MECS 4510
Evolutionary Computation and Design Automation

Xinsheng Gu | UNI: xg2381

Instructor: Hod Lipson

Date submitted: 10/5/2021

Grace hours used: 0 hour

Grace hours remaining: 96 hours

Results Summary

Table 1: Results Summary

		Evaluations	Length
TSP	The shortest path	10000	332.2247
	The longest path	10000	642.7133

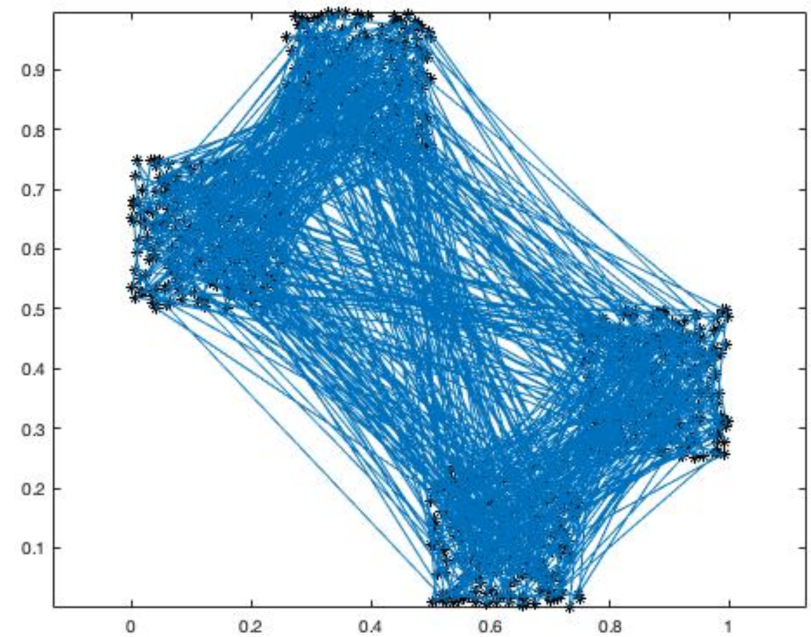

Fig 1 Shortest path found. Length: 332.2247 Evaluations: 10000

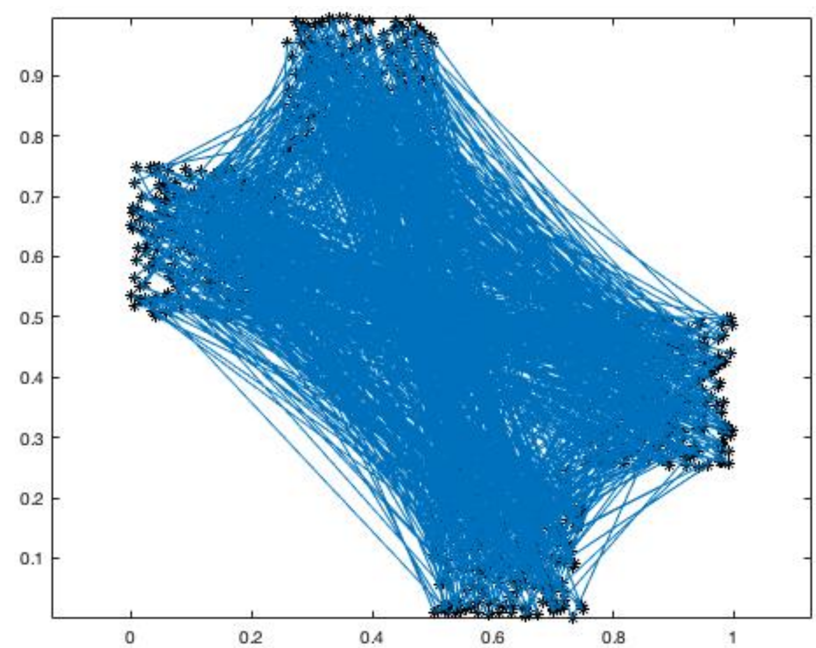

Fig 2 Longest path found. Length: 642.7133 Evaluations: 10000

Methods

Representation: I used an array of city indices to represent the route. The MATLAB function *randperm(N),* where N is the number of cities*,* is conducted to create an integer array indicating the traveling salesman go through each city exactly once.

Random Search: I randomly created an array of city indices and calculated its path length *D*. In order to obtain the shortest path, I first set a variable *Dmin*, which initial value is positive *inf*, and let the first length *D* replace *Dmin*. After that, if the next length D is shorter than *Dmin*, length D is the new value of *Dmin*; if length *D* is longer than *Dmin*, then nothing happens. In this way, the shortest length can be obtained through iteration.

Hill Climber: A bunch of arrays are created as a population. The initial searching point, an individual from the population, is set randomly. Next, the path length of initial point, as well as the point right next to it and left next to it, are calculated. By comparing the path length of these three points, I chose the point with shortest path length as the searching point for the next iteration. If the searching point is stuck in a local optimal point for a certain times of iteration, it will start to search points which are a little bit far from it instead of points next to it. In this way, it can jump out of the local optimum.

Genetic Algorithm: Genetic Algorithm is composed of selection, crossover and mutation.

For the selection method in GA, I calculated fitness values for every individual and then sorted them according to their fitness and selected the top 50% individuals and discard the others.

Single-point crossover is carried out to reproduce offspring. I randomly matched parents one by one and chose the crossover point randomly. When doing the gene exchange, in order to make sure the city index is neither skipped nor repeated, the city index inside the individual switch its position first. This method can be clearly depicted by fig 3.

After crossover, mutation next is aimed at adding diversity for the population. In this case, I randomly selected individuals needed mutating and randomly switched two city indices on each individual. The method is illustrated as fig 4.

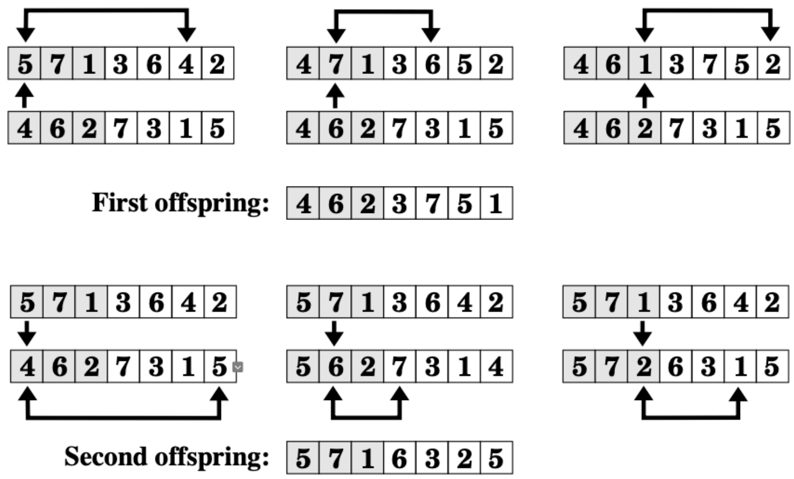

Fig 3 Crossover method between two parents and two children[1]

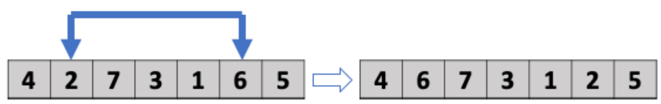

Fig 4 Mutation method by switching the position of two indices

Analysis of performance: Compared with the performance of Random Search, Hill Climber and Genetic Algorithm, I found GA is much better than RS and HC, which means GA can search solution faster and more likely to hit a better solution. Although RS and HC are not as good as GA, they still have some edges. For instance, RS is a global search which is less likely to hit a plateau; while HC can search an optimum quickly during the first hundreds of iterations.

For GA, the bigger population and the bigger times of iteration, the more likely it is to find better solution, which also takes much longer time to calculate. A low mutation rate may let GA stuck in a local optimum, while a high mutation rate would make GA so random that it is far away from evolution algorithm.

[1] Göktürk Üçoluk (2002) Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation, Intelligent Automation & Soft Computing, 8:3, 265-272, DOI: 10.1080/10798587.2000.10642829

Performance Plots

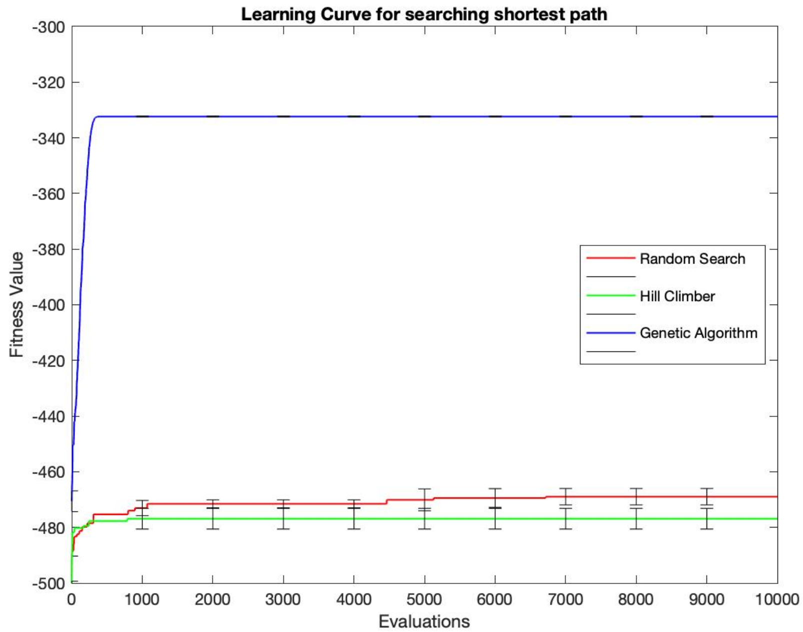

Fig 5 Learning Curve for searching shortest path length (RS, HC, GA) with error bar

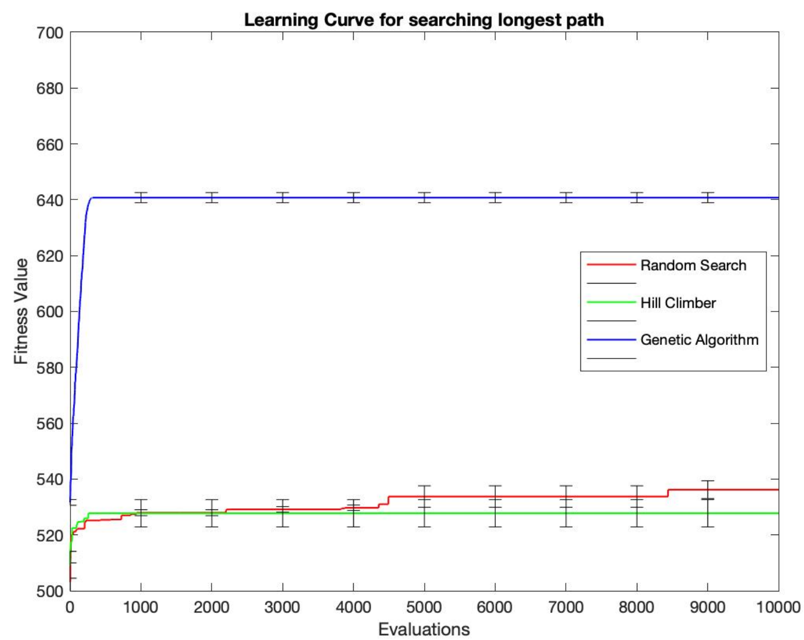

Fig 6 Learning Curve for searching longest path length (RS, HC, GA) with error bar

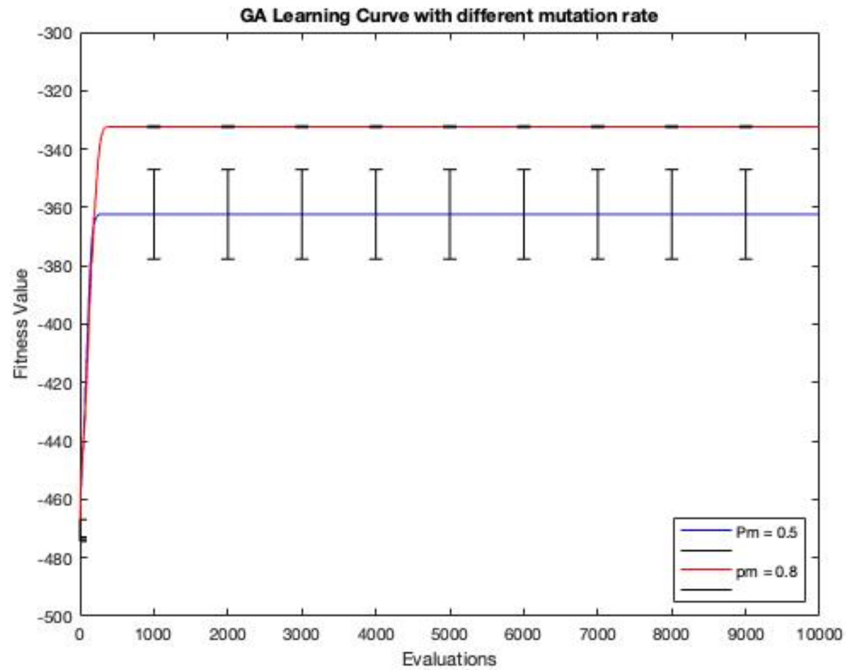

Fig 7 Learning Curve for Genetic Algorithm under mutation rate = 0.5 and 0.8

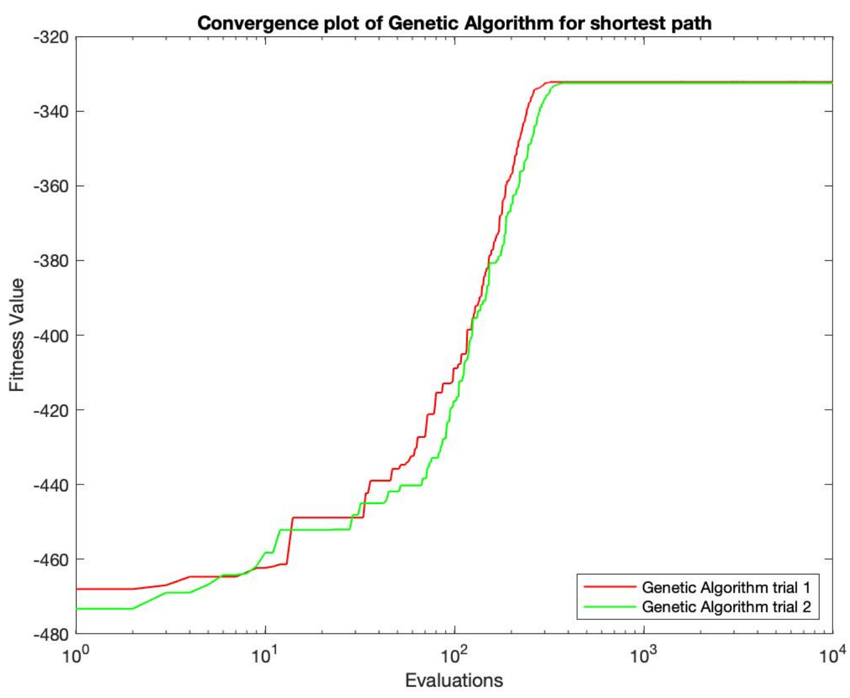

Fig 8 Convergence plot of Genetic Algorithm for searching shortest path length

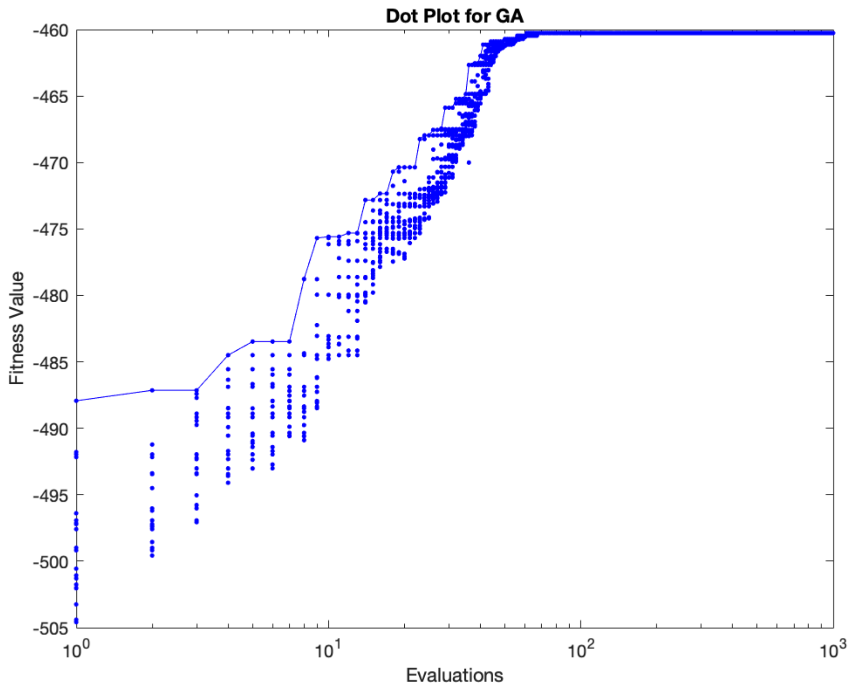

Fig 9 Dot Plot for Genetic Algorithm. The more times of evaluation, the more density of dots.

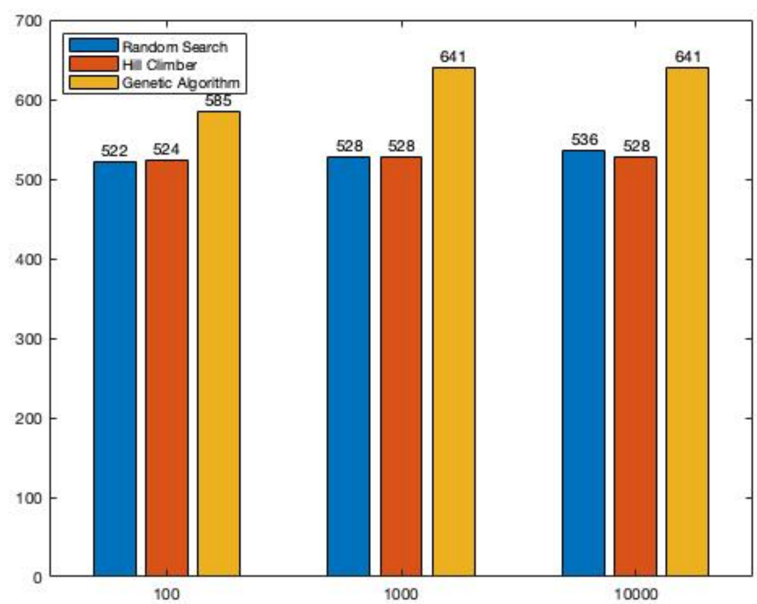

Fig 10 Bar Chart comparing the performance for RS, HC and GA when iteration comes to 100, 1000, and 10000.

Movie of optimizing path (one frame every time path improves) can be found on YouTube: https://www.youtube.com/watch?v=frziYSLrJ0A

Appendix

```
%%%%%%% Algorithm Combo %%%%%%%
%function [Dmin,Rmin,trace] = TSPmin_RS_func(N,iteration,f)
%function [Dmin,Rmin,trace] = TSPmin_HC_func(N,NP,iteration,f)
%function [minFit,r_best,trace] = TSPmin_GA_func(N,NP,G,Pm,f)
close all;
clear;
clc;

f = importdata('tsp.txt');
% N = 100;
N = size(f,1);
NP = 500;
iteration = 10000;
G = iteration;
Pm = 0.8;

tic
[Dmin_RS, Rmin_RS, Trace_RS] = TSPmin_RS_func(N,iteration,f);
t_RS = toc;
tic
[Dmin_HC, Rmin_HC, Trace_HC] = TSPmin_HC_func(N,NP,iteration,f);
t_HC = toc;
tic
[Dmin_GA, Rmin_GA, Trace_GA] = TSPmin_GA_func(N,NP,G,Pm,f);
t_GA = toc;

h = figure(1)
semilogx(-Trace_RS,'r.-')
hold on
semilogx(-Trace_HC,'g.-')
semilogx(-Trace_GA,'b.-')
xlabel('Evaluations')
ylabel('Fitness Value')
title('Learning Curve')
legend('Random Search','Hill Climber','Genetic Algorithm','Location','Southeast')
hold off
savefig(h,'Learning Curve.fig')
close(h)

g = figure(2)
TSP_plot(N,Rmin_GA,f)
savefig(g,'Route.fig')
close(g)

save('workspace.mat')
-----------------------------------------------------------------------------------------------
%RS_learning curve shortest
function [Dmin,Rmin,trace] = TSPmin_RS_func(N,iteration,f)

% N = size(f,1);
% N = 100;
% iteration = 100;
r = zeros(N,iteration);
Dmin = inf; %initial setting for searching min

for i = 1:iteration
    r(:,i) = randperm(N);
    [D, r] = TSP_PathLength(N, r(:,i), f);
    if (D < Dmin)
        Dmin = D;
        Rmin = r;
    end
    trace(i) = Dmin;
end
-----------------------------------------------------------------------------------------------
%%%%%%%% Hill Climber %%%%%%%%%%
function [Dmin,Rmin,trace] = TSPmin_HC_func(N,NP,iteration,f)

r = zeros(NP,N);
```

```
trace = zeros(1,iteration);
step = 1; %range of index for hill climb searching


for i = 1:NP
    r(i,:) = randperm(N);
end
Dmin = inf; %initial setting for searching minimum
count = 0; %counting the times of staying on the same point

for k = 1:iteration
    h = randi([1,NP],1,1);  %randomly select an initial individual/route


    [D_center, r_center] = TSP_PathLength(N, r(h,:), f);

    if (h - step) < 1
        idx = h - step + NP;
    else
        idx = h - step;
    end
    [D_left, r_left] = TSP_PathLength(N,r(idx,:),f);

    if (h + step) > NP
        idx = h + step - NP;
    else
        idx = h + step;
    end
    [D_right, r_right] = TSP_PathLength(N,r(idx,:),f);


    if (D_left < D_center)
        D = D_left;
        R = r_left;
    else
        D = D_center;
        R = r_center;
        count = count + 1;
    end

    if (D_right < D)
        D = D_right;
        R = r_right;
    else
        count = count + 1;
    end


    if (D < Dmin)
        Dmin = D;
        Rmin = r;
    end

    trace(k) = Dmin;

    if (count >= NP) && (step < NP)
        step = step + 1;
        count = 0;
    end
end
-------------------------------------------------------------------------------------------------
function [Dmin,Rmin,trace] = TSPmin_GA_func(N,NP,G,Pm,f)

trace = zeros(1,G); %record of best fitness for each generation
Dmin = inf; %initial setting for searching min

for i = 1:NP    %initialize poplutaion
    r(i,:) = randperm(N); %index of each city
end

%%%%%%%%% iteration of GA %%%%%%%%
for k = 1:G  %test when k = 1
%%%%%%%%% crossover and get children %%%%%%
```

```
    p = randperm(NP);  %initialize matching parents
    pairs = zeros(NP/2,2);
    for i = 1:NP/2
        pairs(i,:) = [p(2*i-1),p(2*i)];
    end
    pairs;

    for pp = 1:size(pairs,1)

    i = pairs(pp,1);
    j = pairs(pp,2);
    ub = N - 1; % single cross point
    lb = 1;
    CrossPoint = round((ub - lb)*rand() + lb);


    Parent1 = r(i,:);
    Parent2 =r(j,:);
    for m = 1:CrossPoint
        idx = find(Parent1 == Parent2(m));
        Parent1(idx) = Parent1(m);
        Parent1(m) = Parent2(m);
    end
    child1 = Parent1;

    Parent1 = r(i,:);
    Parent2 =r(j,:);
    for m = 1:CrossPoint
        idx = find(Parent2 == Parent1(m));
        Parent2(idx) = Parent2(m);
        Parent2(m) = Parent1(m);
    end
    child2 = Parent2;
    r = [r;child1;child2];

    end

%%%%%%%%%%%%% Mutation %%%%%%%%%%%%%%%%%%
    i = 1;
    while i <= round(NP*Pm)
        h = randi([1,NP],1,1);       %select an individual to mutate
        for j = 1:round(N*Pm)
            g1 = randi([1,N],1,1);
            g2 = randi([1,N],1,1);   %select two cities and flip the route with each other
            temp = r(h,g2);
            r(h,g2) = r(h,g1);
            r(h,g1) = temp;
        end
        i = i+1;
    end

%%%%%%%%%%%%% calculate fitness for each route %%%%%%%%%%%
    D = zeros(1,size(r,1));
    for i = 1:size(r,1)
        [D(i),~] = TSP_PathLength(N,r(i,:),f);
        maxFit=max(D);            %maximum fitness
        minFit=min(D);           %minimum fitness
%         rr=find(D == maxFit);
        rr=find(D == minFit);
        fBest=r(rr(1,1),:);         %the best individual
    end
%%%%%%%%%%%%%% Selection %%%%%%%%%%%%%%
    [D_sorted,I] = sort(D,'ascend');    %for finding minimum
    r = r(I(1:round(size(r,1)/2)),:);    %selecting top 50% and discard the others


    D = minFit; %shortest path
    R = r(1,:); %best route
    if (D < Dmin)
        Dmin = D;
        Rmin = R;
    end
    trace(k) = Dmin;
```

```
    disp(['generation:' num2str(k)])

end
```

```
%%%%%%% Algorithm Combo maximum %%%%%%%
%function [Dmin,Rmin,trace] = TSPmin_RS_func(N,iteration,f)
%function [Dmin,Rmin,trace] = TSPmin_HC_func(N,NP,iteration,f)
%function [minFit,r_best,trace] = TSPmin_GA_func(N,NP,G,Pm,f)
close all;
clear;
clc;

f = importdata('tsp.txt');
% N = 100;
N = size(f,1);
NP = 500;
iteration = 10000;
G = iteration;
Pm = 0.8;

tic
[Dmin_RS, Rmin_RS, Trace_RS] = TSPmax_RS_func(N,iteration,f);
t_RS = toc;
tic
[Dmin_HC, Rmin_HC, Trace_HC] = TSPmax_HC_func(N,NP,iteration,f);
t_HC = toc;
tic
[Dmin_GA, Rmin_GA, Trace_GA] = TSPmax_GA_func(N,NP,G,Pm,f);
t_GA = toc;

h = figure(1)
semilogx(Trace_RS,'r.-')
hold on
semilogx(Trace_HC,'g.-')
semilogx(Trace_GA,'b.-')
xlabel('Evaluations')
ylabel('Fitness Value')
title('Learning Curve')
legend('Random Search','Hill Climber','Genetic Algorithm','Location','Southeast')
hold off
savefig(h,'Learning Curve.fig')
close(h)

g = figure(2)
TSP_plot(N,Rmin_GA,f)
savefig(g,'Route.fig')
close(g)

save('workspace.mat')
```

```
%RS_learning curve shortest
function [Dmax,Rmax,trace] = TSPmax_RS_func(N,iteration,f)

r = zeros(N,iteration);
Dmax = -inf; %initial setting for searching max
trace = zeros(1,iteration);

for i = 1:iteration
    r(:,i) = randperm(N);
    [D, r] = TSP_PathLength(N, r(:,i), f);
    if (D > Dmax)
        Dmax = D;
        Rmax = r;
    end
    trace(i) = Dmax;
end
```

```
%%%%%%%% Hill Climber %%%%%%%%%%
function [Dmax,Rmax,trace] = TSPmax_HC_func(N,NP,iteration,f)
trace = zeros(1,iteration);
step = 1; %range of index for hill climb searching
```

```
for i = 1:NP
    r(i,:) = randperm(N);
end
Dmax = -inf; %initial setting for searching maximum
count = 0; %counting the times of staying on the same point

for k = 1:iteration
    h = randi([1,NP],1,1);  %randomly select an initial individual/route


    [D_center, r_center] = TSP_PathLength(N, r(h,:), f);

    if (h - step) < 1
        idx = h - step + NP;
    else
        idx = h - step;
    end
    [D_left, r_left] = TSP_PathLength(N,r(idx,:),f);

    if (h + step) > NP
        idx = h + step - NP;
    else
        idx = h + step;
    end
    [D_right, r_right] = TSP_PathLength(N,r(idx,:),f);


    if (D_left > D_center)
        D = D_left;
        R = r_left;
    else
        D = D_center;
        R = r_center;
        count = count + 1;
    end

    if (D_right > D)
        D = D_right;
        R = r_right;
    else
        count = count + 1;
    end


    if (D > Dmax)
        Dmax = D;
        Rmax = r;
    end

    trace(k) = Dmax;

    if (count >= NP) && (step < NP)
        step = step + 1;
        count = 0;
    end
end
------------------------------------------------------------------------------------------=
function [Dmax,Rmax,trace] = TSPmax_GA_func(N,NP,G,Pm,f)

trace = zeros(1,G); %record of best fitness for each generation
Dmax = -inf; %initial setting for searching max

for i = 1:NP    %initialize poplutaion
    r(i,:) = randperm(N); %index of each city
end

%%%%%%%%% iteration of GA %%%%%%%%
for k = 1:G  %test when k = 1
%%%%%%%%% crossover and get children %%%%%%
    p = randperm(NP);  %initialize matching parents
    pairs = zeros(NP/2,2);
```

```
    for i = 1:NP/2
        pairs(i,:) = [p(2*i-1),p(2*i)];
    end
    pairs;

    for pp = 1:size(pairs,1)

    i = pairs(pp,1);
    j = pairs(pp,2);
    ub = N - 1; % single cross point
    lb = 1;
    CrossPoint = round((ub - lb)*rand() + lb);


    Parent1 = r(i,:);
    Parent2 =r(j,:);
    for m = 1:CrossPoint
        idx = find(Parent1 == Parent2(m));
        Parent1(idx) = Parent1(m);
        Parent1(m) = Parent2(m);
    end
    child1 = Parent1;

    Parent1 = r(i,:);
    Parent2 =r(j,:);
    for m = 1:CrossPoint
        idx = find(Parent2 == Parent1(m));
        Parent2(idx) = Parent2(m);
        Parent2(m) = Parent1(m);
    end
    child2 = Parent2;
    r = [r;child1;child2];

    end

%%%%%%%%%%%%% Mutation %%%%%%%%%%%%%%%%%%
    i = 1;
    while i <= round(NP*Pm)
        h = randi([1,NP],1,1);      %select an individual to mutate
        for j = 1:round(N*Pm)
            g1 = randi([1,N],1,1);
            g2 = randi([1,N],1,1);    %select two cities and flip the route with each other
            temp = r(h,g2);
            r(h,g2) = r(h,g1);
            r(h,g1) = temp;
        end
        i = i+1;
    end

%%%%%%%%%%%%% calculate fitness for each route %%%%%%%%%%%
    D = zeros(1,size(r,1));
    for i = 1:size(r,1)
        [D(i),~] = TSP_PathLength(N,r(i,:),f);
        maxFit=max(D);             %maximum fitness
        minFit=min(D);            %minimum fitness
        rr=find(D == maxFit);
%         rr=find(D == minFit);
        fBest=r(rr(1,1),:);         %the best individual
    end
%%%%%%%%%%%%%% Selection %%%%%%%%%%%%%%
%     [D_sorted,I] = sort(D,'ascend');   %for finding minimum
    [~,I] = sort(D,'descend');   %for finding maximum
    r = r(I(1:round(size(r,1)/2)),:);   %selecting top 50% and discard the others


%     D = minFit;   %shortest path
    D = maxFit;  %longest path
    R = r(1,:); %best route
    if (D > Dmax)
        Dmax = D;
        Rmax = R;
    end
    trace(k) = Dmax;
```

```
    disp(['generation:' num2str(k)])

end
------------------------------------------------------------------------------------------------
function [D, r] = TSP_PathLength(N, r, f) %calculate total length of path

D = 0;
for i = 1:N-1
    d = sqrt((f(r(i),1)-f(r(i+1),1))^2 + (f(r(i),2)-f(r(i+1),2))^2);
    D = D + d;
end
    D = D + sqrt((f(r(1),1)-f(r(N),1))^2 + (f(r(1),2)-f(r(N),2))^2); %link initial dot and the
last dot

end

------------------------------------------------------------------------------------------------
function [] = TSP_plot(N, r, f)

for i = 1:N
    plot(f(i,1),f(i,2),'k*')
    hold on
end



for i = 1:N-1
    line([f(r(i),1),f(r(i+1),1)],[f(r(i),2),f(r(i+1),2)])
end
axis equal

end
```

```
%%%%%%%%%% Data visualization %%%%%%%%%
close all
clear
clc

min_2 = load('workspace_min_2.mat');

min_3 = load('workspace_min_3.mat');

min_4 = load('workspace_min_4.mat');

max_1 = load('workspace_max_1.mat');

max_2 = load('workspace_max_2.mat');

max_3 = load('workspace_max_3.mat');

f = importdata('tsp.txt');
N = size(f,1)

%%
disp(['Dmin_GA:' num2str(min_2.Dmin_GA) ';' num2str(min_3.Dmin_GA) ';' num2str(min_4.Dmin_GA)])
disp(['Dmax_GA:' num2str(max_1.Dmin_GA) ';' num2str(max_2.Dmin_GA) ';' num2str(max_3.Dmin_GA)])
% figure(1)
% TSP_plot(N, min_3.Rmin_GA,f)
% figure(2)
% TSP_plot(N, max_3.Rmin_GA,f)
%%%%%%%shortest length%%%%%%%%


f1 = figure(1)
%%%%%%%% RS %%%%%%%%
Trace_min_RS = [min_2.Trace_RS;min_3.Trace_RS;min_4.Trace_RS];
mean_Trace_min_RS = mean(Trace_min_RS);
std_Trace_min_RS = std(Trace_min_RS);
mean_Trace_min_RS = mean(Trace_min_RS);
err_min_RS = std_Trace_min_RS(1:1000:end);
x = [1:1000:10000];
x_std_min_RS = zeros(size(err_min_RS));
```

```
h(1) = plot(-mean_Trace_min_RS,'Color','r','LineWidth',1);
% legend(h(1),'Random Search')

hold on
errorbare('v',x,-mean_Trace_min_RS(1:1000:end),x_std_min_RS,err_min_RS,'');


%%%%%%%% HC %%%%%%%%
Trace_min_HC = [min_2.Trace_HC;min_3.Trace_HC;min_4.Trace_HC];
mean_Trace_min_HC = mean(Trace_min_HC);
std_Trace_min_HC = std(Trace_min_HC);
mean_Trace_min_HC = mean(Trace_min_HC);
err_min_HC = std_Trace_min_HC(1:1000:end);
x = [1:1000:10000];
x_std_min_HC = zeros(size(err_min_HC));

h(2) = plot(-mean_Trace_min_HC,'Color','g','LineWidth',1);
% legend(h(2),'Hill Climber')
errorbare('v',x,-mean_Trace_min_HC(1:1000:end),x_std_min_HC,err_min_HC,'')

%%%%%%%% GA %%%%%%%%
Trace_min_GA = [min_2.Trace_GA;min_3.Trace_GA;min_4.Trace_GA];
std_Trace_min_GA = std(Trace_min_GA);
mean_Trace_min_GA = mean(Trace_min_GA);
% errorbar(-mean_Trace_min_GA,std_Trace_min_GA(1:100),'vertical')

% semilogx(-mean_Trace_min_GA)
% hold on


err_min_GA = std_Trace_min_GA(1:1000:end);
x = [1:1000:10000];
x_std_min_GA = zeros(size(err_min_GA));

h(3) = plot(-mean_Trace_min_GA,'Color','b','LineWidth',1)
% legend(h(3),'Genetic Algorithm')
errorbare('v',x,-mean_Trace_min_GA(1:1000:end),x_std_min_GA,err_min_GA,'')


%%%%%%%%%%
legend('Random Search','','Hill Climber','','Genetic Algorithm','','Location','East')
% legend(h(1),'Random Search',h(2),'Hill Climber',h(3),'Genetic
Algorithm','Location','Southeast')
xlabel('Evaluations')
ylabel('Fitness Value')
title('Learning Curve for searching shortest path')
axis([0,10000,-500,-300])

% savefig('LearningCurve_shortest.fig')
% saveas(f1,'LearningCurve_shortest.jpg')

%%
%%%%%%%%%%%%% longest path %%%%%%%%%%%%%%
f2 = figure(2)
%%%%%%%% RS %%%%%%%%
Trace_max_RS = [max_1.Trace_RS;max_2.Trace_RS;max_3.Trace_RS];
mean_Trace_max_RS = mean(Trace_max_RS);
std_Trace_max_RS = std(Trace_max_RS);
mean_Trace_max_RS = mean(Trace_max_RS);
err_max_RS = std_Trace_max_RS(1:1000:end);
x = [1:1000:10000];
x_std_max_RS = zeros(size(err_max_RS));

g(1) = plot(mean_Trace_max_RS,'Color','r','LineWidth',1);
% legend(h(1),'Random Search')

hold on
errorbare('v',x,mean_Trace_max_RS(1:1000:end),x_std_max_RS,err_max_RS,'');


%%%%%%%% HC %%%%%%%%
Trace_max_HC = [max_1.Trace_HC;max_2.Trace_HC;max_3.Trace_HC];
mean_Trace_max_HC = mean(Trace_max_HC);
```

```
std_Trace_max_HC = std(Trace_max_HC);
mean_Trace_max_HC = mean(Trace_max_HC);
err_max_HC = std_Trace_max_HC(1:1000:end);
x = [1:1000:10000];
x_std_max_HC = zeros(size(err_max_HC));

g(2) = plot(mean_Trace_max_HC,'Color','g','LineWidth',1);
% legend(h(2),'Hill Climber')
errorbare('v',x,mean_Trace_max_HC(1:1000:end),x_std_max_HC,err_max_HC,'')

%%%%%%%% GA %%%%%%%%
Trace_max_GA = [max_1.Trace_GA;max_2.Trace_GA;max_3.Trace_GA];
std_Trace_max_GA = std(Trace_max_GA);
mean_Trace_max_GA = mean(Trace_max_GA);
% errorbar(-mean_Trace_min_GA,std_Trace_min_GA(1:100),'vertical')

% semilogx(-mean_Trace_min_GA)
% hold on


err_max_GA = std_Trace_max_GA(1:1000:end);
x = [1:1000:10000];
x_std_max_GA = zeros(size(err_max_GA));

g(3) = plot(mean_Trace_max_GA,'Color','b','LineWidth',1)
% legend(h(3),'Genetic Algorithm')
errorbare('v',x,mean_Trace_max_GA(1:1000:end),x_std_max_GA,err_max_GA,'')


%%%%%%%%%%
legend('Random Search','','Hill Climber','','Genetic Algorithm','','Location','East')
% legend(h(1),'Random Search',h(2),'Hill Climber',h(3),'Genetic
Algorithm','Location','Southeast')
xlabel('Evaluations')
ylabel('Fitness Value')
title('Learning Curve for searching longest path')
axis([0,10000,500,700])

% savefig('LearningCurve_longest.fig')
% saveas(f2,'LearningCurve_longest.jpg')

%%
%%%%%%%%% bar chart comparing two methods %%%%%%%%%%
bb = zeros(3,3)
for i = [100,1000,10000]
    bb(i,1) = mean_Trace_max_RS(i);
    bb(i,2) = mean_Trace_max_HC(i);
    bb(i,3) = mean_Trace_max_GA(i);
end

bb(all(bb == 0,2),:) = [];

X = categorical({'100','1000','10000'});
X = reordercats(X,{'100','1000','10000'});
b = bar(X,bb)



xtips1 = b(1).XEndPoints;
ytips1 = b(1).YEndPoints;
labels1 =
string([num2str(round(b(1).YData(1)));num2str(round(b(1).YData(2)));num2str(round(b(1).YData(3)))
]);
text(xtips1,ytips1,labels1,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

xtips2 = b(2).XEndPoints;
ytips2 = b(2).YEndPoints;
labels2 =
string([num2str(round(b(2).YData(1)));num2str(round(b(2).YData(2)));num2str(round(b(2).YData(3)))
]);
text(xtips2,ytips2,labels2,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')
```

```
xtips3 = b(3).XEndPoints;
ytips3 = b(3).YEndPoints;
labels3 =
string([num2str(round(b(3).YData(1)));num2str(round(b(3).YData(2)));num2str(round(b(3).YData(3)))
]);
text(xtips3,ytips3,labels3,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

l{1}='Random Search'; l{2}='Hill Climber'; l{3}='Genetic Algorithm';
legend(b,l,'Location','Northwest');
savefig('barchart.fig')
```