REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Informatıon Technologies

INTELLIGENT PATH OPTIMIZATION OF TRAVELLING SALESMAN PROBLEM BASED ON MODIFY GENETIC ALGORITHEM

Mazin Mohammed HAMID

Master's Thesis

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2023

INTELLIGENT PATH OPTIMIZATION OF TRAVELLING SALESMAN PROBLEM BASED ON MODIFY GENETIC ALGORITHM

Mazin Mohammed HAMID

Information Technologies

Master's Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled INTELLIGENT PATH OPTIMIZATION OF TRAVELLING SALESMAN PROBLEM BASED ON MODIFY GENETIC ALGORITHM prepared by MAZIN MOHAMMED HAMID and submitted on 07/08/2023 has been **accepted unanimously** for the degree of Master of Science in Information Technology.

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. AYÇA KURNAZ TÜRKBEN	Department of Software Engineering, Altınbaş University	_____
Assoc. Prof. Dr. Serdar KARGIN	Department of Biomedical Engineering, Arel University	_____

I hereby declare that this thesis meets all format and submission requirements for a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Mazin Mohammed HAMID

Signature

DEDICATION

First and foremost, I thank the merciful God has guided me in all my life. And I wold like to thank the supervisor for this thesis, Asst. Prof. Dr. Abdullah Abdu IBRAHIM, who encouraged me in my work, and gave me motivation, support, and direction to complete this thesis. Also I present a lot of thanks to those who helped me, particularly my dear father and mother, as well as dear wife, they have always been encouraging in all circumstances. I also thank all the members of my family who supported me for this thesis and hope that God may help us.

PREFACT

All thanks and gratitude go to my supervisor, Asst. Prof. Dr. Abdullah Abdu IBRAHIM, who is known for his knowledge, humility and morals, who gave me his assistance in my writing. Thanks to the university staff and everyone who provided me with science and advice.

ABSTRACT

INTELLIGENT PATH OPTIMIZATION OF TRAVELLING SALESMAN PROBLEM BASED ON MODIFY GENETIC ALGORITHM

HAMID, Mazin Mohammed
M.Sc., Information Technologies, Altınbaş University,
Supervisor: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM
Date: 08/2023
Pages: 85

The travelling salesman problem (TSP) is one of the oldest and most common problems that should be optimized using one of the optimization algorithms to make the salesman's travel itinerary to be short and non-repeated. The solution for travelling salesman problem becomes computationally complex because there are a large number of cities and the travelling salesman requires to visit each city once with shortest path. Multiple algorithms are used to find a good solution to this problem such as genetic algorithm. This study used modify genetic algorithm which is depend on a set of biological changes that occur in living organisms, through a set of steps (natural selection, population, crossover, and mutation). A new intersection method is proposed in this thesis, which uses three techniques together in intersection, namely (flipping, swapping, and sliding). The experiments were applied to the variant number of cities which are 10,50 and 150,200 with two scenarios. In the first scenario, each experiment, a fixed number of iteration (100000) and population size (150) were chosen, the execution time to find solution was (82.00 seconds), (111.41 seconds), (151.10 seconds) and (240.41 seconds) for cities (10,20,100,200), respectively. In the second scenario, the population size was set to (50) or (100) and the number of iteration set to (10000) or (50000). The results to find solution were (03.2 seconds), (04.33 seconds), (05.47 seconds), and (35.05 seconds) for cities (10,20,100,200), respectively, and the results showed that each of (iteration, population) is directly proportional to the number of cities, with the use of the three techniques together, we get a shorter path and less time comparing with the performance of the traditional genetic algorithm.

Keywords: Travelling Salesman Problem, Genetic Algorithm, Modify, Natural Selection, Crossover.

TABLE OF CONTENTS

Pages

ABSTRACT ..... **vii**

LIST OF TABLES ..... **xiii**

LIST OF FIGURES ..... **xiv**

ABBREVIATIONS ..... **xvii**

1. INTRODUCTION **1**

1.1 ARTIFICIAL INTELLIGENCE 2

1.2 TRAVELLING SALESMAN PROBLEM 3

1.3 GENETIC ALGORITHEM 4

1.4 MOVITATION 5

1.5 PROBLEM FORMULATION 5

1.6 THESIS STATEMENT 6

1.7 AN OVERVIEW AND DYNAMIC ON THE GENETIC ALGORITHEM 6

1.8 CONTRIBUTION OF THIS STUDY 6

1.9 STUDY SCOPE 6

1.10 PROBLEM BACKGROUND 7

1.11 ORGNIZATION OF THE THESIS 7

2. BACKGROUND **8**

2.1 INTRODUCTION 8

2.2 DALING WITH TRAVELLING SALESMAN PROBLEM 8

2.3 NP-HARDNESS 8

2.4 LP RELAXATION 9

2.5 APPLICATION OF THE TSP 9

2.6 METHODS FOR DEALING WITH THE TRAVELLING SALESMAN 10

2.7 TOUR CONSTRUCTION HEURISTICS 11

2.7.1 Nerest Neaghbor Algorithem 11

2.7.2 Greedy Algorithem 12

2.7.3 Fierflay Algoritherm 12

2.7.4 Hybrid Genetic Algorithem 13

2.7.5 Method Of Nerest Entry 14

2.7.6 The Method Of Listingin Random 14

2.7.7 The Method Most Distant Insrtation 14

2.7.8 Christofides Algorithem 15

2.7.9 Method of Enhanced Solution 15

2.8 EXTENSIVE LITERATURES 16

3. METHODOLOGY 19

3.1 INTRODUCTION 19

3.2 GENETIC ALGORITHEM ESSENTIAL STEPS 19

3.3 GENERAL STEPS OF GENETIC ALGORITHEM 21

3.3.1 Great Initial Population 21

3.3.2 Objective Function & Fitness Value 21

3.3.3 Selection 22

3.3.3.1 Roulette wheel method of selection 23

3.3.3.2 The methode of selection ranks 24

3.3.3.3 Binary selection 25

3.3.3.4 Goldberg's method of selection 26

3.3.3.5 Whitely's method of selection 27

3.3.4 Crossover 27

3.3.4.1 One-piece mating 29

3.3.4.2 Two-piece mating 29

3.3.4.3 Zigzag crossover 30

3.3.4.4 Partially striped crossover (PSC) 30

3.3.4.5 Order crossovre 32

3.3.4.6 The proposed crossover method based on (flip, swap and sliding) techniques 33

3.3.5 Mutation 33

3.3.6 Genetic Algorithem Stopping Criterion 35

3.3.7 The Method of Representing The Chromosome 36

3.3.7.1 Binary encoding 36

3.3.7.2 Permutation encoding 36

3.3.7.3 Direct values encoding 37

3.3.7.4 Tree coding 37

3.3.8 Differences between Developmental Algorithem and Traditional Reserch Methods 38

3.4 CONCLUSION 39

4. RESULT AND DISCUSSION 40

4.1 EXPERMENT SETUP 40

4.2 ANALYSIS OF RESULT 59

5. CONCLUSTION AND FUTURE WORK 66

5.1 CONCLUSION 66

5.2 FUTURE WORK 66

REFERENCES 67

LIST OF TABLES

Pages

Table 3.1: Differences Between Developmental Genetic Algorithem and Traditional Reserch Methods 39

Table 4.1: Algorithms Results Compared by Distance and Time using Traditional Genetic Algorithm 60

Table 4.2: Algorithms Results Compared by Distance and Time using Modify Genetic Algorithm 65

LIST OF FIGURES

Pages

Figure 3.1: General Steps of Genetic Algorithm 20

Figure 3.2: Code of Fitness Value 22

Figure 3.3: Roulette Method for Selection Best Value for Fitness 24

Figure 3.4: Code of Global Method Selection 26

Figure 3.5: Code of Whitely Method of Selection 27

Figure 3.6: Code of Crossover Method of Selection 28

Figure 3.7: One-Piece Mating 29

Figure 3.8: Two-Piece Mating 29

Figure 3.9: Zigzag Crossover 30

Figure 3.10: PSC Step 1 30

Figure 3.11: PSC Step 2 31

Figure 3.12: PSC Step 3 31

Figure 3.13: Code of Hybrids Methods 33

Figure 3.14: Code of Mutation 35

Figure 3.15: Tree Mating Process 38

Figure 3.16: Mutation Coefficient or Number 38

Figure 4.1: Cities Locations for (10) Cities 42

Figure 4.2: Optimal Solution for (10) Cities 42

Figure 4.3: Best Solution History for (10) Cities 43

Figure 4.4: Cities Locations for (10) Cities 43

Figure 4.5: Total Distance for (10) Cities using (1000) Iterations 44

Figure 4.6: The Best Solution History for (10) Cities using (1000) Iterations..... 44

Figure 4.7: Cities Locations for (50) Cities 45

Figure 4.8: Optimal Solution for (50) Cities 46

Figure 4.9: Best Solution History for (50) Cities ,(100000) Iterations, (150) Population Size 46

Figure 4.10: Cities Locations for (50) Cities 47

Figure 4.11: Optimal Solution for (50) Cities 48

Figure 4.12: The Best Solution History for (50) Cities Using (1000) Iterations..... 48

Figure 4.13: Cities Locations for (100) Cities 49

Figure 4.14: Total Distance for (100) Cities 50

Figure 4.15: The Best Solution History for (100) Cities Using (10000) Iterations..... 50

Figure 4.16: Cities Locations for (100) Cities 51

Figure 4.17: Total Distance for (100) Cities and the Optimal Path Without Repeating 52

Figure 4.18: Best Solution Histor for (100) Cities, (100000) Iterations, (150) Population Size 52

Figure 4.19: Cities Locations for (200) Cities 53

Figure 4.20: Total Distance for (200) Cities 54

Figure 4.21: The Best Solution History for (200) Cities Using (100000) Iterations..... 54

Figure 4.22: The Number of Iteration (221)..... 55

Figure 4.23: The Number of Iteration (805)..... 55

Figure 4.24: The Number of Iteration (1235)..... 55

Figure 4.25: The Number of Iteration (2131)..... 55

Figure 4.26: The Number of Iteration (3138)..... 56

Figure 4.27: The Number of Iteration (4138)..... 56

Figure 4.28: The Number of Iteration (5165)..... 56

Figure 4.29: The Number of Iteration (6178)..... 56

Figure 4.30: The Number of Iteration (7328)..... 57

Figure 4.31: The Number of Iteration (8205)..... 57

Figure 4.32: Cities Locations for (200) Cities Created Randomly..... 58

Figure 4.33: Total Distance for (200) Cities 58

Figure 4.34: The Best Solution History for (200) Cities Using (10000) Iterations..... 59

Figure 4.35: Cities Locations for (200) Cities 61

Figure 4.36: Total Distance (200) Cities using (1000) and (50) Population Size. 62

Figure 4.37: Best Solution History for (200) Cities using (10000) Iterations, (50) Population Size 62

Figure 4.38: Cities Locations for (200) Cities Randomly by Program 63

Figure 4.39: Total Distance for (200) Cities, Iterations (50000), Population Size (100)... 63

Figure 4.40: Best Solution History for (200) Cities, (10000) Iterations, (100) Population Size 64

ABBREVIATIONS

AI	:	Artificial Intelligence
PSC	:	Partially Striped Crossover
NN	:	Nearest Neighbor
FA	:	Fierfly Algorithem

1. INTRODUCTION

With the great development of big data technology, graphics computing is applied to all kinds of fields. For example, in power-inspection route planning, the path length between each point or device can be considered nodes in the graph, thus a power check path graph is formed, and found the best path. In another example, passenger bus routing, product distribution, flight routing, routing vehicles [1].The TSP where the salesman has to go to all randomly selected cities can be summed up in a graph. Where the travelling salesman starts from the first city to all cities and then returns to the starting point without visiting a city more than once, and choosing the shortest path [2]. Despite this, TSP has grown to popular problems, typical methods for solving the TSP can be categorized in to exact algorithms, approximation algorithms, and heuristics, we use genetic algorithm is one of the bio-inspired evolutionary algorithm, uses the ideas of "Normal Selection" and "Genetic Inheritance", to solve it optimally or semi-optimally is really challenging, this problem it has become really standard for many algorithm designs[3][4].

The oldest known technique of random optimization that is based on biological principles motivated by genetics is the genetic algorithm that it uses. This algorithm consists of five stages (initial population, selection, crossing over, fitness functions, and mutations). If the first generation the parents are physically fit, their descendants (the next generation) will outgrow their age and have a better chance of survival. And so on until the eligible generation is discovered genetic algorithms are playing an increasingly important role in studies of complex adaptive systems, ranging from adaptive agents in economic theory to the use of machine learning techniques in the design of complex devices such as aircraft turbines and integrated circuits. Adaptation in natural and artificial systems, presenting the theoretical foundations and exploring applications. In its most familiar form, adaptation is a biological process, whereby organisms evolve by rearranging genetic material to survive in environments confronting them. In this work, presents a mathematical model that allows for the nonlinearity of such complex interactions. He demonstrates the model's universality by applying it to economics, physiological psychology, game theory, and artificial intelligence and then outlines the way in which this approach modifies the traditional views of mathematical genetics, many scientists and researchers used the genetic algorithm because it is considered as one of the most efficient algorithms used for optimization searches, it

contains many operators that can give good results when representing the problem, and one of the reasons for the popularity of its procedures is accuracy and reliability. One of the most important uses of the genetic algorithm is to choose the best global positioning satellites [5], it is also used in the GPS system positioning and solving the problem of the travelling salesman[6][7]**.**

1.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence, which began in computer science, is today a rapidly expanding issue in a variety of professions[8]. Since AI initially refers to building "humanoid" machines, they are believed to have human-like vision and perception as well as the ability to function in complex environments however, the idea of intelligence is still somewhat vague, amnesty International has yet to create a standard definition , it is often assumed that (AI) is a science that examines the process of computer simulation of specific human intelligence behaviours such as perceiving, learning, thinking, communicating, and behaving, among others[9].

In fact, due to technological limitations, the above-mentioned primary goal remains elusive. The primary goal of AI today is to teach machines to do things that people can do better and more efficiently. Artificial intelligence is being created as a powerful tool to improve people's lives, reduce human effort, and improve work experience. As a result, human beings may be freed from many monotonous, physical, and dangerous occupations. The development of artificial intelligence has gone through many ups and downs. After AI was initially proposed in 1956, it has gone through its first golden period, which ran from 1956 to 1974. Many countries have invested in this new field of study, and many initiatives and research programs related to AI have been launched worldwide in that time. Her first "winter" (1974-1980) in the late 1970s. After the British government stopped supporting undirected research in 1973, artificial intelligence came under scrutiny. Japanese investors have also withdrawn their support for AI research. One of the most promising applications of AI is the field of self-driving vehicles, where self-driving cars and trucks are being developed by companies like google, uber, and tesla, which use AI to interpret sensor data and make decisions about how to navigate roads, and avoiding obstacles despite the difficulties, some aspects of artificial intelligence, such as logical programming, logical thinking, etc, have been researched[10] .

Artificial intelligence is greatly increased with the advancement of technology. It can be seen in many synthetic applications intelligence in technological development. Several AI technologies solve the issues. One of the problems that could be solution by artificial intelligence is the TSP. In a travelling salesman program, a salesperson must travel to several destinations with optimized mileage. For a few of the destination, the optimal distance can be reached with a minimum distance. However, in more extensive cases, the optimal distance can be obtained with the lowest search distance practical. The genetic algorithm can find the optimal distance by regenerating each group so that it produces the minimum value. This value is not the minimum value at all points, but it can be the global optimal value. By applying artificial science intelligence in case of a travelling salesman problem the mileage will be improved[11].

1.2 TRAVELLING SALESMAN PROBLEM

The TSP is a well-known problem has been of interest to researchers for many years. This problem is that the salesperson visits a number of locations on the map and returns to the beginning and must visit each point only once, taking care to reduce the total distance between cities starting from the city, with a complete graph representing the problem, the problem can be represented graphically by N (representing N cities) and the distance function d(I, J) reflects the separation between the first town and second town TSP is an NP-hard problem[12]. This means difficulty reliably works out the problem quickly on classical computers, and it is not possible to find a solution in an efficient way in the limited time available. It is also impossible to ascertain the best response to the TSP problem in a reasonable time using classical computers [13].

Therefore, computational and mathematician scientists use algorithms applicable to modern computers to reduce computation time and to find acceptable approximate solutions. However, this technology does not always find the best answer, and thus alternative thinking, like simulated genetic algorithms, has been created increase quality of solutions. TSP has many different applications, including delivery path optimization. Moreover, the expansion of the study of this topic, as it is widely used in mathematics and computer science, leads to the creation of new algorithms approaches to address other combinatorial optimization problems[14].The salesperson starts at a specific location, ends at another, and returns to the initial location, aiming to reduce the total distances travelled.

Solving TSP requires knowledge of the distances between all the locations. If there are (n) total points, there are(n-1)possible connections for the first point and(n-2)connections for the second point, and so on. The scale of the problem is determined based on the total points, denoted as (n) , and quantity of possible routes is calculated as (n-1 factorial). As already stated, when dealing with smaller number of points, we may be able to get a better the equations below show the mathematical form of the problem [15].

$$\gamma \quad = \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} x(i,j) d(i,j) \tag{1.1}$$

$$\sum_{j=1, j \neq i}^{n} x(i,j) = 1, i = 1,2, \dots, n \tag{1.2}$$

$$\sum_{j=1, j \neq i}^{n} x(i,j) = 1, i = 1,2, \dots, n \tag{1.3}$$

$$\sum_{i,j \in S, i \neq j}^{n} x(i,j) \leq = S - 1, \forall S \subset \{1,2, \dots, n\} \tag{1.4}$$

$$x(i,j) = \begin{cases} 1, \\ 0, \end{cases} if\ a\ path\ connects\ points\ i\ and\ j, otherwise\} \tag{1.5}$$

The equation represents the TSP's primary equation (1.1) d is distance among cities (i, j). It shown by the expression x (i, j) whether the target is equation (1.2) and the equation (1.3) are used to move from the spot (i) to spot (j). (1.3) aim to confirm this.

Each spot will only be visited once. As for in formula (1.2), each spot is only left once, so that every spot is just visited once (1.3). For equation (1.4), it gives sub-ellipsis in order to avoid sub-rounds from being erected. When (x) is a value (i, j) = 1 in equation (1.5) indicates that there is a round from i to j, if the value is 0, it indicates that there is no trip.

1.3 GENETIC ALGORITHM

A particular kind of evolutionary algorithm that takes natural evolution is the genetic algorithm. This tool uses heuristics to optimize searches [16]. GA is a population-based search engine method because every generates a novel population through repeating three fundamental population operations: Selection, crossover, mutation. The genetic algorithm is frequently employed in a variety of disciplines, In addition to computer network [17], recognition of voice[18], image processing, systems engineering[19], and so on.

The following is a basic steps for the genetic algorithm works:

a. Create an n-person random population with potential solutions (initial populations).
b. Determine the f (x) each population member's fitness value.
c. To generate a new population, repeat the first three procedures until they are finished.
d. Select two persons from the current generation to mate with.
e. Use a certain ratio of crossing to produce offspring.
f. Use certain mutation ratio.
g. Repeat the preceding processes until the completion requirement is reached.

1.4 MOTIVATION

One of the fundamental problems in many engineering and scientific fields such as optimizing transportation logistics planning, delivery, industrial design, tourist trips, is problem of travelling salesman.

The motivation for the study is due to several factors, including:

a. Efficiency improvement: The genetic algorithm helps to improve the efficiency in solving the problem faster and more accurately compared to algorithms.
b. Practical applications: Solving the travelling salesman problem has wide-ranging practical applications in industry, commerce, logistics, etc., which makes the genetic algorithm useful in these areas.
c. Scientific research: Solving the travelling salesman problem involves various scientific challenges in artificial intelligence, computer engineering, and mathematics, which makes genetic algorithm a popular subject for scientific studies and research.
d. Continuous improvement: Genetic algorithm solutions developed solve TSP can be used as a basis for improving other algorithms used to solve other related problems, leading to the development of more effective techniques in the future.

1.5 PROBLEM FORMULATION

As is the case with certain TSP formulas that employ an increased set of variables compared to another formula, characteristic formulations for the same issue may typically be written in terms of many combinations of variables.

Mathematically, let c= {c1, c2, cn} be the n set of cities to be visited, and define f=a (i , j) as the m by m matrix being the length of the distance among two spot. The idea is to find a permutation p = (p1, p2, ..., pn) of the indices {1, 2, ..., n} that will reduce the total distance of the round. That is, the TSP seeks to find the solution:

$$\text{Reduce } \sum i = 1 \; to \; n - 1 \; di, p(i), p(i + 1) + dn, p(n) \qquad (1.6)$$

Subject to:

Every city should to only be accessed once.

Then you must return to the first starting point.

Where pi is city visited in the tour, and di, j rrepresent total the separation of the cities(i and j).TSP consists of finding the cheapest circuit or ride in the hamiltonian [20].

1.6 THESIS STATEMENT

The propose of this work is to find a solution to the problem of travelling salesmen, therefore the thesis statement is “how efficient is using genetic algorithm for solving travelling salesman problem in terms of total distance, access time, and number of cities”?.

1.7 AN OVERVIEW AND DYNAMIC ON THE GENETIC ALGORITHM

Every organism is made up of cells, and all cells contain that same set of chromosomes, which are strands of DNA could serve as an example for the complete microorganism, or any chromosome composed of blocks of genes, and each gene loads a specific protein. Each gene has its own location in the chromosomes called the locus.

1.8 CONTRIBUTION OF THIS STUDY

The aim of this study is to propose a new solution to the classic combinatorial optimization issue that has captured the attention of many researchers over the past years.

The approach used in this thesis presents a modern proposal for a genetic algorithm that contributes to the development of the algorithm's work efficiency and performance.

1.9 STUDY SCOPE

In this study, the ranges are as follows:

a. Appropriate matrices will be used to show the details of the TSP.

b. GA will be applied to solve a problem Specifically the TSP.

c. The genetic algorithm will find the final answer in a short time.

As mentioned earlier, in this study, to be able to identify most effectively answer for the problem, shall rely on genetic algorithm's optimization. The speed of implementation of the solution, and therefore it may take to achieve the final result little time. Hence, the genetic algorithm is used in the end, in order to speed up process and find an accurate answer quickly.

1.10 PROBLEM BACKGROUND

The TSP eventually acquired prominence as the prototype of a difficult issue in combinatorial optimization. As the needs to use artificial intelligence in every field are increasing, different algorithms are being developed and deployed to tackle this basic problem in the best way possible. It should be highlighted, however, that many of the present algorithms have major shortcomings. Some only work for very tiny difficulties, some fail substantially when it comes to precision, as detailed in Chapter 2 we hope that a method that is faster and more accurate than many of the existing methods can be developed[21][22].

1.11 ORGANIZATION OF THE THESIS

This thesis is divided into five chapters. Chapter 1, which include introduction to the topic that includes a summary of the problem that prompted us to do the research, scope, motivation, thesis statement and contribution of study. In chapter 2, the background to this problem, previous studies by researchers, and the methods that are used to solve this problem are discussed. Variations and applications of TSP, traditional techniques for solving TSP, and population based algorithms for solving TSP, which are important factors in the proposed solution, are covered. In chapter 3, the methodology, which include approach used in this solution, the mechanism of action of the genetic algorithm, and how to apply it to solve the TSP were discussed. Chapter 4 includes details of the data sets that are used. The results of the methods obtained from the different tests are presented and compared the result of traditional genetic algorithm and modify genetic algorithm described in this study. And finally, chapter 5, which contain conclusion and future work.

2. BACKGROUND

2.1 INTRODUCTION

Here, previous studies will be presented, and how to deal with this problem in different ways for several researchers. This research will also address the method that was used to solve this problem, and it will discuss the problem of this research, which is the problem of the travelling salesman, and the different methods used to solve this problem and clarify the advantages and disadvantages of each method.

2.2 DALING WITH TRAVELLING SALESMAN PROBLEM

This problem has attracted the attention of researchers for many years because the field of development and modernization is easy and open source. It entails identifying the hamiltonian cycle with the lowest cost among a group of cities. The issue may be summed up as follows: Find the distances between a specified number of cities. Choose the quickest path feasible to go from one beginning city to all the other cities, stopping in each of them precisely once. The goal is to reduce the overall cost of travel [12]. To make it easier to solve the problem, we create formulate it mathematically and represent it using a graph that depicts the cities and the possible connections between them. By applying genetic algorithms to these equations, the optimal and shortest route can be obtained, resulting in time savings for touring every city and then heading back to the starting spot [23]**.**

2.3 NP-HARDNESS

NP-hardness is a concept in theoretical computer science that indicates the difficulty level of solving an arithmetic problem if it is at least as difficult as the (non-deterministic polynomial time) in other words, each NP-hard task may be reduced to an NP-hard polynomial problem, among the most difficult are NP-hard problems, travelling salesman problem, a graph colouring problem. These important a variety of fields, including computer science, operations research, and mathematics, and finding efficient algorithms to solve them is a major area of research [24].

It should be noted that not all problems are necessarily difficult in NP. Some NP-hard problems may require an exponential time to solve even to check the solution, and they are said to be in the class of problems called exptime. Among most actively researched problems

involving combinatorial optimization is TSP, and many algorithms and approximate inferences have been developed to efficiently find good solutions. However, even with these techniques, the problem remains difficult to solve for large cases, and a true optimal solution is generally possible only for small cases [25] .

2.4 LP RELAXATION

LP formulation for TSP[26] . The following:

$$minimize \quad c^t x \tag{2.1}$$

$$subject\ to\ 0 \leq x_e \leq 1 \qquad for\ all\ edgs\ e \tag{2.2}$$

$$\sum (x_e : v\ is\ edn\ of\ e) = 2 \qquad for\ all\ v \tag{2.3}$$

The decision variable (XE) in this formulation indicates the decision to include edge e in the tour. This decision variable is associated with a cost matrix by the objective function. Whereas the restrictions guarantee that each edge is only utilized once and that each node has two edges". However, the authors go on to highlight how this formulation is not the true problem they wish to tackle, but rather the problem they can solve. The formulation above is a relaxation of the real issue that allows for solutions using sub-tours as well as partial edge assignment.

2.5 APPLICATIONS OF THE TSP

The travelling salesman problem (TSP) has several uses, like:

a. Manufacturing: TSP may be used to optimize manufacturing operations such as machine scheduling and material transportation between workstations.The TSP has been used in a variety of fields such as VLSI chip production, robotics [27]. Where the TSP may be used to optimize the path of a robot visiting a collection of places, minimizing the overall distance travelled or the time necessary to complete the task.
b. Tourism: The TSP may be used to create optimal tourism itineraries, decreasing overall distance travelled or time necessary to visit a group of tourist locate. These are only a handful of the TSP's numerous uses. The topic is a fundamental optimization problem that arises in many fields of study, and developing efficient algorithms to solve it has significant practical ramifications [27].

Transportation and logistics: The TSP may be used to optimize delivery routes for a fleet of vehicles, reducing overall distance travelled or time necessary to deliver products to clients.TSP is useful for designing optimum network topologies for communication networks, power grids, and transportation networks and vehicle routing [28].

2.6 METHODS FOR DEALING WITH THE TRAVELLING SALESMAN

Nowadays combinatorial optimization is one of the most active and important areas in theoretical computer science, operations research and discrete mathematics. The combinatorial optimization poses a strong challenge to solve large complex problems arising in the real world and seeks to improve the quality of life of human beings. For example, the GPS device in our car refers to a pure combinatorial optimization problem aims to find the shortest path between two locations. Design of efficient networks is highly desired in telecommunication and distribution networks, such as gas, water, electricity, and delivery services, it is one of the extensively studied topics in combinatorial optimization field. Transportation, logistics, production, finance, energy, environment and sustainable development, health care, etc., it is hard to find an aspect of our modern-day real-life whose do not critically rely on the solution of one or more combinatorial optimization problems. The routing problems, such as travelling salesman problem (TSP), and vehicle routing problem (VRP) are considered the most used problems in the field of combinatorial optimization specifically in transportation and distribution logistics[29].

These are a few of the most prevalent approaches:

a. Greedy algorithms: produce locally optimum decisions at each stage of the solution creation process. For example, the closest neighbor algorithm begins in a random city and proceeds to the nearest unvisited city. Although this approach is quick and produces decent results, it does not guarantee an optimum answer.
b. Genetic algorithms: These algorithms simulate natural selection by producing a population of candidate solutions and iteratively improving them via mutation and crossover processes. This approach can yield high-quality results, but it is computationally costly.
c. Linear programming: In this technique, the TSP is written as a linear program and then solved using specialized methods. While this approach can yield optimum solutions for small issue sizes, its high temporal complexity makes it unworkable for larger inputs.

There are several different approaches and variants of these methods for solving the TSP, and the method used is determined by the size and needs of the given issue.

2.7 TOUR CONSTRUCTION HEURISTICS

Heuristics used to generate a preliminary a combinatorial optimization's resolution problem, like TSP, are referred to as tour creation heuristics. These strategies are frequently used when it is unfeasible to discover an ideal solution within a reasonable timeframe or when the problem's size makes it difficult to find an optimal solution.[30].
Here are some examples of tour construction heuristics:

2.7.1 Nearest Neighbor Algorithm

One of the algorithms that dealt with the TSP is the nearest neighbor (NN) algorithm. The process begins with the salesperson starting from a specific city and then proceeding to the next closest city. This is followed by moving to the closest unexplored city and repeating the same procedure once all cities have been visited. Finally, the salesperson returns to the initial city.

The algorithm can be divided into the following stages:

a. Randomly select city.
b. Locate and visit the nearest unexplored city.
c. Determine whether any cities remain unexplored. If so, return to point two.
d. Return to the starting position.

Following theorem[31]. shows how this approximation strategy could have significantly bad conduct.
Theorem: Any m-city cases I of triangle inequality,

$$nn\,(i)\leq\ (log_2 m+1)OPT\ (I) \tag{2.4}$$

Additionally, there are m-city instances for arbitrarily high values of m.

$$nn(i)>(log_{\ 2}(m+\frac{4}{3})\ OPT(I) \tag{2.5}$$

The nearest neighbor algorithm's performance definitely leaves a lot to be desired. It would be interesting to get similar guarantee values for this alteration using the aforementioned theory. The NN method is not promising conceptually, yet it performs well in practice[32].

2.7.2 Greedy Algorithm

The greedy algorithm is a simple and intuitive algorithmic model used to solve optimization problems. It takes a prescriptive approach by making optimal choices locally at every step with the hope of finding the best current options. The basic idea behind a greedy algorithm is to make the best possible decision at each step, without considering the overall impact on the final solution. The algorithm builds a solution incrementally, making a series of choices that seem the best at the time. A greedy strategy is often used when a problem can be solved by making a series of decisions, and each decision affects subsequent decisions. A greedy algorithm for solving a TSP is a simple, intuitive approach that provides a reasonably good solution, although it may not always perform to the optimal solution.[33].

Here's how the TSP's greedy algorithm works:

a. Formation: Start with a random city as a starting point.
b. Choose Nearest Neighbors: From the current city, choose the nearest unvisited city as the next city to visit. If several cities are at equal distances, any one of them can be chosen.
c. Tour update: Add the selected city to the tour and mark it as visited.
d. Repeat: Repeat steps 2 and 3 until all cities have been visited.

2.7.3 Firefly Algorithm

One of the most important modern algorithms used to solve the TSP is the firefly algorithm and its application to route planning problems, the vehicle routing problem, and the TSP swarm intelligence is intelligence based on collective behavior in decentralized systems one of the algorithms based on swarm intelligence is the firefly algorithm, firefly algorithm is widely used to solve optimization problems, many researchers jointly develop the standard firefly algorithm to solve the problems encountered due to the different characteristics of the problem, we explored the different firefly algorithm to solve their common characteristics such as route planning, car steering problem, and TSP which are path, time and distance optimization[34]. The (FA) method offers a number of advantages, including its simplicity, ease of implementation, and ability to find several answers at the same time, it may, however, suffer from early convergence and require a high number of iterations to get an ideal solution the (FA) method used in more range of optimization problems, including function

optimization function optimization, feature selection, picture segmentation, and machine learning algorithm parameter tweaking, it has also been integrated with other optimization techniques to increase its performance, such as local search and differential evolution [35].

2.7.4 Hybrid Genetic Algorithm

A hybrid GA used as a technique for incorporation of GA with the local techniques to solve the TSP in efficient manner, the role of the genetic algorithm is to choose best individual solution by satisfying a reverse sequence mutation, tournament selection, and an ordered crossover as well as the initial solution has been select randomly in the very beginning of the algorithm. In contrast, the role of the local method is to escape from local optima by acting and generating some action to improve the over hand solution and trying to reach the global optimal solution or sub global optimal solution depending on the scale of the problem. [36]. It works by iteratively generating and testing candidate solutions, selecting the best ones, and breeding them to produce better solutions genetic algorithm hybrids combine the genetic algorithm with other optimization techniques, like local search, artificial annealing, or gradient descent, to be able to increase search process's effectiveness and effectiveness, hybrid algorithm may use the genetic algorithm to investigate the search area and identify promising regions, and then switch to the other optimization technique to refine the solutions in those regions[37].

There are many different ways to combine the genetic algorithm with other optimization techniques. For example, one strategy is to utilize a genetic algorithm to create an original population of potential solutions before using local search to enhance those answers, using a genetic algorithm is an additional strategy to identify promising regions in the search space, and then use simulated annealing to explore those regions more thoroughly[38] .

Genetic algorithm hybrids are useful when the problem being solved is complex or high-dimensional, making it difficult for a single optimization technique to find good solutions, by combining multiple techniques, genetic algorithm hybrids can take advantage of their strengths while mitigating their weaknesses, resulting in more efficient and effective optimization [39].

2.7.5 Method of Nearest Entry

This method involves selecting a member of the inference family, where you start with a short tour by choosing one city, and then the cities are selected to start in the first round and each time depending on whether all the distinction in length among both towns is shorter than city that the seller will visit. In other words, heuristic for closest insertion selects the original node of the city that follows the starting point, then the city closest to the itinerary, city one will be added if the distance is less between city one and between cities next city and next city, where distance is in the round one that has not started yet. Ultimately, the choice is based on cost[40].

2.7.6 The Method of Listing in Random

The insertion method is not inherently random, but there are variations of the insertion method that use randomness to improve the quality of the solutions produced. These methods are often called randomized insertion heuristics the insertion method is a tour construction heuristic that works by iteratively inserting cities into an existing tour to improve its length there are several variations of the insertion method, including the distinction in length among both towns is can produce good solutions, but they may become trapped in local optimum and fail to find the universal best practice, to address this problem, randomized insertion heuristics use randomness to choose the city to insert into the tour. For example, the random insertion algorithm chooses a random city to insert into the tour, rather than the nearest or farthest city. By using randomness, these algorithms can explore more of the solution space and potentially find better solution, however, it's worth noting that randomized insertion heuristics are not guaranteed in order to determine the best answer to the travelling salesman dilemma. Like all tour construction heuristics, they are designed to find a good initial solution quickly, but they may not find the global optimum in all cases. More sophisticated algorithms, such as metaheuristics or approximation algorithms, may be needed to find better solutions for larger or more complex instances of the TSP[41].

2.7.7 The Method Most Distant Insertion

Farthest listing method starts by selecting two cities that are farther away from each other and adding them to the tour. Then, at each step, the algorithm identifies town farthest derived

from current round and incorporates it into the that spot results in the least increase in the length of the round. The procedure is repeated unless all cities are now included to reunite.

This is a succinct summary of the algorithm:

a. Choose any two cities to start the tour.
b. Determine the separation among this current rounds and all previous cities.
c. Select city farthest from the current round.
d. Enter the selected city at the position that results in the least increase in tour length.
e. Repeat steps 2 through 4 until all cities have been added to the round.

The most dimensional insertion method is the greedy algorithm, which means that it makes the optimal choice locally at each step without considering the overall impact on the solution. While this algorithm is not a given that find the most practical response to TSP, it often produces good results that are computationally efficient[42].

2.7.8 Christofides Algorithm

This algorithm proposes a question about the possibility of giving a fixed ratio polynomial. Christofides developed this method [43] .In 1976, when he proposed that the ratio be (12) to (1.5). After testing results show that in terms of performance guarantee, this is the best algorithm. I first give a basic approximation approach and prove that it yields a round with a length no more than twice as long as the best solution, and then show how this algorithm has modified the scientist Christofides to provide an appropriate approximation of this ratio of (1,5) to TSP. He builds a low-cost spanning tree to select the minimum as the travelling salesman's starting point. Where the cost of tree is less than the value (in terms of cost) of the ideal trip, because it is considered the optimal tour enhancing solutions [44].

2.7.9 Method of Enhanced Solution

Through previous experiences, it has been found that the way rounds work can be rather mediocre. Although these experimental approaches may be effective in certain situations, in general they are not sufficient. On the other hand, there is an opportunity to improve mobile tours developed using tour generation heuristics. The focus should be on improving the local inference of the TSP problem. All of these experiments mainly depend on mutational exchange operators. This action is an action that focuses on each of the adjacent rounds, i.e.

those rounds that are only one step away. The algorithm constantly repeats these steps by turning the current round into a better neighbor round until there is no such round in the vicinity. That is, we have come to the optimal solution locally. However, since the amount of reductions obtained through flight changes depends on how long these methods take, they cannot be constrained by the worst case polynomial. Assuming an integer input, the worst case is that each optimization step reduces the round length by one unit, so the run time is determined by the length of the first and last round[45]. Despite this, these algorithms are frequently used because they give very good results in "real world" situations. The following conclusions differ only in how close rounds are determined.

2.8 EXTENSIVE LITERATURES

J. Bi and other [46].Suggest that a new technique was used in the genetic algorithm in which the author used the greedy switching method for the new community composition in the TSP problem. This system creates genetic algorithms. This algorithm adopts a technique called greedy permutation, which is much better in terms of reliability and effectiveness compared to stochastic approaches in generating the first generation, as this algorithm is used in many issues, the most important of which is solving the TSP problem and superior results were obtained when compared to the previously used approach.

C. Tungom, and other [47]. Presented a novel solution of TSP which was used in order to obtain mature conformations, for which mutation and exploratory crossover were used. These solutions improve concurrent performance while reducing the number of generations. According to empirical data, due to its heuristic performance, these algorithms generate the best results in overcoming the seller's problem, thus obtaining the best possible solution. When this algorithm was implemented, there were no complications.

A. Hameed and others [48].Presented a method for determining the optimal TSP solution using power optimization in GA. In this approach, calculate that euclidean distance between the two cities first. Then the chromosome is counted by coding the values. Moreover, roulette wheel selection was used to obtain a perfect chromosome. Then the method was repeated several times to achieve the best results in the evolutionary algorithms.

J. Stastný and colleagues[30] .Presented a new strategy for improving TSP. In this paper, a graph-based algorithm was used to solve scheduling problems based on lifetime planning.

This was used to calculate the seller's travel route from one city to another. The author compared this method with the traditional GA method commonly used in TSP and noted that it achieved superior results in terms of flight distance, time taken to obtain the result, etc. This method, according to the author, gives better results than GA in less time.

I. B. K. Widiartha, and colleagues [49] .Presented a strategy in which the multicomponent genetic algorithm was applied challenge NP problems one of them TSP. It was a good pick from the others. In the crossover segment, three methods were used: Opaque crossover, map-limited crossover (PMX) (OX), and circular crossover. Multiple item fit function data were calculated using height, fixed obstacles, and non-fixed obstacles. This proportion of the fitness function is divided into three components (70% distance, 30% static hurdles, and 15% for non-static hurdles). This investigation revealed reduced barriers. However, the need for a shorter distance was necessary.

S. Dharm [50]. The author used the greedy method in the genetic algorithm to initialize the population in this study.

Phylogenetics has been used to identify and coordinate different genetic factors. The use of this technique in the results increased the values of individual tasks, speed of improvement, and rate of optimization.

Xu, Haiyang and colleagues[51].Instead of the usual genetic algorithm, an invasive weeding method was used for a random search, in which the normal distribution sequence acts as a random number generator. To address the TSP, a library was used in this study to read the genetic components each time. The results of this work indicate that when solving a TSP, it is necessary to have a high level of convergence speed and accuracy. Full implementation of this functionality in MAT LAB.

Yu. Xiao Feiand colleagues[52].TSP was addressed by a new. The firefly algorithm was used to discover the TSP solution. The author used k mean clustering for node deflection in the first stage of this approach. Sub problems are developed as a result of these sub problems. In the second stage, in each cluster, the best and best path was detected using FA, and the final step was cluster recombination and path recombination between them. Compared to other methods used in the TSP, this strategy produces the best results.

Z.Ahmed and other[53]. This paper develops a new crossover operator, sequential constructive crossover (SCX), for a genetic algorithm that generates high quality solutions to the traveling salesman problem . The sequential constructive crossover operator constructs an offspring from a pair of parents using better edges on the basis of their values that may be present in the parents' structure maintaining the sequence of nodes in the parent chromosomes. The efficiency of the SCX is compared as against some existing crossover operators; namely, edge recombination crossover (ERX) and generalized N-point crossover (GNX) for some benchmark TSPLIB instances. Experimental results show that the new crossover operator is better than the ERX and GNX.

Researcher Jain et al[54]. Applied Jedidi's method of intersection technique. In the genetic algorithm where the using the greedy approach in working to create a new generation Through this operator. As the concept of nearest neighbor used by the author for the development of the algorithm work showed good new results compared with previously used intersection techniques.

3. METHODOLOGY

3.1 INTRODUCTION

Genetic algorithms consist of a group of individuals that make up the community. And the genetic algorithm creates multiple steps called generations, during each generation a new community is formed consisting of a group of individuals called offspring, resulting from the previous community. As the resulting new offspring is configured based on a set of operations for the genetic algorithm selection, crossover, mutation, then the problem is calculated by adopting an equation for the problem called fitness function[55] .

In this chapter new methods of crossover and mutation are proposed which will be explained in this chapter.

One of the search methods that depends on the repetition approach the genetic algorithm, so it is better than other optimization search techniques such as linear programming, especially when the problem space is very large[56].

3.2 GENETIC ALGORITHM'S ESSENTIAL STEPS

The fundamental techniques for genetic algorithm can be represented by the following steps:

a. Proper and optimized coding. The primitive consists of the set of chromosomes that in turn arise from population group
b. The random generation of a population of genes, as each chromosome (which is a gene pool) is a solution.
c. Calculate each's value (resolved) chromosome of the chromosomes. Each method of validity evaluation is supported a solution in the next generation of the perfect solution that we are looking for.
d. Choosing a group of interconnected groups to generate the new generation the network of theories and communication in this field, and the selection procedure mostly relies on the correctness of solution in association follow it, follow it
e. Selection process followed by cross-linking and mutation.
f. Then we go back from the step number

figure 3.1 below explains how the genetic algorithm functions.

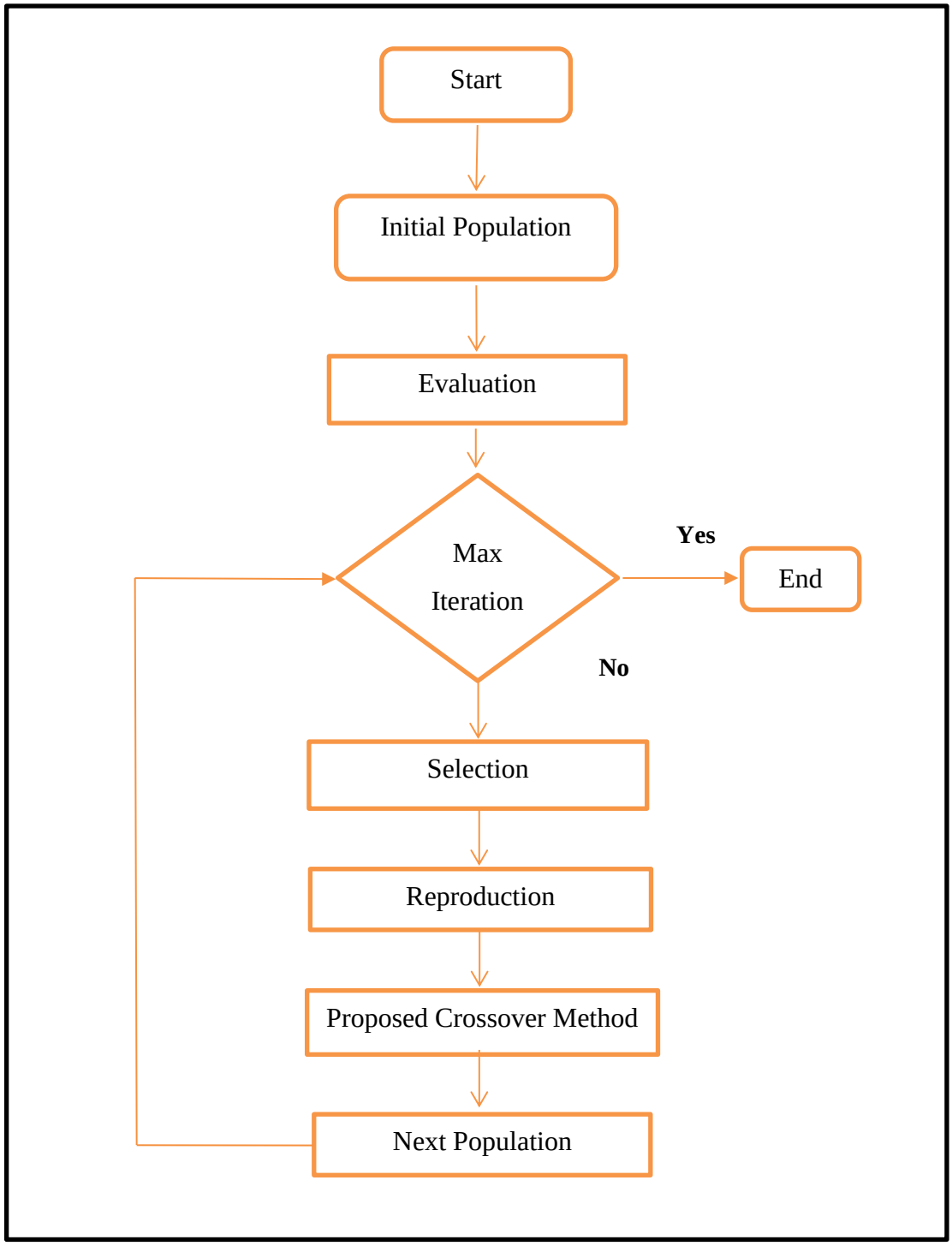

Figure 3.1: General Steps of Genetic Algorithm [57].

3.3 GENERAL STEPS OF GENETIC ALGORITHM

The genetic algorithm's steps work will be explained:

3.3.1 Great Initial Population

Basic community generation process is the first step in the genetic algorithm. Random chromosomes are generated as far as population size, and chromosome is represented in special way according to the nature of the problem. Generation of the basic generation is the optimization point in solving the complex problem. The initial set consists of a set of chromosomes, either randomly generated using the random generation function in (MATLAB) software, or by selecting a fixed number in each experiment. A fixed number was chosen in this research and experiments were conducted on multiple cities. The base generation is the point of improvement in solving the complex problem[58].

3.3.2 Objective Function & Fitness Value

The objective function, also known as the fitness function, is a mathematical representation of the problem's goal. It quantifies how well a particular solution addresses the problem. In optimization problems, the objective function is used to evaluate the quality of a candidate solution. It takes the solution's parameters as inputs and produces a single scalar value as output. For example, in a simple optimization problem where you want to maximize a function. In more complex problems, the objective function can represent various real-world criteria, such as cost, efficiency, accuracy, or any other measure relevant to the problem domain. The fitness value is the result obtained when the objective function is evaluated using a specific candidate solution (also referred to as an individual or chromosome). In the context of genetic algorithms, individuals with higher fitness values are considered better solutions or have a higher likelihood of producing better offspring in the evolutionary process. For example, if you're minimizing a cost function, individuals with lower fitness values lower costs are considered better solutions. Conversely, if you're maximizing a performance metric, individuals with higher fitness values are more favorable. During the evolution process of the genetic algorithm, individuals with higher fitness values have a greater chance of being selected for reproduction (mating) and passing their genetic information (representing good solutions) to the next generation. This selection pressure promotes the exploration of better solutions over successful generations. By iteratively

evaluating the objective function and calculating fitness values for different candidate solutions, the genetic algorithm can identify the best TSP solutions, aiming to find the shortest tour that visits all cities exactly once [59].As for the fitness value for each member of society, according to how valuable the goal function is:

a. Decrease the objective function by calculating the suitability range as

$$fitness\ (i) = -1 * objective\ function\ (chromosome(i)) \quad (3.1)$$

b. If objective function of problem (maximize), then value of fitness range the objective function will have the same value for each community member[60], and it will be as

$$fitness\ (i) = objective\ function(chromosome(i)) \quad (3.2)$$

The working of the fitness value code is explained in the figure below.

```
function best = findBest()
    %FINDBEST Summary of this function goes here
    %   Detailed explanation goes here
    global POPULATION
    global POPULATION_N
    global STATS

    max   = STATS(POPULATION_N + 3, 3);
    best  = POPULATION(1, :);
    dbest = distanceForPath(best);
    for i = 2 : POPULATION_N
        path = POPULATION(i, :);
        dist = distanceForPath(path);
        if (dist < dbest)
            best = path;
            dbest = dist;
        end
    end
end
```

Figure 3.2: Code of Fitness Value.

3.3.3 Selection

Selection refers to the process of selecting individuals from the population (parents) for mating and producing a generation new, as there are types of selection methods that depend

on the basis of the purpose of the problem. Two types of selection models can be categorized, the first known as proportionate selection it means the process of selecting individuals from the community based on the fact that the fitness value is a relative value to the fitness values of the rest of the individuals in the community, and the second known as ordinal selection It refers to the process of selecting individuals from society based on their rank in society and not on the value of fitness[61].

3.3.3.1 Roulette wheel method of selection

In proportional roulette wheel, individuals are selected with a probability that is directly proportional to their fitness values i.e. an individual's selection corresponds to a portion of a roulette wheel. The probabilities of selecting a parent can be seen as spinning a roulette wheel with the size of the segment for each parent being proportional to its fitness. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The fittest individual occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel. The circumference of the roulette wheel is the sum of all fitness values of the individuals roulette-wheel selection is a frequently used method in genetic and evolutionary algorithms or in modelling of complex networks. roulette-wheel selection algorithm, which typically has complexity and is based on stochastic acceptance instead of searching[62]. When use GA with the roulette wheel method is a powerful approach, and finding an optimal solution for large instances remains computationally intensive and may not always be feasible within a reasonable amount of time. Therefore, approximation and heuristic methods are commonly used to tackle real-world instances of the TSP effectively, the roulette wheel method ensures that better solutions have a higher chance of being selected and, consequently, are more likely to be passed on to the next generation. This increases the probability of convergence to a better solution compared to purely random selection methods, the selection process using the roulette wheel relies on the idea of picking people from a population according to how fit they are each person is given a fitness range value, where a greater fitness range value translates into a higher likelihood of selection [59], see figure 3.3.

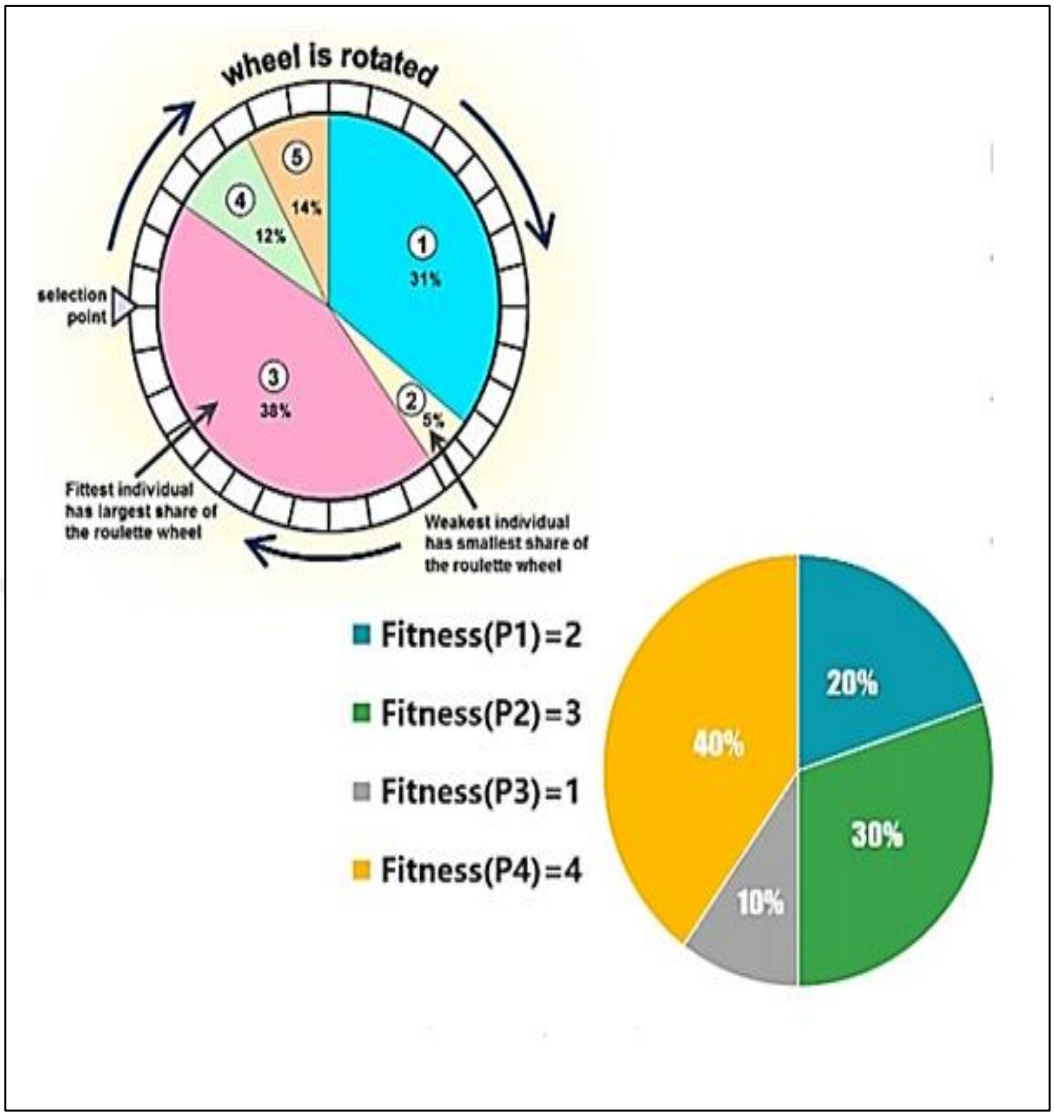

Figure 3.3: Roulette Method for Selection Best Value for Fitness[63].

3.3.3.2 The method of selecting ranks

To address the TSP, several methods and algorithms exist to obtain an optimal or close-to-optimal solution. One widely used approach is the "rank-based" method, which entails assigning ranks to individuals in a genetic algorithm's population.

Below is a general overview of how the rank-based selection method can be utilized to solve the TSP using a genetic algorithm:

a. Initialization: Generate a starting population of potential answers (also known as individuals or chromosomes). Each person stands for a possible TSP resolution.

Calculate each population member's cost or fitness for the population. The inverse of tour length can have used to quantify fitness in the context of TSP.

Assign a rank to each person based on their level of fitness. People who are more physically fit typically obtain lower ranks. You can utilize a variety of ranking techniques, including linear ranking, exponential ranking, and tournament selection.
Rank-based selection of parents for the following generation. Parents are more likely to be chosen from those with lower positions.
b. Reproduction: Produce offspring for the following generation using carefully chosen parents. In this step, we use the crossover to generate new potential generations.
c. Replace: Introduce some of the freshly produced children into the present population. This stage makes sure that throughout generations, the population develops toward better alternatives.

By applying the rank-based selection method within a genetic algorithm framework, the population gradually evolves, and the quality of solutions improves over time.

In this method of selection, the value of the fitness range is regulated based on a method that starts with the minimum fitness range then ends with maximum fitness range, so each chromosome has an arrangement within the community that is called the order of the ch (i) chromosome, as the rank of the ch (i) chromosome is subject to For the condition as per the equation [64]:

$$1 \leq rank\left(ch(i)\right) \leq population\ size \tag{3.3}$$

After calculating the target ratio (target sampling rate) for each chromosome as per the equation:

$$tsr(i) = \min + (max - min)[(rank(i) - \frac{1}{population\ size - 1}] \tag{3.4}$$

3.3.3.3 Binary selection

The possible solution is represented as a binary string in this way, with each bit indicating whether or not a city is included in the current round. The primary goal is to create random set binary sequences (candidate solutions), evaluate their fitness (total mileage), and improve the results over and over again depending on genetic factors, the most important of which are crossover and mutation [65].

Depending on their physical fitness, the selected operator selects the most suitable solutions from the population for the next generation. This is accessed via a selection algorithm, like

a roulette wheel. There are many ways for a crossover agent to combine two chromosomes to form a new chromosome. The new intercept is a typical crossover operator for binary strings where a random intersection point is chosen and then the mutation operator flips several bits in the binary string randomly to make a slight change in the answer, and the method continues until a good solution is discovered after a specified number of iterations. for medium-sized problems. Unfortunately, it may not always discover the best answer and may be slow when dealing with large-scale problems.

The binary selection methods are divided into:

3.3.3.4 Goldberg's method of selection

This method is based on developing a new method for binary selection that works in the following manner, choosing a fixed value (i) and then selecting two individuals from the community randomly, the individual with the lowest value for the fitness range is returned depending on the probability (i) since (i) is a value less than one.

In this code, the randi function is used to generate random integers between 1 and popsize. The ~= operator is used to check for inequality between ch1 and ch2.The constant value k is set to 0.5 in this example, but you can replace it with your desired value. The final selected chromosome is stored in the result variable based on fitness comparison and the random value R being less than K [66].

The working of the fitness value code is explained in the figure (3.4).

```
repeat
    Ch1 = randi(popsize);
    Ch2 = randi(popsize);
until Ch1 ~= Ch2;

if fitness(Ch1) > fitness(Ch2)
    result = Ch1;
else
    result = Ch2;
end
```

Figure 3.4: Code of Global Method Selection [66].

3.3.3.5 Whitely's method of selection

This method works on the basis of selecting the international community at random and then the person who scores the highest on fitness this code represents of Whitely's method of selection in this figure.

```
    Ch1 = randi(popsize);
    Ch2 = randi(popsize);
until Ch1 ~= Ch2;

K = 0.5; % replace 0.5 with the desired constant value

if fitness(Ch1) > fitness(Ch2)
    Min = Ch2;
    Max = Ch1;
else
    Min = Ch1;
    Max = Ch2;
end

R = rand; % generates a random value between 0 and 1

if R < K
    result = Max;
else
    result = Min;
end
```

Figure 3.5: Code of Whitely Method of Selection [66].

In this code the randi function is used to generate random integers between 1 and pop size. The ~= operator is used to check the inequality between ch1 and ch2 in a do-while loop. The final selected chromosome is stored in the score variable based on the fitness comparison.

3.3.4 Crossover

Crossover is a fundamental concept in genetic algorithms (GA), which is a popular optimization technique inspired by the process of natural selection Genetic algorithms are used to evolve solutions to complex problems by mimicking the process of natural evolution. In a genetic algorithm, a population of potential solutions (often represented as individuals or chromosomes) evolves over generations.

Crossover also known as recombination or mating, is a key operator in genetic algorithms that simulates the genetic recombination process that occurs during reproduction in living organisms, in other words, the mating process that takes place in the genetic algorithm can be defined as the pairing of a pair of parents to form children, and in the process of mating the genetic characteristics are transferred from the old community to the new community.

By repeatedly applying selection, crossover, mutation, and replacement, the genetic algorithm explores the solution space and gradually converges towards better solutions. The process mimics the way natural evolution selects and adapts individuals over time, making genetic algorithms effective for solving complex optimization problems in various fields such as engineering, finance, biology, and more [67].

The process of mating takes place randomly for the purpose of obtaining a new chromosome child. For the purpose of applying the mating process, the following is adopted:

a. Since the process of mating is the exchange of syllables between the parents to produce a child, therefore the exchange sites between the parents are determined, and the selection is done either randomly or in a special way imposed by the issue.

b. It must be taken into account that the exchange sites are an integer value confined between 1 and 1: Chromosome Length.

c. The Probability of crossover value is determined, on the basis of which the mating process is performed.

For the purpose of carrying out the process of mating between the parents to produce children, once the parents are selected, crossover is applied to create new offspring. During crossover, genetic material from two or more parents is combined to create one or more offspring. The idea is to mix the characteristics of the parents to potentially generate better solutions.

We can represent code of crossover method of selection in this figure.

```
function childrens = crossover( parents)
    %CROSSOVER Summary of this function goes here
    %   Detailed explanation goes here
    global POPULATION_N
    global POPULATION
    global parent1
    global parent2
    global parents
    pool = parents(randperm(size(parents,1)),:); % Shuffle
    childrens = zeros(size(POPULATION));

    % Crossover
    for i = 1 : 2 : POPULATION_N
        parent1 = pool(i, :);
        parent2 = pool(i + 1, :);

        childrens(i:i + 1, :) = cross(parent1, parent2);
    end
end
```

Figure 3.6: Code of Crossover Method of Selection [67].

3.3.4.1 One-piece mating

This method of mating is one of the easiest methods, as the cutting site is chosen randomly and the two new chromosomes are produced from the selected parents.

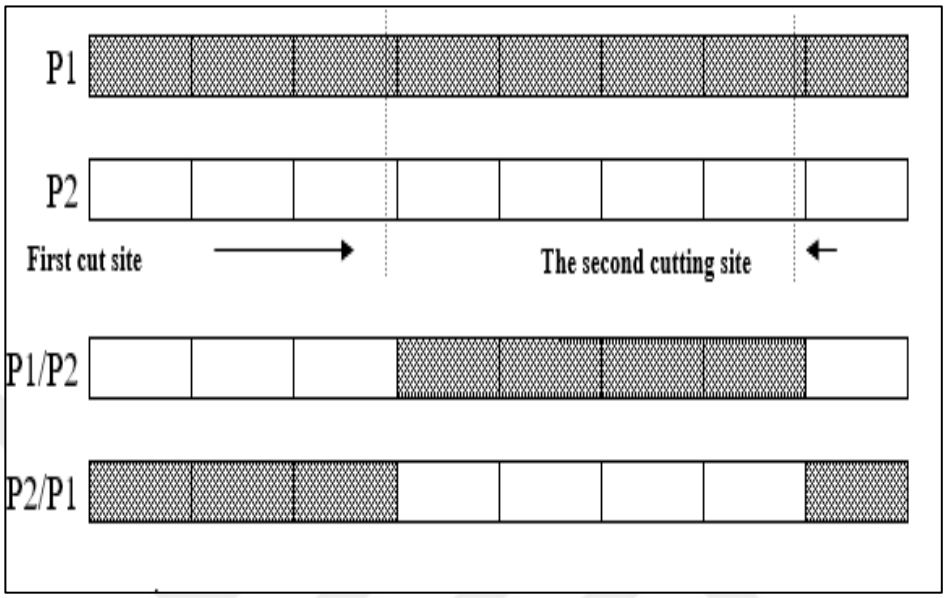

Figure 3.7: One-Piece Mating [68].

3.3.4.2 Two-piece mating

Mating takes place in this way by selecting two random sites for the two pieces, and the blocks are replaced between the cutting sites.

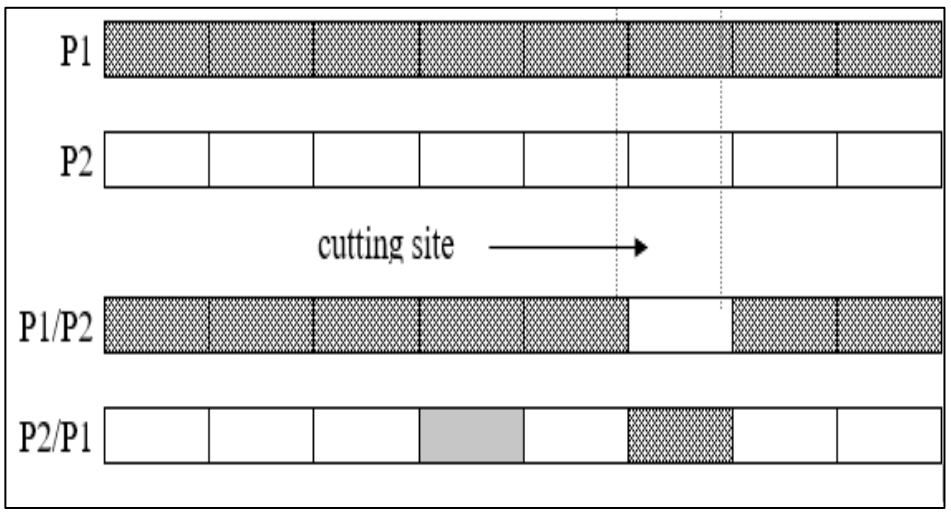

Figure 3.8: Two-Piece Mating [69].

3.3.4.3 Zigzag crossover

In this part, a new method was invented by the mating method, by which we obtained encouraging results compared to the previous known methods, and as we will explain this mating in the following way, as this method depends on the process of substitution between the parents to produce a child in the zigzag manner [70].

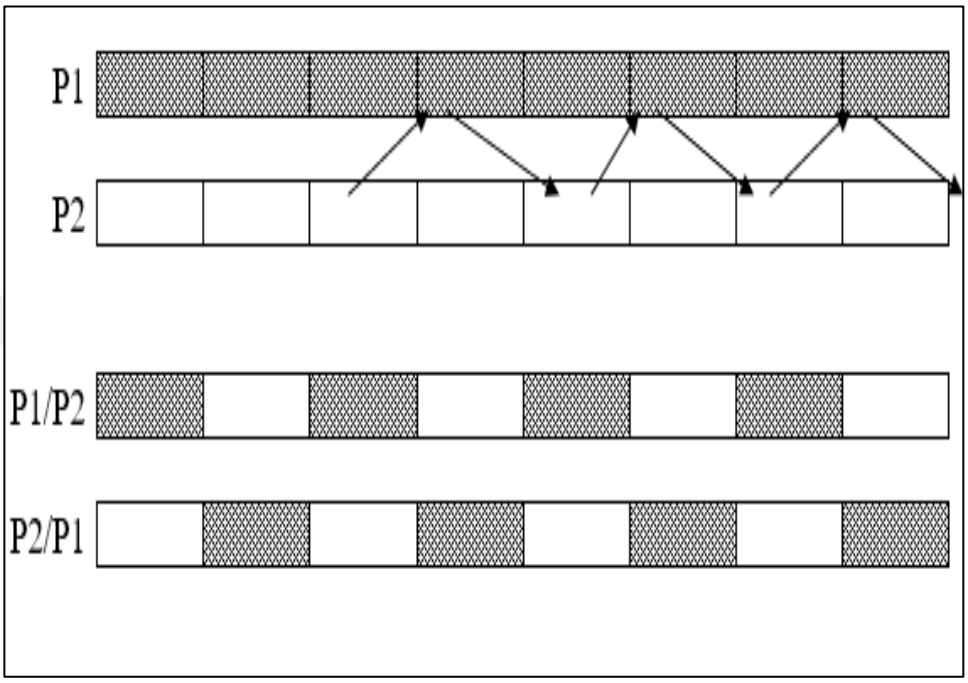

Figure 3.9: Zigzag Crossover [70].

3.3.4.4 Partially striped crossover (PSC)

Partial intersection is one of the most popular methods that relies on recombination to preserve intervals, and the (PSC) operator is one of the most frequently used operators for path type problems. There are many very small variables that have been written about and they are as figure (3.10).

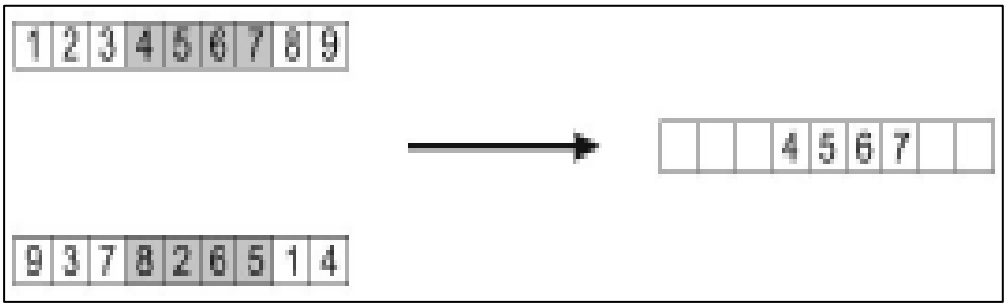

Figure 3.10: PSC Step 1 [71].

a. Copy the segment between two crossing sites from the first parent (P1) into the first offspring at random.
b. Beginning at the first crossover point, search for items in that section of the second parent (P2) that have not been replicated.
c. Check the offspring for each of them (let's say I to check which element (let's say j) has been copied from P1 in its stead.
d. Since we already have j in our string and are aware that we won't be moving it there, move I into the position held by j in P2.

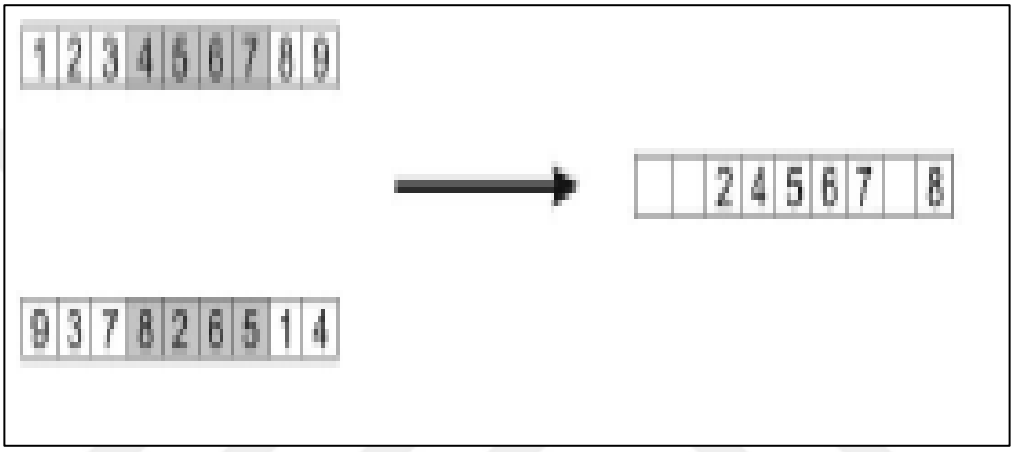

Figure 3.11: PSC Step 2 [71].

e. Place I in the position taken by k in P2 if j's location there has previously been filled in the offspring by an element k.

After dealing with the crossover segment's components, slot number (6) in this offspring may be filled from P2. In a similar way, but with the parental roles switched, the second child is formed.

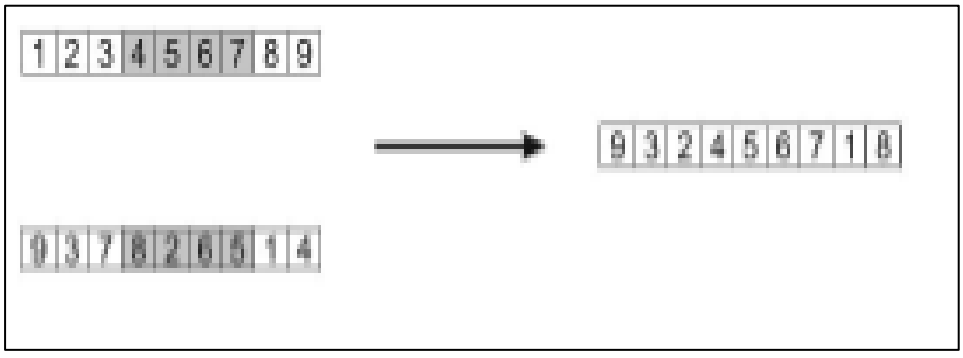

Figure 3.12: PSC Step 3 [71].

Number (6) out of the number (9) linkages that are present in the children in our case, when compared to the parents, are also present in at least one of the kids. Maintaining edges present

in both parents would be a desired attribute of any crossover operator. The edge, however, is present in both parents but absent from the offspring.

3.3.4.5 Order crossover

Order crossover (OX) is a genetic operator used in genetic algorithms to perform crossover (recombination) on two parent chromosomes to produce two offspring chromosomes. This operator is commonly used in permutation-based problems, where the order of elements matters, such as the traveling salesman problem or job scheduling, another factor that falls into the category of maintaining the interval is the intersection of the system. This factor is considered important and has more variables than (pmx) which indicated that the sequence of cities, rather than their exact location, is the only important aspect of the trip assuming that the sequence of cities (12345678) and(24687531)[15].

It creates children by moving away from one parent while maintaining the relative position of cities in the other. Where the change will be as follows, as there are two basic rounds.

Assume that the two randomly chosen cut-off positions are, respectively, located between both the (3) and (2) bit and even the (6) and (5) bit. respectively. Hence, we get it (12|345|678|) and (24|678|531)[71].

Now we produce the first generation as follows:

a. The offspring receives a copy of the flight segment between the cut points from P1. We discover.
b. The following cities repeated in the sequence in which they appear the second parent, going from the second point, and continuing from the second cut-off point of the first parent. We only skip a city if it's already mentioned in the offspring. When a parent's life ends, since the final and first cities on the itinerary are linked together, we start at the beginning. In the case of our example, this produces offspring (87|345|126).

By repeating the segment between the two cut off points of P2 and filling in the remaining bits in the P1 sequence, the second offspring is produced while respecting the parental responsibilities.

3.3.4.6 The proposed crossover method based on (flip, swap, and sliding) techniques

In this research, a new method of intersection was proposed, and this method includes merging the three techniques (flip, swap, and sliding) to apply the intersection process, where first the flip technique is applied to produce a new generation, then the swap technique is applied to the new generation, and finally the technique is applied sliding to produce the final generation, which is a combination of the three technologies.

After applying this technique in the program and after comparing the results, it was found that the performance of the algorithm improved in terms of time, distance travelled, and the best path without repetition in the different cases applied. The following code shows this method.

```
for k=1:4 %Mutate the best to get 3 new routes
tmpPop(k,:)=bestOf4Route;
switch k
case 2 %Flip
tmpPop(k,I:J)=tmpPop(k,J:-1:I);
case 3 %Swap
tmpPop(k,[I,J])=tmpPop(k,[J,I]);
case 4 %Slide
tmpPop(k,I:J)=tmpPop(k,[I+1:J I]);
otherwise
end
end
```

Figure 3.13: Code of Hybrids Methods [15].

3.3.5 Mutation

We consider reciprocal exchange mutation operator with specific mutation probability for our GA. The reciprocal exchange mutation selects any two genes from a parent and then exchanges them to form a new chromosome. However, this operator works with one chromosome only. It is used to increase the diversity of the population rather than to enhance the quality of the population., the charitable properties of the old society are transferred to the new society. The mating process takes place directly between two chromosomes Ch (i) and Ch (j) to reach a new chromosome (Child) a mutation is a simple procedure to change

or replace a specific value within the individual resulting from the mating process, and the specific value chosen for the purpose of replacing it is chosen in a manner random. A mutation is usually a procedure performed on an individual for the purpose of improving his genetic characteristics society the method of mutation differs according to the method of representing the chromosome. In the case of representing the chromosome in the form of a binary representing containing a set of (0, 1), the mutation replaces the location containing the value one to zero or vice versa. This method is called the Inverse. In the case of chromosome representation in the form of real value or in the form of integer value, the mutation adds a specific number to the value of the site to which the mutation is required. However, it's important to strike a balance between mutation and crossover. Too much mutation might disrupt the convergence process, while too little mutation might lead to premature convergence. The proper balance often depends on the problem at hand and requires experimentation to determine the optimal mutation rate.

In summary, mutation in genetic algorithms is a mechanism for introducing diversity into the population by making small random changes to individuals' genetic information. It's a vital component that helps the algorithm explore a wider range of potential solutions and can lead to the discovery of better solutions in complex optimization problems.

Figure (3.14) represents the code used in the mutation process.

```
    % MUTATION 1
    % Swap two random cities
    for i = 1 : length(children)
        child = children(i, :);
        len = length(child);
        for j = 1 : len
            if rand < p_mut1
                prev = child(j);
                index = randi(len);
                child(j) = child(index);
                child(index) = prev;
                children(i, :) = child;
            end
        end
    end

    % MUTATION 2
    % Exchange 2 paths
    for i = 1 : length(children)
        child = children(i, :);
        len = length(child);
        point = randi([2, len - 1]);
        if rand < p_mut2
            children(i, :) = [ child(point + 1:len) child(1:point) ];
        end
    end

```

Figure 3.14: Code of Mutation [65].

3.3.6 Genetic Algorithm Stopping Criterion

In the genetic algorithm, the process of forming new communities continues, and its aim is to improve the solution or approach the optimal solution, and the generation of communities is stopped when the stopping condition is met, and the stopping condition varies according to the nature of the problem, and among the most important and most frequently used stopping conditions:

a. Establish how many communities will be created.

b. fitness function obtains amount needed to solve the problem.

c. When constructing communities, it is important to repeat the goal function's value without altering it.

3.3.7 The Method of Representing the Chromosome

The problem is converted into a mathematical procedure as an initial action of genetic algorithm[23]. Encoding refers to process of marking the encoder's overall shape or appearance while translating an issue, and there are four widely used encoding techniques.

3.3.7.1 Binary encoding

Binary encoding is a technique used in digital communications and computing to represent data using a series of bits (binary digits), which can have only two possible values, typically 0 and 1. In binary encoding, each symbol or character represented in sequence bits, and sequence bits can be interpreted as a number or a character in the receiving system for example, the ASCII code is a type of binary encoding that represents characters using a sequence of 8 bits (a byte). each character is assigned a unique 8-bit code, which is used to represent the character in digital systems, binary encoding is widely used in various applications, including data storage, transmission, and processing.

It is a fundamental technique used in digital electronics, computer programming, and telecommunications. It is the simplest and most widely used type of encoding, as each chromosome and tag is represented as a text string of zeros and ones (0,1) [72].

Chromosome A: 0101101100010011
Chromosome B: 1011010110110101

3.3.7.2 Permutation encoding

Permutation encoding is a concept used in various fields such as computer science, cryptography, and optimization. It involves representing data using permutations, which are arrangements of elements in a particular order. In this encoding scheme, data values are not directly represented as numbers or characters, but rather as sequences of positions or indexes that indicate the desired order of elements. This type of notation is used in matters where there is no redundancy in the values of chromosome positions (order problems), since the chromosome is represented by a set of decimal numbers [73].

Chromosome A: 8 5 4 9 1 0 2 3 6 7
Chromosome B: 9 1 0 2 4 3 8 5 7 6

3.3.7.3 Direct values encoding

Direct value encoding is a simple and efficient method of data representation, as it allows for easy extraction of data values from their binary representation. However, direct value encoding may not be suitable for all types of data, especially for data with a large range of values, as it may require a large number of bits to represent all possible values. Direct value encoding is commonly used in digital signal processing, where data values are represented as binary codes for processing and transmission for example, a common encoding for integers is two's complement representation. Floating-point numbers are often encoded characters can be encoded using ASCII or Unicode. These encodings enable efficient storage, manipulation, and transmission of data while ensuring that values can be extracted accurately. It is also used in computer programming languages, where data types such as integers, characters, and boolean values are directly encoded as binary numbers this code style is utilized in issues. that contain a set of complex values such as real numbers that contain the decimal point, as binary coding for this type of problem is very difficult. Therefore, direct encoding of values is used to represent the chromosome and marker, as each chromosome and marker is represented by a set of values [73].

Chromosome A: Red, Black, Blue, Yellow, Red, Green
Chromosome B: 1.8765, 3.9821, 9.1283, 6.8344, 4.116, 2.192
Chromosome C: C A B K E D I H R F N W M S

3.3.7.4 Tree coding

Tree encoding is used when the solution can be represented as a hierarchical structure. It's common in problems related to expressions or decision trees. Each node in the tree represents a decision or value one mating point is chosen from both parents, and the parents divide at that point, and the sections are exchanged under the mating points in order to produce new offspring, and figure (3.15), (3.16) represents the tree mating process[74].

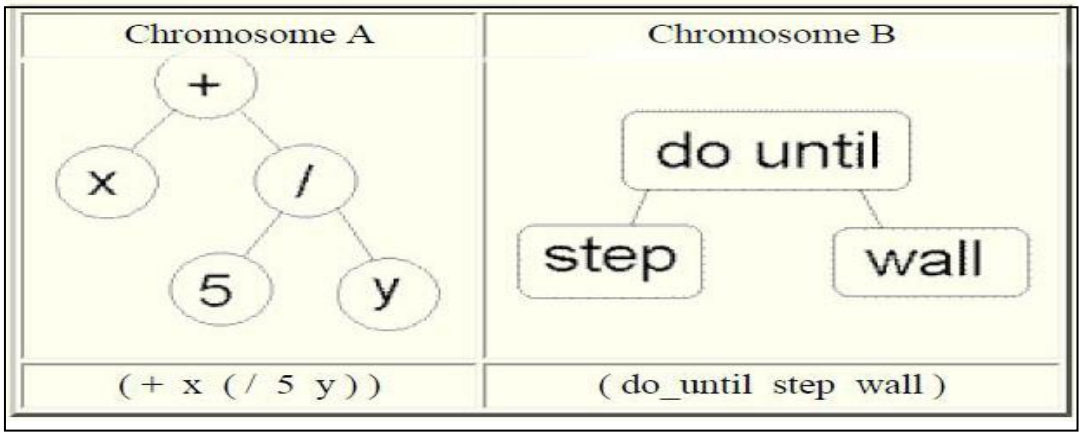

Figure 3.15: Tree Mating Process[74].

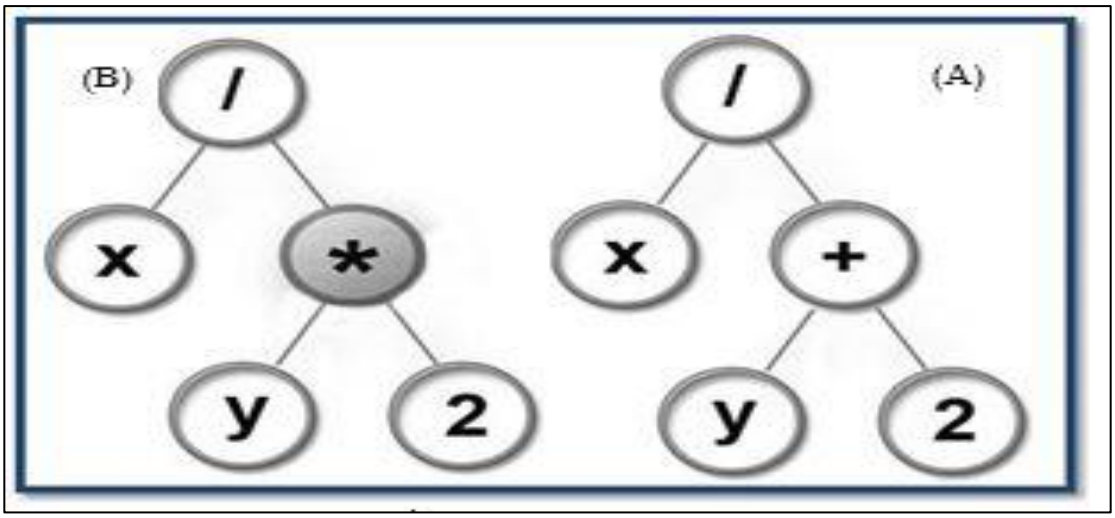

Figure 3.16: Mutation Coefficient or Number[75].

a. Before the mutation occurs $(\frac{x}{y} + 2)$ (3.5)

b. After the mutation occurs $(\frac{x}{y} * 2)$ (3.6)

3.3.8 Difference between Developmental Genetic Algorithms and Traditional Research Methods

Developmental Genetic Algorithms simulate the process of embryonic development in organisms. They start with a simple initial structure and iteratively add more complex structures to it. This approach is inspired by the way organisms grow and develop in nature traditional research methods traditional research methods involve a more direct, human-guided approach. Researchers formulate hypotheses, design experiments, and collect data to test their hypotheses. The focus is on understanding the problem through experimentation and analysis.

There are fundamental differences between evolutionary algorithms and traditional research methods and optimization methods. It summarizes a number of them in table (3.1).

Table 3.1: Differences between Developmental Genetic Algorithms and Traditional Research Method [67].

No	Traditional Research	Developmental GA Algorithms
1	Looking at a single point and sequentially	Searches a generation of points in parallel
2	It requires deriving information and obtaining prior knowledge to determine the direction of the search	It does not require derivation of information or additional knowledge, but only an objective function and equivalent levels of fitness to determine the direction of the research
3	Restricted	Far from being restricted and more probable.
4	Specific in their applications	It is most suitable for application in many matters.
5	You do not have this feature	Allows the user to come up with some potential solutions that support the algorithm.

3.4 CONCLUSION

In this chapter, the genetic algorithm was presented, which includes the method of representing the problem in numerical matrices, and how to implement the algorithm, which facilitates the solution of the problem. Several methods that researchers have used to represent this problem have been addressed and mentioned. The genetic algorithm was modified by using a new hybrid method in crossover with the aim of obtaining better results. In the next chapter, the application of this method will be shown, and the results representing the solution to the TSP will be shown.

4. RESULTS AND DISCUSSIONS

In this chapter, the thesis presents the result of implementing the traditional genetic algorithm and modify genetic algorithm, discusses, and evaluates the results in according when put into outlined in earlier chapter. This section concentrates on using genetic algorithm and its implementation for variable number of cities. Initially, a specific number of iterations is selected while keeping the population size constant, and the outcomes are recorded in tabular format. Subsequently, the experiments are repeated on the same set of cities as in the first experiment, but with an increased number of iterations and population size and using incorporates three hybrid methods (flip, swap, and sliding) to determine the best crossover generation. Tables (4.1) and (4.2) combine the outcomes of these experiments. The table (4.1) represented result of implementing traditional genetic algorithm. And table (4.2) represented result of implementing modify genetic algorithm, which represents the proposed new path. Upon obtaining the results, a comparison is made to assess the algorithm's performance in terms of time efficiency, distance travelled, and the identification of the best path without repetition. The primary objective is to find the optimal route within the shortest possible time, while ensuring that no cities are revisited

4.1 EXPERIMENT SETUP

In this section, our primary objective is to evaluate the performance of the traditional genetic algorithm and proposed genetic algorithm by implementing different configurations for each test case and conducting comparative analysis. The experiments are conducted on varying city sizes, including (10, 50, 100), and (200) cities.

For the first test, involving 10 cities, the genetic algorithm is executed in two cases. The first case utilizes population size (150), (1000) iterations, while the second case employs population size (150), (100000) iterations.

Moving on to the second test, (50) cities are selected, and again, the algorithm is run in two cases. The first case involves (100000) iterations, population size (150).

In the third test (100) cities are chosen, and two cases are considered. The first case employs (1000) iterations, population size of (150), while the second case utilizes (100000) iterations population size of (150).

The fourth test involves (200) cities, and two cases are conducted. The first case utilizes (1000) iterations, population size (150), while the second case employs (100000) iterations, population size (150).

Upon obtaining the results, the experiments are repeated on the same sets of cities (10, 50, 100, 200). However, different recurrence values (1000, 100000, 50000) and population sizes (50, 100, 100, 50) are chosen. The results are recorded in a table and compared with the previous outcomes to identify the best-performing cases.

The purpose of this analysis is to evaluate how different parameter values affect the algorithm's performance and determine the optimal configurations for solving the problem.

We used (MATLAB) to create a code that shows how a genetic algorithm works. The(MATLAB) program is one of the powerful programs used to solve mathematical and scientific problems in general, as it is characterized by having a programming language and a complete development environment as it provides a programming language that allows writing new scenarios for more details in a more customized manner. It also supports mathematical and mathematical operations, as it has an extensive library of mathematical and arithmetic functions which includes linear algebra, in addition to increasing mathematical transformations. It also provides the possibility of plotting data. This helps in understanding the data and visualizing the results better and provides the solution of mathematical equations and systems as it is used to solve a variety of mathematical equations and systems, whether linear or nonlinear, and even equations of virtue and integration. It has the ability to process images and process time as it provides a set of tools for image processing and signals, such as filtering images, studying information from images, and analysing time and frequency. It has the ability to integrate with other tools and other languages such as (C++), (Python), and others, which leads to the integration of integration advantages in solving problems.

The program randomly distributes cities into a two-dimensional (X, Y) square grid. The graph below shows the result of addressing the problem by implementing the smart path solution of the (MATLAB) code when (10) cities are randomly selected for which we chose the number of iterations (100000). With a population size (150).

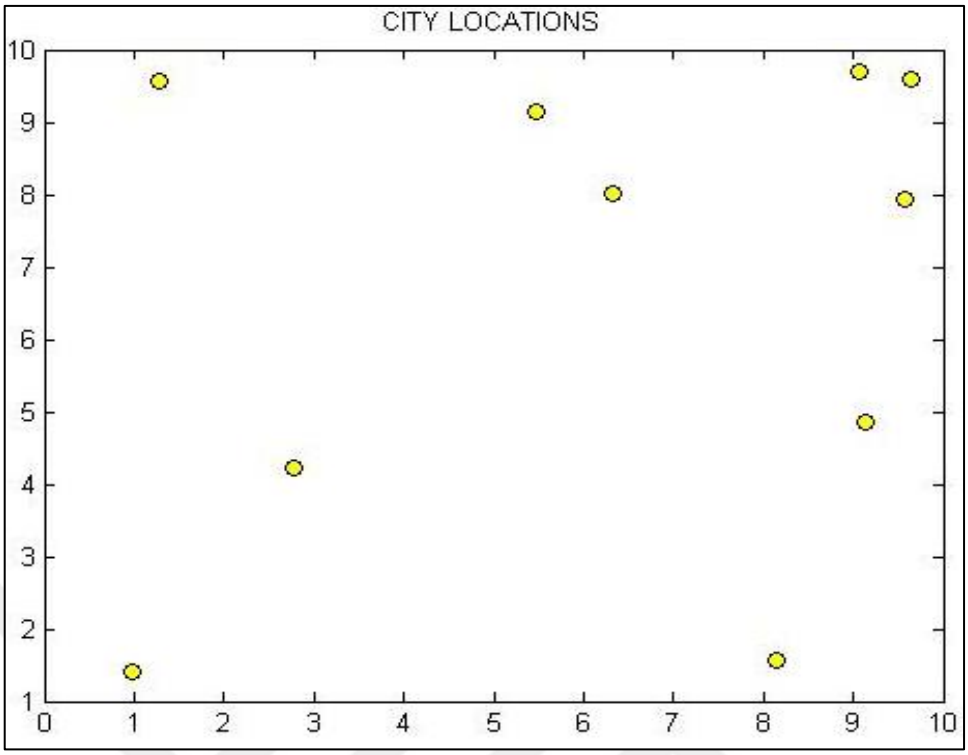

Figure 4.1: Cities Locations for (10) Cities.

Optimal solution

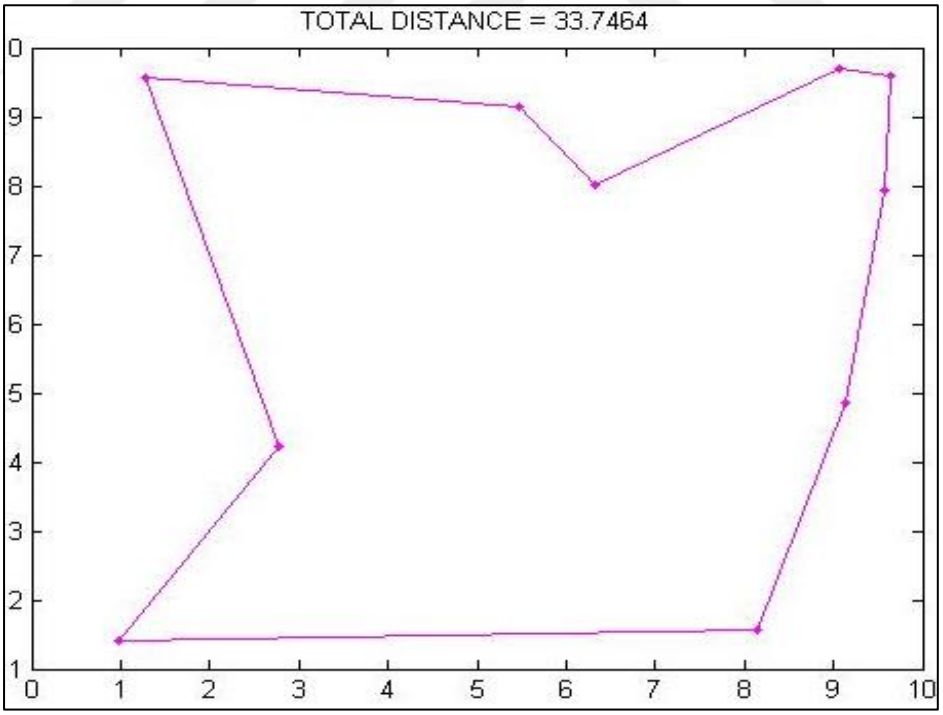

Figure 4.2: Optimal Solution for (10) Cities.

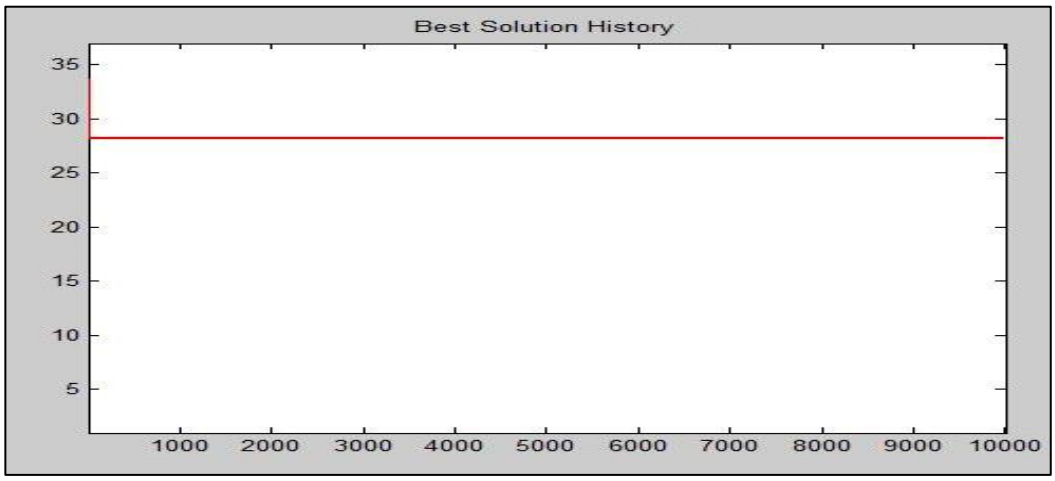

Figure 4.3: Best Solution History for (10) Cities.

We note from the previous models in the implementation of the program that (10) cities were chosen randomly, and the best path was found among these cities. Where we chose iterations (100000). With size population (150), program needed (82) seconds to find the best path. For total distance (33.7464), figure (4.3) determines that the best solution history was when it was less than (100) iterations. Therefore, in the following experiments, we need to reduce the repetition to less than the extent at which the improvement of the solution stops, and the purpose is to obtain less time to implement the program, which leads to an improvement in the results.

We repeat the same experiment (10) cities, with the number of iterations (1000) with a population size of (150).

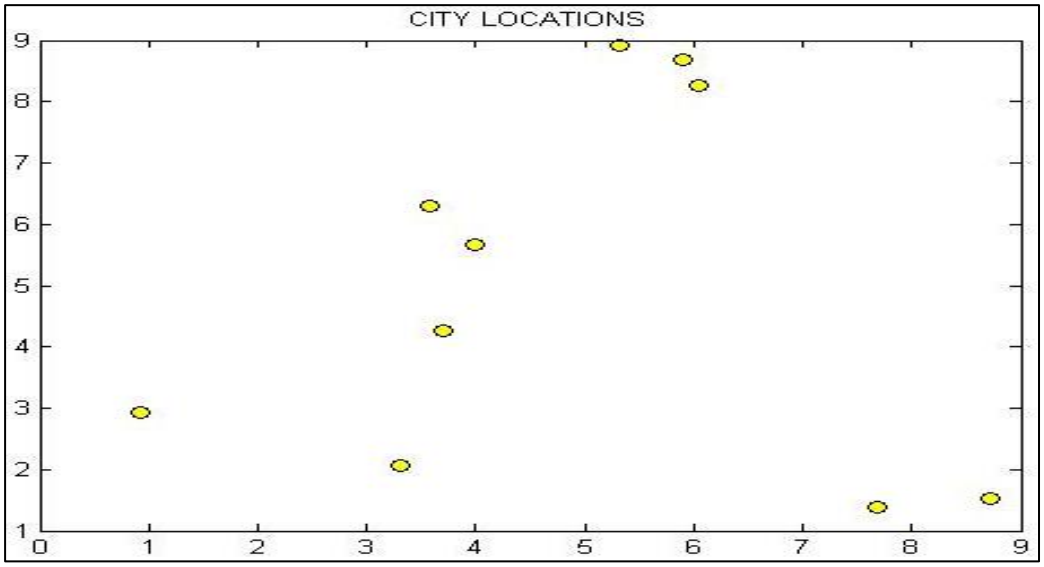

Figure 4.4: Cities Locations for (10) Cities.

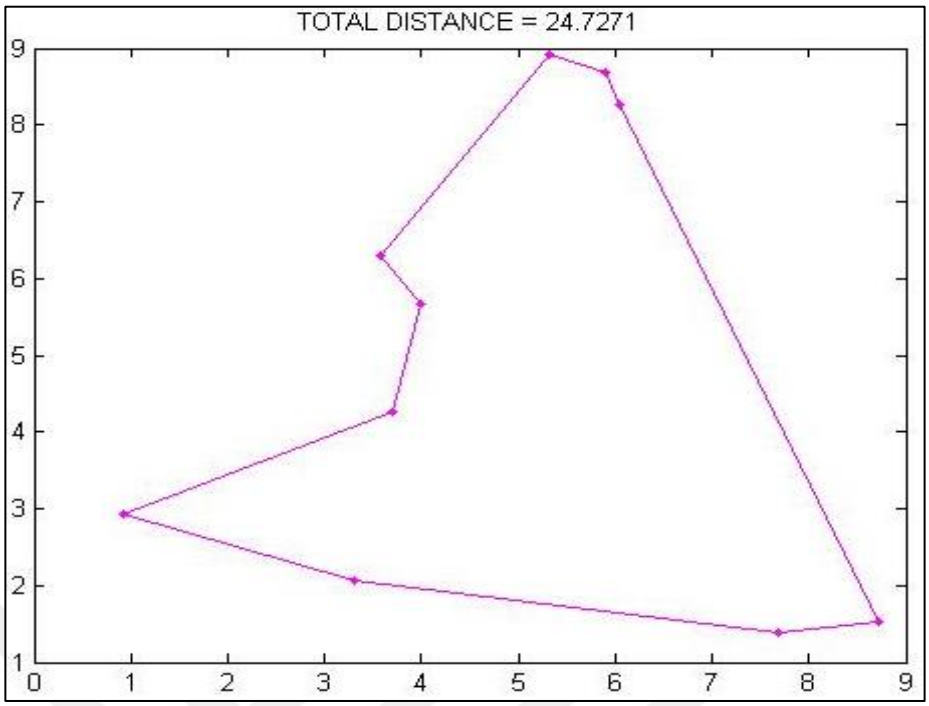

Figure 4.5: Total Distance for (10) Cities using (1000) Iterations.

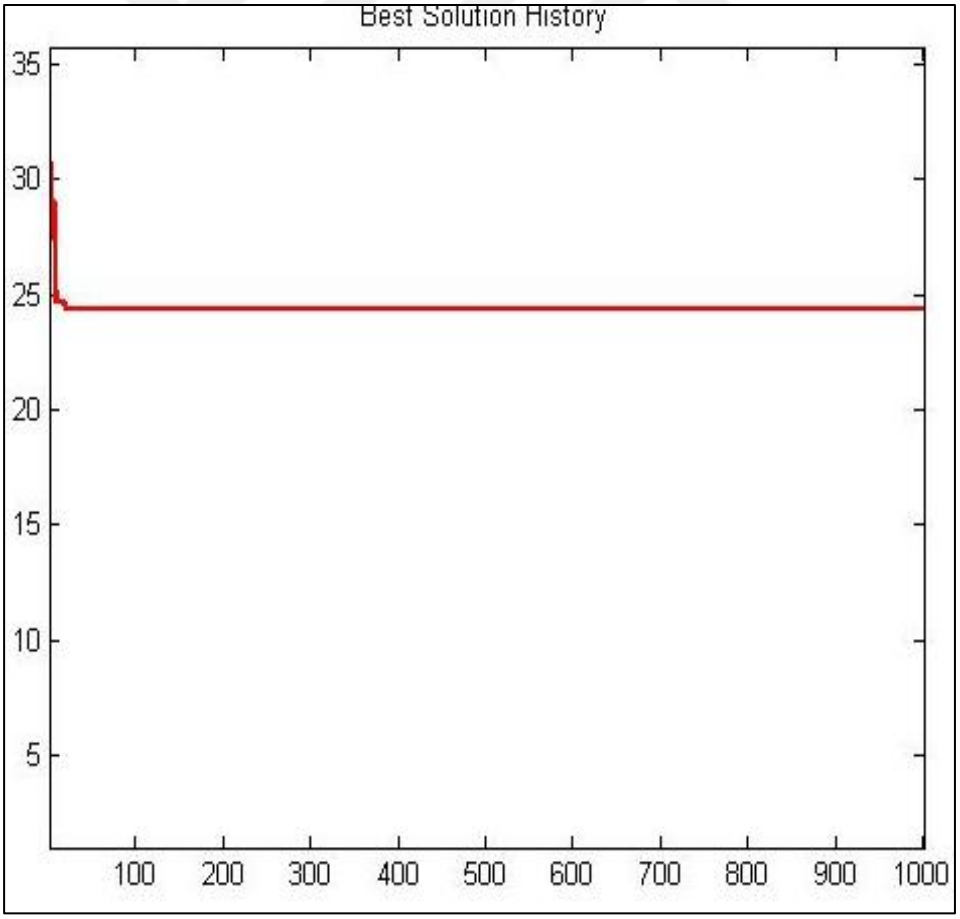

Figure 4.6: The Best Solution History for (10) Cities with (1000) Iterations.

We note from previous experiments when using frequency (1000) and population size (150) for the number of cities (10), we find that the time taken to find the best path is (2.17) seconds for a total distance of (24.7271).

Where it is clear from this that the time is ideal to obtain the result, but here we must note that the number of cities that the traveling salesman must visit is very few and for this we see that the improvement of the solution according to the illustrative scheme stopped early and therefore because the number of iterations did not end and that the prang will continue in the iterations despite that the number of cities is small, so the number of iterations must be proportional to the number of cities in order to reduce the time of implementing the program.

In the following graphs, we present the best record of the solution when we choose (50) cities in a random location, with iterations (100000) and population size (150)

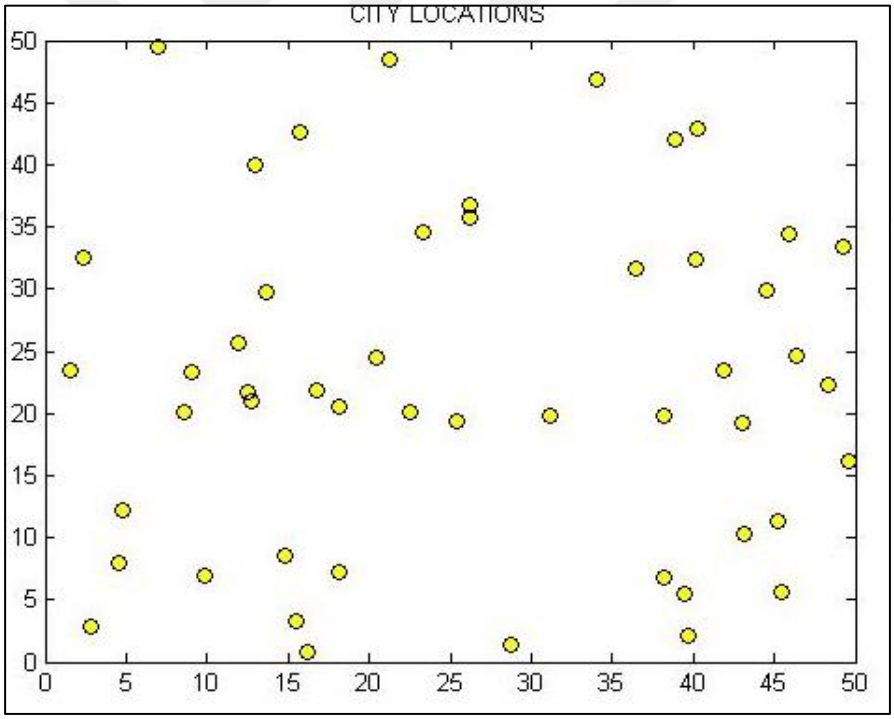

Figure 4.7: Cities Locations for (50) Cities.

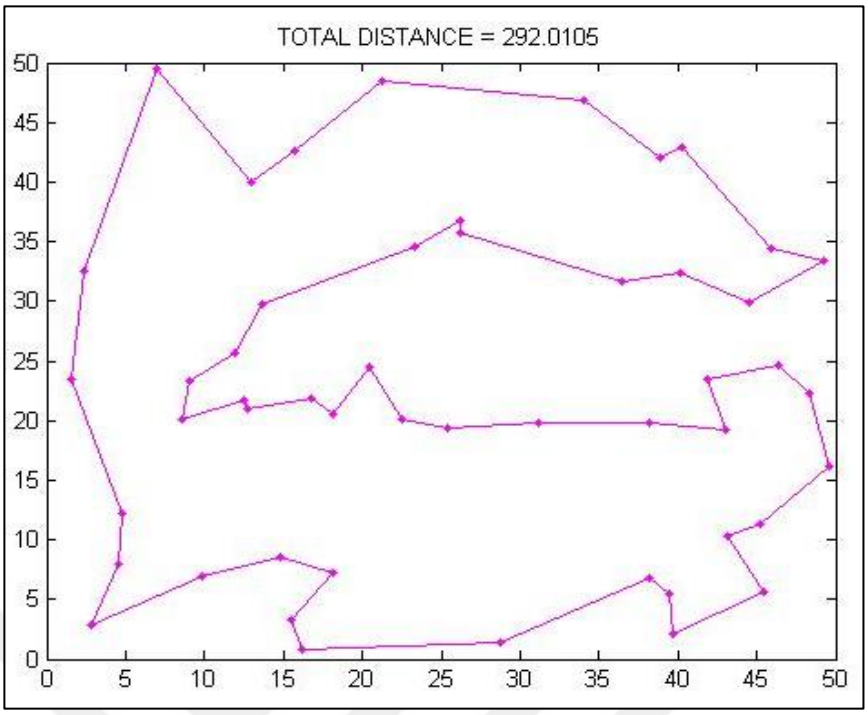

Figure 4.8: Optimal Solution for (50) Cities.

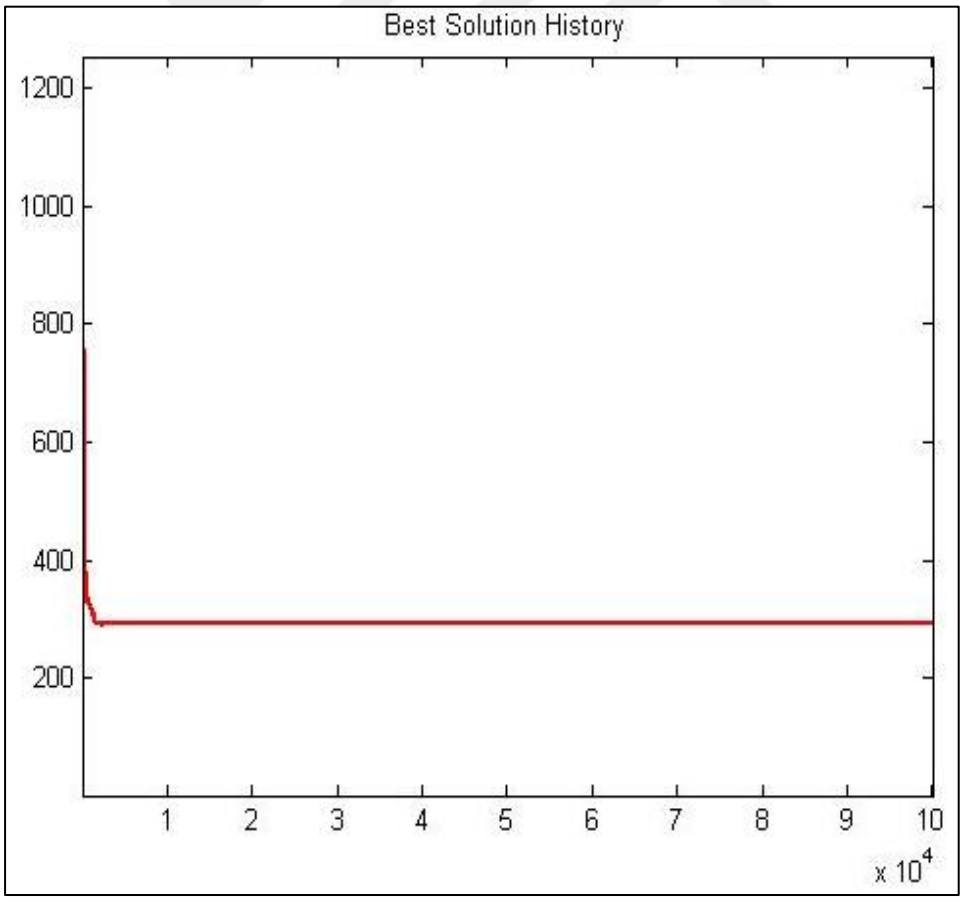

Figure 4.9: Best Solution History for (50) Cities, (100000) Iterations, (150) Population Size.

We note from previous experiments when using iterations (100000) and population size (150) for the number of cities (50), we find that the time taken to find the best path is (111.41) seconds for a total distance of (292.0105).

When looking at the detailed plan, it becomes clear to us that there is no problem, as the traveling salesman chose the ideal route, as there was no repetition on the road, and this leads to reducing the time and effort in the trip, but the time it takes to get to this road is relatively long, so we must change the size of the population or the number of iterations in order to obtain new values lead to better accurate results.

I will redo the same experiment (50 cities), but with a different number of iterations (1000). with population size (150) to make a distinction in the finest solution history.

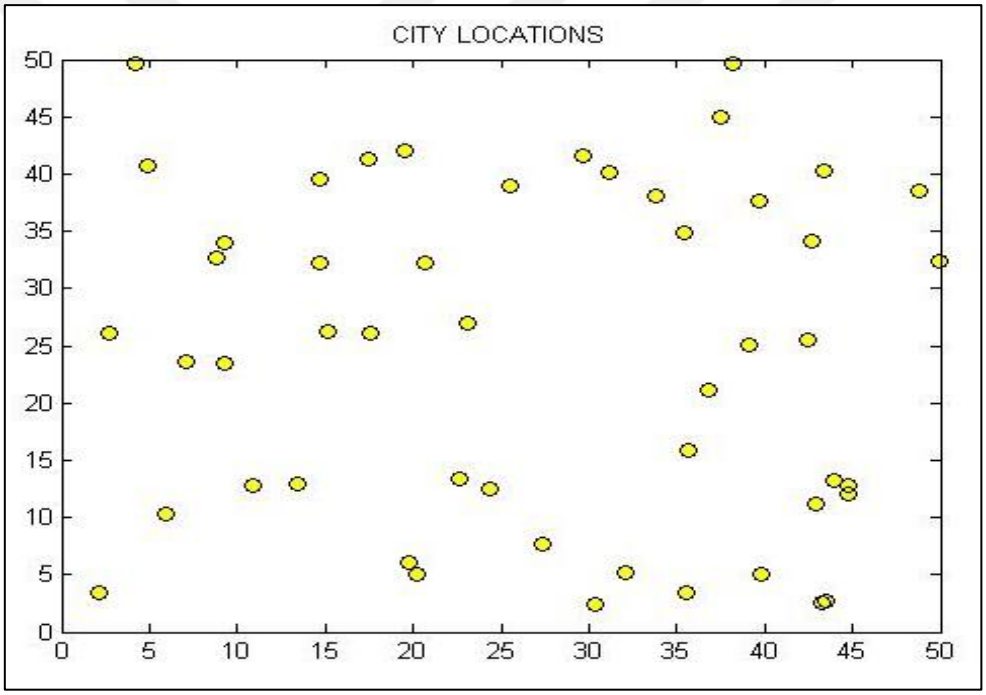

Figure 4.10: Cities Locations for (50) Cities.

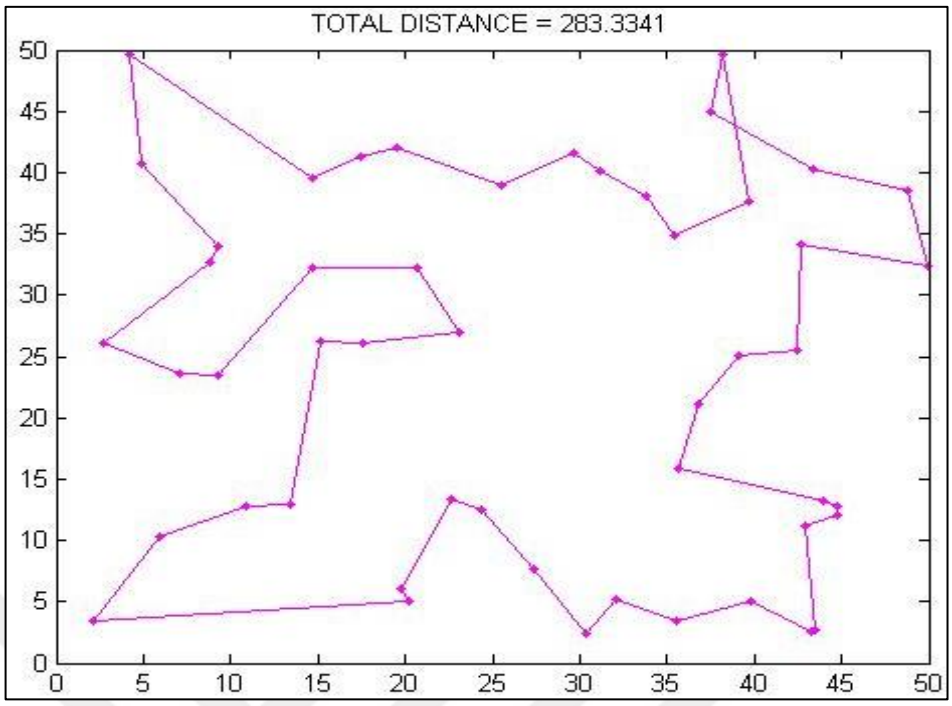

Figure 4.11: Optimal Solution for (50) Cities.

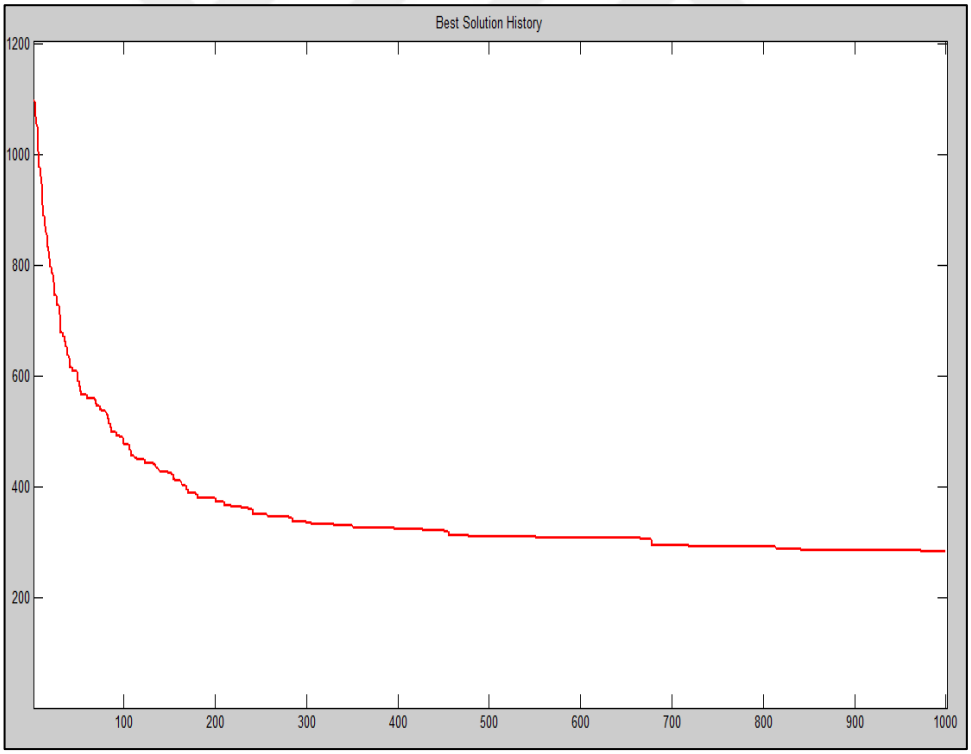

Figure 4.12: The Best Solution History for (50) Cities using (1000) Iterations.

We note from previous experiments when using iterations (1000) and population size (150) for the number of cities (50), we find that the time taken to find the best path is (04.47) seconds for a total distance (283.3341) and the fitness value does not change after iterations (750) therefor we will degree number of iterations in order to get good path.

It is noticeable from the figure (4.12) that there is a problem in choosing the right route for the traveling salesman, as he did not choose the ideal route for the trip, and this was marked with a red circle, and this leads to an increase in the program execution time and an increase in the path travelled, so this problem must be solved by increasing the frequency or increasing population size, which will be conducted in the following experiments in order to obtain better results and solve this problem.

The presented graph shows the observed progression of the best solutions obtained when selecting (100) cities in random locations, with no duplicates (100000) and a population size of (150). It shows the improvement in solution quality over time.

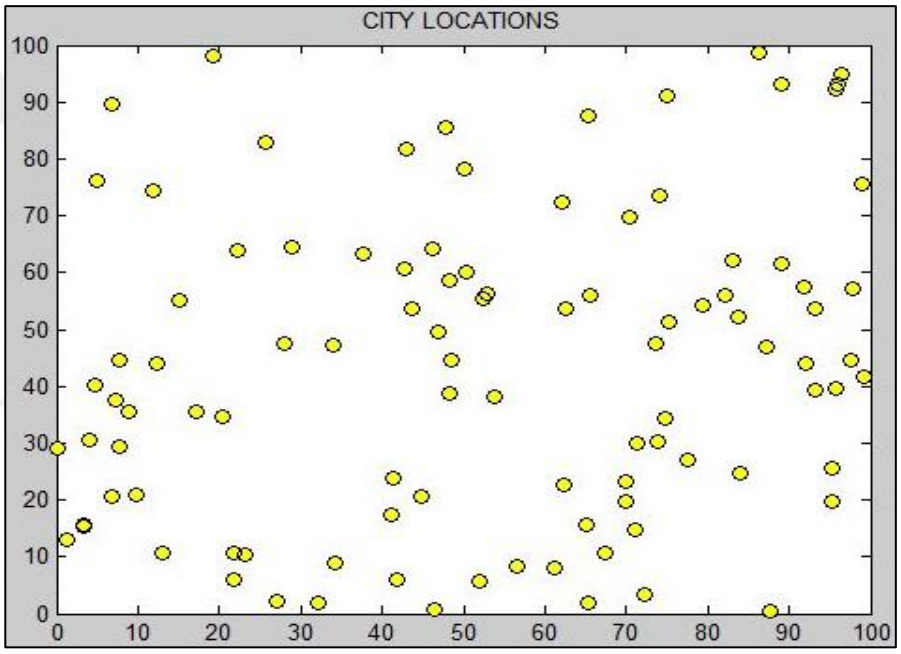

Figure 4.13: Cities Locations for (100) Cities.

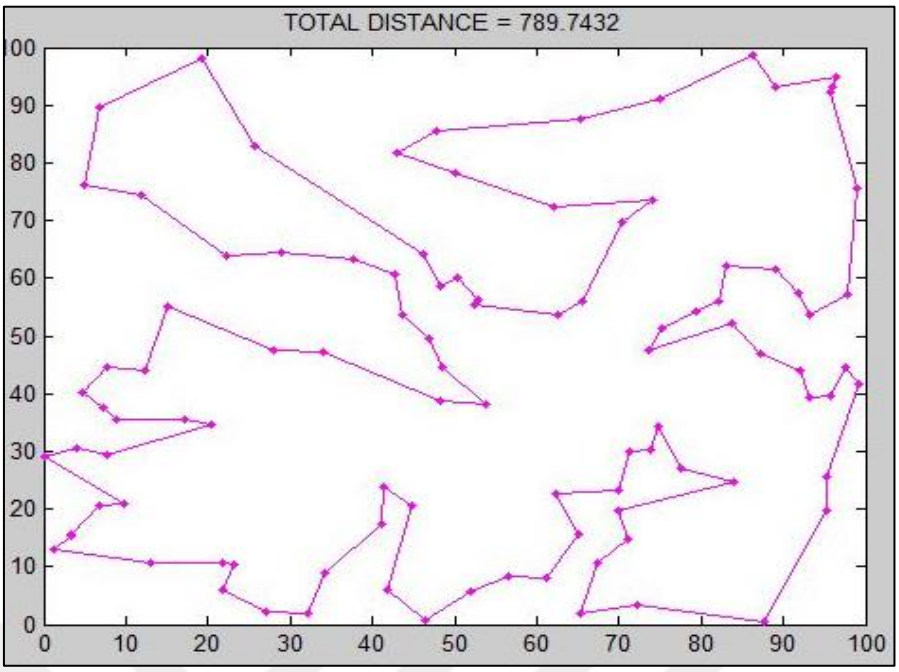

Figure 4.14: Total Distance for (100) Cities.

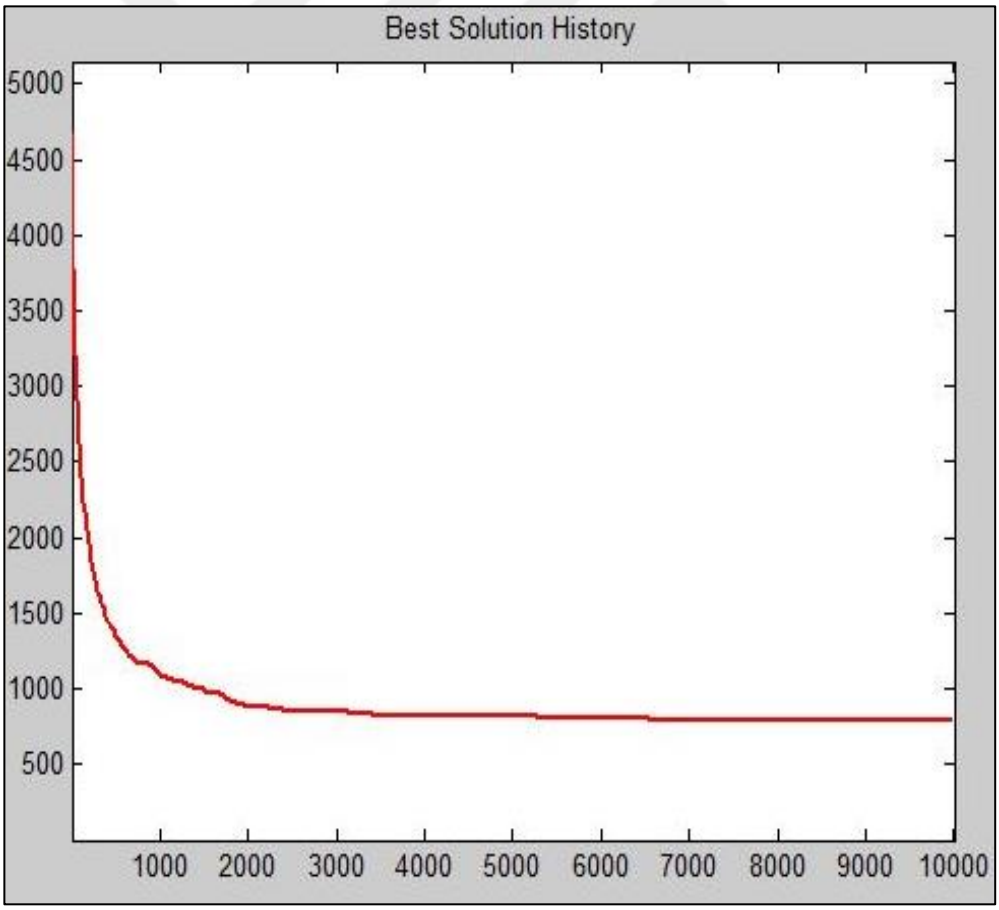

Figure 4.15: The Best Solution History for (100) Cities using (10000) Iterations.

The previous figure (4.15) that explained the result of the case study which is contain from (100) cities, we use iterations (10000) and population size (150) apply on of (100) cities we find that the time taken to find the best path is (22.65) seconds for a total distance (789.7432). And figure (4.15) display when the iterations was (6500).

There was no improvement in the level of the solution. The time taken by the program to find the solution is relatively long, but the solution was without repetition. The travel sales representative reached all cities and without repeating the travel trip. The only problem with this solution is the time it takes to find the solution is relatively long for this we must reduce the number of iterations in order to reduce the execution time of the program.

In the following graph, we present the best solution history when we choices a (100) cities at a random position with iterations (100000) and population size (150).

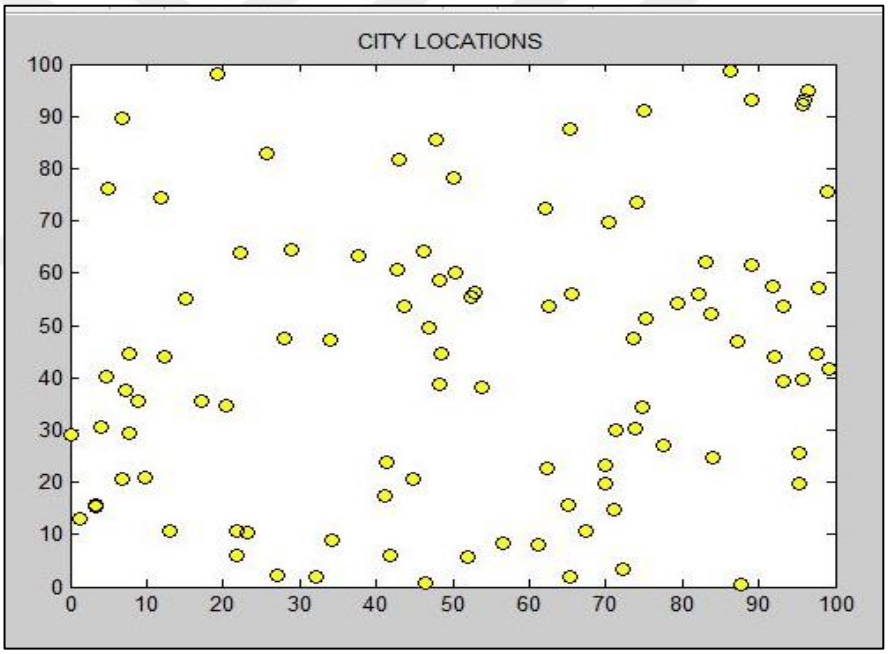

Figure 4.16: Cities Locations for (100) Cities.

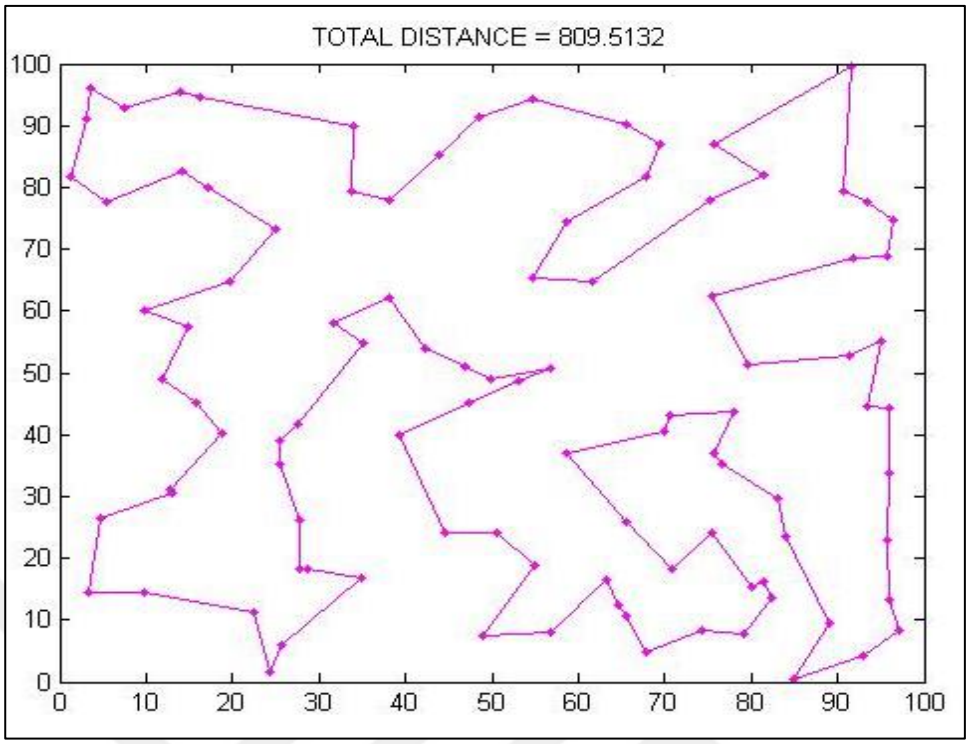

Figure 4.17: Total Distance (100) Cities and the Optimal Path without Repeating.

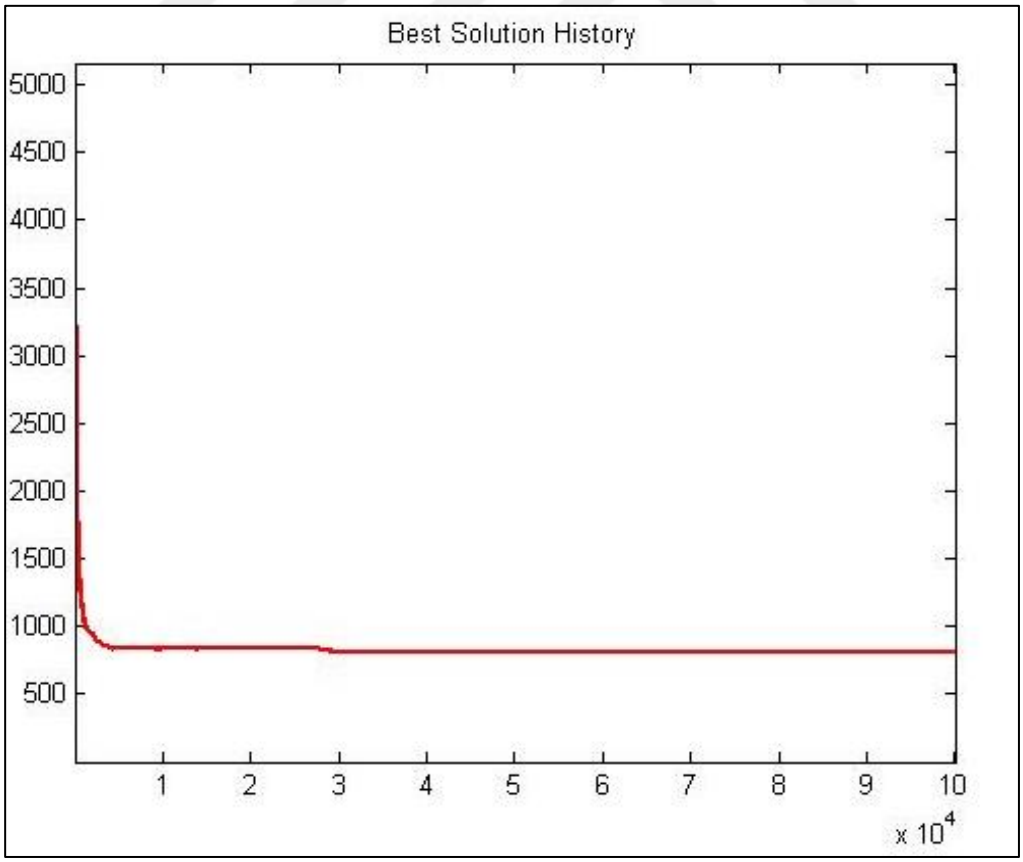

Figure 4.18: Best Solution History for (100) Cities, (100000) Iterations, (150) Population Size.

We note from previous experiments when using iterations (100000) and population size (150) for the number of cities (100), we find that the time taken to find the best path is (120.31) seconds for a total distance (809.5132).

The path is optimal without repeating. and the time is good. Figure (4.18) shows that the solution was good, as the traveling sales representative took a short time to reach all regions without visiting more than one city twice, and this indicates a clear improvement in the solutions, taking into account the number of iterations and the size of the population, we reach good solutions.

In the following graph, we present the best solution history when we choices (200) cities at a random position with iterations (100000) and population size (150).

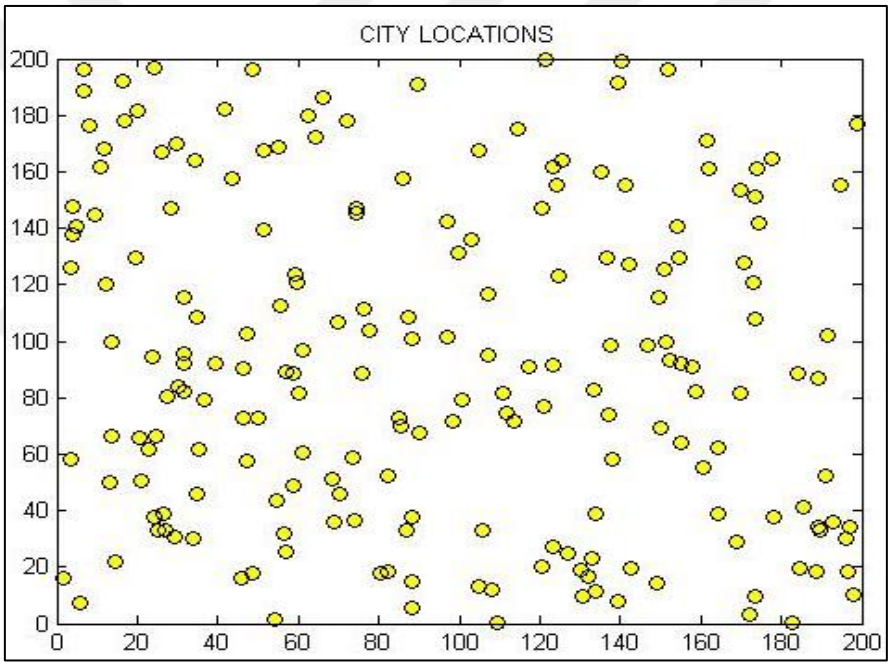

Figure 4.19: Cities Locations for (200) Cities.

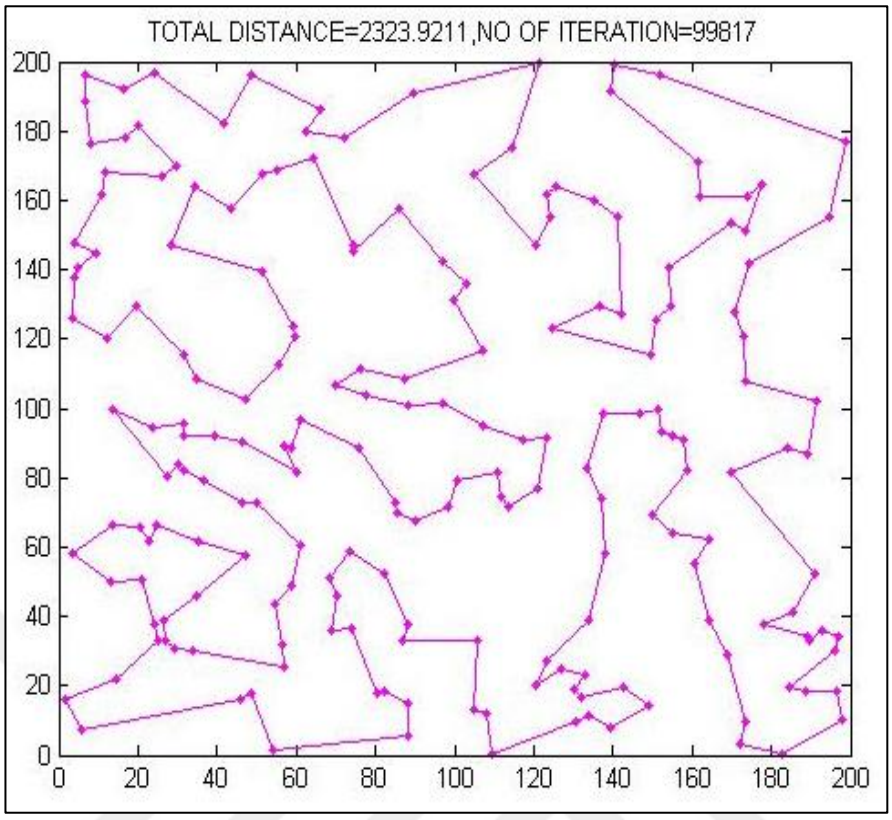

Figure 4.20: Total Distance for (200) Cities.

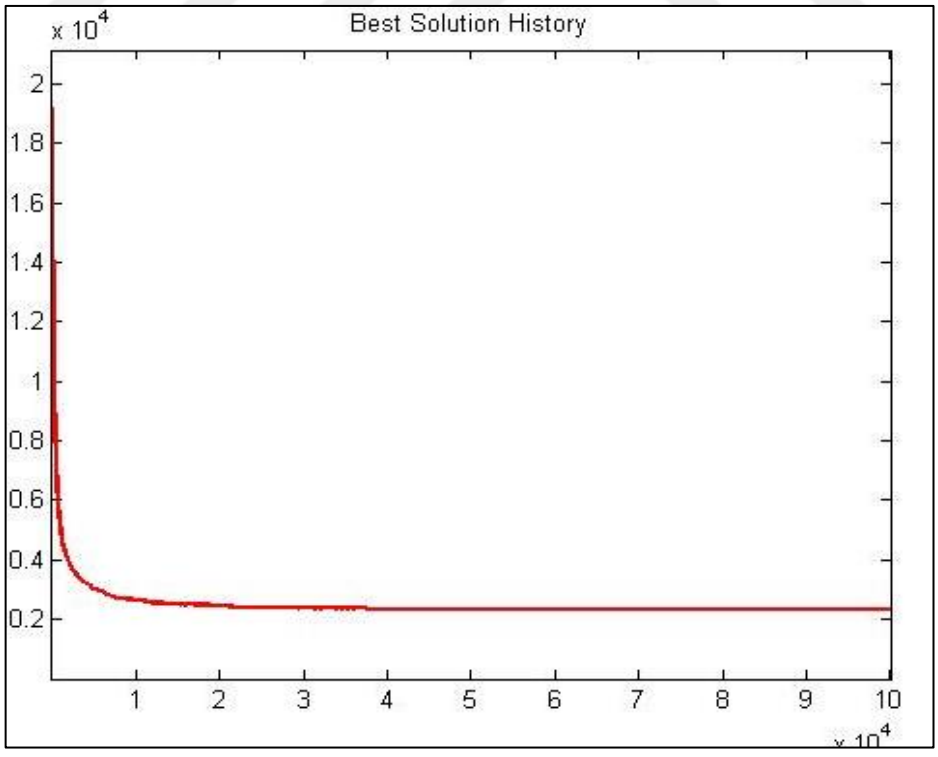

Figure 4.21: The Best Solution History for (200) Cities using (100000) Iterations.

In this example, the efficiency of the traditional genetic algorithm is shown, because the number of cities in this experiment is large, and this represents a challenge for this algorithm in finding the best route for the traveling salesman without repetition. Indeed, the traditional genetic algorithm was impure in finding the best route for the traveling salesman, but it really failed in the matter of time. The search for the best path is (240.41) seconds for a total distance of (2323.9211) is the best path for seals to travel, but the time is very long and for this, changes will be made to the variables in the genetic algorithm in the next experiments to solve the time problem.

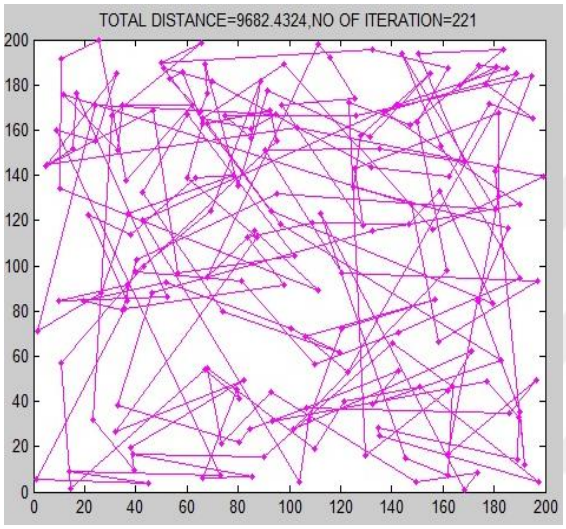

Figure 4.22: The Number of Iterations (221).

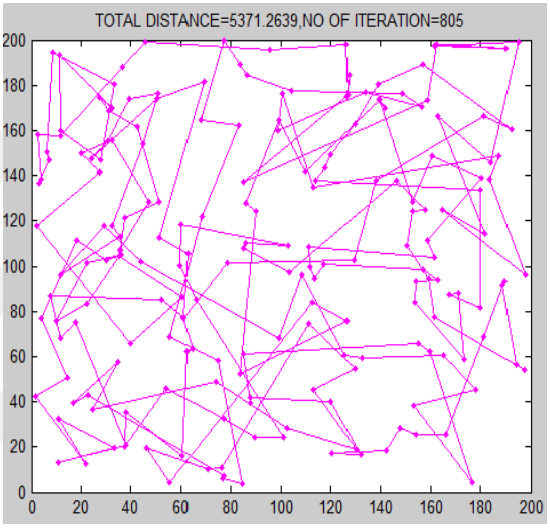

Figure 4.23: The Number of Iteration (805).

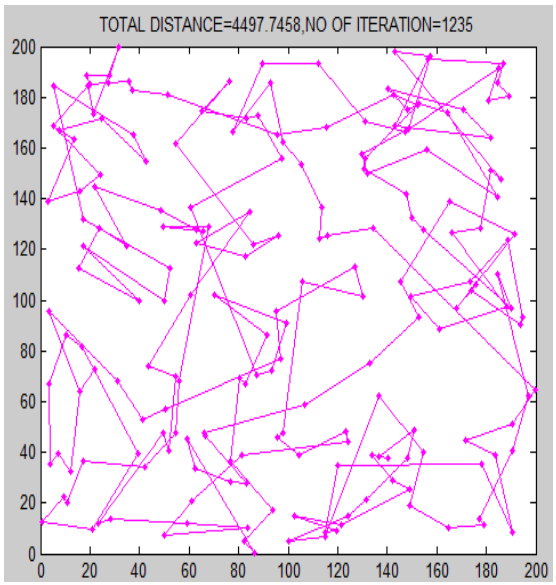

Figure 4.24: The Number of Iteration (1235).

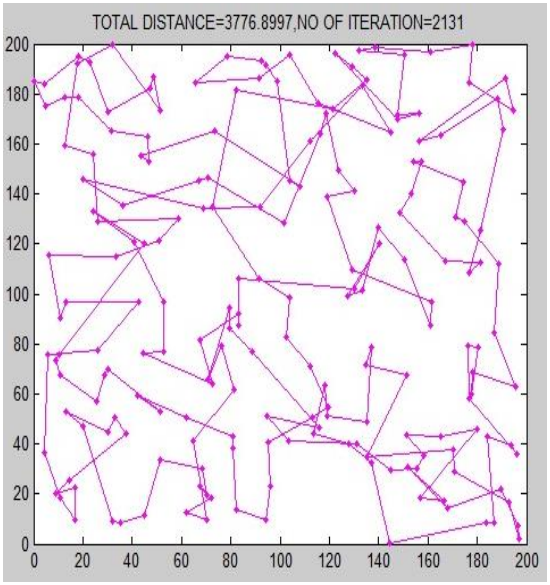

Figure 4.25: The Number of Iteration (2131).

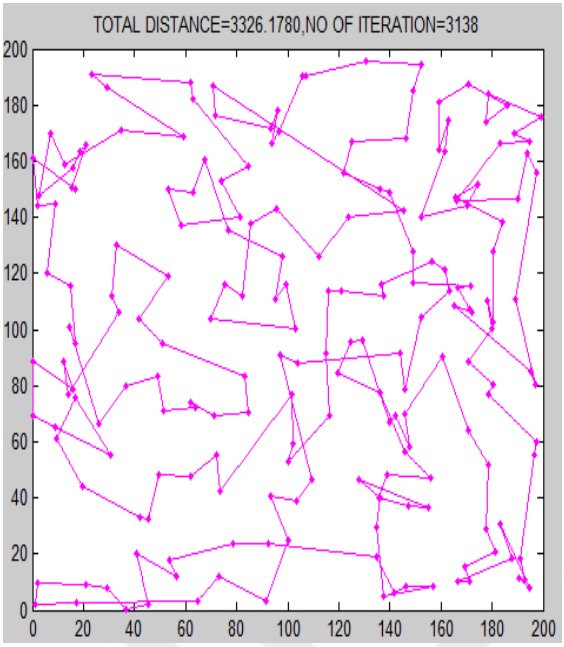

Figure 4.26: The Number of Iteration (3138).

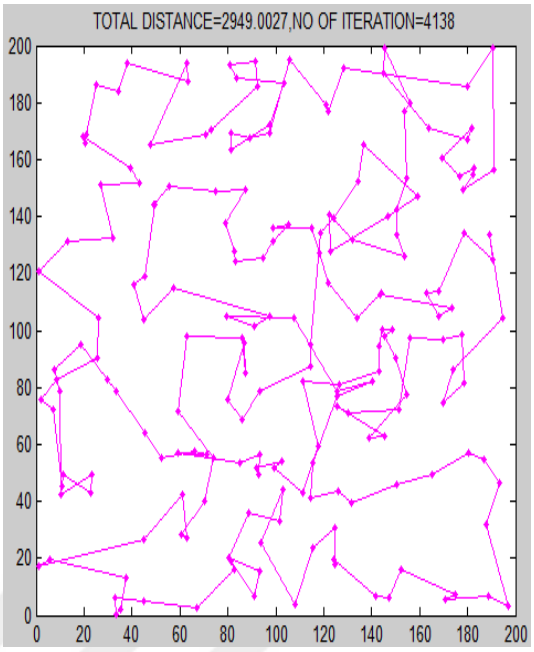

Figure 4.27: The Number of Iteration (4138).

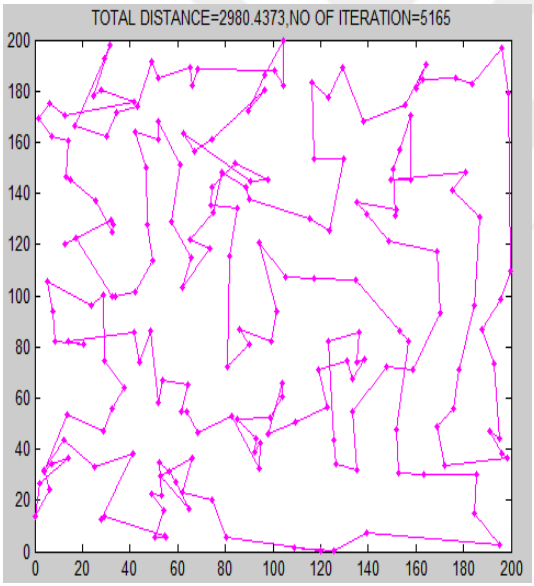

Figure 4.28: The Number of Iteration (5165).

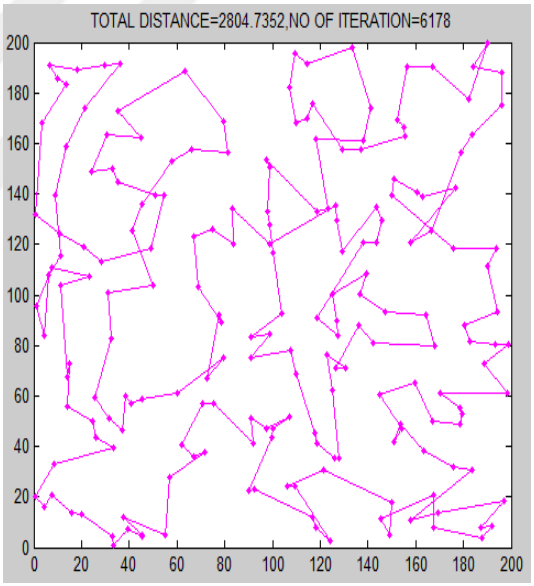

Figure 4.29: The Number of Iteration (6178).

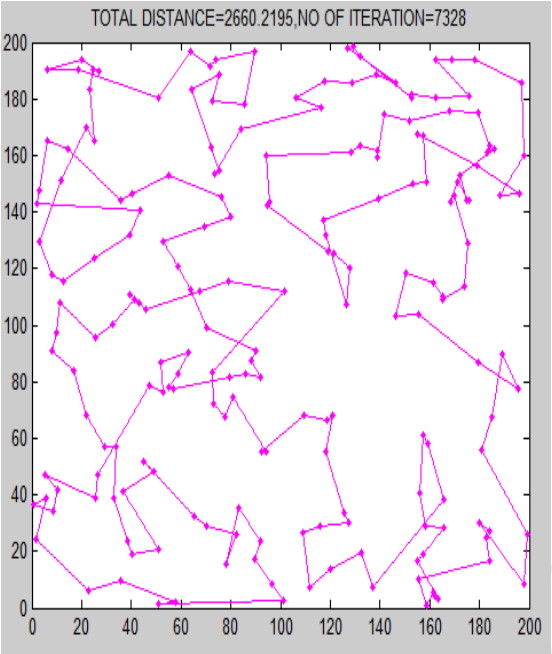

Figure 4.30: The Number of Iteration (7328).

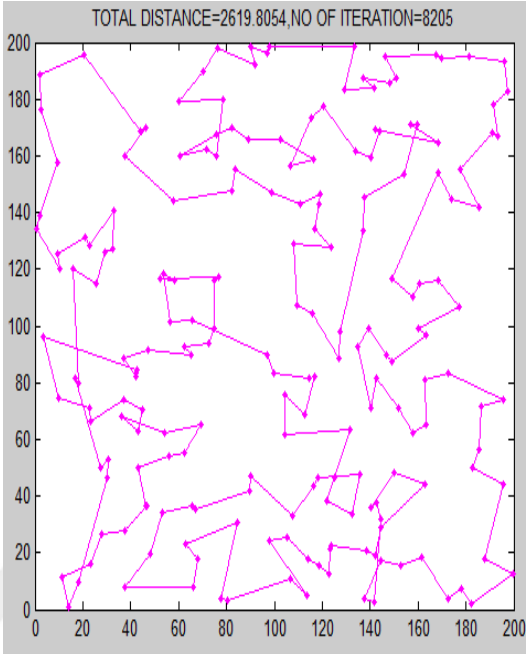

Figure 4.31: The Number of Iteration (8205).

One of the main steps in the algorithm is the result of iteration or generation. The goal of this step is to create a new generation of possible solutions by mutating and simplifying so that solutions can be improved over time. In this phase, members of a new generation of solutions are generated using the experimental tool.

This can be done using intersect (reduction), pivot (random change) operations, and other methods. The task of this species is to diversify phylogenetically into new generations. These steps take iterations in the core algorithm. After starting to set up these steps, it is expected that the quality of the solutions developed will get better and closer to the optimal solution for the given problem.

Through the previous figures, we notice how the genetic algorithm started with computational operations, including intersection, mutation, and iterations, finding the best value for fitness function for choosing is the proper route travelling salesman.

In the following graph, we present the best solution history when we choices (200) cities at a random position with iteration (10000) and population size (150).

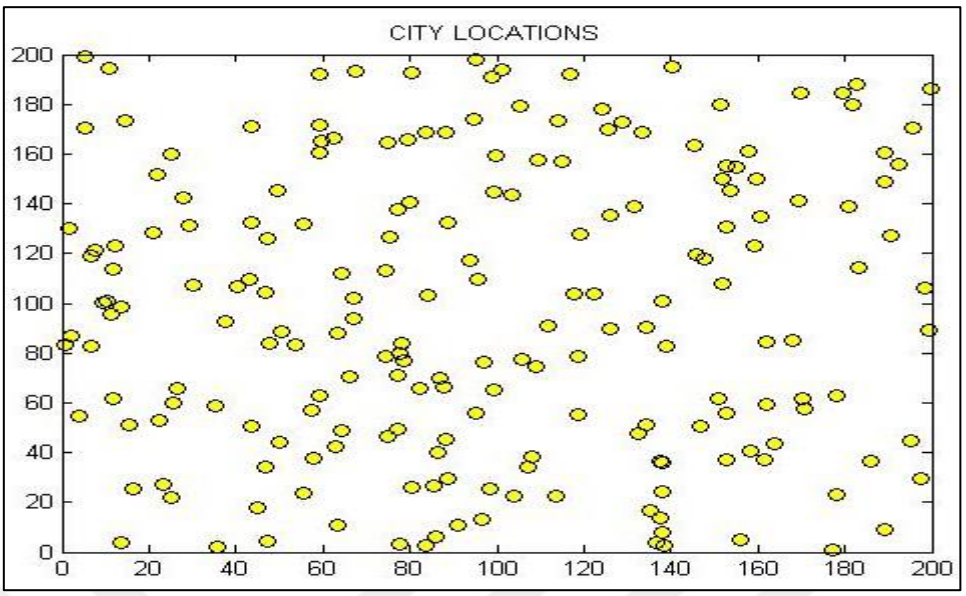

Figure 4.32: Cities Locations for (200) Cities Created Randomly.

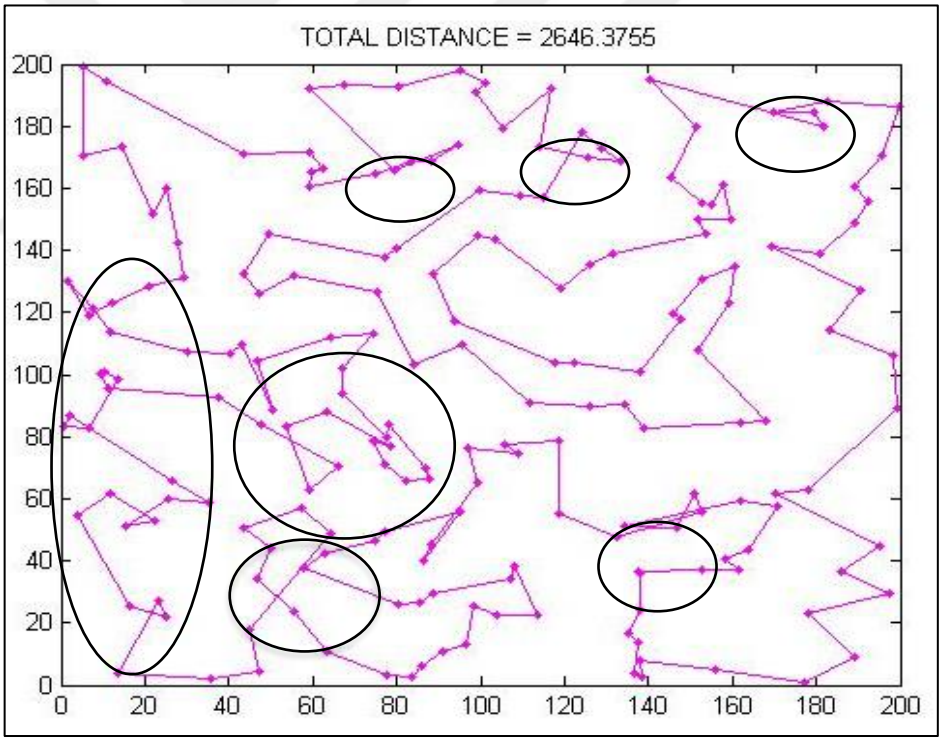

Figure 4.33: Total Distance (200) Cities.

We note from figure (4.33) it is not optimal solution path that there is some incorrect path as marked in circles in figure (4.33) we will solve these incorrect path in the following experiments.

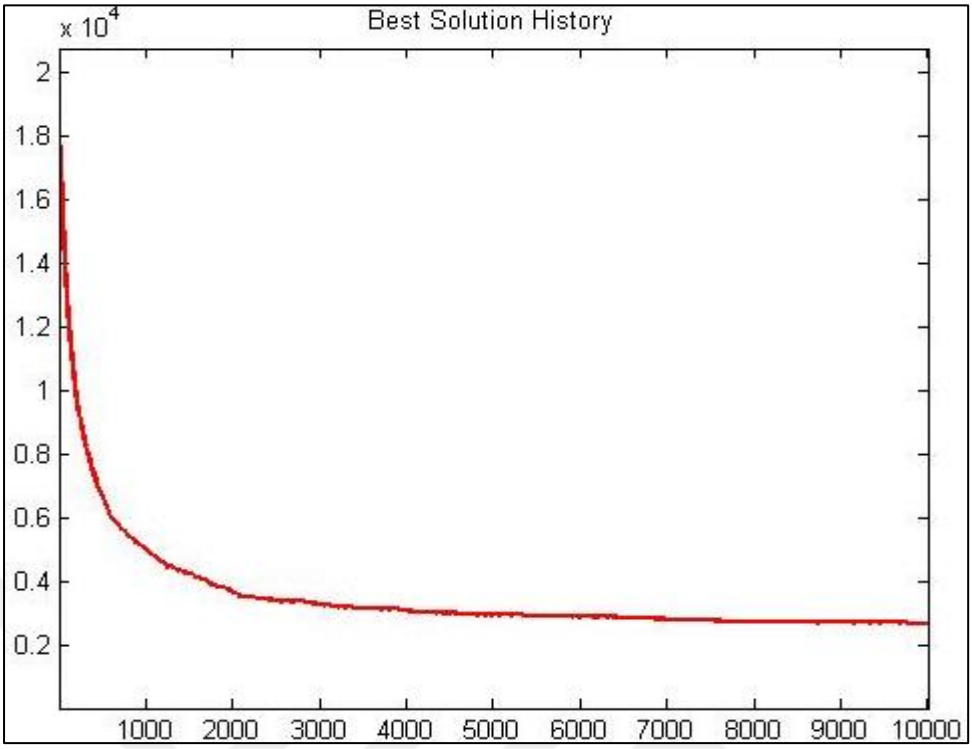

Figure 4.34: The Best Solution History for (200) Cities using (10000) Iterations.

In this last example for the traditional genetic algorithm, an attempt was made to reduce the number of iterations to reduce the time it takes to obtain the solution, and we conducted the experiment, where the best historical solutions were for (200) cities using (10000) iterations. The solutions of the traditional genetic algorithm showed wrong and inaccurate results for the traveling salesman's path, where the path contains many repetitions as indicated in the circles in the figure (4.34).and we note the time to find the best path in (35.79) seconds but there is not the best time.

4.2 ANALYSIS OF RESULTS

When applying the (MATLAB) code, implementing the traditional genetic algorithm and choosing a different number of cities and iterations as well as the generation size, different results were obtained to reach the solution, as shown in the following table.

Table 4.1: Algorithms Results Compared by Distance and Time Using Traditional Genetic Algorithm.

Number of City	Number of Iterations	Population Size	Total Distance	The it Takes to Find the Best Path
10	100000	150	29.8794	82.00 seconds
10	1000	150	24.3419	02.17 seconds
50	100000	150	292.0105	111.41 seconds
50	1000	150	238.3341	04.37 seconds
100	10000	150	789.74320	22.56 seconds
100	100000	150	809.5132	151.10 seconds
200	100000	150	12323.921	240.41 second
200	10000	150	2486.3755	35.79 seconds

Based on the previous experiments conducted using the traditional genetic algorithm to address the traveling salesman problem, different solutions were tried for different city groups, including (10, 50, 100) and (200) cities, where the experiments were conducted depending on the time factor (execution time program), as well as the distance travelled and the number of cities, and then set the selection of a different number of repetitions in each of the experiments for the purpose of comparing the results.

In the beginning, a specific number of (1000) iterations was chosen and the experiment was conducted. In the second experiment, (100,000) iterations were used.

The results show that for fewer cities (50), the time required for the best route to follow was satisfactory. In major cities such as (150) and (200), a problem arose as the proposed route was not ideal and the time taken to implement the program was relatively long.

Further analysis revealed that when the number of iterations was increased to (100000) for cities of varying sizes (10, 50, 100, 200), the program took longer to reach the optimal solution. The accuracy of results was more pronounced for larger cities compared to smaller ones. In the case of small cities (10 to 50) cities, the time taken to achieve the optimal solution ranged from (60 to 120) seconds, which was relatively high considering the number of cities involved.

Based on these experiments, it can be concluded that when dealing with a large number of cities (100, 200) a higher number of iterations is necessary to obtain more precise results without revisiting cities. However, this approach poses the challenge of extended computation period. As opposed to that, for smaller cities, smaller number iterations can be used to obtain optimal results in a shorter timeframe.

After obtaining these results, modifications will be made to the genetic algorithm, mmodifications will be to populations and iterations and crossover method in upcoming experiments. And compare the new results with the obtained results

I will to carry out a similar experiment with a few changes to solve the issue of the extended time needed to find the best route in larger cities with (200) cities. I'll reduce the population size to (50) people and set a cap on iterations at (10000) with use new crossover. This change is intended to observe the influence on the overall outcomes, especially in terms of the amount of time required and the choice of the optimal path.

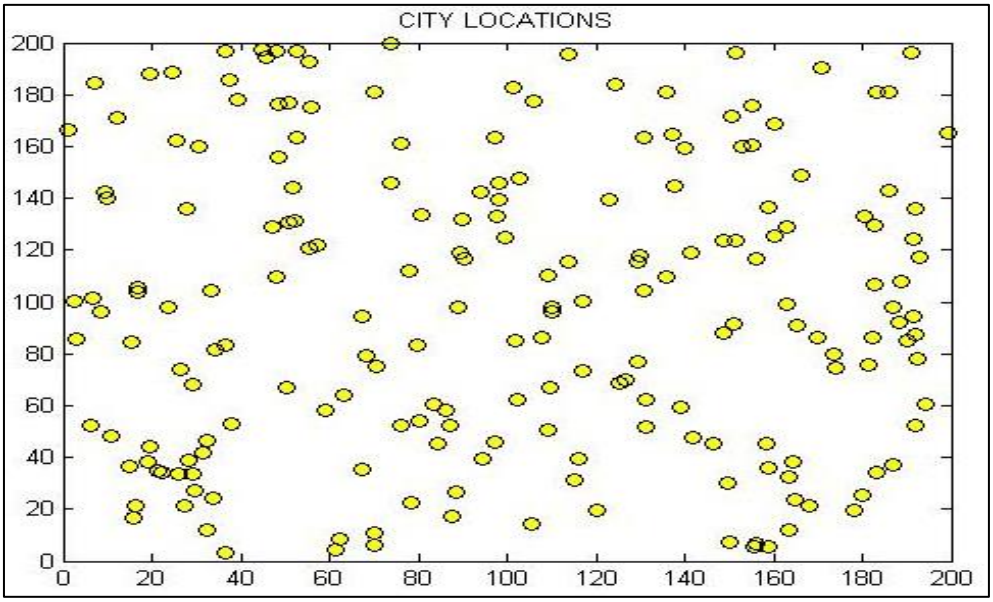

Figure 4.35: Cities Locations for (200) Cities.

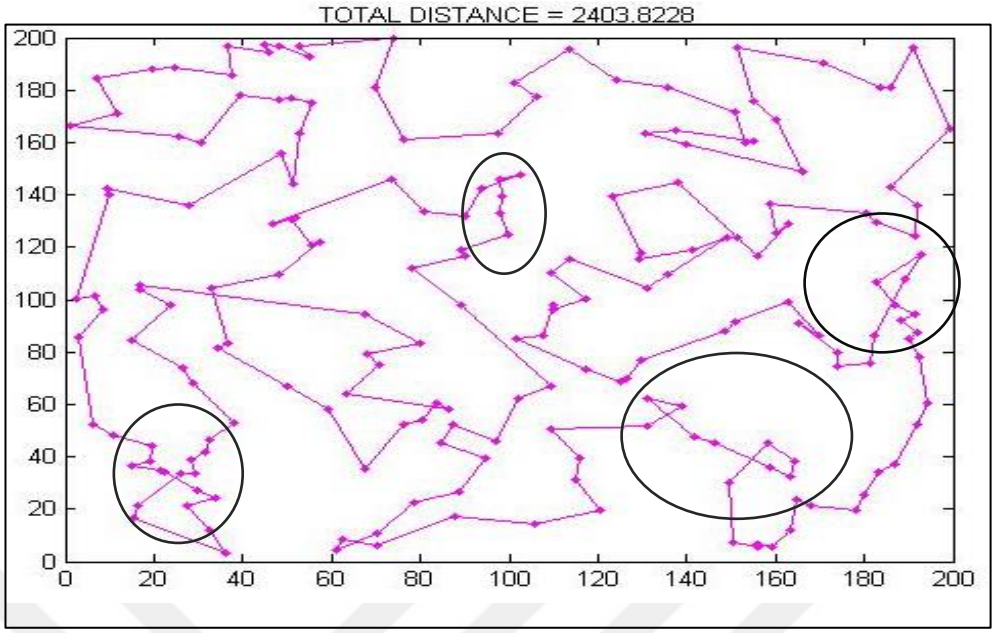

Figure 4.36: Total Distance (200) Cities using (1000) and (50) Population Size.

We note from the figure (4.36) it is not the optimal path because the number iteration is not suitable for a large number of cities.

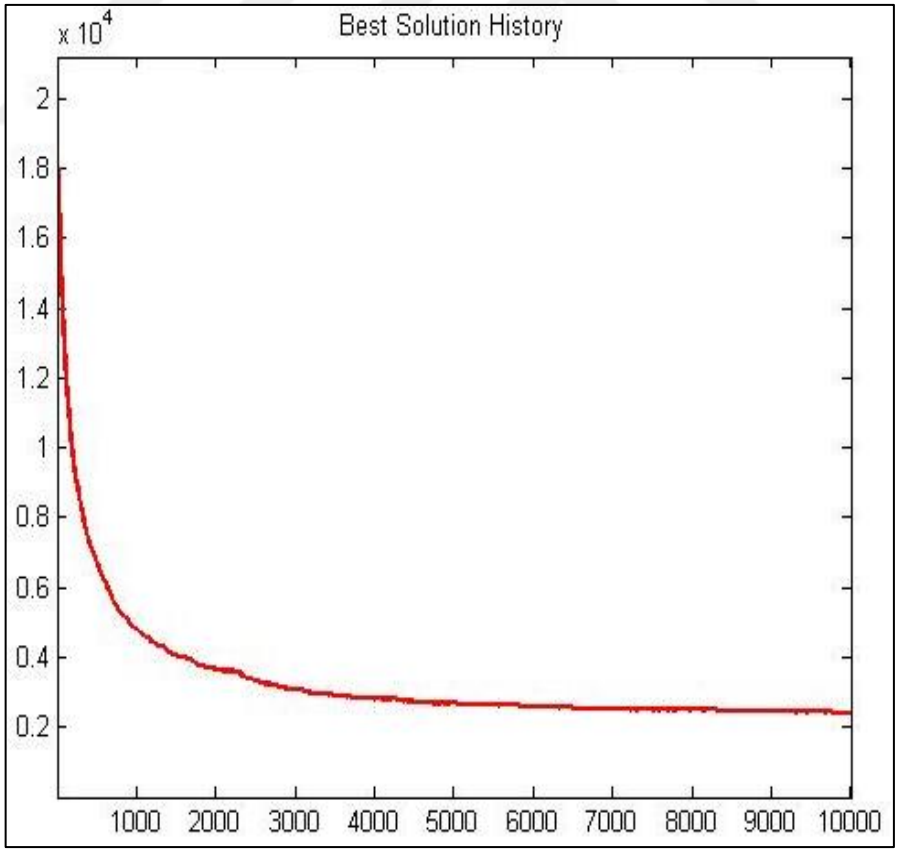

Figure 4.37: Best Solution History for (200) Cities using (10000) Iterations, (50) Population Size.

We notice from Figure (4.37) when using that the date of the best solutions in iterations (10000) and the number of cities (200) using (50) Population Size is not optimal. The curve shows that the improvement in solutions did not stop at the number of iterations (10000), as the graph shows There are better solutions after this limit of repetition, so it is clear from this experiment that the number of iterations must be increased in proportion to the number of cities in order to increase the probability of obtaining better results.

I would repeat the same experiment (200) cities, but with (100) population size and (50000) iterations to see the effect on the results to get the best path and time.

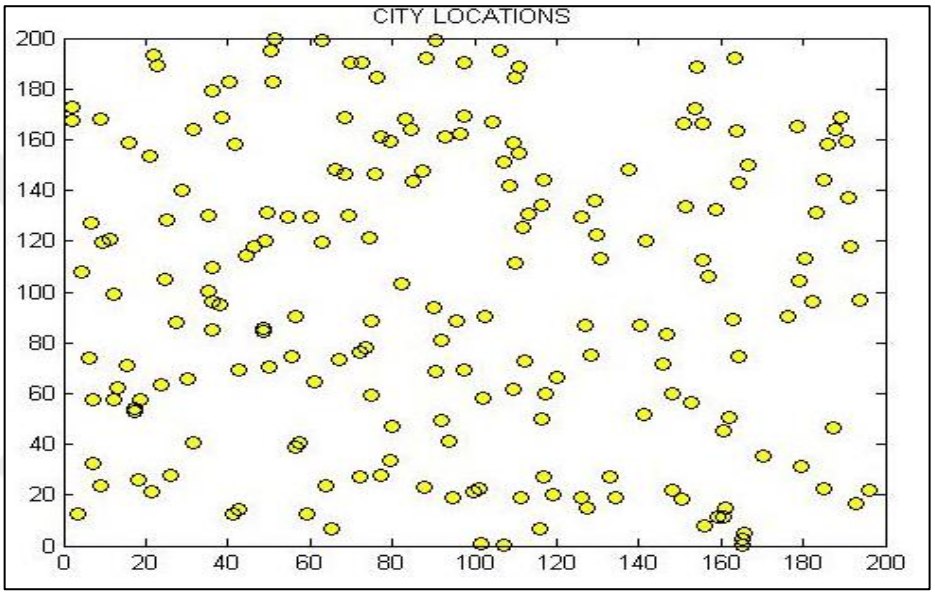

Figure 4.38: Cities Locations for (200) Cities Randomly by Program.

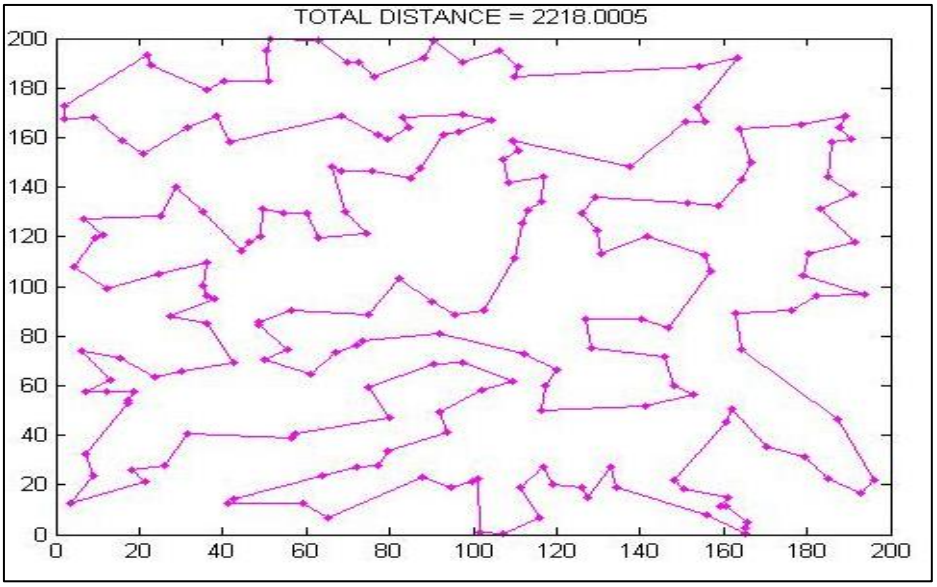

Figure 4.39: Total Distance for (200) Cities, Iterations (50000), Population Size (100).

The result show that the proposed path is an ideal path without repetition.

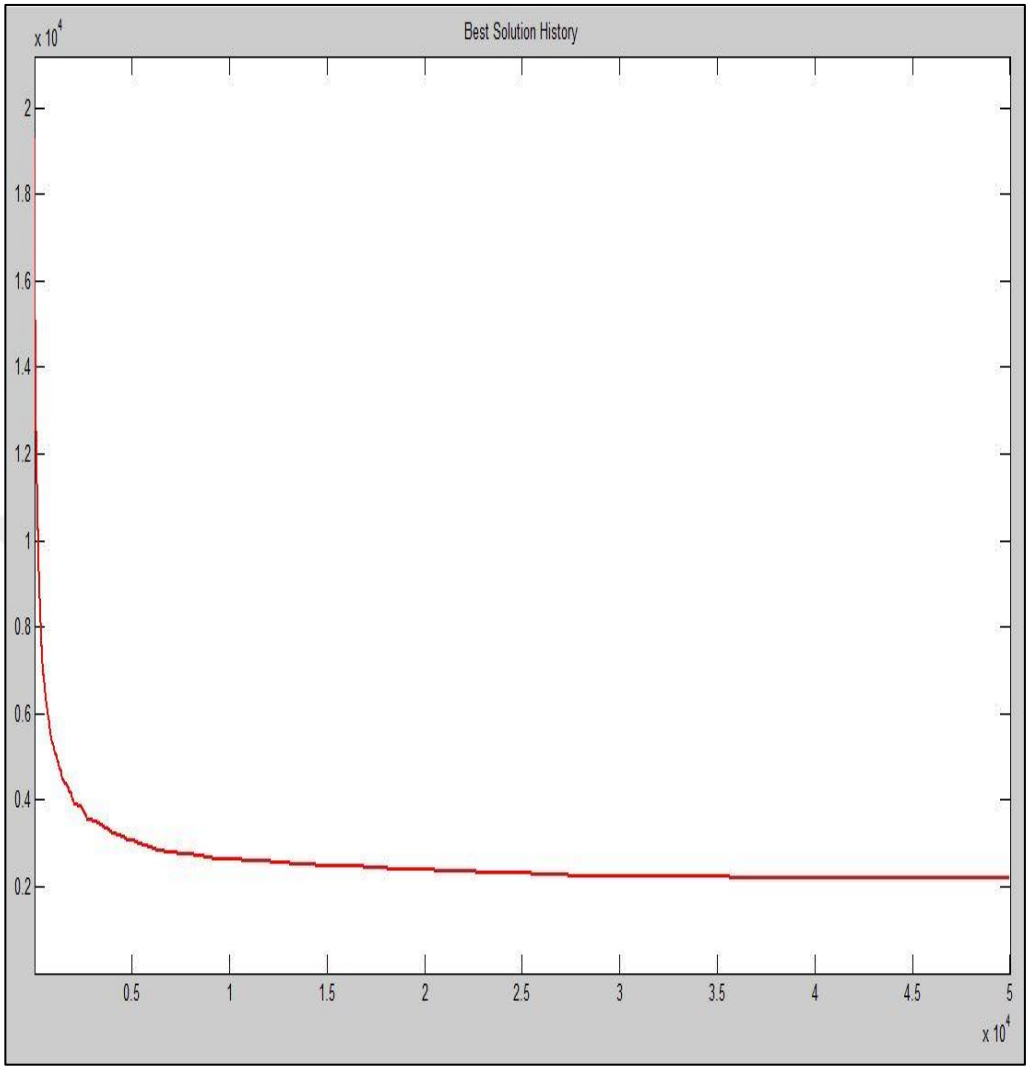

Figure 4.40: Best Solution History for (200) Cities, (10000) Iterations, (100) Population Size.

After applying the program and conducting the second experiment by changing the number of iterations to (10000), and the size of the population to (100), the curve shows that when the iteration reaches (3500), the level of improvement in the solution has stopped.

Therefore, the result showed that this solution is considered the optimal solution for the experiment, as there is no there is repetition in the cities visited, and the time obtained is an excellent time compared to previous experiments for the same number of cities.

Table 4.2: Algorithms Results Compared by Distance and Time Using Modify Genetic Algorithm.

Number of City	Number of Iterations	Population Size	Total Distance	The it Takes to Find the Best Path
10	10000	50	28.8577	06.1 second
10	1000	100	33.7464	03.2 second
50	10000	50	272.0300	06.6 second
50	1000	100	288.1928	04.33 second
100	10000	50	1111.765	06.50 second
100	1000	100	1205.439	05.47 second
200	10000	100	2625.787	35.05 second
200	50000	100	2316.120	85.30 second

The results of applying the modify genetic algorithm with population size change and iteration with using new crossover method are presented in the table that is now presented. The aim was to solve the problem of time delays that occurred when searching for the optimal way in big cities by achieving optimal results in terms of efficiency and pathfinding. The desired results were achieved by selecting a population size of (100) and iterations of (50000). The results were also excellent when the number of iteration was (10000) and the population size was (100). The path was ideal and, and the time to implement the program was less than the results presented in the previous table. This is the goal that is required to be the distance travelled without repetition and the time required to solve the problem is as small as possible.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Genetic algorithm employ to find best solution for travelling salesman issue. However, the technique of working depends on the choice of operators, and the result is greatly reliant on how the issue is encoded (swap and mutation methods). In this research, we used three techniques to solve the mutation, namely (reverse, swap, and sliding). The results shown that when these three techniques were combined, they significantly improved the algorithm's performance by calculating the number of variables for both repetition and generation size. This increased both the correctness of the results and the speed at which the answer was found. This had a definite impact on the algorithm's performance since the travelling salesman issue was solved better utilizing the suggested approach.

The assessments of testing carried out in the earlier chapters have been described. The strengths and weaknesses of the genetic algorithm are highlighted in solving TSP, and its ability to generate reasonably good solutions within a reasonable time frame. In the research, the effect of setting different variables on GA performance was studied, the most important of which was the effect of integrating the three cross over (flipping, swapping, and sliding) techniques. Where the results were positive and more accurate and precise and had a clear impact on the performance of the genetic algorithm. Moreover, he suggested continuing improvements to the algorithm or hybridizing it with other algorithms to improve its performance.

5.2 FUTURE WORK

As future work, the modified GA be used for find the best path in electrical dashboard. Also, use GA algorithm with another algorithm such as, firefly algorithm, cuckoo search algorithm, could be used to solve TSP problem.

REFERENCES

[1] A.-H. Zhou et al., “Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming,” Information, vol. 10, no. 1, p. 7, 2018.

[2] A. Kumar, “Improved Genetic Algorithm to Solve Small Scale Travelling Salesman Problem,” Proc. Int. Conf. Intell. Comput. Control Syst. ICICCS 2020, no. Iciccs, pp. 516–520, 2020, doi: 10.1109/ICICCS48265.2020.9120880.

[3] J. Zheng, J. Zhong, M. Chen, and K. He, “A reinforced hybrid genetic algorithm for the traveling salesman problem,” Comput. Oper. Res., vol. 157, p. 106249, 2023.

[4] S. Akter, N. Nahar, M. Shahadathossain, and K. Andersson, “A New Crossover Technique to Improve Genetic Algorithm and Its Application to TSP,” 2nd Int. Conf. Electr. Comput. Commun. Eng. ECCE 2019, pp. 7–9, 2019, doi: 10.1109/ECACE.2019.8679367.

[5] J. Song, G. Xue, and Y. Kang, “A novel method for optimum global positioning system satellite selection based on a modified genetic algorithm,” PLoS One, vol. 11, no. 3, pp. 1–14, 2016, doi: 10.1371/journal.pone.0150005.

[6] B. Lin, X. Sun, and S. Salous, “Solving Travelling Salesman Problem with an Improved Hybrid Genetic Algorithm,” J. Comput. Commun., vol. 04, no. 15, pp. 98–106, 2016, doi: 10.4236/jcc.2016.415009.

[7] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, “A hybrid genetic algorithm for the traveling salesman problem with drone,” J. Heuristics, vol. 26, no. 2, pp. 219–247, 2020, doi: 10.1007/s10732-019-09431-y.

[8] H. Benbya, T. H. Davenport, and S. Pachidi, “Artificial intelligence in organizations: Current state and future opportunities,” MIS Q. Exec., vol. 19, no. 4, 2020.

[9] A. Pannu and M. T. Student, “Artificial Intelligence and its Application in Different Areas,” Certif. Int. J. Eng. Innov. Technol., vol. 9001, no. 10, pp. 2277–3754, 2008.

[10] S. D. Erokhin, “A Review of Scientific Research on Artificial Intelligence,” 2019 Syst. Signals Gener. Process. F. Board Commun. SOSG 2019, pp. 1–4, 2019, doi: 10.1109/SOSG.2019.8706723.

[11] P. M. Hariyadi, P. T. Nguyen, I. Iswanto, and D. Sudrajat, “Traveling salesman problem solution using genetic algorithm,” J. Crit. Rev., vol. 7, no. 1, pp. 56–61, 2020.

[12] M. A. Al-Furhud and Z. H. Ahmed, “Experimental Study of a Hybrid Genetic Algorithm for the Multiple Travelling Salesman Problem,” Math. Probl. Eng., vol. 2020, 2020, doi: 10.1155/2020/3431420.

[13] A. Uthayasuriyan, H. Chandran, U. V Kavvin, S. H. Mahitha, and G. Jeyakumar, “A Comparative Study on Genetic algorithm and Reinforcement Learning to Solve the Traveling Salesman Problem,” Res. Reports Comput. Sci., pp. 1–12, 2023.

[14] İ. İlhan, “An Application on Mobile Devices with Android and IOS Operating Systems Using Google Maps APIs for the Traveling Salesman Problem,” Appl. Artif. Intell., vol. 31, no. 4, pp. 332–345, 2017, doi: 10.1080/08839514.2017.1339983.

[15] A. Arram and M. Ayob, “A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems,” Comput. Ind. Eng., vol. 133, pp. 267–274, 2019, doi: 10.1016/j.cie.2019.05.012.

[16] J. C. Zhang, “Comparison of various algorithms based on TSP solving,” J. Phys. Conf. Ser., vol. 2083, no. 3, 2021, doi: 10.1088/1742-6596/2083/3/032007.

[17] I. Korejo, S. Yang, and C. Li, “A directed mutation operator for real coded genetic algorithms,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6024 LNCS, no. PART 1, pp. 491–500, 2010, doi: 10.1007/978-3-642-12239-2_51.

[18] K. F. Man, K. S. Tang, and S. Kwong, “Genetic Algorithms in Speech Recognition Systems,” pp. 199–257, 1999, doi: 10.1007/978-1-4471-0577-0_8.

[19] S. Hashemi, S. Kiani, N. Noroozi, and M. E. Moghaddam, “An image contrast enhancement method based on genetic algorithm,” Pattern Recognit. Lett., vol. 31, no. 13, pp. 1816–1824, 2010.

[20] B. Desjardins, R. Falcon, R. Abielmona, and E. Petriu, Planning robust sensor relocation trajectories for a mobile robot with evolutionary multi-objective optimization, vol. 676. 2017. doi: 10.1007/978-3-319-47715-2_8.

[21] A. Öncan, “Laporte, 2009 Öncan T., Altınel İK, Laporte G,” A Comp. Anal. Sev. asymmetric Travel. Salesm. Probl. Formul. Comput. Oper. Res., vol. 36, no. 3, pp. 637–654, 2009.

[22] Ankita, “Travelling Salesman Problem Using GA-ACO Hybrid Approach: A Review,” in ICDSMLA 2020: Proceedings of the 2nd International Conference on Data Science, Machine Learning and Applications, Springer, 2022, pp. 105–113.

[23] A. Riazi, “Genetic algorithm and a double-chromosome implementation to the traveling salesman problem,” SN Appl. Sci., vol. 1, no. 11, 2019, doi: 10.1007/s42452-019-1469-1.

[24] C. Rego, “esar; Gamboa, Dorabela; Glover, Fred; Osterman, Colin (2011),” Traveling salesman prob-lem heuristics: leading methods, implementations and latest advances”,” Eur. J. Oper-ational Res., vol. 211, no. 3, pp. 427–441.

[25] E. A. Silver, “An overview of heuristic solution methods,” J. Oper. Res. Soc., vol. 55, no. 9, pp. 936–956, 2004, doi: 10.1057/palgrave.jors.2601758.

[26] P. D. Jackovich, “AFIT Scholar Theses and Dissertations Student Graduate Works Solving the Traveling Salesman Problem Using Ordered-Lists,” 2019, [Online]. Available: https://scholar.afit.edu/etd/2303

[27] R. G. Bland and D. F. Shallcross, “Large travelling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation,” Oper. Res. Lett., vol. 8, no. 3, pp. 125–128, 1989, doi: 10.1016/0167-6377(89)90037-0.

[28] J. K. Lenstra and A. H. G. R. Kan, “Some simple applications of the travelling salesman problem,” J. Oper. Res. Soc., vol. 26, no. 4, pp. 717–733, 1975.

[29] Y. Saji and M. Barkatou, “A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem,” Expert Syst. Appl., vol. 172, no. January, p. 114639, 2021, doi: 10.1016/j.eswa.2021.114639.

[30] J. Stastný, V. Skorpil, and L. Cizek, “Traveling Salesman Problem optimization by means of graph-based algorithm,” in 2016 39th International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2016, pp. 207–210.

[31] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, “An analysis of several heuristics for the traveling salesman problem,” Fundam. Probl. Comput. Essays Honor Profr. Daniel J. Rosenkrantz, vol. 6, no. 3, pp. 45–69, 2009, doi: 10.1007/978-1-4020-9688-4_3.

[32] G. Kizilateş and F. Nuriyeva, “On the Nearest Neighbor Algorithms for the Traveling Salesman Problem,” Adv. Intell. Syst. Comput., vol. 225, no. 1, pp. 111–118, 2013, doi: 10.1007/978-3-319-00951-3_11.

[33] M. A. Al-Furhud and Z. H. Ahmed, “Genetic algorithms for the multiple travelling salesman problem,” Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 7, 2020.

[34] T. B. Chandrawati, “A Review of Firefly Algorithms for Path Planning , Vehicle Routing and Traveling Salesman Problems,” no. October, pp. 30–35, 2018.

[35] M. Saraei and P. Mansouri, “Solving of Travelling Salesman Problem using Firefly Algorithm with Greedy Approach,” Sci. J., vol. 36, no. 2015, 2015.

[36] Z. A. Ali, S. A. Rasheed, and N. No’man Ali, “An enhanced hybrid genetic algorithm for solving traveling salesman problem,” Indones. J. Electr. Eng. Comput. Sci., vol. 18, no. 2, pp. 1035–1039, 2020, doi: 10.11591/ijeecs.v18.i2.pp1035-1039.

[37] S. Mahmoudinazlou and C. Kwon, “A Hybrid Genetic Algorithm with Type-Aware Chromosomes for Traveling Salesman Problems with Drone,” arXiv Prepr. arXiv2303.00614, 2023.

[38] J. Pacheco, S. Casado, L. Núñez, and O. Gómez, “Analysis of new variable selection methods for discriminant analysis,” Comput. Stat. Data Anal., vol. 51, no. 3, pp. 1463–1478, 2006, doi: 10.1016/j.csda.2006.04.019.

[39] A. Newman, “An improved analysis of the Mömke-Svensson algorithm for graph-TSP on subquartic graphs,” SIAM J. Discret. Math., vol. 34, no. 1, pp. 865–884, 2020, doi: 10.1137/19M1259353.

[40] Z.-J. Bai, D. Chu, and R. C. E. Tan, “Computing the nearest doubly stochastic matrix with a prescribed entry,” SIAM J. Sci. Comput., vol. 29, no. 2, pp. 635–655, 2007.

[41] Y. Sakurai, K. Takada, N. Tsukamoto, T. Onoyama, R. Knauf, and S. Tsuruta, “A simple optimization method based on backtrack and ga for delivery schedule,” in 2011 IEEE Congress of Evolutionary Computation (CEC), IEEE, 2011, pp. 2790–2797.

[42] M. Hahsler and K. Hornik, “TSP-Infrastructure for the traveling salesperson problem,” J. Stat. Softw., vol. 23, no. 2, pp. 1–21, 2007.

[43] G. A. JAYALAKSHMI, S. SATHIAMOORTHY, and R. RAJARAM, “a Hybrid Genetic Algorithm — a New Approach To Solve Traveling Salesman Problem,” Int. J. Comput. Eng. Sci., vol. 02, no. 02, pp. 339–355, 2001, doi: 10.1142/s1465876301000350.

[44] G. Laporte, “A concise guide to the Traveling Salesman Problem,” J. Oper. Res. Soc., vol. 61, no. 1, pp. 35–40, 2010, doi: 10.1057/jors.2009.76.

[45] J. Zhang and W. Si, “Improved enhanced self-tentative PSO algorithm for TSP,” in 2010 Sixth International Conference on Natural Computation, IEEE, 2010, pp. 2638–2641.

[46] J. Bi, G. Zhou, Y. Zhou, Q. Luo, and W. Deng, “Artificial electric field algorithm with greedy state transition strategy for spherical multiple traveling salesmen problem,” Int. J. Comput. Intell. Syst., vol. 15, no. 1, p. 5, 2022.

[47] C. E. Tungom, B. Niu, T. Xing, J. Yuan, and H. Wang, “Copy and Paste: A Multi-offspring Genetic Algorithm Crossover Operator for the Traveling Salesman Problem,” in Advances in Swarm Intelligence: 13th International Conference, ICSI 2022, Xi’an, China, July 15–19, 2022, Proceedings, Part I, Springer, 2022, pp. 272–281.

[48] A. Shakir Hameed, H. M. B. Alrikabi, A. A. Abdul–Razaq, Z. H. Ahmed, H. K. Nasser, and M. L. Mutar, “Appling the Roulette Wheel Selection Approach to Address the Issues of Premature Convergence and Stagnation in the Discrete Differential Evolution Algorithm,” Appl. Comput. Intell. Soft Comput., vol. 2023, 2023.

[49] I. B. K. Widiartha, S. E. Anjarwani, and F. Bimantoro, “Traveling salesman problem using multi-element genetic algorithm,” in 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), IEEE, 2017, pp. 1–4.

[50] D. R. Singh, “Using Group Theory to Generate Initial Population for a Genetic Algorithm for Solving Traveling Salesman,” 2023.

[51] H. Xu and H. Lan, “An Adaptive Layered Clustering Framework with Improved Genetic Algorithm for Solving Large-Scale Traveling Salesman Problems,” Electronics, vol. 12, no. 7, p. 1681, 2023.

[52] X. Yu, L. Yu, M. Zheng, J. Lu, and L. Zhang, “Firefly algorithm and ant colony algorithm to optimize the traveling salesman problem,” in Journal of Physics: Conference Series, IOP Publishing, 2022, p. 12010.

[53] Z. H. Ahmed, “Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator,” Int. J. Biometrics Bioinforma., vol. 3, no. 6, p. 96, 2010.

[54] V. Jain and J. S. Prasad, “Solving Travelling Salesman Problem Using Greedy Genetic Algorithm GGA,” Int. J. Eng. Technol., vol. 9, no. 2, pp. 1148–1154, 2017, doi: 10.21817/ijet/2017/v9i2/170902188.

[55] Y. Gui, B. Gong, and Y. Cheng, “An algorithm for air cargo forwarders consolidation problem,” Proc. - 4th Int. Conf. Nat. Comput. ICNC 2008, vol. 7, pp. 372–376, 2008, doi: 10.1109/ICNC.2008.125.

[56] Z. Zhao, S. Liu, M. Zhou, and A. Abusorrah, “Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem,” IEEE/CAA J. Autom. Sin., vol. 8, no. 6, pp. 1199–1209, 2020.

[57] F. H. Khan, N. Khan, S. Inayatullah, and S. T. Nizami, “Solving TSP problem by using genetic algorithm,” Int. J. Basic Appl. Sci., vol. 9, no. 10, pp. 79–88, 2009.

[58] Y. Deng, Y. Liu, and D. Zhou, “An improved genetic algorithm with initial population strategy for symmetric TSP,” Math. Probl. Eng., vol. 2015, 2015.

[59] S. S. Juneja, P. Saraswat, K. Singh, J. Sharma, R. Majumdar, and S. Chowdhary, “Travelling Salesman Problem Optimization Using Genetic Algorithm,” Proc. - 2019 Amity Int. Conf. Artif. Intell. AICAI 2019, no. February, pp. 264–268, 2019, doi: 10.1109/AICAI.2019.8701246.

[60] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” Phys. A Stat. Mech. its Appl., vol. 391, no. 6, pp. 2193–2196, 2012, doi: 10.1016/j.physa.2011.12.004.

[61] C. Fu, L. Zhang, X. Wang, and L. Qiao, “Solving TSP problem with improved genetic algorithm,” in AIP Conference Proceedings, AIP Publishing, 2018.

[62] N. M. Razali and J. Geraghty, “Genetic algorithm performance with different selection strategies in solving TSP,” in Proceedings of the world congress on engineering, International Association of Engineers Hong Kong, China, 2011, pp. 1–6.

[63] J. Jun, J. Kang, D. Jeong, and H. Lee, “An efficient approach for optimizing full field development plan using Monte-Carlo simulation coupled with Genetic Algorithm and new variable setting method for well placement applied to gas condensate field in Vietnam,” 2017, doi: 10.1177/0144598716680307.

[64] A. Shukla, H. M. Pandey, and D. Mehrotra, “Comparative review of selection techniques in genetic algorithm,” 2015 1st Int. Conf. Futur. Trends Comput. Anal. Knowl. Manag. ABLAZE 2015, pp. 515–519, 2015, doi: 10.1109/ABLAZE.2015.7154916.

[65] M. J. Li et al., “Optimizing emission inventory for chemical transport models by using genetic algorithm,” Atmos. Environ., vol. 44, no. 32, pp. 3926–3934, 2010.

[66] Y. Liu, T.-U. Lee, A. Rezaee Javan, and Y. M. Xie, “Extending Goldberg’s method to parametrize and control the geometry of Goldberg polyhedra,” R. Soc. Open Sci., vol. 9, no. 8, p. 220675, 2022.

[67] Z. H. Ahmed, “Improved genetic algorithms for the travelling salesman problem,” Int. J. Process Manag. Benchmarking, vol. 4, no. 1, pp. 109–124, 2014, doi: 10.1504/IJPMB.2014.059449.

[68] D. Beasley, D. R. Bull, and R. R. Martin, “An overview of genetic algorithms: Part 1, fundamentals,” Univ. Comput., vol. 15, no. 2, pp. 56–69, 1993.

[69] D. Thierens, “Scalability problems of simple genetic algorithms,” Evol. Comput., vol. 7, no. 4, pp. 331–352, 1999.

[70] A. Pandarwidi S, G. Siantoro, and A. Khamidi, “The Effects of Zigzag Ladder Exercise Crossover Shuffle, In Out Shuffle and Ali Shuffle Against Speed and Agility,” Int. J. Educ. Vocat. Stud., vol. 1, no. 8, p. 109, 2020, doi: 10.29103/ijevs.v2i1.2040.

[71] J. Y. Potvin, “Genetic algorithms for the traveling salesman problem,” Ann. Oper. Res., vol. 63, pp. 339–370, 1996, doi: 10.1007/bf02125403.

[72] J. Zhu, C. Zhu, P. Zhou, Z. He, and D. You, “An optimizing diffusion kernel-based binary encoding strategy with genetic algorithm for fringe projection profilometry,” IEEE Trans. Instrum. Meas., vol. 71, pp. 1–8, 2022.

[73] G. Büyüközkan and O. Feyzioğlu, “A fuzzy-logic-based decision-making approach for new product development,” Int. J. Prod. Econ., vol. 90, no. 1, pp. 27–45, 2004, doi: 10.1016/S0925-5273(02)00330-4.

[74] F. Paquin, J. Rivnay, A. Salleo, N. Stingelin, and C. Silva, “Multi-phase semicrystalline microstructures drive exciton dissociation in neat plastic semiconductors,” J. Mater. Chem. C, vol. 3, no. 1, pp. 10715–10722, 2015, doi: 10.1039/b000000x.

[75] B. A. Julstrom, “The Blob Code is competitive with edge-sets in genetic algorithms for the minimum routing cost spanning tree problem,” GECCO 2005 - Genet. Evol. Comput. Conf., pp. 585–590, 2005, doi: 10.1145/1068009.1068108.