© 2014 IJIRT | Volume 1 Issue 11 | ISSN: 2349-6002

Research Paper on Travelling Salesman Problem And it's Solution Using Genetic Algorithm

Kartik Rai, Lokesh Madan, Kislay Anand

Student (B.tech VI^{tth} sem) Department of Computer science

Dronacharya College Of Engineering,Gurgaon-122506

***Abstract*- Genetic Algorithm is used to solve an optimization problems and Travelling Salesman Problem (TSP) is an optimization problem. TSP is an NP hard problem in combinational optimization important in operations research and theoretical computer science. Travelling Salesman Problem (TSP) is always fascinating for the computational scientists and poses interesting challenges to formulate fast and effective algorithms for its solution. It is really challenging to design a robust and effective algorithm, which can give optimal solutions to TSP with large dimensions. There are many algorithms proposed in the literature based on various branches of mathematics, graph theory, operation research and even in fuzzy theory. The amount of computational time to solve this problem grows exponentially as the number of cities. These problems demand innovative solutions if they are to be solved within a reasonable amount of time. This paper explores the solution of Travelling Salesman Problem using genetic algorithms. The aim of this paper is to review how genetic algorithm applied to these problem and find an efficient solution.**

***Index Terms*- TSP, Genetic Algorithm (GA), Selection, Mutation, Crossover.**

I. INTRODUCTION

The idea of the traveling salesman problem (TSP) is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized.

The first instance of the traveling salesman problem was from Euler in 1759 whose problem was to move a knight to every position on a chess board exactly once.

The traveling salesman first gained fame in a book written by German salesman BF Voigt in 1832 on how to be a successful traveling salesman. He mentions the TSP, although not by that name, by suggesting that to cover as many locations as possible without visiting any location twice is the most important aspect of the scheduling of a tour. The origins of the TSP in mathematics are not really known all we know for certain is that it happened around 1931.

The standard or symmetric traveling salesman problem can be stated mathematically as follows:

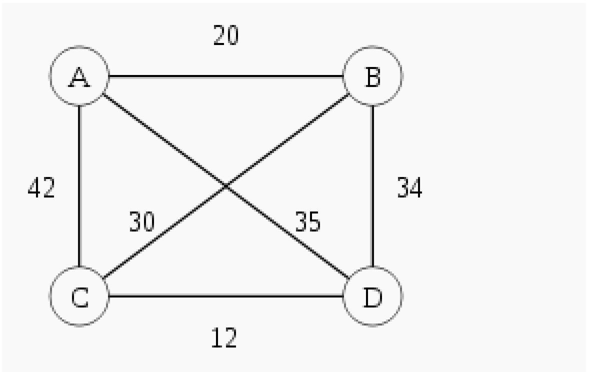
Given a weighted graph G = (V,E) where the weight cij on the edge between nodes i and j is a non-negative value, find the tour of all nodes that has the minimum total cost.

Currently the only known method guaranteed to optimally solve the traveling salesman problem of any size, is by enumerating each possible tour and searching for the tour with smallest cost. Each possible tour is a permutation of 123 . . . n, where n is the number of cities, so therefore the number of tours is n!. When n gets large, it becomes impossible to find the cost of every tour in polynomial time.

II. DEFINITION

Let G = (V, A) be a graph where V is a set of n vertices. A is a set of arcs or edges, and let *C*: *(Cij)* be a *distance* (or *cost)* matrix associated with A. The TSP consists of determining a minimum distance circuit passing through each vertex once and only once. Such a circuit is known as a *tour* or *Hamiltonian circuit* (or *cycle).* In several applications, C can also be interpreted as a cost or travel time matrix.

As a graph problem



Symmetric TSP with four cities

TSP can be modelled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length. It is a minimization problem starting and finishing at a specified vertex after having visited each othervertex exactly once. Often, the model is a complete graph (*i.e.* each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.

Asymmetric and symmetric

In the *symmetric TSP*, the distance between two cities is the same in each opposite direction, forming an undirected graph. This symmetry halves the number of possible solutions. In the *asymmetric TSP*, paths may not exist in both directions or the distances might be different, forming a directed graph. Traffic collisions, one-way streets, and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down.

III. APPLICATIONS OF TSP

The most common practical interpretation of the TSP is that of a salesman seeking the shortest tour through n clients or cities. Several interesting permutation problems not directly associated with routing can also be described as TSPs. Here are some selected examples.

1. Computer wiring (Lenstra and Rinnooy Kan, 1975). Some computer systems can be described as modules with pins attached to them. It is often desired to link these pins by means of wires, so that exactly two wires are attached to each pin
and total wire length is minimized.

2. Wallpaper cutting (Garfinkel, 1977). Here, n sheets must be cut from a roll of wallpaper on which a pattern of length 1 is repeated. For sheet i, denote by *a i* and b i the starting and ending points on the pattern where 0 <= *a i* <= 1 and 0 <= *b i*<= 1. Then cutting sheet j immediately after sheet i results in a waste of *Cij=aj-bi i f bi<=aj or Cij=1+aj-bi if bi>aj* The objective is to order the n sheets so as to minimize total waste.

3. Hole punching (Reinelt, 1989). In several manufacturing contexts, it is necessary to punch holes on boards or metallic sheets. The problem consists of determining a minimum-time punching sequence. Such a problem occurs frequently in metallic sheet manufacturing and in the construction of circuit boards. These problems are often of large scale and must be solved in realtime.

4. Job sequencing:. Suppose n jobs must be performed sequentially on a single machine and that *cij* is the change-over time if job j is executed immediately after job i. Then again, by introducting a dummy job, this problem can be formulated as a TSP.

5. Dartboard design (Eiselt and Laporte, 1991). Dartboards are circular targets with concentric circles, and 20 sectors identified by the numbers 1 to 20. Players throw darts at target points on the board. In the most common version of the game, the objective is to reduce an initial value of 301 to zero by substracting scores. The game rewards accuracy in the throw and it is often more important to hit one's target that to merely register a large score. A natural objective for designing a dartboard is therefore to position the 20 numbers around the board so as to maximize players' risk. For fairly accurate players, it is reasonable to assume that the sector that is hit is always the targeted sector or its neighbour.

6. Crystallography (Bland and Shallcross, 1989). In crystallography, some experiments consist of taking a large number of X-ray intensity measurements on crystals by means of a detector. Each measurement requires that a sample of the crystal be mounted on an apparatus and that the detector be positioned appropriately. The order in which the various measurements on a given crystal are made can be seen as the solution of a TSP. In practice, these problems are of large scale and obtaining good TSP solutions can reduce considerably the time needed to carry out all measurements.

7. Overhauling gas turbine: An application found by Plate, Lowe and Chandrasekaran (cited in [8]) is overhauling gas urbine engines in aircraft. Nozzle-guide vane assemblies, consisting of nozzle guide vanes fixed to the circumference, are located at each turbine stage to ensure uniform gas flow. The placement of the vanes in order to minimize fuel consumption can be modelled as a symmetric TSP.

8. Job scheduling: The scheduling of jobs on a single machine given the time it takes for each job and the time it takes to prepare the machine for each job is also TSP. We try to minimize the total time to process each job.

IV. METHODS OF SOLVING THE TSP

Homaifar states that "one approach which would certainly find the optimal solution of any TSP is the application of exhaustive enumeration and evaluation.
The procedure consists of generating all possible tours and evaluating their corresponding tour length. The tour with the smallest length is selected as the best, which is guaranteed to be optimal. If we could identify and evaluate one tour per nanosecond (or one billion tours per second), it would require almost ten million years (number of possible tours = 3.2×1023) to evaluate all of the tours in a 25-city TSP".
Obviously we need to find an algorithm that will give us a solution in a shorter amount of time. As we said before, the traveling salesman problem is NP-hard so there is no known algorithm that will solve it in polynomial time. We will probably have to sacrifice optimality in order to get a good answer in a shorter time. Many algorithms have been tried for the traveling salesman problem. We will explore a few of these in this section.

Greedy Algorithms are a method of finding a feasible solution to the traveling salesman problem. The algorithm creates a list of all edges in the graph and then orders them from smallest cost to largest cost. It then chooses the edges with smallest cost first, providing they do not create a cycle. The greedy algorithm gives feasible solutions however they are not always good.

The Nearest Neighbor algorithm is similar to the greedy algorithm in its simple approach. We arbitrarily choose a starting city and then travel to the city closest to it that does not create cycle. We continue to do this until all cities are in the tour. This algorithm also does not always give good solutions because often the last edge added to the tour (that is, the edge en1 where n is the number of cities) can be quite large.

A minimum spanning tree is a set of $n - 1$ edges (where again n is the number of cities) that connect all cities so that the sum of all the edges used is minimized. Once we have found a minimum spanning tree for our graph we can create a tour by treating the edges in our spanning tree as bidirectional edges.
We then start from a city that is only connected to one other city (this is known
as a 'leaf' city) and continue following untraversed edges to new cities. If there is no untraversed edge we go back along the previous edge. We continue to do this until we return to the starting city. This will give us an upper bound for the optimal traveling salesman tour. Note, however, that we will visit some cities more than once. We are able to fix this if whenever we need to traverse back to a city we have already been to, we instead go to the next unvisited city. When all cities have been visited we go directly back to the starting city.

Description

Genetic algorithm (GA) as a computational intelligence method is a search technique used in computer science to find approximate solutions to combinatorial optimization problems. The genetic algorithms are more appropriately said to be an optimization technique based on natural evolution. They include the survival of the fittest idea algorithm. The idea is to first „guess" the solutions and then combining the fittest solution to create a new generation of solutions which should be better than the previous generation.

Theory

Genetic algorithms (GAs) are based essentially on mimicking the survival of the fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology. In order to solve any real life problem by GA, two main requirements are to be satisfied:
(a) A string can represent a solution of the solution space, and
(b) An objective function and hence a fitness function which measures the goodness of a solution can be constructed / defined.

The genetic algorithm process consists of the following steps:
1. [Start] Generate random population of *n* chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
4. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
5. [Crossover] is performed with a probability known as crossover probability that crossover the selected parents to produce a new offspring (children). If crossover probability is 0%, children are an exact copy of parents.
6. [Mutation] is performed with a probability known as mutation probability that mutate new offspring at each locus (position in chromosome).
7. [Accepting] Place new offspring in a new population
8. [Replace] Use new generated population for a further run of algorithm
9. [Test] If the termination condition is satisfied, then stop the algorithm, and return the best solution in current population
10. [Loop] Go to step **2**

Genetic coding

To apply GA for any optimization problem, one has to think a way for encoding solutions as feasible chromosomes so that the crossovers of feasible chromosomes result in feasible chromosomes. The techniques for encoding solutions vary by problem and, involve a certain amount of art. For the TSP, solution is typically represented by chromosome of length as the number of nodes in the problem.
Each gene of a chromosome takes a label of node such that no node can appear twice in the same chromosome. There are mainly two representation methods for representing tour of the TSP – adjacency representation and path representation. We consider the path representation for a tour, which simply lists the label of nodes. For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5 node instance, then a tour {1→ 3→4→ 2→ 5 →1} may be represented as (1, 3, 4, 2, 5).

Reproduction operator

In reproduction/selection process, chromosomes are copied into next generation mating pool with a probability associated with their fitness value. By assigning to next generation a higher portion of the highly fit chromosomes, reproduction mimics the Darwinian survival-of-the-fittest in the natural world.
In natural population, fitness is determined by a creature's ability to survive predators, pestilence, and other obstacles to adulthood and subsequent reproduction. In this phase no new chromosome is produced. The commonly used reproduction operator is the proportionate reproduction operator, where a string is selected for the mating pool with a probability proportional to its fitness value. We have considered the stochastic remainder selection method for our genetic algorithms.

The genetic algorithm process consists of the following:

1. Encoding: A suitable encoding is found for the solution to our problem so that each possible solution has unique encoding and the encoding is some form of a string. We do have a graph such as the one described above and we can encode it in the same way, only our matrix will have a one in the i, jth

position if there is an arc from node i to node j in the tour and a zero otherwise. For example, the matrix
[0 0 1]
[1 0 0]
[0 1 0]
represents the tour that goes from city 1 to city3,city 3 to city 2 and city 2 to city 1.
This encoding is known as **matrix representation**

2. Evaluation: The initial population is then selected, usually at random though alternative techniques using heuristics have also been proposed. The fitness of each individual in the population is then computed that is, how well the individual fits the problem and whether it is near the optimum compared to the other individuals in the population. 2.2.2 Evaluation
We use the evaluation function to decide how 'good' a chromosome is. The evaluation function usually comes straight from the problem. In our case the evaluation function would simply be the function f = −2x2 + 4x − 5, and because we are trying to maximize the function, the larger the value for f, the better. So, in our case, we would evaluate the function with the two values 7 and 12. f(7) = −71
f(12) = −241
Obviously 7 is a better solution than 12, and would therefore have a higher fitness.
This fitness is then used to decide the probability that a particular chromosome would be chosen to contribute to the next generation. We would normalize the scores that we found and then create a cumulative probability distribution. This is then used in the crossover process.

3. Crossover: The fitness is used to find the individual‟s probability of crossover. Crossover is where the two individuals are recombined to create new individuals which are copied into the new generation.
Crossover can be a fairly straightforward procedure. In our example, which uses the simplest case of crossover, we randomly choose two chromosomes to crossover, randomly pick a crossover point, and then switch all genes after that point. For example, using our chromosomes
v1 = 0111
v2 = 1100 we could randomly choose the crossover point after the second gene
v1 = 01 | 11
v2 = 11 | 00
Switching the genes after the crossover point would give
v01 = 0100 = 4
v02 = 1111 = 15
We now have two new chromosomes which would be moved into the next population, called the next generation.
Not every chromosome is used in crossover. The evaluation function gives each chromosome a 'score' which is used to decide that chromosome's probability of crossover. The chromosomes are chosen to crossover randomly and the chromosomes with the highest scores are more likely to be chosen. We use the cumulative distribution created in the evaluation stage to choose the chromosomes.

4. Mutation: Next mutation occurs. Some individuals are chosen randomly to be mutated and then a mutation point is randomly chosen. The character in the corresponding position of the string is changed.
Mutation is used so that we do not get trapped in a local optimum. Due to the randomness of the process we will occasionally have chromosomes near a local optimum but none near the global optimum. Therefore the chromosomes near the local optimum will be chosen to crossover because they will have the better fitness and there will be very little chance of finding the global optimum. So mutation is a completely random way of getting to possible solutions that would otherwise not be found.

5. Decoding: Once this is done, a new generation has been formed and the process is repeated until some stopping criteria have been reached. At this point the individual who is closest to the optimum is decoded and the process is complete.

V. SOLUTION OF TSP USING GENETIC ALGORITHM

1. Population Size – When the algorithm starts the initial number of random tours that are created is known as population size. When the population size is larger it will takes longer time to find the result. A small size of population increases the chance in the population that every tour will eventually look the same. This increases the chance that the best solution will not be found.

2. Neighborhood - This number of tours are randomly chosen for each generation from the population. The two best tours become the parents. The worst two tours are replaced by the children. When the group size is high it increases the chance for good tours will be selected as a parent. A large group size will cause the algorithm to run faster, but it might not find the best solution.

3. Mutation Percentage - The percentage that each child after crossover will undergo **mutation** .When a tour is mutated, one of the cities is randomly moved from one point in the tour to another.

4. **Nearby Cities** – As the genetic algorithm is a part of a greedy initial population, it will prefer to link cities that are nearer to each other to make the initial tours.

5. Nearby City Odds percentage - This is the percent chance that any one link in a random tour in the initial population will prefer to use a nearby city instead of a completely random city. If the GA chooses to use a nearby city, then there is an equally random chance that it will be any one of the cities from the previous parameter.

6. Maximum Generations - How many crossovers are run before the algorithm is terminated.

First, create a group of many random tours in what is called a population. This algorithm uses a greedy initial population that gives preference to linking cities that are close to each other. Second, pick 2 of the better (shorter) tours parents in the population and combine them to make 2 new child tours. Hopefully, these children tour will be. A small percentage of the time, the child tours is mutated. This is done to prevent all tours in the population from looking identical. The new child tours are inserted into the population replacing two of the longer tours. The size of the population remains the same. New children tours are repeatedly created until the desired goal is reached. The accuracy of solution in TSP will depend upon factors such as: **->Speed ->Population Size**

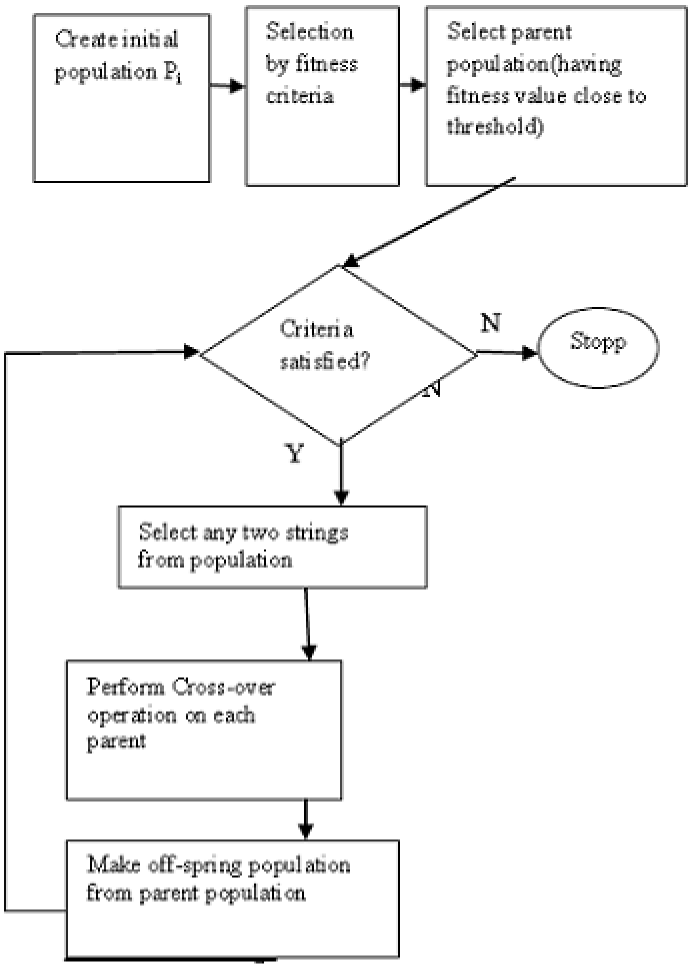
After comparing these factors in each solution the best one will be selected and hence will give the new shortest path in each iteration. The task of comparisons and then representing the solution in every iteration become complex with the increment in population size

VI. METHODOLOGY

A simple GA works by randomly generating an initial population of strings, which is referred as gene pool and then applying (possibly three) operators to create new, and hopefully, better populations as successive generations. The first operator is reproduction where strings are copied to the next generation with some probability based on their objective function value. The second operator is crossover where randomly selected pairs of strings are mated, creating new strings. The third operator, mutation, is the occasional random alteration of the value at a string position. The crossover operator together with reproduction is the most powerful process in the GA search. Mutation diversifies the search space and protects from loss of genetic material that can be caused by reproduction and crossover. So, the probability of applying mutation is set very low, whereas the probability of crossover is set very high.

Steps of Algorithms

1. Randomly create the initial population of individual string of the given TSP problem and create a matrix representation of the cost of the path between two cities.
2. Assign the fitness to each chromosome in the population using fitness criteria measure. $F(x) = 1/x$ where, x represents the total cost of the string. The selection criteria depend upon the value of string if it is close to some threshold value.
3. Create new off-spring population from two existing chromosomes in the parent population by applying crossover operator.
4. Mutate the resultant off-springs if required. NOTE: After the crossover off spring population has the fitness value higher than the parents.
5. Repeat step 3 and 4 until we get an optimal solution to the problem.


Create initial population P_i
Selection by fitness criteria
Select parent population(having fitness value close to threshold)
Criteria satisfied?
N
Stopp
N
Y
Select any two strings from population
Perform Cross-over operation on each parent
Make off-spring population from parent population

For the TSP, solution is typically represented by chromosome of length as the number of nodes in the problem. Each gene of a chromosome takes a label of node such that no node can appear twice in the same chromosome. There are mainly two representation methods for representing tour of the TSP – adjacency representation and path representation. We consider the path representation for a tour, which simply lists the label of nodes. For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5 node instance, then a tour {1 3 4 2 5 1} may be represented as (1, 3, 4, 2, 5)

Fitness function: The GAs are used for maximization problem. For the maximization problem the fitness function is same as the objective function. But, for minimization problem, one way of defining a fitness function **F (x) = 1/ f (x)** where **f (x)** is the objective function. Since, TSP is a minimization problem; we consider this fitness function, where f(x) calculates cost (or value) of the tour represented by a chromosome. Selection Process: In selection process, chromosomes are copied into next generation with a probability associated with their fitness value. By assigning to next generation a higher portion of the highly fit chromosomes, reproduction mimics the Darwinian survival-of-the-fittest in the natural world. In this paper we are using Elitism method for selection. Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

Crossover Operator: The search of the solution space is done by creating new chromosomes from old ones. The most important search process is crossover.

Firstly, a pair of parents is randomly selected from the mating pool.

Secondly, a point, called crossover site, along their common length is selected, then before crossover point we use method of sequential constructive crossover operator and the information after the crossover site of the two parent strings are swapped, if a gene has already been copied into the off-spring then replace that gene by unvisited gene, thus creating two new children. The algorithm for this new crossover technique is as follows:

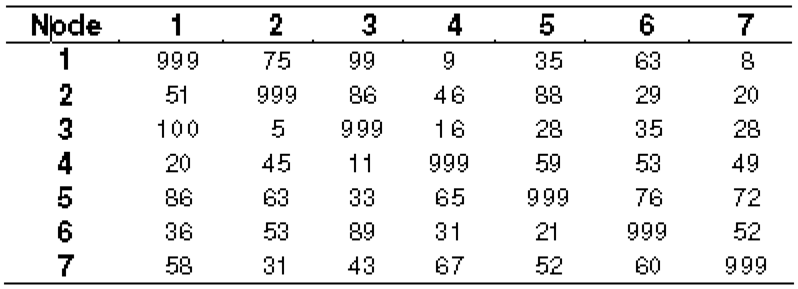
Step 1. Start from the node p(the first node in parents P1 and P2) .

Step 2. Sequentially search both of the parent chromosomes and consider the first legitimate node appeared after the node 1 in both P1 and P2 . Suppose the node x and node y are found in P1 and P2 respectively. Consider the crossover point is selected after 2nd node in both parents P1 and P2 .

Step 3. Now if Cpx < Cpy ,select node x ,otherwise node y as the next node and concatenate it to the partially constructed offspring chromosome.

Step 4. Now if we select node x as the next string in partially constructed offspring chromosome, copy the rest of the genes from parent P2,otherwise copy it from P1. **Step 5.**Suppose a gene has already been copied into the off-spring then replace that gene by unvisited gene.

Let us illustrate the SCX through the example given as cost matrix in Table 1. Let a pair of selected chromosomes be P1: (1, 5, 7, 3, 6, 4, 2) and P2: (1, 6, 2, 4, 3, 5, 7) with values 312 and 331respectively.

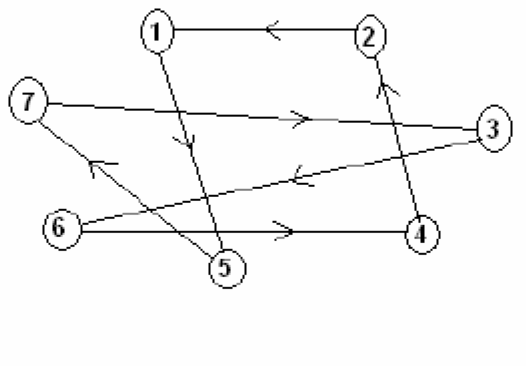


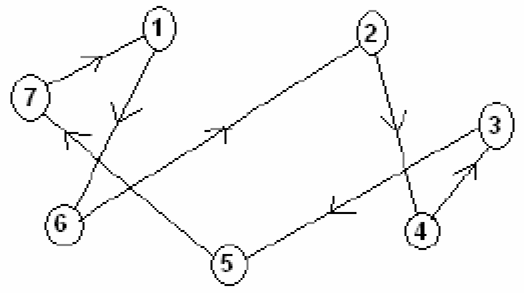
Node	1	2	3	4	5	6	7
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

TABLE 1: The cost matrix.

The cost matrix Select 'node 1' as the 1st gene. The ‘legitimate’ nodes after 'node 1' in P1 and P2 are 'node 5' and 'node 6' respectively with c15=35 and c16=63. Since c15 < c16, we accept 'node 5'. So, the partially constructed chromosome will be (1, 5). The ‘legitimate’ node after 'node 5' in both P1 and P2 is 'node 7'. So, we accept the 'node 7', and the partially constructed chromosome will be (1, 5, 7). The ‘legitimate’ node after 'node 7' in P1 is 'node 3', but none in P2. So, for P2, we consider the first 'legitimate' node in the set {2, 3, 4, 5, 6, 7}, that is, 'node 2'. Since c72 = 31 < 43 = c73, we accept 'node 2'. Thus, the partially constructed chromosome will be (1, 5, 7, 2). Again, the ‘legitimate’ node after 'node 2' in P1 is none, but in P2 is 'node 4'. So, for P1, we consider the first 'legitimate' node in the set {2, 3, 4, 5, 6, 7}, that is, 'node 3'. Since c24 = 46 < 86 = c23, we accept 'node 4'. So, the partially constructed chromosome will be (1, 5, 7, 2, 4). The ‘legitimate’ node after 'node 4' in P1 is none, but in P2 is 'node 3'. So, for P1, we consider the first 'legitimate' node in the set {2, 3, 4, 5, 6, 7}, that is, 'node 3'. We accept 'node 3', which will lead to the partial chromosome (1, 5, 7, 2, 4, 3). The ‘legitimate’ node after 'node 3' in P1 is 'node 6', but none in P2. So, for P2, we consider the first 'legitimate' node in the set {2, 3, 4, 5, 6, 7}, that is, 'node 6'. We accept the 'node 6'. Thus the complete offspring chromosome will be (1, 5, 7, 2,
4, 3, 6) with value 266 which is less than value of both the parent chromosomes. The crossover is shown in Figure 1. The parents are showing as (a) and (b), while (c) is a possible offspring.

Parents' characteristics are inherited mainly by crossover operator. The operator that preserves good characteristics in the offspring is said to be good operator. The SCX is excellent in preserving good characteristics of the parents in offspring. In Figure 1(c), bold edges are the edges which are present either in first parent or in second parent. Out of seven edges five edges are selected from either of the parents. That is, 71.4 % of edges are selected from parents. The edge (5, 7) is common in both the parents, the edges (1, 5) and (3, 6) are selected from the first parent, while the edges (2, 4) and (4, 3) are from the second parent. The edges (1, 5) and (3, 6) are the 2nd and 3rd minimum edges in the first parent, while the edges (4, 3) and (2, 4) are the 1st and 3rd minimum in the second parent. Also, the new edges (6, 2) and (7, 1) have lesser values. In addition, the SCX can generate a wide variety of offspring.


(a) P1: (1, 5, 7, 3, 6, 4, 2)


(b) P2: (1, 6, 2, 4, 3, 5, 7)

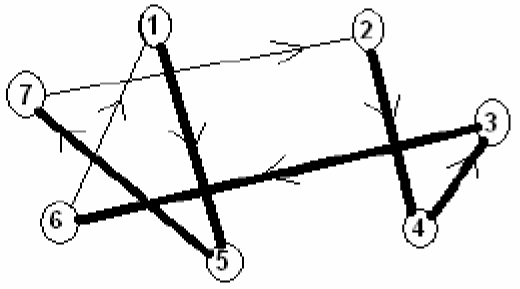

(c) Offspring: (1, 5, 7, 2, 4, 3, 6)

FIGURE 1: Example of Sequential Constructive crossover operator.

Survivor selection

After performing crossover operation survivor selection method is used for selecting next generation population. Traditionally, the survivor selection of GA considers only the fitter chromosomes. The survivor selection of GA considers two kinds of chromosomes for the next generation: (1) parents in current population of size m, and (2) offspring that are generated by crossover of size m. We consider the (μ+λ) survivor selection method that combines chromosomes in (1) and (2), sorts them in ascending order according to their fitness, and considers the first m chromosomes for the next generation. In worst case, all the μ parents in the present generation will survive into the next generation.

Mutation operator

The mutation operator randomly selects a position in the chromosome and changes the corresponding allele, thereby modifying information. The need for mutation comes from the fact that as the less fit members of successive generations are discarded; some aspects of genetic material could be lost forever. By performing occasional random changes in the chromosomes, GAs ensure that new parts of the search space are reached, which reproduction and crossover alone couldn't fully guarantee. In doing so, mutation ensures that no important features are prematurely lost, thus maintaining the mating pool diversity. For the TSP, the classical mutation operator does not work. For this investigation, we have considered the reciprocal exchange mutation that selects two nodes randomly and swaps them.

Control parameters

These are the parameters that govern the GA search process. Some of them are:

(a) Population size: - It determines how many chromosomes and thereafter, how much genetic material is available for use during the search. If there is too little, the search has no chance to adequately cover the space. If there is too much, the GA wastes time evaluating chromosomes.

(b) Crossover probability: - It specifies the probability of crossover occurring between two chromosomes.

(c) Mutation probability: - It specifies the probability of doing bit-wise mutation.

(d) Termination criteria: - It specifies when to terminate the genetic search.

Structure of genetic algorithms

GAs may be summarized as follows:

```
GA( )
{ Initialize random population;
Evaluate the population;
Generation = 0;
While termination criterion is not satisfied
{ Generation = Generation + 1;
Select good chromosomes by reproduction procedure;
Perform crossover with probability of crossover (Pc);
Select fitter chromosomes by survivor selection procedure;
Perform mutation with probability of mutation (Pm);
Evaluate the population;
}
}
```

Computational Experiments

The following common parameters are selected for the algorithms: population size is 200, probability of crossover is 1.0 (i.e., 100%), probability of mutation is 0.01 (i.e., 1%), and maximum of 10,000 generations as the terminating condition. The experiments were performed 10 times for each instance. The solution quality is measured by the percentage of excess above the optimal solution value reported in TSPLIB website, as given by the formula.

$$Excess(\%) = \frac{Solution\ Value - Optimal\ Solution\ Value}{Optimal\ Solution\ Value} \times 100.$$

VII. RESULT

Here we have shown an example of 7 cities graph through a matrix representation and on the basis of our calculation we have reached on an experimental result that our new crossover operator is capable of calculating an approximately optimal path for TSP using genetic algorithmin less time . If we look at the sequential constructive crossover(SCX) than our new crossover technique is better in terms of cost and time . In sequential constructive crossover (SCX) there is a sequential search for each gene in a chromosome and hence comparing it with another chromosome parent genes thereby resulting in a child chromosome. This task is time consuming and also a higher cost is occurring. In our cross overmethod, there is only a single search for a crossover point and after that the child chromosome is developed using the single point crossover technique only. This is leading to less time and cost and hence giving a better optimal solution for the given TSP problem.

Algorithm	**Time**	**Complexity**	**Advantage Disadvantage**	**Disadvantage**
GGenetic AlAgorithm	Expon- ential time	O(Kmn)	Best solution Using “fitness criteria” no optimal solution	Approximation of solution is reached but not an optimal solution.
GGreedy AApproach	5 seconds for 15 cities	O(logn)	Fastest no Solution accuracy	No best choices are consideredand hence no accuracy is achieved .
DDynamic PProgramm mi ng	9 seconds for 15 cities	$O(n^2 2^n)$	Global optimal Solution expensive for both memory and time	Expensive for both memory and time

VIII. CONCLUSION AND FUTURE WORK

Genetic algorithm appear to find good solutions for the Travelling Salesman Problem,however it depends very much on the way the problem is encoded and which crossover and mutation methods are used .We have proposed a new crossover operator named for a genetic algorithm for the Traveling Salesman Problem (TSP). Among all the operators, experimental results show that our proposed crossover operator (SCX) is better in terms of quality of solutions and cost as well as solution times. That is why, we used a local search technique to improve the solution quality. Also, we set here highest probability of crossover to show the exact nature of crossover operator. Mutation with lowest probability is applied wherever required only. We presented a comparative study among Greedy approach, Dynamic programming and Genetic Algorithm for solving TSP.It is very difficult to say that what moderate sized instance is unsolvable exactly by our crossover operator. So an incorporation of good local search technique to the algorithm may solve exactly the other instances, which is under our investigation and is categorized under future work.

Genetic algorithms appear to find good solutions for the traveling salesman problem, however it depends very much on the way the problem is encoded and which crossover and mutation methods are used. It seems that the methods that use heuristic information or encode the edges of the tour (such as the matrix representation and crossover) perform the best and give good indications for future work in this area.

Overall, it seems that genetic algorithms have proved suitable for solving the traveling salesman problem. As yet, genetic algorithms have not found a better solution to the traveling salesman problem than is already known, but many of the already known best solutions have been found by some genetic algorithm method also.

REFERENCES

[1] http://en.wikipedia.org/wiki/Genetic_algorithm

[2] 2 J. Holland, Adaptation in Natural and Artificial Systems : An Introductory Analysis with applications to biology , Control and Artificial Intelligence.‖ The University of Michigan Press,1975.

[3] http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php .

[4] Jean-Yves Potvin , “Genetic Algorithms for the Traveling Salesman Problem” Centre de Recherche sur les Transports Université de Montréal C.P. 6128, Succ. A Montréal (Québec) Canada H3C 3J7

[5] Goldberg, Koza, Michalewicz and Beasley ”Potvin ,Jean-Yves ”new optimization technique for genetic algorithms” (Addison-Wesley, 1989)

[6] Monica Sehrawat, Sukhvir Singh,”Modified Order Crossover (OX) Operator ”

International Journal on Computer Science and Engineering (IJCSE) ISSN 0975-3397 Vol. 3 No. 5 May 2011.

[7] Naveen kumar, Karambir and Rajiv Kumar, "A Genetic Algorithm Approach To Study Travelling Salesman Problem", Journal of Global Research in Computer Science, 2012, Vol. 3, No. (3).

[8] Saloni Gupta,Poonam Panwar "Solving Travelling Salesman Problem Using Genetic Algorithm" International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6, June 2013.

[9] Varunika Arya, Amit Goyal and Vibhuti Jaiswal "An Optimal Solution To Multiple Travelling Salesperson Problem Using Modified Genetic Algorithm" International Journal of Application or Innovation in Engineering & Management (IJAIEM) Volume 3, Issue 1,2014

[10] S.N.Sivanandam , S.N.Deepa "Introduction to Genetic Algorithms" Springer-Verlag Berlin Heidelberg 2008