Traveling Salesman Problem using Genetic Algorithm

Difficulty Level : Hard • Last Updated : 02 Jul, 2021

Prerequisites: Genetic Algorithm, Travelling Salesman Problem

In this article, a genetic algorithm is proposed to solve the travelling salesman problem.

Genetic algorithms are heuristic search algorithms inspired by the process that supports the evolution of life. The algorithm is designed to replicate the natural selection process to carry generation, i.e. survival of the fittest of beings. Standard genetic algorithms are divided into five phases which are:

1. Creating initial population.
2. Calculating fitness.
3. Selecting the best genes.
4. Crossing over.
5. Mutating to introduce variations.

These algorithms can be implemented to find a solution to the optimization problems of various types. One such problem is the Traveling Salesman Problem. The problem says that a salesman is given a set of cities, he has to find the shortest route to as to visit each city exactly once and return to the starting city.

Approach: In the following implementation, cities are taken as genes, string generated using these characters is called a chromosome, while a fitness score which is equal to the path length of all the cities mentioned, is used to target a population.

Fitness Score is defined as the length of the path described by the gene. Lesser the path length fitter is the gene. The fittest of all the genes in the gene pool survive the population test and move to the next iteration. The number of iterations depends upon the value of a cooling variable. The value of the cooling variable keeps on decreasing with each iteration and

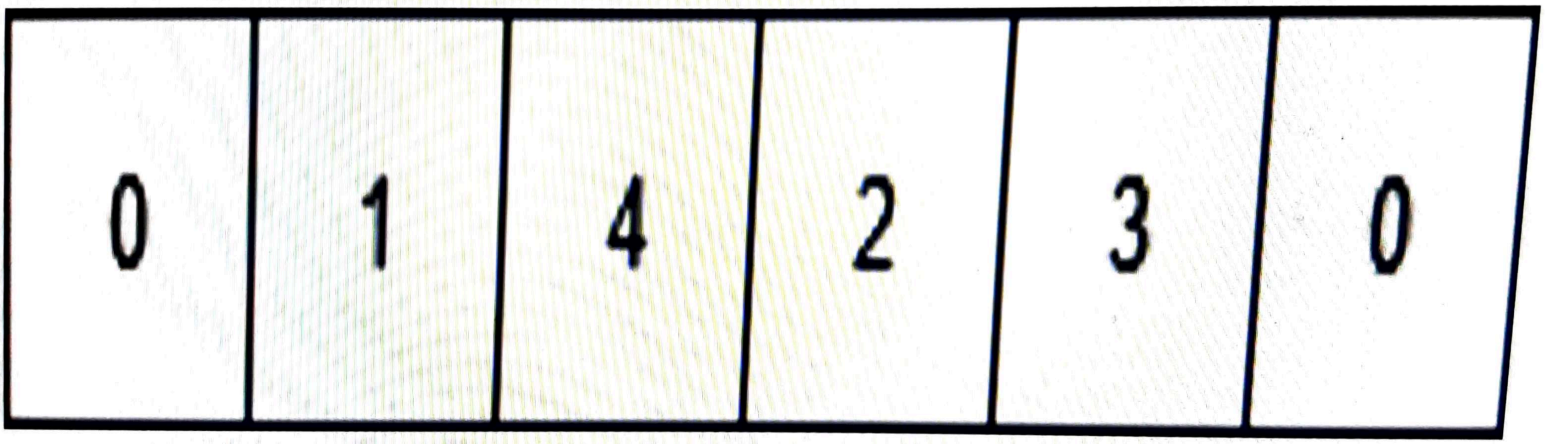
depends upon the value of a cooling variable. The value of the cooling variable keeps on decreasing with each iteration and reaches a threshold after a certain number of iterations.

Algorithm:

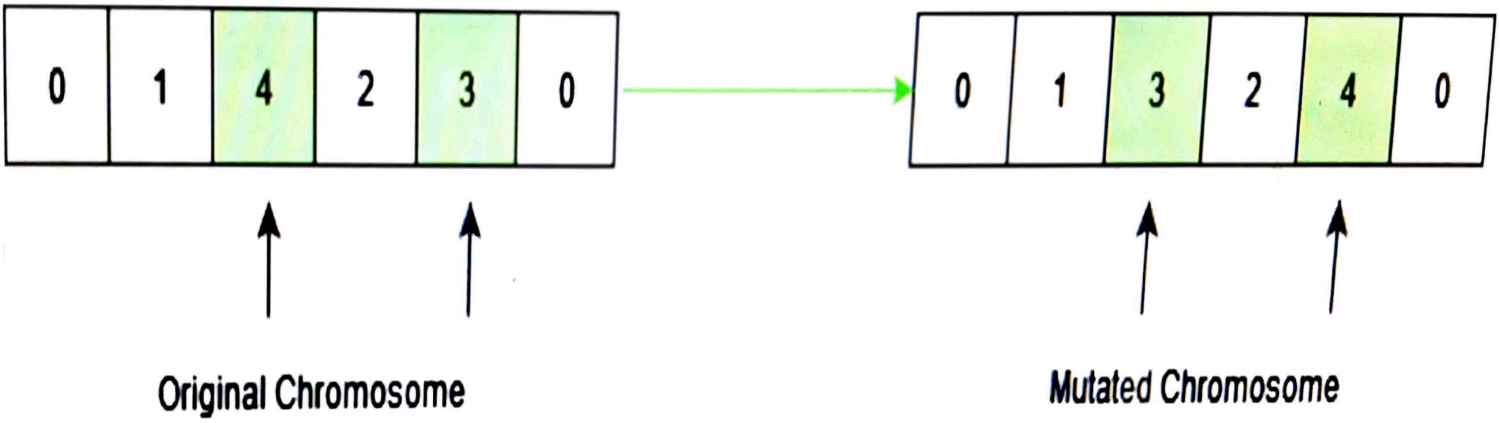
1. Initialize the population randomly.
2. Determine the fitness of the chromosome.
3. Until done repeat:
    1. Select parents.
    2. Perform crossover and mutation.
    3. Calculate the fitness of the new population.
    4. Append it to the gene pool.

How the mutation works?

Suppose there are 5 cities: 0, 1, 2, 3, 4. The salesman is in city 0 and he has to find the shortest route to travel through all the cities back to the city 0. A chromosome representing the path chosen can be represented as:



This chromosome undergoes mutation. During mutation, the position of two cities in the chromosome is swapped to form a new configuration, except the first and the last cell, as they represent the start and endpoint.



Original chromosome had a path length equal to **INT_MAX**, according to the input defined below, since the path between city 1 and city 4 didn't exist. After mutation, the new child formed has a path length equal to **21**, which is a much-optimized answer than the original assumption. This is how the genetic algorithm optimizes solutions to hard problems.

17.3 Optimization of Traveling Salesman Problem using Genetic Algorithm Approach

The traveling salesman problem (TSP) is conceptually simple. The problem is to design a tour of a set of n cities in which the traveler visits each city exactly once and then returns to the starting point. One has to minimize the distance traveled. Solving this combinatorial problem is NP (nondeterministic polynomial time) difficult as the search space is n factorial. Examining all possible Hamiltonian circuits of n vertices by calculating edge weights is certain to reveal the optimum solution, but cannot guarantee to do so in a tractable time for all n.

Perhaps first considered by Euler as the knights' tour problem in 1759, and popularized by the RAND Corporation in the 1950s, TSP has many applications that involve large numbers of vertices. These include VLSI – for which size can exceed one million – circuit board drilling, X-ray crystallography and many

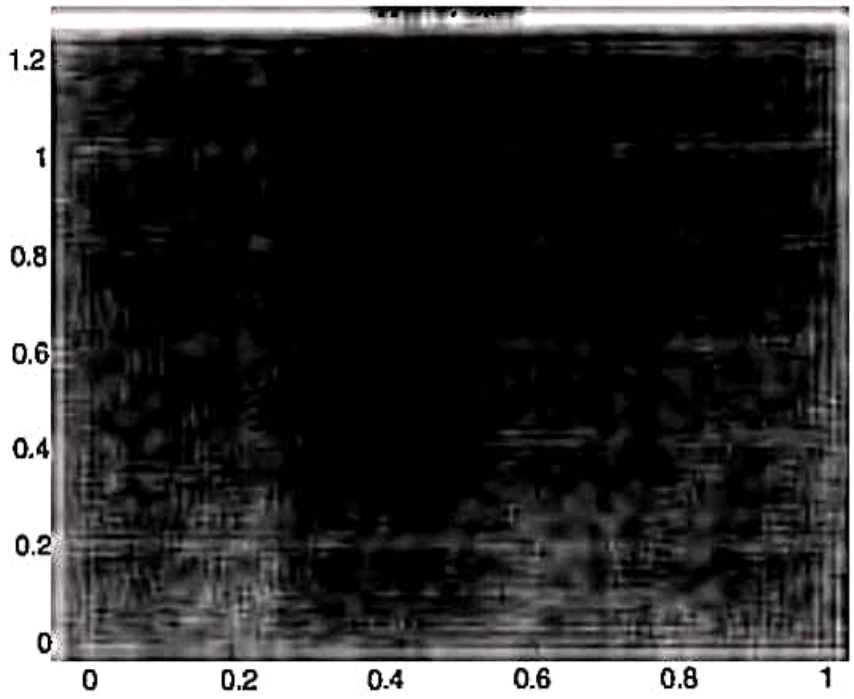


Figure 17-6 Points placed in a unit square representing a landscape of 14 cities.

scheduling problems. Therefore it is important to find algorithms that lower the time costs by providing reasonably good solutions.

This section explores application of GAs to TSP by examining combinations of different algorithms for the binary and unary operators used to generate better solutions and minimize the search space. Three binary and two unary operations were tested. These were compared to a base-line developed from a brute force algorithm for a tractable 14-city problem and to random tour generation, which provides mean and standard deviation statistics very close to brute force methods as the distribution of solutions is normal. For a 14-city problem, Figure 17-6 shows the points placed in a unit square representing a landscape of 14 cities.

The binary methods examined include uniform order-based crossover (OCX), heuristic order-based crossover (HCX) and edge recombination (ER). Unary operators were reciprocal exchange and inversion.