基于改进排队论和元胞自动机算法的出租车决策模型

摘要

随着交通运输行业的不断发展，人们出行频率的不断提高，出租车服务日趋成为人们出行的重要方式。本文利用了元胞自动机算法和改进排队论模型讨论了出租车司机的运营决策问题。

针对问题一，首先，需要将在城市内运营和前往机场运营进行对比，分析各种可能的影响因素，得到前往机场运营的净效率表达式。其次，利用改进排队论模型分析在$\mathrm{M/M/C/m/\infty}$ FCFS 模式下出租车等待服从参数为λ的泊松分布的客流量所需的等待时间。最后，利用机理分析得到净效率表达式中其余影响参数。

针对问题二，以上海市浦东国际机场的数据作为案例，发掘出 9 月 13 日上午 8:00-12:00 的实时数据，代入问题一所建立起的模型中进行分析求解。首先，利用机场的客流量数据，按照出站情况的讨论，得到客流出站花费时间的概率密度函数。接着，将概率密度函数代入改进排队论模型中可以求得出租车司机的等待时间为每辆车$2.0916min$。继而，利用上海强生出租车公司的数据求得净收益表达式中其他相关参数。最后，代入表达式中可以求解出：当前方的出租车数量$\mathrm{M} \leq 80$时，净效益$P_{com} > 0$；反之，$\mathrm{M} > 80$，$P_{com} < 0$。

针对问题三，为了合理地设置“上车点”，采用改进双排队论模型对于出租车和乘客两个排队系统进行分析研究，采用元胞自动机算法进行求解。以上海市浦东国际机场情况为例：首先选定元胞参数：道路长度$L = 850m$、元胞长度$l = 5m$、元胞的数目$N = 170$。其次，在制定的安全规则基础上进行模拟，得到一系列数据值，将运营效率N和“上车点”数量x之间进行函数拟合，可以得到两者的关系为：$N = 169.01 * e^{-\frac{1.16}{x}}$；综合考虑成本因素，上车点数量$x = 7$为最佳。进行函数的检验，其可决系数$\mathrm{R^2} = 0.994$，较为稳定。

针对问题四，考虑到等待时间成本的影响，制定补偿方案为：给予进行短途运营的出租车司机“短途票”作为补偿，使其免于下一次的排队，保障其最基本的运营效率。

关键词：　改进排队论　元胞自动机　决策模型　机理分析法

1 问题重述

1.1 问题背景

大多数乘客下飞机后要去市区的目的地，出租车是主要的交通工具之一。国内多数机场都是将送客与接客通道分开的。送客到机场的出租车司机都将会面临两个选择：

(A) 前往到达区排队等待载客返回市区。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，等待时间长短取决于排队出租车和乘客的数量多少，需要付出一定的时间成本。

(B) 直接放空返回市区拉客。出租车司机会付出空载费用和可能损失潜在的载客收益。

在某时间段抵达的航班数量和“蓄车池”里已有的车辆数是司机可观测到的确定信息。通常司机的决策与其个人的经验判断有关，比如在某个季节与某时间段抵达航班的多少和可能乘客数量的多寡等。如果乘客在下飞机后想“打车”，就要到指定的“乘车区”排队，按先后顺序乘车。机场出租车管理人员负责“分批定量”放行出租车进入“乘车区”，同时安排乘客上车。在实际中，还有很多影响出租车司机决策的确定和不确定因素，影响效果也不尽相同。

1.2 目标任务

(1) 分析研究与出租车司机决策相关因素的影响机理，综合考虑机场乘客数量的变化规律和出租车司机的收益，建立出租车司机选择决策模型，并给出司机的选择策略。

(2) 收集国内某一机场及其所在城市出租车的相关数据，给出该机场出租车司机的选择方案，并分析模型的合理性和对相关因素的依赖性。

(3) 在某些时候，经常会出现出租车排队载客和乘客排队乘车的情况。某机场“乘车区”现有两条并行车道，管理部门应如何设置“上车点”，并合理安排出租车和乘客，在保证车辆和乘客安全的条件下，使得总的乘车效率最高。

(4) 机场的出租车载客收益与载客的行驶里程有关，乘客的目的地有远有近，出租车司机不能选择乘客和拒载，但允许出租车多次往返载客。管理部门拟对某些短途载客再次返回的出租车给予一定的“优先权”，使得这些出租车的收益尽量均衡，试给出一个可行的“优先”安排方案。

2 问题分析

2.1 问题一的分析

问题一需要研究对于出租车司机决策的相关影响因素，建立出租车调度的决策模型并且给出相应的选择策略。通过查阅相关文献[1]，并且需要结合机场服务出租车辆的实际情况，考虑到对于出租车司机做出决策的最重要衡量指标为其运营出租车活动的盈利效率，对应于本问题中，最直接的影响因素为出租车司机前往机场接客过程中的时间成本。并且需要找出对于时间成本的相关影响因素，在如上的种种因素影响下，司机的等待时间成本的计算变为了对于机场出租车排队问题的考量，需要通过建立起排队论数学模型，对于司机前往机场接客的运营效率进行计算，并将之与在市区中接客运营的效率进行对比，得到最终的决策结果。

2.2 问题二的分析

问题二在问题一所建立模型的基础之上，选择实际的机场案例，找寻相关的真实可靠数据，对于实际情况进行出租车司机的决策计算，即比较两种不同运营模式情况下的收益情况，选择更优解。所选取的机场需要满足人流量大、具有典型代表意义等要求，因此，我们选择了上海浦东国际机场作为研究对象。此外，第二题的难点是对于相关机场及出租车资源数据的挖掘。为了解决数据的发掘问题，我们通过各种网站及途径进行数据的寻找，并利用 Python 软件进行数据的发掘和利用。通过对于数据的分析和处理，可以得到问题一模型中相关参数的确定，进而完成问题二的求解。

2.3 问题三的分析

在问题一的基础上，问题三需要进一步考虑两车道设置多接口的问题，使得设置的接口数量使得乘客的排队上车效率最高，出租车的等待时间降低，运营效率得以增长。在此目标下，我们采用双排队论模型，并且利用元胞自动机的算法进行乘客分布的随机模拟，进而求解在不同接口数量情况下，出租车运营的效率情况，找到最优的“上车点”设置方案。

2.4 问题四的分析

在问题一中，我们会发现乘客的目的地的对出租车司机的收入影响很大。但是本着高效率的且公平的“先来后到，不许拒载”原则，司机无法选择乘客。由于等待时间往往很大，在巨大的时间成本下，如果司机接到里程数较少的顾客，

司机一天的收入会大大降低。并且在影响工作状态下，会降低服务质量，甚至造成交通事故。所以，对顾客里程数较少的出租车司机应给予一定的补偿和和优惠政策以平衡随机事件对司机的影响。

要规定一个合理且可行的补贴方案，就要兼顾公平性和可行性两条原则。公平性是指，顾客里程数较低的出租车司机补偿后的期望收益效率η_{com}与平均收益效率接近$\bar{\eta}$，不可过高或者过低。而可行性是指，这一补偿方案符合实际，难以作弊，且在现有的道路规划上能够实现。解决思路是：首先要确定补偿方法的基本类型，以满足可行性，然后在具体的参数上进行调整以满足公平性。

3 模型假设

1. 假设在机场排队候车的过程中，乘客及出租车司机均遵守先来后到的排队规则，没有胡乱插队的现象发生。
2. 由于每一批次乘客上车所需时间差异很小，忽略由于携带多件行李造成的上车时间耽搁，默认每一批次的乘客上车所需时间相同。
3. 机场出租车司机接客不存在跨市的长途旅行，即认为出租车最多在该城市和机场之间进行往返。
4. 假设对于同一地区的出租车收费方案相同，即不考虑由于出租车大小及型号不同所导致的收费差异。

4 符号说明

变量符号	变量名称	单位
P_{air}	机场运营收益	元
P_{ur}	市内运营收益	元
P_{com}	机场运营的净收益	元
ρ	服务强度	无
T_w	等待时间	min
T_y	出租车运营时间	min

5 问题一的模型建立

5.1 影响因素及运营效率对比的分析

伴随着交通的不断发展，人们的出行变得愈发便利，出租车资源作为城市公共交通的重要组成一环，为居民的出行提供了巨大的便捷。因此，如何进行市区和机场出租车运营活动的调度降低出租车司机的空载时间，使得出租车司机的运营效率得以提升便成了一项重要的决策活动[2]。

5.1.1 影响因素的确定

据机场实际的运行需要，出租车运行决策的最终目标是提升对旅客的高水平、高效率交通服务，实现旅客、机场和出租车驾驶员之间的多赢局面[3]。针对出租车驾驶员而言，出租车运营活动的效率便成了重要的衡量指标因素。因此，为了衡量出租车司机运营活动的效率，采用等待时刻到达航班的数量、容客量及满载率（例如，在某一时刻到达了多少辆航班及每辆航班的可载客人数和实际乘坐人数）、天气情况、在机场等候排队接客的出租车数量以及乘客相应的目的地里程数期望值五个指标进行考虑。其中，到达航班的数量、容客量及在机场等候的出租车数量属于确定因素；天气情况、航班的满载率及乘客目的地的里程期望值属于不确定因素，对应关系见于图 1：

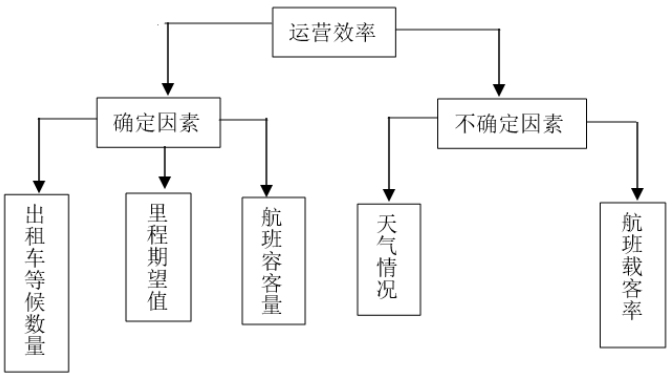

图 1 运营效率影响因素示意图

5.1.2 市区与机场运营效率对比

1.机场运营效率：

相较于在市区运营而言，选择在市区和机场之间往返接客时，往往需要较长的时间空车驶往机场，并且在机场排队接客时需要消耗大量的时间进行排队，带

来较大的时间成本，除此之外，由于市区一般离机场的路途较远，前往接客的过程中也会带来更大的燃油成本，使得司机的单次盈利金额减少；但与此同时，机场的乘客往往会前往目的地较远的地方，也就是意味着出租车司机每一次接单的行驶里程会更远，带来更高的运营效率。基于种种正负影响因素的考量，可以得到机场运营收益P_{air}的表达式为：

$$P_{air} = E(P_{km} - C_{km})$$

式中，E为乘客所到达目的地的期望里程值，P_{km}为单位公里的计价，C_{km}为单位公里的燃油成本。

2.市内运营效率：

在市内运营时，出租车司机不需要前往很远的指定地点进行接客，往往是在市内的地点就近接客，因此可以节省下很大一部分的燃油成本；并且相较于在机场接客，可以省去出租车在机场排队接客的环节，使得时间成本也得到大大的降低;但在市内运营接客时,由于有更多出租车资源分布的原因,竞争压力会更大，除此之外，由于接客地点的不确定性，能否接到乘客也存在很大的随机性，在一定程度上会造成部分空间的资源冗余，使得运营的效率降低。相应地，可以得到市内运营收益P_{ur}的表达式为：

$$P_{ur} = P_w(T_s - T_h) - S_h C_{km}$$

其中，T_s为时间基准， T_h为从机场返回路途中所消耗时间，P_w为在市区内运营收益期望，S_h为从机场返回路途里程。

时间基准T_s包含两个部分，即等待时间T_w和出租车运营时间T_y，其表达式为：

$$T_s = T_w + T_y$$

3.机场运营的净收益效率

为了探究出租车司机是否前往机场进行拉客的决策问题，我们引入目标函数为机场运营的净收益效率：

$$P_{com} = P_{air} - P_{ur} = E(P_{km} - C_{km}) - [P_w(T_s - T_h) - S_h C_{km}]$$

化简可得：

$$P_{com} = EP_{km} - EC_{km} - P_w T_s + P_w T_h + S_h C_{km} \quad (1)$$

其中，目的地的期望里程值E、市区内运营收益期望P_w和从机场返回路途里

程S_h属于可确定的部分，不确定因素的影响主要体现在对于等待时间基准的影响上，因此，需要进一步建立模型探讨不确定因素对于时间基准（主要是等待时间）的影响。

5.2 改进排队论原理简介

排队论是在研究各种排队系统概率规律性的基础上，解决相应排队系统最优设计和最优控制的问题[4]，一般地，排队论的原理是应用在乘客的排队策略当中，解决乘客的等待时间问题。但是，本文基于传统的排队论观念进行一定程度的改进与创新，将排队论的原理逆向使用在出租车资源系统的规划当中，进一步解决出租车在机场等候的效益计算问题。

5.2.1 排队系统的组成

一般的排队系统有三个基本组成要素：输入过程、排队规则及服务机构，且排队过程如图 2 所示。顾客从机场出发，到达乘坐出租车的等候区时，按照排队规则进行排队等待服务，并且在接受完服务后结束服务[5]。

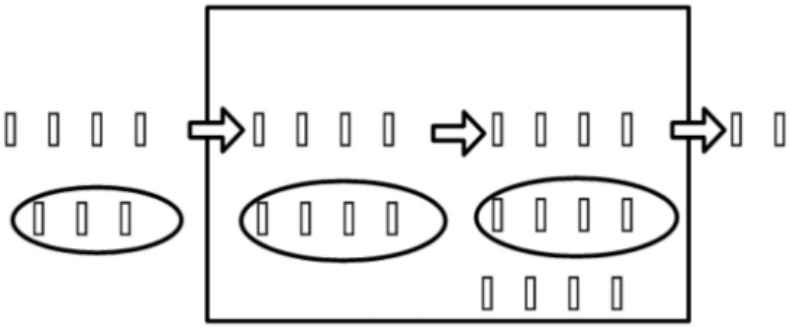

图 2 排队理论流程示意图

5.3 基于改进排队论的出租车等候模型的建立

假设在航班到达之后，旅客到达等车队列的时间呈随机分布，且乘客到达的间隔时间和服务时间是相互独立分布的。因此，排队服务队形服从M/M/C/m/∞ FCFS 模型。此模型是指：乘客到达指定候车地点服从泊松分布，服务时间服从参数为λ的负指数分布，假设每个乘客上车时间服从参数μ的负指数分布，等待时间T_p指平稳状态顾客在系统中排队等待的时间，期望值记作W_q，s是系统中并行服务的工作台数，亦即上车点数量[6]。在此模式下认为工作台接口数量为 m，乘客数量相对于出租车数量无限，并且按照先来后到的规则进行服务。根据服务

强度的定义，可以得到候车系统的服务强度表达式为：

$$\rho = \frac{\lambda}{\mu s}$$

当系统处于稳态时，稳态概率的关系表达式为：

$$P_n = \begin{cases} \frac{\lambda}{\mu s} P_{n-1} & s \le C \\ \frac{\lambda}{\mu C} P_{n-1} & s > C \end{cases} \tag{2}$$

考虑候车系统处于n个工作台同时工作的状态下时，在平衡状态下，根据系统“流入=流出”的原理，可以得到任意状态下的平衡方程：

$$\begin{cases} \mu_1 P_1 = \lambda_0 P_0 \\ \lambda_0 P_0 + \mu_2 P_2 = (\mu_1 + \lambda_1) P_1 \\ \lambda_1 P_1 + \mu_3 P_3 = (\mu_2 + \lambda_2) P_1 \\ \vdots \\ \lambda_{n-2} P_{n-2} + \mu_n P_n = (\mu_{n-1} + \lambda_{n-1}) P_{n-1} \\ \lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} = (\mu_n + \lambda_n) P_n \end{cases} \tag{3}$$

根据概率分布的归一化理论，可以得到：

$$\sum_{n=0}^{\infty} P_n = 1 \tag{4}$$

结合公式（2）和（3），可以得到：

$$\left(1 + \sum_{n=1}^{\infty} C_n\right) P_0 = 1$$

于是，P_0的表达式为：

$$P_0 = \frac{1}{1 + \sum_{n=1}^{\infty} C_n} \tag{5}$$

通过递推关系式（2），即可以得到在同一时刻下，任意i个工作台工作的概率P_i。

可以求出平均被占用的出租车数量为：

$$\bar{s} = \sum_{i=0}^{m} P_i \times i \tag{6}$$

进一步可以推得出租车司机的等待时间：

$$T_w = \frac{K}{\bar{s}} \tag{7}$$

式中，K为排在该出租车司机前的等候出租车数量。

6 问题二的模型的求解与分析

6.1 机场出租车决策模型的求解——以上海市浦东国际机场为例

6.1.1 案例选择及参数确定

1.机场的选择

上海市浦东国际机场位于上海市的东边，是我国三大机场之一，处于上海市东边，属于滨海地带，距离市中心的距离大约为40km[7]。因此，可以认为在机场乘坐出租车的乘客，其里程期望值为40km。

浦东机场的出租车等候区具有两个上车点[8]，即具有 2 个服务接口。因此，我们采用$M/M/C/2/\infty$ FCFS 模型进行分析和计算。

对于上海浦东机场的航班信息，采用 FlightAware 网站的数据信息对于航班进行实时的跟踪和记录，得到了机场的全年大数据，见于图 3 和 9 月 12 日 20:00-9 月 13 日 20:00 期间国际机场的航班数据列于附录中。

图 3 机场全年数据统计图

根据航班到达信息以及 ctuana.variflight.com/fast/对上海浦东国际机场的统计：航空公司载客率在 80%以上，白天到达乘客百分之十五乘坐出租车，晚上到达乘客百分之四十乘坐出租车且平均每辆车乘坐 1.5 个乘客。（数据来源：https://bbs.feeyo.com/thread-5931815-1-1.html）

2.其他参数数据的确定

为了完成出租车司机运营效益的计算，需要根据公式（1）：

$$P_{com} = EP_{km} - EC_{km} - P_w T_s + P_w T_h + S_h C_{km}$$

结合所收集到的数据，完成相关参数的确定。

（1）里程期望值E

考虑到上海市特殊的政治、经济地位，来到上海市进行商务工作上的业务的人员居多，而上海市的主要金融集聚地、商务办公场所以及大型写字楼都集中在上海市中心。因此，考虑普遍情况，认为大多数乘客的目的地都在上海市中心的范围内。浦东国际机场距离上海市中心的距离大约为$40km$，故认为里程期望值$E = 40km$。

（2）市区内运营收益期望P_w及前往从机场返回途中所消耗时间T_h

根据上海市强生出租车公司管理的出租车行驶数据[10]，可以得到上海市出租车上海市中心运营时的单次载客时间、单次载客里程以及空驶率的数据，见于附录中，通过利用 MATLAB 计算，可以得到在市区内运营收益期望P_w见于图 4：

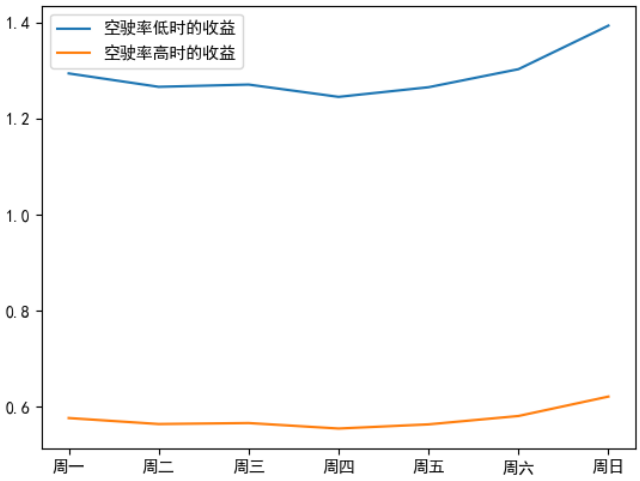

图 4 市区内运营收益期望变化示意图

根据所选取的航班信息为星期五上午 10:00 的情况，所以可以求得P_w为 1.265 元/min。

对于从机场返回途中所消耗时间T_h，有如下关系式：

$$T_h = \frac{S_h}{V_{car}}$$

其中，S_h一般为 40km，V_{car}为出租车于机场和市中心之间的行驶速度，一般为 40 km/h，所以，可以得到：

$$T_h = \frac{S_h}{V_{car}} = \frac{40}{40} = 60min$$

（3）单位公里计价P_{km}

通过查找相关资料，可以得到上海市的出租车计价方案，如表 1 所示：

表 1 上海市出租车计价方案

公里数	**日间**(5:00~23:00)	**夜间**(23:00~5:00)
0~3 公里	14 元	18 元
3~15 公里	2.5 元/公里	3.1 元/公里
15 公里以上	3.6 元/公里	4.7 元/公里

进行近似化处理，在计算单位公里计价时需要将计价随里程数的分段函数关系做平均化处理，得到单位公里计价P_{km}。假设期望里程值为$40km$，则总收费W_{total}为：

$$W_{total} = 14 + (15 - 3) \times 2.5 + (40 - 15) \times 3.6 = 134 \text{ 元}$$

则单位公里计价P_{km}为：

$$P_{km} = \frac{W_{total}}{E} = \frac{134}{40} = 3.35 \text{ 元}/km$$

（4）燃油成本C_{km}

目前上海市的出租车大部分采用燃油方式的汽车，少部分出租车采用燃气的方式[9]。对于上海市的出租车车型，一般分为大众、捷达、起亚几种类型，一般均为 1.6L 排量的类型，平均 100km 的油耗在 8.4L 左右。目前，汽油的价格约为 7.75 元/L,所以可以计算出每公里燃油成本C_{km}为：

$$C_{km} = \frac{W_{oil}}{S_{car}} = \frac{7.75 \times 8.4}{100} = 0.65 \text{ 元}/km$$

6.1.2 客流数据分析

2019 年 9 月 13 日上午 8:00-12:00 上海浦东国际机场的实时机场到港人员变化示意图如图 5 所示：

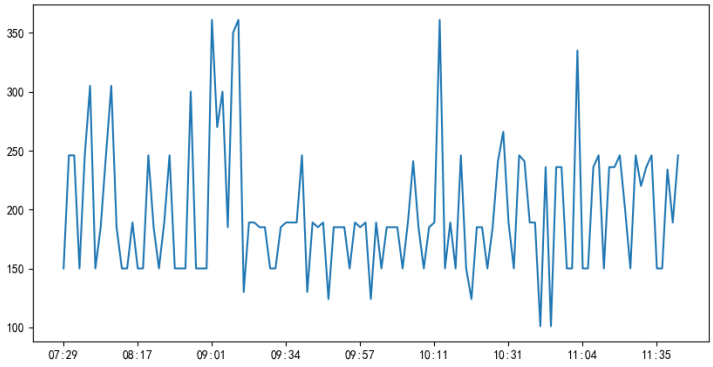

图 5 浦东机场到港客流量变化示意图

从图中可以看到在上午 8:00-12:00 的时段中主要出现了四次到港客流量的高峰期和若干个相对平稳的时段。影响机场客流量波动的主要因素是航班计划，在航班降落的时刻机场会涌入一大批乘客。

对于在机场的乘客，从航班降落到出站时刻之间仍有一系列的活动，存在一定的时间分布，根据 Kim Wonkyu[11]对于出站时间分布的讨论，结合我们所得到的数据，其分布关系如图 6 所示：

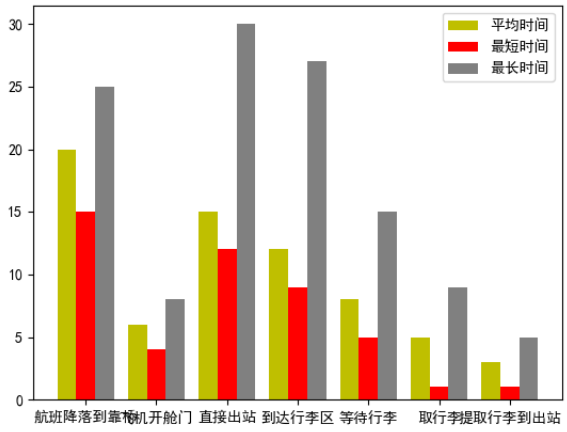

图 6 机场乘客各过程平均花费时间示意图

根据相关文献[12]中对于机场乘客数据的处理方法，结合上海浦东机场的客流人：员构成，可以认为不取行李和取行李的顾客数量之比约为 7:3，则客流到达出站口出站所需时间 X 的概率密度函数为：

$$f(x) = 0.7f_1(x) + 0.3f_2(x)$$

其中，$f_1(x)$和$f_2(x)$分别为不取行李和取行李的顾客出站所需时间 X 的概率密度函数。针对本题的数据，运用 Python 求解可以得到平均概率密度如图 7 所示：

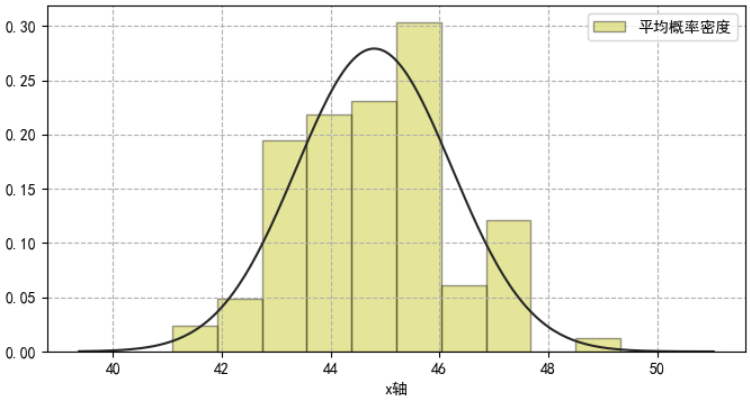

图 7 平均概率密度分布图

6.1.3 等待时间求解

以上，得到了客流数据在机场的等候时间情况，利用问题一中建立起的排队论模型，利用 Python 软件进行编程分析，代码见于附录，可以求解得出出租车前方平均每有一辆车需要等待 2.0916min，此即：

$$T_h = 2.0916min/辆$$

6.1.4 机场运营净收益计算结果

利用公式（1）：$P_{com} = EP_{km} - EC_{km} - P_w T_s + P_w T_h + S_h C_{km}$

在前面的部分中，已经对于各个变量及参数的大小进行了确定，利用 Python 软件进行编程求解，代码见于附录之中。可以得到出租车司机选择去机场接客时的净收益P_{com}随着前方排队出租车数量变化的关系如图 8 所示：

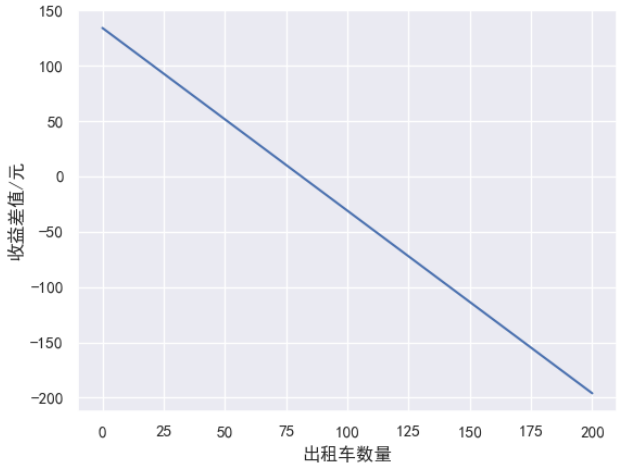

图 8 净收益随排队出租车数量变化关系图

6.2 模型的结果分析

利用上海市浦东国际机场 9 月 13 日上午 8:00-12:00 的数据进行求解得到的结果如图 8 所示，可以看出选择前往机场运营的净收益随机场随排队的出租车数量呈现线性递减关系。这是由于在问题一的分析中，我们采取了改进排队论的模型，对于每一辆出租车没有区分，等待时间为$T_h = 2.0916min/辆$。显然，随着出租车前方排队的车辆数不断增加，其等待时间成本也在随之增长，而这段时间如果选择在市区内运营时，可以取得一定的收益。因此，相较之下，前往机场运营的净效率会降低。

当前方排队的出租车数量$M \leq 80$时，$P_{com} > 0$，此时选择前往机场进行出租

车运营活动的效率更高；当$M > 80$，$P_{com} < 0$，此时应当选择在市区内运营。

6.3 模型的灵敏性分析

为了分析模型的合理性和对相关因素的依赖性，可以通过对于模型中某些变量添加适当的扰动，探究模型的灵敏性，借以判断相关因素的影响以及模型的合理性。

1.对于时间的依赖性：

对于一个星期当中不同的日期，在上海市内运营的出租车的单次载客时间不同，见于表 2，对应着空驶时间段也有着不同的变化。因此，空驶时间段的不同也会在一定程度上影响最后的净效益。

表 2 一周内单次载客时间的变化

日期	均值/分钟	标准差/分钟
周一	18.44	0.31
周二	18.93	0.57
周三	18.91	0.45
周四	19.35	0.88
周五	19.53	0.28
周六	17.76	0.46
周日	16.59	0.54

相应地，通过编程可以计算出一个星期中不同日期当净收益$P_{com} = 0$时，所对应的排队出租车数量，如图 9 所示：

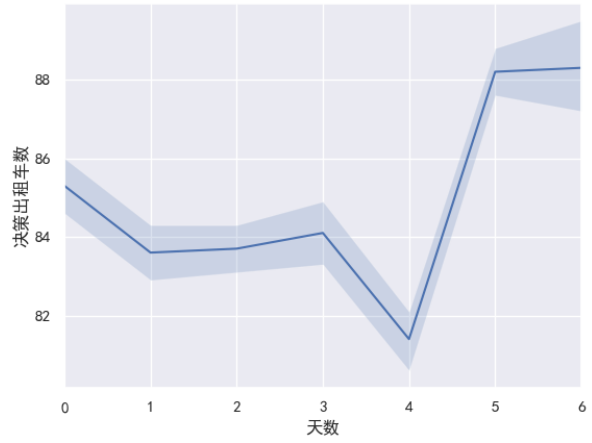

图 9 一周内决策出租车数量随日期变化图

从图中可以发现周末当净收益$P_{com} = 0$时，所对应的排队出租车数量相对会更多，意味着在周末的时候出租车司机更趋向前往机场接客。这是因为从表 2 中可以看出，在周末的时候在上海市内运营，出租车单次载客的时间更短，并且从

附录中的数据可以看出单次载客的路程也更短，空载比例更高，所以在市内运营的收益更低。这是因为在周末人们出行娱乐的人次更多，因此多在市内进行短途出行，并且更多的人次出行导致了交通更加拥堵，使得在市内运营收益大打折扣。

2.对于乘客打车意愿的依赖性：

对于机场运营净效益的影响还有一个很重要的因素是来自人们的打车意愿，包括两个部分：上海市内以及机场人们的打车意愿。显然，在单位区域内出行人数保持一定时，人们的打车意愿越高，出租车司机的运营效率更高。将决策的排队车辆数目与市区内及机场乘客的打车意愿建立三维关系图像，如图 10 所示：

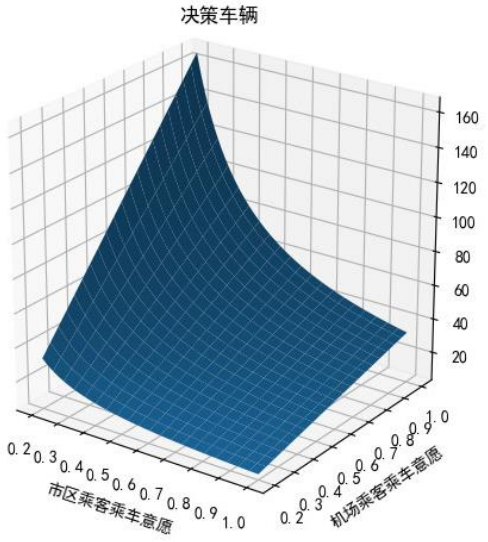

图 10 决策的排队车辆数目与打车意愿的三维关系图

显然，从三维图像中可以看出净收益与机场乘客的打车意愿呈正相关，与市区内乘客的打车意愿呈反相关。乘客的打车意愿受到多种因素的影响：包括航班的延误情况、出行的目的以及当天的天气情况等等。例如，在下雨天时发生了航班的延误，并且乘客有着紧急的事务需要处理时，其打车意愿也会越高。

7 问题三的模型的建立、求解与分析

7.1 基于元胞自动机的双排队模型的建立

7.1.1 元胞参数的规定

问题三中所研究的是双车道的排队问题[13]，因此将道路视为左右两条并列的、长度为L的一维离散格点直链[14]，如图 11 所示：

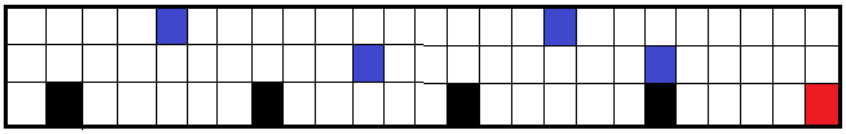

图 11 双车道元胞模型示意图

以上海浦东国际机场的候车道为例，其长度为$850m$，即：

（1） 道路长度：$L = 850m$；

（2） 元胞长度：对于正常出租车的长度约为$5m$，所以规定一个元胞的长度$l = 5m$，一辆出租车对应一个元胞；

（3） 元胞的数目：

$$N = \frac{L}{l} = 170$$

（4） 最大车速：

$$V_m = 1\ \text{元胞长度/秒} = 5\text{m/s} = 18\text{km/h}$$

（5） 模拟总时间：

$$T_{si} = 240min$$

7.1.2 道路安全性的规定

（1）速度限制

由于在这一路段上行人繁多，根据交通安全规章制度，在此双行车道上行驶的时速应当控制在 18km/h。

（2）车辆进站原则

车辆靠边停车接人必须遵循先来的车停在最前面的站点接人，紧随着的车辆在第二个站点的原则。否则，发生一辆出租车准备接人，另一辆出租车接完乘客准备离开的情况下极有可能发生碰撞。

（3）乘客顺次原则

默认乘客了解车辆进站原则后会在人数无差异的情况下选择靠前面的接口。

7.1.3 模型仿真原则

（1）按照发车概率在起始的元胞点产生不同速度的车辆[15]，进入车道占据不同的空格。

（2)在设定好的接口处元胞改为被占据的时间为:平均上车时间+等待乘客时间。

（3）实现仿真乘客到达的频率，在随机数产生后，对应接口的黑点就开始进行

倒计时（上车放行李），并且这个车辆会自动离站。

（4）每个接口最多只能有一辆车正在接站，其余没有接过客的车优先在另一条车道等等候。当整等候车道满员时，称之为饱和状态。

7.2 模型的求解与结果分析

利用 MATLAB 软件对于不同接口数量情况下进行模拟，可以得到在规定时间内出租车接客数量和接口数量之间的关系，并对数据点进行拟合，如图 12 所示：

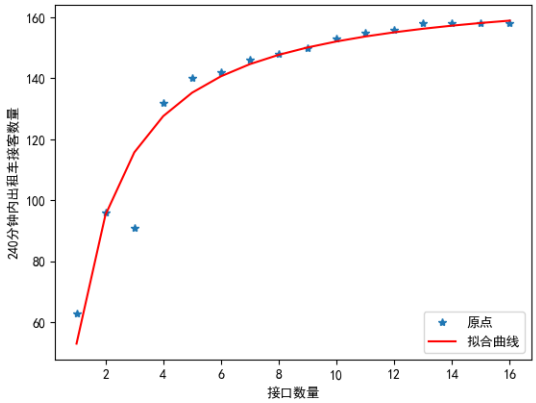

图 12 出租车接客数量随接口数量变化关系图

两者之间的拟合关系式为：

$$N = 169.01 * e^{-\frac{1.16}{x}}$$

式中，N表示接客数量，x表示车道的接口数量。

从表达式及上图中可以分析得到：在接口数$x = 7$之后，接客数随着接口数量的变化逐渐变缓慢；$x = 12$之后，接客数基本不再增长。所以在综合考虑建设成本的情况下，选择接口数$x = 7$为最优情形。

7.3 模型的灵敏性分析

对于模型进行检验，给出拟合曲线的相关检验参数，如表 3 所示：

表 3 拟合曲线检验参数表

参数	置信区间	SSE	MSE	RMSE	$\mathbf{R^2}$
a	[167.37, 171.23]	2.345	0.138	0.371	0.994
b	[-1.17, -1.15]				

其中，可决系数$\mathbf{R^2} = 0.994$，可以看出模型的求解取得了较好的效果。

显然，机场出租车的流入速度会对接口数量的设计产生一定的影响，因此，为了探究模型的灵敏性关系，我们对于出租车流入速度所服从的泊松关系中的指数λ进行变化，探究动态变化，如图 13 所示：

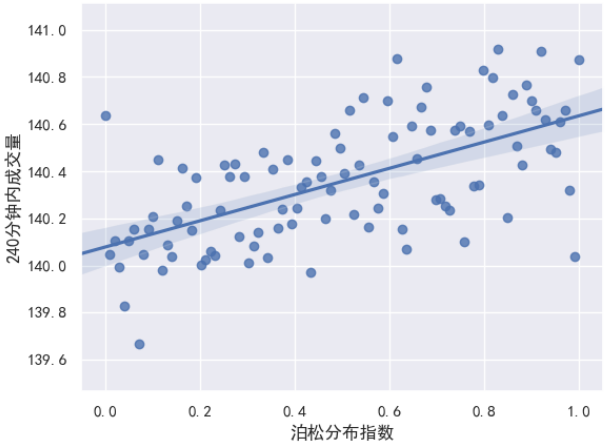

图 13 泊松分布指数对成交量影响

从图中可以看出，泊松分布指数λ的变化对于成交量的影响较小，所以模型的稳定性相对较高。

8 问题四的模型的解决

8.1 安排方案的确定

为了解决某些乘客的期望里程值较低时，对于机场出租车司机的运营效益会产生巨大影响的问题，我们提出了一套补偿方案来保障司机的运营效益：随着现代科技的发展，GPS 系统的完善。我们提出改进方案为：载客离开机场的司机自行用手机上的 GPS 对自己的行程进行记录，考虑如何根据等待时间t_w和载客行程s_l，补偿出租车司机一张短途票，使其免去排队。

8.2 方案合理性的分析

下面我们定量分析如何设定s_l更具有公平性。

我们先定义收益效率

$$\eta = \frac{P}{t_{总}}$$

查得上海浦东机场高峰期平均等待时间大概为$\bar{t_w} = 140\text{min}$，平均载客时间为$\bar{t_g} = 60\text{min}$。盈利$P = 134$ 元，代入计算出：

$$\bar{\eta} = 0.67\ 元/min$$

设一辆出租车满足了设定的补偿条件，那么出租车司机将再进行一次不需要

排队的载客，那么现在对于出租车司机的补偿后的收益效益η_{com}为

$$\eta_{com} = \frac{P_{low} + P_{com}}{t_l + t_{com}}$$

其中，P_{low}表示出租车司机进行该次短途运营所获得的利润，P_{com}表示需要补偿出租车司机此次运营的金额，t_l表示进行短途运营所花的时间，$t_{补}$表示进行补偿活动所花费时间。

进行一次载客的收益期望与之前计算的平均期望一样，所以

$$P_{com} = 134 \text{ 元}$$

同时司机进行的需要补贴的历程的时间分为载客t_{l1}和排队t_{l2}两部分

$$t_l = t_{l1} + t_{l2}$$

这次补偿载客的时间t_{com}还包含再返回到机场的时间t_{l1}和再载客时间$\bar{t}_y$，即

$$t_{com} = t_{l1} + \bar{t}_y$$

其中，P_{low}和t_{l1}都取决于载客里程数。

为了简化模型我们假设行程一定大于 3 公里，车速大于 12 公里/小时的情况。

$$P_{low} = \begin{cases} 14 + 2.5(s_l - 3) & 3 < s_l < 15 \\ 14 + 2.5(s_l - 3) + 3.6(s_l - 15) & s_l > 15 \end{cases}$$

$$t_{l1} = \frac{s_l}{\bar{v}}$$

因为我们的公平性目标是使得补贴后的收益效率η_{com}满足：

$$\eta_{com} = \bar{\eta}$$

在这个基础上我们得到s_l和t_{l2}的关系，如图 14 所示：

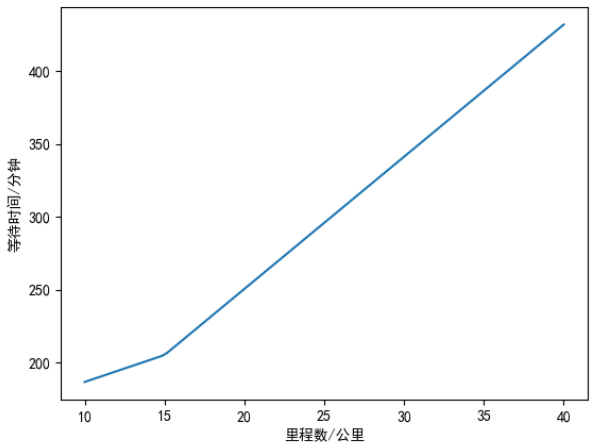

图 14 载客行程和等待时间函数关系图

在 GPS 记录出租车排队等待时间后，对应一个补偿的短途里程数，如果出租车的实际里程数小于等于这个值，那就可以给予出租车司机一张“补偿券”。

9 模型评价

9.1 模型的优点

1.问题一中采用的是改进排队论的模型对于出租车司机的运营决策进行分析。传统的排队论模型是用于解决乘客系统的等待时间，我们的模型中通过修正，将排队论的模型逆用求解出租车系统的等待时间成本，进而计算出出租车司机的运营净效益。将排队论模型进行改进后，可以运用其解决更多的实际问题，是本文的重要创新点之一。

2.问题二搜集的是上海市浦东国际机场的数据，具有典型的代表意义。将问题一建立的模型结合浦东国际机场的数据，使得模型的求解具有更重要的实际指导意义。此外，我们挖掘的是建模期间的实时数据，所以更具现实价值。

3.问题三同样对于排队论模型进行创新：采用双排队模型研究乘客和出租车辆两个排队系统，监测两个系统的动态变化影响。采用元胞自动机算法对于交通系统进行模拟，调整设置接口的数量，观察对于运营效率的影响，在全局中搜索最优解，使得求解结果更加精确。

4.问题四中考虑到前面三问中规定规则中的一些漏洞，提出一套补偿方案，保障了出租车司机运营的基本效益，使得整个模型的体系更加完整，考虑的方面更加全面。

9.2 模型的缺点

1.在第一问中未对多人乘载同一辆出租车的情况进行有效估计。

2.第二问中未能有效量化天气数据对应乘客流量的数据的影响。

9.3 模型的改进

1. 可以利用大数据，对机场出租车平均载客量，即多人载一辆车的情况的期望进行估计，使得计算结果更加精确。

2. 天气数据的寻找方法可以在网上先找到全年的下雨，高温，低温，雾霾等几种类型的天气的日期进行分类，然后在找对应日期的载客量，就可以收集到天气对客流量影响的数据。

10 参考文献

[1]熊笑. 基于梯度上升决策回归树的网约出租车需求动态预测[D].华中科技大学,2017.

[2]颜超. 上海市枢纽机场陆侧公共交通管理研究[D].华东师范大学,2015.

[3]胡稚鸿,董卫,曹流,高忠,陆志勇,吕俊,黄宏标,顾非凡.大型交通枢纽出租车智能匹配管理系统构建与实施[J].创新世界周刊,2019(07):90-95.

[4]何选森. 随机过程与排队论[M].湖南大学出版社,2010.

[5]赵云,张雷,张清,刘学川.基于排队论模型的机场安检流程优化设计[J].科技创新导报,2017,14(16):187-189.

[6]杨晓妍. 排队论在交通控制系统中的应用研究[D].青岛科技大学,2015.

[7]https://flights.ctrip.com/booking/airportpudong/jichangjianjie.html

[8]唐骏杰.上海浦东国际机场卫星厅消防系统监理要点[J].居舍,2019(24):10-11.

[9]龚建兵. 城市出租汽车政府价格规制研究[D].上海师范大学,2017.

[10]吕振华,吴健平,姚申君,朱丽.基于 FCD 的出租车运营特征分析——以上海市为例[J].华东师范大学学报(自然科学版),2017(03):133-144.

[11]Wonkyu Kim,Yonghwa Park,Byung Jong Kim. Estimating hourly variations in passenger volume at airports using dwelling time distributions[J]. Journal of Air Transport Management,2004,10(6).

[12]林思睿. 机场出租车运力需求预测技术研究[D].电子科技大学,2018.

[13]靳文舟,张杰,梅冬芳.基于细胞自动机模型的交通流模拟程序[J].华南理工大学学报(自然科学版),2003(05):34-38.

[14]宇仁德,李大龙.基于元胞自动机的交通流计算机程序[J].计算机仿真,2008(08):271-274.

[15]Kang Rui,Pan Weijun,Yang Kai. A Cellular Automation Model for Two-lane Traffic Flow Based on Turn Signal Effect[J]. Journal of Convergence Information Technology,2013,8(11).

附录一：第二问的 Python 源代码

```
1
import matplotlib.pyplot as plt
import numpy as np

# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
time=['07:29','07:32','07:35','07:40','07:43','07:44','07:47','07:50'
,'07:53','07:56','08:05','08:07','08:13','08:15',

'08:17','08:20','08:22','08:25','08:33','08:35','08:38','08:41','08:5
0','08:52','08:54','08:56','08:56','08:56',

'09:01','09:02','09:04','09:09','09:09','09:13','09:18','09:20','09:2
3','09:27','09:30','09:30','09:32','09:34',

'09:34','09:34','09:36','09:37','09:39','09:41','09:43','09:46','09:4
7','09:48','09:49','09:52','09:54','09:55',

'09:57','09:58','09:59','10:01','10:02','10:02','10:02','10:02','10:0
4','10:05','10:05','10:07','10:08','10:10',

'10:11','10:13','10:13','10:13','10:13','10:15','10:17','10:17','10:1
9','10:21','10:21','10:24','10:27','10:31',

'10:31','10:37','10:42','10:43','10:47','10:50','10:51','10:51','10:5
1','10:54','10:56','10:59','11:00','11:03',
```

```
'11:04','11:10','11:12','11:17','11:19','11:20','11:22','11:24','11:28','11:31',

'11:35','11:36','11:38','11:44','11:47','11:48','11:53','11:55','11:57']

come=[150,246,246,150,246,305,150,185,246,305,185,150,150,189,150,150,246,185,150,189,246,150,150,150,300,150,150,150,

361,270,300,185,350,361,130,189,189,185,185,150,150,185,189,189,189,246,130,189,185,189,124,185,185,185,150,189,

185,189,124,189,150,185,185,185,150,189,241,185,150,185,189,361,150,189,150,246,150,124,185,185,150,185,241,266,

189,150,246,241,189,189,101,236,101,236,236,150,150,335,150,150,236,246,150,236,236,246,200,150,246,220,236,246,
        150,150,234,189,246]

x=np.linspace(1,117,117)

plt.figure(figsize=(10,5))
plt.plot(x,come)
plt.title("9月13日上午8：00-12：00浦东机场到港客流量情况")
plt.xticks([1, 15, 29, 43, 57,71,85,99,113],

["07:29",'08:17','09:01','09:34','09:57','10:11','10:31','11:04','11:35'])
plt.show()
```

```
2
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
time=['07:29','07:32','07:35','07:40','07:43','07:44','07:47','07:50'
,'07:53','07:56','08:05','08:07','08:13','08:15',

'08:17','08:20','08:22','08:25','08:33','08:35','08:38','08:41','08:5
0','08:52','08:54','08:56','08:56','08:56',

'09:01','09:02','09:04','09:09','09:09','09:13','09:18','09:20','09:2
3','09:27','09:30','09:30','09:32','09:34',

'09:34','09:34','09:36','09:37','09:39','09:41','09:43','09:46','09:4
7','09:48','09:49','09:52','09:54','09:55',

'09:57','09:58','09:59','10:01','10:02','10:02','10:02','10:02','10:0
4','10:05','10:05','10:07','10:08','10:10',

'10:11','10:13','10:13','10:13','10:13','10:15','10:17','10:17','10:1
9','10:21','10:21','10:24','10:27','10:31',

'10:31','10:37','10:42','10:43','10:47','10:50','10:51','10:51','10:5
1','10:54','10:56','10:59','11:00','11:03',
```

'11:04','11:10','11:12','11:17','11:19','11:20','11:22','11:24','11:28','11:31',

'11:35','11:36','11:38','11:44','11:47','11:48','11:53','11:55','11:57']

come=[150,246,246,150,246,305,150,185,246,305,185,150,150,189,150,150,246,185,150,189,246,150,150,150,300,150,150,150,

361,270,300,185,350,361,130,189,189,185,185,150,150,185,189,189,189,246,130,189,185,189,124,185,185,185,150,189,

185,189,124,189,150,185,185,185,150,189,241,185,150,185,189,361,150,189,150,246,150,124,185,185,150,185,241,266,

189,150,246,241,189,189,101,236,101,236,236,150,150,335,150,150,236,246,150,236,236,246,200,150,246,220,236,246,
150,150,234,189,246]
sns.set_style("whitegrid")

绘制箱线图

# 竖着放的箱线图，也就是将 x 换成 y
ax = sns.violinplot(y=come)
plt.show()

3

```
import matplotlib.pyplot as plt

# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
name=["航班降落到靠桥","飞机开舱门","直接出站","到达行李区","等待行李","取行李","提取行李到出站"]
time1=[20,6,15,12,8,5,3]
time2=[15,4,12,9,5,1,1]
time3=[25,8,30,27,15,9,5]
x = list(range(len(time1)))
total_width, n = 0.8, 3
width = total_width / n

plt.bar(x, time1, width=width, label='平均时间', fc='y')
for i in range(len(x)):
    x[i] = x[i] + width
plt.bar(x, time2, width=width, label='最短时间', tick_label=name, fc='r')
for i in range(len(x)):
    x[i] = x[i] + width
plt.bar(x, time3, width=width, label='最长时间', fc='gray')
plt.legend()
plt.show()
```

4

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
import pandas as pd
from math import *
from scipy.stats import norm

# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
time=['07:29','07:32','07:35','07:40','07:43','07:44','07:47','07:50'
,'07:53','07:56','08:05','08:07','08:13','08:15',

'08:17','08:20','08:22','08:25','08:33','08:35','08:38','08:41','08:5
0','08:52','08:54','08:56','08:56','08:56',

'09:01','09:02','09:04','09:09','09:09','09:13','09:18','09:20','09:2
3','09:27','09:30','09:30','09:32','09:34',

'09:34','09:34','09:36','09:37','09:39','09:41','09:43','09:46','09:4
7','09:48','09:49','09:52','09:54','09:55',

'09:57','09:58','09:59','10:01','10:02','10:02','10:02','10:02','10:0
4','10:05','10:05','10:07','10:08','10:10',

'10:11','10:13','10:13','10:13','10:13','10:15','10:17','10:17','10:1
9','10:21','10:21','10:24','10:27','10:31',

'10:31','10:37','10:42','10:43','10:47','10:50','10:51','10:51','10:5
1','10:54','10:56','10:59','11:00','11:03',

'11:04','11:10','11:12','11:17','11:19','11:20','11:22','11:24','11:2
```

8',’11:31’,

’11:35’,’11:36’,’11:38’,’11:44’,’11:47’,’11:48’,’11:53’,’11:55’,’11:57’]

time=[29,32,35,40,43,44,47,50,53,56,65,67,73,75,77,80,82,85,93,95,98,101,110,112,114,116,116,116,121,122,124,129,129,

133,138,140,143,147,150,150,152,154,154,154,156,157,159,161,163,166,167,168,169,172,174,175,177,178,179,181,182,

182,182,182,184,185,185,187,188,190,191,193,193,193,193,195,197,197,199,201,201,204,207,211,211,217,222,223,227,

230,231,231,231,234,236,239,240,243,244,250,252,257,259,260,262,264,268,271,275,276,278,284,287,288,293,295,297]

come=[150,246,246,150,246,305,150,185,246,305,185,150,150,189,150,150,246,185,150,189,246,150,150,150,300,150,150,150,

361,270,300,185,350,361,130,189,189,185,185,150,150,185,189,189,189,246,130,189,185,189,124,185,185,185,150,189,

185,189,124,189,150,185,185,185,150,189,241,185,150,185,189,361,150,189,150,246,150,124,185,185,150,185,241,266,

189,150,246,241,189,189,101,236,101,236,236,150,150,335,150,150,236,246,150,236,236,246,200,150,246,220,236,246,

150,150,234,189,246]

```
'''
rs = np.random.RandomState(50)  # 设置随机数种子
s = pd.Series(rs.randn(100)*100)
no_fetch=np.random.normal(loc=41,scale=1.87,size=100)
fetch = np.random.normal(loc=54,scale=2.04,size=100)
all=[]
for i in range(100):
    all.append(no_fetch[i]*0.7+fetch[i]*0.3)
plt.figure(figsize=(8,4))
sns.distplot(all, bins=10, hist=True, kde=False, norm_hist=False,
             rug=False, vertical=False,label='平均概率密度',
             axlabel='x 轴',hist_kws={'color':'y','edgecolor':'k'},
             fit=norm)
# 用标准正态分布拟合
plt.legend()
plt.grid(linestyle='--')
plt.show()
'''
loc1=0.7*41+0.3*54
scale1=sqrt(1.87*1.87*0.7*0.7+2.04*2.04*0.3*0.3)

x = np.linspace(loc1 - 3 * scale1, loc1 + 3 * scale1, 50)
y_sig = np.exp(-(x - loc1) ** 2 / (2 * scale1 ** 2)) / (sqrt(2 * pi)
* scale1)

airport_time=np.linspace(1,350,350)
passenger_number=np.zeros(350)

for i in range(len(time)):
```

```
    for j in range(len(y_sig)):
        passenger_number[time[i]+j]+=y_sig[j]*come[i]*0.15/1.5

plt.plot(airport_time,  passenger_number,color="blue",label="机场等出租车乘客数量")
pd.DataFrame([passenger_number]).to_excel("passenger_number.xlsx")
plt.xlabel("时间/分钟")
plt.ylabel("乘客数量")
print(passenger_number)
plt.legend()
plt.show()
```

5

```
import numpy as np
import matplotlib.pyplot as plt
# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

distance=[8.63,8.68,8.71,8.74,9.03,8.29,8.28]
time=[18.44,18.93,18.91,19.35,19.53,17.76,16.59]
day=["周一","周二","周三","周四","周五","周六","周日"]
earning_low=[]
#空驶比率低时：
for i in range(7):
    earning_low.append(((distance[i]-3)*2.5+14)/(time[i]/0.85))
earning_high=[]
for i in range(7):
    earning_high.append(((distance[i]-3)*3.1+18)/(time[i]/0.3))
```

```
plt.figure()
print(earning_low[4])
plt.plot(day,earning_low,label="空驶率低时的收益")
plt.plot(day,earning_high,label="空驶率高时的收益")
plt.legend()
plt.show()
```

6

```
import numpy as np
import pandas as pd

passenger_number=pd.read_excel("passenger_number.xlsx")
# 每个人到达队伍的时间，得到 100 个值
arrival_time=[]
for i in range(120):
    for j in range(int(passenger_number[i+180])):
        arrival_time.append(i)
#arrival_time = np.random.uniform(0, 3, size=100)
# 达到时间要排序，先来后到进行排队
arrival_time.sort()
# 排队时间为 1~2 分钟，得到 100 个值
duration_time = np.random.uniform(1, 2, size=len(arrival_time))

# 每个人排队起始时间
start_time = [0 for i in range(len(arrival_time))]
# 每个人排队结束时间
end_time = [0 for i in range(len(arrival_time))]
# 每个人等待时间
wait_time = [0 for i in range(len(arrival_time))]
```

```
# 每个位置空闲时间
empty_time = [0 for i in range(len(arrival_time))]

# 位置数量
queue_count = 50
pre_person_index = len(arrival_time) - len(arrival_time) //
queue_count - 1
print(pre_person_index)

start_time[pre_person_index] = arrival_time[pre_person_index]
end_time[pre_person_index] = start_time[pre_person_index] +
duration_time[pre_person_index]
wait_time[pre_person_index] = start_time[pre_person_index] -
arrival_time[pre_person_index]


for i in range(pre_person_index + 1, len(arrival_time)):
    if end_time[i - 1] > arrival_time[i]:
        start_time[i] = end_time[i - 1]
    else:
        start_time[i] = arrival_time[i]
        empty_time[i] = start_time[i] - end_time[i - 1]

    end_time[i] = start_time[i] + duration_time[i]
    wait_time[i] = start_time[i] - arrival_time[i]
    print(wait_time[i])

print("每个人的平均等待时间: %f" % np.mean(wait_time))
```

```
7
import numpy as np

np.random.seed(22)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
'''
sns.set(color_codes=True)
mon = np.random.normal(loc=8.63,scale=0.27,size=10)
tue=np.random.normal(loc=8.68,scale=0.24,size=10)
wed=np.random.normal(loc=8.71,scale=0.18,size=10)
thu=np.random.normal(loc=8.74,scale=0.22,size=10)
fri=np.random.normal(loc=9.03,scale=0.24,size=10)
sat=np.random.normal(loc=8.29,scale=0.25,size=10)
sun=np.random.normal(loc=8.28,scale=0.28,size=10)
time=[18.44,18.93,18.91,19.35,19.53,17.76,16.59]
x = np.linspace(0, 15, 31)
'''
# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
'''
#data = np.sin(x) + np.random.rand(10, 31) + np.random.randn(10, 1)
data=[mon,tue,wed,thu,fri,sat,sun]
print(data)
ax = sns.tsplot(data=data)
plt.show()
```

```
e=40
pkm=3.35
ckm=0.65
pw=1.65
tw=2.0915
ty=60
th=60
sh=40
taxi_number=np.linspace(0,200,201)
pcom=[]
mon_taxi=[]
for i in range(len(mon)):
    pw1=pw/9.03*mon[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            mon_taxi.append(j-1)
            break
tue_taxi=[]
for i in range(len(tue)):
    pw1=pw/9.03*tue[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            tue_taxi.append(j-1)
            break
wed_taxi=[]
for i in range(len(wed)):
    pw1=pw/9.03*wed[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
```

```
            wed_taxi.append(j-1)
            break
thu_taxi=[]
for i in range(len(thu)):
    pw1=pw/9.03*thu[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            thu_taxi.append(j-1)
            break
fri_taxi=[]
for i in range(len(fri)):
    pw1=pw/9.03*fri[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            fri_taxi.append(j-1)
            break
sat_taxi=[]
for i in range(len(sat)):
    pw1=pw/9.03*sat[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            sat_taxi.append(j-1)
            break
sun_taxi=[]
for i in range(len(sun)):
    pw1=pw/9.03*sun[i]
    for j in range(201):
        if((e * (pkm - ckm) - pw1 * taxi_number[j] + 40 * 0.65)<0):
            sun_taxi.append(j-1)
```

```
            break
data=pd.DataFrame([mon_taxi,tue_taxi,wed_taxi,thu_taxi,fri_taxi,sat_t
axi,sun_taxi]).T
print(data)
ax = sns.tsplot(data=data.values)
plt.xlabel("天数")
plt.ylabel("决策出租车数")
plt.show()
'''
'''
plt.figure()
plt.plot(taxi_number,pcom)
plt.xlabel("出租车数量")
plt.ylabel("收益差值/元")
plt.show()
'''
#3d 图形必须的
from mpl_toolkits.mplot3d.axes3d import Axes3D


#计算 Z 轴的值
def mk_Z(X, Y):
    return 33.03/X*Y

#生成 X, Y, Z
x = np.linspace(0.20, 1.00, 100)
y = np.linspace(0.20, 1.00, 100)
X,Y = np.meshgrid(x, y)
Z = mk_Z(X, Y)
```

```
fig = plt.figure(figsize=(14,6))

#创建 3d 的视图，使用属性 projection
ax = fig.add_subplot(1, 2, 1, projection='3d')

ax.plot_surface(X,Y,Z,rstride = 5,cstride = 5)
'''
#创建 3d 视图，使用 colorbar，添加颜色柱
ax = fig.add_subplot(1, 2, 2, projection='3d')
p = ax.plot_wireframe(X, Y, Z, rstride=5, cstride=5,
                      )
'''
plt.xlabel("市区乘客乘车意愿")
plt.ylabel("机场乘客乘车意愿")
plt.title("决策车辆")
plt.show()
```

附录二：问题三的 Python 源代码

```
1
# 使用非线性最小二乘法拟合
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import numpy as np
# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
# 用指数形式来拟合
```

```
x = np.arange(1, 17, 1)
y = np.array([63, 96, 115.7420932237962, 132, 140, 142, 146, 148, 150,
153, 155, 156, 158, 158, 158, 158])
def func(x,a,b):
    return a*np.exp(b/x)
popt, pcov = curve_fit(func, x, y)
a=popt[0] # popt 里面是拟合系数，读者可以自己 help 其用法
b=popt[1]
yvals=func(x,a,b)
plot1=plt.plot(x, y, '*',label="原点")
plot2=plt.plot(x, yvals, 'r',label='拟合曲线')
plt.xlabel('接口数量')
plt.ylabel('240 分钟内出租车接客数量')
plt.legend(loc=4) # 指定 legend 的位置,读者可以自己 help 它的用法
plt.title('出租车接客数量随接口数量变化情况')
plt.show()


def getIndexes(y_predict, y_data):
    n = y_data.size
    # SSE 为和方差
    SSE = ((y_data - y_predict) ** 2).sum()
    # MSE 为均方差
    MSE = SSE / n
    # RMSE 为均方根,越接近 0，拟合效果越好
    RMSE = np.sqrt(MSE)

    # 求 R 方，0<=R<=1，越靠近 1,拟合效果越好
    u = y_data.mean()
```

```
    SST = ((y_data - u) ** 2).sum()
    SSR = SST - SSE
    R_square = SSR / SST
    return SSE, MSE, RMSE, R_square

indexes_1=getIndexes(y, yvals)
#   (1.547750647353422,   0.030955012947068438,   0.1759403675881929,
0.9361164947913276)
print(indexes_1)
```

附录三：问题三的 Matlab 源代码

```
1
clc;clear;close all;
S=ones(40,100);% state matrix
S(end,:)=0; % initial sttae
Ss=zeros(size(S)+[1,0]); % top line is origin of particle
Ss(2:end,:)=S; % showing matrix
N=size(S,2);
II=imagesc(Ss);axis equal;colormap(gray)
set(gcf,'DoubleBuffer','on');
while sum(1-S(1,:))<0.5;
    y=1;
    x=round(rand*[N-1]+1); % random position
    D=0;
    while D<0.5; % random travel
        r=rand;
        if abs(x-1)<0.1;
            SL=1;
        else
```

```
            SL=S(y,x-1);
        end
        if abs(x-N)<0.1;
            SR=1;
        else
            SR=S(y,x+1);
        end
        if SL+SR+S(y+1,x)<2.5; % check the neighbor: left, right, under
            D=1;
            S(y,x)=0; % stop in the position
        end
        if r<=1/3; % travel randomly
            x=x-1;
        elseif r<=2/3;
            x=x+1;
        else
            y=y+1;
        end
        Ss(2:end,:)=S;
        if x<0.5|x>N+0.5;
            D=1; % out of the range
        else
            Ss(y,x)=0; % to show the moving particle
        end
        set(II,'CData',Ss); % to show
        pause(0.1);
    end
end
```

附录四：问题四的 Python 源代码

1

```
import matplotlib.pyplot as plt
import numpy as np
# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
p_plus=134
t_drive=60
s_plus=np.linspace(10,40,100)
p_low=[]
t_low=s_plus/0.67
n_plus=0.67
for i in range(100):
    if(s_plus[i]<15):
        p_low.append(14+2.5*(s_plus[i]-3))
    else:
        p_low.append(14+2.5*(s_plus[i]-3)+3.6*(s_plus[i]-15))
t_low_wait=[]
for i in range(100):
    temp=(p_plus+p_low[i])/n_plus-2*t_low[i]-t_drive
    t_low_wait.append(temp)

plt.figure()
plt.plot(s_plus,t_low_wait)
plt.xlabel("里程数/公里")
plt.ylabel("等待时间/分钟")

plt.show()
```

2

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
sns.set(color_codes=True)

speed=np.linspace(0,1,100)
# 定义目标函数
def func(x, a, b):
    return a * np.linspace(0,1,100) +b

# 这部分生成样本点，对函数值加上高斯噪声作为样本点
# [0, 4]共 50 个点
# a=2.5, b=1.3, c=0.5
y = func(speed, 0.7, 140)
np.random.seed(10086)
err_stdev = 0.2
# 生成均值为 0，标准差为 err_stdev 为 0.2 的高斯噪声
y_noise = err_stdev * np.random.normal(size=speed.size)
ydata = y + y_noise
# 绘图时可以显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
ax = sns.regplot(x=speed, y=ydata)
plt.xlabel("泊松分布指数")
plt.ylabel("240 分钟内成交量")
plt.show()
```