International Journal of Computer Theory and Engineering, Vol. 2, No. 3, June, 2010
1793-8201

A Novel Genetic Algorithm approach for Network Design with Robust Fitness Function

Anand Kumar and Dr. N. N. Jani, *Member IAENG*

***Abstract*—This paper presents a novel genetic algorithm approach for network design with a robust fitness function which finds the best least distance network for any number of nodes. A network design problem for this paper falls under the network topology category which is a minimum spanning tree. Since many researchers have tried to solve this problem for small to mid size, we have explored the use of genetic algorithm with modification but without changing the nature of genetic algorithm. A strong fitness function is developed here for solving this network optimization problem which not only reduces the number of generation rather produces the best result and follow the concept of "Survival of the fittest". Fitness function is the backbone of the concept of genetic algorithm which directly affects the performance; so one of the main focus of this paper is fitness function. Since this is NP problem and traditional heuristics have had only limited success in solving small to mid size problems, in this paper we have tried to show that genetic algorithm is an alternative solution for this NP problem where conventional deterministic methods are not able to provide the optimal solution.**

***Index Terms*—Genetic Algorithm, Network design, Minimum spanning tree.**

I. Introduction

A genetic algorithm approach to design the network is one of the ultimate solutions because traditional heuristics has the limited success. Researchers in operation research have examined this problem under the broad category of 'minimum cost flow problem' [1]. A simple GA approach is applied by many researchers [2],[3],[4] but we have modified the genetic algorithm approach because of its uncertain nature and found the good result. Genetic Algorithms are being used extensively in optimization problem as an alternative to traditional heuristics. It is an appealing idea that the natural concepts of evolution may be borrowed for use as a computational optimization technique, which is based on the principle "Survival of the fittest" given by "Darwin". In this paper various size of network (10 node to 100 node) is considered which is not considered by other researchers. A strong robust fitness is developed and applied which is working in all the size of network. The fitness function consists of many functions which is required according to the nature of problem. We have tried to show that the impact of robust fitness function and the little variation in genetic algorithm approach is very effective.

Manuscript received October 24, 2009. Anand kumar is with AMC Engineering College, Bangalore INDIA (email : kumaranandkumar@gmail.com).

Dr N.N. Jani is with Kadi Sarva Vishwa yidyalya, Gandhinagar INDIA.

(e-mail: drnnjanicsd@gmail.com).

A. Network Design

In this paper, network design is considered as a network topology which is a spanning tree, consists of various nodes and these nodes are represented as vertex. A tree is a connected graph containing no cycles. A tree of a general undirected graph G = (V,E) with a node (or vertex) set V and edge set E is a connected subgraph T = (V',E') containing no cycles with (n-1) edges where n is total no of node.

In this study undirected networks are considered with the weight (distance) associated with each node. For a given connected, undirected graph G with n nodes, a minimum spanning tree T is a sub graph of a G that connects all of G's nodes and contains no cycles [5]. When every edge (i, j) is associated with a distance cij , a minimum spanning tree is a spanning tree of the smallest possible total edge cost

$$C = \sum_{\text{Where } (i,j). \in T} c_{ij} \quad (1)$$

B. Genetic Algorithm

Genetic algorithms (GA) is a powerful, robust search and optimization tool, which work on the natural concept of evolution, based on natural genetics and natural selection..

The simple genetic algorithm has following steps:

```
Simple Genetic Algorithm
    {
        initialize population;
        evaluate population;
            While TerminationCriteriaNotSatisfied
              {
                Select parents for reproduction;
                Perform recombination and
mutation;
                Evaluate population;
              }
        }
```

This is traditional approach of Genetic algorithm, where evaluation is performed after the initialization of population. The next step is the selection of child population on the basis of fitness and application of genetic operators. Genetic algorithm is very uncertain also. It may be possible that the initial parent population has the best result and after so many generation we could not find the better result as it has been found before. At the same time application of genetic

operator means to generate new set of child population. In simple genetic algorithm approach evaluation is done after crossover and mutation but again it may be possible that the required result can be found after crossover only. After this observation we have modified the simple genetic algorithm with evaluation after each new set of population. At the same time we are keeping the best result found previously and it is replaced with the next best result if it is better than the previous result. The modified genetic algorithm has the following steps

C. Work flow of GA

1. Initialisation of parent population.
2. Evaluation
   a) Self loop check
   b) Isolated node or edge check
   c) Cycle check
   d) Store the best result
3. Selection of child population
4. Apply Crossover/ Recombination
5. Evaluation
6. Replace the result if it is better than previously stored.
7. Apply Mutation
8. Evaluation
9. Replace the result if it is better than previously stored.
10. Go to step 3 until termination criteria satisfies

Network Design Problem Presentation

The Network design problem is considered as a unidirectional graph and represented with the help of adjacency matrix. The adjacency matrix stores the distance between one node to another node and contains 0 distances in the case of same node to same node.

TABLE -1. ADJACENCY MATRIX OF THE GRAPH

0	13	40	55	16	44	68	27	57
				54				
13	0	37	38	12	42	45	40	
			21	11				
40	37	0	40	45	39	51	4	
			67	39				
55	38	40	0	51	52	71	42	
			36	18				
16	12	45	51	0	15	31	39	
			23	52				
44	42	39	52	15	0	44	20	
			43	6				
68	45	51	71	31	44	0	39	
			57	40				
27	40	4	42	39	20	39	0	
			39	27				
57	21	67	36	23	43	57	39	
			0	13				
54	11	39	18	52	6	40	27	
			13	0				

In table-1, the first row stores the distance from node 1 to 1,2,3……10. Similarly second row stores the distance from 1 to 10 and so on.

II. GENETIC ALGORITHM APPROACH TO SOLVE THE PROBLEM

Parent population in the form of chromosome is generated randomly according to the size of network. Number of gene in a chromosome is equal to number of node in a network. The total number of chromosome may vary and it is based on user input. Here a chromosome is generated for a 10 node network. The association between nodes is considered between positions to position.

node	1	2	3	4	5	6	7	8	9	10

chromosome	2	10	4	9	6	7	5	9	8	3

The logic behind association is that, the node [1] is connected with node 2; node [2] is connected with 10 and so on.

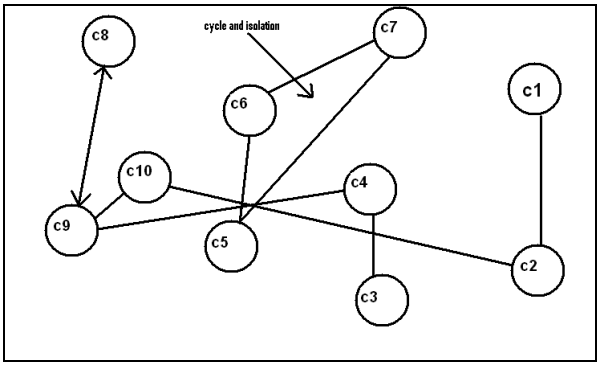

Figure 1

From fig-1 it is clear that this is not a spanning tree because of the isolated circle. Similarly with some other randomly generated chromosome, some other problems have been observed.

node	1	2	3	4	5	6	7	8	9	10

chromosome	8	6	3	1	8	9	4	2	7	1

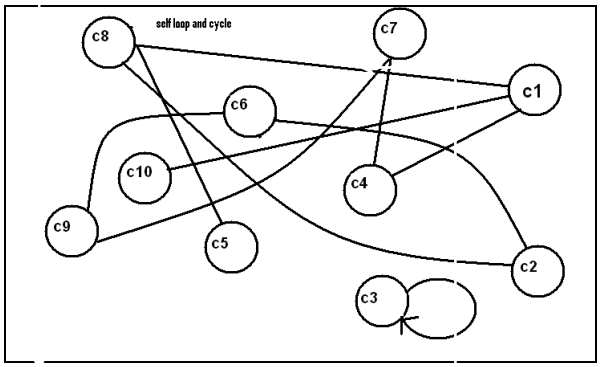

Figure 2

In Figure 2. The chromosome is illegal due to self loop and cycle.

node	1	2	3	4	5	6	7	8	9	10

chromosome	7	6	4	7	10	7	1	3	2	5

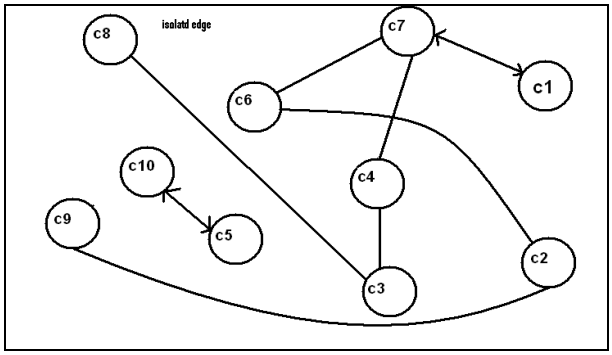

Figure 3

In Figure 3. The chromosome is illegal due to isolated edge.

node	1	2	3	4	5	6	7	8	9	10

chromosome	2	1	10	3	6	7	6	7	4	3

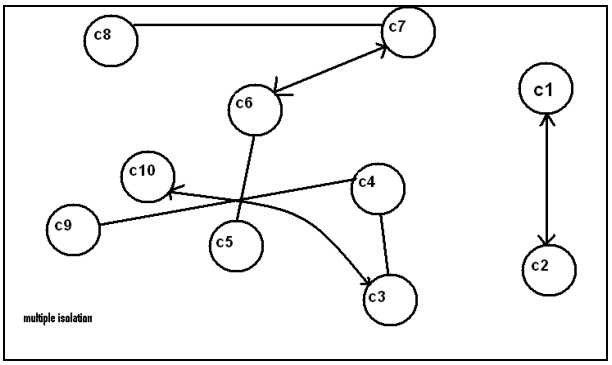

Figure 4

In Figure 4. The chromosome is illegal due to multiple isolated edges.

By observing these problems it has been concluded that there are three main reasons for illegal chromosome:

- Self loop
- Cycle and
- Isolated node or edge

III. Evaluation

By observing these problems, fitness functions have been developed to evaluate these chromosomes. On the basis of these fitness functions, fitness points are given to the chromosomes and on the basis of these fitness points chromosomes are selected as a child population for next generation. Following fitness functions have been developed to evaluate chromosomes.

a) Self Loop
b) Isolated node or edge
c) Cycle

Following data structure have been used for all these developed functions.

Chromosome: it is a matrix of size 10X10 to store 10 randomly generated chromosomes. After the fitness function, fitness point for each chromosome is stored in its last column. Subscript starts from 1.

S (1): it holds the no of row of matrix chromosome
S(2): it holds the no of column of matrix chromosome
Fitnes : it is an array which holds the fitness value for each chromosome.

Self Loop

For the d connected graph

$$G = (V, E)$$

Where $V = \{v_1, v_2 \ldots\ldots v_n\}$
$E = \{e_1, e2 \ldots\ldots e_{n-1}\}$, each edge e_k is associated with vertices (v_i, v_j)

$$(v_i, v_j) \in e_k$$

If (i == j) then it is called self loop for vertex v.
It returns 0 for each self loop and 1 for each non self loop occurrence. For 10 node network, 10 is the maximum fitness point for each chromosome for self loop.

```
function    self_loop()
function [fitnes] = selfloop_calculate(chromosomes)
s=size(chromosomes);
 for i=1:s(1)
      fit = 0;
      for j=1:s(2)
           if(chromosomes(i,j) ~= j)
                fit = fit + 1;
           end
      end
      fitnes(i)=fit;
   end
return;
end
```

Function isolate()
It returns 1 for non isolation and 0 for isolation for each chromosome.

```
function [fitnes] = isolate_calculate(chromosomes)
s=size(chromosomes);
 for i=1:s(1)
      count = 0;
      for j=1:s(2)
           if(j == chromosomes(i,chromosomes(i,j)))
                count = count + 1;
           end
      end
      if ( count > 2)
          fitnes(i)=0;
      else
           fitnes(i)=1;
      end
 end
return;
 end
```

Function cycle()
It returns 1 for non cycle and 0 for cycle for each chromosome.

```
function [fitnes] = cycle_calculate(chromosomes)
a=size(chromosomes);
   for m = 1 : a(1)
      k=0; t=(-1); b=1;e=5;
```

```
    for i = 1:a(2)
        new=0; s=i;
        for j=1:(a(2) + 1)
            if (new == 0)
                check = s;
            else
              check = chromosomes(m,s);
            end
                l=1;
                while(l <=k )
                    if (p(l) == check)
                        if (new == 0)
                            break;
                        end
                        if (l>=b)
                          if (k == (l+1))
                                t=-1;
                                b=k+1;
                                break;
                            end
                           if (k >(l+1))
                                e=0;
                                break;
                            end
                        end
                        if (l<b)
                            t=-1;
                            b=k+1;
                            break;
                        end
                    end
                 l = l + 1;
                end
                 if (e == 0)
                        break;
                 end
                 if (l>k)
                        k=k+1;
                        p(k) = check;
                        t = t + 1;
                 end
                 if (t == -1)
                        break;
                 end
            if (new ~= 0)
            s = chromosomes(m,s);
            end
                 if (new == 0)
                        new = 1;
                 end
        end
          if ( e == 0)
                break;
          end
    end
if( e == 0)
    fitnes1(m)= 0;
else
    fitnes1(m)=1;
end

  end
return;
end
```

Selection

In genetic algorithm fit solution are likely to survive and bad solution are likely to die off. So some of the best fit chromosomes are selected from parent population according to some selection criteria We have use a random number r to select the chromosome. If r is within the range of chromosome then it is selected otherwise it is replaced with the fittest chromosome.

```
function [new_chromosomes] =
simple_selection(chromosomes)
a=size(chromosomes);
row = a(1);
col = a(2)-1;
k=1;
for i=1:row
    r=row*round(rand());
    if((r == 0) || (r > row))
        for l=1:row
  if(chromosomes(l,a(2)) == col+2)
                r=l;
            end
        end

    end

    for j=1:col
new_chromosomes(k,j)=chromosomes(r,j)
    end
    k=k+1;
end
return;
end
```

Crossover/Recombination

This function crossover the chromosome on a single point. First chromosome on first column, with second chromosome. Third chromosome on first and second column , with fourth chromosome and so on.. If crossover point exceeds the maximum column no. then it starts from first column again.

```
function [chromosomes] =
one_point_crossover(chromosomes)
a=size(chromosomes);
row = a(1);col = a(2);
t=1;i=1;
while( i<row)
    for j=1:t
        temp = chromosomes(i,j);
        chromosomes(i,j) = chromosomes((i+1),j);
        chromosomes((i+1),j)=temp;
    end
    i=i+2;
    t=t+1;
    if(t>col)
        t=1;
    end
end
return; end
```

Mutation

Only those chromosomes have been mutated which does not have maximum fitness point.

```
function [mutated_chromosome] =
mutation_simple(new_chromosome)
a=size(new_chromosome);
row = a(1);
k=1;
for i=1:row
   if(new_chromosome(i,a(2)) ~= (a(2)+1))
new_chromosome(i,i) = (a(2)-i);
   end
end
for i=1:row
      for j=1:a(2)-1
  mutated_chromosome(i,j) = new_chromosome(i,j);
      end
end
return; end
```

IV. IMPLEMENATATION

10 Sets of chromosome have been generated randomly for the network of size 10. After fitness function, fitness point is stored in last column of each chromosome.

Randomly generated chromosomes are:

```
     7     8     8     5     5     9     6     7
8     6
     4     4     9     9     1     6    10     6
6     6
     7     9     3     6     1     9     5     2
9     1
     7     4     7     2     1     7     5     7
6     6
     4     8     4     2     7     6     8     1
1     4
     1     4     8     4     5     8     2     6
1     1
     3     8     8     7     2     9     5     7
9     5
     4     8     2     8     5    10     9     7
5     2
     3     4     9     8     1     1     6     9
8     1
     2     2    10     3     1     9     8    10
2     2
```

Chromosomes after cycle, self loop and isolation fitness function

```
     7     8     8     5     5     9     6     7
8     6    10
     4     4     9     9     1     6    10     6
6     6    11
     7     9     3     6     1     9     5     2
9     1     9
     7     4     7     2     1     7     5     7
6     6    11
     4     8     4     2     7     6     8     1
1     4    10
     1     4     8     4     5     8     2     6
1     1     8
     3     8     8     7     2     9     5     7
9     5    10
     4     8     2     8     5    10     9     7
5     2    11
     3     4     9     8     1     1     6     9
8     1    12
     2     2    10     3     1     9     8    10
2     2    11
```

DISTANCE FOR EACH CHROMOSOME

```
   334   321   325   374   345   238   332   310
423   249
```

fittest chromosome is selected on the basis of maximum fitness point(12) and minimum distance

FITTEST CHROMOSOME WITH DISTANCE

```
     3     4     9     8     1     1     6     9
8     1 (dist:   423)
```

**********FIRST GENERATION**********

NEW CHROMOSOME AFTER SELECTION

```
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     2     2    10     3     1     9     8    10
2     2
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     2     2    10     3     1     9     8    10
2     2
     2     2    10     3     1     9     8    10
2     2
```

NEW CHROMOSOME AFTER ONE POINT CROSSOVER

```
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4    10     3     1     9     8    10
2     2
     2     2     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
     3     4     9     8     1     1     6     9
8     1
```

3 4 9 8 1 1 6 9
8 1
2 2 10 3 1 9 8 10
2 2
2 2 10 3 1 9 8 10
2 2

EVALUATED CHROMOSOME AFTER ONE POINT CROSSOVER
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 10 3 1 9 8 10
2 2 **11**
2 2 9 8 1 1 6 9
8 1 **10**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
2 2 10 3 1 9 8 10
2 2 **11**
2 2 10 3 1 9 8 10
2 2 **11**

DISTANCE FOR EACH CHROMOSOME
423 423 314 358 423 423 423 423
249 249

OLD FITTEST CHROMOSOME
3 4 9 8 1 1 6 9
8 1 423

NEW FITTEST CHROMOSOME WITH DISTANCE AFTER ONE POINT CROSSOVER
3 4 9 8 1 1 6 9
8 1 (dist:423.)

NEW CHROMOSOME AFTER MUTATION
3 4 9 8 1 1 6 9
8 1
3 4 9 8 1 1 6 9
8 1
3 4 8 3 1 9 8 10
2 2
2 2 9 7 1 1 6 9
8 1
3 4 9 8 1 1 6 9
8 1
3 4 9 8 1 1 6 9
8 1
3 4 9 8 1 1 6 9
8 1
2 2 10 3 1 9 8 10
2 2
2 2 10 3 1 9 8 10
2 1

EVALUATED CHROMOSOME AFTER MUTATION
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 8 3 1 9 8 10
2 2 **11**
2 2 9 7 1 1 6 9
8 1 **10**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
3 4 9 8 1 1 6 9
8 1 **12**
2 2 10 3 1 9 8 10
2 2 **11**
2 2 10 3 1 9 8 10
2 1 **11**

DISTANCE FOR EACH CHROMOSOME
423 423 279 387 423 423 423 423
249 292

OLD FITTEST CHROMOSOME
3 4 9 8 1 1 6 9
8 1 (423)

NEW FITTEST CHROMOSOME WITH DISTANCE AFTER MUTATION
3 4 9 8 1 1 6 9
8 1 (dist: 423)

**********SECOND GENERATION*********

OLD FITTEST CHROMOSOME
3 4 9 8 1 1 6 9
8 1 (423)

NEW FITTEST CHROMOSOME WITH DISTANCE AFTER MUTATION
2 4 9 8 1 1 6 9
8 1 (dist: 396)

**********THIRD GENERATION*********

OLD FITTEST CHROMOSOME
2 4 9 8 1 1 6 9
8 1 396

NEW FITTEST CHROMOSOME WITH DISTANCE AFTER MUTATION
2 4 9 8 1 1 6 9
8 1 (dist: 396)

**********FOURTH GENERATION*********

OLD FITTEST CHROMOSOME
2 4 9 8 1 1 6 9
8 1 396

NEW FITTEST CHROMOSOME WITH DISTANCE AFTER MUTATION
2 9 10 3 1 9 8 10 2 2 (dist: 270)

Following figure 5. is the spanning tree on the basis of the minimum distance chromosome (270)

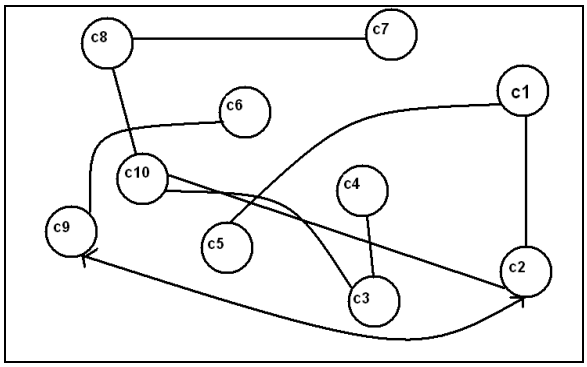

Figure 5

V. Conclusions

In this paper, we have proposed the effect of strong fitness function which reduces the no of generation to find the optimal solution. This is the improved approach of evolutionary computing which gives the very positive result. We have described the problem and GA that encodes the candidate solution to the problem in a simple way and included all the important constraints and developed functions which are required for the solution of Network Optimization Problem. The effectiveness of the methodology however can be increased by applying the various genetic operators with variations and the densely connected locations.

Acknowledgements

My thanks to the reviewers for their cogent and insightful comments.

References

[1] Hamdy A. Taha. 2007. Operation Research An Introduction
[2] S.K. Basu, 2005. Design Methods and Analysis of algorithms (PHI) ISBN : 81-203-2637-7
[3] Melanie M. (1998). An Introduction to genetic Algorithm (PHI) ISBN 81-203-1385-5
[4] Michael D. Vose. 1999. The simple genetic algorithm : (PHI) ISBN 61-203-2459-5
[5] Narsingh Deo, 2000. Graph Theory with Applications to Engineering and Computer science: (PHI)

Anand Kumar is an Assistant Professor in MCA Department of AMC Engineering College Bangalore. He received the B.Sc degree in Physics from B.R.AUniversity, Muzaffarpur, India, the M.C.A degree from IGNOU, Delhi, India in 2001, the M.Phil degree in Computer Science from Madurai Kamaraj University in 2007 and currently pursuing Ph.D from Saurashtra University Rajkot, India. He is a senior member of IACSIT Singapore, Member International Association of Engineers Hong Kong and IEEE listed author. He has authored several papers in reputed international journals and International Conferences.

Dr. N.N.Jani has rich experience of 34 years teaching in computer science and Applications. He is a research guide in Saurashtra University since 1999. He is also the external guide to Ph.D. programme of North Gujarat University, Patan. He has written 15 books in the field of Computer Science and Applications, one of his books on MEMS (Micro Electro Mechanical System) has been appreciated by Ex-honourable president Dr. A.P.J. Abdul Kalam. He has organized 16 workshops, 7 National and 1 International seminar on Emerging Technology and Application