An Efficient Multicast Routing Algorithm for Packet-Switched Networks.

Sung-Jin Chung [*] Sung-Pil Hong [†] Bum Hwan Park [‡]

October 1998

Abstract - *This paper has a dual purpose. First, we consider a relaxation algorithm which seems to be particularly suitable for multicasting routing problems. We show that the algorithm has polynomial complexity. Second, to measure the quality of solutions in comparison to the optimal solutions over a wide range of network sizes for which the computation of the optimal costs is too excessive, we also propose a random graph generation scheme in which an asymptotic lower bound on the expected optimal cost can be computed as a function of network node size.*

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*). To support this type of communication, the network should be equipped with an efficient routing function to establish a corresponding point-to-multipoint connection for each multicast. One of the important issues is to configure the topology of connection so that the network resource utilization is optimized and user QoS requirements are satisfied in accordance with some pre-defined routing goals. If the routing goal is to optimize the bandwidth utilization, the routing problem is typically formulated as the minimum cost Steiner tree problem.

As real-time multimedia services are expected to be popular in the emerging BISDNs, to guarantee QoS requirements is an important issue. In this context, additional constraints may be imposed on the min-cost Steiner tree problem to guarantee that the sum of the delays along the path between the source and each destination is no greater than a predetermined value. We will refer to this problem as *delay-constrained minimum-cost multicast-routing problem with dynamic membership* (DDMP).

There have been some studies on computing the multicast tree for the multicast routing with dynamic membership but without delay considerations [2]. An important observation in these studies is that when a destination joins the current multicast, a good routing solution can be obtained simply by connecting the destination to the tree by a minimum cost path between 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 so that the delay bound is satisfied between the source and the joining destination.

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 current tree. In this paper, we consider a heuristic proposed in [1] (will be referred to as *Heuristic BG*) which can be interpretated as a pragmatic variant of theLagrangian relaxation method. Although the experiments reported in [1] are only preliminary to demonstrate its empirical efficiency, it has a unique multiplier update rule which appears to be particularly suitable for our purposes. It is based on a simple and pragmatic idea, not only rendering computational efforts minimal but also making implementation easy. In this paper, it is shown that the heuristic is, in fact, a polynomial algorithm.

In the literature, for purposes of evaluation of routing algorithms, the random graph called Waxman's network which is of standard use to simulate the real computer networks. we propose a generation scheme for the Waxman's network in which an asymptotic lower bound of the expected optimal cost can be computed as a function of network node size. Hence we can evaluate the quality of solutions a proposed algorithm in comparison to the optimal solutions over a wide range of network sizes for which the computation of the optimal costs is too excessive,

2 Heuristic BG

Consider 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^*, $d(P^*) \equiv \sum_{(i,j)\in P^*} d_{ij}$ is no greater than some bound Δ. This problem is known to be *NP-hard*.

Now we describe Heuristic BG. Let $\mathcal{A}$ be any shortest path algorithm.

Algorithm 2.1 *Heuristic BG [1]*

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 $S \leftarrow R$.*

[*] Dept. of Industrial Eng., Seoul National Univ., Seoul, Korea.

[†] School of Business, Chung-Ang University, An-sung-gun, Kyung-gi-do, 456-756, Korea, tel: +82-334-70-3214, fax: +82-334-675-1384, e-mail: sphong@cau.ac.kr

[‡] Dept. of Industrial Eng., Seoul National Univ., Seoul, Korea.

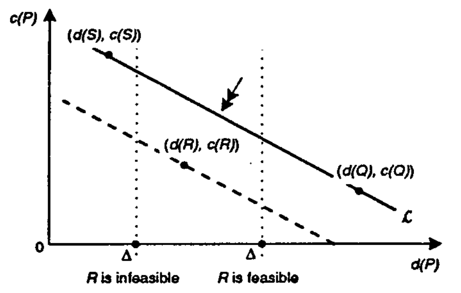

Figure 1: Main iteration of Heuristic BG.

Step 5: (Case 2) *If* $e(R) < \gamma$ *and* $d(R) > \Delta$*, then set* $Q \leftarrow R$*. Go to* Step 3.

The idea of Heuristic BG 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). Thus geometrically, the problem is to find a point lying in the left side of the dotted line (representing the delay bound) which is also as close as possible to the horizontal axis. The algorithm initially computes two s-t paths, one with the minimum cost and the other with the minimum delay. Clearly, these paths can be computed by applying a shortest path algorithm to c and d. These amount to two *push*es in the plane: the push vertically downward and the push horizontally to the left. Let the paths be Q and S respectively as in Figure 1.

In the next step, we push in the direction orthogonal to the straight line, $\mathcal{L}$ intersecting (the points corresponding to) Q and S. This means that two s-t paths are considered indifferent if they are on a straight line parallel to $\mathcal{L}$. Also this amounts to minimize the function $e(P) = \alpha d(P) + \beta c(P)$ which is a linear combination of two parameters, cost and maximum delays, where $\alpha = c(S) - c(Q)$ and $\beta = d(Q) - d(S)$. Notice that this is also can be done by applying shortest path algorithm. Now let R be an obtained path. Then there are two possible cases: R satisfies the delay bound or not, depending on its relative position to Δ as in Figure 1.

If R satisfies the delay bound, then in the next iteration we push in the direction orthogonal to the line intersecting Q and R. (See case 1 in Figure 2.) In other case, push direction is set to be orthogonal to the line intersecting R and S. (See case 2 in Figure 2.) Thus in the former case, because R satisfies the delay bound, we multiply larger weight, that is, to place more emphasis on the cost side in minimization, so that the push direction becomes more vertically downward than in the latter case. And vice versa. Thus the push direction (or multipliers) is updated based on three current solutions, Q, R and S. After R replaces S or Q depending on the cases, we repeat this until we get no improvement in e minimization.

The algorithm has quite pragmatic feature rendering the required iteration minimal. In fact it can be shown that the algorithm has a polynomial complexity.

Theorem 2.2 *Assume that for each solution P, the input sizes of $c(P)$ and $d(P)$ are bounded by a polynomial function $p(I)$ of the input size of the problem. Then Heuristic BG terminates in polynomial, $O(Max(\log C, \log D, \log n) + O(p(I))$ number of iterations, where C and D are the largest absolute values of two parameters, respectively, and $2n$ is the number of all parameters.*

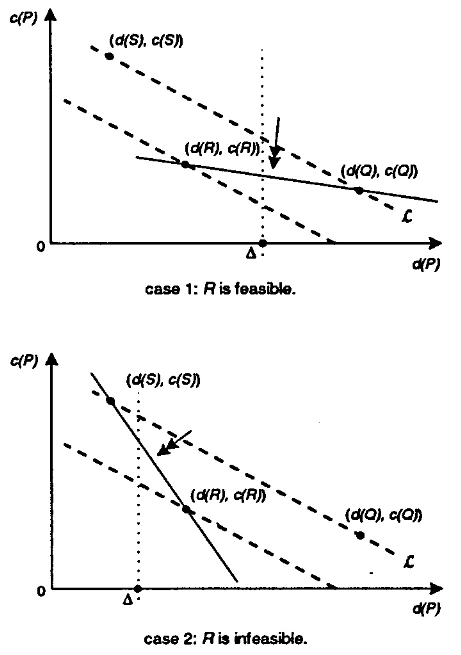

Figure 2: Adaptiveness of subsequent iterations.

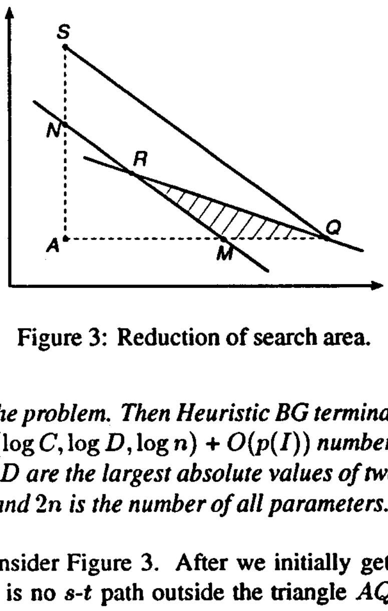

Figure 3: Reduction of search area.

Proof: Consider Figure 3. After we initially get Q and S we see that there is no s-t path outside the triangle AQS because of the minimality of Q and S with respect to c and d respectively. Similarly due to the minimality of R, there can be no path closer to the origin than the straight line intersecting M and N. Thus so far the search region has been reduced to the polyhedron $MQSN$. Then depending on the relative position of R to Δ, the direction of minimization is determined. Say the direction is orthogonal to the line intersecting Q and Rin the next iteration as R satisfies the delay bound Δ. Then we see that the search region is reduced further to QRM. Hence after the first main iteration involving e minimization the area of search region is reduced at least by half.

By an inductive argument, it is easily seen that this is also the case in any subsequent iteration. Thus search area reduces exponentially.

To establish the polynomial bound, it remains to show that before the termination, the area of the remaining search region can not get too small, namely no less than $2^{-p(I)}$, where p is a polynomial function and I is the input size of the problem.

To show this we can rely on the following well-known fact :

Theorem 2.3 *Let $A \in R^{m\times n}$. Suppose the polyhedron $\Pi = \{x|Ax \leq b\}$ is bounded and full-dimensional. Denote by μ be an element of A or b with the maximum input size. Then the volume of Π is no less than $2^{-4n^3\mu}$.*

By induction, it is easy to show that the remaining search region after each iteration is the polytope defined by three halfspaces:

$$\begin{aligned} \alpha_1 x + \beta_1 y &\geq \gamma_1 \\ \alpha_2 x + \beta_2 y &\geq \gamma_2 \\ \alpha_3 x + \beta_3 y &\leq \gamma_3, \end{aligned} \tag{2.1}$$

where, all the coefficients are nonnegative. Since the hyperplanes (straight lines) defining the halfspaces never coincide, the solution set, if not empty, necessarily has an interior except the case when the corresponding hyperplanes intersect at a single point. But then the algorithm immediately terminates in the following iteration as it can not achieve any improvement in e-minimization. Thus before the termination of the algorithm, a nonzero area of the polytope, by Theorem 2.3, can not be smaller than $2^{-O(I')}$, where I' is the maximum value among input sizes of the coefficients in Eq (2.1). But the coefficients of Eq (2.1) are either $c(S') - c(Q')$, $d(Q') - d(S')$, or $d(Q')c(S') - d(S')c(Q')$. By the assumption, $I' = O(p(I))$.

Hence the number of iterations is bounded by

$$\begin{aligned} & \log(\text{Area of} AQS/2^{-O(p(I))}) \\ \leq & \log(d(Q) - d(S))(c(S) - c(Q)) + O(p(I)) \\ \leq & \log d(Q)c(S) + O(p(I)) \\ \leq & \log(nC)(nD) + O(p(I)) \\ = & O(\text{Max}(\log C, \log D, \log n) + O(p(I)) \end{aligned} \tag{2.2}$$

Therefore, the theorem follows. □

3 Waxman's Network

The Waxman's network is a random graph frequently used for evaluation of multicast routing algorithm in the literature. [2]. 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,

$$\Pr(u, v) = \beta \exp \frac{-\delta(u, v)}{2\alpha L}, \tag{3.3}$$

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, longer edges have more chance to exist and hence the connectivity of the network is increased. By increasing β we can uniformly increase the density of the network instances.

For each instance, the members M, namely, the source r and the destinations D are randomly chosen from the nodes. 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 in terms of 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 also better off 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 path minimizes the cost will always have a maximum delay. So such a network is most difficult to solve. In our experiment, the routing algorithm is simulated on two groups of instances.

Theorem 3.1 *Suppose $L = \sqrt{kn}$ for some constant k. Then the expectation, $\bar{c}^*(n)$ of the optimum value of in the positively correlated instances is $\Omega(n)$. In the negatively correlated instances, on the other hand, $\bar{c}^*(n)$ is $\Omega(n^{\frac{3}{2}})$.*

Proof : Clearly for each instance, $\bar{c}^*(n)$ is not less than the number of members $\times$ the expected minimum edge cost. Since the number of members is proportional to n, the theorem will follow if we show that the expected minimum rectilinear distance of existing edges is $\Omega(1)$ and $\Omega(\sqrt{n})$ for the instances with positively and negatively correlated parameters, respectively.

In the case of instances of negatively correlated parameters, $c_{uv} = 4L - \delta(u, v) \times (1 + \omega)$ with ω a random number from $[0, 1]$. Hence the expected minimum cost is $4L - 1.5 \times$ the expectation of the maximum rectilinear distance of existing edges which is not greater than $2L$. Thus the expected value of a minimum cost is at least L hence $\Omega(\sqrt{n})$.

To complete the proof for the instances with positively correlated parameters we need the following lemma.

Lemma 3.2 *Suppose $L = \sqrt{kn}$. If n nodes are randomly chosen among $L \times L$ square lattice points of unit spacing. Then the expected value δ_{min} of the minimum rectilinear distance between pairs of nodes is $\Omega(1)$.*

Before proving Lemma 3.2, let's see why the expected minimum rectilinear distance of existing edges is $\Omega(1)$ for the instances with positively correlated parameters if Lemma 3.2 holds.

Recall that $c_{uv} = \delta(u, v) \times (1+\omega)$ with ω a random number from $[0, 1]$. Hence the expected minimum cost is $1.5 \times$ the expectation

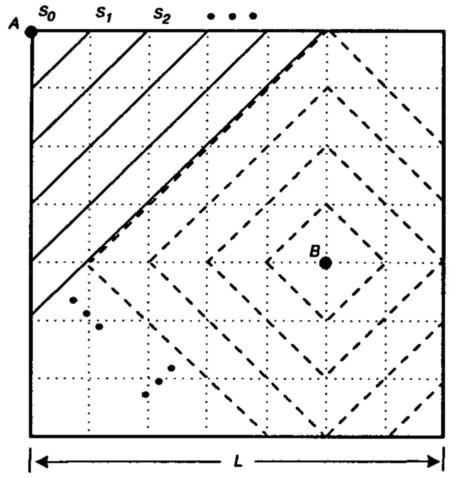

Figure 4: Equi-distance points w.r.t point A and B.

of the minimum rectilinear distance of existing edges. Between a node pair (u, v), the connecting edge of rectilinear distance $\delta(u, v)$ exists with the probability given by Eq (3.3) which is,

$$\Pr(u,v) = \beta \exp \frac{-\delta(u,v)}{2\alpha\sqrt{kn}}. \tag{3.4}$$

Thus obviously the minimum rectilinear distance of existing edges connecting node pairs is not less than the minimum rectilinear distance δ_{min} between node pairs. Thus it suffices to show that δ_{min} is $\Omega(1)$.

Proof of Lemma 3.2

As described above n nodes are randomly chosen among $L \times L$ square lattice points with unit spacing as Figure 4.

To compute δ_{min}, we fix the position of a node, say v arbitrarily chosen among the n nodes. Suppose v is on a corner point of the lattice as A in Figure 4. The lattice points on the solid parallel lines represents the set of points which are at the same rectilinear distances from v at A. Let these sets be $S_0, S_1, S_2, \cdots, S_{L-1}, S_L, S_{L+1}, \cdots, S_{2L}$ in the ascending order of distances. Then the sizes of the sets are$1, 2, 3, \cdots, L\ L+1\ L, \cdots, 1$, which is a concatenation of two sequences of equal differences. The probability that a node is among such points are $\frac{1}{L^2}, \frac{2}{L^2}, \frac{3}{L^2}, \cdots, \frac{L}{L^2}, \frac{L+1}{L^2}, \frac{L}{L^2}, \cdots, \frac{1}{L^2}$.

Now we compute the expected value, $\delta_{min}(A)$of the rectilinear distance of a nearest node from v at the point A. For a nearest node to be at a point in S_k, all $n-1$ nodes other than v must belong to $\cup_{i \ge k} S_i$ but some nodes need to be in S_k. Thus the probability that a nearest node from v to be in S_k $(k = 1, \cdots, 2L)$ is,

$$\begin{aligned} p_k &= \Pr\{\text{all the } n-1 \text{ points are in } \cup_{i \ge k} S_i\} \\ &\quad - \Pr\{\text{all the } n-1 \text{ points are in } \cup_{i \ge k+1} S_i\} \\ &= \left(1 - \sum_{i=0}^{k-1} |S_i|/L^2\right)^{n-1} - \left(1 - \sum_{i=0}^{k} |S_i|/L^2\right)^{n-1}, \end{aligned} \tag{3.5}$$

where, S_{2L+1} is defined to be empty set.

Thus the expected value of of the distance from v to a nearest point is,

$$\begin{aligned} &\textstyle\sum_{k=1}^{2L} kp_k \\ &= 1 \times \left(\left(1 - \sum_{i=0}^{0} \frac{|S_i|}{L^2}\right)^{n-1} - \left(1 - \sum_{i=0}^{1} \frac{|S_i|}{L^2}\right)^{n-1}\right) \\ &+2 \times \left(\left(1 - \sum_{i=0}^{1} \frac{|S_i|}{L^2}\right)^{n-1} - \left(1 - \sum_{i=0}^{2} \frac{|S_i|}{L^2}\right)^{n-1}\right) \\ &+3 \times \left(\left(1 - \sum_{i=0}^{2} \frac{|S_i|}{L^2}\right)^{n-1} - \left(1 - \sum_{i=0}^{3} \frac{|S_i|}{L^2}\right)^{n-1}\right) \\ &\cdots \\ &+(2L-1) \times \\ &\left(\left(1 - \sum_{i=0}^{2L-2} \frac{|S_i|}{L^2}\right)^{n-1} - \left(1 - \sum_{i=0}^{2L-1} \frac{|S_i|}{L^2}\right)^{n-1}\right) \\ &+2L \times \left(\left(1 - \sum_{i=0}^{2L-1} \frac{|S_i|}{L^2}\right)^{n-1}\right) \\ &= \left(1 - \sum_{i=0}^{0} \frac{|S_i|}{L^2}\right)^{n-1} + \left(1 - \sum_{i=0}^{1} \frac{|S_i|}{L^2}\right)^{n-1} \\ &\cdots + \left(1 - \sum_{i=0}^{2L-2} \frac{|S_i|}{L^2}\right)^{n-1} + \left(1 - \sum_{i=0}^{2L-1} \frac{|S_i|}{L^2}\right)^{n-1} \\ &\ge \left(1 - \sum_{i=0}^{0} \frac{|S_i|}{L^2}\right)^{n-1} \\ &= \left(1 - \tfrac{1}{L^2}\right)^{n-1} \\ &= \left(1 - \tfrac{1}{kn}\right)^{n-1} \\ &= \left(\left(1 - \tfrac{1}{kn}\right)^{kn}\right)^{\frac{1}{k}} \left(1 - \tfrac{1}{kn}\right)^{-1} \\ &\to e^{\frac{1}{k}} \end{aligned} \tag{3.6}$$

Thus, when a node is on the corner of the lattice, the rectilinear distance to a nearest point is $\Omega(1)$.

Let's consider the case where the node v is at a more general position like *e.g.* the lattice point B in Figure 4. The lattice points which are at the same rectilinear distances from B form the sets represented by broken lines. While in the previous case, the sequence of equequi-distanceidistance set sizes is a concatenation of two sequences of equal differences, in this case it consists of at most five different sequences of equal differences. To compute these sequences we need to elaborate the arguments by considering the relative position of v in the lattice.

But the point is that an essentially same arguments as Eq (3.6) also applies to this case to derive a asymptotical lower bound on the minimum distance and the bound is the same as in the previous case up to a constant factor. Therefore the expected value of minimum distance δ_{min} is again $\Omega(1)$. Hence the lemma follows. □

References

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

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