ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(24), DOI: 10.17485/ijst/2016/v9i24/96035, June 2016* ISSN (Online) : 0974-5645

Efficient Mobile Agent Path-search Techniques using Genetic Algorithm Processing

Hongil Ji[1] and Chang Jin Seo[2*]

[1]Department of Automotive Software, Youngdong University, Korea;
jihi61@yd.ac.kr
[2]Department of National Defense Intelligence Engineering, SangMyung University, Korea;
cjseo@smu.ac.kr

Abstract

Background/Objectives: Although the efficiency of genetic algorithms improves with the creation of each generation, a number of generations are needed to obtain the desired results. In addition, when ad hoc unit increases are linked to a network, it may be necessary to compare all cases. **Methods/Statistical Analysis:** This requires the simultaneous generation of multiple algorithms at one time. Where a single process is used to manage all such algorithms, the overall efficiency of the network will decrease. **Findings:** The algorithm proposed in this thesis introduces router group cell units for use in the distributed processing of previous genetic algorithms. The experimental results showed that the proposed algorithm reduced path processing costs caused by the alternative path setup by approximately 27% when compared with Dijkstra's and the Munetomo algorithm. Operation time for the alternative path setup was approximately twice as fast as that of Dijkstra's algorithm. These results suggest that the algorithm proposed in this paper is more efficient than either Dijkstra's or the Munetomo algorithm in terms of alternative path setup during router failure. **Application/ Improvements:** The study presents ways to reduce overall search delays across a network through the use of a cell-based genetic algorithm.

Keywords: Ad-hoc Network, Genetic Algorithm, Mobile Agent, Path-search Algorithm, Route Search Method

1. Introduction

Transmission Control Protocol (TCP), the most widely used transmission protocol in the Internet, is used as a transmission protocol for various application programs such as File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP). TCP can transmit data reliably via congestion control techniques, which control the transmission rate according to the available bandwidth in the network, thereby preventing the network from congestion collapse. However, over a broadband wireless network, the congestion control algorithm in TCP increases the congestion window relatively slowly in comparison with the more rapid reduction that takes pace during packet loss, leading to insufficient utilization of broad bandwidth[1]. To solve this problem, network-adaptive transmission-rate adjustments have been proposed to transmit multimedia traffic efficiently by improving the congestion control algorithm in TCP[2,3].

Currently, Dijkstra's algorithm is most commonly used to implement a shortest-path search algorithm, with proven reliability. However, due to the increasing number of network nodes and multimedia streaming transmission, which has a complex tree structure, network transmission delays have increased in proportion to the complexity of long computation time. This makes it difficult to cope dynamically with increases or decreases in receive nodes, changes in transmission amounts, and changes in transmission traffic. To resolve this issue, considerable research attention has been paid to shortest-path network configuration through the use of genetic algorithms or network configurations that imitate neural networks[4,5].

This paper proposes an alternative path search by dividing the network into cell units and then applying a genetic algorithm. The proposed algorithm is more advantageous than applying a genetic algorithm to an

* *Author for correspondence*

entire network because it is able not only to shorten computation time but also to set up an alternative path that bypasses failed routers when specific routers are damaged. The proposed algorithm is able to determine directionality by using an existing shortest-distance algorithm (Dijkstra's algorithm). The study measures the level of convergence to Dijkstra's algorithm by applying a cell-unit genetic algorithm across the mobile agent environment. In addition, the study uses performance analysis to compare the delay time and cost associated with the proposed algorithm with delay time and cost for the Munetomo algorithm, an existing genetic algorithm. Finally, the study analyzes whether the proposed algorithm is better than existing genetic crossover operation-based routing algorithms in terms of alternative path setup, which may be required due to the abnormality of specific routers. Delay time and cost are measured accordingly.

2. Proposed Algorithm

2.1 Proposed Path-search Algorithm

With the mobile agent path-search technique, a genetic algorithm searches for the optimal path during routing using the directionality of the shortest-distance algorithm. A server searches routers across the pathway, using an agent controller to transmit information requested from a client, looking for the shortest distance between them. The proposed algorithm adds to this process by enabling cells that contain failed nodes to re-perform setup tasks via path modification on the client side during node failure. A server consists of services and their agent controls as well as a cell reiterate index that stores optimal paths in each cell. These are called elite groups. A client consists of a path modification module, which restores the failed nodes in a network, and a cell reiterate index, which stores the optimal paths obtained in each cell, as in the server. The selected path pays the lowest cost in cluster-type cells by multiplying the cells via a specific distance of apothem inside the network. These cells are created by the proposed algorithm using C_1, C_2, C_3, C_4, ..., C_n as shown in the Internet route, and C_{31}, C_{32}, C_{33}, C_{34}, ...C_{3n}, which are a group of routers duplicated between cells. Table 1 explains the function of each part in the structure of the proposed algorithm.

2.2 Execution Process of the Proposed Path-search Algorithm

2.2.1 Operation Model

The basic operation of the proposed algorithm is as follows: (1) initialization is carried out; (2) cell multiplication takes

Table 1. Structure of the proposed algorithm

Server		Client	
Component	**Function**	**Component**	**Function**
Service	Composes data to be transmitted for services (web, FTP, etc.) as requested by the client	Sender Socket	Responsible for receiving information sent from the server
Agent Control	Measures the distance between routers Sets up the cell's control criteria Performs the shortest-distance setup task Controls competition between cells Carries out recovery work during network failure Assigns priority rights between agents within the routers Performs agent extinction tasks within the routers	Cell Reiterate Index	Stores cells with optimal paths Stores router information duplicated within the control region between cells Stores the shortest-route information within the routers Stores the optimal paths within cells
Cell Reiterate Index	Stores cells with optimal paths Stores router information duplicated within the control region between cells Stores the shortest-route information within the routers Stores the optimal paths within cells	App	Program executed in the client, which has requested information from the server Responsible for receiving and utilizing requested information (application program)
Sender Socket	Responsible for sending the information requested from the server	Path Modification	Provides failure location tracking inside the network using the cell reiterate index

place; and (3) the optimal path is calculated via a genetic algorithm inside the cells using agents located in each cell. These agents find the destination through a competition process that compares the optimal path and the adjacent cell. Figure 1 illustrates the process[6].

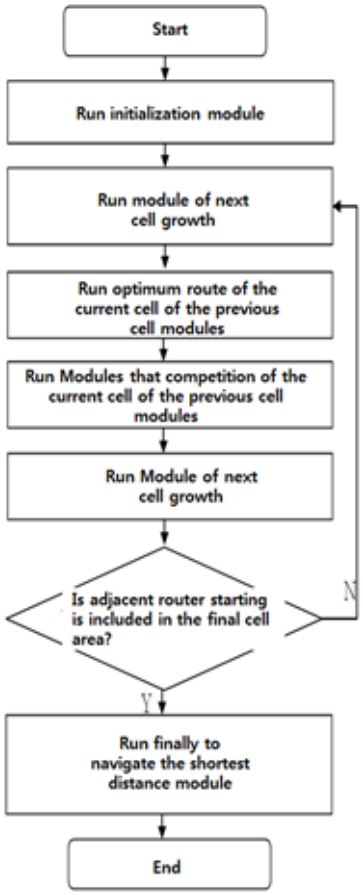

Figure 1. Flowchart of the proposed route-search algorithm.

2.2.2 Identifying the Shortest Distance

- Fitness function in the optimal solution

The following process is carried out to evaluate the optimal solution after it has been located through the crossover process in the genetic algorithm, which is based on a single cluster.

In the proposed algorithm, the total time over the selected path is used as an evaluation criterion to find the optimal solution from the starting node to the finish node in the cluster.

The number of R_n genes per unit of a single cluster is defined and the evaluation function of chromosome A, consisting of the above genes, is defined as follows:

$$\text{eval (A)} = \text{Test S(A)} + \text{Test T(A)} \quad (1)$$

where, the Test S evaluation function evaluates the total time needed to configure a link that is contained in a single cluster; and where the Test T evaluation function refers to the total delay time in a router queue, which is contained in the link configured by generation. In addition:

$$TestS(A) = \sum_{i-1}^{n-1} [\![GLT(R_i \rightarrow R_{i+1}]\!]) \quad (2)$$

where, GLT refers to the time taken to form a link from the i -th chromosome (i-th node R_i) to the i+1-th node R_{i+1} Next:

$$TestT(A) = \sum_{i-1}^{n-2} [\![RBD(R_i \rightarrow R_{i+1}]\!]), MD(R_{i+1} \rightarrow R_{i+2}) \quad (3)$$

where, MD refers to the maximum delay between two nodes and RRD refers to the total sum of delay times in the intermediate routers between two nodes, which correspond to the i-th gene and i+1-th gene and the delay before the conversion into the i+2-th gene node, which corresponds to the i+1-th gene.

2.2.3 Evaluation of Dominant Cell and Path Setup

Figure 2 compares optimal solutions obtained from the C_1 and C_2 cells through the optimal solution evaluation function from the viewpoint of the A_0 agent up to the boundary of the Cs_3 or Cs_4 cell along the optimal solution path, which comprises the dominant cells from the C_1 and C_2 cells. The above procedure is designed to increase the chance of finding the optimal solution through competition by cell region rather than following the optimal solution blindly within each cell.

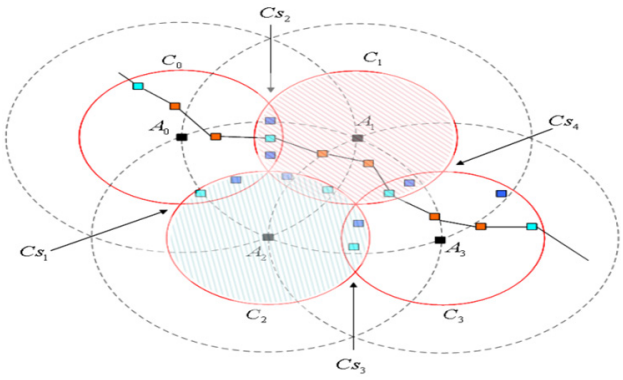

Figure 2. Evaluation of dominant cell and path setup.

Within the C_1 and C_2 cells, in the region marked with deviant crease lines, which follow the optimal path reached up to the A_0 agent, the optimal solution path of each cell is calculated through the chromosome crossover process. The result is passed on to the A_0 agent, which evaluates the paths of two candidates through the optimal solution evaluation function. Based on the result, dominant paths are selected from candidate groups and the flow of the entire path is progressed up to its own cell (A_0). The path defined above is then passed on to the A_1 agent of the selected cell. The A_1 agent applies the optimal path of its own cell region up to $Cs_2 \rightarrow Cs_4$. Using the process described above, an optimal solution is identified with the best performance among the optimal solution values in each cell through a competition process that is iteratively applied up to the destination[7].

2.2.4 Setup of Alternative Paths within the Cell

The proposed algorithm sets up the optimal path within the cell through the crossover operation. In cases where the optimal path may be damaged, the path modification process stores paths by rank. The three-generation crossover result within the cell shows that althoughB3I = 2. 3, 4, 6, 7, 11, 15 is selected as the optimal path for packet transmission, an alternative path is set up where Router 6 is damaged (Figure 3). In the proposed algorithm, the next best path, bypassing Router 6, is identified through the path modification process. In Figure 3, we see that C3I = 1, 3, 4, 5, 12, 14, which is the second rank, is set up as the alternative path, which bypasses Router 6; packets are transmitted through this path. In this case, the setup can be carried out without the recalculation process, thus reducing delay time. As the existing Dijkstra's algorithm requires recalculation time whenever an alternative path is set up, the delay time in the alternative path setup process proposed in this study is more efficient.

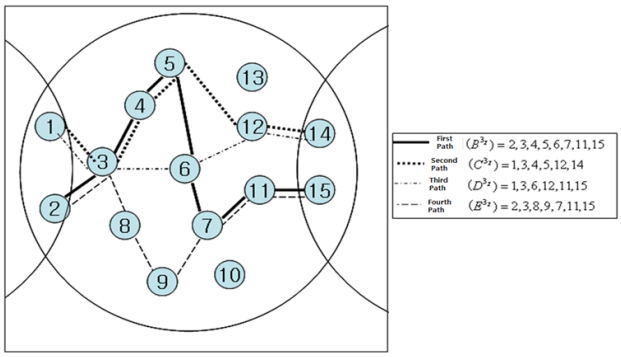

Figure 3. Path setup process within a cell in a single cluster.

Figure 3 shows a ranked search path, based on operation time through a three-generation crossover, within the cell, in a single cluster[8].

3. Experimental Performance Analysis

3.1. Experimental Environment

The transmission delay and processing rate in the Munetomo genetic algorithm, and in Dijkstra's algorithm, which is the optimal path-search algorithm, were compared and analyzed in order to evaluate the performance of the genetic path-search algorithm proposed in this study. The simulation model for this experiment includes the basic elements required for configuration by taking the sensitivity of the network setup into account. Furthermore, a verified topology is extended to test the characteristics of mobile agents[9,10].

3.2 Determination of the Cell Size Required for the Experiment

An appropriate cell size was selected for the simulation model over a hybrid topology environment with 300 routers. Four or more cells were used in the experiments with the proposed algorithm. Less than ten hops occurred if four or more cells were applied to the topology used in this experiment. With 11 or more hops, the 4 or more cells required in the experiment could not be formed, and were therefore unsuitable for the experimental environment. No overlap between cells was generated. With four hops or less, no overlap between cells could be generated and no control cells formed, again unsuitable for the experimental environment. To satisfy the needs of the experimental environment, the cell size had to include between five and ten hops. Since reliability is better when operation time is shorter, a cell size with five hops was selected. This has the shortest operation time.

3.3 Performance Analysis Comparison of the Proposed Algorithm

3.3.1 Operation Delay Time

In this study, optimal solutions derived from Dijkstra's algorithm and the proposed algorithms were compared to measure optimal performance and convergence speed. The performance of the proposed genetic routing algorithm was also compared with the genetic algorithm

suggested by Munetomo for use in configuring network topology. The performance analysis results are shown in Figure 4.

Dijkstra's algorithm was selected as a performance analysis reference point between the two genetic algorithms analyzed in this study. As shown in Figure 4, the path setup delay was relatively longer for the proposed algorithm for up to 40 algorithm generations, which indicates that the proposed algorithm has a longer delay time than the Munetomo algorithm. After 50 generations, however, the proposed algorithm produced a significantly shorter delay time than the Munetomo algorithm, which converged closely to the routing delay produced by Dijkstra's algorithm. After 80 algorithm generations, the proposed algorithm converged to the reference path. The reason for the larger early delay time in the proposed algorithm in comparison with the Munetomo algorithm resulted from the selection of the agent groups needed to set up the initial routing path. Transmission delay was also caused by the need to identify the status of the nodes managed by each agent. Once this process was complete, the proposed algorithm had a shorter delay time than the Munetomo algorithm. Table 2 compares the performance of the routing algorithms.

Table 2. Comparison of routing algorithm performance

Generation(g)	Proposed algorithm (delay time)	Dijkstra's algorithm (delay time)
20	3.225	1.551
40	2.897	1.544
60	2.225	1.523
80	1.525	1.533

3.3.2 Comparison of Path Processing Cost

To compare the path processing cost, an initial value was set by assigning a weight to each node from 110 to 160. The sum of weights in the existing Dijkstra's algorithm was 2.282 as a total path cost; the proposed algorithm was designed to have the optimal solution to converge at the path processing cost of Dijkstra's algorithm. The experiment results show that the total path cost for the Munetomo algorithm was 2.341, whereas that of the proposed algorithm was 2.297, which was 99.34%. The path cost of the Munetomo algorithm was 97.47%, as shown in Table 2. Thus, convergence performance in the proposed algorithm was higher than that of the Munetomo algorithm and it was also more robust.

3.3 Analysis of Performance Comparison according to Alternative Path Setup

3.3.1 Comparison of the Delay Time due to the Alternative Path Setup Operation

The operation time due to the alternative path setup process was compared for all three algorithms. The alternative path was set up owing to the failure Router 100 in the simulation topology. Figure 5 compares operation times due to the alternative path setup. As shown in the figure, the proposed algorithm, the Munetomo algorithm, and Dijkstra's algorithm produced an operational delay time of approximately 0.32 sec, 0.44 sec and 0.63 sec, respectively. This result indicates that the proposed algorithm is approximately twice as fast as Dijkstra's algorithm and approximately 37.5% faster than the Munetomo algorithm in terms of operation time due to the alternative path setup. This is because the proposed algorithm contains second rank path information in each cell, thereby reducing operation time when compared with the other algorithms.

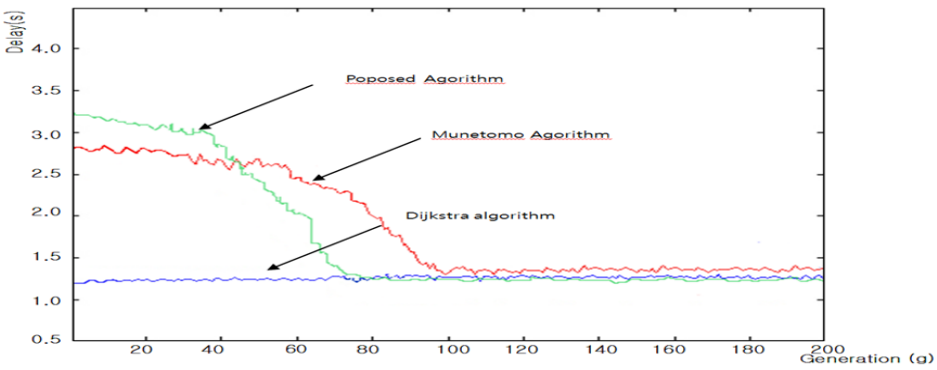

Figure 4. Comparison of the proposed algorithm and existing algorithms.

3.3.2 Comparison of Alternative Path Processing Costs

When a specific router fails over the network path, an alternative path that bypasses the failed router needs to be set up. In this experiment, the alternative path setup in the proposed algorithm was compared with Dijkstra's and the Munetomo algorithm.

In the experiment, the alternative path was required because Router 100 failed in the simulation topology. Dijkstra's algorithm produced an alternative path of 102-168-169-177 to transmit packets. The Munetomo algorithm had the same alternative path.

The proposed algorithm set up the alternative path of 67-99-98-103. As shown in Figure 5, the number of routers that all three algorithms passed through was increased from two before the alternative path setup to three after the alternative path setup. In terms of cost, Dijkstra's and the Munetomo algorithm showed an increased cost of 146, whereas the proposed algorithm showed an increase of 115. This result means that the proposed algorithm has a cost of 31 less than Dijkstra's and the Munetomo algorithm. Therefore, based on the experiment result, the proposed algorithm is better than both the existing Munetomo algorithmand Dijkstra's algorithm, previously regarded as the optimal path setup algorithm, in terms of alternative path setup.

Total path cost after setting up the alternative path was 2,418 for the Dijkstra algorithm, 2,494 for the Munetomo algorithm, and 2,412 for the proposed algorithm. This result suggests that the proposed algorithm can reduce path setup cost, when compared with Dijkstra's algorithm, by a factor of 6 (0.63%) when an alternative path is set up. After the alternative path was set up, the proposed algorithm continued to show better optimality than Dijkstra's algorithm. In addition, the proposed algorithm reduced path setup cost by a factor 72 (3.25%) when compared with the Munetomo algorithm. This suggests that the optimality of the proposed algorithm is better

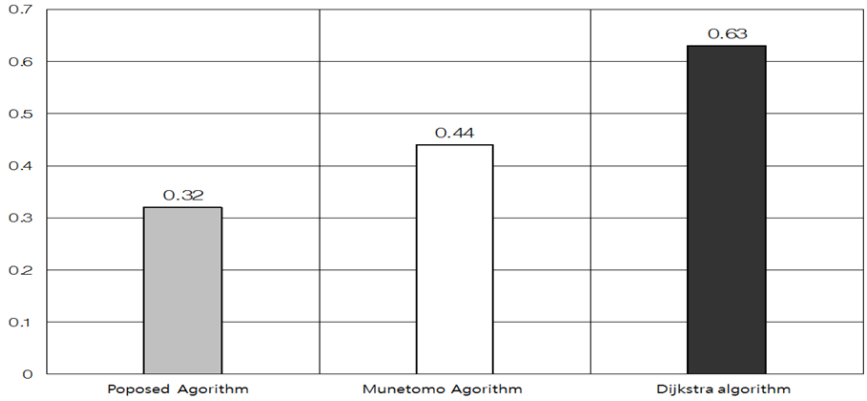

Figure 5. Comparison of the operation time due to alternative path setup.

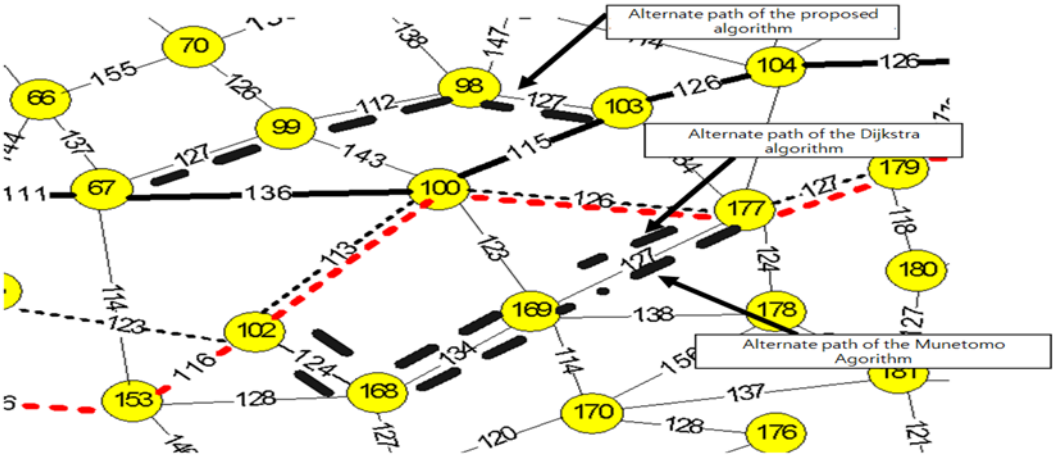

Figure 6. Setup of the alternative path during router failure.

than can be achieved by this existing genetic algorithm. Thus, when an alternative path was set up to bypass failed nodes, the proposed algorithm proved its superiority over both Dijkstra's and the Munetomo algorithm in terms of optimality performance from the cost viewpoint. Figure 6 shows the alternative path setup and compares values when Router 100 failed.

4. Conclusion

In this paper, an existing genetic algorithm and mobile agents were analyzed. A new algorithm was built to improve the path-search function of existing genetic algorithms. The proposed path-search algorithm operates on a cell-unit basis, thereby shortening the distribution of operation processing time.

The execution process of the proposed algorithm works as follows: (1) a first cell is made, followed sequentially by a second, third, and fourth cell at the same location; (2) an agent is copied and transferred to this point; (3) each agent then calculates the shortest path of each cell; and (4) the optimal elements are identified through competition.

Since the proposed algorithm performed a cell-unit genetic operation during the setup process, it decreased delay time significantly after 50 generations when compared with the Munetomo genetic algorithm. That is, the proposed algorithm had a relatively long initial delay time during the early path search while transferring data through the TCP session connection over real network environments. However, the overall efficiency of the proposed algorithm was better than the other algorithms in terms of transmission.

Based on the performance analysis result, the proposed algorithm reduces total path cost by approximately 1.87% when compared with the Munetomo genetic algorithm. This result means that the proposed algorithm is more efficient than the Munetomo genetic algorithm. Compared with Dijkstra's algorithm, which is the current optimal path algorithm, the proposed algorithm had a longer initial operation time. However, after eight seconds, operation time converged to that of Dijkstra's algorithm. The proposed algorithm achieved 99.34% of the total path cost of Dijkstra's algorithm, which suggests a relatively successful convergence to the optimal path.

In addition, path processing cost and operation time were analyzed for an alternative path setup that bypassed a failed router. The experimental results showed that the proposed algorithm reduced path processing costs caused by the alternative path setup by approximately 27% when compared with Dijkstra's and the Munetomo algorithm. Operation time for the alternative path setup was approximately twice as fast as that of Dijkstra's algorithm. These results suggest that the algorithm proposed in this paper is more efficient than either Dijkstra's or the Munetomo algorithm in terms of alternative path setup during router failure.

5. References

1. Persis DJ, Robert TP. Ant based multi-objective routing optimization in mobile ad-hoc network. Indian Journal of Science & Technology. 2015; 8(9):875–88.DOI: 10.17485/ijst/2015/v8i9/59369.
2. Begen C, Akgul T, Baugher M. Watching video over the web part 1: streaming protocols. IEEE Internet Computing. 2011; 15(2):54–63.
3. Floyd S, Ratnasamy S, Shenker S. Modifying TCP's congestion control for high speeds [Internet]. 2002 May.Available from: http://www.icir.org/floyd/papers/hstcp.pdf.
4. Holland JH. Adaptation in natural and artificial systems. The MIT Press; 1992.
5. Karnik NM, Tripathi AR. Design issues in mobile-agent programming systems. IEEE Concurrency. 1998; 6(3):52–61.
6. Baumann J, Hohl F, Rothermel K, Strasser M, Theilmann W.MOLE: A mobile agent system. Software: Practice and Experience. 2002; 32(6):575–603.
7. Inagaki J, Haseyama M, Kitajima H. A genetic algorithm for determining multiple routes and its applications. ProceedingsIEEE International Symposium on Circuits and Systems. 1999;6:137–40.
8. Ahn C, Ramakrishna RS. A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Transactions on Evolutionary Computation. 2002; 6(6):566–79.
9. Liu B, Choo S, Lok S, Leong S, Lee S, Poon F, Tan H. Integrating case-based reasoning, knowledge-based approach and Dijkstra algorithm for route finding. Proceedings of the Tenth Conference on Artificial Intelligence for Applications. 1994;1:149–55.
10. Xi C, Qi F, Wei L. A new shortest path algorithm based on heuristic strategy. Proceedings of the Sixth World Congress on Intelligent Control and Automation. 2006;1: 2531–6.