机场的出租车问题

摘要

现代社会中，人们在日常出行方式当中必然少不了出租车这个选项。如何让乘客和司机利益均得到保障是研究的问题。该文主要通过对宏观和微观两种情况建立数学模型，来得到有关出租车司机该如何进行决策的方法和机场管理部门应如何设置接车方式和对短途载客并再次返回的出租车给予“优先权”的具体解决方案。

对于问题一，该文主要建立出宏观地图与司机等待模型，利用题目已知条件知道司机能看到的具体等待时间 t_{wait} 和计价方式还有出租车的平均收益 a。用分段函数方式来求出了计价方式 f 的具体表达式。求出两种收益分别的表达式之后，直接比较这两个表达式，即可得到结果。

对于问题二，选取郑州机场的目前的数据，有关于这部分数据均用爬虫取自于网上。利用过去半个小时之内离开飞机场和进入飞机场的车辆数，能够得到司机的平均等待时间 t_{wait} 的具体值，再利用郑州的出租车营业方案，得到计价方式 f 的具体表达式，这时候直接利用第一问的表达式即可完成此题。

对于问题三，

对于问题四，

关键字： 元胞自动机　梯度下降

目录

一、 问题重述

1.1 问题背景

由于出租车的选择以及机场管理者所作出的决策对效率，时间，金钱均有着重大的影响。因此，对于机场管理者来说，对于加快机场外交通运营，增加司机收入，提高乘客舒适度，节约出行时间来说，提高机场出租车运营效率是具有重要意义的。

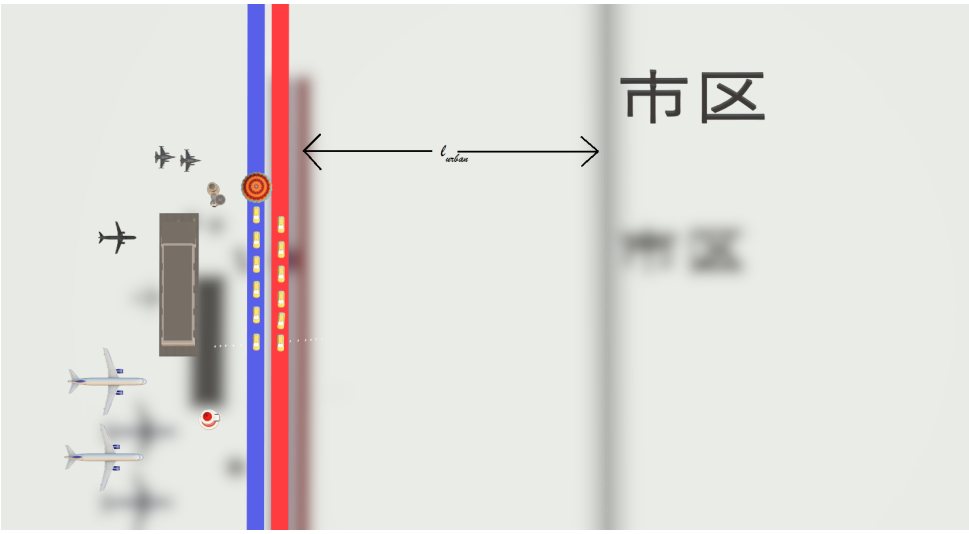

图 1　市区，机场与排队等车模型示意图

1.2 问题的提出

该文章主要是研究不同情况下司机该如何来选择最优的方式和机场管理者该如何调整政策使得宏观效率最高。由于出租车必须到指定的“蓄车池”排队等候，依“先来后到”排队进场载客，这导致司机有一个“排队时间成本”，如果不排队接客，则多出的时间就可以到市区接客。但若直接去市区接客，则从机场到市区中间一段距离就是空置的。

当然，为了解决这个问题，除了司机自身的判断之外，很重要的另外一点就是机场管理者的政策。机场管理者设定不同的上下车点，司机接客的方式，对短途载客再次返回的出租车给予一定的“优先权”，均会影响到具体的效益以及收入情况。

在问题一当中，题设较为简单，只需要建立出租车司机选择决策模型，并给出司机的选择策略即可，所以利用题目所给出的司机已知的条件。

也即大致的出租车司机排队等待时间加服务时间 t_{wait} 和出租车的计价方式 f 还有具体的城市到飞机场的距离 l_{urban}。

第一小问希望利用这些条件，能够得到最终司机分别在哪一种情况下选择哪一种方案的解析解。

在问题二当中，题目直接要求对于一个具体城市的具体司机的具体选择，这需要考察网上的信息搜索能力，在第一问已经得到抽象的解析解的时候，题目要求只需要把具体数值代入表达式当中得到具有现实参考意义的解。

在问题三当中，

在问题四当中，

二、 问题分析

2.1 问题一的分析

问题一较为简单，因此只需要拟定各个不同的定义所代表的含义，目的在于给司机选定到底选择哪一个哪一种方案收益最高。虽然本问并没有给出具体数据，但仍然可以把一些量用字母替代。

考虑到司机是能够观察到具体还有多少乘客和出租车的，所以直接能够假定司机知道还需要等待多久的时间，这是合理的。由于出租车司机的收入司机也是知道的，所以可以假定市区出租车的平均收入。

通过这些条件，选择是否留在机场接客还是直接回市中心是通过比较两者的平均收益得到的。

这时候利用之前得到的出租车收益图函数即可得到最终目标的解析解。

2.2 问题二的分析

问题二需要寻找具体的城市具体分析，由于没有附录提供的任何数据，使用计算机技术在网上进行爬虫，记录并得到了郑州市的有关具体数据。

由于这些数据并不是能够直接得到第一问所设条件的具体值，因此得对这些数据进行处理以得到最终需要的值。

通过这些数据，代入第一问所得到的值，能够得到该机场出租车司机的具体选择方案。最终，通过数据依赖分析来分析模型的合理性和对相关因素的依赖性。

2.3 问题三的分析

2.4 问题四的分析

三、 模型假设

- 认为城市内和城市外行车速度恒定。
- 认为在路上的时候没有堵车情况发生，不考虑出租车关于时间的计费。
- 前两问假设服务时间为定值。
- 假设一辆车只载送一位乘客。
- 假设空载费用损失为油费。
- 假设一个周期内出租车和乘客一一抵消。
- 假设使用 92 号汽油。
- 假设出租车使用汽油总是 $1L$ 走 11.8 公里的距离。

四、 符号系统

符号	意义
fee	出租车的计价收益
t_{wait}	出租车排队等待时间加服务时间
m_{start}	出租车起步价
m_{oil}	每一公里出租车的油费价格
m_{night}	夜价提升价格量
m_{aver}	在市区当中的平均收入
l_{init}	出租车计价起步距离
k	每公里计费价格
l_{urban}	机场到高速路口的平均距离
l_{city}	高速路口到市中心的平均距离
v_{sub}	从机场到市区当中的郊区行车速度
v_{city}	出租车回到市区后平均速度
t_{city}	出租车回到市区后为完成机场订单仍需在城市中行车的时间
t_{wait}	出租车排队等待时间与服务时间总和

注：未列出及重复符号以出现处为准

五、 建模与求解

5.1 出租车的收益模型

对于城市出租车，忽略其型号、车龄等差别，并假设每辆出租车参数相同。

在一个城市中，出租车管理中通常几种计费方式混合使用，不同情况下计费方式不同。在这里为简化模型，选取最常用的计费方式。该计费方式描述如下：

出租车起步价为 m_{start} 元，在 l_{init} 公里后开始按路程距离计价，每多一公里计费 k 元。夜价则在日价的起步价基础上增加 m_{night} 元。

则白天时的出租车计价方式如下：

$$fee(d,\text{day}) = \begin{cases} m_{\text{start}} & ,d < l_{\text{init}} \\ m_{\text{start}} + k \cdot (l - l_{\text{init}}) & ,d \geq l_{\text{init}} \end{cases} \tag{1}$$

夜晚时出租车计价方式如下：

$$fee(d,\text{night}) = fee(d,\text{day}) + m_{\text{night}} \tag{2}$$

但在出租车行驶过程中，如果出现空载情况，司机需要承担一定损失。假设空载损失的主要部分是油费，且油费与行车距离成正比，系数为每公里出租车油费价格 m_{oil}：

$$oilfee(d) = d \cdot m_{\text{oil}} \tag{3}$$

用 matlab 对白天情形下的 $fee - d$ 函数关系与 $oilfee - d$ 函数关系进行可视化，结果如下：

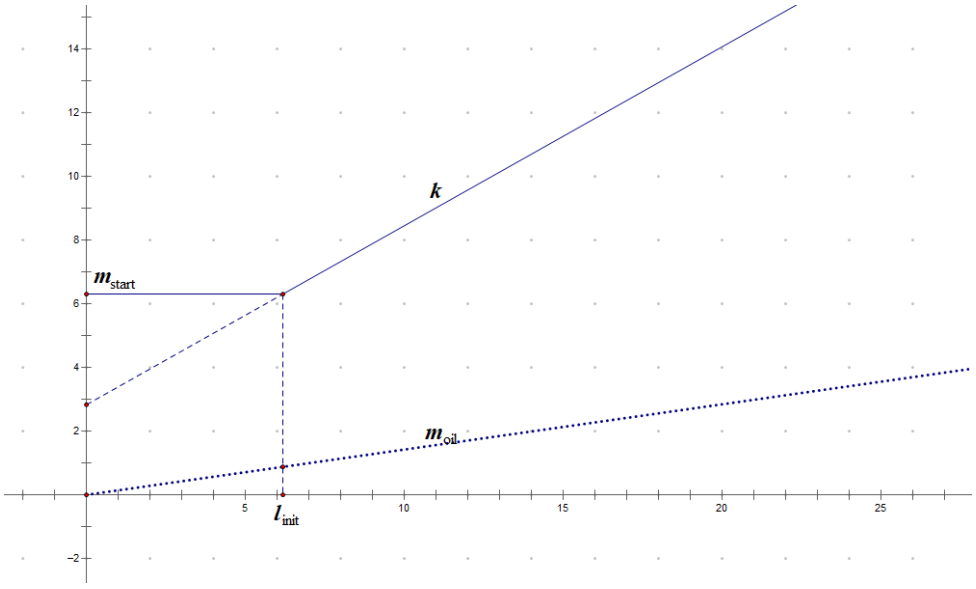

图 **2**　出租车的收益函数图示

在前两问中，出租车总收益将由出租车计价收入 fee 与油费消耗 $oilfee$ 构成, 用符号简记为：

$$sumfee(d) = fee(d) - oilfee(d) \tag{4}$$

5.2 问题一的建模与求解

5.2.1 模型准备

只需要明确相同时间内排队载客后返回市区和空载返回市区这两种情况的平均收益该计算公式，司机便能够就是否排队问题做出决策。假定机场到市区的平均距离为 l_{urban}。从机场到市区行车速度为 v_{out}，市区中行车速度为 v_{city}。

5.2.2 模型的建立

(1) 首先明确要考虑的时间段。从选择排队载客后返回市区的方向对时间段进行刻画，整个时间段划分为三部分：

1. 出租车排队等待时间与服务时间总和 t_{wait}。

2. 出租车从机场返回市区时间 t_{return}，其计算方式为：

$$t_{\text{return}} = \frac{l_{\text{urban}}}{v_{\text{sub}}} \tag{5}$$

3. 出租车回到市区后为完成机场订单仍需在城市中行车的时间 t_{city}。

记总时间为 t_{sum}，其中：

$$t_{\text{sum}} = t_{\text{wait}} + t_{\text{return}} + t_{\text{city}} \tag{6}$$

将该时间段对应到空载返回市区的情形，则有：

1. 出租车从机场返回市区时间 t_{return}。

2. 出租车返回市区后接客时间 $t_{\text{wait}} + t_{\text{city}}$

下面将根据上述上述时间划分对两种选择的平均收益进行计算。

(2) 若司机选择排队载客后返回市区，那么总时间段内平均收益的计算方法如下：

$$aveincome1 = \frac{sumfee(l_{\text{urban}} + v_{\text{city}} \cdot t_{\text{city}})}{t_{\text{sum}}} \tag{7}$$

上式中假定等待时间与服务时间无收入与损失，分子中总收入为机场中接到的一单收入。

(3) 若司机选择空载返回市区，那么总时间段内平均收益的计算方法如下：

$$aveincome2 = \frac{m_{\text{aver}} \cdot (t_{\text{wait}} + t_{\text{city}}) - oilfee(l_{\text{urban}})}{t_{\text{sum}}} \tag{8}$$

上式中 m_{aver} 表示出租车司机在城市内每小时平均收入 (考虑了油费损失)。分子中总收入为市区中行车收入减去从机场返回市区的油费消耗。

我们只需比较这两个的大小：

1. 若 $aveincome1 > aveincome2$, 应选择排队载客后返回市区。

2. 若 $aveincome1 < aveincome2$，应选择空载返回市区。

下面解该不等式：

$$\begin{cases} aveincome1 > aveincome2 \\ aveincome1 = \frac{sumfee(l_{\text{urban}}+v_{\text{city}}\cdot t_{\text{city}})}{t_{\text{sum}}} \\ aveincome2 = \frac{m_{\text{aver}}\cdot(t_{\text{wait}}+t_{\text{city}})-oilfee(l_{\text{urban}})}{t_{\text{sum}}} \\ sumfee(l) = fee(l) - oilfee(l) \\ oilfee(l) = l \cdot m_{\text{oil}} \end{cases} \tag{9}$$

解得：

$$t_{\text{wait}} < \frac{fee(l_{\text{urban}} + v_{\text{city}} \cdot t_{\text{city}}) - m_{\text{oil}} \cdot v_{\text{city}} \cdot t_{\text{city}}}{m_{\text{aver}}} - t_{\text{city}} \tag{10}$$

我们记这个特殊的时间点为判断等待时间 t_0：

$$t_0 = \frac{fee(l_{\text{urban}} + v_{\text{city}} \cdot t_{\text{city}}) - m_{\text{oil}} \cdot v_{\text{city}} \cdot t_{\text{city}}}{m_{\text{aver}}} - t_{\text{city}} \tag{11}$$

等待时间 t_{wait} 比判断等待时间 t_0 小时，选择排队载客后返回市区；当等待时间 t_{wait} 比判断等待时间 t_0 大时，司机空载返回市区更加划算。

那么问题转化为如何计算等待时间与服务时间总和 t_{wait}。由题意可知司机可获取的信息为航班信息与蓄车池中出租车数量。该模型将通过这两个变量计算出等待时间与服务时间总和 t_{wait}。

为建立该计算模型，先定义三个速度：

1. 蓄车池进车速度 v_{in}，表示每分钟进入蓄车池的车辆数。假设其于白天为一定值，于晚上为另一定值。

2. 服务速度 v_{out}，表示每分钟离开蓄车池的车辆数。假设其于单队列单服务口情形下为定值。

3. 乘客队列增长速度 v_{pin}，表示每分钟乘客队列增长人数。假设每时刻 (航班到达时刻间隔为 30 分钟) 到达的航班数量为 n_{fly}，每辆航班上乘客为 300 人，乘客选择乘坐出租车的比例为 α。那么通过下式对乘客队列增长速度进行计算:

$$v_{\text{pin}} = \frac{n_{\text{fly}} \cdot 300 \cdot \alpha}{30} \tag{12}$$

对于 3 个速度大小，会有 13 种情况。下图用图示法进行划分，其中样本点到 3 个关键点的距离用来表示其 3 个速度大小。

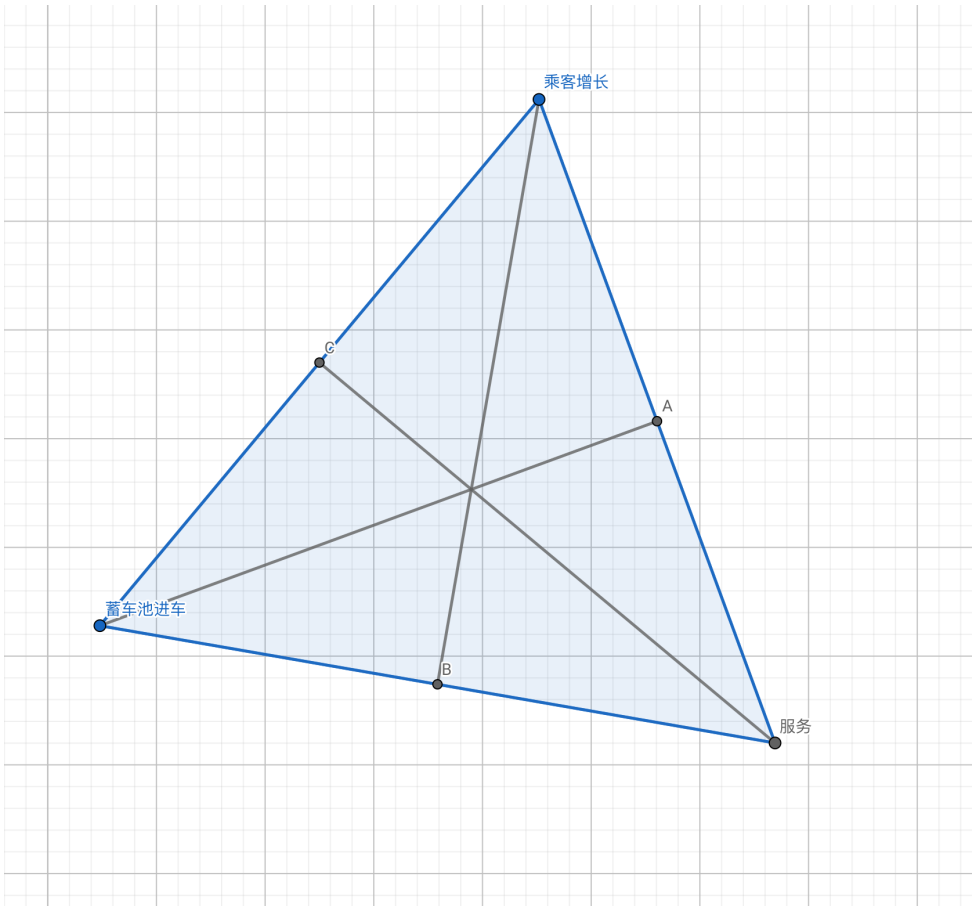

图 **3**　出租车的收益函数图示

13 种情况分别为 6 块区域，6 根分界线，1 个中心点。对各个情况单独讨论，可发现会出现下述两种现象:

1. 人堆积，随时间增长，乘客队列不断增长。

2. 车堆积，随时间增长，蓄车池里车辆数量不断增长。

下列表格详细描述了各个情况:

情形	结果
$v_{\text{pin}} > v_{\text{in}} > v_{\text{out}}$	人堆积，车堆积
$v_{\text{pin}} > v_{\text{out}} > v_{\text{in}}$	人堆积
$v_{\text{pin}} > v_{\text{in}} = v_{\text{out}}$	人堆积
$v_{\text{pin}} = v_{\text{in}} > v_{\text{out}}$	人堆积，车堆积
$v_{\text{pin}} = v_{\text{out}} > v_{\text{in}}$	人堆积
$v_{\text{pin}} = v_{\text{out}} = v_{\text{in}}$	不堆积
$v_{\text{in}} > v_{\text{pin}} > v_{\text{out}}$	人堆积，车堆积
$v_{\text{in}} > v_{\text{out}} > v_{\text{pin}}$	车堆积
$v_{\text{in}} > v_{\text{pin}} = v_{\text{out}}$	车堆积
$v_{\text{in}} = v_{\text{out}} > v_{\text{pin}}$	车堆积
$v_{\text{out}} > v_{\text{pin}} > v_{\text{in}}$	人堆积
$v_{\text{out}} > v_{\text{in}} > v_{\text{pin}}$	车堆积
$v_{\text{out}} > v_{\text{pin}} = v_{\text{in}}$	不堆积

用上图表示结果为:

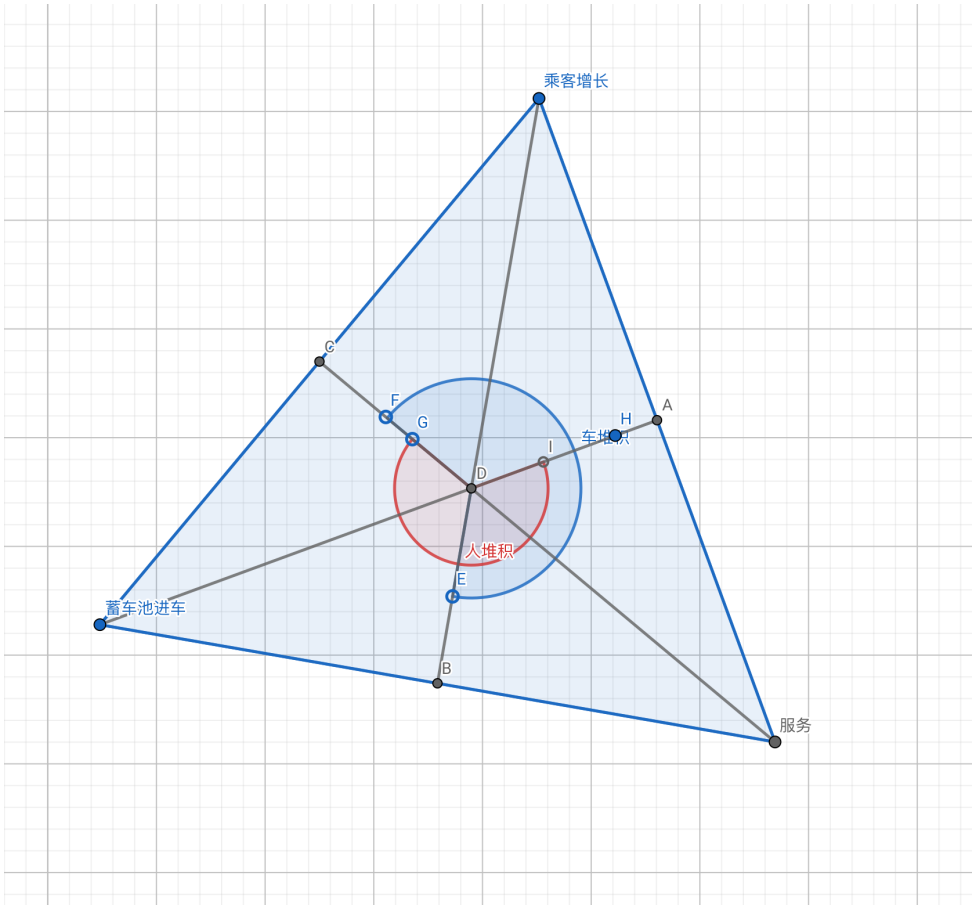

图 **4**　出租车的收益函数图示

将各个情形分别讨论之后，便能够着手进行等待时间加服务时间总和 t_{wait} 的计算：

1. 不堆积或人堆积时：

$$t_{\text{wait}} = \frac{n}{v_{\text{out}}} \tag{13}$$

其中 n 为蓄车池中出租车数量。

2. 人不堆积而车堆积时：

$$t_{\text{wait}} = \frac{n}{v_{\text{pin}}} \tag{14}$$

5.3 问题二的建模与求解

5.3.1 模型的建立

在郑州机场出租车秩序管理站上可以查询到实时的数据。

每个时刻蓄车池里面均有许多车辆在里面。利用数据，可以得到如下的统计图：

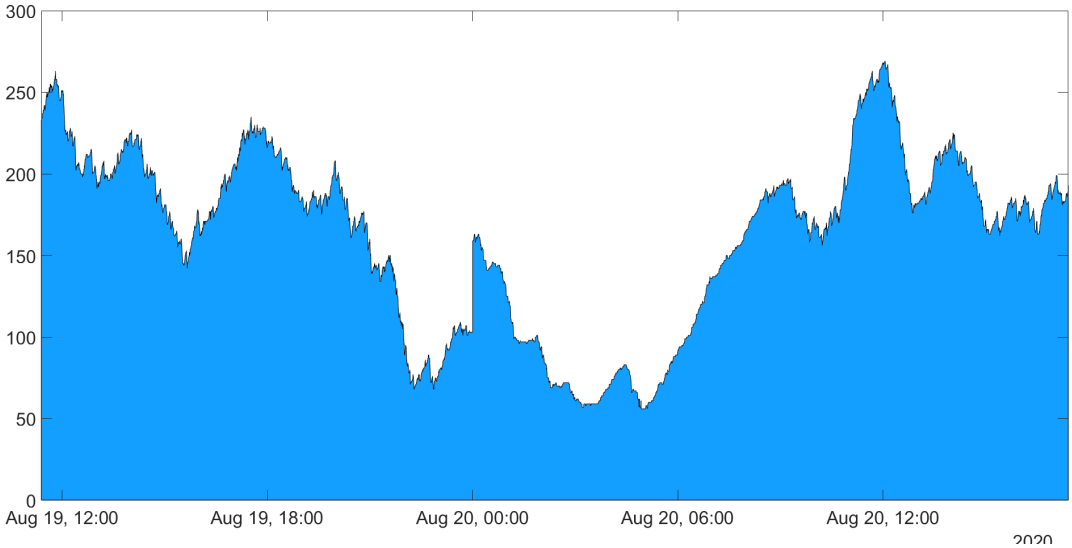

图 **5**　场内待运车辆数统计图

还可以得到每一个时刻汽车出站运载的个数：

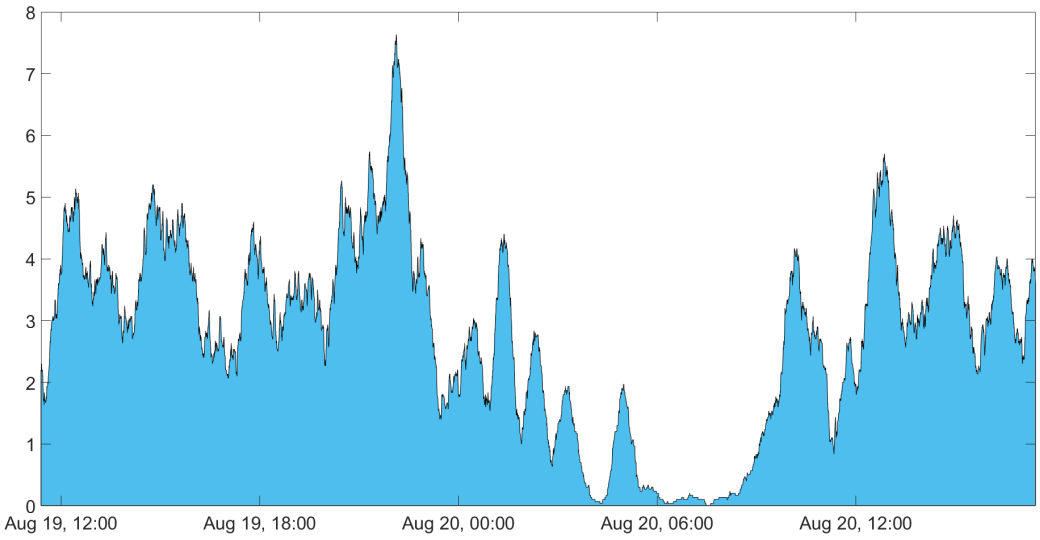

图 **6**　运客速度统计图（单位：辆**/min**）

将场内待运车辆数写成关于时间 t（单位：min）的表达式：$\eta(t)$。

再将出站车辆数写成关于时间 t（单位：min）的表达式：$\theta(t)$

要求出 t_{wait} 的大小，只需要一个积分方程。假设这个时候的时间点为 t，则上面函数以及 t_{wait} 满足约束式：

$$\eta(t) = \int_{t}^{t+t_{\text{wait}}} \theta(t)\mathrm{d}t \tag{15}$$

由于 $\theta(t)$ 无具体解析表达式，是统计得到的图表。有式子变换：

$$\eta(t) = \int_{t}^{t+t_{\text{wait}}} \theta(t)\mathrm{d}t \approx \sum_{i=0}^{[t_{\text{wait}}]} \theta(t+i) \tag{16}$$

找到满足条件的 t_{wait}，使得成立约束式：

$$\sum_{i=0}^{t_{\text{wait}}} \theta(t+i) \le \eta(t) \le \sum_{i=0}^{t_{\text{wait}}+1} \theta(t+i) \qquad (t_{\text{wait}} \in \mathbb{N}^*) \tag{17}$$

解出来的结果用面积统计图表示出来为：

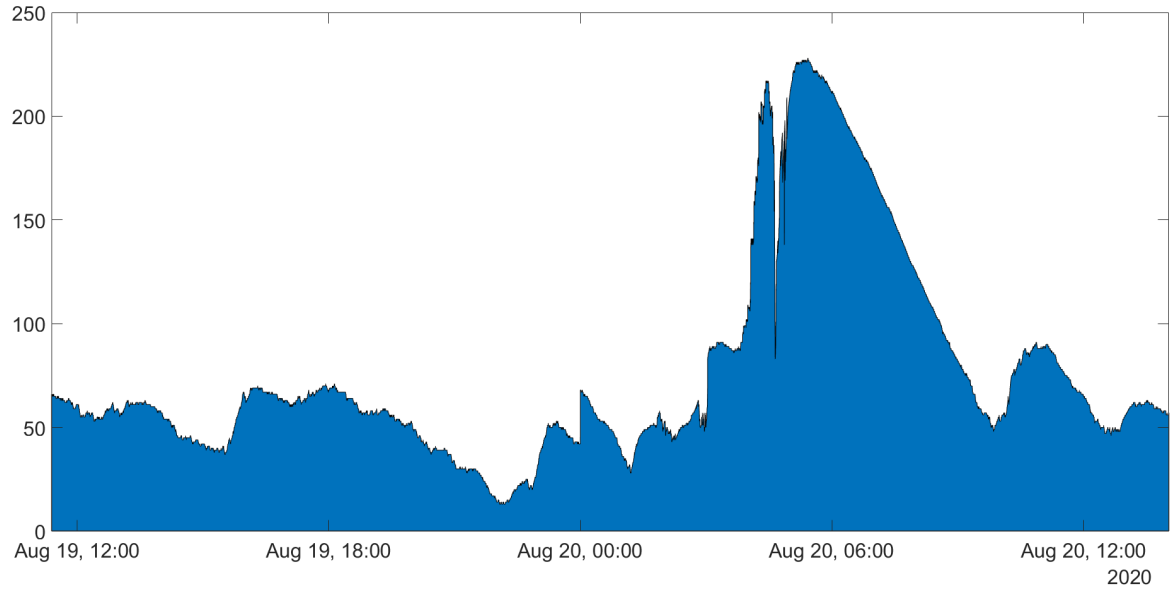

图 **7**　等待时间统计图（单位：**min**）

5.3.2 模型的求解

目前再网络上能够找寻到的数据有：郑州司机的日均收入：400 元人民币。按照每天工作 8 个小时计算，即每分钟 $\frac{5}{6}$ 元人民币。也即 $m_{\text{aver}} = \frac{5}{6}$ 元。

这时候，查询郑州的出租车价格信息网得知，价格费用与路程的关系为：

$$\text{fee}(l) = \begin{cases} c_0, & d \le 2km \\ c_0 + 1.5 \cdot (l-2), & 2 < d \end{cases} \tag{18}$$

c_0 在白天 8 元，在晚上 10 元。

而这时候，查询郑州的汽油价格信息网得知，92 号汽油每一升为 5.60 元，那么，每公里价格为 0.4746 元

查询高速路行车速度得知: 平均速度为 100km/h 即 $\frac{5}{3}$km/min

查询市区行车速度得知: 平均速度为 30km/h 即 0.5km/min

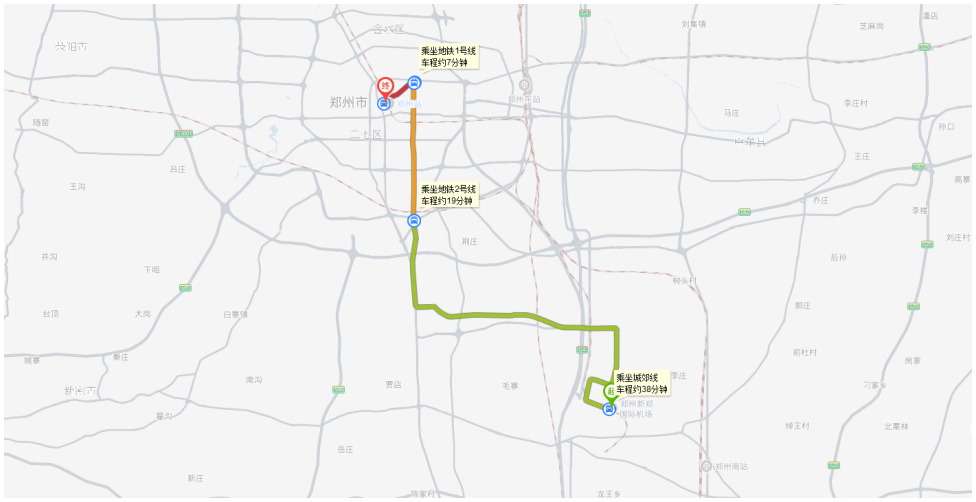

图 **8**　地图信息

根据地图查询结果得知：平均到市中心距离为：50.7km，其中到高速路口距离为 $l_{\text{urban}} = 35.7\text{km}$，高速路口到市中心距离为 $l_{\text{city}} = 15\text{km}$

其中，由于满足式子：

$$l_{\text{city}} = v_{\text{city}} \cdot t_{\text{city}} \tag{19}$$

所以有 $t_{\text{city}} = \frac{l_{\text{city}}}{v_{\text{city}}} = 30\text{min}$

也即最终得到如下信息表格：

表 **1**　得到的结果（部分）

m_{start}	m_{oil}	l_{init}	v_{sub}	m_{night}	m_{aver}	l_{urban}	k	v_{city}
8 元	0.4746 元 /km	2km	$\frac{5}{3}$km/min	2 元	$\frac{5}{6}$ 元	50.7km	1.5 元 /km	0.5km/min
t_{city}	v_{city}							
30min	0.5km/min							

代入式子：

$$t_0 = \frac{fee(l_{\text{urban}} + v_{\text{city}} \cdot t_{\text{city}}) - m_{\text{oil}} \cdot v_{\text{city}} \cdot t_{\text{city}}}{m_{\text{aver}}} - t_{\text{city}}$$

当中，得到

$$t_0 = \frac{6}{5} \cdot (fee(50.7 + 0.5 \cdot 30) - 0.4746 \cdot 0.5 \cdot 30) - 30 = \frac{6}{5} \cdot fee(65.7) - 38.5428$$

则当为白天时，t_0 为 $t_0 = 85.7172\text{min}$，当为夜班凌晨时，t_0 为 $t_0 = 88.1172\text{min}$

由于加收夜补时间为 22：00(含 22：00) 至次日 5：00(含 5：00)，于是可以利用 matlab 绘制出图像，蓝色部分时间为司机应当继续排队等待接客，红色部分代表司机不应该继续排队，应当直接回到郑州市中心。

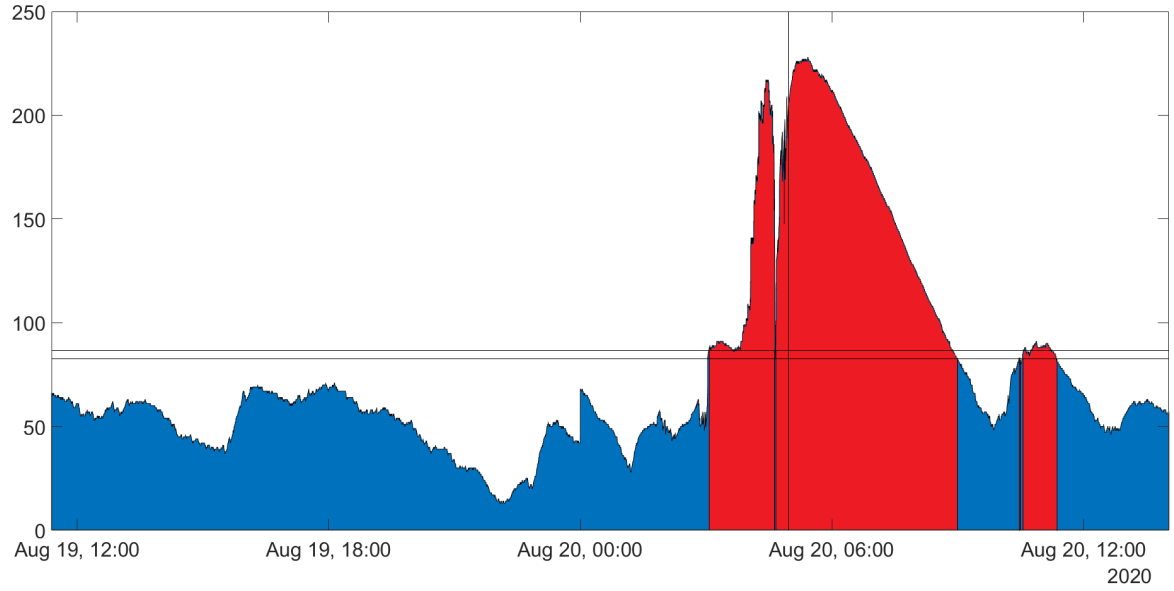

图 **9**

上面这幅图代表司机理应当做出的真正选择，但由于司机指能够根据他所见到的情景推算出大致的 t_{wait}，所以实际情况与应当选择的上一幅图是具有一定出入的，利用 1 问当中列举出的司机应当如何利用所得到的信息大致计算出还需要排多久。

收集到飞机航班信息之后，能够得到下图：

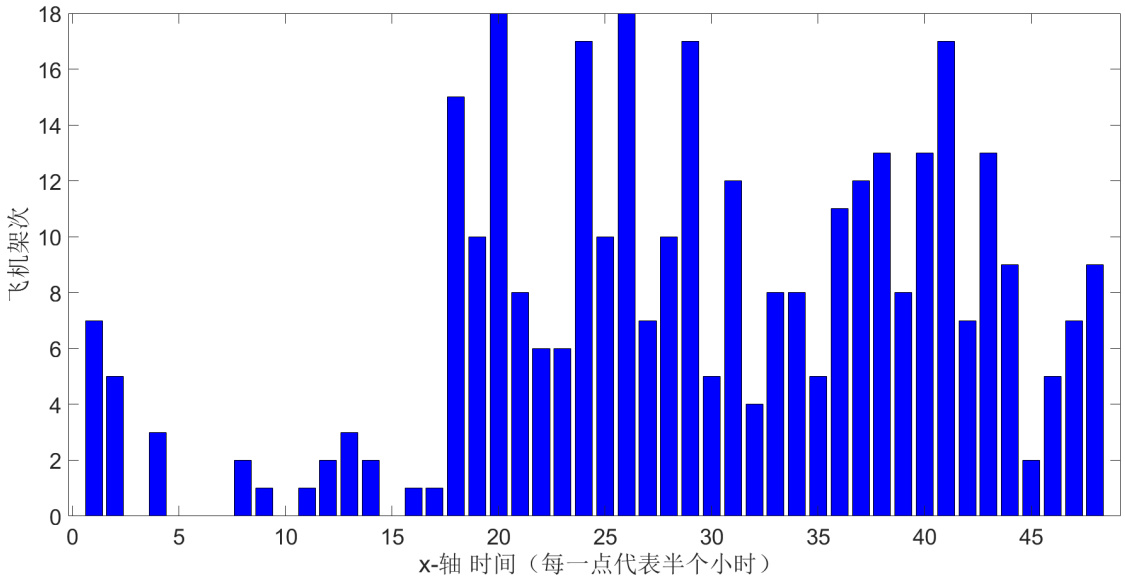

图 **10**

利用 $v_{\text{pin}} = \frac{n_{\text{fly}} \cdot 300 \cdot \alpha}{30}$ 以及当不堆积或人堆积时有 $t_{\text{wait}} = \frac{n}{v_{\text{out}}}$ 亦或者是人不堆积而车堆积时有 $t_{\text{wait}} = \frac{n}{v_{\text{pin}}}$，我们能够得到司机在那个时刻根据具体情况所判断出来的 t_{wait} 的值。

利用司机能够得知的内容与郑州市数据中心所产生的内容，也即下幅图片：

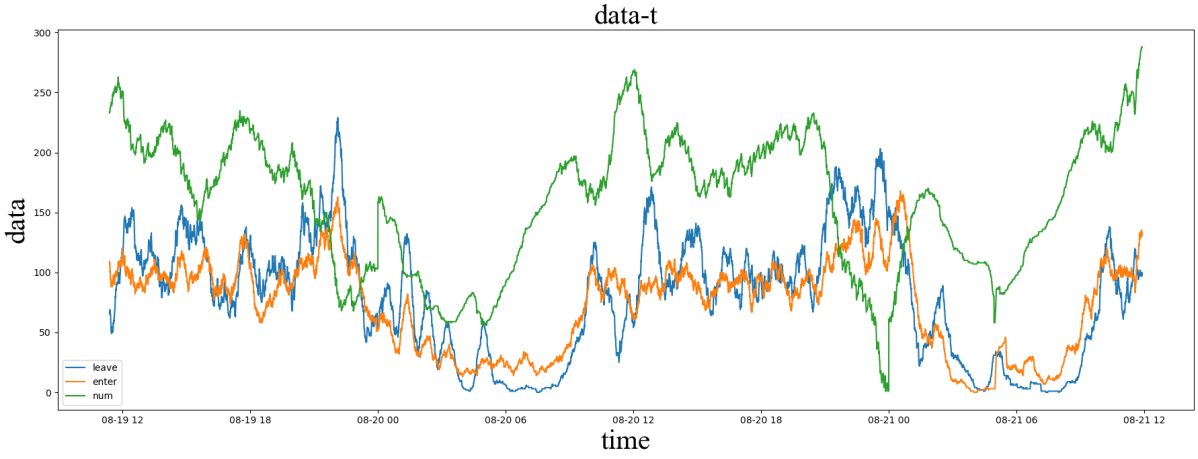

图 **11**

结合 matlab 程序，得到司机所观测到的等待时间 t_{wait}，其中，令 $\alpha \sim \frac{1}{10}$，得出

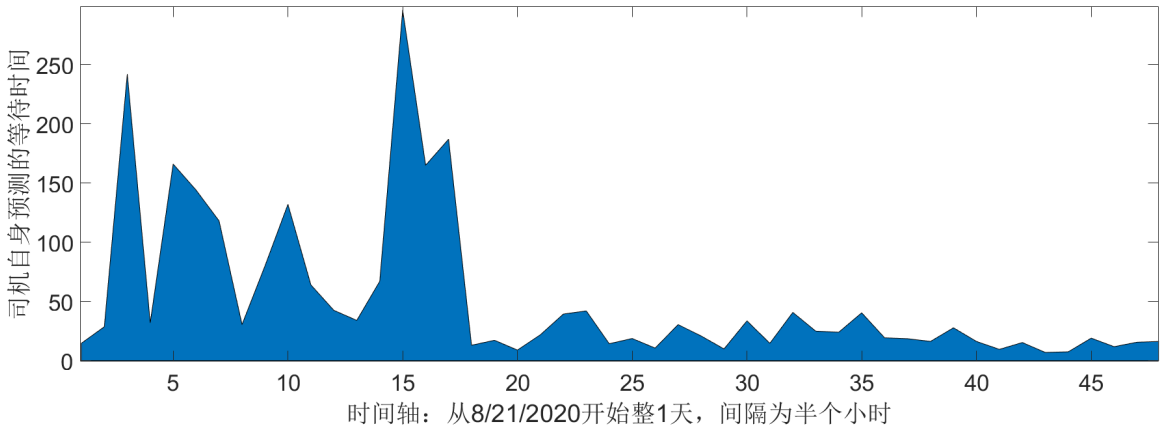

图 **12**

依旧按照上图蓝色部分时间为司机应当继续排队等待接客，红色部分代表司机不应该继续排队，应当直接回到郑州市中心的表示方法，利用 matlab 绘制出图像。

则司机自身的选择会变成：

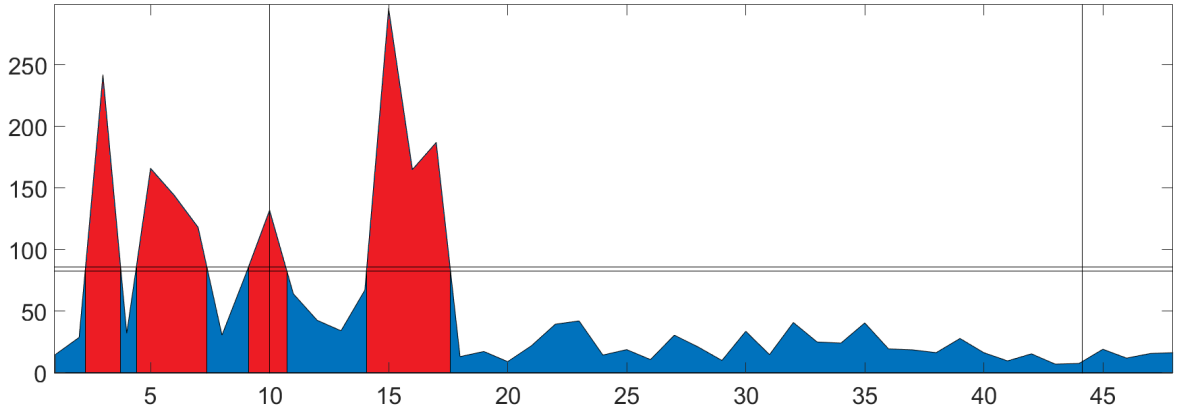

图 **13**

现利用 MSE 算法来检测真实值与预测值 (估计值) 差平方的期望。以分析模型的合理性。

把在 t_x 时间点真实需要等待的 t_{wait} 记作 $t_{t_x,0}$，司机通过自身观测数据得到的 t_{wait} 记作 $t_{t_x,1}$ 的有公式：

$$MSE = \frac{1}{M}\sum_{m=1}^{M}(y_m - \hat{y}_m)^2 = \frac{1}{48}\sum_{m=1}^{48}(t_{t_m,0} - t_{t_m,1})^2 \tag{20}$$

经过程序计算，得到均方误差为 $MSE = t_{\text{sum}} = 5.7433e+03$

5.4 问题三的建模与求解

5.4.1 模型的建立

第三问从机场管理部门角度出发，希望在有两条车道的前提下合理设置“上车点”，使得在保证车辆和乘客安全的条件下，使得总的乘车效率最高。

本文设置“上车点”的主要方案有以下几种:

1. 控制“上车点”的数量，从而控制服务速度 v_{out}。

2. 控制“上车点”的位置，从而控制蓄车池内出租车上限 N。

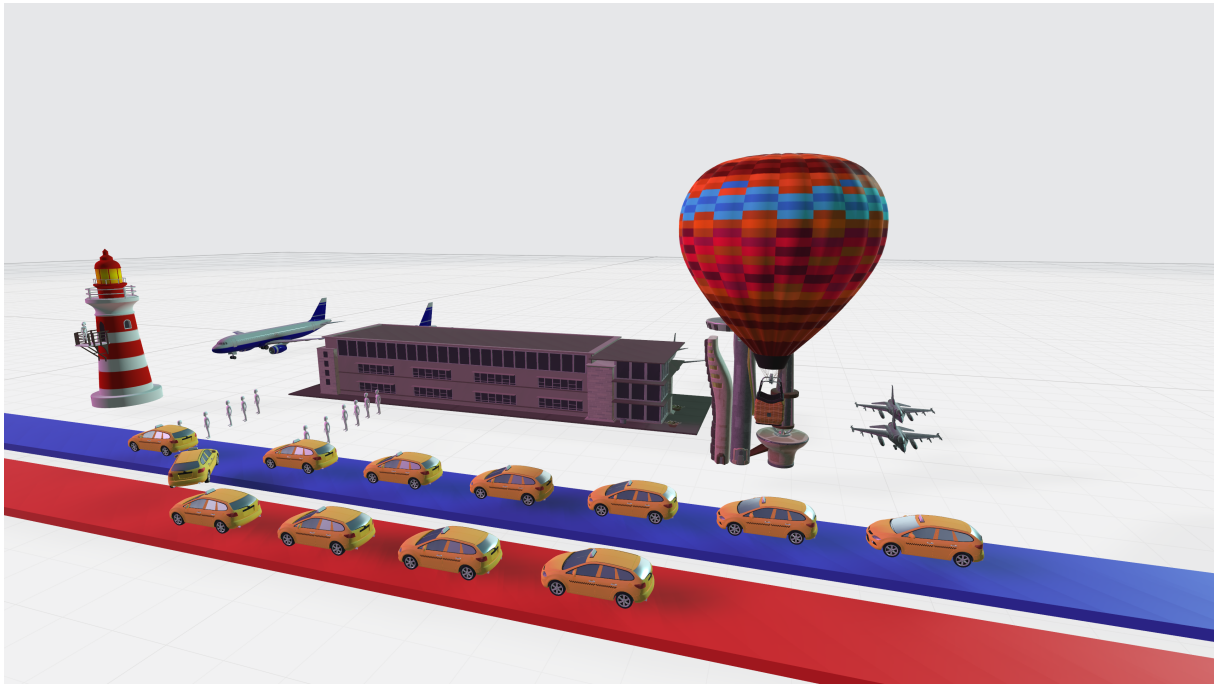

图 **14**　模拟送客示意图

5.4.2 元胞自动机仿真模型

为使得模型简单化，现只考虑蓄车池状态，并将蓄车池看成一个元胞。该元胞的主要参数为出租车数量 $taxinum$。其转移过程如下图所示：

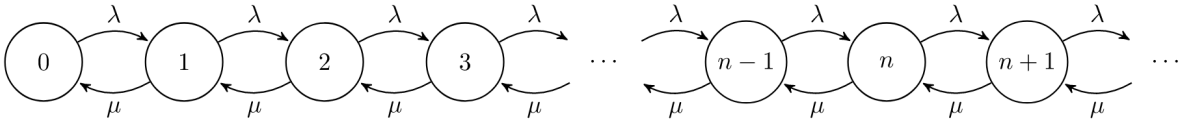

图 **15**　元胞转移过程

上图中 λ 为蓄车池增长概率，即单位时间元胞有 λ 概率增长 1；图中 μ 为服务概率，即单位时间元胞有 μ 概率减小 1。

如此，当“上车点”的数量增加，μ 值等比例增加。

为考察乘车效率，在此处定义流通系数：

$$flu = \frac{servenum}{taxinum} \tag{21}$$

上式中 $servenum$ 为服务状态出租车数量。当总出租车数量增大时，流通系数减小；当服务数量增大时，在总出租车数量保持不变的情形下，流通系数增大。

假定初始态 $\lambda = 0.1$，$\mu = 0.1$，$N = 20$, 用 $python$ 程序模拟元胞变化过程，结果如下图所示：

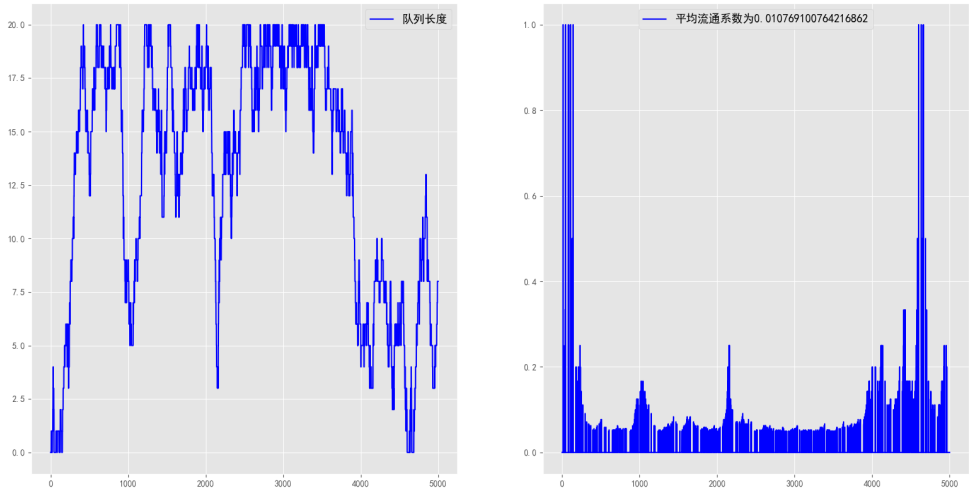

图 **16**　元胞仿真结果 **1**

现考察改变“上车点”数量时，流通系数的变化如下表所示：

上车点数量	蓄车池上限	μ	平均流通系数
1	20	0.1	0.0108
2	20	0.2	0.0724
3	20	0.3	0.0780
4	20	0.4	0.0856
5	20	0.5	0.0882
6	20	0.6	0.0926

可发现，当固定蓄车池上限时，流通系数随上车点数量增加而增加，因而应在保证安全的情形下尽可能多安排上车点。

下面分析“上车点”位置即蓄车池上限对平均流通系数的影响，默认 $\lambda = 0.3$，$\mu = 0.3$。

上车点数量	蓄车池上限	μ	平均流通系数
3	20	0.3	0.0638
3	30	0.3	0.0652
3	40	0.3	0.0509
3	50	0.3	0.0318
3	60	0.3	0.0176

会发现，确定上车点数量，当蓄车池上限增加时，总出租车数量可能性均值随之增加，因而流通系数下降。与实际情况相符。

根据以上分析，可以得到结论：应在保证安全的情形下提高上车点位置；降低蓄车池上限。

5.4.3 模型的求解

5.5 问题四的建模与求解

5.5.1 模型的建立

5.5.2 模型的求解

参考文献

[1] 参考文献

附录 A　爬取郑州机场出租车管理站代码–C-sharp 源代码

```
using System;
using System.IO;
using System.Net;
using System.Text.RegularExpressions;
using System.Threading;

namespace c_2019
{
    internal class ResourcePool
    {
        public static string HttpGet(string Url, string contentType)
        {
            try
            {
                string retString = string.Empty;

                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(Url);
                request.Method = "GET";
                request.ContentType = contentType;

                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream myResponseStream = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(myResponseStream);
                retString = streamReader.ReadToEnd();
                streamReader.Close();
                myResponseStream.Close();
                return retString;
            }
            catch (Exception ex)
            {
                return ex.Message;
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            string head = "time,numbers,enter,leave";
            Console.WriteLine(head);
            using (StreamWriter file = new StreamWriter(@"val.csv", true))
            {
                file.Write(head);
```

```
        }
        for (; ; )
        {
            string ans = NetworkAirport.getContent();
            Console.WriteLine(ans);
            using (StreamWriter file = new StreamWriter(@"val.csv", true))
            {
                file.WriteLine(ans);
            }
            Thread.Sleep(5000);
        }
    }
}

class NetworkAirport
{
    public static string getContent()
    {
        try
        {
            string ans = "";
            string head = "http://www.whalebj.com";
            string content = ResourcePool.HttpGet(head + "/xzjc/default.aspx", "");

            string pattern = "截止目前为止（[0-9]*-[0-9]*-[0-9]* [0-9]*:[0-9]*:[0-9]*）";
            MatchCollection mc = Regex.Matches(content, pattern);
            string time = mc[0].Value.Substring(7, 19);
            ans += time + ",";

            pattern = "场内待运车辆数为：[0-9]*";
            mc = Regex.Matches(content, pattern);
            string vehiclesNumColl = mc[0].Value.Substring(9);
            ans += vehiclesNumColl + ",";

            pattern = "前半小时进场车辆数为：[0-9]*";
            mc = Regex.Matches(content, pattern);
            vehiclesNumColl = mc[0].Value.Substring(11);
            ans += vehiclesNumColl + ",";

            pattern = "前半小时离场车辆数为：[0-9]*";
            mc = Regex.Matches(content, pattern);
            vehiclesNumColl = mc[0].Value.Substring(11);
            ans += vehiclesNumColl;

            return ans;
        }
        catch (Exception e)
```

```
        {
            return "";
        }
    }
  }
}
```

附录 B　爬取郑州机场航班信息代码–JavaScript 源代码

```
console.log('"载体","飞行","起源","状态","到来"');

function getarrivals(){
  for (let i = 1; i < 9; i++) {
    setTimeout(() => {
      var timeperiod = document.getElementsByClassName("timeperiod");
      var timeperiodNode = timeperiod[0].childNodes
      var shouldClick = timeperiodNode[i * 2];
      shouldClick.click();
      setTimeout(() => {
        var arrivals = document.getElementById("arrivals");
        var nodeColl = arrivals.childNodes;
        for (let j = 0; j < nodeColl.length; j++) {
          var subTempText = "";
          var airData = nodeColl[j].childNodes;
          for (let k = 0; k < airData.length - 1; k++) {
            var text = airData[k].childNodes[0].textContent;
            subTempText += '"' + text + '"' + ',';
          }
          subTempText += '"' + airData[airData.length - 1].childNodes[0].textContent +
              '"';
          console.log(subTempText);
        }
      }, 5000);
    }, i * 6000);
  }
}

function getdepartures(){
  for (let i = 1; i < 9; i++) {
    setTimeout(() => {
      var timeperiod = document.getElementsByClassName("timeperiod");
      var timeperiodNode = timeperiod[0].childNodes
      var shouldClick = timeperiodNode[i * 2];
      shouldClick.click();
      setTimeout(() => {
```

```
            var departures = document.getElementById("departures");
            var nodeColl = departures.childNodes;
            for (let j = 0; j < nodeColl.length; j++) {
                var subTempText = "";
                var airData = nodeColl[j].childNodes;
                for (let k = 0; k < airData.length - 1; k++) {
                    var text = airData[k].childNodes[0].textContent;
                    subTempText += '"' + text + '"' + ',';
                }
                subTempText += '"' + airData[airData.length - 1].childNodes[0].textContent +
                    '"';
                console.log(subTempText);
            }
        }, 5000);
    }, i * 6000);
  }
}
```

附录 C　第二小问解题方法及绘图–Matlab 源代码

```
time = val.time;
numbers = val.numbers;
enter = val.enter;
leave = val.leave;

e = enter / 30;
l = leave / 30;

t_wait = 1:18000;

%initializing the t_wait
interval = 12

for i=1:18000
    t_wait(1:i)=0;
end


%the main code
for i=1:18000
    temp = 0;
    for j=1:18000
        if  temp >= numbers(i,1)
            t_wait(i) = j;
            break
```

```
        else
            temp = temp + l(j * interval + i,1);
        end
    end
end


time_temp = time(1:18000).';
area(time_temp,t_wait,'DisplayName','t_wait')
xlim([time_temp(1,1) time_temp(1,18000)])

area(time,l,'DisplayName','l')
xlim([time_temp(1,1) time(20282)])
area(time,numbers,'DisplayName','numbers')
xlim([time_temp(1,1) time(20282)])


time_arr = arrival.time;
status = arrival.status;

t = 1/48

data = 1:48;

for i=1:48
    data(1,i) = 0;
end

for i=1:48
    temp = 0;
    for j=1:368
        if time_arr(j) > (t*(i-1))
            if time_arr(j) < (t*i)
                temp = temp + 1;
            end
        end
    end
    data(1,i) = temp
end

plot(xx,data)
bar(xx,data,'DisplayName','data')


START = 8531

GAP = 360
```

```
t_wait_t = 1:48;

for i=1:48
    t_wait_t(i) = 0;
end

for i=1:48
    num = data(1,i);
    if num == 0
        num = 0.5;
    end
    t_wait_t(i) = numbers(START+(i-1)*GAP) / num
end



>> xx = 1:48;
>> histogram(t_wait_t,xx)
>> area(xx,t_wait_t,'DisplayName','t_wait_t')
>> area(xx,t_wait_t,'DisplayName','t_wait_t')
>> xlim([0.5 48])
>> xlim([1 48])
```