International Journal of Hybrid Information Technology
Vol.11, No.3 (2018), pp.7-12
http://dx.doi.org/10.21742/ijhit.2018.11.3.02

Optimal Routing Path Calculation for SDN Using Genetic Algorithm

Man Soo Han

Dept. Information and Communications, Mokpo National Univ., Republic of Korea
mshan@mokpo.ac.kr

Abstract

Path Computation Element (PCE) is a network entity that calculates the best routing path between a source node and a destination node in networks. It is a key technology to support virtualization of Software Defined Network (SDN) and Network Function Virtualization (NFV) for which packet processing power of virtual network nodes as well as the bandwidth between them should be considered as a major parameter for the routing path calculation. However, current analytic algorithms including Dijkstra widely used can't apply to some critical cases due to the non-linear characteristic of parameters in which both link and performance costs are considered as parameters. This paper addresses that the current shortest path first algorithms are limited to a linear metric, and proposes a new Genetic Algorithm (GA) to support a non-linear metric for both link and performance costs.

Keywords: *PCE, Genetic algorithm, Shortest path, SDN*

1. Introduction

PCE is a network entity that calculates the best routing path between a source node and a destination node [1][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 where the availability of routing paths is determined by a simple sum-up of link costs. Dijkstra's algorithm has widely been used for calculating the best routing path in networks modeled in a linear metric, which makes it possible to calculate the best path in a simple manner and within a reasonable time. However, we find out that the Dijkstra algorithm can't apply to some cases that have performance costs in complex forms, especially when both link and performance costs are combined in a non-linear form. It is a heuristic or meta-heuristic method that can provide a solution to overcome the cases.

Some papers were reviewed to study meta-heuristic algorithms for Virtual Network Embedding (VNE). In [2], a genetic algorithm based on a non-dominated sorting-based multi-objective evolutionary algorithm is introduced for the VNE problem. But the link path between nodes is calculated by the SPF (shortest path first) method which is based on Dijkstra's algorithm. In [4], a particle swarm optimization method is used for the VNE problem. However, the SPF is used for the calculation of the link path between nodes. In [5], an ant colony algorithm is introduced for the VNE problem. But, the link path between nodes is calculated by the SPF. We find out that the above methods can't apply to networks with non-linear costs, as all the methods above use the SPF algorithm for path computation. It is needed to find a new algorithm for path computation to support VNE with non-linear costs.

Article history:
Received (February 26, 2018), Review Result (March 30, 2018), Accepted (May 5, 2018)

Print ISSN: 1738-9968 eISSN: 2652-2233 IJHIT

We studied GA (genetic algorithm) 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 [9] where only link cost is considered. 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 [8] that immigrant individuals are generated from an elite individual and then added to the genetic population. It was shown that EIGA outperforms SGA and RIGA in dynamic environments.

In [7], immigrant individuals are generated from a heuristic method and then inserted into the genetic population. In [10], 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 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[7] and[10] are not desired because of their complexity. In this paper, we propose a noble fast path computation algorithm with a non-linear metric by improving the disadvantages of EIGA.

2. Design of GA for shortest path

2.1. Model

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. Recent applications such as data centers, cloud computing, video, gaming, and mobile can increase connection request rates. For these applications, a rapid path setup has to be provided. Thus the computation speed for the given problem has to be fast. We aim 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 Eq. (1).

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

where L_i is a cost of link i, n_j is a cost of node j, and i,j $\in$P. Also α and β are proportional coefficients.

2.2. Genetic 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.

2.3. Initial 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, chromosomes of the initial population are randomly generated for diversity. Also let s and t be the source node and the destination node, respectively. We build a path from node s to node t by randomly selecting a neighbor node. A node u is a neighbor node of a node v if node u is directly connected to 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 the destination node t. Thus we get a path $P(s,t) = \{s,v_1,v_2,\ldots,t\}$. 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 the destination node t within |V| hops, we discard the path.

2.4. Fitness 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

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

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

2.5. Crossover

Crossover evolves the current chromosomes to become better chromosomes. In this paper, we use single point crossover to exchange partial chromosomes. 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 node is randomly selected. The common nodes are where the paths of parents are intersected. We obtain two child chromosomes by exchanging the substrings of parent chromosomes. The substrings beyond the selected common node are swapped between the parents.

2.6. Mutation

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 the 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.

2.7. Pseudo code

[Figure 1] shows the pseudo code of the proposed algorithm. The termination condition in Figure 1 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 the fitness of the initial population
  repeat
      crossover operation with probability ρ
      for each chromosome Ck
            mutation operation with probability σ
  until the termination condition is met
end
```

Figure 1. Pseudo code of the proposed algorithm

3. Simulation results

We use the network topology in [6] which has 20 nodes and 62 links. [Figure 2] 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 node 0 and the destination node is node 19. We set the crossover probability $\rho = 0.99$ and the mutation probability $\sigma = 0.05$. At each generation, we select the best chromosome from the current population and output the path cost represented by the best one. We execute 1000 independent runs then we obtain the average values of the best solutions at each generation. To check the effect of the population size, we consider three cases that the population size is 50, 100, and 200, respectively.

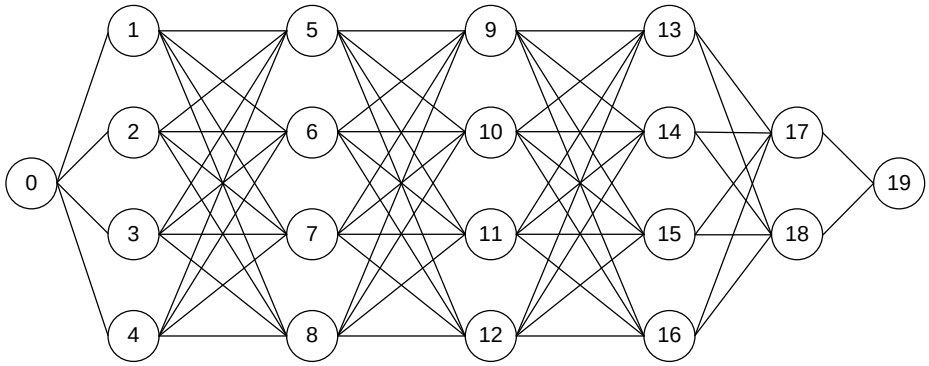

Figure 2. Simulation network topology

[Figure 3] shows the average path cost for the three cases. Note that the quality of the solution is increased as the population size increases in [Figure 3]. The population size has a proportional relationship with genetic diversity. As we can see from [Figure 3], the proposed algorithm converges to a solution within 20 generations if the population size is 200.

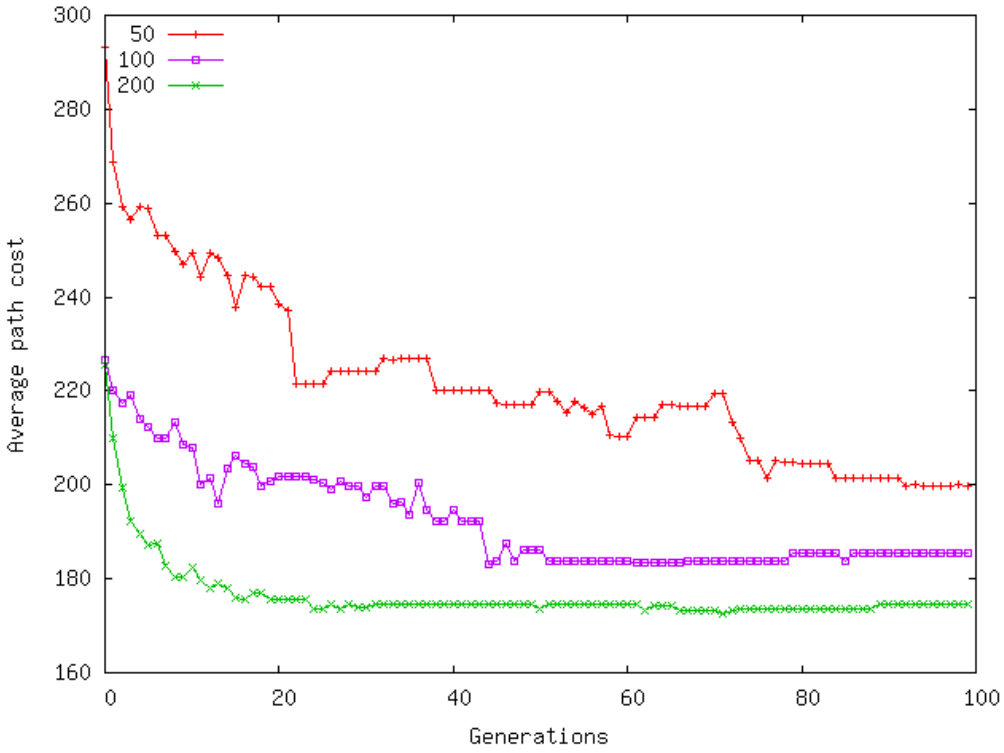

Figure 3. Simulation results

4. Conclusions

This paper addressed the problem of the current shortest path first of computation algorithms limited to a linear metric like link costs and proposes a new GA of optimal path computation that supports a non-linear metric for both link and performance costs. Using simulation, we showed the performance of the proposed method for the optimal path computation problem having the non-linear metric.

References

[1] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach, "Virtual Network Embedding: A Survey," IEEE Communications Surveys & Tutorials, vol.15, no.4, FOURTH QUARTER, **(2013)**

[2] Francesco Paolucci, Filippo Cugini, Alessio Giorgetti, Nicola Sambo, and Piero Castoldi, "A survey on the path computation element (PCE) architecture," IEEE Communications Surveys & Tutorials, vol.15, no.4, FOURTH QUARTER, **(2013)**

[3] Ashraf A. Shahin, "Memetic elitist Pareto evolutionary algorithm for virtual network embedding," Computer and Information Science, vol.8, no.5, pp.73-88, **(2015)**

[4] Zhang Z., Cheng X., Su S., Wang Y., Shuang K., and Luo Y., "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," Int. J. Communication Systems, vol.26, no.8, pp.1054–1073, **(2013)**

[5] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic," 2011 IEEE International Conference on Communications (ICC), pp.1-6, **(2011)**

[6] Bilal Gonen, Genetic Algorithm Finding the Shortest Path in Networks, **(2008)**

[7] D. Gladwin, P. Stewartb, and J. Stewart, "A controlled migration genetic algorithm operator for hardware-in-the-loop experimentation," Engineering Applications of Artificial Intelligence, vol.24, no.4, **(2011)**

[8] Shengxiang Yang, "Genetic Algorithms with Elitism-Based Immigrants for Changing Optimization Problems," Workshops on Applications of Evolutionary Computation, pp.627-636, **(2007)**

[9] Chang Wook Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," IEEE Transactions on Evolutionary Computation, vol.6, no.6, **(2002)**

[10] Hui Cheng and Shengxiang Yang, "Multi-population Genetic Algorithms with Immigrants Scheme for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks," European Conference on the Applications of Evolutionary Computation, pp.562-571, **(2010)**