Havrda and Charvat Entropy Based Genetic Algorithm for Traveling Salesman Problem

Baljit Singh, Arjan Singh and Akashdeep
Department of Computer Science & Engineering, BBSB Engineering College,
Fatehgarh Sahib-140407, Punjab, India

Abstract

The application of evolutionary computation techniques for the solution of combinatorial optimization problems is now the major area of research. Genetic algorithm (GA) is an evolutionary technique that uses crossover and mutation operators to solve such problems using a survival of fittest idea. The traveling salesman problem (TSP) is used as a paradigm for a wide class of problem having complexity due to the combinatorial explosion. TSP has become a target for the GA community, because it is probably the central problem in combinatorial optimization and many new ideas in combinatorial optimization have been tested on the TSP. When GA is applied to TSP, frequently encounter a trap of falling into a local optimal solution rather than a best approximate solution. This paper proposes Havrda and Charvat entropy based genetic algorithm for TSP to obtain a best approximate solution in reasonable time. Havrda and Charvat entropy is a measure of diversity of each population into the process of GA to solve the above-mentioned problem of falling into a local optimal solution. The TSP with 10 nodes is used to evaluate the performance of the proposed algorithm. In order to validate the results, TSPs with 20 and 30 nodes are considered to examine the versatility of the proposed algorithm. It has been observed that the proposed algorithm works effectively for different TSPs as compare to general GA. The results of this study are quite promising.

Keywords: Genetic Algorithm, Traveling Salesman Problem, Combinatorial Optimization, Mutation, Crossover, Entropy.

1. Introduction

Genetic Algorithms can be applied to optimization, constraint satisfaction problems and non constraint problems [1]. Problems of these types are traveling salesman problem, job shop scheduling, space allocation and map coloring, shortest path problem [2], linear transportation problem[3] etc. A genetic algorithm is a computer algorithm that searches for a good solution to a problem among a large number of possible solutions [4]. Genetic algorithms are inspired by the evolution in nature. These search algorithms are based on the mechanism of natural selection and natural genetics [5]. Genetic algorithms maintain a population of some feasible solutions for a given problem. This population undergoes evolution in a form of natural selection and natural genetics. In each generation relatively "good" solutions reproduce and relatively "bad" solutions die, to be replaced by offspring of the good. To distinguish between solutions, an evolution or objective function plays the role of the environment.

In the traveling salesman problem, a salesman seeks the shortest tour through given cities, with each city visited exactly once before returning back to his home city. Consider G be a complete graph with n vertices. Take length (<u, v>) be the length of the edge <u, v>. A path starting at a given vertex v_0, going through every other vertex exactly once and finally returning to v_0 will be called a tour. The length of a tour is the sum of lengths of the edges on the path defining the tour. The problem is to find a tour of minimum length. Such problem is called the traveling salesman problem [6]. The importance of the TSP is that it is representative of a large class of problems, known as combinatorial optimization problems known as NP-hard. To date, no one has found an algorithm that will solve such problems in polynomial bounded time. The best-known algorithms for NP-hard have a worst-case complexity that is exponential in the number of inputs. To produce an algorithm of polynomial complexity to solve an NP-hard optimization, it will be necessary to relax the meaning of solving. Relaxation is to remove the requirement of the algorithm, which solves the optimization problem, which must always generate an optimal solution. This requirement will be replaced by the requirement that the algorithm for optimization problem generates a feasible solution with the value close to the optimal solution. The algorithm that finds out such type of solutions is called approximate algorithm. Genetic algorithm works as approximate algorithm to solve NP-hard problems. Such algorithm is used to find approximate solution that is close to optimal for the TSP in reasonable time. Sometimes, genetic algorithm falls into a local optimal solution in the evolution process. This phenomenon occurs when the diversity of a population at the later generation decrease. Decrement of diversity population causes deterioration of

effects of the crossover and mutation operations. So it is necessary to measure and improve pollution diversity at later stage for achieving global optimal solution of the problem.

In this paper, the concept of Havrda and Charvat entropy [14] is introduced to measure and improve pollution diversity in genetic algorithm for solving the above mentioned problem of falling into a local optimal solution. This proposed algorithm is verified on different TSPs with 10,20 and 30 nodes and compare the results of this algorithm with general GA for obtaining a good solution in mentioned time.

2. Genetic algorithm for TSP

John Holland's simple GA [7] inspired all subsequent GAs and provided the basis for theoretical analysis of GAs. For TSP solving, it is clear that the simple GA is not suitable because bit-string encoding is not suitable and simple genetic operators are not the most effective or appropriate. For TSP solving, the genetic algorithm used will have the following chromosome representation, fitness function, selection technique and genetic operators [8] :

2.1 Chromosome representation scheme

Path representation scheme [9] is employed as a suitable one for representing a schedule of the original problem. A chromosome represents a tour of the salesman. A chromosome T_k ($k = 1, 2, \ldots, m$; m is the population size) is represented as:

$$T_{k=} (C_1\ C_2\ C_3 \ldots\ldots\ldots C_n)$$

Where C_i is the i^{th} city to be visited, $i = 1, 2, \ldots\ldots, n$.
Such type of representation is called permutation representation (encoding).

All the cities are sequentially numbered starting from 1. The route between the cities is described with an array. Each element of the array represents the number of the city. The array represents the sequence in which the cities are traversed to make up a tour. Each chromosome must contain each and every city exactly once.
Example: Considering a TSP with 7 nodes, a tour:

$1\rightarrow3\rightarrow7\rightarrow5\rightarrow4\rightarrow2\rightarrow6$

can be represented using the path representing as

(1 3 7 5 4 2 6)

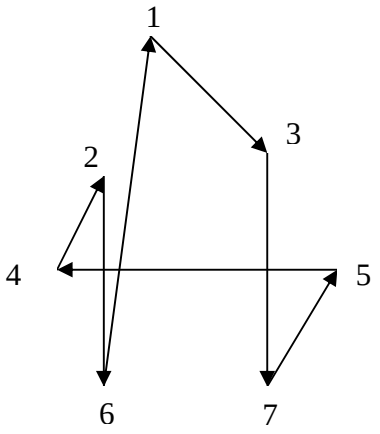

Figure 1

Shows the tour 1→3→7→5→4→2→6

For array representation of such tour, we declare an array t[7].

1	3	7	5	4	2	6
t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]

This chromosomes represents the tour stating from city 1 to city 3, city 3 to city 7, city 7 to city 5, city 5 to city 4, city 4 to city 2, city 2 to city 6 and city 6 to city 1. Chromosome describes the order of cities, in which the salesman will visit them.

2.2 Fitness function

The objective function, the function to be optimized, provides the mechanism for evaluating the fitness of each chromosome. The fitness function fit (T_k) ($k = 1, 2, \ldots, m$) is defined as:

$$fit(T_k) = \frac{1}{\sum_{i=1}^{n} d(C_i, C_{i+1}) + d(C_n, C_1)}$$

Where $d(C_i, C_{i+1})$ is traveling distance from city C_i to C_{i+1}

We use computer screen as platform to describe TSP. Pixels are used to represent cities. If city C_i is represented by pixel $p(x, y)$ and city C_{i+1} is represented by pixel $q(s, t)$, then $d(C_i, C_{i+1})$ is Euclidean distance between two pixels $p(x, y)$ and $q(s, t)$ that is defined as :

$$d(C_i, C_{i+1}) = |(x - s)| + |(y - t)|$$

$$= \sqrt{(x-s)^2 + (y-t)^2}$$

Algorithm for fitness measure

Step 1: Traverse the cities according to the sequence in a tour

Step 2: Calculate $d(C_i, C_{i+1})$ using equation

$$\sqrt{(x-s)^2 + (y-t)^2}$$

and find the total distance in the tour

$$Total\ \ dis\tan ce = \sum_{i=1}^{n} d(C_i, C_{i+1}) + (C_n, C_1)$$

Step 3: Calculate the fitness of the chromosome in the population

$$fit\ (T_k) = 1/Total\ distance$$

2.3 Selection Method

In this algorithm, we use steady-state selection mechanism [10]. In steady-state selection, only a few individuals are replaced in each generation. Usually a small number of the least fit individuals are replaced by offspring resulting from crossover and mutation of the fittest individuals.

2.4 Crossover operator

A large number of crossover have been developed for the permutation encoding such as partially mapped crossover (PMX), order crossover (OX), cycle crossover (CX), edge recombination (ERX) [11], edge assembly crossover (EAX) [12] etc In this algorithm, partially mapped crossover (PMX) mechanism is used. Under PMX, two strings are aligned, and two crossing sites are picked uniformly at random along the strings. These two points define a matching section that is used to affect a cross through 'position-by-position' exchange operations i.e. each element between the two crossover points in the alternate parent are mapped to the position held by this element in the first parent.

Algorithm for PMX

Step 1: Two chromosomes as parent P_1 and P_2 are aligned, and two crossover sites are picked uniformly at random along the chromosomes.

Step 2: Each element between the two crossover points in the alternate parent is mapped to the position held by this element in the first parent.

Step 3: The remaining elements are inherited from the parent without any conflict.

Step 4: If conflict occurs, then for the first child:

(a) Find the position of the element, where conflict occurs, in the second parent. Pick the element from that position in the first parent and place it that position where conflict occur in the first child.

(b) For the second child, parent roles reversed.

2.5 Mutation operator

In this algorithm, swap mutation operator [13] is used. In swap mutation operation, pick two alleles at random and swap their positions. Mutation operator reserves most of adjacency information and disrupts order more.

Algorithm for SMO

Step 1: Randomly choose one tour and randomly select two mutation points.

Step 2: Interchange the cities at these two points.

General genetic algorithm for solving TSP

Step 1: Initialize GA parameters

Set number of cities: n, population size: m, crossover probability: p_c, mutation probability: p_m, and total time: S.

Let *starting time* $s=0$, $maxfit = 0$

Generate m chromosomes (tours) randomly.

Step 2: Evaluate

Step 2.1: Calculate the fitness value of each chromosome

Step 2.2: if $maxfit < max\{fit(T_k)\}$

$bestsol = findbest\{fit(T_k)\}$

and

$maxfit = max\{fit(T_k)\}$

Endif

Where *max* is a operator that find the maximum fitness value of the chromosome from the population i.e. find the minimum distance of the tour from all tours in the population and *findbest* is a operator which takes the T_k (chromosome or tour) having the largest fitness value i.e. take the tour having minimum distance.

Step 3: Crossover

Perform the crossover PMX on chromosomes selected with probability p_c.

Step 4: Mutation

Perform the swap mutation on chromosomes selected with probability p_m.

Step 5: Selection

Select m chromosomes from the parents and offspring for the next generation by steady state selection method.

Step 6: Stop testing

If $s<S$

return to step 2

else
output *bestsol*
Endif

3. Havrda and Charvat entropy based genetic algorithm for TSP

To avoid falling into a local optimal solution in the evolution process, Havrda and Charvat entropy based genetic algorithm is proposed which gives good solution in reasonable time. Such algorithm can keep high diversity of chromosome population at later stage

3.1 Havrda and Charvat entropy

Let $P = (p_1, p_2, \ldots\ldots\ldots\ldots p_n)$ be a probability distribution, then Havrda and Charvat [14] gave the measure

$$H(P) = \frac{1}{1-\alpha}\left(\sum_{j=1}^{n} {p_j}^{\alpha} - 1\right) \qquad \alpha > 0$$

to measure its uncertainty or entropy.

3.2 Measure of chromosome population diversity

The trap of local optimal solution is one of the most important problems of GA. Such problem occurs when all chromosomes in a population become similar. In this case, crossover and mutation operators do not play significant role. To escape from the trap of local optimal solution, we need to maintain high diversity of the population. In the proposed scheme, the diversity of chromosome population is measured using Havrda and Charvat entropy at later stage and then improves low diversity population. Evaluation of the diversity of chromosome population is done by obtaining the locus diversity for each loca of all chromosomes in the population.

The locus diversity H_j of the j^{th} locus *(j=1 ,2,……n)* is defined as:

$$H_j = \frac{1}{1-\alpha}\left(\sum_{i \in Cities} {p_{ji}}^{\alpha} - 1\right) \qquad \alpha > 0$$

where

$$p_{ji} = \frac{na_{ji}}{m}$$

na_{ji} : the number of appearance of city *i* at locus *j*

H_j approaches to the maximum value z= $\left(m^{1-\alpha} - 1\right)/(1-\alpha)$ when each city appears uniformly in the population, conversely it approaches to the minimum value 0 when a city appear in the same locus of all chromosomes in the population.

Procedure to measure chromosome population diversity

count=0
For j=1 to n
Compute H_j
If H_j < z/2
count=count+1
Endif
Endfor
If count> n/a
The diversity of the chromosome population is too low and need to improve it
Else
The diversity of the chromosome population is high and no need to improve it
Endif

Where a is a control parameter. Larger the value of a , higher the probability of doing improvement and vise versa.

3.3 Population diversity improvement

Step1: select *q* chromosomes from the population
q is a random integer number, *(m/4)<q<(m/2)*
Step2: Exchange genes among the loca which have lower locus diversities in selected chromosomes.

Havrda and Charvat entropy based genetic algorithm for TSP

Step 1: Initialize GA parameters
Set number of cities: *n*, population size: *m*, crossover probability: p_c, mutation probability: p_m, and total time: *S*.
Let *starting time s*=0, *maxfit* = 0
Iteration =0
Generate *m* chromosomes (tours) randomly.
Step 2: Evaluate
Step 2.1: Calculate the fitness value of each chromosome
Step 2.2: if *maxfit < max {fit (T_k)}*

bestsol = findbest {fit (T_k)}
and
maxfit = max {fit (T_k)}
Endif

Where *max* is a operator that find the maximum fitness value of the chromosome from the population i.e. find the minimum distance of the tour from all tours in the population and *findbest* is a operator which takes the T_k (chromosome or tour) having the largest fitness value i.e. take the tour having minimum distance.

Step 3: Crossover

Perform the crossover PMX on chromosomes selected with probability p_c.

Step 4: Mutation

Perform the swap mutation on chromosomes selected with probability p_m.

Step 5: Selection

Select *m* chromosomes from the parents and offspring for the next generation by steady state selection method.

Step 6: Measure of chromosome population diversity

If *iteration* <ITER (value of ITER is set experimentally)

Goto step 8

Else

Evaluate chromosome population diversity

Endif

Step 7: Population diversity improvement

If population diversity is too low

Improve it

Endif

Step 8: Stop testing

If *s<S*

Iteration=iteration+1

Goto to step 2

Else

output *bestsol*

Endif

4. Experiment Results

General genetic algorithm and Havrda and Charvat entropy based genetic algorithm for TSP are implemented in C++. TSPs with 10, 20 and 30 nodes are used to analyze general genetic algorithm and Havrda and Charvat entropy based genetic algorithm .The experiment has been made on a PC (Pentium-4: 2GHz CPU, 256 RAM, OS: Windows 98,Turbo C++: version 3.0). Computer monitor is used as a platform to perform experiment. Nodes (Cities) are displayed graphically on the screen. Distance between two nodes is Euclidean distance, which is defined as

$$d = \sqrt{(x-s)^2 + (y-t)^2}$$

where, one node is at pixel *(x, y)* and other node is at pixel *(s, t)*. When program run, randomly nodes are created according to the size of problem. We pick initial random solution (tour) and calculate its cost. After a fixed time, some iterations (generations) are done and we get a solution, which is called best solution (tour). The quality of solution is measured by comparing the cost of final obtained tour after a fixed time with initial random tour. When number of nodes = 20, the initial random tours according to the population size are shown in Figure 2 and the best solution is shown in Figure 3.

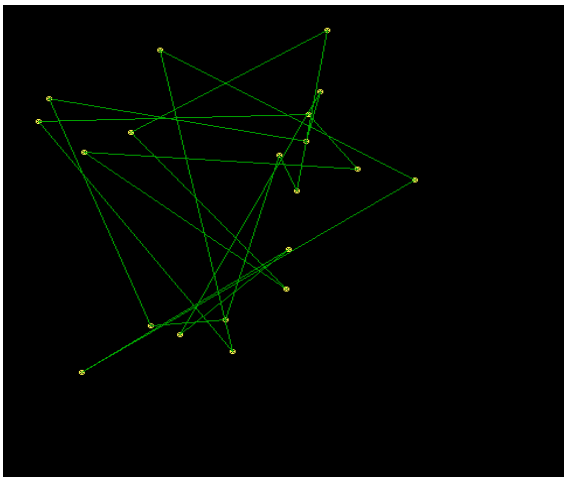

Figure 2: Initial Random Tours

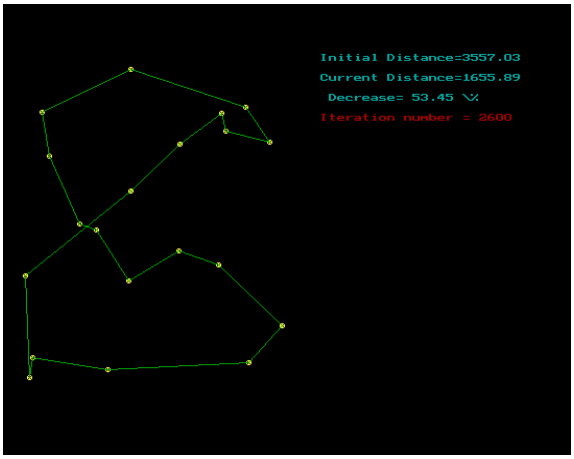

Figure 3: The best tour after 30 seconds

One important characteristics of the current literature on GA is the solution quality has been considered as the only performance measurement. So the performance of both algorithms depends upon the quality of solution in a fixed time. Such fixed time is CPU time. CPU time excludes I/O processing time and is measured using the clock function (library function available in C++) which returns the number of CPU ticks from the beginning of the program. The solution (tour) quality is measured from the cost (length) of solution (tour). Lesser the cost of solution, better the quality. To measure the quality of solution, initial random tour is chosen first and then its cost is calculated. After some fixed time, the solution is found and its cost is calculated. The quality of solution is measured in term of %age decrease between the cost of initial random solution and final solution

$$\%decrease = \frac{\cos tirs - \cos tfs}{\cos tirs} \times 100$$

where $costirs$: cost of initial random solution
$costfs$: cost of final solution

Experiment results that are shown in Table-1 are obtained on a TSP of size n=10. In this case population size=8 and CPU time=30 seconds. 15 independent trials (means run the program 15 times) are performed. In each trial, cities are randomly generated on the screen and initial random tour is picked. The cost of this initial random tour is calculated. After some iterations (generations), which are performed in 30 seconds, a final tour is obtained. The cost of this final tour and the %age decrease between initial random tour and final tour is calculated. Similarly experimental results which are shown in table 2 and table 3 for TSPs of size=20 (set population size=18 and time=35 seconds) and size=30(set population size=18 and time=35 seconds) respectively.

Table-1: Performance of Havrda & Charvat entropy based GA and general GA and when m=8, n=10 & time=30 seconds

Trial No.	%age decrease by Havrda and Charvat entropy based GA	%age decrease by general GA
1	58.13	46.47
2	51.13	26.18
3	58.09	39.42
4	53.34	16.36
5	56.22	30.15
6	59.22	45.17
7	48.21	29.84
8	48.13	35.61
9	52.56	35.93
10	49.42	32.27
11	57.38	23.15
12	58.70	21.80
13	59.98	30.36
14	59.98	38.36
15	58.72	34.29

Table-2: Performance of Havrda &Charvat entropy based GA and general GA when m=18, n=20 & time=35 seconds

Trial No.	%age decrease by Havrda and Charvat entropy based GA	%age decrease by general GA
1	62.87	57.70
2	65.19	58.81
3	62.75	53.69
4	53.84	47.28
5	67.20	51.44
6	63.53	55.05
7	60.05	54.88
8	56.81	47.18
9	58.52	55.74
10	57.87	47.58
11	60.88	55.91
12	64.89	48.64
13	65.07	52.21
14	56.69	53.19
15	61.76	54.02

Table-3: Performance of Havrda & Charvat entropy based GA and general GA and when m=28, n=30 & time=40 seconds

Trial No.	%age decrease by Havrda and Charvat entropy based GA	%age decrease by general GA
1	64.23	46.56
2	60.88	46.60
3	54.95	43.02
4	60.98	35.22
5	59.95	38.39
6	62.57	36.49
7	60.12	41.82
8	57.00	45.31
9	54.30	29.40
10	60.54	37.41
11	57.55	47.36
12	65.24	33.51
13	56.53	48.15
14	63.33	37.38
15	52.93	47.07

15 independent trials are carried out on Havrda and Charvat entropy based GA and general GA, and the statistical results are shown in table 4 .To show the quality of result, the %age decrease between initial random tour and final tour is listed. This table shows the best and average %age decrease Hamilton distance (denoted by BEST and AVG respectively). Furthermore, to show the significant improvement of Havrda and Charvat entropy based GA, statistical test is used that is defined as:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{S}\sqrt{\frac{n_1 n_2}{n_1 + n_2}}$$

where $\overline{X}_1$: mean of %age decrease Hamilton distance of Havrda and Charvat entropy based GA

$\overline{X}_2$: mean of %age decrease Hamilton distance of general GA

n_1 : number of trials for Havrda and Charvat entropy based GA

n_2 : number of trials for general GA

S : combined standard deviation

S *is defined as:*

$$S = \sqrt{\frac{\sum_{i=1}^{15}\left(X_{1i} - \overline{X}_1\right)^2 + \sum_{i=1}^{15}\left(X_{2i} - \overline{X}_2\right)^2}{n_1 + n_2 - 2}}$$

where X_{1i} : %age decrease Hamilton distance in i^{th} trial of Havrda and Charvat entropy based GA

X_{2i} : %age decrease Hamilton distance in i^{th} trial of general GA

Table-4: Statistical comparison of Havrda &Charvat entropy based GA and general GA

Problem size	Havrda &Charvat entropy based GA		General GA		t
	BEST	AVG	BEST	AVG	
10	59.98	55.26	46.47	32.36	10.59
20	65.44	61.19	58.81	52.89	6.05
30	65.24	59.41	48.15	40.91	10.74

From the AVG values in Table-4, it can de concluded that Havrda and Charvat entropy based GA can always obtain better results than general GA. Secondly, from the BEST values, it can be concluded that Havrda and Charvat entropy based GA can obtain the best solution than that obtained by general GA. Meanwhile, with test statistic *t*, the following hypothesis can be tested:

$H_0 : \mu_1 = \mu_2$ *against*

$H_\alpha : \mu_1 > \mu_2$ (To conclude that, in Havrda and Charvat entropy based GA %age decrease Hamilton distance is greater than %age decrease Hamilton distance in general GA)

Where μ_1, μ_2 denote the performance of Havrda and Charvat entropy based GA and general GA.

In these samples, degree of freedom is n_1+n_2-2 =28. As H_α is one-sided, apply one-tailed test for determine the rejection region at 5 percent level (take the level of significance α =0.05) using t-distribution [15] table for 28 degree of freedom: $t_\alpha(n_1 + n_2 - 2)$=1.0701. From Table-4, it can be clearly concluded that Havrda and Charvat entropy based GA perform better than the general GA, since all test statistics t are greater than t_α , where null hypothesis is rejected.

5. Conclusions and Future works

In this study, we proposed Havrda and Charvat entropy based GA for TSP. This improves the population diversity by introducing the concept of Havrda and Charvat entropy to escape from the local optimal trap. The experiment results show that the quality of solution of TSP obtained from Havrda and Charvat entropy based GA is good as compare to the solution obtained from general GA. Thus, Havrda and Charvat entropy based GA find a good tour for TSP from large search space in reasonable time.

For future research works, we can apply Havrda and Charvat entropy based GA to combinatorial optimization problems such as scheduling of processors, graph partitioning, optimal computer network design, and so on.

6. References

[1] R.Shankuriler, Kenyon R.Miller, “genetic algorithms / neural network synergy for nonlinearly constrained optimization problems”, IEEE, 1992.

[2] Mistuo Gen, Runwei Cheng and Dingwei Wang, “genetic algorithms for solving shortest path problem”, IEEE, 1997.

[3] G.A. Vignaux and Z.Michalewicz, “a genetic algorithm for the linear transportation problem”, IEEE 1991.

[4] David E. Goldberg, “genetic algorithms in search, optimization and machine learning”, Pearson Education, 2004.

[5] M.Srinivas, Lalit M. Patnaik, “genetic algorithms: a survey”, IEEE, 1994.

[6] Sartaj Sahni, “algorithms analysis and design”, Galgotia Publications Pvt. Ltd., New Delhi, 1996.

[7] Melanie Mitchell, “ an introduction to genetic algorithms”, PHI, 2002.

[8] Kalyanmoy Deb, “optimization for engineering design”, PHI, 2003.

[9]Lawrence J. Schmitt, Mohammad M. Amini , “performance characteristics of alternative genetic algorithm approaches to the traveling salesman problem using path representation: an empirical study”, European journal of operation research ,1998.

[10] S.Rajasekaran, G.A. Vijayalakshmi Pai, “neural network, fuzzy logic and genetic algorithms”, PHI, 2003.

[11] H.D. Nguyen, I. Yoshihara, M. Yasunaga, “modified edge recombination operators of genetic algorithms for the traveling salesman problem”, IEEE, 2000.

[12]Yuichi Nagata , Shigenobu Kobayashi, “ an analysis of edge assembly crossover for the traveling salesman problem”, IEEE,1999.

[13] Wayne Pullan, “adapting the genetic algorithm to the traveling salesman problem” , IEEE,2003.

[14] J.N. Kapur, “measures of information and their applications”, Wiley Eastern Limited, New Delhi, 1994.

[15] C.R. Kothari, “research methodology”, Wishwa prakashan,2002.