Source Code

**% Optimizing Travelling Salesperson problem using genetic algorithm approach.**

**% Taking Inputs from the user.**

prompt = "Enter the size of the population: "; %Taking the number of instances to be created.

population = input(prompt);

prompt = "Enter the number of cities: "; %Taking the number of cities in the problem.

length = input(prompt);

prompt = "Enter the crossover probability[0-1]: "; %Taking the crossover probability.

crossP = input(prompt);

prompt = "Enter the mutation probability[0-1]: ";

mutP = input(prompt); %Taking the mutation probability.

tempPop = zeros((2\*population),length);

fitarr = zeros(2\*population,1); %Initialize fitness array.

convarr=[];

c=[];

d=[];

count=0;

flag=1;

**% Initializing the current population.**

currentpop = zeros(population,length);

for i=1:population

currentpop(i,:) = randperm(length,length);

end

% fprintf("The Current Population is: \n");

% disp(currentpop);

**%Initializing the adjacency matrix of the graph.**

distmat=[]; %Inputs are taken here i.e. city distances.

distmat=findAdj(distmat);

% for i=1:length

% for j=1:length

% if i==j

% distmat(i,j)=0;

% end

% if i<j

% k=randi(10);

% distmat(i,j)= k\*i+j;

% end

% if i>j

% distmat(i,j)=distmat(j,i);

% end

% end

% end

fprintf("The Adjacency matrix of the graph:\n");

disp(distmat);

**%Creating the matingPool.**

matingPool = zeros(population,length);

while(flag~=0)

for j=1:population

parent1=0;

parent2=0;

while(parent1==parent2)

parent1 = randi(population);

parent2 = randi(population);

end

p1chromosome = currentpop(parent1,:);

p2chromosome = currentpop(parent2,:);

p1fitval = fitnessValue(p1chromosome,length,distmat);

p2fitval = fitnessValue(p2chromosome,length,distmat);

if p1fitval < p2fitval

matingPool(j,:) = currentpop(parent2,:);

else

matingPool(j,:) = currentpop(parent1,:);

end

end

%fprintf("The mating pool:\n");

%disp(matingPool);

**%Performing crossover operation.**

for i = 1:population

prob = unifrnd(0,1);

if prob <= crossP

parent1=0;

parent2=0;

while(parent1==parent2)

parent1 = randi(population);

parent2 = randi(population);

end

p1chromosome = matingPool(parent1,:);

p2chromosome = matingPool(parent2,:);

crossoverPoint = randi([2,length-1]);

offspring1 = [p1chromosome(1:crossoverPoint),p2chromosome(crossoverPoint+1:length)];

offspring2 = [p2chromosome(1:crossoverPoint),p1chromosome(crossoverPoint+1:length)];

offspring1 = unique(offspring1,'stable');

offspring2 = unique(offspring2,'stable');

temp = setdiff(p2chromosome,offspring1);

temp=shuffle(temp);

offspring1 = [offspring1,temp];

temp = setdiff(p1chromosome,offspring2);

temp=shuffle(temp);

offspring2 = [offspring2,temp];

matingPool(parent1,:) = offspring1;

matingPool(parent2,:) = offspring2;

end

end

% fprintf("The new matingpool after crossover:\n");

% disp(matingPool);

**% Performing the mutation operation.**

for k=1:population

prob = unifrnd(0,1);

if prob <= mutP

parent1 = randi(population);

p1chromosome = matingPool(parent1,:);

crossPoint = round(unifrnd(1,(length-1)));

child1 = [flip(p1chromosome(1:crossPoint)),p1chromosome(crossPoint+1:end)];

matingPool(parent1,:) = child1;

end

end

% fprintf("The new matingpool after mutation:\n");

% disp(matingPool);

**%Creating a temporary population with current population and matingPool.**

tempPool = currentpop;

j=1;

for i = (population+1):(2\*population)

tempPool(i,:) = matingPool(j,:);

j=j+1;

end

% fprintf("The new matingpool after combine:\n");

% disp(tempPool);

% Computing the fitness value.

for i=1:(2\*population)

fitarr(i,:)=fitnessValue(tempPool(i,:),length,distmat);

end

% fprintf("The fitness function of the population: \n");

% disp(fitarr);

count=count+1;

avg=findmean(fitarr,(2\*population));

d=[d,avg];

c=[c,count];

fprintf("The minimum distance is: %d(%d)\n",min(fitarr),count);

convarr = [convarr,min(fitarr)];

plot(c,convarr,c,d,'.');drawnow

title('Blue: Minimum Green: Average');

xlabel("Generation");

ylabel("Min Dist");

% convarr

**%Converging the solution.**

if count>500

for i=count:-1:(count-500)

if min(fitarr)==convarr(i)

flag=0;

else

flag=1;

end

end

if flag == 0

break;

end

end

for i=1:(population)

[M,I] = min(fitarr);

currentpop(i,:)=tempPool(I,:);

fitarr(I)=10000000;

end

% for i=(population/2+1):population

% [M,I] = max(fitarr);

% currentpop(i,:)=tempPool(I,:);

% fitarr(I)=0;

% end

end

fprintf("The optimal solution is:\n");

disp(tempPool(I,:));

fprintf("No of generations taken: \n");

disp(count);

% plot(c,convarr,c,d,'.');

% title('Blue: Minimum Green: Average');

% xlabel("Generation");

% ylabel("Min Dist");

**%fitness calculation function.**

function [fitval]=fitnessValue(array,length,distmat)

fitval=0;

for i=1:length-1

fitval=fitval+distmat(array(i),array(i+1));

end

fitval=fitval+distmat(array(length),array(1));

end

function [temp]=shuffle(array)

[m,n]=size(array);

idx=randperm(n);

b=array;

b(1,idx)=array;

temp=b;

end

function [mean]=findmean(array,length)

sum=0;

for i=1:length

sum=sum+array(i,1);

end

mean=sum/length;

end

**%Find adjacency matrix from (x,y) coordinates.**

function [output]=findAdj(distmat)

[m,n]=size(distmat);

r=1;

output=zeros(m,m);

for i=1:m

for j=1:m

output(i,j)=round(power(power((distmat(r,2)-distmat(j,2)),2)+power((distmat(r,3)-distmat(j,3)),2),.5));

end

r=r+1;

end

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Source Code**

% Optimizing the Vehicle Routing Problem (One depot)using GA.

%Taking the number of instances to be created.

prompt = "Enter the size of the population: ";

population = input(prompt);

prompt = "Enter the Number of vehicles: ";

vehicle = input(prompt);

prompt = "Enter the Crossover Probability: ";

crossP = input(prompt);

prompt = "Enter the Mutation Probability: ";

mutP = input(prompt);

fitarr = zeros(2\*population,1);

count = 0;

convarr=[];

c=[];

d=[];

flag = 1;

distmat = [

1 82 76

2 96 44

3 50 5

4 49 8

5 13 7

6 29 89

7 58 30

8 84 39

9 14 24

10 2 39

11 3 82

12 5 10

13 98 52

14 84 25

15 61 59

16 1 65

17 88 51

18 91 2

19 19 32

20 93 3

21 50 93

22 98 14

23 5 42

24 42 9

25 61 62

26 9 97

27 80 55

28 57 69

29 23 15

30 20 70

31 85 60

32 98 5

];

distmat = findAdj(distmat);

distmat(1,1) = 99999999;

fprintf("The Adjacency Matrix: \n");

disp(distmat);

%Initializing the current population.

[row,col] = size(distmat);

len = col + (vehicle - 1);

currentpop = zeros(population, len);

temp = ones(1,vehicle - 1);

%increasing the size of chromosome to #cities+#vehicles.

for i = 1:population

cities = randperm(col);

currentpop(i,:) = [cities,temp];

end

[m,n] = size(currentpop);

%generating the chromosomes and check if valid.

%m is #rows and n is #cols.

for i = 1:m

%the below code is to make the currentpop shuffle.

idx = randperm(n);

currentpop(i,idx) = currentpop(i,:);

%if the last value is 1 in chromosome.

if(currentpop(i,n) == 1)

currentpop(i,:) = flip(currentpop(i,:));

end

%if last and first both are 1 in chromosome.

if((currentpop(i,n) == 1) && (currentpop(i,1) == 1))

for j = (n-1):-1:1

%swapping with the first non zero element from the last.

if(currentpop(i,j) ~= 1)

currentpop(i,:) = swapArrayEl(currentpop(i,:), j, n);

end

end

end

%if the first value if not 1 in chromosome.

if(currentpop(i,1) ~= 1)

for j = 1:n

if(currentpop(i,j) == 1)

%swapping first one from left with the first element.

currentpop(i,:) = swapArrayEl(currentpop(i,:), j, 1);

break;

end

end

end

%if two 1 occurs simultenously.

for k = 1:(n-1)

if((currentpop(i,k) == 1) && (currentpop(i,(k+1)) == 1))

ind = k + 1;

%if ind is last element(i.e., 1) then leave it.

if (ind ~= n)

%if more than one 1 is present go to the 1st non-one value.

while(currentpop(i,ind) == 1)

ind = ind + 1;

end

currentpop(i,:) = swapArrayEl(currentpop(i,:), ind, k+1);

end

end

end

%if still last value is 1 then

if(currentpop(i,n) == 1)

mid = floor(n/2);

currentpop(i,:) = swapArrayEl(currentpop(i,:), mid, n);

end

end

% fprintf("The current population: \n");

% disp(currentpop);

%Creating mating Pool.

matingPool = zeros(population,len);

while(flag~=0)

for j=1:population

parent1=0;

parent2=0;

while(parent1==parent2)

parent1 = randi(population);

parent2 = randi(population);

end

p1chromosome = currentpop(parent1,:);

p2chromosome = currentpop(parent2,:);

p1fitval = fitnessValue(p1chromosome,len,distmat);

p2fitval = fitnessValue(p2chromosome,len,distmat);

if p1fitval < p2fitval

matingPool(j,:) = currentpop(parent1,:);

else

matingPool(j,:) = currentpop(parent2,:);

end

end

% fprintf("The mating pool: \n");

% disp(matingPool);

%Performing crossover operation.

for i = 1:1

prob = unifrnd(0,1);

if prob <= crossP

parent1=0;

parent2=0;

while(parent1==parent2)

parent1 = randi(population);

parent2 = randi(population);

end

p1chromosome = matingPool(parent1,:);

p2chromosome = matingPool(parent2,:);

offspring1 = exchangeCross(p1chromosome, p2chromosome, n);

offspring2 = exchangeCross(p2chromosome, p1chromosome, n);

matingPool(parent1,:) = offspring1;

matingPool(parent2,:) = offspring2;

end

end

% fprintf("Mating Pool after crossover: \n");

% disp(matingPool);

%Performing Mutation operation.

for k=1:population

prob = unifrnd(0,1);

if prob <= mutP

parent = randi(population);

p1chromosome = matingPool(parent,:);

point1 = 1;

point2 = 1;

while(p1chromosome(point1) == 1 || p1chromosome(point2) == 1 || point1 == point2)

point1 = round(unifrnd(2,len));

point2 = round(unifrnd(2,len));

end

p1chromosome = swapArrayEl(p1chromosome, point1, point2);

matingPool(parent,:) = p1chromosome;

end

end

% fprintf("MatingPool after the mutation: \n");

% disp(matingPool);

%Creating a temporary population with current population and matingPool

tempPool = currentpop;

j=1;

for i = (population+1):(2\*population)

tempPool(i,:) = matingPool(j,:);

j=j+1;

end

% Computing the fitness value.

for i=1:(2\*population)

fitarr(i,:)=fitnessValue(tempPool(i,:),len,distmat);

end

count=count+1;

avg=findmean(fitarr,(2\*population));

d=[d,avg];

c=[c,count];

fprintf("The minimum distance is: %d(%d)\n",min(fitarr),count);

convarr = [convarr,min(fitarr)];

plot(c,convarr,c,d,'.');drawnow

title('Blue: Minimum Green: Average');

xlabel("Generation");

ylabel("Min Dist");

%converging the solution.

if (count > 100)

for i=count:-1:(count-100)

if min(fitarr)==convarr(i)

flag=0;

else

flag=1;

end

end

if flag == 0

break;

end

end

for i=1:(population)

[M,I] = min(fitarr);

currentpop(i,:)=tempPool(I,:);

fitarr(I)=10000000;

end

end

fprintf("The optimal solution is:\n");

disp(tempPool(I,:));

fprintf("No of generations taken: \n");

disp(count);

%all required user defined functions.

%function to exchange chromosome in crossover.

function[tempchromosome] = exchangeCross(p1chromosome, p2chromosome, len)

tempchromosome = p1chromosome;

p1left = 2;

p2left = 2;

while(p1left <= len)

if(p2chromosome(p2left) ~= 1 && tempchromosome(p1left) ~= 1)

tempchromosome(p1left) = p2chromosome(p2left);

p1left = p1left + 1;

p2left = p2left + 1;

elseif (tempchromosome(p1left) == 1)

p1left = p1left + 1;

elseif(p2chromosome(p2left) == 1)

p2left = p2left + 1;

end

end

end

%function to swap elements in 1D array.

function[array] = swapArrayEl(array, point1, point2)

temp = array(point1);

array(point1) = array(point2);

array(point2) = temp;

end

%function to calculate adjacency matrix.

function [output] = findAdj(distmat)

[m,n]=size(distmat);

r=1;

output=zeros(m,m);

for i=1:m

for j=1:m

output(i,j) = round(power(power((distmat(r,2)-distmat(j,2)),2)+power((distmat(r,3)-distmat(j,3)),2),.5));

end

r=r+1;

end

end

%calculating fitness value of each chromosome.

function [fitval]=fitnessValue(array,length,distmat)

fitval=0;

for i=1:length-1

fitval=fitval+distmat(array(i),array(i+1));

end

fitval=fitval+distmat(array(length),array(1));

end

%finding the mean of the array.

function [mean]=findmean(array,length)

sum=0;

for i=1:length

sum=sum+array(i,1);

end

mean=sum/length;

end