湖南省人民政府学位委员会办公室
湖南省教育厅学位管理与研究生教育处

2017 湖南省研究生数学建模竞赛参赛承诺书

我们仔细阅读了湖南省研究生数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：A 题

我们的队号为（填写完整的队号）：201718001003

所属学校（请填写完整的全名）：国防科学技术大学

参赛队员（打印并签名）：

1.

2.

3.

指导教师或指导教师组负责人(打印并签名)：

日期：　2017 年　4　月　25　日

评阅编号（由组委会评阅前进行编号）：

2017 湖南省研究生数学建模竞赛

编 号 专 用 页

评阅编号（由组委会评阅前进行编号）：

评阅记录（可供评阅时使用）：

评阅人										
评分										
备注										

湖南省第三届研究生数学建模竞赛

题 目 出租车合乘业务系统设计

摘 要：

近年来随着移动互联网的普及，以滴滴打车和快的打车为代表的网约车在生活中越来越普遍，与此同时城市车流高峰的叫车难和城市道路拥堵等问题越来越突出。网约车合乘方式具有提高车辆使用效率，降低乘客出行成本和提高网约车司机收入等众多优点，从而广泛受到市场的欢迎。网约合乘能有效发挥其优点关键在于建立与之相适应的数学模型。文章系统分析了城市网约车合乘的全过程，建立了三个模型：合乘模型，车辆调度模型，公平收费模型。

合乘模型主要用来解决乘客间如何进行搭配乘车的问题，此模型利用了聚类的思想，将出发地相近且目的地方向相同的乘客搭配一组乘一辆车。同时考虑到减少调用的车辆，就在安排搭乘时尽量使车满载（最多 3 人一辆）；其次还要考虑打车人的体验，不能让其等待过久，同乘者的出发地点需要尽可能接近。

车辆调度模型主要是针对如何将搭配好的乘客以最快的速度接上车而设计的，通过寻找最短路径安排乘客下车的顺序。此模型以就近乘车为原则，来设计车辆的调度，在合乘模型的基础上提出了两种调度车辆的方案：其一是采用优化匹配的方法来分派车辆，其二是按载人量的多少以确定调车的优先权来进行派车。

公平收费模型提出了一个让乘客和司机都能满意的收费方案，此模型主要考虑到两点：1.合乘过程中同行与单行的距离，2.为了进行合乘需要绕行的距离。通过这两点来进行合理制定收费标准。

根据上述模型，编写 MATLAB 程序，用实际数据代入进行检验，结果表明：合乘模型和车辆调度模型具有较好的效果，能够使网约乘车的优点发挥的更加突出。

关键词：网约合乘，车辆调度，公平收费，MATLAB

一、问题的提出

鼓励出行者搭乘公共交通工具是一缓解城市交通日益拥堵不堪问题的有效方法，其中乘坐出租车是现有公共交通中最为灵活的一种出行方式。出租车系统的高效利用将为缓解交通拥堵和方便乘客出行做出重大贡献。

出租车合乘能够改善打车难，出租车空驶率高的现状，并且能够降低乘客出行成本，提高司机收入，辅助以合理的收费方案，该乘车模式具有极大的施行潜力。目前为充分调动乘客、司机参与合乘的积极性并提高出租车利用效率，关键在于构建模型解决以下几个问题:

（1） 设计高效的合乘方案及其相应算法，使得乘客等待时间尽量短，所需出租车数量尽可能少，并结合出租车的分布情况，优化出租车行驶路径，制定合理有效的出租车调度方案；

（2） 设计与合乘方案相应的车费计算方法；

（3） 考察所建模型在具体问题中的适用性：假设某城市的路网为正方形网格，网格边长 500 米，道路均可双向行驶。已知某时刻之前 3 分钟内的打车需求数据以及当前时间点空驰出租车位置信息，按照所设计的合乘方案给出具体计算结果。乘客等待时间均以当前时刻为 0 时刻开始计时，不考虑之前 3 分钟已经等待时间。

二、模型假设

（1） 选择合乘服务的乘客与司机均遵守合乘规则，即乘客只能等待预定出租车，不能另选其它出租车，司机需服从调度并按预定路线行进；

（2） 假设全部出租车车速相同且全程保持平均车速不变，不考虑司机在接人途中其它的可能时间延误，未上车乘客等待时间仅与预订（合乘）出租车间的道路距离成正比；

（3） 暂不考虑道路车流量限制，城市所给道路均可通行；

（4） 每次叫乘只对应一名乘客；

（5） 对于有特殊原因使出发地和目的地相同的乘客暂不予考虑。

三、符号说明

$person\_ID$：乘客编号

$Full\_Num$：出租车最大载客量

N：尚未安排合乘的乘客数

P_i：尚未安排合乘的乘客中第 i 个乘客出发地和目的地所构成的向量，称之为乘客的行程向量

P_0：尚未安排合乘的乘客中模最大的行程向量，也即直线行程最大者的行程向量，称之为基准向量，对应的乘客称为基准乘客

α_{i0}：乘客 i 的行程向量与基准向量之间的夹角

β：可能与基准乘客构成合乘的乘客之行程向量与基准向量的最大偏差角度

r_{start}：为了使最后上车的乘客与首先上车的乘客间的间隔得到控制，设置未安排合乘的人员起点与基准乘客出发点间的最大可接受距离

Co_Num：可与基准乘客构成合乘的乘客数

四、模型建立

出租车合乘模式可以分为静态组合模式和动态组合模式[1]。静态组合指需要合乘的乘客上车前先组合好，上车后按最佳路线行驶，不再搭载组合以外乘客；动态组合则是指车上已有部分乘客，行驶途中根据即时乘客打车信息，在不超载情况下，再与其他乘客合乘。动态组合模式有部分是基于静态组合模式的，即当前时刻的合乘方案可以在前一时刻设计的合乘方案之基础上进行更新改进，所以静态组合模式研究仍十分必要。针对问题(1)～(3)，本文基于静态合乘组合模式依次建立合乘模型、出租车调度模型以及相应于合乘方案设计的合乘收费模型。

问题(1)

乘客等待时间尽量短及调用出租车数量尽量少的实现取决于有效的合乘方案与出租车调度的协同，模型之间联系紧密，两者存在一均衡区间。

4.1 静态合乘模型

当乘客出行的大致方向相同，同时合乘者的候车时间在乘客的忍受范围内时，不同乘客才有可能构成合乘，由此可以构造适合合乘的约束条件，考虑条件约束的聚类方法，找出有可能构成合乘的乘客集合，选择适合度最大的乘客合乘。为使城市的出租车资源得到最大化利用，这里优先考虑乘客的搭配组合，然后再考虑车与人的分派情况，这也能减少交通高峰期乘客就近择车可能造成局部区域出现打车人多，出租车少的情况。

在满足合乘要求的基础上，乘客行程路线越长，则单人乘车的运营成本越高，单人长距离乘车在费用方面对乘客与司机都不利，而且与远距离乘客同路的乘客相对要多，故从降低出租车空载率考虑，以行程较长乘客的出行方向为选择基准，尽量找到能够与其合乘的乘客。

设出租车满载客量为 3 人，下以乘客 P1～P4 为例，以行程最长的乘客 P1 的行程向量为基准，约束其他行程向量与基准行程向量夹角 $\alpha_{j1}(j=2,3,4)$ 不大于 $\pm\beta$ 的乘客为可能合乘者，同时为使每个合乘人员等待乘车的时间尽量短，约束参与合乘的乘客出发点相距尽量近，其示意图如图 4-1 示：

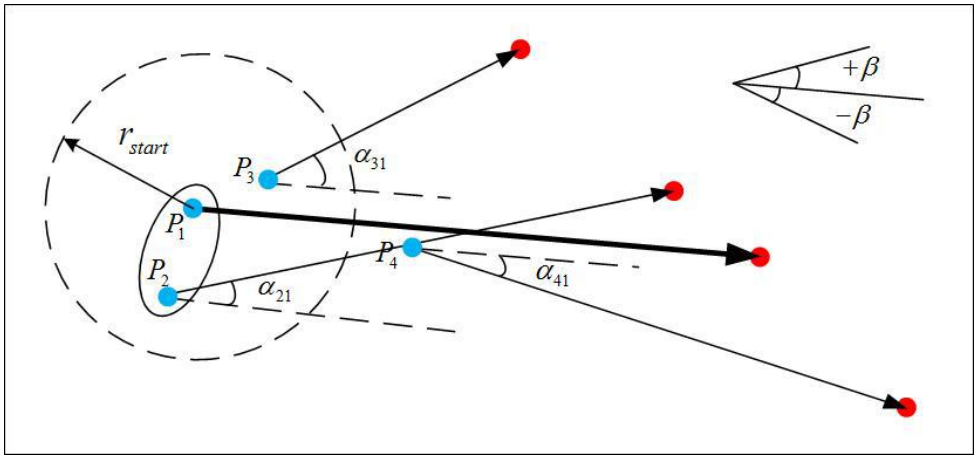

图 4-1 适合合乘的约束条件示意图

则满足与乘客 P1 合乘的仅有乘客 P2，即构成 2 人合乘。若满足合乘条件的人数多于出租车最大载客量，考虑使出租车此次合乘全程绕行路程最短，按与基准向量夹角升序排列，取前 2 人进行 3 人合乘；若没有符合合乘条件乘客，则基准乘客单人乘车；其余情况下分别进行 2 人合乘，3 人合乘。这样一次合乘筛选

可以完成在乘客需求列表中对行程最长乘客的合乘方案设计，认为参与此次合乘乘客的需求得到满足，从乘客需求列表中删除这几位乘客，继续对剩余乘客重复上述操作，直至所有乘客需求得到满足。基于以上思路给出以下合乘方案及算法：

合乘方案算法：

Step 1: 依据当前时刻得到的乘客需求信息，设置乘客需求列表（N 位待设计合乘方案乘客），计算乘客行程向量 P_i；

Step 2: 在需求列表中找出模最大的行程向量，即直线行程最长的乘客作为基准乘客，记录其起终点位置坐标及乘客编号：

$$P_0 = \max\left|P_i\right| \quad ,i \in \{1,2,\ldots,\mathrm{N}\}$$

Step 3: 对于基准乘客，搜寻可能与之合乘乘客，适合合乘约束条件为：

$$s.t.\begin{cases} \alpha_{i0} \le \pm\beta & ,i \in \{1,2,\ldots,\mathrm{N}\} \text{且} P_i \ne P_0 \\ r_{i0} \le r_{start} & ,i \in \{1,2,\ldots,\mathrm{N}\} \text{且} P_i \ne P_0 \end{cases}$$

记录满足约束的乘客集合 $Co = \{person\_ID \,|\, \text{满足合乘条件的乘客}\}$，及其起终点位置信息。满足上述条件，能与基准乘客构成合乘的乘客数为 Co_Num：

$$0 < Co\_Num < N - 1$$

Step 4: 基于 Co_Num 与出租车满载量 $Full\_Num$，进行合乘方案判断：

（1） $Co\_Num + 1 \leqslant Full\_Num$

合乘乘客总人数为：$Co\_Num + 1$，即构成($Co\_Num + 1$)人合乘模式，1 人合乘模式也即没有合乘者，基准乘客单独乘车的情况。同时定义单车空载率：

$$\varphi_{si} = \left(\frac{Co\_Num + 1}{Full\_Num}\right)_i$$

（2） $Co\_Num + 1 > Full\_Num$

此时进行进一步寻优，找寻与基准乘客最适合进行满载合乘的 $Full\_Num$-1 位乘客，使绕行距离最短。对满足合乘约束条件的乘客按其与基准向量夹角升序排列：

$$\underbrace{\alpha_{j,0} < \alpha_{j+1,0} < \ldots < \alpha_{j+Full\_Num-1,0}}_{(Full\_Num\text{-}1)\text{个合乘乘客}} < \ldots < \alpha_{j+Co\_Num-1,0}$$

$$j,\ldots,j+Co\_Num-1 \in Co = \{person\_ID \,|\, \text{满足合乘约束条件的乘客}\}$$

顺序取前 $Full\_Num$-1 个乘客与基准乘客合乘，该情况下出租车为满载合乘模式。

Step 5: 从乘客需求列表中删除已分配合乘的乘客，即更新乘客需求列表；

Step 6: 判断剩余未完成合乘方案设计的乘客人数是否为 0，不为 0，则进行跳至 Step 2，否则进行下一步；

Step 7: 结束此次乘客合乘方案设计，输出所有乘客的合乘方案。

该算法中涉及 2 个可动态调节的参数：β 和 r_{start}，可以根据实际情况结合出租车信息进行调整。程序流程图如下图所示：

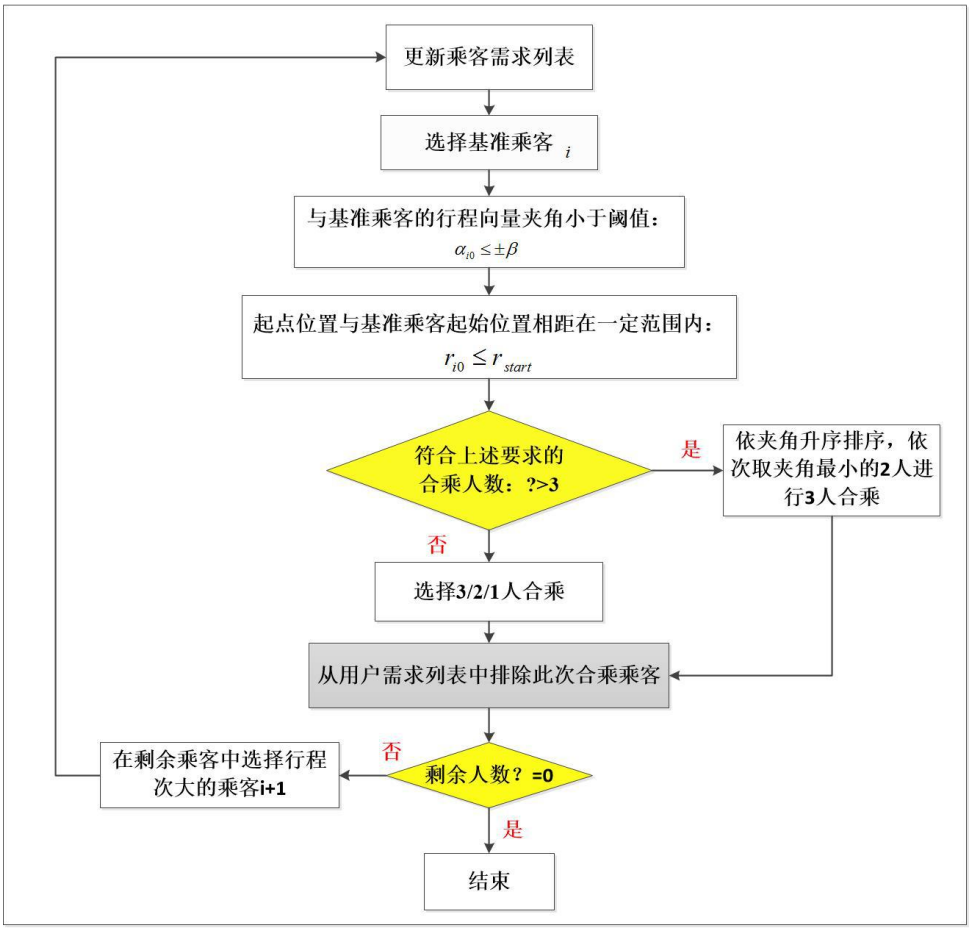

图 4-2 合乘方案算法示意图

4.2 出租车调度模型

以上合乘模型可以总体上使参与调度的出租车空载率最低，达到充分利用出租车资源的目的，但仍需要考虑乘客的等待时间最短问题，基于此提出了两种出租车调度方案：

方案一

从乘车者心理考虑，调度方案应让乘客平均等待时间最短。将乘客按照合乘一辆车进行分组，N 位乘客可分为 K 组，也即现需要从所有可用 M 辆出租车中调度 K 辆出租车去满足所有乘客打车需求，同时需考虑使出租车到达乘客起始位置距离尽量短，也即乘客等待时间在合理范围内。

乘客组顺序排列，并对所有出租车编号，定义第 j 个出租车到第 K_i 个乘客组位置坐标重心的距离为 d_{j,K_i}，称为第 j 辆车到第 K_i 组乘客的距离。则 K 组乘客与 M 辆出租车间有一 K×M 的距离矩阵。实际情况下出租车总数量 M（供给）一般大于需要的数量 K，需要搜寻使距离之和最小的“人-车”配对方案，记一个配对方案：

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{iK}) \quad , x_{i1}, \ldots, x_{iK} \in \{1, 2, \ldots, \mathrm{M}\}$$

其中 x_{i1} 为与第一组乘客对应的出租车编号,依次类推，则对应距离：

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{iK})$$

距离 d_{ij} 与配对车辆编号 x_{ij} 一一对应，最佳分配方案应对应于总距离和最小：

$$F(X_i)=\min(\sum_{j=1}^{K}d_{ij})$$

同时因乘客候车耐心有限，出租车距乘客组距离不能过远，所以每次配对方案中：

$$\max(d_{ij})\le d_{\max}\ ,j=1,2,\ldots,K$$

其中 $d_{\max}$ 为接到乘客前出租车行驶的最大距离。

由于配对方案众多，用粒子群算法[2]以最小化问题 $f(X)$ 为目标函数寻最优解。

适应度函数：
$$f(X_i)=\delta(X_i)\sum_{j=1}^{K}d_{ij}$$

上式中：

$$\delta(X_i)=\begin{cases}1 & ,d_{i1},d_{i2},\ldots,d_{iK}\le d_{\max}\\ 10 & ,\exists d_{ij}>d_{\max},j=1,2,\ldots,K\end{cases}$$

具体算法：

设微粒 i 的当前位置即为：$X_i=(x_{i1},x_{i2},\ldots,x_{iK})\ ,x_{i1},\ldots,x_{iK}\in\{1,2,\ldots,\mathrm{M}\}$

微粒 i 的当前飞行速度：$V_i=(v_{i1},v_{i2},\ldots,v_{iK})$ 且 $v_{\max}=kx_{\max},0.1\le k\le 1$

微粒 i 经历的最好位置：$P_i=(p_{i1},p_{i2},\ldots,p_{iK})$

微粒 i 当前的最好位置：

$$P_i(t+1)=\begin{cases}P_i(t), & f\left(X_i(t+1)\right)\ge f\left(P_i(t)\right)\\ X_i(t+1), & f\left(X_i(t+1)\right)<f\left(P_i(t)\right)\end{cases}$$

适应度函数：
$$f(X_i)=\delta(X_i)\sum_{j=1}^{K}d_{ij}$$

微粒种群规模：s

微粒群中所有微粒经过的最好位置为：$P_g(t)\in\{P_1(t),P_2(t),\ldots,P_s(t)\}$

算法的速度进化方程：

$$v_{ij}(t+1)=v_{ij}(t)+c_1r_{1j}\left[p_{ij}(t)-x_{ij}(t)\right]+c_2r_{2j}\left[p_{gj}(t)-x_{ij}(t)\right]$$

上式中 $c_1=c_2=2$，r_1 和 r_2 为(0,1)之间均匀分布的随机数，t 表示当前时刻

位置进化方程：$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1)$

算法步骤：

Step 1: 确定粒子群规模 s，设定迭代上限次数，用随机方法产生每个粒子的初始位置和初始速度；

Step 2: 计算所有微粒的适应度值；

Step 3: 更新微粒当前最好位置；

Step 4: 更新微粒的全局最好位置,将当前全局最好位置适应度值与当前各粒子最好位置适应值比较，若某个的微粒的当前最好值更好，则更新全局最好位置；

Step 5: 根据速度进化方程和位置进化方程得到下一代微粒的新位置；

Step 6: 若不满足终止条件，则返回 Step 2。

方案二

假设出租车均为 4 座车，即除司机外至多可搭乘 3 位客人。乘客组可以就近寻找距离最近的出租车，从合乘设计模型的特性考虑，按同一出租车搭载的合乘人数，乘客组可以分为：3 人合乘，2 人合乘，1 人合乘。其中 1 人合乘也即没有找到合乘伙伴的乘客需要单人乘车，造成这种结果的可能原因是由于乘客行程方向偏离城市人群集中区域，或者乘客本身在远离市中心的地区，人群密度极为稀疏，不易找到合乘伙伴，同时这种情况下出租车也更难打到，往往需要出租车从较远的地方行驶过来，这时乘客的等待时间不可避免的将明显长于在市区中心附近人群密度和出租车密度均较高的地方打车所需候车时间。

实际生活中，城市中人群密集的区域往往出租车密度也较大，受人们日常出行活动区域的影响，车流量也常集中在某一区域内或部分方向上，这种情况下多人合乘的可能性更大。从以上方面考虑，3 人合乘的乘客组靠近出租车密度较大的地方的可能性较大，1 人合乘为远离人群及出租车密度小的地方的可能性较大，要使乘客平均等待时间最短，故可按就近原则，先对 3 人合乘组合进行出租车配对，再从剩余可用出租车中调度出租车来搭乘 2 人合乘组合，最后对 1 人乘客分派出租车，这样使乘客平均候车时间最短，在由前文确定出的合乘模型基础上，将乘客组按人数分为 3 类，沿用方案一中出租车到乘客组距离 d_{j,K_i} 的定义，按以下步骤进行出租车调度：

Step 1: 3 人合乘组合顺序按最近距离选择出租车，每次分派后将已分派的出租车从可用出租车列表中删除，即出租车列表更新；

Step 2: 类似于 Step 1 操作，对 2 人合乘组合的乘客进行出租车分派及更新可用出租车列表；

Step 3: 类似于 Step 1 操作，从剩余出租车列表中给单人乘客分配出租车。

由于时间上的限制，结合合乘方案，本文最终采用方案二所设计的出租车调度方案。接下来以乘客组合起终点坐标及出租车初始位置信息（坐标），遍历可能行车路径寻求最短路，以此确定出租车接人顺序及乘客下车顺序。

问题(2)

4. 3 合乘收费模型

针对问题二建立合乘收费模型，关键点在于对乘客及司机的公平性权衡。合理的合乘收费标准才会使更多的乘客与司机选择合乘业务。通常合乘条件下乘客所付车费会低于不合乘情况。合乘计费标准同时需要表达形式简单，便于乘客下车时的费用计算。

对乘客 i 来讲，一次合乘总路线可分为乘客 i 的独乘距离 S_1，2 人合乘距离 S_2，以及 3 人合乘距离 S_3 [3]。合乘总距离为 $S_{total} \geq \sum_{i=1}^{3} S_i$，$d_{\min}$ 为乘客单独乘车达到目的地的最短行程距离，则乘客绕行路程：

$$L = \sum_{i=1}^{3} S_i - d_{\min}$$

由乘客的合乘情况，定义 A_i 为对应于 i 人合乘路段的单价（每公里收费），A_1 表示乘客独乘的单价；A_2 表示乘客与 1 人合乘的情况下的单价；A_3 表示乘客与 2 人合乘的情况下的单价。则乘客一次合乘的费用为：

$$W=\sum_{i=1}^{3}A_iS_i-A_{ex}L$$

其中 A_{ex} 为由于绕行原因给予乘客的路程补偿费用单价。路段单价 A_i 主要与出租车运营成本 C，税率 ψ 以及司机的合理利润 λ 这几个因素有关：

$$A_i=A_i(C,\psi,\lambda)$$

通常可以根据实际调研得到这几个变量值的数据，可进行模型拟合，得到路段单价的计算模型。

问题(3)

五、模型求解

针对问题 3 所给背景应用合乘模型，及出租车调度方案二，进行实际合乘规划调度，依合乘模型设计的方案结果数据输出为-plan.txt 文件，其中出租车的调度模型采用方案二，内容包括合乘的乘客编号、对应的出租车编号及乘客上下车次序。部分结果如下：

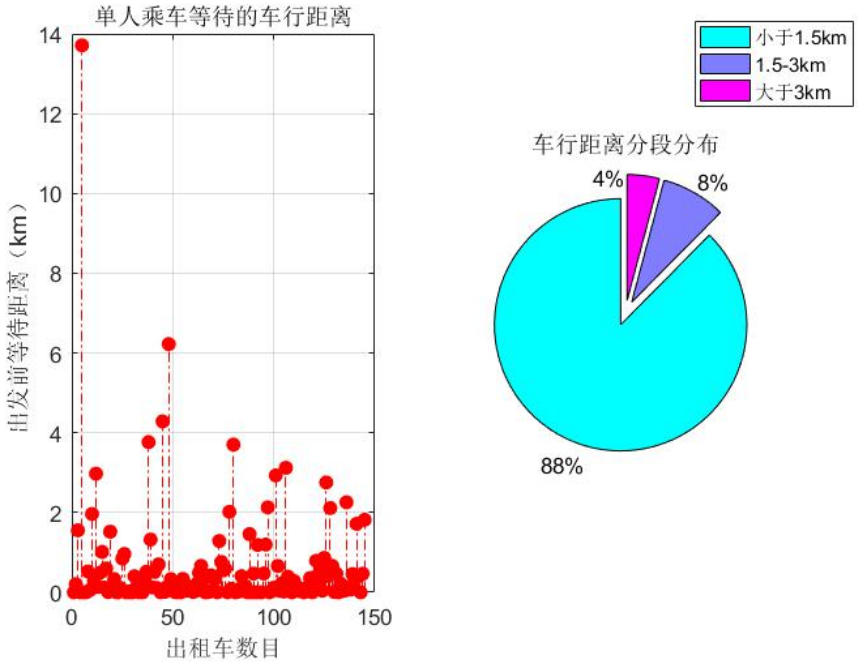

图 5-1 单人乘车时接乘客车需行驶的距离及分段分布

在单人乘车的情况下，出租车为了接到乘客需要行驶一定的距离，画出距离的点散图，如图 5-1（左）所示， 再将行驶的距离进行分段统计作出相应的饼状图，如图 5-1（右）所示，由图可得：以本文所建立的模型来安排乘车，有 88%的车需要行驶 1.5km 以下就可以接到乘客，有 8%的车需要行驶 1.5-3km 即可接到乘客，如果车速为 30km/h（这在城市中是合理的），行驶 3km 需要 6min，照此计算，出租车在接到请求之后，有 96%的乘客只需等待 6min 就可以乘车，这种等待是可接受的。但是也要看到，有 4%的乘客为了乘车需要等待的时间会超过 6min，最长的等待时间达 28min，分析其原因是：有极少数乘客处在相对偏远地区，附近并没有出租车，为了接这类乘客，需要走很远的路，而且这样的情况多出现在单人乘车中，这也正说明了原因在于此。

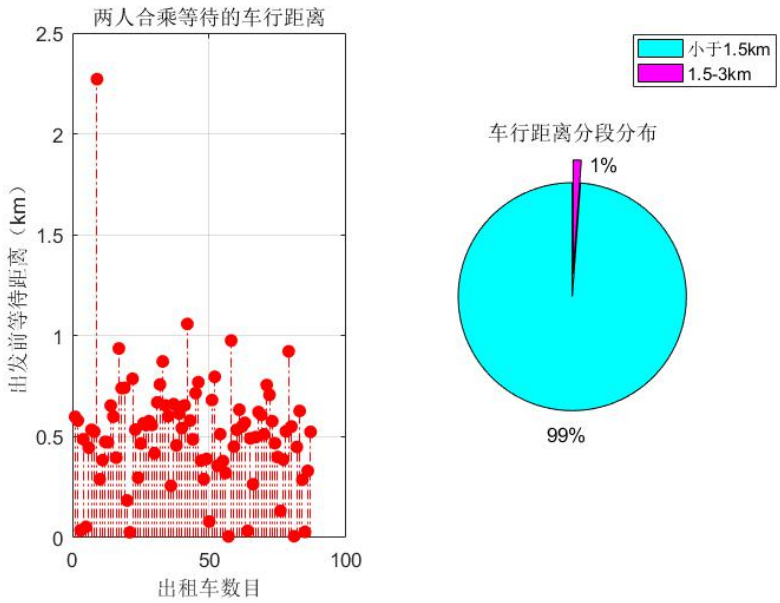

图 5-2 两人合乘时接乘客车需行驶的距离及分段分布

在两人合乘的情况下，出租车为了接到这两个乘客需要行驶一定的距离，画出距离的点散图，如图 5-2（左）所示， 再将行驶的距离进行分段统计作出相应的饼状图，如图 5-2（右）所示，图中显示：模型所安排的两人乘车组，有 99%的车需要行驶 1.5km 以下就可以接到两位乘客，仅有 1%的车需要行驶 1.5-3km 才能接到乘客，如果车速为 30km/h， 99%的两人乘客组合等待时间不超过 3min 就都能够乘车，这种等待时间显然是可接受的。而且由左图结合计算可以看出，两人组乘客最长的等待时间不会超过 5min，即两人在 5min 内都能上车。

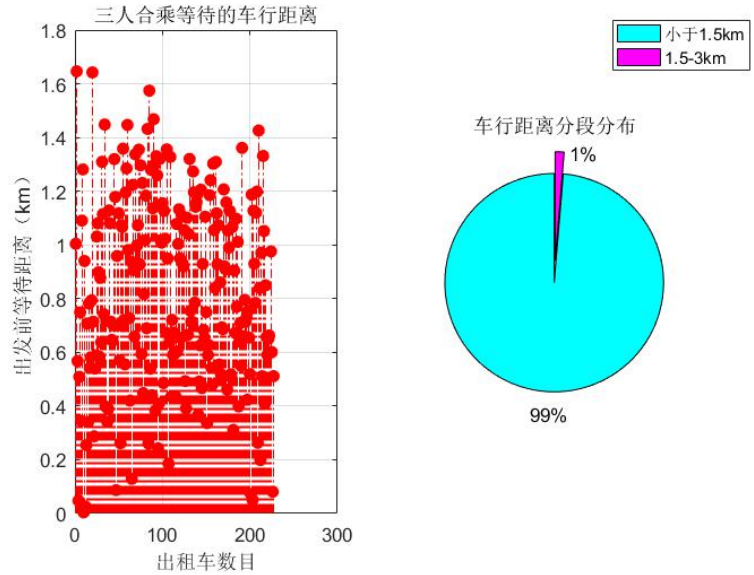

图 5-3 三人合乘时接乘客车需行驶的距离及分段分布

作出在三人合乘的情况下出租车为了接到这三个乘客需要行驶的距离点散图，如图 5-3（左）所示， 再将行驶的距离进行分段统计作出相应的饼状图，

如图 5-3（右）所示，图中显示：模型所安排的三人乘车组，有 99%的车需要行驶不到 1.5km 就可以完全接到三位乘客，仅有 1%的车需要行驶 1.5-3km 方可接到三个乘客，车速仍按 30km/h 计算， 99%的三人乘客组合等待时间不超过 3min 就都能够乘车，这是可接受的。而且由左图结合计算可以看出，三人组乘客最长的等待时间不会超过 3.6min，即三人在 3.6min 内都能上车。

综合上述分析可得：文章所建立的合乘模型和车辆调度模型能够合理的安排所有乘客乘车，单人乘车、两人乘车、三人乘车的安排结果都比较理想，模型也并不复杂。此外，依据这两种模型的安排，出租车为了接到乘客所需要行走的路程也相对较小。因此，文章构建的模型是可靠的，且可操作性强。

六、模型评价

优点：

（1）模型深入分析了可能构成合乘的约束条件，并从使用车数量尽可能的少，乘客相对等待时间较短出发，较为有效和快速地为乘客制定合乘方案，满足所有乘客的打车需求。算法简单，可行性较高，计算方便；

（2）合成方案与出租车调度模型联系紧密，使乘客平均等待时间最短的条件下找到合适的配对出租车；

（3）合乘计价方式既可用于静态合乘，也可用于动态合乘，相对公平合理，同时考虑到补助乘客绕行损失和司机基本利润收入，有利于鼓励乘客选择合乘业务。

缺点：

（1）合乘模型只能计算静态时刻合乘方式，不能用于车辆已经调度完毕后，后一时刻增加乘客的安排。

（2）车辆调度上可能不是全局最优的。

七、模型改进

（1）当车上有乘客但空载率不为零时，不把此车从序列中排除，下一时刻新增乘客时，也把此车也考虑进去成为一个备选车辆，实现动态调度；

（2）偏远地区叫车总会出现等待时间过长，对接人司机来说成本更高，对于此种情况的计价方式有待进一步讨论；

（3）考虑道路实际情况，比如可能存在的交通拥堵等；

（4）确定调度方案后，在出租车的接人顺序上采用最短路确定，这可能导致部分合乘乘客绕行路程增加，可依据路径规划进行进一步优化。

参考文献

[1] 罗超，韩直，乔晓青.城市出租车合乘技术研究.交通运输工程与信息学报.12:79-86. 2014

[2] 张薇，何瑞春，肖强，马昌喜.出租车合乘定价多目标优化研究.武汉理工大学学报.36(6):1105-1109. 2015

[3] 供熙桢，杨珍.基于公平性的出租车合乘算法研究.长沙大学学报.29(2):77-80.2015

附录

```
%使用 Matlab 软件编程
clear
close all
clc
%%导入原始数据乘客乘车信息和出租车信息
requests_data=load('requests.mat');
taxi_data=load('taxi.mat');
requests=requests_data.requests;
taxi=taxi_data.taxi;
%%设计合乘
 Set_Distance=0.8;%设置距离
 Set_Angle=pi/10;%设置角度
 CP1=[];CP2=[];CP3=[];
[ Requests ] = Delete_Zero( requests );%删除不动的乘客及加编号
symble=1;
while symble
    [ Requests ]=Add_Number_Requests( Requests );%加编号
    [ Passenger_maxdistance ] = Search_maxdistance( Requests );%寻找最大距离
    [ PC_Start ] = PossibleCooperator_Start( Requests , Passenger_maxdistance ,
Set_Distance );%按就近原则寻找可能的合乘
    [ Cooperator_Passenger ] = Cooperator_Angle( PC_Start , Set_Angle );%按角度
寻找可能的合乘
    [ CP1 , CP2 , CP3 ] = Statistics_Group( Cooperator_Passenger , CP1 , CP2 ,
CP3 );%分组统计
    [ Requests , symble ] = Delect_Passenger( Requests , Cooperator_Passenger );
    [ Requests ]=Delect_Number_Requests( Requests );%去编号
end
CP1=CP1(:,1:5);%清除临时编号
CP2=CP2(:,1:5);%清除临时编号
CP3=CP3(:,1:5);%清除临时编号
%%分派车辆
Lefted_Taxi=taxi;
[ Assignment_CP3_Result , Lefted_Taxi ]=Assignment_CP3( Lefted_Taxi , CP3 );%
给三人派车及下车顺序
[ Assignment_CP2_Result , Lefted_Taxi ]=Assignment_CP2( Lefted_Taxi , CP2 );%
```

```
给两人派车及下车顺序
[ Assignment_CP1_Result , Lefted_Taxi ]=Assignment_CP1( Lefted_Taxi , CP1 );%
给一人派车
sumNum=size(Assignment_CP1_Result)+size(Assignment_CP2_Result)+size(Assign
ment_CP3_Result);
SumNum=sumNum(1);%总车辆数
%统计分析
[ Proportion ]=Statistics_Analysis( Assignment_CP1_Result ,
Assignment_CP2_Result , Assignment_CP3_Result );
%输出TXT文件
Output_to_Txt( Assignment_CP1_Result , Assignment_CP2_Result ,
Assignment_CP3_Result )


function [ Requests ] = Delete_Zero( requests )
%Delete_Zero  删除不动的乘客
%     requests：输入原始数据
%     Requests：删除后的数据
Size=size(requests);
Num=1:Size(1);
requests=[requests Num'];%加编号
a=sqrt((requests(:,4)-requests(:,2)).^2+(requests(:,3)-requests(:,1)).^2);
Requests=zeros(length(a),5);j=0;
for i=1:length(a)
    if a(i)~=0
        j=j+1;
        Requests(j,:)=requests(i,:);
    end
end
Requests=Requests(1:j,:);
end


function [Requests]=Add_Number_Requests( requests )
%Number_Requests  加编号函数
```

```
%     requests：未乘车人员
%     Requests：加编号的未乘车人员
Size=size(requests);
Num=1:Size(1);
Requests=[requests Num'];%加编号
end


function [ Passenger_maxdistance ] = Search_maxdistance( Requests )
%Search_maxdistance 寻找最大距离
%     Requests：剩余未乘车人员
%     Index_maxdistance：最大距离索引号

Distance=sqrt((Requests(:,3)-Requests(:,1)).^2+(Requests(:,4)-Requests(:,2)).^2);
[~,Index_maxdistance]=max(Distance);
Passenger_maxdistance=Requests(Index_maxdistance,:);
end


function [ Possible_Cooperator_Start ] = PossibleCooperator_Start( requests ,
P_maxdistance , Set_Distance )
%PossibleCooperator_start 按就近原则寻找可能的合乘
%     requests：剩余未乘车人员
%     P_maxdistance：未乘车的乘行距离最远的人员
%     Set_Distance：设置距离
%     Possible_Cooperator_Start：可能的合乘

Distance=abs(requests(:,1)-P_maxdistance(1))+abs(requests(:,2)-P_maxdistance(2));
%计算棋盘距离
Size=size(requests);
Possible_Cooperator_Start=zeros(Size);
k=0;Index=P_maxdistance(6);
for i=1:Size(1)
    if i~=Index&&Distance(i)<=Set_Distance
        k=k+1;
```

```
            Possible_Cooperator_Start(k,:)=requests(i,:);
        end
    end
    Possible_Cooperator_Start(k+1,:)=requests(Index,:);
    Possible_Cooperator_Start=Possible_Cooperator_Start(1:k+1,:);
    end


    function [ Cooperator_Passenger ] = Cooperator_Angle( PC_Start , Set_Angle )
    %Cooperator_Angle寻找合乘人员
    %     PC_Start：可能的合乘人员
    %     Set_Angle：设置角度
    %     Cooperator_Passenger：合乘或单乘
    Size=size(PC_Start);
    Judge_L=Size(1);
    if Judge_L==1
        Cooperator_Passenger=PC_Start(Judge_L,:);
    else
        vector1=[PC_Start(:,3)-PC_Start(:,1) PC_Start(:,4)-PC_Start(:,2)];
        vector2=[PC_Start(Judge_L,3)-PC_Start(Judge_L,1)
    PC_Start(Judge_L,4)-PC_Start(Judge_L,2)];

    Calculate_Angle=acos((vector1*vector2')./(sqrt((vector1(:,1)).^2+(vector1(:,2)).^2)*n
    orm(vector2,2)));
        Calculate_Angle=Calculate_Angle(1:Judge_L-1,:);
        [~ , Sort_Index ]=sort(Calculate_Angle);
        [~,amount]=find(Calculate_Angle<=Set_Angle);
        if sum(amount)==0
            Cooperator_Passenger=PC_Start(Judge_L,:);
        elseif sum(amount)==1
            Cooperator_Passenger(1,:)=PC_Start(Judge_L,:);
            Cooperator_Passenger(2,:)=PC_Start(Sort_Index(1),:);
        else
            Cooperator_Passenger(1,:)=PC_Start(Judge_L,:);
            Cooperator_Passenger(2,:)=PC_Start(Sort_Index(1),:);
```

```
            Cooperator_Passenger(3,:)=PC_Start(Sort_Index(2),:);
        end
end
end
```

```
function [ CP1 , CP2 , CP3 ] = Statistics_Group( Cooperator_Passenger , Cp1 , Cp2 ,
Cp3 )
%Statistics_Group统计合乘分组
%     Cooperator_Passenger：合乘人员
%     Cp1：原一人组
%     Cp2：原两人组
%     Cp3：原三人组
%     CP1：新加一人组
%     CP2：新加两人组
%     CP3：新加三人组
Size1=size(Cp1);
Size2=size(Cp2);
Size3=size(Cp3);
CP1=Cp1;
CP2=Cp2;
CP3=Cp3;
Size=size(Cooperator_Passenger);
if Size(1)==1
    CP1(Size1(1)+1,:)=Cooperator_Passenger;
elseif Size(1)==2
    CP2(Size2(1)+1:Size2(1)+2,:)=Cooperator_Passenger;
else
    CP3(Size3(1)+1:Size3(1)+3,:)=Cooperator_Passenger;
end
end
```

```
function [ Requests , symble ] = Delect_Passenger( requests , Cooperator_Passenger )
%UNTITLED  此处显示有关此函数的摘要
```

```
%     requests：原剩余乘车人员
%     Cooperator_Passenger：合乘人员
%     Requests：未乘车人员
%     symble：分配完毕标志
Index=size(Cooperator_Passenger);
for i=1:Index(1)
      requests(Cooperator_Passenger(i,6),:)=-1000*ones(1,6);
end
Size_requests=size(requests);
Requests=zeros(Size_requests);
k=0;
for j=1:Size_requests(1)
      if requests(j,1)~=-1000
            k=k+1;
            Requests(k,:)=requests(j,:);
      end
end
if k==0;
      Requests=[];
      symble=0;
else
      Requests=Requests(1:k,:);
      symble=1;
end
end


function [ Requests ]=Delect_Number_Requests( requests )
%Delect_Number_Requests  去编号
%     requests：带编号数据
%     Requests：不编号数据
Size=size(requests);
if Size(1)~=0
Requests=requests(1:Size(1),1:5);
else
```

```
    Requests=[];
end
end


function [    Result , Lefted_Taxi ]=Assignment_CP3( Taxi , CP3 )
%Assignment_CP3  给三人派车
%     Taxi：当前可用车租车
%     CP3：三人组合
%     Result：分派结果
%     Lefted_Taxi：剩余出租车

Size=size(Taxi);
Lefted_Taxi=[Taxi (1:Size(1))'];%给车编号
Size_CP3=size(CP3);
Result=zeros(Size_CP3(1)/3,9);
m=0;
for i=1:3:Size_CP3(1)
    Passanger=CP3(i:i+2,:);
    [ Last_Choose ]=Search_Taxi( Lefted_Taxi , Passanger );
    m=m+1;Result(m,:)=Last_Choose;
    Lefted_Taxi_Num=Lefted_Taxi(:,3);
    [Number,~]=find(Lefted_Taxi_Num==Last_Choose(1));
    Lefted_Taxi(Number,:)=[];
end
end

function [ temporary ]=Search_Taxi( Lefted_taxi , Passanger )
Szie_LT=size(Lefted_taxi);
Ps=Passanger(:,1:2);
Pe=Passanger(:,3:4);
P_Num=Passanger(:,5);
Min_Sum_Distance=100000;
for j=1:Szie_LT(1)
    T=Lefted_taxi(j,:);
```

```
        Sum_Distance(1)=S_Distance( T , Ps , 1 , 2 , 3 );
        Sum_Distance(2)=S_Distance( T , Ps , 1 , 3 , 2 );
        Sum_Distance(3)=S_Distance( T , Ps , 2 , 1 , 3 );
        Sum_Distance(4)=S_Distance( T , Ps , 3 , 1 , 2 );
        Sum_Distance(5)=S_Distance( T , Ps , 2 , 3 , 1 );
        Sum_Distance(6)=S_Distance( T , Ps , 3 , 2 , 1 );
        [Min_UD,Num]=min(Sum_Distance);
        if Min_UD<Min_Sum_Distance
            Min_Sum_Distance=Min_UD;
            Up_Sort=Search_Sort( Num );%设计上车秩序
            Sum_Distance(1)=S_Distance( Ps(Up_Sort(3),:) , Pe , 1 , 2 , 3 );
            Sum_Distance(2)=S_Distance( Ps(Up_Sort(3),:) , Pe , 1 , 3 , 2 );
            Sum_Distance(3)=S_Distance( Ps(Up_Sort(3),:) , Pe , 2 , 1 , 3 );
            Sum_Distance(4)=S_Distance( Ps(Up_Sort(3),:) , Pe , 3 , 1 , 2 );
            Sum_Distance(5)=S_Distance( Ps(Up_Sort(3),:) , Pe , 2 , 3 , 1 );
            Sum_Distance(6)=S_Distance( Ps(Up_Sort(3),:) , Pe , 3 , 2 , 1 );
            [Min_DD,Num]=min(Sum_Distance);
            Down_Sort=Search_Sort( Num );%设计下车秩序
            Up_Sort_Num=[P_Num(Up_Sort(1)) P_Num(Up_Sort(2))
P_Num(Up_Sort(3))];
            Down_Sort_Num=[P_Num(Down_Sort(1)) P_Num(Down_Sort(2))
P_Num(Down_Sort(3))];
            temporary=[Lefted_taxi(j,3) Up_Sort_Num Down_Sort_Num Min_UD
Min_UD+Min_DD];
        end
end
end

function [ Sum_Dis ]=S_Distance( T , Ps , i , j , k )
Sum_Dis=abs(T(1)-Ps(i,1))+abs(T(2)-Ps(i,2))+...
            abs(Ps(i,1)-Ps(j,1))+abs(Ps(i,2)-Ps(j,2))+...
            abs(Ps(j,1)-Ps(k,1))+abs(Ps(j,2)-Ps(k,2));
end

function Chose_Sort=Search_Sort( Num )
```

```
if Num==1
    Chose_Sort=[1 2 3];
elseif Num==2
    Chose_Sort=[1 3 2];
elseif Num==3
    Chose_Sort=[2 1 3];
elseif Num==4
    Chose_Sort=[3 1 2];
elseif Num==5
    Chose_Sort=[2 3 1];
else
    Chose_Sort=[3 2 1];
end
end


function [    Result , Lefted_Taxi ]=Assignment_CP2( Taxi , CP2 )
%Assignment_CP2 给两人派车
%     Taxi：当前可用车租车
%     CP2：两人组合
%     Result：分派结果
%     Lefted_Taxi：剩余出租车

Lefted_Taxi=Taxi;
Size_CP2=size(CP2);
Result=zeros(Size_CP2(1)/2,7);
m=0;
for i=1:2:Size_CP2(1)
    Passanger=CP2(i:i+1,:);
    [ Last_Choose ]=Search_Taxi( Lefted_Taxi , Passanger );
    m=m+1;Result(m,:)=Last_Choose;
    Lefted_Taxi_Num=Lefted_Taxi(:,3);
    [Number,~]=find(Lefted_Taxi_Num==Last_Choose(1));
    Lefted_Taxi(Number,:)=[];
end
```

```
end

function [ temporary ]=Search_Taxi( Lefted_taxi , Passanger )
Szie_LT=size(Lefted_taxi);
Ps=Passanger(:,1:2);
Pe=Passanger(:,3:4);
P_Num=Passanger(:,5);
Min_Sum_Distance=100000;
for j=1:Szie_LT(1)
    T=Lefted_taxi(j,:);
    Sum_Distance(1)=S_Distance( T , Ps , 1 , 2 );
    Sum_Distance(2)=S_Distance( T , Ps , 2 , 1 );
    [Min_UD,Num]=min(Sum_Distance);
    if Min_UD<Min_Sum_Distance
        Min_Sum_Distance=Min_UD;
        Up_Sort=Search_Sort( Num );%设计上车秩序
        Sum_Distance(1)=S_Distance( Ps(Up_Sort(2),:) , Pe , 1 , 2 );
        Sum_Distance(2)=S_Distance( Ps(Up_Sort(2),:) , Pe , 2 , 1 );
        [Min_DD,Num]=min(Sum_Distance);
        Down_Sort=Search_Sort( Num );%设计下车秩序
        Up_Sort_Num=[P_Num(Up_Sort(1)) P_Num(Up_Sort(2))];
        Down_Sort_Num=[P_Num(Down_Sort(1)) P_Num(Down_Sort(2))];
        temporary=[Lefted_taxi(j,3) Up_Sort_Num Down_Sort_Num Min_UD
Min_UD+Min_DD];
    end
end
end

function [ Sum_Dis ]=S_Distance( T , Ps , i , j )
Sum_Dis=abs(T(1)-Ps(i,1))+abs(T(2)-Ps(i,2))+...
        abs(Ps(i,1)-Ps(j,1))+abs(Ps(i,2)-Ps(j,2));
end

function Chose_Sort=Search_Sort( Num )
if Num==1
```

```
        Chose_Sort=[1 2];
else
        Chose_Sort=[2 1];
end
end


function [    Result , Lefted_Taxi ]=Assignment_CP1( Taxi , CP1 )
%Assignment_CP1 给单人派车
%     Taxi：当前可用车租车
%     CP1：单人
%     Result：分派结果
%     Lefted_Taxi：剩余出租车

Lefted_Taxi=Taxi;
Size_CP1=size(CP1);
Result=zeros(Size_CP1(1),5);
m=0;
for i=1:Size_CP1(1)
        Passanger=CP1(i,:);
        [ Last_Choose ]=Search_Taxi( Lefted_Taxi , Passanger );
        m=m+1;Result(m,:)=Last_Choose;
        Lefted_Taxi_Num=Lefted_Taxi(:,3);
        [Number,~]=find(Lefted_Taxi_Num==Last_Choose(1));
        Lefted_Taxi(Number,:)=[];
end
end

function [ temporary ]=Search_Taxi( Lefted_taxi , Passanger )
Szie_LT=size(Lefted_taxi);
Ps=Passanger(:,1:2);
Pe=Passanger(:,3:4);
P_Num=Passanger(:,5);
Min_Sum_Distance=100000;
for j=1:Szie_LT(1)
```

```
        T=Lefted_taxi(j,:);
        Min_UD=S_Distance( T , Ps );
        if Min_UD<Min_Sum_Distance
            Min_Sum_Distance=Min_UD;
            Min_DD=S_Distance( Pe , Ps );
            temporary=[Lefted_taxi(j,3) P_Num P_Num Min_UD Min_UD+Min_DD];
        end
    end
end

function [ Sum_Dis ]=S_Distance( T , Ps )
Sum_Dis=abs(T(1)-Ps(1))+abs(T(2)-Ps(2));
end


function [ Proportion ]=Statistics_Analysis( Assignment_CP1 , Assignment_CP2 ,
Assignment_CP3 )
%Statistics_Analysis 进行数据统计分析
%     Assignment_CP1：一人乘车数据
%     Assignment_CP2：两人乘车数据
%     Assignment_CP3：三人乘车数据
%     Proportion：平均空载率

Size1=size(Assignment_CP1);
Size2=size(Assignment_CP2);
Size3=size(Assignment_CP3);
Proportion=(2/3*Size1(1)+1/3*Size2(1))/(Size1(1)+Size2(1)+Size3(1));
Wait_Distance_CP1=Assignment_CP1(:,4);%单人乘车的等待距离
Wait_Distance_CP2=Assignment_CP2(:,6);%单人乘车的等待距离
Wait_Distance_CP3=Assignment_CP3(:,8);%单人乘车的等待距离

Subsection_WP1=Subsection(Wait_Distance_CP1);
Subsection_WP2=Subsection(Wait_Distance_CP2);
Subsection_WP3=Subsection(Wait_Distance_CP3);
```

```
set(0,'defaultfigurecolor','w')
figure(1)
subplot(1,2,1)
stem(Wait_Distance_CP1,'fill','r-.');
title('单人乘车等待的车行距离')
xlabel('出租车数目')
ylabel('出发前等待距离（km）')
grid on
%画饼状图
subplot(1,2,2)
tt={'小于1.5km','1.5-3km','大于3km'};%输入标签
t=tt(1:length(Subsection_WP1));
explode=ones(1,length(Subsection_WP1));
pie(Subsection_WP1,explode);
colormap cool;
legend(t)
title('车行距离分段分布')

figure(2)
subplot(1,2,1)
stem(Wait_Distance_CP2,'fill','r-.');
title('两人合乘等待的车行距离')
xlabel('出租车数目')
ylabel('出发前等待距离（km）')
grid on
subplot(1,2,2)
%画饼状图
t=tt(1:length(Subsection_WP2));
explode=ones(1,length(Subsection_WP2));
pie(Subsection_WP2,explode);
colormap cool;
legend(t)
title('车行距离分段分布')

figure(3)
```

```
subplot(1,2,1)
stem(Wait_Distance_CP3,'fill','r-.');
title('三人合乘等待的车行距离')
xlabel('出租车数目')
ylabel('出发前等待距离（km）')
grid on
subplot(1,2,2)
%画饼状图
t=tt(1:length(Subsection_WP3));
explode=ones(1,length(Subsection_WP3));
pie(Subsection_WP3,explode);
colormap cool;
legend(t)
title('车行距离分段分布')
end

function [ Subsection_P ]=Subsection( Wait_Distance_CP )
Subsection_P=zeros(1,3);
for i=1:length(Wait_Distance_CP)
    if Wait_Distance_CP(i)<=1.5
        Subsection_P(1)=Subsection_P(1)+1;
    elseif Wait_Distance_CP(i)<=3
        Subsection_P(2)=Subsection_P(2)+1;
    else
        Subsection_P(3)=Subsection_P(3)+1;
    end
end
Subsection_P=Subsection_P/sum(Subsection_P);
feiling=zeros(1,length(Subsection_P));k=0;
for i=1:length(Subsection_P);
    if Subsection_P(i)~=0
        k=k+1;
        feiling(k)=Subsection_P(i);
    end
end
```

```
Subsection_P=feiling(1:k);
end


function Output_to_Txt( Assignment_CP1 , Assignment_CP2 , Assignment_CP3 )
%Output_to_Txt 输出乘车方案到文本文件
%     此处显示详细说明
Size1=size(Assignment_CP1);
Size2=size(Assignment_CP2);
Size3=size(Assignment_CP3);
Total_Taxi=Size1(1)+Size2(1)+Size3(1);
[ Assign_CP1 ]=change_to_char( Assignment_CP1(:,1:3) );
[ Assign_CP2 ]=change_to_char( Assignment_CP2(:,1:5) );
[ Assign_CP3 ]=change_to_char( Assignment_CP3(:,1:7) );
fid = fopen('201718001003-plan.txt','wt');
fprintf(fid,'%s\n', num2str(Total_Taxi));
for j = 1:Size1(1)
      fprintf(fid,'%s\n',Assign_CP1(j,:));
end
for j = 1:Size2(1)
      fprintf(fid,'%s\n',Assign_CP2(j,:));
end
for j = 1:Size3(1)
      fprintf(fid,'%s\n',Assign_CP3(j,:));
end
fclose(fid);
end

function [ Assign_CP ]=change_to_char( Assign_CPP )
Size=size(Assign_CPP);
Assign_CP=[];
if Size(2)==3
      for i=1:Size(1)
            taxi=num2str(Assign_CPP(i,1),'%04d');
            taxi=strcat('t',taxi);taxi=strcat(taxi,', p');
```

```
            Up=num2str(Assign_CPP(i,2),'%04d');Up=strcat(Up,'u, p');
            Down=num2str(Assign_CPP(i,3),'%04d');Down=strcat(Down,'d');
            Assign_CP=[Assign_CP;taxi Up Down];
        end
    elseif Size(2)==5
        for i=1:Size(1)
            taxi=num2str(Assign_CPP(i,1),'%04d');
            taxi=strcat('t',taxi);taxi=strcat(taxi,', p');
            Up1=num2str(Assign_CPP(i,2),'%04d');Up1=strcat(Up1,'u, p');
            Up2=num2str(Assign_CPP(i,3),'%04d');Up2=strcat(Up2,'u, p');
            Down1=num2str(Assign_CPP(i,4),'%04d');Down1=strcat(Down1,'d, p');
            Down2=num2str(Assign_CPP(i,5),'%04d');Down2=strcat(Down2,'d');
            Assign_CP=[Assign_CP;taxi Up1 Up2 Down1 Down2];
        end
    else
        for i=1:Size(1)
            taxi=num2str(Assign_CPP(i,1),'%04d');
            taxi=strcat('t',taxi);taxi=strcat(taxi,', p');
            Up1=num2str(Assign_CPP(i,2),'%04d');Up1=strcat(Up1,'u, p');
            Up2=num2str(Assign_CPP(i,3),'%04d');Up2=strcat(Up2,'u, p');
            Up3=num2str(Assign_CPP(i,4),'%04d');Up3=strcat(Up3,'u, p');
            Down1=num2str(Assign_CPP(i,5),'%04d');Down1=strcat(Down1,'d, p');
            Down2=num2str(Assign_CPP(i,6),'%04d');Down2=strcat(Down2,'d, p');
            Down3=num2str(Assign_CPP(i,7),'%04d');Down3=strcat(Down3,'d');
            Assign_CP=[Assign_CP;taxi Up1 Up2 Up3 Down1 Down2 Down3];
        end
    end
end
```