An Efficient Multicast Routing Algorithm for Delay-Sensitive Applications with Dynamic Membership. *

Sung-Pil Hong †
Chung-Ang University

Heesang Lee
Hankuk University

Bum Hwan Park
Seoul National University

Revised: December 1997

Abstract - *We propose an algorithm for finding a multicast tree in packet-switched networks. The objective is to minimize total cost incurred at the multicast path. The routing model is based on the minimum cost Steiner tree problem. The Steiner problem is extended, in our paper, to incorporate two additional requirements. First , the delay experienced along the path from the source to each destination is bounded. Second, the destinations are allowed to join and leave multicasting anytime during a session. To minimize the disruption to on-going multicasting the algorithm adopts the idea of connecting a new destination to the current multicasting by a minimum cost path satisfying the delay bound. To find such a path is an NP-hard problem and an enumerative method relying on generation of delay bounded paths between node pairs is not likely to find a good routing path in acceptable computation time when network size is large. To cope with such difficulty, the proposed algorithm utilizes an optimization technique called Lagrangian relaxation method. A computational experiment is done on relatively dense and large Waxman's networks. The results seem to be promising. For sparse networks, the algorithm can find near-optimal multicast trees. Also the quality of multicast trees does not seem to deteriorate even when the network size grows. Furthermore, the experimental results shows that the computational efforts for each addition of node to the call are fairly moderate, namely the same as to solve a few shortest path problems.*

1 Introduction

Multicast is a communication in which message streams generated by a single node (*source*) are concurrently distributed to more than one nodes (*destinations*). A basic routing goal is to minimize the total connection cost and a basic routing model is typically based on the minimum cost Steiner tree problem. (see e.g., [21], [4], [8], [17], [22], [6], [19], and [2]).

As real-time multimedia services are expected to be popular in the emerging BISDNs, to reduce the delay along the connection path from the source to destinations to a certain allowable range becomes an important issue. For instance, multicast applications such as video and audio conference require quality of service (QoS) guarantees on end-to-end delay and the loss probability to provide a smooth play-out at the receiver. Also, unlike the case in the point-to-point connections, the clock recovery and synchronization among the group of users in the multipoint-to-multipoint connections remain open issues [16]. Thus a mechanism of establishing multicasting tree of a bounded delay may facilitate to resolve such issues. Note that the maximum cell loss probability can be guaranteed for a multicasting by call admission control and the resource reservation [9]. Hence the routing capability of configuration of multicast tree meeting delay bounds is an important issue. Thus the problem is not only to minimize the connection cost but also to guarantee that the sum of the delays on the path between the source and each destination is less than a predetermined bound. Several routing algorithms have been proposed in the literature for this problem [3] [14] [23] [7].

Beside the QoS consideration, the proposed routing model also allows the destinations to join and leave the multicasting anytime during the life of a session. The *membership* of each node of the network to a multicasting changes dynamically in the evolution of time. Considering the characteristics of the real-time applications requiring multicasting, this extension will enhance the applicability of the routing model. We will refer to this problem as *delay-constrained minimum-cost multicast-routing problem with dynamic membership* (DDMP). (To the best of our knowledge this is the first model considering the delay bound and the dynamic membership simultaneously.)

There have been some studies on computing the multicast tree for the multicast routing with dynamic membership but without delay considerations [21] [8]. An important observation in these studies is that when the destinations are added to the current multicasting tree, a good routing solution can be obtained simply by connecting each destination to the tree by a minimum cost path between (a node of) the tree and the destination. This approach has a special advantage of minimizing disruption to on-going session. As an analogy, we can think of a routing algorithm for DDMP in which a new destination is connected the current multicast tree via a minimum cost path satisfying the delay bound.

Thus in such a routing algorithm it is critical to develop an efficient algorithm finding a delay bounded minimum cost path between a new destination and the tree. However this problem is NP-hard as it is more general problem than the one finding a delay bounded minimum cost path between a pair of nodes which is known to be NP-hard [10]. One may suggests an immediate approach (as found often in the literature) which generates all delay bounded paths between the destination and the tree and selects a minimum cost path among them. This enumerative method is, however, prohibitive since there may be an exponentially many such paths and hence the computational efforts can be too large even for the moderate size networks. If, on the other hand, the number of generated paths is restricted (as also found in the literature), the quality of solution may deteriorate. So we need to rely on

*The research is partially supported by a IITA grant 96163-CT-I-1 of Ministry of Information and Communications of Korea.

†corresponding author sphong@cau.ac.kr. Research also partially supported by *Korea Telecom* under contract 96-29

more elaborated approach to avoid such problems. In this paper, we adopt the heuristic proposed by [5]. We can show that this heuristic is a variant of *Lagrangian relaxation method* frequently used to solve NP-hard problems. But, the particular variant has a iteration scheme which is very practical rendering computation time minimal as well as "adaptive" making final solution near optimal. The proposed routing algorithm is based on this heuristic.

This paper is organized as follows. In Section 2, the proposed routing model DDMP is described formally. Section 3 reviews SPH heuristic for the basic routing model and discuss how its idea can be naturally extended to a routing algorithm for the model DDMP. Section 4 briefly discusses the optimization technique (Lagrangian method) proposed in [5] and show how it can be used for our purpose. The section also describes the routing algorithm DDMP which uses the Lagrangian method as a subroutine. Section 5 summarizes the computational experiment and the results.

2 The Routing Model

Let $G = (N, E)$ be a network where $N = \{1, 2, \cdots, n\}$ denotes the set of nodes and E the set of links. The network can be either be a physical or a logical according to the relevant packet-switching technology. Each link $(i, j) \in E$ is assigned a *cost* c_{ij} and a *packet delay* d_{ij}.

A point-to-multipoint connection consists of a *source* $r \in N$ which concurrently sends the same packets to a set of *destinations*, $D \subseteq N - \{r\}$. Each element of the set $M \equiv D \cup \{r\}$ will be referred to as a *member* of the multicast. If all c_{ij} are nonnegative, a minimum cost routing is given as a tree spanning r and the destinations D. This tree is called a minimum cost *Steiner tree* spanning the members M and will be denoted by $T = (N_T, E_T)$. In T there is a unique path $P_T(r, v)$ from r to each destination $v \in D$. For all v the total packet delay over $P_T(r, v)$ is required to be no greater than a bound Δ.

If D is assumed to be unchanged to the conclusion of a session we call the above problem *static delay-constrained minimum-cost multicast-routing problem* (SDMP). (This problem will be used in evaluation of our algorithm performance.) In sum, SDMP can be stated formally as follows.

Problem 2.1 *(SDMP) Given a network $G = (N, E)$, a source $r \in N$, a destination set $D \subseteq N - \{r\}$, link costs c_{ij} and link delays d_{ij} for $(i, j) \in E$, find a minimum cost (Steiner) tree spanning r and D which satisfies*

$$\sum_{(i,j)\in P_T(r,v)} d_{ij} \le \Delta \ \forall v \in D. \tag{2.1}$$

In *dynamic delay-constrained minimum-cost multicast-routing problem* (DDMP), on the other hand, D (hence M) is subject to change as nodes can join or leave multicasting any point during the life of a session. Hence, denoting the destination set after the kth membership change by M_k, it can be written as

$$M_k = M_{k-1} \oplus \{v_k\}, \quad k = 1, 2, \cdots \tag{2.2}$$

where, $\oplus$ is the set operation, $\cup$ or $-$ according to whether v_k is joining or leaving (respectively) multicasting. Note that this dynamic model is compatible with the signalling protocol in ITU-T recommendation [13] , where a *leaf* may be added or dropped from the call any time while the call is in active state. Since in the protocol the multicasting is initiated with a connection between the source r and a destination, M_0 needs to be set to $\{r\}$. With this notation, DDMP can be stated recursively as follows.

Problem 2.2 *(DDMP) Given a (optimal) multicast tree T_{k-1} spanning M_{k-1}, find a minimum cost multicast tree T_k spanning M_k such that*

$$\sum_{(i,j)\in P_{T_k}(r,v)} d_{ij} \le \Delta, \ \forall v \in M_k - \{r\}, k = 1, 2, \cdots. \tag{2.3}$$

An obvious way to deal with DDMP is to apply the algorithm for SDMP to DDMP for each k from scratch. The computational efforts, however, can be too excessive for the actual improvement of the solution. There is more efficient method to configure new multicast tree based on the current tree but still be able to maintain the quality of the solutions as will be seen in subsequent sections. More importantly, it is necessary to minimize the disruption to the on-going multicasting when a node is added or dropped from the multicasting. If the order of packet arrivals or the direction of traffic flows on the links are subject to change due to the changes in the edges involved in the connection, the required control signaling may become prohibitively excessive especially for the delay-sensitive real-time traffics. Hence we adopt the policy in [21] that once an edge is used in the multicast, the edge remains in the connection with the same traffic flow direction until there is no destination which receives the packets via the edge. This implies, in particular, even if a node no longer remains in the multicast as a destination, it will remain in the connection as an intermediate node as long as it has a destination downstream.

Thus when a node is leaving, DDMP is a quite simple problem and the algorithm performance mainly depends on how good connecting paths it can find at the additions of incoming destinations. Hence in the experiment we will consider only node addition at each k:

$$\begin{aligned} M_0 &= \{r\} \\ M_k &= M_{k-1} \cup \{v_k\} \quad k = 1, 2, \cdots, m-1, \end{aligned} \tag{2.4}$$

for some finite number m.

3 The Shortest Path Heuristic

There is a vast literature on the solution method of the Steiner tree problem (See, e.g. [12]). Although the problem is NP-hard, it is well solved in the sense that there are fast heuristic algorithms which provide practically good solutions. Of the various heuristics, one of most frequently adopted ones for multicast is *shortest path heuristic*(SPH) [20].

SPH heuristic [20] is a tree growing algorithm. It begins a single node tree of an arbitrarily chosen multicast member (e.g. the source). At each iteration, among the members currently not in the tree it identifies one which can be connected to the tree with the minimum cost increment. Then he member is added to the current multicast tree via corresponding path. This is repeated until the multicast tree spans all the members.

Algorithm 3.1 *Shortest path heuristic (SPH) [20]*

```
Step 0: $T \leftarrow v_1$, where $v_1$ is any member; $k \leftarrow 1$.
Step 1: Repeat Step 2 until $T$ contains all the members.
Step 2: Find a member $v_{k+1}$ closest to $T$ and the corresponding
    path $P_{k+1}$; $T \leftarrow T \cup P_{k+1}$; $k \leftarrow k+1$.
```

Surprisingly this simple greedy-type heuristic is known to provide the solutions of not only empirically but also theoretically provable quality.

Theorem 3.2 *[20] Let a optimal solution and the solution given by SPH be T^* and T, respectively. Then $c(T)/c(T^*) \leq 2(1-1/|M|)$, where $c(T) \equiv \sum_{(i,j)\in T} c_{ij}$.*

For sparse network the ratio is far less than 2 on average. It is reported in [8] that even a "naive" version of SPH heuristic, in which the members are added via minimum cost path to the source in an arbitrary order, generates the solutions which are only marginally inferior. This suggests that even when the destinations are to be added to the call in an arbitrary order of joining requests, a good routing solution can be obtained simply by adding each destination via a minimum cost path in the requested order. An advantage of this simple method is that the disruption to on-going multicasting can be minimized as it does not require a reconfiguration of the routes already involved in on-going multicasting. Indeed this approach has been incorporated into the algorithms for the routing models with dynamic membership [21], [4], [8], [6].

Thus this approach looks also promising for DDMP. In DDMP, however, at each addition of destination, it is necessary not only to find a minimum cost path from new destination to the current multicast tree but also to make sure the delay bound in Eq. (2.3) is satisfied in the resulting tree. As mentioned in Section 1, this problem is intractable (NP-hard and hence a simple enumerative method can not find a good path within a reasonable computation time.

4 The BG Heuristic

In this section, an algorithm is proposed for finding a minimum cost path from a incoming destination and the current multicast tree so that in the new tree given as the union of the path and the current tree the delay constraint in Eq. (2.3) is still satisfied. The algorithm is based on a heuristic (will be called the *BG heuristic*) proposed by by Blokh and Gutin [5] We can argue that BG heuristic is basically

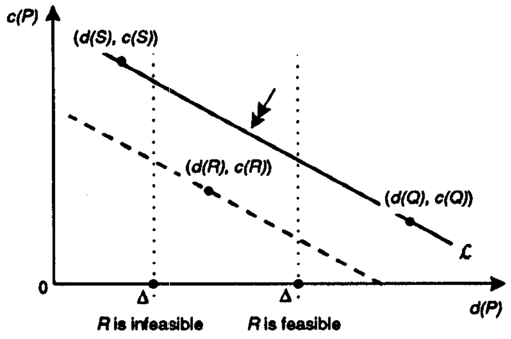


Figure 1: Main iteration of BG heuristic.

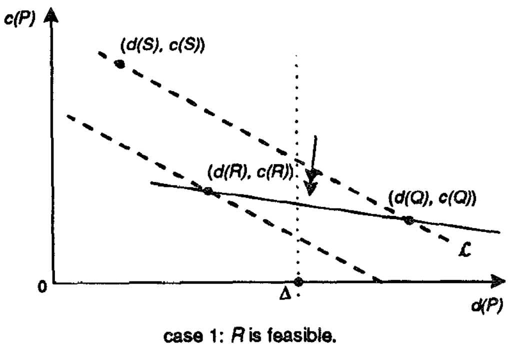
a variant of the *Lagrangian relaxation method*. However, this particular heuristic has very pragmatic iteration scheme rendering computation time minimal as well as an adaptive feature which is likely to produce near optimal solutions.

Consider, as in Section 2, a network $G = (N,\ E)$ in which each link (i, j) is assigned two parameters, a cost c_{ij} and a delay d_{ij}. Let $s, t \in N$ be a pair of nodes. Then the problem is to find an s-t path P^* so that the cost of $P^*, c(P^*) \equiv \sum_{(i,j)\in P^*} c_{ij}$ is minimized while the delay of of P^*, $d(P^*) \equiv \sum_{(i,j)\in P^*} d_{ij} \leq \Delta$. This problem is *NP-hard*. But, notice that finding a path minimizing the sum of a single parameter c or d is easy as it is a shortest path problem. Let $\mathcal{A}$ be any shortest path algorithm.

Algorithm 4.1 *BG heuristic [5]*

```
Step 1: Using $\mathcal{A}$ find an s-t path Q so that $c(Q) \equiv \sum_{(i,j)\in Q} c_{ij}$ is
    minimized. If $d(Q) \leq \Delta$, then Q is an optimal solution. Stop.
Step 2: Using $\mathcal{A}$ find an s-t path S so that d(S) is minimized. If
    $d(S) > \Delta$, then there is no solution. Stop.
Step 3: Set $\alpha \leftarrow c(S) - c(Q)$, $\beta \leftarrow d(Q) - d(S)$ and $\gamma \leftarrow$
    $d(Q)c(S) - d(S)c(Q)$. Compute $e_{ij} \leftarrow \alpha d_{ij} + \beta c_{ij}$ for each
    $(i,j) \in E$. Using $\mathcal{A}$ find an s-t path R so that e(R) is
    minimized.
Step 4: If $e(R) = \gamma$ and $d(R) \leq \Delta$, then output R as the solution.
    If $\gamma = e(R)$ and $d(R) > \Delta$, then output S as the solution.
Step 5: (Case 1) If $e(R) < \gamma$ and $d(R) \leq \Delta$, then set $S \leftarrow R$.
Step 5: (Case 2) If $e(R) < \gamma$ and $d(R) > \Delta$, then set $Q \leftarrow R$. Go
    to Step 3.
```

The idea of this algorithm is best illustrated on the two-dimensional plane whose horizontal and vertical axes correspond to the values $d(P)$ and $c(P)$, respectively, for each s-t path P. (See Figure 1). At each iteration, the algorithm maintains two solutions: the current best feasible (delay bounded) path S and the current best infeasible path Q. Consider the line $\mathcal{L}$ intersecting $(d(Q), c(Q))$ and $(d(S), c(S))$. In *Step 3* the parameters α, β, and γ are defined to construct a cost function $e(P) = \alpha d(P) + \beta c(P)$ for each s-t path P so that every P satisfying $(d(P), c(P)) \in \mathcal{L}$ has the same



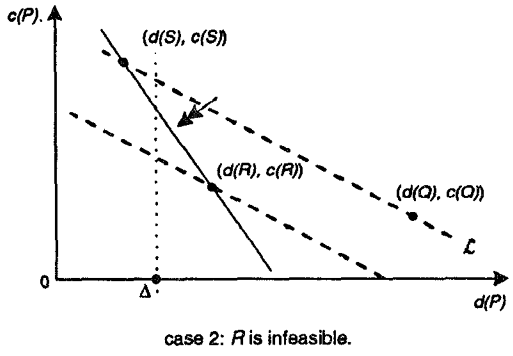


Figure 2: Adaptiveness of subsequent iterations.

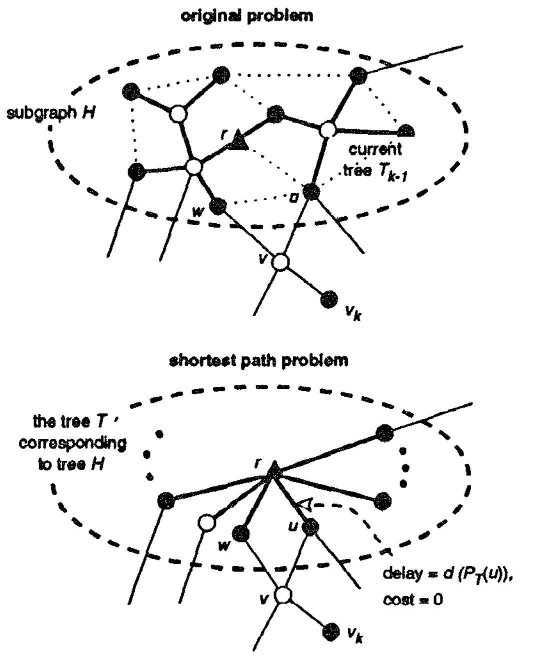


Figure 3: Modification to a delay constrained minimum cost path problem.

value γ. This means that the algorithm consider two solutions indifferent if they are plotted on a straight line parallel to $\mathcal{L}$. The heuristic then tries to reduce c and d simultaneously by minimizing e. If successful, namely the obtained solution R has $e(R) < \gamma$, then two possible cases may arise. First , if R is a feasible solution *i.e.* $d(R) < \Delta$ (*Step 5 (Case 1)*), then R replaces S. Thus in the next iteration, the line $\mathcal{L}$ will intersect the points Q and R as in the case1 in Figure 2. So relatively larger weight is multiplied on the cost term than the previous iteration. If, on the other hand, R does not satisfies the delay bound (*Step 5 (Case 2)*), then R replaces Q. By doing so the algorithm places more emphasis on the delay minimization in the next iteration. The procedure is repeated until an unsuccessful iteration occurs (*Step 4*).

We can see that in *Step 3* the obtained solution R is always a minimal element with respect to the partial order defined by $(d(P), c(P))$ on the set of s-t paths P. Also it is shown in [5] that the algorithm never generates the same Q more than once. Hence the number of iterations is less than the number of minimal elements. The major computational effort at each iteration is to update $e(s)$ for $s \in S$ and to apply the algorithm $\mathcal{A}$.

Theorem 4.2 *[5] The BG heuristic terminates in $O(m(S)\ (|S|+t(\mathcal{A}))$ time where $m(S)$ is the total number of the minimal elements $t(\mathcal{A})$ is the computation time of $\mathcal{A}$.*

It is not known how many minimal points can exist. However, in Figure 1, we can intuitively argue that the heuristic may achieve 50% relative improvement in objective value on average at each successful iteration. Thus it may be conjectured that the heuristic should be very fast in practice. In [5], experimental results reported are not enough to support this conjecture. Our experiment in Section 5 strongly shows that total computation time is bounded by a low order polynomial function.

Now we show how the BG heuristic can be used to solve our problem which can be stated in the context of Problem 2.2. We are given a (optimal) multicast tree T_{k-1} in the network G, connecting the multicasting members M_{k-1}. The node v_k is joining the multicasting. v_k is to be connected to T_{k-1} by a path P_k so that the tree $T_k \equiv T_{k-1} \cup P_k$ connects the increased members $M_k \equiv M_{k-1} \cup \{v_k\}$. The path P_k need to be a minimum cost path while delay constraint is still satisfied in T_k

$$\sum_{(i,j)\in P_{T_k}(r,v)} d_{ij} \leq \Delta, \ \forall v \in M_k - \{r\}. \qquad (4.5)$$

The problem finding such a path can be reduced to a delay constrained minimum cost path problem between r and v_k if the network G is modified as follows (See Figure 3).

- Consider the (star-shaped) tree T' consisting of the nodes of T_{k-1} in which every node except the source r is adjacent to r.
- Assign to each edge (r, v) of T' the zero cost and the delay, $d(P_{T_{k-1}}(v))$ (the total delay along the path from r to v in T_{k-1}).
- In G, replace the subgraph H induced by the nodes of T_{k-1} by the star-shaped tree T'.

It is easy to see that the delay bounded shortest path P_k between r and v_k on the modified network corresponds to a desired path in the original network. Such a path, of course, can be obtained by applying BG heuristic to the modified network. Thus BG heuristic can serve as a subroutine in the main routing algorithm to find a delay constrained minimum cost path along which an incoming destination is connected to the current multicast.

5 Experimental Results

The instances used in the experiments are Waxman's networks [21]. The n nodes of the network G are randomly selected from an $L \times L$ square lattice points with unit spacing. For each pair of nodes u and v, the edge (u, v) is chosen with the probability,

$$\Pr(u,v) = \beta \exp \frac{-\delta(u,v)}{2\alpha L}, \qquad (5.6)$$

where $\delta(u, v)$ is the *Manhattan distance* (*i.e.* the rectilinear distance) between u and v. The parameters β and α are chosen from the interval $(0, 1]$. If α is large, the probability decreases slowly for large distance, and hence longer edges have more chance to exist. Hence a large value of α increases the connectivity of the network. By increasing β we can uniformly increase the density of the network instances.

The node size n is set to 50, 100, 200, 300, 400, and 500. For each case, the number of destination $|D|$ is set to 20% of the nodes. In our experiment, L is set to $\sqrt{10n}$. This will be used later for algorithm performance evaluation.

Each of α and β is set to 0.3, 0.5 and 0.8. For each node size, instances are generated for all possible combinations of the α and β values. For each n, the average edge size is larger than 30% that of the complete graph. Note that the instances includes significantly larger and more dense networks than in the previous studies.

For each network, the members M, the source r and the destinations D are randomly chosen from the nodes. By the notation in Eq. (2.4), $M = M_{m-1}$, $M_0 = \{r\}$, $D = \{v_1, v_2, \cdots, v_{m-1}\}$, and so $|D| = m - 1$. In setting up the delay bound Δ, the method of [23] is adopted. For each instance, two Steiner trees, T_c and T_d spanning M are computed. Here, T_c is a minimum cost Steiner tree and can be obtained by using one of the efficient heuristics, *e.g.* SPH, as in our case. On the other hand, T_d is a tree in which the maximum delay among the paths from r to the destinations is minimized. T_d can be computed by constructing a shortest path tree with respect to d. Denote by $d_{\max}(T)$ the maximum delay among the paths from r to the destinations in T. (Then $d_{\max}(T_c) \geq d_{\max}(T_d)$.) Then set Δ is chosen by the equation

$$\rho = \frac{\Delta - d_{\max}(T_d)}{d_{\max}(T_c) - d_{\max}(T_d)}, \qquad (5.7)$$

for a predetermined value ρ called *delay bound ratio*. Note that $0 \leq \rho \leq 1$. As ρ becomes close to 0, the delay bound becomes tighter. In our experiment, ρ is set to 0.1 and 0.3 and 0.5.

To complete the instance generation, two parameters, the cost and delay need to be assigned to each edge of the network instances. It seems natural to make the delay of an edge to be positively correlated to the geographical distance of its end nodes. In this sense the delays are generated to be $\delta(u, v)$ times $1 + \omega$, with ω a random number from $[0, 1]$. There are two ways to generate the cost with respect to the its sign of correlation to the delay: positive or negative. Regarding the algorithm performance, the correlation between two parameters is a critical factor. If two parameters have the positive correlation of a significant level, the instances should be relatively easy to solve, since it is more probable that minimizing one parameter produces a path better off also in the other one. We call this a *positive correlation effect.* If they have a negative correlation, the situation becomes opposite to have a *negative correlation effect.* The most difficult case is when the correlation coefficient is -1 so that the delay and cost are of completely inverse relation. Then a cost minimizing path will always have a maximum delay. So such a network is most difficult to find a minimum cost multicast tree satisfying the delay bound. In our experiment, the routing algorithm is simulated on two groups of instances. In Group 1, the parameters have some degree of positive correlation. In Group 2, they have the correlation of -1 and hence are instances most difficult to solve.

N	50	100	200	300	400	500
D	10	20	40	60	80	100
ρ	.1, .3, .5	.1, .3, .5	.1, .3, .5	.1, .3, .5	.1, .3, .5	.1, .3, .5
α	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8
β	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8	.3, .5, .8

Table 1: Parameters of used instances.

For each node size there are 27 possible combinations of the three parameters. For each of 162 combinations, 30 instances are generated so that the total number of instances per each group is 4, 860. The parameters are summarized in Table 1.

Group 1: Instances with positively correlated parameters

In this group of instances, the costs are generated by the same manner as used for the delays. Therefore, the parameters are positively correlated as both of them are positively correlated to the Manhattan distance. Note that this is a typical way found in the literature. In [3] and [23], two parameters were set to be identical or positively correlated in their static routing instances.

For 30 of the 50 node network instances, the costs of the multicast tree generated by the algorithm are compared to the exact minimum cost for the static problem, SDMP in Problem 2.1 with $M = M_{m-1}$. (This was done by applying the branch and bound method to the integer programming formulation of SDMP) Then the main algorithm is applied for DDMP recursively for $k = 1, 2, \cdots m-1$. Then the multicast tree for $k = m - 1$ is compared to the optimal tree of SDMP. By doing so the performance of our algorithm is evaluated conservatively as we can obtain a better multicast tree if we know the complete membership *a priori.* The results are summarized in Table 2. The average ratio of the cost, $\bar{c}$ of our multicast tree to the optimal cost c^* is 1.19 on average. So the cost of the multicast tree is quite close to optimal tree on average. This ratio even seems to be comparable to the performance of the heuristics for the minimum Steiner cost problem (without delay constraints).

Figure 4 shows the heuristic costs (averaged for 30 instances) of all162 combinations of node size, α, β and ρ. Each curve corresponds to each node size. The first three consecutive points

	1	2	3	4	5	6	7	8	9	10
c^*	122	109	112	145	116	113	93	134	66	106
$\bar{c}$	143	120	121	176	129	114	125	187	97	108
$\bar{c}/c^*$	1.17	1.10	1.08	1.21	1.11	1.01	1.34	1.40	1.47	1.02
	11	12	13	14	15	16	17	18	19	20
c^*	100	111	129	133	134	90	128	116	108	105
$\bar{c}$	168	122	143	148	157	90	137	143	116	131
$\bar{c}/c^*$	1.68	1.10	1.11	1.11	1.17	1.00	1.07	1.23	1.07	1.25
	21	22	23	24	25	26	27	28	29	30
c^*	100	118	114	121	133	125	141	116	116	128
$\bar{c}$	104	127	128	166	158	151	158	151	152	152
$\bar{c}/c^*$	1.04	1.08	1.12	1.37	1.19	1.21	1.12	1.30	1.31	1.19

Table 2: Group 1: Comparison of heuristic solutions to optimal solution when $|N| = 50$, $\rho = 0.3$, and $\alpha = 0.3$, $\beta = 0.5$. Average ratio is 1.19.

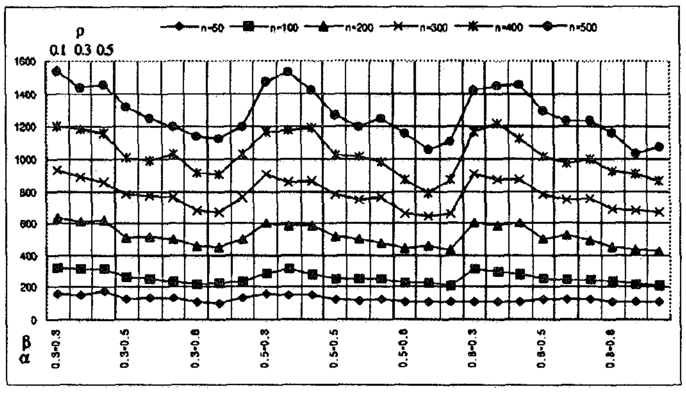


Figure 4: Group 1: Cost of each of162 combinations, averaged over 30 instances.

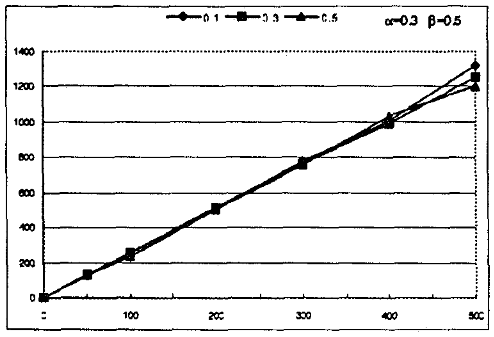


Figure 5: Group 1: Cost vs. node size when $\alpha = 0.3$ and $\beta = 0.5$.

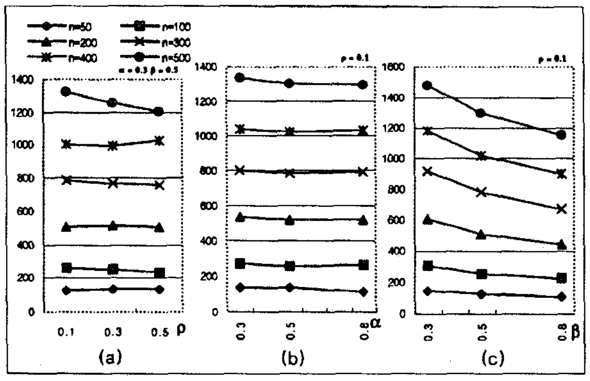


Figure 6: Group 1: Cost effects of ρ, α and β.

at each curves represents the 30 instance average costs of $\rho = 0.1$, 0.3, 0.5 for the same $(\alpha, \beta) = (0.3, 0.3)$, etc.

From this figure, we can make the following observations.

- The marginal increments in cost due to the node size increase are maintained uniformly. This means the cost is a linear function of the node size n. This can be confirmed in Figure 5. We can prove [11] that when L = a constant $\times \sqrt{n}$ as in our experiments, the expected minimum cost is $\Omega(n)$. So we conclude that the quality of solution does not deteriorate when the network size grows.

- The cost effect of delay bound ratio turns out to be not significant as in Figure 6 (a). Due to the positive correlation effect, the algorithm can find near optimal trees even when the delay bound is very tight.

- The effect of α on cost is small (Figure 6 (b)). This means that when a node is added to the on-going multicasting, it can find a good routing path largely relying on the local edges, the edges incident to the nodes in the neighborhood.

- The costs of multicast tree are smaller in a dense network (Figure 4 and 6 (c)). This effect becomes more apparent in large networks. The algorithm is able to find a better multicast tree if the network is dense and there are more choices of path.

The number of the iterations of BG heuristic per addition of a destination to the current tree remains almost as a constant as shown in Figure 7 (a). It is close to 1 on average and never exceeds 5 at maximum . We found that this is common for all instances irrespective of the sizes and parameters. There are several reasons for this. First, again the positive correlation makes the routing easy. Second, the BG heuristic itself is very fast especially in terms of required iterations. This results are compatible to the results reported in [5] and [7]. Third, once the size of multicasting members exceeds a small portion of $|D|$, it takes single iteration of BG heuristic to connect extra nodes to the tree. The CPU time growth for each addition is, therefore, dominated by the time for solving a few shortest path problems (See Figure 7 (b)). We used the original Dijkstra's algorithm for solving the shortest path problem. We expect that the computation time can be considerably reduced by adopting a data structure for large-scale networks [1].

Group 2: Instances with negatively correlated parameters

In the second group of instances, the edge costs are chosen so that the correlation coefficient to be -1: $c_{ij} = 4L - d_{ij}$. (Note that $4L$ is the maximum possible value of d_{ij}.) In a certain sense, it may be plausible that the edge of a larger delay has a lower cost. As mentioned before these are the most difficult instances. Hence the results shown below can be regarded as the worst possible performances of the algorithm. The experiments are done to the same procedure as before.

Figure 3 shows the ratios of the heuristic costs, $\bar{c}$ to the exact static optimal costs c^*. The average is 1.29 on average. So the

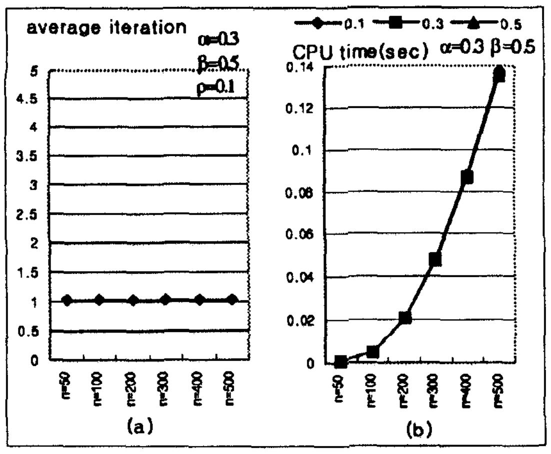


Figure 7: : Group 1: Average iteration and CPU time vs. node size.

	1	2	3	4	5	6	7	8	9	10
c^*	807	747	868	858	813	788	892	849	793	897
$\tilde{c}$	1067	923	1054	1152	1025	961	1032	1183	875	1125
$\tilde{c}/c^*$	1.32	1.24	1.21	1.34	1.26	1.22	1.16	1.39	1.10	1.25
	11	12	13	14	15	16	17	18	19	20
c^*	768	782	891	870	802	842	823	815	876	940
$\tilde{c}$	1038	1209	978	968	1145	1047	940	1101	1161	985
$\tilde{c}/c^*$	1.35	1.54	1.09	1.11	1.43	1.24	1.14	1.35	1.32	1.05
	21	22	23	24	25	26	27	28	29	30
c^*	816	707	808	761	866	860	823	771	884	772
$\tilde{c}$	1191	882	1175	1265	1228	1273	916	819	1142	1041
$\tilde{c}/c^*$	1.46	1.25	1.45	1.66	1.41	1.48	1.11	1.06	1.29	1.35

Table 3: Group 2: Comparison of heuristic solutions to optimal solution when $|N| = 50$, $\rho = 0.3$, and $\alpha = 0.3$, $\beta = 0.5$. Average ratio is 1.29.

quality of solution is only slightly inferior to the one achieved in the positive correlation case.

As before, Figure 8 shows the heuristic costs of all 162 combinations of node size. The cost appears to be slowly increasing concave function. In fact we can prove [11] the expected minimum cost is $\Omega(n^{3/2})$. Thus again we may conclude that the quality of solution does not deteriorate for large networks.

The following are the observations made on the effects of the various parameters on the costs of a multicast tree.

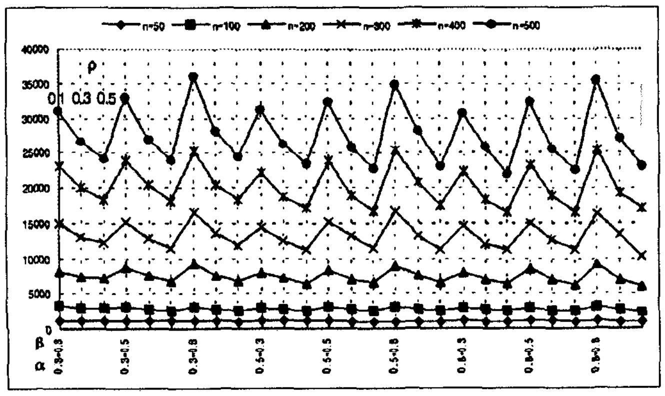


Figure 8: Group 2: Cost of each of162 combinations, averaged over 30 instances.

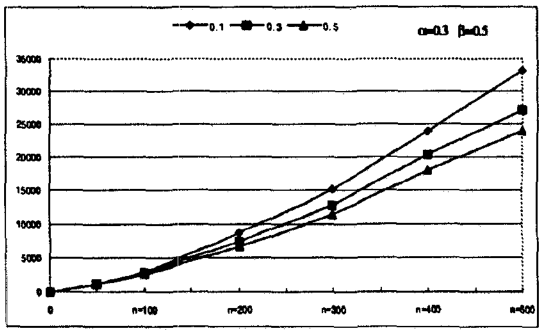


Figure 9: Group 2: Cost vs. node size when $\alpha = 0.3$ and $\beta = 0.5$.

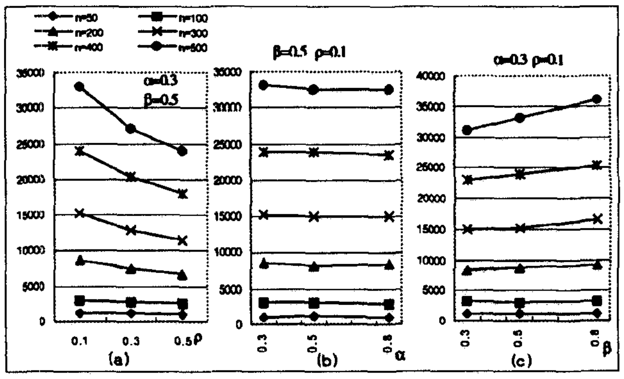


Figure 10: Group 2: Cost effects of ρ, α and β.

- In Group 2, the delay bound effect on cost looks significant (Figure10 (a)). That is, if the delay bound is loose the algorithm can find a multicast tree significantly smaller in cost. This is apparently because the negative correlation effect: the paths of smaller costs tend to have larger delays.
- As in Group 1, the effect of α on the cost is not significant (Figure10 (b)). This shows in any case, the multicasting trees depend largely on the local edges.
- When there are more choices of paths, there would be more paths of smaller costs. However due to the negative correlation, these paths would also have larger delays. It turns out that as the density increases the algorithm is rather "overwhelmed" by the increased choices of the paths under the negative correlation effect. As shown in Figure10 (c), the cost is a somewhat increasing function of the density. So the performance of the algorithm deteriorates in dense networks. (This is compatible to the result in Figure11 (a) showing that it takes more iterations per addition when the density is larger.)

The effect of the delay bound on the average iteration number is significant in Group 2 as shown in Figure 12 (a). Also this is compatible to the effect of the delay bound on the cost discussed above. It is interesting, however, to see that the number of iterations does not increase too fast even if the delay ratio gets close to as small as 0.01 as in Figure 12 (b). (We did extra experiments to obtain this result.) Again we can identify the effect that as the multicasting tree grows, the effort for the addition of an extra node decreases rapidly. To demonstrate this effect, in Figure 12 the

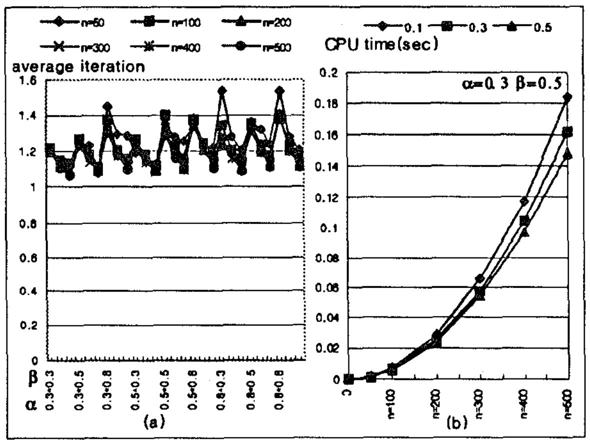


Figure 11: Group 2: (a) Average iteration. (b) CPU time vs. node size .

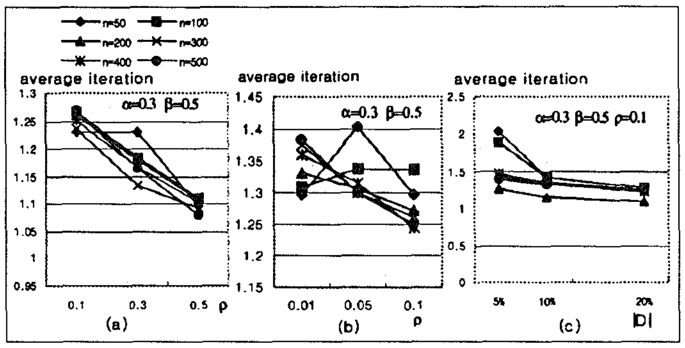


Figure 12: Group 2: Average iteration vs. various factors.

average iteration is plotted to the size of the added multicasting members. The average iterations decreases quickly once the size of members reaches a few percent (3 - 5%) of the nodes. From this, we can see once a multicasting is established with a stable number of members, coping with the dynamic changes in multicast membership is not a difficult problem as far as routing is concerned.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*, Prentice-Hall, New Jersey, 1993.

[2] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," in *IEEE/ACM Trans. on Networking* 4(2):181-191, 1996.

[3] K. Bharath-Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," in *IEEE Trans. on Comm.* com-31(3):343-351, 1983.

[4] L. Berry, "Graph theoretic models for multicast communications," in *Computer networks and ISDN systems* 20:95-99, 1990.

[5] D. Blokh and G. Gutin, "An approximation algorithm for combinatorial optimization problems with two parameters," *manuscript* 1995.

[6] S. Y. Cheung and A. Kumar, "Efficient quorum routing algorithms," in *Proc. IEEE INFOCOM'94* 840-847, 1994.

[7] S.-J. Chung, S.-P. Hong, H.-S. Huh "A fast multicast routing algorithm for delay-sensitive applications," in *Proc. IEEE GLOBECOM'97* 1898-1902,1997.

[8] M. Doar and I. Leslie, "How bad is naive multicast routing," in *Proc. IEEE INFOCOM'93* 82-89, 1993.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, New York, NY, Freeman, 1979.

[10] S.-P. Hong, H. Lee , B.W. Park "An efficient multicast routing algorithm for delay-sensitive applications with dynamic membersip," *full version manuscript*, June (revised, December) 1997.

[11] F. K. Hwang, D. S. Richards, and Pawel Winter, *The Steiner Tree Problem, Annals of Discrete Mathematics 53*, 1992.

[12] ITU-T Recommendation Q.2971: B-ISDN application protocols for access signalling, October 1995.

[13] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communications," *IEEE/ACM Trans. on Networking* 1(3): 286-292, 1993.

[14] L. T. Kou, G. Markowsky and L. Berman, "A fast algortihm for Steiner trees," in *Acta Inf.* 15: 141-145, 1981.

[15] R. O. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues*, Artech House, 1995.

[16] C. S. Ramanathan, "An algorithm for multicast tree generation in networks with asymmetric links," in *Proc. IEEE INFOCOM'96* 353-360, 1996.

[17] G. N. Rouskas and I. Baldine "Multicast routing with end-to-end delay and delay variation constraints," in *Proc. IEEE INFOCOM'96* 353-360, 1996.

[18] A. Shaikh, S. Lu, and K. Shin, "Localized multicast routing," in *Proc. IEEE ICC'95* 1352-1356, 1995.

[19] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," in *Math. Japonica* 6: 573-577, 1980.

[20] B. M. Waxman, "Routing of multipoint connections," in *IEEE J. Select. Areas in Commun.* 6(9):1617-1622, 1988.

[21] B. M. Waxman, "Performance evaluation of multipoint routing algorithm," in *Proc. IEEE INFOCOM'93* 980-986, 1993.

[22] Q. Zhu, P. Parsa, J. J. Garcia-Luna-Aceves "A source-based algorithm for delay-constrained minimum-cost multicasting," in *Proc. IEEE INFOCOM'95* 377-385.