An Artificial Bee Colony Optimization Algorithm for Multicast Routing

Zhenhua Zheng*, Hua Wang*, Lin Yao*
*School of Computer Science and Technology, Shandong University, China
zhenhua_zheng@mail.sdu.edu.cn, wanghua@sdu.edu.cn, happylin@mail.sdu.edu.cn

Abstract— **The most important problem in multicast routing is Steiner tree problem, which has been proved to be NP-complete. This article proposed an algorithm based on Artificial Bee Colony Optimization (ABC). In this algorithm, we optimize the Steiner tree directly. It is not the same as traditional methods that find paths and integrate them to generate a Steiner tree. This algorithm use the characteristic of the ABC, such as fast convergence, get optimum solution quickly. We did a lot experiments in different topology scale, and compared performance of multicast tree, convergence time and multicast tree cost with other algorithms, this algorithm has good improvement in optimization ability and convergence speed.**

Keywords— **Multicast Routing, Steiner Tree Problem, Quality of Service, Tree Transposition, Artificial Bee Colony Algorithm**

I. INTRODUCTION

With the rapid development of the network technology and multimedia technology, network multimedia services such as video on demand, distance learning, video conferencing has been widely attention. Such applications generally involve multiple users, great network resource consumption, and have a higher requirement on bandwidth, delay, jitter, loss rate and other QoS parameters. Because of multicast can effectively reduce network bandwidth consumption, improve the data transmission efficiency and other advantages, this kind of business has been widely used. Multicast routing is to look for a multicast tree connection source node and destination node, information is sent to different destination node parallel along this tree. So networks need to transmit the least copy information, it can save network resources. Actually solving the problem of multicast routing is to build a minimum cost multicast tree. The search and structural problems of multicast tree comes down to Steiner tree problem in mathematics, and the problem is a NP-complete problem [1].

Steiner tree theory and the algorithm is the basis for solving the multicast tree. The study in Steiner tree problem began in the 60s last century, so far there has been a lot of literature in this area. Since the problem is NP-complete, the computation of many exact algorithms to solve the problem increases with the nodes increase in the figure exponentially, does not apply to large-scale application. Some heuristic algorithms such as KMB [2] and SPH [3] which can quickly find a near optimal cost of the Steiner tree, but they can not get a better solution. In recent years, with the rise of ant colony algorithm, particle swarm algorithm, artificial bee algorithm and other swarm intelligence algorithms, people began to introduce swarm intelligence algorithm to solve the multicast tree problem, and have achieved certain results. J Hesser proposed a genetic algorithm(GA) [4] to solve Steiner tree. The algorithm based on the traditional ant colony optimization for Steiner tree (ACO) [5], and the Max–Min ant system algorithm MMAS [6] are application of ant colony in Steiner tree.

Through the combination of artificial bee colony algorithm (ABC) theory and the idea of Steiner tree transformation, the process of bees searching solution space is replaced into the process of tree transformation. Then we generate a new Steiner tree construction algorithm. ABC has been proved that converges well and computes quickly [7]. Previous experiments are basically use 40 nodes of the topological structure, and we carry out experiments in topological structure with 100 nodes in this paper. The experiments proved this algorithm has faster convergence speed and better optimization ability in large scale than other algorithms.

This paper is organized as follows. Section one describes the background and my research. Section two describes the mathematical model of Steiner tree problem. Section three describes the basic principles of swarm algorithm. Section four presents the new multicast routing algorithm based bee swarm intelligence. Section five is the simulation results and analysis of algorithms. Section six is conclusions and prospects for future research.

II. THE MATHEMATICAL MODEL

Definition 1: Given a graph $G = (V, E)$, V is the nodes set of graph G, E is the edges set of a graph G, $|V|$ is the number of nodes in a graph G, $|E|$ is the number of edges or links in graph G. The cost function as follows: cost: $E \rightarrow R$. Destination nodes set $D \subseteq V$, $m = |D|$ is the number of destination nodes. Then the minimum Steiner tree problem is defined as to find out a minimum spanning tree cover all the nodes in the set D from graph G, to minimize the cost of the tree. The minimum spanning tree is called the Steiner tree. It denoted as $T_s(V_T, E_T)$, which $D \subseteq V_T \subseteq V$, $E_T \subseteq E$. Therefore the smallest Steiner trees can form into a formula as follows:

$$\min \sum_{e \in E_T} \cos t(e) \tag{1}$$

Definition 2: Directed graph $G = (V, E)$ if satisfied:

- Do not contain ring.
- There exists a vertex i not have in-arc, and for the rest of the vertex j have only one in-arc.

G is called Arborescence for the root *I*, also called Rooted Tree or Out-Tree.

Definition 3: Given a graph *G = (V, E)*, *V* is the nodes set of graph *G*, *E* is the edges set of a graph *G*, |*V*| is the number of nodes in a graph *G*, | *E* | is the number of edges or links in graph *G*. The cost function as follows: cost: *E*→*R*. Destination nodes set $D \subseteq V$. Root node $r \in V$. Directed Steiner minimum tree is defined as starting from the root to find the minimum spanning tree that cover all the node of *D*, to minimize the cost of the tree and meet the form of Out-Tree in the definition 2.

III. INTRODUCTION TO THE ARTIFICIAL BEE COLONY

The artificial bees colony algorithm(ABC) is a kind of swarm intelligent optimization algorithms, generating through simulation of the honey bees' behavior. In reality bees group collect nectar with very high efficiency in any environment, while adapting to changes in the environment. Based on the research of bees' behavior, Karaboga first proposed the model of human bees algorithm in 2005 [7] and further developed by Karaboga and Basturk [8]. Algorithm model contains three basic elements: employed bees, onlookers and non-employed bees.

Food sources: the target bees searching, the quality evaluated by a variety of factors, such as the distance from the cellular level, the amount of nectar and so on. Unification of these factors in the algorithm is expressed as "fitness" to measure the quality of nectar.

Employed Bees: they correspondence with nectar, record its corresponding nectar information, and share it with other bees according to probability.

Non-employed bees: bees divided into onlookers and scouts, their main task is to find and exploit nectar. Onlookers find high-gains nectar by sharing information with the employed bees, while scouts bees to find new nectar near the hive.

While ABC algorithm solves optimization problems, the nectar corresponds to a feasible solution of the optimization problem, the income degree of nectar corresponds to the objective function value of optimization problem, and the algorithm is an iterative process actually. In the beginning, it randomly generates Size feasible solutions, the top 50% function value of which is seen as nectar, nectar number remains unchanged in the iterative process, each nectar corresponds to an employed bee, and last 50% of which is seen as the location of the onlookers. Employed bees are able to search for new nectar in a nearby area, the formula is as follows:

$$new\_x_{id} = x_{id} + r(x_{id} - x_{kd}) \qquad (2)$$

d is the dimension of the solution vectors , *r* is a random number between the [0,1], $k \in \{1,2,...,Size\} - \{i\}$ is randomly generated number, corresponding to any nectar except the *i* nectar. With the number of iterations increases, the value $(x_{id} - x_{kd})$ will gradually decrease, which means the space bees searching diminishes gradually, helping improving algorithm's search accuracy.

When employed bee completes neighborhood searching, it would compare original one with the neighborhood searching nectar, and save the better one, at the same time, it shares information with the onlookers in dance way, who will select the nectar based on information provided in certain probability. The higher the income of the nectar is, the greater the probability of attracting following bees is. Selecting probability formula is as follows:

$$p_i = \frac{fit_i}{\sum_{n=1}^{size} fit_n} \qquad (3)$$

where fit_i is the fitness of the solution represented by the food source size and *n* is the total number of food sources. Clearly, with this scheme better food sources will get more onlookers than the bad ones.

After the onlookers select the new nectar, it has been transformed into employed bees at this time, and going to complete searching in the neighborhood of nectar and save those better ones. If a nectar corresponding to a feasible solution does not change after many iterations, the bee transform into scout and re-generate a new feasible solution randomly, which means employed bees give up their original nectar to find new nectar.

IV. TREE-BASED ABC ALGORITHM (T-ABC)

This paper adopts ABC algorithm to optimize Steiner tree directly. At the start of the algorithm we generate a group P of initial solution as employed bees randomly, evaluate the fitness. Employed bees produce a solution of modifications depends on local information of its memory according to formula (2), in this paper by merge with other bees, and test the fitness value of new food sources. If the new fitness value is higher, then the employed bee save the new solution, and forget the old one. Otherwise, it still saves old solution. After bees complete the search process, they share information on fitness and food source location in the dancing area. Onlookers assess all the fitness information of employed bees, and select food sources related to the probability of fitness values. As employed bees, onlookers also produce a correction for position in their memory and to detect the competitive position of fitness value. If its fitness value of higher than the previous one, the bees save new location.

A. Initialization of Algorithm

In this paper, bees not only indicate a solution, but each bee save a Steiner tree. If the number of nodes in the network is *N*, each bee save an $N \times N$ matrix *x*. If $x(i,j) = 0$, the link from *i* to *j* is belong the tree, otherwise it is not.

The algorithm initially produces the random source root tree. We suppose V is the set of nodes in the graph, V' is the set of nodes which belong the Steiner tree. The set of candidate edges E_c as follows:

$$E_c = \{(i,j) \mid v_i \in V', v_j \in V - V'\} \qquad (4)$$

Initialization process execution procedure is as follows:

Step 1: Select a node randomly, and put it into V'.

Step 2: Update candidate edge E_c according to the formula (4)

Step 3: Select a link *(i, j)* from E_c randomly, put it into the tree T_k, and then put *j* into V'.

Step 4: If $V' = V$, end process and return T_k, otherwise return step 2.

After all multicast group nodes added to the tree, we need to prune the leaf nodes which don't belong to the multicast group. We prune the useless nodes by checking every leaf node, if it's not the group node and it have an edge e_{mi} (node *m* is the parent node of *i*) , then delete the edge e_{mi}. Do the same operation on *m* until it is the group node or it's not leaf node.

B. Merging and Optimizing the Steiner Tree

The traditional multicast routing optimization algorithm finds the path from source to the destination node separately, and then merge the paths to form a Steiner tree. This paper abandoned the traditional practice of source root tree, optimize the Steiner tree directly. The following article will focus on the central algorithm used in the Steiner tree optimization. It simply requires the following four steps: merging, eliminating directed circles, trimming nodes with in-degree bigger than 1, and pruning non-used leaf nodes.

1) Merging the Steiner Tree: Merging the Steiner tree is the first step to populate evolution. The bees select a tree to merge in accordance with probability. The higher the fitness of the tree is, the greater the probability is. The actual stored form of Steiner tree is the adjacency matrix of the graph. The matrix is a $|V|\times|V|$ bi-dimensional array. Suppose there are two trees T_1, T_2. The merging process of two trees can be simple as $T_3[i][j] = T_1[i][j] \vee T_2[i][j]$. Of course after such processing digraph T_3 is not a tree, it needs to pass the steps after transformations.

2) Eliminating Directed Circles: After the merger of trees circles are most likely to exist, so need to eliminate directed circles. We adopts depth-first algorithm to eliminate rings.

The largest branch of the directed graph obtained after eliminating directed circles includes all the destination nodes. This is due to the definition of root tree that each tree will reach the source of all goal nodes. When two trees merge, the in-degree of destination node *i* may be 1, otherwise its in-degree is bigger than 1. Therefore, it can be found that there are only two instances when a node containing a circle: two trees have common edges, or the node has an in-degree bigger than 1.So after eliminating directed circles, it has not remove the common edges but only except some connection degrees with in-degree bigger than 1. Therefore the biggest branch obtained after eliminating circles include all of members nodes.

3) Trimming Nodes with In-degree Bigger Than 1: After merging and eliminating operation we got directed graph *T*, seeking the in-arc which has the largest fitness value in directed graph *T*. So these in-arc must be the composed of maximum branch of the directed graph and its fitness is the highest one. In this paper it is not keep the larger fitness edge simply but select by adopting probability, with larger probability select weights smaller edge. The fitness of edges is computed according to formula (5).

4) Pruning Non-used Leaf Nodes: After the operation, we got a out-tree with currently branch which has the largest fitness value. Because many leaves on the tree are not destination nodes, thus, it needs a fourth step to eliminate the useless nodes and related edges. The process is the same as the pruning of edges in Section A.

C. Evaluating Fitness

Our algorithm needs to evaluate fitness of the edges and Steiner tree. For the edge $e \in E$ between nodes $v, u \in V$, consider the following parameters: available bandwidth, delay. This paper adds the node parameter into the parameters of edge. So the fitness of an edge in evaluation uses the available bandwidth, cost, delays as evaluation standard. The function can be defined as follows:

$$fitness(e) = \begin{cases} 0, if\ \ bandwidth(e) < bw\_req \\ \alpha \times e^{-cost(e)/acost} + \beta \times e^{-delay(e)/adelay}, otherwise \end{cases} \quad (5)$$

Where α, β are the values of cost and delay function of the links. The variables *acost* and *adelay* express the average cost and delay of the set of edges in network. And *bw_req* means the bandwidth requirements of multicast. Formula (5) shows that an edge of fitness for multicast application is the cost and delay functions with the premise of bandwidth requirements.

While the evaluation of Steiner tree fitness is a compound function of total cost, end-to-end delay, and delay jitter, defined as follows:

$$fitness(u) = a_1 \times e^{-\mathrm{cost}()/cbcost} + a_2 e^{-\mathrm{del}()/cbdel} + a_3 \times e^{-\mathrm{jit}()/cbjit} \quad (6)$$

Where *cost()*, *del()*, *jit()* are function of cost, delay and delay jitter. The variables *cbcost*，*cbdel*，*cbjit* express the best solution of current Steiner tree respectively. a_1, a_2, a_3 are function metric. Finally, $a_1 + a_2 + a_3 = 1, a_1, a_2, a_3 \geq 0$ .In the simulation, the values of a_1, a_2, a_3 are set to 0.8, 0.15, and 0.05, respectively.

D. Algorithm Steps

Step1: Setting iterations counter *iter*, maximum iterating times *max_iter*, population *size*, source stay most restrictions *Limit* and other related parameters.

Step2: Generate *size* multicast trees, evaluating fitness of the trees according to formula (6). The *size/2* trees which have bigger fitness are as the food source position, Corresponding *size/2* employed bees. Initialize mark variable $Bas(i) = 0$, to

record the number of iteration that employed bees stay at the same solution.

Step3: Each employed bees *i* search nearby the food source according to the formula (2). Evaluate fitness of new solutions. If the new solution has a bigger value of fitness, update the employed bee's food source position, set $Bas(i) = 0$. Otherwise, update mark variable by $Bas(i) = Bas(i) + 1$.

Step4: Compute the probability of onlookers choose food sources according to formula (3). Each onlooker choose the food source according the probability and change into employed bee to search solutions. Record the best position of solutions, update $Bas(i)$.

Step5: Judge whether *Bas* is bigger than *Limit*. If $Bas(i) > Limit$, the scouts bee *i* give up current solution and search new solution nearby.

Step6: Record the best solution of all the bees, namely global optimal solution *Best*.

Step7: Update iteration times $iter = iter + 1$. If $iter > \max\_iter$, searching stop and output global optimal solution. Otherwise return to Step 3.

V. The Simulation Experiment and Analysis of The Results

We have implemented the algorithm proposed in this paper in C++ and executed it on a Pentium 4 system with 2GB MB RAM running at 3.0 GHz under Windows XP. The experiment used topological structure is based on random network topology generated according to method Waxman[9], this topology network is similar with the actual network. The mathematical model of the topological is as follows:

$$p(i, j) = \alpha \times \exp\left[-\frac{distance(i, j)}{L\beta}\right] \quad (7)$$

Where *p(i, j)* express the possibility of link exists between node *i* and *j*, *distance(i, j)* is the Eular Distance between node *i* and *j*, *L* is the largest value of *distance(i, j)*. Parameter α is used to control the number of links (the number of links increases with greater values of α) and parameter β is used to indicate the number of short links (a smaller β value means more short links). Following [10], we used a group of 30 bees, half of these bees are employed bees and remaining half are onlookers. In the experiment, we compared our algorithm with the basic ant colony algorithm ACO [11], the genetic algorithm GA in [4] and the Max–Min ant system algorithm MMAS in [6].

In the paper we use a topology with 100 nodes. To make it similar with real networks, the destination nodes should be accounts for 10% to 20% of the topology. We run the algorithm 100 times obtained the average solution. The results are showed in Figure 1 and Figure 2.

Figure 1 shows the cost of tree decreases with the number of bee increases. When the number of bees reaches 30, the algorithm begins to convergence. Figure 2 shows that the convergence time increases with the increase of the number of bees. That's because when the number of bee is low, it's hard for the algorithm to find the best solution. And with the number increases, it can build more random solutions at the beginning of the algorithm, so it's easy to find optimal solution. When the number reaches 30, although the cost of multicast tree still decreases, the changing begins to slow.

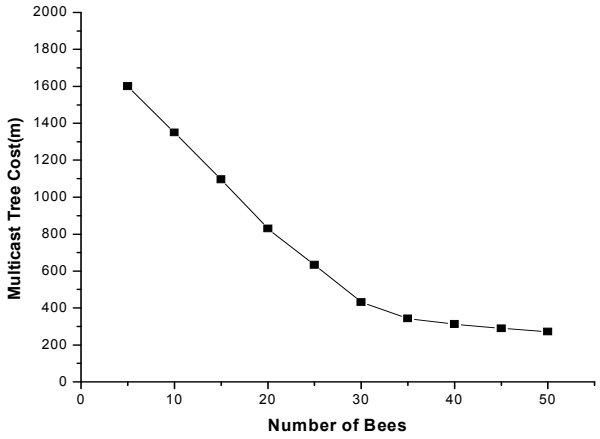

Figure 1. Cost of tree with different number of bees

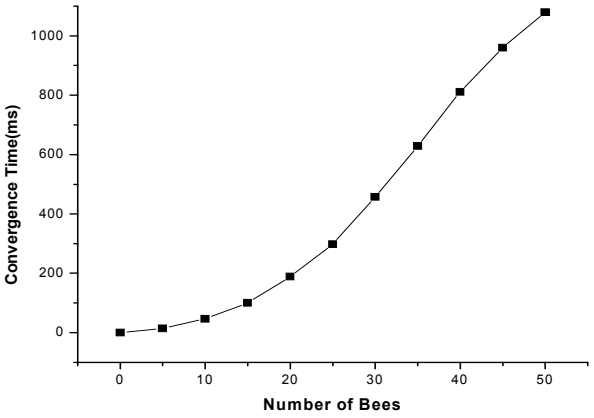

Figure 2. Convergence time with different number of bees

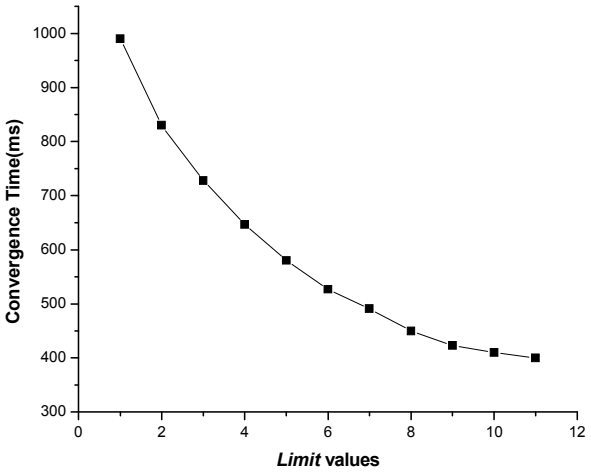

Figure 3. Convergence time with different values of *limit*

Figure 3 shows convergence time decreases with the increase of *limit*, that's because when the value of *limit* is small, means the employed bees have to give up current solution and find new solution, it's hard for the algorithm to convergence. But experiment shows that when *limit* is bigger than 11, the algorithm might fall into local optimal solution, so we use *limit*=11 in this paper.

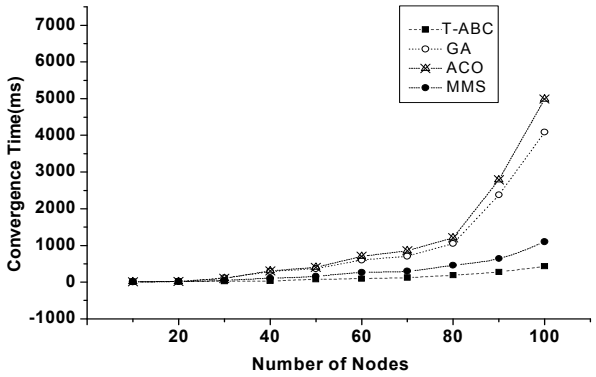

Figure 4. Comparison of convergence time in different topology scales

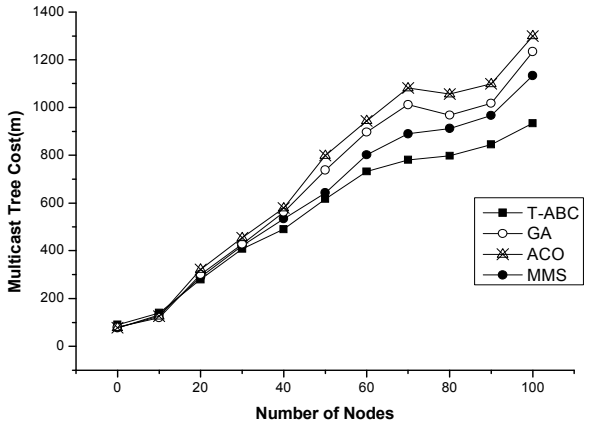

Figure 5. Comparison of results of algorithms in different topology scales

Figure 4 shows that convergence time increases with the increase of the topology scale, the convergence time of each algorithm increases. The time of GA, ACO and MMAS grows faster than the T-ABC. When the number of nodes is 100, T-ABC is still able to maintain a low convergence time while the time of the other three algorithms quickly increase. That is because the T-ABC optimizing the Steiner tree directly through tree transform, it saving a lot of time compared with the traditional ant colony algorithm to find the path and build a tree. And the T-ABC saves the better solutions in every step of the iteration, so it converges faster.

Figure 5 shows that the cost of Steiner trees increase with the increase of topology scale. The tree cost of T-ABC is always smaller than other algorithms.

As above mentioned, the convergence time and multicast tree cost of the T-ABC algorithm are better than GA, ACO and MMS. T-ABC performs better in different topology scales. Therefore, the T-ABC algorithm is an effective solution to Steiner tree problem.

VI. CONCLUSIONS

In this paper we proposed an algorithm based on ABC algorithm for Steiner tree problem. The algorithm introduces the tree transposition to optimize the trees directly, and improves its performance. Simulation results show that the proposed algorithm has very good optimization ability and high search speed in big scale topology. In the future, we will research the relationship between the parameters and topology scale, to make the algorithm have good performance in different topology scales.

ACKNOWLEDGMENT

The study is supported by Natural Science Foundation of Shandong Province (Grant No. ZR2011FM021), and Science and Technology Development Program of Jinan (Grant No.201102010).

REFERENCES

[1] Garey, M.R., R.L. Graham, and D.S. Johnson, *The complexity of computing Steiner minimal trees.* SIAM journal on applied mathematics, 1977. **32**(4): p. 835-859.

[2] Kou, L., G. Markowsky, and L. Berman, *A fast algorithm for Steiner trees.* Acta informatica, 1981. **15**(2): p. 141-145.

[3] Takahashi, H. and A. Matsuyama, *An approximate solution for the Steiner problem in graphs.* Math. Japonica, 1980. **24**(6): p. 573-577.

[4] Hesser, J., R. M nner, and O. Stucky, *On Steiner trees and genetic algorithms.* Parallelism, Learning, Evolution, 1991: p. 509-525.

[5] TIANDE, Y.W.G.U.O., *An Ant Colony Optimization Algorithms for the Minimum Steiner Tree Problem and its Convergence Proof [J].* Acta Mathematicae Applicatae Sinica, 2006. **2**.

[6] Pinto, D. and B. Barán. *Solving multiobjective multicast routing problem with a new ant colony optimization approach*. in *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking ACM* 2005: ACM.

[7] Karaboga, D., *An idea based on honey bee swarm for numerical optimization.* Techn. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.

[8] Karaboga, D. and B. Basturk, *Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems.* Foundations of Fuzzy Logic and Soft Computing, 2007: p. 789-798.

[9] Waxman, B.M., *Routing of multipoint connections.* Selected Areas in Communications, IEEE Journal on, 1988. **6**(9): p. 1617-1622.

[10] Karaboga, D. and B. Basturk, *On the performance of artificial bee colony (ABC) algorithm.* Applied Soft Computing, 2008. **8**(1): p. 687-697.

[11] Wang, Y. and J. Xie, *Algorithm for multimedia multicast routing based on ant colony optimization.* Journal-Shanghai Jiaotong Daxue Xuebao, 2002, 2002. **36**(4): p. 526-528.