Volume 2, Issue 2, February 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Implementation of Shortest Path in Packet Switching Network Using Genetic Algorithm

Gajendra Singh Chandel,Ravindra Gupta
(Computer Sc. &engineerin),RGTU,Bhopal

Arvinda Kushwaha
M.Tech.(Scholar,SSSIST,Sehore)

Abstract- The problem of finding the optimal path between two nodes is a well known problem in network analysis. Optimal routing has been widely studied for interconnection networks this dissertation work considers the problem of finding the optimal path. A Genetic algorithm based strategy is proposed and the algorithm has been developed to find the Optimal Path. This paper work presents a genetic algorithmic approach to the shortest path routing problem. Variable-length chromosomes and their genes have been used for encoding the problem. The crossover operation exchanges partial chromosomes at positionally independent crossing sites and the mutation operation maintains the genetic diversity of the population. The proposed algorithm can cure all the infeasible chromosomes with a repair function. Crossover and mutation together provide a search capability that results in improved quality of solution and enhanced rate of convergence. Even though shortest path routing algorithms are already well established, there are researchers who are trying to find alternative methods to find shortest paths through a network. One such alternative is to use genetic algorithm.

Keywords: Genetic algorithm, crossover, selection, Mutation.

I. INTRODUCTION

For the information-oriented society in the early years of 21st century, communication by packet flow in large-scale computer networks becomes much more important in our daily life than ever before. The problem of finding the shortest path between two nodes is a well-known problem in network analysis. Shortest path algorithms have been a subject of extensive research, resulting in a number of algorithms for various conditions and constrain [1-3].Adaptive routing algorithms [4-8], which can select the route of pack ets dynamically, have been widely studied to make the best use of bandwidth in interconnection networks of massively parallel computers and system area networks (SANs). Most of real SANs for PC clusters [9-10] have not employed adaptive routing. This is because adaptive routing introduces new problems in the networks. First, it does not guarantee in-order packet delivery in which some message passing libraries require. Second, switch complexity may be increased, because they compute alternative output channels and select one of them introducing selection logic. In the context of SANs, some works have also proposed simple methods to support adaptive routing in Infinite Band switches [11]. In a packet switching network, communication between two hosts generally takes place in the following manner: the transmitting host delivers to a node a block of data, called a packet, which are addresses to the destination host. The objective of a routing strategy is essentially to minimize the mean delay of the packets in a network, subject to some reliability or capacity constraints [12-16]

Routing is one of the most important issues that have a significant impact on the network's performance [17], [18]. An ideal routing algorithm should strive to find an optimum path for packet transmission within a specified time so as to satisfy the quality of Service (QoS) [18]-[20]. There are several search algorithms for the shortest path (SP) problem: the breadth-first search algorithm, the Dijkstra algorithm and the Bellman-Ford algorithm, to name a few [17]. Since these algorithms can solve SP problems in polynomial time, they will be effective in fixed infrastructure wireless or wired networks. But, they exhibit unacceptably high computational complexity for real-time communications involving rapidly changing network topologies [19], [20]. In most of the current packet-switching networks, some form of SP computation is employed by routing algorithms in the network layer [18], [20]. Specifically, the network links are weighted, the weights reflecting the link transmission capacity, the congestion of networks and the estimated transmission status such as the queuing delay of head-of-line (HOL) packet or the link failure. The SP problem can be formulated as one of finding a minimal cost path that contains the designated source and desti-nation nodes. In other words, the SP routing problem involves a classical combinatorial optimization problem arising in many designs and planning contexts [18]-[23]. Since neural networks (NNs) [18]-[20] and genetic algorithms (GAs) (and other evo-lutionary algorithms) [21] [24] promise solutions to such complicated problems, they have been used successfully in various practical.

II. PROPOSED APPROACH

A. CHROMOSOME REPRESENTATION

A network can be thought of interconnection of nodes where distance between two nodes is represented by $\{e_i\}$ various paths of a network are shown in fig-1:

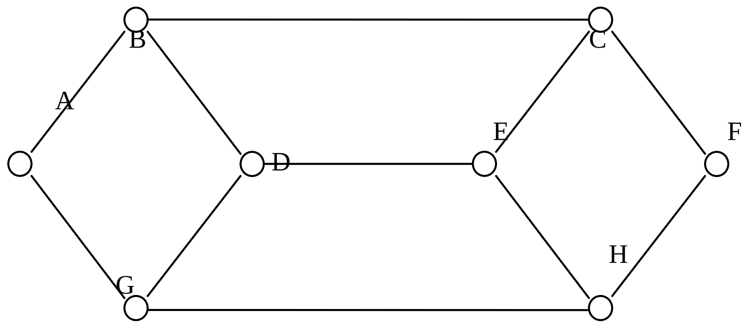


Figure 1: Chromosome Representation

Let, e_1 = AB = 2, e_2 = BC = 7, e_3 = CD = 3, e_4 = GH = 4, e_5 = AG = 6, e_6 = EF = 2, e_7 = BE = 2

e_8 = CF = 3, e9 = DH = 2, e_{10} = FH = 2, e_{11} = GE = 1

One of the combinations of edges can be:

$\{e_5$ (6), e_2 (7), e_4 (4), e_6 (2)$\}$

Each edge is represented by four bit string therefore the above combination of edges can be represented by following strings: 0110 0111 0100 0010

And, $\{e_1$ (2), e_{11} (1), e_{10} (2), e_3 (3)$\}$ these combinations can be represented by following

Strings: 0010 0001 0010 0011

Algorithm: 1(Algorithm for identifying path from source to destination)

```
path (source, destination,*source node)
{Identify source and destination in given graph
 push(sourse);
 while(top!=NULL)
{
 tempnode=pop(&top);
   check whether the node is visited or not
    if not visisted set flag=1
        {
while(tempnode->info!=destination && temparc!=NULL)
          {
push(tempnode);
              p[i][j]=temparc;
              tempnode=temparc->adj;
            temparc=tnode->edg;
             j=j+1;
            if(tempnode->info==destination)
            {
p[i][j]=NULL;
               i=i+1;
 }
            End if
}
  End while
}
End while
}
End
}
```

Algorithm-2(Algorithm for random path from source to destination)

```
 randompath(*source_node)
{
store all edges in an array
 intilize len=0
 repeat step 3,4,5 while(len<length_of_array)
 use mode function index=(3*random_num)%4 to select an edge from array
 assign the selected edge to two diminsion array
 len=len+1
 End
}
```

Algorithm-3(Algorithm for creating a node)

```
CreateNode(**source,*temp,*tc,c,data)
 Assign a block of size from the memory heap in to temp
 temp->next=NULL
 temp->edg=NULL
 temp->info=data
{
 If(*source==NULL)
{
Then *source=temp
 else
 tc=*source
{while(tc->next!=NULL)
{ tc=tc->next
 End While
}
 tc->next=temp
}
End Else
}
End If
}
```

Algorithm-4(Algorithm for Connect Nodes which are present in the whole graph)

```
ConnectNodes(*source,*temp,*back,*track,*tc,*ptr)
 temp=source
 back=temp
while(temp->info!=p)
{temp=temp->next
 back=temp
 End While
} temp=source
 track=temp
 while(temp->info!=c)
{ temp=temp->next
 track=temp
 End While}
Print : Enter the Weight of edge
 Assign a block of size from memory heap to the tc
 tc->w=wit
 tc->adj=track
 tc->nextptr=NULL
 tc->flag=0
 If(back->edg==NULL)
{
Then      back->edg=tc
 End If
} Else
{ ptr=back->edg
 while(ptr->nextptr!=NULL)
{ ptr=ptr->nextptr;
 End While}
 ptr->nextptr=tc;
 End Else
}}
```

B. Crossover

We applied two points crossover on initial population Suppose four randomly generated individuals are:

e_2 (7) e_4 (4) e_3 (3) e_5 (6) e_8 (3) sum of edges→23

e_1 (2) e_{11} (1) e_5 (6) e_2 (7) e_6 (2) sum of edges→18

e_4 (4) e_7 (2) e_3 (3) e_{11} (1) e_6 (2) sum of edges→12

e_1 (2) e_3 (3) e_6 (2) e_2 (7) e_7 (2) sum of edges→16

Before Crossover,

0111 010 0	0011 011 0	0011 sum of edges→23
0010 000 1	0110 011 1	0010 sum of edges→18
0100 001 0	0011 000 1	0010 sum of edges→12
0010 001 1	0010 011 1	0010 sum of edges→16

(Applying two point Crossover)

After Crossover,

```
  (7)  (5)  (6)  (7)  (3)
0111 0101 0110 0111 0011 sum of edges→28
  (2)  (0)  (3)  (6)  (2)
0010 0000 0011 0110 0010 sum of edges→13
  (4)  (3)  (2)  (7)  (2)
0100 0011 0010 0111 0010 sum of edges→18
  (2)  (2)  (3)  (1)  (2)
0010 0010 0011 0001 0010 sum of edges→27
```

Algorithm-5(Algorithm for Crossover operation)

```
Crossover (data [ ][ ])
{      For  i = 0 to 3
  {  For  j= 0 to 15
    {   If (j==4)
      {   Data [i][j] =! data[i][j]
      }
        If (j==11)
      {    Data [i][j]=!data[i][j]
      }
    }
  }
}
```

C. MUTATION

Mutation of a string is implemented through a very simple protocol. We will replace first four bits with source and last four bits with destination.

Since in our network source node is A and destination node is D, therefore we replace first four bits by 0010 and last four bits by 0010

```
  (2)  (5)  (6)  (7)  (2)
0010 0101 0110 0111 0010 sum of edges→22
  (2)  (0)  (3)  (6)  (2)
0010 0000 0011 0110 0010 sum of edges→13
  (2)  (3)  (2)  (7)  (2)
0010 0011 0010 0111 0010 sum of edges→16
  (2)  (2)  (3)  (1)  (2)
0010 0010 0011 0001 0010 sum of edges→10
```

Algorithm-6(Algorithm for Performing Mutation)

```
Mutation(data[i][j],position,counter,limit)
 Initialize position = 0, i = 1, j = 0 and counter = 1
  Start while( i < limt)
{
 pos=pos+4
  Increment  i=i+1
  End While}
Assign  i=0
  Start  while( data[i][0] != -1 )
{
 Assign counter=1
  Assign   j=0
 Start  while( j < 4 )
{
If( counter == 3 ) Then
 data[i][j] = 1
 End If}
 Else data[i][j]=0
}
End Else
}
Increment counter  = counter + 1
 Increment j=j+1
 End while}
Assign j=pos
 Assign counter=1
 Start while( j < ( pos + 4 ) )
{
 If( counter == 3 )
{
 Assign data[i][j]=1
 End If
}
Else data[i][j]=0}
End Else}
Increment counter = counter + 1
 Increment j = j + 1
 End While}
Increment i = i + 1
 Printf "After Mutation " Call displaybin(data)
 }
```

D. SELECTION

In the above problem our fitness function is = min ($\sum e_i$), with continuity

After mutation, we have minimum path length from source node to destination node is:10 (min path length from A->D) (0010 0010 0011 0001 0010)

Path-> AB BE FC EG FH

We Observed that it is not a continuous path, therefore we have to select minimum path with continuity

After iterations we get, minimum path length 10 with continuity 10 (min path length after selection) (0010 0010 0010 0010 0010)

Path-> AB BE EF FH HD

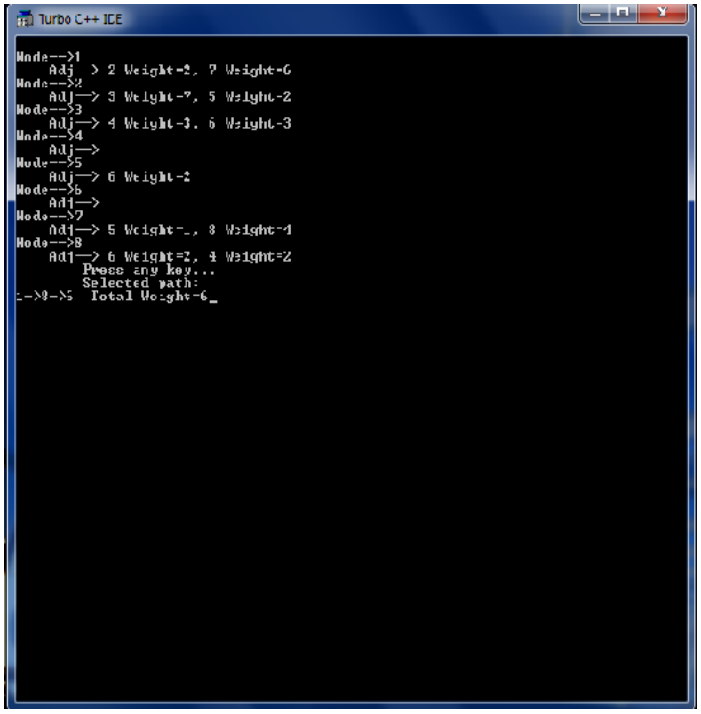
This is the most optimal path.

Algorithm-7(Algorithm for Main Function)

We need a pointer variable source of node structure type and Source and Destination in Graph

```
Main(source,destination,struct node *source)
{
 Initialize source = Null
  Call createNode(&source)
  If(source != Null) Then
{
 Call connectNodes(source)
 Input source and destination
 Call randomPath(source)
 Call path(source,destination,Source)
}
}
```

E. RESULT



```
Node-->1
    Adj  > 2 Weight-2, 7 Weight-6
Node-->2
    Adj--> 3 Weight-7, 5 Weight-2
Node-->3
    Adj--> 4 Weight-3, 6 Weight-3
Node-->4
    Adj-->
Node-->5
    Adj--> 6 Weight-2
Node-->6
    Adj-->
Node-->7
    Adj--> 5 Weight-_, 8 Weight-1
Node-->8
    Adj--> 6 Weight=2, 4 Weight=2
        Press any key...
        Selected path:
[illegible]->8->5  Total Weight-6_
```

III. CONCLUSIONS

Genetic Algorithm provides a useful problem solving technique. The proposed approach shows, how GA can be used to solve a very general version of shortest path problem. A GA encoding along with the genetic operators is defined. The performance of the algorithm is better than previous work. For the practical implementation of the proposed work Coding of the algorithm is also included. This technique can be very useful to evaluate the shortest path in various networks. This research work presented a genetic algorithm for solving the SP routing problem. The crossover and the mutation operations work on variable-length chromosomes. The crossover is simple and independent of the location of crossing site. Consequently, the algorithm can search the solution space in a very effective manner. The mutation introduces, in part, a new alternative route. In essence, it maintains the diversity of population thereby avoiding local traps. A treatment for infeasible solutions (chromosomes) has also been investigated without unduly com-promising on computational requirements. The proposed algorithm can search the solution space effectively and speedily compared with other algorithms. Proposed algorithm is implemented in C language that can search an optimal path in optimum time.

REFERENCES

[1]. E.W. Dijkstra, "A note on two papers in connection with graphs," Numeriske Mathematics 1 pp.269-271 1959.

[2]. D.Eppstein, "Finding the k shortest paths," SIAM journal on Computing 28(2) pp.653-674 1998.

[3]. R.W Floyd, "Algorithm97: Shortest paths, "Communications of the ACM 5 pp.345-357 1962.

[4]. A.A. Chien, J.H. Kim, "Planar-adaptive routing: low-cost adaptive networks for multiprocessors,"J. ACM 42(1) pp. 91-123 1995.

[5]. W.J.Dally,H Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," IEEE Trans. Parallel Distributed. Systems 4(4) pp.466-475 1993.

[6]. J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks,"IEEE Trans.Parallel Distrib. Systems 6(10) pp.1055-1067 1995.

[7]. M. Koibuchi , A . Funahashi , A . Jouraku, H . Amano, "L-turn routing:An adaptive routing in irregular networks," Proc. International Conference on Parallel Processing," pp.374-383 Sep 2001.

[8]. F. Silla, J. Duato, "High –performance routing in networks of workstations with irregular Topology," IEEE Trans. Parallel Distrib. Systems 11(7) pp. 699-719 2000.

[9]. N. J. Boden, et.al, "Myrinet:a gigabit-per-second local area network," IEEE Micro. 15(1) pp. 29-35 1995.

[10]. T. Kudoh, S. Nishimura, J. Yamamoto, H. Nishi,O. Tatebe,H.Amano, "RHINET:A network for high performance parallel computting using locally distributed computing," Proc IWIA pp. 69-73 Nov 1999.

[11]. J.C.Martinez,J.Flich,A.Robles,P.Lopez,J. Duato, "Supporting adaptive routing in IBA switches," Systems Architect 49 pp. 441-449 2004.

[12]. Baransel C,Dobosiewicz W, Gburzynski p, "Routing in multihop packet switching networks:Gb/s challenges,"IEEE Network 9(3) pp.38-61 1995.

[13]. Beaubrun R, Pierre S, "Routing algorithm for distributed communication networks," Proc 22nd IEEE Conference on Computer Networks,LCN 97 pp. 99-105 Nov 1997.

[14] Kershenbaum A, Kermani P,Grover GA, "MENTOR:an algorithm for mesh network topotogical optimization and routing," IEEE Trans Comm pp.503-513 1991.

[15] Khasnabish B, "A new method for evaluating packet routing policies in supra-high-speed metropolitan (or wide) area networks,"Comp Networks ISDN Syst pp.195-2161993.

[16]. Suk-Gwon C, "Fair integration of routing and flow control in communication networks,"IEEE Trans Commun 40(4) pp. 821-34 1992.

[17] W. Stalling, *High-Speed Networks: TCP/IP and ATM Design Princi-ples.* Englewood Cliffs, NJ: Prentice-Hall, 1998.

[18] M. K. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Trans. Neural Networks,* vol. 4, pp. 941-954, Nov. 1993.

[19] D. C. Park and S. E. Choi, "A neural network based multi-destination routing algorithm for communication network," in *Proc. Joint Conf. Neural Networks,* 1998, pp. 1673-1678.

[20] C. W. Ahn, R. S. Ramakrishna, C. G. Rang, and I. C. Choi, "Shortest path routing algorithm using hopfield neural network," *Electron. Lett.,* vol. 37, no. 19, pp. 1176-1178, Sept. 2001.

[21] M. Munemoto, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics,* 1998, pp. 2774-2779.

[22] J. Inagaki, M. Haseyama, and H. Kitajima, "A genetic algorithm for de-termining multiple routes and its applications," in *Proc. IEEE Int. Symp. Circuits and Systems, 1999*, pp. 137-140.

[23] Y. Leung, G. Li, and Z. B. Xu, "A genetic algorithm for the multiple destination routing problems," *IEEE Trans. Evol. Comput,* vol. 2, pp. 150-161, Nov. 1998.

[24] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann, 1989, pp. 2-9.

[25]. Courtois PJ,Semal P, "flow assignment algorithm based on the flow deviation method,"Proc of the ICCC pp.77-83 1980.

[26]. Gavish B, "Topological design of computer networks-the overall design problem,"Eur J Oper Res 58 pp. 149-72 1992.

[27]. Neumann I, "System A. For priority routing and capacity assignment in packet switched networks,"Ann Oper Res 36 pp. 225-46 1992.

[28]. Lee S,Chang S, "Neural network for routing of communication networks with unreliable components," IEEE Trans Neural Networks 4(5) pp. 854-63 1993.

[29]. Mehmet M,Kamoun F, "Neural networks for shortest path computation and routing in computer networks," IEEE Trans Neural networks 4(6) pp.941-54 1993.

[30]. Moopenn A, Thakoor AP, Duong T, "A neural network for Euclidean distance minimization,"Proc IEEE Int Conf Neural Networks 2 349-56 1988.

[31]. Internetworking Technology Handbook: Internet Protocols (IP), Cisco Sytems, Inc., 2002.

[32]. Chyzy Mariusz,Kosinski Witold, "Evolutionary Algorithm for State Assignment of Finite State Machines,"Proc IEEE of the Euromicro Symposium on Digital System Design(DSD'02) pp. 7695-99 2002.

[33] "Evolutionary Computation", IEEE Transactions onVolume 6, Issue 6, Dec 2002 Page(s): 566 – 579 Digital Object Identifier 10.1109/TEVC.2002.804323

[34] "Faster Genetic Algorithm for Network Path", Yinzhen Li1 Ruichun He1 Yaohuang Guo2 ,The Sixth International Symposium on Operations Research and Its Applications Pp 382-389 2000.

[35] G. Tufte and P. C. Haddow, "Prototyping a GA pipeline for complete hardware evolution," in *Proc. 1st NASA/DoD Workshop on Evolvable Hardware,* 1999, pp. 76-84.

[36] X. Hue, "Genetic algorithms for optimization: Background and appli-cations," Edinburgh Parallel Computing Centre, Univ. Edinburgh, Edin-burgh, Scotland, Ver 1.0, Feb. 1997

[37] D. E. Goldberg, K. Deb, andj. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Syst.*, vol. 6, pp. 333-362, 1992.

[38] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller, "The Gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evol. Comput.,* vol. 7, no. 3, pp. 231-253, 1999.

[39] "Multiple processors and the computational resource is distributed among these processors" Cantii-Paz, 2000a; Cantii-Paz, 1997.

[40] "Parallelization approaches 'simple master-slave'" Bethke, 1976; Grefenstette, 1981. [41] "Parallelization approaches 'coarse-grained'" Grosso, 1985; Pettey, Leuze *k*, Grefenstette, 1987; Tanese, 1989.