Network Routing Protocol using Genetic Algorithms

Gihan Nagib and Wahied G. Ali

***Abstract*— This paper aims to develop a genetic algorithm to solve a network routing protocol problem. The algorithm has to find the shortest path between the source and destination nodes. In the literature, the routing problem is solved using search graph techniques to find the shortest path. Dijkstra's algorithm is one of popular techniques to solve this problem. The developed genetic algorithm is compared with Dijkstra's algorithm to solve routing problem. Simulation results are carried out for both algorithms using MATLAB. The results affirmed the potential of the proposed genetic algorithm. The obtained performance is similar as Dijkstra's algorithm.**

***Index Term*— Search methods, Genetic Algorithms, Protocols, Routing.**

I. INTRODUCTION

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959 [1] is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree [2]. This algorithm is often used in routing. An equivalent algorithm is developed by Edward F. Moore in 1957 [3]-[4]. For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. The shortest path first is widely used in network routing protocols, most notably OSPF (Open Shortest Path First). OSPF is a dynamic routing protocol. It is a link state routing protocol and is part of the interior gateway protocols group. OSPF keeps track of the complete network topology and all the nodes and connections within that network. The basic workings of the OSPF routing protocol are as follows:

Gihan Nagib is with the Information Technology Department, College of Computer and Information Sciences, King Saud University, KSA (e-mail: gihan@ksu.edu.sa – gihannagib949@hotmail.com).
Wahied G. Ali is with Electrical Engineering Department, College of Engineering, King Saud University, KSA, on leave from Ain Shams University, Cairo, Egypt. (e-mail: wahied@ksu.edu.sa).

A. Startup

When a router is turned on it sends Hello packets to all neighboring devices, and the router receives Hello packets in response. From here routing connections are synchronized with adjacent routers that agree to synchronize.

B. Update

Each router will send an update message called its “link state” to describe its database to all other devices. So that all the routers connected together have an up to date description of each topology that is connected to each router.

C. Shortest path tree

Each router will calculate a mathematical data structure called “shortest path tree” that describes the shortest path to the destination address, this is where OSPF gets its name. It will try to open the shortest path first.

OSPF routing protocol is a very important protocol to consider when setting up routing instructions on the network. As OSPF gives the routers the ability to learn the most optimal (shortest) paths it can definitely speed up data transmission from source to destination. In the literature, Dijkstra's algorithm is often described as a greedy algorithm. The *Encyclopedia of Operations Research and Management Science* describes it as a "*node labeling greedy algorithm*" and a greedy algorithm is described as "*a heuristic algorithm that at every step selects the best choice available at the step without regard to future consequence*" [4].

Routing is a process of transferring packets from source node to destination node with minimum cost (external metrics associated with each routing interface). Cost factors may be the distance of a router (Round-trip-delay), network throughput of a link or link availability and reliability expressed as simple unit less numbers. Hence routing algorithm has to acquire, organize and distribute information about network states. It should generate feasible routes between nodes and send traffic along the selected path and also achieve high performance [5]. Routing process uses a data structure called routing table at each node to store all the nodes which are at one hop distance from it (neighbor node). It also stores the other nodes (hop count more than one) along with the number of hops to reach

that node, followed by the neighbor node through which it can be reached. Router decides which neighbor to choose from routing table to reach specific destination. In the literature, different approaches are applied to solve this problem as: Dijkstra's algorithm[6], dynamic programming technique [4], and emerged ants with genetic algorithm [5], [7].

This paper is organized as follows. The literature work and the routing problem definition are presented in section I. Section II describes the basics of Dijkstra's algorithm. While section III; gives a brief description of the genetic algorithms as existed in the literature. The developed genetic algorithm to find the shortest path is introduced in Section IV. Simulation results are presented and discussed in Section V. Finally, conclusion is drawn in Section VI.

II. Dijkstra's Algorithm

The Dijkstra's algorithm calculates the shortest path between two points on a network using a graph made up of nodes and edges. It assigns to every node a cost value. Set it to zero four source node and infinity for all other nodes. The algorithm divides the nodes into two sets: tentative and permanent. It chooses nodes, makes them tentative, examines them, and if they pass the criteria, makes them permanent. The algorithm can be defined by the following steps [6]:

1. Start with the source node: the root of the tree.
2. Assign a cost of 0 to this node and make it the first permanent node.
3. Examine each neighbor node of the node that was the last permanent node.
4. Assign a cumulative cost to each node and make it tentative.
5. Among the list of tentative nodes
   a. Find the node with the smallest cumulative cost and mark it as permanent. A permanent node will not be checked ever again; its cost recorded now is final.
   b. If a node can be reached from more than one direction, select the direction with the shortest cumulative cost.
6. Repeat steps 3 to 5 until every node becomes permanent.

If the algorithm is applied to the network in Fig. 1 to calculate the shortest path between the source node a(1) and the destination node b(5); the shortest path will be 1-3-6-5 with cost 20.

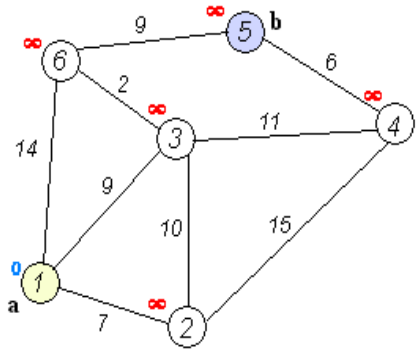

Fig. 1. Network topology

III. Genetic Algorithms

Genetic algorithms (GAs) are global search and optimization techniques modeled from natural selection, genetic and evolution. The GA simulates this process through coding and special operators. The underlying principles of GAs were first published by [8]. Excellent reference on GAs and their applications is found in [9]. A genetic algorithm maintains a population of candidate solutions, where each candidate solution is usually coded as binary string called a chromosome. The best choice of coding has been shown to be a binary coding [8]. A set of chromosomes forms a population, which is evaluated and ranked by fitness evaluation function. The fitness evaluation function play a critical role in GAs because it provides information how good each candidate. The initial population is usually generated at random. The evolution from one generation to the next one involves mainly three steps: fitness evaluation, selection and reproduction [10].

First, the current population is evaluated using the fitness evolution function and then ranked based on their fitness. A new generation is created with the goal of improving the fitness. Simple GA uses three operators with probabilistic rules: reproduction, crossover and mutation. First selective reproduction is applied to the current population so that the string makes a number of copies proportional to their own fitness. This results in an intermediate population.

Second, GA select "parents" from the current population with a bias that better chromosome are likely to be selected. This is accomplished by the fitness value or ranking of a chromosome.

Third, GA reproduces "children" (new strings) from selected parents using crossover and/or mutation operators.

Crossover is basically consists in a random exchange of bits between two strings of the intermediate population. Finally, the mutation operator alters randomly some bits of the new strings. This algorithm terminates when an acceptable solution is

found, when convergence criteria are met or when a predetermined limit number of iteration is reached. The main features of GAs are that they can explore the search space in parallel and don't need the optimized function to be differentiable or have any smooth properties. The precision of the solution obtained depends on the number of bits used to code a particular variable (length of chromosome) and a sufficient number of iterations.

IV. PROPOSED ALGORITHM

The network under consideration is represented as a connected graph with N nodes. The metric of optimization is the cost of path between the nodes. The total cost is the sum of cost of individual hops. The goal is to find the path with minimum total cost between source node and destination node. This paper presents a simple and effective genetic algorithm GA to find the shortest path. The detailed of the algorithm are given in the following subsections; while the investigation of the performance is achieved via a simulation work in the next section.

In the proposed algorithm, any path from the source node to the destination node is a feasible solution. The optimal solution is the shortest one. At the beginning a random population of strings is generated which represents admissible (feasible) or un-admissible (unfeasible) solutions. Un-admissible solutions are strings that cannot reach the destination. That means, the string solution would lead to a path without link between nodes. Admissible solutions are strings that can reach the target. The un-admissible solution has lowest fitness (zero fitness). Consider the network topology in Fig. 2 with 10 nodes. This network will be simulated in the next section.

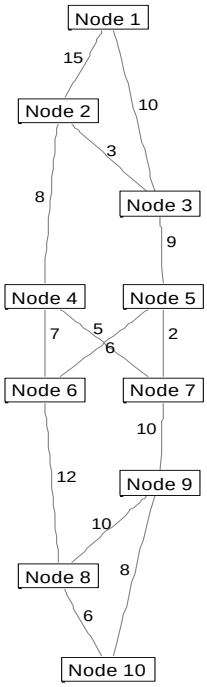

Fig. 2. Network topology for simulation [11]

Each node has a number in Fig. 2 and the nodes are used to encode a path as a string expressed by the order on numbers. For example, 1-3-5-6-8-10 is a feasible path with source node 1 and destination node10; it might be optimal or not. Define via nodes as all nodes except the source and destination (2, 3, … 9). The chromosome is represented by a string bits (i.e. natural representation not binary). Hence special crossover and mutation operators must be adopted. During the crossover, a string that has an efficient fitness is randomly selected as a parent. If the second parent contains the common number in the first one; both strings exchange the part of their strings following the common number. If not another string is selected as the second parent and the same procedure is followed. For example using the network in Fig. 2:

Parent1: 1-3-5-7-9-10
Parent2: 1-2-4-7-9-8-10

At via node 7; the underlined parts of each string are exchanged, yielding: the two children are:

Child1: 1-3-5-7-9-8-10
Child2: 1-2-4-7-9-10

After crossover has been achieved; children are checked to determine whether each string has repeated number. If so, the part of string between the repeated numbers is cut off. Some correction then required because it might be the case that the child is not admissible solution [9]. This approach makes the algorithm more complex. The important question is how to make the binary coding so possible and easy to be used in a fixed length chromosome definition. One of the more challenging aspects in our proposed method is encoding each string in a binary code with fixed length. The path is encoded using binary numbers where each gene (node) in a chromosome is encoded by 4 bits binary as in the Table I. The number of bits has to be sufficient to encode the network nodes.

Table I
Binary coding of network nodes

Node	Binary code	Linked nodes
1	0001	2,3
2	0010	3,4
3	0011	5
4	0100	6,7
5	0101	6,7
6	0110	8
7	0111	9
8	1000	10
9	1001	8,10
10	1010	destination

The following subsections describe the important components of the proposed genetic algorithm.

A. Chromosomes and Initialization

A chromosome corresponds to possible solution of the optimization problem. Thus each chromosome represents a path which consists of a set of nodes to complete the feasible solution, as the sequence of nodes with the source node followed by intermediate nodes (via nodes), and the last node indicating the destination, which is the goal. The default maximum chromosome length is equal to the number of nodes times the gene length (4-bit binary code/gene). The network has 10 points where each point is coded in 4 binary bits. That means; the chromosome length is equal to 10X4=40 bits.

The chromosome structure is given in Fig. 3. The first gene represents node 1 (source node) which is coded in 4 binary bits, next gene is node 3, next gene is node 5 and so on. Successive genes in the chromosome are coded similarly. The initial population of chromosomes can be randomly generated such that each chromosome has a random genes (via nodes), while the start and goal nodes are fixed in the population.

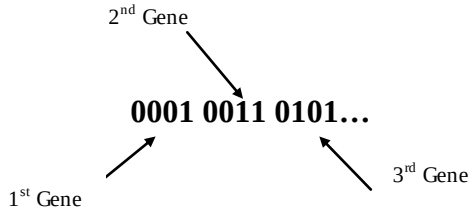

Fig. 3. Chromosome structure

B. Evaluation

The choice of a fitness function is usually very specific to the problem under condition. The evaluation function of a chromosome measures the objective cost function. The cost of a path indicated by the chromosome is used to calculate its fitness. Since the fitness should increase as the cost decreases. Thus, the fitness function (F) of a path is evaluated as defined in equation (1):

$$F = \begin{cases} \dfrac{1}{\sum_{i=1}^{N-1} C_i\,(g_i, g_{i+1})} & ; \quad \text{Feasible path} \\ 0 & ; \quad \text{Infeasible path} \end{cases} \qquad (1)$$

Where $C_i(g_i, g_{i+1})$ is the cost between gene g_i and adjacent gene g_{i+1} in the chromosome of N genes (Nodes). The cost between linked nodes is given in Fig. 2. If the path is not feasible, its fitness is equal to zero. The proposed algorithm can trace the path points to detect if it is feasible or not using the information in table II. The linked nodes are the admissible next nodes to the current one in the solution.

C. Operators

The algorithm uses the common two genetic operators: ***crossover*** and ***mutation***. Crossover recombines two 'parent' paths to produce two 'children' new paths in the next generation. Two points crossover is used. Both parent paths are divided randomly into three parts respectively and recombined. The middle part of the first path between crossover bit positions and the middle part of the second path are exchanged to produce the new children. The crossover bit positions are selected randomly along the chromosome length between bit positions 5 and 36. These limits are chosen in order to keep the start and destination nodes without change during the crossover process. The mutation process is also applied to flip randomly a bit position in the chromosome (between bit position 5 and 36). The pseudo-code of the proposed algorithm is given by:

BEGIN
  Initialize *the start and destination points*
  Generate *randomly the initial population using via nodes in each chromosome*
  While NOT (*convergence condition****) DO***
    Evaluate *the fitness for each chromosome in current population using equation (1)*
    Rank *the population using the fitness values*
    Eliminate *the lowest fitness chromosome*
    Duplicate *the highest fitness chromosome*
    Apply *randomly crossover process between current parents using the given probability, while keeping the start and end nodes without change in the population*
    Apply *the mutation process with the given probability*
    Generate *the new population*
  END
  Output the best individual found
END

V. SIMULATION WORK

Consider the network topology in Fig. 2. The numbers across each link represent distances or weights. The objective is to find the shortest path for the source node 1 to reach destination node 10.

Dijkstra's Algorithm

The network is created using the following MATLAB command:

```
net = sparse([1 1 2 2 3 4 4 5 5 6 7 8 9 9],
[2 3 3 4 5 6 7 6 7 8 9 10 8 10],
[15 10 3 8 9 7 5 6 2 12 10 6 10 8],10,10)
```

The first three row vectors in the sparse command represent source nodes, destination nodes, and equivalent costs respectively. The MATLAB command graphshortestpath is executed to find the shortest path. This command applies Dijkstra's algorithm as the default one to find the optimal solution.

```
[cost, path, pred] = graphshortestpath(net,1,10)
```

The obtained solution is:

```
cost =39
path = 1   3   5   7   9   10
pred = 0   1   1   2   3   5   5   6   7   9
```

That means; the shortest path from node 1 to node 10 has to pass via nodes 3, 5, 7, 9 wit minimum cost 39. The network is viewed using the following command in MATLAB

```
view(biograph(net,[],'ShowArrows','off','ShowWeights','on
```

Proposed Genetic Algorithm

The MATLAB environment has a very powerful string manipulation commands that helps to convert easily the numeric variables into strings and vice versa. Consequently, the bit crossover and mutation were so easily to be implemented in the developed program. Given the source node is 1 and the goal node is 10. The network in Fig. 2 is simulated to find the shortest path. The initial population consists of 6 chromosomes 40 bits of each. The probability of crossover was chosen as 70% and the mutation rate equal to 2.5%. The best results are obtained using two points of crossover. The simulation result after 30 generations is given by:

```
1   3   5   5   7   7   7   9   10   10
1   3   5   5   7   7   7   9   10   10
1   3   5   5   3   7   7   9   10   10
1   3   5   5   7   7   7   9   10   10
1   3   5   5   7   7   7   9   10   10
1   3   5   5   7   7   7   9   10   10
```

The third chromosome is slightly different due to mutation process. The part of repeated numbers is cut off and the final result has been converged to the optimal/shortest individual **{1-3-5-7-9-10}** with the cost of **39;** which is the same results as Dijkstra's algorithm. The program is tested with other source and destination points. The obtained results affirmed the potential of the proposed algorithm where the convergence was guaranteed to obtain the optimal path in each case.

VI. CONCLUSION

A simple genetic algorithm is developed to find the shortest path routing in a dynamic network. The developed algorithm uses an efficient coding scheme. The chromosome length depends on the number of nodes in the network. The MATLAB environment searches the shortest path using Dijkstra's algorithm as the default one. The algorithm is simulated to solve the network of 10 nodes for the first one as the source node. Also, the developed GA is simulated to find the solution for the same problem. The obtained results affirmed the potential of the proposed algorithm that gave the same results as Dijkstra'a algorithm. In the future, the developed GA will be more investigated to decrease the chromosome length especially for network with a large number of nodes.

REFERENCES

[1] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematlk* **I**, 269 – 271, l 959.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Section 24.3: Dijkstra's algorithm. *Introduction to Algorithms* (Second ed.), MIT Press and McGraw-Hill, pp. 595–601, 2001.

[3] E. F. Moore, "The shortest path through a maze", *Proceedings of an International Symposium on the Theory of Switching (Cambridge, Massachusetts, 2–5 April 1957). Cambridge: Harvard University Press*, pp. 285–292, 1959.

[4] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion". *Journal of Control and Cybernetics* **35** (3): 599–620, 2006.

[5] N. K. Cauvery and K. V. Viswanatha, "Routing in Dynamic Network using Ants and Genetic Algorithm", *Int. J. of Computer Science and Network Security*, Vol. 9 No.3, pp.194-200, March 2009.

[6] B. A. Forouzan, *Data Communications and Networking*, Fourth Edition McGraw Hill, 2007.

[7] N. Selvanathan and W. J. Tee, "A Genetic Algorithm Solution to Solve The Shortest Path Problem OSPF and MPLS", *Malaysian Journal of Computer Science*, Vol. 16 No. 1, pp. 58-67, 2003.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI, The University of Michigan Press, 1975.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.

[10] Y. John and R. Langari, *Fuzzy Logic: Intelligence, Control, and Information*, Prentice-Hall Inc., Ch (17), pp.469-470, 1999.

[11] Finding shortest path in a Network using MATLAB, Available online at http://hubpages.com/hub/Shortest-Path-Routing---Finding-Shortest-Path-in-Network-using-MATLAB