Malaysian Journal of Computer Science, Vol. 16 No. 1, June 2003, pp. 58-67

A GENETIC ALGORITHM SOLUTION TO SOLVE THE SHORTEST PATH PROBLEM IN OSPF AND MPLS

N. Selvanathan and Wee Jing Tee
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur, Malaysia

ABSTRACT

This paper explores the potential of using genetic algorithm to solve the shortest path problem in Open Shortest Path First (OSPF) and Multi-protocol Label Switching (MPLS). The most critical task for developing a genetic algorithm to this problem is how to encode a path in a graph into a chromosome. The proposed approach has been tested on ten randomly generated problems with different weights. The experimental results are very encouraging and the algorithm can find the optimum solution rapidly with high probability.

Keywords: ***OSPF, MPLS, Genetic, Chromosome, Mutation, Crossover***

1.0 INTRODUCTION

One of the most common problems encountered in analysis of networks is the shortest path problem: finding a path between two designated nodes having minimum total length or cost. Several good methods have been proposed to solve the problem to optimality [1]. This study provides an interesting alternative: using genetic algorithms to find out a shortest path.

In data communication and networking, a packet-switching network, or a frame relay or cell network can be viewed in an obvious way as a digraph, with each packet-switching node corresponding to a vertex, and each communication link between nodes corresponding to a pair of parallel edges, each carrying data in one direction. In such a network, a routing decision is needed to transmit a packet from a source node through various links and packet switches to a destination node; this is equivalent to finding a path through the graph.

Network routing in packet-switched networks and the Internet is a major component at the network layer and is concerned with the problem of determining feasible paths or routes from each source to each destination. Most routing algorithms are based on variants of shortest-path algorithms, which try to determine the shortest path for a packet according to some cost criterion.

Open Shortest Path First (OSPF) [2, 3] is the interior gateway routing protocol which uses link-state routing protocol and uses Dijkstra algorithm [4] to find the shortest path for hop-by-hop routing in an Internet. The metrics or weights of the links, and thereby the shortest path routes, can be changed by the network operator.

The Multi-protocol Label Switching (MPLS) technologies [5, 6] proposed by the Internet Engineering Task Force (IETF) is to be the networking technology to deliver traffic engineering capability. MPLS supports traditional hop-by-hop routing and provides explicit-routing capability. The most significant initial application of MPLS is in traffic engineering, which gives network operators a great deal of flexibility to divert and route traffic around link failures and congestion. MPLS traffic engineering employs constraint-based routing (CBR) algorithm, in which the path for a traffic flow is the shortest path that meets the resource requirements (constraints) of the traffic flow.

This paper discusses the implementation of a Genetic algorithm in solving the shortest path problem in OSPF and MPLS. The most thorny problem when applying genetic algorithms to this problem is how to encode a path in a graph into a chromosome. In this paper, a solution called Previous-node-based Encoding to solve the shortest path problem in OSPF by using genetic algorithm instead of the traditional Dijkstra algorithm has been proposed and developed. The proposed algorithm has been tested on ten randomly generated problems with different size. The results show that genetic algorithm can find the optimal solution which is the same as the solution found using Dijkstra algorithm. Next, to solve the shortest path problem in MPLS, another solution called Previous-node-based Encoding using genetic algorithm has been proposed and tested with the same test model. The experimental results

are very encouraging, and the algorithm is able to find the optimum solution, which is the shortest path that meets the resource requirements (constraints) of the traffic flow.

2.0 FORMULATION OF THE SHORTEST PATH PROBLEM IN OSPF

In OSPF, the network operator assigns a weight to each link, and shortest paths from each router to each destination are computed using these weights as lengths of the links. In each router, the next link on all shortest paths to all possible destinations is stored in a table, and a demand going in the router is sent to its destination by splitting the flow between the links that are on the shortest paths to the destination. The exact mechanic of the splitting can be somewhat complicated, depending on the implementation. Here, as a simplifying approximation, we assume that it is an even split. The quality of OSPF routing depends highly on the choice of weight.

In OSPF, the Internet is represented using a weighted directed graph according to RFC 1583 [3]. A weighted directed graph G = (V, E) comprises a set of nodes V=$\{v_i\}$ and a set of edges E $\in$ V x V connecting nodes in V. Corresponding to each edge, there is a non-negative number $\mathbf{w}_{ij}$ representing the cost (distance, transit times, or others of interest) from node v_i to node $\mathbf{v}_j$. A path from node v_i to node v_j is a sequence of edges ($\mathbf{v}_i$, $\mathbf{v}_l$), ($\mathbf{v}_l$, $\mathbf{v}_m$), …, ($\mathbf{v}_h$, $\mathbf{v}_j$) from E in which no node appears more than once. A path can also be equivalently represented as a sequence of nodes ($\mathbf{v}_i$, $\mathbf{v}_l$, $\mathbf{v}_m$, …, $\mathbf{v}_k$, $\mathbf{v}_j$). The problem is to find a path between two given nodes having minimum total cost. The integer programming model is formulated as follows:

$$min \quad \sum_i \sum_j \boldsymbol{w}_{ij}\boldsymbol{x}_{ij}$$

where $\mathbf{x}_{ij}$ is an indicator variable defined as follows:

$$\boldsymbol{x}_{ij} = \begin{Bmatrix} 1, \text{ if edge } (i, j) \text{ is included in the path} \\ 0, \text{ otherwise} \end{Bmatrix}$$

In order to achieve Internet Traffic Engineering, we could optimise the OSPF weights as proposed in [7]. The objective function is to minimise the total cost (metric) of the network, which is the summation of 2 parameters, i.e. distance (d) and utilisation's weight (w(u)). The integer programming model is formulated as follows:

$$\phi'_{\min} = \sum_i \sum_j \left(\boldsymbol{d}_{ij}\boldsymbol{x}_{ij} + \boldsymbol{w(u)}_{ij}\boldsymbol{x}_{ij} \right)$$

Having decided on a routing, the load l(a) on an arc a is the total flow over a, that is l(a) is the sum over all demands of the amount of flow for that demand which is sent over a and c(a) is the capacity of a. The utilisation of a link a, u(a), is defined as follows:

$$\boldsymbol{u(a) = l(a) / c(a)}$$

Loosely speaking, our objective is to keep the loads within the capacities. More precisely, our cost function sums the cost of between l(a) and c(a). Here, A represents a set of all links and a is a particular link. In our experimental study, we had,

$$\Phi = \sum_{a \in A} \Phi_a(\ell(a))$$

where for all $a \in A, \Phi_a(0) = 0$ and

$$\Phi_a(\ell(a)) = \begin{Bmatrix} 1 & for & 0 \le \ell(a)/c(a) < 1/3 \\ 3 & for & 1/3 \le \ell(a)/c(a) < 2/3 \\ 10 & for & 2/3 \le \ell(a)/c(a) < 9/10 \\ 70 & for & 9/10 \le \ell(a)/c(a) < 1 \\ 1073741824 & for & 1 \le \ell(a)/c(a) < 11/10 \\ 2147483647 & for & 11/10 \le \ell(a)/c(a) < \infty \end{Bmatrix}$$

and the maximum value is 2147483647, which is the maximum value of integer data type. The reason we choose this maximum value is because it is more effective and efficient in programming since the maximum value of integer data type is provided as a constant in Java standard library. The graph, which reflects the above cost function, is illustrated in Fig. 1.

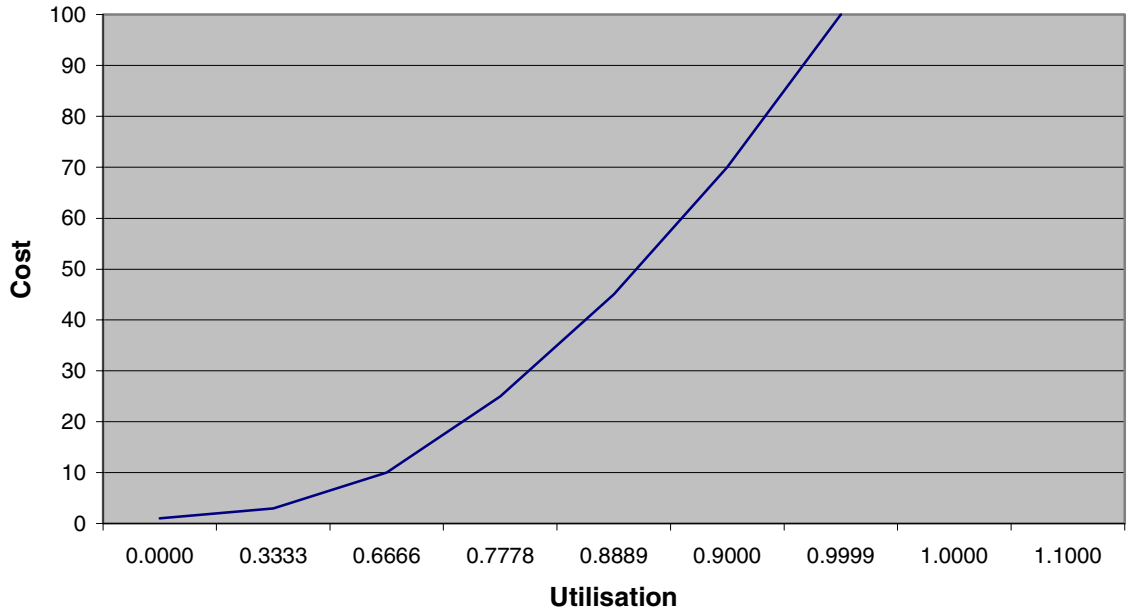

Fig. 1: Graph for the Cost versus Utilisation function

The above definition of the general routing problem and the objective function is equivalent to the one used in Bernard et. al. [7]. The idea behind Φ_a is that it is cheap to send flow over an arc with a small utilisation. As the utilisation approaches 100%, it becomes more expensive, for example because the algorithm becomes more sensitive to bursts. If the utilisation goes above 110%, the penalty becomes so high that this should never happen.

3.0 FORMULATION OF THE SHORTEST PATH PROBLEM IN MPLS EXPLICIT-ROUTING

MPLS is an IETF initiative that integrates Layer 2 information about network links (bandwidth, latency, utilisation) into Layer 3 (IP) within a particular autonomous system or ISP in order to simplify and improve IP packet exchange. MPLS gives network operators a great deal of flexibility to divert and route traffic around link failures, congestion, and bottlenecks.

MPLS supports traditional hop-by-hop routing and provides explicit-routing capability for traffic engineering. While the hop-by-hop routing follows the path that normal Layer 3 routed packets will take, the explicit-routing can be specified and controlled by the network operators or network management applications to direct the network traffic, independent of the Layer 3 topology. In this way, the explicit-routing gives network operators a great deal of flexibility to divert and route traffic around link failures, congestion, and bottlenecks.

In MPLS, the Internet is represented using a weighted directed graph. A weighted directed graph G = (V, A) comprises a set of nodes, V={v_i} and a set of edges, A ∈ V x V, connecting nodes in V. Let $\boldsymbol{x}_{ij}$ be an indicator variable defined as follows:

$$\boldsymbol{x}_{ij} = \begin{cases} 1, \text{if edge } (i, j) \text{ is included in the path} \\ 0, \text{otherwise} \end{cases}$$

Corresponding to each edge, there is a non-negative number d_{ij} (distance) and c_{ij} (capacity) from node *vi* to node *vj*. The objective function is to minimise the total cost (metric) of the network, which is the summation of 2 parameters, i.e. distance *(d)* and utilisation's weight *(w(u))*. The integer programming model is formulated as follows:

$$\phi'_{min} = \sum_i \sum_j (\boldsymbol{d}_{ij}\boldsymbol{x}_{ij} + \boldsymbol{w}(\boldsymbol{u}_{ij})\boldsymbol{x}_{ij})$$

As in the case of OSPF, our objective in MPLS is also to keep the loads within the capacities. More precisely, our cost function sums the cost of between l(a) and c(a). In our experimental study, we had the utilisation of the link a defined as $Ua = f_a / c_a$, for all a ∈ A, and

$$\Phi_a(\ell(a)) = \begin{cases} 1 & for \quad 0 \le \ell(a)/c(a) < 1/3 \\ 3 & for \quad 1/3 \le \ell(a)/c(a) < 2/3 \\ 10 & for \quad 2/3 \le \ell(a)/c(a) < 9/10 \\ 70 & for \quad 9/10 \le \ell(a)/c(a) < 1 \\ 1073741824 & for \quad 1 \le \ell(a)/c(a) < 11/10 \\ 2147483647 & for \quad 11/10 \le \ell(a)/c(a) < \infty \end{cases}$$

4.0 TESTING ON THE GENETIC ALGORITHM PARAMETERS

Initial experiments were carried out to study the effects of the genetic algorithm parameters towards the algorithm performance in finding the optimal solution. The performance issues are diversity and convergence. The setting of genetic algorithm parameters used in the experiment includes population size, maximum generation, crossover rate, mutation rate, and overlapping rate for the steady state genetic algorithm. The matrix experiment layouts are as in Table 1. A total of 100 tests for network 1 (N1) have been tested for each experiment set.

Table 1: Results of the matrix experiment

Experiment Set	Genetic Algorithm Parameters					Performance (average=X/100)	
	Overlapping Ratio (%)	Population Size (unit)	Generation Size (unit)	Crossover Rate (%)	Mutation Ratio (%)	Diversity (cost value)	Convergence (generation value)
Control	50	500	30	80	25	199	23.8
1(a)	50	500	30	100	0	205.9	25.8
1(b)	50	500	30	0	100	254.5	15.9
2(a)	10	500	30	80	25	237.7	22.1
2(b)	90	500	30	80	25	241.3	21.8
3(a)	50	10	30	80	25	534	9.9
3(b)	50	1000	30	80	25	199.4	25.1
4(a)	50	500	10	80	25	281.8	8.5
4(b)	50	500	50	80	25	199.4	24.6

The results of the matrix experiment are as shown in Table 1. The performance issues discussed here are diversity and convergence. As discussed in the previous section, the diversity of the population will determine the scope of the choice of the solution encoded in the chromosome. The higher the diversity the larger the search scope and the more choices of solution we can have. However, the higher the diversity, the longer the time it takes to converge and reach the optimal solution. Ideally, the optimal solutions should have the minimum cost value (diversity) and minimum generation value (convergence). However, there is often a trade-off between these two performance issues. Here, in the shortest path problem, we want to achieve minimum cost value within a reasonable time (generation value). From the graph of the experiment results as in Fig. 2, the optimal solutions are obtained from the control experiment set with the parameter setting of:

(a) population size = 500,
(b) maximum generation = 30,
(c) crossover rate = 80%,
(d) mutation rate = 25%, and
(e) overlapping rate for the steady state genetic algorithm is = 50%.

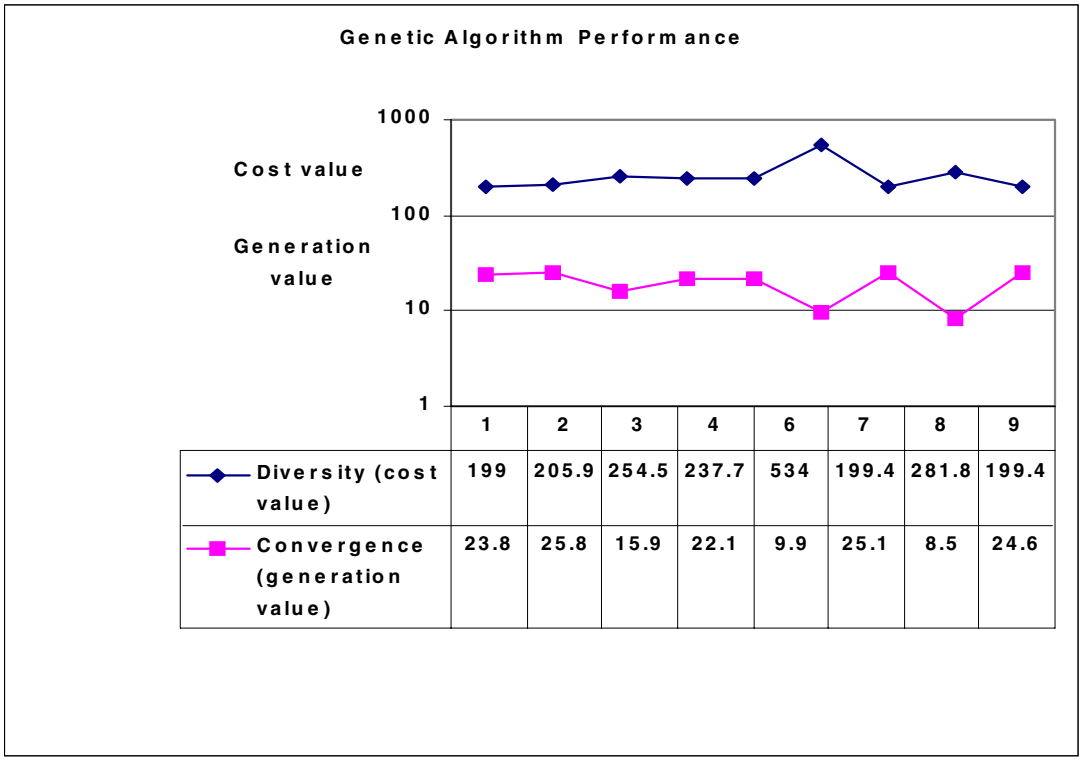

Fig. 2: The graph of the matrix experiment results

5.0 SOLUTION METHOD I: PREVIOUS-NODE-BASED ENCODING FOR OSPF HOP-BY-HOP ROUTING

How to encode a path in a graph is critical for developing a genetic algorithm to this problem. It is not as easy as the Traveling Salesmen Problem (TSP) to find out a nature representation. Special difficulties arise from: (a) a path contains a variable number of nodes and the maximal number is n-1 for a n node graph, and (b) a random sequence of edges usually does not correspond to a path. To overcome such difficulties, we adopted an indirect approach: encode some guiding information for constructing a path, but a path itself, in a chromosome. The path is generated by a sequential code appending procedure, beginning at a specified node 1 and terminating at a specified node n. As we know, a gene in a chromosome is characterised by two factors:

(a) allele, the value the gene takes, and
(b) locus, the position of the gene located within the structure of chromosome.

In the proposed previous-node-based encoding method, the position of a gene is used to represent node ID and the value is used to represent the previous node of the current node for construction a path among candidates. A shortest path tree from the start node to all the other nodes can be uniquely determined from this encoding. Let us see an example of a network, which consists of 6 nodes and 13 links as shown in Fig. 3.

Suppose we are going to find a path from node 1, the start node to all the other nodes. From Dijkstra algorithm, the shortest path tree from the start node (N0) to all the other nodes is shown in Fig. 4. Using genetic algorithm, the solution obtained is the same as the solution obtained from Dijkstra algorithm.

To encode the solution into a chromosome, the previous-node-based encoding method is used as shown in Fig. 5, where the locus of a gene is used to represent node ID and the allele is used to represent the previous node of the current node.

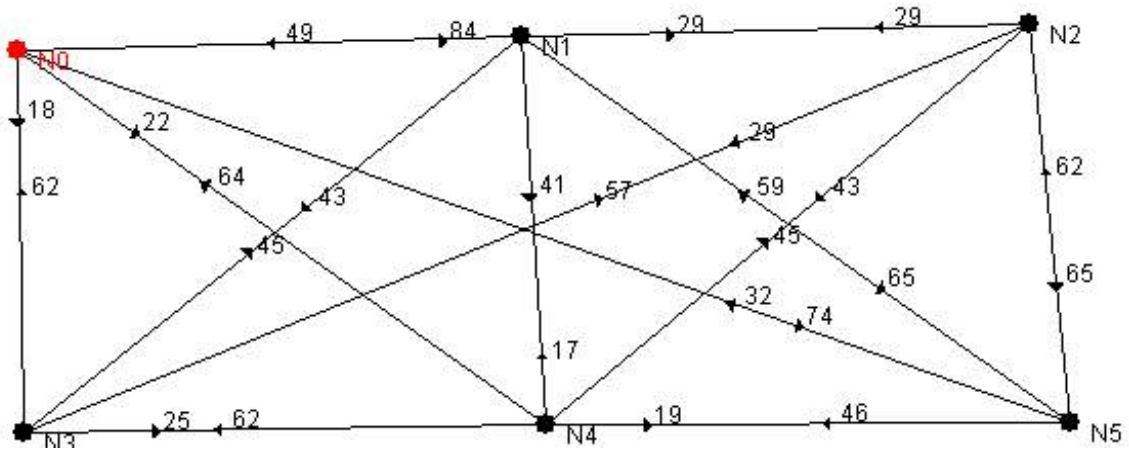

Fig. 3: Example of mesh topology network, which consists of 6 nodes and 13 links

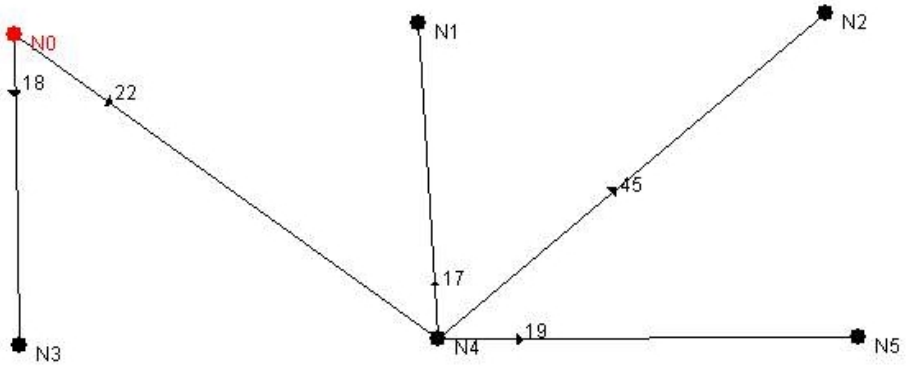

Fig. 4: The shortest path tree obtained from Dijkstra algorithm

N0	N1	N2	N3	N4	
0	4	4	0	0	4

Locus: Node ID

Allele: Previous node ID

Fig. 5: Previous-node-based encoding in a chromosome

In this proposed solution, the steady-state genetic algorithm is used to find the optimal solution. This algorithm uses overlapping populations; only a portion of the population is replaced in each generation. The amount of overlap (percentage of population that is replaced) may be specified when tuning the genetic algorithm. The use of a genetic algorithm requires the definition of initialisation, crossover, and mutation operators specific to the data type in the genome. The crossover operator is the array one-point crossover with the condition that the crossover node ID of the 2 parent chromosomes must be the same number. Mutation is performed by the replacement of the mutation operator, where a node ID (locus) is randomly selected, then the previous node (allele) is replaced with another random node. Here, the roulette wheel approach is adopted as the selection procedure, which is one of the fitness-proportional selections to produce the next generation. One can evaluate the individuals in a population using an individual-based evaluation function, or a population-based evaluator. Here, we use an individual-based objective, then the function is assigned to each genome. The genome performance measure, often referred to as the objective function, is based upon the path the genome represents. The objective function calculation is straightforward. Each link consists of a weight. The fitness value of a genome is calculated by adding up all the weights of the links used in the path. The fitness scoring is the reverse of the fitness value. So, the higher the weight a genome has, the lower is its fitness scoring.

6.0 SOLUTION METHOD II: PRIORITY-BASED ENCODING FOR MPLS EXPLICIT-ROUTING

To solve the shortest path problem in MPLS explicit-routing, the priority-based encoding has been proposed. In the proposed priority-based encoding method, the position of a gene is used to represent node identity (ID) and this value is used to represent the priority of the node for construction a path among candidates. A path can be uniquely determined from this encoding. Priority-based encoding in a chromosome follows the following guidelines:

- The node with the biggest priority value = start node
- The node with the smallest priority value = end node
- Possible next node with bigger priority value will be selected

A simple graph and its adjacency metrics are given in Fig. 6 and Fig. 7. Let us see an example of a chromosome generated randomly as shown in Fig. 8.

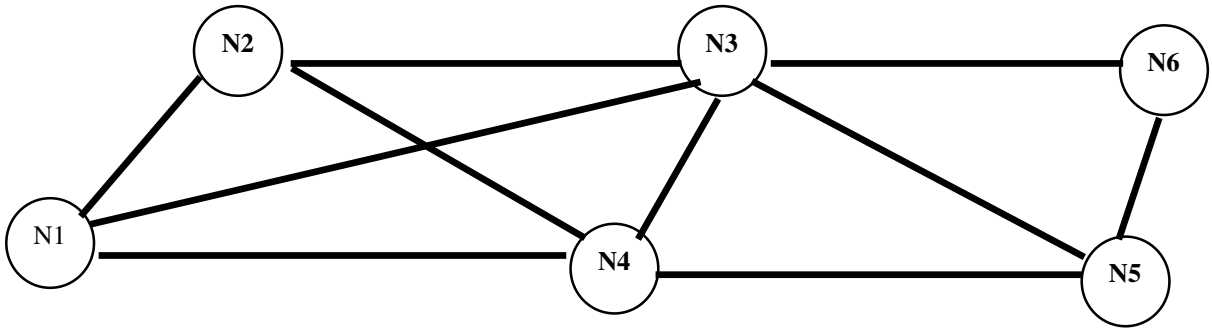

Fig. 6: A simple graph

	V_1	V_2	V_3	V_4	V_5	V_6
V_1	0	1	1	1	0	0
V_2	1	0	1	1	0	0
V_3	1	1	0	1	1	1
V_4	1	1	1	0	1	0
V_5	0	0	1	1	0	1
V_6	0	0	1	0	1	0

Fig. 7: The adjacency metrics

N1	N2	N3	N4	N5	N6
3	4	2	5	6	1

Position: Node ID

Allele: Priority Value

Fig. 8: An example of chromosome generated randomly

Based on the guidelines of priority-based encoding, the node with the biggest priority value, which is node 5 (N5) will be the start node and the node with the smallest priority value, which is node 6 (N6), will be the end node. Suppose we are going to find a path from N5 to N6. At the beginning, we try to find a node, which is connected to node 5. Nodes 3, 4 and 6 are possible for consideration. This can be easily fixed due to adjacent relation among nodes. The priorities for them are 2, 5 and 1 respectively. Node 6 is the end node and is not considered. The node 4 has the highest priority and is put into the path. Then we form the set of nodes available for next consideration and select the one with the highest priority among them. Repeat the steps until we obtain a complete path (5, 4, 2, 1, 3, 6). It is easy to see that any permutation of the encoding can always yield to a path as shown in Fig. 9.

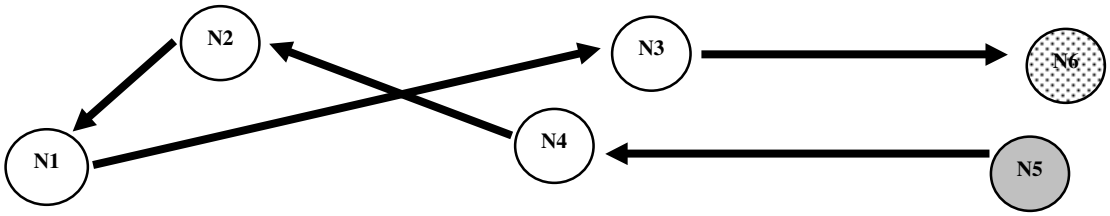

Fig. 9: Permutation of the encoding which yields to a valid path

In this proposed solution, the steady-state genetic algorithm is used to find the optimal solution. This algorithm uses overlapping populations; only a portion of the population is replaced in each generation. Use of a genetic algorithm requires the definition of initialisation, crossover, and mutation operators specific to the data type in the genome. Here, the roulette wheel approach is adopted as the selection procedure, which is one of the fitness-proportional selections to produce the next generation. The crossover operator is the array two-points crossover with the condition that the crossover node ID of the 2 parent chromosomes must be the same number. Mutation is performed by swapping mutation operator where two nodes (locus) are randomly selected and the priority values (allele) are swapped. In MPLS, the objective function calculation is quite straightforward. Each link consists of a total weight, which is the summation of two parameters, i.e. distance *(d)* and utilisation's weight *(w(u))*. The fitness value of a genome is calculated by adding up all the weights of the links used in the path. The fitness scoring is the reverse of the fitness value. So, the higher the weight a genome has, the lower is its fitness scoring.

7.0 COMPUTATIONAL EXPERIMENTS AND RESULTS

In this section, we describe some computational experiments comparing the proposed method using genetic algorithm with traditional algorithm approach. A routing simulator is developed using Java programming language to run the computational experiments. All algorithms were written in the Java programming language using Java 2 Standard Development Kit (J2SDK) Version 1.3 and run on the Window 98 operating system with 1 GHz Pentium PC and 128 MB of memory. Table 2 is the test results of the overall computational experiments using the simulator. In order to test the performance of the proposed algorithm, ten randomly generated networks (N1 – N10) with different weights setting using mesh topology of nine nodes and thirty six edges have been selected with the same condition for each of the shortest path problem domain.

From the 100 test results as shown in Table 2, we can conclude that:

- In OSPF hop-by-hop routing, genetic algorithm can find the optimal solutions exactly the same as in Dijkstra algorithm. The frequency of obtaining the optimal solutions using genetic algorithm is 100%.
- In OSPF hop-by-hop routing with optimised weight setting, genetic algorithm can find an optimal solution to avoid congestion to achieve traffic engineering. The frequency to avoid the congestion using genetic algorithm is 100%.
- In MPLS explicit-routing, genetic algorithm is able to find the optimum solution, that is the shortest path that meets the resource requirements (constraints) of the traffic flow. The frequency of obtaining the optimal solutions using genetic algorithm is 100%.

Table 2: Test results of the overall computational experiments

		OSPF HBH-RT	OSPF HBH-RT*	MPLS EX-RT
		Dijkstra's/Genetic Algorithm	Genetic Algorithm	Genetic Algorithm
Network No.	Re-Test	Congested Links/Test	Congested Links/Test	Congested Links/Test
N1	100	3	0	0
N2	100	2	0	0
N3	100	3	0	0
N4	100	1	0	0
N5	100	3	0	0
N6	100	5	0	0
N7	100	4	0	0
N8	100	3	0	0
N9	100	1	0	0
N10	100	3	0	0
Total Congested Links		N	N-N	N-N

*optimised weights

8.0 DISCUSSION AND FUTURE RESEARCH

The potential of genetic algorithms for solving the shortest path problem in minimising the weight has been studied and the results are very encouraging: it can find the known optimum very rapidly with very high probability [8, 9]. In general, evaluation of the relative merits of an algorithm should consider the processing time of the algorithms and the space involved. The purpose of these studies [8, 9], however, is not to compare the genetic algorithms with conventional algorithms, because genetic algorithms will be unable to compete in both space and time [10, 11]. As a variety of network optimisation problems could be solved, either exactly or approximately, by identifying the shortest path, these studies provide a base for constructing an effective solution for more specific shortest-path based network optimisation problems.

In this study, evaluation of the relative merits of a genetic algorithm solution is focus on the optimal solution which minimising the space, which is the total weight. This project demonstrates the potential of using genetic algorithms to solve the shortest path problem in optimising the weight in OSPF and MPLS. This solution can provide a potential alternative for such problem. If the genetic algorithm solution can reduce congestion and the elimination of many of the manual configuration tasks, then the genetic algorithm solution is justified to be a potential alternative for such problems.

Although genetic algorithm has been proven in this study to be an effective solution for optimising the space but genetic algorithm is also a computation-intensive algorithm. In order to enhance the performance of the genetic algorithm solution, the development of more systematic techniques for the shortest path problem including parallel genetic algorithm is subject of future research. Moreover, an analysis of the computation time and memory space required is also needed.

Further research should be carried out to develop a hybrid solution combining the strength of the genetic algorithm and traditional algorithm to overcome the weaknesses of genetic algorithm at this stage. An example of an algorithm based on hybridisation of genetic algorithm and the shortest path routing is proposed in [9]. The hybrid algorithm has shown simplification in implementation and improvement in the performance in solving the joint optimisation of capacity and flow assignment in the packet switched network.

9.0 CONCLUSIONS

In this study, the potential of using genetic algorithm to solve the shortest path problem in Open Shortest Path First (OSPF) hop-by-hop routing, and Multi-protocol Label Switching (MPLS) explicit-routing has been explored. The most critical task for developing a genetic algorithm to this problem is how to encode a path in a graph into a chromosome. For each of the shortest path problem domain, the proposed solution model has been tested on ten randomly generated networks with different weights setting using mesh topology of nine nodes and thirty six edges using the routing simulator. The test runs were performed with tuning of the genetic algorithm parameters. From the testing and results analysis, the genetic algorithm solution has achieved the following results:

(a) In OSPF hop-by–hop routing, genetic algorithm is able to find the optimum solution, which is the shortest path with the minimum weight.

(b) In MPLS explicit-routing, genetic algorithm is able to find the optimum solution, which is the shortest path that meets the resource requirements (constraints) of the traffic flow.

As a conclusion, genetic algorithm could be a promising approach for solving the shortest path problem. In the long term, as with natural evolution, the strengths of this mutation may be combined with the strengths of current practices to yield a superior hybrid protocol. The eventual result of this work, when combined with other ongoing research, will be improved reliability and performance on tomorrow's networks.

REFERENCES

[1] A. Ravindran, D. T. Phillips, and J. J. Solberg, *Operations Research: Principles and Practice*. John Wiley & Sons, 1987.

[2] J. Moy, "OSPF Version 2", *Request For Comment 2328*. Internet Engineering Task Force, April 1998.

[3] J. Moy, "OSPF Version 2", *Request For Comment 1583.* Internet Engineering Task Force, March 1994.

[4] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs". *Numerical Mathematics*, October 1959.

[5] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering over MPLS". *Internet Request for Comments 2702*, Internet Engineering Task Force, September 1999.

[6] E. C. Rosen, A. Viswanathan, and R. Callon, "Multi-Protocol Label Switching Architecture". *Internet Draft (work in progress),* Internet Engineering Task Force, 1999.

[7] Bernard Fortz and Mikkel Thorup. "Internet Traffic Engineering by Optimizing OSPF Weights", in *INFOCOMM 2000*. IEEE, March 2000.

[8] Mitsuo Gen, Runwei Cheng, and Dingwei Wang, "Genetic Algorithms for Solving Shortest Path Problems". *IEEE Computer*, Vol. 30, No. 8, 1997, pp. 401-406.

[9] Mohamed E. Mostafa, and Saad M. A. Eid, "A Genetic Algorithm for Joint Optimization of Capacity and Flow Assignment in Packet Switched Networks", in *17th National Conference*, Egypt, Feb. 2000, pp. 1-6.

[10] Z. Michalewicz, *Genetic Algorithm + Data Structure = Evolution Programs*. Springer-Verlag, New York, second edition, 1994.

[11] M. Gen, and R. Cheng, *Genetic Algorithms and Engineering Design*. John Wiley & Sons, 1987.

BIOGRAPHY

N. Selvanathan is an Associate Professor and Deputy Dean of the Faculty of Computer Science and Information Technology, University Malaya. His research interest is in Medical Imaging, Computer Aided Design (CAD), Modelling, and in the Application of Artificial Intelligence in solving Computer Science Problems. He has a keen interest in the synergy of Physics, Consciousness, Indian Philosophy and Artificial Intelligence.

Wee Jing Tee obtained his Masters in Computer Science degree from University Malaya. At present he is a lecturer in Computer Science at Multimedia University. His research interest are networking and Artificial Intelligence.