Primal-Dual Algorithms for QoS Multimedia Multicast

G. Calinescu* C.G. Fernandes† I.I. Mandoiu‡ A. Olshevsky§ K. Yang¶ A. Zelikovsky¶

***Abstract*— The QoS Steiner Tree Problem asks for the most cost-efficient way to multicast multimedia to a heterogeneous collection of users with different consumption rates. We assume that the cost of using a link is not constant but rather depends on the maximum bandwidth routed through the link. Formally, given a graph with costs on the edges, a source node and a set of terminal nodes, each one with a bandwidth requirement, the goal is to find a Steiner tree containing the source, and the cheapest assignment of bandwidth to each of its edges so that each source-to-terminal path in the tree has bandwidth at least as large as the bandwidth required by the terminal. Our main contributions are: (1) new covering-type integer linear program formulations for the problem; (2) two new heuristics based on the primal-dual framework; (3) a primal-dual constant-factor approximation algorithm; (4) an extensive experimental study of the new heuristics and of several previously proposed algorithms.**

I. INTRODUCTION

Recent progress in audio, video, and data storage techniques has given rise to a host of high-bandwidth real-time applications such as video conferencing. Quality of Service (QoS) requirements for these applications must be provided by the underlying networks. In light of this, multicast routing algorithms which manage network resources efficiently and satisfy the QoS requirements have come under increased scrutiny in recent years. As applications such as video conferencing gain popularity, the focus on multimedia data transfer capability in networks will grow [12].

Multimedia distribution is usually done via multicast trees. There are two reasons for basing efficient multicast routes on trees: the data can be transmitted concurrently to destinations along the branches of the tree and only a minimum number of copies of the data is transmitted since information needs to be replicated only at forks in the tree [14]. The bandwidth savings obtained from the use of multicast trees can be maximized by using optimal or nearly optimal tree algorithms. Future networks will no doubt integrate such algorithms into basic operational performance [3].

Several versions of the QoS multicast problem have been studied in the literature. These versions seek routing tree cost minimization subject to (1) end-to-end delay, (2) delay variation, and/or (3) minimum bandwidth constraints (see, e.g., [3], [11], [7]). In this paper, we consider the case of minimum bandwidth constraints, that is, the problem of finding an optimal multicast tree when each terminal possesses a different rate of receiving information. This problem is a generalization of the classical Steiner tree problem and known to be NP-hard [4]. Formally, given a graph $G = (V, E)$, a source s, a set of terminals S, and two functions: *length* : $E \to \mathbb{R}^+$ representing the length of each edge and *rate* : $S \to \mathbb{R}^+$ representing the rate of each node, a *multicast tree* T is a tree in G spanning s and S. The *rate* $\mathit{rate}(e, T)$ of an edge e in a multicast tree T is the maximum rate of a terminal in the connected component of $T - e$ which does not contain s. The *cost* of a multicast tree T is defined as

$$\mathit{cost}(T) = \sum_{e \in T} \mathit{length}(e) \cdot \mathit{rate}(e).$$

QUALITY OF SERVICE STEINER TREE PROBLEM (QOSST): Given a network $G = (V, E, \mathit{length}, \mathit{rate})$ with a source s in V and a set of terminals $S \subseteq V$, find a minimum cost multicast tree in G.

This formulation assumes that each link in the network has maximum possible rate, that is, has the maximum possible bandwidth. This is somewhat unrealistic due to different cables and traffic influence. A more sophisticated version of this problem would include a maximum possible rate function *maxrate* : $E \to \mathbb{R}^+$. Moreover, in practice the rates are rarely constant; another possible generalization would consider the case of dynamic rates.

The rest of the paper is organized as follows. In the next section we give a short summary of the algorithms proposed by [7], [4], and [6] and show that the approximation ratio of the algorithm from [7] is unbounded. In Section III, we consider an integer linear program formulation (ILP) and describe two heuristics based on the primal-dual framework. Then we prove that a primal-dual algorithm based on an enhanced ILP has an approximation ratio 4.311. Finally, in Section IV we conclude with an experimental comparison of our two primal-dual heuristics with algorithms from [7], [4].

*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: calinesc@cs.iit.edu.

†Department of Computer Science, University of São Paulo, Rua do Matão, 1010, 05508-900 São Paulo, Brazil. E-mail: cris@ime.usp.br.

‡ Department of Computer Science & Engineering, UC San Diego, La Jolla, CA 92093. E-mail: mandoiu@cs.ucsd.edu.

§ Department of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA 30332. E-mail: alex.olshevsky@resnet.gatech.edu.

¶ Department of Computer Science, Georgia State University, Atlanta, GA 30303. E-mail: {kenan,alexz}@cs.gsu.edu.

Input: A graph $G = (V, E, \mathit{length}, \mathit{rate})$ with a source s in V and a collection of terminals $S \subseteq V$.
Output: A QoS Steiner tree spanning the source and the terminals.

(1) Initialize the current tree to $\{s\}$.
(2) Find a non-reached terminal t of highest rate with the shortest distance to the current tree.
(3) Add t to the current tree along with a shortest path connecting it to the current tree.
(4) Repeat until all terminals are spanned.

Fig. 1. Maxemchuk's Algorithm for the QoS Steiner Tree Problem.

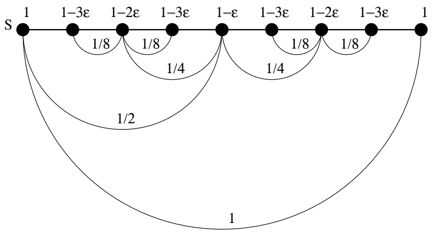

Fig. 2. A bad example for Maxemchuk's algorithm, with $k = 4$ rates. In the figure, $\varepsilon = 1/2^{2k-1}$. The rate of each node is given above the node. The edge lengths are given on the thin curved arcs, while on the solid horizontal line each segment has length $1/2^{k-1} + \varepsilon$. The optimum, of total cost $1 + 2^{k-1}\varepsilon = 1 + 2^{k-1}(1/2^{2k-1}) = 1 + 1/2^k$, uses the solid horizontal line at rate 1. Maxemchuk's algorithm picks the thin curved arcs at a cost of $1 + (1/2)(1-\varepsilon) + 2(1/4)(1-2\varepsilon) + 4(1/8)(1-3\varepsilon) \geq ((k+1)/2)(1 - 1/2^k)$.

II. Previous Work

A. Maxemchuk's Approach

Maxemchuk [7] proposed a heuristic algorithm for the QoS Steiner Tree Problem. His algorithm is a modification of the MST heuristic for Steiner Trees [13] (see Figure 1).

The extensive experiments given in [7] demonstrate that this method works well in practice. Nevertheless, the following example shows that the method may produce arbitrarily large error (linear in the number of rates) compared with the optimal tree. Consider the natural generalization of the example in Figure 2 with an arbitrary number k of distinct rates. Its optimal solution has a cost of about 1, whereas Maxemchuk's method returns a solution of cost about $(k+1)/2$. As there are $2^{k-1} + 1$ nodes, this cost can also be written as $1 + \frac{1}{2}\log_2(n-1)$, where n is the number of nodes in the graph. We conclude that the approximation ratio of Maxemchuk's algorithm is at least linear in the number of rates and at least logarithmic in the number of nodes in the graph.

B. The Charikar-Naor-Schieber Algorithm

The Charikar-Naor-Schieber algorithm [4] is the first constant-factor approximation algorithm given for the QoS Steiner tree problem. In its first step, all rates are rounded to the closest power of two to produce the rounded up instance of this problem (clearly, this at most doubles the cost of an optimal solution). In its second step, Steiner trees are computed separately for each rate (within some approximation ratio α). The union of these trees is the final solution.

Replace each edge of rate 2^i by edges of the rate $2^0, 2^1, \ldots, 2^{i-1}, 2^i$, respectively. In the new network, all edges of a specific rate form a Steiner tree. Since the optimal cost in this new network is no more than twice the cost of the rounded up instance, taking the union of all the computed Steiner trees introduces another factor of two to the approximation ratio. Thus the final approximation factor is $2 \cdot \alpha \cdot 2 = 4\alpha$.

Using a randomization technique, Charikar, Naor, and Schieber [4] reduce the approximation ratio to $e\alpha \approx 4.21$, where $e \approx 2.71$ is the Euler constant and $\alpha \approx 1.55$ is the currently best approximation ratio for the Steiner Tree problem. The approximation factor has been further improved to 3.802 by Karpinski et al. [6].

C. Algorithms for Two or Three Rates

In practice, it is often the case that only few distinct rates are requested by the terminals. This is why the QoS problem with two or three rates has a long history, having been considered in [1], [2], [8], and [15]. The previously-best results of [8] and [15] have produced algorithms with approximation factor equal to 2.667 (provided that the MST heuristic is used to compute Steiner trees). Karpinski et al. [6] improved this ratio to 2.414 and showed that it can be further improved to 1.96 if more sophisticated time consuming Steiner tree algorithms are used.

III. Primal-Dual Motivated Algorithms

The problem can be formulated as an integer program as follows. Consider a network $G = (V, E, \mathit{length}, \mathit{rate})$ with a source node s and a set of terminal nodes. Let $r_1 < r_2 < \cdots < r_k$ be all rate values assigned to the terminals. It simplifies notation to assume that every node has a rate by considering an extra rate $r_0 = 0$ (assign rate r_0 to each non-terminal node). Also, we may assume that s has the highest rate. Construct a new network $G' = (V, E', \mathit{cost}, \mathit{rate})$ by replacing each edge e of G with k edges $(e, r_1), (e, r_2), \ldots, (e, r_k)$ and setting $\mathit{cost}((e, r_i)) = r_i \cdot \mathit{length}(e)$.

Let $x_{(e,r)}$ be a boolean variable denoting whether edge e is used at rate r in an optimum tree. The QoS Steiner tree problem can be formulated as

$$\min \sum_{(e,r)\in E'} x_{(e,r)} \cdot r \cdot \mathit{length}(e) \tag{III.1}$$

$$\text{s.t.} \sum_{\substack{(e,r)\in\delta(C)\\ r\geq r_C}} x_{(e,r)} \geq 1, \quad \forall C \subseteq V \setminus \{s\} \tag{III.2}$$

$$x_{(e,r)} \in \{0,1\} \tag{III.3}$$

where $\delta(C)$ denotes the set of edges with exactly one endpoint in C and r_C denotes the maximum rate of a node in C. Note that (III.1) gives the cost of an optimal solution, while (III.2) guarantees that each terminal is connected to the source through a collection of edges of rate no less than its rate.

We relax the integrality constraints (III.3) and consider the dual linear program. For each (e, r), we define $C^*(e,r) = \{C \in V \setminus \{s\} : (e,r) \in \delta(C), r \geq r_C\}$. In words, $C^*(e, r)$ is the set of subsets C of $V \setminus \{s\}$ such that (e, r) has at least one endpoint in C and r is at least as large as r_C. Using this definition, the dual is as follows:

$$\begin{aligned} \max \quad & \sum_C y_C \\ \text{s.t.} \quad & \sum_{C \in C^*(e,r)} y_C \;\leq\; r \cdot \mathit{length}(e), \quad \forall (e,r) \\ & \qquad\quad y_C \;\geq\; 0 \end{aligned}$$

A. *The Naive Primal-Dual Method*

The primal-dual framework applied to network design problems usually grows uniformly the dual variables associated to the "active" components of the current forest [5]. This approach fails to take into account the different rates of different nodes in the QoS problem. In Figure 3 we give a modification, referred to as the "Naive Primal-Dual" algorithm. Our modification takes into account the different rates by varying the speed at which each component grows. While the simulations in the ensuing sections show that this is a good method in practice, the solution it produces on some graphs may be very large compared to the optimal solution, as shown by the following example.

Input: A graph $G = (V, E, \mathit{length}, \mathit{rate})$ with a source s in V and a collection of terminals $S \subseteq V$.
Output: A QoS Steiner tree spanning the source and the terminals.

(1) Start from the spanning forest of G with no edges.
(2) Grow y_C with speed r_C for each "active" component C of the current forest. (A component C is *inactive* if it contains s and all vertices of rate r_C.)
(3) Stop growing once the dual inequality for a pair (e, r) becomes tight, with e connecting two distinct components of the forest.
(4) Add e to the forest, collapsing the two components.
(5) Terminate when there is no active component left.
(6) Keep an edge of the resulting tree at the minimum needed rate.

Fig. 3. The Naive Primal-Dual algorithm for the QoS Steiner Tree Problem.

The Frame Example. Consider two nodes of rate 1 connected by an edge of length 1 (see Figure 4). There is an arc between these two nodes, and on this arc there is a chain of nodes of rate ϵ. Each two consecutive nodes in the chain are at a distance δ from each other, where $\delta < 1$. Each extreme node in the chain is at a distance $\delta/2$ of its neighboring rate-1 node.

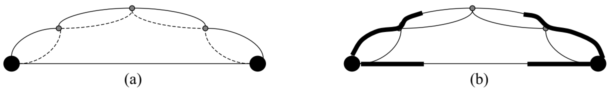

Fig. 4. The Restarting Primal-Dual avoids the mistake of the Naive Primal-Dual. Part (a) shows duplication of the edges. Part (b) shows the components growing along the respective edges.

Input: A Graph $G' = (V, E, cost, rate)$ with source s, and a collection of terminals S.
Output: A QoS Steiner Tree spanning the source and the terminal.

(1) Grow each active C_{r_i} with speed r_i along incident edges (e, r_j), for $j \leq i$, picking edges which become tight.
(2) Continue this process until there is no active component of rate r_k.
(3) Remove all edges which are not necessary for maintaining connectivity of nodes of rate r_k.
(4) Accept (keep in the solution) and contract all edges of C_{r_k} (i.e., set their length/cost to 0)
(5) Restart the algorithm with the new graph

Fig. 5. The Restarting Primal-Dual algorithm for the QoS Steiner Tree Proble.

The Naive Primal-Dual applied to this graph connects the rate-ϵ nodes first, since $\frac{\delta}{2} < \frac{1}{2}$. So, the algorithm connects the rate-1 nodes via the rate-ϵ nodes, and not via the direct edge connecting them. Thus, the Naive Primal-Dual can make arbitrarily large errors (just take an arbitrarily long chain).

B. *Restarting Primal-Dual Algorithm*

An improved algorithm is given in Figure 5. One can easily see that this is a primal-dual algorithm. Indeed, each addition of an edge to the current solution is the result of growing dual variables. Moreover, since the feasibility requirement for edge a is $\Sigma_{a \in \delta(C)} y_C \leq r \cdot \mathit{length}(a)$, this addition preserves the feasibility of the dual solution. The algorithm maintains forests F^{r_i} given by the edges picked at rate r_i, and the connected components of F^{r_i}, seen as sets of vertices, are denoted in the algorithm by C_{r_i}. Such a component is *active* if $r_{C_{r_i}} = r_i$ and C_{r_i} is disjoint from components of higher rate.

The Restarting Primal-Dual avoids the mistake made by the Naive Primal-Dual on the frame example in Figure 4(a). Then, at time $\frac{\delta}{2}$ the rate-ϵ nodes become connected. This means that $\delta(1 - \epsilon)$ of each rate-1 edge between the ϵ-rate nodes is not covered. Meanwhile, the rate-1 nodes are growing on the respective edges as shown in Figure 4(b).

Let us assume that the Restarting Primal-Dual uses the chain of rate-ϵ nodes to connect the two rate-1 nodes instead of the direct edge. This would imply that it takes less time to cover the chain, i.e., $\frac{1}{2}\delta(1 - \epsilon)n \leq \frac{1}{2} - \frac{\delta}{2}$, where n is the number of rate-ϵ nodes. With ϵ small, we obtain $n\delta \leq 1$, so if the Restarting Primal-Dual uses the chain then it is correct to do so.

C. Primal-Dual 4.311-Approximation Algorithm

A primal-dual constant-factor approximation algorithm can be obtained based on the enhanced integer linear programming formulation below. It takes into account the fact that if a set $C \subset V \setminus \{s\}$ is connected to the source with edges of rate $r' > r_C$, then there should be at least **two** edges of rate r' with exactly one endpoint in C. The integer program is

$$\begin{aligned} \min \quad & \sum_{(e,r)\in E'} x_{(e,r)} \cdot r \cdot \textit{length}(e) \\ \text{s.t.} \quad & \sum_{\substack{e\in\delta(C)\\ r=r_C}} x_{(e,r)} + \frac{1}{2} \sum_{\substack{e\in\delta(C)\\ r>r_C}} x_{(e,r)} \ \geq\ 1, \ \ \forall C \subseteq V \setminus \{s\} \\ & x_{(e,r)} \in \{0,1\} \end{aligned}$$

The corresponding dual of the LP relaxation is

$$\begin{aligned} \max \quad & \sum_{C\subseteq V\setminus\{s\}} y_C \\ \text{s.t.} \quad & \sum_{\substack{C\ :e\in\delta(C)\\ r_C=r}} y_C + \frac{1}{2} \sum_{\substack{C\ :e\in\delta(C)\\ r_C<r}} y_C \ \leq\ r \cdot \textit{length}(e) \qquad \text{(III.4)} \\ & y_C \ \geq\ 0 \end{aligned}$$

The core of the algorithm is presented in Figure 6. Before that, we do a random bucketing of rates following [4]. Let a be a real (to be picked later) and γ be a real picked uniformly at random from the interval $[0\,..\,1]$. Every node of rate r is replaced by a node of rate $a^{\gamma+j}$, where j is the integer satisfying $a^{\gamma+j-1} < r \leq a^{\gamma+j}$.

The primal-dual part follows the classical framework [5], and works in stages starting from the lower rate to the highest. During the execution of the algorithm, edges are picked at a certain rate (in other words, $x_{(e,r)}$ is set to 1) one by one. Before executing step 3 at rate r for the ith time, the set of edges picked at rate r by the algorithm forms a forest F_i^r. (An edge can be picked at several rates, but it is kept in at most one such rate in the final solution because of the reverse delete step.) A component C of F_i^r is called an *r-component* if $r_C = r$.

Input: A graph $G = (V, E, \textit{length}, \textit{rate})$ with source s in V and a collection of terminals $S \subseteq V$.
Output: A QoS Steiner tree spanning the source and the terminal.

(1) For each $r = r_1, r_2, \ldots, r_k$, execute steps 2-6.
(2) Start from the spanning forest F^r of G with no edges.
(3) Grow y_C uniformly for each r-component C of the current forest F^r.
(4) Stop growing once the dual inequality for a pair (e, r) becomes tight, with e connecting two distinct components of F^r.
(5) Add (e, r) to F^r, collapsing two of its components.
(6) Terminate when there is no r-component of F^r left.
(7) Traversing the list of picked edges in reverse order, remove an edge (e, r) from F^r if after (e, r)'s removal the set of edges picked form a feasible tree.

Fig. 6. The 4.311-approximation algorithm for QoS Steiner Tree.

Using Constraint (III.4), it follows by induction on i that, for an edge e and a rate $a^{\gamma+j}$, we have

$$\begin{aligned} \sum_{\substack{C\ :e\in\delta(C)\\ r_C\leq a^{\gamma+j}}} y_C \ &\leq\ \textit{length}(e) a^{\gamma+j} \sum_{j=0}^{i} \left(\frac{1}{2a}\right)^j \\ &\leq\ \textit{length}(e) a^{\gamma+j} \frac{2a}{2a-1}. \end{aligned}$$

For an edge picked by the algorithm at rate r, Constraint (III.4) is tight and therefore

$$\sum_{\substack{C\ :e\in\delta(C)\\ r_C\leq a^{\gamma+j}}} y_C \geq \textit{length}(e) \frac{2a-2}{2a-1} a^{\gamma+j}. \qquad \text{(III.5)}$$

Exactly as in [5], we have that the number of edges of rate r in the final solution which cross the active r-components at some moment (an edge being counted twice if it crosses two r-components) is at most twice the number of active r-components. Using Equation (III.5) and exactly the same argument as in Theorem 4.2 of [5], we obtain that the cost of the solution of the algorithm is bounded by $(2(2a-1)/(2a-2)) \sum y_C \leq ((2a-1)/(a-1))\,\textit{opt}$, as any feasible solution for the dual linear program has value at most the value of any feasible solution of the primal.

The same argument as in [4] shows that the approximation ratio of the algorithm above is $(2a-1)/\ln a$. Numerically picking the best value for a, we obtain:

Theorem III.1: The output cost of the algorithm on Figure 6 is at most 4.311 times the optimum cost.

IV. Experimental Study

All algorithms except the very recent 4.311-approximation Primal-Dual were implemented in C++. The heuristics were compiled using `gpp` with `-O2` optimization, and run on a Sun workstation Ultra-60. The experiments were run on randomly generated testcases. Table I gives a comparison of the performance of of the aforementioned algorithms. The experiments were conducted in the presence of no Steiner nodes, respectively 50% Steiner nodes. Moreover, both arithmetic and geometric distributions of rates were tested.

Table I gives the results of a multitude of experiments; however, the results are fairly uniform throughout. It can be observed that the Naive Primal-Dual and the Charikar-Naor-Schieber algorithms most often produce comparable results which are slight improvements over the results produced by Maxemchuk's algorithm. The Restarting Primal-Dual typically produces the best result, which is typically $0.25 - 6$ percent better than the result produced by Maxemchuk's algorithm; this, however, occurs at the expense of greater CPU time. It can also be observed

TABLE I

COST IMPROVEMENT OVER MAXEMCHUCK'S ALGORITHM (%) AND CPU SECONDS FOR CHARIKAR-NAOR-SCHIEBER AND PRIMAL-DUAL ALGORITHMS (AVERAGES OVER 10 TESTCASES).

R	N	Maxem.	Charikar		Naive-PD		Restart-PD	
		CPU	%G	CPU	%G	CPU	%G	CPU
50% steiner nodes, geometric progression rates								
1	200	0.017	0.00	0.017	-0.01	0.544	-0.01	0.325
1	300	0.050	0.00	0.052	0.04	1.372	0.04	0.946
2	200	0.027	0.00	0.026	0.43	1.271	1.03	1.125
2	300	0.070	0.00	0.072	0.93	4.573	2.17	3.747
5	200	0.044	0.00	0.044	-2.13	1.490	1.30	5.321
5	300	0.123	0.00	0.120	-0.91	5.221	1.10	16.798
10	200	0.065	0.00	0.068	-2.53	1.636	0.66	17.848
10	300	0.180	0.00	0.176	-2.61	6.582	0.24	107.125
50% steiner nodes, arithmetic progression rates								
1	200	0.016	0.00	0.017	-0.01	0.541	-0.01	0.327
1	300	0.052	0.00	0.051	0.04	1.370	0.04	0.946
2	200	0.027	0.00	0.023	-0.69	1.373	-0.00	1.136
2	300	0.071	0.00	0.070	-0.32	4.491	0.24	3.773
5	200	0.043	-0.01	0.040	1.70	1.564	2.66	5.256
5	300	0.123	-0.10	0.107	1.92	5.392	4.19	17.271
10	200	0.067	1.79	0.043	4.25	1.556	6.11	16.856
10	300	0.181	2.36	0.126	3.38	5.444	5.73	92.575
0% steiner nodes, geometric progression rates								
1	100	0.002	0.00	0.002	0.00	0.052	0.00	0.077
1	200	0.028	0.00	0.028	0.00	0.251	0.00	0.465
2	100	0.007	0.00	0.007	1.21	0.088	1.69	0.185
2	200	0.038	0.00	0.033	2.14	0.698	2.31	1.517
5	100	0.012	0.00	0.013	1.24	0.120	2.82	0.665
5	200	0.059	0.00	0.056	-0.25	1.296	1.70	6.314
10	100	0.019	0.00	0.018	-0.68	0.133	1.63	1.953
10	200	0.090	0.00	0.091	-1.97	1.466	0.73	20.525
0% steiner nodes, arithmetic progression rates								
1	100	0.005	0.00	0.005	0.00	0.054	0.00	0.078
1	200	0.026	0.00	0.026	0.00	0.247	0.00	0.457
2	100	0.005	0.00	0.006	-0.11	0.111	-0.04	0.187
2	200	0.036	0.00	0.034	-0.02	1.078	0.30	1.570
5	100	0.011	-0.17	0.011	3.70	0.114	4.60	0.656
5	200	0.059	-0.15	0.052	3.13	1.235	3.85	5.952
10	100	0.019	2.62	0.012	6.65	0.113	7.12	1.922
10	200	0.091	2.67	0.058	5.83	1.203	6.38	17.689

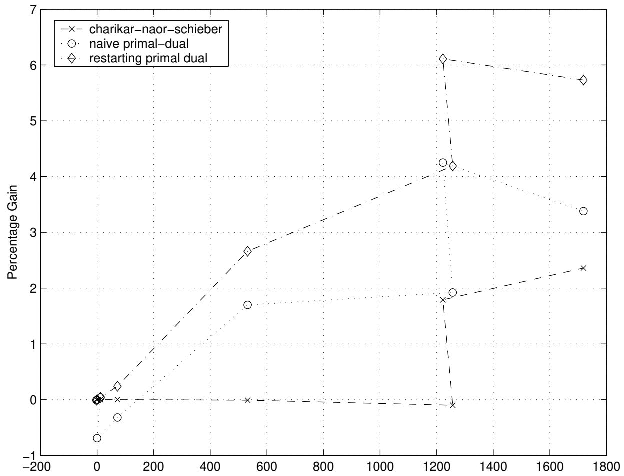

Fig. 7. The gain of several algorithms versus Maxemchuk's algorithm, 50% Steiner nodes.

that the difference between the algorithms increases as the number of rates increases. Figures 7 and 8 illustrate these results in graphical form.

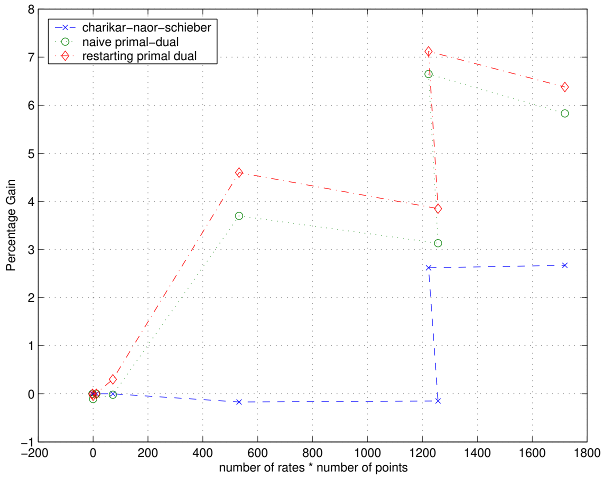

Fig. 8. The gain of several algorithms versus Maxemchuk's algorithm, 0% Steiner nodes.

REFERENCES

[1] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani, *Modeling and Heuristic Worst-Case Performance Analysis of the Two-Level Network Design Problem*, Management Science **40**:846–867, (1994).

[2] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani, *Heuristics, LPs, and Trees on Trees: Network Design Analyses*, Operations Research **44**:478–496, (1996).

[3] R. Bajaj, C.P. Ravikumar, and S. Chandra, *Distributed Delay Constrained Multicast Path Setup High Speed Networks*, Proceedings of the Fourth International Conference on High Performance Computing, 438–442, 1997.

[4] M. Charikar, J. Naor, and B. Schieber, *Resource Optimization in QoS Multicast Routing of Real-Time Multimedia*, Proceedings of the 19th Annual IEEE INFOCOM, (2000).

[5] M. Goemans and D. Williamson, *The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems*, in Approximation Algorithms, D. Hochbaum, Ed., 1997.

[6] M. Karpinski, I. Mandoiu, A. Olshevsky, and A. Zelikovsky, *Improved Approximation Algorithms for Some Generalization of the Steiner Tree Problem*, Proc. Int. Workshop on Algorithms and Data Structures, LNCS 2748, (2003), pp. 401–411.

[7] N. Maxemchuk, *Video Distribution on Multicast Networks*, IEEE Journal on Selected Issues in Communications **15**:357-372 (1997).

[8] P. Mirchandani, *The multi-tier tree problem*, INFORMS Journal on Computing **8**:202–218 (1996).

[9] H.J. Promel and A. Steger, *A new approximation algorithm for the Steiner tree problem with performance ratio* $\frac{5}{3}$, Journal of Algorithms **36**:89–101, (2000).

[10] G. Robins and A. Zelikovsky, *Improved Steiner Tree Approximation in Graphs*, Proc. of ACM/SIAM Symposium on Discrete Algorithms (SODA 2000), 770–779.

[11] G.N. Rouskas and I. Baldine, *Multicast routing with end-to-end delay and delay variation constraints*, IEEE J. on Selected Areas in Communications **15**:346–356, (1997).

[12] H.F. Salama, D.S. Reeves, and Y. Viniotis, *Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks*, IEEE Journal on Selected Areas in Communications, **3**:332–345 (1997).

[13] H. Takahashi and A. Matsuyama, *An Approximate Solution for the Steiner Tree Problem in Graphs*, Math. Japonic, **6**:573–577, (1980).

[14] H. Ural and K. Zhu, *An Efficient Distributed QoS Based Multicast Routing Algorithm*, Proceedings of the 21st IEEE International Performance, Computing, and Communication Conference, 27–36 (2002).

[15] G. Xue, G.-H. Lin, and D.-Z. Du, *Grade of service Steiner minimum trees in the Euclidean plane*, Algorithmica **31**:479–500, (2001).