ISSN: 0975-766X
CODEN: IJPTFI

Available Online through ***Research Article***
www.ijptonline.com

SHORTEST PATH TRAVERSAL FOR A FULLY CONNECTED NETWORK USING GENETIC ALGORITHM

Dr. B.Sreedevi*, K.Surya, G.Senbhaga Priya**, S.Gayathri****
* Assistant Professor, **B.Tech Student
Department of Computer Science and Engineering,
Srinivasa Ramanujan Centre, SASTRA University, Kumbakonam. 612 001.
Email: sreedevi@src.sastra.edu

Received on: 20.10.2016 *Accepted on: 25.11.2016*

Abstract

Needs are increasing as days passes by for shortest path traversal of nodes in the field of Networking. Shortest path traversal is mainly required for speedy transfer of data from source node to destination node. There are several techniques in existence for the shortest path traversal. In this paper we have proposed the use of genetic algorithm (GA) to find the shortest path traversals using an objective function, for producing an optimized path, for a fully connected network, by mutation of genes.

Keywords - objective function, gene mutation, genetic algorithm, shortest path.

I. Introduction

A technique that helps in finding exact or approximate solutions to search problems and optimization in computing is Genetic Algorithm (GA). It is a particular kind of evolutionary algorithm. It is a particular kind of evolutionary algorithm. They are heuristic search and optimization technique, which mimic the process of natural evolution. Thus, GA implements the optimization strategies of simulating the evolution of species through natural selection. The natural evolution techniques are inheritance, mutation, selection and crossover [1, 2]. The fitness function value quantifies the optimality of a solution. A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem. The main principle on selection is “select the best, discard the rest” [3, 4].

II. Working mechanism

Genetic Algorithm is a class of machine-learning techniques that gain their name from a similarity to certain processes that occur in the interactions of natural, biological genes. In our algorithm we use objective function, which is the judge

of the genetic algorithms attempt to solve a problem. Genetic algorithm does not know how to derive a problem's solution, but they do know, from the objective function, how close they are to a better solution. Each attempt a genetic algorithm makes towards a gene to find a traversal such that all the nodes are exactly visited once is called gene- a sequence of information that can somehow be interpreted in the problem space to yield a possible solution. The genetic algorithm which we have presented is analogous to a biological gene in that both are representation of alternative solutions to a problem. In the biological world, the problem is evolutionary survival, and a particular gene represents one possible solution to survival within a competitive environment [2, 5]. Our genetic algorithm maintains a collection or population of genes.

Each gene in the population may represent a different sequence and a different about how to solve the problem, thus satisfying the objective function. The objective function which we have crafted for our algorithm is able to interpret the data contained within a gene and decide how good a solution it represents. Fitness value is associated with each generated gene. After an encoding and an objective function have been formulated, the person running a genetic algorithm may step back and let it do the work. Through a sequence of steps, the genetic algorithm will work toward making its genes more fit.

Again, we find a biological analogy in the optimization steps. Each pass through the set of optimization steps is called a generation. If all is going well, we expect the overall fitness of the genes to increase as we pass through generations. Optimization steps in each generation involve reproduction, crossover and mutation. We use the fitness of a gene, conferred by the objective function; to decide how likely that gene is to reproduce. To put it all together then, a genetic algorithm can be seen as a series of steps. Initially, a random, population of genes is created. Then we attempt to optimize the fitness of the genes by running through generations of optimization steps. An objective function is used throughout, to judge the fitness of members of the population [6, 7].

III. Traversal

This paper dealt about the Mesh topology networks we have been traversed using Genetic algorithm to find optimum solutions. The condition of traversal of 3 nodes is simple and the implementation in GA also will be less complex. The complexity increases as the number of nodes increases.

Traversal of four nodes. In this we are going to traverse four nodes of a Mesh topology shown in the fig.1.

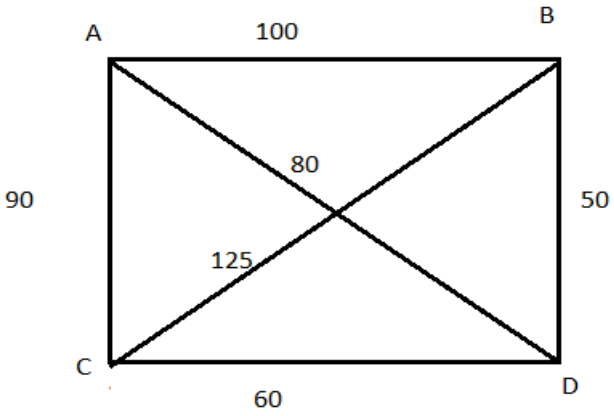

Fig. 1.Traversal of four nodes.

Some of the sequences in which the nodes can be traversed are given below:

1. A-B-D-C (210)

2. A-C-D-B (200)

3. A-B-C-D (285)

4. B-A-D-C (240)

5. C-D-B-A (210)

6. C-A-B-D (240)

7. C-B-D-A (315)

8. D-C-A-B (250)

9. D-B-A-C (240)

Since the traversal A-C-D-B has the smallest weight, it is concluded that as the shortest path. Genetic algorithm will perform huge number of passes or attempts to find the correct solution. Each time Mutations are also done in order to find the nearest solution. The process will proceed as per the Fitness function in order to achieve the best value [2].

IV. Glimpse of the entire process

1. Formulating appropriate objective function

2. Mutation

3. Calculating the fitness value

The three steps that followed previously give us a slight glimpse of the entire process in the algorithm which we have proposed. The first step is the main one which is concerned with the objective function for our genetic algorithm. It

predicts a distance randomly via a gene which is used in our objective function for calculating the fitness value of other genes produced after the initial step.

The second step involved is mutation, which forms the core of the algorithm. Mutation is carried out till the number of steps we prefer [3, 5]. It should be noted that higher value of mutation is generally preferable because it has got the higher probability of producing a gene with good fitness value.

After the production of gene, we make it to traverse a path such that in its traversal it covers every node uniquely one time. And the distance taken by that gene is calculated so that it can be further used by us for calculating the fitness value of that particular gene.

V. Pseudocode

The pseudo code for the entire process is given below:

Algorithm mutate ()

1. for i=0 to mut_val then
2. flag=true
3. while (flag)
4. index=produce a node for the gene
5. check whether index is unique in array path
6. if (index_is_unique)
7. path[path_index]=index//path_index value 0
8. path_index++
9. else
10. goto step 4 again.
11. if (path_index==no_nodes)
12. flag=false

The pseudo code presented above illustrates the mutation process involved in the algorithm. First of all we are looping the whole function till the number of optimization steps required. This value is given explicitly by the user. Generally a higher value of optimization steps produces a healthy result.

We are then producing a node for a particular gene and appending it to an array path if no duplicate node exists previously in that array. But if a duplicate node already exists then we produce a node again for the gene and apply the same conditions as previously stated.

Then if at a certain stage of that algorithm, the index of the array path equals with the total number of nodes present in the network then, the gene is completely produced. Then other trivial things can be performed like calculating the distance in that path, so that it can be used further up for calculating the fitness value for that particular gene.

Algorithm objective_function ()

1. index=produce a node
2. check whether index is unique in array path
3. if (index_is_unique)
4. objpath_array[array_index]=index
5. array_index++
6. else
7. goto step 1
8. if (array_index==no_nodes)
9. calculate distance in that path and store it in a variable opt_dist

The algorithm for objective function nearly follows the same procedure of mutation with few exceptions. First of all a node is produced for a gene. The produced node is checked in the array objpath_array for its uniqueness. If it is found to be unique, then it is appended in the objpath_array and the array index is updated as required. But if the produced node is not a unique one, then the node is produced again and the same steps are repeated as stated earlier. As each node is produced, we also check whether objpath_array's index (array_index) is equal to the number of nodes present in the network.

If the above condition becomes true, then we calculate the distance totally covered by that gene in that particular path and store it in a variable called opt_dist which stands for optimized distance for a gene.

Algorithm fitness_value(gene_dist)

1. fitness=opt_dist-gene_dist

2. return fitness

The most important parameter required before calculating the fitness value for any particular gene is that we should know its gene distance beforehand. We also use the opt_dist variable previously calculated in objective function for calculating the fitness value for a given gene. We take the difference of the opt_dist and the gene distance for getting the fitness value for a particular gene. We should note here that calculating the fitness value for any gene is much simpler here because we explicitly know the value of the distance covered by a gene in its traversal.

VI. Experimental Results

We have implemented this algorithm in C++ using Qt library under Linux platform [8]. Several Mesh topology networks of varying nodes have been traversed using genetic algorithm and the values are entered in the table given below:

Table 1. Table Observations.

No of Nodes	Values observed		
	Distance covered by objective function	Best fitness values	Time taken to traverse (min)
4	54	9	1.16
8	156	50	3.14
12	353	167	10.12
16	580	67	17.15
20	802	143	14.17

Here the time taken to traverse the nodes is taken along X-plot and the best fitness value achieved is taken along Y-plot. The unit of time is taken in minutes.

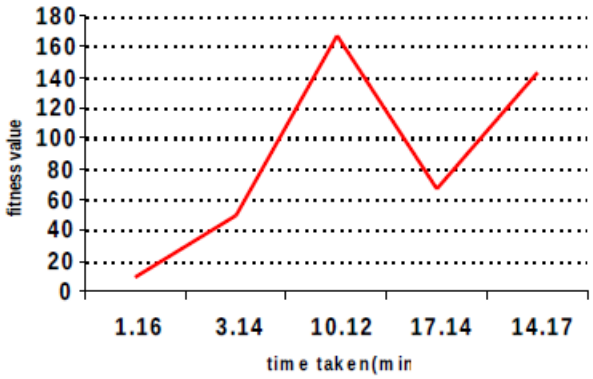

Fig.2. A graph depicts the fitness value and time taken during traversal.

Conclusion

In this paper, we have seen in our algorithm that, as the number of optimization steps involved in it increases, the fitness value of a gene also gradually increases. Best solution can be offered by a gene as the number of optimization steps involved in it, increases. The formulation of the objective function is a tricky issue. It is the major parameter deciding the fitness of a gene.

We present a simple yet efficient way of calculating objective function in our algorithm. In this algorithm, we illustrate a way of calculating the best path for even n number of nodes for a mesh topology network. This is one of the major advantages of the algorithm. One of the major factors affecting genetic class of algorithms is its time taken to produce an optimized gene. Even our algorithm suffers from this, as we can infer from the experimental results. But still, our algorithm holds true in a long run where it can produce gene of high fitness value. In future, we are going to extend this type of algorithm for other network topologies.

References:

1. David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", 2nd ed., 1989.
2. L. D. Davis , Melanie Mitchell, Van Nostrand Reinhold Ed., Handbook of Genetic Algorithms, 1991.
3. Melanie Mitchell , An Introduction to Genetic Algorithms, feb 1998.
4. Reeves, Colin R., Rowe, Jonathan E, Genetic Algorithms – Principles and Perspectives A Guide to GA Theory, Series: Operations Research Computer Science Interfaces Series, Vol. 20, pg. 344, 2002.
5. Gilbert Held,Wireless Mesh Networks,2005.
6. Mike Marrow,"Genetic Algorithms-A new class of searching algorithms".
7. Richard Spillman, "Genetic Algorithms-Nature's way to search for the best".
8. Jasmine Blanchette, Mark Summerfield, C++ GUI programming with Qt4, 2006.