数学建模 C 题

2019 年 9 月 15 日

摘要

本论文主要建立了两个模型：选择决策模型和排队模型。选择决策模型通过分析出租车载客到达机场后放空回市区的全天收益，和在机场等候载客的全天收益，帮助司机进行留去选择。第一问的解决方案中，假设出租车在市区正常运营单位时间内的收益期望 λ，出租车每公里的物力成本 c，出租车载客时的收益 p(d)（元/千米），由 t 时刻航班数目和蓄车池内出租车的数目确定的等待时间 Tw（小时）。运用基本的数学运算，得到两种选择的利益之差，该差值最终是由司机可观测到的数据——当前航班数目和蓄车池内出租车数目共同决定的。在第二问的解决方案中，本论文收集了 2019 年 9 月 14 日深圳市航班离港数目及时间等数据，通过调查论文得到了深圳市机场出租车的数量与时间的关系以及机场乘客下飞机后选择网约车/出租车的比例，计算得到机场每天不同时间段的乘客数量以及出租车的容纳量，易得出租车司机等待载客的时间 Tw 与航班数和出租车数的关系，进而求得司机选择在机场等待载客的可观测条件，第二问得以解决。该选择决策模型的优点是计算量小，司机进行不同选择的依据简单明了；模型的缺点是数据灵敏度低，受现实因素如节假日、季节等影响。选择决策模型解决的问题一结果为：由司机根据个人经验和现有的航班数、出租车数量判断等待时间长短决定空载离开或者等待载客；问题二的结果为当航班数大于 41 且出租车少于 280 辆时，等待时间 T_w>2.8h，司机选择空载回市区更明智，其他大部分情况选择等待载客获取利润更多。

排队模型是在有两条并行车道的前提下，确定上车点的位置使得出租车接客效率最高且保证人车安全。第三问有两个解决方案，一是在右车道设置 N 个上车点，每个上车点间隔 20 米，每次分批放行 N 辆出租车沿着左车道按顺序前往上车点，最前面的车去最远的上车点接客，模型内乘客上车时间符合指数分布，要求最长上车时间为 3min，待所有出租车均接客离开上车点以后再放行下一批车，该方案满足了“绝对安全”，但接乘效率不高；第二个方案为在第 k 个车位的车接客离开后，可以马上放行一辆空车补位，该空车途中会路过 k-1 个上车点，会车时可能有该上车点的车恰好要变道驶出的冲突，因此安全系数降低，计算单位时间 (h) 内每个上车点平均能够接乘的次数得到效率，通过规划得到安全系数和效率之和最高的最优解，确定 N 的个数。第四问的解决方案是由短途载客的距离确定出租车司机可以插队的位置，以此保证接到短途客人的出租车返回机场利用优先权接长途乘客所得收益与直接接长途乘客的利润均衡。该模型的优点是：利用 python 模拟排队系统，求得数据更加精确；缺点是，现实中的突发状况和服务时间的不确定性可能会导致冲突较多，效率低。排队模型解决问题三的结果为：在右侧道路设置 6 个上车点，分批放行出租车，没有补位车，达到了绝对安全；或者在右侧设置 4 个上车点，每当一个上车点离开一辆车，就可以有下一辆空车驶入补位，安全系数为 0.83，效率为 0.90。问题四的结果为：路程在 5km 以内的短途载客可以插队到上车点直接载客，路程超过 5km 但是少于 10km 的短途载客要插队在进入上车点之前?? 辆车处继续等待载客。

一　问题重述

1.1　问题背景

第一、二问：由于国内多数机场都是将送客（出发）与接客（到达）通道分开的，送客到机场的出租车司机常面临两个选择：前往到达区排队等待载客返回市区，和直接放空返回市区拉客。在各种因素的影响下，做出不同的决策分别会付出不同的成本，决策正确与否直接关系到司机的利益。

第三问：当交通流量较大时，会出现出租车排队载客和乘客排队乘车的情况。如若不能合理管控，很容易出现上车点交通状况乱、上车时间长的状况，影响交通效率。在乘车区具有两条并行车道的情况下，需要做出合理规划，进而保证司乘安全，提升乘车效率。

第四问：在出租车司机不能选择乘客和拒载的前提下，短途载客意味着司机在完成当前订单后将面临机场附近人流稀少而导致的低上客率。机场出租车管理人员负责“分批定量”方形出租车进入“乘车区”，在这一过程中，管理部门可以对短途载客再次返回的出租车给予“优先权”，减少他们排队载客的机会成本，进而使长途、短途载客的司机之间收益尽量均衡。

1.2　解决任务

(1)　分析送客到机场的出租车司机载客返回市区或直接放空回市区拉客做决策所需要的相关因素，建立决策模型，给出选择策略。

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

(3)　在有两条并行车道的前提下设置“上车点”，合理安排出租车和乘客，使乘车区安全高效。

(4)　为短途载客再次返回的出租车设置“优先权”，使出租车之间收益尽可能均衡。

二　模型的假设

- 出租车收费是与载客行程有关的已知函数。
- 候车区的上车点为质点，市区可看作以市中心为圆心的圆。
- 从机场空载前往市区途中遇客概率较小，在模型中忽略不计。
- 从送客通道前往接客通道时间较短，在模型中忽略不计。
- 出租车行驶时的燃油成本可看作与路程成正比。

三　符号说明

表 1: 符号说明

符号	意义
N_f	机场的航班数
N_c	机场的出租车数
T_1	空载回市区的时间
d_0	机场到市区的距离
T_w	在机场等待载客的时间
P_w	等待收益
λ_1	在市区正常运营时单位时间内收益期望
c	单位时间内行车的物力成本
P_2	载客回市区收益
T_n	在市区正常运营时间
P_n	在市区收益

四　问题分析

问题一　题目要求我们分析出租车司机决策的影响因素、建立模型并给出选择策略。题干已经说明，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，此外在某个季节与某时间段抵达航班的多少和可能乘客的多寡也属于司机的个人经验。当司机回到市区后，环境中打车需求由较低水平恢复至正常水平，此后可以正常运营，无需考虑。若前往蓄车池等待载客回市区，收益为该笔订单的金额，成本为等待载客的时间成本以及载客回市区的时间、人力、物力成本。若直接放空回市区，则可以尽快回到打车需求正常的运营环境，但需要消耗空载回市区的时间、人力物力成本且在此过程中无收益。欲分析司机该做何决策，则需要比较不同决策下的收益与成本之差，也就是净利润，便可解决此问题。

问题二　题目要求我们收集国内某一机场及其所在城市的相关数据，给出方案并分析模型的合理性及对相关因素的依赖性。在这里我们以深圳为例，调查一天内深圳宝安国际机场进港、离港的航班信息及出租车运营数据。本题可处理为问题一的实际应用，收集近期权威数据，整理解析并验证，即可完成题目要求。

问题三　题目要求我们在已有两条并行车道的前提下给出合理方案，使乘车区司乘安全、秩序稳定、运营高效。高效指最大化每个上车点的乘降时间与空闲时间的比值，减少车辆会车导致的低速运行；安全指尽可能避免路线交叉，减少后来空车补位时与刚装载乘客的车相撞的可能性。

设每个乘客上车需要的时间符合均值为 λ 的指数分布，总上车点数目为 N，假设不允许后车补位，待每组 N 台车辆完全通过后再放行新一批次出租车，那么这种情况不存在交叉，定义为“绝对安全”，可以通过 N 计算上车点的使用效率和平均用时。

如果允许后车补位，假设空位发生在第 k 位，则这辆空车（下称“补位车”）上前补位时会依次经过第 1,2，……，k-1 辆车。k 为随机变量，在 1 到 N 均匀分布。定义“会车事件”：补位车的车头与正在乘降的车的车尾相齐开始，到补位车的车尾与正在乘降的车的车头相齐为止，为一次“会车事件”；定义“危险会车事件”：在“会车事件”中，正在乘降的车可能会恰好结束装载准备变道，称此次会车事件为“危险会车事件”。这样，补位车进入 k 乘车位途中发生危险会车的概率 p 越大，模型的危险系数就越高，通过调整 N 的大小，最小化危险系数 p，我们可以得到符合要求的

安全模型。

同理，易求第 k 乘车位空出后到补位车进入乘车位的时间。第 k 乘车位占用时间如前所述是均值为 λ 的指数分布。定义占用时间与空闲时间的比值为第 k 车位的使用效率。则整个乘车区的“平均使用效率” δ 定义为所有乘车位的使用效率的平均值。通过调整 N 的大小，我们可以最大化“平均使用效率” δ，进而获得符合要求的高效模型.

问题四 题目要求我们为短途载客再次返回的出租车设置“优先权”，使长途、短途出租车之间收益尽可能均衡。定义“载客率”为载客时间与行驶时间的比值。在付出同样排队等客的时间成本后，司机接单后的一段时间 t 内（t 大于短途订单所需运营时间，小于长途订单所需的运营时间），接到长途订单的出租车载客率为 1，而接到短途订单的出租车载客率为小于 1 的正数，司机单位时间内的收益出现不均。为解决此问题，可对短途载客后返回机场重新拉客的司机给予优先权，减少他们的时间成本，使得“单位时间内的收益”p：总利润与总时间的比值相等。在此我们分别计算长途和短途的利润期望，以此求出返回后司机应得的等待接客时间，进而为他们设置合理的优先权。

五　模型的建立与解决

5.1　问题一：选择决策模型

5.1.1　机场与市区的模型

机场候车区的上车点为质点，市区可看作以市中心为圆心的圆。

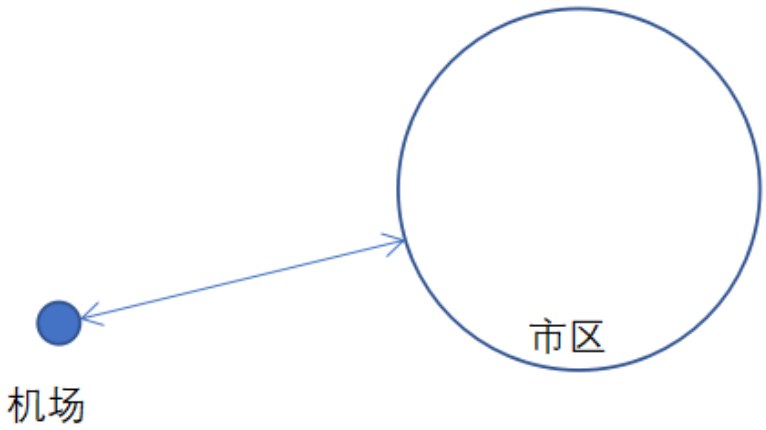


图 1: 机场与市区模型

5.1.2　出租车在时间轴上的收益模型

假设空载回市区所需时间为 T_1，车程为 d, 收益为 P_1；在候车区等候载客时间为 T_w，收益为 P_w；出租车载客到达市区内目的地所需时间为 $T-2$，收益为 P_2，其他正常运营情况下时间为 T_n，收益为 P_n。

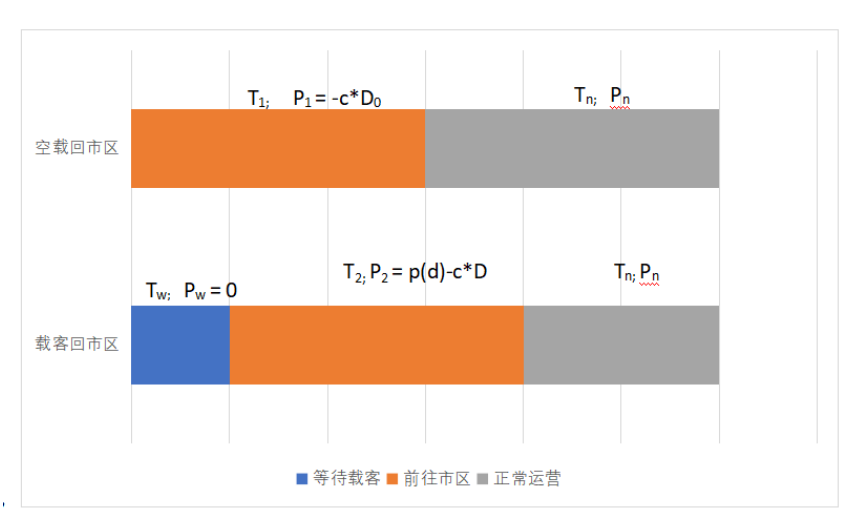


图 2: 收益模型

5.1.3 构建等待时间和收益关系的数学模型

假设当前时刻为 t，航班数量 N_f 及蓄车池内车辆数 N_c 为与 t 有关的函数:

$$N_f = f_1(t) \tag{1}$$

$$N_c = f_2(t) \tag{2}$$

在蓄车池等待接客时间:

$$T_w(x) = \begin{cases} k_1 * N_c & N_p \geq N_c \\ k_1 * N_p + k_2 & N_p \leq N_c \end{cases} \tag{3}$$

其中 k_1 为每乘次平均服务时间 (30s)，k_2 为平均每车等待乘客下一波的时间出租车司机收费为与载客路程有关的分段函数:

$$P = p(d) \tag{4}$$

在市区正常运营时单位时间的利润期望为 λ_1，因此，在送客到达机场后选择空载回市区的全天利润是:

$$P_1 = -c * d_0 + \lambda_1 * (t + T_1 - 7) \tag{5}$$

选择在机场等待载客回市区的全天利润为:

$$P_2 = -c * d + p(d) + \lambda_1 * (t + T_1 + T_w - 7), \tag{6}$$

计算两者的差值:

$$different(t) = P_1 - P_2 = -c(d_0 - d) - p(d) + \lambda_1 * T_w \tag{7}$$

若差值大于 0, 那么选择空载回市区，反之选择在机场等待载客。

5.2 问题二

5.2.1 出租车运营仿真模型

查阅相关资料 [1]，出租车日均运营金额为 1027，除去物力费用和公司提成，出租车在市区正常运营时单位时间内收益期望大约为 λ_1 =35 元。由于深圳已经实现出租车全面电动化，续航为 450km, 充电需要 90 元，因此燃油费由电费代替：c = 0.2 元/km。深圳出租车收费标准为 10 元/2km, 里程价为 2.6 元/km.[2] 出租车司机工作时间为 7:00am-7:00pm。出租车空载回深圳市中心（福中路）需要 40 分钟，路程为 d_0=36km。

深圳市现有数据表明市内有 53330 辆网约车，21789 辆巡游出租车，所以乘客选择坐出租车的概率 p 为 21789/(21789+53330)=0.29，由资料可知 [3] 有 46.6% 的乘客下飞机选择坐网约车或者出租车，那么就有 0.29*46.6%=13.5% 的人选择坐出租车离开机场。

统计得深圳宝安国际机场某日（7：00-19：00）每小时降落航班数 N_f(t)，如图 3。

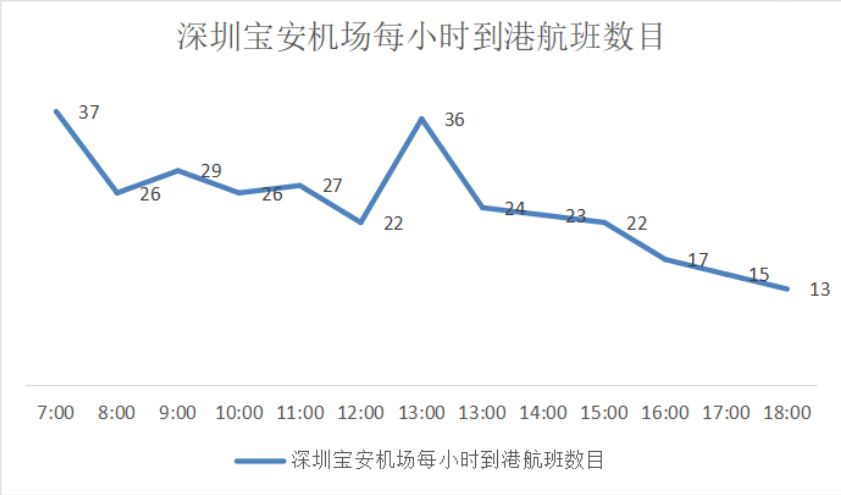


图 3: 降落信息

平均每架飞机坐 150 人,那么就可以得到平均每小时要坐出租车的人数,即 N_p=0.135*150*N_f(t) 人/h。该机场内每小时出租车数量 N_c[4] 分布如图 4。

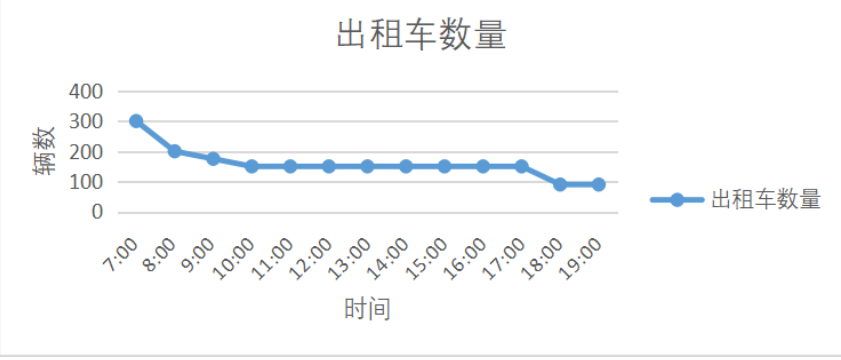


图 4: 出租车数量

在 t 时刻，出租车容纳量（设每辆车平均坐两个人）和乘客的数量关系（dff=N_p-2*N_c）如图 5 所。

时间	出租车数量	机场每小时到港航班数目	到达机场的人数估测	乘坐出租车的人数N_p	出租车最大容纳量与乘客数量关系
7:00	300	37	3700	499.5	-100.5
8:00	200	26	2600	351	-49
9:00	175	29	2900	391.5	41.5
10:00	150	26	2600	351	51
11:00	150	27	2700	364.5	64.5
12:00	150	22	2200	297	-3
13:00	150	36	3600	486	186
14:00	150	24	2400	324	24
15:00	150	23	2300	310.5	10.5
16:00	150	22	2200	297	-3
17:00	150	17	1700	229.5	-70.5
18:00	90	15	1500	202.5	22.5
19:00	90	13	1300	175.5	-4.5

图 5: N_c 与 N_p 数量关系

由图可知，当容纳量大于乘客数量时，dff 数值为负数，且可以求得航班数与蓄车池内出租车数量关系为:

$$\frac{N_f}{N_c} \quad < 0.148$$

某司机在 t 时刻来到机场，若出租车容纳量大于乘客数，则每辆车平均等待下一波乘客的时间 k_2 设为 0.5h，那么他的等待时间为

$$T_w(x) = \begin{cases} 30 * N_c & dff \geq 0 \\ 30 * (N_p/2) + k_2 & dff \leq 0 \end{cases} \tag{8}$$

表 2: t 时刻对应等待时间

到达时间 t	等待时间 T_w(h)
7:00	2.60
8:00	2.00
9:00	1.50
10:00	1.25
11:00	1.25
12:00	1.70
13:00	1.25
14:00	1.25
15:00	1.25
16:00	1.70
17:00	1.50
18:00	0.75
19:00	1.23

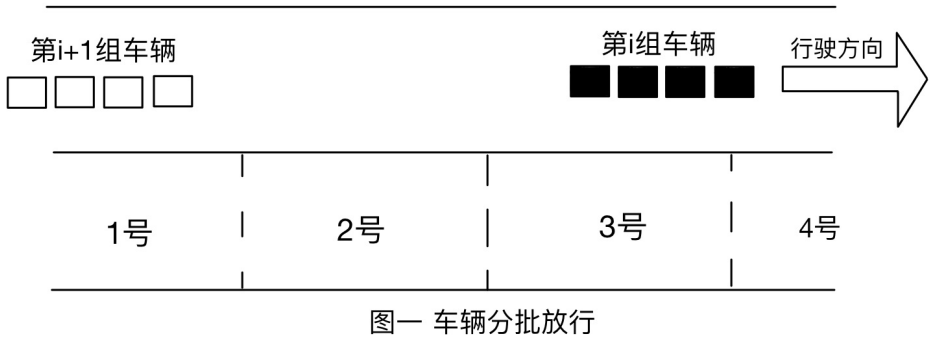
通过代入数据得到两种决策收益差值为:

$$different(t) = -p(d) + \lambda_1 * T_w = -(10 + 2.6 * 34) + 35 * T_w \tag{9}$$

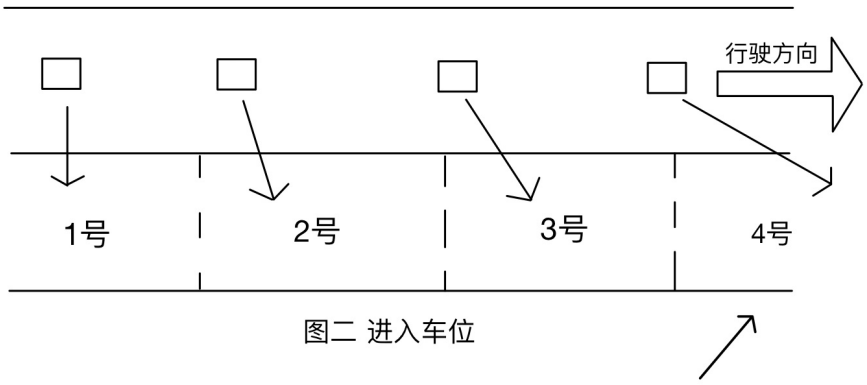
当 different(t)>0 时，求得 T_w>2.8h，司机应该选择空载回市区，否则选择留下来待客。对于司机来说，当到港航班数与出租车数之比小于 0.148 时，出租车总容纳量小于乘客数量，需要等待的时间更长一些，当航班数大于 41 且出租车小于 280 辆时，等待时间 T_w>2.8h，司机选择空载回市区更明智。其他大部分情况选择留下来载客一天的盈利更多。

5.3 问题三：排队模型

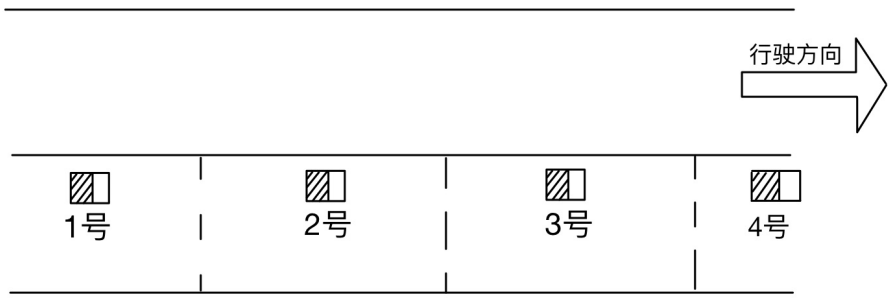
5.3.1 绝对安全上车模型



图一 车辆分批放行



图二 进入车位



图三 乘客上车

图 6: 绝对安全上车模型

假设不允许后车补位，待每组中 N 台车辆完全通过后再放行新一批次出租车，那么这种情况不存在车辆路线交叉，定义为“绝对安全”，可以通过 N 计算上车点的使用效率和平均用时。由于计算的是“绝对安全”的时间，只有前面一组完全通过，后面一组才可放行，每组通过的时间为行

车时间与本组中最长上客时间之和。设其中最长的上客时间为 T，则

$$P(T_{max} < t) = P(T_1 < t, T_2 < t, ..., T_N < t) = \prod_{i=1}^{N} PT_i < t = (\int_0^t \frac{1}{\lambda} e^{-\frac{x}{\lambda}} dx)^N = F_{Tmax}(t) \quad (10)$$

故

$$f_{Tmax} = F'_{Tmax}(t) = N(\int_0^t \frac{1}{\lambda} e^{-\frac{x}{\lambda}} dx)^{N-1} \frac{1}{\lambda} e^{-\frac{t}{\lambda}} = N(1 - e^{-\frac{t}{\lambda}})^{N-1} \frac{1}{\lambda} e^{-\frac{t}{\lambda}} \quad (11)$$

因此，在“绝对安全的情况下，每辆车平均接客时间：

$$\frac{1}{N} \int N(1 - e^{-\frac{t}{\lambda}})^{N-1} \frac{1}{\lambda} e^{-\frac{t}{\lambda}} t dt \quad (12)$$

乘车位平均使用效率的计算 已设车身长 5m，停车点之间的距离（车头到车头）为 20m，并行车道内限速为 9m/s，每个乘客上车需要的时间服从均值为 λ =30s 的指数分布，总上车点数目为 N。因此易求从第 k 乘车位空出后到补位车进入乘车位的时间: T_0=20(k-1)/9s, 也就是此乘车位的空闲时间。此乘车位的使用时长为 T_1，T_1 服从均值为 λ 的指数分布。

当所有司机都以安全范围内尽可能最快的速度运行（这一点切合他们的利益，符合生活实际）且乘客上车的时长条件不变，并行车道效率高低就取决于乘车位的使用效率。其中，第 k 个乘车位的使用效率为 δ_k=$\frac{T_1}{T_0+T_1}$.

将 T_0 与 λ 代入, 有

$$\delta_k = \frac{27}{25 + 2k} \quad (13)$$

因此，在整个乘车区中，每个乘车点的平均使用效率：

$$\overline{\delta} = \frac{1}{N} \sum_{k=1} N \frac{27}{25 + 2k} \quad (14)$$

安全系数的计算 假设空位发生在第 k 位，则补位车上前补位时会依次经过第 1,2，……，k-1 辆车。k 为随机变量，在 1 到 N 均匀分布。定义“会车事件”：补位车的车头与正在乘降的车的车尾相齐开始，到补位车的车尾与正在乘降的车的车头相齐为止，为一次“会车事件”；定义“危险会车事件”：在“会车事件”中，正在乘降的车可能会恰好结束装载准备并道，称此次会车事件为“危险会车事件”。

由前面得，当补位车经过第 i 辆车时，乘车点有车的概率：

$$P_i = \delta_i = \frac{27}{25 + 2i} \quad (15)$$

若乘车位有车，则在此乘车位旁会车时间为 5m/(9m/s), 且每个乘客上车需要的时间服从均值为 λ=30s 的指数分布。因此此乘车位中出租车刚好结束装载，准备并道的概率是 P(会车期间第 i 个乘车位中的车准备并道 | 会车期间第 I 个乘车位中有车）=$\frac{1}{30} * \frac{5}{9} = \frac{1}{54}$. 所以在补位车路过第 i 个乘车位时内部载客车有并道倾向，也就是在第 i 个乘车位有发生危险的可能为

$$P_i = \frac{27}{25 + 2i} * \frac{1}{54} = \frac{1}{50 + 4i} \quad (16)$$

因此，在第 k 个乘车位的空车补位过程中，一路上没有安全隐患的概率为

$$P_k = \prod_{i=1}^{k-1} 1 - \frac{1}{50 + 4i} \quad (17)$$

在 N 辆出租车分别于整片乘车区所有乘车位上完成补位载客的过程中，一路上没有安全隐患的概率为

$$P_{safe} = \prod_{k=1}^{N} P_k \quad (18)$$

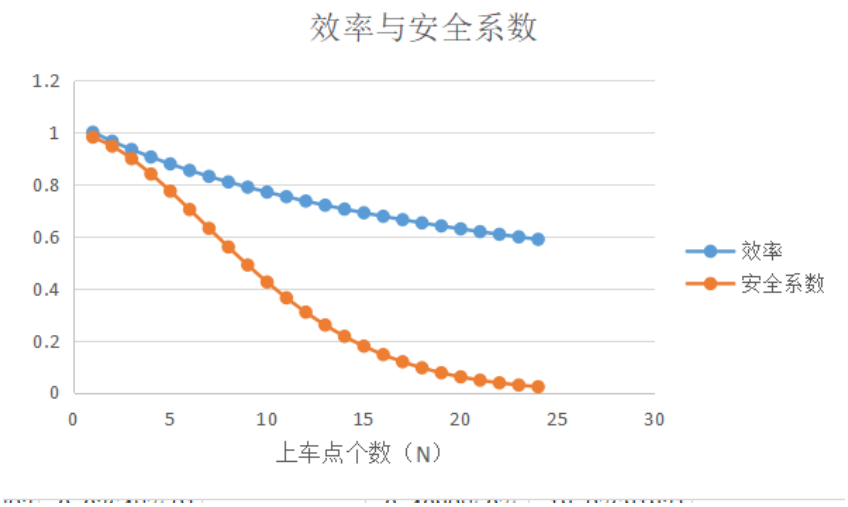


图 7: 效率与安全系数

根据 python 编程得出的结果和图像分析，当设置上车点数目位 4 时，乘车效率较高且安全系数也较高。

5.4 问题四

机场短途司机的“优先”安排模型 假设司机接到短途订单后回到机场。设在普通情况下，司机所需等待时间为 t_1。假设机场人员将接到短途订单 (事件 A_x) 的司机排队时间缩短为 $t_2(t_2 \leq t_1$（事件 B)，下一笔订单收益的期望 E(B), 此时与未接到短途订单（事件 C）的司机收益均衡，则二者收益期望与二者盈利时间有如下关系:

$$\frac{Profit(A_x) + E(B)}{t_A + t_2 + t_B} = \frac{E(C)}{t_1 + t_C} \tag{19}$$

假设接到短途订单（事件 A）的概率为 p，则 E(B)=p*E(A)+(1-p)*E(C). 代入上式, 则

$$t_2 = \frac{t_1 * Profit(A_x)}{E(C)} + (1-p) * t_1 - t_A \tag{20}$$

其中

$$E(C) = \int_{d_{short}}^{\infty} pdf(x) * price(x)dx \tag{21}$$

则

$$t_2 = \frac{t_1 * Profit(A_x}{\int_{d_{short}}^{\infty} pdf(x) * price(x)dx} + (1-p) * t_1 - t_A \tag{22}$$

六 模型的优缺点

6.1 模型的优点

1. 对影响司机收益的参数进行了有效量化，求解速度较快。
2. 模型充分利用了现实生活中可获取到的信息，与现实生活联系紧密，便于推广。
3. 最终通过比较不同决策的期望确定了司机预估等待时间的最大上限。
4. 模型对到客速率较小的情况作出了分析，提出了相应的优化方案。

6.2 模型的缺点

1. 出租车运营模型中，在一天内不同时段，由于人流密度不同，出租车两单间的空驶距离不同，因此一天不同时段单位时间获得收益期望不同。由于未搜寻到精确的数值，在本次运用模型中作常数处理。

2. 司机选择空载回市区或等待载客回市区可能获得的收益波动不同，即方差不同。但由于缺少大量的统计数据，在本次运用的决策模型中只比较了二者的期望大小。

七 附录

```
# 运行约需10s
#### 标准库
from collections import defaultdict
#### 第三方库
import numpy as np


#### 定义常数
n = 0 # 上车点个数
l_c = 5.00 # 车辆长度
d_c = 20.00 # 上车点间距
t_s = 30.00 # 服务时间期望值为30s
v = 30/3.6 # 安全车速
t = 0.01 # 模拟精度为 0.01s

#### 定义全局变量
current_code = 1 # 为每一辆入场的出租车编号，每次加一
conflicts = set() # 记录已发生的冲突
remaining_service_time = {} # 每个上车点此刻剩余的服务时间
cars_on_road = {} # 键为目标上车点，值为目前所在位置，若已载客，目标上车点标为0
end_time = 1000 # 模拟终止时间


#### 定义函数
# 根据指数分布随机生成服务时间
def random_service_time(t_s=t_s):
    return np.random.exponential(t_s)

# 模拟情境初始化
def init_start(N, v=v,d=d_c):
    global current_code, conflicts
    conflicts = set()
    for idx in range(N):
        # idx+1同时为第一批车每辆的编号
        current_code = idx + 1
        remaining_service_time[idx] = [random_service_time(), current_code]

    return remaining_service_time
```

```
# 开始模拟
# 假设：每辆车起点距离为0，第一个上车点距离为 l_c（车辆长度）
def simulate(N, l_road, d=d_c, t=t, v=v):
    global conflicts, current_code
    remaining_service_time = init_start(N=N)
    current_time = 0
    while True:
        # 若达到模拟时间，返回结果
        if current_time >= end_time:
            return current_code - len(conflicts)
        # 每次推进单位时间（0.01s）
        current_time += t
        # 更新所有车辆距离
        for k in cars_on_road.keys():
            # 若不是已载客车
            if k != 0:
                try:
                    cars_on_road[k][0] += v * t
                except:
                    pass
            # 若是已载客车
            else:
                if k in cars_on_road.keys():
                    for car in cars_on_road[k]:
                        if type(car) == list:
                            car[0] += v * t
                        else:
                            cars_on_road[k].remove(car)
                else:
                    pass

        for k in remaining_service_time.keys():
            if type(remaining_service_time[k][0]) == float:
                # 剩余服务时间相应减少
                remaining_service_time[k][0] -= t

        # 将已驶离出候车区的车移出路面
        remove_out_cars(l_road)
        # 将已完成服务的车移回行驶通道，并发派补位车
        done_service()
        # 将到达目标上车点的补位车填入服务车道
        move_cars_to_service()
        # 计算每辆车是否发生危险会车事件
        calculate_conflicts()
    return



# 更新事件：出租车驶离候车区
def remove_out_cars(l_road):
    # 若道路上无载客车
    if 0 not in cars_on_road.keys():
        return
    else:
        # 对所有已载客车
        for car in cars_on_road[0]:
            if type(car) == list:
                if car[0] >= l_road: # 当距离大于通道长度时
                    # 将车移出通行通道
                    cars_on_road[0].remove(car)
                # print(car)
        if len(cars_on_road[0]) == 0: del(cars_on_road[0])
    return

# 更新事件：出租车到达目标上车点，开始服务
def move_cars_to_service(d_c=d_c,t_s=t_s):
    for k in list(cars_on_road.keys()):
        if k == 0: # 跳过已载客，正在驶离的车
            continue
        else:
            v = cars_on_road[k]
            if v[0] >= l_c + d_c * (k - 1): # 若在0.01s内，该车到达服务点
                del(cars_on_road[k]) # 将该车移出通行通道
                # 更新该上车点服务时间，及正在服务的车的编号
                remaining_service_time[k] = [random_service_time(), v[1]]
    return
```

```
# 更新事件：服务完成
def done_service(l_c=l_c, d_c=d_c):
    global current_code
    undefined = 99.99 # 代表上车点空闲中
    for k, v in remaining_service_time.items():
        # v[0]为剩余服务时间，v[1]为出租车编号
        if v[1] != 0:
            try:
                if v[0] <= 0:
                    # 如果路上没车
                    if any_car_on_road() == False:
                        remaining_service_time[k] = [undefined, 0]
                        current_code += 1
                        cars_on_road[k] = [0, current_code]
                    else:
                        # 最后发车的车离起点距离
                        last_car = undefined
                        # 更新k点服务状态：空闲中
                        remaining_service_time[k] = [undefined, 0]
                        # 载客完成的车开始驶离
                        if 0 in cars_on_road.keys():
                            cars_on_road[0].append(  [l_c + d_c, v[1]]  )
                        else:
                            cars_on_road[0] =[  [ l_c + d_c, v[1] ]  ]
                            # 若此时上一辆发车还未完成（即驶出l_c/v距离）
                            if last_car <= 0:
                                current_code += 1
                                # 则此辆车排在上一辆车后发车
                                cars_on_road[k] = [last_car - l_c, current_code]
                            # 若上一辆车已完成发车
                            else:
                                current_code += 1
                                # 则直接从起点发车
                                cars_on_road[k] = [0, current_code]
            except: pass
    return



# 更新事件：路上每辆车是否发生危险会车事件
def calculate_conflicts(l_c=l_c):
    global conflicts
    heading_cars = [] # 已载客车列
    leaving_cars = [] # 未载客车列
    for goal, car in cars_on_road.items():
        if goal != 0:
            if len(car) > 1: heading_cars.append(car)
        else:
            if len(cars_on_road[goal]) > 0:
                leaving_cars = cars_on_road[goal]

    for c1 in heading_cars:
        d1, code1 = c1[0], c1[1]
        for c2 in leaving_cars:
            try:
                d2, code2 = c2[0], c2[1]
                # 距离过近则用集合记录发生
                if abs(d1 - d2) < l_c:
                    conflicts.add(tuple((code1, code2)))
            except: pass
    return

def any_car_on_road():
    for k in cars_on_road.keys():
        if len(cars_on_road[k]) > 0:
            return True
    return False


if __name__ == "__main__":
    results = []
    for n in range(2, 20):
        # 返回评价函数
        l_road = l_c + (n-1) * d_c # 候车区道路长度
        current_code = 0
        goodness = simulate(n, l_road)
        # 存储评价结果
        results.append(tuple((goodness, n)))
    # 从好到坏进行排序
    results.sort(reverse=True)
    print(results[0][1])
```

参考文献

[1] 《深圳网约车司机赚的多吗？市交通局：日均每车订单运营金额约为 296.2 元》, 深圳新闻网, 2019

[2] 2019 年深圳出租车最新收费标准一览, 深圳本地宝, 2019

[3] 孙昊，单铮，朱晶. 基于 NL 模型的机场旅客陆侧交通方式选择行为研究, 科学技术创新,2019

[4] 刘丽, 张丰, 杜震洪, 刘仁义, 贾玉杰. 基于深圳市出租车轨迹数据的高效益寻客策略研究 [J]. 浙江大学学报 (理学版), 2018