A Method for QoS Multicast Routing Based on Genetic Simulated Annealing Algorithm

Bo Peng and Lei Li

Graduate School of Engineering, Hosei University,
Koganei, Tokyo 184-8584 Japan
bo.peng.8k@stu.hosei.ac.jp, lilei@hosei.ac.jp

Abstract

With the increasing demand of multimedia applications, efficient and effective support of quality of service (QoS) has become more and more essential. A multicast communication creates a distribution tree structure, on which a multicast source sends a single copy of data to a group of receivers instead of sending a separate copy of the data to each receiver as in a unicast communication. In this paper, we study the bandwidth, delay, delay jitter, and packet loss ratio constrained least-cost multicast routing problem which is known to be NP-complete, and present an adaptive genetic simulated annealing algorithm (AGSAA) to solve the QoS multicast routing problem. The simulation results show that this algorithm has fast convergence and excellent cost performance.

Keywords: *Quality of Service, Multicast routing, Genetic algorithm, Simulated annealing algorithm*

1. Introduction

As the technology and popularity of the Internet have grown, multicast based applications have pervasive presence and influence in wide−area networks. For example, multicast has been used to transport real-time audio and video for news, entertainment, and distance learning. For applications involving group communication, multicast is more efficient than unicast as a multicast source sends a single copy of data to a group of receivers instead of sending a separate copy of the data to each receiver as in a unicast communication.

With the high demand of fast and better quality of services (QoS), a number of rigid QoS criteria, such as delay, bandwidth, delay jitter, and packet loss ratio, have been considered. Multicast routing algorithms capable of satisfying the QoS requirements of real-time applications will be essential for the high-speed information networks. The main goal of multicast routing algorithm is to minimize the communication resources used by the multicast session. This is achieved by minimizing the cost of the multicast tree, which is the sum of the costs of the links in the multicast tree.

The least-cost multicast routing problem is known to be NP-Complete. Most of the multicast routing algorithms use heuristic algorithms, such as the Kompella-Pasquale-Polyzos (KPP) heuristic [2], the Bounded Shortest Multicast Algorithm (BSMA) heuristic [3] and others. However, the simulation result has shown that most of the proposed heuristic algorithm either work too slowly or cannot compute QoS constrained multicast trees with low costs. Since deterministic heuristic algorithms for QoS constrained multicast routing are usually very slow, methods based on computational intelligence such as neural networks, genetic algorithms (GA) and ant colony optimization algorithms (ACO) have been proposed for solving the QoS multicast routing problem. In the field of computational intelligence, GA-based algorithms have emerged as powerful tools for solving NP-complete constrained optimization problems. Several GA-based algorithms [1] [4-12] have been proposed for solving the QoS multicast routing problem. Xiang et al. [1] have proposed a GA-based algorithm for QoS multicast routing in general case. This algorithm adopts the one-dimensional binary code as the encoding scheme of GA, but this encoding scheme makes the algorithm too complicate. Wang et al. [7] have proposed an efficient GA-based algorithm for delay-constrained least-cost multicast routing problem. They have used a tree data structure for genotype representation, but this algorithm is lack of local search ability.

Aiming to those shortage, this paper presents a new adaptive genetic simulated annealing algorithm (AGSAA) for QoS multicast routing. This algorithm combines GA and SA adequately. The SA has the character of probability-sudden-jump so as to be able to escape from the local optimum and approach the global minimum. The character of SA is used to avoid the premature convergence of GA. GA has the character of parallel processing and high convergence speed. The character of GA is used to avoid the lower convergence speed of SA [15].

2. Problem Description and Formulation

A network can be expressed as a weighted graph $G = (V, E)$, where V is the set of nodes representing routers, E is the set of links that connect the routers. Let $P_T(s, d_i)$ be the route from source node $s \in V$ to destination node d_i. $T(s, M)$ is a multicast tree. $M \in \{V - \{s\}\}$ is destination nodes set in multicast tree.

Let R^+ be non-negative real numbers set. For each link $e \in E$, five non-negative real value functions can be defined: cost function $Cost(e): E \rightarrow R^+$, delay function $Delay(e): E \rightarrow R^+$, bandwidth function $Bandwidth(e): E \rightarrow R^+$, delay jitter function $Delay\_jitter(e): E \rightarrow R^+$, packet loss ratio function $Packet\_loss(e): E \rightarrow R^+$.

The total cost of multicast tree $T(s, M)$ is defined as the sum of the cost of all links in that tree and can be given by

$$cost\big(T(s, M)\big) = \sum_{e \in T(s,M)} Cost(e) \tag{1}$$

$P_T(s, d_i)$ is the routing path between source node s and destination node d_i of multicast tree $T(s, M)$, which has following relations:

$$Delay\big(P(s,d)\big) = \sum_{e \in P_T(s,d_i)} Delay(e) \tag{2}$$

$$Bandwidth\big(P(s,d)\big) = min_{e \in P_T(s,d_i)}\{Bandwidth(e)\} \tag{3}$$

$$Packet\_loss\big(P(s,d)\big) = 1 - \prod\nolimits_{e \in P_T(s,d_i)}[1 - Packet\_loss(e)] \tag{4}$$

$$Dealy\_jitter\big(P(s,d)\big) = \sum_{e \in P_T(s,d_i)} Delay\_jitter(e) \tag{5}$$

The QoS multicast routing problem can be described as follows: Given network graph G, a source node s, a set of multicast destination nodes M, and the delay constraint D_{max}, delay jitter constraint DJ_{max}, bandwidth constraint B_{min} and packet loss ratio constraint PL_{max}. The problem is defined as minimization of the cost function $cost(T(s,M))$ subject to the following conditions [16]:

1. $Delay(P_T(s,d_i)) \leq D_{max}$
2. $Bandwidth(P_T(s,d_i)) \geq B_{min}$
3. $Dealy\_jitter(P_T(s,d_i)) \leq DJ_{max}$
4. $Packet\_loss(P_T(s,d_i)) \leq PL_{max}$

3. Genetic Algorithm

Genetic algorithm (GA) is a part of evolutionary computing. Genetic Algorithms were first proposed by Holland (1975) [22] and more recently reviewed and enhanced by Goldberg (1989) [23], Forrest (1993) [24] and many others. The genetic algorithm is an optimization technique derived from the principles of evolutionary theory. It has been applied to a myriad of optimization problems, such as the function optimization, traveling salesman problem, neural network optimization and others. It is also an efficient search method that has been used for path selection in networks.

An implementation of a genetic algorithm begins with a population of chromosomes (individuals). The evolution of individuals is based on the laws of natural selection and genetic information recombination within the population. A "fitness" value is assigned to each member of the population by a fitness function that gives a measure of the solution quality. The fitter chromosomes are then selected to produce offspring for the next generation in such a way that those chromosomes which represent a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions.

The main operators of genetic algorithm are selection, crossover and mutation. The genetic operators give each individual the chance of optimization and ensure the evolutionary tendency with the selection mechanism of survival of the fittest. Crossover simply combines the symbol strings of two parents to form two new individuals with different chromosome string that inherits solution characteristics from both parents. However, crossover cannot produce information that doesn't already exist within the population. Mutation covers this need by injecting new information in the produced chromosome string. The injection is done by randomly altering symbols of the chromosome [13].

4. Simulated Annealing Method

Simulated annealing algorithm (SA) is based on the annealing process in the physics of solids. It is observed that a metal body heated to high temperature cools slowly and tends to a state with the least internal energy. The goal is to bring the system, from an initial state, to a state with the minimum internal energy. SA regards the optimization problem as a physical system and the value of the objective function as its internal energy. With this analogy, annealing is the process of determining a solution with the least value of the objective function [14, 20]. The flowchart of the SA is shown in Figure 1.

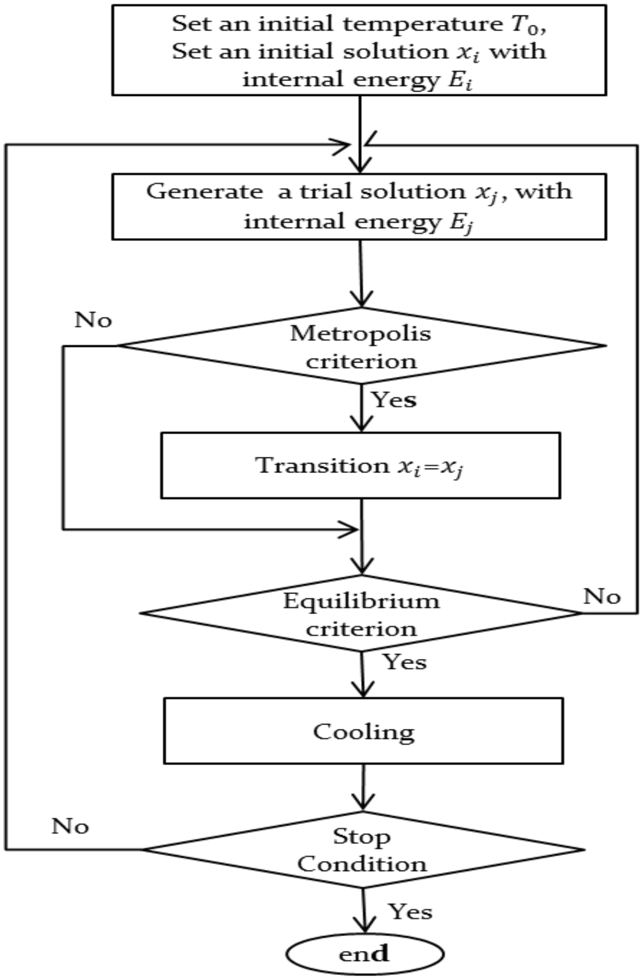

Figure 1. Flowchart of the Simulated Annealing Algorithm

5. The Proposed AGSAA Algorithm

5.1. Pre-processing Phase

Firstly, we concern about the bandwidth constraint. We delete all the links, which their bandwidth are less than the minimum required bandwidth.

5.2. Initial Population

The creation of the initial population in this paper is based on Dijkstra kth shortest path algorithm.

For each destination node $d_i \in M$, we compute least-cost paths from s to d_i by using Dijkstra kth shortest path algorithm to construct a candidate paths set. Let Q_i be the candidate paths set for destination node d_i:

$$Q_i = \{P_i^{\,1}, \dots, P_i^{\,j}, \dots, P_i^{\,k}\} \tag{6}$$

where $P_i^{\,j}$ is the jth path for destination node d_i. In this paper, we select $k = 20$ in our experiments. For example, a topology of a multicast network is shown in Figure 2, and the cost of each link is defined. Node 1 is source node, and destination nodes set is (4, 6, 7, 8). Table 1 shows the candidate paths set for destination node 7.

A multicast tree $T(s, M)$ is encoded as an array of $m = |M|$ elements, where each element is a path from source node s to a destination node $d_i \in M$, i.e., $T(s, M) = \{p_1, p_2, \dots, p_m\}$, where $p_i = p(s, d_i)$, $d_i \in M$, p_i is a path selected from the candidate paths set of destination node d_i. The relation between chromosome, gene, and candidate paths set is shown in Figure 3.

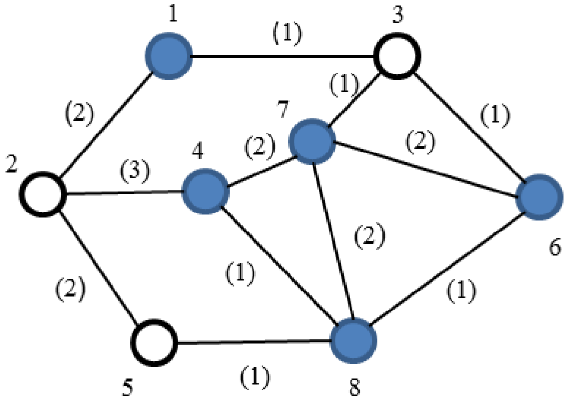

Figure 2. Topology of a Multicast Network

Table 1. Example of Candidate-paths-set for Node 7

Route number	Route list	Route cost
1	(1,3,7)	2
2	(1,3,6,7)	4
3	(1,3,6,8,7)	5
...	(1,…,7)	…
k	(1,…,7)	…

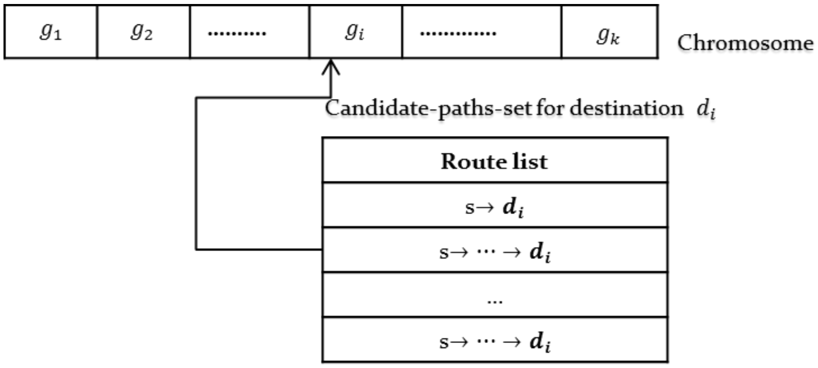

Figure 3. Chromosome Structure

According to the feature of the candidate paths algorithm, we choose a path from every candidate paths set, and the chosen paths can form a multicast tree, which covers all the destination nodes in network. Therefore, we can use the multicast tree to describe the chromosome structure. This process must be called pop_size times to create an initial population, where pop_size is the size of population.

5.3. Fitness Function

In this paper, we define the fitness function for each individual, the multicast tree $T(s,M)$, using the penalty function, as follows:

$$f(T(s,M)) = f_c \times f_d \times f_{dj} \times f_{pl} \qquad (7)$$

$$f_c = \frac{\alpha}{cost\big(T(s,M)\big)} = \frac{\alpha}{\sum_{e\in T(s,M)} Cost(e)} \qquad (8)$$

$$f_d = \prod_{d_i\in M} \delta_d(Delay(p(s,d_i)) - D_{max}) \qquad (9)$$

$$f_{dj} = \prod_{d_i \in M} \delta_{dj}(Delay\_jitter(p(s,d_i)) - DJ_{max}) \tag{10}$$

$$f_{pl} = \prod_{d_i \in M} \delta_{pl}(Packet\_loss(p(s,d_i)) - PL_{max}) \tag{11}$$

$$\delta_d = \begin{cases} 1\,, & T \le 0 \\ r_d\,, & T > 0 \end{cases} \tag{12}$$

$$\delta_{dj} = \begin{cases} 1\,, & T \le 0 \\ r_{dj}\,, & T > 0 \end{cases} \tag{13}$$

$$\delta_{pl} = \begin{cases} 1\,, & T \le 0 \\ r_{pl}\,, & T > 0 \end{cases} \tag{14}$$

where α is a positive real coefficient, $\delta(T)$ is the penalty function. When the individual satisfies the QoS constraint, its value is 1, or else $r\,(0 < r < 1)$. The value of r determines the degree of penalty. In this algorithm, we select $r_d = r_{dj} = r_{pl} = 0.5$ in our simulation experiments.

5.4. Selection

Selection operation in this paper adopts roulette wheel selection method and best individual preservation strategy. The probability P_i that an individual is selected is given by:

$$P_i = \frac{f(i)}{\sum_{i=1}^{N_{pop}} f(i)} \tag{15}$$

where $f(i)$ is the fitness value of the individual.

5.5 Crossover and Mutation Operation with Adaptive Probability

Crossover and mutation probability play an important role in GA, and directly affect the convergence of the algorithm. Therefore, the improved adaptive crossover and mutation probability is adopted in this paper [17] [19]. In the improved adaptive genetic algorithm, the crossover probability p_c and the mutation probability p_m, are varied depending on the fitness values of the solutions. Crossover probability p_c and mutation probability p_m, can be expressed as follows:

$$p_c = \begin{cases} p_{c1} - \frac{(p_{c1}-p_{c2})(f'-f_{avg})}{f_{max}-f_{avg}} & f' \ge f_{avg} \\ p_{c1} & f' < f_{avg} \end{cases} \tag{16}$$

$$p_m = \begin{cases} p_{m1} - \frac{(p_{m1}-p_{m2})(f_{max}-f)}{f_{max}-f_{avg}} & f \geq f_{avg} \\ p_{m1} & f < f_{avg} \end{cases} \tag{17}$$

where f_{avg} is the average fitness function value in current generation population, f_{max} is the maximum fitness function value in current generation population, f' is the larger fitness function value in the two crossover individuals, and f is the fitness function value of the mutation individual.

In the improved adaptive genetic algorithm, low values of p_c and p_m are assigned to high fitness solutions, while low fitness solutions have very high values of p_c and p_m. The adaptive crossover and mutation probability can not only improve the convergence rate of the GA, but also prevent the GA from getting stuck at a local optimum.

5.5.1 Crossover Operation: In this algorithm, two multicast trees, $T_F\ (s, M)$ and $T_M\ (s, M)$, are randomly selected as parents and the crossover operation produces an offspring $T_O(s, M)$. The processing of crossover operation is shown as follows:

step 1 Randomly select two individuals from the population according to the crossover probability p_c.

step 2 Select the path with higher value of fitness function between the parents for each destination node $d_i \in M$ and copy them to the offspring $T_O(s, M)$ directly.

step 3 This process is repeated $pop\_size - 1$ times, where pop_size is the size of the initial population.

The pseudo code is shown as follow:

```
Procedure crossover (parent1, parent2)
{
 for i=1 to lengthchrom    // lengthchrom is length of chromosome
      if (parent1 [i] > parent2 [i])
           offspring [i]=parent1[i];
      else
           offspring [i]=parent2 [i];
     return offspring;
}
```

In this paper the crossover operator selects the paths with higher value of fitness function between two parents for quicker convergence of the genetic algorithm.

5.5.2. Mutation operation: The mutation operation is performed according to the probability of mutation p_m. First, a mutation gene g_i is selected randomly. Then, the gene g_i is replaced with a new candidate path p_i, which is selected randomly from the candidate paths set of the destination node d_i.

5.6 Simulated Annealing Algorithm

In this paper, genetic algorithm and simulated annealing algorithm are combined to solve QoS multicast routing problem, where the cost function is just the objective function and the cost value just simulates the internal energy. Metropolis [20] was the first to propose an SA algorithm to simulate this process. SA algorithms approach optimization problems by randomly generating a neighboring solution, and then making successive random modifications.

A neighboring solution is created by parent and accepted with probability p of the Metropolis criterion. The calculation formula of p is

$$p(x_i \to x_i') = \begin{cases} 1 & f(x_i') \geq f(x_i) \\ exp\left(-\frac{\Delta f_{x_i x_i'}}{t_k}\right) & f(x_i') < f(x_i) \end{cases} \tag{18}$$

where, $\Delta f_{x_i x_i'} = f(x_i) - f(x_i')$, $f(x_i)$ is the fitness value of current solution, $f(x_i')$ is the fitness value of neighboring solution. When $f(x_i') \geq f(x_i)$, namely, the neighboring solution is better than current solution, neighboring solution individual will be copied into the population of the next generation. When $f(x_i') < f(x_i)$, the neighboring solution is accepted by probability $exp\left(-\Delta f_{x_i x_i'}/t_k\right)$.

In an iteration, the procedure for generating and testing the neighboring solution is repeated for a specified number of trials. The last accepted neighboring solution is then taken as the starting population for genetic algorithm in the next generation. In the next generation, the temperature is reduced according to:

$$t_k = \alpha \times t_0 \tag{19}$$

where t_0 is the initial temperature, t_k is the temperature at the *kth* iteration, and α is the temperature reduction factor. In this paper, we select $\alpha = 0.95$ in our experiments. The solution process of SA algorithm continues until the maximum number of iterations is reached or the optimum solution is found.

5.6.1. Initial Temperature and Initial Solution: The initial temperature t_0 should be large enough to accept all neighboring solutions with high probability and ensure excellent convergence. In this paper, we select $t_0 = 9000$. The population generated by mutation operation of genetic algorithm in the current generation is taken as the initial population of simulated annealing algorithm.

5.6.2. Creating of Neighboring Solution: For each individual, a gene g_i is selected randomly. Then, the gene g_i is replaced with a new path p_i, which is selected randomly from the candidate paths set Q_i of the destination node d_i (i.e. $Q_i = \{P_i{}^1, \dots, P_i{}^j, \dots, P_i{}^k\}$). Then this new tree is the neighboring solution. The process of creating a neighboring solution is shown in Figure 4.

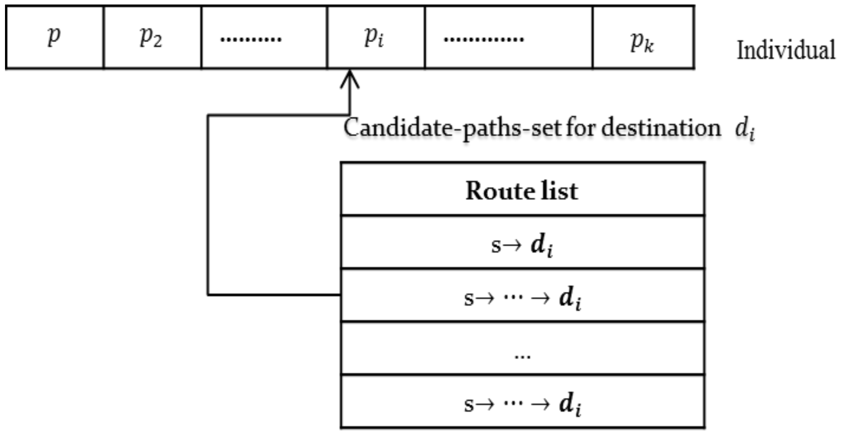
p
p_2
....
p_i
....
p_k
Individual
Candidate-paths-set for destination d_i
Route list
s→ $\boldsymbol{d_i}$
s→ ⋯ → $\boldsymbol{d_i}$
...
s→ ⋯ → $\boldsymbol{d_i}$

Figure 4. Creating of Neighboring Solution

The flowchart of proposed algorithm AGSAA is shown in Figure 5.

6. Simulations and Discussions

In this section, we have used the simulation results to compare the performance of the proposed AGSAA algorithm with the Wang-GA-based-algorithm [7].

The experimental simulations are achieved by MATLAB 7.11.0 to implement both algorithms. All simulation experiments are run on a personal computer (Intel Core i7 Processor (3.4GHz), 8.00GB RAM). The network topology is created by Salama graph generator in this experiment [18]. The average degree of each node in the random generated graphs is 4. The degree is close to the average node degree of current networks. The multicast group is randomly selected in the graph. The numbers of network nodes are 20, 40, 60, 80 and 100 nodes respectively, and the size of multicast group is 5.

In this simulation the following parameter settings are achieved: maximum number of generations $max.generation = 300$, size of population $pop\_size = 60$, crossover probability $p_{c1} = 0.9$, $p_{c2} = 0.6$, mutation probability $p_{m1} = 0.1$, $p_{m2} = 0.001$, initial temperature $t_0 = 9000$, temperature reduction factor $\alpha = 0.95$. Dijkstra *kth* shortest path algorithm was designed to automatically generate the shortest 20 paths for each destination node.

The least-cost QoS multicast routing problem is known to be NP-complete. For a large-scale network, it is time-consuming to obtain the optimal solution to the least-cost QoS multicast routing problem. This problem can be overcome by setting an appropriate iteration time of the genetic algorithm. In this way, we can obtain a near global optimal solution within a reasonable time limit.

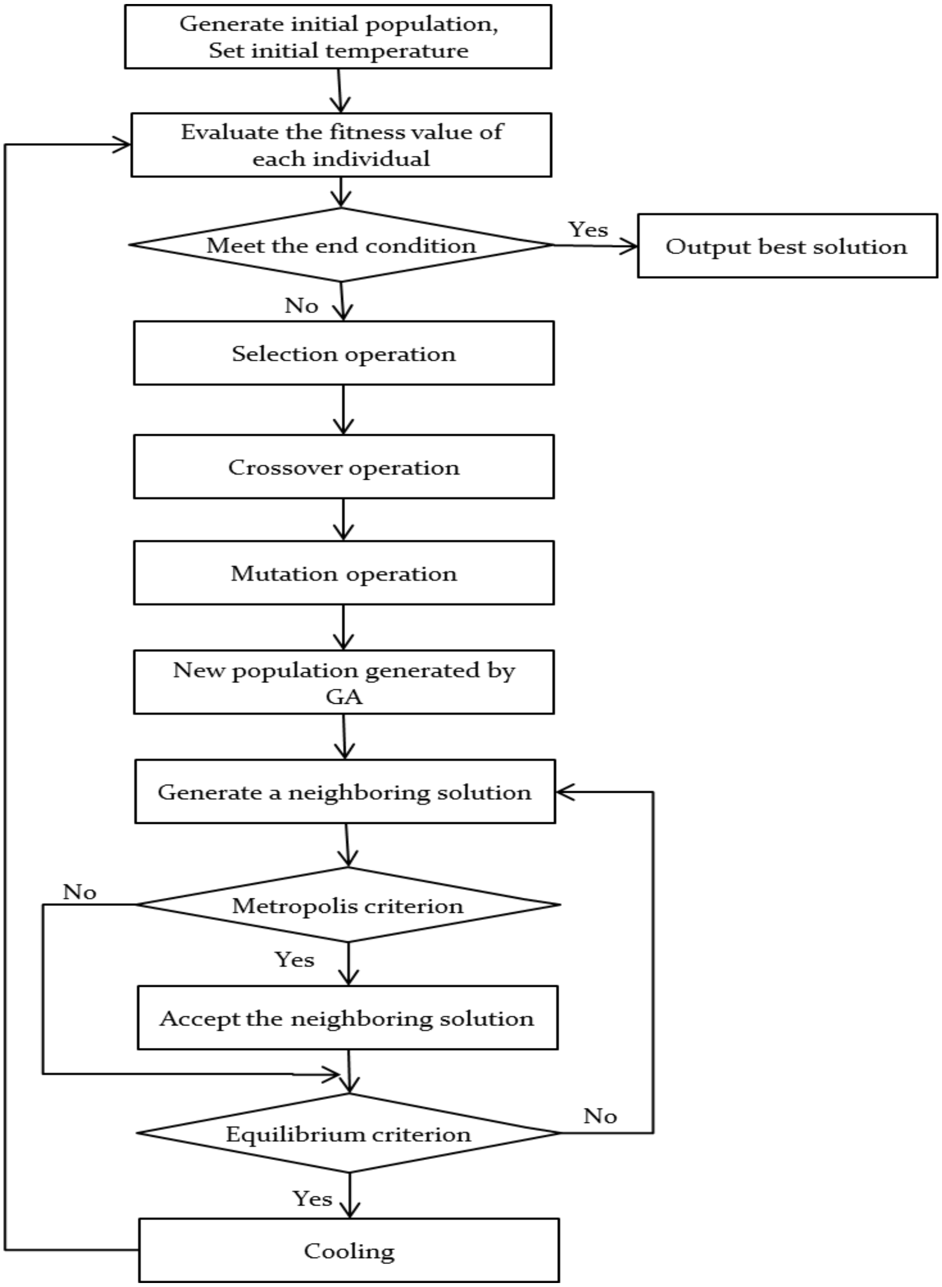

Figure 5. Flowchart of AGSAA

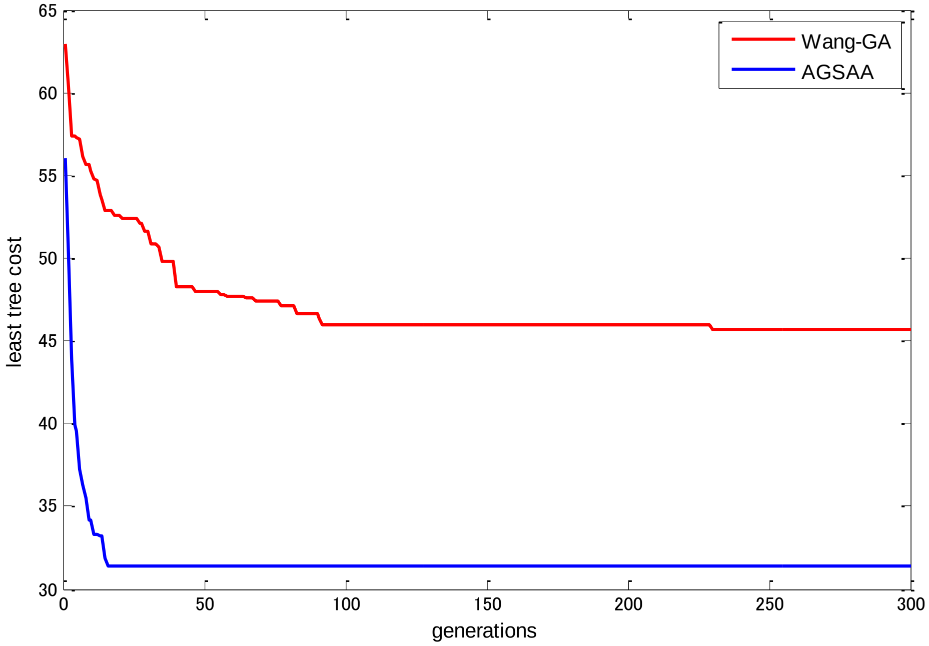

Figure 6. Comparison of Least Tree Cost in 20 Nodes Network

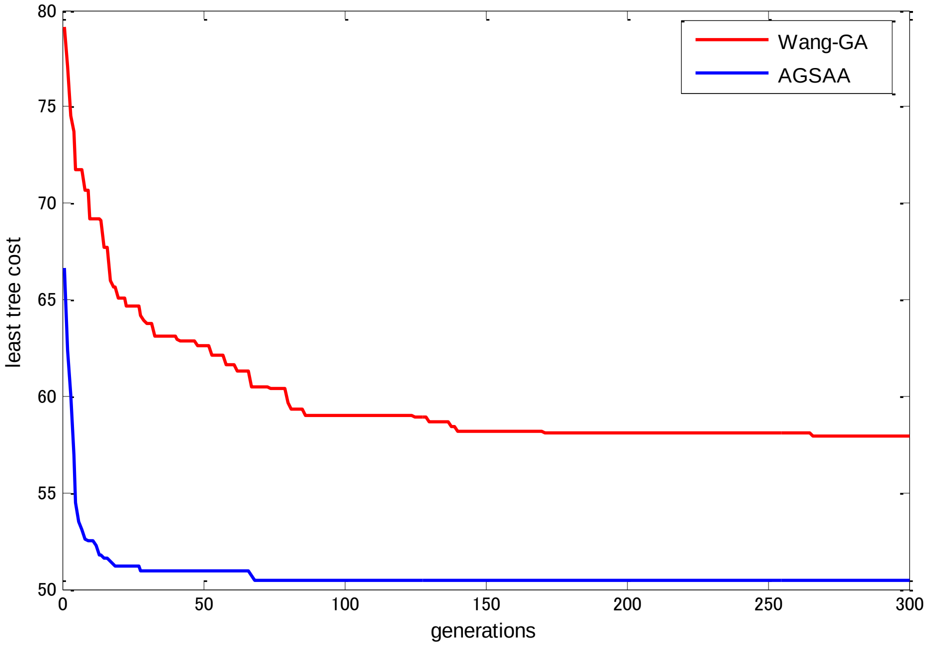

Figure 7. Comparison of Least Tree Cost in 40 Nodes Network

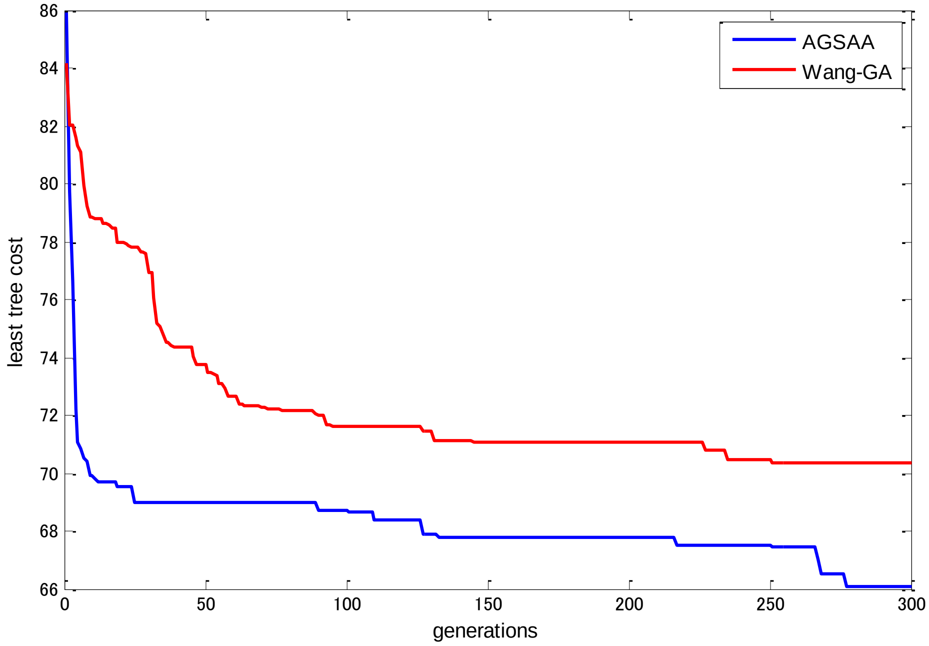

Figure 8. Comparison of Least Tree Cost in 60 Nodes Network

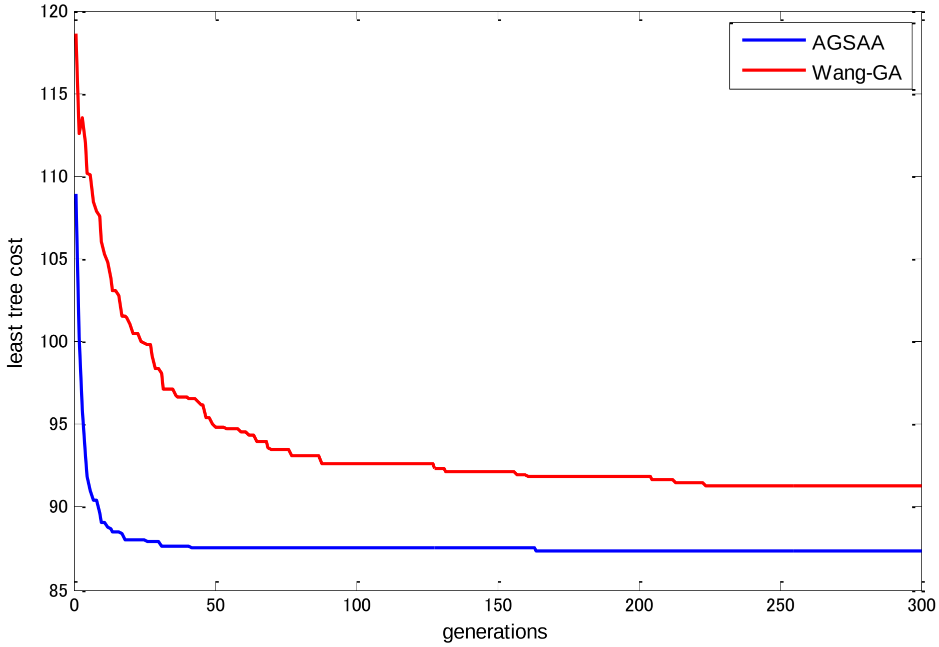

Figure 9. Comparison of Least Tree Cost in 80 Nodes Network

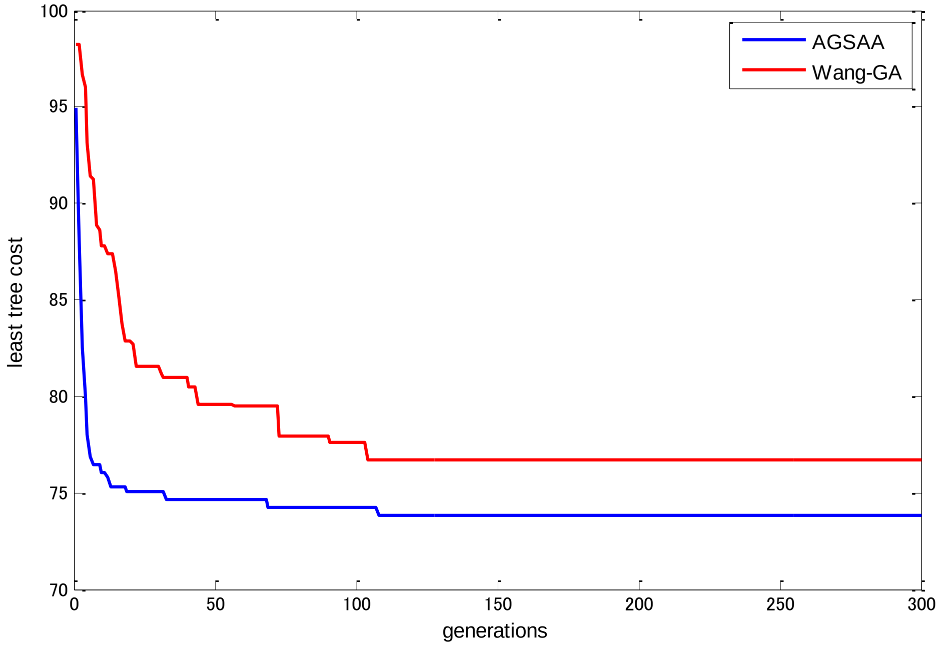

Figure 10. Comparison of Least Tree Cost in 100 Nodes Network

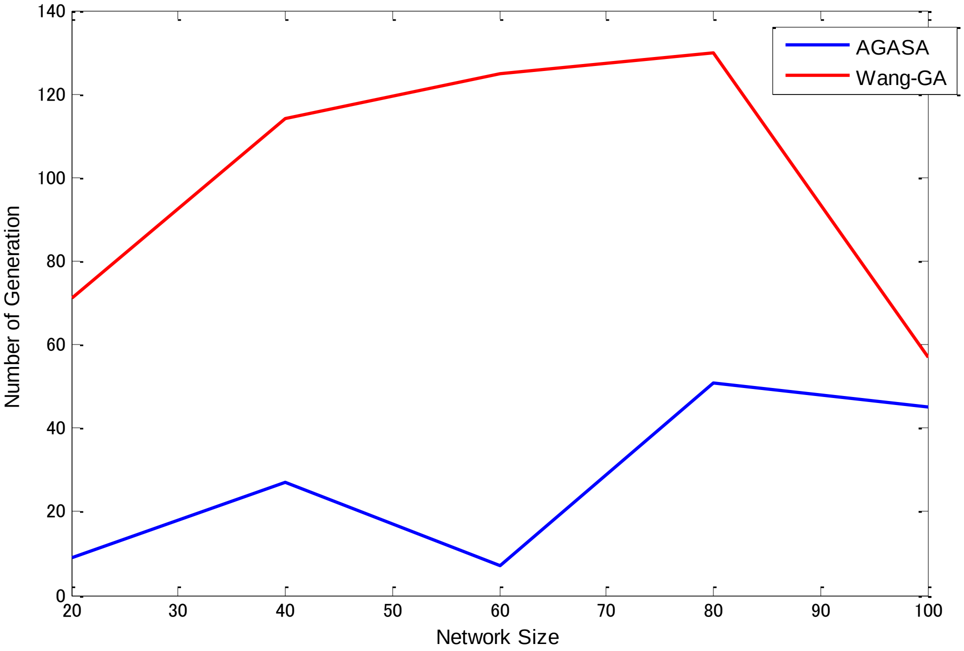

Figure 11. Average Iteration Times of Reaching the Minimum Multicast Tree Cost in Different Networks

The experiments mainly test the convergence ability, the convergence speed, and the least-cost of achieved solutions in each generation.

Figures 1, 2, 3, 4 and 5 show the least tree cost of our proposed AGSAA algorithm and Wang-GA-based algorithm versus iteration times in 20, 40, 60, 80 and 100 nodes networks respectively. These figures indicate that our proposed AGSAA algorithm can result in a smaller average tree cost than Wang-GA-based algorithm.

Figure 6 shows the average iteration times of reaching the minimum tree cost in 20, 40, 60, 80 and 100 nodes networks. This result indicates that the AGSAA provides a significant improvement for obtaining a global optimum solution or a near global optimum solution in early generation. This simulation result shows that this AGSAA algorithm has fast convergence speed.

7. Conclusion

Multicast is an essential technology for many real-time multimedia applications involving group communication. QoS multicast routing become one of important technique in network real-time information transmission. Least-cost QoS multicast routing algorithm will be essential for the high-speed information networks.

In this paper, a least-cost QoS multicast routing algorithm AGSAA that combined Genetic Algorithm (GA) and simulated annealing algorithm (SA) is proposed. Our proposed algorithm has main features as follows:

- Using pre-processing phase to delete all the links, which their bandwidth are less than the minimum required bandwidth.
- Candidate-paths-set encoding method based on Dijkstra *kth* shortest path algorithm is adopted in this paper, and the encoding and decoding operations in genetic algorithm are simplified.
- In GA, we use fitness function to evaluate each individual. In this paper, we define the fitness function using the penalty technique. A penalty method replaces a constrained optimization with an unconstrained one. The unconstrained problems are formed by adding a term to the fitness function that consists of a penalty parameter and a measure of violation of the constraints.
- Selection operation in this paper adopts roulette wheel selection method and best individual preservation strategy.
- The improved adaptive crossover and mutation probabilities are adopted in this paper, and the speed of convergence is improved.
- The method of crossover operation ensures that the excellent link gene can be saved in offspring.
- Metropolis criterion of SA is combined with, and the algorithm can escape from the local optimal solution. SA can avoid the premature convergence of GA, and the character of GA is used to avoid the lower convergence speed of SA.

Therefore, this AGSAA algorithm can not only escape from local optimum but also have high convergence speed and excellent convergence ability. The simulation results show that this AGSAA algorithm has fast convergence speed and good performance of cost than

Wang-GA-based algorithm. This algorithm is an effective solution to the least-cost QoS multicast routing problem.

Acknowledgements

The author would like to express gratitude to all those who have helped her during the writing of this paper. Firstly, the author gratefully acknowledges the help of her supervisor Professor Lei Li. She appreciates his patience, encouragement, and professional instructions during this paper writing. Also, the author deeply appreciates the contribution to this paper made in various ways by her friends and classmates.

REFERENCES

[1] Fei X, Luo JZ, Wu JY, Gu JQ, “QoS Routing Based on Genetic Algorithm”, Computer Communications, vol. 22, **(1999)**, pp. 1392–1399.

[2] Kompella VP, Pasquale JC, Polyzos GC, “Multicast routing for multimedia communication”, IEEE/ACM Transactions on Networking, vol. 1, No. 3, **(1993)**, pp. 286–292

[3] Parsa M, Zhu Q, Garcia-Luna-Aceves JJ, “An iterative algorithm for delay constrained minimum-cost multicasting”, IEEE/ACM Transactions on Networking, vol. 6, no. 4, **(1998)**, pp. 461–474.

[4] Wang ZY, Shi BX, Zhao ED, “Bandwidth Delay Constrained Least Cost Multicast Routing Based on Heuristic Genetic Algorithm”, Computer Communications, vol. 24, **(2001)**, pp. 685–692.

[5] Haghighat AT, Faez K, Dehghan M, Mowlaei A, Ghahreman Y, “GA-Based Heuristic Algorithms for QoS Based Multicast Routing”, Knowledge-Based Systems, vol.16, **(2003)**, pp. 305–312.

[6] Kapsalis A, Rayward-Smith VJ, Smith GD, “Solving the graphical Steiner tree problem using genetic algorithms”, Journal of the Operational Research Society, vol. 44, no. 4, **(1993)**, pp. 397–406.

[7] Wang XH, Wang GX, “A multicast routing approach with delay-constrained minimum-cost based on genetic algorithm”, Journal of China Institute of Communications, vol.23, no.3, **(2002)**, pp. 112–117.

[8] Hamdan M, El-Hawary ME, “Multicast Routing with Delay and Delay Variation Constraints Using Genetic Algorithm”, Proc. The Canadian Conference on Electrical and Computer Engineering (CCCECE 2004), **(2004)**, pp. 2363-2366.

[9] Sun Q, “A genetic algorithm for delay-constrained minimum-cost multicasting”, Technical Report, Insititute of Operationg Systems and Computer Networks, TU Braunschweig, Butenweg, 74/75, 38106, Braunschweig, Germany, **(1999)**.

[10] Leung Y, Li G, Xu ZB, “A genetic algorithm for the multiple destination routing problems”, IEEE Transactions on Evolutionary Computation, vol.2, no.4, **(1998)**, pp.150–161.

[11] Zhang Q, Lenug YW, “An orthogonal genetic algorithm for multimedia multicast routing”, IEEE Transactions on Evolutionary Computation, vol.3, no.1, **(1999)**, pp.53–62.

[12] Esbensen H, “Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm”, Computer Science Department, Aarhus University, DK-8000 Aarhus C, Denmark, **(1994)**.

[13] Kazarlis SA, Bakirtzis AG, Petridis V, “a genetic algorithm solution to the unit commitment problem”, IEEE Transactions on Power Systems, vol. 11, no. 1, **(1996)**, pp.83-92.

[14] Zhang K, Wang H, Feng YL, “Distributed multicast routing for delay and delay variation-bounded Steiner tree using simulated annealing”, Computer Communications, vol.28, no.11, **(2005)**, pp. 1356–1370.

[15] Zhang L, Cai LB, Li M, Wang FH, “A method for least-cost QoS multicast routing based on genetic simulated annealing algorithm”, Computer Communications, vol. 32, **(2009)**, pp. 105–110.

[16] Rehab FAK, “Hybrid discrete PSO with GA operators for efficient QoS multicast routing”, Ain Shams Engineering Journal, vol. 2, **(2011)**, pp. 21–31.

[17] Liu JM, Liu GY, “Algorithm of QoS Multicast Routing Based on Genetic Simulated Annealing Algorithm”, International Conference on Computer Application and System Modeling, ICCASM, **(2010)**.

[18] Salama HF, Reeves DS, Viniotis Y, "Evaluation of multicast routing algorithms for real-time communication on high-speed networks", IEEE Journal on Selected Areas in Communications, vol. 15, no.3, **(1997)**, pp. 332–345.

[19] Srinivas M, Patnaik LM, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", IEEE Transactions On Systems, Man And Cybernetics, vol. 24, no. 4, **(1994)**, pp. 656–667.

[20] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller H, "Equation of calculations by fast computing machines", The Journal of Chemical Physics, vol. 21, no. 6, **(1953)**, pp. 1087–1092.

[21] Wang XW, Cao JN, Cheng H, Huang M, "QoS multicast routing for multimedia group communications using intelligent computational methods", Computer Communications 29 **(2006)**, pp. 2217–2229.

[22] Holland JH, "Adaptation in Natural and Artificial Systems", Ann Arbor, The University of Michigan Press **(1975)**.

[23] Goldberg DE, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, **(1989)**, pp. 432.

[24] Forrest S, Mitchell M, "Relative building-block fitness and the building-block hypothesis", In L. D. Whitley, editor, Foundations of Genetic Algorithms 2, Morgan Kauffman, San Mateo, CA, **(1993)**.

Authors

Bo Peng received the M.Sc. degrees in Electronic Technology from Technische Universit ät Dresden (TU Dresden), Germany, in 2008. She is currently working toward the Ph.D. degree in the Department of Information Electronics Engineering at Hosei University, Tokyo, Japan. Her research interests include multicast routing, mobile ad hoc networks, etc.

Dr. Lei Li is a full Professor and the Vice-Dean of Faculty of Science and Engineering, Hosei University, Koganei, Tokyo in Japan. Dr. Lei Li was born in Yancheng, Jiangsu, China on May 1, 1961. He received his Doctor Degree of Science from Xian Jiaotong University, China in 1989, and Doctor Degree of Engineering from Tohoku University, Japan in 1994 separately. He studied at Hirosaki University, Japan as a Post-Doctoral and Research Assistant from April, 1989 to March 1992. From April 1992 up to present, he has been on the Faculty of Computer Science and Engineering [1992-1997: Associate Professor, Aomori University, Japan, 1997-2001: Associate Professor, Yamaguchi University, Japan, 1999-2000: Visiting Professor, State University of New York at Stony Brook, 2002-present, Professor, Hosei University, Japan]. His research interest includes the Fast Algorithms, Parallel Algorithms, Genetic Algorithms, Neural Networks, Machine Learning Algorithms etc. He has published around 220 papers in refereed journals, conference proceedings and book chapters in these areas. He has been involved in more than 30 conferences and workshops as a program/general/organizing chair. He is serving as the president of International Information Institute. In addition, he is the Editor-in-Chief of Information, An International Interdisciplinary Journal and as an Associate Editor for some other International Journals. He has been acting as an author/co-author or an editor/co-editor of 12 books. He served as Dean of Graduate School of Engineering, Hosei University, Japan, from 2008 to 2009. Dr. Li received the International Award of HITACHI TO Software Papers in Japan (1990), the Outstanding Leadership Award from International Conference on Computer Convergence Technology in Seoul, Korea (2011), and the Outstanding Service Award from ACM Research in Applied Computation Symposium in Miami, USA (2011) separately.