基于最优化的出租车补贴方案设计模型

摘要

本文针对出租车时空安排不合理导致的“打车难”问题，对形成因素进行分析，利用排队论的方法建立乘客出租车排队模拟模型，从乘客和公司的角度出发，建立多目标优化模型，通过模拟模型进行验证，设计出了最佳补贴方案，能够有效的缓解“打车难”的问题。

建立模型之前，通过滴滴平台提供的“苍穹”数据库网站，收集到了北京市 2016 年 8 月 10 日所有的出租车数据。通过投点分析，北京五环外的数据较少，所以只考虑五环内的数据，并通过四环、三环和二环将五环内的区域划分成四块。

针对出租车“供求匹配”程度分析的问题，首先分析了三个影响“供求匹配”程度的因素：单位车辆需求人数、乘客平均满意度和平均等待时间。然后，利用熵值法分析三个影响因素的权重，建立了合理的评价指标模型，得出了具体的评价指标计算公式。以北京市 2016 年 8 月 10 日为例分析，将得到的数据带入计算公式得到所有区域所有时刻的评价值。最后，通过比较相同时间不同区域、相同区域不同时间的评价值分析了出租车资源“供求匹配”程度的情况：北京市早高峰时间二环到三环以及三环到四环的“供需匹配情况”较差，这是上班族集体出行导致的；二环到三环一整天出现了三次较严重的“供需不匹配”情况，是由于早高峰、下午晚上集体出行导致的。符合现实情况。

针对出租车补贴政策对缓解“打车难”问题有效性的问题，首先分析了打车难的原因，收集了“滴滴”和“快的”两家公司的补贴政策，进行初步的比较分析，建立了排队模型。接着利用 MATLAB 编写出租车乘客的排队系统模拟模型，以北京市 8 月 10 日”打车难“问题最明显的第 8 个时间段进行分析，通过泊松流来模拟乘客和出租车的到达情况，利用排队论的方法来模拟整个打车系统的运行情况。最后通过分别改变出租车和乘客的到达强度，以此来模拟补贴前后的情况。分析后得知，出租车公司对乘客的补贴对缓解”打车难“问题没有帮助，而补贴司机对缓解”打车难“问题有帮助。

为了重新设计补贴方案，首先分析了目前各家公司补贴方案的弊端，将补贴方案分为初期推广阶段和后期稳定运营阶段两个阶段：在初期推广阶段，针对不同时间、不同区域进行不同力度的补贴；后期稳定运营阶段主要是调整出租车不同区域分布比例，根据不同时间、不同区域、调度金额、调度距离、公司收益以及司机体验方面建立对司机主动调度、被动调度的弹性补贴方案。对于乘客对于乘客拼车角度，同样从等待时间和公司支出出发建立多目标优化模型。对司机补贴根据地区不同而不同，最终方案为：对北京 8 月 10 日早高峰时段：二环内每单补 7 元、二三环间、三四环间、四五环间每单补 1 元；晚高峰时段：二环内每单补 5 元，二三环间每单补 1 元，三四环间每单补 3 元，四五环间每单 8 元。对拼车乘客的补贴方案为早晚高峰时期分别补贴 57% 和 49%。其余补贴方案见正文。

关键字： 熵值法　排队论　系统模拟　多目标优化

一、 问题重述

随着经济的不断发展，出租车渐渐成为了市民出行的重要交通工具之一，可时常碰到的问题便是出租车“供不应求”。随着近几年“互联网”的发展，网约车逐渐占领了出租车的大部分市场。通过互联网，网约车服务平台实现了乘客与出租车之间的信息交流，同时推出了多种网约车补贴方案。通过收集相关数据，建立数学模型解决以下问题：

1. 建立合理的指标，并分析不同时空出租车资源的“供求匹配”程度。
2. 分析各公司的出租车补贴方案是否对“缓解打车难”有帮助？
3. 如果创建一个新的打车软件服务平台，应怎么样设计补贴方案，并论证其合理性。

二、 问题分析

互联网给出租车行业带来了天翻地覆的变化，而网约车平台如何利用好资源，如何设计良好的补贴方案便成为平台能否长期运营的关键。

建立评价指标、分析补贴方案的前提便是有足够的数据进行分析，所以首先需要从互联网获取出租车的数据。可以从滴滴出行的“苍穹”数据库网站进行数据的获取。根据三个问题的要求，需要首先对目标城市进行区域划分和时间划分，时间可以以一天 24 个时间段划分。空间划分则可以先通过将得到的数据投点，作热力图，筛选出有效数据并选取边缘区域，根据具体情况便可以进行划分。

第一问需要建立合理的评价指标和分析出租车“供需匹配”程度。首先，可以从影响“供求匹配”程度的因素下手，分析出主要的影响因素。接着利用收集到的数据，通过合适的评价体系建立方法建立评价模型。最后将所有的数据带入评价体系求解，比较相同时间不同区域、相同区域不同时间出租车“供求匹配”程度的不同，结合实际情况分析即可。

第二问要求分析各公司补贴方案的有效性。首先需要收集各公司的补贴方案，分别探究讨论其对乘客的补贴方案和对出租车司机的补贴方案，对这些公司的补贴方案对”打车难“问题的影响做一个初步的分析。然后分析乘客数量、出租车数量与补贴金额定量的关系，通过这些关系得出乘客等待时间的表达式。接着可以建立乘客与出租车体系的实际运行模型，通过建立模拟系统能够有效定量分析出补贴前后的影响。最后，将补贴前后利用模拟系统计算得到的乘客平均等待时间进行比较，得出最终结论。

第三问要求设计新的补贴方案并验证合理性。首先可以分析目前各家公司补贴方案存在的弊端，并且结合这些方案再推广期间所展现出的优点，根据具体情况得出推广初期的补贴方案。然后接着再讨论正常运营期间的合理补贴方案。分别考虑对司机和乘客的补贴，最能够缓解“打车难”问题的安排便是与各地区需求比率相同，所以可以通过尽量向最大匹配的方案靠近；但是也要从公司的角度出发考虑支出问题，对拼车角度也是一样的考虑。最后结合问题寻找合适的约束条件，建立优化模型求解，带入第二问的模型得出等待时间的变化并进行比较论证。

三、 模型假设

1. 假设得到的滴滴打车的数据真实可靠。
2. 假设 2016 年 8 月 10 日，北京举办的大型活动等不对出租车运营造成影响。
3. 假设乘客上车下车时间忽略不计。
4. 假设拼车的乘客在同一地点上车。

四、 变量说明

变量名称	含义
x_{ij}	第 i 个区域或时间段第 j 个影响因素对应的数值
a	单位车辆需求人数或等待时间或满意度
P_{ij}	第 j 个影响因素下第 i 个区域或第 i 个时间段 A_i 的贡献度
W_j	各影响因素的权重
F_i	“供求匹配”程度的评价指标值
α	单个地点乘客队列计算系数
β	单个地点空载出租车队列计算系数
λ_1	乘客到达强度，即单位时间乘客的平均到达数
λ_2	空载出租车到达强度，即单位时间空载出租车的平均到达数

五、 建模准备

5.1 数据收集

出租车的出现，极大的方便了市民日常出行。但是由于车辆及人口流动的原因，经常会出现打车难的问题。因此而生的“互联网 +”出租车，借助了智能手机的便利，成功将出租车行业提高到一个新高度。网约车有着比传统出租车得天独厚的优势：统一的平台管理，对于乘客和司机的安全都有良好的保障；智能化的车辆调度，能够有效控制每个区域的出租车数量，并且能够有效提高乘客打车的成功率。由于网约车的发展极其迅速，所以在研究出租车数据时便可以以网约车的相关数据来代替整体出租车的数据。

为了研究不同时空出租车资源的“供求匹配”程度以及合理的补贴方案，以首都北京为例，通过查询滴滴打车提供的数据库网站——“苍穹滴滴快的智能出行平台”，获取北京市 2016 年 8 月 10 日多个地点出租车的信息，然后进一步进行建模分析。

查询到的数据类型如下：

表 **1**　出租车数据结构说明

数据名称	说明
打车需求量（demand）	某日某地某时间段需要乘坐出租车的人数（需求量）
出租车分布（distribute）	某日某地某时间段内出租车的数量
车费（money）	某日某地某时间段订单成交总价
被抢单时间（response）	某日某地某时间段乘客下单（打车）到司机接单的时间
满意度（satisfy）	某日某地某时间段乘客能被接单的难易程度

5.2 数据清洗

为了更加准确的对所收集到的数据进行分析，需要对不符合实际情况的数据进行剔除。首先将收集到的所有数据在地图上按照经纬度标出，如图 1。

图 **1**　原始数据所对应位置

由上图可以看出存在多组数据偏离北京范围，对此部分数据进行剔除：

北京市的大致经纬度范围为:西北角:40.164499°N,116.055207°E;东南角:39.697996°N,116.71576°E。经纬度在这两个区间的数据认为是北京市合理的数据，保留；对于经纬度不在这个范围内的数据，认为有所偏差，将其剔除。

5.3 数据分析

通过数据清洗所得到的北京市的数据，首先对其出租车分布大致情况进行分析，绘出 10 日北京市 24 小时出租车需求总量曲线图，如图 2。

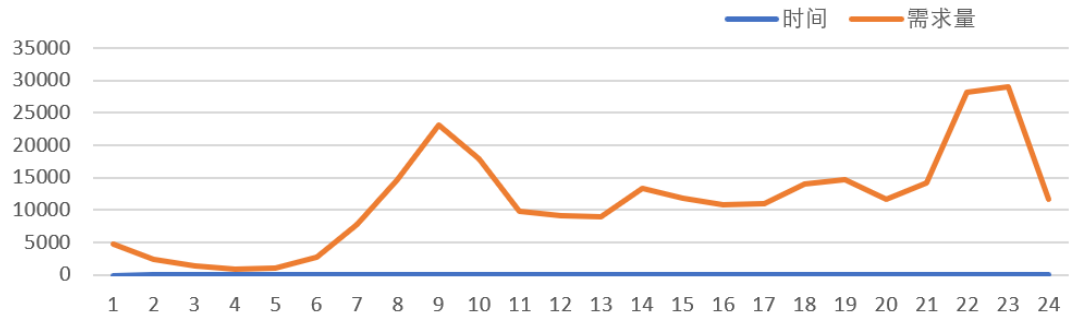


图 **2**　北京市 **2016** 年 **8** 月 **10** 日 **24** 小时出租车需求总量

由图 1 观察到出租车需求量由凌晨的低谷到早高峰 8 点突然增高，之后有所下降趋于平稳。在 17 点左右，出租车需求量增高，这是因为晚高峰，上班族下班回家有需求。晚上人们出行娱乐、回家等因素导致一个深夜高峰，之后的需求量下降趋于平稳，直到凌晨达到低谷。

六、 模型的建立与求解

6.1 问题一

出租车由于流动性较大，随机性比较高，经常导致“供求匹配”程度不足或者过剩的情况。为了解决这类问题，首先需要寻找对“供求匹配”影响的因素，接着使用恰当的方式建立评价指标，对相同空间不同时间、不同空间相同时间出租车“供求匹配”程度进行评价分析即可。

6.1.1 模型准备

数据预处理

通过调用百度地图 API 做出了出租车的热力分布图，代码见附录 A，如图 3。

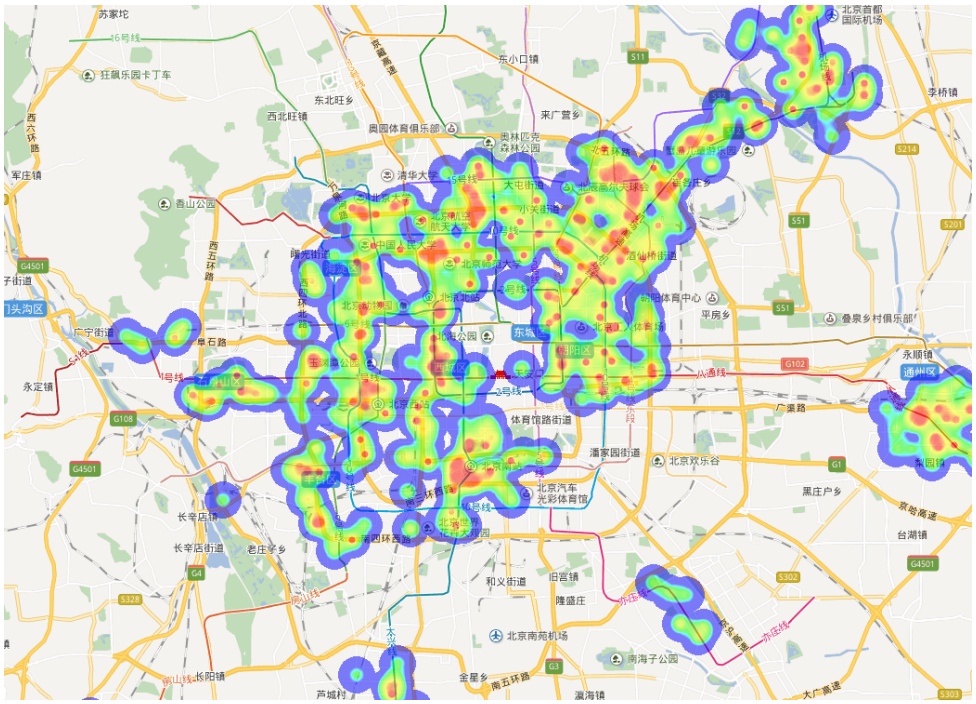


图 **3**　北京 **2016** 年 **8** 月 **10** 日 **8** 时出租车的热力分布图

由图 3，可以观察到北京市出租车需求量的分布集中在五环以内，五环以外的需求量锐减了很多，所以我们将研究区域限定在北京五环以内，将五环以外的数据筛除。由于北京的环城路近似于矩形，于是便可以通过五环西北角和东南角的经纬度来筛选数据，五环经纬的范围如下表：

表 **2**　五环经纬度范围

	二环		三环		四环		五环	
	西北	东南	西北	东南	西北	东南	西北	东南
北纬	39.9453	39.8725	39.9646	39.8590	39.9780	39.8373	40.0003	39.8076
东经	116.3525	116.4389	116.3037	116.4528	116.2738	116.4762	116.2290	116.5171

为了研究不同地区出租车“供求匹配”程度，按照上表数据，将北京市以五环为边

界，将北京分为二环内、二环到三环、三环到四环以及四环到五环四块不同的矩形区域。划分的区域如图 4。

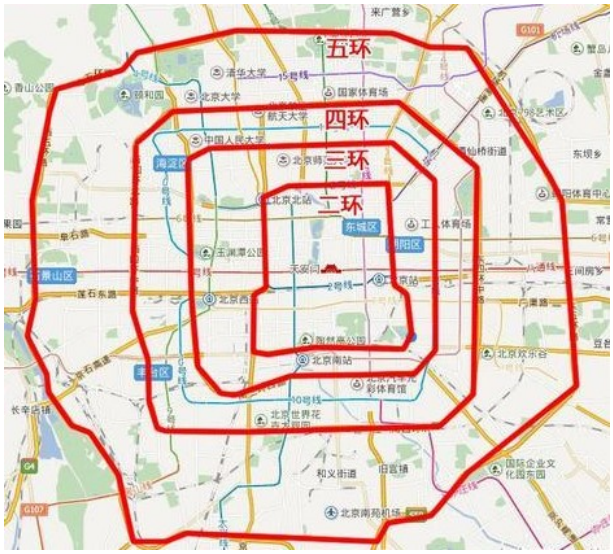


图 **4**　北京环城公路区域划分图

订单费用计算方法

出租车订单费用由路程费用和时长费用组成。通过百度地图，收集到了四个区域中相邻区域的距离数据 d_j 以及对应地区的车辆平均行驶速度 v_i。通过咨询，得到了出租车的具体收费标准如下：

表 **3**　滴滴快车普通型收费标准

时间段	里程收费标准 a_i	时长收费标准 b_i
0：00-6：00	2.4 元/km	0.3 元/min
6：00-23：00	1.6 元/km	
23：00-00：00	2.4 元/km	

于是，订单的单位时间费用可以表示为：

$$s_i = (d_j \times a_i + \frac{d_j}{v_i} \times b_i)/(\frac{d_j}{v_i})$$

$$s_i = v_i a_i + b_i \tag{1}$$

利用上述公式，可以求出每个地区的乘车单位时间费用：

表 **4**　出租车不同区域乘车单位时间费用

地区	二环内	二环到三环之间	三环到四环之间	四环到五环之间
单位时间费用（元/min）	0.6627	0.724	0.7747	0.884

6.1.2 供求匹配程度影响因素分析

出租车运力规模是否合理，不仅直接关系到出租车经营者的利益，而且对于出租车行业资源优化配置、城市居民的出行质量、整个城市交通系统的协调发展以及和谐社会的构建都具有非常重要的意义。

评价城市出租车供给量与需求量的关系一般有以下三大指标：

（1）单位车辆需求人数 [1]

单位车辆需求人数是指城市或某一特定区域有打车需求的人数与该区域出租车现有量之比，这一指标显示了不同地区出租车相对量的多少。

$$\text{单位车辆需求人数} = \frac{\text{总需要打车的人数}}{\text{总出租车数}}$$

单位车辆需求人数的倒数为出租车人均需求量，目前对城市出租车人均需求量的控制标准没有确定其上限，现行的《城市道路交通规划设计规范》只提供了出租车人均需求量的下限：即大城市不低于每万人 20 辆，小城市不低于每千人 0.5 辆，中等城市在之间取值。

（2）乘客平均满意度

乘客满意度是指乘客对打车难易以及行驶过程体验的直观感受。通过乘客每次乘车后对行程的评价，可以间接的反应出出租车的供求程度。如果出租车的服务质量较低，乘客乘坐出租车就不满意，一部分乘坐出租车的乘客就会选择其他的出行方式。这对出租车的供需匹配程度的影响也是很大的。

由于得到的数据乘客的满意度与打车需求量的数据不同，不能够通过这两组数据求出乘客的平均满意度，所以需要将不同地点的数据通过合并的方式处理，具体从处理方法，即乘客平均满意度的算法如下：

Step1 把五环内看作一个 200×200 的网格面，每个网格的边长约为 1.1km。

Step2 对于单个时间点，将得到的多组乘客满意度数据通过经纬度在网格上投点，将同一时间点乘客需求量数据也在网格上投点。

Step3 以每一个网格为一个新的数据，得到新的数据组。其中，每个数据的经纬度为对应网格的中点坐标，出租车需求量为对应网格内需求量的之和，乘客满意度为对应网格内乘客满意度的平均值。

Step4 每个数据通过其中点坐标确定其所属区域。则每个区域的平均满意度则可以通过以下公式求解：

$$\text{乘客平均满意度} = \frac{\sum \text{单个网格乘客满意度} \times \text{单个网格内出租车需求总量}}{\text{此区域出租车需求总量}}$$

由此，便得到了单个时刻单个区域乘客平均满意度，具体结果见附录 C。

（3）乘客平均等待时间

乘客平均等待时间在一定程度上反映了出租车的服务质量的高低，指的是乘客在有出租车需求时到出租车到来时的这段候车时间。由于收集到的数据与满意度遇到的问题相同，所以使用了相同的网格化重新构造数据的方法求解出了乘客平均等待时间，单个

网格的等车时间为对应网格内等车数据的平均值，具体结果见附录 C：

$$\text{乘客平均等待时间} = \frac{\sum \text{单个网格乘客等待时间} \times \text{单个网格内出租车需求总量}}{\text{此区域出租车需求总量}}$$

这是显示出租车服务水平高低、出租车工序是否合理的直观指标。乘客平均等待时间越短，说明出租车数量充足，乘客当然希望等待时间越短越好，但是也不能通过肆意增加出租车数量来减少其等待时间。如果出租车数量过多，必然导致出租车运力资源的浪费以及出租车行业的恶性竞争；而当乘客平均等待时间过长，就容易出现司机拒载、乘客因打不到车耽误正常生活、工作的情况，这就需要增加出租车供给量。因此，把乘客平均等待时间控制在一个合理的区间，能体现城市出租车处于科学的发展规模。

（4）出租车空驶率

出租车空驶率一定程度上反应了乘客与出租车的比率。出租车的空驶率表示某个时间段内空载出租车占总出租车数量的比率，空驶率越低，说明出租车运营效率越高，但同时也说明了出租车的数量可能存在不足，反之，则说明出租车数量充足。其计算方法如下：

$$\text{各环出租车空驶率} = 1 - \frac{\text{各环出租车载客总里程}}{\text{出租车行驶总里程}}$$

出租车的载客总里程数可以由收集到的每单订单价格和单位里程的费用求出；行驶总里程可以通过出租车数量乘上平均速度以及单位时间得到：

$$\text{各环出租车载客总里程} = \frac{\text{各环订单总价}}{\text{单位里程费用}}$$

$$\text{出租车行驶总里程} = \text{出租车数量} \times \text{各环平均行驶速度} \times \text{时间段长度}$$

单位里程的费用由里程长度费和时长费构成：

$$\text{单位里程费用} = \text{每公里里程费用} \times \text{时长费} \times \frac{\text{里程长度}}{\text{平均各环速度}}$$

6.1.3 供需匹配评价模型的建立

通过上面对供需匹配程度影响因素的分析，便可以利用这三个主要的影响因素建立供需匹配程度评价模型。为了尽量减少和避免权重确定过程中的主观因素及某些客观局限, 对此问题采用熵值法对各指标进行赋权, 在此基础上利用加权求和方法对出租车供需匹配进行测度评价。[2]

熵值法的原理

熵最初来源于物理学中的热力学概念, 主要反映系统的混乱程度, 现已广泛应用于可持续发展评价及社会经济等研究领域。在信息论中, 熵是系统混乱程度的度量, 而信息则是有序程度的度量, 二者绝对值相等，符号相反。

在由 n 个待评方案 (本问中指 4 个不同区域和 24 个不同时间段构成的 96 个数据)、3 个评价影响因素所构成的指标数据矩阵 $X = \{x_{ij}\}_{n\times m}$ 中，数据的离散程度越大，信息熵越小，其提供的信息量越大，该指标对供需匹配程度评价的影响越大，其权重也应越大；反之，各指标值差异越小，信息熵就越大，其提供的信息量则越小，该指标对评价结果的影响也越小，其权重亦应越小。

用熵值法确定指标权重，既可以克服主观赋权法无法避免的随机性、臆断性问题，还可以有效解决多指标变量间信息的重叠问题。所以，本文尝试根据各样本数据的离散程度，用信息熵来确定指标权重，对出租车供需匹配进行评价。

熵值法的算法

Step1 首先为了统一各因素对评价体系的影响程度，需要对原始数据进行处理。根据图1，可观察到12时数据趋于平稳，没有早晚高峰的影响，是最接近于“供需平衡”的一个状态。以8月10日北京市12时的平均数据为标准，对8月10日数据按照如下方式进行正向化处理：

$$a' = -|a - \overline{a}|$$

其中a代表单位车辆需求人数或等待时间或满意度，a’代表正向化后的数据，$\overline{a}$ 代表对应的北京市12时的平均数据。

Step2 由于所给的三个指标所代表的物理含义不同，因此存在着量纲上的差异，所以需要对数据进行无量纲化。算法公式为：

$$b = \frac{x - \min x}{\max x - \min x}$$

其中x表示原指标的值，b表示对应的x标准化后的值。

Step3 建立决策矩阵，3列表示3个影响因素，行表示不同地点不同时间组成的数据。求第j个影响因素下第i个数据 A_i 的贡献度 P_{ij}，先求出各列的和，然后用每行的数值比上列和，形成新的矩阵。求解不同地点不同时间组成的第i个数据 A_i 的贡献度的公式如下：

$$P_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}}$$

其中 x_{ij} 表示第i个数据的第j个属性的正向化后的值，m表示4个不同区域24个不同时间段组成的96个数据。

Step4 用 E_j 来表示不同区域不同时间段组成的全部数据对影响因素 X_j 的贡献总量：

$$E_j = -K \sum_{i=1}^{m} P_{ij} ln(P_{ij})$$

其中，常数 $K = \frac{1}{ln(m)}$，这样，就能保证 $0 \leq E_j \leq 1$，即 E_j 最大为1。

由式中可以看出，当某个影响因素下各区域或时间段的贡献度趋于一致时，E_j 趋于1；特别是当全相等时，也就可以不考虑该目标的影响因素在决策中的左右，也即此时影响因素的权重为0。可看出影响因素值由所有区域或时间段差异大小来决定权系数的大小，为此可以定义 $d_j = 1 - E_j$。

Step5 各影响因素的权重 w_j 计算方式：

$$w_j = \frac{d_j}{\sum_{j=1}^{n} d_j}$$

当 $d_j = 0$ 时，第j个影响因素可以剔除，其权重等于0。

Step5 最后通过得到的三种影响因素的权重，通过对原数据加权求和，即可得到“供求匹配”程度的评价指标值。指标值越大，说明此地此刻的出租车“供求匹配”程度

越好；指标越小，说明“供求匹配”程度越差。评价指标值 F_i 的计算公式如下：

$$F_i = \sum_{j=1}^{m} w_j x_{ij} \tag{2}$$

6.1.4 供需匹配评价模型的求解

通过熵值法的计算，得到了四个“供求匹配”程度的影响因素的权值 (系数) 如图 4。

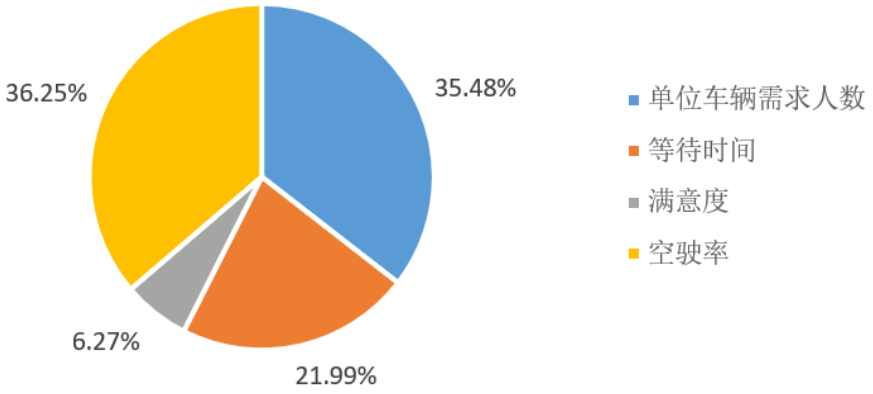


图 **5**　指标权重比

由图可知单位车辆需求人数所占权重比为 35.48%，等待时间权重为 21.99%，满意度权重为 6.27%，空驶率权重为 36.25%。从所得数据中可得出满意度对“供求匹配”程度的影响较低，其他三个因素对“供求匹配”程度影响都较大。可以说“供求匹配”程度主要由单位车辆需求人数、等待时间和空驶率决定。

把图中的数据带入 (2) 式，最终得到的北京市 2016 年 8 月 10 日“供求程度”匹配评价指标值的计算公式为：

$$F_i = 0.3548x_{i1} + 0.2199x_{i2} + 0.0627x_{i3} + 0.3625x_{i4} \tag{3}$$

利用相同的方法，将正向化处理后的数据带入 3, 便求出北京市 2016 年 8 月 10 日四个区域 24 个时间段的评价指标值。具体结果见附录 D。为了更好的对数据进行分析，将北京市 8 月 10 日四个区域所有的评价值绘制成折线图如下：

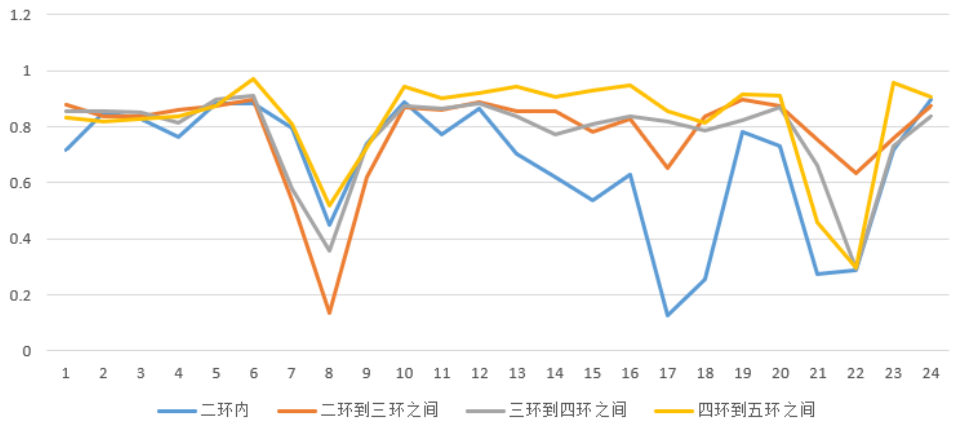


图 **6**　北京市 **8** 月 **10** 日四个区域“供求匹配”程度评价值变化图

图 6的曲线代表了四个区域一天“供需匹配”程度的变化情况。数值越大，越接近于 0，说明此地区此时间段的“供需匹配”程度越好，反之越差。由此可以初步分析：早高峰和晚高峰时，都分别出现了不同区域的严重“供需不匹配”情况。同时在午夜出现了较严重的“供求不匹配”情况。而在不同时间段，每个区域的供需匹配情况也有着较大的差异。

6.1.5 同一时间不同地点“供需匹配”程度比较

由于一天之内不同地点不同时间“供需匹配”程度变化较大，所以对每个时间点四个区域的程度比较都需要单独的讨论。以北京市 8 月 10 日第 8 个时间段、第 17 个时间段为例分析，其它时间段的分析方式相同。

将这两组数据用柱状图的形式表现出来，如图 7。

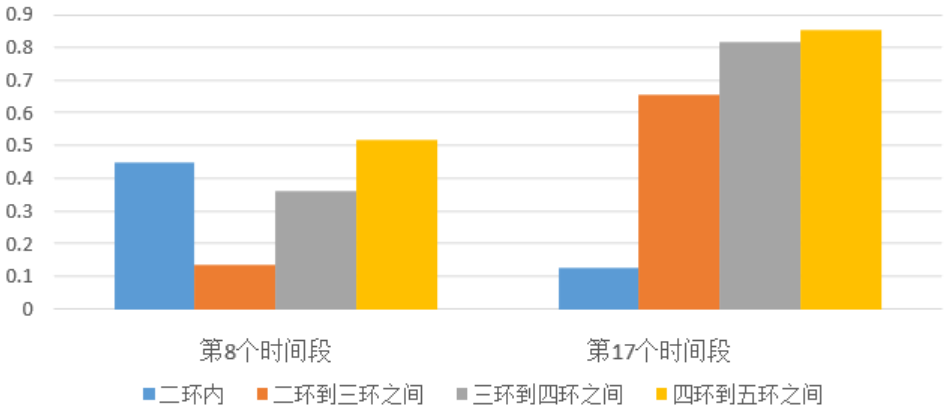


图 7　第 8、17 个时间段四个区域“供求匹配”程度评价值变化图

第 8 个时间段处于一个上班的早高峰时期，二环和四环到五环之间的评价值较高，出租车“供需匹配”程度较好，乘客能够很容易打到出租车；而对于二环到三环以及三环到四环之间，出租车的“供需匹配”程度很差，非常不容易打到出租车，这是因为早高峰上班族的集体出行，从居住区集中前往内环的商业区上班工作。评价值得到的结果合乎现实情况。

第 17 个时间段处于下班的晚高峰，人群集体由内环向外环移动，所以二环内此时的“供需匹配”程度较差；相比而言，其他三个区域的“供需匹配”程度都较好，乘客能够及时打到出租车，符合现实情况。

6.1.6 同一地点不同时间“供需匹配”程度比较

以二环内以及二环到三环两个地区 8 月 10 日整日“供需匹配”程度平评价值为例分析。分别分析其折线图，如图**??**、图**??**。

二环到三环凌晨时的“供求匹配”程度一直处于很高的状态，这是由于出租车的需求量相对较低，出租车的数量相对其他时间段更能满足乘客出行，甚至会呈现一种“供过于求”的现象。随着早高峰的到来，“供需匹配”程度急速下降，大量的上班族需要从他们居住的二环到三环间赶往内环上班，而过了这个时期，“供需匹配”程度又趋于稳定状态。下午 15 时出现的“供需匹配”程度下降是因为人群集体外出进行娱乐活动。晚上 23 时可能是因为此时公共交通已经停运，机场的乘客只能选择出租车进内环。“供不应求”，符合现实情况。

二环内，由于不是上班族的集中居住区域，所以没有早高峰，一直处于很好的“供求匹配”状态，下午由于外环的人群向内环商业区移动，造成正常的出租车短缺情况，符合实情。

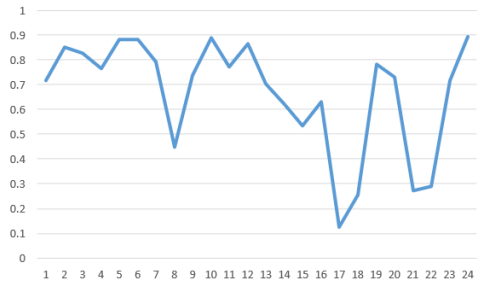


图 **8**　二环内“供需匹配”程度变化

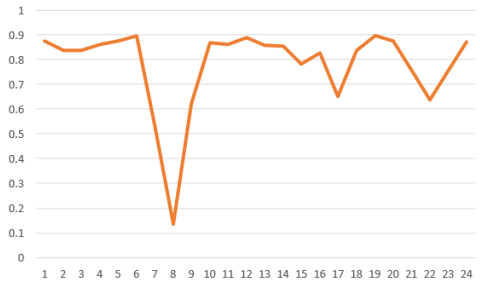


图 **9**　二环到三环“供需匹配”程度变化

6.2 问题二

出租车司机经常会由于行程距离短、道路拥堵而不愿意接单，各大公司为了解决“打车难”的问题，都推出了自己的出租车补贴方案。为了分析补贴方案的有效性，收集了“快的”和“滴滴”两家公司的补贴方案进行分析。接着通过建立乘客出租车的排队模型，对乘客打车坐车进行模拟。最后通过比较补贴前后乘客平均等待时间的变化来判定补贴方案的有效性。

局部假设

1. 单个地点乘客队列长度计算系数 α=0.01
2. 单个地点空载出租车队列长度计算系数 β=0.002
3. 假设乘客只会在乘客队列满的情况下才会取消订单。
4. 假设同一时间段内，同一地点乘客和空闲出租车的到达均满足泊松流。

6.2.1 出租车乘客“打车难”因素分析

随着经济的快速发展, 市民对出租车的需求量在逐渐增加, “打车难”问题同样在以某种方式“发展”。通过查询相关资料，得出“打车难”问题的主要原因如下：

- **司机拒载**：由于司机接单时距离乘客过远，或者是乘客目的地所经过的地区拥挤，司机会选择拒载，因为接这样的订单不仅会导致时间的浪费，还会有可能亏本，就会形成乘客“打车难”的局面。
- **出租车数量少，市场垄断**：一线城市的经济发展较快, 但据资料统计, 北京上海等发达地区的打车难度居高不下, 一部分原因在于出租车市场的地域垄断, 一些公司抢占市场份额形成垄断局面, 竞争力度的大大削弱使得出租车资源配置效率低下, 造成“打车难”的情况。
- **交通拥堵与高峰期**：高峰期与交通拥堵让打车的空驶率居高不下。特别是早晚高峰期很难打到车, 原因就在于交通拥堵, 一线、二线城市这种现象屡见不鲜。一旦出租车堵在路上，便会形成“打车难”的局面。

通过第一问分析，得到三个影响“供求匹配”程度的因素：单位车辆需求人数、乘客平均满意度和平均等待时间，这三个因素也同样是影响“打车难”问题的重要因素。综上所述，由于乘客平均等待时间是使乘客感受到“打车难”问题最直接的指标。“打车难”的整体原因可以归结成乘客平均等待时间的增加。所以只需要研究乘客平均等待时间的长短便可以解决“打车难”的问题。

6.2.2 “滴滴”与“快的”公司补贴方案的说明与比较

通过查阅相关文献 [3]，收集到了”滴滴”与“快的”两家推出的出租车补贴政策如表 5。两个公司的补贴政策都很直接，对顾客或者是司机进行现金补贴。对司机进行补

表 **5**　打车软件补贴表

时间	滴滴打车	时间	快的打车
1.10	顾客车费立减 10 元、司机立降 10 元	1.20	顾客车费返现 10 元，司机奖励 10 元
2.17	顾客返 10~15 元、新司机首单奖 50 元	2.17	顾客返现 11 元，司机返 5~11 元
2.18	顾客返现 12~20 元	2.18	顾客返现 13 元
3.7	顾客每单减免随机 6~15 元	3.4	顾客返现 10 元/ 单，司机端补贴不变
3.23	顾客返现 3~5 元	3.5	顾客补贴金额变为 5 元
5.17	顾客补贴“归零”	3.22	顾客返现 3~5 元
7.9	司机端补贴降为 2 元/单	5.17	顾客补贴“归零”
8.12	取消对司机接单的常规补贴	7.9	将司机端补贴降为 2 元/单

贴的政策不仅能够缓解司机拒载的问题，还能够吸引更多的司机参与到出租车的行列中，同时出租车公司的效益也会增加。而对乘客进行补贴，只能够增加乘客打车的难度，反而增加了“打车难”问题的严重性。

为了理性的分析两个公司打车补贴政策对“打车难”问题的缓解程度，建立模拟乘客和出租车的排队模型，求解补贴政策前后乘客平均等待时间的变化情况。

6.2.3 模型准备

由于乘客出租车排队安排系统包含诸多随机因素，很难给出解析的结果，因此可以借助计算机模拟对该排队系统进行模拟。

通过查阅相关资料，得知 2016 年北京出租车日订单为 100 万左右，而收集到的数据只有 32 万。由此认为收集到的数据的时间段为 20 分钟，即 t=20min。

乘客到达的泊松分布检验

将搜集到的北京市 8 月 6 日-8 月 12 日第 8 个时间段和第 17 各时间段的出租车需求量的数据进行统计，结果如下表：

表 **6**　七日出租车需求量统计

	8.6	8.7	8.8	8.9	8.10	8.11	8.12
8 时	16448	12824	22321	15837	26860	27696	32425
17 时	125697	107404	69880	80849	143551	159048	219861

利用 SPSS 中的非参数检验（K-S 检验），得到结果如下：

		需求总量17点	需求总量8点
个案数		7	7
泊松参数[a,b]	平均值	129470.0000	22058.7143
最极端差值	绝对	.571	.532
	正	.571	.429
	负	-.429	-.532
柯尔莫戈洛夫-斯米诺夫 Z		1.512	1.408
渐近显著性（双尾）		.021[c]	.038

图 **10** **K-S** 检验结果

由结果可看出：两时刻得到的显著性水平均大于 0.05。因此可以认为乘客到达服从泊松分布。

乘客与出租车的到达概率

以单个地点的一段时间段 t 内为研究对象。首先假设同一时间段内，同一地点乘客和空闲出租车的到达均满足泊松流则有：

在 t 时间段内来 x 个乘客的概率为：

$$P_x(t) = \frac{(\lambda_1 t)^x}{x!} e^{-\lambda_1 t} \tag{4}$$

在 t 时间段内来 y 辆空载出租车的概率为：

$$P_y(t) = \frac{(\lambda_2 t)^y}{y!} e^{-\lambda_2 t} \tag{5}$$

其中 λ 表示乘客或空载出租车的到达强度。这两个泊松流概率对应的数学期望分别为：

$$E(x) = \lambda_1 t, E(y) = \lambda_2 t$$

为了求出泊松流概率 $P_x(t)$ 与 $P_y(t)$，需要求解乘客和空载出租车的到达强度 λ_1 和 λ_2。由第一问的数据，可以得到单个时间段的出租车需求总量 d，即为 t 时间段乘客到达的总量，而 t 时间段乘客到达总量等于时间段 t 乘上乘客到达强度 λ_1，便可以认为这个值就是乘客数量 x 的数学期望值：

$$d = \lambda_1 t = E(x) \tag{6}$$

对于空载出租车的到达强度 λ_2 来说，没有空载出租车的数量数据，所以不能用同样的方法求解。在排队模型中存在两个队列，如果排队的乘客数或空载出租车数超过队列长度，乘客或者是空载出租车会选择离开，具体情况在后面模型建立时讨论。

对于“打车难”的情况来说，乘客队列从一开始便会增加，直到到达队列长度，排队乘客数便会稳定在这个值上下。而空载出租车队列会保持常空状态，因为实际情况下，一但空载出租车到达，马上会带着乘客队列前端的乘客离开。此时，乘客的减少强度即为空载出租车到达的强度 λ_2。

单一乘客从队尾排到上车的时间便是问题一中的乘客平均等待时间，是已知的数据。则可知空载出租车到达的强度 λ_2 即为队列长度 L_c 除以此时此地乘客平均等待时间 t_r。现在需要求出乘客队列长度。

由于乘客队列长度 L_c 和对应区域区域出租车需求量 d_j 存在一定的关系，其关系一定呈正相关，由此假设：

$$L_c = \alpha d_j \tag{7}$$

通过探究，假设 α=0.01。而出租车需求量在第一问中是已知量，最终便可以通过乘客队列长度 L_c 和乘客平均等待时间 t_r 得到：

$$\lambda_2 = \frac{L_c}{t_r} \tag{8}$$

由于早高峰是最典型的”打车难“的时候，所以取 8 月 10 日第 8 个时间段的数据来计算 λ_1 和 λ_2，结果如表 4。

表 **7**　不同地区”打车难“泊松概率的出现强度 λ 表

	二环内	二环到三环之间	三环到四环之间	四环到五环之间
λ_2	0.3751	0.3026	0.3662	0.7859
λ_1	1.4558	3.1317	3.0258	3.7700

补贴金额与乘客数量或出租车数量的关系

建立模型之前还需要对补贴金额与乘客数量以及补贴金额与出租车数量的关系进行分析求解。不补贴乘客或者司机的情况，将会是乘坐出租车的乘客和出租车数量都最少的时候；当公司对乘客进行一定的补贴，会吸引更多的人乘坐出租车，于是乘客数量便会增加，并且由于有意向乘坐出租车的人数是一定的，出租车的数量是有限的，所以随着补贴金额的增加，乘客增加的速度会放缓，最终趋近于某一个最大值；对于司机也是一样的原理，随着补贴金额的增加，出租车数量的增长速度会放缓，并最终趋近于一个值。于是可以假设乘客数量 x_1 和补贴金额 s 的关系，以及出租车数量 x_2 和补贴金额 s 的关系如下：

$$\text{乘客数量：} x_1 = N_1(1 - e^{-a_1 s_1}) + b_1 \tag{9}$$

$$\text{出租车数量：} x_2 = N_2(1 - e^{-a_2 s_2}) + b_2 \tag{10}$$

其中 b_1 和 b_2 表示补贴 $s = 0$ 的情况下，乘客数量和出租车数量，即为收集到的数据，是已知量。

乘客等待时间

通过乘客数量、出租车数量以及乘客、出租车的到达强度，可以求出乘客的等待时间。由式 (7) 和式 (8) 可得等待时间 t_w：

$$t_w = \frac{\alpha d_j}{\lambda_2}$$

其中出租车需求量 d_j 等于乘客数量 x_1。λ_2 表示出租车到达强度，与出租车的数量呈正比，所以：

$$\lambda_2 = kx_2 = kN_2(1 - e_2^{-a_2 s}) + kb_2 = kN_2(1 - e^{-a_2 s_2}) + \lambda_2'$$

x_2 表示出租车的数量，λ_2' 表示不补贴时，车的到达强度。

综上，乘客等待时间 t_w 的表达式为：

$$t_w = \frac{\alpha N_1(1 - e^{-a_1 s_1}) + \alpha b_1}{k N_2(1 - e^{-a_2 s_2}) + \lambda_2'} \tag{11}$$

6.2.4 乘客出租车排队安排及系统模拟模型的建立

根据系统模拟的一般方法，需要考虑系统的如下数据和参数。

1. 输入：
   - 出租车、乘客到达强度
   - 出租车、乘客队列长度
   - 模拟时间
2. 状态 (变量)：
   - 等待乘车的乘客
   - 等待乘客的空载出租车
3. 实体: 出租车、乘客
4. 事件：
   - 一位乘客到达队列，需要乘车
   - 一辆出租车到达队列，等待乘客
   - 乘客乘坐出租车离开
   - 乘客队列已满时，新到达乘客离开
   - 出租车队列已满时，新到达空载出租车离开
5. 活动：
   - 乘客到达、离开队列的时刻
   - 乘客在队列内等待出租车的时间
   - 出租车到达、离开队列的时刻
   - 出租车在队列内等待乘客的时间
6. 输出：
   - 乘客等待时间图像
   - 乘客平均等待时间
   - 空驶率

通过上述数据和参数，便可以根据模拟系统的方法利用 MATLAB 编写程序，建立模拟系统，具体代码见附录 E，算法流程图见附录 F。

单个区域单个时间段具体模拟方法如下：

Step1 利用对应地区的泊松流函数随机生成出租车和乘客的到达时刻 $arriveTime1$ 和 $arriveTime2$。

Step2 如果 $arriveTime1 < arriveTime2$，则出租车进入出租车队列 $queue1$ 进行排队，按顺序等待乘客到达；反之如果 $arriveTime1 > arriveTime2$，则乘客进入乘客队列 $queue2$，等待出租车到达。

Step3 如果队列 $queue1$ 或 $queue2$ 排队的乘客或出租车达到队列长度上限，则对应的出租车或乘客选择离开，而不是继续排队。

Step4 如果队列 $queue1$ 和 $queue2$ 中同时存在出租车和乘客，则同时出对，即模拟乘客坐上出租车离开的过程。此时出租车或乘客离开时间以及等待时间分别为：

$$leaveTime1 = leaveTime2 = \max\{arriveTime1, arriveTime2\}$$

$$waitTime1 = leaveTime1 - arriveTime1$$

$$waitTime2 = leaveTime2 - arriveTime2$$

Step5 若模拟时间达到 T，则结束模拟，输出 $waitTime1$ 和 $waitTime2$；否则回到 Step1。

通过系统模拟出租车乘客排队模型，能够有效的对各公司推出的”补贴政策“进行定量分析。

6.2.5 乘客出租车排队安排及系统模拟模型的求解

首先对模型进行验证，带入第 8 个时间段的数据，和原始数据进行比较，判断模型准确性。接着分别模拟对乘客和出租车司机补贴的效果，与原情况进行比较分析。

通过分析，得出式 (11) 中的参数为 $N_1 = 6000, kN_2 = 0.5, a_1 = 0.1, a_2 = 0.01$，乘客等待时间表达式即为：

$$t_w = \frac{60 \times (1 - e^{-0.1 s_1}) + 0.01 b_1}{0.5 \times (1 - e^{-0.01 s_2}) + \lambda_2'}$$

模型准确度验证

将 8 月 10 日第 8 个时间段的数据带入建立的模拟系统，T 取 1 小时。输入如下：

表 **8**　模拟系统输入数据

	λ_2	λ_1	L_c	L_t
二环内	0.3751	1.4558	17	7
二环到三环之间	0.3026	3.1317	37	12
三环到四环之间	0.3662	3.0258	36	15
四环到五环之间	0.7859	3.7700	45	16

其中 L_c 表示乘客队列长度，L_t 表示出租车队列长度。
得到结果如下图：

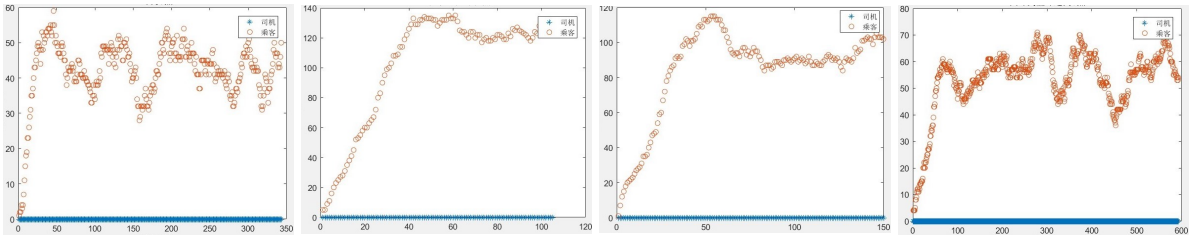


图 **11**　四个地区随着时间增长，等待时间变化

图中横轴表示乘车累计人数，纵轴表示等待时间。由图像可以看出，当等待人数增加到一定数量时，等待时间趋于稳定。符合现实生活中，某一段时间内某个地点，乘客的平均等待时间处于饱和的状态。

表 **9**　乘客平均等待时间真实值与模拟值比较表

	二环内	二环到三环之间	三环到四环之间	四环到五环之间
实际等待时间（秒）	45.32	122.29	98.32	57.26
模拟等待时间（秒）	43.06	107.75	86.41	54.19

由上表，观察到真实值与模拟系统的模拟结果相差最大是在二环到三环之间这个区域，相差值为 14.5456 秒；最小相差为 2.2605 秒。模拟值与真实值的差距很小，由此可以说明模拟系统的准确性很高。

补贴政策的影响

(1) 对司机进行补贴

出租车公司对司机进行补贴，不对乘客进行补贴，即式 6.2.5中的 $s_1 = 0$，取 $s_2 = 3, 6, 9, 12, 15$ 表示分别补贴 3，6，9，12，15 元的情况，对应不同程度的补贴。对

应的等待时间如下表：

表 **10**　对司机补贴前后乘客平均等待时间变化情况表

补贴金额（元）	0 元	3	6	9	12	15
二环内	45.32	41.84	38.94	38.57	38.16	36.77
二环到三环之间	122.29	110.70	103.08	99.71	95.30	93.27
三环到四环之间	98.32	85.21	82.73	81.71	80.87	77.36
四环到五环之间	57.26	51.26	49.32	48.80	47.40	46.60

通过观察上述表格，可以看出随着 λ_2 值的增大，四个区域乘客的平均等待时间均减少。有力的证明了之前的分析：给司机补贴可以提高单位时间内出租车的到来次数从而可以减少乘客等待时间，进而缓解“打车难“的问题。

(2) 对乘客进行补贴

出租车公司对乘坐出租车的乘客进行补贴，不对司机进行补贴，即式 6.2.5中的 $s_2 = 0$，取 $s_2 = 3, 6, 9, 12, 15$ 表示分别补贴 3，6，9，12，15 元的情况，对应的等待时间如下表：

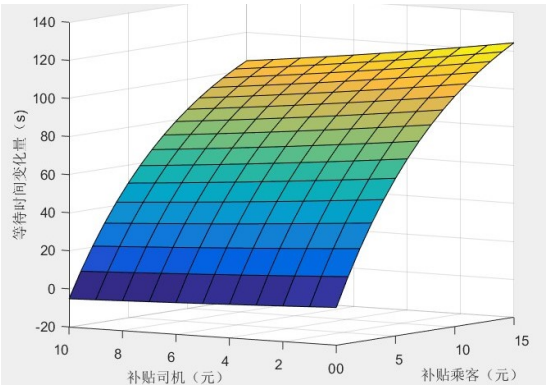
表 **11**　对乘客补贴前后乘客平均等待时间变化情况表

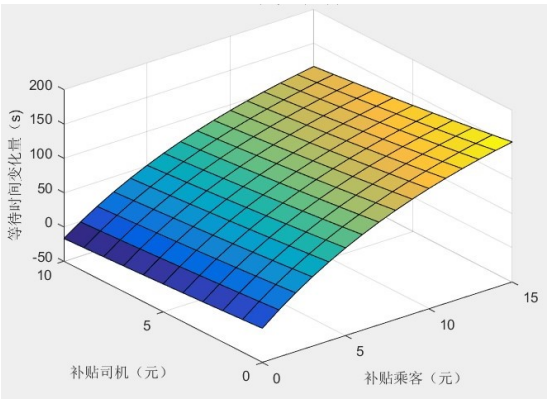
补贴金额（元）	0 元	3	6	9	12	15
二环内	45.32	56.87	67.86	71.20	73.24	73.51
二环到三环之间	122.29	131.04	138.82	140.33	142.45	143.13
三环到四环之间	98.32	107.65	113.88	115.24	117.86	118.25
四环到五环之间	57.26	68.55	76.81	80.45	83.16	84.49

通过观察上述表格，可以看出随着 λ_1 值的增大，四个区域乘客的平均等待时间均增大。同样的证明了之前的分析：给乘客进行补贴提高了乘客出行选择出租车的比率，越来越多的乘客选择出租车从而增加了“打车难”的问题。

(3) 对司机和乘客都进行补贴

若对司机和乘客共同进行补贴，则同时改变式 6.2.5中的 s_1 和 s_2，最终得到的结果如下：





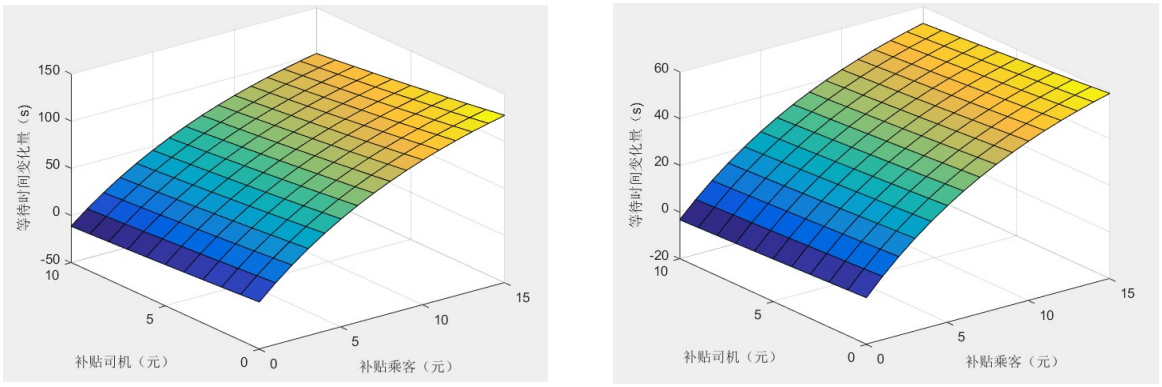


图 **12**　各地区补贴随等待时间的变化

通过上述结果，可以看出在不对乘客补贴的情况下，对司机进行补贴能够有效的减少乘客平均等待时间，能够缓解打车难的问题；而对乘客进行补贴，会导致乘坐出租车的人数大幅度增长，反而使“打车难”问题更加严重了，但是能够很有效的吸引乘客，对抢占市场有一定的作用。

6.2.6 结果分析

通过建立乘客出租车排队系统模拟模型，利用已有数据进行分析并证明得到了如下结论:

通过表 5可知，两家出租车公司都有对乘客进行补贴，进一步的增加了人们出行选择出租车的几率，虽然这对于出租车公司是有利的，但是数量不变的出租车会导致乘客”打车难“的问题更加严重，完全没有起到缓解打车难的作用，但能够有效积攒出租车的人气。而另一方面，出租车公司对司机也有补贴，促进了更多的人加入出租车司机的行列，在乘客不变的情况下，对”打车难“问题的帮助是有效的。

6.3 问题三

上述问题对目前市场上两种不同软件平台补贴方案进行了详细的探讨分析，这些方案大都是单纯的以载客次数为补贴依据，在一定程度上可以缓解“打车难”的问题，但是并没有将这些补贴资金完全利用起来，还存在相当大的改进空间，本问题将从几个不同的方面进行探讨，从而建立更好的补贴模型，以实现将补贴资金完全利用起来。

为了解决“打车难”的问题，同时也兼顾到出租车公司的利润，需要设计新的补贴方案。结合前面一二问的分析，分开探讨公司初期和平稳期发展计划，提出了对司机补贴和对拼车补贴两种方式。通过补贴司机来提高司机调度的积极性，通过对拼车进行补贴能够提高乘客拼车的机率，有效减少乘客排队等待时间。最后通过分别建立优化模型求解，把优化后的等待时间与优化前的相比，论证其合理性。

为了研究更合理的补贴方案，需对现行的出租车补贴方案进行分析，对他们的特点进行了深度挖掘，发现他们都有几个相同的弊端：

1. 现行补贴方案没有对时间段进行精细划分，依然存在高峰期出租车资源严重不足的局面，补贴利用率不高的情况；
2. 城市不同区域对出租车资源的需求也不尽相同，现存补贴方案没有对地区作区分，针对不同的地区应该有不同的补贴方案；
3. 出租车没考虑到平台派单的情况，实际上网约车分为抢单和平台派单。
4. 出租车公司没有对拼车乘客的补贴，而拼车能够有效缓解打车难的问题。

6.3.1 软件推广阶段补贴方案

软件推广阶段是软件服务平台的起始阶段，是一个需要快速提高软件知名度和用户量的阶段。此时的推广模式应该具有补贴力度大、形式多样、范围广泛等特点。这样的推广方式主要以收单优惠为主。

对于所有乘客以及出租车司机首次使用打车软件，一方面是免单，另一方面是对乘客采取赠送代金券，同时对司机赠送话费，且不同时间段的补贴方案有所不同，对于需求量大的时间段补贴高，反之则低，以第 8 个时间段（早高峰）和第 17 个时间段（晚高峰）为例，补贴方案如下：上表中对乘客的奖励金额包括话费、代金券和现金，对司机的奖励包括话费和现金。

表 **12**　早晚高峰补贴方案

	早高峰				晚高峰			
	二三环间	三四环间	二环内	四五环间	二环内	二三环间	三四环间	四五环间
乘客	200-250	150-200	100-150	≤ 100	200-250	150-200	100-150	≤100
司机	50	45	40	35	50	45	40	35

6.3.2 软件稳定运营阶段补贴方案

当一个新的打车软件平台在前期推广阶段之后，在市场上累积了一定的客户群体之后，便不需要再进行大额度的补贴来支撑软件的日常运行，这时的补贴方案转变为更为理性、更有针对性的补贴方案，联系现行的补贴方案存在的种种弊端，从司机和乘客两个角度进行更为合理的补贴。

6.3.3.1 司机被动调度补贴

第一种补贴方案便是对出租车司机进行区域指派并给予现金补贴，使得司机愿意从出租车较多的区域赶往出租车较少的区域接订单。为了方便分析四个区域的车辆调度情况，以此来寻找最优的司机补贴方案。在相同的时间段内，设 λ_{1i} 为第 i 个地区乘客的到达强度，i=1，2，3，4，分别表示北京市的四个区域。同理设 λ_{2i} 为第 i 个地区空载出租车的到达强度。t_{ij} 表示 i 区域到 j 区域所需要的时间，j=1，2，3，4，同样表示北京市的四个区域，且易知 t_{ij}=t_{ji}。T_i 表示在 i 区域出租车司机接到单的时间。

如果能将出租车的分布按照乘客分布比例安排，则能够在车辆有限的情况下最大的缓解”打车难“的问题。通过第二问求出的每个地区乘客到达强度的比率，即可确定乘客数量的分布比率，因为乘客到达强度与乘客分布比率是成正比的。同理，每个地区出租车的数量也是与此地空载出租车到达强度成正比，同样可以使用多个地区到达强度比率代替出租车数量比率。由以上的比率关系可以得到补贴后的单个地区出租车的到达强度：

$$\frac{\lambda'_{2i}}{\lambda'_{21}+\lambda'_{22}+\lambda'_{23}+\lambda'_{24}}=\frac{\lambda'_{1i}}{\lambda'_{11}+\lambda'_{12}+\lambda'_{13}+\lambda'_{14}}$$

其中 λ'_{2i} 代表补贴过后，即调度过后，各地区出租车的到达强度。通过化简，可得到：

$$\lambda'_{2i}=\frac{\lambda'_{1i}}{\lambda'_{11}+\lambda'_{12}+\lambda'_{13}+\lambda'_{14}}(\lambda'_{21}+\lambda'_{22}+\lambda'_{23}+\lambda'_{24}) \tag{12}$$

由于对司机补贴前后引起的调度效果不影响各个地区乘客的到达强度，即乘客到达强度不变，所以有：λ'_{ji}=λ_{ji}；并且对司机的补贴不允许出租车所有区域总的数量，即总的到

达强度不变，所以：$\lambda'_{21}+\lambda'_{22}+\lambda'_{23}+\lambda'_{24}=\lambda_{21}+\lambda_{22}+\lambda_{23}+\lambda_{24}$。式 (12) 可以化成：

$$\lambda'_{2i}=\frac{\lambda_{1i}}{\lambda_{11}+\lambda_{12}+\lambda_{13}+\lambda_{14}}(\lambda_{21}+\lambda_{22}+\lambda_{23}+\lambda_{24})$$

对于司机来说，只有当补贴费用足以抵消地区间流动的损失时，司机才会接受指派并进行调度。设 m_{ij} 为司机从 i 区域转移到 j 区域的利润差，即不转移与转移两种情况下司机利润的差值：

$$m_{ij}=((t_{ij}+T'_j)-T_i)s_i \tag{13}$$

其中 s_i 表示在 i 区域载客时，单位时间的收益，T'_j 表示补贴后在 j 地司机接到单的时间，T_i 表示补贴前，司机在 i 地接到单的时间。上式可理解为赶路以及在 j 区域接单时间所获得的收益减去同样时间在原 i 地接单获得的收益，即利润差值，即公司需要对出租车司机补贴的最小金额。

缓解打车难的目标优化模型的建立

首先需要判断各地区出租车调入和调出情况:$\lambda_k-\lambda'_k$>0，则认为 k 地需要调出多余的出租车；$\lambda_k-\lambda'_k$<0，则认为 k 地需要从其他地区调入出租车。

设各地区调度后，出租车到达强度的变化量为 $\Delta\lambda_{ij}$，i,j=1,2,3,4，表示从 i 区域调往 j 区域的出租车到达强度。i 表示需要调出出租车的区域；j 表示需要调入出租车的区域。为此约束问题的四个自变量。

(1) 目标函数的确定

目标一：最大匹配程度

为了最大程度的缓解“打车难”问题，出租车调度后各地区的比率要更加接近出租车需求比率，即与最佳比率 (λ'_{21}：λ'_{22}：λ'_{23}：λ'_{24}) 的几何“距离”最小：

$$\min\quad G=\sum_i[(\lambda_{2i}-\sum_j\Delta\lambda_{ij})-\lambda'_{2i}]^2+\sum_j[(\lambda_{2j}+\sum_i\Delta\lambda_{ij})-\lambda'_{2j}]^2 \tag{14}$$

其中第一项和表示有出租车调出的区域,i 表示所有的需要调出出租车的区域,则 $\sum_j\Delta\lambda_{ij}$ 表示所有从 i 区域调出的出租车的到达强度之和；第二和项表示有出租车调入的区域，j 表示所有需要调入出租车的区域，$\sum_i\Delta\lambda_{ij}$ 表示所有调入 j 区域的出租车的到达强度之和，i 不等于 j。

目标二：最少补贴

在匹配最优方案的同时，也需要考虑公司的利益，使公司支出的尽量少，所以有以下约束条件：

$$\min\quad H=\sum_{i,j}m_{ij}\Delta\lambda_{ij} \tag{15}$$

此式表示各调度总共需要的补贴金额。

多目标转化单目标

这是一个多目标优化的模型，因此考虑使用理想点法。

理想点法：先用单目标优化方法求得各子目标的最优解，然后使各子目标函数离各自优化解的距离相对值，并且在量纲规范化后引入方差进行求和，从而得到衡量综合偏

离程度的单目标函数 将两个消除量纲后的函数相加，便可以构造新的目标函数如下：

$$\min[\frac{G-G_{min}}{G_{max}-G_{min}}]^2+[\frac{H-H_{min}}{H_{max}-H_{min}}]^2$$

(2) 约束条件的确定

每个区域的调度量都小于其原来的出租车到达强度与最佳比率下出租车的到达比率之差的值，即：

$$\sum_j \Delta\lambda_{ij} \le |\lambda_{2i}-\lambda'_{2i}| \tag{16}$$

所有的调度都需要为正：

$$0 \le \lambda_{ij} \tag{17}$$

(3) 优化模型汇总

综上得到的优化模型为一个多目标优化模型：

$$\min[\frac{G-G_{min}}{G_{max}-G_{min}}]^2+[\frac{H-H_{min}}{H_{max}-H_{min}}]^2$$
$$s.t.\begin{cases}\sum_j \Delta\lambda_{ij} \le |\lambda_{2i}-\lambda'_{2i}| \\ 0 \le \lambda_{ij}\end{cases}$$

目标优化模型的求解

为了能够更好的验证优化模型的有效性，选取了早晚高峰（第 8 个时间段和第 17 个时间段）为例计算并论证合理性。首先对调度前和最优调度方案进行对比（第 15 个时间段的数据见附录 F）：

表 **13**　第 **8** 个时间段最优调度前后对比

	调度前	最优调度后	差值（前减后）	进出情况
二环内	0.3751	0.2340	0.1411	出
二环到三环之间	0.3026	0.5034	-0.2008	进
三环到四环之间	0.3662	0.4864	-0.1202	进
四环到五环之间	0.7859	0.6060	0.1799	出

可以知道第 8 个时间段需要的调度 (优化函数自变量) 为：$\Delta\lambda_{12}$,$\Delta\lambda_{13}$,$\Delta\lambda_{42}$,$\Delta\lambda_{43}$，即从二环内向二四环内域调度，从四五环间向二四环间调度。同理第 17 个时间段的自变量为：$\Delta\lambda_{31}$,$\Delta\lambda_{32}$,$\Delta\lambda_{41}$,$\Delta\lambda_{42}$。

通过 LINGO 软件求解的最优结果为：

表 **14**　早晚高峰最优调度方案

早高峰				晚高峰			
$\Delta\lambda_{12}$	$\Delta\lambda_{13}$	$\Delta\lambda_{42}$	$\Delta\lambda_{43}$	$\Delta\lambda_{31}$	$\Delta\lambda_{32}$	$\Delta\lambda_{41}$	$\Delta\lambda_{42}$
0	0	0	0.0056	0	0	0	0.1003

从上表可以看出对于早高峰的情况，从四五环间调度 0.0056 到达强度对应的出租车数量到三四环间，便可达到最佳方案。对于晚高峰，从三四环间向四五环间调度 0.1003 到达强度对应的出租车数量。调度前后的 λ_{2i} 的值如下：

表 **15**　早晚高峰空载出租车到达强度调度前后变化

	早高峰调度前	早高峰调度后	晚高峰调度前	晚高峰调度后
二环内	0.3751	0.3751	0.2342	0.2342
二环到三环之间	0.3026	0.3026	0.2910	0.3913
三环到四环之间	0.3662	0.3718	0.2894	0.2894
四环到五环之间	0.7859	0.7803	0.5651	0.4648

最后带入第二问的排队系统模拟模型中，可得 (以第 8 个时间段为例，第 17 个时间段的数据见附录 H)：

表 **16**　早晚高峰乘客平均等待时间调度前后变化

	早高峰调度前	早高峰调度后	早高峰人数	晚高峰调度前	晚高峰调度后	晚高峰人数
二环内	45.320602	45.320602	1747	93.92	93.92	2240
二环到三环之间	122.29309	122.29309	3758	65.29	38.80	1978
三环到四环之间	98.317995	64.7414	3631	48.38	48.38	1490
四环到五环之间	57.258927	60.3775	4524	38.93	40.45	2228
平均等待时间	84.537667	76.645413		62.79	56.62	

对于早高峰：指派从四五环间到三四环间的出租车，单位时间指派 0.0056 辆，补贴金额为每辆车每趟补 15.0145 元；此时对司机补贴金额最少，人均等待时间减少了 7.8923 秒。

对于晚高峰：指派从四五环内到二三环间的出租车，单位时间指派 0.1777 辆，补贴金额为每辆车补 22.8200 元，此时对司机补贴金额最少，人均等待时间减少了 6.1754 秒。

6.3.3.2 司机主动调度补贴

第二种方案是公司从司机的角度出发，设计合理的补贴方案来使司机自愿向需求高的区域移动。由于每个地区设置的补贴金额不同，各地区对司机的吸引力也不尽相同。对吸引力大小影响的主要因素为：

- **调度区域补贴金额差值**：司机所在区域和调度目的区域的补贴金额的差值。如果调度目的区域的补贴更高，司机则更愿意主动调度。
- **调度距离**：司机从所在区域到调度目的区域的距离，距离越大，司机主动调度的意愿将越低，反之，司机更愿意主动调度。
- **调度目的区域平均车速差值**：司机相对于拥堵的区域，更愿意选择通畅的道路驾驶，即更愿意在平均速度更高的区域行驶。

最小补贴目标优化模型的建立

通过上述三个因素构造吸引力函数，进而从最优分配和公司成本两方面考虑最佳的补贴方案，即各区域补贴金额。

模型准备

设司机在 i 区域时，目的区域 j 的吸引力为 q_{ij}，目的区域补贴金额为 u_j，i、j 两区域的距离为 d_{ij}，目的区域的平均行驶速度为 v_j。分析表明，目的区域的吸引力 q_{ij} 与三个主要因素 u_i、d_{ij} 和 v_i 大体满足如下关系：

$$q_{ij} = [1 - e^{-\alpha(u_j - u_{min})}] + [1 - e^{-\beta(d_{max} - d_{ij})}] + [1 - e^{-\gamma(v_j - vmin)}]$$

其中，u_{min} 各区域中的最小补贴金额，d_{max} 表示两个区域最远的距离，这里指四五环内到二环内，v_{min} 指各区域中最小的平均行驶速度。

由于各个地区对司机的吸引力值的大小不同，所以最终主动调度到各区域的数量与吸引力大小成正比，各区域吸引力的比值也和调度后的各区域出租车的流向的比值相同。由前面的分析，出租车数量的变化即可等价于出租车出现强度的变化。所以出现强度的变化可以表示为：

$$\lambda_{ij} = \lambda_i \frac{q_{ij}}{q_i}$$

其中，λ_{ij} 表示 i 区域主动调度向 j 区域的出租车数量对应的出现强度，λ_i 表示 i 区域主动调度前出租车数量对应的出现强度，q_{ij} 表示 i 区域向 j 区域移动的吸引力，q_i 表示 i 区域向所有区域流动（包括 i 区域）的吸引力之和。

经过各区域司机的主动调度之后，i 区域新的出租车数量对应的出现强度为所有区域向 i 区域移动的数量对应的出现强度之和：

$$\lambda_{newi} = \lambda_{1i} + \lambda_{2i} + \lambda_{3i} + \lambda_{4i} \tag{18}$$

其中 λ_{newi} 表示司机的主动调度之后，i 区域新的出租车数量对应的出现强度。

(1) 目标函数的确定

目标一：最大匹配程度

在得到各区域新的出租车出现强度后，希望它与最优调度方案 (λ_1'：λ_2'：λ_3'：λ_4') 的差距最小，能够尽量满足各区域乘客需求：

$$\min Z = (\lambda_{new1} - \lambda_1)^2 + (\lambda_{new2} - \lambda_2)^2 + (\lambda_{new3} - \lambda_3)^2 + (\lambda_{new4} - \lambda_4)^2 \tag{19}$$

目标二：最少补贴

从公司角度考虑，希望在向最优分配方案靠近的同时，减少公司的支出：

$$\min W = u_1\lambda_{new1} + u_2\lambda_{new2} + u_3\lambda_{new3} + u_4\lambda_{new4} \tag{20}$$

其中，W 表示公司本次调度总支出，u 表示每个区域的补贴金额。

多目标转化单目标

利用和第一种补贴方案相同的理想点法，可将两个目标函数合并成：

$$\min[\frac{G - G_{min}}{G_{max} - G_{min}}]^2 + [\frac{Z - Z_{min}}{Z_{max} - Z_{min}}]^2$$

(2) 约束条件的确定

通过查询有关文献并结合实际情况，各环补贴金额在 0-15 元为合理区间，所以可以得到约束条件：

$$0 \leq u_i \leq 15, i = 1, 2, 3, 4 \tag{21}$$

(3) 优化模型汇总

综上分析，建立的优化模型为：

$$\begin{aligned} \min \quad & [\frac{G - G_{min}}{G_{max} - G_{min}}]^2 + [\frac{Z - Z_{min}}{Z_{max} - Z_{min}}]^2 \\ s.t. \quad & 0 \leq u_i \leq 15, i = 1, 2, 3, 4 \end{aligned}$$

目标优化模型的求解

由于目标函数中存在函数 G 或函数 H 的最大值和最小值，而这些值是未知的，所以利用常规目标优化的解法不可行。于是利用网格遍历的办法首先寻找 G 和 H 的最大值和最小值，然后再利用 MATLAB 求解，求解得到早晚高峰的最优方案为：

表 **17**　早晚高峰最优补贴方案

	二环内	二环到三环之间	三环到四环之间	四环到五环之间
早高峰	7	1	1	1
晚高峰	5	1	3	8

利用上述最优结果，求得各区域调度前后变化情况如下：

表 **18**　早晚高峰乘客平均等待时间调度前后变化

	早高峰			晚高峰		
	调度前	调度后	人数	调度前	调度后	人数
二环内	45.32	40.04	1747	93.92	55.91	2240
二环到三环之间	122.29	69.79	3758	65.29	40.18	1978
三环到四环之间	98.32	80.07	3631	48.38	30.98	1490
四环到五环之间	57.26	67.83	4524	38.93	50.74	2228
平均等待时间	84.54	68.07		55.73	45.86	

对于早高峰：对二环内每单补贴 7 元，二环到三环内、三环到四环内、四环到五环内每单补贴 1 元，平均等待时间减少 16.47 秒。对于晚高峰：对二环内、二环到三环内、三环到四环内、四环到五环内每单分别补贴 5、1、3、8 元，平均等待时间减少 9.87 秒。

6.3.3.3 乘客拼车的补贴

解决“打车难”问题的另一个途径便是给乘客提供拼车的选项，并给予乘客一定比率的补贴 (相比不拼车的价格)。

局部假设

- 假设乘客出行均为一个人。
- 假设拼车人数只为 2。

拼车优化模型的建立

设公司对乘客拼车的补贴比例为 r，每单的平均价格为 w。由于补贴越高，乘客拼车的概率 p' 越高，所以两者成正比，两者概率相等。

(1) 确定目标优化函数

优化函数需要从两个方面确定：乘客平均等待时间和公司利益。

目标一：乘客最少等待时间

对于乘客等待时间 t，需要其尽可能的小。其值可以由乘客等待队列长度和乘客被接走的强度确定，而乘客被接走的强度等于空载出租车到达强度，即：

$$t = \frac{L_c}{\lambda_1} \tag{22}$$

对于单个乘客来说，他有 $p = 1 - r$ 的概率不拼车，此时，乘客被接走的强度不变，即：$\lambda_1' = \lambda_1$；它有 $p' = r$ 的概率拼车，此时，乘客被接走的强度变为原来的两倍，即：$\lambda_1' = 2\lambda_1$，则此时乘客被接走的强度就为：

$$\lambda_1' = (1 - r)\lambda_1 + 2r\lambda_1$$

带入式 (22)，化简得到乘客等待时间的目标函数为：

$$\min T = \frac{L_c}{(1 + r)\lambda_1} \tag{23}$$

目标二：公司收益最大

对于公司效益方面，希望公司尽可能多盈利，公司的盈利为拼车盈利加上不拼车的盈利，在 t 时间内，公司收益的目标函数为：

$$M = \lambda_1 t \times r \times 2 \times (1 - r)w + \lambda_1 t \times (1 - r)w$$

其中 $\lambda_1 t \times r$ 表示 t 时间内接到拼车的订单数，$(1 - r)w$ 表示补贴乘客后公司单笔订单的收入。由于 w 平均每单价格，在研究单位时间内的收益时，上述目标函数可以转化为：

$$\max M = (2r + 1)\lambda_1(1 - r)wt$$

多目标转化单目标

综上，这也是一个多目标优化问题，和前面对司机补贴的优化问题处理方法一样。首先单独对这两个目标函数求解最优解，然后将两个函数分别减去其最优解的值，再除以最优解。分别对前面得到的值平方后相加（将最大约束的目标函数取负以统一量纲），

得到新的目标函数：

$$\min[\frac{\frac{L_c}{(1+r)\lambda_1}-T_{min}}{T_{max}-T_{min}}]^2-[\frac{(2r+1)\lambda_1(1-r)-M_{max}}{M_{max}-M_{min}}]^2 \tag{24}$$

其中 T_{min} 为第一个目标函数的最优解，M_{max} 为第二个目标函数的最优解。

(2) 确定约束条件

出租车公司对乘客的补贴比例一定是再 0 到 1 之间的，所以：

$$0 \le r \le 1 \tag{25}$$

(3) 优化模型汇总

综上分析，建立的优化模型为：

$$\begin{aligned} \min \quad & [\frac{\frac{L_c}{(1+r)\lambda_1}-T_{min}}{T_{max}-T_{min}}]^2-[\frac{(2r+1)\lambda_1 t(1-r)w-M_{max}}{M_{max}-M_{min}}]^2 \\ s.t. \quad & 0 \le r \le 1 \end{aligned}$$

6.3.3.3 拼车优化模型的求解

利用 LINGO 求解优化模型得到结果如下：

表 **19** 拼车优化模型结果

早高峰			晚高峰		
r	最优值	λ_1	r	最优值	λ_1
0.5752	0.1081	0.5909	0.4989	0.0123	0.3511
0.5752	0.1081	0.4766	0.4989	0.0123	0.4362
0.5752	0.1081	0.5768	0.4989	0.0123	0.4338
0.5752	0.1081	1.2380	0.4989	0.0123	0.8471

利用第二问的乘客出租车排队系统模型对其进行验证，结果如下：

表 **20** 早晚高峰实行拼车补贴前后乘客等待时间变化

	早高峰			晚高峰		
	补贴前	补贴后	人数	补贴前	补贴后	人数
二环内	45.32	26.78	1747	45.32	33.72	2240.00
二环到三环之间	122.29	71.82	3758	122.29	42.05	1978.00
三环到四环之间	98.32	59.76	3631	98.32	41.10	1490.00
四环到五环之间	57.26	33.95	4524	57.26	23.91	2228.00
平均等待时间	84.54	50.31		77.81	34.43	

在每环对拼车的乘客进行补贴，补贴的比例为 0.5752，此时不但公司收益达到最优，人均等待时间也减少了 43.3806 秒。

6.3.3 结果分析

根据上述优化模型结果针对北京市早高峰时段、各环分别设计了对司机和对拼车乘客最优的补贴方案。

- 平台推广阶段：
  - 早高峰时段：二环内的司机共补贴 40 元，乘客共补贴 100 150 元；二三环间的司机共补贴 50 元，乘客共补贴 200 250 元；三四环间环的司机共补贴 45 元，乘客共补贴 150 200 元；四五环间的司机共补贴 35 元，乘客共补贴 0 100 元。
  - 晚高峰时段：二环内的司机共补贴 50 元，乘客共补贴 200 250 元；二三环间的司机共补贴 45 元，乘客共补贴 150 200 元；三四环间环的司机共补贴 40 元，乘客共补贴 100 150 元；四五环间的司机共补贴 35 元，乘客共补贴 0 100 元。
- 平台稳定阶段：
  - 对司机被动接收调度（平台派单）的补贴方案如下：
    - 早高峰时段：指派从四五环间到三四环间的出租车，单位时间指派 0.0056 辆，每辆车每趟补 15.01 元；
    - 晚高峰时段：指派从四五环间到二三环间的出租车，单位时间指派 0.1777 辆，每辆车每趟补 22.82 元；
  - 对司机被主动调度的补贴方案如下：
    - 早高峰时段：二环内每单补 7 元、二三环间、三四环间、四五环间每单补 1 元；
    - 晚高峰时段：二环内每单补 5 元，二三环间每单补 1 元，三四环间每单补 3 元，四五环间每单 8 元。
  - 对拼车乘客的补贴方案如下：
    - 早高峰时段：对每位拼车乘客补贴 57%；
    - 晚高峰时段：对每位拼车乘客补贴 49%；

七、 模型的评价及改进

7.1 模型优点

1. 第一问对于所在地不重合的数据点，采取了网格法进行统计。
2. 第二问对于乘客的等待时间，建立了排队模型。
3. 第二问建立了排队模拟系统，能对真实情形进行仿真。
4. 第三问对司机的指派补贴和乘客的拼车补贴均进行了综合考虑公司效益和缓解打车难问题的双目标优化模型。

7.2 模型的改进

由于第二问的模拟系统忽略了里程因素，因而无法准确地根据出租车里程给出补贴方案，需要搜集相关数据，细化第三问的补贴方案。

参考文献

[1] 崔亚明, 徐硕, and 邓强. 浅谈出租车供需平衡的影响因素及指标. 文摘版: 经济管理, (10):35–36, 2015.

[2] 王富喜, 毛爱华, 李赫龙, and 贾明璐. 基于熵值法的山东省城镇化质量测度及空间差异分析. 地理科学, 33(11):1323–1329, 2013.

[3] 吕纪荣, 刘昕, 关凯霖, and 王荣翔. “互联网 +”时代的出租车资源配置. 现代电子技术, 40(2):69–72, 2017.

附录清单

编号	名称
A	北京市 2016 年 8 月 10 日第 8 个时间段出租车分布热力图 HTML 代码
B	北京市 2016.8.10 网约车数据
C	乘客平均满意度与平均等待时间
D	北京市 2016.8.10 日“供求匹配”程度评价指标值
E	出租车乘客排队模型系统模拟的 MATLAB 代码
F	模拟系统流程图

附录 A 北京市 2016 年 8 月 10 日第 8 个时间段出租车分布热力图 HTML 代码

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<script type="text/javascript"
    src="http://api.map.baidu.com/api?v=2.0&ak=OwDGDQPC7jrAZQRvQEGVdmmSGxVIiADc"></script>
<script type="text/javascript"
    src="http://api.map.baidu.com/library/Heatmap/2.0/src/Heatmap_min.js"></script>
<title>热力图功能示例</title>
<style type="text/css">
ul,li{list-style: none;margin:0;padding:0;float:left;}
html{height:100%}
body{height:100%;margin:0px;padding:0px;font-family:"微软雅黑";}
#container{height:90%;width:100%;}
#r-result{width:100%;}
</style>
</head>
<body>
<div id="container"></div>
<div id="r-result">
<input type="button" onclick="openHeatmap();" value="显示热力图"/><input type="button"
    onclick="closeHeatmap();" value="关闭热力图"/>
</div>
</body>
</html>
<script type="text/javascript">
var map = new BMap.Map("container");    // 创建地图实例

var point = new BMap.Point(116.418261, 39.921984);
map.centerAndZoom(point, 15);             // 初始化地图，设置中心点坐标和地图级别
map.enableScrollWheelZoom(); // 允许滚轮缩放

var points =[
{"lat":40.0042,"lng":116.5112,"count":213},
{"lat":39.9626,"lng":116.4627,"count":205},
{"lat":39.8811,"lng":116.3948,"count":205},
{"lat":39.967,"lng":116.4616,"count":196},
{"lat":40.0099,"lng":116.5157,"count":194},
{"lat":39.9972,"lng":116.4877,"count":190},
{"lat":39.8679,"lng":116.3841,"count":187},
{"lat":40.0044,"lng":116.4895,"count":184},
{"lat":40.0048,"lng":116.5026,"count":182},
{"lat":40.0009,"lng":116.5041,"count":178},
{"lat":39.9908,"lng":116.4831,"count":174},
{"lat":40.0685,"lng":116.5898,"count":169},
{"lat":39.9591,"lng":116.4738,"count":167},
{"lat":39.9187,"lng":116.3609,"count":167},
{"lat":39.9883,"lng":116.3561,"count":166},
{"lat":39.9219,"lng":116.3663,"count":165},
{"lat":39.9817,"lng":116.4744,"count":163},
{"lat":40.0245,"lng":116.5242,"count":162},
{"lat":39.8829,"lng":116.3847,"count":160},
{"lat":39.9737,"lng":116.4819,"count":158},
{"lat":39.9765,"lng":116.3836,"count":157},
{"lat":39.9889,"lng":116.4876,"count":157},
{"lat":40.0412,"lng":116.5854,"count":157},
{"lat":39.9443,"lng":116.3666,"count":153},
{"lat":39.9597,"lng":116.4675,"count":153},
```

```
{"lat":39.971,"lng":116.3623,"count":152},
{"lat":40.0477,"lng":116.6298,"count":152},
{"lat":39.921,"lng":116.4731,"count":149},
{"lat":39.9825,"lng":116.4805,"count":149},
{"lat":39.9912,"lng":116.3874,"count":148},
{"lat":39.9257,"lng":116.4686,"count":147},
{"lat":39.9481,"lng":116.4686,"count":147},
{"lat":39.8715,"lng":116.3763,"count":145},
{"lat":40.0004,"lng":116.5052,"count":144},
{"lat":39.995,"lng":116.3386,"count":144},
{"lat":39.9908,"lng":116.3735,"count":144},
{"lat":39.9439,"lng":116.3663,"count":144},
{"lat":39.9925,"lng":116.393,"count":143},
{"lat":39.9376,"lng":116.4753,"count":142},
{"lat":40.0544,"lng":116.6085,"count":139},
{"lat":40.0832,"lng":116.5892,"count":138},
{"lat":39.8688,"lng":116.3977,"count":138},
{"lat":39.9878,"lng":116.4164,"count":138},
{"lat":39.9146,"lng":116.2409,"count":136},
{"lat":39.9139,"lng":116.4659,"count":135},
{"lat":39.9018,"lng":116.3127,"count":134},
{"lat":39.9976,"lng":116.4266,"count":134},
{"lat":39.9446,"lng":116.3189,"count":134},
{"lat":40.0575,"lng":116.5909,"count":134},
{"lat":39.9969,"lng":116.3456,"count":133},
{"lat":40.0099,"lng":116.3914,"count":133},
{"lat":39.8445,"lng":116.2929,"count":133},
{"lat":39.9066,"lng":116.2435,"count":132},
{"lat":39.9,"lng":116.3201,"count":132},
{"lat":40.0697,"lng":116.5942,"count":132},
{"lat":39.982,"lng":116.4279,"count":131},
{"lat":40.0639,"lng":116.5929,"count":130},
{"lat":39.8663,"lng":116.3762,"count":130},
{"lat":39.8732,"lng":116.3783,"count":130},
{"lat":39.9523,"lng":116.4335,"count":130},
{"lat":39.9745,"lng":116.4453,"count":128},
{"lat":40.0267,"lng":116.5214,"count":128},
{"lat":39.8838,"lng":116.3172,"count":127},
{"lat":40.0578,"lng":116.6079,"count":127},
{"lat":39.9896,"lng":116.32,"count":126},
{"lat":39.9753,"lng":116.4465,"count":126},
{"lat":40.0275,"lng":116.534,"count":126},
{"lat":40.0425,"lng":116.5552,"count":125},
{"lat":40.0341,"lng":116.5645,"count":125},
{"lat":39.9761,"lng":116.3253,"count":125},
{"lat":39.8737,"lng":116.4108,"count":125},
{"lat":39.9335,"lng":116.3164,"count":125},
{"lat":39.9273,"lng":116.362,"count":124},
{"lat":40.0507,"lng":116.6001,"count":124},
{"lat":40.0752,"lng":116.5913,"count":123},
{"lat":40.0133,"lng":116.4992,"count":123},
{"lat":39.9323,"lng":116.3627,"count":123},
{"lat":39.9109,"lng":116.3209,"count":123},
{"lat":39.9665,"lng":116.4539,"count":123},
{"lat":39.8735,"lng":116.3593,"count":122},
{"lat":39.9194,"lng":116.4424,"count":121},
{"lat":40.0321,"lng":116.5619,"count":121},
{"lat":39.9083,"lng":116.3261,"count":118},
{"lat":39.9988,"lng":116.4865,"count":118},
{"lat":39.9689,"lng":116.4784,"count":118},
{"lat":40.0226,"lng":116.4684,"count":117},
{"lat":39.9741,"lng":116.469,"count":117},
{"lat":40.0533,"lng":116.6295,"count":117},
{"lat":39.9999,"lng":116.388,"count":117},
```

```
{"lat":39.8645,"lng":116.3887,"count":117},
{"lat":39.9545,"lng":116.4505,"count":117},
{"lat":40.0122,"lng":116.4774,"count":117},
{"lat":39.9775,"lng":116.4743,"count":116},
{"lat":39.9804,"lng":116.4103,"count":116},
{"lat":39.9941,"lng":116.3125,"count":116},
{"lat":39.9148,"lng":116.4986,"count":115},
{"lat":39.8971,"lng":116.664,"count":115},
{"lat":39.9715,"lng":116.4184,"count":115},
{"lat":39.9346,"lng":116.3183,"count":114},
{"lat":39.9912,"lng":116.3272,"count":114},
{"lat":39.9361,"lng":116.4715,"count":114},
{"lat":40.0216,"lng":116.5161,"count":114},
{"lat":39.8684,"lng":116.2921,"count":113},
{"lat":39.9892,"lng":116.4513,"count":113},
{"lat":39.912,"lng":116.4811,"count":113},
{"lat":40.014,"lng":116.3903,"count":113},
{"lat":39.9929,"lng":116.4057,"count":113},
{"lat":39.9426,"lng":116.4396,"count":112},
{"lat":39.9475,"lng":116.3191,"count":112},
{"lat":39.9981,"lng":116.4587,"count":112},
{"lat":39.978,"lng":116.4409,"count":111},
{"lat":39.9745,"lng":116.3419,"count":111},
{"lat":39.9725,"lng":116.3692,"count":111},
{"lat":40.0019,"lng":116.486,"count":111},
{"lat":39.9578,"lng":116.3854,"count":110},
{"lat":39.8804,"lng":116.6754,"count":110},
{"lat":39.9163,"lng":116.3649,"count":110},
{"lat":39.9686,"lng":116.3153,"count":109},
{"lat":39.9201,"lng":116.2896,"count":109},
{"lat":39.9664,"lng":116.321,"count":108},
{"lat":40.0538,"lng":116.5796,"count":108},
{"lat":39.9419,"lng":116.4658,"count":108},
{"lat":39.8966,"lng":116.3328,"count":107},
{"lat":39.9694,"lng":116.3772,"count":107},
{"lat":39.9863,"lng":116.3873,"count":106},
{"lat":39.9769,"lng":116.3667,"count":106},
{"lat":39.911,"lng":116.3677,"count":106},
{"lat":39.9707,"lng":116.3973,"count":105},
{"lat":39.8443,"lng":116.2902,"count":105},
{"lat":39.8927,"lng":116.3518,"count":105},
{"lat":39.9062,"lng":116.3577,"count":105},
{"lat":39.8815,"lng":116.6853,"count":104},
{"lat":39.8988,"lng":116.6754,"count":104},
{"lat":39.8937,"lng":116.6646,"count":104},
{"lat":39.885,"lng":116.6736,"count":103},
{"lat":39.8796,"lng":116.3847,"count":103},
{"lat":39.8688,"lng":116.2849,"count":103},
{"lat":39.9066,"lng":116.4673,"count":103},
{"lat":39.9236,"lng":116.3133,"count":102},
{"lat":39.9883,"lng":116.4551,"count":102},
{"lat":39.9225,"lng":116.2876,"count":102},
{"lat":39.8362,"lng":116.3123,"count":102},
{"lat":39.9928,"lng":116.318,"count":102},
{"lat":39.7626,"lng":116.3367,"count":102},
{"lat":40.0895,"lng":116.5956,"count":101},
{"lat":39.898,"lng":116.6951,"count":101},
{"lat":39.8994,"lng":116.3531,"count":101},
{"lat":39.8768,"lng":116.3203,"count":101},
{"lat":40.0081,"lng":116.4774,"count":101},
{"lat":39.8732,"lng":116.7025,"count":101},
{"lat":39.8596,"lng":116.3747,"count":100},
{"lat":39.9267,"lng":116.4412,"count":100},
{"lat":39.9209,"lng":116.4569,"count":100},
```

```
{"lat":40.0084,"lng":116.4692,"count":99},
{"lat":40.0034,"lng":116.3827,"count":99},
{"lat":40.0085,"lng":116.4868,"count":99},
{"lat":39.9483,"lng":116.3301,"count":99},
{"lat":39.9107,"lng":116.2843,"count":98},
{"lat":39.9859,"lng":116.5022,"count":98},
{"lat":39.97,"lng":116.4922,"count":98},
{"lat":39.9186,"lng":116.4982,"count":98},
{"lat":39.9022,"lng":116.2257,"count":98},
{"lat":39.866,"lng":116.3782,"count":98},
{"lat":39.9394,"lng":116.4344,"count":98},
{"lat":39.9095,"lng":116.2301,"count":97},
{"lat":39.8932,"lng":116.6863,"count":97},
{"lat":39.9051,"lng":116.2131,"count":97},
{"lat":39.9591,"lng":116.4526,"count":96},
{"lat":40.0637,"lng":116.6099,"count":96},
{"lat":39.8497,"lng":116.3739,"count":96},
{"lat":39.8942,"lng":116.6802,"count":96},
{"lat":39.8429,"lng":116.3764,"count":96},
{"lat":39.8761,"lng":116.4307,"count":96},
{"lat":39.9324,"lng":116.2026,"count":96},
{"lat":39.9238,"lng":116.3716,"count":95},
{"lat":40.0687,"lng":116.5837,"count":95},
{"lat":40.0344,"lng":116.5435,"count":95},
{"lat":39.9871,"lng":116.4342,"count":95},
{"lat":39.9598,"lng":116.3186,"count":95},
{"lat":39.9957,"lng":116.3901,"count":94},
{"lat":39.8927,"lng":116.6643,"count":94},
{"lat":39.951,"lng":116.4391,"count":94},
{"lat":40.137,"lng":116.6563,"count":94},
{"lat":39.9269,"lng":116.3061,"count":93},
{"lat":40.0116,"lng":116.4796,"count":92},
{"lat":39.9066,"lng":116.3192,"count":92},
{"lat":39.9455,"lng":116.3537,"count":92},
{"lat":40.0758,"lng":116.5732,"count":91},
{"lat":39.9719,"lng":116.3551,"count":91},
{"lat":39.9324,"lng":116.489,"count":91},
{"lat":39.9809,"lng":116.3171,"count":91},
{"lat":39.8626,"lng":116.3857,"count":91},
{"lat":39.9056,"lng":116.2533,"count":91},
{"lat":40.0144,"lng":116.535,"count":91},
{"lat":39.9377,"lng":116.495,"count":90},
{"lat":39.973,"lng":116.3591,"count":90},
{"lat":39.9181,"lng":116.3138,"count":90},
{"lat":39.8718,"lng":116.2796,"count":90},
{"lat":39.8795,"lng":116.2896,"count":90},
{"lat":39.7522,"lng":116.1503,"count":89},
{"lat":39.9558,"lng":116.2896,"count":89},
{"lat":39.8373,"lng":116.3015,"count":89},
{"lat":39.9599,"lng":116.3227,"count":89},
{"lat":39.9064,"lng":116.6784,"count":89},
{"lat":39.8935,"lng":116.3096,"count":89},
{"lat":39.7962,"lng":116.5155,"count":89},
{"lat":39.9598,"lng":116.287,"count":89},
{"lat":39.9122,"lng":116.4394,"count":88},
{"lat":39.9627,"lng":116.4898,"count":88},
{"lat":39.9714,"lng":116.4992,"count":88},
{"lat":39.9803,"lng":116.3327,"count":88},
{"lat":39.9084,"lng":116.3338,"count":87},
{"lat":39.7489,"lng":116.3419,"count":86},
{"lat":39.7678,"lng":116.3392,"count":86},
{"lat":39.9359,"lng":116.2811,"count":86},
{"lat":39.9793,"lng":116.4558,"count":86},
{"lat":39.9289,"lng":116.1824,"count":86},
```

```
{"lat":39.8847,"lng":116.4,"count":86},
{"lat":39.927,"lng":116.4598,"count":86},
{"lat":39.9058,"lng":116.2612,"count":86},
{"lat":39.9958,"lng":116.4717,"count":86},
{"lat":39.8616,"lng":116.4046,"count":86},
{"lat":39.8415,"lng":116.3482,"count":85},
{"lat":39.8835,"lng":116.6728,"count":84},
{"lat":40.0117,"lng":116.4163,"count":84},
{"lat":39.9571,"lng":116.4324,"count":84},
{"lat":39.9908,"lng":116.3031,"count":84},
{"lat":39.9047,"lng":116.6996,"count":84},
{"lat":39.8756,"lng":116.348,"count":84},
{"lat":39.8763,"lng":116.4185,"count":83},
{"lat":39.7665,"lng":116.3353,"count":83},
{"lat":39.8975,"lng":116.7042,"count":83},
{"lat":39.9473,"lng":116.4498,"count":83},
{"lat":39.748,"lng":116.1391,"count":83},
{"lat":39.9465,"lng":116.3025,"count":83},
{"lat":39.8881,"lng":116.3931,"count":83},
{"lat":39.8728,"lng":116.6714,"count":83},
{"lat":39.7447,"lng":116.3489,"count":83},
{"lat":39.9079,"lng":116.2484,"count":82},
{"lat":39.9977,"lng":116.4613,"count":82},
{"lat":39.9934,"lng":116.4746,"count":82},
{"lat":39.8563,"lng":116.2862,"count":82},
{"lat":40.0082,"lng":116.4003,"count":82},
{"lat":39.8879,"lng":116.3566,"count":82},
{"lat":40.1293,"lng":116.647,"count":82},
{"lat":39.9375,"lng":116.2077,"count":81},
{"lat":40.0728,"lng":116.5547,"count":81},
{"lat":39.9205,"lng":116.4429,"count":81},
{"lat":39.985,"lng":116.336,"count":81},
{"lat":40.1388,"lng":116.6422,"count":81},
{"lat":39.9779,"lng":116.4913,"count":81},
{"lat":39.9107,"lng":116.6872,"count":81},
{"lat":39.9932,"lng":116.5091,"count":80},
{"lat":39.7402,"lng":116.1391,"count":80},
{"lat":39.8554,"lng":116.4111,"count":80},
{"lat":39.9802,"lng":116.4617,"count":80},
{"lat":39.9767,"lng":116.4605,"count":80},
{"lat":39.8499,"lng":116.2901,"count":80},
{"lat":39.8781,"lng":116.6971,"count":80},
{"lat":39.8662,"lng":116.3963,"count":80},
{"lat":40.0045,"lng":116.4018,"count":79},
{"lat":40.05,"lng":116.5932,"count":79},
{"lat":39.9036,"lng":116.277,"count":79},
{"lat":39.9068,"lng":116.6666,"count":79},
{"lat":40.0473,"lng":116.597,"count":78},
{"lat":39.9846,"lng":116.3638,"count":78},
{"lat":39.9852,"lng":116.5142,"count":78},
{"lat":39.7902,"lng":116.5268,"count":78},
{"lat":39.9113,"lng":116.4904,"count":78},
{"lat":40.0034,"lng":116.419,"count":78},
{"lat":39.9178,"lng":116.4173,"count":78},
{"lat":39.7396,"lng":116.1345,"count":78},
{"lat":39.979,"lng":116.4962,"count":78},
{"lat":39.8712,"lng":116.311,"count":78},
{"lat":39.9226,"lng":116.2896,"count":77},
{"lat":39.9539,"lng":116.3116,"count":77},
{"lat":39.9418,"lng":116.3068,"count":77},
{"lat":39.8309,"lng":116.3011,"count":77},
{"lat":39.8593,"lng":116.3466,"count":77},
{"lat":39.8979,"lng":116.6639,"count":76},
{"lat":39.8972,"lng":116.3231,"count":76},
```

```
{"lat":39.9444,"lng":116.4565,"count":76},
{"lat":40.0204,"lng":116.4646,"count":76},
{"lat":39.9938,"lng":116.3366,"count":76},
{"lat":39.9469,"lng":116.3375,"count":76},
{"lat":39.8891,"lng":116.6491,"count":76},
{"lat":39.9919,"lng":116.4224,"count":75},
{"lat":39.9103,"lng":116.4335,"count":75},
{"lat":39.9917,"lng":116.3852,"count":75},
{"lat":40.0115,"lng":116.4681,"count":75},
{"lat":40.0428,"lng":116.6138,"count":75},
{"lat":39.9066,"lng":116.3781,"count":75},
{"lat":39.8108,"lng":116.5089,"count":75},
{"lat":39.8913,"lng":116.3603,"count":75},
{"lat":39.8555,"lng":116.2831,"count":75},
{"lat":39.9013,"lng":116.2219,"count":75},
{"lat":39.7947,"lng":116.5304,"count":75},
{"lat":39.9283,"lng":116.4984,"count":75},
{"lat":40.0171,"lng":116.4777,"count":75},
{"lat":39.9655,"lng":116.2857,"count":75},
{"lat":39.9791,"lng":116.4243,"count":74},
{"lat":40.011,"lng":116.4646,"count":74},
{"lat":39.7267,"lng":116.121,"count":74},
{"lat":40.0443,"lng":116.6256,"count":74},
{"lat":39.75,"lng":116.3389,"count":74},
{"lat":39.8995,"lng":116.6813,"count":74},
{"lat":39.8158,"lng":116.499,"count":74},
{"lat":39.9521,"lng":116.4602,"count":74},
{"lat":40.0425,"lng":116.5404,"count":74},
{"lat":39.9149,"lng":116.2824,"count":74},
{"lat":39.8433,"lng":116.3199,"count":73},
{"lat":39.8816,"lng":116.3582,"count":73},
{"lat":40.0209,"lng":116.4562,"count":73},
{"lat":39.9841,"lng":116.4367,"count":73},
{"lat":39.7357,"lng":116.1473,"count":73},
{"lat":39.9751,"lng":116.5014,"count":73},
{"lat":39.9332,"lng":116.1727,"count":72},
{"lat":39.9433,"lng":116.4944,"count":72},
{"lat":39.9393,"lng":116.4945,"count":72},
{"lat":39.9711,"lng":116.3013,"count":72},
{"lat":39.9083,"lng":116.4686,"count":72},
{"lat":40.004,"lng":116.4162,"count":72},
{"lat":40.0804,"lng":116.5584,"count":72},
{"lat":39.9936,"lng":116.4091,"count":72},
{"lat":39.8663,"lng":116.2797,"count":72},
{"lat":39.95,"lng":116.3637,"count":72},
{"lat":40.1376,"lng":116.6638,"count":72},
{"lat":39.8949,"lng":116.2817,"count":72},
{"lat":39.8947,"lng":116.6571,"count":71},
{"lat":40.0884,"lng":116.5992,"count":71},
{"lat":39.9035,"lng":116.4547,"count":71},
{"lat":39.7387,"lng":116.1512,"count":71},
{"lat":39.9093,"lng":116.4839,"count":71},
{"lat":39.9694,"lng":116.3615,"count":71},
{"lat":39.9497,"lng":116.37,"count":71},
{"lat":40.0043,"lng":116.3844,"count":71},
{"lat":39.9556,"lng":116.3284,"count":71},
{"lat":39.7561,"lng":116.3506,"count":70},
{"lat":40.0106,"lng":116.371,"count":70},
{"lat":39.9794,"lng":116.3674,"count":70},
{"lat":39.9495,"lng":116.4928,"count":70},
{"lat":39.8965,"lng":116.3016,"count":70},
{"lat":39.9042,"lng":116.3839,"count":70},
{"lat":40.0382,"lng":116.6048,"count":70},
{"lat":40.1331,"lng":116.6526,"count":70},
```

```
{"lat":39.8694,"lng":116.2733,"count":70},
{"lat":39.8763,"lng":116.7076,"count":69},
{"lat":39.8995,"lng":116.6941,"count":69},
{"lat":39.9177,"lng":116.4844,"count":69},
{"lat":39.8987,"lng":116.6433,"count":69},
{"lat":39.9543,"lng":116.4157,"count":69},
{"lat":39.9143,"lng":116.4593,"count":69},
{"lat":39.969,"lng":116.3877,"count":69},
{"lat":39.8443,"lng":116.3681,"count":68},
{"lat":39.9622,"lng":116.3639,"count":68},
{"lat":39.9069,"lng":116.3092,"count":68},
{"lat":39.9607,"lng":116.3777,"count":68},
{"lat":40.0036,"lng":116.4467,"count":68},
{"lat":39.8432,"lng":116.3815,"count":68},
{"lat":40.0281,"lng":116.6093,"count":68},
{"lat":39.9135,"lng":116.3022,"count":68},
{"lat":40.0301,"lng":116.5292,"count":68},
{"lat":39.8101,"lng":116.5194,"count":67},
{"lat":39.9555,"lng":116.3678,"count":67},
{"lat":39.8798,"lng":116.649,"count":67},
{"lat":39.9651,"lng":116.4762,"count":67},
{"lat":39.9966,"lng":116.3731,"count":67},
{"lat":39.9135,"lng":116.3896,"count":67},
{"lat":39.8926,"lng":116.3965,"count":67},
{"lat":39.9078,"lng":116.3695,"count":67},
{"lat":39.7773,"lng":116.3404,"count":67},
{"lat":39.8755,"lng":116.3433,"count":66},
{"lat":39.9167,"lng":116.3881,"count":66},
{"lat":39.9206,"lng":116.3483,"count":66},
{"lat":40.0262,"lng":116.5424,"count":66},
{"lat":39.9047,"lng":116.3313,"count":66},
{"lat":39.8552,"lng":116.2325,"count":66},
{"lat":39.9032,"lng":116.2162,"count":66},
{"lat":39.9299,"lng":116.4475,"count":65},
{"lat":39.8797,"lng":116.6603,"count":65},
{"lat":39.956,"lng":116.4103,"count":65},
{"lat":39.929,"lng":116.4427,"count":65},
{"lat":39.7719,"lng":116.315,"count":65},
{"lat":39.8914,"lng":116.3818,"count":65}];

if(!isSupportCanvas()){
alert('热力图目前只支持有canvas支持的浏览器,您所使用的浏览器不能使用热力图功能~')
}
//详细的参数,可以查看heatmap.js的文档 https://github.com/pa7/heatmap.js/blob/master/README.md
//参数说明如下:
/* visible 热力图是否显示,默认为true
* opacity 热力的透明度,1-100
* radius 势力图的每个点的半径大小
* gradient {JSON} 热力图的渐变区间 . gradient如下所示
* {
.2:'rgb(0, 255, 255)',
.5:'rgb(0, 110, 255)',
.8:'rgb(100, 0, 255)'
}
其中 key 表示插值的位置, 0~1.
value 为颜色值.
*/
heatmapOverlay = new BMapLib.HeatmapOverlay({"radius":20,"opacity":.5});
map.addOverlay(heatmapOverlay);
heatmapOverlay.setDataSet({data:points,max:100});
//是否显示热力图
function openHeatmap(){
heatmapOverlay.show();
}
```

```
function closeHeatmap(){
heatmapOverlay.hide();
}
closeHeatmap();
function setGradient(){
var gradient = {
0:'rgb(102, 255, 0)',
.005:'rgb(255, 170, 0)',
.01:'rgb(255, 0, 0)'
};
var colors = document.querySelectorAll("input[type='color']");
colors = [].slice.call(colors,0);
colors.forEach(function(ele){
gradient[ele.getAttribute("data-key")] = ele.value;
});
heatmapOverlay2.setOptions({"gradient":gradient});
}
//判断浏览区是否支持canvas
function isSupportCanvas(){
var elem = document.createElement('canvas');
return !!(elem.getContext && elem.getContext('2d'));
}
</script>
```

附录 B 北京市 2016 年 8 月 10 日网约车数据

见附件 A

附录 C 乘客平均满意度与平均等待时间

满意度	二环内	二环到三环之间	三环到四环之间	四环到五环之间
1	1.158536585	0	0	0.392638037
2	0.298013245	0.171779141	0	1.158357771
3	0	0	0.099502488	0
4	1.633587786	0	0.298136646	0
5	0.244755245	0.034285714	0.28	0.071428571
6	0.2268431	0.32059448	0	0.023225806
7	0	0	0.16243149	0
8	0.56284488	0.231495281	0.063100137	0
9	0.184593023	0.1486893	0.24516129	0.390250261
10	0.091179386	0.30125523	0.256615878	0.104718067
11	0.142628205	0.424489796	0.162931034	0.180327869
12	0.160944206	0.338888889	0.160291439	0.189668466
13	0	0.163009404	0.132937685	0.071495767
14	0.370629371	0	0.090653153	0.399595346
15	0.442071197	0.235140431	0.258934592	0.039949271
16	0.067249496	0.223061084	0	0.358541526
17	0	0.009972299	0.390117035	-0.014658727
18	0	0.035687732	0.1426547	0
19	0.329581994	0	0.184354154	0.33313783
20	0.111041009	0.086572438	0	0.207045291
21	0.519951264	0.234118487	0.072520073	0.518431488
22	0.595541401	0.382522671	3.367818149	0.531965769
23	0.697806086	0.095081967	0.894341116	0.206194315
24	0.25087108	0.380952381	0.239229025	0

附录 D 北京市 2016 年 8 也 10 日“供求匹配”程度评价指标

	二环内	二环到三环之间	三环到四环之间	四环到五环之间
1	-3.1104	-0.7880	-1.4487	-2.2742
2	-2.9126	-2.6437	-1.0913	-2.9909
3	-1.6727	-1.9449	-3.3933	-3.5279
4	-7.8232	-0.5167	-5.1657	-2.9985
5	-0.5306	-2.1729	-0.2772	-2.4326
6	-1.7034	-3.4106	-3.1827	-0.4163
7	-3.7252	-12.1014	-12.7048	-4.2342
8	-4.4376	-22.9311	-17.1435	-7.2452
9	-3.0127	-12.7825	-9.9687	-5.6522
10	-5.4469	-4.8659	-5.3392	-0.3041
11	-10.7005	-4.0571	-3.8285	-0.5764
12	-3.6562	-1.7316	-2.7049	-0.0299
13	-11.0260	-3.8446	-8.7078	-1.2781
14	-6.5512	-3.4845	-9.1150	-3.0691
15	-20.1399	-13.7991	-8.9136	-1.3133
16	-8.8447	-9.7048	-8.7337	-1.1043
17	-16.2042	-9.1468	-5.0055	-2.7565
18	-10.2362	-4.3437	-9.2877	-3.5973
19	-4.8349	-0.2682	-8.3237	-0.3699
20	-3.6560	-0.8749	-2.4176	-0.1968
21	-6.9247	-3.7358	-5.2735	-4.5653
22	-6.3034	-5.4780	-14.3864	-7.2936
23	-8.9226	-12.5462	-14.6173	-0.0690
24	-2.8587	-1.0978	-7.3585	-0.4625

附录 E 出租车乘客排队模型系统模拟的 MATLAB 代码

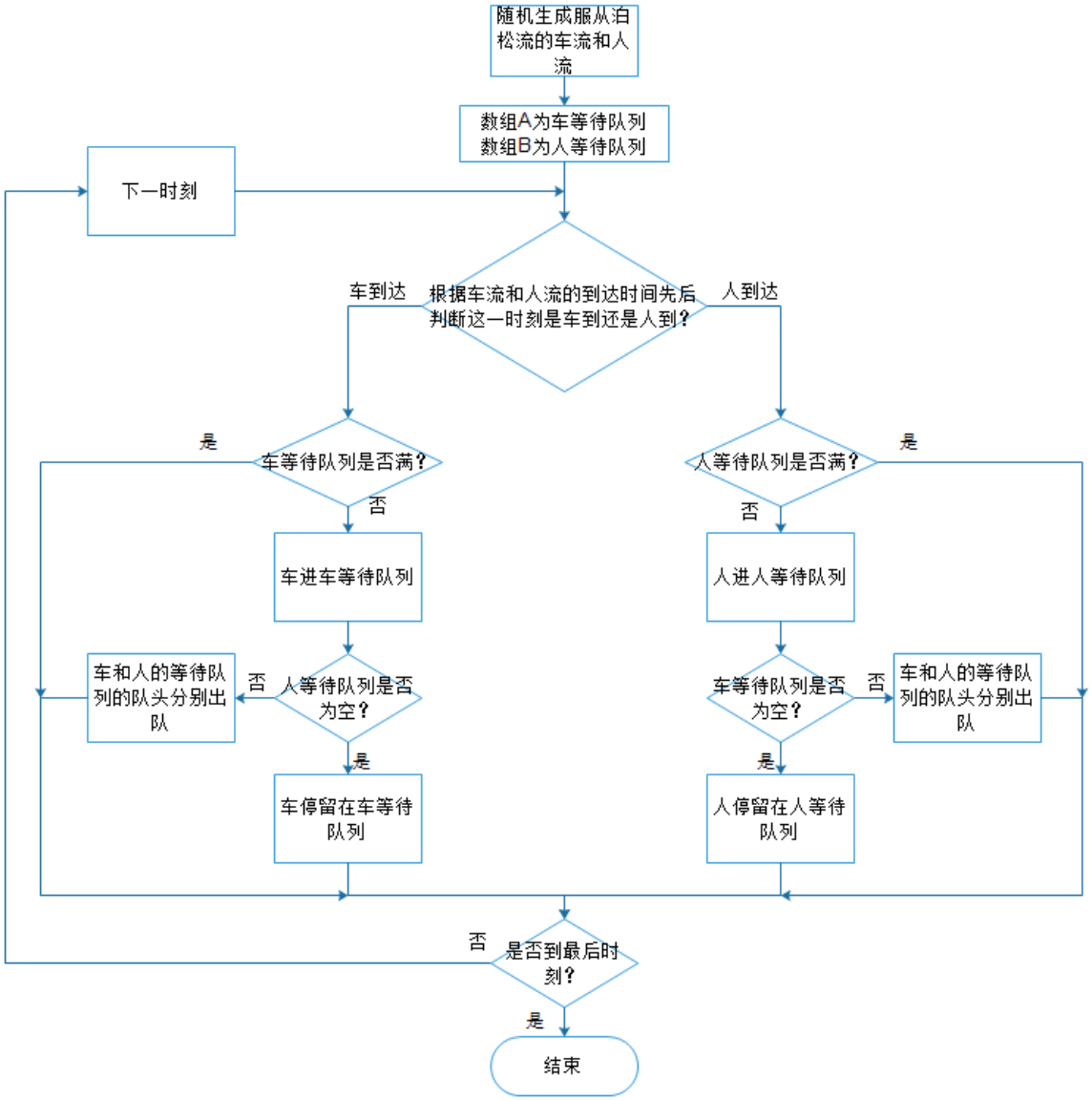
```
%模拟某时间某地的乘车情况
clc,clear
lamda1=0.847096;%单位时间内空闲车到达平均数
lamda2=1.8566;%单位时间内打车人的数量
R=22 ;%乘客等待上限:人队列长度
S=18;%车等待上限:车队列长度
T=60*60;%时间间隔为10min

%随机生成人和车的到达时间
sumTime=0;
i=2;
t1=poissrnd(1/lamda1);
t2=poissrnd(1/lamda2);
arriveTime(1,1)=t1;arriveTime(1,2)=t2;%到达时间矩阵 第一列司机到达 第二列乘客到达
while sumTime<T;
t1=poissrnd(1/lamda1);
t2=poissrnd(1/lamda2);
arriveTime(i,1)=arriveTime(i-1,1)+t1;arriveTime(i,2)=arriveTime(i-1,2)+t2;
sumTime=max(arriveTime(i,1),arriveTime(i,2));
i=i+1;
end
length=size(arriveTime);
top1=0;%队头: 司机到来
top2=0;%队头: 乘客到来
N=1000000;

A=zeros(1,S);%车队列
B=zeros(1,R);%人队列
m=0;
n=0;
j=1;
Q=zeros(1,2);
leaveTime=zeros(length(1),2);
waitTime=zeros(length(1),2);
for i=1:N
if arriveTime(top1+1,1)<arriveTime(top2+1,2)%车来临
if m==S %车队列到达上限
top1=top1+1;%车离开
else
m=m+1;
A(m)=arriveTime(top1+1,1);%车入队
top1=top1+1;
end
else
if n==R
top2=top2+1;%人离开
else
n=n+1;
B(n)=arriveTime(top2+1,2);%人入队
top2=top2+1;
end
end
if m>=1 && n>=1
t=max(A(1),B(1));
leaveTime(j,1)=t;leaveTime(j,2)=t;%离开时间
waitTime(j,1)=t-A(1);%车等待时间
waitTime(j,2)=t-B(1);%人等待时间
j=j+1;
Q=[Q;m n];
if m>n;%车拥堵
for g=1:m-1
```

```
A(g)=A(g+1);
end
elseif m<n%人拥堵
for h=1:n-1
B(h)=B(h+1);
end
else
A(1)=0;
B(1)=0;
end
m=m-1;
n=n-1;
end
if top1==length(1) || top2==length(1)
d=min(top1,top2);
break;
end
end
a=size(waitTime);
plot(1:d,waitTime(1:d,1),'*',1:d,waitTime(1:d,2),'O')
title('四环到五环之间8点')
xlabel('乘车顺序/人')
ylabel('等待时间/s')
legend('司机','乘客')
j=1;
sum=zeros(1,d-10);
for i=1:d-10;
count=0;
for k=1:10
count=count+waitTime(i+k-1,2);
end
sum(j)=count/10;
j=j+1;
end
lengthSum=size(sum);
% figure
% plot(1:lengthSum(2),sum,'*')
for i=1:d
if Q(i,2)==R;
x=i;
break;
end
end
summ=0;
for i=x:d
summ=summ+waitTime(i,2);
end
tt=summ/(d-x)%等待时间均值
```

附录 F 模拟系统流程图


随机生成服从泊松流的车流和人流
数组A为车等待队列
数组B为人等待队列
下一时刻
根据车流和人流的到达时间先后判断这一时刻是车到还是人到？
车到达
人到达
车等待队列是否满？
是
否
人等待队列是否满？
是
否
车进车等待队列
人进人等待队列
车和人的等待队列的队头分别出队
否
人等待队列是否为空？
车等待队列是否为空？
否
车和人的等待队列的队头分别出队
是
是
车停留在车等待队列
人停留在人等待队列
否
是否到最后时刻？
是
结束