clear all;

clc;

C0=load('c\_23.csv'); % depot+clients

C0Size=size(C0);

C=C0(2:C0Size(1),:);

C\_N=size(C,1);

V\_N=2;

% define C\_N\*C\_N matrix to store the distance between tow cities.

D\_C=[];

D\_C=load('d\_100.csv');

max\_capacity=2000;

% for i=1:size(C0,1)

% for j=1:size(C0,1)

% %D\_C(i,j)=i+j;

% D\_C(i,j)=sqrt((C0(i,2)-C0(j,2)).^2+(C0(i,3)-C0(j,3)).^2);

% end

% end

% initialize

popSize=200;

empty\_pop.routes=zeros(V\_N,C\_N);

empty\_pop.clients=[];

empty\_pop.distances=[];

empty\_pop.total\_distance=0;

pop=repmat(empty\_pop,popSize,1);

for i=1:popSize

pop(i)=completion(pop(i),C\_N,V\_N,C,D\_C,max\_capacity);

end

%% sort pop by totalDistance from min to max

for i=1:popSize

for j=1:i

if pop(j).total\_distance>pop(i).total\_distance

temp=pop(i);

pop(i)=pop(j);

pop(j)=temp;

end

end

end

%% process

pc=0.8;

pm=0.08;

max\_NFC=200;

numberofcrossover=round(pc\*popSize);

numberofmutation=round(pm\*popSize);

numberofrec=popSize-(numberofcrossover+numberofmutation);

for i=1:max\_NFC

pop2=repmat(empty\_pop,0,1);

% selecting population for crossover

selected=repmat(empty\_pop,numberofrec,1);

for j=1:numberofrec

selected(j)=pop(j);

pop2(end+1)=pop(j);

end

%% crossover

c=randperm(popSize);

individualsForCross=repmat(empty\_pop,numberofcrossover,1);

for j=1:numberofcrossover

individualsForCross(j)=pop(c(j));

end

% crossover of parent1 and parent2

offspringset=repmat(empty\_pop,0,1);

for z=1:numberofcrossover

n\_parent1=ceil(numberofcrossover\*rand);

n\_parent2=ceil(numberofcrossover\*rand);

parent1=individualsForCross(n\_parent1);

parent2=individualsForCross(n\_parent2);

crossoverpoint=ceil((C\_N-1)\*rand);

parent12=parent1.clients(1:crossoverpoint);

parent21=parent2.clients(1:crossoverpoint);

s1=[];

s2=[];

for j=1:numel(parent1.clients)

if sum(parent1.clients(j)~=parent21)==numel(parent21)

s2=[s2 parent1.clients(j)];

end

if sum(parent2.clients(j)~=parent12)==numel(parent12)

s1=[s1 parent2.clients(j)];

end

end

offspring1=repmat(empty\_pop,1,1);

offspring2=repmat(empty\_pop,1,1);

offspring1.clients=[parent12 s1];

offspring2.clients=[parent21 s2];

% evaluate fitness

offspring1=completion(offspring1,C\_N,V\_N,C,D\_C,max\_capacity);

offspring2=completion(offspring2,C\_N,V\_N,C,D\_C,max\_capacity);

offspringset(end+1)=offspring1;

offspringset(end+1)=offspring2;

pop2(end+1)=offspring1;

pop2(end+1)=offspring2;

end

%% Mutation

% selecting a parent for mutation

m=randperm(popSize);

individualsForMutation=repmat(empty\_pop,numberofmutation,1);

for j=1:numberofmutation

individualsForMutation(j)=pop(m(j));

end

%doing the mutation

Mutatedset=repmat(empty\_pop,0,1);

for z=1:numberofmutation

parent=individualsForMutation(z).clients;

mutationPoint1=ceil(C\_N\*rand);

mutationPoint2=ceil(C\_N\*rand);

temp=parent(mutationPoint1);

parent(mutationPoint1)=parent(mutationPoint2);

parent(mutationPoint2)=temp;

% Evaluate fitness of mutateed members

mutated\_parent=repmat(empty\_pop,1,1);

mutated\_parent.clients=parent;

mutated\_parent=completion(mutated\_parent,C\_N,V\_N,C,D\_C,max\_capacity);

Mutatedset(end+1)=mutated\_parent;

pop2(end+1)=mutated\_parent;

end

%sort the new population

for z=1:size(pop)

for j=1:z

if pop(j).total\_distance>pop(z).total\_distance

temp=pop(z);

pop2(z)=pop(j);

pop2(j)=temp;

end

end

end

%find the best root

for j=1:popSize

pop(j)=pop2(j);

end

end

best=pop(1);

%% plot

figure;

color=['r','g','b','c','m','y','k','w'];

routes=best.routes;

for i=1:V\_N

route=routes(i,:);

X=C0(1,2);

Y=C0(1,3);

for j=1:numel(route)

if route(j)~=0

X(j+1)=C(route(j),2);

Y(j+1)=C(route(j),3);

end

end

X=[X,C0(1,2)];

Y=[Y,C0(1,3)];

plot(X,Y,'--',"Marker","o",'LineWidth',2);

hold on;

end

plot(C0(1,2),C0(1,3),'\*','Color','b',"LineWidth",10);

text(C0(1,2),C0(1,3),'Depot');

%title("Figure 1");

hold off;

function chromsome=completion(chromsome,C\_N,V\_N,C,D\_C,max\_capacity)

satisfy\_capacity=0;

while satisfy\_capacity==0

x=randperm(C\_N);

chromsome.clients=x;

chromsome.routes=[];

%divide into V\_N routes

indexs=randperm(C\_N,V\_N-1);

indexs\_order=sort(indexs);

d\_sum=0;

flag=0;

for j=1:V\_N

if(j==V\_N)

if indexs\_order(j-1)==C\_N

start\_p=C\_N;

else

start\_p=indexs\_order(j-1)+1;

end

end\_p=C\_N;

else

index=indexs\_order(j); % 5,9

start\_p=1;

end\_p=index;

if j~=1

index\_1=indexs\_order(j-1);

start\_p=index\_1+1;

end

end

s=chromsome.clients(1,start\_p);

e=chromsome.clients(1,end\_p);

d=D\_C(C(s,1),1); % distance from first client to depot

d2=D\_C(C(e,1),1); % distance from last client to depot

for k=start\_p:end\_p-1

C\_index=chromsome.clients(1,k);

C\_index\_next=chromsome.clients(1,k+1);

d=d+D\_C(C(C\_index,1),C(C\_index\_next,1));

end

chromsome.distances(j)=d;

d\_sum=d\_sum+d+d2;

%vehicle route

r=[];

%vehicle capacity

demand=0;

for n=start\_p:end\_p

C\_index=chromsome.clients(1,n);

r=[r,C\_index];

demand=demand+C(C\_index,4);

end

if(demand>max\_capacity)

break;

else

flag=flag+1;

end

for m=1:numel(r)

chromsome.routes(j,m)=r(m);

end

end

chromsome.total\_distance=d\_sum;

% check the demand

if(flag==V\_N)

satisfy\_capacity=1;

break;

end

end

end