共享单车区域分布和调度问题研究

摘要

共享单车是指企业在校园、地铁站点、公交站点、居民区、商业区、公共服务区等提供自行车单车共享服务，是一种分时租赁模式。共享单车是一种新型共享经济。共享单车已经越来越多地引起人们的注意，由于其符合低碳出行理念，政府对这一新鲜事物也处于善意的观察期。很多共享单车公司的单车都有 GPS 定位，能够实现动态化地监测车辆数据、骑行分布数据，进而对单车做出全天候供需预测，为车辆投放、调度和运维提供指引。 在很多时候，用户想用车，附近却无车可用或有车却用不了。

针对问题一，根据一千辆自行车的骑行数据，可以分析出在不同时段内，不同区域中各地段自行车的时空情况，以及区域之间的距离关系。利用 java 语言，通过正则表达式，提取出骑行数据，再使用动态规划算法得到各区域的某时间段的车辆数值。

针对问题二，我们对附件一中共享单车的骑行数据进行挖掘。通过自行车出发时间和到达时间的间隔，通过计算每条数据取平均值的方法求得各取区域间的距离矩阵。我们得到距离矩阵后以调度总距离最短为目标函数，以及供需和实际条件为约束条件，建立优化模型。我们采用了模拟退火的方法来求解模型，最后得到一个可行的调度方案。

针对问题三,我们先确定可以从材料中获得的指标,发现有些指标并不是能够用一天的数据来衡量的。所以我们选取了区域单车存量 区域单车可用量和区域单车供需矛盾程度等三个方面六个指标建立 AHP 模糊综合评价模型，再根据评价得分进行排名。我们将 100 辆自行车优先分配给排名靠后的地区。

针对问题四，我们对附件三中的数据进行可视化并进行简单拟合。发现了共享单车投放量与打车减少量之间的关系。我们研究其机理发现其符合新产品扩散对互补产品的影响关系。所以我们建立离散型增长阻滞模型来拟合打车减少量和随时间投入共享单车数量的关系。此外，为了进一步研究这个问题 ，我们搜集资料另外引了区域共享单车使用频率 共享单车营收和出租车司机营收等变量，建立 BP 神经网络模型，进一步探究共享单车对打车市场的冲击情况。

关键字：共享单车调配 正则表达式 动态规划 AHP 模糊综合评价 BP 神经网络 阻滞增长模型 目标规划 VRP 模型 模拟退火

一、 问题重述

1.1 问题背景

共享单车是指企业在校园、地铁站点、公交站点、居民区、商业区、公共服务区等提供自行车单车共享服务，是一种分时租赁模式。共享单车是一种新型共享经济。共享单车已经越来越多地引起人们的注意，由于其符合低碳出行理念，政府对这一新鲜事物也处于善意的观察期。很多共享单车公司的单车都有 GPS 定位，能够实现动态化地监测车辆数据、骑行分布数据，进而对单车做出全天候供需预测，为车辆投放、调度和运维提供指引。 在很多时候，用户想用车，附近却无车可用或有车却用不了。而企业呢？一方面，用户有用车需求，却无法及时响应；另一方面，在其他区域，有大量的单车处于暂时闲置状态。这个需求其实是供与求在时间上、空间上相匹配的问题。匹配度越高，问题解决的就越好，匹配成本越低，效益就越好。

1.2 问题描述

（1）根据附件 1 中共享单车的骑行数据，估计共享单车的时空分布情况。如从某地点 A 出发，到达不同地点的分布情况。可分时间段讨论。

（2）假如根据调查，得到人们的骑行需求估计数据，见附件 2。

根据问题 1 的估计结果，建立数学模型解决如何优化共享单车的调度问题。

（3）根据附件 1 的骑行数据和附件 2 的需求数据，判断各区域所需共享单车的满足程度，给出你的度量指标。若增加 100 辆单车，如何进行投放更优。

（4）附件 3 是某地区投入不同数量共享单车后打车人次的数据。据此分析研究共享单车的投入对该地区打车市场的影响。同时请你收集实际数据进行量化研究。

二、 问题分析

针对第一问：

我们根据一千辆自行车的骑行数据，分析出在不同时段内，不同区域中各地段自行车的时空情况，以及区域之间的距离关系。首先，通过 java 正则表达式，提取出自行车的骑行数据；其次，通过每辆车的停车时间再使用动态规划算法(dynamic programming)，利用某辆车的停车时间（即某辆车上一次停车时间（到达时间）减去下一次出发时间）得到各区域的某时间段的车辆数值，并通过计算自行车骑行时间（当次到达时间减去当次出发时间），通过平均值求取区域间的距离矩阵。

针对第二问：

我们可以对附件一中共享单车的骑行数据进行挖掘。通过自行车出发时间和到达时间的间隔，通过计算每条数据取平均值的方法求得各取区域间的距离矩阵。我们得到距离矩阵后以调度总距离最短为目标函数，以及供需和实际条件为约束条件，建立优化模型。我们采用了模拟退火的方法来求解模型，最后得到一个可行的调度方案。

针对第三问：

我们可以先确定可以从材料中获得的指标,发现有些指标并不是能够用一天的数据来衡量的。所以我们选取了区域单车存量 区域单车可用量和区域单车供需矛盾程度等三个方面六个指标建立 AHP 模糊综合评价模型,再根据评价得分进行排名。我们将 100 辆自行车优先分配给排名靠后的地区。

针对第四问：

我们对附件三中的数据进行可视化并进行简单拟合。发现了共享单车投放量与打车减少量之间的关系。我们研究其机理发现其符合新产品扩散对互补产品的影响关系。所以我们建立离散型增长阻滞模型来拟合打车减少量和随时间投入共享单车数量的关系。此外，为了进一步研究这个问题，我们搜集资料另外引了区域共享单车使用频率 共享单车营收和出租车司机营收等变量，建立 BP 神经网络模型，进一步探究共享单车对打车市场的冲击情况。

三、 模型假设

1. 假设附件和获取资料的数据准确

2. 骑行数据中,基本不受由于人们不是直达某地而造成计算区域距离的影响。

3. 假定附件一中给出的自行车活动数据完整且对应 24 小时

4. 假设骑行数据在误差允许范围内可以表现各个区域间的距离

四、 符号说明

符号	说明解释
Time	车辆的间隔时间
Location	表示区域 1-10 的车辆数
GoOff	车辆的出发时间
ArrivalTime	车辆的到达时间
Loca	距离矩阵
P	专家评分
$\sigma_h(k)$	偏导数
X	打车人次减少量
p	P=r/N
r, N	增长率 单车影响饱和值
k	递增投放单车时间段
E /e	误差
W	权值
f () /s()	函数
X	距离
k	移动量
w	需求量
s	状态

表 1 符号说明

五、 模型建立与求解

5.1 单车时空分布模型

5.1.1 数据的提取

对于问题一，首先利用记事本文档中的替换，将附件一中的“自行车序号”替换成“z 自行车序号”，这样做的目的是做一个标记，便于更好

车辆判断	^z.*$
车辆序号以及出发区域	^\d{1,}$
出发时间，到达时间及 目的地	^\d{1,4}(.)?\d{0,2}$

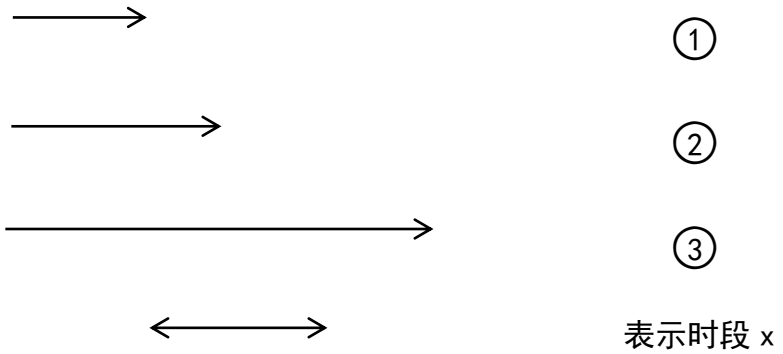
的提取出自行车的数据信息，至此我们可以列出如下三个正则表达式：

通过正则表达式以及 IO（input&output）处理，我们可以得到所有数据在数据中，我们得到了所有数据中的最小时间为 360.21，最大时间为 1423.1，所以我们可以将数据分为时差为 100，从 300 开始的 12 个时段。

5.1.2 动态规划分析

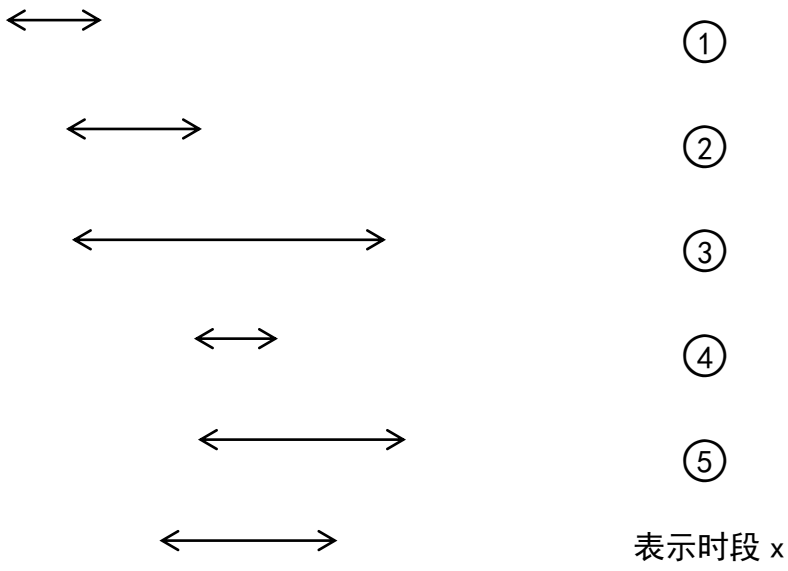
将所有信息保存到 car 类里，分析自行车出发时间，我们可以通过动态规划算法，先分析车辆停车可能的情况：

对于开始的情况：



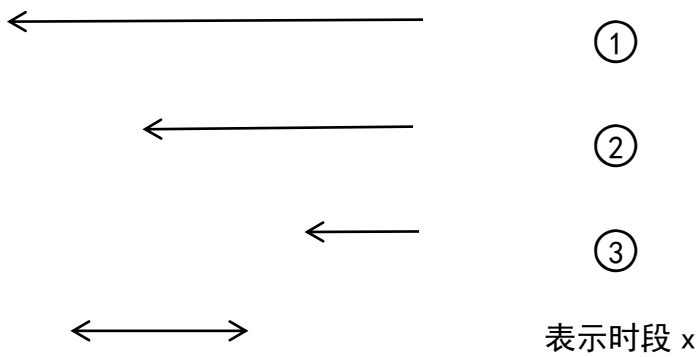
具体解释一下这张图的含义单项箭头表示该车的停车时段，箭头处表示该车第一个出发时间，①表示第一个出发时间在时段 x 之前，②表示第一个出发时间在时段 x 中，③表示第一个出发时间在时段 x 后；

对于中间的情况：



图中的双向箭头①②③④⑤表示该车的停车情况，其中左箭头表示这辆车上一次的到达时间，右箭头表示这辆车的出发时间，；

对于最后的情况：



单项箭头表示该车的停车时段，箭头处表示该车最后一个到达时间。

所以有如下表达式：

$$\text{time}_j = GoOff_i(\text{或}300 + i*100) - ArrivalTim\,e_{i-1}(\text{或}200 + \text{i}*100)$$

对于所有边界在区域外面的时段，都用括号里的数值

有了以上分析，我们可以计算出区域中停车的数量，表达式为

$$location_i = \sum_{j=0} time_i / 100 \quad i = 1、2、3...10$$

这里我们得到的是区域 1-10 的在该时段内的停车数量，由于车并不会在整个时段内都停在该区域，所以这里用小数表示该车停在该区域在整个时域占比，作为该区域停车的数值。

最后用 matlab 绘图，如图所示为 1-12 个时间段内，区域 1-10 的停车情况。

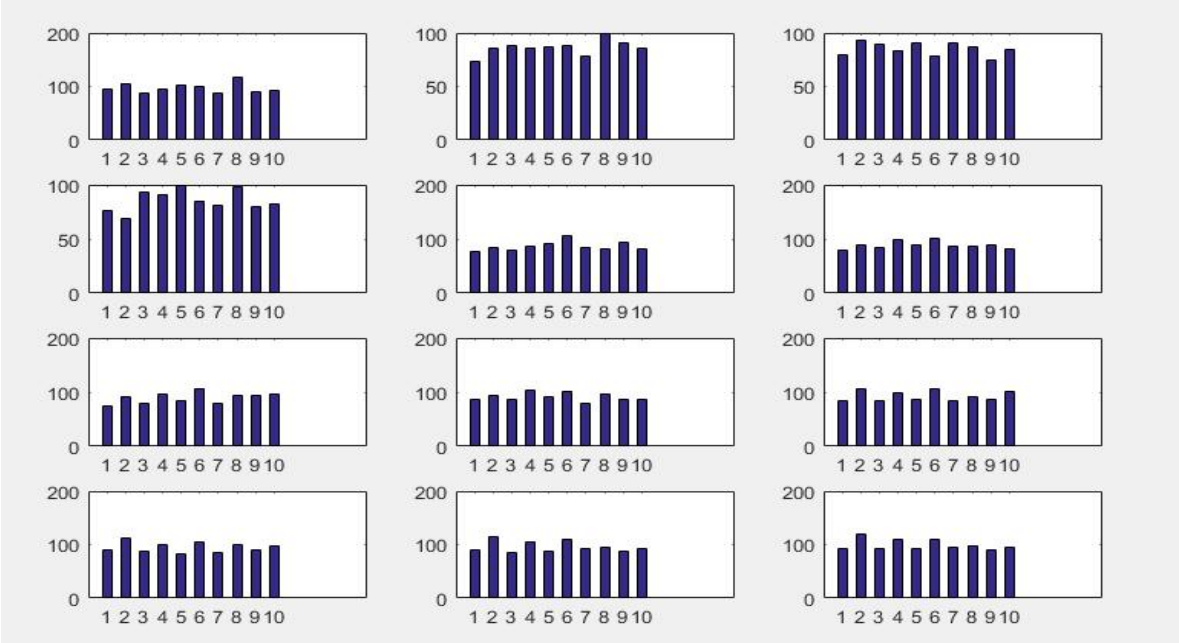


图 1 单车时序分布图

5.2 区域单车调度模型

5.2.1 计算距离矩阵

我们可以根据附件一里的骑行记录估算出每个区域之间的距离，我们得到距离矩阵才能进一步建立车辆调度优化模型。

我们可以通过附件一里的骑行数据和 5.1.2 模型的结果，通过时间，我们可以对区域之间的距离进行估算

距离矩阵的表达式如下：

$$loca_{ij} = \sum_{k=1}^{n} \frac{(ArrivalTime - GoOff_k)}{n} (i, \mathrm{j} = 1、2...10)$$

由此可得关于时间的距离矩阵。（其中横轴为目的地，纵轴为出发地）

0	10.37961	14.31574	10.81695	8.153616	9.036612	10.905	11.37639	12.32275	8.236616
10.44045	0	4.89172	6.886625	13.70023	12.53375	10.11392	8.580256	3.50743	10.92664
14.44842	4.818122	0	13.37393	8.942425	5.382111	14.04131	3.995541	13.96117	7.099734
10.57297	6.801937	13.42963	0	4.082678	6.644184	12.17658	4.258555	6.898228	11.04575
8.057095	13.60421	8.883125	4.027595	0	10.32809	12.2	11.11814	14.78705	13.56333
9.043856	12.55195	5.447053	6.760467	10.35899	0	6.112134	4.295611	5.20645	13.6115
10.86964	10.14425	13.79962	12.1366	12.27815	6.071479	0	4.460117	13.9727	13.57951
11.388	8.57094	4.037588	4.25088	11.18888	4.331299	4.426699	0	12.70536	3.690453
12.39754	3.514056	14.05666	6.882447	14.76473	5.278876	13.88732	12.56999	0	4.453657
8.221534	10.91788	7.101232	11.0323	13.72306	13.5269	13.56153	3.668815	4.453834	0

表 2 各区域距离矩阵

5.2.2 单车调度模型

我们根据附件二中的数据得到每个地区的净需求量，每个地区的净需求量是需要我们通过调配来平衡的。

调配车辆应该避开高峰期，所以我们选择 2,3,4 这三个时间段自行车停考量最多的时间进行调配。取此时区间内车辆数值的均值 z（1..10）。

我们的目标函数是：

$$\min \sum_{i=1}^{10} \sum_{j=1}^{10} x_{ij} \times s_{ij}$$

即各区域间调配里程最小 s 取 0 或 1

约束条件是：

$$w_i = w_i + \sum_{j}^{10} k_{ij} - \sum_{i}^{10} k_{ji}$$

最终净需求量应该是移入和移出之后值。

$\sum_{i=1}^{10} w_i > 0$ 每个区域的净需求量保证足够。

$s_{ij} = 0$ when $k_{ij} = 0$ 将不移动的路线置为 0.

$k_{ij} < z_i$ 调动数量应该不大于调动时含有的车辆

我们可以采用模拟退火的方法来求解这种 NP 难题。模拟退火算法的基本原理是来自于固体加热至一定的温度后由固体结构瓦解变为液体结构，在对其降温过程甲乙控制，是的分子在变回固定结构时，能重新排列成我们所预期的稳定状态。模拟退火算法已在理论上被证明是一种概率 1 收敛与全局的全局最优解算法。我们下边的算法描述，该问题的一个解 i 极其目标函数 f（i）分别于固体的一个微观状态 i 机器能量 E（i）等价.

模拟退火算法的基本步骤如下：

第 1 步 随机得到一个初始可行解 x_0，设定初始温度 t_0，令当前解 $x_i = x_0$，当前迭代步数 k = 0，当前温度 $t_k = t_0$

第 2 步 若在该温度达到内循环停止条件，则转第 3 步；否则，从邻域 $N(x_i)$，中随机选择一个邻解 x_j，并计算 $\Delta E_{ij} = E(x_j) - E(x_i)$. 若 $x_i = x_j$；否则，若 $\exp(-\Delta E_{ij}/t) > rand$ (0, 1)，则 $x_i = x_j$，重复第 2 步。

第 3 步 k=k+1, $t_{k+1} = y(t_k)$，若满足种植条件，则转到第 4 步；否则，转到第 2 步。

第 4 步 输出计算结果，算法停止。

内循环为第 2 步，他表示在同一温度下进行随机搜索。外循环主要包括第 3 步的温度下降变化，迭代步数的增加和停止准则。

我们可以得到 2 到 4 时段调度情况结果：

1→4　13 辆

1→3　7 辆

6→5 2　5 辆

10→5　10 辆

10→9　20 辆

7→8　4 辆

此时即满足需求，又不会影响车辆分布变化造成的短缺。

在此需要说明的是我们选取的目标函数是要求调配总距离最短，其实也可以考虑其他因素例如调配车的饱和程度，采用调配距离和调配数量总乘积最小的目标函数等。

另外我们没有考虑建立调配中心，我们采用每个点都参与调度，在一定程度上是不够的最优的。

5.3 区域单车满足情况模型

5.3.1 指标选取

我们可以建立综合评价模型，从多个角度利用 5.1 和 5.2 材料里的数据综合评测一个区域实际的车辆满足程度。

一级指标： 单车拥有程度 单车可用量程度 单车供求矛盾程度

选取一下二级指标：

a:单车时段平均分布数

数据可以从 5.1 中可以得到，它反映了一个区域分布车辆的平均整体水平。

b:高峰期单车拥有率

我们选取停留总量较少几个时段假定为高峰期，用车高峰期是供求矛盾集中的时期，此时区域中车辆的保有率能够反应现实的需求。

c:单车需求与占有比率

从 5.2 中可以得到，反应供需关系。

d:单车停留平均时长

从附件一中可以得到，反应可供选择单车使用情况。

e:每日需求出入净值

可从附件二中得到

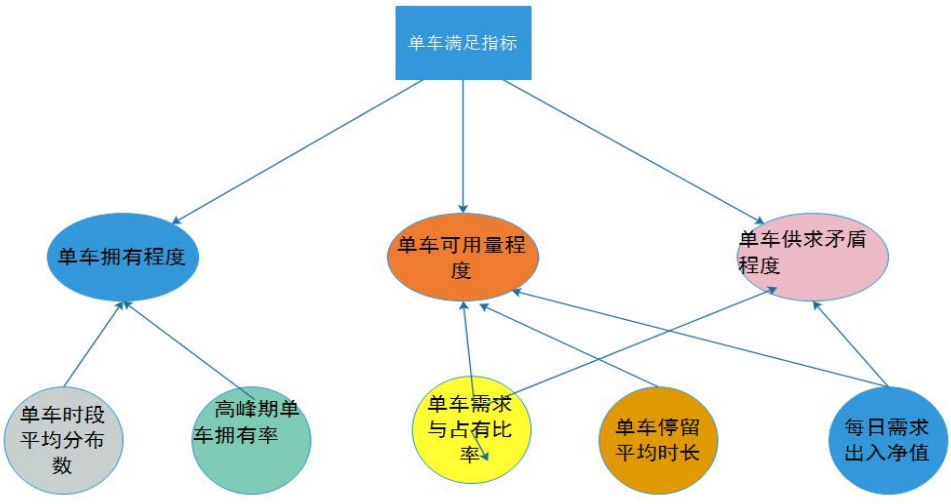


图 2 指标选取

5.3.2 基于 AHP 的模糊综合评价模型

AHP 模型

1：我们将 6 个指标分成三大类，其中有些指标是公用的但他们的权值不一样。
2：依据专家打分构造评价矩阵
①对每列的判断矩阵元素进行归一化处理

$$\bar{P}_{ij}=\frac{P_{ij}}{\sum_{k=1}^{n}P_{kj}}$$

②计算出归一化后的矩阵行与构量的平均值，该平均值就是各指标的权重值

$$\overline{w_i}=\sum_{j=1}^{n}\overline{P_{ij}}$$

③然后在向量 $\overline{W_i}=\left[\overline{W_1},\overline{W_2},....\right]^T$ 归一化以后计算判断矩阵的最大特征根：

$$\lambda_{max}=\sum_{i=1}^{n}\frac{(PW)_i}{nW_i}$$

④一致性检查
根据测试结果调整判断矩阵，直到 CR ＜ 0.1。

含有模糊指标的综合评测模型

因为我们只有一天的数据所以我们对一项指标好坏的评价不能准确地给出，这里存在偶然性，但是我们可以通过对这些指标就行模糊打分，可以大致反应这些指标的所处的水平和区间，在一定程度上减少误差。
1：对一些不能具体量化的指标就行模糊打分
2：确定评价集，在这里我们采用从一个 A 到 E 评价机制
3：确定隶属函数，由于我们的模糊指标正相关类型，所以我们选择较大的柯西函数：

$$f(x)=\begin{cases}\left[1+\alpha(x-\beta)^{-2}\right]^{-1}, 1\le x\le 3\\ a\ln x+b \quad\quad ,3\le x\le 5\end{cases}$$

当评价为“A”时，隶属度为 1，当评价为“C”时，隶属度为 0.8
f（3）=0.8；
f（1）=0.01；
解得:
α=1.1086； a=0.3915；

β=0.3915； b=0.3699；

3：因为数据间存在差异，我们将只能模糊评判的指标如高峰期车辆占有情况就

行数据统计打分.（因为我们只有一天的数据，完整的数值不能确切量化统计）
4:用 AHP 模型中得到的权重对 10 各区域进行单因素评价. 根据评分数据和 f（x）构建单因素评价矩阵 $R_i = \{r_{ij}\}$.

单因素评价： $B_i = W_i \cdot R_i$.

5：构建多因素综合评价矩阵 B

$$R = \begin{bmatrix} W_1 \cdot R_1 \\ W_2 \cdot R_2 \\ W_3 \cdot R_3 \end{bmatrix} \qquad B = W \cdot R$$

最后计算各个区域的综合得分

$$E = B \cdot H^T$$

单车满足等度指标排名情况如下：

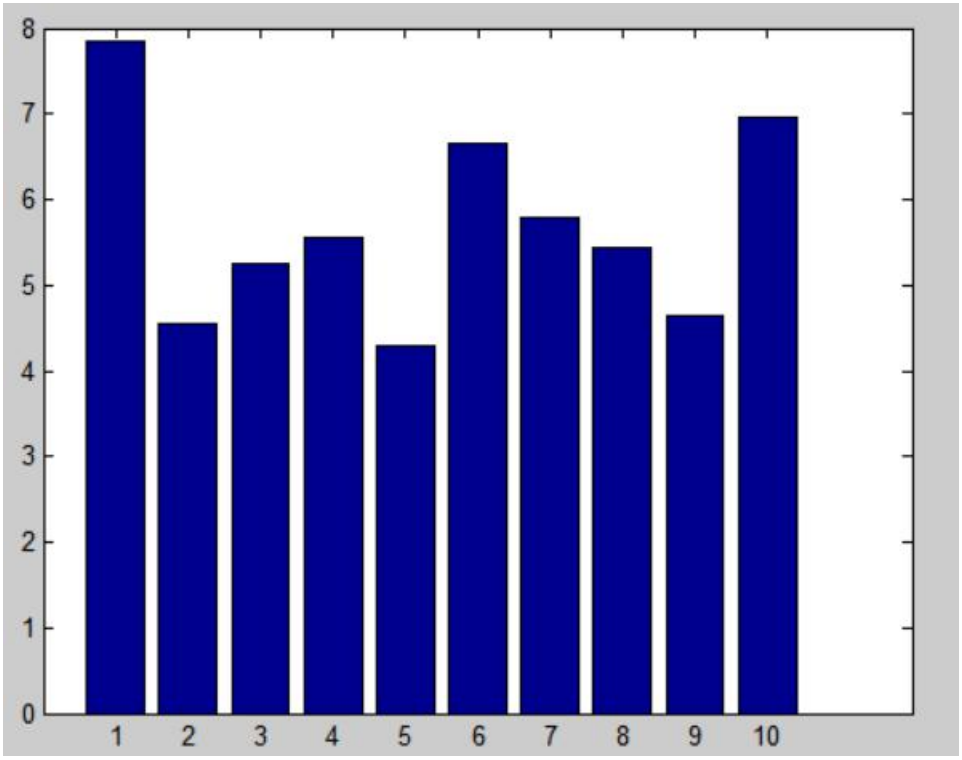


图 3 各区域满足程度指标

如果 100 辆车我们应该优先分配给排名靠后的地区，使各个区域的满足程度尽量保持一致。

5.4 共享单车投放对打车市场的影响

5.4.1 数据的预分析和拟合

我们根据附件三中给出的数据进行绘图，得到了变化曲线的基本图像。然后我们对数据进行拟合，使其更加易于理解和分析。

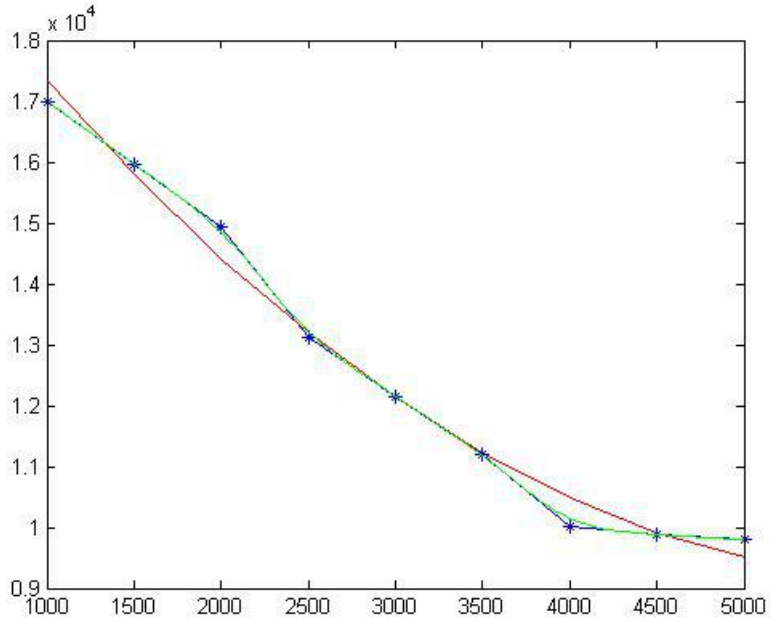


图 4 数据拟合

我们不难发现随着共享单车的投入数量变多，当地打车的数量不断减少，并且斜率也是在不断变小的。

5.4.2 离散型阻滞增长模型

共享单车相比于打车具有方便使用，环保，价格低廉等优点。所以当共享单车刚投入市场时可以借鉴种群入侵模型，此时打车相当于原始种群，共享单车相当于入侵种群。由于入侵种群在与原始种群的竞争中具有压倒性的优势。所以刚开始时打车对共享单车的增长影响可以忽略不计。

我们这里取打车人次的减少量为 x 构建离散阻滞增长模型。

首先我们假设用前差公式计算的增长率为常数 r，既假设：

$$\frac{\mathrm{x}_{\mathrm{k+1}}-x_k}{x_k}=r,k=0,1,2,...$$

其中 k 为每一个时段并且每一个时段都有投放自行车数量的增长。也就是每个 k 阶段实质上的变化是由投放共享单车来引起的。

然后我们建立一节线性常微分方程模型：

$$\mathrm{x}_{\mathrm{k+1}}=(1+r)x_k$$

解为等比数列：

$$\mathrm{x}_k=x_0(1+k)^k$$

如果 r>0,则受公享单车投放的影响，打车减少量将无限增长。

但是我们知道在某些情况下共享单车是不能够完全取代出租车的,比如长距离运输，老年人病人的出行等方面。我们可以称这些人群为“刚需”人群，另外二者作为互补产品有公共的目标人群，这部分人群既可以选择共享单车，也可以选择打车出行。所以我们需要对共享单车对打车的影响进行限制，我们称这种制约作用为“阻滞作用”。

假设由于受到阻滞作用,用前叉公式计算的增长率随着种群数量的增加而现行递减，即：

$$\frac{\mathrm{x}_{k+1}-x_k}{x_k}=r\left(1-\frac{x_k}{N}\right)$$

其中 N 为共享单车饱和时对打车减少量的影响。

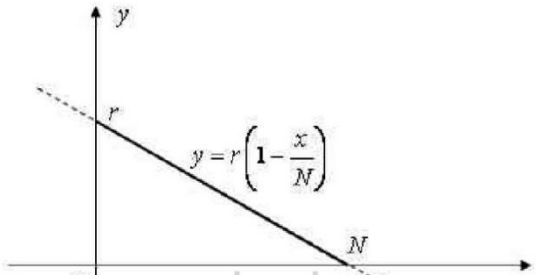
离散阻滞增长模型就是一阶线性差分方程：

$$\Delta x_k=rx_k\left(1-\frac{x_k}{N}\right)$$

即 $x_{k+1}=x_k+rx_k\left(1-\frac{x_k}{N}\right)$

分别记 x 和 y 是同一时段的打车减少量和前差公式计算的增长率,则在 x-y 直角坐标系内直线方程:

y =r(1-x/N)



其中纵截距为 r，横截距为 N

参数 r 为固有增长率，既然 r 是直线方程的纵截距，所以在理论上是当打车减少量 x=0 时的增长率；实际上 r 是共享单车投放初期打车减少量的增长率。

参数 N 是最大容量，实际上是共享单车随时间变化投放量对打车减少量影响的饱和值。

而$(1-x_k/N)$体现了打车硬性需求对打车减少量的阻滞作用，所以被称为阻滞作用因子。随着打车减少量的不断增加，阻滞作用因子越来越小，逐渐趋向于 0.

由此我们可以得到离散阻滞增长模型的等价形式：

$$x_{k+1}=x_k+px_k(N-x_k)$$
$$x_{k+1}=x_k+rx_k-px_k^2$$

离散阻滞增长模型很难写出解析解，不过我们可以根据递推公式绘制其拟合效果图，来验证模型的准确性。

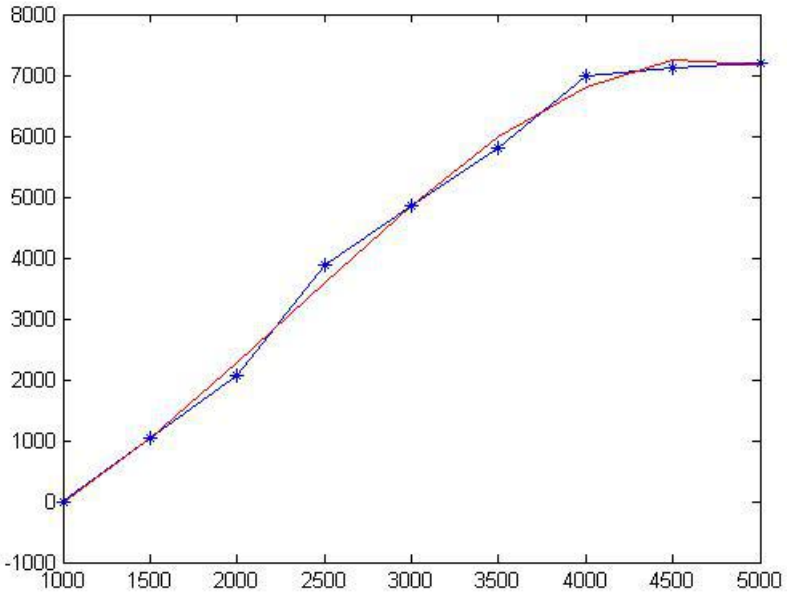


图 5　离散阻滞增长模型拟合结果图

我们发现可以明显反映给出的数据的变化趋势。

5.4.3 BP 神经网络模型

我们在前边的基础上，自己手机数据进一步进行量化研究。我们收集到共享单车投放数量，以及共享单车企业盈利额还有单车使用频率对打车次数和出租车司机人均收入的数据。然后我们建立 BP 神经网络模型进一步探究二者影响的关系之间的关系。

BP 神经网络模型

BP 神经网络的基本思想是利用输出后的误差来估计输出层的直接前导层的误差，再用这个误差估计更前一层的误差，如此一层一层的反传下去，就获得了所有其他各层的误差估计。 “BP 神经网络模型拓扑结构包括输入层、隐层和输出层”。

基本 BP 算法包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。误差反传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，使误差沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数(权值和阈值)，训练即告停止。此时经过训练的神经网络即能对类似样本的输入信息，自行处理输出误差最小的经过非线形转换的信息。

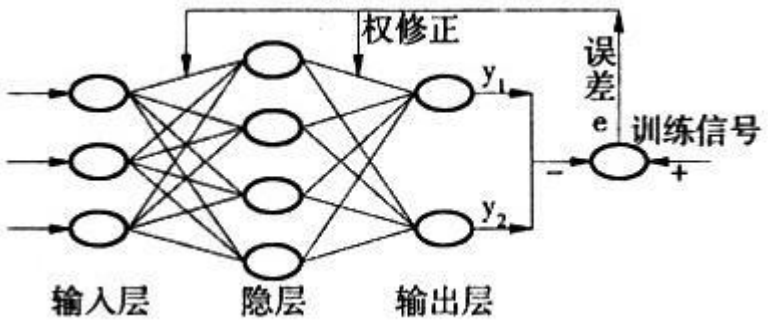


图 6 BP 神经网络示意图

BP 神经网络具体的算法步骤：

1. 网络初始化

给各连接权值分别赋一个区间（-1，1）内的随机数，设定误差函数 e，给定计算精度值　和最大学习次数 M。

2. 随机选取第 k 个输入样本及对应期望输出

$$d_o(k)=\left(d_1(k),d_2(k),\cdots,d_q(k)\right)$$

$$x(k)=\left(x_1(k),x_2(k),\cdots,x_n(k)\right)$$

3. 计算隐含层各神经元的输入和输出

4. 利用网络期望输出和实际输出，计算误差函数对输出层的各神经元的偏导数 $\sigma_0(\mathrm{k})$ a

5. 利用隐含层到输出层的连接权值、输出层的 $\sigma_0(\mathrm{k})$ 和隐含层的输出计算误差函数对隐含层各神经元的偏导数 $\sigma_h(\mathrm{k})$

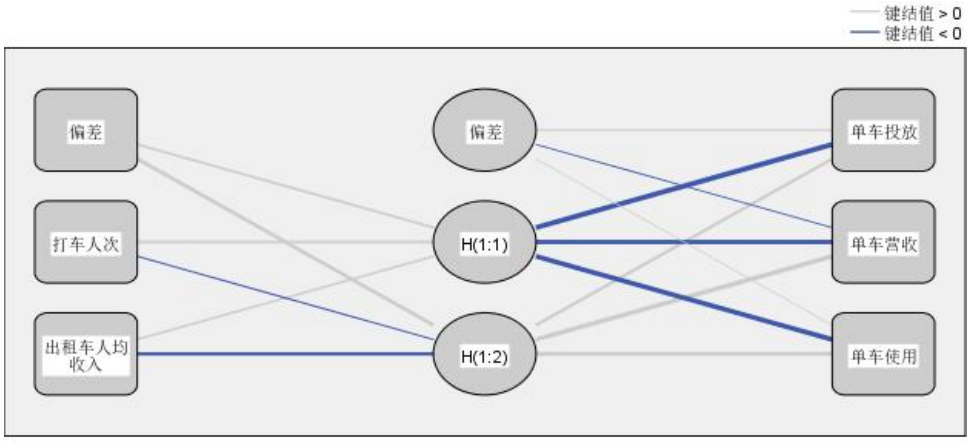
6. 利用输出层各神经元的 $\sigma_h(\mathrm{k})$ 和隐含层各神经元的输出来修正连接权值 $w_{ho}(\mathrm{k})$

7. 利用隐含层各神经元的 $\sigma_h(\mathrm{k})$ 和输入层各神经元的输入修正连接权。

8. 计算全局误差 $E=\frac{1}{2m}\sum_{k=1}^{m}\sum_{o=1}^{q}(d_0(k)-y_0(k))^2$

9. 判断网络误差是否满足要求。当误差达到预设精度或学习次数大于设定的最大次数，则结束算法。否则，选取下一个学习样本及对应的期望输出，返回到第三步，进入下一轮学习。

我们经过训练得到以下结果：



隐藏层激活函数： 双曲正切

输出层激活函数： 恒等

模型汇总

	平方和错误		.209
	平均整体相对错误		.023
		单车投放	.048
训练	尺度因变量的相对错误	单车营收	.004
		单车使用	.017
	中止使用的规则		错误未减少的 1 连续步骤[a]
	培训时间		0:00:00.00
	平方和错误		.021
	平均整体相对错误		.019
测试		单车投放	.011
	尺度因变量的相对错误	单车营收	.020
		单车使用	.030

a. 基于检验样本的错误计算。

图 7 神经网络基本信息

六、 模型评价

6.1 模型优点

1. 在模型的选取和建立过程中，使用了多种丰富的算法，使问题的解决和阐述更加合理也更加科学。

2. 对于以上问题的分析，模型可以做到很好的数据匹配，即通过我们的模型可以更好地容纳更多的数据，对于数据处理过程以及处理结果都有很好的通用性。

3. 在问题的处理及数据的处理过程中，做到了足够的客观处理，数据全部由科学合理的算法解决，保证了模型建立的严格与准确。

6.2 模型不足

1. 对于我们的数据，缺乏实际应用检验。

2. 模型侧重对给出的数据进行分析，缺乏可拓展性。

6.3 模型推广

1. 可以应用到实际的共享资源运维中去。

2. 可以获取更多的详细数据，进行更加细致的分析。

七、 参考文献

[1]周彩英. 基于 AHP 和模糊综合评判的档案信息利用服务评价[J]. 档案学通讯,2011,03:88-91.

[2]武兴睿. Java 中的正则表达式与模式匹配研究[J]. 科技传播,2011,15:180+186.

[3]向速林,刘占孟,尤本胜. 水资源调配的动态规划研究[J]. 新疆环境保护,2005,01:18-20.

[4]黄丽. BP 神经网络算法改进及应用研究[D].重庆师范大学,2008.

[5]张建国. 城市公共自行车车辆调配问题研究[D].西南交通大学,2013.

[6]李续扬. 公共自行车系统车辆调配优化研究[D].兰州交通大学,2016.

[7]王越. 太原市公共自行车车辆调配问题研究[D].山西大学,2016.

八、 附录

附录 1. 问题一 附件 1 骑行数据. txt（文件有所改动）

z 自行车序号： 1, 出发区域: 8

出发时间：468. 28, 到达时间：471. 67, 到达区域:10

出发时间：569. 13, 到达时间：580. 61, 到达区域: 2

出发时间：655. 88, 到达时间：669. 49, 到达区域: 6

出发时间：724. 07, 到达时间：728. 71, 到达区域: 8

出发时间：778. 06, 到达时间：782. 69, 到达区域: 6

出发时间：877. 31, 到达时间：891. 67, 到达区域:10

（以下数据略）

附录 2. 问题一 java 代码

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
class car{
     int carnumber;//车的编号
     int carlocation;//车的初始位置
     int time;//车使用的次数
     float GoOff[];//出发时间
     float ArrivalTime[];//到达时间
     int destination[];//到达后的地点
    public int getCarnumber() {
         return carnumber;
    }
    public void setCarnumber(int carnumber) {
         this.carnumber = carnumber;
    }
    public int getCarlocation() {
         return carlocation;
    }
    public void setCarlocation(int carlocation) {
         this.carlocation = carlocation;
    }
    public int getTime() {
         return time;
    }
    public void setTime(int time) {
         this.time = time;
    }
    public float[] getGoOff() {
         return GoOff;
    }
    public void setGoOff(float[] goOff) {
         GoOff = goOff;
    }
    public float[] getArrivalTime() {
         return ArrivalTime;
    }
    public void setArrivalTime(float[] arrivalTime) {
         ArrivalTime = arrivalTime;
    }
    public int[] getDestination() {
         return destination;
```

```
    }
    public void setDestination(int[] destination) {
         this.destination = destination;
    }
    public void set(car a){
         this.setArrivalTime(a.getArrivalTime());
         this.setCarlocation(a.getCarlocation());
         this.setDestination(a.getDestination());
         this.setCarnumber(a.getCarnumber());
         this.setTime(a.getTime());
         this.setGoOff(a.getGoOff());
    };

     public car(int c,int cl,float go[],float at[],int d[],int j){
         setCarlocation(cl);
         setArrivalTime(at);
         setCarnumber(c);
         setGoOff(go);
         setDestination(d);
         setTime(j);
     }
     public car(){

     }
}
 class zhengze {
         public boolean zhengze(String str){
            /*str 为要验证的字符串*/
             // 验证规则，此规则为是否为一个新的数据
             String regEx = "z.*";
             // 编译正则表达式
             Pattern pattern = Pattern.compile(regEx);
             Matcher matcher = pattern.matcher(str);
             // 字符串是否与正则表达式相匹配
             boolean rs = matcher.matches();
             return rs;
         }
    }
```

```
public class Mathorcap{
    /**
     * 功能：Java 读取 txt 文件的内容
     * 步骤：1：先获得文件句柄
     * 2：获得文件句柄当做是输入一个字节码流，需要对这个输入流进行读取
     * 3：读取到输入流后，需要读取生成字节流
     * 4：一行一行的输出。readline()。
     * 备注：需要考虑的是异常情况
     * @param filePath
     */
    public static void main(String args[]){
     car[] c=new car[1000];
     String filePath = "E:\\附件 1 骑行数据.txt";
     zhengze z=new zhengze();
        try {
               int j=0;//i 表示车
               int i=0;//j 表示数据
               int carn=0;
               int carl=0;
               float[] f1=new float[50];
                 float[] f2=new float[50];
                 int[] aaaa=new int[50];
                 String encoding="GBK";
                 File file=new File(filePath);
                 if(file.isFile() && file.exists()){ //判断文件是否存在
                     InputStreamReader read = new InputStreamReader(
                     new FileInputStream(file),encoding);//考虑到编码格式
                     BufferedReader bufferedReader = new BufferedReader(read);
                     String lineTxt = null;

                     while((lineTxt = bufferedReader.readLine()) != null){
                        /*IO 核心算法*/



                        if(z.zhengze(lineTxt)){
                        if(j!=0){
                             car temp=new car(carn, carl, f1, f2, aaaa, j);
                             c[i]=temp;
```

```
                                j=0;
                                i++;
                                f1=new float[50];
                                f2=new float[50];
                                aaaa=new int[50];

                            }
                            Pattern p = Pattern.compile("\\d{1,}");//这个 1 是指连续数
字的最少个数
                            Matcher m = p.matcher(lineTxt);
    //                          c[i].carnumber =
Integer.valueOf(m.group()).intValue();
    //
c[i].carlocation=Integer.valueOf(m.group()).intValue();
                        /**
                         * 赋值
                         */
                            {   if(m.find()) {
    //                              System.out.println(m.group());
                                    String a=m.group();
                                    carn=Integer.valueOf(a).intValue();

                                }
                                if(m.find()) {
    //                              System.out.println(m.group());
                                    String a=m.group();
                                    carl=Integer.valueOf(a).intValue();

    //                              System.out.println(m.group());
                                }

                            }
                            }else{
                                Pattern p =
Pattern.compile("\\d{1,4}(.)?\\d{0,2}");//这个 1 是指连续数字的最少个数
                                Matcher m = p.matcher(lineTxt);
                                if(m.find()){
    //                              System.out.println(m.group());
                                    float aa = Float.parseFloat(m.group());
```

```
                                f1[j]=aa;
                                }
                            if(m.find()){
//                              System.out.println(m.group());
                                float aa = Float.parseFloat(m.group());
                                f2[j]=aa;
                                }
                            if(m.find()){
//                              System.out.println(m.group());
                                int aa=Integer.valueOf(m.group()).intValue();
                                aaaa[j]=aa;
                                }
                            j++;
                            }

                    }
                    read.close();
                    car temp=new car(carn, carl, f1, f2, aaaa, j-1);
                   c[i]=temp;//最后一次赋值
            }else{
                System.out.println("找不到指定的文件");
            }
            } catch (Exception e) {
                System.out.println("读取文件内容出错");
                e.printStackTrace();
            }
            float k=c[0].GoOff[0];
            float kl=c[0].ArrivalTime[0];
     //     System.out.println(c[999].GoOff[4]);
           System.out.println(c[41].time);
            /**
             *统计下应该设置的时间段
             */
            for(int i=0;i<1000;i++){
                for(int j=0;j<c[i].getTime();j++){
                    if(c[i].GoOff[j]<k) k=c[i].GoOff[j];
                    if(c[i].ArrivalTime[j]>kl)kl=c[i].ArrivalTime[j];
                }
            }
```

```
System.out.println("最短时间"+k+"最长时间"+kl);
/**
 * 最后得到 location 是一个二维的数组
 */
float[][] location=new float[12][11];
for(int i=0;i<12;i++){//i 表示时间段
for(int j=0;j<1000;j++){//j 表示车
 int p=c[j].getCarlocation();
 /*开始的情况*/

if(c[j].GoOff[0]<(400.0f+i*100.0f)&&c[j].GoOff[0]>(300.0f+i*100.0f)){//排除第一种
      location[i][p]+=(c[j].GoOff[0]-(300.0f+i*100.0f))/100.0f;//第二种情况
 }else if(c[j].GoOff[0]>(400.0f+i*100.0f)){
      location[i][p]+=1.0f;//第三种情况
 }else{}
 /*中间的情况*/
 for(int l=1;l<c[j].time;l++){//l 表示 time
      int l_1=l-1;
      p=c[j].destination[l_1];

if(c[j].ArrivalTime[l_1]<(400.0f+i*100.0f)&&c[j].GoOff[l]>(300.0f+i*100.0f)){//排除第一个&第六个可能性
      if(c[j].ArrivalTime[l_1]<(300.0f+i*100.0f)){//第 123 种情况
           if(c[j].GoOff[l]<(400.0f+i*100.0f)){//第 124 种情况

location[i][p]+=(c[j].GoOff[l]-300.0f-i*100.0f)/100.0f;//综合为第 2 种情况
           }else{//第 3,5,6 种情况
                location[i][p]+=1.0f;//综合，为第 3 种情况
           }
      }else{//456 种情况
           if(c[j].GoOff[l]<(400.0f+i*100.0f)){//124

location[i][p]+=(c[j].GoOff[l]-c[j].ArrivalTime[l_1])/100.0f;//4
           }else{//356

location[i][p]+=(400.0f+i*100.0f-c[j].ArrivalTime[l_1])/100.0f;//5
      }
 }
```

```
                }
            }
             /*最后的情况*/
             int time_1=c[j].time-1;
             if(c[j].ArrivalTime[time_1]<(400.0f+i*100.0f)){//排除 3
                  if(c[j].ArrivalTime[time_1]<300.0f+i*100.0f){//第一种情况
                       location[i][p]+=1;
                  }else{

location[i][p]+=(400.0f+i*100.0f-c[j].ArrivalTime[time_1])/100.0f;//第二种情况
                  }
             }
             }
            }
            /*for(int i=0;i<12;i++){//输出第一道题各区域的某个时间段的车辆数值
             for(int j=1;j<11;j++){
                  System.out.print(location[i][j]+"\t");
             }
             System.out.println("\t");
            }*/
            /**
             * 统计距离矩阵
             */
            float[][] loca=new float[11][11];
            int[][] nloca=new int[11][11];
            int gol;
            int arrl;
            for(int i=0;i<=999;i++){
             gol=c[i].carlocation;
             for(int j=0;j<c[j].time;j++){
                  if(j!=0) gol=c[i].destination[j-1];
                  arrl=c[i].destination[j];
                  loca[gol][arrl]+=c[i].ArrivalTime[j]-c[i].GoOff[j];
                  nloca[gol][arrl]+=1;

             }
            }
           /*输出距离矩阵*/
            for(int i=1;i<11;i++){//距离矩阵横轴为目的地，纵轴为出发地
```

```
                for(int j=1;j<11;j++){
                    if(nloca[i][j]==0){
                        System.out.print("0\t");
                    }else{
                        System.out.print(loca[i][j]/nloca[i][j]+"\t");
                    }
                }
                System.out.println("\t");
            }
        //        for(int i=1;i<11;i++){
        //            for(int j=1;j<11;j++){}
        //        }
        }
    }
```

附录 3.问题一 matlab 代码

```
x1=[93.8535 106.1641 87.9754 95.0272 103.0347 99.7145 87.7834 117.4334 89.0695 91.3788
];
x2=[73.63951 85.7974 88.78649 86.10962 87.014595 88.21428 78.356514 99.31748
91.19677 85.23519
];
x3=[79.17672 92.81789 88.959465 83.94537 91.08857 78.86899 90.52871
86.82518 74.75249 84.19372
];
x4=[76.09477 68.647514 93.42313 91.20604 99.34302 85.1752 80.61445
97.865005 79.57778 82.841515
];
x5=[77.27392 83.21029 79.34378 85.77731 91.6527 105.5617 85.116005 82.24297
92.86375 81.12458
];
x6=[78.34728 88.5112 83.21131 97.59367 89.60221 101.52932 87.03251 86.97879
88.49565 82.60593
];
x7=[74.87261 92.23538 78.786 96.56861 84.986725 106.70829 79.397385 94.03879
93.911316 97.32038
];
x8=[86.35099 94.57372 85.92449 103.85062 90.8609 102.03881 79.121086 97.50979
87.45282 86.407684
];
x9=[84.07969 105.37769 84.51103 98.99307 86.74359 105.7789 83.845695
92.4707 86.32072 101.31551
];
x10=[89.41821 111.1599 88.06069 99.78691 83.11459 104.975006 84.97247
```

```
99.293175    89.828415    96.7565
];
x11=[90.568596   115.0345     85.5042  105.9347     87.9957  109.308205   92.00151     94.09302
87.705605    93.089195
];
x12=[92.8329     120.3775     92.4158  109.71591    91.744995    109.6428     94.7222  98.72309
90.70521     95.577095
];
subplot(4,3,1),bar(x1,0.5);
subplot(4,3,2),bar(x2,0.5);
subplot(4,3,3),bar(x3,0.5);
subplot(4,3,4),bar(x4,0.5);
subplot(4,3,5),bar(x5,0.5);
subplot(4,3,6),bar(x6,0.5);
subplot(4,3,7),bar(x7,0.5);
subplot(4,3,8),bar(x8,0.5);
subplot(4,3,9),bar(x9,0.5);
subplot(4,3,10),bar(x10,0.5);
subplot(4,3,11),bar(x11,0.5);
subplot(4,3,12),bar(x12,0.5);
```

附录 4 模型二代码：

附录 5 模型三代码：

```
clear;    clc;
 E=input('输入计算精度 e:')
  Max=input('输入最大迭代次数 Max:')
  F=input('输入优先关系矩阵 F:');    %计算模糊一致矩阵    N=size(F);    r=sum(F');
  for i=1:N(1)    for j=1:N(2)    R(i,j)=(r(i)-r(j))/(2*N(1))+0.5;    end    end
  E=R./R';     % 计算初始向量 ----------
  % W=sum(R')./sum(sum(R)); % 和行归一法
  %
  for i=1:N(1)    S(i)=R(i,1);    for j=2:N(2)    S(i)=S(i)*R(i,j);
  end    end     S=S^(1/N(1));     W = S./sum(S);%方根法
  % a=input('参数 a=?');     %W=sum(R')/(N(1)*a)-1/(2*a)+1/N(1); %排序法
  % 利用幂法计算排序向量----V(:,1)=W'/max(abs(W));
  %归一化    for i=1:Max    V(:,i+1)=E*V(:,i);
  V(:,i+1)=V(:,i+1)/max(abs(V(:,i+1)));     if max(abs(V(:,i+1)-V(:,i)))  k=i;
  A=V(:,i+1)./sum(V(:,i+1));    break    Else    End    End
```

附录 6 模型四代码：

自然状态下，用 plot 画的是折线，而不是平滑曲线。

有两种方法可以画平滑曲线，第一种是拟合的方法，第二种是用 spcrv，其实原理应该都一样就是插值。下面是源程序，大家可以根据需要自行选择，更改拟合的参数。

```
clc,clear;
a = 1:1:6;  %横坐标
b = [8.0 9.0 10.0 15.0 35.0 40.0]; %纵坐标
plot(a, b, 'b');  %自然状态的画图效果
hold on;
%第一种，画平滑曲线的方法
c = polyfit(a, b, 2);  %进行拟合，c 为 2 次拟合后的系数
d = polyval(c, a, 1);  %拟合后，每一个横坐标对应的值即为 d
plot(a, d, 'r');       %拟合后的曲线

plot(a, b, '*');       %将每个点 用*画出来
hold on;
%第二种，画平滑曲线的方法
values = spcrv([[a(1) a a(end)];[b(1) b b(end)]],3);
plot(values(1,:),values(2,:), 'g');
```

```
N=1;   r=[.1;1.9;2.3;2.5;2.7;2.7];
x=[.1;.1;.1;.1;.1;.10001];
for k=1:100
    x(:,k+1)=x(:,k)+r.*x(:,k).*(1-x(:,k)./N);
end
s{1}=……      s{6}='r>2.57,混沌(r=2.7,x_1=0.10001)';
for k=1:6
    subplot(3,2,k), plot(0:100,x(k,:),'k.')
    axis([0,100,0,1.6]), xlabel(s{k})
end
```


SPSS 软件操作略