Genetic Algorithm for Solving the Travel Salesman Problem

By:

Bhargav Dineshbhai Patel

Gaurav Pathak

Harshkumar Mehta

Mohasina Shaikh

Niraliben Dipakkumar Mistry

Vikaskumar Chaudhary

CPSC-5506 Project

Research Report submitted in Partial Fulfilment

of the Requirements of the Master’s Course

in Computational Science

(Introduction to Computational Science, CPSC-5506)

Department of Mathematics and Computer Science

Faculty of Sciences, Engineering and Architecture

Laurentian University

Sudbury, Ontario, Canada

Winter 2021

Table of Contents

[List of Figures II](#_Toc68029862)

[Abstract III](#_Toc68029863)

[Keywords III](#_Toc68029864)

[CHAPTER 1: THE TRAVEL SALESMAN PROBLEM 1](#_Toc68029865)

[1.1 Introduction 1](#_Toc68029866)

[1.2 The Travel salesman Problem 1](#_Toc68029867)

[1.2.1 Genetic Algorithm 2](#_Toc68029868)

[1.3 Extension of the Travel Salesman Problem 5](#_Toc68029869)

[1.3.1 Quantum Computing 5](#_Toc68029870)

[1.3.2 The Proposed Algorithm 6](#_Toc68029871)

[1.4 Conclusion for the Extension of the Travel Salesman Problem 7](#_Toc68029872)

[CHAPTER 2: LITERATURE REVIEW OF THE TRAVEL SALESMAN PROBLEM 8](#_Toc68029873)

[2.1 Introduction 8](#_Toc68029874)

[2.2 Solution methods for the Travel Salesman Problem 9](#_Toc68029875)

[2.2.1 Method of total enumeration 9](#_Toc68029876)

[2.2.2 Method of branches and bounds 9](#_Toc68029877)

[2.2.3 Efficient algorithm of Clarke and Wright 10](#_Toc68029878)

[2.2.4 Computer Simulation 10](#_Toc68029879)

[2.2.5 Ant Colony Optimization Algorithm (ACO) 11](#_Toc68029880)

[2.2.6 Particle Swarm Optimization Algorithm (PSO) 12](#_Toc68029881)

[2.2.7 Genetic algorithms 12](#_Toc68029882)

[2.3 Conclusion 14](#_Toc68029883)

[CHAPTER 3: GENETIC ALGORITHM FOR THE TRAVEL SALESMAN PROBLEM 15](#_Toc68029884)

[3.1 Introduction 15](#_Toc68029885)

[3.2 Genetic Algorithm 15](#_Toc68029886)

[3.3 Numerical Experiments 18](#_Toc68029887)

[3.4 Conclusion 19](#_Toc68029888)

[Bibliography 20](#_Toc68029889)

# List of Figures

[Figure 1: Representation of Population Chromosome and Genes 7](#_Toc68026906)

[Figure 2: Distance matric of a TSP instance 9](#_Toc68026907)

[Figure 3: The proposed algorithm 10](#_Toc68026908)

[Figure 4: A TSP with 5 cities and distances 19](#_Toc68026909)

[Figure 5: Illustration of crossover with parents and its offspring 21](#_Toc68026910)

# **Abstract**

This research report is the result of implementation of travelling salesman problem (TSP) using genetic algorithm (GA)in python carried out by the authors. TSP is one of the most intensively studied problem in optimization. The main attraction of TSP is a salesman visiting all the cities in his tour at the least possible cost. In genetic algorithms crossover and mutation are the preferred technique to solve the optimization problem using survival for the fittest idea.The implementation can solve the travelling salesman problem up to 29 cities in < 2 minutes on a standard testbed with 8GB of RAM. For our experimental investigation, results have shown that genetic algorithms lead to a good optimization as high as 70 percent even with less population in consideration.

# Keywords

Travelling Salesman Problem, Genetic Algorithms, Path Representation, Optimization, Shortest distance.

# CHAPTER 1: THE TRAVEL SALESMAN PROBLEM

## 1.1 Introduction

Travelling salesman problem are known classical permutation based combinatorial optimisation problems which has been extensively studied over past few decades. This program requires a colossal amount of system resources to be solved efficiently, as when the size of a problem increases the solution space also increases exponentially. The problem objective is to find the shortest route for the travelling salesman who, starting from his home city must visit every city given on the list precisely once and then return to his home city this problem is mainly focused to find the shortest such trip. The city visit ends back at the starting city, this problem is known as NP-hard as it cannot be solved in polynomial time [1, 2]. The coordinate of the city are known in advance, in order to find the pairwise distance between the cities.The main difficulty is the immense number of possible tours for n cities: (*n*-1)! /2[3]

## 1.2 The Travel salesman Problem

Travelling salesman problem is relatively an old problem.The idea of this problem was introduced as early as 1759 by Euler like TSP where a knight visits each square of chessboard exactly once in his tour. Although the term ‘*travelling salesman*’ coined in early 1930’s in a German book written by a travelling salesman [3].

Over the years TSP has occupied the interest of numerous researchers. The reason for this is the lack of a polynomial time algorithm to resolve the problem Although there are several techniques available to solve the travelling salesman problem [4, 5], but these techniques are not optimized.TSP is applicable on a variety of routing and scheduling problems [6]. Multiple heuristic approacheshave been developed to solve TSP as described in [7]. Using genetic algorithm the first researcher to tackle the travelling salesman problem was Brady [8]. The genetic algorithm provided by researchers to solve the travelling salesman problem up to 531 cities have provided very good results but the solution was not optima [9, 10].

Recently there is increasingly many reasons now to believe that TSP is very hard [11]. There is evidence that there is no polynomial time algorithm for obtaining the exact solution even if the distance is restricted [12] and a solution for guaranteed accuracy [13]. The problem is verify whether the solution is optimal exactly or approximately is also seems to be intractable [11].

### 1.2.1 Genetic Algorithm

Genetic Algorithm are adaptive search technique based on principal and mechanism of natural selection and the survival of the fittest. GA gained popularity from Holland’s study in 1975 [14] of adaption in artificial and natural systems in search problems. In recent years numerous papers have been published on the optimization of NP-hard problems in different application domains such as computer science, biology, telecommunication. GA operate on an iterative fixed size population or pool of candidate solution. The candidate solution represents an encoding of the problem which is like the chromosomes of a biological system.

Each chromosome is associated with the fitness value. It is the ability of chromosome which determine the ability to survive and further produce an offspring. Space search problem are represented as ‘*individuals*’ which are represented by character of strings referred as ‘*chromosomes’*. Integer and floating point can also be used [15].

Part of space search which is to be examined is called ‘*population*’. Genetic algorithm working described in (Figure 1). To start, an initial population is chosen along with this the quality of population is determined and then evaluating everyone with fitness function. In further iterations parents are selected from the population and in turn these parents produce children which are further added back to the population. Offspring are generated through a process called crossover and mutation.It can further be defined as the operations which define the child production process and mutation process are known as crossover operator and mutation operator.

Offspring are generally placed back into the population thus replacing other individuals. Mutation helps algorithm to explore the new stats by avoiding the local optima[3]. Crossover increases the average quality of the population thus by choosing the adequate crossover and mutation operators, the probability that genetic algorithm will produce the nearly optimal solution will increases with respect to the increased number of iterations. GA algorithm relies on three genetic operators: *selection, crossover, mutation.* The selection operation use the fitness value to select the parents of next generation [15].

**BEGIN** AGA

Make initial population at random.

**WHILE NOT** stop **DO**

**BEGIN**

*Select parents* from the population.

*Produce children* from the selected parents.

*Mutate* the individuals.

*Extend* the population adding the children to it.

*Reduce* the extend population.

**END**

Output the best individual found.

**END** AGA

*Figure 1*. The pseudo-code of the Abstract Genetic Algorithm (AGA) [3].

We generate the finite set of individuals which we called *‘population’.* The size of population set is predetermined before applying the genetic algorithm procedure. An individual characterised by the set of variables is known as ‘*gene*’. We calculate the fitness of everyone which is commonly done by calculating the sum of Euclidean distance between cities in the solution. During selection process the initial population get chosen arbitrarily among the possible individuals.



*Figure 1: Representation of Population Chromosome and Genes*

Gene is joined to form a set of string usually known as chromosome depicted in figure 2. In genetic algorithm the fitness is defined by using a fitness function, it determines how fit is an individual to compete with other individuals by assigning a fitness score to everyone. The probability of selecting individual based on fitness score highlights that individual is selected for reproduction.The selection phase usually selects the individual who are fittest so that their genes can be passed to next generation. The classical crossover operation was proposed by Holland in 1975 [14], as shown below where two solutions of 6 cities are available for travelling salesman problem:

(000 001 010 011 100 101) and

(101 100 011 010 001 000)[3]

Randomly among the strings a crossover point is selected from where the string is broken into two separate parts, considering we have chosen the below crossover point highlighted with pipe.

(000 001 010 | 011 100 101) and

(101 100 011 | 010 001 000)[3]

After recombining the parts result in two separate offspring:

(000 001 010 010 001 000) and

(101 100 011 011 100 101)[3]

The mutation operator which was developed by Holland [14] alters one or more bit with the probability equivalent of mutation rate. A tour represented by string 1-2-3-4-5-6:

(000 001 010 011 100 101)

Consider the last and second last bits are selected for mutation, hence these bits will change its value from 0 to 1 and 1 to 0:

(000 001 010 011 100 110)

## 1.3 Extension of the Travel Salesman Problem

We can extend the problem using different approaches or methods to find out best possible solution of the TSP.

### 1.3.1 Quantum Computing

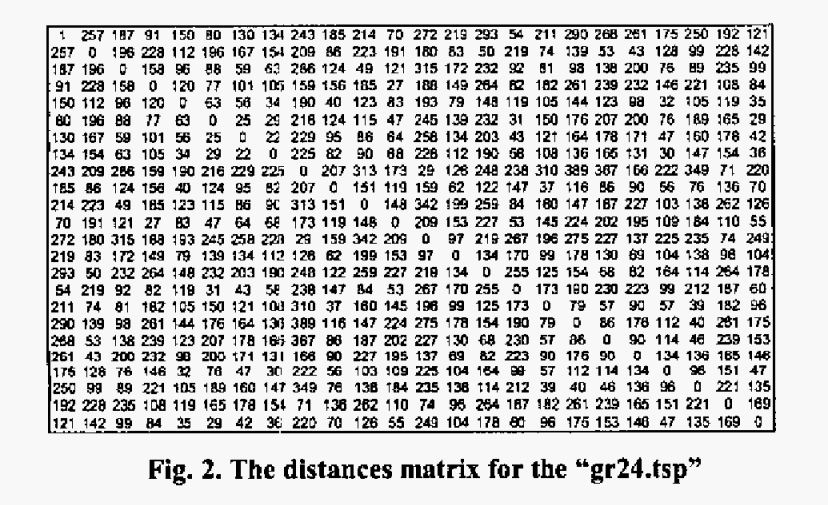
In early 80, Richard Feynman's observed that certain quantum mechanical effects cannot be simulated efficiently on a computer. His observation led to speculation that computation in general could be done more efficiently if it used this quantum effects. This speculation proved justified in 1994 when Peter Shor described a polynomia1 time quantum algorithm for factoring numbers.

In quantum systems, the computational space increases exponentially with the size of the system which enables exponential parallelism. This parallelism could lead to exponentially faster quantum algorithms than possible classically [16].

### 1.3.2 The Proposed Algorithm

We introduce here a new algorithm inspired from both genetic programming and quantum computing fields to find the shortest Hamiltonian circuit relating N cities. The symmetry of the problem has no special importance. The algorithm deals indifferently with symmetric and asymmetric instances of the TSP.

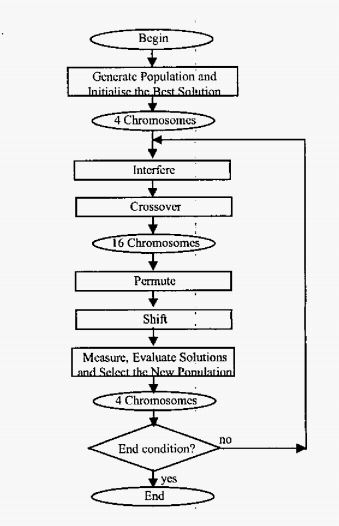
The algorithm has as input data the distances between each pair of cities. These distances are arranged within a square matrix D of NxN element. The element D[ i, j] denotes s the distance between the city labelled i and the one labelled j. The figure below gives the distances matrix of a TSP instance (“gr24” [16]).



*Figure 2: Distance matric of a TSP instance*

A solution for a TSP dealing with N cities is a circuit which relates in a suitable order these cities. So we can represent the solution by an NxN matrix “A” associating to every city its range in the circuit, i.e. if A (i, j) = 1, j is the ith visited city. Elsewhere if d (i, j) = 0. The figure below gives a representation of the best solution for the problem above.

Starting with the initial population, we apply cyclically a set of4 operations followed by a measurement (figure 3)



*Figure 3: The proposed algorithm*

There are some other approaches to find out efficient solution for the travel salesman problem. The approaches are mentioned below:

I) An analogue approach to the travelling salesman problem using an elastic net method

II) The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem

III) The travel salesman problem using the Brute-Force approach (Naive approach) which uses nearest neighbour method

## 1.4 Conclusion for the Extension of the Travel Salesman Problem

We have suggested a new algorithm inspired from both genetic algorithms and quantum computing to solve the travelling salesman problem as a representative of combinatorial optimisation problems class. Our algorithm provides a great diversity by using quantum coding of solutions, i.e. all the solutions exist within each chromosome and what change are the probabilities to have one of them as a result of a measurement. Therefore, the size of the population does not need to be great. So, we have chosen to have only 4 chromosomes at the origin of each generation. Another advantage is that the interference provides in some way a guide for the population individuals and reinforces therefore the algorithm convergence. This has allowed obtaining good solutions after a small number of iterations. Introducing permutation and shifting operations has improved the algorithm’s performance by permitting it to avoid been blocked in local minima.

# CHAPTER 2: LITERATURE REVIEW OF THE TRAVEL SALESMAN PROBLEM

## 2.1 Introduction

The Travelling salesman problem results in more than one solution, but the aim is to find the best solution in a decreased time and the performance is also improved. So, the heuristic genetic algorithm is used. The optimal solution choses the route that minimizes the total distance travelled. Traveling Salesman Problem [17, 18] is an optimization problem and has a vast search space and is said to be NP-hard, which means it cannot be solved in polynomial time. The Pictorial and mathematical structure of the TSP is a graph in which the nodes, edges, vertices etc. are termed as the attributes. If the nodes are the cities available in the problem. Edges are the vectors connecting the pair of cities and each edge has a cost associated with it which can be distance, time or other attribute. If n is the input number of vertices representing cities, for a weighted graph G, the TSP problem is to find the cycle of minimum costs that visit each of the vertices of G exactly once. There are (n – 1)! possible tours.

Let us consider an example describing the Traveling salesman problem. We have a set of five cities A, B, C, D and E. The travelling salesman problem has to find the minimized distance cycle that starts from some specified vertex, visits all other vertices exactly once return to the same specified vertex.

Example 1 illustrates the collection of the cities and their distances among each other. Here (5-1)! that is 4! route can be generated. The tour with A→B→C→D→E→A will be the optimal route for given problem.

The travelling salesman problem can be classified as:

1. STSP: In STSP the distance between two cities is same in both the directions that mean this will result in an undirected graph.
2. ATSP: In ATSP the distance between two cities is not same in both directions. It is a directed graph and distance is different in both the directions.
3. MTSP: In each set of nodes, let there be ‘m’ salesmen located at a single depot node. The remaining nodes (cities) that are to be visited are intermediate nodes. Then, the MTSP finds the tours for all ‘m’ salesmen, who all start and end at the place, such that each intermediate node (city) is visited exactly once and the total time of visiting all nodes is minimized.

Many heuristic techniques have been used to find the efficient solution to the problem like greedy method, ant algorithms, simulated annealing, tabu search and genetic algorithms [19]. But as the number of cities increases the computation to find the solution becomes difficult. Despite the computational difficulty, we can use methods like genetic algorithms and tabu search which can give near to optimal solution for thousands of cities. In this paper we provide overview of different approaches used for solving travelling salesman problem.

## 2.2 Solution methods for the Travel Salesman Problem

Literature cites an entire range of TSP methods differing in various styles to the solutions, efficiency of the methods and the results. Let us quote brief characteristics of the most often used ones.

### 2.2.1 Method of total enumeration

In principle it is a combinatorial solution. The method rests in evaluation of all potential routes (sequences) in the total number of (n – 1)! . The advantage is that a global optimum is always found, however, it is not employable if higher numbers of visited places are considered. With every added element (node) the number of possible solutions grows exponentially and not even nowadays do we have computers powerful enough for being able to provide optimum solution within reasonable time [20].

### 2.2.2 Method of branches and bounds

This method belongs to the oldest ones and the most frequently used algorithms for the TSP solutions. The merit of the method rests in a gradual decomposition of a possible solution set into several mutually disjunctive subsets labelled as branches. In each step the following is estimated:

● The upper limit of the objective function that is most often the value of the objective function zH without respecting limits and

● Maximum lower bound of an objective function zD of acceptable solutions which are known to us within the step.

Both the estimates can be employed for seeking non-prospective directions of further procedures: if for any branch zH < zD, then the given direction can be excluded. However, this method is also, especially for higher n, too laborious and does not always guarantee an optimum solution at the first attempt [21].

### 2.2.3 Efficient algorithm of Clarke and Wright

A significant progress in TSP solutions was provided by the Clarke’s and Wright’s method. The initial situation assumes that each place is supplied separately and always a return to the starting base follows. The basic idea is based on the calculation of economies achieved through integrating other places into the circular route. An indisputable asset of this algorithm is its function to respect further restrictions often generated by the practice, e.g., the need to optimize more orbital routes, to use more vehicles while respecting their various capacities etc.

Guerra, Murino and Romano worked with this algorithm for optimize the routing phase in a Location-Routing Problem (LRP) in [22]. LRP can by conform to a Vehicle Routing Problem (VRP) and after that they combine and balance VRP with TSP. Both problems were solved with Clarke and Wright saving algorithm and the Branch and Bound model.

### 2.2.4 Computer Simulation

The development of simulation models, their program support and increasing computing power brought about attempts at use of simulation techniques for solving large TSPs. Their value rests in random PC sampling in large scale that is later evaluated according to a selected objective function. Though the solution does not guarantee the global optimum, a sufficiently large number of simulations will issue in achieving the best possible solution, the value of which will be close to the optimum [23].

### 2.2.5 Ant Colony Optimization Algorithm (ACO)

ACO is one of the metaheuristic methods for solving TSP. Jalali, Afshar and Marino [24] refer to ACO as observation of real ants, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. There is a higher probability that the trail with a higher pheromone concentration will be chosen. The pheromone trail allows ants to find their way back to the food source and in the opposite way. The trail is used by other ants to locate food source discovered by any ant. When several paths available from the nest to a food source, a colony of ants may by able to exploit the pheromone trail left by individual members of the colony to discover the shortest path from the nest to the food source and back. As more ants choose a path to follow, the pheromone on the path builds up, making it more attractive to be followed by other ants. To solve TSP, we keep the strength of pheromone trail Tij for each combination of two points. The role of each ant is to find a valid solution, thus possible routes. From the starting point, the ant gradually repeats a move when it chooses a place where it has not been yet while moving to it from its current location. Once there are no more vacancies left, the ant returns to the starting location. As a result, the ant keeps its path T. If the ant k is currently at the location i, then the probability that it goes to the city j is

Text

Description automatically generated with low confidence

Where Tkij is the total pheromone deposited on path ij, ηk ij is the heuristic value of path ij according to the measure of the objective function (a priori knowledge, typically 1/cij, where cij is distance). α, β are parameters that control the relative importance of the pheromone trail versus heuristic value. When all the ants have completed a solution, the trails are updated by



where p is the pheromone evaporation coefficient.

Ant colony optimization algorithms have been used to produce near-optimal solutions to the travelling salesman problem. The first ACO algorithm was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities. It is able to find the global optimum in a finite time.

### 2.2.6 Particle Swarm Optimization Algorithm (PSO)

PSO proceed from the social behaviour of organisms such as bird flocking and fishing schooling. Through cooperation between individuals, the group often can accomplish their goal efficiently and effectively. PSO simulates this social behaviour as an optimization tool to solve some optimization problems. Each particle flies in the search space with a velocity that is dynamically adjusted based on its own flying experience and its companions’ flying experience. In other word, every particle will utilize both the present best position information of its own (pbest) and the global best position information (gbest) that swarm has searched up-to now to change its velocity and thus arrives in the new position [28].

### 2.2.7 Genetic algorithms

In recent years there have been attempts to use so called genetic algorithms for TSP solutions. Simply stated, genetic algorithms transfer evolution principles in living organisms into intelligent searching and model optimization in other fields. Biological terminology is applied also to this very description of the algorithm. Genetic algorithm, as well as nature, works with population of individuals (P) defined by one or more mathematical genes – chromosomes (i.e., sequences of numbers in binary notation).

**The genetic algorithm (GA) uses the following steps:**

**(a) Initial Population Generation:** The GA randomly samples values of the changing cells between the lower and upper bounds to generate a set of (usually at least 50) chromosomes. The initial set of chromosomes is called the population. The genetic algorithm starts by initialization of population. The initial population is generated randomly by the algorithm. It will encode all possible solutions for the problem. The initial population can be of any size.

**(b) Fitness Evaluation:** The fitness evaluation phase assigns a fitness value for each individual solution which is produced in the previous step. The fitness value can be calculated in many ways. Based on the user requirement the fitness value is calculated. One common method is given by the equation F = 1/f, where ‘F’ is the fitness value and ‘f’ is the total path length of the individual. The fitness value shows how fit the chromosome is. In the new generation, chromosomes with a smaller fitness function (in a minimization problem) have a greater chance of surviving to the next generation.

**(c) Parent Selection:** The parent selection phase selects the fit parents for the further processing of genetic algorithm. This phase observes the fitness values of each individual and selects the individual with best fitness value of the next phase. There are many parent selection methods such as Elitism method, Roulette Wheel method, Tournament selection method and so on. Different selection methods select a different parent, and it is given to the crossover process.

**Crossover:** This is the important stage in genetic algorithms. The result of genetic algorithm mainly depends on how efficient the crossover method is. So, care must be taken while selecting the crossover method. The crossover phase takes two parents and combines them to produce the new child solution. The child solution is also known as the offspring. Crossover can be one point, or two-point crossover and it may produce the offspring which has the same edges as the parent, or it may have some new edges which are not present in the parent solutions.

**Mutation:** Mutation operator is applied to maintain the uniqueness between the chromosomes. It will make some changes in the solution so that it can generate new values which are not already present and may produce better results. Mutation operator does the modifications to the solution by swapping the values of a solution.

**Stopping conditions:** At each generation, the best value of the fitness function in the generation is recorded, and the algorithm repeats step 2. If no improvement in the best fitness value is observed after many consecutive generations, the GA terminates.

The genetic algorithm is complete if all the above genetic algorithm operators are applied. Genetic algorithm optimizes the initial solution to produce better results for a problem. Mapping of Genetic Algorithm to TSP The main focus of the travelling salesman problem is to find the shortest path to travel through the given cities and to minimize the time.

More authors propose new modifications and improvements of GA. For example, an effective parallel model was presented from Bai Xiaojuan and Zhou Liang in [29]. It is based on the traditional genetic algorithm, but a new operating mechanism of GA was improved means of adaptive crossover and mutation. It uses probability of GA, which can keep the solution space effective. Further, 2-opt neighbourhood search optimization techniques are imported, which can ensure the evolution process is not stagnation, and improve the efficiency of solving. Another improving proposals of Genetic Algorithm were presented in [30, 31]. The interest of many authors about GA proved to be the promising and effective solving method for the specific group of optimization models.

## 2.3 Conclusion

The travelling salesman problem is used in many application areas like planning, logistics, manufacturing of microchips, DNA sequencing, vehicle routing problems, robotics, airport flight scheduling, time and job scheduling of machines. In this paper we provide overview of different approaches used for solving travelling salesman problem. The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem, which is simple to state but very difficult to solve. The problem is to find the shortest tour through a set of N vertices so that each vertex is visited exactly once. This problem is known to be NP-hard and cannot be solved exactly in polynomial time. Many exact and heuristic algorithms have been developed in the field of operations research (OR) to solve this problem. In this paper we study Genetic algorithm approach, Ant Colony Optimization Technique, Particle Swarm Optimization method & branch and bound algorithm to solve the travelling salesman problem.

# CHAPTER 3: GENETIC ALGORITHM FOR THE TRAVEL SALESMAN PROBLEM

## 3.1 Introduction

Genetic algorithms (GAs) are stochastic approach which does not rely on derivative based modelling (no-model approach). It is a simple yet way more effective algorithm designed based on the idea of biological evolution of the species. Core idea behind this algorithm is that the most suitable individuals are likely to survive and mate; which will result into better generation & that will be more healthy and fitter than their parents. GAs work with population of chromosomes that are represented by some underlying parameters set codes.

As species passes their genes into their next generation to make species capable to survive their life even better than them, same idea that have been used in genetic algorithm which brings the biological terms into it i.e., chromosome in the context of algorithm: a sample selection procedure from the available population, crossover is the way to generate child of the parents which means from two parents one next generation will be created, mutation is the way to mutate the current samples with other chromosome which helps to generate more accurate results from the sample & finally evaluation and selection; is to find the best mates out of selected samples.

## 3.2 Genetic Algorithm

As we have already introduced the terms like chromosome, crossover and mutation, in this topic we will explain in-depth approach of this algorithm to solve TSP (Traveling Salesman Problem).

**Algorithm flow in the context of TSP:**

Step-1 Randomly generate initial population.

Step-2 Based on the cost function, evaluate all individual chromosome.

Step-3 If generation limit exceeded, then go to step 5

Step-4 a) Perform Selective reproduction,

b) Do Crossover,

c) Do Mutation, then, Go to Step 3

Step-5 Select the best individual from the result and display.

**Step wise explanation:**

**Step-1:** Randomly generate initial population

Generate M number of paths from N! possibilities where N is the number of cities and M is the number of samples that we will select i.e., M = Initial population size.

Diagram

Description automatically generated

Figure 4: A TSP with 5 cities and distances

For example, considering the one of test case of the problem statement, where number of cities N = 5. Thus, the possible valid paths in the TSP would be 5! = 120. Now, if we select M = 4 which means out of 120 we have selected the 4 number of samples, hence M = initial population size = 4. E.g., initial population P = [[5, 2, 4, 3, 1], [4, 1, 5, 3, 2], [3, 5, 2, 4, 1], [5, 1, 4, 2, 3]]

**Step-2:** Based on the cost function, evaluate all individual chromosome.

In this step, Individual path (chromosome) will be evaluated based on the cost or fitness function where the cost function = 1 / (total distance of individual path).In the previous example corresponding fitness value V for the path vector P would be, V = [0.029411, 0.017857, 0.016949, 0.012987]. Individual fitness values vector V calculated based on the formula,

Text

Description automatically generated

Where dist(Pi[j],Pi[j+1]) is the distance between each pair of node. It indicates distance between jth and j+1th index of the node of ith path vector (chromosome) of initial population P. E.g., This fitness values vector V indicates less the distance, more fitness score it has.

**Step-3:** If generation limit exceeded, then go to step 5

Generation limit is the parameter which depicts how many times loop will iterate and generate new evolved batches of various paths.

**Step-4:**

a) Perform Selective reproduction,

Selective reproduction is to Select the number of paths from the population based on the elite size given. Elite Size E is a parameter which indicates how many best paths (chromosomes) should be selected from the population. For example, Here, population size is M = 4 and the Elite size E = 3. So, it indicates out of 4 generated population we will select only 3 best paths (chromosomes).

b) Do Crossover,

After selecting the elite chromosome in selective reproduction, now randomly two of them will merge together by copying some random path portion of one chromosome to another and generates new one. E.g.,

Graphical user interface

Description automatically generated

*Figure 5: Illustration of crossover with parents and its offspring*

c) Do mutation,

Mutation serves an important function in GA, as it helps to avoid local convergence by introducing novel routes that will allow to explore other parts of the solution space. Which means in the path with specified low fitness valued two cities will swap places in the rout which mimics the behavior of mutation in genes. E.g., [5, 2, 4, 3, 1] path will have [5, 2, 3, 4, 1] 3rd and 4th index mutation. Eventually this mutation will help to swap explore other parts when crossover will be performed. If the mutation rate like 0.25 is given then 1 path from vector P will have mutation effect. i.e., length(P) \* 0.25 = 4 \* 0.25 = 1 path of vector P.

**Step-5:** Select the best individual from the result and print it.

If whole algorithm approaches to maximum generation limit then at the end, from all elite population P with highest fitness valued path will be selected and displayed along with the distance cost.

## 3.3 Numerical Experiments

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of cities** | **Generations** | **Population Size** | **Elite Size** | **Mutation Rate** | **Initial Path distance** | **Last generation’s**  **Elite Path distance** |
| 5 | 100 | 4 | 3 | 0.25 | **35** | **31** |
| 5 | 250 | 4 | 3 | 0.25 | **35** | **32** |
| 5 | 500 | 4 | 3 | 0.25 | **34** | **31** |
| 6 | 100 | 25 | 20 | 0.25 | **87** | **76** |
| 6 | 250 | 25 | 20 | 0.25 | **81** | **76** |
| 6 | 500 | 25 | 20 | 0.25 | **76** | **76** |
| 6 | 100 | 25 | 20 | 0.25 | **1272** | **1248** |
| 6 | 250 | 25 | 20 | 0.25 | **1427** | **1248** |
| 6 | 500 | 25 | 20 | 0.25 | **1272** | **1248** |
| 15 | 100 | 100 | 20 | 0.25 | **2049** | **1225** |
| 15 | 250 | 100 | 20 | 0.25 | **2241** | **1194** |
| 15 | 500 | 100 | 20 | 0.25 | **2276** | **1194** |
| 29 | 100 | 200 | 20 | 0.25 | **80966** | **35548** |
| 29 | 250 | 200 | 20 | 0.25 | **86412** | **33734** |
| 29 | 500 | 288 | 20 | 0.25 | **81833** | **31315** |

We have conducted the numerical experiments on test cases given, there are three generation limits (hyper parameter) on which algorithm was tested, i.e., 100, 250, 500. As generation increases elite generation increases and provides the best result. Thus, we can say that more the generation, more the accurate elite result will be.

Similarly, variation is possible into the mutation rate, elite size and population size as well but that can generate multiple rows of the experiments approx. more than 405 rows into the experiment table. Thus, instead of considering them, team have given the priority to Generations because that can show the perfect comparison of elite path distance in the large number of cities case scenario.

## 3.4 Conclusion

Evolutionary algorithms have been around since the early sixties. They apply the rules of nature: evolution through selection of the fittest individuals, the individuals representing the solutions to a mathematical problem. Genetic algorithms are so far generally the best and most robust kind of advanced algorithms which gradually approaches to the global optima of the entire problem. Such as in TSP case, 5,6 cities scenario the greatest result was approached within 100 generations. Similarly, global optima can be achieved for the large number of cities as well by manipulating the hyper parameters. So, out of N! it efficiently finds solution in just O(GXPXM) complexity where G is generations, P indicates Population and M indicates elite selection of individuals.

# Bibliography

[1] S. Lin and B. W. Kernighan, ‘An Effective Heuristic Algorithm for the Traveling-Salesman Problem’, *Oper. Res.*, vol. 21, no. 2, pp. 498–516, Apr. 1973, doi: 10.1287/opre.21.2.498.

[2] N. Biggs, ‘THE TRAVELING SALESMAN PROBLEM A Guided Tour of Combinatorial Optimization’, *Bull. Lond. Math. Soc.*, vol. 18, no. 5, pp. 514–515, 1986, doi: https://doi.org/10.1112/blms/18.5.514.

[3] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, ‘Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators’, *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, Apr. 1999, doi: 10.1023/A:1006529012972.

[4] H. Talbi, A. Draa, and M. Batouche, ‘A new quantum-inspired genetic algorithm for solving the travelling salesman problem’, in *2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT ’04.*, Dec. 2004, vol. 3, pp. 1192-1197 Vol. 3, doi: 10.1109/ICIT.2004.1490730.

[5] D. Goldberg, ‘Genetic Algorithms in Search Optimization and Machine Learning’, 1988, doi: 10.5860/choice.27-0936.

[6] M. M. Flood, ‘The Traveling-Salesman Problem’, *Oper. Res.*, vol. 4, no. 1, pp. 61–75, Feb. 1956, doi: 10.1287/opre.4.1.61.

[7] R. Johnson and M. G. Pilcher, ‘The traveling salesman problem, edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B Shmoys, John Wiley & Sons, Chichester, 1985, 463 pp’, *Networks*, vol. 18, no. 3, pp. 253–254, 1988, doi: https://doi.org/10.1002/net.3230180309.

[8] R. M. Brady, ‘Optimization strategies gleaned from biological evolution’, *Nature*, vol. 317, no. 6040, Art. no. 6040, Oct. 1985, doi: 10.1038/317804a0.

[9] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, ‘Evolution algorithms in combinatorial optimization’, *Parallel Comput.*, vol. 7, no. 1, pp. 65–85, Apr. 1988, doi: 10.1016/0167-8191(88)90098-1.

[10] H. Braun, ‘On solving travelling salesman problems by genetic algorithms | SpringerLink’. https://link.springer.com/chapter/10.1007/BFb0029743 (accessed Mar. 27, 2021).

[11] C. H. Papadimitriou, ‘The Euclidean travelling salesman problem is NP-complete’, *Theor. Comput. Sci.*, vol. 4, no. 3, pp. 237–244, Jun. 1977, doi: 10.1016/0304-3975(77)90012-3.

[12] R. M. Karp, ‘Reducibility among Combinatorial Problems’, in *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Boston, MA: Springer US, 1972, pp. 85–103.

[13] S. Sahni and T. Gonzales, ‘P-complete problems and approximate solutions’, in *15th Annual Symposium on Switching and Automata Theory (swat 1974)*, Oct. 1974, pp. 28–32, doi: 10.1109/SWAT.1974.22.

[14] T. M. Press, ‘Adaptation in Natural and Artificial Systems | The MIT Press’. https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems (accessed Mar. 26, 2021).

[15] K. Katayama, H. Sakamoto, and H. Narihisa, ‘The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem’, *Math. Comput. Model.*, vol. 31, no. 10, pp. 197–203, May 2000, doi: 10.1016/S0895-7177(00)00088-1.

[16] http://www.informatik.uni-heidclberg.dc/groups/omopt‘ softwarci TSPLIB95/STSP.html

[17] Gerard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, 1994.

[18] D. B. Fogel, “An Evolutionary Approach to the Traveling Salesman Problem”, Biol. Cybern. 60,139-144 (1988)

[19] Kylie Bryant, Arthur Benjamin, Advisor, “Genetic Algorithms and the Traveling Salesman Problem”, Department of Mathematics, December 2000.

[20] Kovarik V.: Heuristic Algorithms for decision Support, Dissertation, Prague: VŠCHT, 2008

[21] Gros, I.: Quantitative Methods for managerial decision making, Prague: Grada publishing a.s., 2003, ISBN 80-247-0421-8 (In Czech)

[22] L. Guerra, T. Murino, E. Romano, The Location - Routing Problem: an innovative approach, 6th WSEAS International Conference on System Science and Simulation in Engineering, Venice, Italy, November 21-23, 2007

[23] Wisniewski, M.: Methods for Managerial Decision Making, Prague: Grada publishing, 1996, ISBN 80- 7169-089-9 (In Czech)

[24] Mohammad Reza Jalali, Abbas Afshar, Miguel A. Marino: Ant Colony Optimization Algorithm (ACO); A new heuristic approach for engineering optimization, Proceedings of the 6th WSEAS Int. Conf. on Evolutionary Computing, Lisbon, Portugal, June 16- 18, 2005, pp. 188-192

[25] Izzatdin A. Aziz, Nazleeni Haron, Mazlina Mehat, Low Tan Jung, Aisyah Nabilah: Solving Traveling Salesman Problem on High Performance Computing using Message Passing Interface, Proc. Of the 7th WSEAS Int. Conf. on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS '08), pp. 83-87

[26] Rabanal P., Rodríguez I., Rubio F.: Using River Formation Dynamics to Design Heuristic Algorithms, Springer, 2007, ISBN 978-3-540-73553-3

[27] Luping Fang, Pan Chen, Shihua Liu: Particle Swarm Optimization with Simulated Annealing for TSP, Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, February 16-19, 2007, pp. 206-210

[28] Bai Xiaojuan, Zhou Liang: A New Genetic Algorithm for solving Travelling Salesman Problem, Proceedings of the 8th WSEAS International Conference on Applied Computer and Applied Computational Science, 20, pp. 451-455

[29] Fu Yiping, Chen Guangxi, An improved genetic algorithm for solving TSP problem, Journal of Guilin University of Electronic Technology, Vol.27, No.4, 2007, pp.287-290

[30] Han Lixia, Wang Yuping, A Novel Genetic Algorithm for Travelling Salesman Problem, systems Engineering-Theory & Practice, No.12,2007, pp.145-151

[31] Sangwan, Shabnam. (2018). Literature Review on Travelling Salesman Problem. International Journal of Research. 5. 1152.