% Course: Nonlinear Optimization. %

% FALL.2018. Dr. Cheng. %

% Assignment: (8) %

% Date:(2018.10.31) %

% By: 卢博 %

% ID NUMBER: 11849159 %

% LAB:(8) %

%

1. Genetic Algorithms with Floating-PointRepresentation

clear;clc;  
f = @(x) (1-x(1)).^2 + 100\*(x(2) - x(1).^2).^2;  
[x,y] = fga(f,10,-2,2,0.8,0.02,2,5000)

x =  
 0.9057 0.8319  
y =  
 0.0223

# Genetic Algorithms with Floating-Point Representation

function [xbest,fbest] = fga(fun,N,lb,ub,pc,pm,L,k\_max)  
pm0 = pm;  
P = initialize\_population(N,L,lb,ub); %% P is the poluation  
[F,xbest,fbest] = evaluate(fun,N,P); %% F is the fitness of the individuals in P  
k = 0;  
while k < k\_max  
 Ptemp = selection(P,F);  
 P = crossover(Ptemp,pc);  
 P = mutation(P,pm);  
 P(1,:) = xbest;  
 [F,xbest,fbest] = evaluate(fun,N,P,fbest);  
 pm = adjust\_mutation\_rate(P,k,k\_max,pm,pm0);  
 k = k+1;  
end

function P = initialize\_population(N,L,lb,ub)  
P = zeros(N,L);  
for i = 1:N  
 P(i,:) = (lb-ub)\*rand(1,L) - (lb-ub)/2;  
end

function [F,xbest,fbest] = evaluate(fun,N,P,varargin)  
j0 = 1; fbest = Inf;  
if nargin>3  
 j0 = 2;  
 xbest = P(1,:);  
 fbest = varargin{1};  
 F(1) = fbest;  
end  
for j = j0:N  
 F(j) = feval(fun,P(j,:));  
 if F(j) < fbest  
 xbest = P(j,:);  
 fbest = F(j);  
 end  
end  
end

function R = selection(P,F)  
[N,L] = size(P);  
R = zeros(2\*N,L);  
for j = 1:2\*N  
 c = ceil(N\*rand(1,2)); %% random values [c1 c2] ci=1~N  
 if F(c(1)) < F(c(2))  
 R(j,:) = P(c(1),:);  
 else  
 R(j,:) = P(c(2),:);  
 end  
end

function R = crossover(P,pc)  
[N,L] = size(P);  
R = [];  
for j = 1:N/2  
 if rand() < pc  
 c = rand(1,L); %% P{2\*j} and P(2\*j-1} are a pair of parents  
 for k = 1:L  
 R(j,k) = P(j,:)\*[c(k),(1-c(k))]';  
 end  
 else  
 R(j,:) = P(2\*j,:);  
 end  
end

function P = mutation(P,pm)  
[N,L] = size(P);  
x\_max = max(P);  
r = rand();  
for j = 1:N %% Go through all genes in each individual for the  
 for k = 1:L %% entire population  
 if rand() < pm  
 P(j,k) = (x\_max(k)-P(j,k))\*(2\*abs(r-0.5))^1.2; %%  
 end  
 end  
end

function pm = adjust\_mutation\_rate(P,k,k\_max,pm,pm0)  
S = population\_diversity(P); %% check population diversity  
if k < k\_max/2  
 if S < 0.1  
 pm = pm\*1.3;  
 else  
 pm = pm/1.2;  
 end  
else  
 pm = pm0\*(2\*(k\_max-k)/k\_max)^2;  
end

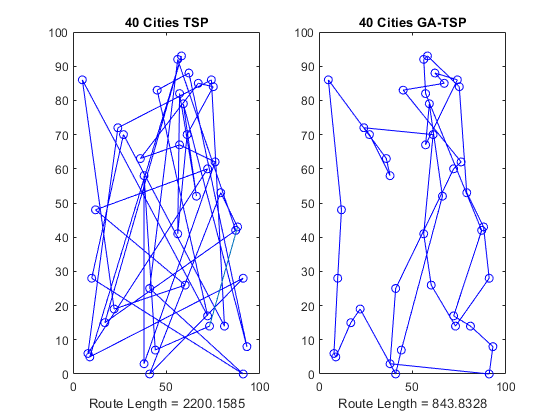
function S = population\_diversity(P)  
[N,L] = size(P);  
for j = 1:L  
 for k = 1:N  
 temp(k) = P(k,j);  
 end  
 R(j) = std(temp); %% standard deviation of jth gene  
end  
S = sum(R)/L;

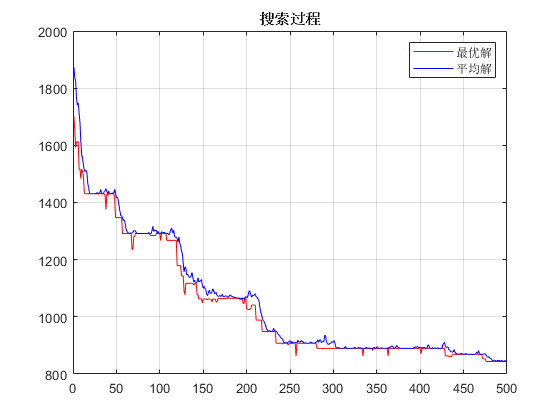
[*Published with MATLAB® R2018a*](https://www.mathworks.com/products/matlab)

1. Evolutionary Algorithm for TSP

function GA\_TSP  
clear;clc  
CityNum=40;  
[dislist,P]=initialize\_population(CityNum);  
  
N=30; %初始种群大小  
kmax=500; %最大代数  
pc=0.8; %交叉概率  
pm=0.02; %变异概率  
  
%产生初始种群  
s=zeros(N,CityNum);  
for i=1:N  
 s(i,:)=randperm(CityNum);  
end  
[~,p]=evaluate(s,dislist);  
  
k=1;  
ymean=zeros(k,1);  
ymax=zeros(k,1);  
xmax=zeros(N,CityNum);  
scnew=zeros(N,CityNum);  
smnew=zeros(N,CityNum);  
while k < kmax+1  
 for j=1:2:N  
 seln=selection(p); %选择操作  
 scro=crossover(s,seln,pc); %交叉操作  
 scnew(j,:)=scro(1,:);  
 scnew(j+1,:)=scro(2,:);  
 smnew(j,:)=mutation(scnew(j,:),pm); %变异操作  
 smnew(j+1,:)=mutation(scnew(j+1,:),pm);  
 end  
 s=smnew; %产生了新的种群  
 [f,p]=evaluate(s,dislist); %计算新种群的适应度  
 %记录当前代最好和平均的适应度  
 [fmax,nmax]=max(f);  
 ymean(k)=1000/mean(f);  
 ymax(k)=1000/fmax;  
 %记录当前代的最佳个体  
 x=s(nmax,:);  
 xmax(k,:)=x;  
 k=k+1;  
end  
[min\_ymax,index]=min(ymax);  
subplot(122)  
drawTSP(P,xmax(index,:),min\_ymax);  
  
figure;  
plot(ymax,'r'); hold on;  
plot(ymean,'b');grid;  
title('搜索过程');  
legend('最优解','平均解');  
fprintf('遗传算法得到的最短距离:%.2f\n',min\_ymax);  
fprintf('遗传算法得到的最短路线');  
disp(xmax(index,:));  
end  
  
%城市位置坐标  
function [DLn,P]=initialize\_population(n)  
P = randi([0 100], n, 2);  
DLn=zeros(n,n);  
for i=1:n  
 for j=1:n  
 DLn(i,j)=((P(i,1)-P(j,1))^2+(P(i,2)-P(j,2))^2)^0.5;  
 end  
end  
subplot(121)  
nor\_path(P);  
end  
  
%计算所有种群的适应度  
function [f,p]=evaluate(s,dislist)  
  
N=size(s,1); %读取种群大小  
f=zeros(N,1);  
for i=1:N  
 f(i)=CalDist(dislist,s(i,:)); %计算函数值，即适应度  
end  
f=1000./f'; %取距离倒数  
%根据个体的适应度计算其被选择的概率  
fsum=0;  
for i=1:N  
 fsum=fsum+f(i)^15;% 让适应度越好的个体被选择概率越高  
end  
ps=zeros(N,1);  
for i=1:N  
 ps(i)=f(i)^15/fsum;  
end  
  
%计算累积概率  
p=zeros(N,1);  
p(1)=ps(1);  
for i=2:N  
 p(i)=p(i-1)+ps(i);  
end  
p=p';  
end  
  
%"选择"操作  
function seln=selection(p)  
  
seln=zeros(2,1);  
%从种群中选择两个个体，最好不要两次选择同一个个体  
for i=1:2  
 r=rand; %产生一个随机数  
 prand=p-r;  
 j=1;  
 while prand(j)<0  
 j=j+1;  
 end  
 seln(i)=j; %选中个体的序号  
 if i==2&&j==seln(i-1) %%若相同就再选一次  
 r=rand; %产生一个随机数  
 prand=p-r;  
 j=1;  
 while prand(j)<0  
 j=j+1;  
 end  
 end  
end  
end  
  
%"交叉"操作  
function scro=crossover(s,seln,pc)  
  
bn=size(s,2);  
scro(1,:)=s(seln(1),:);  
scro(2,:)=s(seln(2),:);  
if rand() < pc %根据交叉概率决定是否进行交叉操作  
 c1=round(rand\*(bn-2))+1; %在[1,bn-1]范围内随机产生一个交叉位  
 c2=round(rand\*(bn-2))+1;  
 chb1=min(c1,c2);  
 chb2=max(c1,c2);  
 middle=scro(1,chb1+1:chb2);  
 scro(1,chb1+1:chb2)=scro(2,chb1+1:chb2);  
 scro(2,chb1+1:chb2)=middle;  
 for i=1:chb1 % 找出重复的城市并且换回原来的城市  
 while find(scro(1,chb1+1:chb2)==scro(1,i))  
 zhi=find(scro(1,chb1+1:chb2)==scro(1,i));  
 y=scro(2,chb1+zhi);  
 scro(1,i)=y;  
 end  
 while find(scro(2,chb1+1:chb2)==scro(2,i))  
 zhi=find(scro(2,chb1+1:chb2)==scro(2,i));  
 y=scro(1,chb1+zhi);  
 scro(2,i)=y;  
 end  
 end  
 for i=chb2+1:bn  
 while find(scro(1,1:chb2)==scro(1,i))  
 zhi=logical(scro(1,1:chb2)==scro(1,i));  
 y=scro(2,zhi);  
 scro(1,i)=y;  
 end  
 while find(scro(2,1:chb2)==scro(2,i))  
 zhi=logical(scro(2,1:chb2)==scro(2,i));  
 y=scro(1,zhi);  
 scro(2,i)=y;  
 end  
 end  
end  
end  
  
%"变异"操作  
function snnew=mutation(snew,pm)  
  
bn=size(snew,2);  
snnew=snew;  
  
%根据变异概率决定是否进行变异操作，1则是，0则否  
if rand() < pm  
 c1=round(rand\*(bn-2))+1; %在[1,bn-1]范围内随机产生一个变异位  
 c2=round(rand\*(bn-2))+1;  
 chb1=min(c1,c2);  
 chb2=max(c1,c2);  
 x=snew(chb1+1:chb2);  
 snnew(chb1+1:chb2)=fliplr(x);  
end  
end  
  
%适应度函数  
function F=CalDist(dislist,s)  
  
alldis = 0;  
n = size(s,2);  
for i = 1:(n-1)  
 alldis = alldis + dislist(s(i),s(i+1));  
end  
alldis = alldis + dislist(s(n),s(1));  
F = alldis;  
end  
  
%画图  
function drawTSP(Clist,BSF,bsf)  
CityNum=size(Clist,1);  
for i=1:CityNum-1  
 plot([Clist(BSF(i),1),Clist(BSF(i+1),1)],[Clist(BSF(i),2),Clist(BSF(i+1),2)],'bo-');  
 hold on;  
end  
plot([Clist(BSF(CityNum),1),Clist(BSF(1),1)],[Clist(BSF(CityNum),2),Clist(BSF(1),2)],'bo-');  
xlabel(['Route Length = ',num2str(bsf)]) % 显示最后总距离  
title([num2str(CityNum),' Cities GA-TSP']);  
hold off;  
end  
  
function distance = nor\_path(dots)  
axis([0 100 0 100])  
n = length(dots); % n个点  
plot(dots(:,1), dots(:,2), 'bo');  
hold on  
distance = 0; % 总距离此时为0  
for i = 1:n-1 % 起点到终点画n-1条线  
 x1 = dots(i,1);  
 y1 = dots(i,2);  
 x2 = dots(i+1,1);  
 y2 = dots(i+1,2);  
 distance = distance + norm([x1 y1] - [x2 y2]);  
 plot([x1,x2],[y1,y2],'bo-');  
% plot(x2,y2,'ro');  
% line([x1 x2],[y1 y2]); % 打点划线  
end  
x0 = dots(1,1);  
y0 = dots(1,2);  
xf = dots(end,1);  
yf = dots(end,2);  
line([x0 xf],[y0 yf]) % 连接终点和起始点  
distance = distance + norm([x0 y0] - [xf yf]);  
xlabel(['Route Length = ',num2str(distance)]) % 显示最后总距离  
title([num2str(n),' Cities TSP']);  
end

遗传算法得到的最短距离:843.83  
遗传算法得到的最短路线 Columns 1 through 13  
  
 37 11 26 14 23 8 29 9 6 32 2 1 4  
  
 Columns 14 through 26  
  
 40 20 30 31 16 13 38 34 22 19 12 17 5  
  
 Columns 27 through 39  
  
 7 21 3 39 28 27 35 15 25 33 24 36 18  
  
 Column 40  
  
 10





[*Published with MATLAB® R2018a*](https://www.mathworks.com/products/matlab)