STEADY STATE GENETIC ALGORITHM APPROACH TO ROUTING IN ADHOC NETWORK

[1]SUMMIYA.A.PATHAN, [2]BASAVARAJ S. MALAPUR

[1,2]Computer Science and Engineering Department,
Basaveshwar Engineering college, Bagalkot, India
E-mail: [1]summias@gmail.com, [2]Basu.malapur@gmail.com

Abstract— Routing policies plays essential role in how traffic is forwarded across the network. Adhoc network is a collection of wireless mobile nodes forming a temporary network without the aid of any established infrastructure or centralized administration. Adhoc networks require a highly adaptive routing scheme. One of the most problems encountered in these networks, is finding the shortest path (SP) between source and destination in a specified time so as to satisfy the Quality of Service (QoS). QoS routing in an Adhoc network is difficult as network topology changes constantly and time consuming because of different QoS parameters like distance/cost and energy and the available state information for routing is inherently imprecise. This work presents Genetic algorithm approach to find the optimal path for Adhoc networks, GA generate solutions to optimization problems using techniques inspired by natural evolution. Appropriate chromosome structure, crossover and mutation operations and fitness functions are defined.

Keywords— Adhoc Network, Quality of Service, Genetic Algorithm.

I. INTRODUCTION

A wireless Adhoc network is a decentralized type of network. Here nodes are not familiar with the topology of their networks, Instead they have to discover it. The network is Adhoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in infrastructure. Each node participates in routing by forwarding data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. Routing is the process of selecting paths in a network along which to send network traffic. 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. Routing in conjunction with congestion control and admission control defines the performance of the network. Routing directs packet forwarding from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Broadly there are mainly two types of routing policies - static and dynamic. In static routing, the routes between the nodes are pre-computed based on certain factors for example bandwidth, buffer space etc. and are stored in routing table. All packets between any two nodes follow the same path [1]. When network topology changes the path between two nodes may also change and static routing fails. Hence in dynamic routing policy, the routes are not stored but are generated when required. The new routes are generated based on the factors like traffic, link utilization, bandwidth, jitter, network delay, hop count, path cost, energy, load, MTU, reliability, communication cost etc which is aimed at having maximum performance.

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. GA's are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [6]. New offspring are generated /evolved from the chromosomes using operators like selection, crossover and mutation. Most fit chromosomes are moved to next generation. The weaker candidates get less chance for moving to next generation. This is because GA is based on the principle of Darwin theory of evolution, which states that the "survival is the best" [1]. The GA object determines which individuals should survive, which should reproduce, and which should die. It also records statistics and decides how long the evolution should continue. This process repeats until the chromosomes have best fit solution to the given problem. The summary is that the average fitness of the population increases at each iteration, so by repeating the process for many iterations, better results are discovered. Typically a GA has no obvious stopping criterion. We must tell the algorithm when to stop.

There are four basic categories of genetic algorithms. The first is the standard 'simple genetic algorithm'. This algorithm uses non-overlapping populations and each generation creates an entirely new population of individuals. The second is a 'steady-state genetic algorithm' that uses overlapping populations. In this variation, you can specify how much of the population should be replaced in each generation. The third variation is the 'incremental genetic algorithm', in which each generation consists of only one or two children. The incremental genetic algorithms allow custom replacement methods to define how the new generation should be integrated into the population.

So, for example, a newly generated child could replace its parent, replace a random individual in the population, or replace an individual that is most like it. The fourth type is the 'deme' genetic algorithm. This algorithm evolves multiple populations in parallel using a steady-state algorithm. Each generation the algorithm migrates some of the individuals from each population to one of the other populations. In our work we have used steady-state genetic algorithm [8,12,13] .

II. PROBLEM DEFINITION AND PROPOSED MODEL

Given the set of nodes (N) the Adhoc network under consideration is represented as G = (V, E), a fully connected graph with N nodes. The total distance (∑D) is the sum of distance between the nodes in path. The total energy (∑E) is the sum of energy between the nodes in path. The goal is to find the optimum path between source node Vsrc and destination Vdest, where Vsrc and Vdest belong to V. We start out with initial population (first generation). Every string in this generation is evaluated according to its fitness value. Next, a new generation is produced by applying the reproduction operator. Pairs of strings of the new generation are selected and crossover is performed, with a certain probability genes and mutated before all solutions are evaluated again. This procedure is repeated until the termination criteria are met. While doing this, the all time best solution is stored and returned at the end of the algorithm.

2.1 Mathematical Model

Given a undirected graph G(V, E) where V is the set of nodes and E is the set of edges, each link V1->V2 ϵ E is associated with two criteria. Distance D_i(V1->V2) and Energy E(V1->V2) for i=1 to |E|. The problem is to find the optimal path between given source 'S' and destination 'D' having maximum value of the fitness function.

$$f = \frac{\sum E \times path}{\sum D} \quad \text{where path} = \begin{cases} 1000 & \text{if path exists} \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

Algorithm 1: Routing in Adhoc Network using Genetic Algorithm
Input: Source & Destination Nodes
Output: Optimal Path

Step 1: Generate initial population of chromosomes based on number of nodes in network.
Step 2: Initialize max no of iterations & termination_count

Step 3: for 1: max no of iteration do

1. Evaluate fitness of each chromosomes & store in a fitness array.
2. Find max fitness of populations.
3. If there is no change in max fitness within termination_count then break out as there is no improvement.
4. Select two parents (Parent1 & Parent2) for reproduction from population using roulette wheel selection based on their fitness value.
5. Generate two offspring (Child1 & Child2) using crossover operator.
6. Apply mutation operator on Child1 & Child2.
7. Find the fitness value of Child1 & Child2.
8. Find the least fit chromosomes in the current population.
9. If the fitness of child1 & child2 are greater than the least fit of the population replace such chromosomes with child1, child2
10. Repeat step3 till iteration is greater than max no of iteration break out .

Step 4: End for

Step 5: Find chromosome with the highest fitness in the population. The genes of this chromosome give the solution of optimal path, between source and destination.

Step 6: Stop

III. IMPLEMENTATION

The GA to find the optimal path is applied on the networks with N number of nodes designed in Matlab. The genetic structure of a chromosome is described using a fixed, finite alphabet. In GAs, the alphabet 0, 1 is usually used [2,4]. An example of a chromosome implemented is given in Table No 1. The chromosome represents a path between nodes n1, n2, n4. The nodes in graph are represented as 1 and other nodes which are not a part of the graph are represented as 0. Thus the chromosome is '1 2 4 0 0'. The size of the initial population depends on number of nodes. Time required for the program for execution is recorded. This helps us to decide on the time taken by the network that utilizes the GA.

Table No (1) : Genetic representation of 5 nodes

n1	n2	n3	n4	n5
1	1	0	1	0

In our implementation
populationSize = 10*numberofNodes;

The initial population for a certain run of the algorithm is shown in figure 1(nodes=5,popSize=50) first few are shown below

1	3	4	0	0
3	4	5	0	0
1	3	0	0	0
1	2	4	0	0
1	3	4	5	0
1	2	5	0	0
1	3	4	5	0
1	2	3	0	0
2	5	0	0	0
1	2	3	0	0
2	4	0	0	0
1	2	3	0	0
1	3	0	0	0
3	5	0	0	0
2	5	0	0	0
2	5	0	0	0

Fig 1 Example of initial population with 5 nodes

3.1 Fitness Function

A function is written to check if the chromosome holds any path. Using Equation 1 fitness is calculated based on the distance and energy so that the chromosomes can be selected based on their fitness. The equation is multiplied by a large number so that the chromosomes which contain a path have a higher probability of selection in the next step

3.2 Selection

In our implementation we have chosen the 'Roulette wheel selection' technique which we narrowed down as the best selection technique. The basic advantage of proportional roulette wheel selection is that it discards none of the individuals in the population and gives a chance to all of them to be selected [3]. Therefore, diversity in the population is preserved. Based on fitness value selection of two chromosomes from the initial population is done. The selected chromosomes will be called parent chromosomes which are passed on to next step for the application of other genetic operators.

Algorithm 2: Roulette Wheel Selection
Input: Fitness of all chromosome(fit), popSize
Output: Selection of two Chromosomes, Parent1 & Parent2

Step1: Find the sum of fitness values of all chromosome [CumulativeSum] using equation fitnessSum=cumu(fit)
Step2: Generate 2 different random numbers r1 & r2
Step3: In the Population,
If the cumulative Sum of ith chromosome >= r1
Then ith chromosome is selected as parent1
Step4: In the Population,
If the cumulative Sum of ith chromosome >= r2 then ith chromosome is selected as parent2 & Parent2!=Parent1
Step5: Stop

3.3 Crossover

Crossover is applied to both the selected chromosomes. There are various techniques for crossover. We have selected the single point crossover in our implementation of the GA. Single-point crossover calls for a point to be selected on the parent organism strings [2,3]. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the newly generated children which are passed onto the next step. The crossover probability is decided after experimentation and we have selected a crossover of 0.9. The figure 2 shows single point crossover.

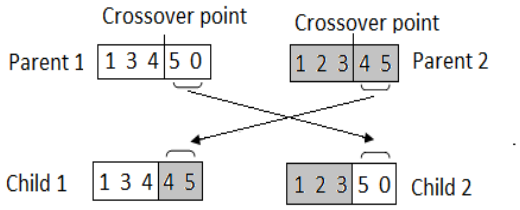

Fig 2: Single Point Crossover

Algorithm2: Single Pont crossover Operation
Input: Parent1, Parent2 (two chromosomes)
Output: child1, child2 (new two Chromosomes)

Step1 : index=Find number of common nodes in parent1 & parent2
Step2: Generate a random number which will indicate crossover point on parent1, parent2
Step3: if index>0 then do
1. All genes beyond crossover point is copied from parent2 keeping genes of parent1 from1 to crossover point, this gives child1
2. All genes beyond crossover point is copied from parent1 keeping genes of parent2 from 1 to crossover point, this gives child2
3. Eliminate the repeated nodes from child1, child2 else
Parent1, Parent2 is taken as child1, child2
Step4: Stop

3.4 Mutation

Mutation process brings in more randomness in the GA. We have implemented the multi-point mutation in which mutation operator replaces the value of two chosen genes with a random value selected between upper and lower bounds for that gene. Mutation occurs during evolution, according to a user-definable mutation probability which is usually low and we have defined a probability of 0.5. Mutation Technique as shown in figure 3, we have two random points generated and the node at that position is replaced with some other randomly generated node within the range and care is taken that replaced node is not already present in Child chromosome. After the mutation process the fitness of each of the child chromosomes is calculated and it is compared with all other chromosomes from the initial population. If any

of the chromosomes in the initial population has fitness less than the fitness of the children then that chromosome is eliminated and replaced by the child chromosome. This process is repeated until the termination criterion is met.

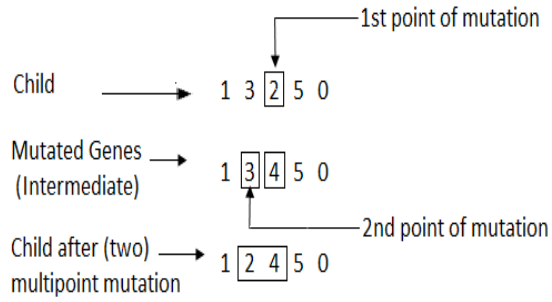

Figure 3: Mutation technique

Algorithm 3: Multi-point Mutation Operation
Input: child & range (Number of nodes)
Output: mutated_child

Step1: based on number of genes in child, generate I , J two random points on child for mutaion
Step2: while Ith point on child
1. mutnum1= Generate randomly a node within the range
2. If mutnum1 is not in child then Child(I)=mutnum1
   Else Repeat step2

Step3: while Jth point on child
1. Mutnum2= Generate randomly a node within the range
2. If mutnum2 is not in child then Child(J)=mutnum2
   Else Repeat step3

Step4: Stop

3.5 Termination

Maximum generations, No change in population fitness and stall generation are considered as algorithm stopping condition [4]. When the termination criterion is met it is assumed that the required fitness is achieved i.e the optimal path. In our implementation we have defined a certain number of iteration along with the criteria that the fitness generated does not change for a specified number of runs. At this time the clock is stopped to record the total time taken by the algorithm to arrive at the solution to the routing problem.

IV. EXPERIMENTS AND RESULTS

Figure 4 displays the optimal path, the time taken by the program to arrive at the solution, the fitness of the solution chromosome for 10 nodes Figure 5 displays the percentage of optimal paths obtained for networks with 5, 10, 15, 20, 25, 30, 40, 50 nodes and Figure 6 displays the time complexity with respect to all the topologies.

CONCLUSIONS

In the present work Genetic algorithms have been used for finding optimal paths between given source and destination in Adhoc Network. A novel Steady State node based chromosome structure has been devised to represent the possible solutions of the optimal path in the Adhoc networks. Further the crossover and mutation operators are devised for better efficiency. The Genetic algorithm has been tested on variety of network topologies and the results have been brought out. The overall performance of genetic algorithm for finding optimal paths under constraints such as cost/distance and Energy has been satisfactory

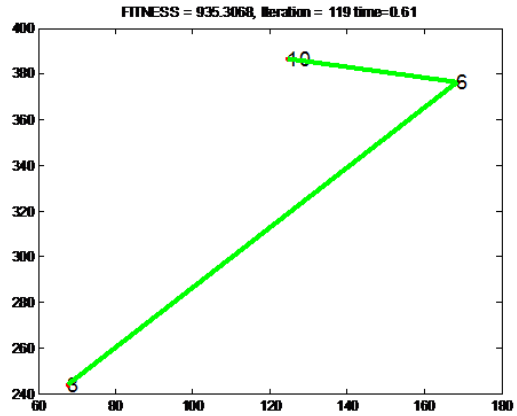

Figure 4: Optimal Path

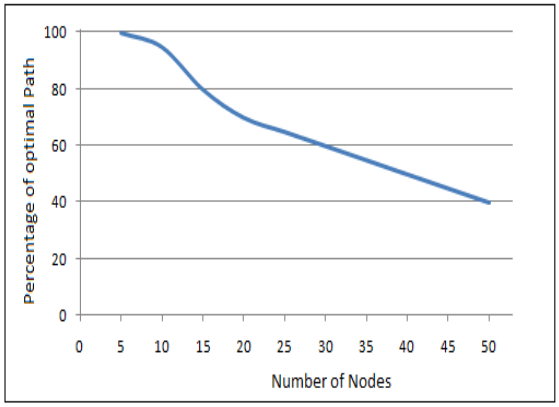

Figure 5: Optimal Path different Topology

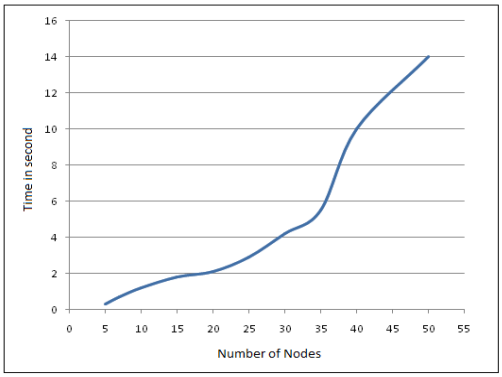

Figure 6: Time Complexity for different topology

REFERENCES

[1] Dr. Rakesh Kumar, Mahesh Kumar, "Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks ", Vol. 10 Issue 11 (Ver. 1.0) October 2010.

[2] Yousra Ahmed Fadil , "Routing Using Genetic Algorithm for Large Networks" ,Vol. 03 , No. 02 , pp. 53 -70 , December 2010.

[3] Susmi Routray, "An enhanced genetic algorithm for dynamic routing in ATM networks", Associate Professor (ITM), IMT Ghaziabad, India – 201001, Vol. 16 No.2 June , 2010 .

[4] Chao-Hsien Chu , G. Premkumar , Hsinghua Chou , "Theory and Methodology Digital data networks design using genetic algorithms", European Journal of Operational Research 127 (2000) .

[5] Bilal Gonen, "Genetic Algorithm Finding the Shortest Path in Networks", Department of Computer Science and Engineering, University of Nevada, Reno, Reno, Nevada, U.S.A.

[6] Usama Mehboob, Junaid Qadir, Salman Ali, and Athanasios Vasilakos, "Genetic Algorithms in Wireless Networking: Techniques, Applications, and Issues", arXiv:1411.5323v1 [cs.NI] 19 Nov 2014

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

[8] N. Selvanathan and Wee Jing Tee, "A genetic algorithm solution to solve the shortest path problem in OSPF and MPL's",Malaysian Journal of Computer Science, Vol. 16 No. 1, June 2003.

[9] Pravin.M, Munaf.S and Vetrivelan.P, " An approach for DSP routing in MANET using genetic algorithm", Research International Journal of Recent Scientific Research vol. 3, Issue, 7, pp. 647 - 651, July, 2012

[10] Stavroula Siachalou, Leonidas Georgiadis,"Efficient QoS Routing",0-7803-7753-2/03/$17.00 (C) 2003 IEEE IEEE INFOCOM 2003

[11] Noraini Mohd Razali, John Geraghty, " Genetic Algorithm Performance with Different Selection Strategiesin Solving TSP", Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011, London, U.K.

[12] F.Vavak and T.C.Fogarty, "A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments"

[13] Alexandru Agapie, Bucharest, Alden H. Wright, Missoula,"Theoretical Analysis of Steady State Genetic Algorithms", University of Montana ScholarWorks Computer Science Faculty Publications.

★★★