Project Report on

"**Optimization of Travelling Salesperson Problem using Genetic Algorithm**"

Submitted by

Pranab Kumar Paul

Roll: 90/MCS No.210008

Reg.No. 2080001 of 2021-2022

M.Sc. Computer Science 3rd Semester

Under the supervision of

Dr. Priya Ranjan Sinha Mahapatra

Department of Computer Science and Engineering

University of Kalyani, Kalyani-741235

West Bengal, India

Abstract

As our project is on optimizing the travelling salesperson problem so at first, we should know about the problem. The Traveling Salesperson Problem also known as travelling salesman problem or TSP is a well-known and popular optimization problem. The **solution** of the problem is to find the lowest distance to cover all the cities and return back to the initial city with the constraint that each city must be visited only once except the initial city. Despite the simple problem statement, solving the problem is much more difficult task as it belongs to **NP-complete** class. The importance of the TSP arises because of its variety of applications. General applications include computer wiring, vehicle routing, and job sequencing. In this paper we introduce the Genetic Algorithm (GA) approach for handling and solving the traveling salesperson problem in a simple but effective process.[1]

Keywords: NP-Complete, Solution, Genetic Algorithm.

1. Introduction

The travelling salesperson problem seeks to find the shortest tour that visits every city exactly once and returns to the starting city. This problem is considered NP-complete in the field of combinatorial optimization, and is equally significant in the fields of operations research and theoretical computer science.

The travelling salesperson problem (TSP) can be described as a unweighted graph, denoted as G=(V,E), where each edge (u,v) Є E has a non-negative integer cost d(u,v) associated with it. The graph's vertices represent the cities, while the edges represent the paths between the cities, with the weights of the edges indicating the distance between the corresponding cities.

Problem Statement of TSP: The complete graph of cities is stored in **adjacency matrix** say A with elements C_{ij} where i, j=1...m where 'm' is the number of cities and the diagonal elements of matrix A are zero. A tour can be represented by a Hamiltonian cycle Π of {1, 2, . . ., m} where $\Pi_{(i)}$ represents the next city i on the tour. The traveling salesperson problem is the optimization problem to find a permutation Π of cities that minimizes the **total distance of the tour** denoted by

$$\sum_{i=1}^{m} C_{i\Pi(i)}$$

The brute force method requires calculating (m - 1)! permutations for each of the n iterations in the minimization task. As a result, the overall complexity of the method becomes O(m!), which is a very large number.

Keywords: Distance Matrix, NP complete, Applications.

2. Motivation

The traveling salesman problem (TSP) is a significant research area that seeks to minimize the total distance traveled by a salesman on a complete tour. When the distance between any two vertices U and V is identical to the distance between V and U, the problem is classified as Symmetric TSP (STSP). On the other hand, if there exists at least one pair of vertices in the graph where the distance between U and V is not equal to the distance between V and U, the problem is known as an Asymmetric TSP (ATSP).
In general, there are two methods for solving a TSP: brute force method and **heuristic methods**. The brute force method requires enormous time for larger n; which is O(n!) time for brute force solution. By **dynamic programming** it is solved in $O(2^n n^2)$ time which is still exponential time complexity. Numerous heuristic algorithms have been created and suggested in the field of operations research to tackle the travelling salesman problem. Solving TSP is relatively simple when the number of cities is small, but it becomes more difficult as the number of cities increases due to the significant amount of computation time required. So here we use a heuristic approach named Genetic Algorithm (GA) to solve TSP to get a nearest optimal solution with in less time.

Keywords: Symmetric-TSP, Asymmetric-TSP, Complexities, Heuristic Algorithms

3. Proposal

The Genetic Algorithm (GA) is a type of **evolutionary search algorithm** that aims to find the closest optimal solution for various NP problems. In order to achieve an optimized result using GA, it requires an effective chromosome representation, carefully designed crossover and mutation operators. As the number of cities increases, the complexity of the problem also increases, and the solution becomes less accurate. Initially, GA system starts with an adjacency matrix of cities, which represents the Distance Matrix, and a randomly selected city sequence as the initial population. New generations are formed recursively until a more optimal path is reached. Genetic algorithms employ techniques such as inheritance, mutation, selection, and crossover, which are inspired by the evaluation system. The reason for using GA is that sometimes generating near-optimal solutions quickly is more desirable than finding the optimal solution, which can take a huge amount of time, i.e., exponential time.

Keywords: Inheritance, Selection, Crossover, Mutation, Initial Population

4. Algorithm

The concept of genetic algorithm was introduced by **John Holland** in the 1970s, but it became well-known in the late 1980s. The approach is rooted in Darwin's theory of natural selection.

4.1. Idea Behind Genetic Algorithm

The optimization method, Genetic Algorithm, is grounded in the principle of **Natural Selection** introduced by Darwin. The principle suggests that the best individuals should be chosen while others should be discarded. A case in point is an animal population of a specific species residing in an ecosystem where some animals are more capable of surviving in that environment than others due to their fitness. These animals compete for the limited resources, such as water and food, available in the ecosystem. Eventually, only the strongest or fittest individuals survive while the others perish, thus leading to the **survival of the fittest.**

4.2. Working Principle

To improve the quality of solutions, Genetic Algorithm operates on an initial population of the given problem instance, which is evolved through multiple generations until a termination condition is met. During each iteration, the population is replaced with a newly obtained better generation. The GA operators process the individuals of the current population to enhance the quality of the new generation as compared to the old one. This iterative process leads to the improvement of solutions, and at the end of the search, we can expect to get the best or near-best solution generated by the GA.

4.3. Mapping with TSP

A GA possesses several distinct characteristics, including the chromosomes, encoding and decoding procedures for a solution to a chromosome, fitness function for evaluating each chromosome, population size, initial population, mating pool selection, GA operators such as selection, crossover, and mutation, various GA parameters like crossover probability (pc), mutation probability (pμ), and population size, and finally the termination condition.

1. Chromosomes: Here in the context of Travelling Salesperson problem the chromosomes are the random permutation of the cities which cover all the cities exactly once and return to the initial city. For e.g., with a map of 5 cities represented by digits i.e., 1 2 3 4 5, one possible chromosome is 4 3 2 1 5.

2. Fitness Value: Here the Fitness value is the total cost of the path in the corresponding tour.

3. Initial Population: Here the initial population is the randomly generated city sequences i.e., set of randomly generated city sequences. And the total number of populations is taken as input from the user.

4. Mating Pool: After evaluating the current population the best chromosomes among the current population is selected in Mating Pool and further operations are done on the mating pool.

5. GA Operators: There are three logical operators applied on the mating pool in each iteration i.e., selection, crossover, mutation which will be further elaborated in the upcoming slides.

6. Terminating Condition: Here I implemented terminating condition as follows, if the minimum fitness value of the population does not change for the 100 iterations, then the algorithm will converge and give the solution as the tour length and the city sequences i.e., best chromosome.

4.4. The Algorithm:

Procedure GA:

The following steps can be followed to solve a problem using the Genetic Algorithm:

1. Begin by initializing the population, which is the **current population**.
2. Iterate through steps 3 to 5 until the **termination condition** is met.
3. Use the **selection** operation to obtain the mating pool from the current population.
4. Use the **crossover** and **mutation** operators on the mating pool to generate the new population.
5. Replace the **current population** with the newly generated population.
6. Return the **best solution** found in the current population.

This is a basic outline of the steps that can be followed to use the Genetic Algorithm to solve a given problem.

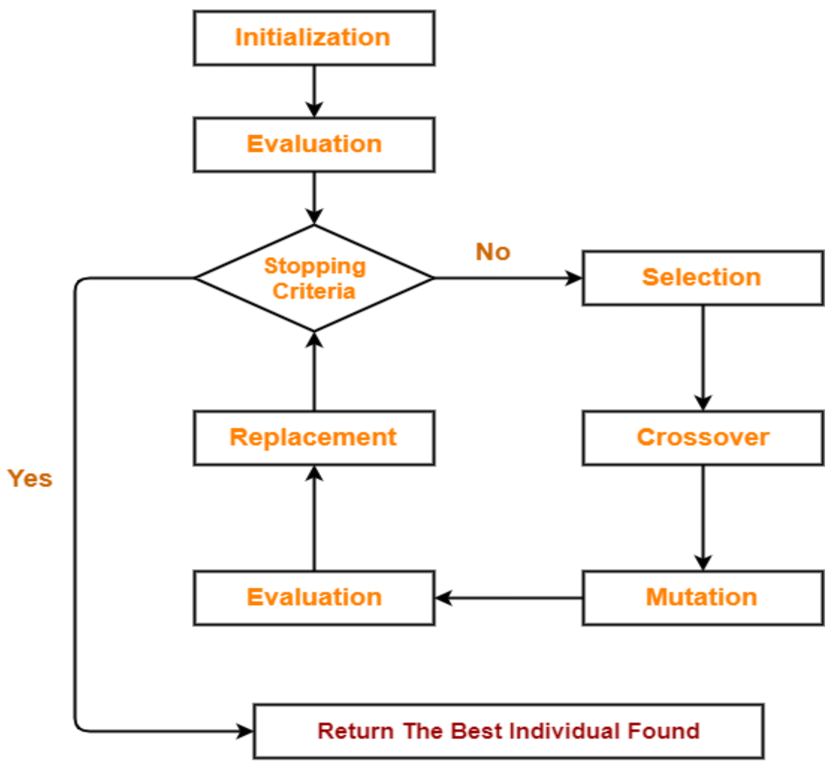


Fig: Flow Chart of Genetic Algorithm.[2]

5. Implementation

In order to utilize genetic algorithm to solve TSP, a chromosome is used to represent the encoding solution. The length of the chromosome corresponds to the number of cities in the problem. The procedure for implementing GAs on a TSP problem with 6 cities is explained below.

5.1. Encoding:

In this example, we have chosen a problem with six cities and assigned a number to each city. The path between these cities can be represented as a sequence of integers from 1 to 6, using permutation encoding where the order of the integers determines the fitness of the solution. The distances between the cities are represented using an **adjacency matrix**, which is symmetric. This means that the distance between city 'u' and city 'v' is the same as the distance between city 'v' and city 'u'. The first row and column of the matrix indicate the names of the cities

Table. Distance matrix of 6 cities (distance in Km).

	City 1	City 2	City 3	City 4	City 5	City 6
City 1	0	261	1499	1417	203	1170
City 2	261	0	1454	1207	939	922
City 3	1499	1454	0	541	1679	890
City 4	1417	1207	541	0	1806	362
City 5	203	939	939	1806	0	1552
City 6	1170	922	890	362	1552	0

In this scenario, the initial population is comprised of six chromosomes. Each chromosome represents a random sequence of the cities, adhering to the constraint that each city can only appear once. The number assigned to a city is used to represent each chromosome.

Initial Population:

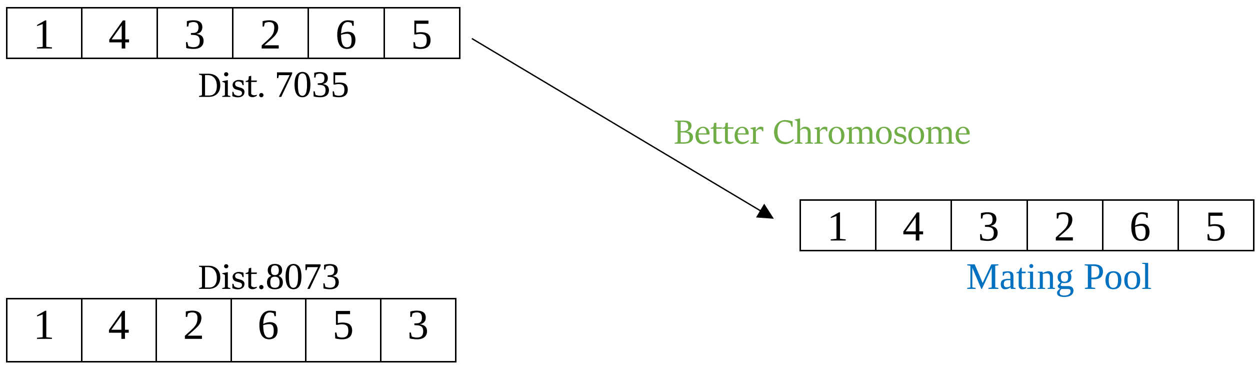
Chromosome 1:	1 3 4 2 5 6
Chromosome 2:	1 4 2 6 5 3
Chromosome 3:	3 6 1 4 2 5
Chromosome 4:	6 3 1 5 2 4
Chromosome 5:	5 2 6 1 3 4
Chromosome 6:	2 6 3 1 5 4

5.2. Fitness Function:

The primary goal of the fitness function is to identify the best individual in the population based on their performance. In the case of the travelling salesman problem, the **fitness value** is determined by the tour cost of a given chromosome. This tour cost is the sum of the distances covered by the chromosome's sequence, and the solution is considered fitter if this sum is smaller.

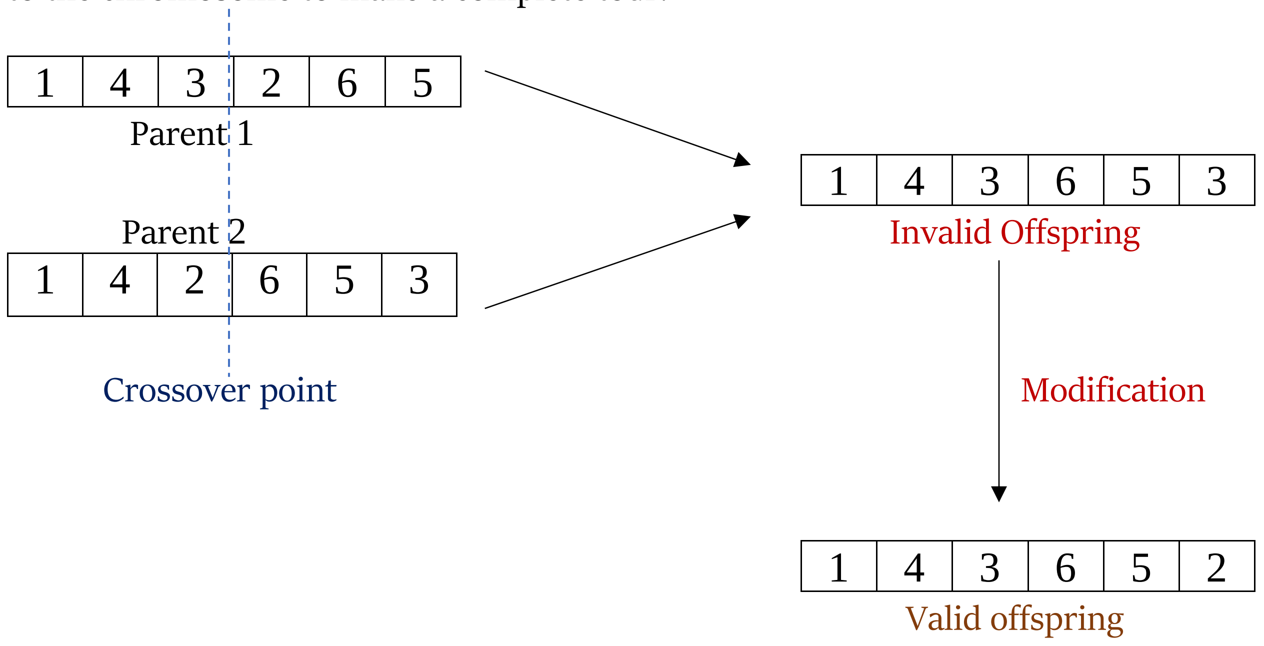
5.3. Selection:

Selection is a process used to choose chromosomes from the population based on their fitness values. In this case, we have employed **tournament selection**, which involves pitting two solutions against each other and selecting the fittest one to move on to the mating pool. This process repeats until the mating pool is full.



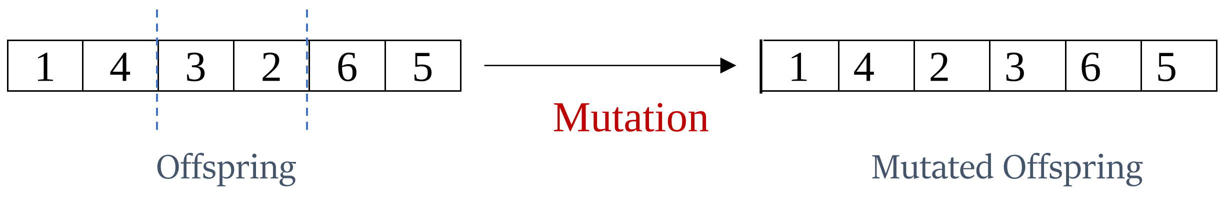
5.4. Crossover:

Crossover means exchanging the genetic elements between the chromosomes. Crossover is of two types: single point crossover and multi-point crossover. We have chosen to use the single-point crossover method, which involves selecting a random crossover point in the chromosome and exchanging the substrings between two selected chromosomes. However, this method is not suitable for the travelling salesman problem since it can generate offspring that do not meet the problem's requirements. The crossover point is chosen randomly. Here I slightly modified the crossover operation. After the crossover of two chromosomes, we apply unique function on the set of chromosomes and take the unique cities only then we apply difference operator to identify the missing cities. Then the missing cities are added to the chromosome to make a complete tour.



5.5. Mutation:

Mutation is a crucial process in generating a new and improved generation. It involves making random changes to the genetic sequence of a chromosome. In our study, we have implemented mutation by randomly selecting two points in the chromosome and swapping the cities between them to create a new tour. This process is depicted in the figure.



5.6. Termination Condition:

Here I implemented terminating condition as follows, if the *minimum fitness value of the population does not change for the 100 iterations*, then the algorithm will converge and give the solution as the tour length and the city sequences i.e., best chromosome. The figure illustrates the optimal tour with the minimum distance of 3968km between six cities.

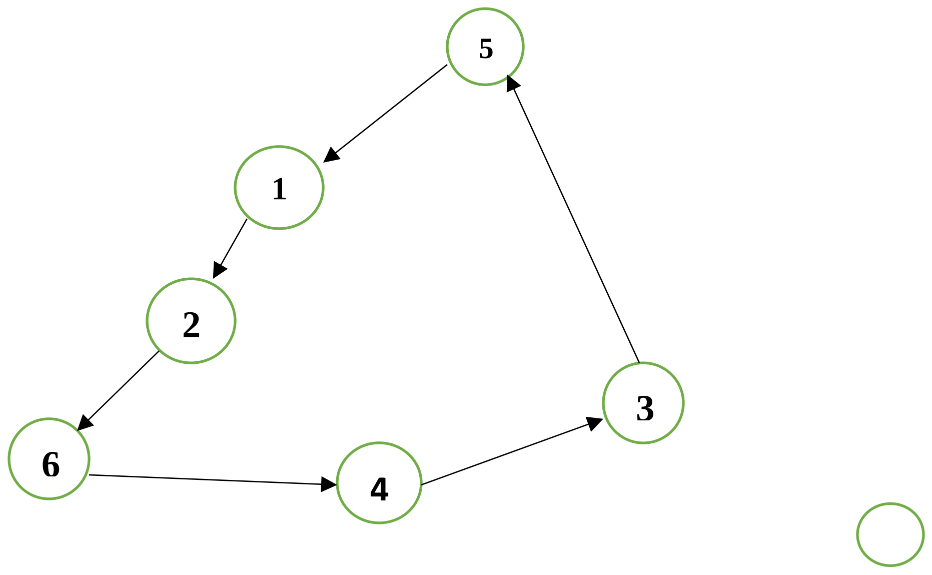


Figure. Minimum Distance between 6 cities

6. Results

I implemented the above-mentioned solution using MATLAB programming language in MATLAB 2021a with the system configuration: Intel(R) Pentium(R) CPU G4400 @ 3.30GHz, 3300 MHz, 2 Core(s), 2 Logical Processor(s), Windows 10 Operating System and 4 GB RAM.

$$\text{Error Rate} = \frac{Best\ Solution - Optimal\ Solution}{Optimal\ Solution * Cities}$$

Inputs are taken from these TSP Libraries:

- TSP - Data for the Traveling Salesperson Problem (fsu.edu).
- CITIES - City Distance Datasets (fsu.edu).
- National Traveling Salesman Problems (uwaterloo.ca).
- jorlib/jorlib-core/src/test/resources/tspLib/tsp at master · coin-or/jorlib · GitHub

Problems From TSPLIB	**Cities**	**Population**	**Generation**	**Best solution**	**Optimal Solution**	**Error Rate(%)**
P01	15	100	100	291	291	0.00
GR17	17	100	146	2085	2085	0.00
FRI26	26	400	280	937	937	0.00
WI29	29	3000	378	26703	26703	0.00
DJ38	38	5000	454	6656	6656	0.00
DANTZIG42	42	300	347	699	699	0.00
SWISS42	42	1000	500	1366	1273	0.17
ATT48	48	3000	846	33889	33523	0.02
ELI51	51	1000	1002	442	426	0.07
ST70	70	1000	1341	786	675	0.23
kroA100	100	1500	2508	24386	21282	0.14
pr124	124	5000	1586	66286	59030	0.12
pr152	152	5000	2342	85962	73682	0.11
QA194	194	1000	5369	11563	9352	0.19
kroA200	200	5000	3659	41205	29368	0.20

X-axis: Generation **Y-axis:** Minimum Distance Plotting
Orange: Average Fitness value Green: Minimum Distance

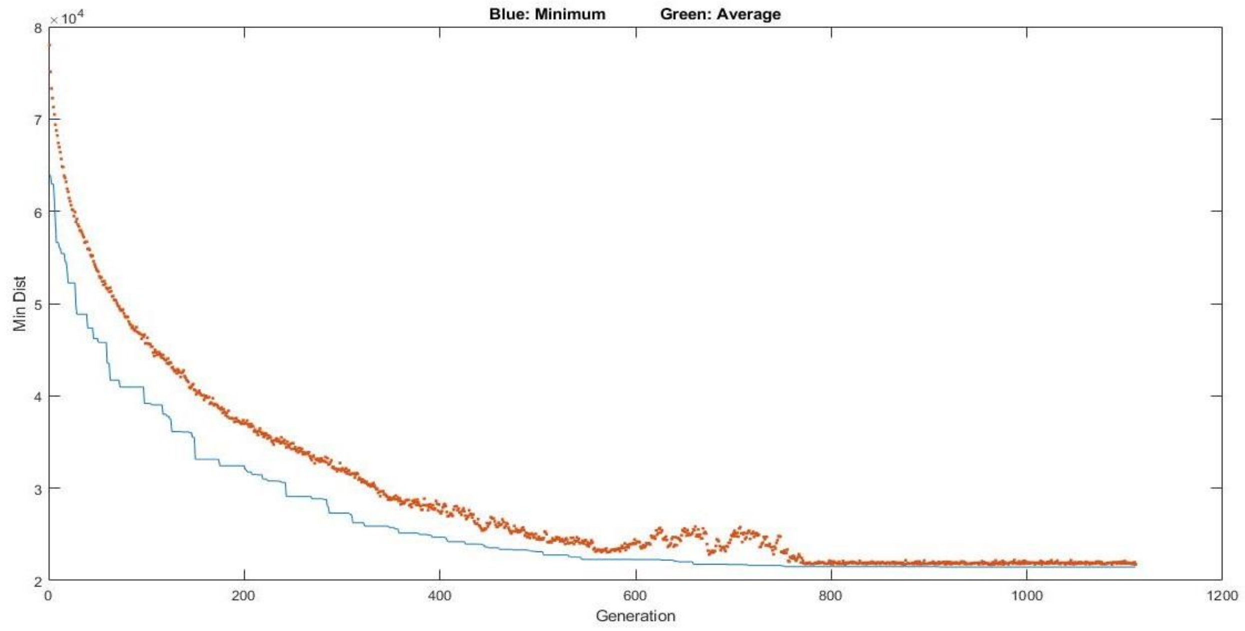


Fig: Graphical Representation of Average and Minimum Distance

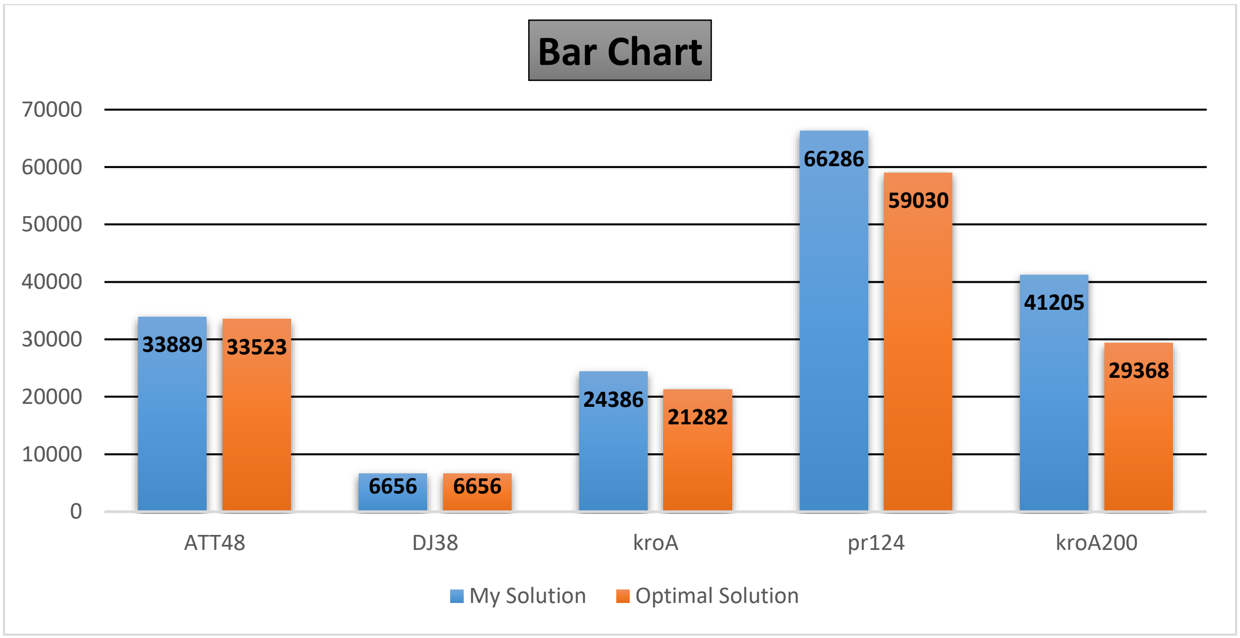


Fig: Bar Plot of some instances of Optimal solution and My Solution

Y-Axis: Minimum Distance (in km) **X-Axis:** Problem Instances

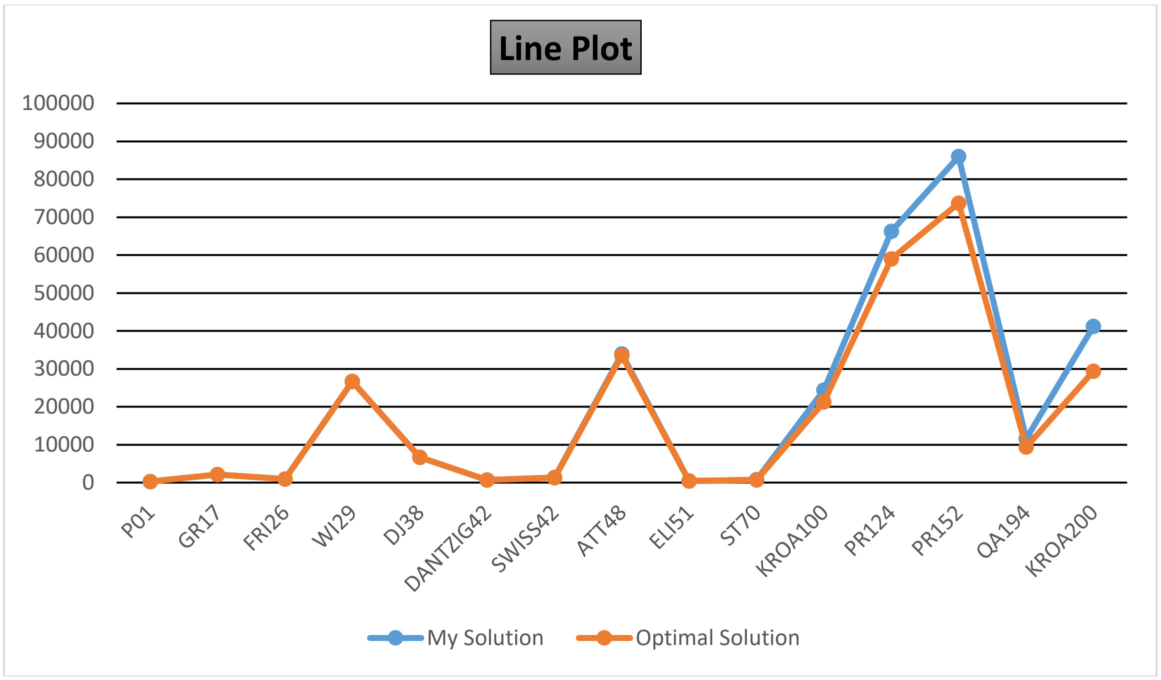


Fig: Line Plot of all instances of Optimal solution and My Solution.

7. Conclusion

The use of genetic algorithms to solve the travelling salesman problem is a promising approach. Our **aim** with this study is to develop a more efficient method for solving TSP using GA. While we have currently addressed symmetric TSP, we plan to enhance our solution to achieve greater accuracy and tackle asymmetric TSP as well. Additionally, we seek to minimize the error rate for larger city problems.

Keywords: our objective, Future Goals

8. Acknowledgement

I would like to express my sincere gratitude to my supervisor, **Prof. Priya Ranjan Sinha Mahapatra**, as well as **Dr. Soumen Atta** and **Prof. Anirban Mukhopadhyay** for their constant encouragement and support throughout this project. I also extend my thanks to the Research LAB at KU for providing the necessary support for this study.

9. References

[1] (PDF) "Solving Travelling Salesman problem by New Optimization Algorithm" (researchgate.net).

[2] introduction-to-soft-computing-neuro-fuzzy-and-genetic-algorithms-by-samir-roy-udit-chakraborthy_compress.

[3] Exploring Travelling Salesman Problem using Genetic Algorithm – IJERT.

[4] Exploring Travelling Salesman Problem using Genetic Algorithm (ijert.org).

[5] An Introduction to Genetic Algorithms: Method and Implementation (Lecture 1) by Anirban Mukhopadyay.