{-}\ldots\text{-}d_{k-2}$, $P_2^* = s\text{-}d_{k-1}\text{-}d_k\text{-}d_{k+1}\text{-}\ldots\text{-}d_{2k-2}$, $\ldots$, $P_m^* = s\text{-}d_{(m-1)k-1}\text{-}d_{(m-1)k}\text{-}d_{(m-1)k+1}\text{-}\ldots\text{-}d_{mk-2}$. The weights are $w(s, d_{ik-1}) = M$ for $i = 1, 2, \ldots, m$, and $w(d_j, d_{j+1}) = 1$ when $j \neq ik - 2$ for some i. The underlying communication network is the completion of this tree. Note that the cost of the optimal k-path routing is $R^* = m(M + k - 1)$. The minimum spanning tree has a cost the same as the optimal routing, and the cost of the Hamiltonian cycle is exactly twice R^*. According to the partitioning, $d_1, d_2, \ldots, d_k$ are on a same segment and d_{k-1} is the closest to source s. Therefore, the final k-path routing has a cost $m(4M + 2k - 3)$, which is asymptotically 4 times R^*.

In the following we propose another way to partition the obtained Hamiltonian cycle into segments each containing exactly k distinct destinations (again, the last segment might contain less than k distinct destinations), by which the added paths connecting them to the source can be made to be from one of the ending destination nodes on the segments and the total length of these added paths is also no more than R^*.

Observe that in $P_1^*, P_2^*, \ldots, P_m^*$, the distance from every destination node d_i to the source s is an upper bound on the weight of edge (s, d_i) in the underlying network G (recall that we assume the shortest-path distance weight). Suppose the destination nodes are $d_1, d_2, \ldots, d_n$. It follows that

$$\sum_{i=1}^{n} w(s, d_i) \leq k \times R^*,$$

since there are at most k destination nodes in every path P_j^* for $j = 1, 2, \ldots, m$. Suppose the destination nodes on the obtained Hamiltonian cycle are indexed consecutively from 1 to n (with source s lying in between d_1 and d_n). Partition the term $\sum_{i=1}^{n} w(s, d_i)$ into k sub-terms:

$$\sum_{i=0}^{\lfloor \frac{n}{k} \rfloor} w(s, d_{ik+j}),\ j = 1, 2, \ldots, k.$$

(Note: when index is out of range, there is no such destination node.) It follows that there is at least one index j^* such that

$$\sum_{i=0}^{\lfloor \frac{n}{k} \rfloor} w(s, d_{ik+j^*}) \leq R^*.$$

Now partition the Hamiltonian cycle into segments of which the first one contains destination nodes d_{j^*}, d_{j^*+1}, d_{j^*+2}, ..., d_{j^*+k-1}, the second one contains destination nodes d_{j^*+k}, d_{j^*+k+1}, d_{j^*+k+2}, ..., d_{j^*+2k-1}, ..., and so on. For the ith segment, the path used to connect it to the source s is the edge $(s, d_{j^*+(i-1)k})$. It is clear that every segment appended with the connecting edge is still a path and thus they form a feasible routing. Note that the cost of the segments is no more than $2R^*$ and the cost of added edges/paths is no more than R^*. Therefore, the cost of this routing has cost no more than $3R^*$.

INPUT: an edge-weighted graph G on $s \cup D$;
OUTPUT: a k-path routing.

1. Compute a minimum spanning tree T_0 on $s \cup D$;
2. Double edges in T_0 and short-cut a Hamiltonian cycle C on $s \cup D$;
3. Suppose the order of nodes on C is s-d_1-d_2-...-d_n:
   $j^* = \arg\min_j \sum_{i=0}^{\lfloor \frac{n}{k} \rfloor} w(s, d_{ik+j})$;
4. Make the ith path d_{ik+j^*}-d_{ik+j^*+1}-d_{ik+j^*+2}-...-d_{ik+j^*+k-1};
5. Connect source node s to d_{ik+j^*} for every i;
6. Output the k-path routing.

Fig. 1. A high-level description of the 3-approximation algorithm for kMPR, where $k \geq 3$.

Theorem 1. *The kMPR ($k \geq 3$) problem admits a 3-approximation algorithm which runs in $O(|D|^3)$ time.*

Proof. The algorithm presented in the above, and its high-level description in Figure 1, is an approximation algorithm for the kMPR problem and its is worst case performance ratio is 3. Note that completing the graph might take $O(|D|^3)$ time. After that, computing a minimum spanning tree can be done in $O(|D|^2 \log |D|)$ time and forming the Hamiltonian cycle in $O(|D|^2)$ time. It takes $O(|D|)$ time to compute the partition, or equivalently, the optimal index j^*. Therefore, the overall running time is in $O(|D|^3)$. □

3 A $(2+\rho)$-Approximation Algorithm for kMTR

In the kMTR problem, the underlying communication network is an edge-weighted complete graph $G(s, V, D)$ where s is the source node, $D = \{d_1, d_2, \ldots, d_n\}$ is the destination node set, and V is a superset of D containing also Steiner nodes which can be used as intermediate nodes to save the routing cost. The edge weight function satisfies the triangle inequality. The goal is to find a least cost *k-tree routing*, which is a set of Steiner trees rooting at s and spanning all destination nodes, and every tree contains at most k destination nodes. Note that

in a feasible k-tree routing, one destination node assigned in some tree can be used as a Steiner node in the others.

Let $T_1^*, T_2^*, \ldots, T_m^*$ be the set of trees in an optimal k-tree routing. Recall that every T_i^* might contain some Steiner nodes and might also contain some destination nodes which are *not* allowed to received data (but act as Steiner nodes).

Let $c(T_i^*)$ denote the cost of tree T_i^*, which is defined to be the sum of the weights of edges in the tree. Let $R^* = \sum_{i=1}^{m} c(T_i^*)$ be the cost of the routing tree. Since every destination node d_i in tree T_j^* satisfies $w(s, d_i) \leq c(T_j^*)$, we have

$$\sum_{i=1}^{n} w(s, d_i) \leq k \times R^*.$$

In the $(2+\rho)$-approximation algorithm, we firstly apply the currently best approximation algorithm for the Steiner Tree problem (which has the worst-case performance ratio ρ) to obtain a Steiner tree T on $s \cup D$ in the underlying network G. Since the cost of an optimal Steiner tree is a lower bound on R^*, we know that the cost of tree T is upper bounded by ρR^*, that is, $c(T) \leq \rho R^*$. Note that tree T is not necessarily a feasible routing tree yet since some branch rooted at the source s might contain more than k destination nodes. We treat T in the following way: if there is any branch of it which contains no more than k destination nodes, we can just leave the branch alone in the next step. For branches containing more than k destination nodes, we perform the following partition on each of them in the next step.

To present the partition technique, we need the following lemmas, the first two of which are from [7].

Lemma 1. [7] *Given a tree T containing $n \geq 3$ nodes, it is always possible to partition it into two subtrees which overlap at at most one node and the numbers of nodes in both subtrees fall in the closed interval $[\frac{1}{3}n, \frac{2}{3}n]$.*

Lemma 2. [7] *Given a Steiner tree T containing $n \geq 3$ destination nodes, it is always possible to partition it into two subtrees which overlap at at most one node, either Steiner or destination, and the numbers of destination nodes in both subtrees fall in the closed interval $[\frac{1}{3}n, \frac{2}{3}n]$.*

Lemma 3. *Given a Steiner tree T containing n destination nodes, where $k < n \leq \frac{3}{2}k$ and $k \geq 3$, randomly select $n - \frac{1}{2}k + 1$ destination nodes from the tree to form a set D_0. Then, it is always possible to partition the tree into two subtrees T_1 with destination node set D_1 and T_2 with destination node set D_2 which overlap at at most one node (either destination or Steiner), $0 < |D_1|, |D_2| \leq k$, $D_1 \cap D_0 \neq \emptyset$, and $D_2 \cap D_0 \neq \emptyset$.*

Proof. Root tree T at any node, which could be either destination or Steiner. In this rooted tree, for every node v, let $c(v)$ denote the number of destination nodes in the subtree rooted at v (inclusive). Let r denote the farthest (from the root) node which has $c(r) > n - \frac{1}{2}k$. Note that in the case that there is no node

having the c-value greater than $n - \frac{1}{2}k$, r is set to be the root. Since $k < n \leq \frac{3}{2}k$, r is uniquely defined. Re-root tree T at node r.

By duplicating root node r, we can partition T into two subtrees (both rooted at r) T_1 with destination node set D_1 and T_2 with destination node set D_2. Our partition goal is to minimize $|D_2| - |D_1|$, assuming without loss of generality that $|D_2| \geq |D_1|$. If it already holds $0 < |D_1|, |D_2| \leq k$, $D_1 \cap D_0 \neq \emptyset$, and $D_2 \cap D_0 \neq \emptyset$, then we got the two desired subtrees. Otherwise there must be $|D_1| < \frac{1}{2}k$ and $|D_2| > n - \frac{1}{2}k$. We proceed to examine subtree T_2, which must have multiple branches and each of them contains at most $n - \frac{1}{2}k$ destination nodes.

Number these branches as T_{21}, T_{22}, ..., $T_{2\ell}$, with the destination node sets D_{21}, D_{22}, ..., $D_{2\ell}$, respectively. We distinguish two cases. In the first case there is a branch say T_{2i} such that $|D_{2i}| \geq \frac{1}{2}k$. It follows from $|D_{2i}| \leq n - \frac{1}{2}k \leq k$ that re-partitioning T to have only T_{2i} in subtree T_2, while all the other branches rooted at r are included into subtree T_1, gives the desired partition. That is, $0 < |D_1|, |D_2| \leq k$, $D_1 \cap D_0 \neq \emptyset$, and $D_2 \cap D_0 \neq \emptyset$. In the other case, every branch contains less than $\frac{1}{2}k$ destination nodes: $|D_{2i}| < \frac{1}{2}k$, for $i = 1, 2, \ldots, \ell$. Since $|D_0| = n - \frac{1}{2}k + 1 > \frac{1}{2}k + 1$, there are at least two branches, say T_{21} and T_{22}, both contain destination nodes from D_0 (which is not the root node r). Again, we do the re-partitioning by removing T_{21} from T_2 while including it into T_1. This gives us a new pair of subtrees T_1 and T_2 that satisfy $0 < |D_1|, |D_2| \leq k$, $D_1 \cap D_0 \neq \emptyset$, and $D_2 \cap D_0 \neq \emptyset$.

This proves the Lemma. □

Recall that every branch of T rooted at the source s is ignored for further consideration. In the following, we will focus on the operations performed on one branch of T (rooted at the source s) containing more than k destination nodes. First of all, we delete the edge incident at s from the branch to get a subtree denoted as T_1. Secondly, if T_1 contains more than $\frac{3}{2}k$ destination nodes, we apply Lemma 2 to partition T_1 into two subtrees. We then repeatedly apply Lemma 2 to partition the resultant subtrees if they contain more than $\frac{3}{2}k$ destination nodes. At the end of this repeatedly partition, there will be a set of subtrees each contains no more than $\frac{3}{2}k$ destination nodes. It should be noted that each of them contains at least $\frac{1}{2}k$ destination nodes since we started with T_1 which contains more than $\frac{3}{2}k$ destination nodes. At this point, for those subtrees which contain no more than k destination nodes, we may leave them alone. For ease of presentation, we call subtrees containing at most k destination nodes *final trees*. The subtrees become final at this point are called *type-1 final trees*. The non-final subtrees will become *type-2 final trees* after the next step of partitioning.

For each non-final-yet subtree again denoted by T_1, our third step is to apply Lemma 3 on it to partition it into two final subtrees. To this purpose, we let D_0 denote the set of closest $n - \frac{1}{2}k + 1$ (to the source s) destination nodes in T_1, where n is the total number of destination nodes in T_1. Let T_{11} and T_{12} denote the two resultant subtrees having destination node sets D_1 and D_2, respectively. By Lemma 3, $0 < |D_1| \leq k$, $D_1 \cap D_0 \neq \emptyset$, $0 < |D_2| \leq k$, and $D_2 \cap D_0 \neq \emptyset$. It is

clear that type-2 final trees always come in a pair, since they are resulted from one single partition by Lemma 3.

For each final tree, we pick the closest destination node therein and connect it to the source s. This gives a feasible k-tree routing. In what follows, we will estimate the total cost of these added edges. We will show that this total is at most twice of R^*.

First of all, for every type-1 final tree, we pick the $\frac{1}{2}k$ closest destination nodes therein to be the representatives for the tree. Suppose there are ℓ_1 type-1 final trees T_1, T_2, ..., T_{ℓ_1}. Let the representatives for T_i be $d_{i,1}$, $d_{i,2}$, ..., $d_{i,\frac{k}{2}}$, in the order of non-decreasing distance from the source s. Secondly, for every pair of type-2 final trees T_1 and T_2, if any one of them contains no less than $\frac{1}{2}k$ destination nodes, then the $\frac{1}{2}k$ closest ones are picked to be the representatives for the tree; otherwise all the destination nodes, say m, are picked to be the representatives and additionally the $\frac{1}{2}k-m$ farthest (to the source s) destination nodes in the other tree are picked to be the representatives. Therefore, every type-2 final tree has exactly $\frac{1}{2}k$ representatives, although some of them might not come from its own but its partner. Note that the reason we can do this is that the total number of destination nodes in this pair of type-2 final trees is greater than k. Similarly, assume that there are ℓ_2 pairs of type-2 final trees T_{11}, T_{12}, T_{21}, T_{22}, ..., $T_{\ell_2 1}$, $T_{\ell_2 2}$. Let the representatives for T_{ih} be $d_{ih,1}$, $d_{ih,2}$, ..., $d_{ih,\frac{k}{2}}$, where h is either 1 or 2, in the order of non-decreasing distance from the source s. Also for every pair of trees T_{i1} and T_{i2}, let $d^0_{i,1}$, $d^0_{i,2}$, ..., $d^0_{i,\frac{k}{2}}$ be the $\frac{1}{2}k$ closest destination nodes among all the destination nodes in both of them, and let $d^0_{i,\frac{k}{2}+1}$, $d^0_{i,\frac{k}{2}+2}$, ..., $d^0_{i,k}$ be the $\frac{1}{2}k$ farthest destination nodes among all the destination nodes in both of them.

It follows that

$$\sum_{i=1}^{\ell_1}\sum_{j=1}^{\frac{k}{2}} w(s,d_{i,j}) + \sum_{i=1}^{\ell_2}\sum_{j=1}^{k} w(s,d^0_{i,j}) \le \sum_{i=1}^{n} w(s,d_i) \le k \times R^*.$$

Using the non-increasing distance orderings of these destination nodes, we have

$$\sum_{i=1}^{\ell_1} w(s,d_{i,1}) + \sum_{i=1}^{\ell_2} \Big(w(s,d^0_{i,1}) + w(s,d^0_{i,\frac{k}{2}+1}) \Big) \le 2R^*.$$

Clearly, for every type-1 final tree T_i, destination node $d_{i,1}$ is connected to the source s; also is true that $d^0_{i,1}$ must serve as a representative for either type-2 final tree T_{i1} or type-2 final tree T_{i2} and thus it is connected to the source s. Suppose without loss of generality that $d^0_{i,1}$ is a representative for T_{i1}, then the closest destination node d in T_{i2} which is picked to be a representative has a distance no larger than the distance from the source to destination node $d^0_{i,\frac{k}{2}+1}$. It follows that the total cost of the edges added to connect the source to the final trees to produce a feasible k-tree routing is at most $2R^*$. Therefore, the thus produced routing tree has a cost no more than $(2+\rho)R^*$.

INPUT: an edge-weighted graph $G(s, V, D)$;
OUTPUT: a k-tree routing.

1. Compute a Steiner tree T_0 on $s \cup D$, using the currently best approximation;
2. Delete the edges incident at s to get subtrees of T_0;
3. For each subtree T_1, and the resultant, containing $k' > k$ destinations:
3.1 if $k' > \frac{3}{2}k$, apply Lemma 2 to partition T_1;
3.2 if $k' \leq \frac{3}{2}k$, apply Lemma 3 to partition T_1;
4. Pick for every final tree the closest destination therein and connect it to s;
5. Output the k-tree routing.

Fig. 2. A high-level description of the $(2 + \rho)$-approximation algorithm for kMTR, where $k \geq 3$.

Theorem 2. kMTR ($k \geq 3$) *admits a* $(2 + \rho)$*-approximation algorithm, where* ρ *is the best performance ratio for approximating the* Steiner Tree *problem.*

Proof. The theorem holds according to the above discussion, since the algorithm presented in the above, and its high-level description in Figure 2, is a $(2 + \rho)$-approximation for the kMTR problem. It improves the previous best approximation algorithm which has a performance guarantee of $(2.4 + \rho)$ [7]. It is known that ρ is about 1.55 [8,1]. Therefore, our approximation algorithm has a performance ratio about 3.55, while the previous best one has a performance ratio about 3.95. It is worth mentioning that the running time is dominated by the approximation algorithm for the Steiner Tree problem. □

4 Conclusions

We have proposed a weight averaging technique which gives a better way to estimate the cost of the paths connecting to the source and thus guarantees a 3-approximation algorithm for the kMPR problem. We have also designed a better approximation algorithm for the kMTR problem, with the worst case performance ratio $(2 + \rho)$, via another design technique — tree partitioning — which is very interesting on its own. We hope these two design techniques can be further combined with the others developed in the literature leading to even better approximations.

References

1. C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Approximation algorithms for the Steiner tree problem in graphs. In D.-Z. Du and X. Cheng, editors, *Steiner Trees in Industries*, pages 235–279. Kluwer Academic Publishers, 2001.
2. J. Gu, X. D. Hu, X. Jia, and M. H. Zhang. Routing algorithm for multicast under multi-tree model in optical networks. *Theoretical Computer Science*, 314:293–301, 2004.

3. J. Gu, X. D. Hu, and M. H. Zhang. Algorithms for multicast connection under multi-path routing model. *Information Processing Letters*, 84:31–39, 2002.
4. R. L. Hadas. Efficient collective communication in WDM networks. In *Proceedings of IEEE ICCCN*, pages 612–616, 2000.
5. C. Huitema. *Routing in the Internet*. Prentice Hall PTR, 2000.
6. F. Kuo, W. Effelsberg, and J. J. Garcia-Luna-Aceves. *Multimedia Communications: Protocols and Applications*. Prentice Hall, Inc., 1998.
7. G.-H. Lin. An improved approximation algorithm for multicast k-tree routing. *Journal of Combinatorial Optimization*, 2004. Submitted.
8. G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the* 11*th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, pages 770–779, 2000.
9. A. S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ, 1996.
10. Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14:1228–1234, 1996.
11. X. Zhang, J. Wei, and C. Qiao. Constrained multicast routing in WDM networks with sparse light splitting. In *IEEE INFOCOM*, pages 1781–1790, March 26–30 2000.