Genetic Algorithm Finding the Shortest Path in Networks

Bilal Gonen[1], Sushil J. Louis[1]

[1]Department of Computer Science and Engineering, University of Nevada, Reno, Reno, Nevada, U.S.A.

Abstract— *With the growth of the Internet, Internet Service Providers (ISPs) try to meet the increasing traffic demand with new technology and improved utilization of existing resources. Routing of data packets can affect network utilization. Packets are sent along network paths from source to destination following a protocol. Open Shortest Path First (OSPF) is the most commonly used intra-domain Internet routing protocol (IRP). Traffic flow is routed along shortest paths, splitting flow at nodes with several outgoing links on a shortest path to the destination IP address. Link weights are assigned by the network operator. A path length is the sum of the weights of the links in the path. In this paper, we study the problem of optimizing OSPF weights, given a set of projected demands, with the objective of minimizing network congestion. The weight assignment problem is NP-hard. We developed a genetic algorithm (GA) to solve this problem.*

Keywords: genetic algorithm, shortest path, computer networks

1. Introduction

Routing is a fundamental engineering task on the Internet. It consists in finding a path from a source to a destination host. Routing is complex in large networks because of the many potential intermediate destinations a packet might traverse before reaching its destination [2]. The link weights are assigned by the network operator. The lower the weight, the greater the chance that traffic will get routed on that link [3]. When one sends or receives data over the Internet, the information is divided into small chunks called packets or datagrams. A header, containing the necessary transmission information, such as the destination Internet Protocol (IP) address, is attached to each packet. The data packets are sent along links between routers on Internet. When a data packet reaches a router, the incoming datagrams are stored in a queue to await processing. The router reads the datagram header, takes the IP destination address and determines the best way to forward this packet for it to reach its final destination [3].

The configuration of network protocols is widely considered a black art and is normally performed based on network administrators' experience, trial and error, etc... These manual methods are often error-prone and not scalable to large complex networks. The emphasis of the search algorithm should be on finding a better operating point within the limited time frame instead of seeking the strictly global optimum. Network conditions vary with time and the search algorithm should quickly find better network parameters before significant changes in the network occur. Another feature of these problems; for example, AT&T's network has 1000s of routers and links. If all OSPF link weights of this network are to be configured, there will be thousands of parameters present in the optimization [4].

2. The Shortest Path Problem

The shortest path problem is defined as that of finding a minimum-length (cost) path between a given pair of nodes [5]. Shortest path problem is a classical research topic. It was proposed by Dijkstra in 1959 and has been widely researched. The Dijkstra algorithm is considered as the most efficient method. It is based on the Bellman optimization theory. But when the network is very big, then it becomes inefficient since a lot of computations need to be repeated. Also it can not be implemented in the permitted time [9].

3. Genetic Algorithm

As a special kind of stochastic search algorithms, genetic algorithm is a problem solving method which is based on the concept of natural selection and genetics [6].

In the 1970s, Holland first introduced genetic algorithms to explain the adaptive processes of natural systems and to design an artificial system, which retains the robust mechanism of natural systems [5].

The steps of a GA are shown in Algorithm 1.

Algorithm 1 *GA Steps*[7]

1: Choose initial population
2: Evaluate the fitness of each individual in the population
3: **while** <terminating condition not satisfied> **do**
4:    Select best-ranking individuals to reproduce
5:    Breed new generation through crossover and mutation
6:    Evaluate the individual fitnesses of the offspring
7:    Replace worst ranked individuals with offsprings
8: **end while**

---

4. The GA Algorithm That We Implemented

The steps of my GA algorithm are explained in this section.

4.1 Choose Initial Population

When initializing the population, my algorithm starts from the SOURCE. SOURCE is a constant in the program, so the user may want to pick another node as the starting point. The algorithm selects one of the neighbors provided that it has not been picked before. It keeps doing this operation until it reaches to DESTINATION. Like SOURCE, DESTINATION is also a constant that user may change as they wish.

4.2 Evaluate the fitness of each individual in the population

The evaluation function takes a path in the population. It gets the distance between each node pair in the path, by calling a function to read from the distance array. Adds them together and returns the sum as the cost of the path.

4.3 Select best-ranking individuals to reproduce

My algorithm selects two individuals from the population with the lowest costs.

4.4 Crossover and Mutation

With some probability, the program mates the two individuals. The crossover function takes two parents to mate. It looks for the common points in the parents. The common nodes are where these two paths intersect. Among the common points, the program selects one of them randomly. It makes the crossover from that point. The crossover operation is illustrated in Figure 1.

4.5 Evaluate the individual fitnesses of the offspring

We send these offspring to the evaluation function to get their fitnesses. If the offsprings' fitnesses are less than the nodes with maximum fitnesses in the population, we replace them with the nodes with the maximum fitnesses.

4.6 Terminating Condition

My terminating condition is a predefined number of iterations. Because, in the network topology, the goal is not to find the global optimum, but to find a path with a reasonable cost in a limited time.

5. Experiment Results

We generated a network topology with 20 nodes and 62 links to test my Genetic Algorithm. Each link has a cost associated with them. We set two nodes as source and destination. The goal of my GA application is to find a path between source and destination with the lowest cost.

In Figure 3, the cells with 10,000 in them represent that there is no direct link between those nodes. Because, 10,000 is too big compared to other small costs, therefore my implementation ignore those big numbers, and pick the links
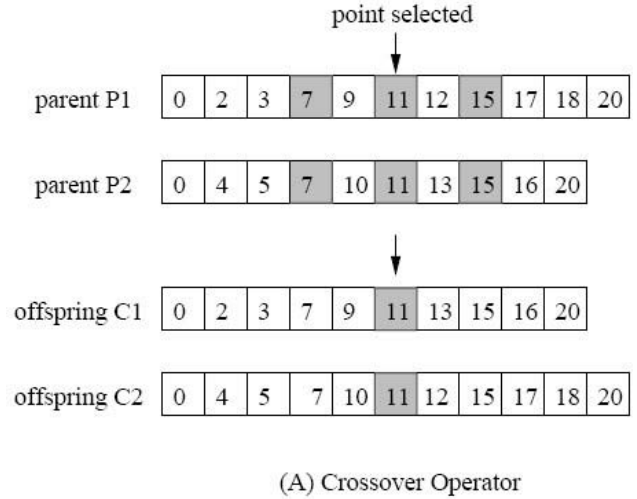

(A) Crossover Operator
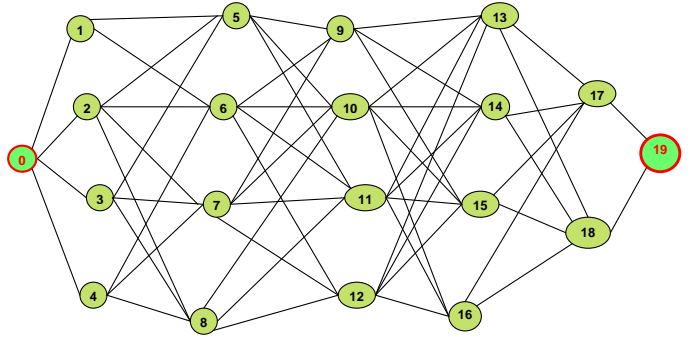
Fig. 1: Crossover Operator [7]

Fig. 2: Network topology used

with small costs, instead. Figure 4 shows a sample of initial population, and their fitnesses.

We set several parameters for the experiment. They are as follows; Population size=50, Number of runs=30, Number of generations=50, Crossover probability = 0.99, Mutation probability=0.1. We run the steps selection, crossover, and replace part 50 times (number of generations). Figure 5 and Figure 6 shows the average of maximum numbers of 30 runs, the average of minimum numbers of 30 runs, and the average of average numbers of 30 runs.

6. Analysis of Results

The results show that GA gets close to optimum very quickly. This is a promising result for my research. When using this GA algorithm besides other search algorithms in the USF [8], such as, multi-start hill-climbing, simulated annealing, Controlled Random Search and RRS (Recursive Random Search), we can start searching the space with GA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	10000	52	61	8	16	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
1	52	10000	10000	10000	10000	78	41	6	92	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
2	61	10000	10000	10000	10000	84	63	2	99	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
3	8	10000	10000	10000	10000	71	48	223	73	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
4	16	10000	10000	10000	10000	63	55	44	88	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
5	10000	78	84	71	63	10000	10000	10000	10000	11	22	33	44	10000	10000	10000	10000	10000	10000	10000
6	10000	41	63	48	55	10000	10000	10000	10000	21	32	43	54	10000	10000	10000	10000	10000	10000	10000
7	10000	6	2	223	44	10000	10000	10000	10000	74	85	96	14	10000	10000	10000	10000	10000	10000	10000
8	10000	92	99	73	88	10000	10000	10000	10000	46	64	75	35	10000	10000	10000	10000	10000	10000	10000
9	10000	10000	10000	10000	10000	11	21	74	46	10000	10000	10000	10000	66	55	44	11	10000	10000	10000
10	10000	10000	10000	10000	10000	22	32	85	64	10000	10000	10000	10000	91	97	73	19	10000	10000	10000
11	10000	10000	10000	10000	10000	33	43	96	75	10000	10000	10000	10000	45	65	25	85	10000	10000	10000
12	10000	10000	10000	10000	10000	44	54	14	35	10000	10000	10000	10000	73	37	87	16	10000	10000	10000
13	10000	10000	10000	10000	10000	10000	10000	10000	10000	66	91	45	73	10000	10000	10000	10000	86	84	10000
14	10000	10000	10000	10000	10000	10000	10000	10000	10000	55	97	65	37	10000	10000	10000	10000	74	76	10000
15	10000	10000	10000	10000	10000	10000	10000	10000	10000	44	73	25	87	10000	10000	10000	10000	2	6	10000
16	10000	10000	10000	10000	10000	10000	10000	10000	10000	11	19	85	16	10000	10000	10000	10000	7	9	10000
17	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	86	74	2	7	10000	10000	52
18	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	84	76	6	9	10000	10000	25
19	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	52	25	10000

Fig. 3: The costs on the links

```
Individual-0    Fitness: 222   path: 0 - 1 - 6 - 12 - 16 - 17 - 19 -
Individual-1    Fitness: 1025  path: 0 - 4 - 5 - 9 - 15 - 17 - 16 - 12 - 14 - 11 - 8 - 1 - 6 - 3 - 7 - 10 - 13 - 18 - 19
Individual-2    Fitness: 231   path: 0 - 3 - 5 - 11 - 16 - 18 - 19 -
Individual-3    Fitness: 879   path: 0 - 1 - 8 - 12 - 14 - 17 - 13 - 11 - 6 - 2 - 7 - 3 - 5 - 9 - 16 - 18 - 19 -
Individual-4    Fitness: 728   path: 0 - 3 - 7 - 11 - 13 - 10 - 5 - 2 - 6 - 9 - 15 - 18 - 19 -
Individual-5    Fitness: 574   path: 0 - 2 - 7 - 9 - 13 - 10 - 8 - 12 - 14 - 18 - 16 - 17 - 19 -
Individual-6    Fitness: 390   path: 0 - 3 - 5 - 1 - 6 - 12 - 14 - 18 - 19 -
Individual-7    Fitness: 901   path: 0 - 4 - 6 - 1 - 5 - 12 - 16 - 10 - 15 - 17 - 14 - 11 - 7 - 2 - 8 - 9 - 13 - 18 - 19
Individual-8    Fitness: 755   path: 0 - 3 - 6 - 1 - 5 - 9 - 13 - 12 - 7 - 4 - 8 - 11 - 14 - 18 - 16 - 17 - 19 -
Individual-9    Fitness: 466   path: 0 - 2 - 5 - 9 - 8 - 10 - 13 - 18 - 19 -
Individual-10   Fitness: 364   path: 0 - 4 - 8 - 12 - 13 - 18 - 16 - 17 - 19 -
Individual-11   Fitness: 210   path: 0 - 1 - 7 - 12 - 14 - 18 - 19 -
Individual-12   Fitness: 380   path: 0 - 3 - 7 - 9 - 15 - 18 - 19 -
Individual-13   Fitness: 281   path: 0 - 2 - 8 - 9 - 15 - 18 - 19 -
Individual-14   Fitness: 792   path: 0 - 2 - 8 - 11 - 15 - 18 - 14 - 10 - 6 - 1 - 5 - 12 - 7 - 9 - 16 - 17 - 19 -
Individual-15   Fitness: 531   path: 0 - 4 - 7 - 1 - 8 - 11 - 6 - 10 - 14 - 17 - 19 -
Individual-16   Fitness: 954   path: 0 - 2 - 8 - 4 - 5 - 12 - 7 - 11 - 13 - 18 - 16 - 9 - 15 - 10 - 14 - 17 - 19 -
Individual-17   Fitness: 700   path: 0 - 3 - 7 - 12 - 6 - 11 - 13 - 18 - 14 - 9 - 15 - 17 - 19 -
Individual-18   Fitness: 243   path: 0 - 3 - 6 - 11 - 16 - 17 - 19 -
Individual-20   Fitness: 219   path: 0 - 1 - 5 - 11 - 15 - 18 - 19 -
Individual-21   Fitness: 566   path: 0 - 3 - 7 - 10 - 15 - 17 - 14 - 18 - 19 -
Individual-22   Fitness: 476   path: 0 - 1 - 8 - 2 - 5 - 10 - 15 - 17 - 19 -
Individual-23   Fitness: 683   path: 0 - 2 - 7 - 3 - 6 - 11 - 13 - 9 - 15 - 10 - 16 - 17 - 19 -
Individual-24   Fitness: 292   path: 0 - 3 - 6 - 12 - 13 - 18 - 19 -
Individual-25   Fitness: 329   path: 0 - 1 - 6 - 12 - 13 - 18 - 19 -
Individual-26   Fitness: 147   path: 0 - 1 - 7 - 12 - 16 - 17 - 19 -
Individual-27   Fitness: 765   path: 0 - 2 - 7 - 9 - 14 - 11 - 8 - 3 - 5 - 12 - 16 - 10 - 13 - 17 - 15 - 18 - 19 -
Individual-28   Fitness: 188   path: 0 - 4 - 5 - 9 - 15 - 17 - 19 -
Individual-29   Fitness: 712   path: 0 - 4 - 5 - 11 - 13 - 9 - 8 - 12 - 6 - 2 - 7 - 10 - 14 - 17 - 15 - 18 - 19 -
Individual-30   Fitness: 290   path: 0 - 3 - 6 - 11 - 14 - 17 - 19 -
Individual-31   Fitness: 271   path: 0 - 2 - 5 - 10 - 15 - 18 - 19 -
Individual-32   Fitness: 407   path: 0 - 1 - 6 - 9 - 16 - 12 - 5 - 10 - 13 - 18 - 19 -
Individual-33   Fitness: 565   path: 0 - 4 - 7 - 11 - 15 - 9 - 16 - 18 - 13 - 12 - 14 - 17 - 19 -
Individual-34   Fitness: 288   path: 0 - 3 - 5 - 11 - 15 - 10 - 16 - 17 - 19 -
Individual-35   Fitness: 471   path: 0 - 3 - 8 - 9 - 16 - 18 - 14 - 12 - 13 - 17 - 19 -
Individual-36   Fitness: 391   path: 0 - 4 - 8 - 10 - 14 - 17 - 19 -
Individual-37   Fitness: 1081  path: 0 - 4 - 6 - 12 - 5 - 2 - 8 - 1 - 7 - 10 - 15 - 11 - 16 - 9 - 13 - 18 - 14 - 17 - 19
Individual-38   Fitness: 653   path: 0 - 2 - 6 - 10 - 15 - 12 - 14 - 18 - 16 - 9 - 13 - 17 - 19 -
Individual-39   Fitness: 190   path: 0 - 4 - 6 - 9 - 15 - 17 - 19 -
Individual-40   Fitness: 477   path: 0 - 2 - 6 - 9 - 13 - 11 - 5 - 12 - 14 - 17 - 15 - 18 - 19 -
Individual-41   Fitness: 354   path: 0 - 3 - 8 - 11 - 16 - 18 - 19 -
Individual-42   Fitness: 308   path: 0 - 3 - 8 - 9 - 14 - 17 - 19 -
Individual-43   Fitness: 1134  path: 0 - 2 - 6 - 11 - 7 - 3 - 8 - 4 - 5 - 10 - 15 - 18 - 16 - 9 - 14 - 12 - 13 - 17 - 19
Individual-44   Fitness: 349   path: 0 - 1 - 6 - 9 - 15 - 11 - 14 - 18 - 19 -
Individual-46   Fitness: 808   path: 0 - 3 - 5 - 4 - 8 - 1 - 7 - 11 - 15 - 9 - 16 - 17 - 13 - 12 - 14 - 18 - 19 -
Individual-47   Fitness: 373   path: 0 - 1 - 7 - 12 - 6 - 2 - 5 - 10 - 16 - 17 - 19 -
Individual-48   Fitness: 543   path: 0 - 1 - 7 - 12 - 14 - 10 - 5 - 4 - 6 - 11 - 13 - 18 - 19 -
Individual-49   Fitness: 929   path: 0 - 3 - 7 - 2 - 8 - 1 - 5 - 12 - 14 - 11 - 15 - 17 - 16 - 10 - 6 - 9 - 13 - 18 - 19
```

Fig. 4: A sample of initial population, and their fitnesses

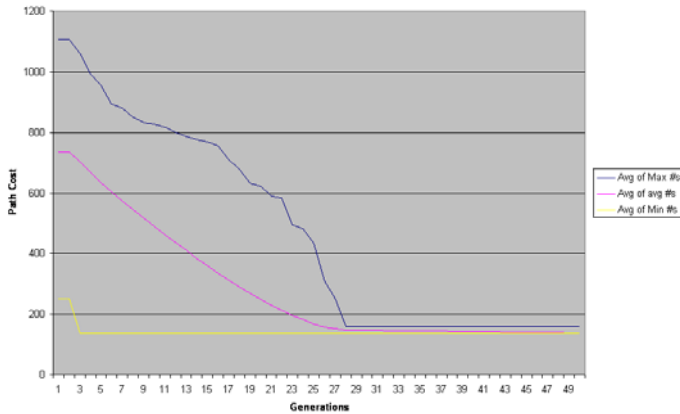| Averages of 30 runs			
gens	Avg of Max #s	Avg of Avg #s	Avg of Min #s
1	1105	736.287	252
2	1105	736.287	252
3	1061	701.767	136
4	992	667.847	136
5	956	634.327	136
6	892	603.207	136
7	881	573.607	136
8	851	544.407	136
9	832	516.427	136
10	826	489.107	136
11	817	462.027	136
12	798	435.587	136
13	787	409.767	136
14	776	384.227	136
15	769	359.167	136
16	756	334.527	136
17	711	311.047	136
18	678	288.627	136
19	631	267.667	136
20	621	248.407	136
21	588	229.507	136
22	583	211.927	136
23	495.6	195.087	136
24	482.8	181.396	136
25	434	168.12	136
26	309	157.22	136
27	252	151.22	136
28	159	147.5	136
29	159	147.04	136
30	159	146.58	136
31	159	146.12	136
32	159	145.66	136
33	159	145.2	136
34	159	144.74	136
35	159	144.28	136
36	159	143.82	136
37	159	143.36	136
38	159	142.9	136
39	159	142.44	136
40	159	141.98	136
41	159	141.52	136
42	159	141.06	136
43	159	140.6	136
44	159	140.14	136
45	159	139.68	136
46	159	139.22	136
47	159	138.76	136
48	159	138.3	136
49	159	137.84	136
50	159	137.38	136

Fig. 5: Average values of runs

first, and then after GA gets close to optimum, then we can switch to other search techniques.

7. Conclusions and Future Research

In this project, we developed a genetic algorithm that finds a shortest path in a limited time. This algorithm is meant to be used in OSPF routing, which is the most commonly used intra-domain Internet routing protocol (IRP). As the future research, we would like to test my GA algorithm on some real network topologies containing much more nodes and links.

References

[1] A Genetic Algorithm for the Weight Setting Problem in OSPF Routing, M. Ericsson, M.G.C. Resende, and P.M. Pardalos
[2] T.M. Thomas II. OSPF Network Design Solutions. Cisco Press, 1998
[3] U. Black. IP Routing Protocols, RIP, OSPF, BGP, PNNI & Cisco routing protocols. Prentice Hall, 2000.
[4] A Recursive Random Search Algorithm for Network Parameter Optimization, Tao Ye , Shivkumar Kalyanaraman
[5] Genetic Algorithms for Solving Disjoint Path Problem with Proportional Path-Costs, Burcu Ozcam, North Carolina State University
[6] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989.
[7] http://en.wikipedia.org/wiki/Genetic_algorithm
[8] A Unified Search Framework for Large-scale Blackbox Optimization, Tao Ye, Shivkumar Kalyanaraman, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute
[9] Faster Genetic Algorithm for Network Paths, Yinzhen Li, Ruichun He, Yaohuang Guo.
[10] Minimizing Packet Loss by Optimizing OSPF Weights Using Online Simulation, Hema Tahilramani Kaur, Tao Ye, Shivkumar Kalyanaraman, Kenneth S. Vastola, Electrical Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY-12180

Fig. 6: Average of 30 runs for 50 generations