A Report on Project

Efficient Path Module

ABSTRACT

Genetic Algorithm is a search technique used in computing to find the optimal solution to a computational problem that maximizes or minimizes a particular function. Genetic Algorithm is used to solve the Travelling Salesman Problem where one has to find the shortest route among the cities from the origin. The Travelling Salesman Problem (TSP) is well known in the field of combinatorial optimization. Since it is an NP-complete problem, there is no efficient method to solve this problem and give the best result.

TABLE OF CONTENTS

Problem Definition

Calculating shortest distance in Network is one of the important network functions for QoS routing, MLPS Path selection, ATM networking and traffic networking engineering. In voice and video calling finding cheapest path is not an easy task. In order to find solution for this problem many researches and algorithms are designed. This project proposes two techniques which will reduce decentralization errors and provide shortest routing distance by developing faster algorithms. Providing shortest path is the important factor for successful design in QOS system. Simulator is used to test new techniques which show execution time efficiency in new system.

Genetic Algorithm is a search technique used in computing to find the optimal solution to a computational problem that maximizes or minimizes a particular function. Genetic Algorithm is used to solve the Travelling Salesman Problem where one has to find the shortest route among the cities from the origin. The Travelling Salesman Problem (TSP) is well known in the field of combinatorial optimization. Since it is an NP-complete problem, there is no efficient method to solve this problem and give the best result.

Many algorithms are used to solve travelling salesman problem. Some algorithms give optimal solution, but some other algorithms give the nearest optimal solution. The genetic algorithm is a heuristic method which is used to improve the solution space for the Travelling Salesman Problem. The genetic algorithm results in nearest optimal solution within a reasonable time. This project mainly focuses on the comparative study of different selection methods and crossover operators in genetic algorithm to solve Travelling Salesman Problem and finally report the results.

TECHNOLOGY USED

2.1. JSP:

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent methods for building Web-based applications. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

A Java Server Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands. Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.

Why Use JSP?

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets; JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP, etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

2. HTML:

HTML is the language for describing the structure of Web pages. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

With HTML, authors describe the structure of pages using markup. The elements of the language label pieces of content such as “paragraph,” “list,” “table,” and so on.

3. CSS:

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

CSS can be added to HTML elements in 3 ways:

- Inline – by using the style attribute in HTML elements
- Internal – by using a <style> element in the <head> section
- External – by using an external CSS file

4. JAVA:

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.
Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995

How Java is used in web applications?

Java Web Application is used to build dynamic websites. Java offers support for the web application through JSPs and Servlets. We can build a website with static HTML web pages but when we want data to be dynamic, we require the web application.

Why is Java used for web applications?

Java is the name for both the programming language that can be used for building complex web applications and for the software platform that used this programming language as its most essential component. It is widely used by development companies to build secure, robust and scalable web applications.

5.GlassFish Server:

GlassFish is an open-source application server project started by Sun Microsystems for the Java EE platform and as such supports Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, servlets, etc. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. Optional components can also be installed for additional services.

TRAVELLING SALESMAN PROBLEM

1 INTRODUCTION:

Travelling salesman problem (TSP) is a well- known classic problem from computer science. Its mathematical formulation is simple, and one can state a simple strategy to solve. Such a strategy is often impractical and as yet there is no efficient algorithm for this problem that works consistently in all instances.TSP is a choice problem for combinatorial optimization problems and one that has received the most attention.

Another thing to note is that one can‘t find for the Travelling Salesman Problem a consistently efficient method of solution, as it belongs to a set of combinatorial problem that are called NP-complete problems. NP-complete problems have the distinction that there is no known algorithm that is efficient, practical and works in all instances.

The Travelling Salesman Problem is easy to state: given a finite number of cities along with the cost of travel between each pair of them, TSP is the search of the shortest tour that visits the given cities exactly once.

Genetic Algorithm is a search technique used in computing to find the optimal solution to a computational problem that maximizes or minimizes a particular function. Genetic Algorithm is used to solve the Travelling Salesman Problem where one has to find the shortest route among the cities from the origin. The Travelling Salesman Problem (TSP) is well known in the field of combinatorialoptimization. Since it is an NP-complete problem, there is no efficient method to solve this problem and give the best result. Many algorithms are used to solve travelling salesman problem. Some algorithms give optimal solution, but some other algorithms give the nearest optimal solution. The genetic algorithm is a heuristic method which is used to improve the solution space for the Travelling Salesman Problem. The genetic algorithm results in nearest optimal solution within a reasonable time.

2. MATHEMATICAL FORMULATION OF TSP:

Mathematically, the goal of this problem is to find a tour, among all the possible tours, that minimizes the total distance the salesman travels. The other criterion that the problem has to satisfy is that each city should be visited once and only once, except that he returns to the city from which the salesman starts.

TSP can be formulated as an integer linear program. Label the cities with the numbers 0,…,n. For n cities to visit, let xij be the variable that has a value 1 if the salesman goes from city i to city j and a value 0 if the salesman does not go from city i to city j. Letdijbe the distance from city i to city j.

Then the TSP can be stated as-

Minimize the linear objective function:

$Z = xnj \neq ij=0\ ni=0$ i, j, d[i][j]

$0 \leq xij \leq 1 \qquad i, j = 0, \cdots, n$

Subject to the constraint,

$xni=0, \neq j$i, j=1 j=0,…n $xnj=0, j \neq i$ij=1 i=0,…,n

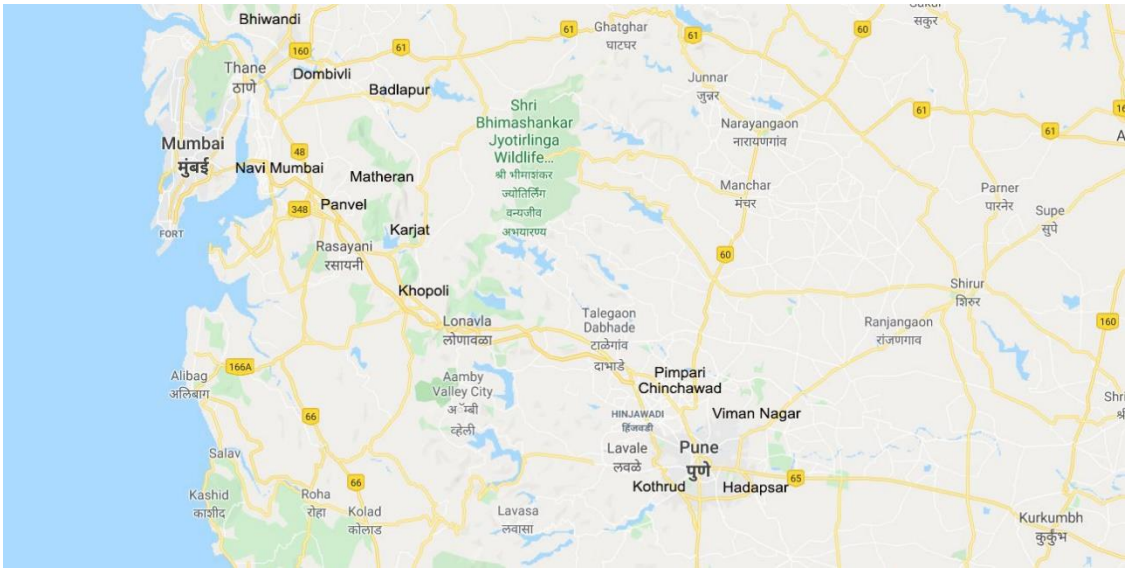
4.3 METHODS TO SOLVE TSP

Branch and Bound method

This method belongs to the oldest ones and the most often used algorithms for the TSP solutions. The merit of the method rests in a gradual decomposition of a possible solution set into a number of mutually disjunctive subsets labeled as branches.

In each step the following is estimated: The upper limit of the objective function that is most often the value of the objective function zH without respecting limits and maximum lower bound of an objective function zD of acceptable solutions which are known to us within the step. Both the estimates can be employed for seeking non-prospective directions of further procedures: if for any branch zH.

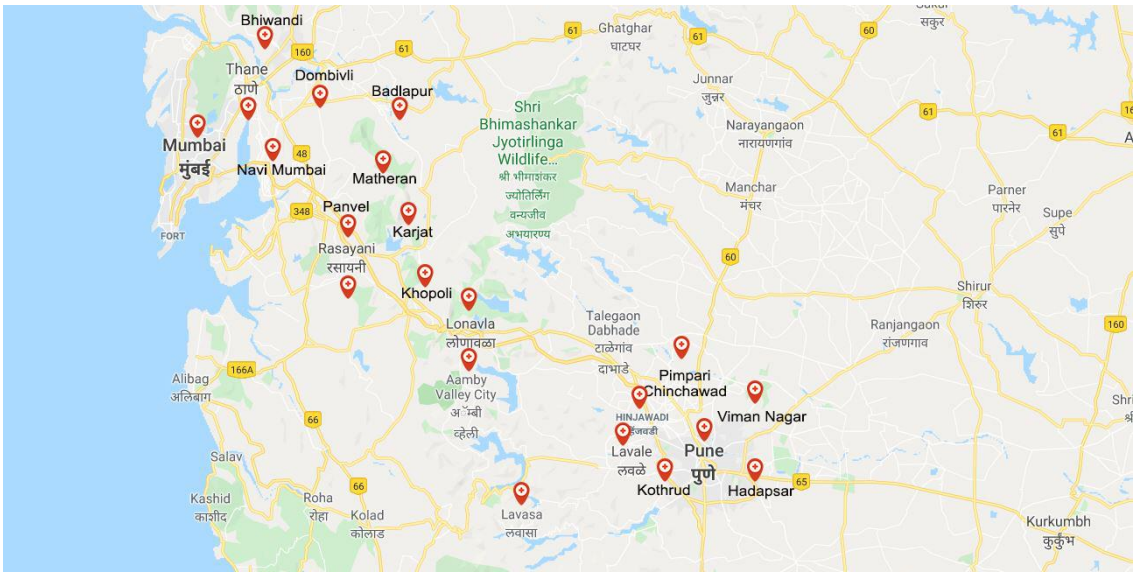
The map we used are:



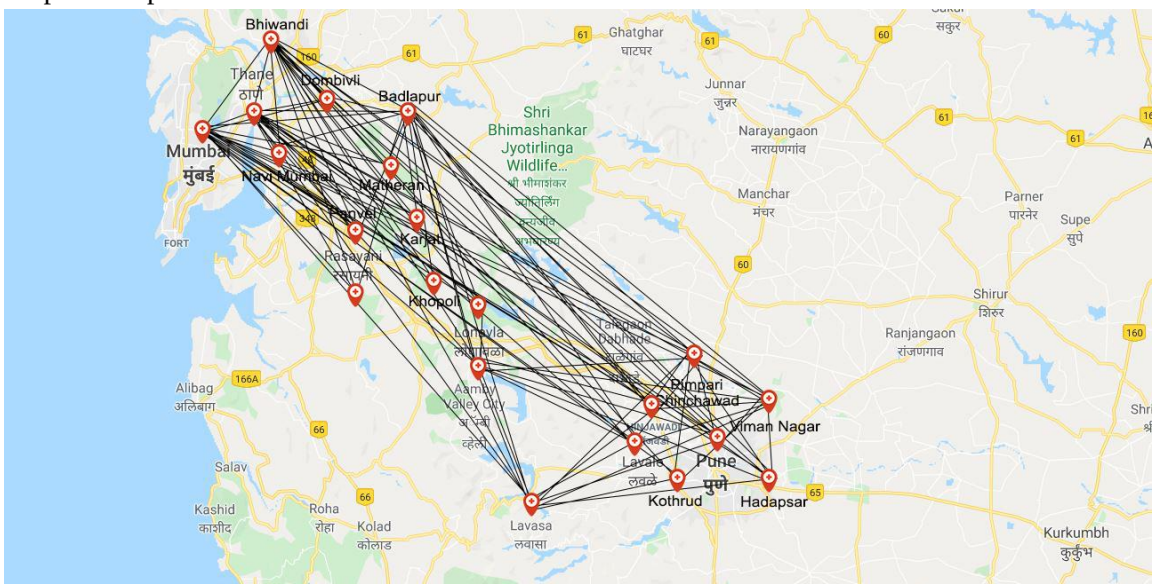
The places for choice are:

- Mumbai,
- Thane,
- NaviMumbai,
- Lonavla,
- Dombivli,
- Bhiwandi,
- Panvel,
- Rasayani,
- Karjat,
- Matheran,
- Badlapur,

- Lavale,
- Hinjawadi,
- Kothrud,
- Viman Nagar,
- PimpariChinchawad,
- Khopoli,
- Hadapsar,
- Pune,
- Lavasa,
- Aamby Valley City



The all possible path network are:



APPLICATIONS OF TSP:-

VEHICLE ROUTING:-

Suppose that in a city _n‘ mail boxes have to be emptied everyday within a certain period of time, say one hour. The problem is to do the particular work with less number of vehicles in minimum time.

SCHOOL BUS ROUTING PROBLEM:-

The objective of this problem is to obtain a bus loading pattern such that the number of routes is minimized.The total distance travelled by all buses is kept at minimum.

MISSION PLANNING PROBLEM:-

The objective is to determine an optimal path for army men to accomplish the goals of the mission in the minimum possible time.

PRINTING PRESS SCHEDULING PROBLEM:-

One of the major and primary applications of the mTSP arises in scheduling a printing press for a periodical with multi-editions. Here, there exist five pairs of cylinders between which the paper rolls and both sides of a page are printed simultaneously. There exist three kind of forms, namely 4-, 6- and 8-page forms, which are used to print the editions. The scheduling problem consists of deciding which form will be on which run and the length of each run. In the mTSP vocabulary, the plate change costs are the inter-city costs

CREW SCHEDULING PROBLEM:-

An application for deposit carrying between different branch banks is reported by Svestka&Huckfeldt (1973). Here, deposits need to be picked up at branch banks and returned to the central office by a crew of messengers. The problem is to determine the routes having a total minimum distance.

GENETIC ALGORITHM

Genetic Algorithm is a branch of Artificial Intelligence stochastic search technique that is widely used in the field of optimization. GA‘s are computer algorithms that search for good solutions to a problem from various possible solutions. This is well suited to resolve variety of practical problems, computational problems. It randomly generates a set of possible solutions to a problem, representing each as a fixed length character string. It then test each possible solution against the problem using a fitness function to evaluate each solution. Keep the best solutions, and use them to generate new possible solutions. Repeat the previous two steps until either an acceptable solution is found. GA was proposed by John Holland, his students and his colleagues at the University of Michigan. Carrasco et al (2001) Genetic Algorithm (GA) model introduced by John Holland in 1975 triggered a wide interest in the application of types of heuristic, which mimic the natural evolutionary characteristic present in the biological species with the purpose of solving optimization problem efficiently. Randy et al (1998) GAs is a branch of artificial intelligence‘s stochastic search technique that is widely used in the field of optimization. According to Biesbroek (1999) the study of GA was originated from studies done on cellular automata, which took place when Goldberg was at the University of Michigan. John Koza(1992)has used genetic algorithm to evolve programs to perform certain tasks. He called his method ―genetic programming‖ (GP).

ADVANTAGES AND DISADVANTAGES

3.8.1 ADVANTAGES

1. Easy to understand.
2. Unlike older AI systems, the GA‘s do not break easily even if the inputs changed slightly.
3. GA can be employed for a wide variety of optimization problems.
4. GA performs very well for large scale optimization problems which may be very difficult or impossible to solve by other traditional methods.

3.8.2 DISADVANTAGES

1. Sometimes have trouble finding the exact global optimum because there is no guaranty to find best solution.
2. It requires large number of fitness function evaluations depending on the number of individuals and the number of generations.
3. GA may take long time to evaluate the individuals.

3.9APPLICATIONS OF GENETIC ALGORITHM

I. **Scheduling**:
Facility, Production, Job and Transport Scheduling.

II. **Design**:
Circuit board layout, Communication Network design, parametric design in aircraft.

III. **Control**:
Missile evasion, Gas pipeline control, Pole balancing.

IV. Machine Learning:
Designing Neural Networks, Classifier Systems, and Learning rules.

V. **Robotics**:
Trajectory Planning, Path Planning

VI. Combinatorial Optimization:
TSP, Bin Packing, Set Covering, Routing.

VII. Image Processing:
Pattern recognition.

System Design

Backend code:
Here we have written backend code for implementing and maintain Web Page and Traveling Salesman Algorithm.

1. home.jsp

```
<%--
   Document   : sample
   Created on : 23 Feb, 2020, 10:25:39 AM
   Author     : Lenovo
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
                <title>Effective Path Module</title>
                <meta charset="utf-8" />
                <meta name="viewport" content="width=device-width, initial-scale=1" />
                <link rel="stylesheet" href="assets/css/main.css" />
        </head>
        <body style="background-color: #edeff6">

                <!-- Banner -->
                        <section id="banner" >
<h1 style="font-family: sans-serif;color: #5162a7">Effective Path Module</h1>
                                <p style="color: #404c82">Effective Path Module</p>
                        </section>

<!-- Start Point -->
<section id="one" class="wrapper" style="background-color: #ffffff">
                                <div class="inner">
                                        <div class="flex flex-3">
                                                <article>
                                                        <header>
                                                                <h3>Starting Point</h3>
                                                        </header>
```

```
                        <select id="cities" name="cities">
<option value="mumbai">Mumbai</option>
<option value="thane">Thane</option>
<option value="navi_mumbai">Navi Mumbai</option>
<option value="lonavla">Lonavla</option>
<option value="dombivli">Dombivli</option>
<option value="bhiwandi">Bhiwandi</option>
<option value="panvel">Panvel</option>
<option value="rasayani">Rasayani</option>
<option value="karjat">Karjat</option>
<option value="matheran">Matheran</option>
<option value="badlapur">Badlapur</option>
<option value="lavale">Lavale</option>
<option value="hinjawadi">Hinjawadi</option>
<option value="kothrud">Kothrud</option>
<option value="viman_nagar">Viman Nagar</option>
<option value="pimpari_chinchawad">PimpariChinchawad</option>
<option value="khopoli">Khopoli</option>
<option value="hadapsar">Hadapsar</option>
<option value="pune">Pune</option>
<option value="lavasa">Lavasa</option>
<option value="aamby_valley_city">Aamby Valley City</option>
</select>
</article>
                </div>
                        </div>
                </section>

                <!-- One -->
<form action="newjsp.jsp" method="post">
                <section id="one" class="wrapper" >
                        <div class="inner">
                                <div class="flex flex-3" >
                                        <article>
<input type="checkbox" id="mumbai" name="city" value="0">
<label for="mumbai" style="color: #000000">Mumbai</label>
</br>
<input type="checkbox" id="thane" value="1" name="city">
<label for="thane" style="color: #000000">Thane</label>
</br>
<input type="checkbox" id="navi_mumbai" value="2" name="city">
<label for="navi_mumbai" style="color: #000000">Navi Mumbai</label>
</br>
<input type="checkbox" id="lonavla" value="3" name="city">
```

```
<label for="lonavla" style="color: #000000">Lonavla</label>
</br>
<input type="checkbox" id="dombivli" value="4" name="city">
<label for="dombivli" style="color: #000000">Dombivli</label>
</br>
<input type="checkbox" id="bhiwandi" value="5" name="city">
<label for="bhiwandi" style="color: #000000">Bhiwandi</label>
</br>
<input type="checkbox" id="panvel" value="6" name="city">
<label for="panvel" style="color: #000000">Panvel</label>
</article>
<article>
<input type="checkbox" id="rasayani" name="city" value="7">
<label for="rasayani" style="color: #000000">Rasayani</label>
</br>
<input type="checkbox" id="karjat" name="city" value="8">
<label for="karjat" style="color: #000000">Karjat</label>
</br>
<input type="checkbox" id="matheran" name="city" value="9">
<label for="matheran" style="color: #000000">Matheran</label>
</br>
<input type="checkbox" id="badlapur" name="city" value="10">
<label for="badlapur" style="color: #000000">Badlapur</label>
</br>
<input type="checkbox" id="lavale" name="city" value="11">
<label for="lavale" style="color: #000000">Lavale</label>
</br>
<input type="checkbox" id="hinjawadi" name="city" value="12">
<label for="hinjawadi" style="color: #000000">Hinjawadi</label>
</br>
<input type="checkbox" id="kothrud" name="city" value="13">
<label for="kothrud" style="color: #000000">Kothrud</label>
</article>
<article>
<input type="checkbox" id="viman_nagar" name="city" value="14">
<label for="viman_nagar" style="color: #000000">Viman Nagar</label>
</br>
<input type="checkbox" id="pimpari_chinchawad" name="city" value="15">
<label for="pimpari_chinchawad" style="color: #000000">PimpariChinchawad</label>
</br>
<input type="checkbox" id="khopoli" name="city" value="16">
<label for="khopoli" style="color: #000000">Khopoli</label>
</br>
<input type="checkbox" id="hadapsar" name="city" value="17">
<label for="hadapsar" style="color: #000000">Hadapsar</label>
```

```
</br>
<input type="checkbox" id="pune" name="city" value="18">
<label for="pune" style="color: #000000">Pune</label>
</br>
<input type="checkbox" id="lavasa" name="city" value="19">
<label for="lavasa" style="color: #000000">Lavasa</label>
</br>
<input type="checkbox" id="aamby_valley_city" name="city" value="20">
<label for="aamby_valley_city" style="color: #000000">Aamby Valley City</label>
</article>
</div>
</div>
</section>
<footer>
<input style="margin-left: 47%" type="submit" value="Find The Path" name="Submit" class="button
special"/>
</footer>
</form>
</br></br>
<!-- Footer -->
<footer id="footer">
<div class="inner">
<div class="flex">
<div class="copyright">&copy; Effective Path Module </div>
<ul class="icons">
<li><a href="#" class="icon fa-facebook"><span class="label">Facebook</span></a></li>
<li><a href="#" class="icon fa-twitter"><span class="label">Twitter</span></a></li>
<li><a href="#" class="icon fa-linkedin"><span class="label">linkedIn</span></a></li>
<li><a href="#" class="icon fa-pinterest-p"><span class="label">Pinterest</span></a></li>
<li><a href="#" class="icon fa-vimeo"><span class="label">Vimeo</span></a></li>
</ul>
</div>
</div>
</footer>
<!-- Scripts -->
                    <scriptsrc="assets/js/jquery.min.js"></script>
                    <scriptsrc="assets/js/skel.min.js"></script>
                    <scriptsrc="assets/js/util.js"></script>
                    <scriptsrc="assets/js/main.js"></script>
          </body>
</html>
```

2. result.jsp

```
<%--
    Document   : newjsp
    Created on : 15 Feb, 2020, 10:51:24 PM
    Author     : Lenovo
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"
import="com.mrunal.jsp.NewClass,com.mrunal.jsp.DrawImage"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Effective Path Module</title>
            <meta charset="utf-8" />
            <meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="stylesheet" href="assets/css/main.css" />
</head>
<body>
<%
boolean[] cities =
{true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false};
String[] selected;
selected = request.getParameterValues("city");
if(selected != null){
for(inti=0;i<selected.length;i++){
cities[Integer.parseInt(selected[i])] = true;
              }
          }
         %>


<!-- Banner -->
                                <section id="banner" >
<h1 style="font-family: sans-serif;color: #5162a7">Effective Path Module</h1>
                                        <p style="color: #404c82">Effective Path Module</p>
                                </section>
<!-- Three -->
                                <section id="three" class="wrapper special">
```

```
<div class="inner">
<header class="align-center">
<h2>Distance =  <%= NewClass.game(cities)%> km</h2>
</head>
</div>
</section>
<!-- Two -->
<section id="two" class="wrapper style1 special">
	<div class="inner">
		<header>
		<h2><%
String[] str = NewClass.ans.split(",");
			//out.println(str.length);
for(inti=0;i<str.length;i++){
out.println(str[i]);
if(i != str.length-1) out.println("-->");
			}
NewClass.ans="";
			%></h2>
		</header>
		</div>
</section></br>
<section><imgsrc="images/map.jpeg"/></section>
<!-- Footer -->
	<footer id="footer">
	<div class="inner">
	<div class="flex">
	<div class="copyright">&copy; Effective Path Module</div>
	<ul class="icons">
	<li><a href="#" class="icon fa-facebook"><span
class="label">Facebook</span></a></li>
<li><a href="#" class="icon fa-twitter"><span class="label">Twitter</span></a></li>
<li><a href="#" class="icon fa-linkedin"><span class="label">linkedIn</span></a></li>
<li><a href="#" class="icon fa-pinterest-p"><span class="label">Pinterest</span></a></li>
<li><a href="#" class="icon fa-vimeo"><span class="label">Vimeo</span></a></li>
		</ul>
		</div>
		</div>
		</footer>
<!-- Scripts -->
			<scriptsrc="assets/js/jquery.min.js"></script>
			<scriptsrc="assets/js/skel.min.js"></script>
			<scriptsrc="assets/js/util.js"></script>
			<scriptsrc="assets/js/main.js"></script>
	</body>
</html>
```

3. DrawImage.java

```
packagecom.mrunal.jsp;
importjava.awt.*;
importjava.io.File;
importjavax.imageio.ImageIO;
importjavax.swing.JFrame;
public class DrawImage {
static Image image;
public static void main(String[] args) {
	String imageURL = "C:\\Users\\Lenovo\\Desktop\\banner.jpg";
image = Toolkit.getDefaultToolkit().getImage(imageURL);
	Frame frame = new Frame();
frame.add(new CustomPaintComponent());
intframeWidth = 1291;
intframeHeight = 648;
frame.setSize(frameWidth, frameHeight);
frame.setVisible(true);
try
	{
BufferedImage image = new BufferedImage(frameWidth, frameHeight,
BufferedImage.TYPE_INT_RGB);
	Graphics2D graphics2D = image.createGraphics();
frame.paint(graphics2D);
ImageIO.write(image,"jpg", new File("C:\\Users\\Lenovo\\Desktop\\map.jpg"));
	}
catch(Exception exception)
{ //code	}
 }
static class CustomPaintComponent extends Component {
public String game = "Mumbai,Thane,NaviMumbai,Lonavla,Dombivli,Bhiwandi";
public void paint(Graphics g) {
	Graphics2D g2d = (Graphics2D)g;
	String[] citys = {"Mumbai","Thane","Navi
Mumbai","Lonavla","Dombivli","Bhiwandi","Panvel","Rasayani","Karjat","Matheran","Badlapur"
,"Lavale","Hinjawadi","Kothrud","Viman
Nagar","PimpariChinchawad","Khopoli","Hadapsar","Pune","Lavasa","Aamby Valley City"};
String[] city = game.split(",");
float[] dash1 = { 5f, 0f, 5f };
	Stroke stroke = new BasicStroke(5f, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER,5.0f, dash1,5f);
```

```
int[] x =
{214,280,310,529,364,300,390,395,467,434,454,718,730,755,863,776,484,865,797,592,528}
;
int[] y =
{181,81,183,367,63,18,232,274,263,185,82,503,454,566,444,452,314,544,497,563,416};
g2d.drawImage(image, 0, 0, this);
g2d.setStroke(stroke);
int a = 214, b=181;
for(inti = 1;i <city.length;i++){
for(int j=0;j<21;j++){
if(city[i].equalsIgnoreCase(citys[j])){
g2d.drawLine(a, b, x[j], y[j]);

int temp1 = a;
                    a = x[i];
x[i] = temp1;

int temp2 = b;
                    b = y[i];
y[i] = temp2;
                } }} } }
```

4. Result.java

```
packagecom.mrunal.jsp;
// Java program to solve Traveling Salesman Problem
// using Branch and Bound.
importjava.util.*;

public class NewClass
{
staticint N = 21;
public static String ans;
public static String[] r_city;
      public static String[] city = {"Mumbai","Thane","Navi
Mumbai","Lonavla","Dombivli","Bhiwandi","Panvel","Rasayani","Karjat","Matheran","Badlapur"
,"Lavale","Hinjawadi","Kothrud","Viman
Nagar","PimpariChinchawad","Khopoli","Hadapsar","Pune","Lavasa","Aamby Valley City"};
```

```
staticintfinal_path[] = new int[N + 1];
        staticboolean visited[] = new boolean[N];
        public static intfinal_res;
        static void copyToFinal(intcurr_path[])
        {
                for (inti = 0; i< N; i++)
                        final_path[i] = curr_path[i];
                final_path[N] = curr_path[0];
        }
        staticintfirstMin(intadj[][], inti)
        {
                int min = Integer.MAX_VALUE;
                for (int k = 0; k < N; k++)
                        if (adj[i][k] < min &&i != k)
                                min = adj[i][k];
                return min;
        }
        staticintsecondMin(intadj[][], inti)
        {
                int first = Integer.MAX_VALUE, second = Integer.MAX_VALUE;
                for (int j=0; j<N; j++)
                {
                        if (i == j)
                                continue;

                        if (adj[i][j] <= first)
                        {
                                second = first;
                                first = adj[i][j];
                        }
                        else if (adj[i][j] <= second &&
                                        adj[i][j] != first)
                                second = adj[i][j];
                }
                return second;
        }

        static void TSPRec(intadj[][], intcurr_bound, intcurr_weight,
                                int level, intcurr_path[])
        {
                if (level == N)
                {
                        if (adj[curr_path[level - 1]][curr_path[0]] != 0)
                        {
                                intcurr_res = curr_weight +
                                                adj[curr_path[level-1]][curr_path[0]];
```

```
if (curr_res<final_res)
                                {
                                        copyToFinal(curr_path);
                                        final_res = curr_res;
                                }
                        }
                        return;
                }

                for (inti = 0; i< N; i++)
                {
                        if (adj[curr_path[level-1]][i] != 0 &&
                                        visited[i] == false)
                        {
                                int temp = curr_bound;
                                curr_weight += adj[curr_path[level - 1]][i];

                                if (level==1)
                                curr_bound -= ((firstMin(adj, curr_path[level - 1]) +
                                                        firstMin(adj, i))/2);
                                else
                                curr_bound -= ((secondMin(adj, curr_path[level - 1]) +
                                                        firstMin(adj, i))/2);

                                if (curr_bound + curr_weight<final_res)
                                {
                                        curr_path[level] = i;
                                        visited[i] = true;

                                        TSPRec(adj, curr_bound, curr_weight, level + 1,
                                                curr_path);
                                }

                                curr_weight -= adj[curr_path[level-1]][i];
                                curr_bound = temp;

                                Arrays.fill(visited,false);
                                for (int j = 0; j <= level - 1; j++)
                                        visited[curr_path[j]] = true;
                        }
                }
        }
```

```
static void TSP(intadj[][])
        {
                intcurr_path[] = new int[N + 1];
                intcurr_bound = 0;
                Arrays.fill(curr_path, -1);
                Arrays.fill(visited, false);

                for (inti = 0; i< N; i++)
                        curr_bound += (firstMin(adj, i) +
                                                secondMin(adj, i));

                curr_bound = (curr_bound==1)? curr_bound/2 + 1 :
                                                                curr_bound/2;

                visited[0] = true;
                curr_path[0] = 0;

                TSPRec(adj, curr_bound, 0, 1, curr_path);
        }

        public static int game(boolean[] cities)
        {
final_res = Integer.MAX_VALUE;
                TSP(matrix(cities));
cities[0] = false;
cities[1] = false;
cities[2] = false;
cities[3] = false;
cities[4] = false;
cities[5] = false;
cities[6] = false;
cities[7] = false;
cities[8] = false;
cities[9] = false;
cities[10] = false;
cities[11] = false;
cities[12] = false;
cities[13] = false;
cities[14] = false;
cities[15] = false;
cities[16] = false;
cities[17] = false;
cities[18] = false;
cities[19] = false;
cities[20] = false;
```

```
System.out.printf("Minimum cost : %d\n", final_res);
                    System.out.printf("Path Taken : ");
ans="";
                    for (inti = 0; i<= N; i++)
                    {
ans += r_city[final_path[i]]+",";
                    }
returnfinal_res;
          }
staticint[][] matrix(boolean[] cities){
int original[][] = {{0,22,21,83,40,35,34,53,62,11,52,151,134,148,154,136,69,157,148,151,105},
                    {23,0,24,86,25,10,34,56,65,11,37,154,137,150,157,139,72,160,151,190,108},
                    {21,22,0,62,26,33,13,32,41,11,38,130,113,126,133,114,48,136,127,130,84},
                    {83,85,62,0,80,92,50,38,35,11,66,67,51,64,71,52,35,74,66,67,24},
                    {40,26,25,77,0,18,29,47,57,11,22,145,129,142,149,130,64,152,142,182,99},
                    {38,13,35,91,19,0,42,61,71,11,30,159,143,156,163,144,77,166,162,196,120},
                    {35,36,13,50,29,41,0,15,29,11,39,118,101,115,121,103,36,124,115,155,72},
                    {53,54,32,37,50,65,15,0,19,11,51,105,89,102,109,90,25,112,102,140,64},
                    {62,64,41,33,59,61,30,19,0,11,34,103,86,99,106,87,19,109,99,139,55},
                    {11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11},
                    {53,37,41,86,22,30,38,51,34,11,0,134,117,131,137,118,50,140,131,171,86},
                    {151,152,129,65,147,162,117,106,102,11,134,0,17,12,28,27,79,29,20,44,55},
                    {134,136,113,50,131,146,101,87,86,11,117,14,0,16,23,10,63,26,17,53,61},
                    {148,149,126,62,144,159,114,102,99,11,131,12,17,0,16,19,76,15,7,50,88},
                    {158,159,137,72,154,170,124,113,109,11,141,28,25,16,0,18,93,9,11,66,98},
                    {136,137,115,52,131,146,102,90,87,11,117,25,10,20,18,0,64,21,15,63,76},
                    {68,71,47,18,64,78,35,23,19,11,50,87,69,84,91,72,0,93,84,124,40},
                    {159,160,138,74,156,171,127,114,112,11,143,28,27,15,8,22,85,0,10,65,99},
                    {148,150,127,65,146,160,115,103,100,11,131,20,18,6,10,16,84,9,0,57,89},
                    {188,189,167,104,185,200,154,143,140,11,171,44,53,50,68,64,115,69,57,0,86},
                    {107,108,85,24,104,119,73,61,58,11,90,58,61,63,95,77,37,98,89,86,0},
                    };

int counter = 0;
for(boolean k:cities) if(k==true) counter++;
       N = counter;
r_city = new String[counter];
int[][] distances = new int[counter][counter];
intindex_x=0,index_y=0;

for(inti=0;i<21;i++){
if(cities[i]==true){
for(int j=0;j<21;j++){
if(cities[j]==true){
```

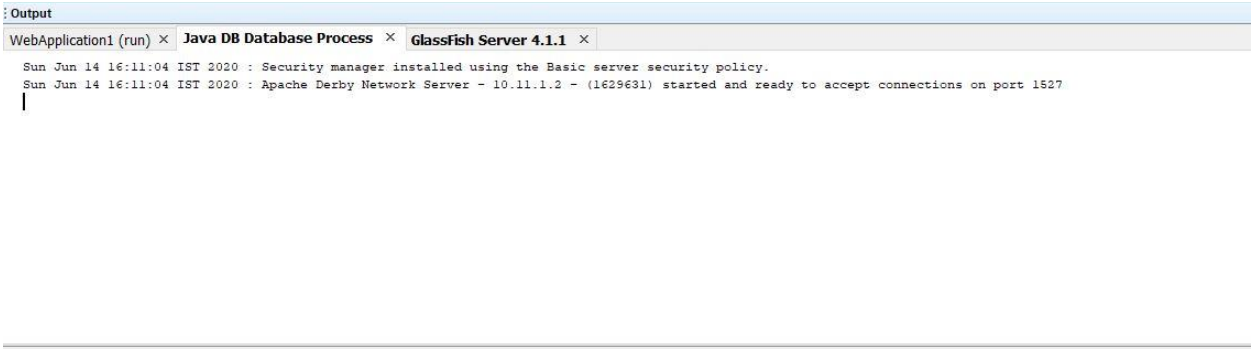
```
for(int j=0;j<21;j++){
if(cities[j]==true){
distances[index_x][index_y] = original[i][j];
index_y++;
                    }
                }
r_city[index_x] = city[i];
index_x++;
index_y=0;
            }
        }

for(inti=0;i<counter;i++){
for(int j=0;j<counter;j++){
System.out.print(distances[i][j]+" ");
            }
System.out.println();
        }
return distances;
        }
public static String distance(){
returnans;
        }
}
```

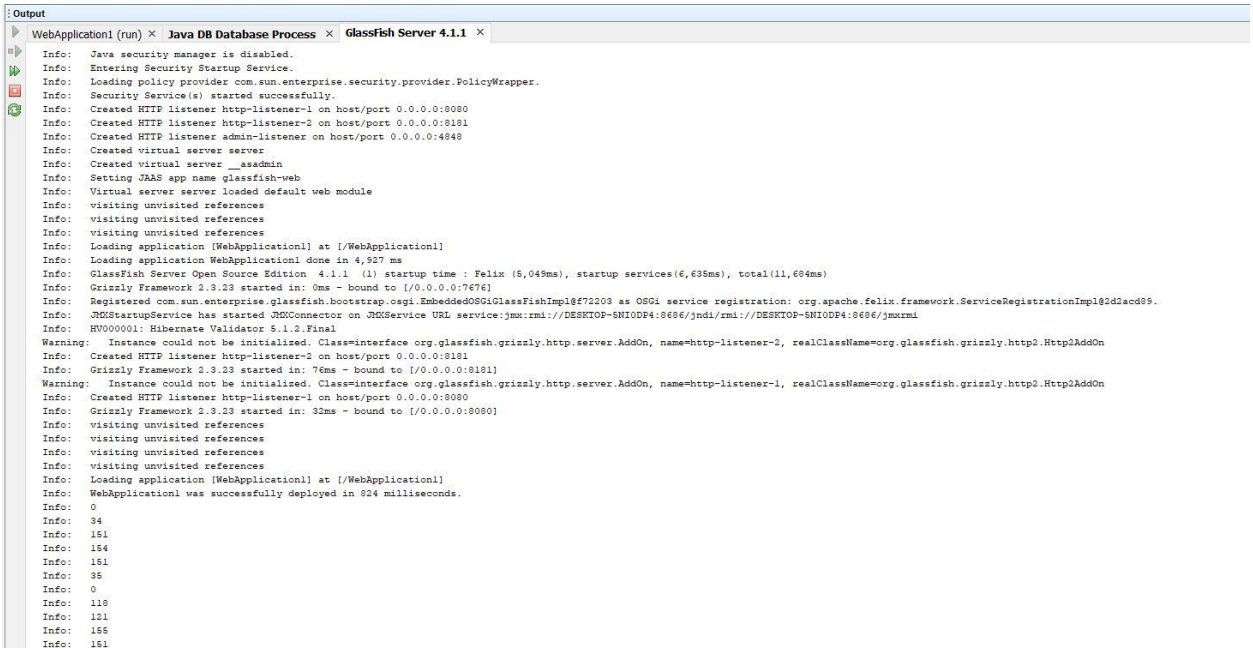
METHODOLOGY

STARTING A SERVER:

The IDE opens an output window that shows the progress of running the application.

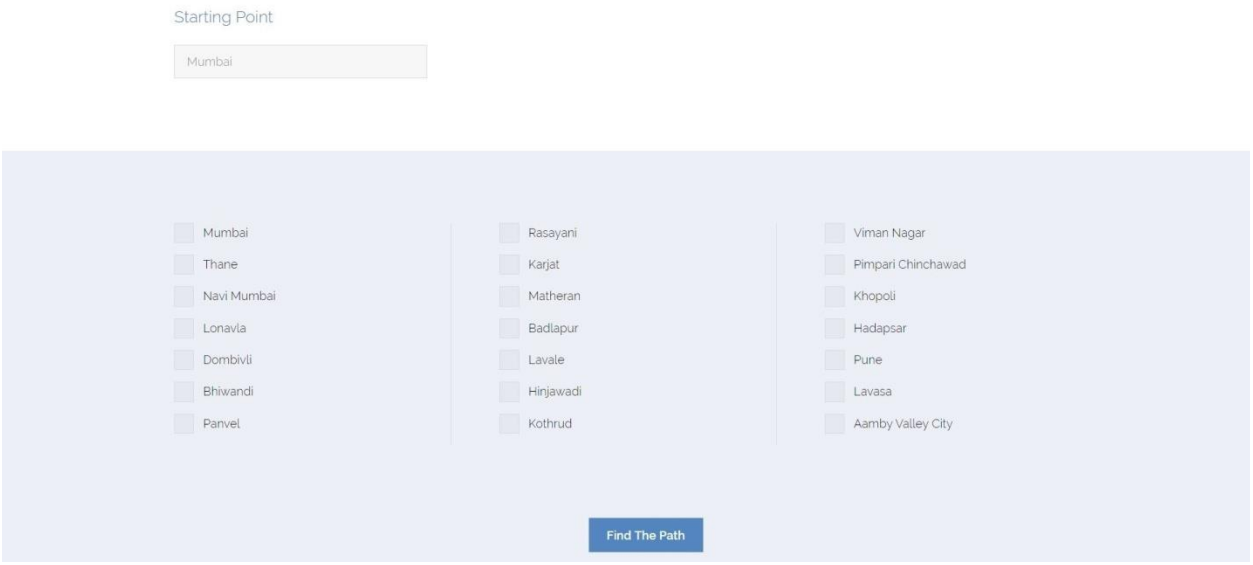
The Java DB Database Process CMD traces the responses and working of Server.

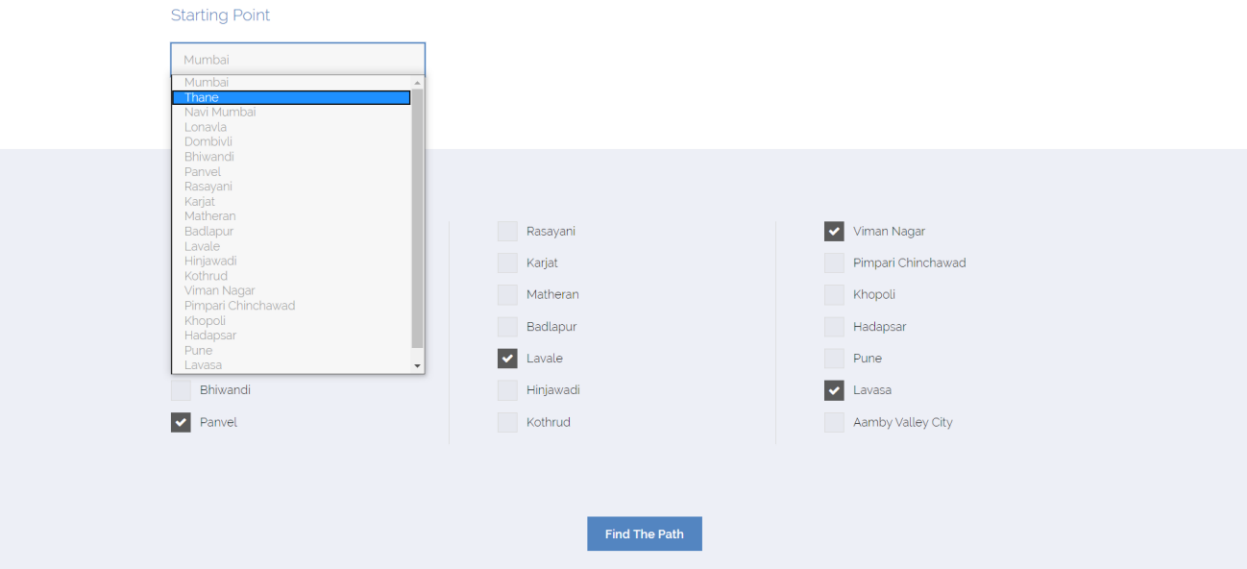




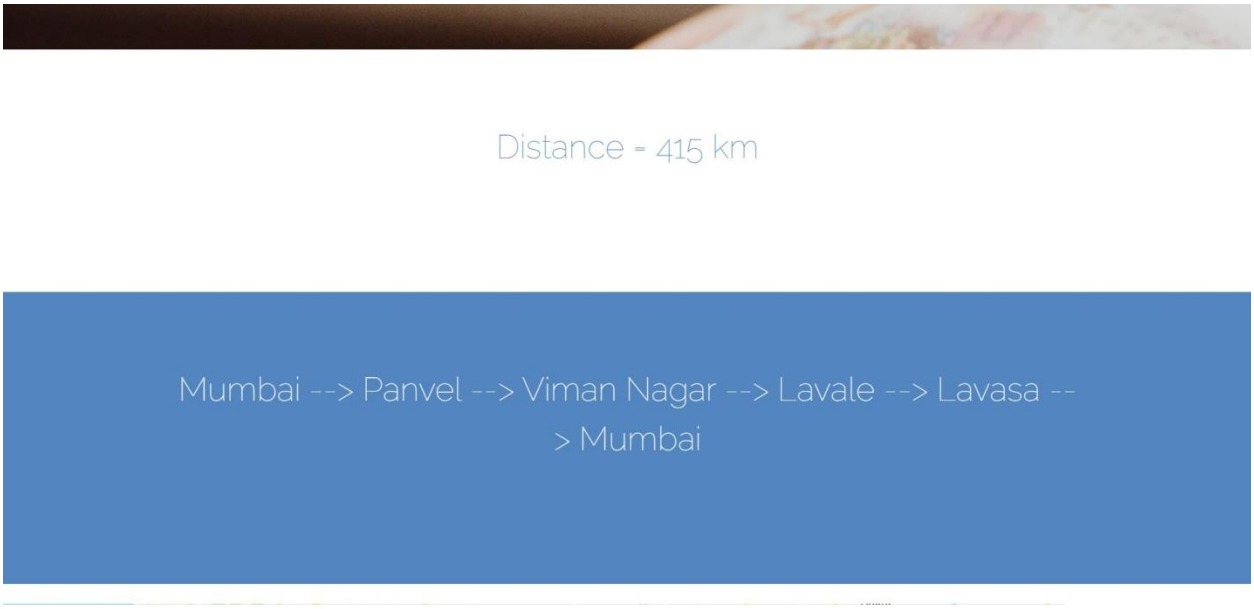
OPENING A REQUIRED ADDRESS IN BROWSER:

This link from the above image is open in browser and the home page of the program is open here you can select the starting point and also the cities you have to visit once.





Required shortest path as got:



CONCLUSION

In this project, the three selection methods viz. Roulette wheel selection, Rank selection and Tournament selection methods and the three crossover methods viz. uniform crossover, one point crossover and two point crossover methods are implemented. A city map is drawn to show the cities travelled by the salesman. It also shows the route taken to visit all the cities. The results obtained are analyzed and tabulated for Roulette wheel selection, Rank selection and Tournament selection using Uniform crossover, One point crossover and Two point crossover methods. The best selection method was found to be Roulette wheel selection based on the analysis of fitness cost. The minimum fitness cost found is 0.00123 which is for Two point crossover method.

FUTURE SCOPE

In this project, basic genetic algorithm is used to solve the Travelling Salesman Problem. As a future work, it can also be extended for hybrid genetic algorithm and parallel genetic algorithm. The performance between the two algorithms can also be analyzed based on the selection methods and crossover operators.