基于微观离散动力学系统的小区开放问题的研究

一、 摘 要

2016 年小区开放成为热点问题，本文建立了基于马氏距离的 TOPSIS 方法评价模型，与小区交通网络微观离散动力学系统模型，针对小区开放对周边道路的影响问题进行了分析与评价。

对于问题一，本文从道路的通畅性与安全性两个角度出发，分别设立了通行效率和通行安全两个一级指标，下设拥挤度、混乱度和出行时间延误、干道事故率、事故易发速率段频率五个二级指标。通过运用 VISSIM 软件对小区进行交通仿真，同时确定了各二级指标的相关性。为了区分各指标的重要程度，本文采用分形维数法进行权值的计算，求解得出，从通畅性指标的重要程度来看：拥挤度＞混乱度＞出行时间延误；从安全性角度来看：事故易发速率段频率＞干道事故率。接着本文运用基于马氏距离的 TOPSIS 排序方法，对各指标进行综合评价。分析得出，开放小区后能改善周边道路通行状况。

对于问题二，通过对小区开放前后道路与通行状况的比较分析，本文建立了小区交通网络微观离散动力学系统模型。通过分析，小区开放前主要通行路线为主干道，道路宽，车流量大，以此建立了小区开放前直行主干道路模型。小区开放后，由于增设了交叉路口，使得车辆需要进行路线选择，以此建立了小区开放后增设交叉路口模型，与小区开放后通行道路选择模型。在充分考虑各种阻碍车辆行驶的因素的基础上，本文结合路口的转弯与信号灯规则，成功对小区开放前后的动态交通网络进行建模。

对于问题三，本文基于元胞自动机的思想，从小区结构的角度选取了向心式布局、片块式布局、轴线式布局和混合式布局四种布局结构，在车流量上考虑了流量较大的双向流通和流量较小的混合流通。根据实际的小区结构图，运用问题一、二的模型，定量评价小区开放对周边道路的影响。基于问题一的评价模型对四种布局小区开放前后各指标进行对比分析，得出当车流量一定时，不同小区开放都会使得道路通行状况得到改善和提高，但是不同布局改善程度不同，当车流量较小时，混合式布局改善程度最大，轴线式布局改善程度最小，当车流量较大时，混合式布局改善程度最大，向心式布局改善程度最小。小区布局一定时，不同车流量，道路通行状况的改善程度也不同：片块式布局与向心式布局适合于车流量较小的情形，轴线式布局、混合式布局适合于车流量较大的情形。同时，对混合式布局小区进行深入分析，得出开放小区后车辆拥堵现象明显减弱，混合流通与双向流通会对车辆道路选择产生影响。

对于问题四，本文根据问题三的结果，从道路通畅性、交通安全两个角度，向城市规划和交通管理部门提出了关于交通通行的小区开放合理化的建议。

关键词：TOPSIS　分形维数法　微观离散动力学系统　Dijkstra 算法　状态转移自动机

二、 问题重述

2.1 问题的背景

2016 年 2 月 21 日，国务院发布《关于进一步加强城市规划建设管理工作的若干意见》，其中第十六条关于推广街区制，原则上不再建设封闭住宅小区，已建成的住宅小区和单位大院要逐步开放等意见，引起了广泛的关注和讨论。

除了开放小区可能引发的安保等问题外，议论的焦点之一是：开放小区能否达到优化路网结构，提高道路通行能力，改善交通状况的目的，以及改善效果如何。一种观点认为封闭式小区破坏了城市路网结构，堵塞了城市“毛细血管”，容易造成交通阻塞。小区开放后，路网密度提高，道路面积增加，通行能力自然会有提升。也有人认为这与小区面积、位置、外部及内部道路状况等诸多因素有关，不能一概而论。还有人认为小区开放后，虽然可通行道路增多了，相应地，小区周边主路上进出小区的交叉路口的车辆也会增多，也可能会影响主路的通行速度。

2.2 问题的提出

建立数学模型，就小区开放对周边道路通行的影响进行研究，为科学决策提供定量依据，解决以下问题：

1.选取合适的评价指标体系，用以评价小区开放对周边道路通行的影响。

2.建立关于车辆通行的数学模型，用以研究小区开放对周边道路通行的影响。

3.选取或构建不同类型的小区，从小区结构及周边道路结构、车流量等角度，应用建立的模型，定量比较各类型小区开放前后对道路通行的影响。

4.根据研究结果，向城市规划和交通管理部门提出关于交通通行的小区开放合理化建议。

三、 问题分析

3.1 问题一的分析

要建立评价指标体系评价小区开放对周边道路通行的影响，可以从道路的通畅性和安全性两个角度考虑。设立拥挤度、混乱度、出行时间延误、干道事故率和事故易发速率段频率五个二级指标。为了区分各指标的重要程度，采用分形维数法进行权值的计算。由于小区开放对周边道路通行的影响是一个多方面的问题，可以采用逼近理想解的排序方法（TOPSIS），对不同时刻不同道路的通行进行定量分析评价。

3.2 问题二的分析

对于问题二，要研究小区开放对周边道路通行的影响，首先对小区开放前后道路的通行状况进行比较分析，小区开放前主要通行路线为主干道，道路较宽，车流量较大，以直行为主，可以建立小区开放前直行主干道路模型来刻画。小区开放后，道路增多，内部道路较窄，由于增设了交叉路口，车辆需要进行路线选择，在充分考虑了其他车辆与信号灯对车辆行驶速度的阻碍的基础上，我们可以结合路口的转弯与信号灯规则，通过增加对转弯时的分析与设计，建立小区开放后增设交叉路口模型，再运用最短路算法对车辆路线选择进行刻画，就可以建立小区开放后通行道路选择模型。这就构成了小区交通网络微观离散动力学系统模型。这样就可以详细地刻画小区开放前后的动态交通状况。

3.3 问题三的分析

对于问题三，通过结合实际，构建不同类型的小区，从小区结构及周边道路结构、车流量等角度，应用建立的模型，定量比较各类型小区开放前后对道路通行的影响。由于真是数据的缺乏与可靠度低，我们可以基于元胞自动机的思想，基于实际情况，构建和模拟交通状况。从小区结构的角度，我们可以选取向心式布局、片块式布局、轴线式布局和混合式布局四种布局结构，从车流量的角度，可以考虑模拟上下班时期和平时 的情况，分别考虑流量较大的双向流通和流量较小的混合流通。根据实际的小区结构图，运用问题一、二的模型，对比不同车流量下，不同小区各时刻各道路的车辆数，计算问题一的各项指标，定量分析评价小区开放对周边道路的影响。

3.4 问题四的分析

对于问题四，可以根据问题三的结果，从道路通畅性、交通安全问题和社会服务的改进问题三个角度，向城市规划和交通管理部门提出了关于交通通行的小区开放合理化的建议。

四、 模型假设

假设 1：不考虑行人及非机动车在交通中的影响；

假设 2：小区周边道路为双向主干道，小区内部道路为单向次干道；

假设 3：所有信号灯时长相同；

假设 4：不考虑车辆损坏、交通事故等意外事件的发生；

假设 5：所有车辆的型号、性能相同。

五、 符号说明

符号	说明
£	拥挤度
Ω	混乱度
$\overline{t}$	出行时间延误
P	事故易发速率段频率
K	干道事故率
D_j	第 j 个指标的分形维数
ω_j	第 j 个指标的权重
S^+、S^-	正理想解、负理想解
d_i^+、d_i^-	与正理想解、负理想解的马氏距离
$a_i(t)$	第 i 辆车行驶的加速度
α	粘滞系数
η	阻滞系数
v_m	车辆行驶最大速度
ρ	车流密度

六、 模型的建立与求解

6.1 问题一：基于马氏距离的 TOPSIS 方法评价模型

研究小区开放对周边道路通行的影响是一个多方面的问题。逼近理想解的排序方法（TOPSIS）方法是一种解决有限方案多指标决策问题的有效方法，通过计算个方案与决策问题的理想解和福利详解的距离和贴合度，从而对各方案进行排序确定优劣。为了准确、全面、系统地刻画小区开放的影响，本文以小区周边某条干道为例，建立了综合评价指标体系，使用分形维数法确定权重，建立基于马氏距离的 TOPSIS 方法评价模型对开放小区的影响进行评价。

在本问中为了简化问题，我们提出以下几点假设：

假设 1：暂不考虑信号灯的设立；

假设 2：小区周边道路为主干道，设为双向车道，小区内部次干道为单向车道；

假设 3：干道交通暂不考虑行人及非机动车，只考虑机动车的行驶。

6.1.1 评价指标体系的建立

在小区开放对周边道路通行的影响时本文从道路的通畅性和是否安全两个评价方面入手，分别设立了对应的通行效率、通行安全两个一级指标，和拥挤度、混乱度、出行时间延误、干道事故率和事故易发速率段频率五个二级指标。建立的评价指标体系如表 1 所示。

表 1. 综合评价指标体系

评价方面	一级指标	二级指标
通畅性	通行效率	拥挤度 混乱度 出行时间延误
安全性	通行安全	事故易发速率段频率 干道事故率

6.1.2 二级指标的定义

（1）拥挤度

我们定义小区周边道路为主干道，小区内部道路为次干道。小区周边道路根据与小区之间的距离关系分为相邻主干道，次主干道。主干道与主干道、主干道与次干道的交汇口称为主交叉口。

对于拥挤度£的设定，我们以道路通行密度即单位面积车辆数来衡量道路车辆与车辆间的拥挤程度。小区开放后，次干道可以起到一定的引流作用，使主干道在一定路段内行车数量减少，但是由于交叉路口的出现，会使得小区周围道路局部车辆数增多。为了量化该现象对道路通行的影响，采用拥挤度指标定量刻画。如图 1 假设某小区周边某一干道长度为l_1,干道宽度为l_2，则干道面积$S=l_1\times l_2$。假设某一时刻在该干道上的车辆总数为q，则干道拥挤度为

$$£=\frac{q}{S} \qquad (1)$$

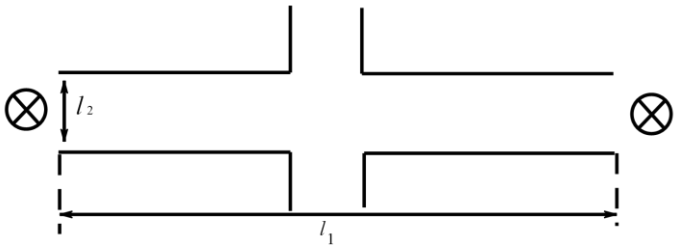

图 1. 小区周边某一主干道示意图

(2) 混乱度

车辆通过小区周边某一干道时如果保持匀速行驶，说明该通路通行状态比较稳定有序，如果车辆通行速度处于时刻变化，则说明桐庐通行状态具有波动性，在一定程度上反映了道路的混乱程度。而车辆车速状态的变化可以用车辆加速度 $a(t)$ 随时间的变化来描述。假设在单位时间内（本节以一小时为单位时间段），通过小区周边某一干道的车次数车辆总数为 n，第 i 辆车在干道上行驶的加速度为 $a_i(t)$。混乱度 Ω 的表达式为：

$$\Omega=\sqrt{\sum_{i=1}^{n} a_i^{\,2}(t)} \qquad (2)$$

(3) 出行时间延误

小区开放前周边主干道如图 2。此时，车辆出行时间延误分为两部分，一是正常行驶下的时间延误（l_1 段），二是排队时间延误（l_2 段）。正常行驶路段小，车辆速度由于受该道路内车流量、车流密度、车流速度的影响，造成该车无法正常以理想速度 v_I 行驶。（v_I 为车辆在道路最大行车速度，只与道路本身有关，与其他因素无关）小区开放后，会对车辆产生分流作用，使行驶在干道上的车辆数发生变化。车辆在被迫行驶状态下造成的时间延误等于车辆从进入干道时刻开始至小车离开干道时刻结束的时段实际行驶的时间 c_i 与理想速度下车辆的理想行驶时间 b 的差。车辆总数为 n，道路总长度为 L。

$$b=\frac{L}{v_I} \qquad (3)$$

$$\bar{t}=\frac{1}{n}\sum_{i=1}^{n}(c_i-b),\quad i=1,2,\cdots,n \qquad (4)$$

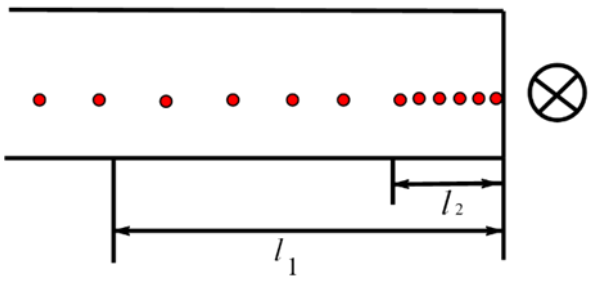

图 2. 小区开放前主干道示意图

(4) 事故易发速率段频率

小区周边某干道上某一时刻不同速率段分为低速段（0-20km/h）、中速段（20km/h -40km/h）、高速段（40km/h -60km/h）。车辆频率（处于该车速段的车辆占总数的百分比）不同，我们将高速段称为事故易发速率段，对已有数据进行统计分析可得不同时段高速段的频率，该指标在一定程度上反映了该路段某时段的行车安全性。

$$P=\frac{t_{高速段}}{t_{总}} \tag{5}$$

(5) 干道事故率

由实际数据可知，当£=0 时，干道事故率 $K=0$；当 $\bar{v}=0$ 时，$K=0$。不妨设 £=0.05，当 $\bar{v}=60km/h$ 时，$K=0$。由此拟合出干道事故率与混乱度与平均速度的关系为：

$$K=e^{12-\frac{1}{£}} \tag{6}$$

6.1.3 基于 VISSIM 的小区仿真

我们选取最简单的小区模式，运用 VISSIM 对其进行交通仿真。C、O、Q、H 所围区域内部代表小区，JL、FP 代表小区内部道路，其余线段代表小区外部道路。所得数据见附录。

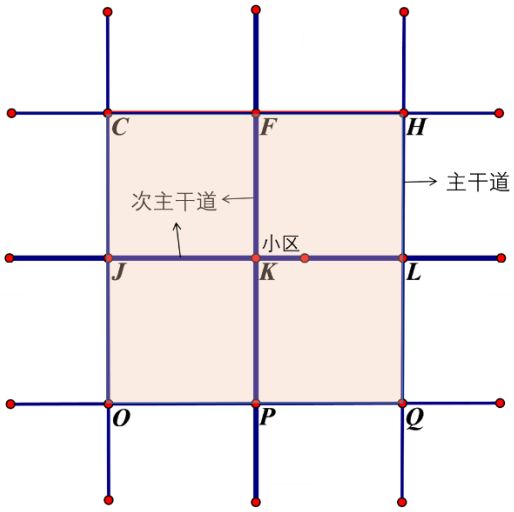

图 3. 仿真模拟小区示意图

6.1.4 二级指标的相关性分析

根据仿真结果，从二级指标的定义来看，有的指标间存在明显的相关性，进一步使用相关系数来定量检验指标间的相关性。

（1）通行效率指标的相关性分析

表 2. 通行效率指标的相关性

	拥挤度	混乱度	出行时间延误
拥挤度	1	0.6331	0.8523
混乱度	0.6331	1	0.7174
出行时间延误	0.8523	0.7174	1

从表 2 可以看出，拥挤度和出行时间延误存在非常强的相关性。拥挤度和混乱度的相关性较弱。

（2）通行安全指标的相关性分析

表 3. 通行安全指标的相关性

	事故易发速率段频率	干道事故率
事故易发速率段频率	1	0.8324
干道事故率	0.8324	1

从表 3 可以看出，干道事故率和事故易发速率段频率存在非常强的相关性。

6.1.5 使用分形维数法确定权重

考虑到数据的离散性，本文使用分形维数法，该方法被广泛应用于多指标参数权重的确定，具有极强的可靠性。

根据分形维数法的一般过程，首先将原始数据进行规范化处理，计算分形维数 D_j。

设 k_{ij} 为第 i 个研究对象第 j 个指标的位次，$P_{k_{ij}}$ 为对应的数值，$P_{1_{ij}}$ 为第 j 个指标的第一位次的数值，然后利用 k_{ij} 与 $P_{k_{ij}}$ 组成数组，利用

$$\ln P_{k_{ij}} = \ln P_{1_{ij}} - q_j \ln k_{ij}, i = 1,2,\cdots,m, j = 1,2,\cdots,n \quad (7)$$

进行回归，就可计算出 q_j，接着计算出 $D_j = 1/q_j$，就是分形维数。

最后求出第 j 个指标的权重 ω_j 为

$$\omega_j = \frac{D_j}{\sum_{j=1}^{n} D_j}, j = 1,2,\cdots,n \quad (8)$$

得到的权重的计算结果如表 4 所示。

表 4. 指标权重

一级指标	权重	二级指标	权重
通行效率	0.6081	拥挤度	0.3604
		混乱度	0.3293
		出行时间延误	0.3103
通行安全	0.3919	事故易发速率段频率	0.5357
		干道事故率	0.4643

6.1.6 结果分析：

从通畅性角度来看：拥挤度＞混乱度＞出行时间延误

从安全性角度来看：事故易发速率段频率＞干道事故率

6.1.7 基于马氏距离的 TOPISIS 方法的评价模型

由表 3 可知，各指标间存在一定程度的相关性，而传统的 TOPISIS 方法所采用的欧氏距离不能有效地处理指标之间的共同信息。指标之间的相关性越强，重复计算的部分也越大。本文借鉴印度统计学家马哈拉诺比斯（P.C.Mahalanobis）的思想，建立基于马氏距离的 TOPISIS 法评价模型，该方法通过样本协方差矩阵放映不同决策指标的相关性，能够有效剔除决策信息的重复，决策数据的任意非奇异线性变换不会影响该方法决策结果，进一步完善了 TOPSIS 决策方法体系。

按照 TOPSIS 方法的一般步骤，首先建立规范化矩阵，

$$Z=(z_{ij})_{m\times n}=\begin{bmatrix} 0.077732 & 0.008403 & 0.159603 & 0.102869 & 0.331985 \\ 0.148684 & 0.094521 & 0.308565 & 0.154303 & 0.382505 \\ 0.323642 & 0.241194 & 0.404327 & 0.360041 & 0.324768 \\ 0.312866 & 0.576529 & 0.457527 & 0.46291 & 0.259814 \\ 0.209223 & 0.008072 & 0.113495 & 0.051434 & 0.41859 \\ 0.425832 & 0.086578 & 0.340485 & 0.154303 & 0.339202 \\ 0.580206 & 0.305518 & 0.425607 & 0.46291 & 0.476326 \\ 0.455496 & 0.706776 & 0.44334 & 0.617213 & 0.230946 \end{bmatrix}$$

然后确定正理想解 S^+ 和负理想解 S^-

$$S^+=\{0.077732,0.008072,0.113495,0.051434,0.230946\}$$

$$S^-=\{0.580206,0.576529,0.44334,0.6172313,0.476326\}$$

最后计算各方案与正理想解和负理想解的马氏距离

$$d_i^+=\{0.1606,0.5629,0.2825,0.4695,0.3815,0.3245,0.4587,0.6968\}$$

$$d_i^-=\{0.5927,0.3444,0.5650,0.2792,0.3989,0.2895,0.5895,0.7455\}$$

6.1.8 结果的分析

由此计算出五个二级指标的贴合度为

表 5. 二级指标贴合度

贴合度	时段 1	时段 2	时段 3	时段 4
开放前	0.1039	0.2441	0.4737	0.6718
开放后	0.1815	0.3565	0.6547	0.7888

从结果看出，开放小区有利于周边道路通行状况的改善。

6.2 问题二：小区交通网络微观离散动力学系统模型的建立

小区开放的交通能力与小区的结构、周边道路的结构及车流量有着密切的关系。

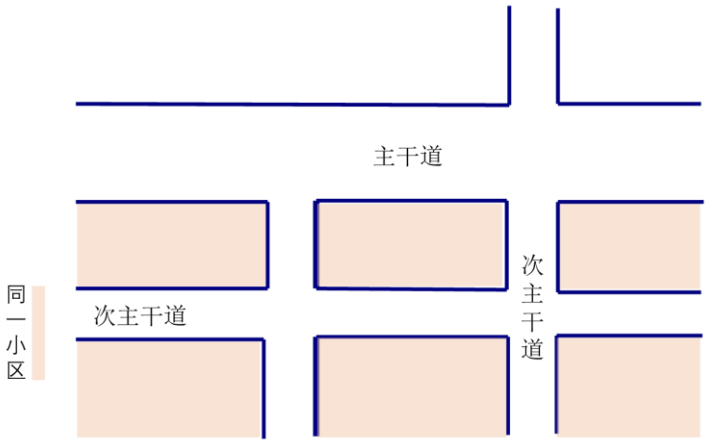

图 4. 小区主次干道示意简图

图 4 为小区主次干道示意简图。在小区开放前车辆从主干道通过，道路宽阔，大多数情况处于直行状态。小区开放后主次干道会产生十字路口的交叉和 T 型路口的交叉，同时可选道路增多，但内部道路宽度较窄，车辆需要选择合适的路线通行。

6.2.1 小区开放前直行主干道路模型的建立

在小区开放前，车辆通过小区外围的主干道通过小区，主干道的特点是道路宽阔、转弯少、信号灯设置少。因此，我们建立直行主干道路模型来刻画小区开放前的交通情况。

（1）粘滞系数的设定

车辆在道路上行驶通常不会一帆风顺，会受到一些其他因素的干扰，如行人、道路宽度和车流量等。由于车辆行驶是双向行驶如图 3，以 A 车为例，对其产生阻碍的是同向行驶的车辆 B，而 C、D 对其阻碍可以忽略不计。根据流体物理的思想，我们设计了粘滞系数α，可以很好地反映车辆同向行驶受到的阻碍程度。但是由于行人的随机性，随机的设定不能反映真实的情况，反而对结果会增加干扰，故本文重点考虑道路宽度和车流量两个因素。

$$\alpha = k\frac{\rho}{v_m} \tag{9}$$

其中k为归一化常数，v_m表示车道允许车辆的最大速度，ρ为车道的车流密度。主干道路允许速度较大，车流密度较小，粘滞系数较小，对车辆的阻碍较小；相对的，次主干道路允许速度较小，车流密度较大，粘滞系数较大，对车辆的阻碍较大。

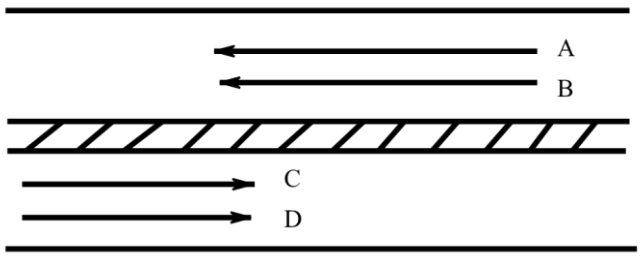

图 3.双向车辆行驶示意图

当车辆在前方无车辆和信号灯的道路上行驶时，会尽量提高速度来抵达目的地。为了刻画这一现象，假定道路允许最大速度为v_m，则车辆在无障碍道路上会保持匀加速行驶，加速度为在阻滞情况下的最大加速度，在数值上，

$$a(t) = \frac{dv(t)}{dt} = a_{\max} - \alpha v(t) \tag{10}$$

（2）行驶速度的设定

车辆的加速度与当前道路的状况和当前车速有关。$a_{\max}$为车辆自身允许的最大加速度，为车辆固有属性。在存在其他因素的干扰情况下，当车辆处于起步状态时，受到道路状况的影响会使加速度减慢。

当车辆正常行驶时，车辆前方不会有其他车辆和信号灯的限制，这时驾驶员可以在道路允许的情况下以最大加速度进行加速，即车辆下一时刻的速度为：

$$v(t+1)=v(t)+a(t)\Delta t \tag{11}$$

考虑到实际情况中车辆不可能无限加速，这里限定车辆速度不超过道路允许最大速度，即当$v(t+1)>v_m$时，令$v(t+1)=v_m$。

（3）障碍减速的设定

由于人眼可视范围的限定，驾驶员只有在靠近信号源一定距离时才会做出反应。当车辆发现前方有其他车辆时，为了避免发生碰撞，司机会选择减速慢行，并将速度控制在前车的速度。为刻画这一现象，设L_2为车辆与前车的距离，d为车辆间的安全距离，l为车身长度。a_1为车辆减速的最大加速度。认为当车辆与前车车尾距离大于两倍安全距离时是安全的，车辆可以保持正常的变加速行驶；当车辆进入两倍安全距离时就可能会产生危险，这时车辆开始以最大减速度在阻滞存在的情况下减速。故设计车辆下一时刻的速度计算公式如式下：

$$v(t+1)=v(t)-(1-\alpha)a_1\Delta t \tag{12}$$

6.2.2 小区开放后增设交叉路口模型的建立

小区开放后，不仅道路增加，最主要的改变是交叉路口的增加，它承载了小区内外的车辆流通，是交通的枢纽，使得车辆行驶增加了转弯与交汇。因此格外重要。

在生活中，交叉路口共有十字路口、三叉路口和折角路口三种情况如图 5 所示。

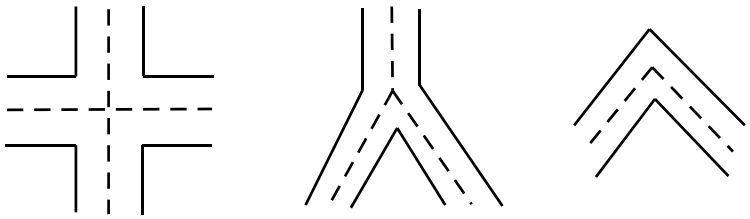

图 5. 三种交叉路口

（1）十字路口

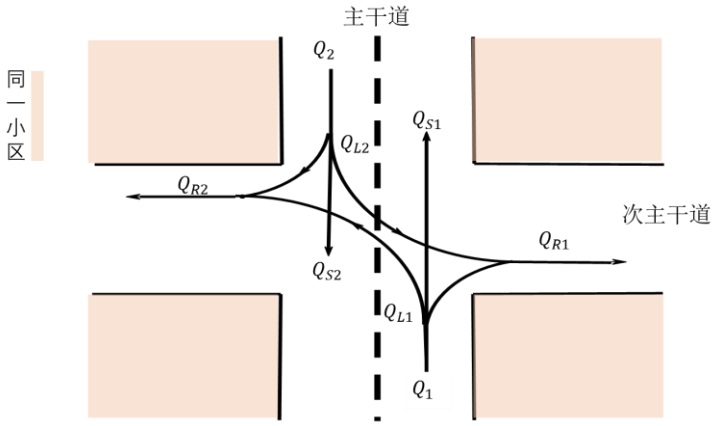

图 6. 小区开放后十字路口示意图

车辆进行转弯时要考虑两个问题，即信号灯与车流的影响。如图 6 为十字路口车辆转弯示意图，车辆在通过路口时有直行、左转和右转三种选择。右转一般情况下不受信号灯控制，可以畅通无阻，直行和左转都会对车道上的车辆产生影响。假设向上总车流量为Q_1，向下总车流量为Q_2，它们分别分成三束支流：向左、向右和直行。

$$Q_i = Q_{Si} + Q_{Ri} + Q_{Li} (i = 1,2) \tag{13}$$

通常情况下，直行与左转是不能同时进行的，它们会发生相互干扰，而右转只会增加下一道路的车流量，不会对路口产生影响。但分流时直行车辆数大于左转车辆数，即$Q_{Si} > Q_{Li}$。综合来看，我们认为左转的影响所占比重更大，设为 0.6，对应的直行的比重设为 0.4。由此得出路口车流对车辆通行的影响系数：

$$\eta = f(Q_S, Q_L) = 0.4\frac{Q_S}{Q_{S\max}} + 0.6\frac{Q_L}{Q_{L\max}} \tag{14}$$

其中Q_S、Q_L分别为同一时刻直行和左转的车辆数，$Q_{S\max}$、$Q_{L\max}$分别为同一时刻直行和左转的车辆数最大值。

这个阻滞系数是道路交通对车辆的影响，与车辆通过交叉路口时的粘滞系数相结合，新的交叉路口粘滞系数α'的表达式为：

$$\alpha' = \alpha + \eta \tag{15}$$

（2）T 型路口

在小区开放后，主干道与次主干道会产生 T 型路口。由于 T 型路口是三叉路口的一种特殊情况，本文假设所有的三叉路口的转弯规则都与 T 型路口一致。

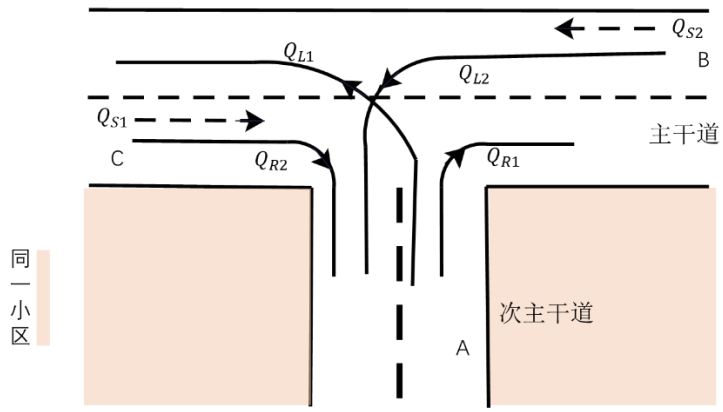

图 7. T 型路口的转弯示意图

对于 T 型路口，当车辆从 A 方向进入时，路口车流对车辆通行的阻滞系数只与左转车流量有关，

$$\eta'=f(Q_L)=\frac{Q_L}{Q_{L\max}} \tag{16}$$

从 B、C 口进入的车辆的阻滞系数仍保持不变。

①当有信号灯时

设车辆的可视距离为C，设车辆距路口的距离为L_1。当车辆在可视范围内出现了信号灯立即做出减速的反应，并且以一个恒定的加速度a_0减速，为了保证每辆车都能在路口信号灯前减速到 0，应该以一个恒定的加速度减速到 0。由动力学公式知：

$v^2(t)=2a_0L_1$

由此得到a_0的计算公式如下：

$$a_0=\frac{v^2(t)}{2L_1} \tag{17}$$

由于一次信号灯变化时间有限，不一定所有的车辆都可以在一次信号灯变换中通过路口。因此我们规定，一次可以通过的车辆队伍长度为一次单位时间内汽车加速可达的距离，即$\frac{1}{2}a_2t_0^2\eta$，其中t_0为单位时间，a_2为车辆的加速度。

②当无信号灯时

- 左转和直行的车辆：距离路口最近的车辆优先通过；
- 右转的车辆：无需等待，按次序直接通过。

6.2.3 小区开放后通行道路选择模型的建立

小区开放后的另一个改变是车辆需要根据实际路况进行路线选择。车辆自起点至终点的路线选择并不能随机选择，而是根据路线的长度、车道的流量和交叉路口的流量等多重因素进行的综合决策。因此我们对传统的最短路算法进行改进。

在直行道路模型和交叉路口转弯模型的基础上，本文使用道路权值Γ_1和交叉路口权值Γ_2两个权值对交通网路进行考察，道路值由道路的最大阻滞系数、路线总长度和车辆最大速度决定，交叉路口值由选择路线中所有红灯时间的总和决定。由于经过信号灯时不一定会遇到红灯，故这里乘遇到红灯的概率 1/2。在确定网络图中的边权与点权后，采用最短路算法，求解该个体达到目标点的节点通行序列。

$$\begin{cases} \Gamma_1 = \dfrac{\alpha_{\max} \cdot S}{v_m} \\ \Gamma_2 = \dfrac{1}{2}\sum_{i=1}^{h} \tau_i, i = 1,2,\cdots,h \end{cases} \tag{18}$$

其中$\alpha_{\max}$为最大阻滞系数，S为路线总长度，v_m为车辆最大速度，τ_i为每个信号灯的红灯时长，h为信号灯总数。

6.3 问题三：对不同结构小区的分析与评价

小区结构及周边道路结构、车流量等都会对开放式小区的交通产生影响。由于实际数据的查找不够全面且难以实现，现有的交通仿真工具 VISSIM 又并不能实现贴合实际的仿真，故本文基于元胞自动机的思想，结合实际数据，运用建立的模型，定量比较各类型小区开放前后对道路通行的影响。

6.3.1 小区的选取与构建

通过查阅相关专业资料得知，中国现代居民小区大体分为四种类型：向心式布局、片块式布局、轴线式布局和混合式布局。本文选取了四种类型的小区代表如图 8 所示。

图 8. 四种典型小区结构图

本文将圆弧道路分割成几个小线段，化曲为直。如图 8 所示即为四个小区的内部及周边道路结构。

其中道路的宽度严格按照比例尺进行计算，分别运用区域网络交通模型对实际路况进行模拟与仿真。

由选取的小区结构的实际情况得知，比例尺为 1:2500。设车长为 5m，可视距离为 50m，安全距离为 30m，信号灯的转换周期为 15s, 设定车辆最大加速度为 $2m/s^2$，最大速度为 40km/h，

6.3.2车流量的设定

在考虑车辆投放问题时，我们按泊松分布在小区外投放车辆总数随时间变化的车辆。从车流量的角度，我们考虑上下班时期和平常时期，以此为基础设定车流量大小。

（1）双向流通

在上下班时期，车辆出行的主要方向呈现对流的状态，我们在小区两端同时投放一半数量的车辆。由于上班人群集中，车流量较大。

（2）混合流通

在平常时期，车辆的行驶方向是杂乱无章的，因此我们在四周节点随机设置车辆的目的地和初始点进行模拟。由于平常时期人们出行相对稀松，车流量较大。

6.3.3 车辆状态转移的选择机制

根据模型的刻画，我们设计出了车辆状态转移的选择机制，如图 9。

车辆在直行道路行驶时首先判断前方有无车辆，再根据所处状态做出反应。若前方无车辆，则继续判断可视范围内有无信号灯。若遇到红灯，当处于行驶状态时，则变为遇红灯的减速状态，若正处于减速状态则减速到停车，若正处于停车状态则保持此状态。若遇到绿灯，正处于行驶状态时则保持此状态；若正在减速则继续减速，若正处于停车状态；则起步加速通过。若前方看不到红绿灯，则保持现有状态。当车辆发现前方有车，若正在行驶或减速，则跟车减速，若停止则保持此状态。

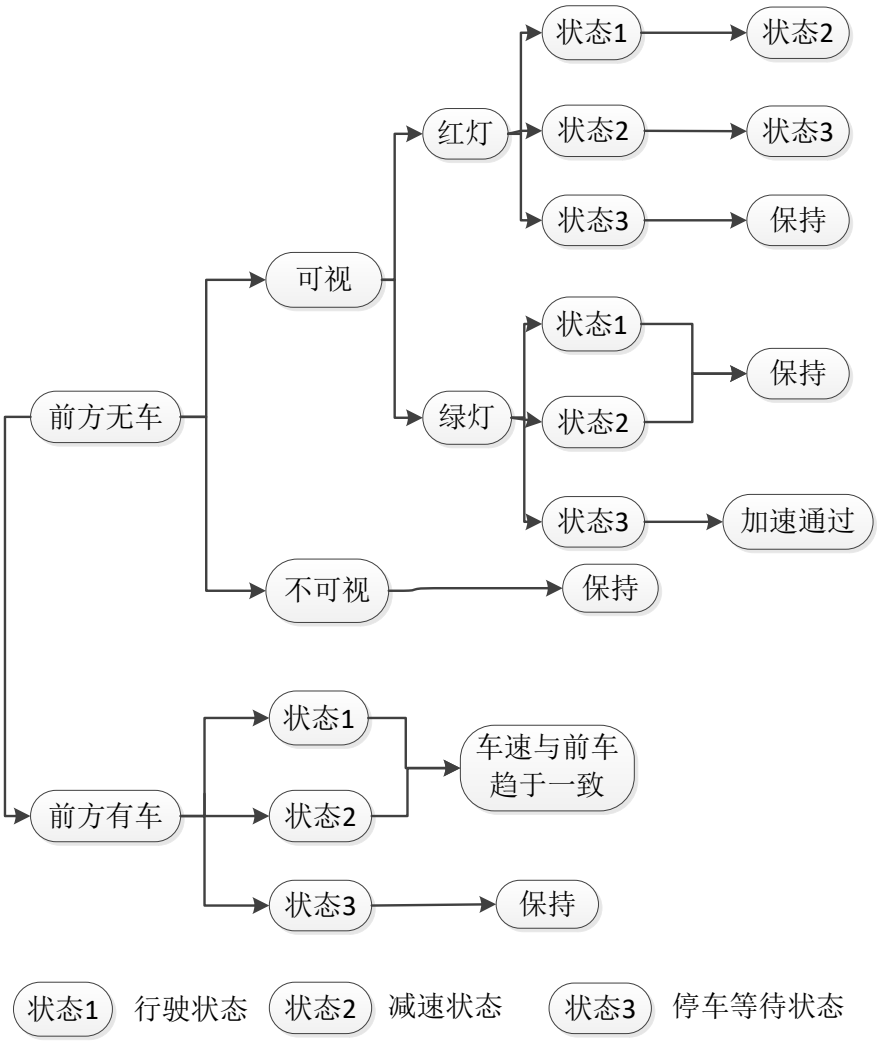

图 9. 车辆状态转移选择机制

表 6.状态转移自动机算法框架

Step1:构建小区区域网络图，以路口为节点，道路为双向可行边。 赋予节点额外属性：信号灯状态、瞬时阻滞系数。 赋予边额外属性：宽度、粘滞系数
Step2:依据输入流特性，产生一定数量的出行需求，即随机初始化目标点与终点。同时调用改进的 Dijkstra 算法，计算每个出行需求的边权点权和最短路，即阻抗最短路。
Step3:依据设计好的状态转移自动机依次对所有车辆进行转移，转移规则由当前图中的信息修正。
Step4:更新所有路口节点的信号灯信息。
Step5:更新所有道路和路口的粘滞系数与阻滞系数信息，并依据这些信息，重新计算阻抗最短路决策矩阵。
Step6:判断系统运行次数是否达到要求，如果不足，回到步骤 2 继续循环。

6.3.4结果的分析与评价

（1）基于评价模型小区开放前后各指标的对比分析

在进行双向流通时，

表 7.混合流通四种小区开放前后各指标

		拥挤度	混乱度	出行时间延误	事故易发速率段频率	干道事故率
片块式布局	开放前	8403	714	324	0.002	0.50
	开放后	7344	678	457	0.032	0.51
向心式布局	开放前	7462	826	485	0.007	0.48
	开放后	5863	1143	364	0.006	0.63
轴线式布局	开放前	9436	534	306	0.005	0.69
	开放后	8960	248	732	0.003	0.72
混合式布局	开放前	8532	682	428	0.004	0.43
	开放后	7936	469	658	0.005	0.96

据此计算出四种小区布局的贴合度为：

表 8.混合流通时四种小区开放前后贴合度

贴合度	片块式布局	向心式布局	轴线式布局	混合式布局
开放前	0.4263	0.5834	0.7322	0.6526
开放后	0.5436	0.6826	0.8236	0.9024

在进行混合流通时，

表 9.混合流通四种小区开放前后各指标

		拥挤度	混乱度	出行时间延误	事故易发速率段频率	干道事故率
片块式布局	开放前	9802	714	439	0.003	0.60
	开放后	6222	678	692	0.019	0.67
向心式布局	开放前	7462	826	592	0.010	0.59
	开放后	8163	1143	579	0.008	0.66
轴线式布局	开放前	10236	557	499	0.011	0.71
	开放后	8960	367	888	0.008	0.82
混合式布局	开放前	9625	845	671	0.009	0.50
	开放后	10089	663	859	0.006	0.97

据此计算出四种小区布局的贴合度为：

表 10.混合流通时四种小区开放前后贴合度

贴合度	片块式布局	向心式布局	轴线式布局	混合式布局
开放前	0.5363	0.6244	0.8534	0.7245
开放后	0.6345	0.6822	0.8896	0.8994

进一步的，通过贴合度的计算结果，我们计算出贴合度的变化率如表 10.

表 11.贴合度变化率

贴合度变化率	片块式布局	向心式布局	轴线式布局	混合式布局
双向流通	27.52%	17.00%	12.48%	38.28%
混合流通	18.31%	8.25%	4.24%	24.1%

通过对贴合度和变化率的对比分析得出以下几点分析：

1）当车流量一定时，不同小区开放都会使得道路通行状况得到改善和提高，但是不同的小区类型，改善程度各不相同，当车流量较小时，混合式布局改善程度最大，轴线式布局改善程度最小，当车流量较大时，混合式布局改善程度最大，向心式布局改善程度最小。

2）当小区类型一定时，对于不同车流量，道路通行状况的改善程度也不同：片快式布局与向心式布局适合于车流量较小的情形；轴线式布局、混合式布局适合于车流量较大的情形。

（2）混合式布局小区的深入分析

这里我们针对混合布局模型进行深入分析。通过求解与计算得出混合式布局小区开放前后各时刻各道路的车辆数量分布图如图 10、11 所示。

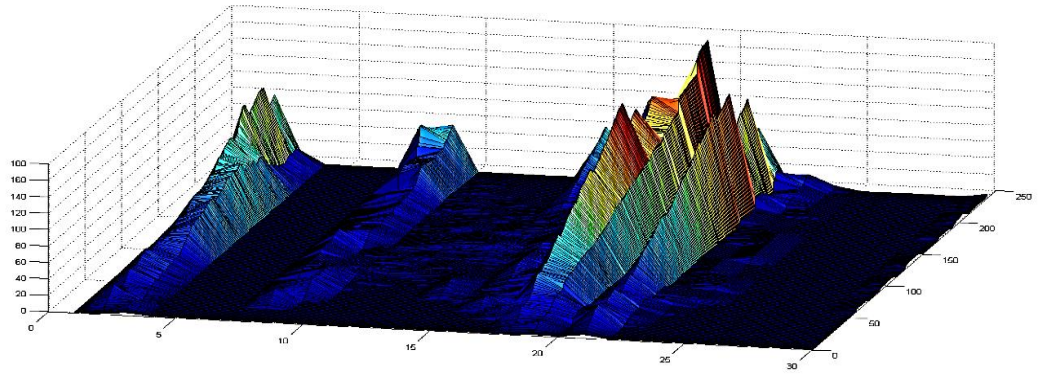

图 10. 混合式布局开放前双向流通车辆分布图

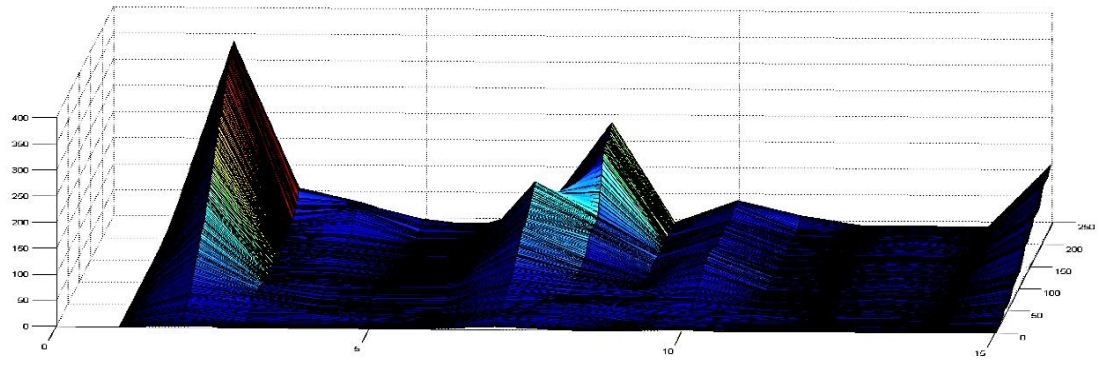

图 11. 混合布局开放后双向流通车辆分布图

图中平面坐标表示道路数与时刻，竖直坐标为车辆数目。通过图像可以看到，开放小区后车辆拥堵现象明显减弱，无论是峰值还是各时刻的总数都明显小于开放前。因此我们判断开放后比开放前的交通状况好。

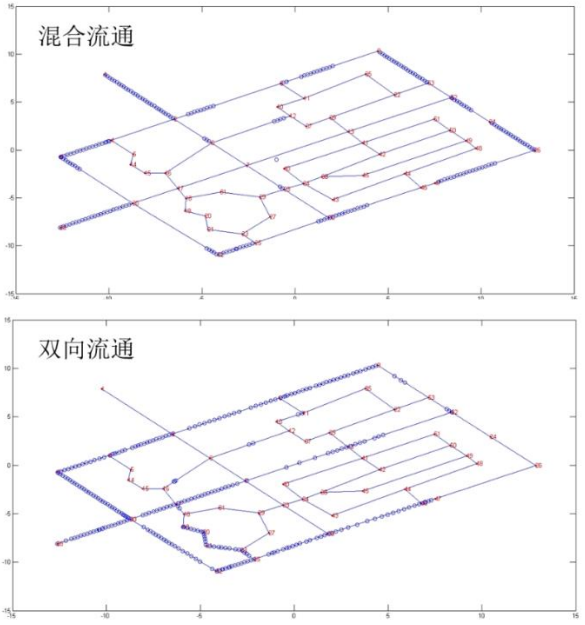

图 12. 混合布局混合、双向流通末态图

图 12 为混合流通与双向流通是混合布局小区末态第（2400 秒）车辆状态图。图中每一个小圆圈代表一辆车，可以发现不同流通情况时车辆聚集的相同点是大多集中在边界处，不同点是双向流通选择次主干道的几率大一些。分析其原因主要是由于双向流通时车流量大，路线最短的主干道聚集大量车辆时，次主干道更加通畅，使得驾驶员更容易选择次主干道，以此来缩短通行时间。

6.4 问题四：基于模型评价分析对小区开放合理化的建议

通过本文的评价与分析，结合实际，本文从道路通畅性、交通安全问题和社会服务的改进问题三个角度，对小区开放合理化提出以下几点建议：

（1）对于道路通畅性：

1）开放小区道路时适当拓宽道路，避免车辆拥堵与事故的发生。

2）合理设置信号灯的位置，避免由于信号灯过多与过少而造成的车辆惯性聚集与拥堵现象的发生。

（2）对于交通安全：

1）合理增设信号灯，避免由于转弯路口车辆交通规则不明确造成的事故。

2）增设路标与避让行人标志，加强交通法律法规的宣传与普及，加强对小区居民的人身安全的保障。

七、 模型分析与评价

7.1.1 模型的优点

1. 本文采用逼近理想解的排序方法（TOPSIS）方法与分形维数法相结合，有效地解决多指标决策问题，尤其是指标间相关性比较强的问题，使得计算简便、便于理解、与其他方法融合性强。

2. 本文采用微观离散的动力学系统来刻画交通网络的各项特征，相对以往的概率性模型以及近些年来的元胞自动机更能详细、客观地描述区域交通网络的状态。

7.1.2 模型的缺点

1. 本文没有考虑行人对该系统的影响，这与实际情况有些出入，不够完善。

2. 本文在问题三中忽略了系统更新过程中先后顺序的影响，在今后的应用中应当改进。

八、 模型的推广

本文采用的基于马氏距离的 TOPSIS 方法评价模型可应用于社会经济等诸多领域，如公司绩效评价、供应链管理、电子商务等。

九、 参考文献

[1] 尤晓，张恩杰，张青熹，现代道路交通工程学，北京，清华大学出版社，北京交通大学出版社，2008。

[2] 李向朋，城市交通拥堵对策 封闭型小区交通开放研究，中国学位论文全文数据库，2014。

[3] 王永明，基于元胞自动机的道路交通堵塞仿真研究，第 22 卷第 9 期，2149-2154，2010

[4] 王正新，基于马氏距离的 TOPSIS 决策方法及其应用，第 29 卷第 2 期，2012

十、 附 录

附录一 问题一数据

		拥挤度	混乱度	出行时间延误	事故易发速率段频率	干道事故率
时段 1	开放前	642	310	45	0.002	0.46
	开放后	1728	297.8	32	0.001	0.58
时段 2	开放前	1288	826	592	0.003	0.53
	开放后	3517	1143	114	0.007	0.47
时段 3	开放前	2673	8898	499	0.011	0.45
	开放后	8960	367	888	0.007	0.66
时段 4	开放前	2584	21209	129	0.009	0.45
	开放后	3762	26074	125	0.006	0.32

附录二 问题三的源程序

```
#include <bits/stdc++.h>
#define inf 100000
using namespace std;
const int numnode=34;//道路节点总数
const int numroad=41;//道路总数
const int putcar=2;//单位时间投放汽车数量
const int firstputcar=4;//一开始投放的车辆
const int seconds=240;//总迭代时间
int fa[numnode+1];//获得最短路径信息
int g[numnode+1][numnode+1];//双节点索引道路编号
int putstart[6]={1,5,7,3,9,2};//随机初始点
int putend[6]={1,5,7,3,9,2};//随机终点
```

```
int roadtime[numroad+1][240];
int nodetime[numnode][240];
int ceshi1[3]={3,28,32};
int ceshi2[3]={52,53,24};//测试序列
int major_dir[numnode+1][numnode+1];//道路口的主要方向判别
double antig[numnode+1][numnode+1];//阻抗权值决策矩阵
double seedis=2;//可视距离
double fixeddis=0.8;//起步距离
double beginspeed=0.22;//起步速度
double length_car=0.2;//车长
double safe_dis=1.2;//安全距离
double tjr=0.8;//统计半径
double kconst=0.00125;//粘滞系数归一化常数
double fconst=0.001;//阻滞系数常数
struct node
{
    double x;
    double y;
    double n1;//主要道路阻滞系数
    double n2;//次要道路阻滞系数
    int had;//有无信号灯
    int is_cross;//是否是交通路口
    int redtime;
    int greentime;
    int is_red;//是否是红灯
    int cur_redtime;//当前红灯时间
    int cur_greentime;//当前绿灯时间
}t[numnode+1];
```

```
struct road
{
    int st;//起点和终点
    int en;
    double acceleration;//最大加速度
    double length;//道路长度
    double w;//道路宽度
    double antir;//当前阻抗系数
    double vm;//道路允许最大速度
}r[numroad+1];
struct car
{
    double x;
    double y;
    double vx;//速度
    double vy;
    double acc;//减速度
    double hopedis;
    int state;//1 行驶 2 红灯减速 3 等红灯 0 达到目的地
    int cur;//当前在?
    int aimlist[30];//车辆道路目标序列
    int cur_road;//当前所在道路编号
};
vector<car> c;
double distance2(double x1,double y1,double x2,double y2)
{
    double res=0;
    res=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
```

```
    return res;
}
void standard(double& x)
{
    if(x>=1||x<=0)
    {
        x=1/(1+exp(-x));
        x=x*0.9;
    }
}
int nearnode(int id)//寻找车辆附近最近节点
{
    int i,res=-1;
    double dist=inf;
    for(i=0;i<numnode;i++)
    {
        double d;
        d=(t[i].x-c[id].x)*(t[i].x-c[id].x)+(t[i].y-c[id].y)*(t[i].y-
c[id].y);
        if(d<dist)
        {
            dist=d;
            res=i;
        }
    }
    return res;
}
int nearcar(int id)//寻找最近车辆
```

```
{
    int i,num_car;
    num_car=c.size();
    int r1=g[c[id].aimlist[c[id].cur]][c[id].aimlist[c[id].cur+1]];
    int res=id;
    double mindis=inf;
    for(i=0;i<num_car;i++)
    if(i!=id&&c[i].state!=0)
    {
        int r2=g[c[i].aimlist[c[i].cur]][c[i].aimlist[c[i].cur+1]];
        if(r1==r2)
        {
            double dx=c[i].x-c[id].x;
            double dy=c[i].y-c[id].y;
            if(sqrt(dx*dx+dy*dy)>length_car)
            {
                if(dx*c[id].vx>0&&dy*c[id].vy>0)
                {
                    double dist=sqrt(dx*dx+dy*dy);
                    if(dist<mindis)
                    {
                        mindis=dist;
                        res=i;
                    }
                }
            }
        }
    }
```

```
    if(res==id)
        return -1;
    else
        return res;
}
double shortdist[numnode+1];
int pan[numnode+1];
void shortestpath(car &aimcar,int start,int endd) //求解阻抗最短路
{
    int i,j;
    for(i=1;i<=numnode;i++)
        shortdist[i]=inf;
    shortdist[start]=0;
    memset(pan,0,sizeof(pan));
    for(i=1;i<=numnode;i++)
    {
        double d=inf;
        int x;
        for(j=1;j<=numnode;j++)
         {
             if(pan[j]==0&&shortdist[j]<d)
             {
               d=shortdist[j];
               x=j;
             }
         }
        pan[x]=1;
        for(j=1;j<=numnode;j++)
```

```
        {
            if(pan[j]==0)
            {
                if(antig[x][j]>0)
                if(shortdist[x]+antig[x][j]+t[j].redtime/10<shortdist[j])
                    {

shortdist[j]=shortdist[x]+antig[x][j]+t[j].redtime/10;
                        fa[j]=x;
                    }
            }
        }
    }
    int con=0;
    stack<int> tr;
    int next=fa[endd];
    tr.push(endd);
    while(next!=start)
    {
        tr.push(next);
        next=fa[next];
        con++;
    }
    tr.push(next);
    for(i=0;i<=con;i++)
    {
        aimcar.aimlist[i]=tr.top();
        tr.pop();
```

```
    }
    aimcar.aimlist[i]=tr.top();
    aimcar.aimlist[i+1]=-1;
    aimcar.hopedis=shortdist[endd];
    aimcar.cur_road=g[aimcar.aimlist[0]][aimcar.aimlist[1]];
    double dx,dy,len,vmax;
     dx=t[aimcar.aimlist[1]].x-t[aimcar.aimlist[0]].x;
     dy=t[aimcar.aimlist[1]].y-t[aimcar.aimlist[0]].y;
     int pl=rand()%6+2;
     vmax=r[aimcar.cur_road].vm*(double)pl/10;
     len=r[aimcar.cur_road].length;
    aimcar.vx=vmax*(dx/len);
    aimcar.vy=vmax*(dy/len);
}
int main()
{
    srand((unsigned)time(NULL));
    int cnt=0;
    int i,j;
    //读取图论信息
    freopen("c1.txt","r",stdin);
    freopen("out.txt","w",stdout);
    int k=1;
    int id1,id2,we;
    double x1,y1,x2,y2;
     for(i=0;i<numroad;i++)
    {
        scanf("%d(%lf,%lf)-%d(%lf,%lf)%dm",&id1,&x1,&y1,&id2,&x2,&y2,&we);
```

```
        t[id1].x=x1;
        t[id1].y=y1;
        t[id2].x=x2;
        t[id2].y=y2;
        r[k].st=id1;
        r[k].en=id2;
        r[k].w=we;
        if(we==50)
           {
            major_dir[id1][k]=1;
            major_dir[id2][k]=1;
            r[k].vm=40/(3.6*25);
           }
        if(we==30)
            r[k].vm=30/(3.6*25);
        if(we==10)
            r[k].vm=20/(3.6*25);
        r[k].antir=0;
        r[k].length=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        antig[id1][id2]=r[k].length;//阻抗初始化为距离
        g[id1][id2]=k;
        antig[id2][id1]=r[k].length;
        g[id2][id1]=k;
        k++;
    }
   for(i=1;i<numnode;i++)
    t[i].is_cross=1;
        t[9].had=1;
```

```
    t[9].redtime=10;
    t[9].greentime=10;
   t[31].had=1;
    t[31].redtime=10;
    t[31].greentime=10;
 for(i=0;i<firstputcar;i++)//系统开始投放的车辆
    {
      car newcar;
      int s1=rand()%6;
      int s2=rand()%6;
      newcar.x=t[putstart[s1]].x;
      newcar.y=t[putstart[s1]].y;
      newcar.cur=0;
      newcar.state=1;
      shortestpath(newcar,putstart[s1],putend[s2]);
      c.push_back(newcar);
    }
while(cnt<seconds)
{
    for(i=1;i<=numnode;i++) //节点红绿灯处理变化
    if(t[i].had)
    {
        if(t[i].is_red)
        {
            if(t[i].cur_redtime>=t[i].redtime)
              {
                  t[i].cur_redtime=0;
                  t[i].is_red=0;
```

```
            }
        else
            t[i].cur_redtime++;
    }
    else
    {
        if(t[i].cur_greentime>=t[i].greentime)
        {
            t[i].cur_greentime=0;
            t[i].is_red=1;
        }
        else
            t[i].cur_greentime++;
    }
}
int num_car=c.size();
for(i=0;i<num_car;i++)//先投放的车辆先处理
if(c[i].state!=0)//车辆未销毁
{
    if(i==53)
        i=53;
    int fron,last_node,next_node,croad;
    double v,dx,dy,cur_len,new_v,l;
    fron=nearcar(i);
    last_node=c[i].aimlist[c[i].cur];
    next_node=c[i].aimlist[c[i].cur+1];
    croad=g[last_node][next_node];
    v=sqrt(c[i].vx*c[i].vx+c[i].vy*c[i].vy);//当前车速
```

```
            if(v==0)
                v=0;
            dx=t[next_node].x-t[last_node].x;
            dy=t[next_node].y-t[last_node].y;
            cur_len=r[croad].length;
            if(fron<0)//如果前方没有车辆判断红灯
                {

l=distance2(c[i].x,c[i].y,t[next_node].x,t[next_node].y);
                    if(l>seedis)
                    {
                        c[i].state=1;
                        if(v>r[croad].vm)
                        {
                            c[i].vx*=r[croad].vm/v;
                            c[i].vy*=r[croad].vm/v;
                            c[i].x+=c[i].vx;
                            c[i].y+=c[i].vy;
                        }
                        else
                        {
                            if(abs(v)<0.0001)
                                v=beginspeed;
                            new_v=v+(1-r[croad].antir)*(r[croad].vm-v);
                            c[i].vx=new_v*dx/cur_len;
                            c[i].vy=new_v*dy/cur_len;
                            c[i].x+=c[i].vx;
                            c[i].y+=c[i].vy;
```

```
                    }
                }
                else if(t[next_node].had&&(major_dir[next_node][croad]-
t[next_node].is_red==0))//红灯
                {
                    if(c[i].state==2)
                    {
                      double nvy,nvx;
                      nvx=c[i].vx-c[i].acc*dx/cur_len;
                      nvy=c[i].vy-c[i].acc*dy/cur_len;
                      if(nvx*c[i].vx<0||nvy*c[i].vy<0)
                        {
                            c[i].state=3;
                            c[i].x=t[next_node].x;
                            c[i].y=t[next_node].y;
                            c[i].vx=0;
                            c[i].vy=0;
                        }
                    }
                    else if(c[i].state==1)
                        {
                            c[i].acc=v*v/(2*l);
                            c[i].state=2;
                            c[i].vx-=c[i].acc*dx/cur_len;
                            c[i].vy-=c[i].acc*dy/cur_len;
                        }
                    if(c[i].state!=3)
                    {
```

```
            c[i].x+=c[i].vx;
            c[i].y+=c[i].vy;
            if((c[i].x-t[next_node].x)*c[i].vx>0)
            {
                c[i].x=t[next_node].x;
                c[i].y=t[next_node].y;
                c[i].state=3;
                c[i].vx=0;
                c[i].vy=0;
            }
        }
    }
    else if(t[next_node].had==1)//绿灯
    {
        double fixdis=fixeddis;
        if(c[i].state==3)//原来处于等红灯状态
        {
            if(l>=fixdis)
            {
                c[i].vx=beginspeed*dx/cur_len;
                c[i].vy=beginspeed*dy/cur_len;
                c[i].x+=c[i].vx;
                c[i].y+=c[i].vy;
                c[i].state=1;
            }
            else
            {
                double dd=fixdis-l;
```

```
            c[i].cur+=1;
            int new_node=c[i].aimlist[c[i].cur+1];
            if(new_node==-1)
                {
                    c[i].state=0;
                    break;
                }
            else
            {
                double dxn,dyn,lnew;
                dxn=t[new_node].x-t[next_node].x;
                dyn=t[new_node].y-t[next_node].y;
                lnew=r[g[new_node][next_node]].length;
                c[i].x=dd*dxn/lnew+t[next_node].x;
                c[i].y=dd*dyn/lnew+t[next_node].y;
                c[i].vx=beginspeed*dxn/lnew;
                c[i].vy=beginspeed*dyn/lnew;
                c[i].state=1;
            }
        }
    }
    else//行驶而来
    {
        new_v=v+(1-r[croad].antir)*(r[croad].vm-v);
        if(new_v<beginspeed)
            new_v=beginspeed;
        c[i].vx*=new_v/v;
        c[i].vy*=new_v/v;
```

```
                    c[i].x+=c[i].vx;
                    c[i].y+=c[i].vy;
                    if((c[i].x-t[next_node].x)*c[i].vx>0)//转弯
                    {
                        double d1,d2,dd;

d1=distance2(c[i].x,c[i].y,t[last_node].x,t[last_node].y);
                        d2=r[croad].length;
                        dd=d1-d2;
                        c[i].cur+=1;
                        int new_node=c[i].aimlist[c[i].cur+1];
                        if(new_node==-1)
                            {
                                c[i].state=0;
                                break;
                            }
                        else
                        {
                            double dxn,dyn,lnew;
                            dxn=t[new_node].x-t[next_node].x;
                            dyn=t[new_node].y-t[next_node].y;
                            lnew=r[g[new_node][next_node]].length;
                            c[i].x=dd*dxn/lnew+t[next_node].x;
                            c[i].y=dd*dyn/lnew+t[next_node].y;
                            c[i].vx=new_v*dxn/lnew;
                            c[i].vy=new_v*dyn/lnew;
                        }
                    }
```

```
                    c[i].state=1;
                }
            }
            else if(t[next_node].is_cross==1)//无信号灯但是是交通路
口
            {
                double bck;
                if(major_dir[next_node][croad]==1)
                    bck=t[next_node].n1;
                else
                    bck=t[next_node].n2;
                new_v=v+(1-r[croad].antir-bck)*(r[croad].vm-v);
                if(new_v<beginspeed)
                    new_v=beginspeed;
                c[i].vx*=new_v/v;
                c[i].vy*=new_v/v;
                c[i].x+=c[i].vx;
                c[i].y+=c[i].vy;
                if((c[i].x-t[next_node].x)*c[i].vx>0)//转弯
                {
                double d1,d2,dd;

d1=distance2(c[i].x,c[i].y,t[last_node].x,t[last_node].y);
                d2=r[croad].length;
                dd=d1-d2;
                c[i].cur+=1;
                int new_node=c[i].aimlist[c[i].cur+1];
                if(new_node==-1)
```

```
                {
                    c[i].state=0;
                    break;
                }
            else
                {
                    double dxn,dyn,lnew;
                    dxn=t[new_node].x-t[next_node].x;
                    dyn=t[new_node].y-t[next_node].y;
                    lnew=r[g[new_node][next_node]].length;
                    c[i].x=dd*dxn/lnew+t[next_node].x;
                    c[i].y=dd*dyn/lnew+t[next_node].y;
                    c[i].vx=new_v*dxn/lnew;
                    c[i].vy=new_v*dyn/lnew;
                }
            }
        }
        else //无信号灯不是交通路口
            {
                c[i].x+=c[i].vx;
                c[i].y+=c[i].vy;
                if((c[i].x-t[next_node].x)*c[i].vx>0)//转弯
                {
                    double d1,d2,dd;
d1=distance2(c[i].x,c[i].y,t[last_node].x,t[last_node].y);
                    d2=r[croad].length;
                    dd=d1-d2;
```

```
                    c[i].cur+=1;
                    int new_node=c[i].aimlist[c[i].cur+1];
                    if(new_node==-1)
                        {
                            c[i].state=0;
                            break;
                        }
                    else
                    {
                        double dxn,dyn,lnew;
                        dxn=t[new_node].x-t[next_node].x;
                        dyn=t[new_node].y-t[next_node].y;
                        lnew=r[g[new_node][next_node]].length;
                        c[i].x=dd*dxn/lnew+t[next_node].x;
                        c[i].y=dd*dyn/lnew+t[next_node].y;
                        c[i].vx=v*dxn/lnew;
                        c[i].vy=v*dyn/lnew;
                    }
                }
            }
    }
else//如果前方有车辆判断速度变化
    {
        if(c[i].state!=3)
        {
            int stf=c[fron].state;
            double dist;
            dist=distance2(c[fron].x,c[fron].y,c[i].x,c[i].y);
```

```
                if(stf==3)
                {
                    c[i].x=c[fron].x-length_car*dx/cur_len;
                    c[i].y=c[fron].y-length_car*dy/cur_len;
                    c[i].state=3;
                    c[i].vx=0;
                    c[i].vy=0;
                }
                else
                {
                    new_v=0.8*(dist-length_car);
                    c[i].vx=new_v*dx/cur_len;
                    c[i].vy=new_v*dy/cur_len;
                    c[i].x+=c[i].vx;
                    c[i].y+=c[i].vy;
                    c[i].state=1;
                }

            }
        }
}
for(i=1;i<=numnode;i++){t[i].n1=0;t[i].n2=0;}
for(i=1;i<=numroad;i++) r[i].antir=0;
for(i=0;i<num_car;i++) //计算节点阻滞系数和道路阻抗系数
if(c[i].state!=0)
{
    int next_node=c[i].aimlist[c[i].cur+1];
    int _road=g[c[i].aimlist[c[i].cur]][c[i].aimlist[c[i].cur+1]];
```

```
            double
dis=distance2(c[i].x,c[i].y,t[next_node].x,t[next_node].y);
            if(dis<tjr)
               {
                   if(major_dir[next_node][_road])
                    t[next_node].n1+=1;
                   else
                    t[next_node].n2+=1;
               }
            r[_road].antir+=1;
            roadtime[_road][cnt]+=1;
            nodetime[next_node][cnt]+=1;
        }
        for(i=1;i<=numnode;i++)
            {
                t[i].n1=t[i].n1*fconst;
                t[i].n2=t[i].n2*fconst;
                standard(t[i].n1);
                standard(t[i].n2);
            }
        for(i=1;i<=numroad;i++)
            {

r[i].antir=kconst*(r[i].antir/(r[i].length*r[i].w))/(r[i].vm);
                standard(r[i].antir);
            }
        for(i=0;i<putcar;i++)//投放车辆
        {
```

```
   car newcar;
  int s1=rand()%6;
  int s2=rand()%6;
  newcar.x=t[putstart[s1]].x;
  newcar.y=t[putstart[s1]].y;
  newcar.cur=0;
  newcar.state=1;
  shortestpath(newcar,putstart[s1],putend[s2]);
  c.push_back(newcar);
}
for(i=0;i<putcar;i++)//投放车辆
{
   car newcar;
  int s1=rand()%6;
  int s2=rand()%6;
  newcar.x=t[putstart[s1]].x;
  newcar.y=t[putstart[s1]].y;
  newcar.cur=0;
  newcar.state=1;
  shortestpath(newcar,putstart[s1],putend[s2]);
  c.push_back(newcar);
}
cnt++;
int j;
if(cnt%10==0)
{
    for(i=1;i<=numnode;i++)
        for(j=i+1;j<=numnode;j++)
```

```
                {
                    antig[i][j]=r[g[i][j]].length*(r[g[i][j]].antir*2+1);
                    antig[i][j]=antig[j][i];
                }
        }
       // printf("%.4f %.4f\n",c[0].x,c[0].y);
    }
    for(i=1;i<=numroad;i++)
    {
        for(j=1;j<seconds;j++)
        {
            printf("%d ",roadtime[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```