A Genetic Algorithm for Optimal Path Routing in Computer Networks

Sowmya KS[#1], N Raksha Rao[#2], Disha P Khanted[#3]

[#1] *Assistant Professor, Department of Information Science Engineering, BMS College of Engineering*
[#2] *Student, Department of Information Science Engineering, BMS College of Engineering*
[#3] *Student, Department of Information Science Engineering, BMS College of Engineering*
Bull Temple Rd, Basavanagudi, Bengaluru, Karnataka 560019, India

Abstract *— Optimal path routing in computer networks refers to the process of finding paths through a network that have a minimum cost. A network mainly consists of nodes and edges, with each edge having a specific weight. The optimal path may be defined in terms of various factors such as throughput, propagation delay, latency and bandwidth, which can be represented as weights of the edges in the network, as a combination of the aforementioned factors. The simulation and algorithm developed achieves convergence at a high rate for large networks. The computer simulation is not dependent on the source and destination nodes. This paper provides a genetic algorithm (GA) approach to arrive at the most optimal path in a large network of computer network nodes.*

Keywords — *genetic algorithms, optimization, meta-heuristic, optimization, chromosome, gene, mutation, crossover, routing problem, multi-hop, networks*

I. INTRODUCTION

Routing is the pivot around which IP connectivity revolves. At the simplest level, routing establishes basic internetwork communications, implements addressing frameworks that identifies each device distinctively, and organizes individual devices into a hierarchical structure. Routing in multi-hop networks is critical. In order to achieve routing, certain algorithms such as breadth first search, depth first search, Dijkstra's, Bellman Ford may be implemented. All these algorithms can be scaled up to a certain level since they run in polynomial time. Modern wireless networks communicate with each other without any fixed structure. The changes in the network are dynamic. Therefore, the routing needs to also be done dynamically in accordance with these changes. The routing must also adapt to these changes quickly, namely in a few milliseconds. The most optimal path in a network may depend on a variety of factors such as bandwidth, propagation delay, and congestion and so on. The weight may be represented as a combination of these factors. The values change over time and the routing must therefore change dynamically. In order to accommodate these changes, meta-heuristic approaches such as Genetic Algorithms may be executed. Genetic Algorithms can be proved to be efficient for such networks, but they are independent to network size. They iterate a large number of times to converge at an optimal solution, and therefore are convenient for large-sized networks.

A. Inspection of Genetic Algorithms

Genetic Algorithms are a meta-heuristic approach to optimization problems and a sub domain of evolutionary computation. It is based on the idea- "survival of the fittest". In nature, the most suitable individuals are likely to survive and mate, thereby leading the next generation to be healthier and fitter.These evolutionary algorithms have three crucial characteristics:

a) They are population-oriented: Evolutionary algorithms optimize a process in which current solutions generate new and better solutions. The set of current solutions from which the new solutions are generated is called the population. The new solutions must ideally be better than the current solution set.

b) They are fitness-oriented: If there are several solutions, we must be able to choose the most optimal solutions from the solution space There is a fitness value associated with each individual solution which is calculated from a fitness function. This fitness value reflects how good the solution is.

c) They are adaptable: If there is no acceptable solution in the current population according to the fitness function calculated for each individual, the algorithm should be able to generate better solutions in that generation. As a result, chosen individual solutions (chosen randomly or probabilistically) will undergo a number of variations to generate new solutions which are fitter.

There are 5 stages in a traditional Genetic Algorithm, namely-

1) Initialization of initial population
2) Fitness function
3) Selection
4) Crossover
5) Mutation

1) Initial Population

The first step is the initialization of population. The population consists of a set of individuals. Each individual is a part of the solution space. Each individual is called a chromosome. These chromosomes are characterized by a set of parameters uniquely. Each parameter is a gene which may be represented as a string in terms of alphabets, digits or binary values. It can be said that we encode the genes in a chromosome.

2) The Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness value.

3) Selection

The selection phase is to select the fittest individuals in order to allow them to pass their genes to the next generation. Pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have a higher chance of being selected for mating. At the end of this step, we will be selecting a subset of the population. These individuals comprise of the mating pool. Random selection or sequential selection of parental chromosomes can be carried out.

4) Crossover

This is the most significant step in the algorithm.
A crossover point is chosen in the chromosome for exchange of genetic materials. The point maybe exactly in the middle to exchange equal parts, or it may be at any random point depending on the type that is opted. There also may be a single point or multi-points for crossover. The crossover point(s) is/are generally chosen at random. Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population.

5) Mutation

When any of the genes are subjected to a random change in their values within the domain of the possible values, with a low probability, it is called mutation. Primarily, one or more genes are flipped to some other value. Without mutation, the offspring will have all of the properties of its parents. To add new features to the offspring, mutation needs to take place. But because mutation occurs randomly, one cannot say with certainty that the offspring may be fitter than its parents.

Termination of the algorithm may occur when the number of iterations is over, or once the most optimal solution is obtained.

II. GENETIC ALGORITHM PROPOSED

A. Representation

A chromosome represents a possible path from the source node to the destination node. Each gene in the chromosome holds an integer representing the label of a node in the given network. The first gene and the last gene in the chromosome is fixed as the source and destination, which is given as the input. The intermediate genes are filled with labels of the remaining nodes in the network. The length of the chromosome is the total number of nodes in the network,
since we will need a maximum of all nodes once to form a path from the source to destination.

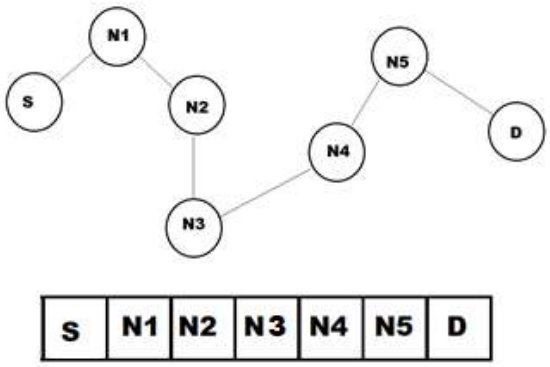

Fig. 1: Structure of chromosome

Above is an example of a network with S as the source node, D as the destination node. Nodes N1, N2, N3, N4 and N5 are the intermediate nodes in the network. The chromosome has N number of genes for an N-node network, the first gene is the source node and the last gene is the destination node.

B. Initialization of population

To form the first set of chromosomes, we fix the source and destination nodes as the first and last gene respectively. The intermediate genes in the chromosome is filled by using a randomizer function. We could also choose heuristic initialization, but it may not be able to explore the entire solution space and may get stuck at a local optimum. To add diversity to our solution, we use a random initialization method. The population size is a crucial part of the algorithm. The larger the population size, the faster we can reach the global optimum solution. But a larger population means an increase in space and time complexity of the solution. For an efficient solution, we must use this trade-off and choose an

optimum population size which accommodates the mating pool.

C. Fitness Function Calculation

The fitness of a chromosome in a network is calculated based on the following metrics

1. Cost
2. Hop Count
3. Bandwidth

Based on the given factors we define the fitness function as

$$f_i = \frac{1}{\sum_{i=1}^{l-1} C_{g(i)\rightarrow g(i+1)}}$$

Fig.2: Fitness function

where f(i) represents the fitness value of the given chromosome, l is the length of the chromosome g(i) represents the ith gene in the chromosome and C represents the cost from g(i) to g(i+1). The fitness of a chromosome is inversely proportional to the combined cost of moving from the source gene to the destination gene. A chromosome with a higher value of fitness is considered to be better than other chromosomes in the population. Using these fitness values, we select chromosomes.

D. Selection

The selection process is necessary to send the chromosomes to the next generation. We must make sure that only higher fitness value chromosomes are passed on, in order to improve the quality of the subsequent generations and thereby converge to an optimal solution. The selection process thus takes the promising region from the solution space for further exploration. We must make sure that if we explore the current region, we may reach the optimal solution quickly but with a trade-off of it being a suboptimal solution on account of lack of diversity in the sample space.

There are various methods of selection used in practice. Here we use a combination of proportionate and ordinal based schemes. In the proportionate based selection scheme, chromosomes are picked out based on their fitness values relative to the other chromosomes in the population. By this method, we continue to explore promising regions of the subspace. To avoid being stuck at a local optimum, we add more selection schemes such as roulette wheel selection. Then we rank the chromosomes of the new population according to their fitness values. This is ordinal based scheme. After ranking, we can pick the top chromosomes to be passed into the next generation.

E. Crossover

Crossover is required to stochastically generate new solutions from the existing population. Two chromosomes are selected from the current population to perform a crossover. These two parents can be chosen by their relative fitness ranking or by using a randomizer.

There are various methods of crossover that can be used.

In a single point crossover, one random point in the chromosome is chosen and the genes to the right of that point are swapped resulting in two off springs, each carrying some information from both parents. In a multi-point crossover, two crossover points are picked randomly. The bits between these two points are swapped.

In the proposed genetic algorithm, we use a modification of the single point crossover to generate new population. First, we randomly generate a crossover point using a randomizer. Then we take the genes from the source to the crossover point from parent 1, and from the crossover point to the destination node from parent 2 and combine them to form the child chromosome.

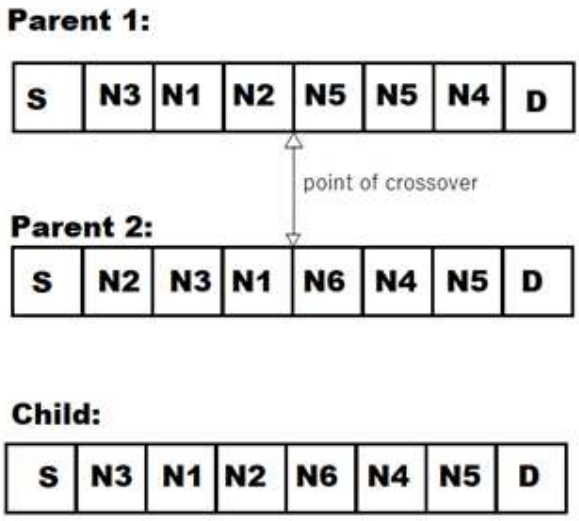

Fig.3: Single Point Crossover

In the above diagram, we take two chromosomes parent one and parent two. A point of crossover is generated using a randomizer. Here, we see that the fourth index is the point of crossover. The child chromosome is generated by taking the nodes from source gene S to the point of crossover from parent one and from the point of crossover to the destination gene D from parent two.

F. Mutation

We need a mutation operator to maintain genetic diversity in our population. In mutation, the solution may change entirely from the previous solution. This way the genetic algorithm may move towards a better solution. Mutation occurs based on a user-defined mutation probability. The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.
The value of the mutation probability is critical. If the value is set too high, the search will turn into a primitive random search. Thus, we must keep it just enough to enable our algorithm to reach the solution optimally without being stuck at the local optimum. In the proposed genetic algorithm, we use a probability of five percent for mutation.

Here, a combination of a randomizer and swap is used. We choose two random genes based on a randomizer and then swap the values of these genes.

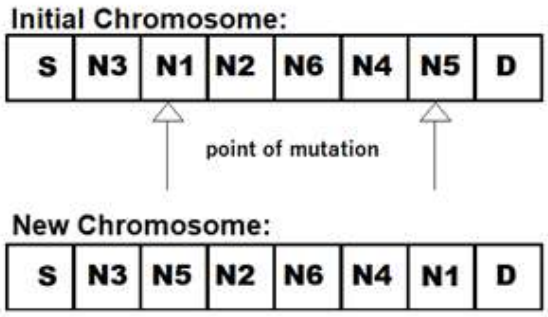

Fig.4: Mutation

In the above diagram, we can see that we have chosen two genes from initial chromosome using a randomizer. These two genes are swapped to generate a new chromosome which is the mutated version of the initial chromosome.

III. IMPLEMENTATION

1) Pseudocode

While (1/fitness is minimized)
{
Generate Chromosome
->Initialize the first and last gene of the chromosome as the source and destination respectively

Crossover (parent1,parent2)
->perform single point between the 2 parents

Selection (prev_population, curr_population)
->pop=Merge(prev_population, curr_population)
->Sort based on fitness value
->Pick 2 best individuals from the population as parents

Fitness (Chromosome)

```
def fitness(self, chromosome):
    chromosome_list = chromosome.get()
    return sum([self.weights[i][j] for i, j in zip(chromosome_list[:-1]
chromosome_list[1:]])
```

Generate Population(population_size)

while gen<max_gen
->gen++
->p=1
->select 2 parents randomly from the population
->newbie=Crossover (parent1, parent2)
newbie. Mutate ()
->fit=fitness(newbie)
->append newbie to the population

}

2) Results for random network size topologies

Shown below are the results. We compare the proposed algorithm to a well-known Bellman-Ford algorithm. We test both the algorithms with network sizes ranging from ten to one thousand units. A graph is plotted comparing the time taken by each algorithm to reach the required solution.
We also compare the output given by proposed algorithm for different mutation probabilities.

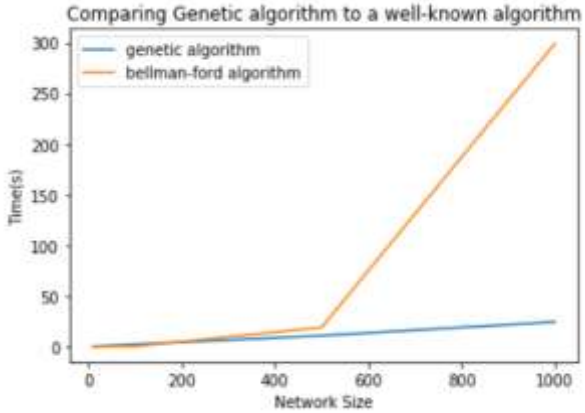

Fig.5: Mutation Probability: 5%

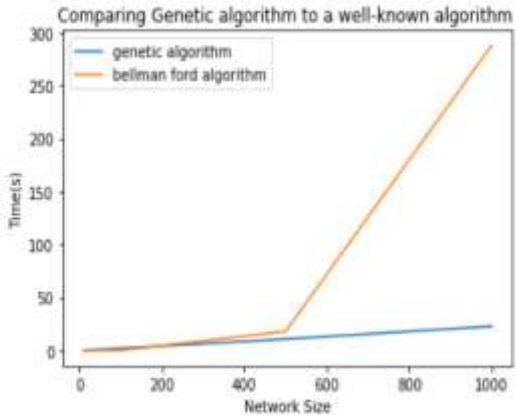

Fig.6: Mutation Probability: 20%

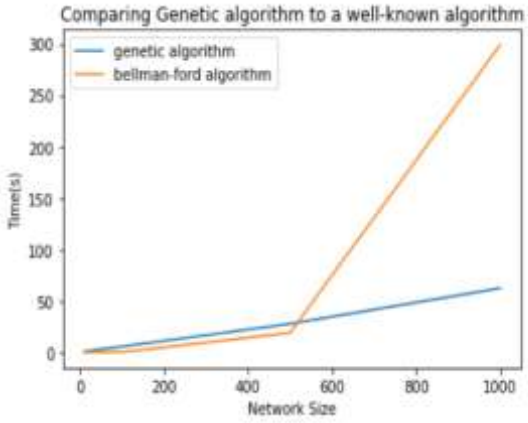

Fig.7: Mutation Probability: 50%

CONCLUSION

Genetic Algorithms has emerged immensely in the past two decades. It is an important device for optimization problems and a tool to approach solutions heuristically. Many of these are NP-hard problems. This means that as the input size grows, the computing time for traditional algorithms increases exponentially. But evolutionary computation algorithms like Genetic Algorithms provide a good approach to the optimal solution. Since Genetic Algorithms obey universal evolutionary concepts, this good solution can be achieved in reasonable computing time.

ACKNOWLEDGEMENTS

We would also like to thank The Information Science and Engineering Department, BMS College of Engineering, Bangalore, for providing us the required resources and facilities to carry out the research.

REFERENCES

[1] Gihan Nagib and Wahied G.Ali, Network Routing Protocol using Genetic Algorithms, International Journal of Electrical & Computer Sciences IJECS-IJENS Vol:10 No:02*(references)*

[2] Yong Deng, Yang Liu and Deyun Zhou, School of Electronics and 1)Information, Northwestern Polytechnical University, Xian, Shaanxi 710071, China 2)School of Computer and Information Science, Southwest University, Chongqing 400715, China- An IMproved Genetic Algorithm with Initial Population Strategy for Symmetric TSP

[3] Cooper, Jason. "Improving performance of genetic algorithms by using novel fitness functions." (2006).

[4] Y. Qiu, F. Liu and X. Huang, "Network Optimization based on Genetic Algorithm and Estimation of Distribution Algorithm," *2008 International Conference on Computer Science and Software Engineering*, Hubei, 2008, pp. 1058-1061.
doi: 10.1109/CSSE.2008.1511

[5] Chang Wook Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566-579, Dec. 2002.
doi: 10.1109/TEVC.2002.804323

[6] Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators P. LARRANAGA, C.M.H. KUIJPERS, R.H. MURGA, I. INZA and ˜ S. DIZDAREVIC Dept. of Computer Science and Artificial Intelligence, University of the Basque Country, P.O. Box 649, E-20080 San Sebastian, The Basque Country, Spain