1. Use the GA to solve the travel salesmen problem with 25 cities. You can assume the missing information.

Solution:

Assumption:

1. Assume the population is 5 \* 4 = 20.
2. The reproduction procedure does not have mating but mutation only (To simplify the problem). For each group with 4 possible TSP solutions, the new group is form by the best solution within the group, and 3 new solutions, which are mutated (three types of mutations) from the best solution inside the group.
3. Mutation points are random.
4. There are three types of mutations. The first one is Flip, The second one is Swap and the third type is Slide.
5. Suppose one of the solution is [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25] and the start point of mutation is 1 and the end point of mutation is 4. Then Flip will produce solution [4 3 2 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]. Swap will produce solution [4 2 3 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]. Slide will produce solution [2 3 4 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25].
6. City locations are 2-D only.

The Matlab code is as followed

In genCityLocation.m file:

% This function generates the city locations and

% make sure all the city have different locations [x y]

function xy = genCityLocation(num\_city)

bound = 2 .\* num\_city;

x = randperm(bound, num\_city);

y = randperm(bound, num\_city);

xy = [x' y'];

end

In tsp\_ga1.m file:

function [optD optR] = tsp\_ga1(xy, times)

%times = 100;

%x = randperm(100,25);

%y = randperm(100,25);

%xy = [x' y'];

% Initiate the population

popSize = 20; % Population size is 20

groupSize = 4; % Group size is 4

numCity = 25; % number of different cities is 25

pop = zeros(popSize, numCity); % A matrix with 20 rows and 25 columns

for i = 1:popSize

pop(i, :) = randperm(numCity);

end

% Initiate the distance matrix

disMatx = zeros(numCity, numCity);

for j = 1:numCity

for k = 1:numCity

disMatx(j, k) = sqrt((xy(j, 1) - xy(k, 1))^2 + (xy(j, 2) - xy(k, 2))^2);

end

end

% Run the GA

globalMin = Inf; % Global minimum distance

totalDist = zeros(1, popSize); % Total distance for each solution

disHistory = zeros(1, times);

tempPop = zeros(4, numCity);

newPop = zeros(popSize, numCity); % New population after mutaions

for t = 1:times

% Calculate the total distance

for p = 1:popSize

d = disMatx(pop(p, numCity), pop(p, 1));

for q = 2:numCity

d = d + disMatx(pop(p, q - 1), pop(p, q));

end

totalDist(p) = d;

end

% Find the best solution in the population

[minDist index] = min(totalDist);

disHistory(t) = minDist;

if minDist < globalMin

globalMin = minDist;

optRoute = pop(index, :); % Best solution for now

end

% GA opeartors

randomOrder = randperm(popSize);

for p = 4:4:popSize

group = pop(randomOrder(p - 3:p), :);

groupDist = totalDist(randomOrder(p - 3:p));

[groupMinDist groupIndex] = min(groupDist);

groupOptRoute = group(groupIndex, :); % Get the best solution in one group

routeInsertionPoints = sort(ceil(numCity\*rand(1,2))); % Randomly get the mutaion point

I = routeInsertionPoints(1);

J = routeInsertionPoints(2);

for k = 1:4 % Mutate the best and get three new solution

tempPop(k, :) = groupOptRoute;

switch k

case 2 % Flip

tempPop(k, I:J) = tempPop(k, J:-1:I);

case 3 %Swap

tempPop(k,[I J]) = tempPop(k, [J I]);

case 4 %Slide

tempPop(k, I:J) = tempPop(k, [I+1:J I]);

end

end

newPop(p-3:p, :) = tempPop;

end

pop = newPop;

end

% Plot the optRoute

figure('Name', 'TSP\_GA Result');

subplot(1, 2, 1);

plot(xy(:,1), xy(:,2), '.');

title('City Locations');

subplot(1, 2, 2);

rte = optRoute([1:numCity 1]);

plot(xy(rte, 1), xy(rte, 2), 'r.-');

title(sprintf('Final Minimum Distance = %1.4f', globalMin));

optD = globalMin;

optR = optRoute;

end

In the command line window:

xy = genCityLocation(25);

tsp\_ga1(xy, 10);

tsp\_ga1(xy, 100);

tsp\_ga1(xy, 1000);

tsp\_ga1(xy, 10000);

And the result of each function is shown in Figure 1, 2 , 3, 4.

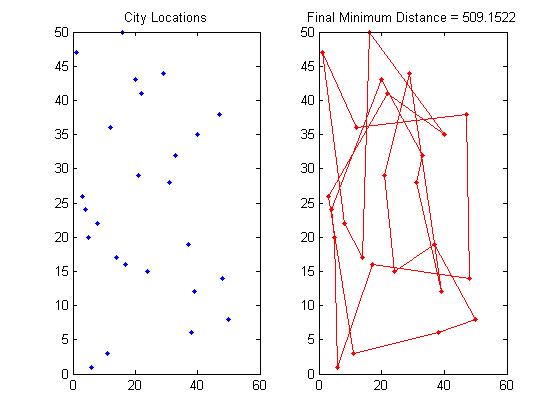


Figure 1: Result with iteration time = 10

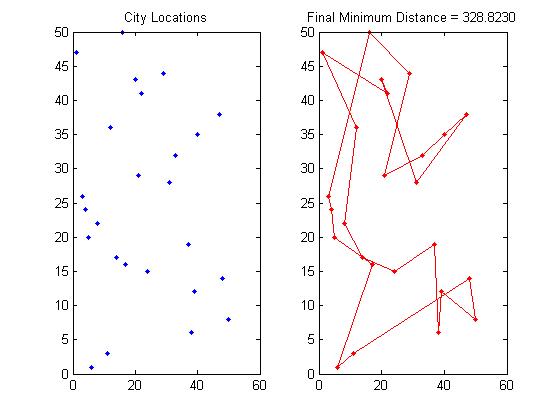


Figure 2: Result with iteration time = 100

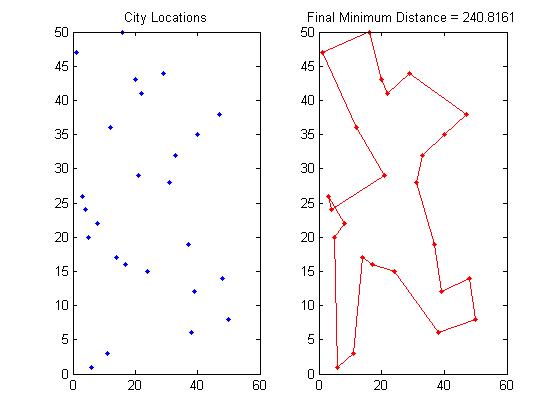


Figure 3: Result with iteration time = 1000

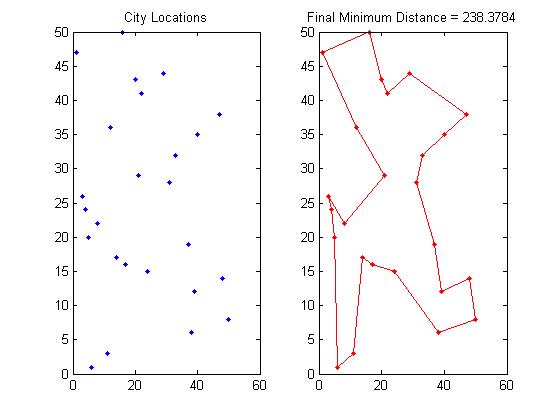


Figure 4: Result with iteration time = 10000