A Novel Genetic Algorithm Approach to Routing in Data Network

Summiya.A.Pathan[1], Basavaraj N Bhusani[2] and Prof. Basavaraj S. Malapur[3]

1 2 3 Department of computer Science & Engineering , Basaveshwar Engineering College, Bagalkot, India.

ABSTRACT

Routing is the process of moving information from source to destination, and is one of the major issues found in computer network literature. For data networks, the complexity of routing increases due to the multiple constraints such as distance/cost and transmission success rate (TSR). These constraints are to be properly addressed for the reliability, robustness, manageability and functionality of the data networks. The biggest challenge in this kind of network is to find the optimal path between the common end points satisfying various requirements. Genetic algorithms (GA) are a class of search strategy based on the mechanism of biological evolution. GA is able to reduce search space and can also converge to a good solution of the problem efficiently. In this project an approach using genetic algorithm to routing in data network is explained. Appropriate chromosome structure, crossover operator and mutation operators are devised and tested on variety of network topologies. The results are found to be encouraging and satisfactory.

Key Words: Routing, Data Network, Genetic Algorithm.

1. INTRODUCTION

A data network is an electronic communications process that allows for the orderly transmission and receptive of data, such as letters, spreadsheets, and other types of documents. Routing is the process of selecting paths in a network along which to send network traffic so as to achieve high performance. Routing is complex in large networks because of the many potential intermediate destinations a packet might traverse before reaching its destination. Hence routing algorithm has to acquire, organize and distribute information about network states. In a large network like the Internet, information is routed from one communications network to the next until that information reaches its destination.

Routing is usually performed by a dedicated device called a router. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analyzing a routing table to determine the best path. 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. 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, delay etc which is aimed at having maximum performance. The shortest path problem is defined as that of finding a minimum-length (cost) path between a given pair of nodes. The Dijkstra algorithm is considered as the most efficient method for shortest path computation in data networks. But when the network is very big and the QoS parameters are more than one then it becomes inefficient since a lot of computations need to be repeated. Also it cannot be implemented in the permitted time.

GA works on the search space called population. Each element in the population is called as chromosome. GA begins with randomly selecting set of feasible solution from population. Each chromosome is a solution by itself. Each chromosome is evaluated for fitness and this fitness defines the quality of solution. GA uses adaptive heuristic search technique which finds the set of best solution from the population. New offsprings 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

ISSN 2229-5518
generation. This is because GA is based on the principle of Darwin theory of evolution, which states that the "survival is the best". 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. GA has been widely studied and experimented on many fields of engineering. GA provides alternative methods for solving problems which are difficult to solve using traditional methods.

2 -RELATED WORK

[1] In this paper, applied Hadoop-based system with Cellular Data Network flow upload and analyze components. It combines data uploading and CDR (Calling Detail Records) query with Map/Reduce framework and HDFS. It uses Map/Reduce framework to process the users' request, HDFS to manage the data flow files. Our system has the ability to upload and for the network manager to query the CDR more conveniently and reliable. More importantly, our system is applied to a real-world and has a good scalability, effectiveness and efficiency. [2] The requirements of interworking are presented and discussed here. Interworking different wireless network systems requires internetworking the different networks, interoperability off the different systems, as well as the additional mobility across different networks, involving security, accounting, and other technologies. These requirements may be used to guide future developments as well as to point out technical issues that need future exploration. [3] Author propose a combinatorial algorithm for solving the optimal shortest-path routing problem, which consists of finding the link-distance metric assignment to all links in a network that results in optimal shortest-path routing and minimizes the average delay of packets in the network. [4] A novel maximum-energy shortest path tree algorithm (MESPT), which is a two-phase algorithm, for data aggregation in wireless sensor networks was proposed. [5] This paper presents new efficient shortest path algorithm to solve single origin shortest path problems (SOSP problems) and multiple origins shortest path problems (MOSP problems) for a class of hierarchically clustered data networks with n nodes. [6] A distributed algorithm is presented that can be used to solve the single-destination shortest path(SDSP) problem or the all-pairs shortest path (APSP) problem for a class of clustered data networks. [7] This a model of data networks with delay, packet loss ratio and network delay and an optimization of them using genetic algorithm. [8] This paper presents an algorithm based on hierarchical hybrids parallel genetic algorithm for the shortest path routing. [9] This paper addresses a shortest path problem in network optimization, and proposes a model with constraints. In order to solve the problem, author present an improved genetic algorithm through optimal selection and crossover strategy of genetic algorithm, and explore the framework and key steps of improved genetic algorithm for solving shortest path problem. [10] In this investigated the possibility of using genetic algorithms to solve shortest path problems. A priority-based encoding method is proposed which can potentially represent all possible paths in a graph.

3-PROBLEM DEFINITION

Given the set of nodes, N aim is to find the shortest path between each source and destination. We aim to optimise the path between nodes in the data network using genetic algorithm. The data network under consideration is represented as G = (V, E), a connected graph with N nodes. The metric of optimization is cost and transmission success rate (TSR) between the nodes. The total cost is the sum of cost of individual edges. Success rate is the probability that the data will be transferred between the two nodes. The goal is to find the optimum path between source node Vsrc and destination Vdest, where Vsrc and Vdest belong to V. Our work is to apply efficient and robust genetic algorithm which will meet the above demands. We start out with a randomly selected first generation. Every string in this generation is evaluated according to its quality, and a fitness value is assigned. 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 are 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.

Algorithm 1: Routing in Data Network using Genetic Algorithm
Input: Number of nodes & edges
Output: Optimal Path

Step1: Get the inputs, no_edges, no_nodes, weights of each edge (weights), Transfer Success Rate of each edge (tsr).
Step2: Initialize mutation_probability and crossover_probability
Step3: Generate initial population of chromosomes based on no_edges and population_size.
Step4: Evaluate fitness of each chromosomes based on cost (weights of each edge in the chromosome) and tsr, Store in a fitness array.
Step5: for 1: max_number_of_iteration do

1. Select two parents (Parent1 & Parent2) for reproduction from population using Roulette wheel selection based on their fitness value.
2. Based on the crossover_probalibity, generate two offspring (Child1 & Child2) using crossover operator.
3. Based on the crossover_probability, apply mutation operator on Child1 & Child2.
4. Find the fitness value of Child1 & Child2.
5. Find the least fit chromosomes in the current population.
6. If the fitness of child1 & child2 are greater than the least fit of the population, And replace such chromosomes with child1, child2.
7. Find maximum fitness of populations
8. If there is no change in maximum fitness within termination_count then break out as there is no improvement.
9. Repeat step 4 till iteration is greater than max_no of iteration break out.

Step 6: End for
Step 7: Find chromosome with the highest fitness in population, genes of this chromosome give the solution of optimal path, between source and destination.
Step 8: Stop.

3.1 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. Cost C_i(V1->V2) and transmission success rate TSR_i(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,

$$F = \frac{\prod TSR \times path}{\sum C} \qquad (1)$$

Where,

$\prod$ TSR = product of transmission success rate
$\sum$ Cost = sum of total cost of the chromosome
Path = a large number like 1000 if path exists.

$$\text{Where path} = \begin{cases} 1000 & \text{if path exists} \\ 1 & \text{Otherwise} \end{cases}$$

4- IMPLEMENTATION

The genetic algorithm to find the optimal path is applied on the networks designed in matlab. Various static networks of varying size are created. In our implementation genetic algorithm is applied on topologies mentioned in table (1)

Table (1): Topologies

NODES	EDGES
5	5 , 7 , 9
10	14 , 25 , 35
15	20 , 30 , 50
20	30 , 60

After the number of nodes required and edges are created weights are assigned to all the edges which correspond to the cost of that particular edge. The total cost is the summation of all the costs of all the edges. One QoS considered is the transmission success rate (TSR) which is assigned to all the edges randomly ranging between the values of 90 to 100. The probability that data is transferred successfully increases with the increasing value of the TSR, 90 being the least and 100 being the highest.

4.1 Genetic representation

In the first step we initialize a population on which we will be applying the genetic algorithm. Each unit of the population is called a chromosome which is considered as a possible solution to our problem. The chromosomes in our implementation are edge based and contain binary sequence of 0's and 1's. An example of a chromosome implemented can be represented as shown in table (2) for a network with 9 edges

Table (2): Edge Representation

E1	E2	E3	E4	E5	E6	E7	E8	E9
1	1	0	1	0	0	1	1	0

The chromosome represents a path. This path above has edges E1, E2, E4, E7 and E8 which are represented as 1 and other edges which are not a part of the graph are represented as 0.

4.2 Initialization

ISSN 2229-5518

The initial population includes arrays where the number of elements is equal to the number of edges. The size of the initial population depends on the size of the network under consideration. The initial population for a certain run of the algorithm is shown in figure 4.1 (edges=9,size=25):

1	0	1	1	1	0	1	0	1
1	1	1	0	1	1	0	0	0
0	1	1	0	0	1	0	0	1
0	0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0
0	0	0	1	0	1	0	1	1
0	1	0	1	1	0	1	0	0
0	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1	0
1	1	1	1	0	1	0	0	1
0	0	1	1	1	0	0	0	1
1	1	0	0	0	1	1	1	1
0	0	1	1	1	0	0	1	1
0	1	1	0	1	0	1	0	1
0	0	0	0	0	1	1	1	1
1	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1
1	1	0	0	0	1	0	0	0
1	1	1	0	0	1	1	1	1
0	0	1	1	0	1	0	0	0
1	1	1	1	1	1	0	1	0

Fig 4.1 Initial Population

4.3 Fitness function

The fitness function is given in equation 1. The fitness of each chromosome is calculated based on the cost and the transmission success rate so that the chromosomes can be selected based on their fitness. A function is written to check if the chromosome holds any path.

4.4 Selection

The next step is the selection of two chromosomes from the initial population which will be called parent chromosomes. Each chromosome has a fitness function assigned based on the formula described above. In our implementation we have chosen the 'Roulette wheel selection' technique which we narrowed down as the best selection technique. In roulette wheel, individuals are selected with a probability that is directly proportional to their fitness values. The probabilities of selecting a parent can be seen as spinning a roulette wheel with the size of the segment for each parent being proportional to its fitness. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The circumference of the roulette wheel is the sum of all fitness values of the individuals. After the selection of both the parent chromosomes they 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=cumsum(fit)
Step2: Generate two 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

4.5 Crossover

Crossover is applied to both the selected chromosomes. There are various techniques for crossover. We have selected the multiple point crossover in our implementation of the genetic algorithm. The crossover probability is decided after experimentation and we have selected a crossover of 0.95. Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms. The newly generated children are passed on to the next step. The figure 4.2 shows two point crossover.

Algorithm 3: Two Point crossover Operation
Input: Parent1, Parent2 (two chromosomes) and no_edges
Output: child1, child2 (new two Chromosomes)

Step1: Generate two random number using no_edges as co_index1, co_index2
Step2: if co_index1>co_index2 then swap their values
Step3: Initially all genes of parent1 are copied till co_index1, beyond co_index1 genes from parent2 are copied till co_index2, then rest of genes from parent 1 is copied.This gives child1.
Step4: Initially all genes of parent2 are copied till co_index1, beyond co_index1 genes from parent1 are copied till co_index2, then rest of genes from parent2 is copied. This gives child2.
Step5: Stop

ISSN 2229-5518

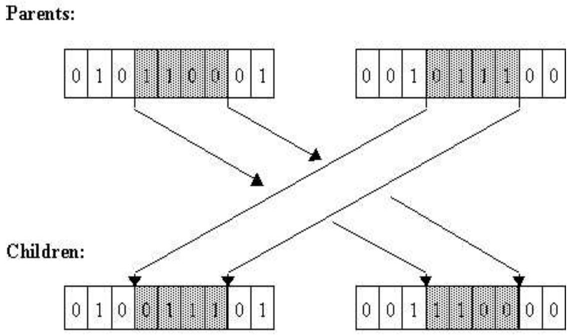

Fig 4.2 Two Point Crossover

4.6 Mutation

Mutation process brings in more randomness in the genetic algorithm. We have found the multipoint (three point) mutation to be best suited for our implementation. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. We have implemented the mutation using bit flipping. This mutation operator takes the chosen genome and inverts the bits. (i,e. if the genome bit is 1,it is changed to 0 and vice versa). Mutation occurs during evolution according to a user-definable mutation probability. The probability is usually low and we have defined a probability of 0.5.

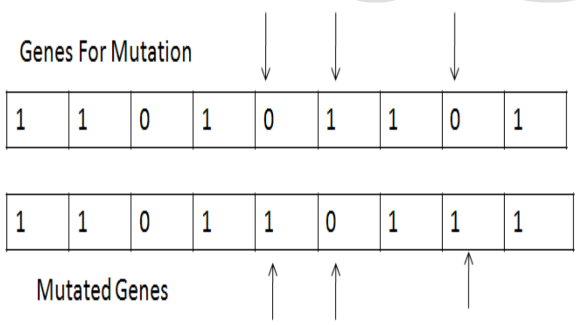

Fig 4.3 Three Point Mutation

In the figure 4.3 shows mutation technique three random points is generated and the bit at that position is flipped. 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.

Algorithm 4: Three-point Mutation Operation
Input: child, no_edges, mutation probability
Output: mutated_child

Step 1: for 1: 3 (for 3 point mutation)

1. based on no_edges in child, generate m_index random point on child for mutation.
2. If child(m_index)==1 then flip bit to 0[i.e. child(m_index)=0] Else child(m_index) = 1
3. Repeat step1

Step 2: return child

4.7 Termination Criteria

Maximum generations, No change in population fitness and stall generation are considered as algorithm stopping condition. 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.

5-EXPERIMENTS AND RESULTS:

The user is allowed to select the network on which the genetic algorithm will be implemented. Hence a GUI is created where the provision is provided for the user to select the number of nodes, the number of edges, the source node and also the destination node.After the execution of the program GUI as shown in Figure 5.1 also displays the time taken by the program to arrive at the solution, the fitness of the solution chromosome and also the solution chromosome itself. The path obtained is highlighted in the biograph as shown in figure 5.2

The graphs as in figure 5.1 to 5.7 shows the percentage of optimal paths obtained for networks with 5, 10, 15 and 20 nodes and also the time complexity with respect to all the topologies

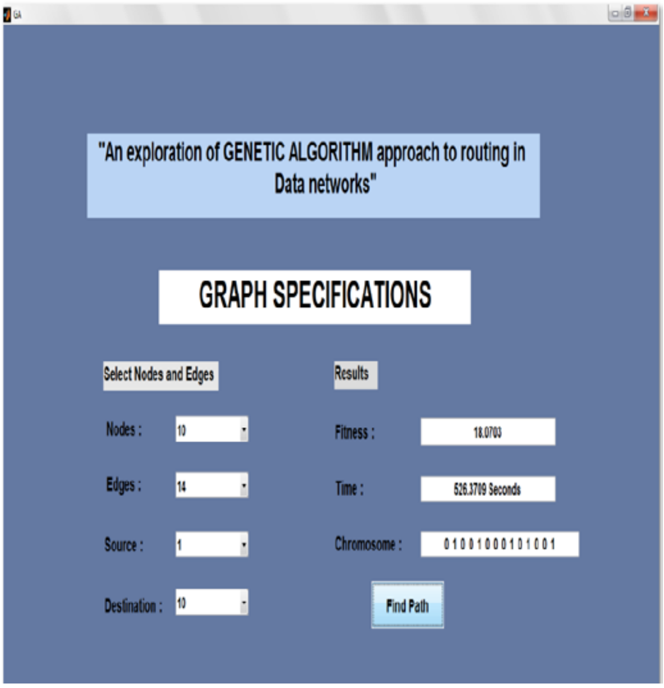

Fig 5.1 GUI with Inputs & Outputs

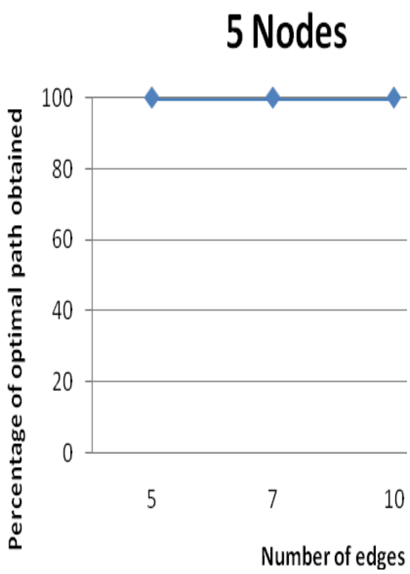

Fig 5.3 Optimal path for 5 Nodes

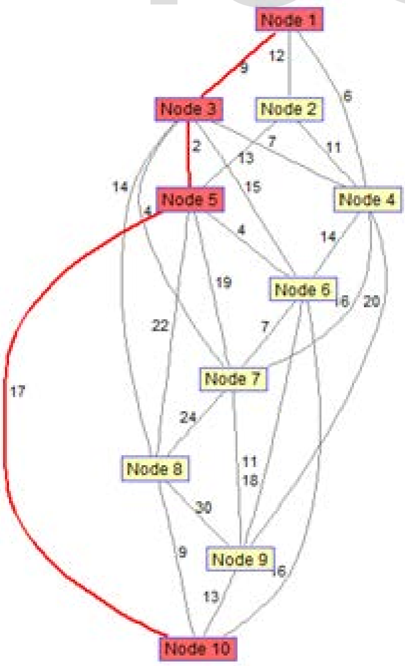

Fig 5.2 Path obtained

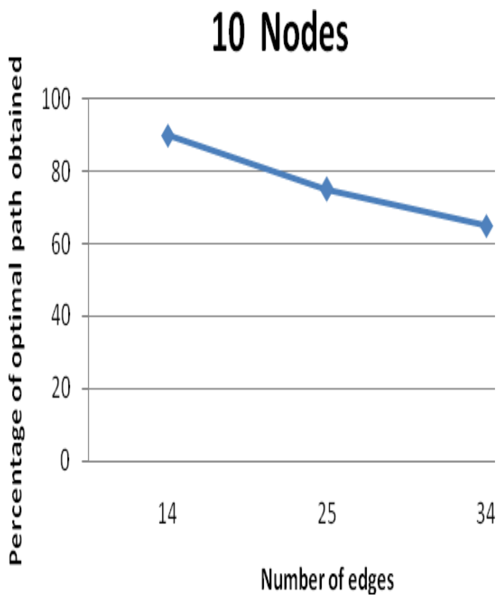

Fig 5.4 Optimal path for 10 Nodes

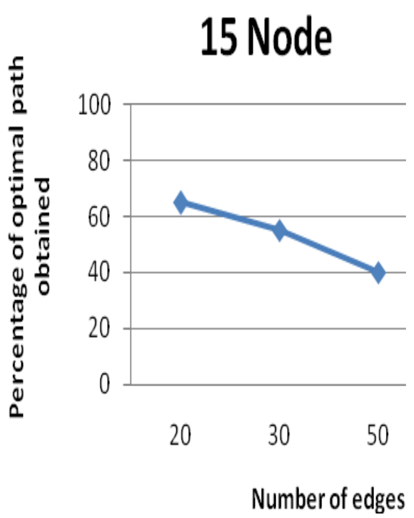

Fig 5.7 Optimal path for 15 Nodes

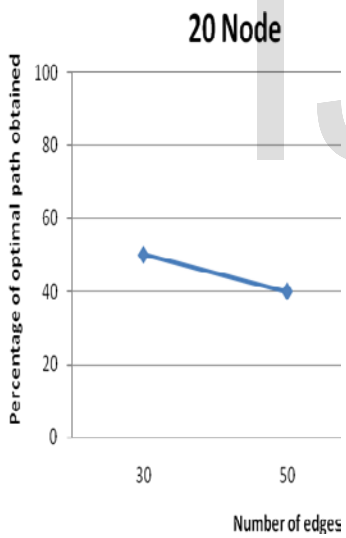

Fig 5.6 Optimal path for 20 Nodes

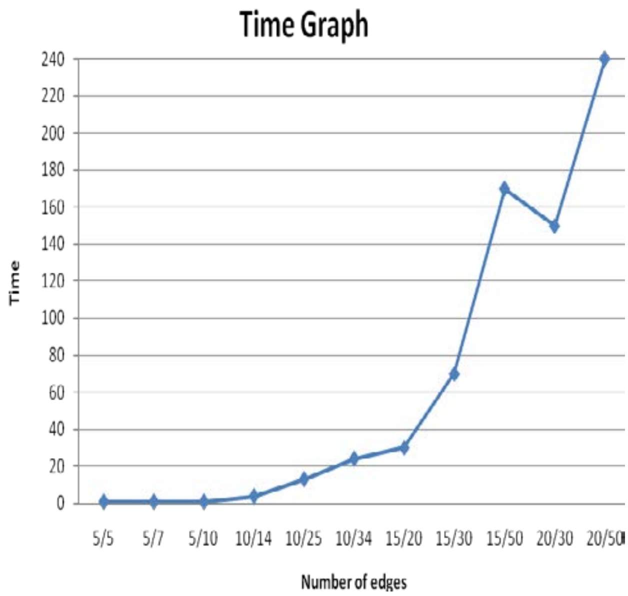

Fig 5.8 Time graph

7 – CONCLUSION AND FUTURE WORK

In the present work Genetic algorithms have been used for finding optimal paths between given source and destination in a network. A novel edge based chromosome structure has been devised to represent the possible solutions of the optimal path in the data 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 transmission success rate (TSR) has been satisfactory. More complex crossover and mutation operators can be explored to further enhance the performance of genetic algorithm in finding the optimal paths in data network.

5 - THE REFERENCES SECTION

[1] Toward scalable and high performance data processing for Cellular Data Network", Zhengxiang Ke; Jun Liu; Dazhong He; Qing Yan Communication Technology (ICCT), 2013 15th IEEE International Conference on Year: 2013.

[2]"Requirements of interworking wirelessLAN and PLMN wireless datanetwork systems" , Chan

H.A ,AFRICON, 2004. 7th AFRICON Conference in Africa Year: 2004, Volume: 1.

[3]“Optimal routing in shortest-path data networks”, Ramakrishnan, K.G.; Rodrigues, Manoel A. Bell Labs Technical Journal Year: 2001, Volume: 6, Issue: 1.

[4]”Maximum-Energy Shortest Path Tree for Data Aggregation in Wireless Sensor Networks”, Yunyi Zhang; Yi Wang; Deyun Zhang; Chongzheng Huang. Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on Year: 2007

[5] ”A new distributed shortest path algorithm for hierarchically clustered datanetworks”, Huang, G.; Zhu, S., American Control Conference, Proceedings of the 1995Year: 1995, Volume: 3 Pages: 2031 – 2035
vol.3, DOI: 10.1109/ACC.1995.531249

[6] “A fast distributed shortest path algorithm for a class of hierarchically clustered data networks” , Antonio, John K.; Huang, G.M.; Tsai, W.K. Computers, IEEE transactions on Year: 1992, Volume: 41, Issue: 6.

[7] “Optimization of multiple objectives and topological design of data networking using genetic algorithm” , Dasgupta, M.; Biswas, G.P.; Bhar, C.Recent Advances in Information Technology (RAIT), 2012 1st International Conference on Year: 2012.

[8]“Efficient hierarchical hybrids parallel gentic algorithm for shortest pathrouting”, Neeraj; Kumar, A. Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference - Year: 2014.

[9] “A Study on the Shortest Path Problem Based on Improved GeneticAlgorithm” , Zongyan Xu; Haihua Li; Ye Guan. Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on Year: 2012.

[10]“Genetic algorithms for solving shortest path problems” , Gen, M.; Runwei Cheng; Dingwei Wang. Evolutionary Computation, 1997., IEEE International Conference on Year: 1997.