道路运行情况模拟程序：

clc;

clear;

Tmax = 500; %考虑的时间的上限

carnum = 0; %carnum 是当前通过的车的总数目

fieldCarNum = 0; %fieldCarNum 进入小区的车辆数

beta = 0.5; %出现车的概率,体现车流量

fieldCapure = 800; %小区内道路的承载量

fieldDistance = 100; %小区内道路的逻辑长度

fRpb = 0.8; %行人自行车修正系数

R = 0.8; %主路右转车进入小区的比例

L1 = 5; %车身长度,km

h = 25; %标准饱和车头时距

alpha = pi / 4; %转弯角度

miu = 0.18; %横向力系数

v = 60/3.6; %车辆速度

t\_avg = L1 / v ; %畅通情况下车辆直行通过计算截面的平均耗时

ER = (L1 + alpha \* R)/(h \* sqrt(127 \* R \* miu)); %右转车转换系数

fR = 100/(100 + R \* (ER - 1)); %右转修正系数

theta = 0; %车辆入口延时的影响因子

delay = zeros(Tmax, 1); %记录入口延时

T = 20; %路灯周期

Tg = 10; %绿灯时间

Car = cell(carnum,1); %Car 是所有的车的集合

Car0 = cell(carnum,1); %Car 是所有的车的集合

car = struct('road',0,'distance',0,'state',0);

%road 是当前车所在的道路 distance 表示在这条路上的位置 state 表示是否在区域里 1-在里面 0-在外面

t = 0; %当前时间

Dt = cell(t,1); %每个时刻的平均延误

Dt0 = cell(t,1); %每个时刻的平均延误

Light = 0; % Light 表示当前灯是红灯 0，还是绿灯 1

roadnum = 4; %roadnum 是所有的道路数目

Outroad = []; % 与出口相连的路

Outroad(1) = 4;

Inroad = [2,3]; %与入口相连的路

FieldRoad = []; %小区内的路

FieldRoad(1) = 3;

Troad = zeros(roadnum,1);%每条路上的路灯周期

Troad(:) = 20;

Tgroad = zeros(roadnum,1);%每条路上绿灯时间

Tgroad(:) = 5;

RoadMap = zeros(roadnum,roadnum); %路的可达性矩阵

RoadMap0 = zeros(roadnum,roadnum); %路的可达性矩阵

Roadcapture = zeros(roadnum,1); %Roadcapture 路的设计承载量

Roadcarnum = zeros(roadnum,1); %Roadcarnum 路当前有的车数量

Roadcarnum0 = zeros(roadnum,1); %Roadcarnum 路当前有的车数量

Roaddistance = zeros(roadnum,1);%Roaddistance 路的距离

Roaddt = zeros(roadnum,1); %Roaddt 每条路上的平均延迟时间

Roadcapture = [1000,1000,1000,1];

Roadcapture(3) = fieldCapure;

Roadcapture = Roadcapture';

Roaddistance = [100,100,100,1];

Roaddistance(3) = fieldDistance;

RoadMap = [0,0,0,0;

1,0,0,0;

1,0,0,0;

0,1,1,0];

RoadMap0 = [0,0,0,0;

1,0,0,0;

0,0,0,0;

0,1,0,0];

myres = zeros(500,4);

Flag = [0,1,0,0]';

mainflow = 3600 \* beta; %主路的车流量

for t = 1:Tmax

%%%%%%判断当前是否有车到来 0 没有，1 有

Dt{t} = 0;

Dt0{t} = 0;

ra = rand();

if ra >= beta

ra = 0;

else

ra = 1;

end

%%%%%%增加车的数量,更新车的情况

if ra == 1

carnum = carnum+1;

car.road = 1;

car.distance = Roaddistance(1);

car.state = 1;

Car{carnum} = car;

Car0{carnum} = car;

Roadcarnum(1) = Roadcarnum(1)+1; Roadcarnum0(1) = Roadcarnum0(1)+1;

end

%%%%%%判断当前红绿灯情况

Light = mod(t,T);

if Light <= Tg

Light = 1;

else

Light = 0;

end

for cari = 1:carnum

if(Car{cari}.state == 0)

continue;

end

%%不考虑小区开放时

if Light == 1 %%%绿灯时

[nextroad,nextdistance,nextstate] = nextdir(cari,RoadMap0,Car0,Roaddistance,Outroad,Roadcarnum0);

%%%函数 nextdir 返回 cari 这辆车下一次所在的路的在路上的距离

else %%%%红灯时

[nextroad,nextdistance,nextstate] = nextdir\_red(cari,RoadMap0,Car0,Roaddistance,Outroad,Roadcarnum0);

end

Roadcarnum0(Car0{cari}.road) = Roadcarnum0(Car0{cari}.road)-1;

Car0{cari}.road = nextroad;

Car0{cari}.distance = nextdistance;

Car0{cari}.state = nextstate;

if nextstate == 1

Roadcarnum0(nextroad) = Roadcarnum0(nextroad)+1; %%%%%%当前这条路上的车的数量

end

%%考虑小区开放时

% disp(strcat('car', num2str(cari)));

if Light == 1 %%%绿灯时

[nextroad,nextdistance,nextstate] = nextdir(cari,RoadMap,Car,Roaddistance,Outroad,Roadcarnum);

%%%函数 nextdir 返回 cari 这辆车下一次所在的路的在路上的距离

else %%%%红灯时

[nextroad,nextdistance,nextstate] = nextdir\_red(cari,RoadMap,Car,Roaddistance,Outroad,Roadcarnum);

end

myres(t, 1) = nextroad;

myres(t, 2) = nextdistance;

myres(t, 3) = nextstate;

myres(t, 4) = cari;

Roadcarnum(Car{cari}.road) = Roadcarnum(Car{cari}.road)-1;

Car{cari}.road = nextroad;

Car{cari}.distance = nextdistance;

Car{cari}.state = nextstate;

if nextstate == 1

Roadcarnum(nextroad) = Roadcarnum(nextroad)+1; %%%%%%当前这条路上的车的数量

end

end

if( ra == 1 && ismember(nextroad, FieldRoad) )

fieldCarNum = fieldCarNum + 1;

theta = theta + 1;

else

if(theta >= 0.3)

theta = theta - 0.3;

else

theta = 0;

end

end

if(carnum > 0)

R = R \* 0.5 + 0.5 \* fieldCarNum / carnum;

ER = (L1 + alpha \* R)/(h \* sqrt(127 \* R \* miu));%右转车转换系数

fR = 100/(100 + R \* (ER - 1));%右转修正系数

x = (mainflow) / Roadcapture(1); %道路饱和度

C1 = x \* (fRpb + fR); %主流向通行能力

NDT = (1/C1 - t\_avg) \* (1/t\_avg + 1) \* t\_avg / 2; %信号交叉路口平均延误

Dt{t} = theta \* NDT;

delay(t) = Dt{t};

end

%%%%%%计算每条路的平均延误时间

dt = ((0.5\*Troad).\*(1-Tgroad./Troad))./(1-min(1,Roadcarnum.\*(720./Roaddistance')./Roadcapture).\*(Tgroad./Troad)).\*Flag;

Dt{t} = sum(dt);

dt0 = ((0.5\*Troad).\*(1-Tgroad./Troad))./(1-min(1,Roadcarnum0.\*(720./Roaddistance')./Roadcapture).\*(Tgroad./Troad)).\*Flag;

Dt0{t} = sum(dt0);

end

Dt\_Mat = cell2mat(Dt) - 7.5;

Dt\_Sum = zeros(1, Tmax);

Dt\_Sum(1) = Dt\_Mat(1);

for i = 2 : Tmax

Dt\_Sum(i) = Dt\_Sum(i - 1) + Dt\_Mat(i);

end

Dt0\_Mat = cell2mat(Dt0) - 7.5;

Dt0\_Sum = zeros(1, Tmax);

Dt0\_Sum(1) = Dt0\_Mat(1);

for i = 2 : Tmax

Dt0\_Sum(i) = Dt0\_Sum(i - 1) + Dt0\_Mat(i);

end

dtDelta = cell2mat(Dt0) - cell2mat(Dt);

% save(strcat(num2str(fieldDistance), 'dtDelta.mat'), 'dtDelta');

save('type1\_Dt.mat', 'Dt\_Sum');

% save(strcat(num2str(beta), 'Dt0.mat'), 'Dt0');

交叉口选择道路函数：

function [nextroad] = chooseRoad(cari,currentRoad, RoadMap,Car,Roaddistance, RoadCarNum)

%%%交叉路口选择路的方向

% nextroad:选择的下一条路的标号

% cari:车的序号

% currentRoad: 当前所在的路的标号 % RoadMap: 路的可达性矩阵

% Car:当前各辆车的状态

% Roaddistance:每条路的长度

% RoadCarNum: 当前每条路上有多少车

%

k1 = 1;%%%%两个参数

k2 = 1;

nextroad = 0;

min = intmax();

roadNum = length(RoadMap);

% disp(strcat('currentroadNum', num2str(currentRoad)));

myNeighbor = zeros(roadNum);

myNeighborNum = 0;

for i = 1 : roadNum

if(RoadMap(i, currentRoad) == 0) %%道路不通的情况

continue;

end

myNeighborNum = myNeighborNum + 1;

myNeighbor(myNeighborNum) = i;

% disp(strcat('roadNum', num2str(i)));

myvalue = 0;

for j = 1 : (cari - 1)

if(Car{j}.state == 1 && Car{j}.road == i && Car{j}.distance == 1)

%%%% 判断走这条路是否需要等

myvalue = intmax() - 100000;

break;

end

end

%%计算每条路的指标

myvalue = myvalue + k1 \* Roaddistance(i) + k2 \* RoadCarNum(i);

if(myvalue < min)

min = myvalue;

nextroad = i;

end

end

% 随即选择

nextroad = myNeighbor(ceil(rand() \* (myNeighborNum)));

绿灯时车辆移动函数：

function [nextroad,nextdistance,nextstate]=nextdir(cari,RoadMap,Car,Roaddistance,Outroad, RoadCarNum)

% 函数 nextdir 返回 cari 这辆车这一秒之后的状态

% nextroad: 1s后所在的路

% nextdistance:1s后在路上距路入口的逻辑距离

% nextstate: 1s后，该车是否还在区域内

% cari:车的序号

% RoadMap: 路的可达性矩阵

% Car:当前各辆车的状态

% Roaddistance:每条路的长度

% Outroad:所有可能出去的路

% RoadCarNum: 当前每条路上有多少车

car = Car{cari}; %当前正在考虑的车

nextroad = car.road;

nextdistance = car.distance;

if car.distance >= Roaddistance(Car{cari}.road) && ismember(car.road,Outroad)

nextstate = 0;

return;

else

nextstate = 1;

end

nextdistance = nextdistance + 1;

if (Roaddistance(nextroad) < nextdistance)

nextroad = chooseRoad(cari,nextroad,RoadMap,Car,Roaddistance, RoadCarNum); %%%交叉路口选择路的方向

nextdistance = 1;

% disp(strcat('car', num2str(cari)));

% disp(strcat('change road to', num2str(nextroad)));

end

for i = 1 : (cari - 1)

if(Car{i}.state == 1 && Car{i}.road == nextroad && Car{i}.distance == nextdistance)

%%%% 车辆不能走，

nextroad = car.road;

nextdistance = car.distance;

end

end

红灯时车辆移动函数：

function [nextroad,nextdistance,nextstate]=nextdir\_red(cari,RoadMap,Car,Roaddistance,Outroad, RoadCarNum)

% 函数 nextdir 返回 cari 这辆车这一秒之后的状态

% nextroad: 1s后所在的路

% nextdistance:1s后在路上距路入口的逻辑距离

% nextstate: 1s后，该车是否还在区域内

% cari:车的序号

% RoadMap: 路的可达性矩阵

% Car:当前各辆车的状态

% Roaddistance:每条路的长度

% Outroad:所有可能出去的路

% RoadCarNum: 当前每条路上有多少车

car = Car{cari}; %当前正在考虑的车

nextstate = 1;

nextroad = car.road;

nextdistance = car.distance;

if car.distance >= Roaddistance(Car{cari}.road) && ismember(car.road,Outroad)

return;

end

nextdistance = nextdistance + 1;

if(Roaddistance(nextroad) < nextdistance)

nextroad = chooseRoad(cari,nextroad,RoadMap,Car,Roaddistance, RoadCarNum); %%%交叉路口选择路的方向

nextdistance = 1;

end

for i = 1 : (cari - 1)

if(Car{i}.state == 1 && Car{i}.road == nextroad && Car{i}.distance == nextdistance)

%%%% 车辆不能走，

nextroad = car.road;

nextdistance = car.distance;

end

end

画图函数：

main\_test\_figure:

load('100Dt.mat');

Dt\_100 = Dt\_Sum;

load('200Dt.mat');

% load('200Dt0.mat');

% load('200dtDelta.mat');

Dt\_200 = Dt\_Sum;

% Dt0\_200 = cell2mat(Dt0);

% DtDelta\_200 = dtDelta;

load('400Dt.mat');

Dt\_400 = Dt\_Sum;

load('600Dt.mat');

Dt\_600 = Dt\_Sum;

load('800Dt.mat');

Dt\_800 = Dt\_Sum;

x=1:500;

plot(x, Dt\_100, x, Dt\_200,x, Dt\_400,x, Dt\_600,x, Dt\_800, 'LineWidth', 2);

legend('小区路长100', '小区路长200', '小区路长400', '小区路长600', '小区路长800');

xlabel('时间/s');

ylabel('累积延时/s');

main\_test\_figure\_2:

test\_x = 1 : 500;

% subplot(1, 2, 1)

% plot(test\_x, Dt\_Mat,'r', test\_x, Dt0\_Mat)

% legend('小区开放', '小区不开放');

% xlabel('时间/s');

% ylabel('车辆平均延误时间Dt/s')

% % axis([0 500 7 10])

% subplot(1, 2, 2)

% plot(test\_x, Dt\_Sum, 'r', test\_x, Dt0\_Sum, 'b');

% legend('小区开放', '小区不开放');

load('type1\_Dt.mat');

type1\_Dt = Dt\_Sum;

load('type2\_Dt.mat');

type2\_Dt = Dt\_Sum;

load('type3\_Dt.mat');

type3\_Dt = Dt\_Sum;

plot(test\_x, type1\_Dt, 'r', test\_x, type2\_Dt, 'b', test\_x, type3\_Dt, 'LineWidth', 2);

xlabel('时间/s');

ylabel('车辆累积延误时间Dt/s');

legend('1号小区', '2号小区', '3号小区');

Dt\_Sum(end)

Dt0\_Sum(end)

main\_test\_figure\_3:

load('type1\_Dt.mat');

type1\_Dt = Dt\_Sum;

load('type2\_Dt.mat');

type2\_Dt = Dt\_Sum;

plot(test\_x, type1\_Dt, 'r', test\_x, type2\_Dt, 'b', 'LineWidth', 2);

xlabel('时间/s');

ylabel('车辆累积延误时间Dt/s');

legend('1号小区', '2号小区');