Lab-5 Optimization using Genetic Algorithm

Software: MATLAB, optimization Toolbox

Problem statement: A Traveling Sales Person (TSP) needs to visit 50 different locations. The salesman must not revisit the same location more than once and he must return to the starting location after visited all other location. In order to do so, the salesman want to find out the shortest route to visit all 50 locations. Thus, in this lab, the problem will be solve using generic algorithm.

Objective: To find the shortest distance of a salesman traveling 50 different location subject to the condition that the person should not revisit the same location and return to the stating location.

Genetic Algorithm (GA)

GA is a method that mimics the process of natural selection and uses it for optimisation purpose. This method can be very useful in finding solutions to a problem that has many variables. This method can be very effective finding solution approximate to the actual.

The technique used in GA are designed to simulate processes in natural system necessary for evolution. The principle of GA is similar to Charles Darwin of "survival of the fittest'. In nature, competition among the individuals for resource often down to where the fittest individuals dominating over the weaker ones.

Task:

(i) Prepare the data set of different location from the file as uploaded in Moodle "Data Set GA"
(ii) Design for experiment in Genetic Algorithm (Generate initial population, Selection Operator, Crossover Operator, Mutation Operator) using steps 1 to 6.
(iii) Demonstrate the effectiveness of optimization through performance comparison for multiple combinations of mutation and crossover types.
(iv) Demonstrate the relationship/effects of P_m and P_c on the optimization performance.

Laboratory Activities:

Assessment rubrics: Marking rubrics to evaluate Lab-5 Optimization using Genetic Algorithm Note: Maximum mark provided for this Experiment is 15.

Assessment rubrics: Marking rubrics to evaluate Lab-5 Optimization using Genetic Algorithm. Note: Maximum mark provided for this Experiment is 15.

Preliminary studies (Max. Mark 2): Preparation of Data sets such as Inputs/ Demonstration and explanation is poor (40%)/ acceptable (60%)/Excellent (100%)	Usage of GA method (Max.Mark: 2). Display the level/ ability in using and demonstrating the effectiveness of data handling; Not proficient (20%)/ Novice level (40%)/ Intermediate level (60%/commendable (100%)	Design for experiments (GA Parameters) (Max. Mark: 3) Methods and agruments to support decisions: presented poorly (40% / Vague (60%) / Clearly 100%)	Simulation results (with appropriate figures) Max. Mark (3): Little information on results (20%) Analysis of results (50%) Critical assessment of results (100%)	Comparative study on the performance: Max. Mark (5). Achievements made are poor (20%), Acceptable (50%) and Commendable/examplary (100%)

Refer EXCEL SHEET attached in Moodle.

Implementation

Based on Natural Selection, step:

1. Generate initial population
2. Selection Operator
3. Crossover Operator
4. Mutation Operator

Step-1 Chromosome Design

The parameters of the GA are set constant,

Crossover probability, PC = 0.5

Mutation probability, PM = 0.1

The number of chromosomes (traveling route), chrom = 500

The number of gene (locations), gene = 50

Since that the salesman needs to reach back the starting point after visiting all other location, the number of gene is equal to 50+1, indicating both the start and the end will be the same value to ensure they are the same location.

The MATLAB code for generating chromosomes are as the following:

```
Generate population of chromosomes

chromgen=[];

for k=1:chrom
    num=gene;
    city_array=1:num;
    xxx=[];
    for n=1:num
        a=rand(1);
        for i=1:num
            if a<i/num
                xxx=[xxx city_array(i)];
                break
            end
        end
        city_array(i)=[];
        num=num-1;
    end
    chromgen(k,:)=xxx;
```

```
end

chromgen;
rout=[chromgen chromgen(:,1)];
```

Step-2 Fitness Function Design

In this lab, the objective is to find the shortest route. For the shortest route to be chosen, its fitness needs to be high. There are about 500 different routes in total. Every route will have its own distance.

```
function totaldist=evaldist(chromosome,city_distance,chrom,gene)
totaldist=[];
for k=1:chrom
      path=0;
    for i=1:gene
        path = path + city_distance(chromosome(k,i),chromosome(k,(i+1)));
    end
    totaldist(k)=path;
end
```

The following formula can be used to find fitness for each route:

$$Fitness = \frac{1}{Total\ Distance}$$

From the formula above, it is clear that when the total distance is shorter, the fitness becomes bigger. Hence, the formula suits the requirement.

Step-3 Selection

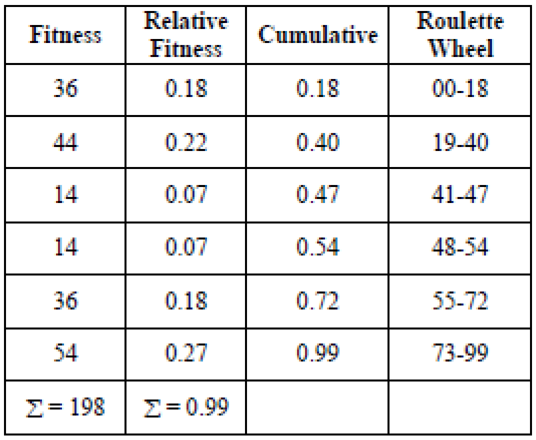
In the natural world, it has been said that the only the fittest will survive. The same concept is used here where the higher the fitness would most likely to survive. In this case, it will be chosen as the wanted gene for the used of crossover.

The method used for selection is roulette wheel. For the fitness to be used, it needs to be transform to relative fitness first, and then cumulate it to get the range for each section in the wheel.

The relative fitness formula:

$$Relative\ Fitness = \frac{Fitness_{i=1}}{\sum Fitness} x\ 100$$

The example for the cumulative fitness can be seen as below:



Fitness	Relative Fitness	Cumulative	Roulette Wheel
36	0.18	0.18	00-18
44	0.22	0.40	19-40
14	0.07	0.47	41-47
14	0.07	0.54	48-54
36	0.18	0.72	55-72
54	0.27	0.99	73-99
Σ = 198	Σ = 0.99		

A number is generated randomly. When a number appears, that particular chromosome which lies in the range of the roulette wheel of the generated number will then be chosen.

Step 4 Crossover Operator

Crossover Operator is the prime distinguished factor from other optimisation technique. In crossover, two chromosomes will be selected as parents to form offspring chromosomes.

In this lab, the crossover operators are custom-made. The characteristic of this custom-made crossover operator is that the parents swapped the sequence of their value instead of swapping their actual value. The first and last value will remain unchanged so that the start and end of the location will be at the same location.

Crossover 1

In this method, a point is generated to separate the chromosome into 2 sections, left and right. The right section is selected, excluding the last value, for both the parents. The sequence of the value is found for both parents, then swapped them.

Crossover 2

In this method, a point is generated to separate the chromosome into 2 sections, left and right. The left section is selected, excluding the last value, for both the parents. The sequence of the value is found for both parents, then swapped them.

Crossover 3

In this method, two points in generated to separate the chromosome into 3 sections, left, middle and right. The middle section is selected, excluding the last value, for both the parents. The sequence of the value is found for both parents, then swapped them.

- The MATLAB code for all 3 methods for crossover is attached to the appendix.

Step 5 Mutation Operator

Mutation is a rare occurrence, just as the same as in nature. It represent a change in the gene that may improve the system or reduce the system.

Mutation 1

In this method, two points is generated to pin point two points in the offspring chromosome. The two points then exchange with each other.

Mutation 2

In this method, two points is generated in order to create a section in the offspring chromosome. Inside the section, the array will move to the right by one space, the value at the most right will then move to the front of the section.

Step 6 The MATLAB code

M-file for the MATLAB codes of the Genetic Algorithm:

```
% Genetic Algorithm
% ========================================================================
clc; clear; close all;

% read the data from excel
citydata = xlsread('Data Set for GA Assignment/pr76.csv');
citydata = citydata(1:50,:); % take only the first 50 data
x = citydata(:,1); % X coordinate
y = citydata(:,2); % Y coordinate

% number of cities to be visited by the salesman
num = 50;

% show the plot of the city location
figure;hold on;
plot(x,y,'.r','markersize',25)

% Assign a numerical name for each city
for i = 1:num;
    text(x(i)+0.02,y(i),sprintf('%g',i));
end

% plot all available roads between cities
for i = 1:num
    for j = 1:num
        plot([x(i) x(j)],[y(i) y(j)])
    end
end
title('All Possible Route');

% finding distance between cities
citydist = dist(citydata');

% setting GA parameters
chrom = 500; % number of different combination(chromosomes)
gene = num; % number of cities(gene)
PC = 0.5; % probability of crossover
PM = 0.1; % probability of mutation
generation = 2000; % number of generation

disp('')
disp('GA parameters')
disp('==================================')
fprintf(1,' chrom = %.0f Size of the chromosome population\n',chrom);
fprintf(1,' Pc = %.1f Crossover probability\n',PC);
fprintf(1,' Pm = %.3f Mutation probability\n',PM);
fprintf(1,' generation = %.0f Number of generations\n',generation);
disp(' ')

% Generate population of chromosomes
```

```
chromgen=[];

for k=1:chrom
    num=gene;
    city_array=1:num;
    xxx=[];
    for n=1:num
        a=rand(1);
        for i=1:num
            if a<i/num
                xxx=[xxx city_array(i)];
                break
            end
        end
        city_array(i)=[];
        num=num-1;
    end
    chromgen(k,:)=xxx;
end

chromgen;
rout=[chromgen chromgen(:,1)];

% Calculate the chromosome fitness
totaldist = evaldist(rout,citydist,chrom,gene);
best = min(totaldist);

% Plot the best route found in the initial chromosome population
[a b] = min(totaldist);
figure('name','The best rout found in the initial population');
plot(x,y,'.r','markersize',25)
title(['The total distance: ',num2str(a)]);
hold on

for i = 1:gene;
    text(x(i)+0.02,y(i),sprintf('%g',i));
    plot([x(rout(b,i)) x(rout(b,(i+1)))],[y(rout(b,i)) y(rout(b,(i+1)))]);
end

%=========================================================================
% RUN GENETIC ALGORITHM
%=========================================================================
tic;
for m = 1:(generation)

%=========================================================================
    % Selection

%=========================================================================
    fitness = (1./totaldist)';
    relativefitness = (fitness./sum(fitness))*100;
    cumfit=cumsum(fitness);

    % Roulette wheel selection
    numsel=round(chrom); % The number of chromosomes to be selected for
reproduction
    cumfit=repmat(cumsum(fitness),1,numsel); % Replicating the cumulative
fitness array
    chance=repmat(rand(1,numsel),chrom,1)*cumfit(chrom,1);
    [selind,j]=find(chance < cumfit & chance >=
[zeros(1,numsel);cumfit(1:chrom-1,:)]);
    selected=(rout(selind,:));


%==========================================================================
    % CrossOver

%==========================================================================
    crossed = selected;
    for n = 1:2:chrom
        if (rand<PC)
            crossed(n:n+1,:) = crossover3(selected(n,:),selected(n+1,:));
        else
        end
    end

%==========================================================================
    % Mutation

%==========================================================================
    mutated = crossed;
    for n = 1:chrom
        if (rand<PM)
            mutated(n,:) = mutation2(crossed(n,:));
        else
        end
    end
    rout = mutated;

    %calculate total distance
    totaldist = evaldist(rout,citydist,chrom,gene);
    best = [best min(totaldist)];
end
[a b] = min(totaldist);
Best_Fitness_Value = a

% Plotting the best rout found in the current population
figure('name','The best rout found in the current population');
plot(x,y,'.r','markersize',25)
title(['Generation # ',num2str(generation),' The total
distance:',num2str(a)]);
hold on

for i=1:gene;
    text(x(i)+0.02,y(i),sprintf('%g',i));
    plot([x(rout(b,i)) x(rout(b,(i+1)))],[y(rout(b,i)) y(rout(b,(i+1)))])
end
toc;

%==========================================================================
% Performance Graph
%=========================================================================
disp(' ')
figure('name','Performance graph');
plot(0:generation,best);
legend('Best','Average',0);
title(['Pc = ',num2str(PC),', Pm = ',num2str(PM)]);
xlabel('Generations');
ylabel('Distance')
```

MATLAB File for Crossover 1:

```
function crossed = crossover1(array1,array2)

crossed = [array1;array2];
gene = length(array1);
crossline = ceil(rand*gene-4); % the separation point
while crossline<1
    crossline = ceil(rand*gene-4);
end

% choose the right side of the separation point
crossarray(1,:) = array1(crossline+1:end-1); % choose the right side of the
separation point
crossarray(2,:) = array2(crossline+1:end-1);
cross_size = length(crossarray); %length of the array
sequence = zeros(2,cross_size);

for i = 1:2
    crossarrayloop = crossarray(i,:);
    sequenceloop =  sequence(i,:);
    for j = 1:cross_size
        % each value mark based on its value in ascending order
% the marked value will show the sequence of the array
minimum = min(crossarrayloop);
min_index = find(crossarrayloop==minimum);
sequenceloop(min_index) = j;
crossarrayloop(min_index) = NaN;
    end
    sequence(i,:) = sequenceloop;
end

% the sequence of the array is swapped
cross_sequence = [sequence(2,:);sequence(1,:)];
crossing = zeros(1,cross_size);
for i=1:2
    sequenceloop = cross_sequence(i,:);
    crosslooparray = crossarray(i,:);
    for j= 1:cross_size
        %the array rearranged based on the swapped sequence
        crosslooparray = sort(crosslooparray);
        crossing(j) = crosslooparray(sequenceloop(j));
    end
    crossed(i,(crossline+1:end-1))=crossing;
end
```

```
end
```

MATLAB File for Crossover 2:

```
%sequence swapping to the left

function crossed = crossover2(array1,array2)
crossed = [array1;array2];
gene = length(array1);
crossline = ceil(rand*gene-4); % the separation point

while crossline<3
    crossline = ceil(rand*gene-4);
end

% choose the left side of the separation point
crossarray(1,:) = array1(2:crossline); % choose the left side of the
separation point
crossarray(2,:) = array2(2:crossline);
cross_size = length(crossarray); %length of the array
sequence = zeros(2,cross_size);

for i = 1:2
    crossarrayloop = crossarray(i,:);
    sequenceloop =  sequence(i,:);
    for j = 1:cross_size
        % each value mark based on its value in ascending order
        % the marked value will show the sequence of the array
        minimum = min(crossarrayloop);
        min_index = find(crossarrayloop==minimum);
        sequenceloop(min_index) = j;
        crossarrayloop(min_index) = NaN;
    end
    sequence(i,:) = sequenceloop;
end

cross_sequence = [sequence(2,:);sequence(1,:)];
crossing = zeros(1,cross_size);

for i=1:2
    % the sequence of the array is swapped
    sequenceloop = cross_sequence(i,:);
    crossloparray = crossarray(i,:);
    for j= 1:cross_size
        %the array rearranged based on the swapped sequence
        crossloparray = sort(crossloparray);
        crossing(j) = crossloparray(sequenceloop(j));
    end
    crossed(i,(2:crossline))=crossing;
end
end
```

MATLAB File for Crossover 3:

```
%sequence swapping to the middle
```

```
function crossed = crossover3(array1,array2)
crossed = [array1;array2];
gene = length(array1);
point1 = ceil(rand*gene-4); % the separation point1

while point1<3
    point1 = ceil(rand*gene-4);
end
point2 = ceil(rand*gene-2); % the separation point2

while point2<point1+2 || point2>gene
    point2 = ceil(rand*gene-2);
end

% choose the right side of the separation point
crossarray(1,:) = array1(point1:point2); % choose the array between the two
separation points
crossarray(2,:) = array2(point1:point2);
cross_size = length(crossarray); %length of the array
sequence = zeros(2,cross_size);

for i = 1:2
    crossarrayloop = crossarray(i,:);
    sequenceloop = sequence(i,:);
    for j = 1:cross_size
        % each value mark based on its value in ascending order
        % the marked value will show the sequence of the array
        minimum = min(crossarrayloop);
        min_index = find(crossarrayloop==minimum);
        sequenceloop(min_index) = j;
        crossarrayloop(min_index) = NaN;
    end
    sequence(i,:) = sequenceloop;
end

% the sequence of the array is swapped
cross_sequence = [sequence(2,:);sequence(1,:)];
crossing = zeros(1,cross_size);

for i=1:2
    sequenceloop = cross_sequence(i,:);
    crossloooparray = crossarray(i,:);
    for j= 1:cross_size
        %the array rearranged based on the swapped sequence
        crossloooparray = sort(crossloooparray);
        crossing(j) = crossloooparray(sequenceloop(j));
    end
    crossed(i,(point1:point2))=crossing;
end
end
```

MATLAB File for Mutation 1:

```
%swap between two random point

function mutated = mutation1(array)
mutate = array(2:end-1); %prevent changing start point and end point
mutateref = mutate;
point1 = ceil(rand*length(mutate));
point2 = ceil(rand*length(mutate));

if point1 == point2
    point2 = ceil(rand*length(mutate));
end

%swap between two random point
mutate(point1) = mutateref(point2);
mutate(point2) = mutateref(point1);
mutated = [array(1) mutate array(end)];
end
```

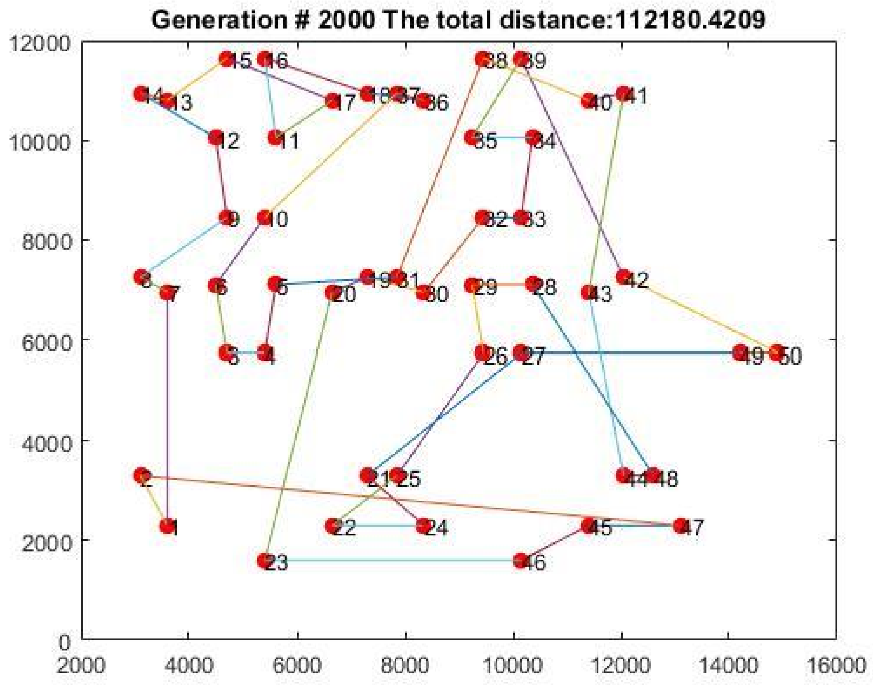
MATLAB File for Mutation 2:

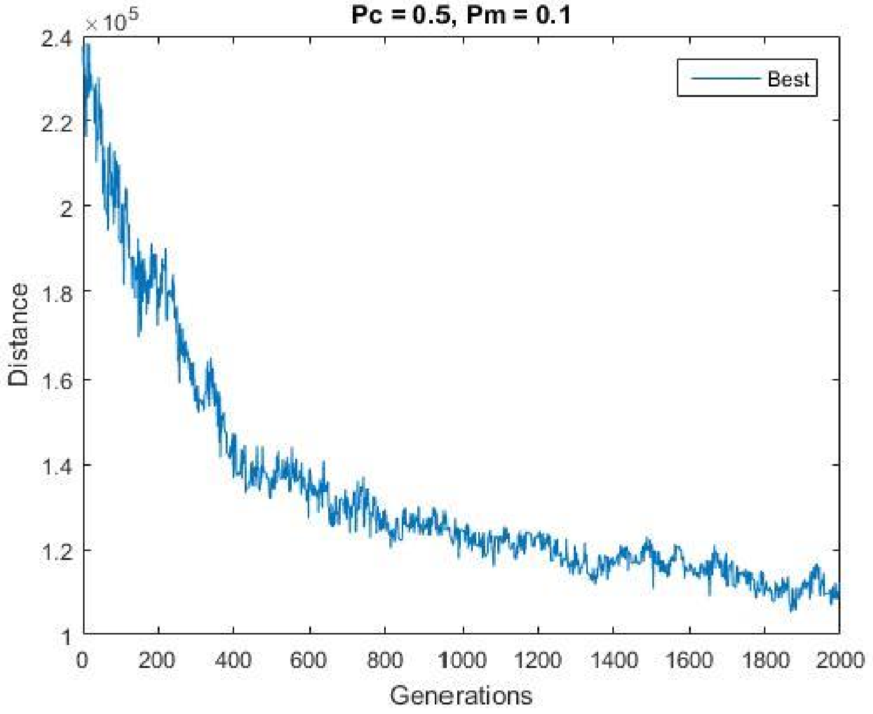
```
%shift a random section to the right by one
%last number of that section move to the front of the section

function mutated = mutation2(array)
mutate = array(2:end-1); %prevent changing start point and end point

% front of the random section
point1 = ceil(rand*length(mutate)-1);
if point1<1
    point1=1;
end

% back of the random section
point2 = ceil(rand*length(mutate));
while point2<=point1
    point2 = ceil(rand*length(mutate));
end
mutating = mutate(point1:point2); % extract the random section from the array
mutating = [mutating(end) mutating(1:end-1)]; % shift the section to the
right by one
mutated = [array(1) mutate(1:point1-1) mutating mutate(point2+1:end)
array(end)];

end
```

Step-7 Sample results

Comparison for Performance of Different Combination of Crossover and Mutation Operator

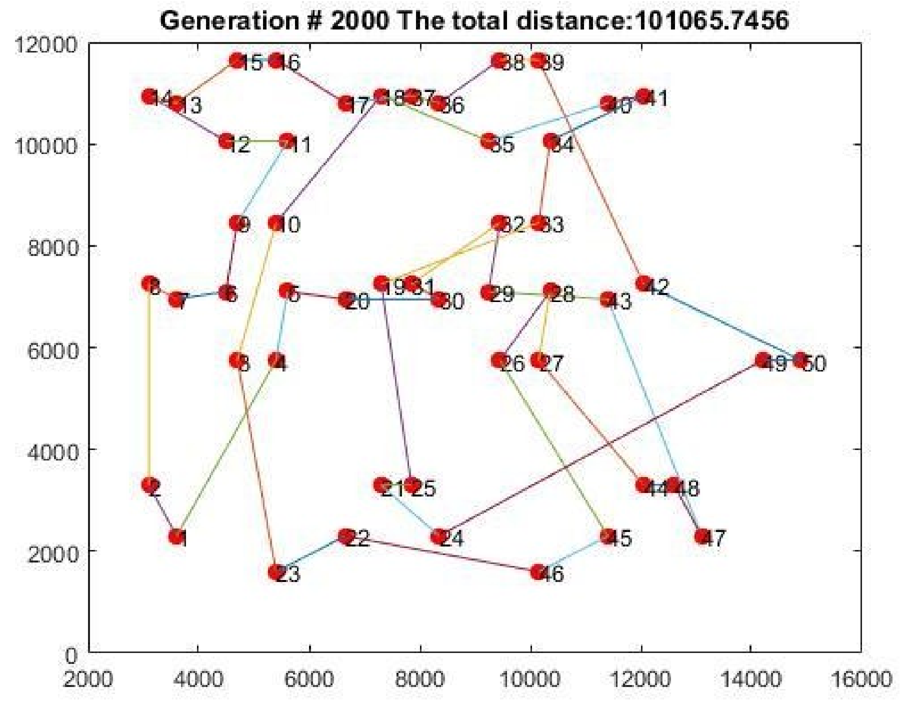
Crossover 1, Mutation 1

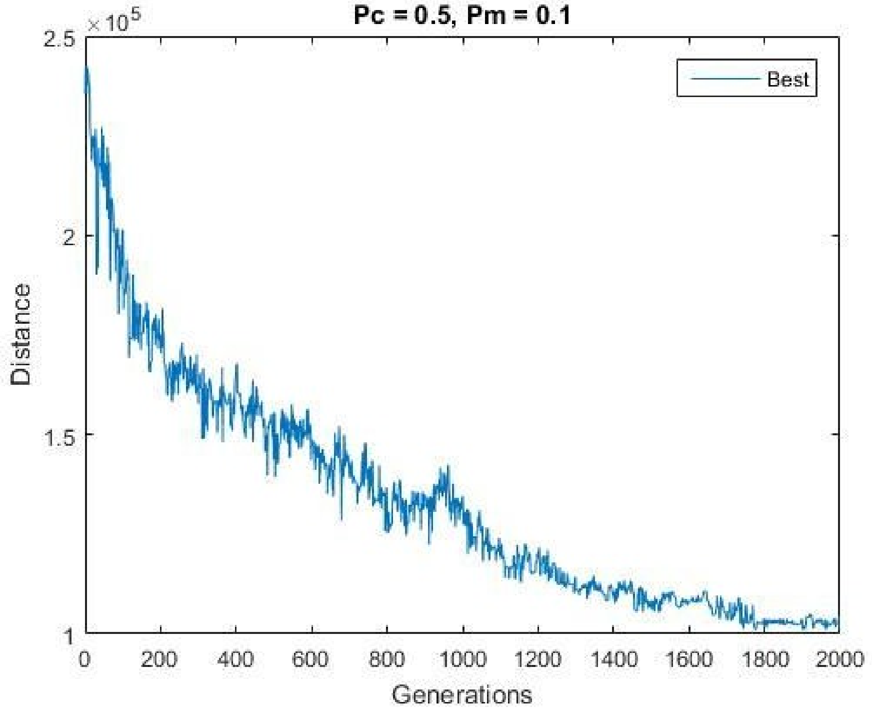




Time taken = 77.985797 seconds, Total distance = 112180.4209

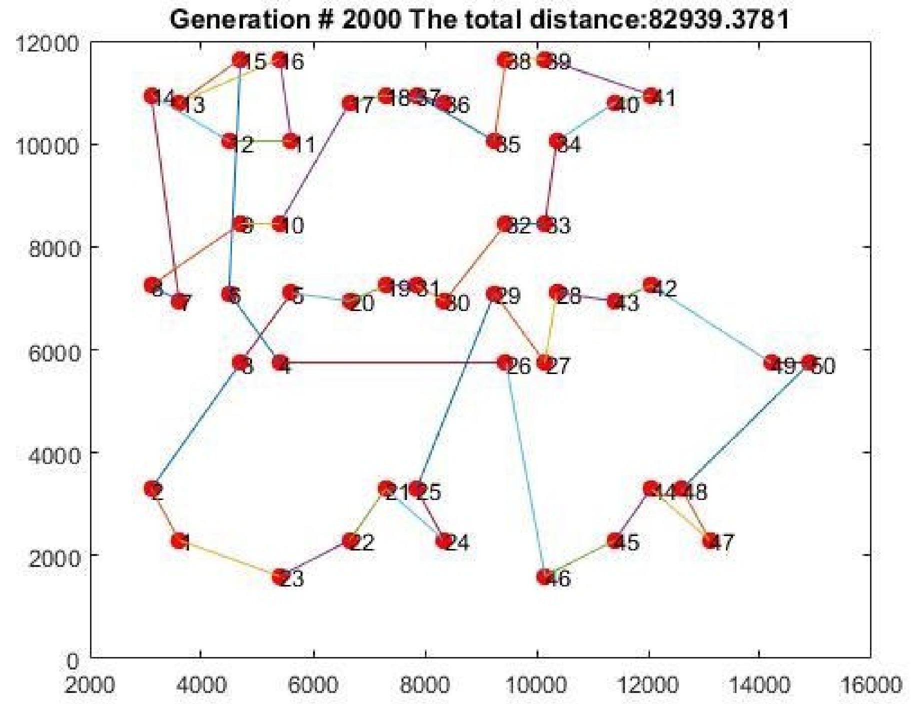
Crossover 2, Mutation 1

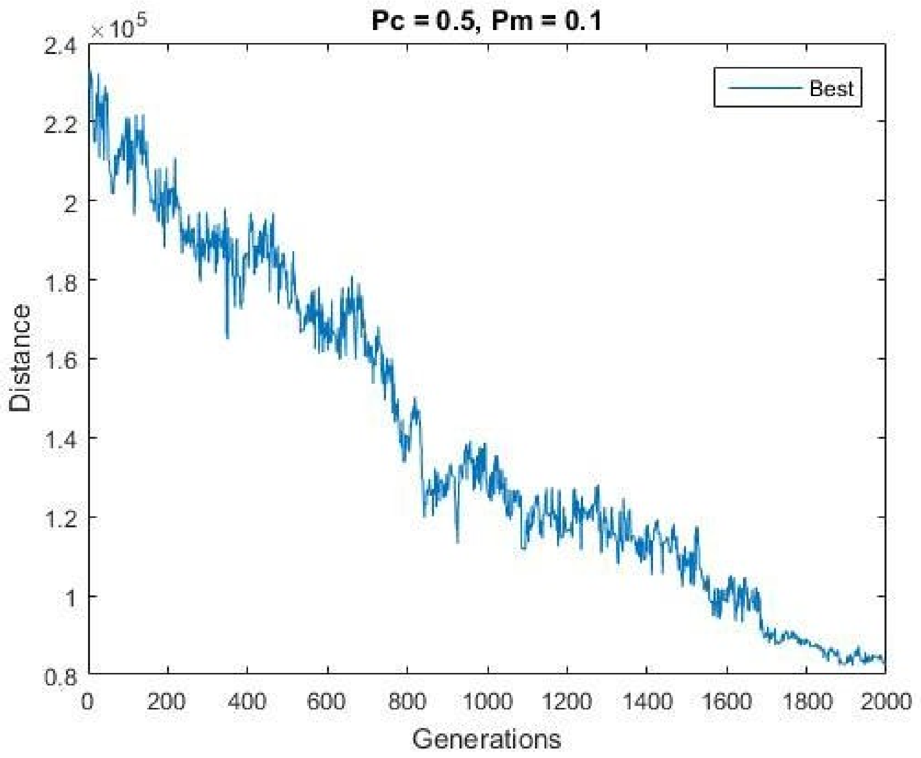




Time taken = 69.181294 seconds, Total distance = 101065.7456

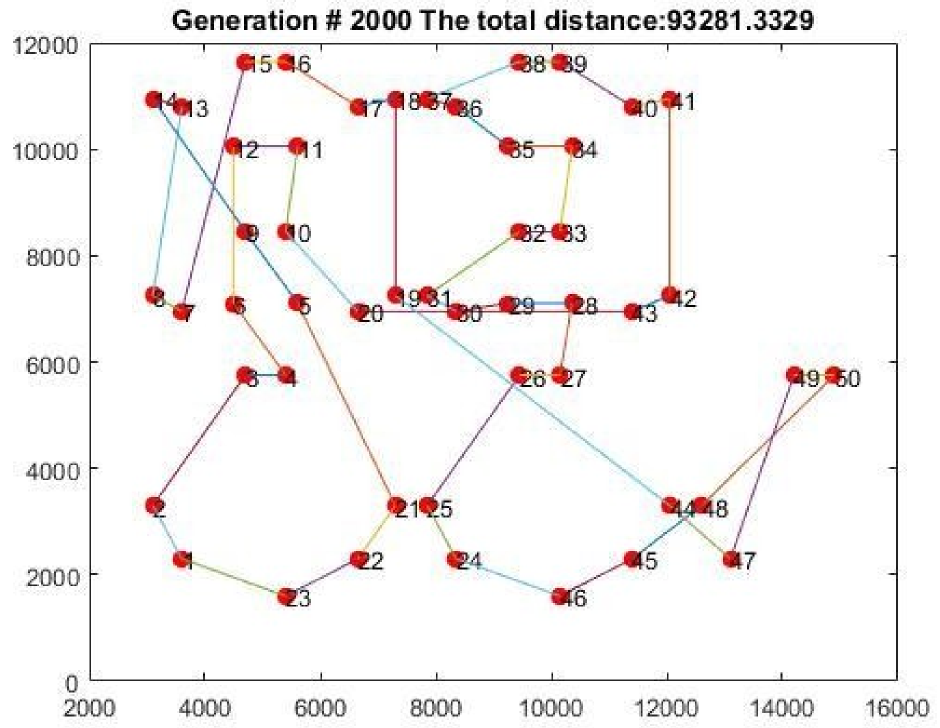
Crossover 3, Mutation 1

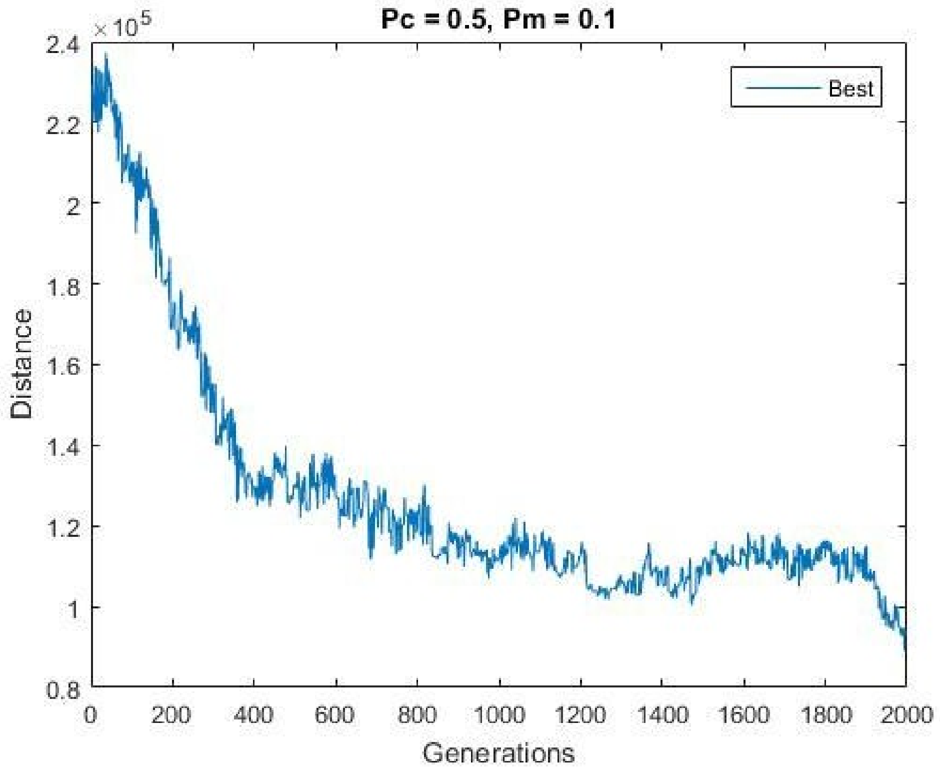




Time taken = 49.061004 seconds, Total distance = 82939.3781

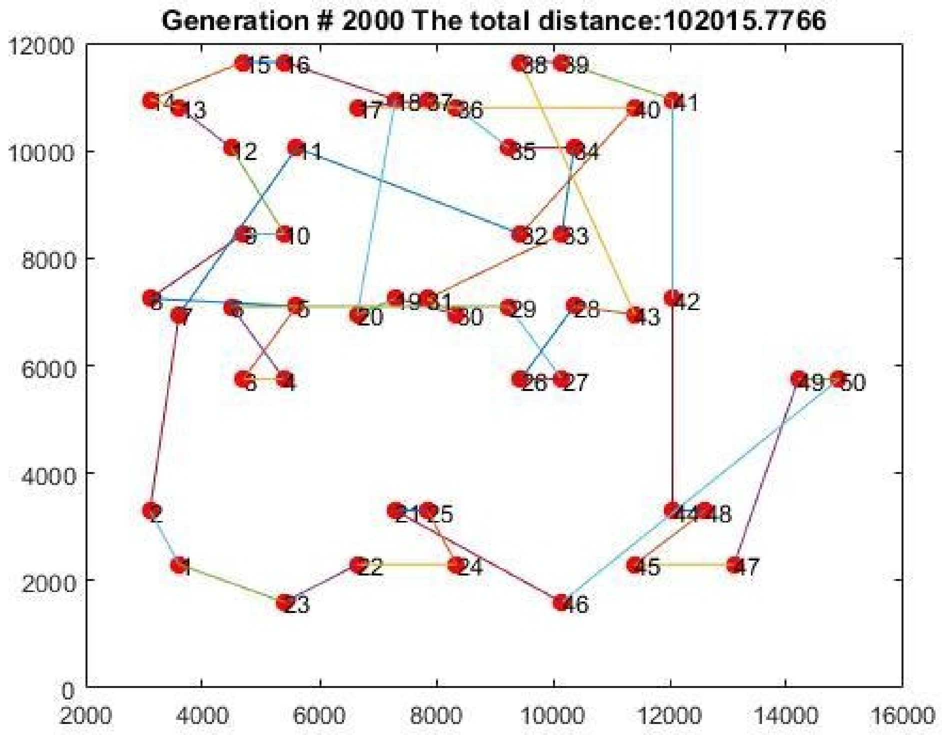
Crossover 1, Mutation 2

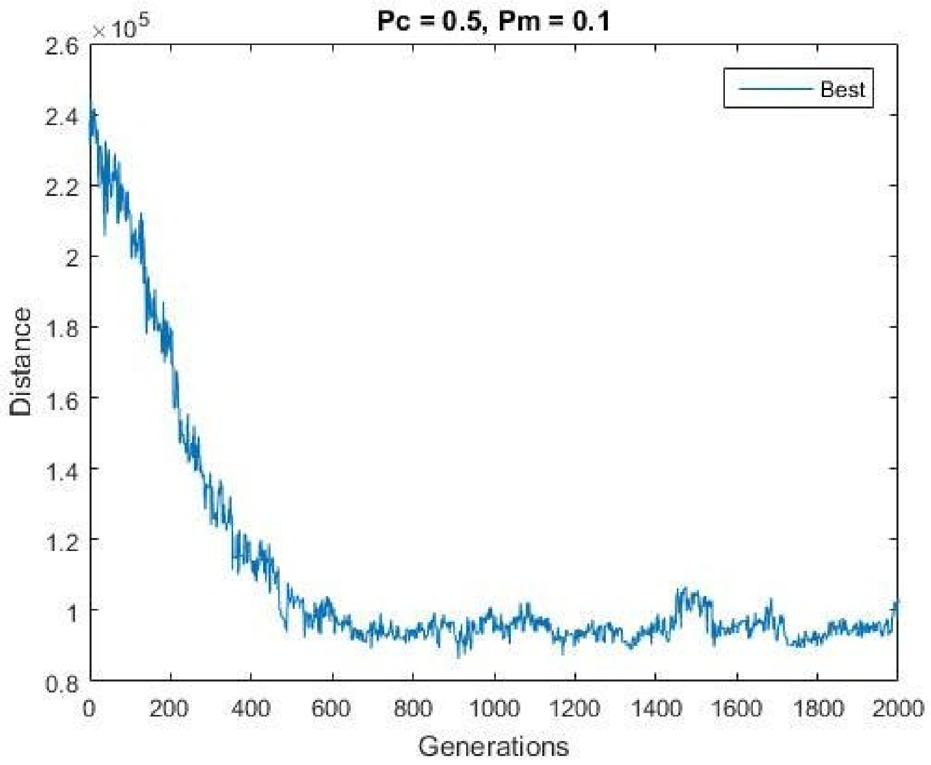




Time taken = 79.047696 seconds, Total distance = 93281.3329

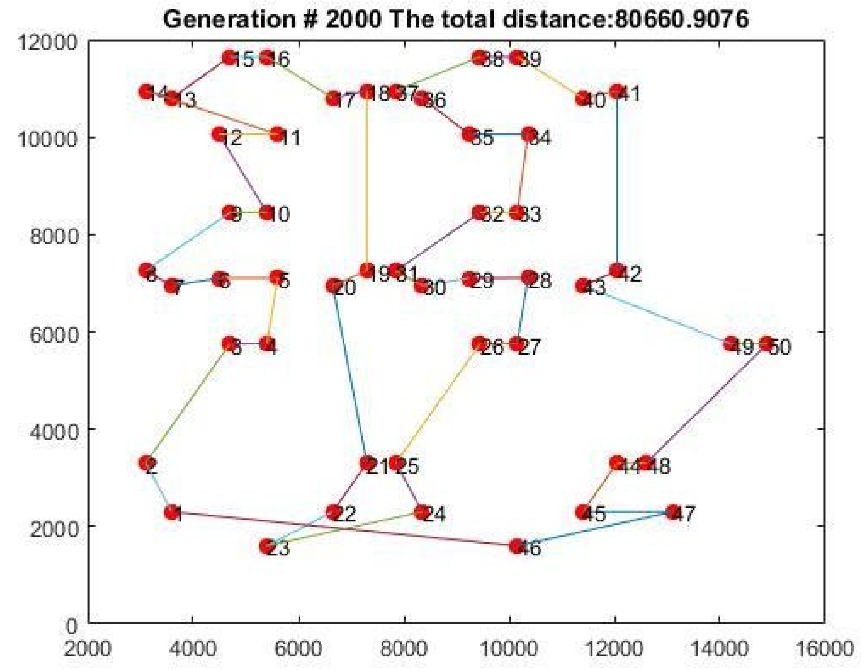
Crossover 2, Mutation 2

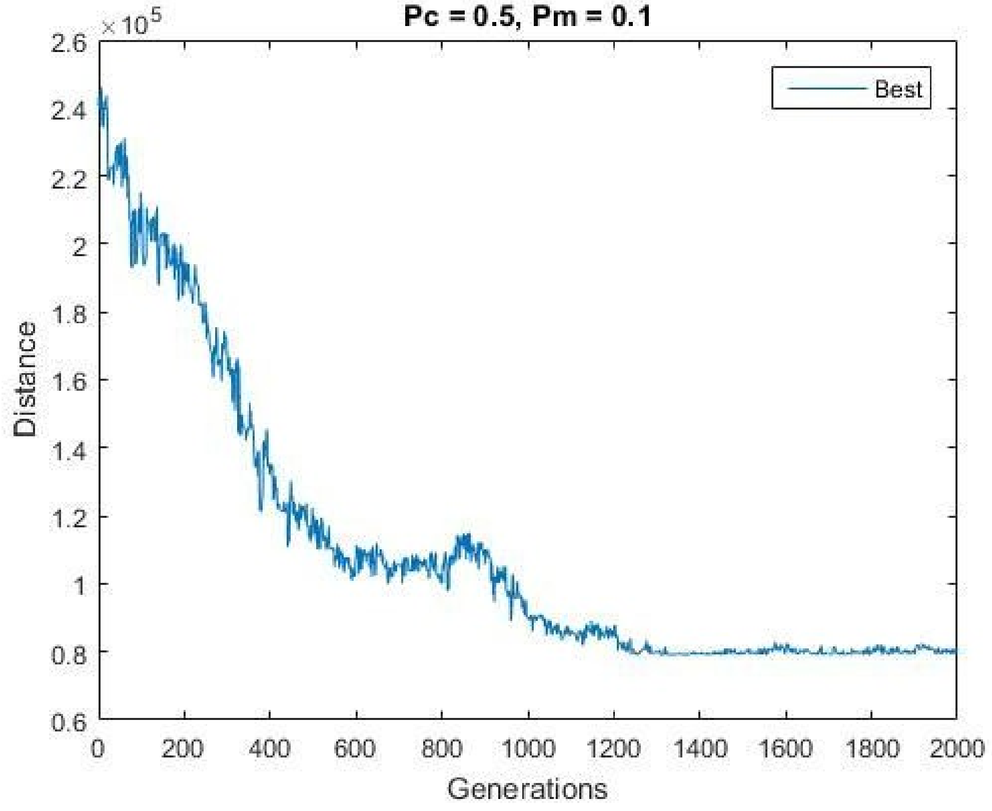




Time taken = 71.619697 seconds, Total distance = 102015.7766

Crossover 3, Mutation 2





Time taken = 56.984316 seconds, Total distance = 80660.9076

Table of Performance Comparison

Combination	Time Taken, s	Best Final Distance	Best Fitness Value
Crossover 1, Mutation 1	77.985797	112180.4209	1.1218e+05
Crossover 2, Mutation 1	69.181294	100863.061	1.0717e+05
Crossover 3, Mutation 1	49.061004	82939.3781	8.2939e+04
Crossover 1, Mutation 2	79.047696	93281.3329	9.3281e+04
Crossover 2, Mutation 2	71.619697	102015.7766	1.0202e+05
Crossover 3, Mutation 2	56.984316	80660.9076	8.0661e+04

Step-8 Performance Evaluation and Discussions

Students will be required to do the following:

i. Discuss about the performance of different crossover & mutation combinations (in terms of time taken, best final distance, fitness value, etc).

ii. Discuss on the effects of probability of crossover/mutation on the performance of the optimization algorithms (i.e. what happens to the results and optimization performance when the probabilities are increased/decreased). **Use the combination which provided "best" results from part (i).**

Relevant explanations and evidences are required when discussing the above. The parameters which are set constant should be clearly mentioned in the report.