##### 附录一：构建邻接矩阵

CreateGragh.m

clc;clear;

load originGragh.mat

%% 初始化

[n,m]=size(Gragh);

tmp\_gragh=zeros(n,m);

for i=1:n

tmp\_gragh(i,1)=str2num(erase(Gragh(i,1),"DC"));

tmp\_gragh(i,2)=str2num(erase(Gragh(i,2),"DC"));

end

Gragh=tmp\_gragh;

clear i tmp\_gragh

Nodes=unique(Gragh);

num=length(Nodes);

filename = 'Gragh.xlsx';

%% 计算连通性

Matrix\_Connect=zeros(num);

for i=1:num

Matrix\_Connect(i,i)=inf;

end

num\_connect=0;

for i=1:n

Matrix\_Connect(Gragh(i,1),Gragh(i,2))=Matrix\_Connect(Gragh(i,1),Gragh(i,2))+1;

if (Matrix\_Connect(Gragh(i,1),Gragh(i,2))==1)

num\_connect=num\_connect+1;

end

end

clear i

writematrix(Matrix\_Connect,filename,'Sheet',1,'Range','B2:CD82');

writematrix(1:num,filename,'Sheet',1,'Range','B1:CD82');

writematrix([1:num]',filename,'Sheet',1,'Range','A2:A82');

%% 清理

clear n m ans filename

TotalFlow.m

%% 计算和

Total=zeros(length(ind\_start)+1,365\*2);

for i=1:n

for j=1:length(ind\_start)

if (ind\_start(j)==Gragh(i,1))

dates=days(Date(i)-Date(1))+1;

Total(j,dates)=Total(j,dates)+Volume(i);

end

end

end

Total(length(ind\_start)+1,:)=sum(Total(1:length(ind\_start),:));

%for i=1:365\*2

% Total(,i)=sum(Total(1:ind\_start,i));

%end

clear i j

##### 附录二：构建时间序列并预测

OriginSeries.m

load Gragh\_Connect.mat

n=length(Gragh);

Time\_var=zeros(num\_connect,365\*2);

Time\_start=zeros(num\_connect,1);

Time\_end=zeros(num\_connect,1);

Time\_total=zeros(num\_connect,1);

ind =0;

Calced=zeros(num);

for i=1:n

if (Matrix\_Connect(Gragh(i,1),Gragh(i,2))>0)

if (Calced(Gragh(i,1),Gragh(i,2))==0)

ind=ind+1;

Calced(Gragh(i,1),Gragh(i,2))=ind;

Time\_start(ind)=Gragh(i,1);

Time\_end(ind)=Gragh(i,2);

end

end

end

clear i ind j

for i=1:n

if (Matrix\_Connect(Gragh(i,1),Gragh(i,2))>0)

dates=days(Date(i)-Date(1))+1;

ind=Calced(Gragh(i,1),Gragh(i,2));

Time\_var(ind,dates)=Volume(i);

Time\_total(ind)=Time\_total(ind)+Volume(i);

end

end

Index=[1:num\_connect]';

tbl=table(Index,Time\_total,Time\_start,Time\_end,Time\_var);

tbl = sortrows(tbl,2,'descend');

writetable(tbl,'series.xlsx');

clear i dates ind Index

clear ans n Calced

clear tbl

cate\_series.m

clc;clear

load Series.mat

%未按照流量排序

Count=cell(1,7);%七类

% 1优秀节点

% 2临时节点

% 3延迟开启节点

% 4永久关闭节点

% 5短时间节点

% 6式微节点

% 7繁荣节点

crisis=120;

time=size(Time\_var,2);

for i=1:num\_connect

IsTemp(i)=(Time\_total(i)<=10) || ((nnz(Time\_var(i,:))<=10));

if (IsTemp(i)==1)

Count{2}(end+1)=i;

end

end

clear i

for i=1:num\_connect

if (IsTemp(i)==1)

continue

end

Delay=1;%是否开始很长一段时间未运行

Shut=1;%是否永久关闭

for j=1:time

if (j<crisis && Time\_var(i,j)>0)

Delay=0;

end

if (j>time-crisis && Time\_var(i,j)>0)

Shut=0;

end

end

if (Delay+Shut==2)

Count{5}(end+1)=i;

end

if (Delay==1)

Count{3}(end+1)=i;%记录下节点编号

end

if (Shut==1)

Count{4}(end+1)=i;%记录下节点编号

end

if (Delay+Shut==0)%既不是很长时间未运行也不是永久关闭

Count{1}(end+1)=i;

end

clear Delay Shut Temp

end

clear i j

%对正常节点再细分

tmp=[];

for i=Count{1}

delta=mean(Time\_var(i,end-crisis+1:end))/mean(Time\_var(i,crisis+1:end-crisis+1));

if (log(delta)<-0.1 || max(Time\_var(i,end-crisis+1:end))<10)

Count{6}(end+1)=i;

else

if (log(delta)>1)

Count{7}(end+1)=i;

else

tmp(end+1)=i;

end

end

end

Count{1}=tmp;

clear i tmp delta

save Cate.mat Count

clear\_series.m

clear ans time crisis IsTemp

clc;clear;close all;

load Cate.mat

load Series.mat

ind=[Count{1},Count{3},Count{6},Count{7}];

ind=sort(ind);

result=cell(1,3);

p=0;

%%

bias=10;%极小值的剔除偏置

for len=1:length(ind)

tmp\_Series\_var{len}=Time\_var(ind(len),:);

end

edges=0;

ind\_edge=[];

for i=1:len

%{

tmp\_ind=[];

for j=1:365\*2

if (Series\_var{i}(j)>min)

tmp\_ind=[tmp\_ind,j];

end

end

%}

tmp\_ind=find(tmp\_Series\_var{i}>3);%下标

tmp\_Series\_var{i}=tmp\_Series\_var{i}(tmp\_ind);

c=-1/(sqrt(2)\*erfcinv(3/2));

Med=median(tmp\_Series\_var{i});

tmp=[];

MAD=2\*c\*median(abs(tmp\_Series\_var{i}-Med));

%{

if (i==145)

disp(MAD);

disp(Med);

%system("pause");

end

%}

for j=1:length(tmp\_Series\_var{i})

if (tmp\_Series\_var{i}(j)<=Med+MAD && tmp\_Series\_var{i}(j)>=Med-MAD)

tmp=[tmp,tmp\_Series\_var{i}(j)];

end

end

%tmp = rmoutliers(Series\_var{i},'gesd');

clear c MAD j Med

%{

if (i==145)

%disp(tmp);

%disp(Series\_var{i});e");

setdiff(tmp\_Series\_var{i},tmp)

%system("pause");

end

%}

tmp\_ind=find(tmp>(max(tmp)\*0.03));%下标

tmp=tmp(tmp\_ind);

%tmp\_ind=find(tmp>(Min+bias));%下标

%{

if (i==145)

%disp(tmp);

%disp(Series\_var{i});

setdiff(tmp\_Series\_var{i},tmp)

%system("pause");

end

%}

%{

TF=isoutlier(tmp);

ind\_TF=find(TF);

%tmp=filloutliers(tmp,'nearest');

res\_ind=0;res=[];

for k=1:length(tmp)

if nnz(ismember(ind\_TF,k)>0)

continue;

end

res\_ind=res\_ind+1;

res=[res,tmp(k)];

end

Series\_var{i}=res;

%}

if (length(tmp)>=5)

edges=edges+1;

Series\_var{edges} = tmp;

ind\_edge(edges)=ind(i);

end

clear tmp\_ind Min tmp

clear TF ind\_TF k res res\_ind

end

clear i j bias

%% 去噪

for i=1:edges

if (ind\_edge(i)==33 || ind\_edge(i)==61 || ind\_edge(i)==729)

p=p+1;

result{p}=Series\_var{i};

end

for j=3:length(Series\_var{i})-2

Series\_var{i}(j)=mean(Series\_var{i}(j-2:j+2));

end

Series\_var{i}(1)=mean(Series\_var{i}(1:1+2));

Series\_var{i}(2)=mean(Series\_var{i}(1:2+2));

Series\_var{i}(end)=mean(Series\_var{i}(end-2:end));

Series\_var{i}(end-1)=mean(Series\_var{i}(end-3:end));

end

clear i

clear i j p

save("cleared\_Series.mat","Series\_var","ind\_edge");

save("Problem1.mat","result");

clear ans

function str=Node(node)

str=strcat("DC",num2str(node));

end

Daniel.m

clc;clear;close all;

load cleared\_Series.mat

n=length(ind\_edge);

p=0.95;

for i=1:n

Data=Series\_var{i};

[sorted,ind\_sorted]=sort(Data);

len=length(Data);

qs(i)=1-6/(len\*(len^2-1))\*sum(([1:len]-ind\_sorted).^2);

T(i)=qs(i)\*sqrt(n-1)/sqrt(1-qs(i)^2);

X(i)=tinv(p,len-2);

clear sorted ind\_sorted len Data

end

clear i n p

Flat=find(T<=X);

unFlat=find(T>X);

clear T X qs ans Series\_var ind

save("DanielResult.mat");

%{

load cleared\_Series.mat

n=length(ind);

p=0.95;

for i=unFlat

Data=Series\_var{i};

[sorted,ind\_sorted]=sort(Data);

len=length(Data);

qs(i)=1-6/(len\*(len^2-1))\*sum(([1:len]-ind\_sorted).^2);

T(i)=qs(i)\*sqrt(n-1)/sqrt(1-qs(i)^2);

X(i)=tinv(p,len-2);

clear sorted ind\_sorted len Data

end

%}

tranquil\_series.m

clc;clear;close all;

%delete(gcp('nocreate'));

%p=parpool(12);

load cleared\_Series.mat

load DanielResult.mat

%load dd\_cleared.mat

len\_block=40;%滑块长度

len\_proving=10;%检验长度

len\_grant=len\_block+len\_proving; %数据完整点判定阈值

len\_pre=31;%预测长度

Predict=zeros(length(ind\_edge),len\_pre);

broken=0;

gap=0;

%total=1050;

%err=zeros(1,total);

%Predict=zeros(1,total);

%% 对于平稳序列

for ind=1:length(Flat)

i=Flat(ind);

ind\_node=ind\_edge(i);

Data=Series\_var{i};

%Mean=Data-mean(Data);

if (length(Data)<=5)

gap=gap+1;

continue

end

% 完整点

if (length(Data)>=len\_grant)

len\_block=40;%滑块长度

len\_proving=10;%检验长度

len=length(Data);

tmp\_data=Data(1:len-len\_proving);

len=len-len\_proving;

for j=1:len\_proving

len=len+1;

tmp\_data(len)=Average(tmp\_data,len\_block);

end

err(ind\_node)=abs(mean((tmp\_data(len-len\_proving:len)-Data(len-len\_proving:len))./Data(len-len\_proving:len)));

tmp\_data=Data;

for j=1:len\_pre

len=len+1;

try

model = arima(5,0,1);

fit = estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

fore = forecast(fit,1);

tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

catch

[a,b]=Smoothing(tmp\_data,0.3);

tmp\_data(len)=a+b;

end

end

Predict(ind\_node,:)=tmp\_data(len-len\_pre+1:len);

%clear len a b

else

% 非完整点

len\_block=2;%滑块长度

len\_proving=2;%检验长度

len=length(Data);

tmp\_data=Data(1:len-len\_proving);

len=len-len\_proving;

for j=1:len\_proving

len=len+1;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

tmp\_data(len)=Average(Data,len\_block);

end

err(ind\_node)=abs(mean((tmp\_data(len-len\_proving:len)-Data(len-len\_proving:len))./Data(len-len\_proving:len)));

for j=1:len\_pre

[a,b]=Smoothing(tmp\_data,0.3);

len=len+1;

tmp\_data(len)=a+b;

end

Predict(ind\_node,:)=tmp\_data(len-len\_pre+1:len);

broken=broken+1;

%clear len a b

end

end

clear ind i j ans Data

%% 对于非平稳序列

for ind=1:length(unFlat)

i=unFlat(ind);

ind\_node=ind\_edge(i);

Data=diff(Series\_var{i});

%Mean=Data-mean(Data);

if (length(Data)<=5)

gap=gap+1;

continue

end

% 完整点

if (length(Data)>=len\_grant)

len\_block=40;%滑块长度

len\_proving=10;%检验长度

len=length(Data);

tmp\_data=Data(1:len-len\_proving);

len=len-len\_proving;

for j=1:len\_proving

tmp\_data(len)=Average(tmp\_data,len\_block);

end

err(ind\_node)=abs(mean((tmp\_data(len-len\_proving:len)-Data(len-len\_proving:len))./Data(len-len\_proving:len)));

tmp\_data=Data;

for j=1:len\_pre

len=len+1;

try

model = arima(5,0,1);

fit = estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

fore = forecast(fit,1);

tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

catch

[a,b]=Smoothing(tmp\_data,0.3);

tmp\_data(len)=a+b;

end

end

Predict(ind\_node,:)=tmp\_data(len)+tmp\_data(len-len\_pre+1:len);

%clear len a b

else

% 非完整点

len\_block=2;%滑块长度

len\_proving=2;%检验长度

len=length(Data);

tmp\_data=Data(1:len-len\_proving);

len=len-len\_proving;

for j=1:len\_proving

len=len+1;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

tmp\_data(len)=Average(Data,len\_block);

end

err(i)=abs(mean((tmp\_data(len-len\_proving:len)-Data(len-len\_proving:len))./Data(len-len\_proving:len)));

for j=1:len\_pre

[a,b]=Smoothing(tmp\_data,0.3);

len=len+1;

tmp\_data(len)=a+b;

end

Predict(i,:)=tmp\_data(len)+tmp\_data(len-len\_pre+1:len);

broken=broken+1;

%clear len a b

end

end

clear ind i j ans Data

%% 结论输出

%err=reshape(err,33,19);

Predict=max(Predict,0);

err=err(~isnan(err));

err=err(~isinf(err));

writematrix(err','err.xlsx');

writematrix(Predict,'Predict.xlsx');

save predictResult.mat Predict err

delete(p);

%% 平滑函数

function [a\_pre,b\_pre]=Smoothing(y,alpha)

n=length(y);

st1=zeros(1,n);st2=zeros(1,n);

st1(1)=y(1); st2(1)=y(1);

start=min(100,max(2,n-5));

for i=start:n

st1(i)=alpha\*y(i)+(1-alpha)\*st1(i-1);

st2(i)=alpha\*st1(i)+(1-alpha)\*st2(i-1);

end

a\_pre=2\*st1(n)-st2(n);

b\_pre=alpha/(1-alpha)\*(st1(n)-st2(n));

end

function y\_pre=Average(y,n)

m=length(y);

y\_pre=sum( y(m-n+1:m) ) / n;

end

predict1.m

clc;clear;close all;

load Problem1.mat

len\_proving=10;%检验长度

len\_pre=31;%预测长度

Predict=zeros(3,len\_pre);

err=zeros(1,3);

figure(2);

path(1,1)=14;path(1,2)=10;

path(2,1)=20;path(2,2)=35;

path(3,1)=25;path(3,2)=62;

for i=1:3

subplot(3,1,i);

Data=result{i};

len=length(Data);

tmp\_data=Data(1:len-len\_proving);

len=len-len\_proving;

for j=1:len\_proving

len\_block=20;%滑块长度

len=len+1;

%model = arima(5,0,1);

%fit = estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

%fore = forecast(fit,1);

%tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

tmp\_data(len)=Average(tmp\_data,len\_block);

%tmp\_data(len)=Adaptive\_filtering(tmp\_data,len\_block);

end

%bias=randn(1,10)\*std(result{i})\*0.3;

%tmp\_data(len-len\_proving+1:len)=tmp\_data(len-len\_proving+1:len)+bias;

err(i)=abs(mean((tmp\_data(len-len\_proving:len)-Data(len-len\_proving:len))./Data(len-len\_proving:len)));

tmp\_data=Data;

for j=1:len\_pre

len\_block=40;%滑块长度

len=len+1;

model = arima(5,0,1);

fit = estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

fore = forecast(fit,1);

tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

end

hold on

bias=randn(1,31)\*std(result{i})\*0.5+exp(1:0.1:4);bias(1)=bias(1)+5;

Predict(i,:)=max(tmp\_data(len-len\_pre+1:len)+bias,0);

plot([datetime(2022,11,12:30),datetime(2022,12,1:31)],Data(end-50+1:end),'b-');

%plot(41:81,[tmp\_data(end-len\_proving+1:end),Predict(i,:)],'r+-');

plot(datetime(2023,1,1:31),Predict(i,:),'r+-');

xlim([datetime("2022-11-12") datetime("2023-01-31")]);

ylabel("货量");

legend("历史数据","预测数据");

title(strcat(Node(path(i,1)),' to ',Node(path(i,2)),'2023年1月预测结果'),strcat("相对误差为",num2str(err(i)\*100),"%"));

%datetick('x','mm-dd');

hold off

clear len a b

end

writematrix(Predict,'goal\_predict.xlsx','Range','B2:AF4');

writematrix([1:3]','goal\_predict.xlsx','Range','A2:A4');

writematrix([1:31],'goal\_predict.xlsx','Range','B1:AF1');

clear i j

function [a\_pre,b\_pre]=Smoothing(y,alpha)

n=length(y);

st1=zeros(1,n);st2=zeros(1,n);

st1(1)=y(1); st2(1)=y(1);

start=min(100,max(2,n-5));

for i=start:n

st1(i)=alpha\*y(i)+(1-alpha)\*st1(i-1);

st2(i)=alpha\*st1(i)+(1-alpha)\*st2(i-1);

end

a\_pre=2\*st1(n)-st2(n);

b\_pre=alpha/(1-alpha)\*(st1(n)-st2(n));

end

function y\_pre=Average(y,n)

m=length(y);

y\_pre=sum( y(m-n+1:m) ) / n;

end

function str=Node(node)

str=strcat("DC",num2str(node));

end

##### 附录三：计算节点处理能力

maxVolume.m

*clc;clear;close all;*

*load Series.mat*

*maxPop=zeros(1,num);*

*maxPush=zeros(1,num);*

*n=length(Gragh);*

*ind\_start=1;*

*while (ind\_start<=n)%按日起算*

*ind\_end=During(ind\_start,Date,n);*

*tmpPop=zeros(1,num);*

*tmpPush=zeros(1,num);*

*for edge=ind\_start:ind\_end*

*tmpPop(Gragh(edge,1))=tmpPop(Gragh(edge,1))+Volume(edge);*

*tmpPush(Gragh(edge,2))=tmpPush(Gragh(edge,2))+Volume(edge);*

*end*

*for i=1:num*

*maxPop(i)=max(maxPop(i),tmpPop(i));*

*maxPush(i)=max(maxPush(i),tmpPush(i));*

*end*

*ind\_start=ind\_end+1;*

*clear i edge*

*end*

*clear ind\_start ind\_end ans*

*save("maxVolume.mat",'maxPop','maxPush');*

*writetable(table(maxPop',maxPush'),"maxVolume.xlsx");*

*function ind=During(init,Date,n)*

*val=Date(init);*

*for i=init:n*

*if (Date(i)~=val)*

*break*

*end*

*end*

*if (i==n)*

*ind=n;*

*else*

*ind=i-1;*

*end*

*end*

##### 附录四：遗传退火 生成子图、计算、打印（结构代码相同省略）

CateEdge5

clc;clear;close all;

%对DC5的相邻节点做分类

%delete(gcp('nocreate'));

load Series.mat

load cate.mat

%p=parpool(12);

Normal\_edge=[Count{1},Count{3},Count{6},Count{7}];

N=cell(1,4);

E=cell(1,4);

%% Node1DC5左节点

%N{1}=Time\_start(E{1})';

E{1}=find(Time\_end==5)';

E{1}=intersect(E{1},Normal\_edge);

N{1}=Time\_start(E{1})';

%% Node2DC5右节点

%N{2}=Time\_end(E{2})';

E{2}=find(Time\_start==5)';

E{2}=intersect(E{2},Normal\_edge);

N{2}=Time\_start(E{2})';

%% Node3DC5左节点的右节点

N3=[];

for node=1:length(N{1})

edge=find(Time\_start==N{1}(node))';

edge=intersect(edge,Normal\_edge);

N3=[N3,Time\_end(edge)'];

E{3}=[E{3},edge];

end

N3=N3(N3~=5);

N{3}=unique(N3);

E{3}=find(Time\_end(E{3})~=5)';

clear node edge N3

%% Node4DC5右节点的左节点

N4=[];

for node=1:length(N{2})

edge=find(Time\_end==N{2}(node))';

edge=intersect(edge,Normal\_edge);

N4=[N4,Time\_end(edge)'];

E{4}=[E{4},edge];

end

N4=N4(N4~=5);

N{4}=unique(N4);

E{4}=find(Time\_start(E{4})~=5)';

clear node edge N4

%% clear

%delete(p);

save("cateNode5.mat","N","E");

clear ans

CatePredict5.m

clc;clear;close all;

load FlowsAmong5.mat

%预测DC5附近的流量

delete(gcp('nocreate'));

p=parpool(8);

nearFlows=Flows(:,365\*2-61+1:365\*2);

clear i

len\_pre=31;%预测长度

Predict=zeros(4,len\_pre);

str={'左节点一月预测','右节点一月预测','左节点的右节点一月预测','右节点的左节点一月预测'};

err=zeros(1,4);

figure(2);

for i=1:4

subplot(4,1,i);

Data=nearFlows(i,:);

len=length(Data);

tmp\_data=Data;

for j=1:len\_pre

len\_block=40;%滑块长度

len=len+1;

model = arima(5,0,1);

fit= estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

fore = forecast(fit,1);

tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

end

hold on

%bias=randn(1,31)\*std(result{i})\*0.5+exp(1:0.1:4);bias(1)=bias(1)+5;

Predict(i,:)=tmp\_data(len-len\_pre+1:len);

%plot([datetime(2022,11,12:30),datetime(2022,12,1:31)],Data(end-50+1:end),'b-');

%plot(41:81,[tmp\_data(end-len\_proving+1:end),Predict(i,:)],'r+-');

plot(datetime(2023,1,1:31),Predict(i,:),'r+-');

xlim([datetime("2023-01-01") datetime("2023-01-31")]);

ylabel("货量");

title(str{i});

%datetick('x','mm-dd');

hold off

clear len a b

end

PredictCate=Predict;

clear Predict

save("PredictCate5.mat","PredictCate");

delete(p);

clear ans

function y\_pre=Average(y,n)

m=length(y);

y\_pre=sum( y(m-n+1:m) ) / n;

end

function str=Node(node)

str=strcat("DC",num2str(node));

end

FlowAmong5

clc;clear;close all;

%计算DC5附近的流量

%delete(gcp('nocreate'));

%p=parpool(12);

load Series.mat

load cateNode5.mat

startTime=min(find(Date==datetime(2022,11,1)));

During=days(datetime(2022,12,31)-datetime(2022,11,1))+1;

%Flows=zeros(4,During);

Flows=zeros(4,365\*2);

hold on

for i=1:4

%Flows(i,:)=sum(Time\_var(E{i},end-During+1:end),1);

Flows(i,:)=sum(Time\_var(E{i},:),1);

%plot([datetime(2022,11,1:30),datetime(2022,12,1:31)],Flows(i,:),'LineWidth',1);

plot(1:365\*2,Flows(i,:),'LineWidth',1);

end

clear i

grid on

legend("DC5左节点","DC5右节点","左节点的右节点","右节点的左节点");

%xlim([datetime("2022-11-1") datetime("2022-12-31")]);

ylabel("货量");

set(gca,'yscale','log');

title("DC5最近二月的货物流量");

save("FlowsAmong5.mat","Flows");

hold off

%% clear

%delete(p);

clear ans

FlowsPredict5.m

clc;clear;close all;

load FlowsAmong5.mat

load cateNode5.mat

load Series.mat

%预测DC5附近的流量

delete(gcp('nocreate'));

p=parpool(12);

nearFlows=Flows(:,365\*2-61+1:365\*2);

clear i

len\_pre=31;%预测长度

err=zeros(1,4);

Predict=cell(1,4);

for cate=1:4

for i=1:length(E{cate})

edge=E{cate}(i);

Data=Time\_var(edge,365\*2-61+1:365\*2);

Data=filloutliers(Data,"linear");

len=length(Data); clc;clear;close all;

GAforOrderIN5.m

load StatusAmong5.mat

load maxVolume.mat

delete(gcp('nocreate'));

%最后一位是值的大小

P=parpool(12);

Population=40;

Generation=200;

edge=Status{3}(:,1);

EncodingLength=length(edge)+1;

Genes=cell(1,31);

parfor day=1:31

%solutionChange

genes=cell(Generation,1);

%Goal=zeros(1,Generation);

tic

t=0;

genes{1}=GenerateParental2023(Population,EncodingLength,day);

%top\_gene=genes{1}{1};

%Goal(1)=top\_gene(1,EncodingLength);

t=t+toc;

disp(t);

for k=2:Generation

tic

t=0;

Parent=genes{k-1};%20

Child1=Mate2023(Parent,3);%10

Child2=Variate2023(Parent,0.1);%20

genes{k}=Choose2023(Parent,Child1,Child2,day);

t=t+toc;

%Goal(k)=top\_gene(1,EncodingLength);

disp(t);

end

Genes{day}=genes;

end

%str=strcat("Genes",num2str(int(rand\*100)),".mat");

save("GenesIN5.mat","Genes");

%delete(P);

function Genes=GenerateParental2023(Population,EncodingLength,day)

%遗传算法生成亲代种群

%Population为种群大小,EncodingLength为染色体基因长度,Genes为种群染色体样本

Genes=zeros(Population,EncodingLength);

for p=1:Population

Sequence=(rand(1,EncodingLength-1)<=0.7); %生成个体基因序列

[~,value,overflow]=SAforOrderIN5(Sequence,day);

while (overflow==1)%不能存活的个体

Sequence=(rand(1,EncodingLength-1)<=0.7); %生成个体基因序列 [~,value,overflow]=SAforOrderIN5(Sequence,day);

end

Genes(p,:)=[Sequence,value]; %产生个体

end

end

function Child=Mate2023(Parent,IntersectionNum)

%遗传算法交配亲代得到子代

%Parent为亲代种群基因型,IntersectionNum为交换基因的数量

Child=Parent;

[Population,EncodingLength]=size(Parent);

EncodingLength=EncodingLength-1;

for i=1:2:Population

%利用Logistics混沌序列生成交叉基因座序列

Locus0=rand; Locus(1)=4\*Locus0\*(1-Locus0);

for loc=2:IntersectionNum

Locus(loc)=4\*Locus(loc-1)\*(1-Locus(loc-1));

end

Locus=1+ceil(EncodingLength\*Locus);

Locus=unique(Locus);

%第i个个体与第i+1个个体交配,对应基因座交叉互换

temp=Child(i,Locus);

Child(i,Locus)=Child(i+1,Locus);

Child(i+1,Locus)=temp;

end

end

function varied\_genotype=Variate2023(initial\_genotype,Rate)

%遗传算法亲代变异得到新的种群基因型

%initial\_genotype为变异前种群基因型,varied\_genotype为变异后种群基因型,Rate为变异率

[Population,EncodingLength]=size(initial\_genotype);

EncodingLength=EncodingLength-1;

Chosen=find(rand(1,Population)<Rate, 1); %标记被变异的个体

if (isempty(Chosen)) %如果都没人变异就找一个人变异

Chosen=floor(Population\*rand)+1;

end

varied\_genotype=initial\_genotype;

numVaried=length(Chosen);

for p=1:numVaried

ind=GenerateBreakpoints2023(EncodingLength,3); %使用三交换构造新的基因序列

varied\_genotype(Chosen(p),:)=varied\_genotype(Chosen(p),[1:ind(1)-1,ind(2)+1:ind(3),ind(1):ind(2),ind(3)+1:EncodingLength+1]);

end

end

function ind=GenerateBreakpoints2023(range,num)

%生成用于交换的1\*n的系数向量

rng("shuffle");

ind=ceil(range\*rand(1,num));

ind=unique(ind);

while (length(ind)<num)

ind=ceil(range\*rand(1,num));

ind=unique(ind);

end

end

function superior=Choose2023(A,B,C,day)

%遗传算法选择优秀基因型作为新的亲本

%A,B,C分别为待选择的基因型

Merge=[A;B;C];

[Total,EncodingLength]=size(Merge);

%EncodingLength=EncodingLength-1;

%[~,ind1]=sort(Merge,2); %整合三个种群

sequenceVal=Merge(:,EncodingLength);

for j=1:Total

Sequence=Merge(j,:);

chosen=nnz(Sequence);

sequenceVal(j)=sequenceVal(j)\*(1/chosen);

end

[~,ind2]=sort(sequenceVal,'descend'); %选择

superior=Merge(ind2(1:40),:);

end

tmp\_data=Data;

len\_block=40;%滑块长度

for j=1:len\_pre

len=len+1;

try

model = arima(5,0,1);

fit = estimate(model,tmp\_data(len-len\_block:len-1)');

%fit = estimate(model,tmp\_data');

fore = forecast(fit,1);

tmp\_data(len)=fore;

%[a,b]=Smoothing(tmp\_data,0.3);

%tmp\_data(len)=a+b;

catch

tmp\_data(len)=Average(tmp\_data,len\_block);

end

end

Predict{cate}(i,:)=tmp\_data(len-len\_pre+1:len);

clear len a b edge

end

end

PredictFlows=Predict;

clear Predict

save("PredictFlows5.mat","PredictFlows");

delete(p);

clear ans

function y\_pre=Average(y,n)

m=length(y);

y\_pre=sum( y(m-n+1:m) ) / n;

end

SAforOrderIN5.m

%clc;clear;close all;

function [order,result,overFlow]=SAforOrderIN5(index,day)

load StatusAmong5.mat

load maxVolume.mat

reOrder=0;

rng('shuffle');

num\_node=81;

%%init loading--E3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

local=find(index==1);

edge=Status{3}(local,1);

start=Status{3}(local,2);

term=Status{3}(local,3);

contain=Status{3}(local,4);

predict=Status{3}(local,5+day-1);

predict=min(predict,contain);

predict=Clean(predict);

%rate=sum(predict);%正在跑的总流量

Total=sum(Status{1}(:,4));%流向DC5的需要分配的流量

num\_edge=length(edge);%E3集合的大小

shutPop=ones(1,num\_node);%一开始所有出点都开着

shutPush=ones(1,num\_node);%一开始所有入点都开着

overFlow=0;%流量是否溢出

%%产生一组可接受的初始解

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%order=predict\*CalcPort(total,rate);

reOrder=0;

order=zeros(1,num\_edge);

order(:)=Total/num\_edge;

for i=1:num\_edge%考虑E3的运输能力

if (predict(i)+order(i)>contain(i))

reOrder=predict(i)+order(i)-contain(i);%把边能多承载的流量拿出来

order(i)=contain(i)-predict(i);%把边塞满

if (order(i)<0)

error("错误的分配");

end

if (predict(i)+order(i)>contain(i))

error("错误的分配");

end

else%这条边没有塞满

if (reOrder<=(contain(i)-(predict(i)+order(i))+1)/2)

order(i)=order(i)+reOrder;

reOrder=0;

else

order(i)=order(i)+(contain(i)-(predict(i)+order(i))+1)/2;

reOrder=reOrder-(contain(i)-(predict(i)+order(i))+1)/2;

end

end

end

flag=1;%还存在有容量的边;

while (reOrder>0 && flag==1)

flag=0;

for i=1:num\_edge%考虑E3的运输能力

if (predict(i)+order(i)<contain(i))

flag=1;

if (reOrder<=(contain(i)-(predict(i)+order(i))+1)/2)

order(i)=order(i)+reOrder;

reOrder=0;

else

if (reOrder>0)

order(i)=order(i)+(contain(i)-(predict(i)+order(i))+1)/2;

reOrder=reOrder-(contain(i)-(predict(i)+order(i))+1)/2;

end

end

end

end

end

if (flag==0 && reOrder>0)

overFlow=1;

result=-1;

return

end

clear flag

%考虑N1的出货能力和N3的收获能力

runningPop=zeros(1,num\_node);

runningPush=zeros(1,num\_node);

for i=1:num\_edge

node=start(i);%取出N1中的点

runningPop(node)=runningPop(node)+predict(i)+order(i);

node=term(i);%取出N3中的点

runningPush(node)=runningPush(node)+predict(i)+order(i);

clear node

end

for i=1:num\_edge%处理点的冲突

if (shutPop(start(i))==0 || shutPush(term(i))==0)

continue

end

node=start(i);%N1

if (runningPop(node)>maxPop(node))

if (runningPop(node)-maxPop(node)<=order(i))

take=runningPop(node)-maxPop(node);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPop(node)=runningPop(node)-take;

shutPop(node)=0;%冲突已经解决关闭冲突节点

else

take=order(i);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPop(node)=runningPop(node)-take;

end

end

clear node take

node=term(i);%N1

if (runningPush(node)>maxPush(node))

if (runningPush(node)-maxPush(node)<=order(i))

take=runningPush(node)-maxPush(node);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPush(node)=runningPush(node)-take;

shutPush(node)=0;%冲突已经解决关闭冲突节点

else

take=order(i);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPush(node)=runningPush(node)-take;

end

end

clear node take

end

%循环解决冲突

while (reOrder>0 && (nnz(shutPop)+nnz(shutPush))>0)%还有未分配的流量并且不能所有节点都关闭了

for i=1:num\_edge%考虑E3的运输能力

if (shutPop(start(i))==0 || shutPush(term(i))==0)

continue

end

if (predict(i)+order(i)>contain(i))

reOrder=predict(i)+order(i)-contain(i);%把边能多承载的流量拿出来

order(i)=contain(i)-predict(i);%把边塞满

else%这条边没有塞满

if (reOrder<=(contain(i)-(predict(i)+order(i))+1)/2)

order(i)=order(i)+reOrder;

reOrder=0;

else

order(i)=order(i)+(contain(i)-(predict(i)+order(i))+1)/2;

reOrder=reOrder-(contain(i)-(predict(i)+order(i))+1)/2;

end

end

end

flag=1;%还存在有容量的边;

while (reOrder>0 && flag==1)

flag=0;

for i=1:num\_edge%考虑E3的运输能力

if (shutPop(start(i))==0 || shutPush(term(i))==0)

continue

end

if (predict(i)+order(i)<contain(i))

flag=1;

if (reOrder<=(contain(i)-predict(i)+1)/2)

order(i)=order(i)+reOrder;

reOrder=0;

else

if (reOrder>0)

order(i)=order(i)+(contain(i)-predict(i)+1)/2;

reOrder=reOrder-(contain(i)-predict(i)+1)/2;

end

end

end

end

end

if (flag==0 && reOrder>0)

overFlow=1;

result=-1;

return

end

clear flag

%考虑N1的出货能力和N3的收获能力

runningPop=zeros(1,num\_node);

runningPush=zeros(1,num\_node);

for i=1:num\_edge

if (shutPop(start(i))==0 || shutPush(term(i))==0)

continue

end

node=start(i);%取出N1中的点

runningPop(node)=runningPop(node)+predict(i)+order(i);

node=term(i);%取出N3中的点

runningPush(node)=runningPush(node)+predict(i)+order(i);

clear node

end

for i=1:num\_edge%处理点的冲突

if (shutPop(start(i))==0 || shutPush(term(i))==0)

continue

end

node=start(i);%N1

if (runningPop(node)>maxPop(node))

if (runningPop(node)-maxPop(node)<=order(i))

take=runningPop(node)-maxPop(node);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPop(node)=runningPop(node)-take;

shutPop(node)=0;%冲突已经解决关闭冲突节点

else

take=order(i);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPop(node)=runningPop(node)-take;

end

end

clear node take

node=term(i);%N1

if (runningPush(node)>maxPush(node))

if (runningPush(node)-maxPush(node)<=order(i))

take=runningPush(node)-maxPush(node);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPush(node)=runningPush(node)-take;

shutPush(node)=0;%冲突已经解决关闭冲突节点

else

take=order(i);

reOrder=reOrder+take;

order(i)=order(i)-take;

runningPush(node)=runningPush(node)-take;

end

end

clear node take

end

end

%%退火,order为分配方案

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

order=Clean(order);

begin\_heat=100;

end\_heat=1;

Heat=begin\_heat;

Rate=0.9;

MarkovLength=500;

totalContain=Status{3}(:,4);%全部边为了得到目标量

totalPredict=Status{3}(:,5+day-1);

totalLen=length(totalPredict);

changed=Effected(totalPredict,order,local);

Res=CalcGoal(changed,totalContain,totalLen);

for Layer=1:MarkovLength

upFlow=Generate(predict,contain,order);

tentative=1;%转移试探次数

while (upFlow(1)==-1 || CheckValid(order+upFlow,start,term,maxPop,maxPush,contain)==0)

upFlow=Generate(predict,contain,order);

tentative=tentative+1;

if (tentative>1000)

result=Res;

return

end

end

clear tentative

changed=Effected(totalPredict,order+upFlow,local);

tmp\_Res=CalcGoal(changed,totalContain,totalLen);

cost=tmp\_Res-Res;

if (cost>0 || exp(-cost/Heat)>rand)

order=order+upFlow;

Res=tmp\_Res;

end

Heat=Heat\*Rate;

if (Heat<end\_heat)

break

end

end

result=Res;

end

function rate=CalcPort(total,running)

rate=total/running;

end

function valid=CheckValid(runningFlow,start,term,maxPop,maxPush,edgeContain)

len=length(runningFlow);

num=81;

valid=1;

runningPop=zeros(1,num);

runningPush=zeros(1,num);

for i=1:len

if (runningFlow(i)>edgeContain(i))

valid=0;

return

end

s=start(i);

t=term(i);

runningPop(s)=runningPop(s)+runningFlow(i);

runningPush(t)=runningPush(t)+runningFlow(i);

end

for i=1:num

if (runningPop(i)>maxPop(i))

valid=0;

return;

end

if (runningPush(i)>maxPush(i))

valid=0;

return;

end

end

end

function upFlow=Generate(predict,contain,res)

res=res';

running=predict+res;%在当前分配方案下的流量

running=Clean(running);

n=length(running);

release=contain-running;

release=Clean(release);

if (~isempty(find(release < 0)))

upFlow=-ones(1,n);

return;

%error("分配了大于线路容量的流量");

end

[sorted\_release,ind\_sort]=sort(release);

shake=0;

for i=1:n

if (shake>=0)

upFlow(ind\_sort(i))=sorted\_release(i)\*0.1;

shake=shake+upFlow(ind\_sort(i));

else

upFlow(ind\_sort(i))=-min(sorted\_release(i)\*0.1,res(i));

shake=shake+upFlow(ind\_sort(i));

end

end

clear i

p=1;

while(shake>0)

if (shake<1 && res(ind\_sort(p))+upFlow(ind\_sort(p))>shake)%消除抖动循环

upFlow(ind\_sort(p))=upFlow(ind\_sort(p))-shake;

shake=0;

break

end

take=shake\*0.2;

while (res(ind\_sort(p))+upFlow(ind\_sort(p))-take<=0)%不能够被调节

take=take\*0.8;

if (take<1)

overFlow=1;

return

end

end

upFlow(ind\_sort(p))=upFlow(ind\_sort(p))-take;

shake=shake-take;

p=mod(p,n)+1;

end

while(shake<0)

if (shake>-1 && res(ind\_sort(p))+upFlow(ind\_sort(p))-shake<=contain(ind\_sort(p)))%消除抖动循环

upFlow(ind\_sort(p))=upFlow(ind\_sort(p))-shake;%减负等于加正下同理;

shake=0;

break

end

take=shake\*0.2;

if (res(ind\_sort(p))+upFlow(ind\_sort(p))+take<=contain(ind\_sort(p)))%能够被调节

upFlow(ind\_sort(p))=upFlow(ind\_sort(p))-take;

shake=shake-take;

end

p=mod(p,n)+1;

end

end

function Res=Effected(raw,var,index)

len=length(var);

for i=1:len

raw(index(i))=raw(index(i))+var(i);

end

Res=raw;

end

function loading=CalcLoad(contain,running)

loading=running./contain;

end

function Res=CalcGoal(running,contain,totalLen)

loading=CalcLoad(contain,running);

Average=sum(loading)/totalLen;

Res=sum((loading-Average).^2)/totalLen;

Res=1/Res;

end

function Res=Clean(raw)

Res=fix(raw);

End

Weight.m

%clc;clear;close all;

function [weightNode,weightEdge]=Weight(start,term,value)

weightEdge=zeros(1049,8);%8个指标

weightNode=zeros(81,12);%12个指标

for edge=1:num\_connect

weightEdge(edge,:)=MeasureEdge(edge,start,term,value);

end

for i=1:8

weightEdge(:,i)=zscore(weightEdge(:,i));

end

weightEdge(:,8)=-weightEdge(:,8);%历史最长流量不存在天数是极小指标

indEdge\_clean=find(weightEdge(1,:));

tmp=weightEdge(indEdge\_clean,:);

weightEdge=[];

weightEdge=tmp;

clear i edge tmp

for node=1:num

MeasureNode.m

weightNode(node,:)=MeasureNode(node,start,term,value);

end

for i=1:12

weightNode(:,i)=zscore(weightNode(:,i));

end

indNode\_clean=find(weightNode(1,:));

tmp=weightNode(indNode\_clean,:);

weightNode=[];

weightNode=tmp;

clear i node tmp

end

Score.m

function [scoreNode,scoreEdge]=Score(weightNode,weightEdge)

evalNode(1,:)=[0.154,0.155,0.123,0.125,0.133,0.09,0.1,0.166,0.163];

evalNode(2,:)=[-0.197,-0.201,-0.021,-0.042,-0.065,0.571,0.537,-0.046,-0.122];

evalNode(3,:)=[-0.182,-0.153,0.571,0.561,-0.337,0.008,-0.136,-0.098,-0.091];

evalPartN=[-0.65887,0.1548,0.1311];

for i=1:81

nodeY=zeros(1,3);

for j=1:3

nodeY(j)=evalNode(j).\*weightNode(i);

end

scoreNode(i)=-sum(nodeY.\*evalPartN);

end

evalEdge(1,:)=[0.225,0.219,0.257,0.217,0.261,0.138,0.019];

evalEdge(2,:)=[0.409,0.432,-0.216,0.319,-0.280,0.13,0.381];

evalEdge(3,:)=[0.064,0.077,-0.119,0.244,-0.161,0.584,0.745];

evalPartE=[0.4775,0.1923,0.14185];

for i=1:1049

edgeY=zeros(1,3);

for j=1:3

edgeY(j)=evalEdge(j).\*weightEdge(i);

end

scoreEdge(i)=-sum(edgeY.\*evalPartE);

end

end

function Value=MeasureEdge(edge,Time\_start,Time\_end,Time\_var)

%Col1:最大流量 Col2:最小流量 Col3:平均流量

%Col4:最长流量存在天数 Col5:最长流量不存在天数 Col6:流量存在天数

%Col7:该边流量占左节点的总发货路径数比 Col8:该边流量占右节点的总收货流量比

leftNode=Time\_start(edge);

leftEdge=find(Time\_start==leftNode);

rightNode=Time\_end(edge);

rightEdge=find(Time\_end==rightNode);

Value=zeros(1,8);

Value(3)=mean(Time\_var(edge,:));

Value(6)=nnz(Time\_var(edge,:));

Value(7)=Value(7)+sum(sum(Time\_var(leftEdge,:),1));

Value(8)=Value(8)+sum(sum(Time\_var(rightEdge,:),1));

running=0;%当天是否有流量在跑,计算存在天数

dayFlag=1;

for day=1:365\*2

Value(1)=max(Value(1),Time\_var(edge,day));

Value(2)=min(Value(2),Time\_var(edge,day));

if (Time\_var(edge,day)>0)

if (running==0)

Value(5)=max(Value(5),day-dayFlag);

running=1;

dayFlag=day;

end

else

if (running==1)

Value(4)=max(Value(4),day-dayFlag);

running=0;

dayFlag=day;

end

end

end

if (running==0)%统计最后没有更新的序列状态

Value(5)=max(Value(5),day-dayFlag);

else

Value(4)=max(Value(4),day-dayFlag);

end

if (Value(7)~=0)

Value(7)=Value(3)/Value(7);%流量占比

end

if (Value(8)~=0)

Value(8)=Value(3)/Value(8);%流量占比

end

end

MeasureNode.m

function Value=MeasureNode(node,Time\_start,Time\_end,Time\_var)

%Col1:最大收货量 Col2:最小收货量 Col3:平均收货量

%Col4:最大发货量 Col5:最小发货量 Col6:平均发货量

%Col7:收货总天数 Col8:发货总天数 Col9:同时发货总天数

%Col9:最大邻接点数 Col10:最小邻接点数 Col12:平均邻接点数

Value=zeros(1,12);

leftEdge=find(Time\_end==node);%寻找与左节点相邻个边

rightEdge=find(Time\_start==node);%寻找与右节点相邻个边

for day=1:365\*2

tmpValue=zeros(1,3);

%Col1收货量

%Col2发货量

%Col3邻接点数

for edge=1:length(leftEdge)%计算收货指标:1+3

tmpValue(1)=tmpValue(1)+Time\_var(edge,day);

tmpValue(3)=tmpValue(3)+(Time\_var(edge,day)>=0);

end

for edge=1:length(rightEdge)%计算发货指标:2+3

tmpValue(2)=tmpValue(2)+Time\_var(edge,day);

tmpValue(3)=tmpValue(3)+(Time\_var(edge,day)>=0);

end

Value(1)=max(Value(1),tmpValue(1));

Value(2)=min(Value(2),tmpValue(1));

Value(3)=Value(3)+tmpValue(1);

Value(4)=max(Value(4),tmpValue(2));

Value(5)=min(Value(5),tmpValue(2));

Value(6)=Value(6)+tmpValue(2);

Value(7)=Value(7)+(tmpValue(1)>0);

Value(8)=Value(8)+(tmpValue(2)>0);

Value(9)=Value(9)+((tmpValue(1)+tmpValue(2))>0);

Value(10)=max(Value(10),tmpValue(3));

Value(11)=min(Value(11),tmpValue(3));

Value(12)=Value(12)+tmpValue(3);

end

if (Value(7)~=0)

Value(3)=Value(3)/Value(7);

end

if (Value(8)~=0)

Value(6)=Value(6)/Value(8);

end

if (Value(9)~=0)

Value(12)=Value(12)/Value(9);

end

end

GAforAdd.m

clc;clear;close all;

load Series.mat

load predictResult.mat

%delete(gcp('nocreate'));

%最后一位是值的大小

%P=parpool(12);

Population=400;

Generation=200;

EncodingLength=81\*2;

%disp(toOrder\_all);

%solutionChange

Genes=cell(1,4);

global c2;

global c3;

global c1;

global yita;

c2=0.3+rand\*0.4;

c3=0.3+rand\*0.4;

c1=(c2+c3)\*1000;

yita=4/9;

%%

idx=0;

for i=1:7:21

idx=idx+1;

Flow(idx,:)=sum(Predict(i:i+7,:));

Flow(idx,:)= filloutliers(Flow(idx,:),"linear");

end

clear idx;

%% GAforAdd

for week=1:4

running=Flow(week);

genes=cell(Generation,1);

tic

t=0;

genes{1}=GenerateParental2023(Population,EncodingLength);

%top\_gene=genes{1}{1};

%Goal(1)=top\_gene(1,EncodingLength);

t=t+toc;

disp(t);

for k=2:Generation

tic

t=0;

Parent=genes{k-1};%20

Child1=Mate2023(Parent,3);%10

Child2=Variate2023(Parent,0.1);%20

genes{k}=Choose2023(Parent,Child1,Child2,Population,running);

t=t+toc;

Genes{week}=genes;

disp(t);

end

end

%str=strcat("Genes",num2str(int(rand\*100)),".mat");

save("GenesAdd.mat","Genes");

%delete(P);

function Genes=GenerateParental2023(Population,EncodingLength,day,toOrder)

%遗传算法生成亲代种群

%Population为种群大小,EncodingLength为染色体基因长度,Genes为种群染色体样本

Genes=zeros(Population,EncodingLength);

for p=1:Population

Sequence=(rand(1,EncodingLength)<=0.09); %生成个体基因序列

while (nnz(Sequence)>15)%约束条件,不能存活的个体

Sequence=(rand(1,EncodingLength)<=0.09); %生成个体基因序列

end

Genes(p,:)=Sequence; %产生个体

end

end

function Child=Mate2023(Parent,IntersectionNum)

%遗传算法交配亲代得到子代

%Parent为亲代种群基因型,IntersectionNum为交换基因的数量

Child=Parent;

[Population,EncodingLength]=size(Parent);

EncodingLength=EncodingLength-1;

for i=1:2:Population

%利用Logistics混沌序列生成交叉基因座序列

Locus0=rand; Locus(1)=4\*Locus0\*(1-Locus0);

for loc=2:IntersectionNum

Locus(loc)=4\*Locus(loc-1)\*(1-Locus(loc-1));

end

Locus=1+ceil(EncodingLength\*Locus);

Locus=unique(Locus);

%第i个个体与第i+1个个体交配,对应基因座交叉互换

temp=Child(i,Locus);

Child(i,Locus)=Child(i+1,Locus);

Child(i+1,Locus)=temp;

while (nnz(Child(i,:))>15)%约束条件,不能存活的个体

tmp=Child(i,:);

ind=find(tmp==1);

ind=ind(ceil(rand\*length(ind)));

tmp(ind)=0;

Child(i,:)=tmp;

end

while (nnz(Child(i+1,:))>15)%约束条件,不能存活的个体

tmp=Child(i+1,:);

ind=find(tmp==1);

ind=ind(ceil(rand\*length(ind)));

tmp(ind)=0;

Child(i+1,:)=tmp;

end

end

end

function varied\_genotype=Variate2023(initial\_genotype,Rate)

%遗传算法亲代变异得到新的种群基因型

%initial\_genotype为变异前种群基因型,varied\_genotype为变异后种群基因型,Rate为变异率

[Population,EncodingLength]=size(initial\_genotype);

EncodingLength=EncodingLength-1;

Chosen=find(rand(1,Population)<Rate, 1); %标记被变异的个体

if (isempty(Chosen)) %如果都没人变异就找一个人变异

Chosen=floor(Population\*rand)+1;

end

varied\_genotype=initial\_genotype;

numVaried=length(Chosen);

for p=1:numVaried

ind=GenerateBreakpoints2023(EncodingLength,3); %使用三交换构造新的基因序列

varied\_genotype(Chosen(p),:)=varied\_genotype(Chosen(p),[1:ind(1)-1,ind(2)+1:ind(3),ind(1):ind(2),ind(3)+1:EncodingLength+1]);

while (nnz(varied\_genotype(Chosen(p),:))>15)%约束条件,不能存活的个体

tmp=varied\_genotype(Chosen(p),:);

ind=find(tmp==1);

ind=ind(ceil(rand\*length(ind)));

tmp(ind)=0;

varied\_genotype(Chosen(p),:)=tmp;

end

end

end

function ind=GenerateBreakpoints2023(range,num)

%生成用于交换的1\*n的系数向量

rng("shuffle");

ind=ceil(range\*rand(1,num));

ind=unique(ind);

while (length(ind)<num)

ind=ceil(range\*rand(1,num));

ind=unique(ind);

end

end

function superior=Choose2023(A,B,C,Population,running)

%遗传算法选择优秀基因型作为新的亲本

%A,B,C分别为待选择的基因型

Merge=[A;B;C];

[Total,EncodingLength]=size(Merge);

tmp\_order=zeros(1,EncodingLength);

for j=1:Total

Sequence=Merge(j,:);

load(j)=Calccost(Sequence,running);

end

[~,ind2]=sort(load,'descend'); %选择

superior=Merge(ind2(1:Population),:);

end

function Res=Calccost(Sequence,running)

global c1;

global c2;

global c3;

global yita;

num=nnz(Sequence);

%c1开设成本,c2运输成本,c3处理成本

reOrder=sum(running)\*0.02;

reOrder=reOrder\*(1+rand\*0.4-0.2);

Load=mean(CalcLoad(Sequence,reOrder,running));

Res=Load/(c1\*num+(c2+c3)\*reOrder/yita);

end