ROUTING ALGORITHM OF MOBILE AGENTS FOR QUERY RETRIEVAL USING GENETIC ALGORITHM

Ali Selamat and Md. Hafiz Selamat

Faculty of Computer Science and Information Systems,
Universiti Teknologi Malaysia,
81310 UTM Skudai, Johor, Malaysia.
Tel.: 6-07-5532009 Fax: 6-07-5565044
E-mail: aselamat@fsksm.utm.my, hafiz@fsksm.utm.my

ABSTRACT

Mobile agents often have a task to collect data from several predefined sites. This should be done in an efficient way by minimizing the elapsed time. Usually these agents only know the list of sites but not the distances between them. This paper proposes a method to minimize a network routing time taken by the mobile agents to collect information from different sites using genetic algorithm (GA). The mobile agents repeat travelling over short routes and avoid longer ones. Mobile agents for query retrieval have used the GA to select the best routes that minimize the query retrieval time. The result shows that the proposed method provides good time minimization in retrieving the query results by the mobile agents based on different GA parameters.

Keywords: Mobile Agent, Query Retrieval, Genetic Algorithm, Network Routing.

1.0 INTRODUCTION

The introduction of mobile agent in the field of distributed computing has proven useful where the mobile agents will roam the networks to search for information requested by the users. The mobile agents will also cooperate with other agents to accomplish the assigned tasks. There are four main properties belonging to mobile agents such as intelligence, communication, autonomy, and mobility [1]. The biological insects have inspired most of the research related to agent based network routing and their colonies [2]. It relies on the principles that individual insects will perform a simple behaviour while the collective communities of these insects will perform complex problem solving capabilities [3]. A research has been conducted in mapping the biological insects to the network routing management by using mobile agents. These agents are represented as artificial agents that traverse the network to collect specific information from the designated hosts. They will visit these hosts and coordinate with other agents to accomplish the assign tasks on behalf of users. They will also make several decisions to adapt their behavior according to the current environment in which they are currently resided.

Many researchers have investigated the adaptations of mobile agents for network routing. For example, a method to reduce the number of agents to be used to retrieve the information from the Internet that will minimize the routing time taken by the mobile agents has been proposed in [4]. An adaptive routing algorithm for network routing by using a rule-based method has been proposed in [5]. A method to reduce the cost and network traffic used by mobile agents to retrieve information from the Internet, namely, the highest probability first search (HPFS) algorithm has been proposed in [6]. The HPSF has been used to locate the agent that has been dispatched to the Internet to collect the required information. However, there is a problem with the HPSF where the control function of agents after the target object has been located has not been discussed. The performance migration of mobile agents from one host to another has been analyzed in [7]. However, the mobile agent routing mechanism has not been investigated.

In this paper, a network routing by mobile agents for query retrieval using a genetic algorithm (GA) is proposed. The agent repeats traveling over short routes and avoids longer ones. The GA approach for selecting the best routes has been applied to the mobile agents technology. We have applied the route selection of query retrieval by mobile agents in the Mobile Agent Search System (MaSS) [8]. The result shows that the proposed method provides good time minimization in retrieving the query results by the mobile agents based on different GA parameters.

This paper is organized as follows: The query retrieval using mobile agent is discussed in Section 2. The MaSS architecture is described in Section 3. The network routing using the GA applied to the MaSS search agent is discussed in Section 4. The experiments and results of proposed network routing method with different parameters are discussed in Sections 5 and 6. The conclusions are described in Section 7.

2.0. QUERY RETRIEVAL USING MOBILE AGENTS

The mobile agent technology for prefetching the results from the search engines in the MaSS is shown in Figs. 1 and 2. The term prefetching means that our mobile agents were used to retrieve the URL datasets that exist in the search engines databases and store them in the local database servers. The process has been executed during night-time by employing the activation and deactivation functions that exist in our mobile agent system. Activation is a process of activating the mobile agent to start retrieving the search engine databases based on specific queries. Also the activation time was set to 1-hour where in each hour, the mobile agent will retrieve the query results from the designated search engines. If the search query result is new, then it will be added to the local database, otherwise, it will be discarded. A deactivation is a process to stop the mobile agent from collecting the new search results, which will be determined by the user. Further description on the MaSS for query retrieval approach is described in the next section.

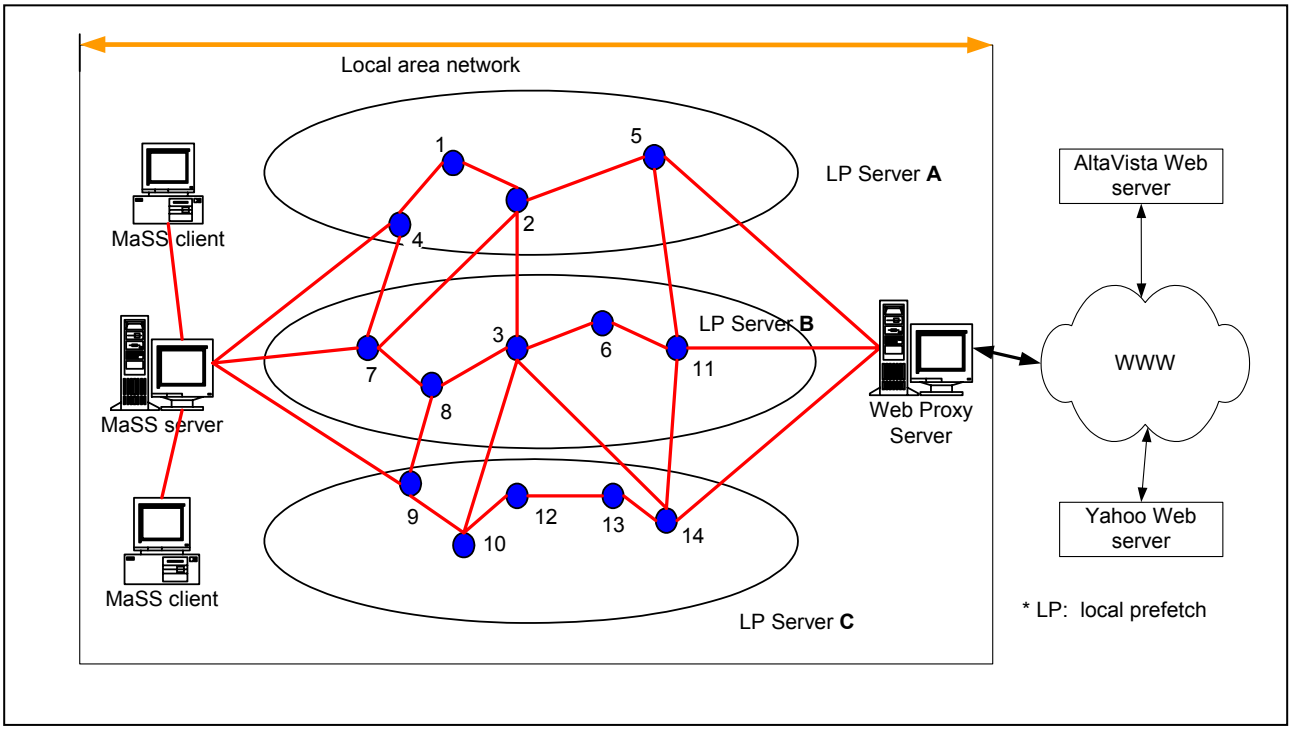

Fig. 1: The Mobile Agent Search System (MaSS) architecture.

3.0 THE MASS ARCHITECTURE

Mobile Agent Search System (MaSS) was developed to support retrieval query results from a few numbers of search servers and its architecture is shown in Fig. 2. The architecture consists of a MaSS client, a MaSS server, and a collection of the local prefetch (LP) servers. The algorithm used to select the best routes that minimizes the query retrieval time by mobile agents is shown in Fig. 3. Fig. 2 represents the detail architecture of the query retrieval using the MaSS. Further description on each component of the MaSS is described as follows:

3.1. The MaSS client

The MaSS client comprises of the MaSS client agent **(W)** and a simple user interface. A user interface is used as a medium for a user to interact with the MaSS.

3.2. The MaSS client agent

A user will enter a search keyword on the query form in the HTML browser. The MaSS client agent **(W)** will send a request to the MaSS server agent **(X)** in order to get the query search results.

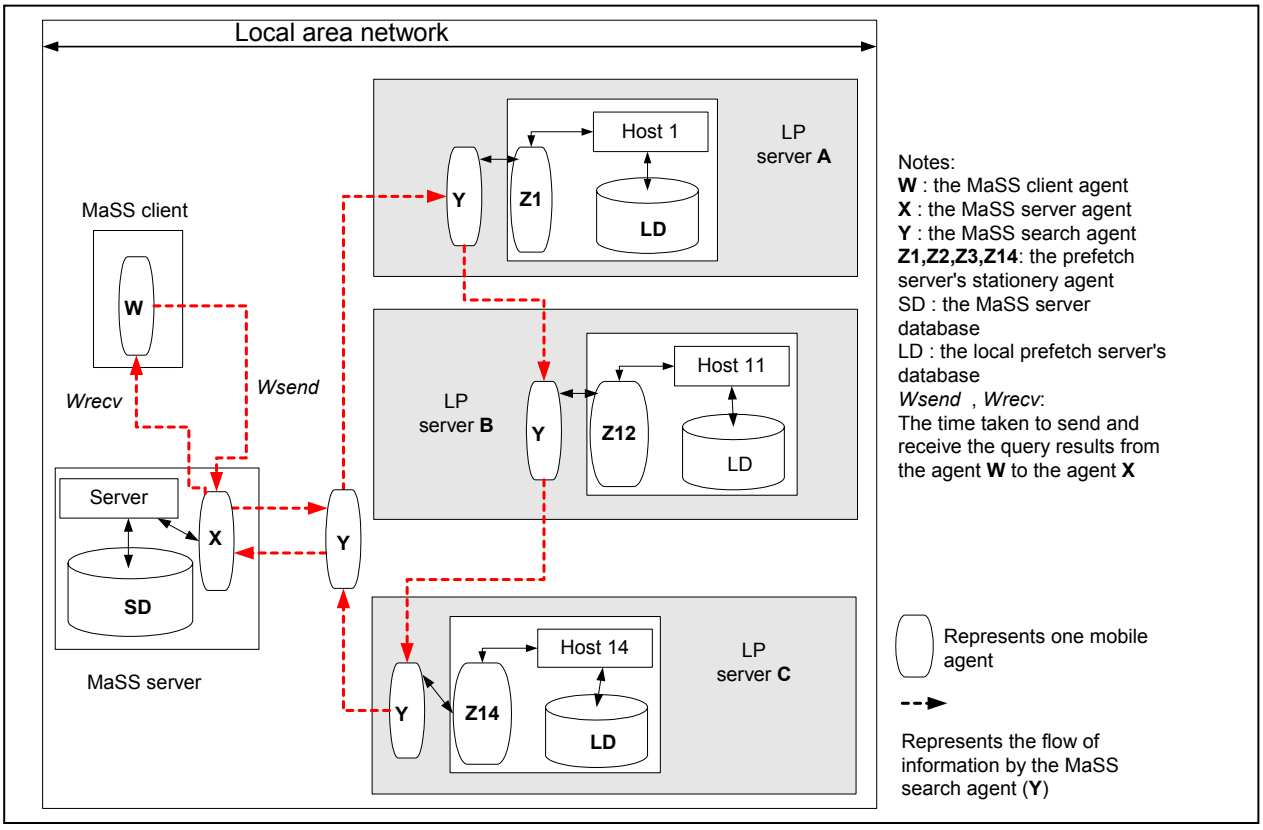

Fig. 2: The detail of the MaSS architecture.

3.3. The MaSS server

The MaSS server comprises of the MaSS server agent **(X)** and the MaSS search agent **(Y)**. Further descriptions on each of them are as follows:

3.3.1. The MaSS server agent

The MaSS server agent **(X)** is a stationery agent. Upon receiving request from the MaSS client agent **(W)**, the user's id will be checked at this stage. Then the MaSS server agent will delegate the search tasks to the MaSS search agent **(Y)**. After receiving the search results from the MaSS search agent, the MaSS server agent will rank them. The ranked search results will be stored into the MaSS server database **(SD)** before returning them to the MaSS client agent to be presented to the user.

3.3.2. The MaSS search agent

When receiving a query request from the MaSS server agent **(X)**, the MaSS search agent (Y) will start to mobile to a collection of local prefetch servers as shown in Fig. 2. Then it will communicate with the local agents at each of the hosts at the collection of the LP servers **A**, **B**, and **C**. At host 1, the MaSS search agent will ask the local agent about the query that is requested by the MaSS server agent. The query results will be given to the MaSS search agent by a local agent. The same process will be repeated at hosts 2, 3,…,14 with different local agents. Once the tasks have been completed, the MaSS search agent **(Y)** will return home and pass the search results to the MaSS server agent **(X)**.

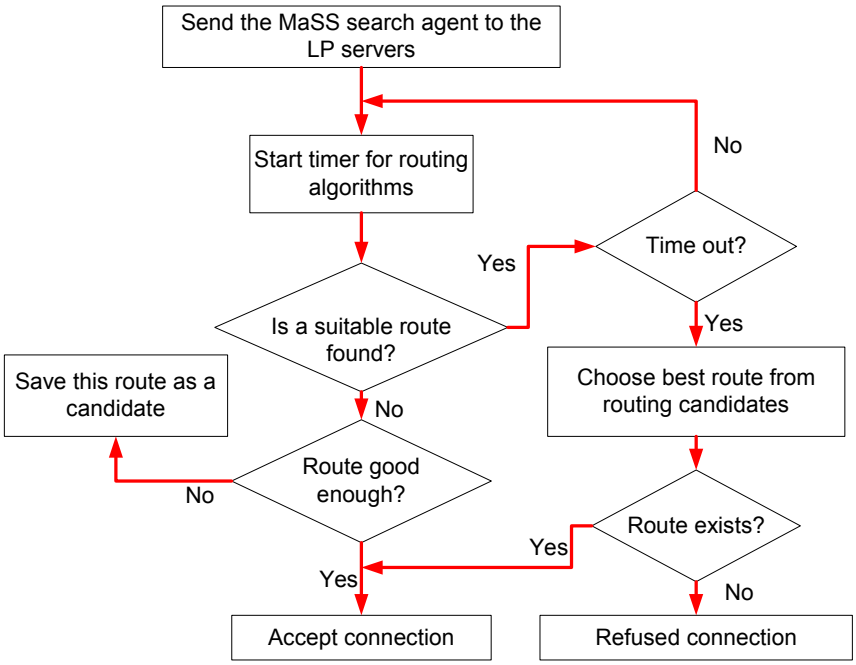

Fig. 3: The algorithm that has been used to select a suitable route by the MaSS search agent in order to collect the query results from the LP servers.

3.4. A collection of LP servers

The operational architecture of the MaSS in retrieving the search results from the collection of LP servers is shown in Fig. 2. The W_{send} and W_{recv} are the time taken to send and receive the query results from the MaSS client agent **(W)** to the MaSS server agent **(X)**. The LP servers consist of a local Yahoo database server (host 1), a local AltaVista database server (host 2), a local HotBot database server (host 3), etc. The idea of LP server is to store the query results locally instead of searching them from the World Wide Web. They will reduce the time of searching and retrieving the query results by the MaSS search agent **(Y)**. The LP servers will send their local agents to roam the WWW and collect the search results from designated search engines and store the query results in a local database **(LD)** at hosts 1, 2, …, 14. The process of removing the broken and duplicated links, stemming, and stopping will take place before storing them into a local database. This is performed in a night-time using activation and deactivation function that exist in the local agents.

4.0 NETWORK ROUTING BY MOBILE AGENTS USING GA

In order to retrieve the query results in an optimal time, the MaSS search agent has applied the GA approach for route selection in order to minimize the query retrieval time as shown in Fig. 4. Further formulation of the route optimization is as follows:

$$Min\ Qrt(route)\ \ s.t.\ Delay(route)\ \leq MaxDelay \qquad (1)$$

where *Qrt(route)* is the query retrieval time taken to retrieve the query results, *route* is the paths that have been used by the MaSS search agent **(Y)** to send and retrieve the results from the LP servers. *Delay(route)* is the time constraint due to network bottleneck. The operational process of the MaSS search agent in using GA for selecting an optimal route is shown in Fig. 4. *MaxDelay* is the maximum time delay applied to the path used by the MaSS search agent to retrieve the query results.

4.1. Encoding of the GA

The encoding of the GA that has been used by the mobile agents to select the optimal route in retrieving the query results is shown in Fig. 4. The path representation approach was chosen to encode a route due to its easy implementation. There are 15 LP servers including with the web proxy server as shown in Fig. 1. The route from the MaSS Server to the web proxy server can be represented as [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]. If the route does not exist then a value of 0 is included to the route such as [1 2 3 4 5 6 7 8 9 10 0 0 0 0 0].

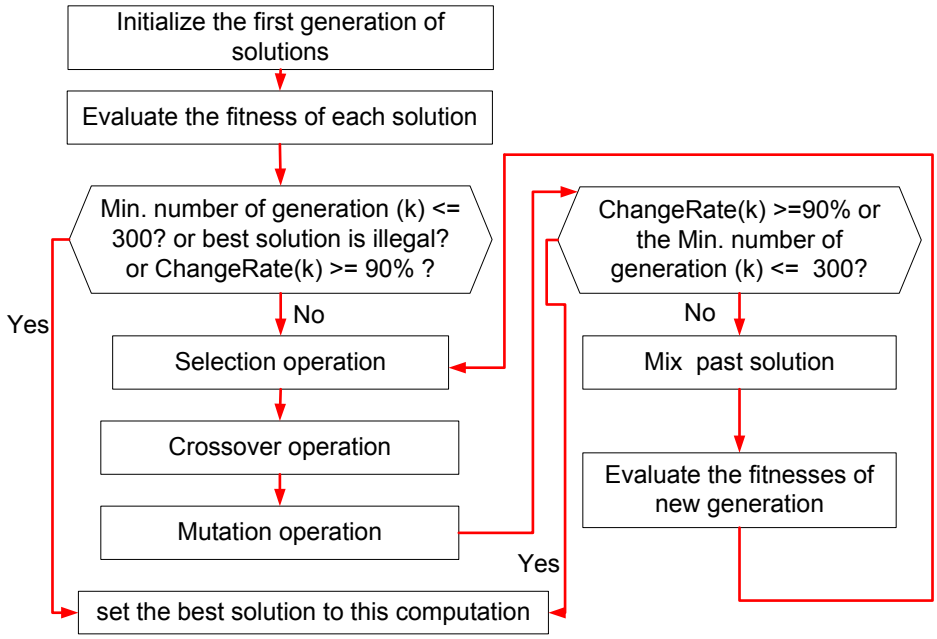

Fig. 4: The GA used to select a suitable route by the MaSS search agent.

4.2. Population initialization

We can randomly determine how many nodes the route will pass through and randomly determine which node will be in the route and the sequence of nodes of the route. However, there will be some solutions that may violate constraint of delay connectivity. A penalty method was used to deal with these constraints. For those routes that do not exist, a very large delay value was assigned to them. For those routes that violate the delay constraint, a penalty was added to their cost. The following expression to evaluate the weighted cost of those illegal routes was employed in the algorithm. It is given by

$$Qrt(route) = Cost(route) + (\alpha + Delay(route)) \qquad (2)$$

where *Qrt(route)* is the weighted query retrieval time for a selected route as described at the beginning of this section, α is the penalty constraint if the route does not exist, e.g., α=1,2,3,…,10, and *Cost(route)* is the function that evaluates the total cost of the links that the route may pass through. The details of *Cost(route)* and *Delay(route)* are shown in Tables 1 and 2 (in Appendix).

4.3. Fitness evaluation

Fitness of the solutions is proportional to the chromosomes survivability during the GA operation where the good values are selected and the bad values are discarded. In this study, the fitness of solutions was normalized to $0 \leq Fitness(route) \leq 1$ by the following expression:

$$Fitness(route) = \frac{Costs(route)}{TotalCosts(route) + MaxDelay} \qquad (3)$$

where the *Costs(routes)* has been described in previous paragraph and the *TotalCosts(routes)* is the sum of *Costs(routes)* for all populations at generation *k*. The *TotalCosts* is given by

$$TotalCosts(route) = \sum_{i=1}^{i=k} Costs(route) . \qquad (4)$$

For the first 30 generations, the *Fitness(route)* value was adjusted when the *Fitness(route)* ≤ 0.005 to *Fitness(route)* = 0.005 to prevent premature efficiency.

4.4. Selection operation

In order to keep the "good" solutions and discard the "bad" solutions at the same time, two selection operators have been used in the algorithm. First, the fitness of all solutions was added by generating number randomly between zero and the total value of fitness. Second, the fitness value was added on each of the solutions until the value is greater than the current fitness value of each solution. Then the highest fitness value among the solutions will be selected.

4.5. Crossover operation

In this paper, the traditional one-point crossover method is used. Firstly, a certain point of the array was determined and swap the part before and after the cross point to generate two new solutions. However, because the number of nodes the routes may pass through is not fixed, it is difficult to determine a fixed crossover point. So, a new dynamic crossover point method was assigned such as [((A+B)/4)] where A and B are the number of nodes that the two routes will pass through; respectively, and the operator "[]" is the rounding function. For example, two routes before the crossover operations are [1 2 3 4 5 0 0 0 0 0 0 0 0 0 0] and [6 7 8 0 0 0 0 0 0 0 0 0 0 0 0]. According to this procedure [((A+B)/4)] after crossover so that [1 2 8 0 0 0 0 0 0 0 0 0 0 0 0] and [6 7 3 4 5 0 0 0 0 0 0 0 0 0 0] were obtained. In the proposed approach, only part of the population will exercise the crossover operation.

4.6. Mutation operation

A solution has been randomly chosen among the population and the solution was changed slightly to generate a new solution. In this way, there are some chances to find better solution that cannot be found by only crossover operation.

4.7. Repair operation

During crossover and mutation operations, illegal representation of route may be generated because duplicated elements (node) may appear in the same route. In the proposed algorithm, those duplicated nodes were deleted that bring high cost to the route.

4.8. Finding route efficiently and dynamically

Although GA can be used to search in large solution space and obtain an optimal solution, it may take a lot of time to coverage to the optimal solution. In some cases, GA can only find some other sub optimal solution. In practice, usually a sub optimal solution is sought, which, however, is close to the optimal one. So, in this approach, a combination of conditions was used to determine when to stop the algorithm's computation. The basic idea is as follows:

After a minimum number of k generations (i.e., $k \leq 300$), if the algorithm has found a feasible solution and has made no improvement for a specific period of time, then stop. In the proposed algorithm, the "improvement" is presented by the coverage cost rate of the best solution of certain generation. This change rate is evaluated as follows,

$$ChangeRate\,(k) = \frac{Costs\,(k-1)}{Costs\,(k)} \times 100 \qquad (5)$$

where the cost of the best solution of that generation changes at *k*-th step and *ChangeRate(k)* is the average change rate of cost at *k*-th step. Once this value is greater than a certain value (i.e., *ChangeRate(k)* ≤ 90%), then stop the GA

computation. The flow of the GA process for a route selection by the MaSS search agent applied in the MaSS is shown in Fig. 3.

5.0 EXPERIMENTS

The experiments and the performance parameters of mobile agent using the GA routing have been conducted in a local area network (LAN) as shown in Tables 1 and 2. In these tables, only the nodes of interest (i.e., access nodes or edge nodes) and their interconnections are represented. In the following the cost and delay of links between two nodes at certain period time are assumed given. If there are no links exist between nodes or bandwidth of the link cannot support the traffic request between the MaSS server and the web proxy server, a large delay value (i.e., 999) is assigned to that link.

The objective of the experiment is to find the route from the MaSS server to the web proxy server that has the lowest cost and the delay is less than the maximum delay requirement. The cost matrix and the delay matrix are given in Tables 1 and 2 respectively. In these tables, element $(i,j,\ i \neq j)$, is the Cost(delay) from node i to node j where $i, j \in [1,2,,3,\ldots,15]$, element (i,i) is the Cost(delay) from the MaSS server to node i, element $(i, 15)$ is the Cost(delay) from node i to the web proxy server. The upper-bound requirement of time delay (MaxDelay) is assumed to be 25 times in this study. The parameters of the GA are shown in Table 3.

Table 3: Genetic algorithm parameters.

Items	Run-1	Run-2	Run-3
No. of populations	320	320	320
No. of genomes	14	14	14
Crossover rate	0.85	0.85	0.85
Mutation rate	0.7	0.5	0.24
No. of generations	420	420	420

6.0 EXPERIMENTAL RESULTS

The results based on three runs, which are shown in Figs 5,6,7, and 8 respectively. The cost of Run-1 is better than Run-2 and Run-3 as shown in Fig. 8. Also, the parameters that affect the populations' size and generations are analyzed and shown in Fig. 5, and the effect of mutation rate with the generations size as shown in Fig. 6. Furthermore, the effect of the cost values and the generations are also shown in Fig. 8. The mutation rates of 0.1 and 0.7 reduce the routes costs taken by the MaSS search agent when requesting and retrieving the query results from the LP servers as shown in Fig. 6. However, when the effects of costs are compared with the k generations as shown in Fig. 8, the mutation rate of 0.7 in Run-1 performs better compared with the mutation rates of 0.5 and 0.24 in Run-2 and Run-3, respectively. This is due to the network bottleneck that occurs while the MaSS search agent is retrieving the query results from the LP servers.

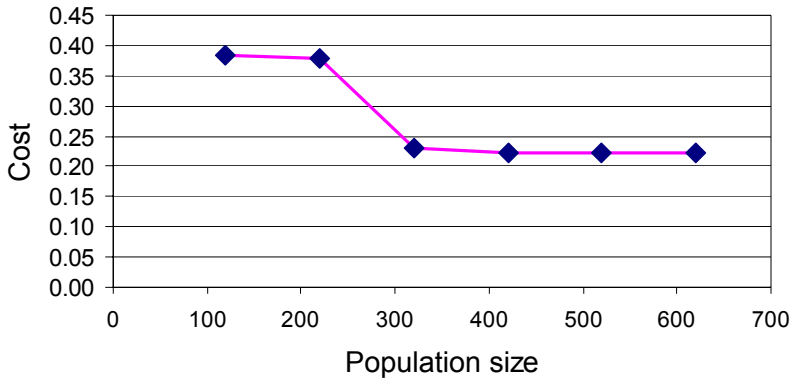

Fig. 5: The effect of the size of chromosome pool on the performance of the proposed algorithm.

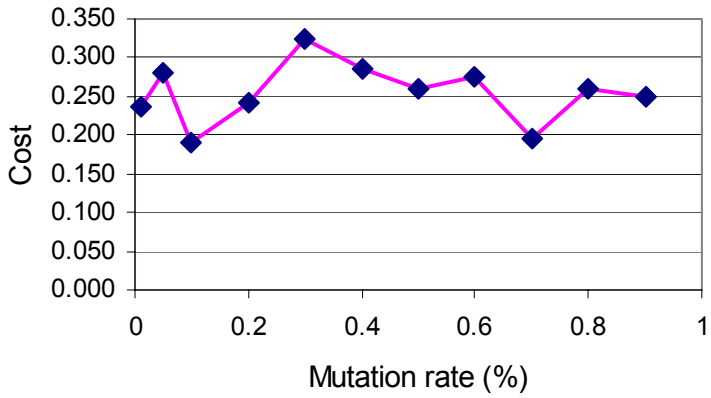

Fig. 6: The effect of the mutation rate on the performance of the proposed algorithm.

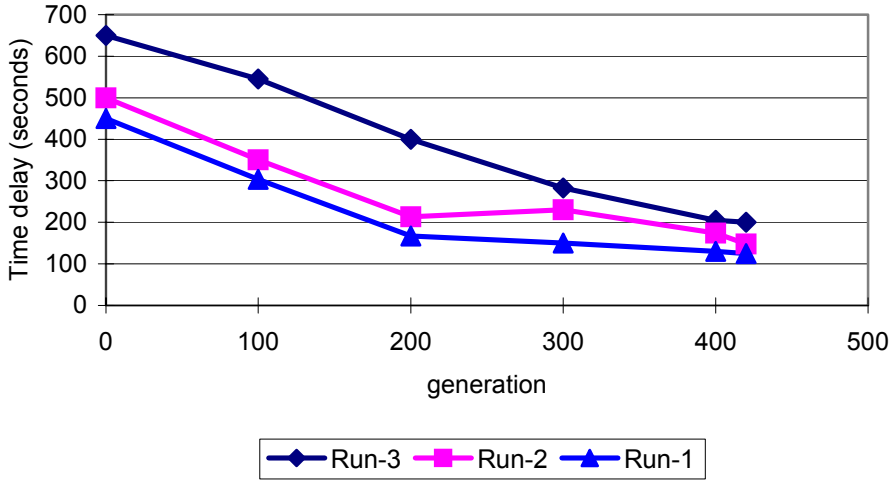

Fig. 7: The optimal time to retrieve query results is 125s by using GA parameters in Run-1

The parameters used in Run-1 provide a good time minimization in retrieving query results by mobile agents as shown in Fig. 7. As there are 15 hosts in the local prefetch (LP) servers, the time used to retrieve query results can be significantly long if some of the hosts serve many agents at one time. In the experiments, the minimum time to retrieve query results is 125 seconds by using GA parameters in Run-1, 180 seconds in Run-2, and 200 seconds in Run-3, respectively.

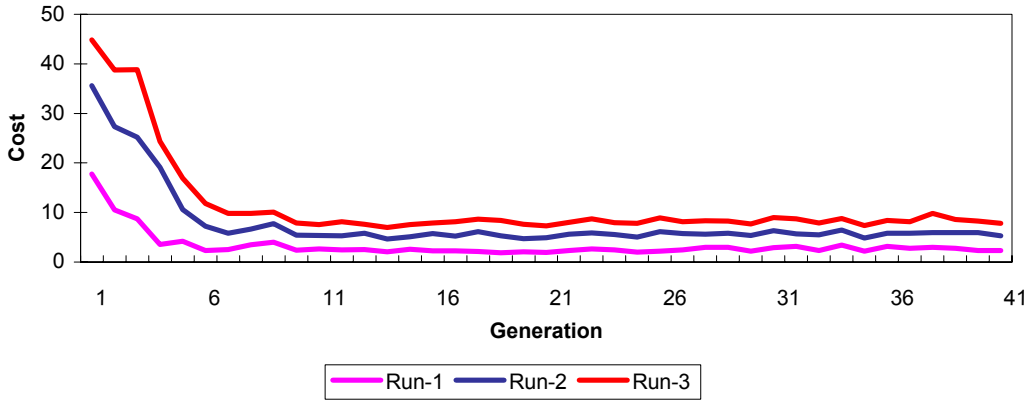

Fig. 8: The effect of the number of generations on the performance of the proposed GA.

7.0 CONCLUSIONS

In this paper, a routing algorithm based on GA applied to a mobile agent for query retrieval in the MaSS was proposed. The proposed mobile agent for routing algorithm tries to minimize the query retrieval cost while maintaining a reasonable path delay. The number of generations required to reach a good solution has been reduced significantly by preferring shorter routes in initializing the chromosome pool and reusing the past solutions as the initial chromosomes for the new search. The simulation results show that, with properly setting of the GA's parameters, such as the size of chromosome pool and the number of generations, the proposed routing algorithm is able to obtain a better solution with different running parameters.

8.0 REFERENCES

[1] P. Maes, Y. Lashkari and M. Metral, "Collaborative interface agents", Readings in Agents, edited by Michael N. Huhns & Munidar P. Singh, Morgan Kaufmann Publishers, Inc., 1997.

[2] R. Schoonderwoerd, O. Holland, and J. Bruten, "Ant-like agents for load balancing in telecommunication networks", in *Proceedings of the First Int. Conf. on Autonomous Agents*, 1997, pp. 209-216, ACM Press.

[3] G. Di Caro, and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communication Networks", *Journal of Artificial Intelligence Research,* Vol. 9, 1998, pp. 317-365.

[4] J. Sum, H. Shen, and C-S. Leung, G. Young, "Analysis on a Mobile Agent-Based Algorithm for Network Routing and Management", *IEEE Transactions on Parallel and Distributed S*ystems, IEEE Press. , Vol. 14, No. 3, 2003, pp. 193-202

[5] H. Kashiwazaki, and H. Takai, "Adaptive Network Routing by Using the Multiagent", in *Proc. of IASTED International Conference Networks, Parallel and Distributed Processing, and Applications*, 2002.

[6] W.-S.E. Chen, C.W.R. Leng, Y-N. Lien, "A Novel Mobile Agent Search Algorithm", in *the Sixth International Conference on Computer Communications and Networks (ICCCN '97*), 1997.

[7] H-S. Park, "Agent Migration Information System for the Efficient Migration of the Mobile Agent", in *V. Kumar et. al. (Eds.): ICCSA2003, LNCS 2668*, pp. 607-613, Springer-Verlag Berlin Heidelberg, 2003.

[8] A. Selamat, S. Omatu, H. Yanagimoto, T. Fujinaka, and M. Yoshioka, "Effectiveness of Mobile Agent for Query Retrieval", IEEJ Transactions on Electronic and Information Systems, Vol. 122, No. 8, 2003, pp. 1367-1373.

BIOGRAPHY

Ali Selamat has received B.Sc.(Hons.) in IT from Teesside University, U.K. and M.Sc. in Distributed Interactive Systems from Lancaster University, U.K. in 1997 and 1998, respectively. He has received a Ph.D. degree from Osaka Prefecture University, Japan in 2003. He was a lecturer at the Department of Graphics and Multimedia, Faculty of Computer Science & Information Systems (FSKSM), UTM from 1999-Sept. 2004. Currently, he is a Head of Department at the Department of Software Engineering, FSKSM, UTM. His research interests include mobile agents, open source technology, information retrieval, bioinformatics, genetic algorithms, neural networks and soft-computing.

Md. Hafiz Selamat has received B.Sc. .(Hons.) and M. Sc. in Computer Science from Universiti Teknologi Malaysia (UTM) in 1988 and 1998, respectively. Currently, he is a lecturer at the Department of Information Systems, Faculty of Computer Science & Information Systems, Universiti Teknologi Malaysia since 2000. His research interests include mobile agent, information retrieval, and knowledge management.

Appendix

Table 1: Cost matrix of simulation.

Host's ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99	15	99	99	99	23	99	99	99	99	99	99	99	99	99
2	15	18	99	99	23	99	99	99	99	99	99	99	99	99	99
3	99	99	15	99	99	99	99	300	99	99	99	99	99	99	99
4	99	23	99	99	99	50	30	99	99	99	99	99	99	99	99
5	99	23	99	99	99	50	30	99	99	99	99	99	99	99	99
6	23	99	99	99	50	99	20	99	18	99	99	99	99	99	99
7	99	99	99	99	30	20	99	10	99	30	30	99	99	99	99
8	99	99	300	99	99	99	10	99	99	99	15	99	99	99	99
9	99	99	99	99	18	99	99	99	99	25	5	99	99	99	99
10	99	99	99	99	99	99	99	99	25	99	99	99	99	99	99
11	99	99	99	99	99	99	99	99	25	99	99	99	99	99	99
12	99	99	99	99	99	99	99	99	25	99	99	99	99	99	99
13	99	99	99	99	99	99	99	99	25	99	99	99	99	99	99
14	99	99	99	99	99	99	99	99	25	99	99	99	99	99	99

											999				