Laboratory Manual

For

Evolutionary Computing

(MF 202)

M.Tech (IT)

SEM II

June 2016

Faculty of Technology
Dharmsinh Desai University
Nadiad.

www.ddu.ac.in

Table of Contents

Sample experiment

1 AIM: Solve Travelling Salesman Problem using simple genetic algorithm The travelling sales man problem deals with the fact that a salesman travels between cities taking the path that is of minimum distance. Salesman has to travel each and every city without repetition.

2 TOOLS / APPARATUS: Python / Matlab / Jdk1.6 / Microsoft Visual Studio 2010

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

Here we have to travel n cities with minimum distance. For minimizing distance we use genetic algorithm. For this purpose we need to define various parameters of GA like representation scheme, population size, individuals, recombination methods, mutation type, crossover type, parent selection mechanism, survivor selection mechanism , etc.

3.2 Designing the Solution:

- Selection criteria 1
    - Filling of population is ***random***.
    - Evaluation of fitness : ***sum of distance.***
- Selection criteria 2
    - Selection of individual for crossover and mutation.
        - Crossover : ***Partial mapped crossover (permutation kind of problem).***
        - Mutation: ***swap***
- Selection criteria 3
    - Survivor Selection: ***Replace best with worst.***

3.3 Implementing the Solution

```
importjava.util.Random;
importjava.util.Scanner;

public class TVSP {
staticint n1,n2;
staticint N = 5;
static Random random = new Random();
public static void initialize(intpathlen[][],int path[][])
{
    inti,j,k;
    //obtaining pathlengths
    for(i=0;i<n1;i++)
    {
        for(j=0;j<n1;j++)
        {
            if(j<i)         //path length from a to b will be same as b to a
        {
                pathlen[i][j]=pathlen[j][i];
            }

            if(j==i)        // path length from a to a will be 0
            {
                pathlen[i][j]=0;
            }
        if(j>i)         // rest initialize
        {                                   pathlen[i][j]= random.nextInt(20);
                    }
                }
        }
    // display the path lengths
    `System.out.print("\n\tThe PATH LENGTHS ARE: \n" );
    for(i=0;i<n1;i++)
    {
        for(j=0;j<n1;j++)
        {
            System.out.print(pathlen[i][j]+" \t");
        }
        System.out.print("\n");
    }
```

```
// generating the population
for(i=0;i<n2;i++)
          {
                    for(j=0;j<n1;j++)
                    {
                              path[i][j]=random.nextInt(n1);

                              for(k=j-1;k>=0;k--)
                              {
                                        if(path[i][j]==path[i][k])  //checking to avoid repeatition
                              {
                                                  path[i][j] = random.nextInt(n1);
                                                  k=j;
                                        }
                              }
                    }
          }

}
// evaluating the fitness function or total distance
public static void evaluate(intpathlen[][],int path[][],intfx[])
{
          int sum =0,i,j,a,b;

//obtaing the sum of the path taken
          for(i=0;i<n2;i++)
          {
                    sum=0;
                    for(j=0;j<n1-1;j++)
                    {
                              a=path[i][j];
                              b=path[i][j+1];
                              sum=sum+pathlen[a][b];
}
                    fx[i]=sum;
}
//display the paths generated
          System.out.print("\n");
          System.out.print("\n\tPATH \t\tf(x) \n\n");
```

```
        for(i=0;i<n2;i++)
        {
                System.out.print("\t");
                for(j=0;j<n1;j++)
                {
                        System.out.print(path[i][j]);
                }
                System.out.print("\t\t"+fx[i]);
                System.out.print("\n");
}
}
//selecting the two points for cross over and then performing partial Crossover
public static void selection(intfx[],intpos[],intposmax[])
{
        int min1=fx[0],min2=fx[0],i,max1=fx[0],max2=fx[0];
        pos[0]=0;
        pos[1]=0;
        posmax[0]=0;
        posmax[1]=0;
        //calculating the minimum postion
        for(i=1;i<n2;i++)
        {
                if(fx[i]<min1)
                {
        min1=fx[i];
                        pos[0]=i;

                }
        }
        //calaculating the second minimum position
        for(i=1;i<n2;i++)
        {
                if(fx[i]<min2&&i!=pos[0])
                {
                        min2=fx[i];
                        pos[1]=i;
                }
        }
//calculating the max position
```

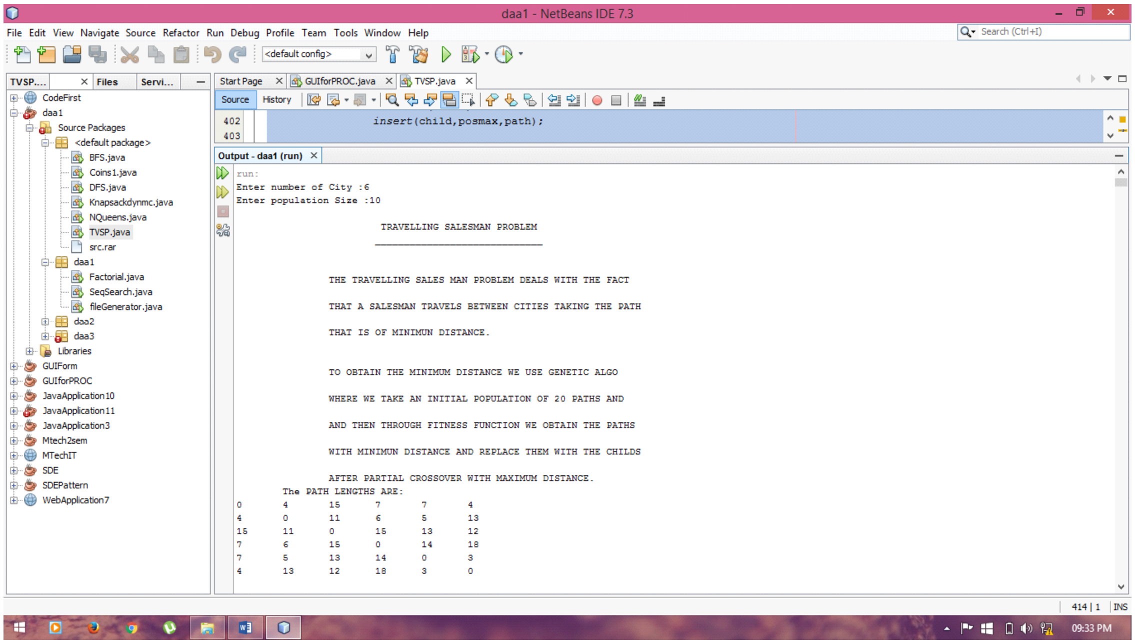
```
        for(i=1;i<n2;i++)
        {
                if(fx[i]>max1)
                {
                        max1=fx[i];
                        posmax[0]=i;
                }
        }
        //calculating the second max position

        for(i=1;i<n2;i++)
        {
        if(fx[i]>max2&&i!=posmax[0])
                {
                        max2=fx[i];
                        posmax[1]=i;
                }
        }
        System.out.print("\n\tFIRST MINIMUM="+min1+"
        \tPOSITION="+pos[0]+"\n\tSECOND MINIMUN="+min2+"
        \tPOSITION="+pos[1]+"\n\tFIRST MAXIMUM="+max1+"
        \tPOSITION="+posmax[0]+"\n\tSECOND MAXIMUM="+max2+"
        \tPOSITION="+posmax[1]+"\n");
}

//PERFORMING PARTIAL CROSSOVER
public static void crossover(intpos[],int path[][],int child[][])
{
        int crosspt1,crosspt2,j,i,temp,temp2;
        int temp1[][] = new int[2][n1];
        //TAKING 2 CROSS POINTS
        do
        {
                crosspt1=random.nextInt(n1-1);
        }while(crosspt1>2) ;
        do
        {
```

```
crosspt2=random.nextInt(n1-1);
}while(crosspt2<=3);

System.out.print("\n\n\t The CROSSOVER POINTS ARE : "+crosspt1+" , "+crosspt2);
System.out.print("\n\n\tTHE PATHS FOR CROSSOVER ARE");
System.out.print("\n\n\t\t");
for(j=0;j<n1;j++)
{
        child[0][j]=path[pos[0]][j];
        System.out.print(child[0][j]);
}
System.out.print("\n\t\t");
for(j=0;j<n1;j++)
{
        child[1][j]=path[pos[1]][j];
        System.out.print(child[1][j]);
}
intcnt=0;
//swapping the paths between two crosspoints
for(j=crosspt1+1;j<=crosspt2;j++)
{
        temp1[1][cnt]=child[0][j];
        temp1[0][cnt]=child[1][j];
        temp=child[0][j];
        child[0][j]=child[1][j];
        child[1][j]=temp;
        cnt++;
}

//performing partial crossover
intk,m;
for(m=0;m<2;m++)
{
        for(i=0;i<crosspt1+1;i++)   //taking the path before crosspoint
        {
                for(j=0;j<cnt;j++)   //comparing the path within crossover point
                {
```

```
if(child[m][i]==temp1[m][j]) //if found then
                                {
                                        if(m==0)   //for child 1
                                        {
                                temp2=temp1[1][j];   //take the path from child2 crossover
                                                for(k=0;k<n1;k++)
                                                {
                                if(child[m][k]==temp2)
                                //if still the path repeats then repeat the process again
                                { temp2=child[1][k];
                                  k=0;
                                }
                        }
                        child[m][i]=temp2;   //finally putting the value in child
                                        }
                                        else  //for child 2
                                        {
                                                temp2=temp1[0][j];
                                                for(k=0;k<n1;k++)
                                                {
                                                        if(child[m][k]==temp2)
                                                        {temp2=child[0][k];
                                                         k=0;

                                                        }
                                                }
                                                child[m][i]=temp2;
                                        }
                                }
                        }
                }
        }


for(m=0;m<2;m++)
        {
        for(i=crosspt2+1;i<n1;i++)   //now chehcking the path after the second cross point
                {
```

```
for(j=0;j<cnt;j++)   //comparing the path within crossover point
                        {
                                if(child[m][i]==temp1[m][j])  //if found then
                                {
                                        if(m==0)   //for child 1
                                        {
                                temp2=temp1[1][j];   //take the path from child2 crossover
                                for(k=0;k<n1;k++)
                                                {
                if(child[m][k]==temp2) //if still the path repeats then repeat the process again
                                {temp2=child[1][k];
                                 k=0;
                                 }
                        }
                                child[m][i]=temp2;  //finally assigning the value
                                        }
                                        else   //for child 2
                                        {
                                        temp2=temp1[0][j];
                                                for(k=0;k<cnt;k++)
                                                {
                                                        if(child[m][k]==temp2)
                                                        {temp2=child[0][k];
                                                         k=0;
                                                         }
                                                }
                                                child[m][i]=temp2;
                                        }
                                }
                        }
                }
        }
        //display AfTER  CROSSOVER
        System.out.print("\n\tAFTER CROSSOVER\n\t\t");

        for(j=0;j<n1;j++)
        {
                System.out.print(child[0][j]);
        }
        System.out.print("\n\t\t");
        for(j=0;j<n1;j++)
        {
                System.out.print(child[1][j]);
        }
}
        //insering the paths in population removing those having maximum populaiton
        public static void insert(int child[][],intposmax[],int path[][])
        {
        for(int j=0;j<n1;j++)
        {
                path[posmax[0]][j]=child[0][j];
                path[posmax[1]][j]=child[1][j];
        }
}
// performing mutation
public static void mutation(int child[][])
{
        intsel=random.nextInt(2);
        int pos1=random.nextInt(n1);
        int pos2=random.nextInt(n1);
        int temp=child[sel][pos1];
        child[sel][pos1]=child[sel][pos2];
        child[sel][pos2]=temp;
}
public static void main(String args[])
{
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of City :");
        n1 = sc.nextInt();
        intpathlen[][] = new int[n1][n1];
        System.out.print("Enter population Size :");
        n2 = sc.nextInt();
        int path[][] = new int[n2][n1];
        intfx[] = new int[n2];
        intpos[] = new int[2];
        intposmax[] = new int[2];
        int child[][] = new int[2][n1];
        System.out.print("\n\t\t\t TRAVELLING SALESMAN PROBLEM ");
```

```
          System.out.print("\n\t\t\t______________________________");
          System.out.print("\n\n\n\t\tTHE TRAVELLING SALES MAN PROBLEM DEALS
WITH THE FACT");
          System.out.print("\n\n\t\tTHAT A SALESMAN TRAVELS BETWEEN CITIES
TAKING THE PATH");
          System.out.print("\n\n\t\tTHAT IS OF MINIMUN DISTANCE.");
System.out.print("\n\n\n\t\tTO OBTAIN THE MINIMUM DISTANCE WE USE GENETIC
ALGO");
          System.out.print("\n\n\t\tWHERE WE TAKE AN INITIAL POPULATION OF 20
PATHS AND ");
          System.out.print("\n\n\t\tAND THEN THROUGH FITNESS FUNCTION WE OBTAIN
THE PATHS ");
          System.out.print("\n\n\t\tWITH MINIMUN DISTANCE AND REPLACE THEM WITH
THE CHILDS ");
          System.out.print("\n\n\t\tAFTER PARTIAL CROSSOVER WITH MAXIMUM
DISTANCE.");
initialize(pathlen,path);
          evaluate(pathlen,path,fx);
          selection(fx,pos,posmax);
          crossover(pos,path,child);
          mutation(child);
          insert(child,posmax,path);
for(inti=1;i<N;i++)
          {
                    evaluate(pathlen,path,fx);
                    selection(fx,pos,posmax);
                    crossover(pos,path,child);
                    mutation(child);
                    insert(child,posmax,path);
}
          evaluate(pathlen,path,fx);
          selection(fx,pos,posmax);
          crossover(pos,path,child);
          insert(child,posmax,path);
          evaluate(pathlen,path,fx);

}
}
```



```
                    insert(child,posmax,path);
```

```
run:
Enter number of City :6
Enter population Size :10

                         TRAVELLING SALESMAN PROBLEM
                        _____________________________


                 THE TRAVELLING SALES MAN PROBLEM DEALS WITH THE FACT

                 THAT A SALESMAN TRAVELS BETWEEN CITIES TAKING THE PATH

                 THAT IS OF MINIMUN DISTANCE.


                 TO OBTAIN THE MINIMUM DISTANCE WE USE GENETIC ALGO

                 WHERE WE TAKE AN INITIAL POPULATION OF 20 PATHS AND

                 AND THEN THROUGH FITNESS FUNCTION WE OBTAIN THE PATHS

                 WITH MINIMUN DISTANCE AND REPLACE THEM WITH THE CHILDS

                 AFTER PARTIAL CROSSOVER WITH MAXIMUM DISTANCE.
        The PATH LENGTHS ARE:
0       4       15      7       7       4
4       0       11      6       5       13
15      11      0       15      13      12
7       6       15      0       14      18
7       5       13      14      0       3
4       13      12      18      3       0
```

output

```
Enter number of City :6
Enter population Size :10
The PATH LENGTHS ARE:
0       5       5       19      18      3
5       0       0       12      3       17
5       0       0       10      13      2
19      12      10      0       4       6
18      3       13      4       0       1
3       17      2       6       1       0

        PATH            f(x)

        140523                  36
        423501                  37
        345120                  27
        325014                  23
```

```
342105              25
250134              26
013425              36
352041              34
345120              27
130245              50

FIRST MINIMUM=23          POSITION=3
SECOND MINIMUN=25         POSITION=4
FIRST MAXIMUM=50          POSITION=9
SECOND MAXIMUM=37         POSITION=1
The CROSSOVER POINTS ARE : 2 , 4
THE PATHS FOR CROSSOVER ARE
        325014
        342105
AFTER CROSSOVER
        325104
        342015
PATH            f(x)

140523              36
342015              44
345120              27
325014              23
342105              25
250134              26
013425              36
352041              34
345120              27
325104              52

FIRST MINIMUM=23          POSITION=3
SECOND MINIMUN=25         POSITION=4
FIRST MAXIMUM=52          POSITION=9
SECOND MAXIMUM=44         POSITION=1

 The CROSSOVER POINTS ARE : 0 , 4
```

```
THE PATHS FOR CROSSOVER ARE

                325014
                342105
        AFTER CROSSOVER
                342105
                325014

        PATH            f(x)

        140523                  36
        345012                  13
        345120                  27
        325014                  23
        342105                  25
        250134                  26
        013425                  36
        352041                  34
        345120                  27
        342105                  25

        FIRST MINIMUM=13                POSITION=1
        SECOND MINIMUN=23               POSITION=3
        FIRST MAXIMUM=36                POSITION=0
        SECOND MAXIMUM=36               POSITION=0


         The CROSSOVER POINTS ARE : 2 , 4

        THE PATHS FOR CROSSOVER ARE

                345012
                325014

        AFTER CROSSOVER
                345012
                325014
```

```
PATH          f(x)

        025314              28
        345012              13
        345120              27
        325014              23
        342105              25
        250134              26
        013425              36
        352041              34
        345120              27
        342105              25

        FIRST MINIMUM=13        POSITION=1
        SECOND MINIMUN=23       POSITION=3
        FIRST MAXIMUM=36        POSITION=6
        SECOND MAXIMUM=34       POSITION=7


         The CROSSOVER POINTS ARE : 2 , 4

        THE PATHS FOR CROSSOVER ARE
                345012
                325014
        AFTER CROSSOVER
                345012
                325014
        PATH                f(x)
        025314              28
        345012              13
        345120              27
        325014              23
        342105              25
        250134              26
        342015              44
        325014              23
        345120              27
        342105              25
```

```
FIRST MINIMUM=13          POSITION=1
SECOND MINIMUN=23         POSITION=3
FIRST MAXIMUM=44          POSITION=6
SECOND MAXIMUM=28         POSITION=0

 The CROSSOVER POINTS ARE : 1 , 4
THE PATHS FOR CROSSOVER ARE
        345012
        325014
AFTER CROSSOVER
        345012
        325014
PATH                f(x)
325014              23
345012              13
345120              27
325014              23
342105              25
250134              26
345012              13
325014              23
345120              27
342105              25

FIRST MINIMUM=13          POSITION=1
SECOND MINIMUN=13         POSITION=6
FIRST MAXIMUM=27          POSITION=2
SECOND MAXIMUM=27         POSITION=8

The CROSSOVER POINTS ARE : 1 , 4
THE PATHS FOR CROSSOVER ARE
        345012
        345012
```

```
AFTER CROSSOVER
            345012
            345012

PATH            f(x)

325014                  23
345012                  13
345012                  13
325014                  23
342105                  25
250134                  26
345012                  13
325014                  23
345012                  13
342105                  25
```

EXPERIMENT-1

Aim: Study about Matlab / Python

Tools/ Apparatus: Matlab / Python.

Procedure:

- Explore the matlab / python environment, its features and coding standards.
- Perform some basic matrix operations on it.
- Prepare the report about Matlab / Python.

EXPERIMENT-2

Aim: Implement following programs in Python

- Display the Fibonacci series by taking input from user

- Display factorial of a number entered by user

Tools/ Apparatus: Python.

Procedure:

- Identify the use of variables and loops in python and apply it for implementing given programs.

EXPERIMENT-3

Aim: Solve AND / OR problem using single layer perceptron
Tools/ Apparatus: Python
Procedure:

- Identify whether the problem is linearly separable or not
- Prepare the truth table for it
- Apply perceptron learning algorithm for finding errors and learning the network for AND / OR problem.

EXPERIMENT-4

Aim: Study and install lilgp / ECJ toolkit.

Tools/Apparatus:

Procedure:

- Get familiar with the toolkit and its environment
- Download the lab manual for installation of lilgp / ECJ toolkit
- Mention all the steps you have performed for installation of the toolkit

EXPERIMENT-5

Aim: Execute a sample code in lilgp / ECJ toolkit based on Genetic algorithm / Genetic programming.

Tools/Apparatus: Lilgp / ECJ.

Procedure:

- Identify the sample problems based on Genetic Algorithm / Genetic Programming
- Thoroughly understand the problem.
- Identify the problem, candidate solution, fitness function, population size, criteria for parent selection, survivor selection, crossover operator, mutation and termination condition.
- Run the sample code and conclude its result.

EXPERIMENT-6

Aim: Solve Travelling Salesman Problem using simple genetic algorithm

Tools/Apparatus: JDK1.6 / Microsoft Visual Studio 10 / python.
Procedure:

- •Select appropriate representation scheme.
- •Take any four cities (For eg:- Baroda, Anand, Nadiad, Ahmedabad)
- •Create a matrix containing distance between two cities
- •Eg:-
- •Initialize the population with candidate solutions.
- •Apply selection criteria for parent selection, survivor selection.
- •Identify crossover operator and mutation.
- •Identify the fitness function.
- •Specify the termination criteria.
- •Implement the program and display the results

EXPERIMENT-7

Aim: Implement Stochastic / Batch gradient descent algorithm
Tools/Apparatus: Python
Procedure:

- Take any maximization or minimization problem
- Apply stochastic as well as batch gradient descent algorithm to solve it.

EXPERIMENT-8

Aim: Implement alpha LMS and Mu – LMS algorithm

Tools/Apparatus: Python
Procedure:

- Take any maximization or minimization problem
- Apply alpha LMS and Mu LMS algorithm to solve it.

EXPERIMENT-9

Aim: Solve XOR problem using back propagation neural network.

Tools/Apparatus: Python
Procedure:

- Draw the truth table of XOR
- Identify the type of problem (linear / non linear)
- Draw the neural network architecture to solve it
- Implement back propagation algorithm to solve it.

EXPERIMENT-10

Aim: Study and simulate any algorithm based on collective intelligence (Eg: ACO,PSO,BCO,CSO, etc)

Tools/Apparatus: Weka / R

Procedure:

- Select any collective intelligence based algorithm.
- Identify the application to be solved using that algorithm
- Simulate the algorithm to solve the problem.

EXPERIMENT-11

Aim: Implement RBFN for data sample classification

Tools/Apparatus: Rapid Miner / Weka / R / Python.
Procedure:

- Down load the data set for classification from given link https://archive.ics.uci.edu/ml/datasets.html
- Identify the attributes and class labels from the dataset.
- Draw RBFN architecture to classify the data set
- Simulate RBFN to classify the data sample

EXPERIMENT-12

Aim: Implement MLBPNN for digit recognition.

Tools/Apparatus: Rapid Miner / Weka / R / Python.
Procedure:

- Download the samples of english digits from the internet
- Draw the appropriate Multilayer Back propagation neural network for it
- Implement the code for digit recognition.

References

Reference books:

- Neural Network
    - Simon Haykin.

- Neural Network
    - Satish Kumar

- Genetic Algorithms in search, optimization and machine learning
    - David E. Goldberg

- Programming Collective Intelligence
    - Toby Segaram