A Genetic Algorithm for Delay-Constrained Minimum-Cost Multicasting

Quan Sun

Institute of Operating Systems and Computer Networks

Technical University of Braunschweig

Bűltenweg 74/75, 38106 Braunschweig, Germany

Abstract

The problem of computing delay-constrained minimum-cost multicast trees is of great interest in the last few years. The best known solution to this problem is the Bounded Shortest Multicast Algorithm (BSMA), which has unfortunately very large computation time. In this paper, we propose a genetic algorithm for computing such delay-constrained low-cost multicast trees. The proposed algorithm is based on a bit-string encoding of selected Steiner nodes and an efficient deterministic delay-constrained multicast heuristic. Simulation results show that the proposed genetic algorithm can achieve a smaller average tree cost than BSMA with a much smaller running time for relatively large networks.

Keywords: Multicast, minimum-cost tree, delay constraint, multimedia, genetic algorithm

1 Introduction

Multicasting involves the transport of the same information from a sender to multiple receivers along a multicast tree. In the past, much work has been focused on algorithms for computing low-cost multicast trees so that network resources can be efficiently managed. Since the problem of computing minimum-cost multicast trees (called *Steiner trees*) in a network is NP-complete [8], most of the proposed algorithms are heuristic ones. Some of the well-known Steiner tree heuristics are the KMB heuristic [9], the RS heuristic [10] and the TM heuristic [11]. Recently, methods based on neural networks [13] and genetic algorithms [16] have also been proposed for computing low-cost multicast trees.

As more and more delay-sensitive networking applications such as video conferencing, tele-teaching and many other distributed multimedia applications appear, efficient computation of low-cost multicast trees satisfying a given *delay constraint* is of great interest. In the last few years, a lot of delay-constrained multicast routing heuristics have been proposed (see e.g. [1] [2] [3] [4] [17]). The simulation results given in [5] have shown, however, that most of the proposed algorithms either work too slowly or cannot compute delay-constrained multicast trees with low-costs.

Deterministic computation of low-cost delay-constrained multicast trees usually requires large amount of computation time. For this reason, methods based on neural networks or genetic computation may be of great help. To our knowledge, techniques based on artificial intelligence have until now not been reported for the delay-constrained multicast routing problem.

In this paper, we propose a genetic algorithm computing delay-constrained low-cost multicast trees. The paper will show that the given genetic algorithm can perform better than the best deterministic delay-constrained multicast routing algorithm proposed so far, namely the Bounded Shortest Multicast Algorithm (BSMA) [1], with a much smaller computation time. The rest of the paper is arranged as follows. The network model and problem definition are given in section 2. Section 3 describes the proposed genetic algorithm. The simulation results are presented in section 4. Section 5 concludes the paper.

2 Network Model and Problem Definition

A network is modeled as a connected, directed graph *G=(V,E)* where *V* is the set of network nodes and *E* is the set of links. Two non-negative real value functions are associated with each link e ($e \in E$): delay *D(e)* and cost *C(e)*. The link delay *D(e)* is the delay a data packet experiences on the corresponding link, and the link cost *C(e)* is a measure of the utilization of the corresponding link's resources. Links are asymmetrical, namely the cost and delay for the link *e=(i,j)* and the link $e' = (j, i)$ may not be the same.

Given a source node s ($s \in V$), a set of destination nodes S ($S \subseteq V - s$), a tree T ($T \subseteq G$) rooted at s and spanning all of the nodes in S (with all leaf nodes $\subseteq S$) is called a *multicast tree*. The cost (C_T) of the multicast tree T is defined as follows:

$$C_T = \sum_{e \in T} C(e) \tag{1}$$

The nodes in S are often called multicast *group members*, $|S|$ is the multicast group size. The nodes in V which are not in $\{s\} \cup S$ are called *Steiner nodes*. Let $P_T(s, v)$ denote the unique path from s to $v \in S$ in the multicast tree T, then the *delay-constrained minimum-cost multicast problem* can be formally expressed as follows:

For a network *G=(V,E)*, given a link delay function $D : E \rightarrow \Re^+$, a link cost function $C : E \rightarrow \Re^+$, a source node s ($s \in V$), a set of destination nodes S ($S \subseteq V - s$), and a delay tolerance $\triangle$, find a multicast tree T ($T \subseteq G$) rooted at s and spanning all of the nodes in S such that C_T is minimized and

$$\sum_{e \in P_T(s,v)} D(e) < \triangle \quad \forall v \in S \tag{2}$$

The choice of $\triangle$ depends on the applications. For most real-time video and audio applications, $\triangle$ should be less than 150 ms [12]. In this paper, we assume that the least-delay tree (T_l)

rooted at s and spanning all other network nodes satisfies $\triangle$. Namely, the choice of $\triangle$ satisfies

$$\sum_{e \in P_{T_l}(s,v)} D(e) < \triangle \quad \forall v \in V - s \tag{3}$$

where $P_{T_l}(s,v)$ is the least-delay path from s to v.

The delay-constrained minimum-cost multicast problem is shown to be *NP*-complete [2]. Among all the heuristic solutions for the delay-constrained minimum-cost multicast problem proposed so far, the Bounded Shortest Multicast Algorithm (BSMA) [1] is considered to have the best performance regarding the resulting tree cost [5]. BSMA is based on a feasible search optimization method which starts with the least delay tree and monotonically decreases the cost by iterative improvement of the delay-bounded tree. It uses *path switching* to improve the cost of the tree currently obtained. Path switching means that a path in tree T_i is replaced by a new path that is not in T_i, resulting in a new tree configuration T_{i+1}. The kth-shortest path algorithm [7] is used to find cheaper constrained paths for path switching. Although BSMA can achieve near-optimal cost performance, its computation time is very large [5].

In the following section, we propose a genetic algorithm for the delay-constrained minimum-cost multicast problem. Simulation results in Section 4 will show that the proposed algorithm has a much smaller computation time and can result in even lower tree costs than the algorithm BSMA.

Throughout this paper, we let $n=|V|$ and $m=|S|$.

3 The Proposed Genetic Algorithm

Genetic algorithms are search algorithms based on the mechanisms of genetic adaptation in biological systems [14] [15]. The algorithm maintains a *population* of *individuals* (*chromosomes*). Each individual corresponds to a specific solution to the given problem. The quality of an individual is judged by its *fitness* measure. A simple genetic algorithm works according to the following steps:

(*S1*) Generate an initial population of individuals.

(*S2*) Evaluate the fitness of every individual.

(*S3*) Produce a new *generation* of population based on the fitnesses of individuals in the current population (by using genetic operators like *crossover* and *mutation*).

(*S4*) Evaluate the fitness of every individual in the new generation.

(*S5*) Continue to execute Steps (*S3*) and (*S4*) until a given termination criterion is satisfied.

The genetic algorithm starts with an initial population of random individuals and simulates the process of evolution. After a number of generations, highly fit individuals will emerge corresponding to good solutions to the given problem. One of the most difficult tasks in designing a genetic algorithm lies in the representation (encoding) of individuals. The genetic encoding (representation) of an individual is called a *genotype*, and the corresponding physical appearance of an individual is called a *phenotype*. For a given genotype, the corresponding phenotype is computed by a *decoder*. Another difficult task is how to design such a decoder. Note that fitness is defined in terms of phenotype.

In the following, we first define and/or describe the concepts of genotype, decoder and fitness in the context of solving our delay-constrained minimum-cost multicast problem before presenting the genetic algorithm.

3.1 Genotype

The genotype encoding in our genetic algorithm solution for the delay-constrained multicast problem is similar to that used in the genetic algorithm for the Steiner Problem in a graph proposed in [16]. The genotype specifies a set of selected Steiner nodes. The selected Steiner nodes are specified by a bit-string in which each bit corresponds to a specific node. Assume a fixed numbering 0, 1, ..., (n-m-2) of the Steiner nodes V-S-$\{s\}$, a genotype is the set:

$$\{g(0),\ g(1),\ \ldots,\ g(n\text{-}m\text{-}2)\}$$

where $g(i) \in \{0,1\}$, i=0, 1, ..., (n-m-2). The set of Steiner nodes $S_T \subseteq V - S - \{s\}$ specified by the genotype is $S_T = \{v_i \in V \,|\, g(i) = 1, i = 0, 1, ..., n - m - 2\}$.

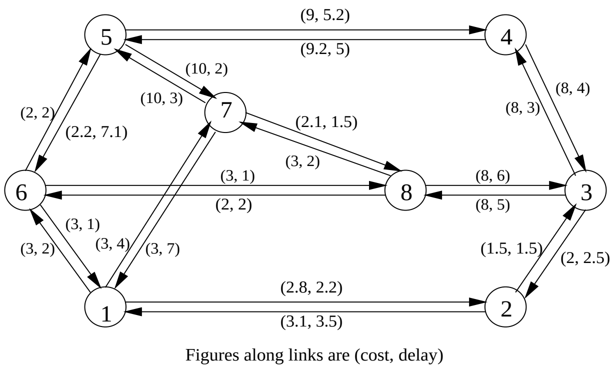

Figure 1: Example graph

Example: In Fig. 1, if the source node $s = 7$ and destination node set $S = \{2,3,5\}$, and we assume a fixed numbering 0,1,2,3 of the Steiner nodes V-S-$\{s\}$=$\{1,4,6,8\}$, then the genotype is the set: $\{g(0),\ g(1),\ g(2),\ g(3)\}$. A genotype set of $\{0,1,1,0\}$ specifies the Steiner nodes S_T=$\{4,6\}$.

3.2 Decoder

The genotype of an individual must be decoded so that the fitness function can be computed. The proposed genotype decoder is based on a deterministic delay-constrained multicast routing algorithm (CKMB) proposed in [17]. CKMB works like the Steiner tree heuristic KMB proposed in [9]. We first briefly describe CKMB, referring the reader to [17] for more details. Given a source node s, a set of destination nodes S ($s \notin S$) and a delay bound $\triangle$ for a directed network *G(V,E)* with link costs C and link delays D, the CKMB(s, S, $\triangle$, C, D) procedure roughly takes the following steps:

(a) Construct a directed graph G' with the nodes being $\{s\} \cup S$. The link (v_i, v_j) in G' is a path from v_i to v_j in *G(V,E)* satisfying a delay constraint that is equal to or less than $\triangle$.

(b) Construct a delay-constrained spanning tree T' of G'.

(c) Convert the links of T' to the corresponding paths they represent and remove possible loops to form a solution.

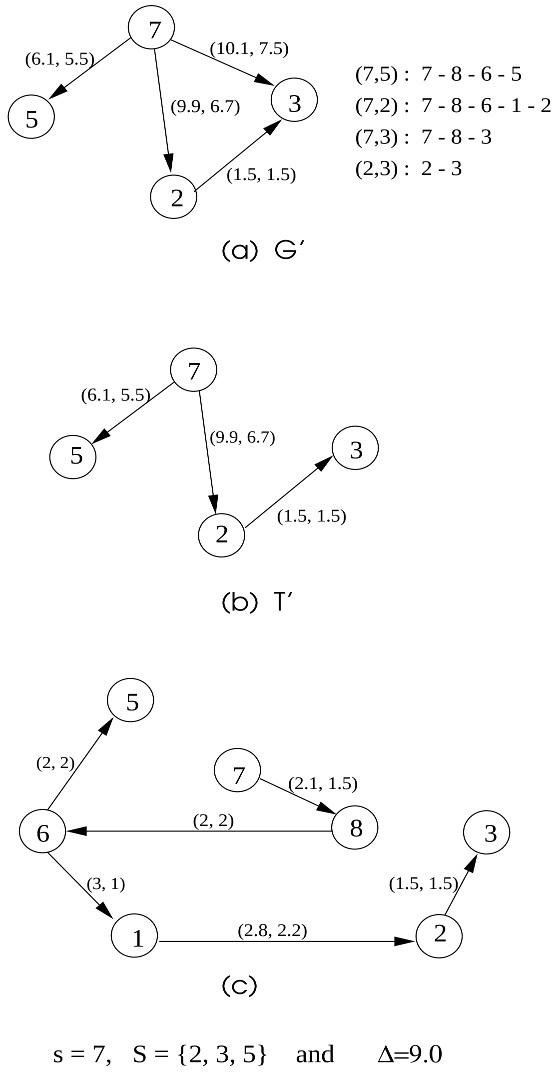

Figure 2: The application of CKMB to the example graph

Fig. 2 illustrates the application of the CKMB to the graph shown in Fig.1, with s=*7*, S={*2,3,5*} and $\triangle$=9.0. In Fig. 2 (a), a directed graph G' consisting of the nodes {2,3,5,7} is constructed. A link in this graph corresponds to a path in Fig. 1. For example, the link (7,5) in Fig. 2 (a) represents the path 7→8→6→5 in Fig. 1. In Fig. 2 (b), a delay-constrained spanning tree T' of G' is computed. Finally, the links of T' are converted to their corresponding paths computed in Fig. 2 (a), and a delay-constrained multicast tree is obtained in Fig. 2 (c).

It is shown in [17] that CKMB can always find a delay-constrained multicast tree in the network G if one exists, and the time complexity of CKMB is only O(mn^2). Besides, CKMB performs well regarding the resulting tree cost.

Remember that the genotype defined in the above subsection gives a set of Steiner nodes S_T. The proposed genotype decoder takes the following two steps:

(1) Call CKMB(s, $S \cup S_T$, $\triangle$, C, D) to compute a multicast tree T rooted at s, spanning all nodes in $S \cup S_T$ and satisfying the delay constraint $\triangle$.

(2) Prune those leaf nodes in T that are not in S.

Lemma 1: The decoder computes a multicast tree rooted at the source node s, spanning all the destination nodes in S and satisfying the delay constraint $\triangle$ under the condition of (3).

Proof: According to [17], Step (1) of the decoder can always compute a delay-constrained multicast tree T rooted at s and spanning all nodes in $S \cup S_T$ if the least-delay path from s to every node in $S \cup S_T$ satisfies $\triangle$. Therefore, under (3), the decoder can always compute a delay-constrained multicast tree with the source node s and destination nodes $S \cup S_T$.

In Step (2) of the decoder, those leaf nodes in T that are in S_T are pruned, which means that the leaf nodes of the resulting tree are only nodes in S. $\square$

Because the procedure CKMB(s, S, $\triangle$, C, D) has a time complexity of O(mn^2), the computational complexity of the decoder is therefore O($(m + |S_T|)n^2$).

In order to reduce the decoding time in the evolution process of the genetic algorithm, the genotype decoder can be divided into two sub-procedures: PREPARATION() and DECODING(). The PREPARATION() constructs a directed graph G_0' with the nodes being V by using the method described in CKMB (namely by executing Steps 1 - 3 of CKMB with S set to be V-s (see [17]) for detail). The link (v_i, v_j) in G_0' is a path from v_i to v_j in *G(V,E)* satisfying a delay constraint. The DECODING() takes the following steps:

(A) Construct a directed graph G_1' with the nodes being $s \cup S \cup S_T$ from G_0' by deleting nodes $V - s - S - S_T$ and the links connecting these nodes in G_0'.

(B) Construct a delay-constrained spanning tree T_1' of G_1'.

(C) Convert the links of T_1' to the corresponding paths they represent (generated in PREPARATION()), and remove possible loops.

(D) Prune those leaf nodes in T that are not in S.

From [17], it is easy to infer that the computational complexities of the procedures PREPARATION() and DECODING() are O(n^3) and O(n^2) respectively. In Subsection 3.4, we will see that in the proposed genetic algorithm the procedure PREPARATION() only needs to be executed once for decoding all genotypes.

3.3 Fitness Measure

The genetic algorithm maintains a population of individuals: $\Phi = \{\phi_0, \phi_1, ..., \phi_{N-1}\}$ where ϕ_i is an individual and N is the population size. Let $C_T(i)$ denote the cost of the tree obtained in decoding the genotype of the individual ϕ_i. Furthermore, let

$$C_{TM} = max\{C_T(j)\} \quad for\ j = 0, 1, ..., N-1 \tag{4}$$

The fitness of the individual ϕ_i is computed as follows:

$$F(\phi_i) = 1 - \frac{C_T(i)}{C_{TM}} \tag{5}$$

A large value of $F(\phi_i)$ ($0 \leq F(\phi_i) < 1$) corresponds to an individual with good performance.

3.4 The Genetic Algorithm (GA)

The proposed genetic algorithm is shown in Fig. 3. In Fig. 3, the PREPARATION() procedure is first executed. The initial population Φ_c is constructed from randomly generated individuals by the routine *Generate*(). Throughout the simulation the population size, namely the number of individuals $N = |\Phi_c|$, is kept constant. Routine *Evaluate*() decodes every genotype by calling

(1) PREPARATION();

(2) $\Phi_c \leftarrow$Generate();

(3) Evaluate(Φ_c);

(4) **repeat**

(5) $\quad \Phi_n = \emptyset$;

(6) $\quad$ **for** $i \leftarrow 1$ **to** $N/2$ **do**

(7) $\qquad$ select $\phi_1 \in \Phi_c,\ \phi_2 \in \Phi_c$;

(8) $\qquad \{\phi_1^c, \phi_2^c\} = crossover(\phi_1, \phi_2, P_c)$;

(9) $\qquad \psi_1 \ = \ mutation(\phi_1^c, P_m)$;

(10) $\qquad \psi_2 \ = \ mutation(\phi_2^c, P_m)$;

(11) $\qquad \Phi_n \leftarrow \Phi_n \cup \{\psi_1, \psi_2\}$;

(12) $\quad$ **end for**

(13) $\quad \Phi_c \leftarrow \Phi_n$;

(14) $\quad Evaluate(\Phi_c)$;

(15) **until** StopCriterion;

(16) Output the individual with the largest fitness in Φ_c;

Figure 3: Outline of the genetic algorithm (GA)

the DECODING() procedure and computes the fitness of the corresponding individual in the given population according to (5). Lines 5 - 12 are a typical evolution process of a genetic algorithm [15], which generates a new population Φ_n from the current population Φ_c. First, Line 7 selects two individuals from Φ_c. Each of the two individuals ϕ_1 and ϕ_2 is selected with a probability proportional to its fitness. Then, two offspring ϕ_1^c and ϕ_2^c are generated using the *crossover* operator (the pair of the genotypes of ϕ_1 and ϕ_2 undergoes crossing over with a given probability P_c). Finally, a *mutation* operator is exerted on the genotypes of ϕ_1^c and ϕ_2^c with a small given probability P_m to generate two new individuals ψ_1 and ψ_2 for the new population Φ_n. The *crossover* and *mutation* are two fundamental operators used in genetic algorithms

[15]. The *crossover* operator involves partial exchange of information between two genotypes, while the *mutation* operator randomly changes each of the components of the genotype with a small given probability (please see [15] for detail). The genetic algorithm terminates when the StopCriterion is satisfied. In our simulation, the StopCriterion is that at least *N/2* individuals in the current population have a fitness value larger than 99% of the largest fitness among all the individuals in the current population and at least 5 generations are simulated.

Lemma 2: Each generation of the proposed genetic algorithm (Lines 5-14) has a computational complexity of O(Nn^2).

Proof: The genetic operator *crossover* or *mutation* requires O$(n - m - 1)$ time. Therefore, Lines 6-12 have only a computational complexity of O(Nn). The *Evaluate*() routine needs to execute DECODING() procedure N times, which results in a time complexity of O(Nn^2). $\square$

The total computational complexity of the proposed genetic algorithm is O(MNn^2), where M is the number of generations. In our simulation experiments described in the next section, we found that in many cases only a few generations could already give a result. Note that the proposed genetic algorithm will always give a delay-constrained multicast tree under (3).

4 Simulation Results

We used simulation experiments to compare the performances of the proposed genetic algorithm (GA) and the near-optimal Bounded Shortest Multicast Algorithm (BSMA) [1] regarding the resulting tree cost and the computation time. The simulation was conducted in the multicast routing simulation environment described in [5].

In the simulation, full duplex ATM networks with homogeneous link capacities of 155 Mbps (OC3) were used. The positions of the network nodes were kept in a rectangle of size 3000 * 2400 Km^2, roughly the area of the USA. A random graph generator based on Waxman's generator [8] was used to create networks. The generator first creates a list of nodes (at random locations) and then creates links between these nodes. The probability of a link to exist between two nodes u and v is given by

$$P(u,v) = \beta exp\frac{-l(u,v)}{L\alpha} \tag{6}$$

where $l(u,v)$ is the distance between u and v, and L is the maximum distance between any two nodes, α and β are parameters between (0,1]. A large value of α increases the ratio of the number of long links to short links, and a large β increases the average node degree of the network. The simulated networks were always connected [5] and had an average node degree of 4 through adjusting the parameters.

Each node in the created network represented a non-blocking ATM switch. Each link was assumed to have a small output buffer. The propagation speed through the links was taken to be two thirds the speed of light. The propagation delay was dominant under these conditions, and the queuing component was neglected when calculating the link delay $D(e)$. In the simulation, the link cost $C(e)$ was set to be the total currently reserved bandwidth on the link, which is a suitable measure of the utilization of both the link's bandwidth and its buffer space. For more information about the simulation environment, please see [5]. Note that in our simulation the delay bound $\triangle$ was set to be 50 ms, and all the generated networks satisfied (3).

For each run of the experiment, we generated a given number of nodes at random locations and a random set of links to interconnect these nodes using the graph generator described above. Random background traffic for each link was also generated. The equivalent bandwidth of each link's background traffic is a random variable uniformly distributed between Bmin and Bmax. As the difference between Bmin and Bmax increases, the asymmetry of the link loads also increases. In the simulation, we set Bmin=10 Mbps and Bmax=120 Mbps. The routing algorithm was applied to create a multicast tree for a random source node generating video traffic with an equivalent bandwidth of 0.5 Mbps and a group of randomly chosen destination nodes. We measured the cost of the created tree and the running time of the algorithm. The experiment was run repeatedly until confidence intervals of less than 5%, using 95% confidence level, were achieved for the mentioned measured quantities. All simulation experiments were run on a SUN SPARCstation 5.

In the simulation, the parameter settings for the GA algorithm were: population size N=20, crossover probability P_c=0.6 and mutation probability P_m=0.005. These values were used in all simulation experiments.

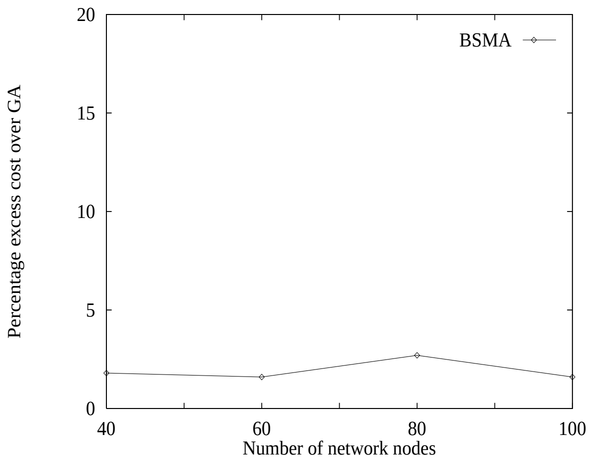

Figure 4: Percentage excess cost over GA versus number of network nodes, multicast group size equal to 5% of the number of network nodes

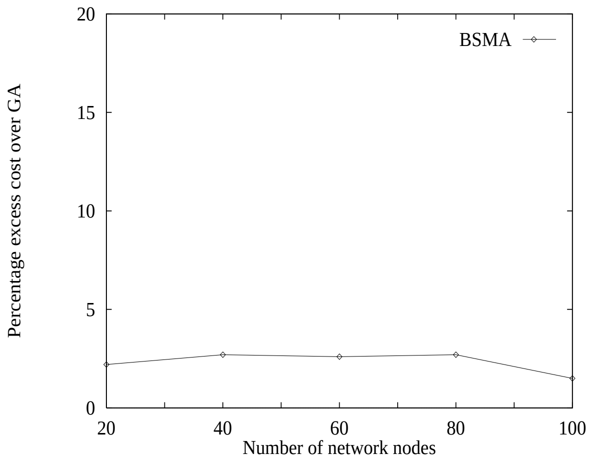

Figure 5: Percentage excess cost over GA versus number of network nodes, multicast group size equal to 15% of the number of network nodes

Fig. 4, 5 and 6 show the percentage excess tree cost of BSMA over GA versus number of

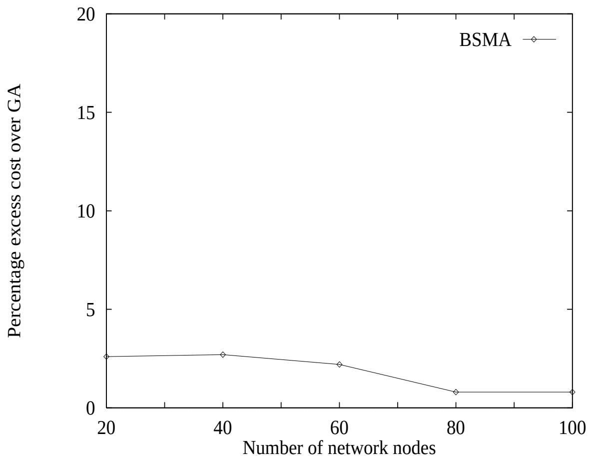

Figure 6: Percentage excess cost over GA versus number of network nodes, multicast group size equal to 30% of the number of network nodes

network nodes for different multicast group sizes. From these Figures, we see that the proposed genetic algorithm (GA) can result in a smaller average tree cost than the near-optimal delay-constrained multicast routing algorithm (BSMA).

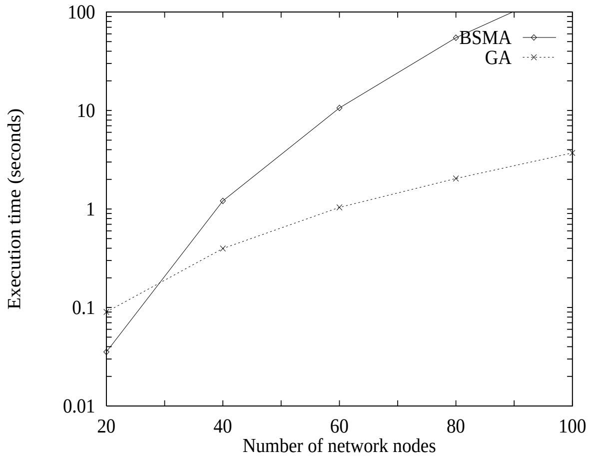

Figure 7: Execution time versus number of network nodes, multicast group size equal to 15% of the number of network nodes

A further encouraging result is that GA has a much better running time performance than BSMA for relative large networks. Fig. 7 shows a typical example of the execution time of BSMA and GA. It is easy to see that the execution time of BSMA is much larger than that of

GA for relative large networks. In fact, the execution time of BSMA is so large that we finally gave up running simulation experiments for networks with more than 100 nodes. From Fig. 7, we can easily see that the execution time growth rate of BSMA is also much larger than GA.

5 Concluding Remarks

The delay-constrained minimum-cost multicast problem is a typical optimization problem. Up to now, only deterministic heuristic solutions are known for this problem. The best known delay-constrained multicast algorithm, the Bounded Shortest Multicast Algorithm (BSMA) [1], that can compute near-optimal delay-constrained multicast trees, has unfortunately very large computation time.

In this paper, we proposed a genetic algorithm for constructing the delay-constrained minimum-cost multicast trees. The proposed genetic algorithm can even achieve a smaller average tree costs than BSMA with a much smaller running time for relative large networks.

One of the difficulties in designing a genetic algorithm is to encode the individuals (chromosomes). In our proposed algorithm, the encoded individual (genotype) specifies a set of Steiner nodes, which are represented by a bit-string. The genotype decoder is based on an efficient delay-constrained multicast heuristic [17]. It should be pointed out that performance of the genetic algorithm also depends on the choice of the parameters like population size, crossover probability and mutation probability. In our simulation, we have experimented with various values of these parameters before achieving the desired results.

Using an appropriate fitness function, the proposed genetic algorithm can also be applied to other multicast problems, such as computing delay-constrained multicast trees with low costs and low inter-destination delay variations [6]. In [18], we give fitness functions which can make tradeoffs between the multicast tree cost and the inter-destination delay variation or between the tree cost and the largest end-to-end delay.

The proposed algorithm works for the situation that the least-delay tree rooted at the source node and spanning all other network nodes satisfies the delay bound $\triangle$. Our future work includes modifying the proposed genotype decoder so that the proposed algorithm can also work for other delay bound values.

References

[1] M. Parsa, Q. Zhu, and J.J. Garcia-Luna-Aceves, "An Iterative Algorithm for Delay-Constrained Minimum-Cost Multicasting", *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, pp. 461-474, 1998.

[2] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicasting for Multimedia Applications", in: *IEEE INFOCOM'92*, pp. 2078-2085, 1992.

[3] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels", Technical Report TR-94-024, Tenet Group, Department of EECS, University of California at Berkeley, 1994.

[4] A.G. Waters, "A new heuristic for ATM multicast routing", *2nd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, 1994.

[5] 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*, vol. 15, no. 3, pp. 332–345, 1997.

[6] G.N. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints", *IEEE Journal on Selected Areas in Communications*, vol. 15, no.3, pp. 346-356, 1997.

[7] E. Lawler, *Combinational Optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1976.

[8] B.M. Waxman, "Routing of Multipoint Connections", *IEEE Journal of Selected Areas in Communications*, vol. 6, no. 9, pp. 1617-1622, 1988.

[9] L. Kou, G. Markowsky and L. Berman, "A fast algorithm for Steiner trees", *Acta Informatica*, vol. 15, pp. 141-145, 1981.

[10] V. Rayward-Smith, "The Computation of Nearly Minimal Steiner Trees in Graphs", *International Journal of Mathematical Education in Science and Technology*, vol. 14, no. 1, pp. 15-23, 1983.

[11] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs", *Mathematica Japonica*, vol. 22, no. 6, pp.573-577, 1980.

[12] S. Rampal and D.S. Reeves, "An Evaluation of Routing and Admission Control Algorithms for Multimedia Traffic", *Computer Communications*, vol. 18, no. 10, pp. 755-768, 1995.

[13] E. Gelenbe, A. Ghanwani, and V. Srinivasan, "Improved Neural Heuristics for Multicast Routing", *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 2, pp. 147-155, 1997.

[14] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[15] D. E. Goldberg, *Genetic Algorithms in search, optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[16] H. Esbensen, "Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm", *Networks*, vol.26, pp. 173-185, 1995.

[17] Q. Sun and H. Langendőrfer, "An Efficient Delay-Constrained Multicast Routing Algorithm", *Journal of High-Speed Networks*, vol. 7, no. 1, pp.43-55, 1998.

[18] Q. Sun, *Routing for Supporting Real-Time Applications in High-Speed Networks*, Dissertation, Institute of Operating Systems and Computer Networks, Technical University of Braunschweig, Bueltenweg 74/75, 38106 Braunschweig, Germany, Jan. 1999.