Research Article

International Journal of Advanced Computer Research, Vol 9(40)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.SOC20

Optimal path calculation for virtual networks using genetic algorithm

Man Soo Han*
Professor, Department of Information and Communications, Youngsan-ro, Muan-gun, Jeonnam, Republic of Korea

Received: 28-May-2018; Revised: 25-July-2018; Accepted: 5-October-2018

Abstract
With the advent of software- defined networks, network virtualization becomes a key technology to implement software-defined networks. Network virtualization requires a path computation element (PCE) to calculate virtual paths to connect virtual network nodes. The Dijkstra's algorithm has been widely used in the PCE to calculate the shortest path between two virtual nodes. In this paper, we address that the Dijkstra's algorithm cannot be applicable when a non-linear cost metric is used in the path cost evaluation. This paper proposes a new genetic algorithm (GA) to find the shortest path when a non-linear metric is used. The proposed GA generates the immigrants from ordinary chromosomes not from the elite chromosome for the genetic diversity. Also, the proposed GA does not use the sorting process to replace the worst chromosomes. The proposed GA uses the random replacing mechanism to decrease the path computation time. Using simulations, we showed that the proposed method is better than existing algorithms.

Keywords
PCE, Genetic algorithm, Shortest path, Virtual network.

1.Introduction
Network virtualization is that a network operator provides for a customer a virtual network abstracted from a physical network that does not interfere with other customers' virtual networks. It requires a virtual network embedding that maps a requested virtual network of a customer into substrate networks of a provider. Virtualization makes a challenging issue usually referred to as the virtual network embedding (VNE) problem [1]. VNE problem solving is known as NP-hard that makes it hard to get an exact solution for large networks in a reasonable time. Because of the NP-hard complexity, many solutions are based on heuristic or meta-heuristic approaches. Most of them use the Dijkstra's algorithm for path computation element (PCE) to find the best path between network nodes where PCE plays an important role of optimal virtual network embedding into underlying substrate networks. PCE is a network entity that calculates the best routing path between a source node and a destination node [2]. Current calculation algorithms of the PCE are typically based on a linear metric like link costs that are used to model the capacity of network nodes to transmit packets.

Dijkstra's algorithm has been widely used for calculating a best routing path in networks for the PCE with the linear metric. Dijkstra's algorithm calculates the best path for the linear metric problem in a simple manner and within reasonable time. However, we find out that the Dijkstra algorithm cannot be applied in some cases that have performance costs in complex forms, especially when both link and performance costs are combined in a non-linear form.

Many papers studied meta-heuristic algorithms for the VNE problem. In [3], a genetic algorithm (GA) based on a non-dominated sorting based multi-objective evolutionary algorithm is introduced to the VNE problem. But the link path between nodes is calculated by a shortest path first (SPF) method which is based on Dijkstra's algorithm. In [4], a particle swarm optimization method is used for the VNE problem. However, the SPF algorithm is used for calculation of the link path between nodes. In [5], an ant colony algorithm is introduced to the VNE problem. But, the link path between nodes is calculated by the SPF algorithm. We find out that the above methods cannot be applied to networks with non-linear costs, as all the methods above use the SPF algorithm for path computation. It is needed to

*Author for correspondence

find a new algorithm for path computation to support VNE with non-linear costs.

We studied GA as a meta-heuristic method to support non-linear metrics considering both link and performance costs. A standard GA (SGA) is well described in [6] and [7] where only link cost is considered. The SGA is studied for the PCE problem with the non-linear cost in [8]. A random immigrant GA (RIGA) is introduced in [6] that random immigrant individuals are added to the genetic population for the search diversity. An elitism-based immigrant GA (EIGA) is introduced in [9] that immigrant individuals are generated from an elite individual and then added to the genetic population. It was shown that the EIGA outperforms the SGA and the RIGA in dynamic environments.

In [10], immigrant individuals are generated by a heuristic method and then inserted the genetic population. In [11], a large population is split into several small populations. Each small population independently genetically evolves. Random immigrants are added to some small populations to enhance the genetic diversity. Since not only the quality of the solution, but also the computation speed is important in the path computation of VNE, the methods such as [10] and [11] are not desired because of their complexity. Network applications require fast path setup times [12]. The setup time is expected to be less than 100 ms for future applications [12]. We propose a noble fast path computation algorithm with a non-linear metric by improving disadvantages of EIGA.

Since the EIGA generates immigrants using an elite individual, the genetic diversity of immigrants is limited in the EIGA. In the proposed algorithm, the immigrants are generated from normal individuals not from the elite individual for the genetic diversity. Since the EIGA replaces the worst individuals with the immigrants, a sorting mechanism is required to sort individuals based on their fitness. The sorting mechanism consumes time and computing power. To decrease the time burden, the proposed algorithm randomly replaces individuals with the immigrants without using the sorting mechanism. We show that the proposed method is better than the EIGA and the SGA in the solution quality and the proposed method is comparable to the EIGA in the convergence speed using computer simulations.

In section 2, we address the current computation algorithms of the shortest path are limited to a linear metric like link costs. In section 3, we describe the design process of GA to support a non-linear metric consisting of both link and node costs. Then, we explain the proposed algorithm in section 4. In section 5, we evaluate the performance of the proposed algorithm and existing algorithms. Finally, conclusions are given in section 6.

2.Limitation of Dijkstra's algorithm

Dijkstra's algorithm is the most widely used method to calculate the shortest path in VNE. Also, it is the basic method to calculate the optimal path in the open shortest path first (OSPF) protocol of PCE technology. The execution time of Dijkstra's algorithm is in time $O(|V|^2)$ where $|V|$ is the number of nodes. If a Fibonacci heap is used then its runtime is in $O(|V| + |E|\log|E|)$ where $|E|$ is the number of links. Therefore, the runtime of Dijkstra's algorithm is increased as the number of nodes or the number of links is increased.

As we mentioned before, a metric can be complex in VNE technology. For example, a metric can be a combination of a link cost and a node failure cost. The node failure cost can be the sum of the node failure rates or the largest one among the node failure rates. In the former case, it is hard to know a failure rate of an individual node from the cost. In the latter case, it is obvious that the failure rate of every node is less than or equal to the cost. If we use the former cost, a node with a high failure rate can be included in a solution path despite its high failure rate. In contrast, a node with a high failure rate is easily avoided when a path is calculated if we use the latter cost.

When a path passes through a node, the node allocates a service rate as well as a buffer space for the path. The service rate is related with a mean delay and the buffer space is associated with a packet loss rate. In every node on the path, both indexes have to be larger than or equal to what the path requires. The sum of service rates of nodes does not provide sufficient information, whether or not a service rate of a node is satisfied. Also, the sum of buffer spaces of nodes does not give satisfactory information if a right amount of buffer size is allocated to the path to a node. For both indexes, the costs of the minimum value type are better. If the minimum service rate among the service rates of the nodes on the path is larger than the requirement, all of the nodes satisfy the service rate requirement. Also, if the minimum buffer size among the buffer sizes of the nodes is

larger than the requirement, all of the nodes meet the buffer size requirement.

Resource reservation protocol-traffic engineering (RSVP-TE) supports a path computation for a label switched path (LSP) of multiprotocol label switching (MPLS) by considering constraints such as the bandwidth requirement and resource attributes [13]. The resource attributes can be a link bandwidth or a buffer resource [14]. In MPLS networks, constraint-based routing is used for the path computation with the constraints. The basic idea of constraint-based routing is to remove any network elements that do not satisfy the constraints [14]. Then it finds a path by running the shortest path algorithm on the residual network.

Constraint-based routing can give a feasible path, but not the best path. For example, consider two paths (A-B-D) and (A-C-D) where A, B, C, and D are nodes. Assume the link cost of all links between nodes is the same. Suppose that the constraint is the available service rate of each node and it must be larger than or equal to 100 Mbps. Let the available service rates of all nodes be 1 Gbps except the node B whose available service rate is 100 Mbps. Since both paths satisfy the constraint and both paths have the same link costs, constraint-based routing can select the path (A-B-D) for the solution. However, the path (A-C-D) is the best in the sense of load balancing. Moreover, a bandwidth of an LSP can be increased [15]. We cannot use the path (A-B-D) if the service rate is increased over 100 Mbps.

The shortest path algorithm such as Dijkstra's algorithm is not efficient when a metric is not a linear form. *Figure 1* shows an example that Dijkstra's algorithm fails. The path cost is a sum of link costs and the maximum node cost among the nodes on a path. That is the path cost c is given by $c = \sum L_i + \max\{n_j \mid j \in P\}$ where L_i is a cost of link i, n_j is a cost of node j, and $i,j \in P$ and P is a path. Let the path cost of a node k be the path cost from a source node to the node k. In the *Figure 1*, the alphabet in a circle means a node number and the number on a link represents a cost of the link. Also the number above a circle denotes a cost of the node. The start node is A and the end node is F.

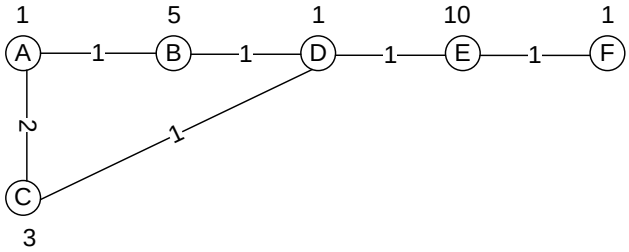

Figure 1 Example of Dijkstra's algorithm failure

Dijkstra's algorithm calculates the shortest path by using hop by hop approach. The shortest path between the source node A and the node D is given by A-C-D and the path cost of D is 6 where the sum of link costs is 3 and the node cost is 3. Since Dijkstra's algorithm calculates the path cost of the node E by adding a cost to the path cost of the node D, the shortest path between the source node A and the node E is given by A-C-D-E and the path cost of E is 14 where the sum of link costs is 4 and the node cost is 10. The shortest path between the source node A and the node F is calculated as A-C-D-E-F and its path cost is 15. However, the true shortest path is A-B-D-E-F and its path cost is 14. Therefore, Dijkstra's algorithm cannot find the true shortest path in this example.

3.Design of GA for shortest path

3.1Model

In this paper, the network is modeled by an undirected and connected topology graph G(V,E) where V is a set of nodes and E is a set of links that connect nodes. We summarize some notations used in this paper.

- G(V,E): the topology graph.
- s: the source node.
- t: the destination node.
- P(s,t): a path from s to t on G(V,E).
- L_i: the cost of the link i.
- n_j: the cost of the node j.
- c(P): a total cost of path P.

The path computing problem can be explained as follows. For a given network, link costs, node costs, a path cost function, a source node and a destination node, we wish to find a path that minimizes the path cost. The two main objectives of the problem are the optimality of the solution and the computation speed.

The optimal solution minimizes the network resource waste and provides a better quality of service (QoS). For example, the longer a path, the more resources have to be reserved for the path. Since nodes and links take some time to process packets, a packet delay is affected by the path length.

The path setup times for network applications have to be low. Future applications are expected to require setup times as fast as 100 ms [12]. Recent applications such as data centers, cloud computing, video, gaming, and mobile can increase connection request rates [12]. For these applications, the rapid path setup has to be provided. Thus the computation speed for the given problem has to be fast. Our aim is to develop a fast GA that finds an optimal path that minimizes a path cost.

More formally, consider a network G(V,E) and a path setup request from the source node s to the destination node t. The shortest path problem is to find a path P over a network G(V,E) which minimizes the path cost as shown in Equation (1).

$$c(P) = \alpha \sum L_i + \beta \max \{n_j | j \in P\} \quad (1)$$

Where L_i is a cost of link i, n_j is a cost of node j, and i,j ∈P. Also α and β are proportional coefficients.

3.2Genetic representation

A routing path is encoded by a string of numbers that denote nodes which the path passes through. The order in the string represents a node order in the routing path. The string is called a chromosome in GA. The first locus of the chromosome is the source node and the last locus of the chromosome is the destination node of the path. The chromosome length is less than or equal to the maximum length |V| which is the total number of nodes.

3.3Initial population

In GA, a chromosome is a genetic representation of a possible solution. To obtain a good solution, the initial population has to be genetically diverse. In this paper, the chromosomes of the initial population are randomly generated for the diversity. Also let s and t be the source node and the destination node, respectively. We build a path from the node s to the node t by randomly selecting a neighbor node. A node u is a neighbor node of a node v if the node u is directly connected to the node v via a link. First, we randomly select a neighbor node, v_1, of the source node s. Then we randomly choose a neighbor node, v_2, of the node v_1, and so on until we reach to the destination node t. Thus we get a path P(s,t)= {s,v_1,v_2,…,t}. To prevent a loop in a path, a node already included in a path is excluded in the random selection. Also, if a path cannot reach to the destination node t within |V| hops, we discard the path.

3.4Fitness function

For a chromosome, we evaluate its quality for a solution i.e., its fitness using a fitness function. In this paper, the aim is to find a path from a source node to a destination node with the lowest path cost which is given by a sum of link costs and the maximum node cost among the nodes on the path. The fitness function of a chromosome C_k, $f(C_k)$ is given by

$$f(C_k) = \left(\alpha \sum L_i + \beta \max \{n_j | j \in P(s,t)\}\right)^{-1} \quad (2)$$

Where L_i is a cost of link i, n_j is a cost of node j, and i,j ∈ P(s,t). Also α and β are proportional coefficients and P(s,t) is a path encoded by the chromosome C_k.

3.5Crossover

Crossover evolves the current chromosomes so as to become better chromosomes. In this paper, we use single point crossover to exchange partial chromosomes. *Figure 2* shows an example of the single point crossover.

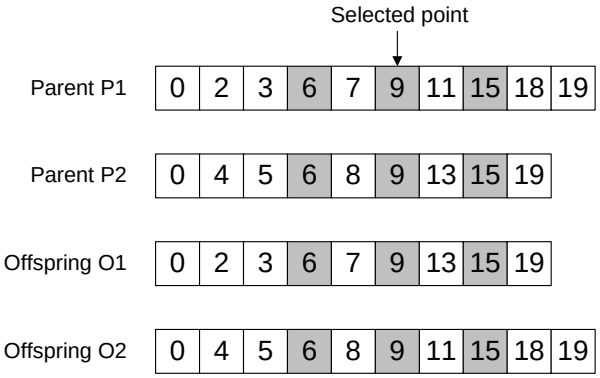

Figure 2 Example of single point crossover

Parent chromosomes are randomly selected to mate. The selection probability of a chromosome is proportional to the fitness of the chromosome. The crossover is performed with a crossover probability ρ. The common nodes of both parents are checked and one common node is randomly selected. The common nodes are where the paths of parents are intersected. In *Figure 2*, the common nodes are 6, 9, and 15 and the node 9 is selected. We obtain two child chromosomes by exchanging the substrings of parent chromosomes. The substrings beyond the selected common node are swapped between the parents. In *Figure 2*, the substrings 13-15-19 and 11-

15-18-19 are swapped to produce child chromosomes.

If parent chromosomes do not have a common node, crossover does not occur and the child's chromosomes are the exact copy of the parents. The child's chromosomes may be infeasible because of the node duplication or the path disconnection. We first remove the duplicated node from the child's chromosomes and then check the path connectivity of the child. If two adjacent nodes of a child chromosome are not directly connected via a link, the child chromosome is infeasible. In this paper, we do not repair the infeasible child. If a child is infeasible after crossover, we discard the infeasible child, but use one of the exact copies of the parents as a new child.

3.6Mutation

Mutation helps to escape from local optima. In this paper, we use a single point mutation. Each chromosome can mutate with a mutation probability σ. The mutation point of a chromosome is randomly selected except the start and destination nodes. Let the i-th point of a chromosome C_k be the mutation point. To prevent an infeasible path, a node in the i-th point must be directly connected to a node in the (i-1)-th point. Also the node in the i-th point must be directly connected to a node in the (i+1)-th point. In this paper, we randomly select one of neighbor nodes of a node in the (i-1) point for the new node in the i-th point. Then we test if the new node is directly connected to the node in the (i+1)-th point and check if the new node is duplicated in C_k. The mutation is accepted only if the new node makes a feasible path. Otherwise, the mutation is canceled and the original C_k is used.

3.7Elite chromosome

After a mutation process is over, we evaluate fitness of all chromosomes and then find the best one in the current generation. Then we compare the best of the current generation, with the best of the previous generation. The best of them is the elite chromosome and we keep the elite chromosome in the evolution process of the next generation.

4.Mutation-based immigrant GA

The SGA generates an initial population without checking duplication of chromosomes. It generates new populations using crossover and mutation. By repeating these processes, it selects relatively fitter individuals and stops when a certain stop condition is met. The elite chromosome is not used in the SGA. The RIGA was introduced to improve convergence speed under changing environments. To adapt to a new environment, the diversity of chromosomes is required. In RIGA, randomly generated new chromosomes replace old chromosomes for the diversity. The randomly generated chromosomes are called random immigrants. Two strategies were introduced to replace old chromosomes: replacing random chromosomes or replacing the bad ones. To prevent that random immigrants disrupt the ongoing evolutionary process too much, the number of the random immigrants is set to less than 20% of the total chromosome population.

The EIGA is motivated by the RIGA. The disadvantage of the RIGA is that the evolution processes can be disrupted by the random immigrants especially when the environment changes slowly. The random immigrants may not have any actual effect on the slowly changing environments [9]. To overcome the disadvantage, the EIGA uses an elite chromosome to generate immigrants. To obtain the immigrants, the elite chromosome is mutated. Then the immigrants replace the worst chromosomes in the current generation. The total number of immigrants is limited to a certain level to prevent disruption of the ongoing evolutionary process. It was shown that the EIGA outperforms the SGA and the RIGA when an environment changes slowly and slightly [9]. Moreover, the EIGA converges faster than the SGA and the RIGA. In our problem, the network topology does not change. Thus the EIGA is more suitable for our problem compared to the RIGA.

The disadvantage of the EIGA is that the elitism-based immigrants may not helpful to escape from the local minima since the immigrants were generated from the same elite. To avoid the local minima, we need a certain level of the diversity in the chromosomes. Based on this consideration, we propose a new immigrant approach, called a mutation-based immigrant GA (MIGA). The immigrants are generated by mutation from the current multiple chromosomes not from the elite chromosome. In a mutation stage, a chromosome can undergo an immigrant mutation process with a probability η and a normal mutation process with a probability $(1 - \eta)$. If the probability η is large, the evolution process can be disrupted. Like other methods, the probability η must be less than 0.2.

The immigrant mutation process is the same as the normal mutation process except a chromosome mutates with an immigrant mutation probability μ

which is larger than the normal mutation probability σ. We use a single point mutation for the immigrant mutation. At the end of the immigrant mutation process, we check if the newly mutated node is duplicated and if the newly mutated node makes a valid path. If the newly mutated node yields a feasible path, the mutation is accepted. Otherwise, the mutation is canceled and the original chromosome is used as an immigrant. *Figure 3* shows the pseudo code of the proposed MIGA. The termination condition in *Figure 3* is the predetermined number of generations. Since not only the optimality of the solution, but also the fast path setup is important, the number of generations is used.

```
begin
initial population
evaluate fitness of initial population
repeat
        crossover operation with probability ρ
        for each chromosome C_k
                if random value < η then
                        immigrant mutation operation with probability μ
                else
                        mutation operation with probability σ
                end if
                elite chromosome calculation
until the termination condition is met
end
```

Figure 3 Pseudo code of the proposed algorithm

The difference between the proposed MIGA and the EIGA is twofold. The first one is the diversity of the immigrants. Since the MIGA generates the immigrants from multiple chromosomes, the diversity of the MIGA is better than that of the EIGA. The second one is the simplicity. The EIGA has to sort the chromosomes using their fitness functions to replace the worst chromosomes with the immigrants. As the population size of the chromosomes increases, the computation time for the sorting process increases. The proposed MIGA does not use the sorting process. The MIGA is more suitable than the EIGA when the fast path setup is required in network applications.

5.Simulation results

We use the network topology in [6] which has 20 nodes and 62 links. *Figure 4* illustrates the network topology. We use the link costs of [6]. We set the cost of the unconnected link as 10000 as in [6]. We set the node costs as follows: $n_7 = n_{14} = 80$, $n_3 = n_6 = n_9 = n_{12} = n_{15} = 50$, and other $n_j = 30$. The start node is the node 0 and the destination node is the node 19.

We compare the performance of MIGA with those of SGA and EIGA. The crossover operations of the SGA and the EIGA are the same as those of the MIGA. The normal mutation operation of the MIGA is used in the mutation operations of the SGA and the EIGA. For the generation of the elitism-based immigrants in the EIGA, we use the single point mutation with probabilityω.

We set the crossover probability ρ = 0.99 for all GAs. The mutation probability σ = 0.05 for the SGA, the EIGA and the normal mutation process of the MIGA. For the MIGA, the immigrant ratio probability η = 0.2. For the EIGA, the ratio of the number of immigrants to the number of chromosomes is 0.2. Also, for the EIGA, we set α = 1.0 and β = 1.0. For each GA, the termination is that the number of generations reaches to 100. At each generation, for each GA, we select the best chromosome from the current population and output the path cost represented by the best one. For each GA, 1000 independent runs were executed and we obtain the average values of the best solutions at each generation.

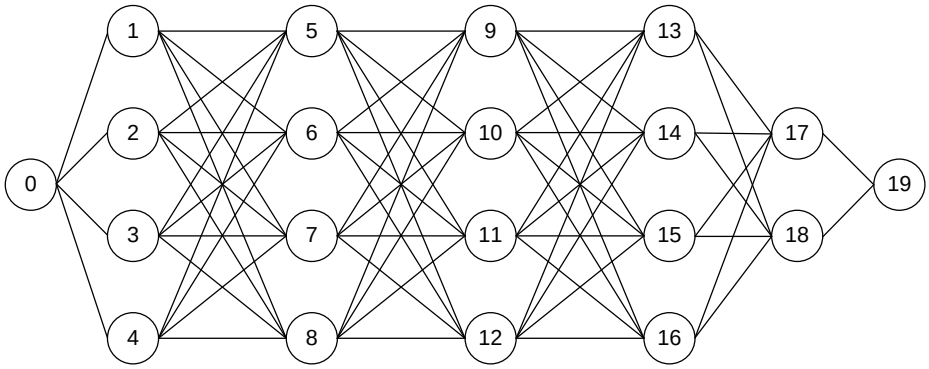

Figure 4 Simulation network topology

We compare the proposed MIGA with the SGA and the EIGA. We set the immigrant mutation probability $\mu = 0.9$ for the MIGA. For the EIGA, we set the immigrant mutation probability $\omega = 0.5$ for comparison purpose. Since the immigrant ratio probability $\eta = 0.2$, the average ratio of the number of immigrants to the population size is 20% for the MIGA. For the EIGA, we set the ratio of the number of immigrants to the population size is 20% for comparison purpose. To check the effect of the immigrants, we compare the three GAs as we set the population size to 100 and 200 separately. *Figures 5* and 6 show average values of the best solutions at each generation for each GA when the population size is 100 and 200, respectively.

As we can see from *Figures 5 and 6*, the proposed MIGA outperforms the other GAs in the quality of the solution in all cases. The immigrant scheme is more effective when the population size is small in the proposed MIGA. The reason is that the immigrants provide a genetic diversity that helps to find the best solution. But the immigrants are less helpful if the population size is large since the large population is genetically diverse. The EIGA has the fastest convergence speed to a solution thank to the elitism-based immigrants. The immigrants based on the elite replace the original chromosomes. Therefore, the chromosome pool is filled with chromosomes, which are similar to the elite as the generations increase. This helps the fast converge to a solution. But the problem is that the suboptimal chromosome can be the elite. The chromosomes generated from the suboptimal elite disrupt converging to the optimal solution. Hence the quality of the solution of the EIGA is worse than that of the MIGA.

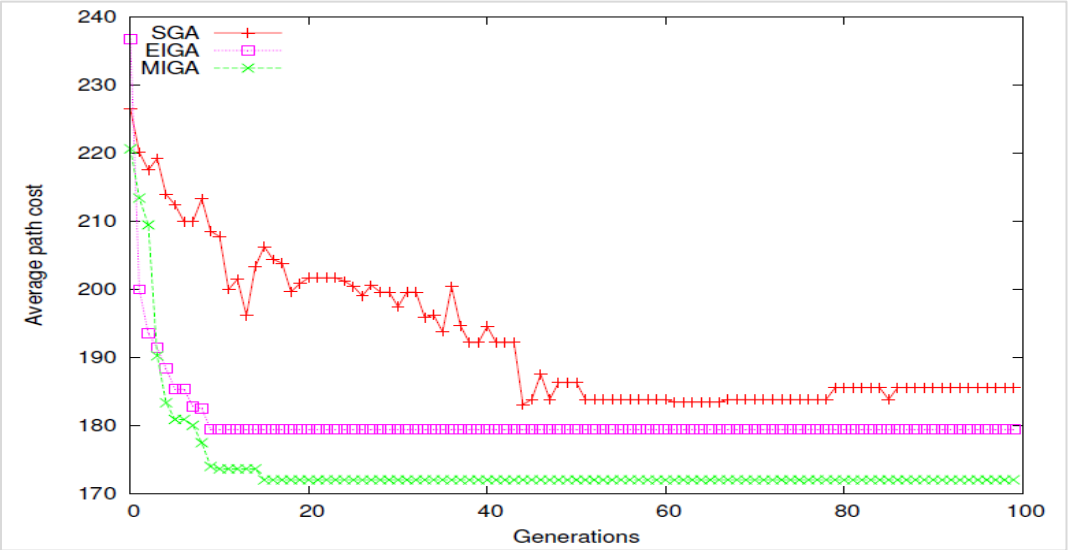

Figure 5 Population size = 100

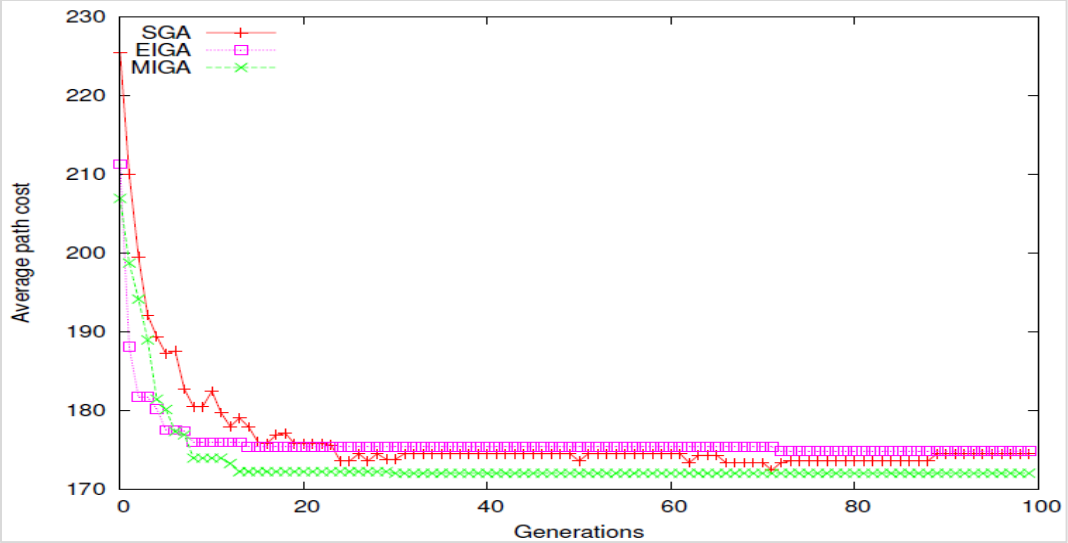

Figure 6 Population size = 200

Note that the quality of the solution of the SGA is increasing as the population size increases in *Figures 5 and 6*. The population size has a proportional relationship with the genetic diversity. The impacts of the immigrants of the MIGA and the EIGA are decreased as the population size increases. However, the increased population size means the algorithm run time is increased. Since the fast path setup is also important in network applications, the population size has to be small. As we can see from *Figures 5 and 6*, the MIGA converges to a solution within 20 generations. Therefore, the proposed MIGA has the best quality of a solution and a good convergence time among the considered GAs.

6.Conclusion

We addressed that the current path computation algorithms are limited to a linear metric like link costs in the VNE problem. Then we proposed a fast GA of optimal path computation that supports a non-linear metric consisting of link and node costs. The proposed method generates the immigrants from ordinary chromosomes not from the elite chromosome for the genetic diversity. Also, the proposed method does not use the sorting process to replace the chromosomes. The proposed method uses the random replacing mechanism to decrease the path computation time. Using simulations, we showed that the proposed method is better than existing algorithms.

Acknowledgment

None.

Conflicts of interest

The author has no conflicts of interest to declare.

References

[1] Fischer A, Botero JF, Beck MT, De Meer H, Hesselbach X. Virtual network embedding: a survey. IEEE Communications Surveys & Tutorials. 2013; 15(4):1888-906.

[2] Paolucci F, Cugini F, Giorgetti A, Sambo N, Castoldi P. A survey on the path computation element (PCE) architecture. IEEE Communications Surveys & Tutorials. 2013; 15(4):1819-41.

[3] Shahin AA. Memetic elitist pareto evolutionary algorithm for virtual network embedding. Computer and Information Science. 2015; 8(2):73-88.

[4] Zhang Z, Cheng X, Su S, Wang Y, Shuang K, Luo Y. A unified enhanced particle swarm optimization-based virtual network embedding algorithm. International Journal of Communication Systems. 2013; 26(8):1054-73.

[5] Fajjari I, Saadi NA, Pujolle G, Zimmermann H. VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic. In international conference on communications 2011 (pp. 1-6). IEEE.

[6] Gonen B. Genetic algorithm finding the shortest path in networks. Reno: University of Nevada. 2006.

[7] Ahn CW, 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.

[8] Han MS. Optimal routing path calculation for SDN using genetic algorithm. International Journal of Hybrid Information Technology. 2018; 11(3): 7-12.

[9] Yang S. Genetic algorithms with elitism-based immigrants for changing optimization problems. In workshops on applications of evolutionary computation 2007 (pp. 627-36). Springer, Berlin, Heidelberg.

[10] Gladwin D, Stewart P, Stewart J. A controlled migration genetic algorithm operator for hardware-in-the-loop experimentation. Engineering Applications of Artificial Intelligence. 2011; 24(4):586-94.
[11] Cheng H, Yang S. Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile ad hoc networks. In European conference on the applications of evolutionary computation 2010 (pp. 562-71). Springer, Berlin, Heidelberg.
[12] Malis A, Wilson B, Clapp G, Shukla V. Requirements for very fast setup of GMPLS LSPs, RFC-7709; 2015:1-9.
[13] Farrel A, Ayyangar A, Vasseur J. Inter-domain MPLS and GMPLS traffic engineering, RFC-5151; 2008.
[14] Awduche D, Malcolm J, Agogbua J, ODell M, McManus J. Requirements for traffic engineering over MPLS, RFC-2702. 1999:1-29.
[15] Awduche D, Berger L, Gan D, Li T, Srinivasan V, Swallow G. RSVP-TE: extensions to RSVP for LSP tunnels. 2001:1-61.

Man Soo Han received his BS, MS, and PhD in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 1992, 1994, and 1999, respectively. He is a professor in the Department of Information and Communications Engineering at Mokpo National University, Jeonnam, Republic of Korea.
Email: mshan@mokpo.ac.kr