机场出租车载人决策及短途优先模型

——以成都市双流机场为例

摘要

在日常生活里，出租车在我们的交通出行中扮演着很重要的角色。本文通过搭建出租车司机的机场决策模型并对其进行检验来提高出租车司机的决策效率。通过建模分析，探究如何改善机场设施以及规章制度，实现出租车司机在机场等待时间的缩短以及出租车司机盈利风险的降低；

针对问题一，本文首先构建了单位时间内盈利率指标——收益效率 w，通过对方案 A、B 的收益效率进行比较，帮助出租车司机做出更优的决策。与收益效率有关的变量有盈利变量和时间变量。将盈利变量简化为只与油耗和路程有关，而将时间变量分为行驶时间和等待时间两部分。根据现实情况，假定行驶时间为一个概率分布函数，通过设立期望行驶时间来求解方案 A 的期望收益。拟定时间、月份、蓄车池车辆数，队伍乘客数等因素作为影响司机在机场内等待时间的自变量，并设定了从机场到市区的路程、下机乘客选择乘坐出租车的概率等多个可通过客观数据收集到的变量。之后使用枚举和迭代方法构造模型，得到与自变量相对应的机场等待时间。从而得出在特定时间、特定“蓄车池”已有出租车数量下方案 A 的期望收益效率的表达式。为了方便比较，计算司机采用 B 方案在与 A 方案相同的时间段内的期望收益效率。而 B 方案的收益来源于提前回到市区拉客所享受的收益效率 w_0。将两个方按的期望收益效率比较公式进行移项，得出机场出租车司机的决策方程。这个方程与 w_0 有关，当决策方程右边的表达式大于 w_0 时，司机应选择方案 A；相等时，两种方案均可考虑；反之，选择方案 B。

针对问题二，我们通过 2014 年成都市出租车 GPS 数据、2016 年全国机场吞吐量数据以及 2014 年 11 月 27 日全国所有机场的航班信息，计算出了下机乘客选择乘坐出租车的概率。通过对市区内的 GPS 数据进行处理，计算出了从机场到市区的期望路程。将所有已知的数据代入第一问的模型中得出，当 “蓄车池”已有出租车数量为 22 辆、队伍乘客数为 5 人时，在 2016 年 8 月份的任意一天的早上 8：32 之前应该选择方案 A，早上 8：32 之后应该选择方案 B。最后我们选取“蓄车池”初始车辆数、每批次出租车进入“乘车区”耗时、每批次放行的出租车数量以及这三个因素作为自变量，对收益率进行依赖性分析，前两者的数值越大，选择方案 B 的性越大，最后一个变量的数值越大，选择 A 方案的合理性越大。

针对问题三，通过缩短在机场的出租车司机的等待时间来提升总乘车效率。在建立某批进入“乘车区”的车辆的移动方程的过程中发现“上车点”的位置由每批进入“乘车区”的车辆数量 α 决定。而结合问题一中的模型，发现等待时间也可以由这个变量的函数表示。找寻出最佳的 α 值使等待时间最短从而就可求出“上车点”的最佳位置。

针对问题四，给予某个时间段内返回机场的出租车一定的“优先权”，即行车时间在给定范围内的出租车可以进入“优先蓄车池”中。之后建立了两个收益指标：一个是第一趟搭载了长途乘客的出租车司机在一段时间内的收益率，另一个是第一趟搭载了短途乘客的司机完成生意后返回并享受优先权且第二趟搭载乘客的综合收益率。两个收益指标都是关于行车时间的函数，计算两个函数的交点，即一个合理的行车时间范围。最后通过计算，在拟定条件下，我们得出优惠策略应该设定为：在 16 分钟内返回机场的出租车可以享受优先权，这个条件下两个收益率数值相等，且都为 0.6789 元/分钟。

关键词：枚举法 迭代法 依赖性分析 地理位置分析

1 问题重述

1.1 问题的背景

出租车是大多数乘客下飞机后去往其他目的地选择的交通工具之一。对于送客到机场的出租车司机而言，一般都面临着直接在机场载客或回市区拉客的选择。每个出租车司机必然会根据可观测信息，如某时间抵达的航班数量和机场的出租车数量等以及相关因素并结合经验来做出决策。但仅仅通过经验来主观决断可能并不能保证在机场的出租车司机每次都能做出最优的决策。如果将司机的决策过程量化，以客观数据来为司机定制决策方案，在一定的程度上能使司机做出正确选择的概率得到提升。而如何选择和利用准确的相关数据有待研究。而从机场的角度来看，如何根据机场的道路情况设置上车点提升乘车效率、合理制定出租车等待方案来均衡出租车收益以吸引更多的出租车也是有待研究的问题。

1.2 需解决的问题

(1) 第一问需要根据机场乘客打车数量变化规律和出租车司机的收益以及其他相关因素，建立出司机的选择策略模型，并分析不同条件下司机的选择策略。

(2) 第二问需要收集某机场及其所在城市的相关现实数据，并代入第一问的模型中，求解出该机场出租车司机的选择方案。根据现实情况分析模型的合理性，同时通过对比各因素的系数分析出决策模型对相关因素的依赖性。

(3) 第三问需要求解在“乘车区”拥有两条并行车道的情况下，“上车点”设置的最佳位置，使得总的乘车效率最高。

(4) 第四问需要拟定一个“优先”安排方案，增加短途载客再次返回的出租车的收益，使得选择在机场载客的出租车收益尽量均衡。

2 问题分析

2.1 问题一的分析

在问题一中，司机既可以选择花费一定的时间成本在飞机场载客并将乘客群体送回市区，也可以选择损失一定的空载费用提前回到市区拉客盈利，为了衡量两种方案的优劣，我们建立单位时间盈利数目这个指标，以该指标的大小来代表方案的优劣性。与单位时间盈利数目相关的变量有盈利和时间，在第一问中，我们简单的将盈利和时间看作一个变量，实际上会有很多实际因素影响这两个变量的大小，针对实际因素的探讨我们放在了第二问中。

两个变量中的时间变量可以分为两个部分求解，一个是司机行驶所花的时间，另一个是司机在机场等待的时间，等待时间则和“蓄车池”中的车辆数，当前的航班数量，当下的月份以及时间都有着密切的关系；我们在 wind 金融终端收集了多个机场的多年所有月份的吞吐量[2]和 2016 年 11 月 27 日全国各地机场的每一趟航班的详细数据[3]，从中得出了月度层面和分钟层面的规律并将两者整合，最后建立模型。将各个因素联系起来，帮助司机做出理性决策。

盈利变量在第一问中被我们简化为一个只与油耗和路程有关的变量，在第二问中我们会联系实际增加出租车收费制度等客观因素，在此不再赘述。

2.2 问题二的分析

观察第一问的决策模型，可以发现飞机场的出租车司机做出决策取决于等待时间、出租车司机在市区各时段能享受的收益效率以及汽车的计价方式和油费这几个变量。汽车的计价方式和油费在各个时间段比较固定。等待时间由“蓄车池”中车辆数目、“乘车区”每次放行车辆数目、每批出租车进入“乘车区”耗时、“乘车区”乘客群体数、

相应时间段客流量[4]这几个因素共同决定。通过查找客观数据可以拟合出决策模型右端表达式的曲线。而出租车司机在市区各时段能享受的收益效率也可以通过收集数据拟合出一条市区收益效率曲线。将这两条曲线放在同一坐标轴内做出比较，即可分析出在机场的出租车司机在相应客观条件下更优做出的决策。

而通过观察两条拟合曲线的取值范围、相交状况、异常取值数量和与客观现实相符合的程度，即可在一定程度上证明模型的合理性。由于决策方程左端的市区收益效率可以通过客观数据大致确定，因此通过控制变量法找寻关键变量的变化对决策方程右端表达式数值的影响即可分析决策方程对相关因素的依赖性。

2.3 问题三的分析

设置“上车点”使得总的乘车效率最高，结合问题一建立的模型，就是缩短等待时间。而等待时间这个变量主要被管理人员分批次放行的标准出租车数量与每批出租车进入“乘车区”耗时这两个因素影响。通过对出租车在“乘车区”的移动过程分析可以建立上述两个因素之间的关系模型。

在建模过程中发现“上车点”的位置由管理人员分批次放行的标准出租车数量决定。因此可以将问题三转化为求解管理人员分批次放行的最优标准出租车数量，使得等待时间最短，从而求解出“上车点”的最佳位置。

2.4 问题四的分析

问题四的要求是制定一个“优先”安排方案来增加短途载客再次返回的出租车司机的收益。如果另开一条连接到“乘车区”的“优先蓄车池”，只供短途载客再次返回的出租车使用，缩小该类出租车的二次等待时间，就可以改善其两趟载客的总收益效率。而关键点就是定义出租车短途载客行驶时间的临界值，当某出租车司机再次返回机场与上一次间隔的时间小于该临界值的两倍时，即可选择进入“优先蓄车池”候客，享受“优先权”。

建立短途载客出租车司机两趟载客的总收益效率与临界值的关系模型，使得总收益效率与期望收益效率相等的临界值就是最优临界值。将客观数据代入模型即可拟合出一条总收益效率曲线曲线。通过前几问的分析可知选择在机场接客的出租车司机获得的收益效率是一个概率分布函数，而具体的期望收益效率数值也可以由客观数据求解。如果在“优先”安排方案下总收益效率曲线有与期望收益效率相等或相近的数值点，即可说明该方案具有一定的可行性。

3 模型假设

(1) 假设出租车匀速驾驶。但在高速路和市区内的驾驶速度不同。
(2) 假设出租车在市区范围内就会享受一定的收益效率。
(3) 假设每辆出租车只载一个乘客群体。
(4) 假设在机场的出租车司机根据航班数即可确定大概的客流量。
(5) 假设每批放行的出租车进入“乘车区”时以相同的加速度同时启动，并且达到最高限速时保持原速。

4 符号说明

变量符号	变量含义	变量单位
Δt	每批出租车进入“乘车区”耗时	*min*
m	队伍群体数	个
α	管理人员分批次放行的标准出租车数量	辆
T_0	该出租车司机进入“蓄车池”的时刻	*min*
T'	队伍人员数量与T_0时刻“蓄车池”中车辆数目相等的时刻	*min*
n	T时刻到达的航班数量	架
q	“蓄车池”中初始状态T_0已有的车辆数目	辆
d	出租车行驶距离	公里
$W=g(d)$	出租车收益	元
t_1	出租车司机在机场载客等待的时间	*min*
t_2	机场到目的地所需要的时间	*min*
t_0	机场到市区所需要的时间	*min*
w	收益效率	元/*min*
w_0	出租车在市区范围内享受的收益效率	元/*min*
k	油耗成本与路程的比值	元/*km*
$N(T_x)$	T_x时刻机场管理人员放行的出租车数量	辆
$h(T_x)$	T_x时刻“乘车区”新增加的群体数量	个
p	下飞机的乘客选择出租车作为交通工具的概率	-
v_0	出租车在高速路上的驾驶速度	*km/h*
v_1	出租车在市区内的驾驶速度	*km/h*

$t_{1,short}$	出租车进入“优先蓄车池”后的等待时间	min
$t_{2,short}$	出租车短途载客行驶的时间	min
Y	乘车区的长度	m
L	辆出租车的车身长度	m
D	两辆出租车之间的安全距离	m
t_c	出租车在“乘车区”的行驶时间	min
t_p	所有乘客都已走到车门处的时间	min
t_p'	乘客放置行李、上车的时间	min
v_c	出租车在“乘车区”的限速	m/min
a	出租车在“乘车区”的启动加速度	m/min^2
v_p	乘客行走的时间	min
x_1	车道 1 的上车点位置	m
x_2	车道 2 的上车点位置	m

5 模型的建立与求解

5.1 出租车司机选择决策模型

5.1.1 模型的建立

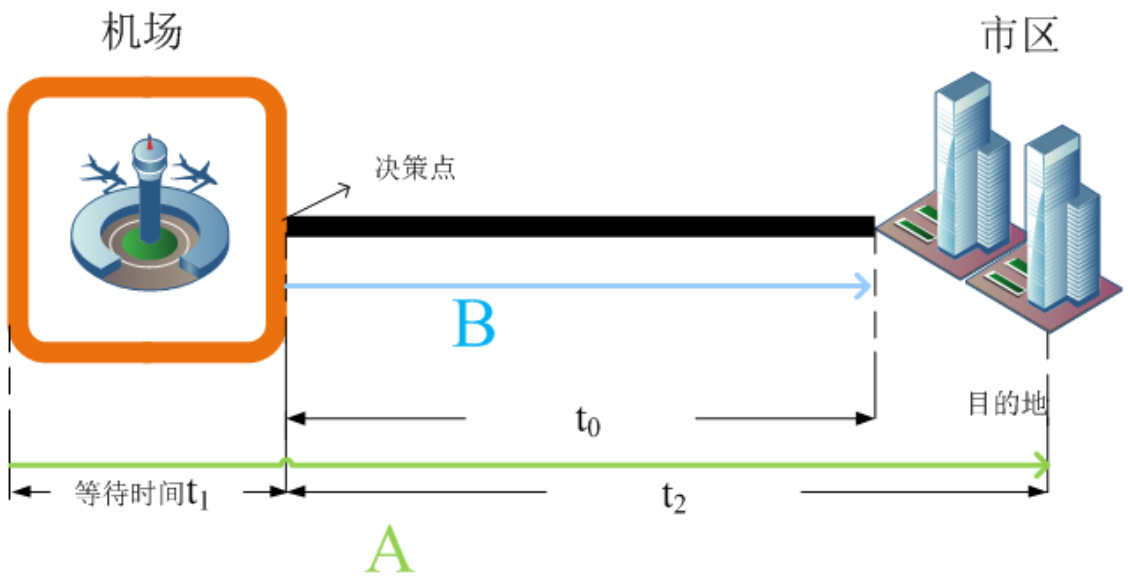

图 5-1 决策方案耗时示意图

如图 5-1 所示，当出租车司机将乘客送到机场，即到决策点时，将面临 A：直接在机场载客与 B：回市区拉客两个选择。如果司机选择 A 方案，那么当他返回市区并到达目的地所花费的时间为(t_1+t_2)，并且支出 t_2 时间段内的油费，但可获得载客收益。在这段时间内的收益效率为：

$$w_A = \frac{g(d_2) - kd_2}{t_1 + t_2} \tag{1}$$

$$d_2 = v_0 t_0 + v_1 (t_2 - t_0) \tag{2}$$

如果司机选择 B 方案，那么他回到市区的最短时间为 t_0，会付出空载费用并可能损失潜在的载客收益，但能最快享受到在市区范围内的收益效率。为了方便比较，计算司机采用 B 方案在与 A 方案相同的时间段(t_1+t_2)内的收益效率：

$$w_B = \frac{w_0(t_1 + t_2 - t_0) - kd_0}{t_1 + t_2} \tag{3}$$

$$d_0 = v_0 t_0 \tag{4}$$

根据现实情况可知 t_2 服从某种概率分布，如果方案 A 的期望收益效率大于 B，那么出租车司机选择方案 A 更优的概率较大，即：

$$E(w_A) \geq E(w_B) \tag{5}$$

$$\frac{g\left[E\left(d_2\right)\right] - kE(d_2)}{t_1 + E(t_2)} \geq \frac{w_0\left[t_1 + E\left(t_2\right) - t_0\right] - kd_0}{t_1 + E\left(t_2\right)} \tag{6}$$

5.1.2 模型的求解

由公式(4)可推出：

$$w_0 \le \frac{g\left[E\left(d_2\right)\right]-k\left[E(d_2)-d_0\right]}{t_1+E(t_2)-t_0} \tag{7}$$

其中出租车收益W、d、t与k都可以通过现实数据推导。

出租车司机在机场载客等待的时间 t_1 取决于排队出租车和乘客的数量。“蓄车池”中的出租车数量是可观测的，而乘客的数量变化规律与到达的航班数量变化规律有关，即：

$$t_1 = f\left(q,n,T,Month\right) \tag{8}$$

若$m_{T_0} \le q+1$，假设经过“乘车区”的队伍人群总量与T_0时刻经过“蓄车池”的车辆总数相等距初始时刻T_0需经过t_A：

$$t_A = T'-T_0 \tag{9}$$

若$m_{T_0} > q+1$，则令$t_A = 0$。

以下首先考虑$t_A \neq 0$的情况：

在T'时刻时，队伍人群总量与T_0时刻“蓄车池”中车辆数目相等，即：

$$m_{T_0} + \int_{T_0}^{T'} h(T)dT \times p = q+1 \tag{10}$$

由客观数据可知$h(T)$的方程，代入方程(10)可解出T'。

根据现实情况，可知某时刻的队伍群体数是上个时刻的队伍群体数加这一间隔时间段新增的群体数再减去已乘车的群体数，用数学表达式列出来为：

$$\begin{cases} m_{T_1} = m_{T_0} + h(T_0)\times\Delta t \times p - N(T_0) \\ m_{T_2} = m_{T_1} + h(T_1)\times\Delta t \times p - N(T_1) \\ \vdots \qquad\qquad \ddots \\ m_{T'} = m_{T'-1} + h(T'-1)\times\Delta t \times p - N(T'-1) \end{cases} \tag{11}$$

其中，m_{T_0}设为已知量（后续用穷举法），$h(T_0)\times\Delta t$与p都可由客观数据得出。

易知每时刻放行的出租车量与“乘车区”的人数有关：

$$N(T_x) = \min(\alpha, m_{T_x}) \tag{12}$$

结合公式(11)和公式(12)可求各个时间段的m_{T_x}与$N(T_x)$。

假设距初始时刻T_0总共进入“乘车区”的出租车数量为N_{total}，即：

$$N_{total} = N(T_0) + N(T_1) + \cdots + N(T'-1) \tag{13}$$

结合公式(10)、公式(11)和公式(13)可推得

$$m_{T'} = q + 1 - N_{total} \tag{14}$$

当$t_A = 0$时，

$$m_{T'} = q + 1 \tag{15}$$

假设T'时刻后，做决策的出租车还需等待t_B：

$$t_B = \left\lceil \frac{m_{T'}}{\alpha} \right\rceil \Delta t \tag{16}$$

结合公式(9)与公式(16)可以推得

$$t_1 = t_A + t_B = T' - T_0 + \left\lceil \frac{m_{T'}}{\alpha} \right\rceil \Delta t \tag{17}$$

将公式(17)的结果代入公式(7),得出机场出租车司机的决策方程：

$$\begin{cases} w_0 \le \dfrac{g\left[E\left(d_2\right)\right] - k\left[E(d_2) - d_0\right]}{T' - T_0 + \left\lceil \dfrac{q + 1 - N_{total}}{\alpha} \right\rceil \Delta t + E(t_2) - t_0}, m_{T_0} \le q + 1 \\ w_0 \le \dfrac{g\left[E\left(d_2\right)\right] - k\left[E(d_2) - d_0\right]}{\left\lceil \dfrac{q + 1}{\alpha} \right\rceil \Delta t + E(t_2) - t_0}, m_{T_0} > q + 1 \end{cases} \tag{18}$$

5.1.3 选择策略分析

由公式(18)可知，当出租车在市区范围内享受的平均收益效率w_0小于公式(18)右侧的数值时，飞机场的出租车司机选择 A 方案更为合理。当等式成立时，两种方案均可考虑。反之，选择 B 方案更为合理。

5.2 成都双流机场出租车司机决策分析

5.2.1 数值收集与计算

第二问解题的重点在于收集准确的相关数据并将客观数据代入第一问所建立的决策模型中，从而帮助在相应机场的司机做出选择。并考虑是否会出现异常例如所求结果不符合现实情况等。之后分析出相关因素对司机做决策的影响程度。

收集到的相关数据有：成都市双流机场 2013 年到 2018 年每个月份的月吞吐量、2016 年 11 月 27 日全国所有航班的起落信息、成都市 2014 年 8 月 15 日的所有出租车各个时间点的经纬度坐标、成都市每日出租车承担的客流量、成都市的出租车计价标准，以及成都出租车的百公里油耗和油价。

成都市的出租车计价标准采用网站上收集的排气量 1.6L 的出租车在白天的计价标准[2]，即：

$$g(d_2)=\begin{cases}8, d_2 \le 2\\ 1.9(d_2-2)+8, d_2 > 2\end{cases} \tag{19}$$

油耗成本与路程的比值 k 为百公里油耗与油价的乘积。百公里油耗根据客观现实大致定为 8L/100km，油价在 wind 数据库上找寻 2016 年 8 月份的油价[4]取平均值为 6.2825 元/L。

首先统计 2016 年 11 月 27 日全国所有航班的起落信息，得出当日吞吐量排名前十的机场。之后对每个时刻的吞吐量加权平均得到每个时刻的一般吞吐量。最后综合成都双流机场的规模和该机场每个月份吞吐量的大小，可以得出成都双流机场 2016 年各个月份中任意一天的吞吐量规律。

我们在此次模拟中假设 m_{T_0} =5， q =22；

由第一问模型可知，当 B 方案的期望单位时间盈利率小于 A 方案的期望单位时间盈利率时，出租车司机选择 A 方案更好，即留在机场排队等待降落乘客。即公式(5)与公式(7)。

其中出租车在市区范围内享受的平均收益效率，可由论文中的数据推导得出，具体数据表 5-1 所示[5]。

表 5-1 成都市出租车信息表

时间段	载客率	平均载客距离 (km/h)	盈利状况 (km/h)
6：00-6:30	0.36585	4.82068619	13.3593
6：30-7:00	0.38715	5.479803333	14.61163
7：00-7:30	0.4276	6.54062	16.62718
7：30-8:00	0.4145	6.89696	17.30422
8：00-8:30	0.4241	7.17658	17.8355
⋱	⋱	⋱	⋱
19：30-20:00	0.8299	9.834166667	22.88492
20：00-20:30	0.8139	8.95335	21.21137
20：30-21:00	0.7947	8.850586667	21.01611
21：00-21:30	0.7809	8.967516667	21.23828
21：30-22:00	0.7573	9.304683333	21.8789
22：00-22:30	0.720467	9.198546667	21.67724
22：30-23:00	0.6895	9.056036667	21.40647
23：00-23:30	0.665033	9.405893333	22.0712
23：30-24:00	0.6221	9.1508	21.58652

每个时间段的盈利状况乘以载客率再除以每个时间段的时长（30 分钟）就得到了每个时间段的市区范围内出租车收益效率，可知其是关于时间段的函数，如图 5-2 所示。

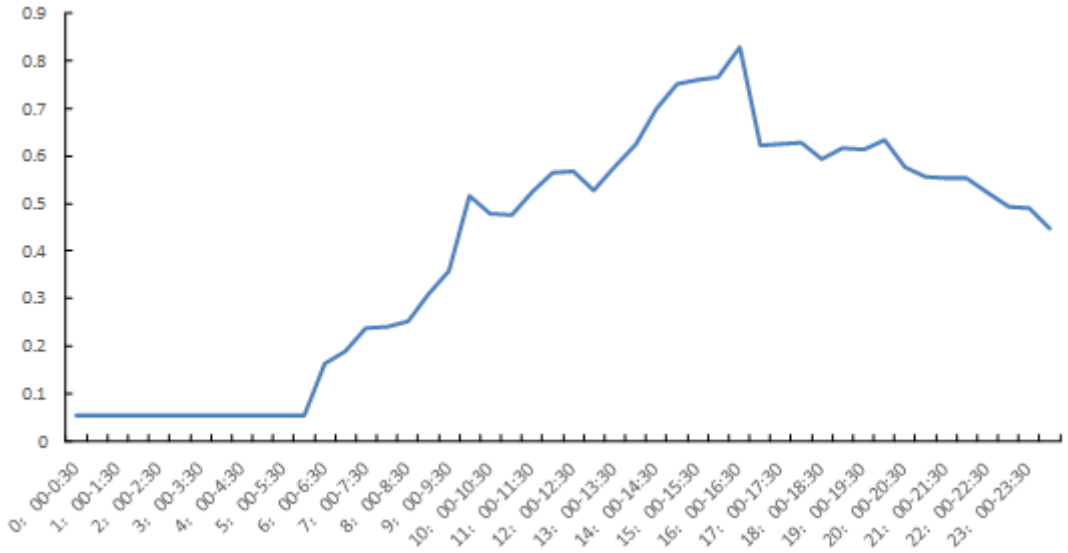

图 5-2 出租车在市区范围内的收益效率 w_0 数值图

同样的，公式(18)右端也是一个关于时间的表达式，这个表达式中的其他变量都可以通过现有数据求得：

市区内期望收益，可综合租车司机在市区入口位置到市区其它各个位置的期望距离与成都市的出租车计价标准计算。期望距离可由以下步骤得出：

首先从收集到的 59,209,838 条成都市出租车 GPS 数据中随机抽取 600,000 条，用核密度估计画出六边形密度图：

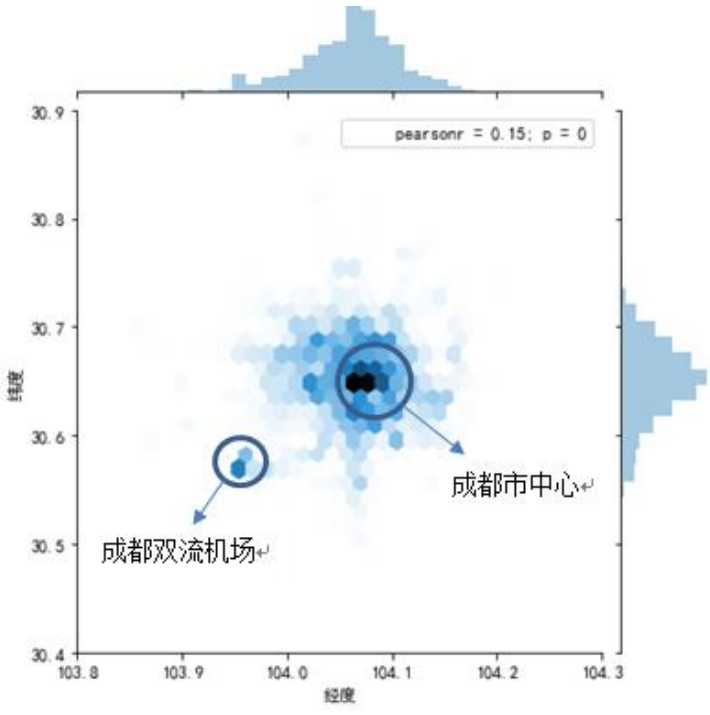

图 5-3 成都市出租车部分 GPS 数据六边形密度图

然后从 600,000 条数据中随机抽取 3000 条在地图上标注：

图 5-4 成都市出租车部分 GPS 数据地图标记图

由图 5-3 和图 5-4 可以看出，在市区内，出租车密度从市中心向郊区成发散状，为了简化模型，将三环内的区域作为市区区域，并将其等分为 16 个长方形区域，标号为 1~16。如图 5-5 所示。

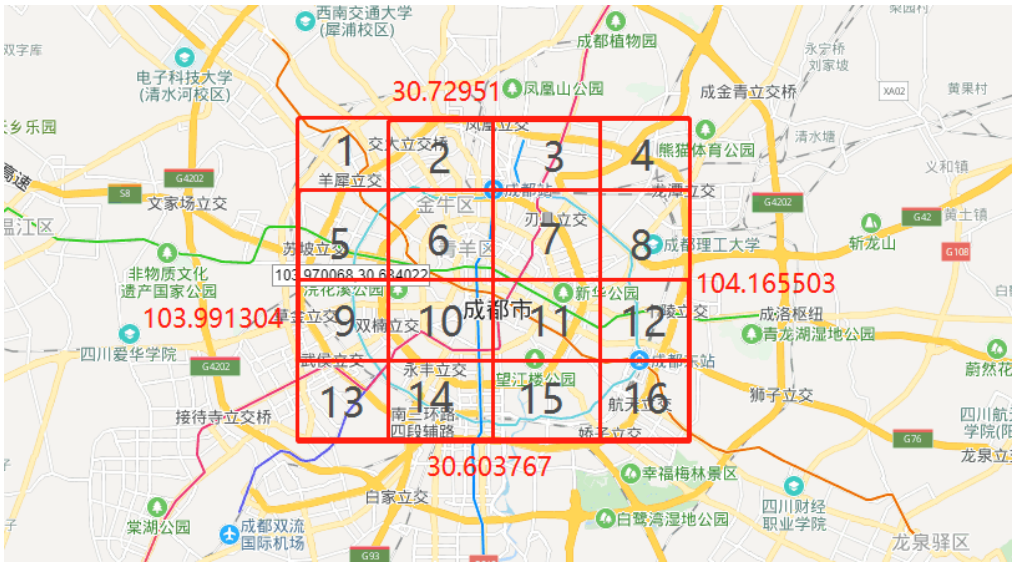

图 5-5 成都市区域划分图

继续使用前面的 600,000 个样本，计算每个矩形区域内的样本数量，以此计算出每个矩形区域的样本占比，如图 5-6 所示。

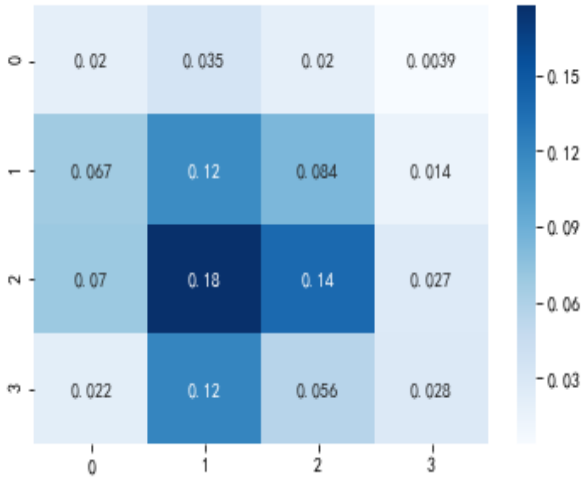

图 5-6 区域样本占比图

再分别计算每个矩形区域的中心点的经纬度，使用散点图表示：

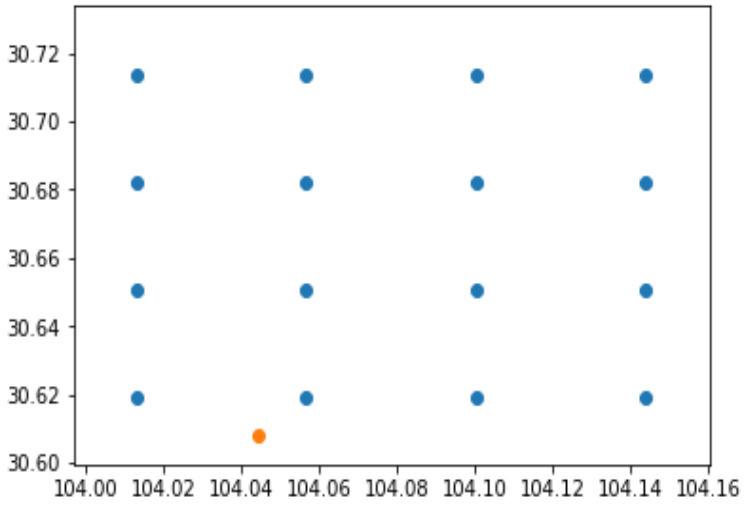

图 5-7 区域中心点经纬度图

图 5-7 中蓝色点为每个矩形区域的中心点，橙色点为市区入口(即机场高速出口)。计算橙色点到所有蓝色点的距离(km)：

表 5-2 各区域中心与市区入口距离表

12.10236	11.78953	12.89808	15.11852
8.76706	8.32975	9.83672	12.60914
5.61672	4.90597	7.17313	10.66317
3.24430	1.74471	5.51707	9.62848

用上表所示距离分别乘以对应样本占比，可得到在市区内行驶的期望距离：6.9347811820369305km。

而通过成都双流机场与市区路口的经纬度信息可求得机场到市区入口的路程 d_0=8.702km。由此可以算出从机场到目的地的期望距离 $E(d_2)$约为 15.637km。

根据现实状况，拟定出租车在高速路上的驾驶速度 v_0 为 100km/h。出租车在市区内

的驾驶速度 v_1 为 70km/h。

将上述数值代入公式(2)和公式(4)即可求出 $E(t_2)$与 t_0。

接下来求出等待时间 t_1 的具体值，由决策模型可知，当蓄车池数目大于队伍长度时，即决策模型的第一种情况：

先求出 t_A：由公式(10)可以求得T'，其中概率 p 为下飞机的旅客选择出租车出行的概率，p 的计算过程如下：

(1) 双流机场中出租车数量占成都市出租车数量的比例：

上文提到，我们一共收集到 59,209,838 条成都市出租车 GPS 数据。为了计算机场中出租车占成都市的比例，我们需要计算这约六千万条数据中有多少出现在机场。计算机场中的 GPS 数据点数量，需要先规定机场的范围：我们用下图中两个红色矩形近似表示机场的两个航站楼的范围，如下图所示：

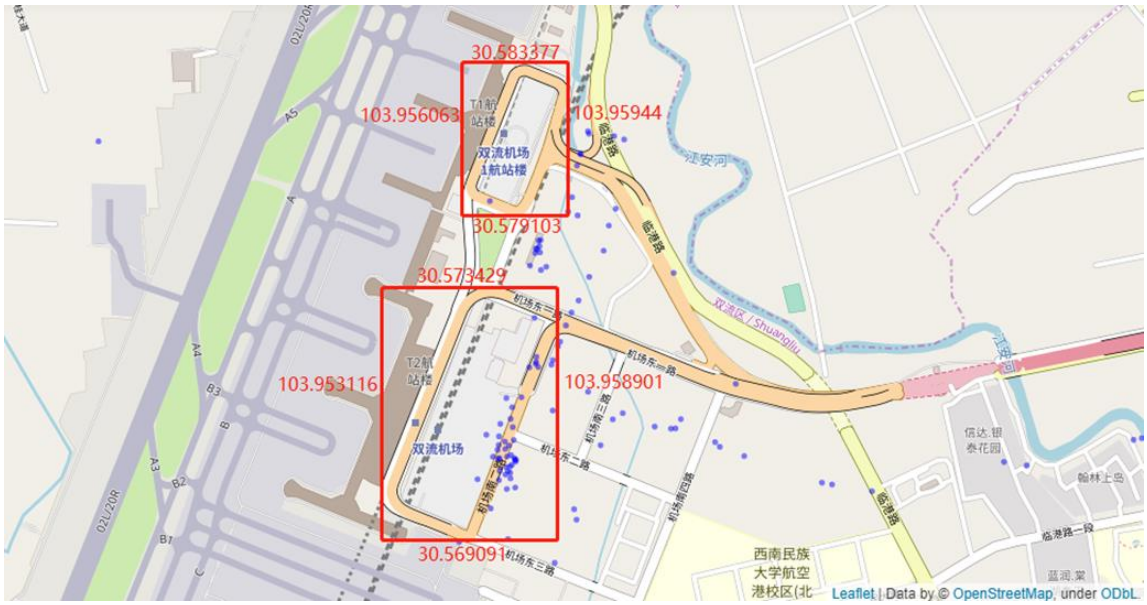

图 5-8 成都双流机场航站楼范围示意图

其中 T1 航站楼的经纬度范围为：经度[103.956063,103.95944]，纬度[30.579103,30.583377]

T2 航站楼的经纬度范围为：经度[103.953116, 103.958901]，纬度[30.569091, 30.573429]

计算两个航站楼里 GPS 数据的数量为 498,148

所以机场中出租车在成都市出租车中的比例为：498148÷59,209,838≈0.0084133

(2) 每一天城市中出租车承担的客流量：

假设时间处于 2016 年的 8 月份，当下时刻为晚上八点钟，司机可知此时五分钟内的航班数量约为三到四架，下机乘客约为 760 人，根据论文——《供需平衡状态下的出租车发展规模研究》[5]中的数据我们可以得出成都市的市民出行情况：

表 5-3 成都市居民出行情况表

年份	人口（万人）	居民日均出行次数（万人次）	出租车分担率
2012	586.256	1437	6.4%
2015	652.243	1697	6.2%

以 2015 年的数据为例，可以求解出出租车分担的日均客流量为 105.214 万人次，我们从数据竞赛网站上下载了 2014 年 8 月 5 日到 8 月 13 日期间，成都市每一辆出租车每半分钟在成都市内的经纬度坐标，并使用 Python 从数据集中随机抽取了 100 辆车的 60 万条经纬度坐标数据。同时在百度地图上可视化标识出来，如图 5-9 所示。600000 条 GPS 位置中(约 150 辆车)，有 8273 条(约 1.862 辆)在机场里，我们可视化其中的 3000 条，如图 5-10 所示。

图 5-9 成都市 100 辆出租车样本的经纬度坐标数据图

图 5-10 成都市双流机场 GPS 样本数据位置图

用 8273 条定位在机场的数据量除以 60 万条经纬度坐标的数据量，可以得出成都有 1.378%的出租车客流量来源于双流机场。

(3) 结合两个比例：

1.378%乘以出租车分担日均客流量 105.214 万人次，可得成都双流机场出租车日均客流量约为 14498（人次/日），根据成都民航规律得出的日均客流量 146881（人次/日），我们可以得出，到达成都双流机场的乘客约有 14498/146881=9.87%选择乘坐出租车。即 p=9.87%；

公式(10)中的积分部分可以根据成都市双流机场精确到每分钟的吞吐量规律累加得到，至此，将上述所求相应变量数据代入公式(10)即可求得T'，从而求得 t_A。

接下来求t_B：

利用 python 对迭代过程公式(11)进行运算，将每一次迭代得出的T_x时刻机场管理人员放行的出租车数量相加得到出租车的总放行数量，如公式(13)所示。

然后再将所得到的数值代入公式(14)以及公式(16)求得 t_B;

最后将 t_A 与 t_B 相加得到等待时间 t_1。

当蓄车池数目小于队伍长度时，即决策模型的第二种情况：只需将第一种情况求解的对应数值代入第二种情况下的决策模型，即可求解。

5.2.2 决策分析与合理性分析

由上一部分的数值计算，我们可以将公式(18)不等式左右两端表达式的图像拟合出来，如下图所示：

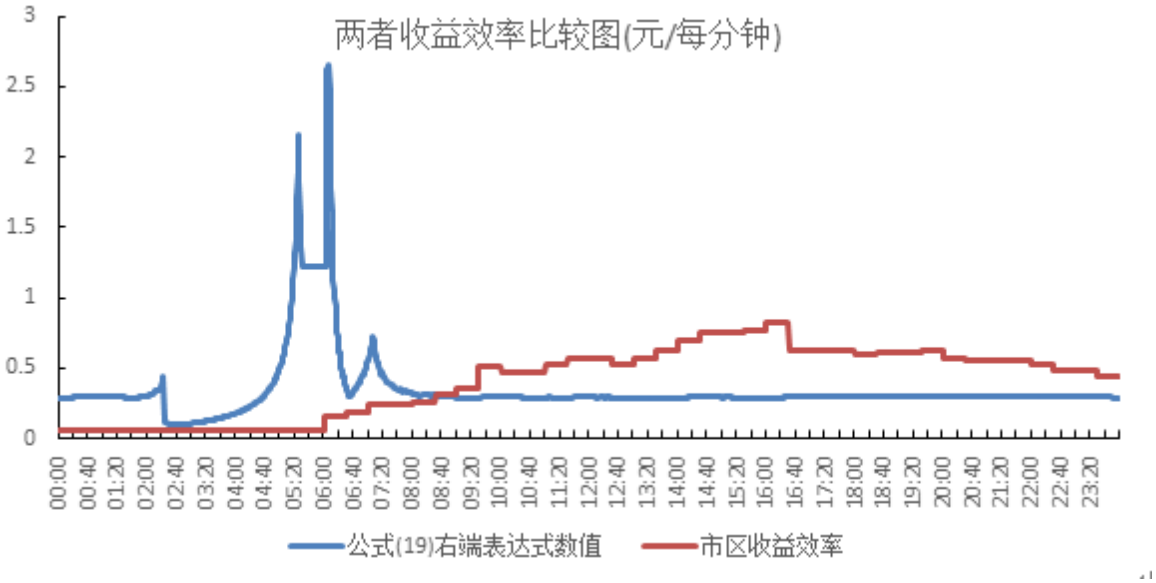

图 5-11 决策公式两端表达式对比图

由于一些数据难以找寻且具有一定的随机性，结合现实，拟定每批出租车进入“乘车区”耗时为 2.5 分钟，管理人员分批次放行的标准出租车数量为 1（辆/每次），“蓄车池”中初始状态已有的车辆数目为 22 辆，“乘车区”内等待乘客群体数为 5 个。在 8 月份成都双流机场的环境下，两个收益效率比在 8 月某天的 8：32 分相等。根据决策方程，在这种情况下，8：32 之前司机应该选择方案 A，在 8：32 之后应该选择方案 B。

观察图 5-11 中的两条曲线，它们的取值范围相近、数值与现实相比较为合理、有交叉点且异常数值较少，说明第一问所求决策方程具有一定的合理性。

5.2.3 依赖性分析

根据前面的模型与分析，飞机场的出租车司机做出决策取决于等待时间、出租车司机在市区各时段能享受的收益效率以及汽车的计价方式和油费这几个变量。除了等待时间以外的变量都可以根据客观数据计算，且在一定时间内较为固定。而等待时间主要取决于起始时间点的队伍群体数、起始时间点的“蓄车池”中已有的车辆数目以及每批出租车进入“乘车区”耗时。而这三个变量难以找寻客观数据且数值具有一定的随机性。

因此，在其他因素不变的情况下，分别改变这三个变量来研究其对决策方程的影响。第一次先拟定 3 个合理数值，往后每次增加原值的 10%，直到连续取出十个值，如下表所示。

表 5-4 变量取值表

q	9	10	11	12	13	14	15	16	17	18	19
α	2	2.2	2.4	2.6	2.8	3	3.2	3.4	3.6	3.8	4
Δt	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2

每次仅变换一个变量的值并算出决策方程右端表达式的当日平均收益率，可分别得出三个变量对决策方程右端表达式数值的影响，如下图所示：

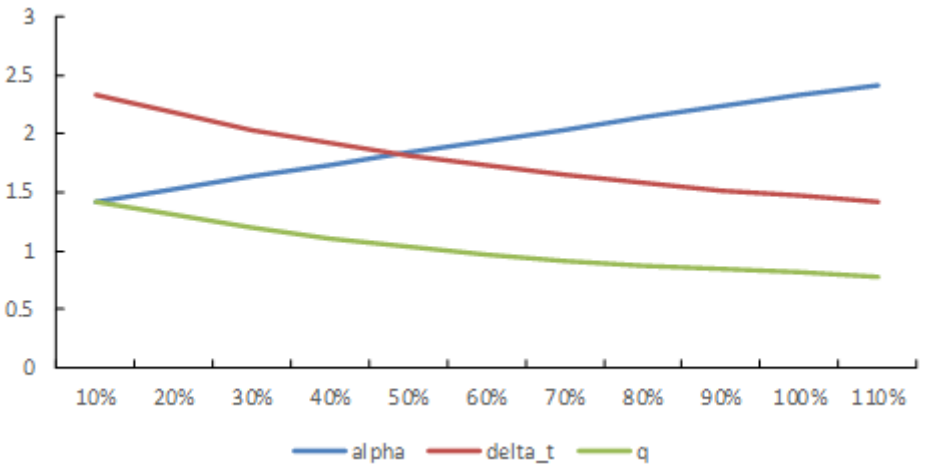

图 5-12 选取变量对决策方程右端表达式数值的影响图

由上图易知，起始时间点的“蓄车池”中已有车辆数目和每批出租车进入“乘车区”耗时的大小与决策方程右端表达式的数值成较弱的负相关关系，表明这两个变量的数值越大，在飞机场的出租车司机选择 B 方案的合理性更大。管理人员分批次放行的标准出租车数量与决策方程右端表达式的数值成较强的正相关关系，表明这个变量的数值越大，在飞机场的出租车司机选择 A 方案的合理性越大。

5.3 并行车道“上车点”的设置

第三问的要求是使总的乘车效率最高，结合第一问建立的模型，这意味着出租车在机场的等待时间 t_1 最短。

5.3.1 乘客排队乘车

由公式(17)可知当乘客排队乘车时 t_1 由“蓄车池”中已有出租车数量 q、每批出租车进入“乘车区”耗时 Δt 以及管理人员分批次放行的标准出租车数量 α 共同决定。已有出租车数量 q 根据现实状况而定，是一个固定值。因此，可以通过控制 Δt 与 α 的数值来使总的乘车效率最高。

根据假设，每批放行的出租车进入“乘车区”时以相同的加速度同时启动，并且达到最高限速时保持原速。所以不论管理人员分批次放行的标准出租车数量 α 的取值为多少，两个车道所有车辆在 x 轴方向的间距是不会发生任何改变的。

(1) α 的计算

由于车道数量为偶数，而 α 可能取奇数也可能取偶数，所以将这两种情况分开讨论：

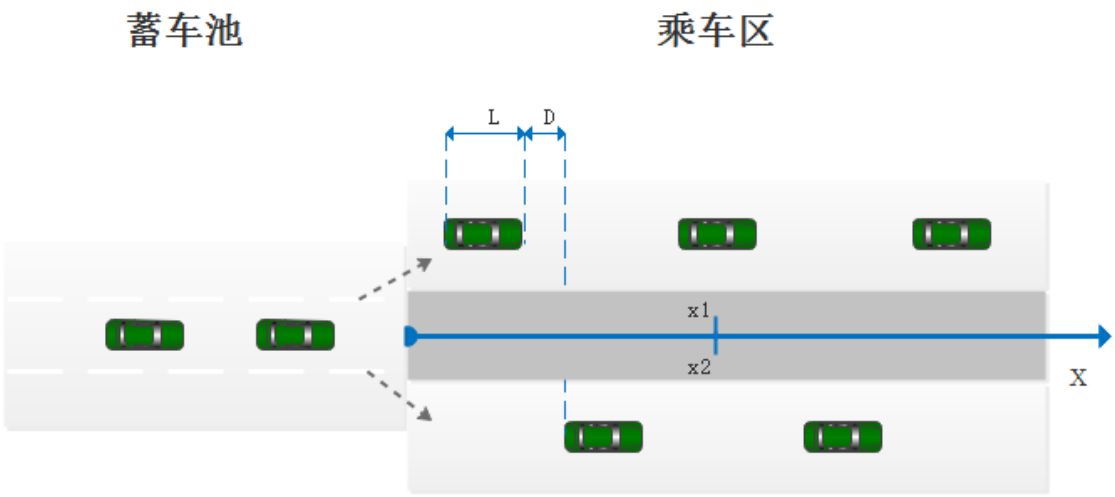

图 5-13 α 为奇数时每批出租车进入“乘车区”的停车示意图

当α为奇数时，如图 5-13 所示，以“蓄车池”和“乘车区”的边界中点为坐标原点，沿 “乘车区”车辆驶入方向建立一条坐标轴 X 轴。将“上车点”的位置放在 X 轴上表示。且第α辆车的车尾在 X 轴上的位置位于坐标原点。

易知车道 1 与车道 2 的上车点位置相同，均为两条车道车辆流的中心，即：

$$x_1 = x_2 = \frac{(\alpha-1)D+L}{2} \tag{20}$$

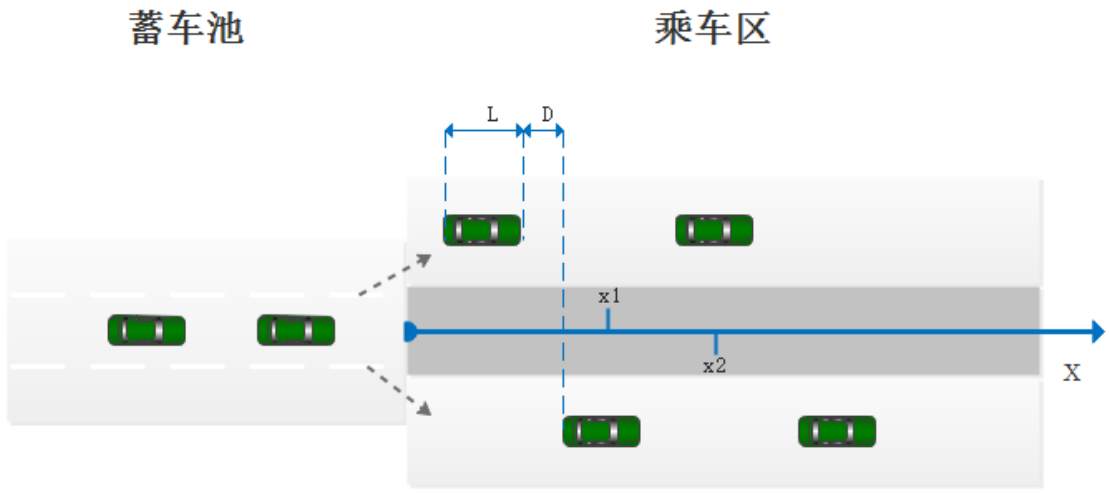

图 5-14 α 为偶数时每批出租车进入“乘车区”的停车示意图

当α为偶数时，如图 5-14 所示，“上车点”的位置依旧位于每条车道车辆流的中心，但在 X 轴上的位置不再相等：

$$\begin{cases} x_1 = \dfrac{(\alpha-2)D+(\alpha-1)L}{2} \\ x_2 = \dfrac{(\alpha-1)D+\alpha L}{2} \end{cases} \tag{21}$$

由于“乘车区”的长度是固定的，因此每条车道的车辆流长度不能超过“乘车区”的长度，即：

$$\alpha L+(\alpha-1)d\le Y \tag{22}$$

由公式(20)与公式(21)可知“上车点”x 可以表达为每批次放行的标准出租车数量α的函数。因此，确定了最优的α数值，即可确定每条车道最优的“上车点”位置。

(2) Δt 的计算

假定Δt 由出租车在“乘车区”的行驶时间、所有乘客都已走到车门处的时间与乘客放置行李、上车的时间共同决定，即：

$$\begin{aligned}\Delta t &= t_c + t_p + t_p' \\ &= \frac{\alpha L+(\alpha-1)D}{v_c}-\frac{v_c}{2a}+\frac{(\alpha-1)(L+D)}{2v_p}+t_p' \\ &= (\frac{1}{v_c}+\frac{1}{2v_p})(L+D)\alpha-(\frac{D}{v_c}+\frac{v_c}{2a}+\frac{L+D}{2v_p}+t_p')=t(\alpha)\end{aligned} \tag{23}$$

公式(23)中出租车在“乘车区”的行驶时间通过速度-位移公式以及两条车道车辆流总长度的表达式计算求得。在公式(23)中直接以结果形式呈现。

观察公式(23)可以发现Δt 是α的一次函数，因此结合公式(15)和公式(17)可以求得等待时间 t_1 由α表达的函数公式：

$$t_1=\left\lceil\frac{q+1}{\alpha}\right\rceil t(\alpha),\ m_{T_0}>q+1 \tag{24}$$

将客观现实数据代入公式(24)，结合公式(22)的约束条件，找寻使等待时间 t_1 值最小的α数值，即为最优α值。将最优α值根据奇偶性代入公式(20)或公式(21)即可求得最优 “上车点”的位置。

5.3.2 出租车排队载客

对于乘客来说，出租车是召之即来的，这里我们换一种模型方法：当客流量非常大时，只统计可能出现的α值即可。

当α为奇数时：

(1) 当队伍人数大于“乘车区”的最大容纳量，即 $m_{T_0}\ge\frac{Y+D}{L+D}$时，两条车道的“上车点”位置为：

$$x_{2,\alpha}=x_{1,\alpha}=\frac{\alpha L+(\alpha-1)D}{2} \tag{25}$$

(2) 当队伍数量小于“乘车区”的最大容量，即 $m_{T_0}<\frac{Y+D}{L+D}$时 ，α 的数值大小即为队伍数量的大小，队伍数量可取：1，2，3，…,n ,其中 $n=\left\lceil\frac{Y+D}{L+D}\right\rceil=\alpha$，则：

$$x_{2,\alpha}=x_{1,\alpha}=\frac{\alpha L+(\alpha-1)D}{2} \tag{26}$$

到这里可以得到α在任意值下所有$x_{1,\alpha}$和$x_{2,\alpha}$的数值。

$$F(x_2)=\sum_{\alpha=1}^{n}[\frac{(\alpha-1)L+(\alpha-2)D}{2}] \tag{27}$$

对公式(27)求最小值可以求得最优x_1和x_2。

当α为偶数时：

(1) 当队伍人数大于乘车区的最大容纳量，即$m_{T_0}\geq\frac{Y+D}{L+D}$时，两条车道的上车点位置为：

$$x_{1,\alpha}=\frac{(\alpha+1)L+\alpha D}{2},\quad x_{2,\alpha}=\frac{(\alpha-1)L+(\alpha-2)D}{2} \tag{28}$$

(2) 当队伍数量小于乘车去的最大容量，即$m_{T_0}<\frac{Y+D}{L+D}$时，α的数值大小即为队伍数量的大小，队伍数量可取：1，2，3，…,n ,其中 $n=\left\lceil\frac{Y+D}{L+D}\right\rceil=\alpha$ ，则：

$$x_{1,\alpha}=\frac{(\alpha+1)L+\alpha D}{2},\quad x_{2,\alpha}=\frac{(\alpha-1)L+(\alpha-2)D}{2} \tag{29}$$

到这里可以得到α在任意值的情况下所有$x_{1,\alpha}$和$x_{2,\alpha}$的数值。

$$F(x_1)=\sum_{\alpha=1}^{n}[\frac{（\alpha+1）L+\alpha D}{2}],\quad F(x_2)=\sum_{\alpha=1}^{n}[\frac{(\alpha-1)L+(\alpha-2)D}{2}] \tag{30}$$

$$x_1=x_2+D+L \tag{31}$$

对公式(30)中任意一个函数求最小值可以求出最优x_1或x_2，由公式(31)可以得到另一个“乘车道”的最优“上车点”。

5.4 “优先”安排方案

5.4.1 模型的建立

我们拟定的“优先方案”是另开一条连接到“乘车区”的“优先蓄车池”，只供短途载客再次返回的出租车使用，缩小该类出租车的二次等待时间，从而改善总收益效率。而定义供短途载客再次返回的出租车需设立一个临界时间值，若某出租车从机场离开到再次返回机场的时间小于该临界值，则将其排入专门的“优先蓄车池”。

首先计算该类出租车两趟载客的总收益效率：

$$\omega_{total}=\frac{\mathrm{g}\left(d_{2,short}\right)-kd_{2,short}+\mathrm{g}\left[E\left(d_2\right)\right]-kE\left(d_2\right)}{E(t_1)+t_{2,short}+t_{1,short}+E\left(t_2\right)} \tag{32}$$

其中$d_{2,short}$是关于$t_{2,short}$的一个函数：

$$d_{2,short}=l\left(t_{2,short}\right)=\begin{cases}v_0t_{2,short},d_{2,short}\le d_0\\ v_0t_0+v_1(t_{2,short}-t_0),d_{2,short}>d_0\end{cases} \tag{33}$$

其他变量可以继续沿用问题一中的出的已知数据。

易知$t_{1,short}$与$t_{2,short}$之间有关联：

将$t_{2,short}$看作一个自变量，由公式(21)可知，$t_{2,short}$能够决定第一次短途载客行驶的距离$d_{2,short}$。由数据集可得出载客行驶距离的概率分布，从而能够解出载客距离小于等于$d_{2,short}$的概率p'，将概率p'当作蓄车池中上次短途载客的车辆数占蓄车池总车辆数的比例。令每日 “蓄车池”已有出租车平均总数为 22，可认为其中 22* p'个车辆上一次的载客路程是小于$d_{2,short}$的，再将 22* p'带入决策方程，可以得到$t_{1,short}$。因此$t_{1,short}$可以表达为$t_{2,short}$的一个函数。将两者带入公式(20)中得到ω_{total}，最后对每一个$t_{2,short}$进行运算，选取在相同条件下ω_{total}与期望收益率$E(\omega_{\mathrm{A}})$的数值相等时的$t_{2,short}$，该$t_{2,short}$对应的$d_{2,short}$就是临界距离值，$t_{2,short}$就是优先权制度中应该规定的临界时间值。

5.4.2 模型的求解

拟定$t_{2,short}$是1分钟到40分钟内的任意值，找出最优的$t_{2,short}$数值，使得$\omega_{total}=E(\omega_{\mathrm{A}})$。首先对$t_{2,short}$采用穷举法得出一定数量的$t_{2,short}$值。

由于d_0段和(d_2 - d_0)段，即机场到市区入口的路段和市区入口到达市区目的地的车速是不同的，将d_0段的配速v_0设为 70km/h，(d_2 - d_0)段的配速v_1设为 50km/h。将上述数值代入公式(21)，即可求得$d_{2,short}$的数值。

之后从 2014 年 8 月 15 日的出租车 GPS 经纬度数据中，抽取 10 辆在当日搭载过双流机场乘客的出租车的信息，并描绘出每次生意的完整路线图，计算出路线的长度。通过这十条线路的长度以及其出现的频率，拟合出一条机场载客路线长度的概率密度函数。通过拟合，我们发现拟合程度最高的概率密度函数是期望为 12.5 公里，标准差为 2.5 公里的正态分布概率密度函数，即：

$$p(d_{2,short}) = \frac{1}{2.5\sqrt{2\pi}} e^{-\frac{(d_{2,short}-12.5)^2}{12.5}} \tag{34}$$

拟合函数图像如图 5-15 所示。

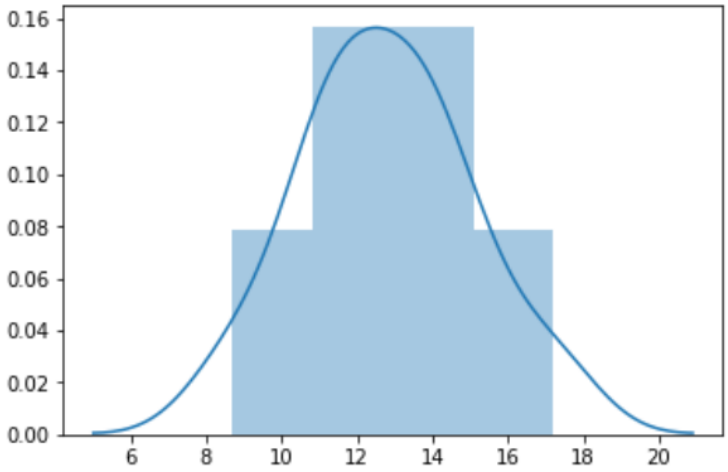

图 5-15 概率密度函数拟合图

将所求得的 $d_{2,short}$ 数值代入公式(21)即可求出 $p(d_{2,short})$。

接下来需要求出第二趟载客的等待时间 $t_{1,short}$。依旧选用第一问中的假设：初始的队伍群体数 m_{T_0} 为 5 个、“蓄车池”中初始车辆数目 q 为 22 辆、其中有 $p(d_{2,short})$ 的出租车会遇到短途的乘客，即“优先蓄车池” 中初始车辆数目数量为 22 $p(d_{2,short})$、当前时间处于 2016 年 8 月份的某一天。将这些数值代入公式(8)可求出 $t_{1,short}$。

之后计算 ω_{total}，将 $t_{1,short}$，$d_{2,short}$ 用 $t_{2,short}$ 来表达，并将其他计算出来的数值代入公式(20)可得：

$$\omega_{total} = \frac{\mathrm{g}\left(l(t_{2,short})\right) - 0.5026 \times l(t_{2,short}) + 26.051}{30.5859 + t_{2,short} + f\left(22 \times p(l(t_{2,short})), n, T, Month\right) + 15.781} \tag{35}$$

$t_{2,short}$ 取 1、2、3、…、29、40，共四十个值，每一个 $t_{2,short}$ 对应一个两次载客的总收益效率 ω_{total}，如图 5-16 所示：

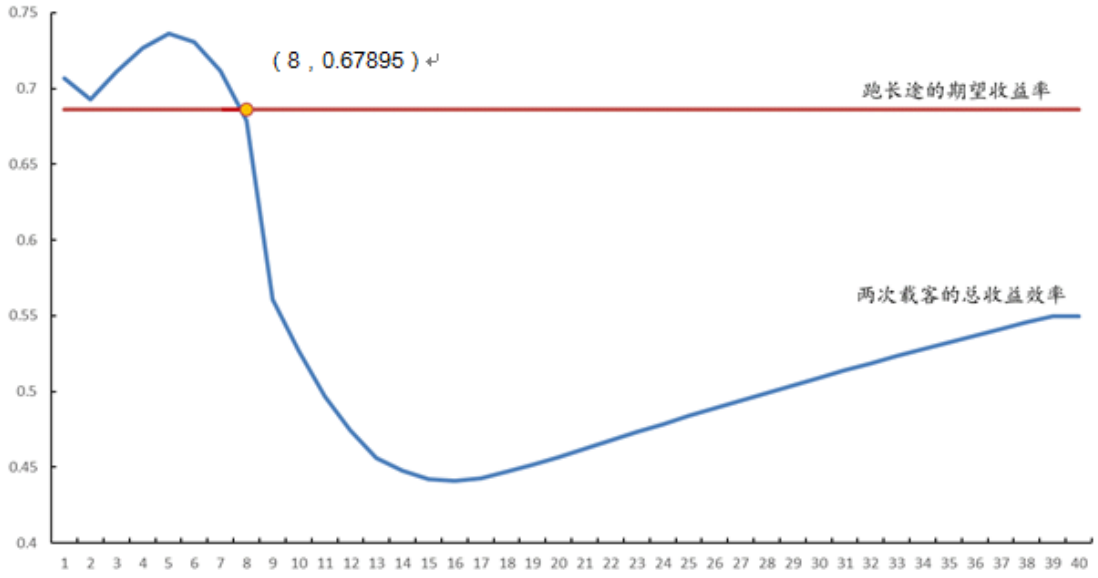

图 5-16 两次载客的总收益效率 ω_{total} 与出租车短途载客行驶的时间 $t_{2,short}$ 的关系图

易知，当 $t_{2,short}$ 取 8 分钟时，两次载客的总收益效率与期望收益效率相等，为 0.67895（元/分钟）。

5.4.3 结果分析

在假定的情况下，如图 5-16 所示，当某辆出租车短途载客行驶的时间小于或等于 8 分钟，即它返回机场的时间小于或等于 16 分钟时，该车可选择进入“优先蓄车池”等待载客。由于上图的两条曲线相交且相交点数值合理，可以说明，在假定情况下，该“优先”安排方案具有一定的合理性。

6 模型的评价与分析

6.1 模型的优缺点

6.1.1 模型的优点

(1) 本文在建立模型之前对大量数据进行了认真的分析和处理，通过将出租车的GPS 数据显示在基于 Leaflet 的地图上，可以更加清晰地观察分析，寻找数据中的规律。

(2) 在计算 A 方案的收益效率时，引入了乘客到达的速率、初始时刻乘客数量、乘客选择乘用出租车的概率等多个相关变量，使模型更加可靠，更加贴近实际。

6.1.2 模型的缺点

(1) 有些数值由于找不到确切的官方数据，被根据现实情况拟定，可能会使模型结果具有一定的误差。

(2) 由于未收集到成都双流机场平均出租车载客的人数，直接令乘客数量为乘客群体数，会对运算结果造成一定的影响。

(3) 由于出租车的 GPS 数据点是离散且高度重合的，在计算离开机场后的行驶距离时容易错误判断乘客的下车点，所以会影响行驶距离密度函数的准确性。

6.2 模型的改进方向

为了缓解机场出租车司机因为收益低而不愿搭载短途乘客的社会问题，本文对出租车收益的影响因素进行分析，建立了司机决策模型；对，建立了短途优先模型。通过改变是否为短途的判断条件，给予某些短途载客再次返回的出租车一定的“优先权”，从而达到使机场出租车司机的收益尽量均衡的目的。

本文建立的“短途优先模型”能够广泛应用于机场、车站等人流量较大的地方，能够在一定程度上缓解“客流拥堵”的问题，体现了一定的管理智慧。

参考文献

[1] DC 竞赛，交通线路通达时间预测，https://www.dcjingsai.com/common/cmpt/%E4%BA%A4%E9%80%9A%E7%BA%BF%E8%B7%AF%E9%80%9A%E8%BE%BE%E6%97%B6%E9%97%B4%E9%A2%84%E6%B5%8B_%E8%B5%9B%E4%BD%93%E4%B8%8E%E6%95%B0%E6%8D%AE.html, 2019-9-13.

[2] wind 金融终端宏观数据平台, 2012 年成都机场各个月份吞吐量, 2019-9-14.

[3] CSDN 论坛，国内航班数据，https://download.csdn.net/download/weixin_44572004/10926732, 2019-9-14.

[4] 成都市政府数据平台,客运出租汽车车辆营运信息, http://www.sohu.com/a/259646430_697917, 2019-9-14.

[5] 冯晓梅，供需平衡状态下的出租车发展规模研究，西南交通大学: 52 页表-18，49 页表 5-6, 2010.

附录一、程序的源代码（Python）

1. 问题一：计算每分钟的乘客到达数

```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns',40)

import matplotlib.pyplot as plt
import folium
from geopy.distance import geodesic

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

import warnings
warnings.filterwarnings('ignore')

data=pd.read_excel('./datasets/flight_new.xls')
data['降落时间_hm']=[x.strftime('%H:%M:%S')[0:5] for x in data['降落时间']]

dict_plane={'JET':180,'波音 737(中)':175,'ERJ-190(中)':115,'其他机型':210,
            '空客 320(中)':170,'CRJ(小)':70,'空客 321(中)':220,'空客 319(中)':128,
            '庞巴迪 CRJ900':90,'波音 787(大)':300,'空客 321(窄体机)':150,'波音 777(大)':350,
            '空客 330(宽体机)':335,'波音 757(中)':200,'空客 380(大)':555,'新舟 60(小)':60,
            '波音 747(大)':370,'ERJ(小)':50,'波音 767(大)':280}
data['人数']=0
for i in range(data.shape[0]):
    plane=data.iloc[i,9]
    num=dict_plane[plane]
    data.iloc[i,30]=num
lst_time=[]
for i in range(0,24):
    for j in range(0,60,5):
        if i<10 and j<10:
            lst_time.append('0'+str(i)+':0'+str(j))
        if i<10 and j>=10:
            lst_time.append('0'+str(i)+':'+str(j))
        if i>=10 and j<10:
            lst_time.append(str(i)+':0'+str(j))
        if i>=10 and j>=10:
            lst_time.append(str(i)+':'+str(j))
lst_airport=data['降落机场'].value_counts()[0:10].index

def mean_num_by_airport(data):
    lst_mean=[]
```

```
    for t in lst_time:
        df=data[data['降落时间_hm']==t].groupby(['降落机场']).sum()
        lst_num=[]
        for name in lst_airport:
            if name in df.index:
                num=df.loc[name,'人数']
                lst_num.append(num)
        lst_mean.append(np.mean(lst_num))
    return lst_mean
lst=mean_num_by_airport(data)

plt.figure(figsize=(40,4))
plt.xticks(rotation=90)
plt.plot(lst_time,lst)
plt.show()
l=[]
for i in range(24):
    l.append(np.mean(lst[0+i:0+i+6]))
plt.figure(figsize=(8,4))
plt.xticks(rotation=90)
plt.ylim(0,900)
plt.plot(l)
plt.show()
```

2. 问题一：筛选出客流量前十的机场

```
import pandas as pd
pd.set_option('display.max_columns',29)

import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

import warnings
warnings.filterwarnings('ignore')
data=pd.read_excel('./datasets/flight_new.xls')

lst_name=data['降落机场'].value_counts()[0:10].index
lst_num=data['降落机场'].value_counts()[0:10].values

plt.figure(figsize=(8,5))
plt.bar(lst_name,lst_num)
for x, y in enumerate(lst_num):
    plt.text(x, y+10, '%s' % y, ha='center', va='bottom')
```

```
plt.show()
data['航班有效期结束'].value_counts()
```

3. 问题二：计算双流机场中出租车数量占成都市出租车数量的比例

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import folium
from geopy.distance import geodesic

file_name=[]
lst_1=['./datasets/20140815_train/20140815_train_00'+str(i)+'.txt' for i in range(1,10)]
lst_2=['./datasets/20140815_train/20140815_train_0'+str(i)+'.txt' for i in range(10,100)]
file_name.extend(lst_1)
file_name.extend(lst_2)
file_name.append('./datasets/20140815_train/20140815_train_100.txt')

def get_airport(data):
    df1=data[(data['经度']>103.956063)&(data['经度']<103.95944)&(data['纬度']>30.579103)&(data['纬度']<30.583377)]
    df2=data[(data['经度']>103.953116)&(data['经度']<103.958901)&(data['纬度']>30.569091)&(data['纬度']<30.573429)]
    df=pd.concat([df1,df2])
    return df

lst_data_len=[]
lst_sub_len=[]
for i in range(100):
    data=pd.read_table(file_name[i],
                       header=None,sep=',',names=['出租车 ID','纬度','经度','载客状态(1 表示载客，0 表示无客)','时间点'])
    lst_data_len.append(data.shape[0])

    sub_len=get_airport(data).shape[0]
    lst_sub_len.append(sub_len)

print(np.sum(lst_sub_len)/np.sum(lst_data_len))
np.sum(lst_sub_len),np.sum(lst_data_len)

lst_car=[]
for i in range(100):
    data=pd.read_table(file_name[i],
                       header=None,sep=',',names=['出租车 ID','纬度','经度','载客状态(1 表示载客，
```

```
0 表示无客）','时间点'])
    lst_car.append(len(data['出租车 ID'].unique()))

np.sum(lst_car)

data0=pd.read_table(file_name[0],
                    header=None,sep=',',names=['出租车 ID','纬度','经度','载客状态（1 表示载客，0
表示无客）','时间点'])
data0_airport=get_airport(data0)



a=data0_airport.iloc[0:3000,:]
d_lat=0.0021
d_lng=0.0009

incidents = folium.map.FeatureGroup()
for lat, lng, in zip(a['纬度'],a['经度']):
    incidents.add_child(
        folium.Circle(
            [lat+d_lat, lng-d_lng],
            radius=4,
            color='blue',
            opacity=0.5
        )
    )

san_map = folium.Map(location=[30.575389,103.960367], zoom_start=12)
san_map.add_child(incidents)
```

4. 问题二：计算机场高速路的路程和市区内期望车程

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import folium
from geopy.distance import geodesic

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
import warnings
warnings.filterwarnings('ignore')

file_name=[]
```

```
lst_1=['./datasets/20140815_train/20140815_train_00'+str(i)+'.txt' for i in range(1,10)]
lst_2=['./datasets/20140815_train/20140815_train_0'+str(i)+'.txt' for i in range(10,100)]
file_name.extend(lst_1)
file_name.extend(lst_2)
file_name.append('./datasets/20140815_train/20140815_train_100.txt')

data0=pd.read_table(file_name[0],
                    header=None,sep=',',names=['出租车 ID','纬度','经度','载客状态（1 表示载客，0
表示无客）','时间点'])

#取 60000 个点
data_sub=data0.iloc[list(range(0,592099,10)),:]
plt.scatter(data_sub['经度'],data_sub['纬度'])

sns.jointplot(x='经度', y='纬度',kind='kde',data=data_sub,xlim=(103.8,104.3),ylim=(30.4,30.9))
plt.show()

sns.jointplot(data0['经度'],data0['纬度'],kind='hex',xlim=(103.8,104.3),ylim=(30.4,30.9))
plt.show()

a=data0.iloc[list(range(0,592099,100)),:].iloc[0:3200,:]

incidents = folium.map.FeatureGroup()

for lat, lng, in zip(a['纬度'],a['经度']):
    incidents.add_child(
        folium.Circle(
            [lat, lng],
            radius=4,
            color='blue',
            opacity=0.5
        )
    )

san_map = folium.Map(location=[30.643429,104.123414], zoom_start=12)
san_map.add_child(incidents)

def split_image(j1,j2,w1,w2):
    mean_j=np.mean([j1,j2])
    mean_w=np.mean([w1,w2])
    lst_j=[j1,np.mean([j1,mean_j]),mean_j,np.mean([mean_j,j2]),j2]
    lst_w=[w1,np.mean([w1,mean_w]),mean_w,np.mean([mean_w,w2]),w2]
    return lst_j,lst_w
```

```
lst_j=split_image(103.991304,104.165503,30.72951,30.603767)[0]
lst_w=split_image(103.991304,104.165503,30.72951,30.603767)[1]

#分数据
def split_data(data,lst_j,lst_w):
    lst_data=[]
    for i in range(4):
        for j in range(4):
            df=data[(data['经度']>lst_j[j])&(data['经度']<lst_j[j+1])&(data['纬度']>lst_w[i+1])&(data['纬度']<lst_w[i])]
            lst_data.append(df)
    return lst_data
lst_data=split_data(data0,lst_j,lst_w)


array=[[0.019662083538124823,0.03502039513205375,0.019880527794931255,0.003923594920330859],

[0.06737745067150605,0.11614933185463375,0.08372716312324877,0.014129562649392767],

[0.06985385239049431,0.17742294588883709,0.13924141030972875,0.02677832528870349],

[0.02185282738282776,0.12102862039849273,0.05576209740093343,0.028189811255760 42]]
sns.heatmap(array,cmap='Blues',annot=True,fmt='.2')

def get_center(lst_j,lst_w):
    lst_center=[]
    for i in range(4):
        for j in range(4):
            lst_center.append((np.mean([lst_j[j],lst_j[j+1]]),np.mean([lst_w[i],lst_w[i+1]])))
    return lst_center

lst_center=get_center(lst_j,lst_w)

def get_distance(lst_center,start):
    lst_distance=[]
    for center in lst_center:
        lst_distance.append(geodesic(center,start).km)
    return lst_distance
start=(30.607994,104.044196)
lst_center_back=[x[::-1] for x in lst_center]
lst_distance=get_distance(lst_center_back,start)
lst_distance
```

```
# lst_center_back
plt.scatter([x[0] for x in lst_center],[x[1] for x in lst_center])
plt.scatter(start[1],start[0])

df=pd.DataFrame([lst_center,lst_proba,lst_distance],index=['中心点','概率','到起点距离']).T

np.sum([x*y for x,y in zip(lst_proba,lst_distance)])

airport=(30.576348,103.966654)
turn=(30.586608,104.024721)
l1=geodesic(airport,turn)
l2=geodesic(turn,start)
l1,l2

import folium

minus_j=0.008759
minus_w=0.003028
A=[airport[1]-minus_j,airport[0]-minus_w]
B=[turn[1]-minus_j,turn[0]-minus_w]
C=[start[1]-minus_j,start[0]-minus_w]

#创建底层 Map 对象
m = folium.Map(location=[30,105],
                    zoom_start=8,
                    control_scale=True)

#画三个点
gj1 = folium.GeoJson(data={ "type": "MultiPoint",
  "coordinates": [A]
  })
gj2 = folium.GeoJson(data={ "type": "MultiPoint",
  "coordinates": [B]
  })
gj3 = folium.GeoJson(data={ "type": "MultiPoint",
  "coordinates": [C]
  })
gj1.add_to(m)
gj2.add_to(m)
gj3.add_to(m)

#画两条条线
gj4 = folium.GeoJson(data={ "type": "LineString",
  "coordinates": [A,B]
```

```
  })
gj5 = folium.GeoJson(data={ "type": "LineString",
  "coordinates": [B,C]
  })
gj4.add_to(m)
gj5.add_to(m)

#显示 m
m
```

5. 问题二：查看收益效率随等待人数、每批次车辆数、间隔时间的变化规律

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import   math
data1=pd.read_excel('./datasets/成都机场规律(1).xlsx')

lst_t=data1.iloc[:,7].index
lst_t

lst_h=data1.iloc[:,7].values
# lst_h

lst_t_m=[]
for i in range(24):
    for j in range(60):
        if i<10 and j<10:
            lst_t_m.append('0'+str(i)+':0'+str(j))
        if i<10 and j>=10:
            lst_t_m.append('0'+str(i)+':'+str(j))
        if i>=10 and j<10:
            lst_t_m.append(str(i)+':0'+str(j))
        if i>=10 and j>=10:
            lst_t_m.append(str(i)+':'+str(j))

lst_h_m=[]
for i in range(1440):
    lst_h_m.append((lst_h[math.floor(i/5)])/5)

m_T0=5
p=0.0987
lst_q_and_1,lst_i,lst_j,lst_i_minus_j,lst_ijdeta=[],[],[],[],[]
for q in range(9,20):
```

```
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<q+1):
            left+=lst_h_m[i+1]*p
            i+=1
        print('q+1:{0} ,i:{1} ,j{2} ,i-j:{3}'.format(q+1,i,j,i-j))
        lst_q_and_1.append(q+1)
        lst_i.append(i)
        lst_j.append(j)
        lst_i_minus_j.append(i-j)
        #
        s=0
        for k in range(j,i):
            s+=lst_h_m[k]*p
        deta=(q+1-m_T0-s)/(lst_h_m[i]*p)

        print('i-j+deta',i-j+deta)
        lst_ijdeta.append(i-j+deta)

lst_time_i=[lst_t_m[i] for i in lst_i]
lst_time_j=[lst_t_m[j] for j in lst_j]

df=pd.DataFrame([lst_q_and_1,lst_time_i,lst_time_j,lst_i_minus_j,lst_ijdeta]).T
df.columns = ['q+1','i','j','i-j','i-j+deta']
df
# df.to_excel('qijdeta.xlsx')
# df.head()
plt.plot(df[df['q+1']==10]['i-j+deta'])
m_T0=5
p=0.0987
alpha=2
lst_n,lst_m=[],[]
lst_tb=[]
for q in range(9,20):
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<q+1):
            left+=lst_h_m[i+1]*p
            i+=1
        #循环
```

```
            m=m_T0
            lst_nj,lst_mj=[],[]
            for k in range(i-j):
                n=min(alpha,m)
                m=m+2*lst_h_m[j+k]*p-n
                lst_nj.append(n)
                lst_mj.append(m)
            n_total=sum(lst_nj)
            tb=(q+1-n_total)/alpha*2
            lst_tb.append(tb)

lst_haha=[]
for x,y in zip(lst_tb,lst_ijdeta):
    lst_haha.append(x+y)
df=pd.DataFrame([lst_tb,lst_ijdeta,lst_haha]).T
df.columns=['tb','i-j+deta','tb+i-j+deta']
df.head()
# df.to_excel('waiting_time_alpha2.xlsx')
plt.figure(figsize=(20,5))
plt.plot(lst_ijdeta)
plt.plot(lst_haha)

lst_q=list(range(9,20))
lst_deta_t=[x/10 for x in list(range(10,21))]
lst_alpha=[x/10 for x in list(range(20,41,2))]

print('q:',lst_q)
print('deta_t:',lst_deta_t)
print('alpha:',lst_alpha)

m_T0=5
p=0.0987

def get_lst_ijdeta(q):
    lst_ijdeta=[]
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<q+1):
            left+=lst_h_m[i+1]*p
            i+=1
        #
        s=0
```

```
            for k in range(j,i):
                s+=lst_h_m[k]*p
            deta=(q+1-m_T0-s)/(lst_h_m[i]*p)
            lst_ijdeta.append(i-j+deta)
        return lst_ijdeta
def get_lst_tb(q,deta_t,alpha):
    lst_tb=[]
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<q+1):
            left+=lst_h_m[i+1]*p
            i+=1
        #循环
        m=m_T0
        lst_nj,lst_mj=[],[]
        for k in range(i-j):
            n=min(alpha,m)
            m=m+deta_t*lst_h_m[j+k]*p-n
            lst_nj.append(n)
            lst_mj.append(m)
        n_total=sum(lst_nj)
        tb=(q+1-n_total)/alpha*deta_t
        lst_tb.append(tb)
    return lst_tb

deta_t,alpha=2,2

lst_ijdeta=[]
for q in lst_q:
    lst_ijdeta.append(get_lst_ijdeta(q))

lst_n,lst_m=[],[]
lst_tb=[]
for q in lst_q:
    lst_tb.append(get_lst_tb(q,deta_t,alpha))

lst_haha_q=[]
for x,y in zip(lst_tb,lst_ijdeta):
    lst=[]
    for tb,ijdeta in zip(x,y):
        lst.append(tb+ijdeta)
    lst_haha_q.append(lst)
```

```
q,alpha=9,2

lst_ijdeta=[]
for deta_t in lst_deta_t:
    lst_ijdeta.append(get_lst_ijdeta(q))

lst_n,lst_m=[],[]
lst_tb=[]
for deta_t in lst_deta_t:
    lst_tb.append(get_lst_tb(q,deta_t,alpha))

lst_haha_t=[]
for x,y in zip(lst_tb,lst_ijdeta):
    lst=[]
    for tb,ijdeta in zip(x,y):
        lst.append(tb+ijdeta)
    lst_haha_t.append(lst)

q,deta_t=9,2

lst_ijdeta=[]
for alpha in lst_alpha:
    lst_ijdeta.append(get_lst_ijdeta(q))

lst_n,lst_m=[],[]
lst_tb=[]
for alpha in lst_alpha:
    lst_tb.append(get_lst_tb(q,deta_t,alpha))

lst_haha_a=[]
for x,y in zip(lst_tb,lst_ijdeta):
    lst=[]
    for tb,ijdeta in zip(x,y):
        lst.append(tb+ijdeta)
    lst_haha_a.append(lst)

plt.plot(np.array(lst_haha_q).flatten())
plt.plot(np.array(lst_haha_t).flatten())
plt.plot(np.array(lst_haha_a).flatten())

df_q=pd.DataFrame([lst_haha_q[i] for i in range(0,11)]).T
df_q.columns=lst_q
```

```
df_deta_t=pd.DataFrame([lst_haha_t[i] for i in range(0,11)]).T
df_deta_t.columns=lst_deta_t

df_alpha=pd.DataFrame([lst_haha_a[i] for i in range(0,11)]).T
df_alpha.columns=lst_alpha
# df_q.to_excel('df_q.xlsx')
# df_deta_t.to_excel('df_deta_t.xlsx')
# df_alpha.to_excel('df_alpha.xlsx')
q,deta_t,alpha=22,2.5,1

lst_ijdeta=get_lst_ijdeta(q)

lst_n,lst_m=[],[]
lst_tb=get_lst_tb(q,deta_t,alpha)

lst_haha=[]
for x,y in zip(lst_tb,lst_ijdeta):
    lst_haha.append(x+y)
plt.plot(lst_haha)

df_mcx=pd.DataFrame(lst_haha)
df_mcx.columns=['q=22,deta_t=2.5,alpha=1']
```

6. 问题四：计算出租车行驶距离的密度函数

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from geopy.distance import geodesic

file_name=[]
lst_1=['./datasets/20140815_train/20140815_train_00'+str(i)+'.txt' for i in range(1,10)]
lst_2=['./datasets/20140815_train/20140815_train_0'+str(i)+'.txt' for i in range(10,100)]
file_name.extend(lst_1)
file_name.extend(lst_2)
file_name.append('./datasets/20140815_train/20140815_train_100.txt')
data=pd.read_table(file_name[0],
                   header=None,sep=',',names=['出租车 ID','纬度','经度','载客状态','时间点'])

data.groupby(['出租车 ID']).count()

p1=(30.580,104.000)
p2=(30.586,104.025)
```

```
ID=50
car=data[data['出租车 ID']==ID]
# car=car[(car['纬度']>p1[0])&(car['纬度']<p2[0])&(car['经度']>p1[1])&(car['经度']<p2[1])]
# print(car['时间点'])
car=car.iloc[0:3000,:]

car_0=car[car['载客状态']==0]
car_1=car[car['载客状态']==1]

d_lat=0.0026
d_lng=0.0023
incidents = folium.map.FeatureGroup()
for lat, lng, in zip(car_0['纬度'],car_0['经度']):
    incidents.add_child(
        folium.Circle(
            [lat+d_lat, lng-d_lng],
            radius=4,
            color='green',
            opacity=0.5
        )
    )
for lat, lng, in zip(car_1['纬度'],car_1['经度']):
    incidents.add_child(
        folium.Circle(
            [lat+d_lat,lng-d_lng],
            radius=4,
            color='red',
            opacity=0.5
        )
    )

san_map = folium.Map(location=[30.5868,104.025], zoom_start=12)
san_map.add_child(incidents)

lst=[8.702+3.6,8.702+3.1,8.702+2.7,8.702+4.5,8.702,8.702+5.8,8.702+5.0,8.702+1.8,8.702+4.9,8.70
2+8.5,5.684+9.6,8.702+2.4]
sns.distplot(lst)

l=np.random.normal(12.5,2.5,2000)
sns.kdeplot(l)
sns.kdeplot(lst)
```

7. 问题四：查看不同的短途判定时间和收益效率的关系

```
import pandas as pd
import numpy as np
import math
import sympy as sp
from scipy.integrate import quad
import matplotlib.pyplot as plt

def t_to_d(t_2_short):
    if t_2_short<=8.702/70*60:
        d_2_short=70*t_2_short/60
    else:
        d_2_short=8.702+50*(t_2_short-8.702/70*60)/60
    return d_2_short

def f(x):
    return (1/2.5/math.sqrt(2*3.1415))*(2.718**(-((x-12.5)**2)/(2*2.5**2)))

def get_p(a):
    x=sp.Symbol('x')
    i=quad(f,0,a)
    return i

get_p(12.5)

lst_p=[]
lst_d=[]
for t_2_short in range(1,41):
    d_2_short=t_to_d(t_2_short)
    lst_d.append(d_2_short)
    p=get_p(d_2_short)
    lst_p.append(round(p[0],3))

plt.plot(lst_d)
plt.ylabel('d_2_short')
plt.xlabel('t_2_short')
plt.show()

plt.plot(lst_p)
plt.ylabel('Probability')
plt.xlabel('t_2_short')
plt.show()

data1=pd.read_excel('./datasets/成都机场规律(1).xlsx')
```

```
lst_t=data1.iloc[:,7].index
lst_h=data1.iloc[:,7].values

lst_t_m=[]
for i in range(24):
    for j in range(60):
        if i<10 and j<10:
            lst_t_m.append('0'+str(i)+':0'+str(j))
        if i<10 and j>=10:
            lst_t_m.append('0'+str(i)+':'+str(j))
        if i>=10 and j<10:
            lst_t_m.append(str(i)+':0'+str(j))
        if i>=10 and j>=10:
            lst_t_m.append(str(i)+':'+str(j))

lst_h_m=[]
for i in range(1440):
    lst_h_m.append((lst_h[math.floor(i/5)])/5)

def get_lst_ijdeta(q):
    lst_ijdeta=[]
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<(q+1)*proba):
            left+=lst_h_m[i+1]*p
            i+=1
        #
        s=0
        for k in range(j,i):
            s+=lst_h_m[k]*p
        deta=((q+1)*proba-m_T0-s)/(lst_h_m[i]*p)
        lst_ijdeta.append(i-j+deta)
    return lst_ijdeta

def get_lst_tb(q,deta_t,alpha):
    lst_tb=[]
    for j in range(len(lst_t_m)):
        #计算 T'
        left=m_T0+lst_h_m[j]*p
        i=j
        while(left<(q+1)*proba):
            left+=lst_h_m[i+1]*p
```

```
            i+=1
        #循环
        m=m_T0
        lst_nj,lst_mj=[],[]
        for k in range(i-j):
            n=min(alpha,m)
            m=m+deta_t*lst_h_m[j+k]*p-n
            lst_nj.append(n)
            lst_mj.append(m)
        n_total=sum(lst_nj)
        tb=((q+1)*proba-n_total)/alpha*deta_t
        lst_tb.append(tb)
    return lst_tb
m_T0=5
q,deta_t,alpha,p=22,2.5,1,0.0987

lst_mean=[]
#前 8 个，只算 tb，不算 i-j+deta
for proba in lst_p[1:9]:
    lst_n,lst_m=[],[]
    lst_tb=get_lst_tb(q,deta_t,alpha)

    lst_ijdeta=[0]*len(lst_tb)

    lst_t_1_short=[]
    for x,y in zip(lst_tb,lst_ijdeta):
        lst_t_1_short.append(x+y)
    lst_mean.append(np.mean(lst_t_1_short))
#9-30 个正常计算
for proba in lst_p[9:41]:
    lst_ijdeta=get_lst_ijdeta(q)

    lst_n,lst_m=[],[]
    lst_tb=get_lst_tb(q,deta_t,alpha)

    lst_t_1_short=[]
    for x,y in zip(lst_tb,lst_ijdeta):
        lst_t_1_short.append(x+y)
    lst_mean.append(np.mean(lst_t_1_short))
len(lst_mean)

plt.plot(lst_mean)
plt.xlabel('t_2_short')
plt.ylabel('t_1_short')
```

```
k=0.5026

def get_price(d):
    if d<=2:
        return 8
    else:
        return 8+(d-2)*1.9

def get_w(t2,d,t1):
    w=(get_price(d)-k*d+26.051)/(30.5859+t2+t1+15.781)
    return w
lst_w=[]
for t2,d,t1 in zip(list(range(1,41)),lst_d,lst_mean):
    lst_w.append(get_w(t2,d,t1))

plt.plot(lst_w)
plt.xlabel('t_2_short')
plt.ylabel('w')

lst_w
df=pd.DataFrame([list(range(0,30)),lst_d,lst_p,lst_mean,lst_w]).T
df.columns=['t_2_short','d_2_short','probability','t_1_short','w']
print(df.shape)
df.head()
df.to_excel('t2_d2_t1_w.xlsx')
df.drop(columns=['w','probability']).plot()

# plt.plot(df['probability'])
plt.plot(df['w'])
```