公共自行车服务系统规划建设与运行调度方案

摘要

本文主要研究西安市经开区公共自行车的分配和调度优化问题，讨论了租赁点的设计选择、自行车的分配和调度方案。建立了基于公平分配原则的自行车分配调度模型、0-1 规划模型，借助 K-means 聚类和模拟退火等算法，给出了合理可行的公共自行车系统规划建设与运行调度方案。

针对问题一，为了确定系统自行车分配数目和运行时的调度方案，我们将其分为两个小问题进行讨论：（1）对自行车分配问题，首先根据租赁点需求量等约束条件，综合运用公平分配原则和 Lingo 编程初步生成公共自行车的分配方案，然后根据经纬度求出各租赁点之间的距离，进而计算其余两个时间段的自行车数量变化量，从而修正最初的分配方案；（2）对调度方案问题，首先引入 0-1 变量表示各租赁点之间是否调度，以租赁点之间的往来时间与装卸自行车的时间为权重，然后采用 K-means 聚类算法和模拟退火算法(Simulated Annealing)，通过 Matlab 编程确定调度车的近似最优调度方案，最终求得整个服务系统完成一次调度花费的近似平均时间为 106.08 分钟。

针对问题二，为了确定第三期建设站点个数和选址，我们建立了站点数目预测和选址模型，并借助问题一的自行车配给模型，给出了三期建成后系统自行车分配方案。首先，采用先估算后修正的方法，对第三期新增站点数目进行合理预测；其次，在 70 个预选站点中，通过考虑自行车平均使用率和满足的总需求数这两个因素，借助 0-1 规划模型，最终花费 200 万元，购置 750 辆自行车，建设了包括确定了 50、56、60、81…共 25 个最佳站点选址；最后，通过问题一建立的自行车配给模型，确定了 55 个站点每处自行车配给数量。

针对问题三，为了确定三期完工后调度车辆能否按时完成调度工作，我们首先结合问题一和问题二的结果将原有的站点与新增站点混合，重新根据前两问的相应方法和分析建立数学模型；然后对调度点进行了区域划分并确定了最佳路径，求得单车调度时间大于 150mins，由此判断在第三期建设后需要重新购置调度车辆。接着通过建立的数学模型与 Matlab 编程 确定需要 3 辆调度车才能满足服务需求，并且求出 3 辆调度车的最佳调度路径。

关键词：公共自行车服务系统　K-means 聚类算法　0-1 规划　模拟退火算法

一、 问题重述与分析

1、问题一要求针对已有的30个租赁点设计最优车辆分配方案、调度方案，即调度平均耗时尽量少。这个问题是在已知租赁点具体的位置的条件下求解两个小问题：租赁点的自行车分配问题和调度问题。首先对于第一个小问题，需要找到一个初始需求量，建立目标模型，求出初始分配量，再不断经过负反馈修正初始分配量；然后根据调度平均耗时尽量少的原则，给出30个租赁点的调度方案，运用0-1变量表示各租赁点是否需要调度，接着建立规划模型进行各个时间段的调度。

2、问题二给出了三期预选站点的基本信息和发展状况，需要确定新增站点数目，位置和规划每个站点自行车的配置。可以用0-1规划的角度予以解决：需要我们选定合适的约束条件和确定合适的评价指标以构成目标函数，在选定站点后，还需要我们结合实际情况，对算法初步筛选出的站点做进一步的筛选，以确保选址的科学性和合理性。

3、 问题三要求系统在150min内完成调度，从而确定是否需要增加调度车辆。在第一、二问的基础上，租赁点的具体位置、需求量和分配方案都已知，即需要确定在给定时间内能否完成调度；如能则直接给出调度方案，如调度车辆不够，则给出增加的车辆数目和调度方案。

二、 模型假设

1、为简化模型，不考虑调度车启动和停止的时间，假设调度车运输过程匀速行驶，不会受到交通事故、红绿灯等外界因素的影响。

2、假设各个租赁点每天的自行车需求量不变，且仅考虑高峰时段自行车数辆的变化。

3、假设居民的骑行距离不超过2公里，在某个租赁点还车的概率与租车点和还车点的距离成反比，骑行距离超过2公里的情况一定不会发生。

4、求解过程中，自行车数量出现小数时采用四舍五入取整代替。

三、 符号说明

表 1 符号说明及含义

符号	含义
G	租赁点的集合
T	调度总时间（不包括完成调度后调度车返回原点的时间）
d_{ij}	租赁点间的最短距离
M_{ij}	从租赁点i 还到了租赁点 j 的概率
c_i	概率系数
K_{ij}	从租赁点i 还到了租赁点 j 的约化概率
u_i^t	t 时间段内租赁点 i 的需求量
u_i	租赁点 i 的需求量
r_{ij}	从 i 点出发到达 j 租赁点的自行车数量
x_i	i点的单车数量
p_{ij}	从 i 点出发到达j租赁点 j 的调度量
J_{ij}	从租赁点i 到租赁点 j 是否进行了调度
$x_{i'}$	骑处和归还已经结束、调度之前i点的单车数量
a	建立一个租赁点耗资
b	维护每辆自行车耗资
w	每一个租赁点的耗资数

注：未列出及重复的符号以出现处为准

四、 模型建立与求解

4.1 问题一的模型建立与求解

为得出现有站点最佳自行车分配方案和自行车系统运行期间的调度方案，本文建立了基于公平分配原则的自行车分配模型，同时引入聚类算法为两辆调配车分配目标站点，并通过模拟退火算法搜索每辆配送车的最佳运行路线。

首先，根据车辆需求数据，综合运用公平分配原则[1]生成初步车辆分配方案；然后，通过计算一天中其余两个时间段的自行车数量变化量，对现有租赁点自行车的原始车辆进行再调整，得出优化后的车辆分配方案；最后，结合西安市经开区具体行驶数据，采用 K-means 聚类算法和模拟退火算法确定自行车的调度方

案。

4.1.1 自行车分配模型建立

依据西安市 30 个租赁点的车辆需求数据，并且基于公平分配原则对自行车进行分配。首先引入分配的不平衡度，进而构造出不平衡度的单值最小目标函数；然后综合分析各种约束构造非线性规划模型，使用 Lingo 对非线性模型求解，得到最初的分配方案；最后根据其他站点归还的自行车数量确定租赁点车辆数的变化量，进而对分配方案进行调整，得到最终的分配方案。

1） 公平分配原则[2]

针对附件二所给出的不同站点的需求量，引入变量 r 衡量自行车分配的不平衡程度

$$r(x_i, q_i) = \sqrt{\frac{x_i^2 - q_i^2}{x_i}} \tag{1}$$

其中，q_i为第 i 个站点的需求量，r越大，表示分配越不平衡。

2） 整数规划模型确定初始分配方案

由附件二知，系统一天要经历早、中、晚三个用车高峰。不妨先从早高峰出发（7:00—8:30），以此作为确定初始分配量的高峰时段，选取当天 7:00 之前各站点分配的自行车数量 x_i 为决策变量，可以建立下列关系：

（1）目标函数：根据公平分配原则确定最小分配的不平衡函数：

$$f = min \sum r(x_i, q_i) \tag{2}$$

（2）约束条件：

$$s.t. \begin{cases} (1 + 10\%) \sum q_i \le N \\ \sum x_i = N \\ 0 \le x_i (i = 1,2,3,\cdots,30) \le n_{max} \end{cases} \tag{3}$$

其中，N 为目前车辆总数，q_i为第 i 个站点的需求量，x_i为第 i 个站点分配的车辆数，n_{max}为单个站点所能容纳的最大车辆数。

3） 初始分配方案的优化

（1）各站点最短路程求解

在实际运行中，调度车在两个站点间的形式距离由两站点的经纬度和站点间的具体道路分布决定。但由于具体路线长度求解较为繁琐，而直接用经纬度求两战点的直线距离误差又较大，站点间实际最短路程的求解存在很大的困难。通过观察道路的特点，本文探索了一种时间复杂度较低且误差较小的简化处理方法。

以附件一出口加工厂站 i 与管委会 j 之间的距离 d_{ij} 的求解为例。通过观察附件一所给地图，我们发现西安经开区道路几乎全部为南北走向和东西走向，因此，我们可以通过分别求两站经纬度差换算出的路程，得到两站点东西方向上的距离和南北方向上的距离，结合西安市经开区道路的走向特点，将两个距离加和就是两站点间的最短路程。

图 1 由经纬度计算路程示意图

a. 路线的横向距离：

$$d_{BC} = R_1 \frac{J_i - J_j}{57.3} \tag{4}$$

b. 路线的纵向距离：

$$d_{AB} = R_2 \frac{W_i - W_j}{57.3} \tag{5}$$

c. 自行车租赁点 i 到 租赁点 j 之间的距离：

$$d_{ij} = R_1 \frac{J_i - J_j}{57.3} + R_2 \frac{W_i - W_j}{57.3} (i = 1,2,\dots,30; j = 1,2,\dots,30) \tag{6}$$

（2）归还概率

已知从站点 i 出发的自行车，在站点 j 的归还的概率与两站点的距离成反比，且已假设自行车骑行距离不超过 2 km。因此，可以出站点 i 的自行车，在

站点 j 的归还的概率为

$$M_{ij}=\begin{cases}0\ ,\ d_{ij}>2\\ \frac{1}{d_{ij}},0\le d_{ij}\le 2\end{cases} \tag{7}$$

由于站点的位置有限，所以需要对 M_{ij} 的分布做离散化处理。又因为站点 i 的自行车在附近所有站点归还的概率之和必须等于 1，所以我们要对 M_{ij} 的分布做归一化处理，得出概率系数 c_{ij} 进而约化归还概率

$$K_{ij}=c_{ij}M_{ij}=\frac{c_{ij}}{d_{ij}} \tag{8}$$

（3）其他站点归还的自行车数量以及站点车辆数目的变化量

其他 29 个租赁点（2 km 以内）归还到站点 i 的自行车数量以及变化量

$$\begin{cases}r_i=\sum u_jK_{ij}\\ \epsilon(i,t)=r_{i,t}-u_{i,t}\end{cases} \tag{9}$$

其中， u_j表示每个站点的需求量，M_{ij}表示从站点 j 出发的自行车归还到站点 i 的概率，t 用于区别一天之中的三次高峰时段 $(t=1,2,3)$。

4.1.2 自行车分配模型的求解

1）分配方案初步求解

选取一天中的早高峰作为初步求解的数据来源，将附件二中编号为 1~30 的站点在 7:00-8:30 时段的需求量带入公式(1)~(3)中，使用 Lingo 软件求解非线性整数规划问题，得到初步分配方案。求解结果数据量过大，不便于在正文中展示，见附录一。

2）优化方案所用数据的准备

将附件三各站点的经纬度数据代入公式(6)，即可得出各站点之间的最短路程。求解结果见附件。

将上述求得的各站点间的最短路程代入公式(8)，可以得到从站点 i 出发的自行车，在站点 j 的归还的约化概率。求解结果见附件。

将求得的各站点的约化概率矩阵 Cij 代入公式(9)，可以计算出 30 个自行车租赁点在各时间段的变化量，求解结果见附录二。

3）对分配方案的进一步优化

根据变化量的多少，我们可以得出每个时间段都发生了自行车数量的变化，

有些租赁点变化较大。根据变化量的变动大小，我们根据以下原则将 30 个租赁点自行车的原始分配方案进行调整：

a.每个租赁点的分配量满足时间段 7:00-8:30 的自行车需求数；

b.如果接下去两个时间段变化量均正值，且能满足下两个时间段的需求量，则以原来需求量当做分配量；

c.如果接下去两个时间段变化量均负值，则用变化量之和与需求量的变化量进行比较，使得能够同时尽量使分配量与下两个时段的分配量的差值尽量小；

d.如果接下去两个时间段变化量为一负一正，比较正负变化量的大小，并以是否满足下两个时段为目标，以满足当前时间段为条件进行增加一定数量自行车数；

e.考虑特殊的两个点：出口加工区广场和鼎新花园分别为整个区域图的左上和右下，所以尽量给这两个点最大的需求量，以减少调度的车辆数和车辆时间；

按照以上原则，对自行车初始分配方案进行调整，得到最终分配方案，见附录三。

4.1.3 调度车的调度方案设计与求解

本次调运系统有 2 辆调运车，每辆调运车拥有负荷数为 $q=50$，当有租赁点达到上下限时（小于 20%或大于 90%），调运车从最近的停车站点出发，负责对各租赁点进行自行车的需求调度服务，完成调度服务后就近回到停车站点。所以首先要确定一个租赁点的自行车是否需要调度，这取决于各租赁点的需求量u_i、从其他租赁点起来的自行车r_i、以及租赁点自行车数量的变化量；然后基于 k-Means 算法对调度点进行区域划分。

1） 调配时间最短的模型建立

设拥有最大负荷为 Q 的调度车从指定的节点出发，对集合为 G 的节点进行 调度。完成任务后返回原点。 调度需求量 u_i 和租赁点间的距离 d_{ij} 已经求得。建立单目标规划模型：

（1）目标函数

一辆调度车一次调度所需要的时间函数：

$$minT=\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\frac{d_{ij}}{V}+|p_j|)J_{ij} \tag{10}$$

（2）决策变量

n 个租赁点的初始投放量：

$$x_1, x_2, x_3, \cdots, x_n \quad (n = 1,2, \ldots\ldots, n)$$

（3）约束条件

保证调度车调度时调度车上的数量能够满足任意节点的调度需求量，同时装载量不能超过调度车的最大载重；

给出总需求量、分配量、调度量从该节点出发的自行车数目和到达该节点的自行车的数量之间的关系；

每个租赁点后仅有一个租赁点与其相接；

每个租赁点前也仅有一个租赁点与其相接；以上两式确定了调度车对特定节点只能调度 1 次，不能重复经过一个 1 节点；

规定了每次服务都能完成并且不超过车辆最大载车数 Q

$$s.t.\begin{cases} 0 \le p_j + r_{ij} \\ p_j + r_{ij} \le Q \\ \sum_{i=1}^{n} x_{ij} = 1(j = 1,2,\cdots,30) \\ \sum_{j=1}^{n} x_{ij} = 1(i = 1,2,\cdots,30) \\ p_i(i,t) = x_i - u_i - \sum_{i=1}^{n} q_i\, k_{ij} \\ x_{i,t} \ge (1 + 10\%)u_{i,t}\ (i = 1,2,\cdots,30) \\ x_{i,t} = x_i + \epsilon^0(i,t)(i = 1,2,\cdots,30) \\ \sum_{i=1}^{j} \epsilon^0\,(i,t) \le 50(i = 1,2,\cdots,30) \\ \sum_{i=1}^{30} x_{i,t} = N(i = 1,2,\cdots,30) \\ 0 \le -u_i + r_{ji} \le Q_j \\ J_{ij} = \begin{cases} 0, & p_{ij} = 0 \\ 1, & p_{ij}\text{不等于 } 0 \end{cases} \end{cases} \tag{11}$$

2） 对调度点进行区域划分

首先通过统计各个站点的运营状况和与其他站点的距离选择K个初始聚类中心点，其次利用优化的 K-means 算法[3][4]，得到聚类结果，然后引入各站点的调度需求量，修正聚类结果中的边缘点，得到最终聚类结果。

分析初次使用 K-means 算法后数据集的分类情况与修正 K-means 算法后数据集的分类情况，对两种算法各区域的调度需求量进行对比，可以知道：由于事先对初始聚类中心点做了改进，该算法得到的聚类中心要比传统 Kmeans 算法更接近全局最优，同时调度车的数量也有所减少。

因此该算法能够在保证聚类精度的同时，通过降低聚类数据量可以有效提高聚类效率。

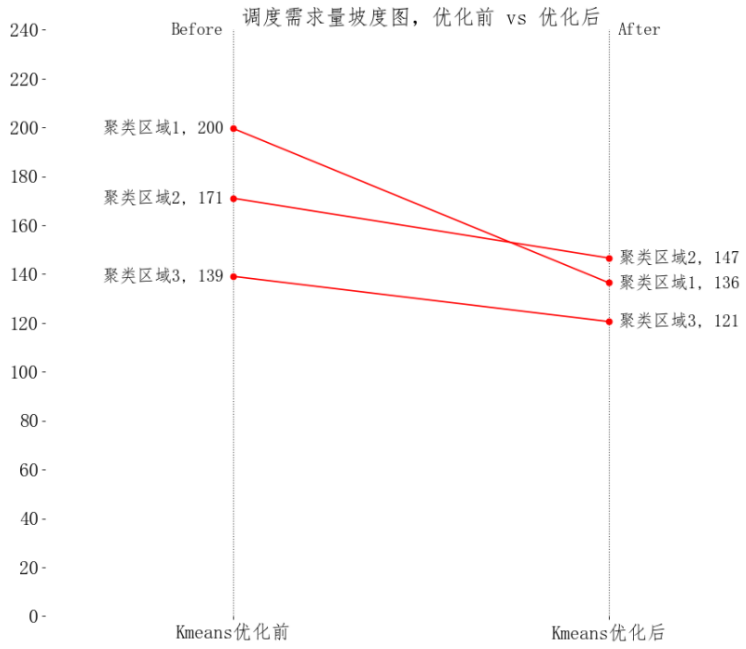

图 2 用两种算法各区域的调度需求量对比

为了验证上述算法的有效性，本文选择了公共自行车的调度实例进行分析。数据包括该区域站点总数为 55，各站点的经纬度及调度需求量已知。如果用于配送的调度车辆为 3，利用改进的 K-means 聚类算法 将 55 个站点分为 3 个类，从上图 可以看出改进的 K-means 算法应用到公共自行车 系统后能有效地解决调度 时间耗损大的问题，节约了调度成本。

3） 划分区域后的最短时间的最优解

在经典 TSP 问题中，经常使用模拟退火算法来求解最短路程及其路径。在本文中求解调配的最短消耗时间时，模拟退火算法同样能够能通过概率突跳方式避免陷入局部最小并最终趋于全局最优[5]。首先从有盈余车的租赁点中随机确定一个调度出发点x_i，求出从这个租赁点 x_i 出发的最佳路径和最短时间；然后根据更新函数产生一个随机扰动，如果找到的最短时间比上一次的最短时间还小，即比上一个解更接近最优解，则需要判断是接受这一个解，还是以人为设定的概率 Φ 接受一个比上一次时间更长的最短路径的解。接着不断重复上述步骤，直到找到最短调度时间，或者达到指定的寻找次数，结束循环。

（1）随机确定调度出发点 x_i

通过类比物理退火过程，调度出发点 x_i 的选择应该尽可能大以保证几乎所有的候选解都能被接受。实验表明，初始值 x_i 越大，获得全局最优解的几率就越大[6]，但所需要的计算时间也越长。在本问题中，假设不考虑计算时间仅考虑最佳路径是否对应最短调配时间，所以选取一个初始调度时间 T 足够大的调度出发点 x_i 作为初始值。

（2）评价是否接受随机扰动产生的新的最短调度时间

在预设的随机波动下，每一次循环将会解出一个新的最短调度时间 T'；首先确定温度更新函数，即最短调配时间的下降函数，查阅资料可得，较为常用的下降函数为指数函数 ，$T_{K+1} = \alpha T_k = \alpha T + 1$ 其中 $0 < \alpha < 1$，且其大小可以不断变化[7];然后计算增量 $\Delta f = f(T') - f(T)$，其中$f(x)$ 为评价函数；若$\Delta f = f(x') - f(x) < 0$，则接受 T' 作为当前最短调度时间，否则以概率 $e^{-\frac{\Delta f}{T}}$ 接受 T' 作为当前最短调度时间；

（3）在随机扰动下循环得到最短调配时间 T 与最佳调配方案

要得到最短调配时间，首先需要确定循环终止准则：本循环中通过设置终止温度的阈值与外循环迭代次数来判断是否终止循环，得出最优解；然后不断对不同的调度出发点 x_i（其中$i = 1,2,\cdots,n$ ）进行上述循环计算，若满足终止条件，则得到最优解，终止循环；否则，执行更新函数 $tmp = tmp \times \alpha$，继续步骤（2）。

相应的最佳路线及最短时间如下表：

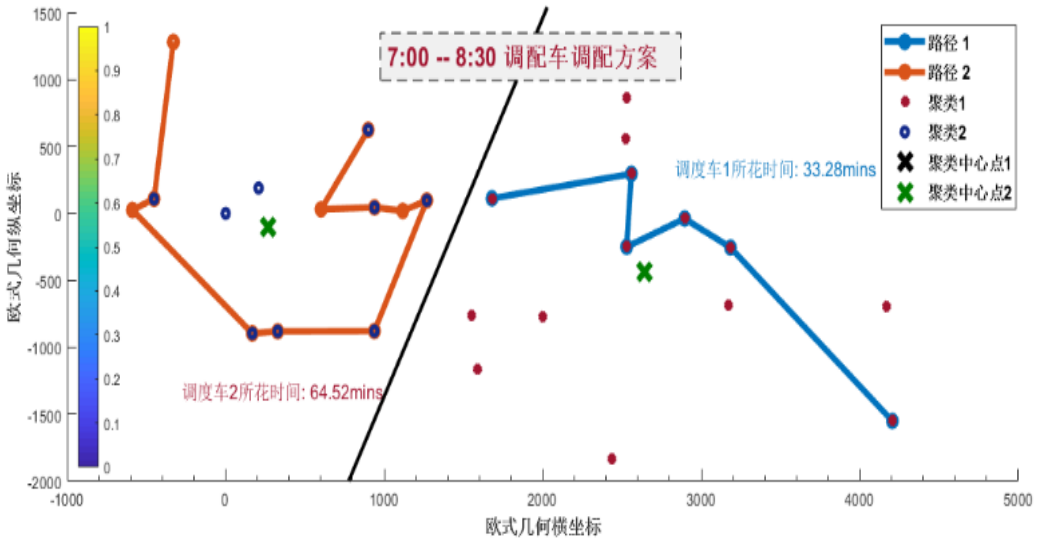

图 3 7:00-8:30 调配车的最佳调配方案

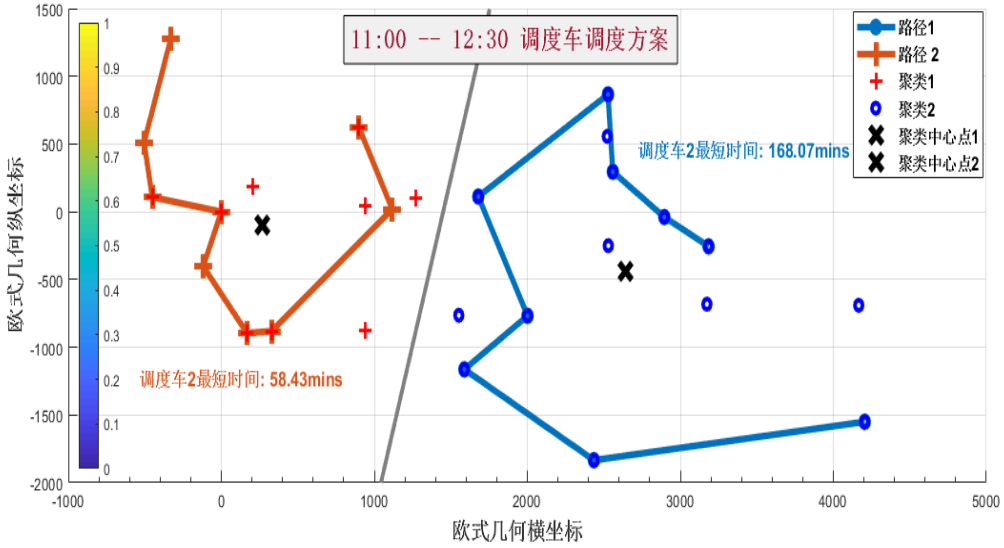

图 4 11:00-12:30 调配车的最佳调配方案

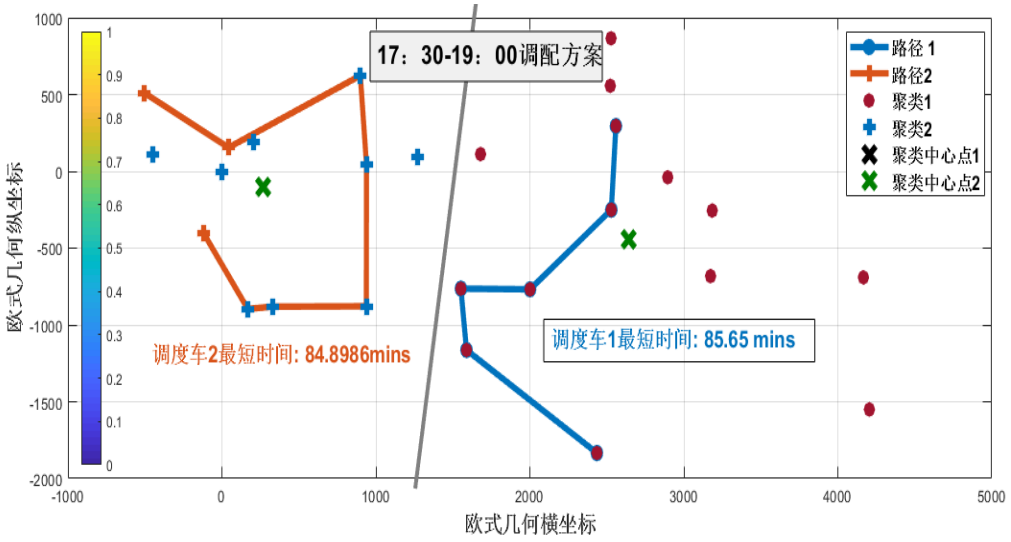

图 5 17:30-19:00 调配车的最佳调配方案

表 2　最佳路线及最短调度时间表

调度时间段	最短时间/mins	调配返回方案
07:00-08:30	路线一：33.28(右)	21-27-8-7-30-10(右)
	路线二：64.52(左)	11-13-14-3-17-4-20-5-19-29-18(左)
11:00-12:30	路线一：168.07(右)	30-7-27-26-25-21-23-22-24-10(右)
	路线二：58.43(左)	11-12-13-1-2-3-17-5-18(左)
17:30-19:00	路线一：85.65(右)	27-8-23-6-22-24(右)
	路线二：84.90(左)	2-3-17-4-5-18-15-12(左)

4.2 问题二：站点选址模型的建立

我们建立了站点数目预测和选址模型，并借助问题一已建立的自行车配给模型，确定第三期工程的具体实施方案。首先，我们采用先估算后修正的方法，对第三期新增站点数目进行合理预测；其次，在 70 个预选站点中，通过考虑自行车平均使用率和满足的总需求数这两个因素，借助优化模型，确定最合适的站点选址；最后，通过问题一建立的自行车配给模型，确定每个站点的自行车配给数量。

4.2.1 新增站点数目预测

为了降低算法求解的时间复杂度，将模型转化为易于求解的形式，我们将站点数量和站点选址分开考虑。首先根据一二期的建设预测第三期单个站点的建设支出，然后再由总预算估算出第三期新建站点的数目。

1）预测第三期单个站点的建设支出

设前两期建设车辆总数为 X，总站点数为 total，则平均每个站点配备车辆数为：

$$\overline{X} = \left[\frac{X}{total}\right] \tag{12}$$

假设第三期按照同样的配置进行采购，则单个站点的建设支出为：

$$\overline{Y} = Y_1 + \overline{X} \cdot Y_2 \tag{13}$$

其中，Y1 为修建单个站点的固定费用，Y2 为购进一辆自行车所需的费用。

2）估算第三期新建站点的数目

在已知单个站点的建设支出的情况下，通过总预算 S，可以估算出第三期新建站点的数目为：

$$N_3 = \left[\frac{S}{\overline{Y}}\right] \tag{14}$$

4.2.2 新增站点选址优化

在上文已估算出第三期新建站点数量的情况下，解决三期建设选址问题，我们可以借助 0-1 型正数规划优化模型[8]，通过设定合适的目标函数，找到最合适的站点选址。

1）决策变量

$$C_i = \begin{cases} 0, \text{第 i 个站点被选为三期站点} \\ 1, \text{第 i 个站点不作为三期站点} \end{cases} \tag{15}$$

2）约束条件

$$\sum_{31}^{100} Ci = \mathrm{N}_3 \tag{16}$$

3）目标函数

为了选取最合适的站点位置，我们通过问题一的求解和附件一材料分析，选择了三个在站点筛选时，最为重要的三个衡量指标：能够满足的需求总量，自行车平均使用频率，调配所需总时间。

（1）满足的总需求数

考虑到自行车租赁系统的目的是方便市民的出行，所以判断租赁站点选址是否合理的一个重要指标是：可以满足的需求数。

由附件二可知各预选站点每天自行车的需求量，加和可得每天的需求总量为：

$$Need = \left(\sum_{j=1}^{3} need_{ij}\right) \cdot {C_i}^T \tag{17}$$

（2）自行车平均使用频次[9]

合理的设置自行车租赁站点，可以提高自行车的利用率，加速自行车的流动循环，因此我们把自行车平均使用频次也作为衡量站点选择是否合理的判断依据，表达式为：

$$\varphi = \left(\frac{\sum_{j=1}^{3} need_{ij}}{\overline{\mathrm{X}} \cdot \mathrm{N}_3}\right) \cdot {C_i}^T \tag{18}$$

（3）目标函数的构建

为防止指标数据本身的大小对目标函数评价结果造成干扰，我们需要对以上三个指标进行一致化处理和归一化处理：

$$\begin{cases} Need_0 = \dfrac{Need - min(Need)}{max(Need) - min(Need)} \\ \varphi_0 = \dfrac{\varphi - min(\varphi)}{max(\varphi) - min(\varphi)} \end{cases} \tag{19}$$

假设需求满足量的权重为 γ，则自行车使用率的权重为(1- γ)，因此目标函数可以构建为：

$$max\, y_{aim} = \gamma \cdot Need_0 + (1-\gamma)\varphi_0 \tag{20}$$

4.2.2 站点选址模型的求解

1）初步计算

由附件一知，自行车租赁系统前两期建设站点总数为 30 个，车辆总数为 850 辆，带入公式(12)我们可以得到，每个站点平均配备车辆数取整后为 28 辆。

已知每个站点建设的固定费用为 5 万元，每辆自行车的采购维护费用为 1000 元，因此，由公式(13)，我们可以估算出第三期工程单个站点建设所需费用为 7.8 万元。将上述求得的单个站点开支带入公式(14)，我们可以初步得到第三期新建站点数目为 25 个，自行车共 700 辆，开销为 195 万元，余下 5 万元。

2）站点位置的初步选定

附件三包含了 100 个规划站点的位置信息，附件二包含了各个规划站点在三个高峰时段的自行车需求量。将附件三所给出的各站点经纬度信息导入地图软件中，我们可以得到一二期和第三期的位置分布情况：

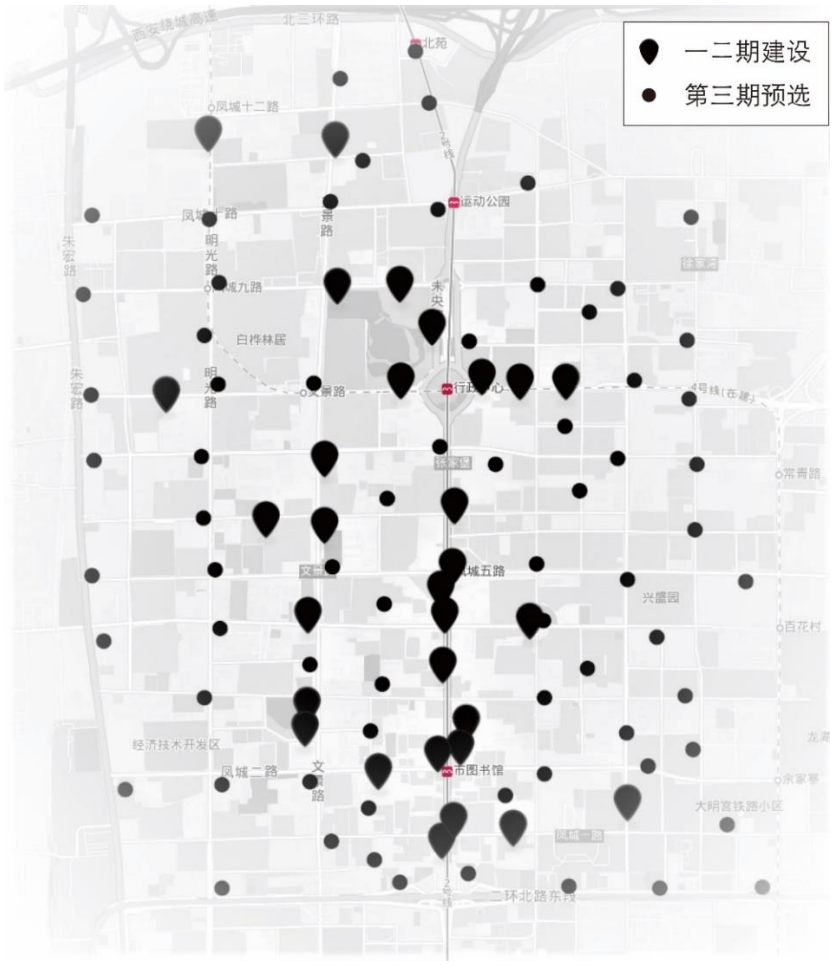

图 6 自行车站点分布图

我们将编号为 31~100 的各个站点的位置坐标和每日三个时段的需求量代入到模型中，通过 matlab 的 Optimization Tool 工具箱的 intlinprog“0-1 规划”命令，求解出最合适的 27 个站点决策。

表 3 地址优化模型运行结果

初步选取的站点序号													
31	33	34	36	45	46	47	49	50	56	59	60	62	63
69	72	73	76	77	80	81	84	87	88	90	91	81	

3）站点配置优化

为了确定出最终 25 个站点的位置，将上述求得的 27 个站点位置标在地图上，同时，通过热力图的形式，在同一张地图上画出各站点处的日平均高峰需求量。通过观察站点需求量与站点位置的匹配关系，以及站点重合程度，对所选取的 27 个站点进行筛选。

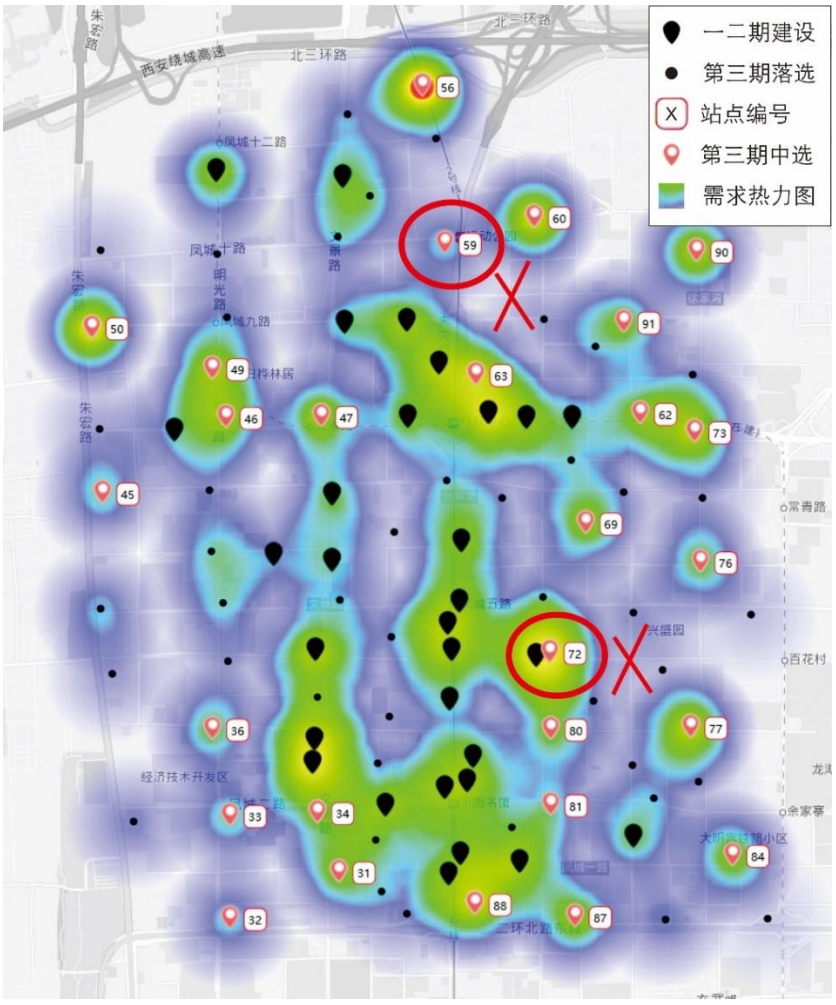

图 7 三期站点选址

由图 7 可以看出，编号为 72 的站点，虽然需求量高，但是距离已建成站点过近，因此二者共同分担需求量，这将导致公共自行车平均使用频次降低，因此，我们应将这个站点删去。

对于 59 号站点来说，其日均需求量在初选的 27 个站点中处在最低位，考虑到建设站点的固定费用较高，需求量较小的站点边际成本[10]就会偏高，进而拉低项目资金利用率。同时，根据附件一的描述，59 号站点位于别墅区，距离购物中心的较远，配套较差，商业主要依靠苏宁等家电售卖商家，前来购买家电的顾客及居住在别墅的业主对于公共自行车的需求不大。因此，我们将 59 号站点删去。

最终，我们得到了 25 最合适的站点选址：

表 4 手动剔除个别不合理项后最终的选择结果

最终通过筛选的车站序号												
56	50	60	77	73	88	90	47	87	49	63	46	69
84	62	80	31	76	36	91	33	45	34	32	81	

4）预算的使用优化

考虑到边际成本以及调配车的工作量，我们将剩余的 5 万元用于购买自行车。

由于每个站点建设都需要高昂的固定费用，而将余额用于采购自行车，以降低单个站点的平均边际成本，服务更多的市民，从而提高了投资收到的成效。同时，适当的增加自行车辆，可以缓解调配车辆的运行压力以及调度所需要的时间。因此，我们将剩余的 5 万元预算用于采购自行车。即，我们用 200 万元预算，建设了 25 个站点，并配备了 750 辆自行车。

5）三期站点的车辆分配和调度方案求解

通过上述对模型的求解，我们确定了第三期的站点建设及自行车采购方案。目前站点总数为 55 个，可供分配的公共自行车数量为 1600 辆。将 55 个站点的位置数据和各点不同时段的需求量数据带入问题一所建立的模型中，我们可以得到具体的自行车分配方案，求解结果见附录四。

4.3 问题三的模型建立与求解

将原有的站点与新增站点进行混合，重新根据租赁点的密集度、各个租赁点

自行车需求量u_i、其他站点归还的自行车数量r_{ij}等因素对调度点进行区域划分，根据分组情况确定调度车辆数量以及各个调度车辆的调配方案。

4.3.1 车辆调度模型

1）车辆调度模型修正

为了保证自行车调度系统在 150min 内完成调度，设调度车辆数为$m(m \geq 2)$，则采用同方向调度方式时，所消耗的时间由调运时间最长的调运车决定。为了合理配置调配资源，避免资源浪费每辆调运车服务的租赁点个数尽量保持均匀，则对问题一中的目标函数模型修正如下：

已知运输车速度为 $V = 30$ km/ h，即 $V = 500$ m/ min，

$$z = minT = \frac{1}{m}(\sum_{i=1}^{m}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\frac{d_{ij}}{V} + |p_j|)J_{ij}) \leq 150 \qquad (21)$$

2）求解目标函数以确定所需调度车辆数量、方案

首先根据问题二的结果确定了新增网点的位置与各个网点车辆初步分配信息。 然后采用 k-Means 聚类算法寻找最快调度方案，可以将网点进行分区（相距近的网点为一个分区）来提高调度效率降低时间复杂度，假设需要 m 辆调 度车，每辆调度车只在一个分区行驶且各分区相互独立，采用问题一中模拟退火算法(10) 得出每辆调度车在各分区的调度时间。然后我们改变 m 值重复上述步骤，知道所有分区的调度时间的最大值不大于 150 分钟。

4.3.2 车辆调度模型求解

1）各个网点需求量

基于问题二的结果我们可以确定各个网点车辆初始分配，结果见附录三。

2）求解模型确定调度车辆数量

首先在 2 辆调度车辆的情况下，根据模拟退火算法求解目标函数（约束条件和问题一中的约束条件相同，此处不再赘述），解得目标函数 $T > 150$ min， 可知仅有 2 辆运输车无法满足题设需求，所以接下来就 3 辆运输车的情形进行了区域分块，进而求解目标函数。

3）3 辆调度车情况下的自行车调度方案

与问题一的调度方案求解相同，本题首先利用优化的 K-means 算法，修正聚类结果中的边缘点，得到最终聚类结果；然后基于模拟退火算法求得最短调配时间 T 与最佳调配方案。

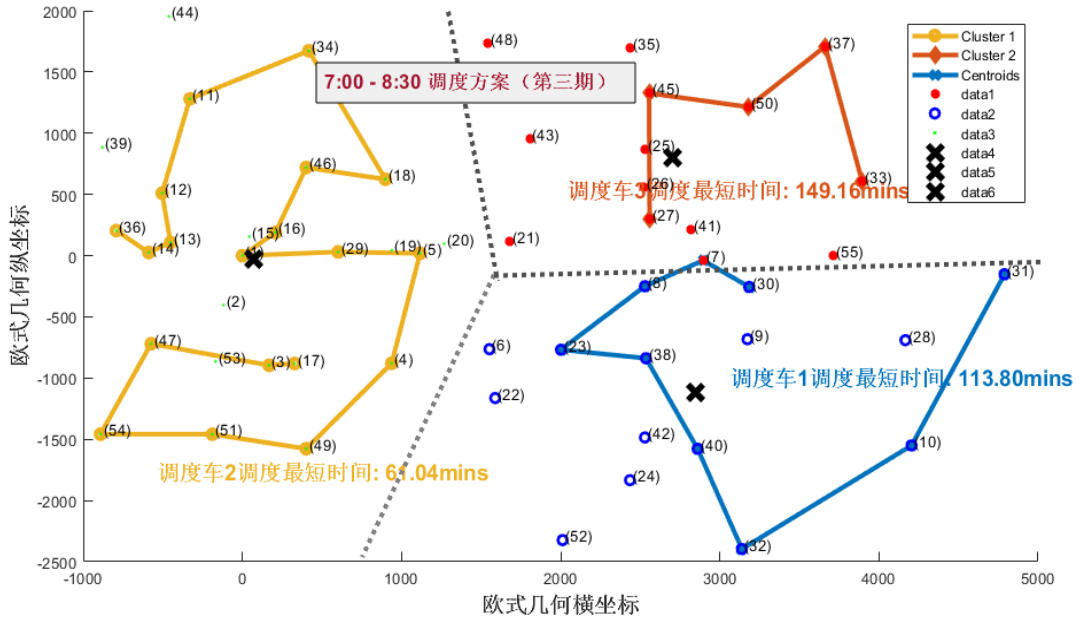

图 8 7:00-8:30 调度方案（第三期）

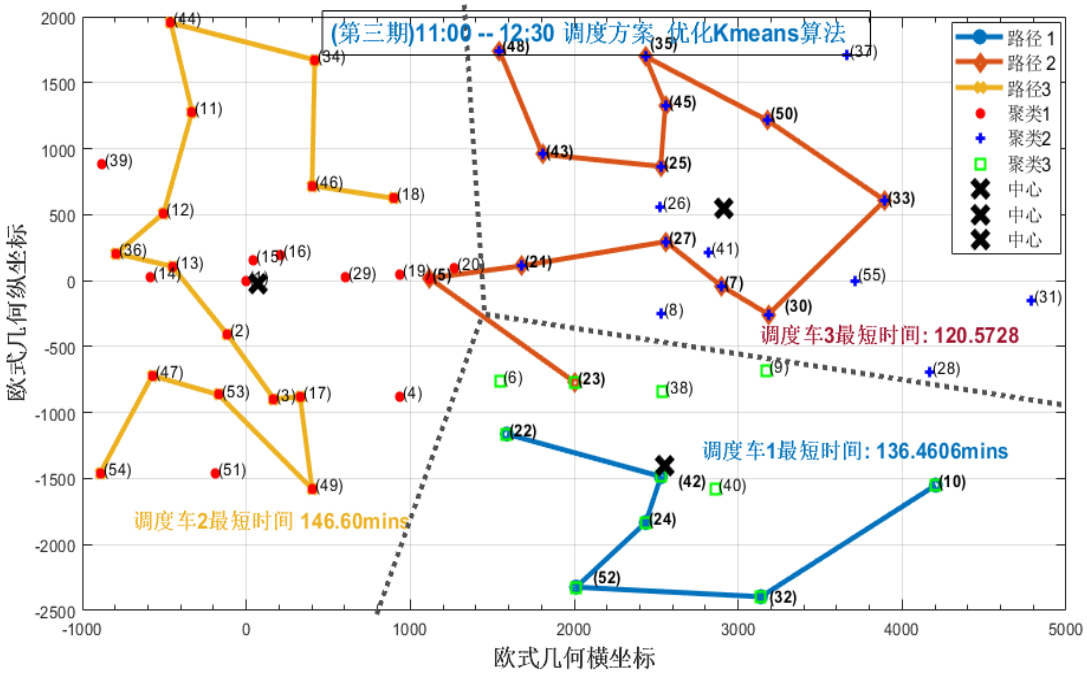

图 9 11:00—12:30 调度方案（第三期）

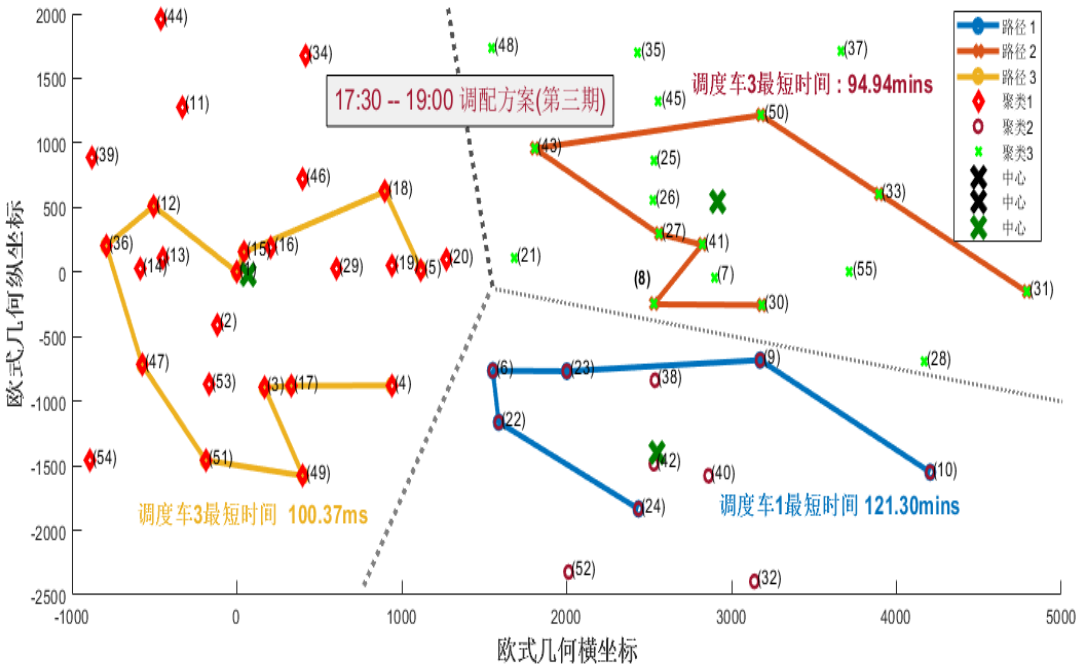

图 10 17:30—19:00 调度方案（第三期）

表 5 最短时间及调度方案表（第三期）

调度时间段	最短时间/min	调配返回方案
07:00-08:30	113.80(右下)	10-32-40-38-23-8-30 (右下)
	61.04(左)	51-54-47-14-13-36-12-11-16-15-1-3-17-49-4-5-19-29-18-46-342 (左)
	149.16(右上)	1-7-27-26-25-45-50-37-33-31(右上)
11:00-12:30	136.46(右下)	10-32-52-24-42-22(右下)
	146.60 (左)	54-47-53-49-17-3-2-13-36-12-11-44-34-46-18(左)
	120.57(右上)	23-5-21-27-7-30-33-50-39-45-25-43-48(右上)
17:30-19:00	121.30(右下)	10-9-23-6-22-24 (右下)
	100.37(左)	4-17-3-49-51-47-36-12-11-15-18-5-20 (左)
	94.94(右上)	31-33-50-43-27-41-8-30(右上)

五、 模型检验

5.1 检验模型是否合理

在第一问中，我们已经得出了最短时间的目标函数，随着时间的推移，也能预测未来某一天的某一段高峰期的公共自行车调配方案。所以首先，为了检验我们在第一问中根据平均分配原则分析调整求得的分配方案是否合理，先对未来一

个星期内的自行车调度最短时间进行预测和分析；然后，将初始的分配方案代入目标规划模型内，不断迭代求出新解；最后分析一次调度所需的最短时间与自行车投入使用以来的时长的关系，如果随着时间的推移，自行车的调度时间波动幅度越来越小直至趋于稳定，那么证明分配方案的确定和模型的建立合理。

将基于公平分配原则得出并已经过分析修正的分配方案代入问题一的模型当中，求出一周内每天三个高峰时间段的最短调度时间，数据见附件。

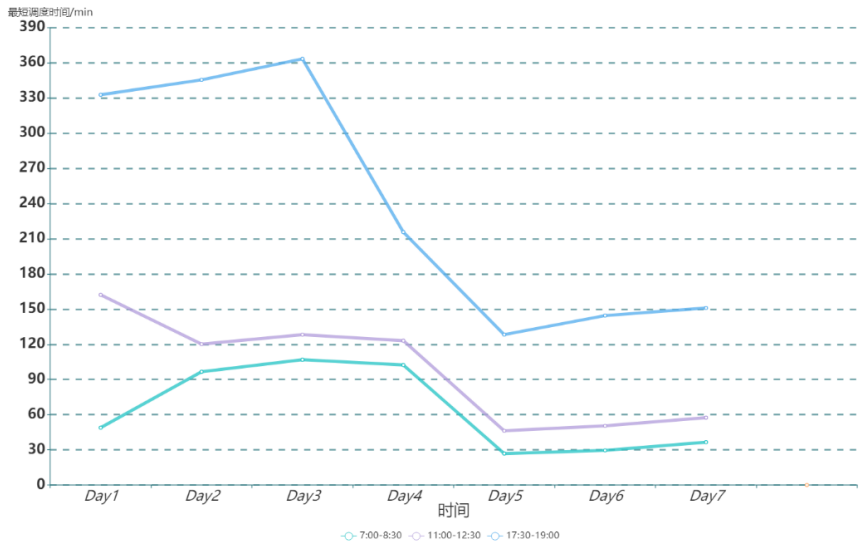

图 11 连续运行一周的三个高峰维护时间变化

由表中数据和图可得：公共自行车的最短调度时间随着其投入使用的时间增加，逐渐减小且趋于稳定；由此推导可知，依据公平分配原则确定并经过分析修正后的分配方案具有一定的合理性。

参考文献

[1] 钱丽丽,邓桂丰.一个公平分配席位的新方案[J].数学的实践与认识,2012,42(18):13-20.

[2] 江志华, 齐文静. 常用作业调度算法的分析与评价[J]. 乐山师范学院学报, 2008, 23(012):57-59.

[3] 周爱武,崔丹丹,潘勇.一种优化初始聚类中心的 K-means 聚类算法[J].微型机与应用. 2011(13).

[4] Kriegel, Hans-Peter; Schubert, Erich; Zimek, Arthur (2016). "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?". Knowledge and Information Systems. 52 (2): 341–378. doi:10.1007/s10115-016-

1004-2. ISSN 0219-1377.

[5] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J., The Traveling Salesman Problem, 2006, ISBN 0-691-12993-2.

[6] Steinbrunn M ,Moerkotte G, Kemper A. Heuristic and Ran2 domized Optimization for the Join Ordering Problem[J] . The VLDB Journal , 1997 , 6 (3) :8 - 17.)

[7] 姚新, 陈国良. 模拟退火算法及其应用[J]. 计算机研究与发展, 1990(7):1-6.

[8] 程龙. 城市公共自行车租赁点选址及调度模型研究[D]

[9] 王懿. 小城市公共自行车选址及需求优化设计[J]. 交通科学与工程, 2016, 32(01):99-103.

[10] 鲍娜. 城市公共自行车租赁点选址决策及调度模型研究[D]. 长安大学, 2012.

附录

附录 1　三个时段 30 个租赁点车辆的需求量与变化量

编号	网点位置	7:00-8:30 需求量	7:00-8:30 改变量	11:00-12:30 需求量
1	经发大厦	15	9	22
2	可口可乐北门	23	0	24
3	经发国际会馆	38	-15	28
4	昆仑银行	38	-14	32
5	赛高街区	17	7	23
6	西安中学西门	32	-7	12
7	运动公园东门	13	12	35
8	运动公园南门	40	-17	22
9	管委会	26	-3	19
10	出口加工区广场	18	6	31
11	鼎新花园	18	5	24
12	西安外国语学校	35	-15	20
13	雅荷花园	7	16	27
14	御道华城	12	9	19
15	天地时代广场	38	-18	16
16	市图书馆	17	8	22
17	文景观园	21	7	39
18	长庆电视台	23	0	38
19	移动公司	28	-6	20
20	凤城五路	23	-1	21
21	凤城六路	15	9	33
22	中登家园北门	35	-10	7

23	万华园	15	11	20
24	粤华凤城家园	34	-10	9
25	市人大	21	4	9
26	市委	23	2	21
27	政务大厅	35	-12	30
28	首创国际城	18	6	21
29	中登广场	13	11	22
30	运动公园北门	18	6	30

附录二 三个时段 30 个租赁点车辆的需求量与变化量

编号	网点位置	7:00-8:30 需求量	7:00-8:30 改变量	11:00-12:30 需求量
1	经发大厦	0	15	9
2	可口可乐北门	0	23	0
3	经发国际会馆	-1	38	-15
4	昆仑银行	-9	38	-14
5	赛高街区	0	17	7
6	西安中学西门	11	32	-7
7	运动公园东门	-12	13	12
8	运动公园南门	2	40	-17
9	管委会	5	26	-3
10	出口加工区广场	-9	18	6
11	鼎新花园	-1	18	5
12	西安外国语学校	3	35	-15
13	雅荷花园	-5	7	16
14	御道华城	5	12	9
15	天地时代广场	7	38	-18
16	市图书馆	0	17	8
17	文景观园	-15	21	7
18	长庆电视台	-15	23	0
19	移动公司	4	28	-6
20	凤城五路	3	23	-1
21	凤城六路	-11	15	9
22	中登家园北门	16	35	-10
23	万华园	2	15	11
24	粤华凤城家园	14	34	-10
25	市人大	15	21	4
26	市委	2	23	2
27	政务大厅	-7	35	-12
28	首创国际城	3	18	6
29	中登广场	1	13	11
30	运动公园北门	-7	18	6

附录三 调整后各租赁点的初始分配量表

租赁点	1	2	3	4	5	6	7	8	9	10
分配数	15	15	30	40	17	32	23	40	26	40

租赁点	11	12	13	14	15	16	17	18	19	20
分配数	40	40	7	12	38	17	28	40	28	23

租赁点	21	22	23	24	25	26	27	28	29	30
分配数	23	35	15	34	21	21	40	20	13	32

附录四 55 个租赁点按需求量大小顺序表格

租赁点	1	2	3	4	5	6	7	8	9	10
分配数	15	15	30	40	17	32	23	40	26	40

租赁点	11	12	13	14	15	16	17	18	19	20
分配数	40	40	7	12	38	17	28	40	28	23

租赁点	21	22	23	24	25	26	27	28	29	30
分配数	23	35	15	34	21	21	40	20	13	32

租赁点	56	50	60	77	73	88	90	47	87	49
分配数	40	38	33	33	20	25	23	23	13	33

租赁点	63	46	69	84	62	80	31	76	36	91
分配数	24	18	39	21	15	18	40	19	15	32

租赁点	33	45	34	32	59					
分配数	8	18	17	40	15					

附录五 kMeans 聚类对调度区域进行分块

```
%带入 100 个点的实际距离值
%以第一个点为坐标原点，下列值为其横纵坐标，画图
X = xlsread('data_co.xlsx','data_co');
isNeedTrans = xlsread('firstTrans.xlsx','sheet1','F2:F31');
opts = statset('Display','final');

flag=zeros(30,1);
X_new=zeros(30,2);
isNeedTrans_new=zeros(30,1);
isNeedTrans_new1=zeros(30,1);
```

```
isNeedTrans_new2=zeros(30,1);
j = 1; k = 1; n = 1;
%  数据预处理
for i = 1: 30
    % if(isNeedTrans(i) <= 5) && (isNeedTrans(i) >= 0)
    % flag(i) = 0;
    % end
    X_new(j,:) = X(i,:);
    isNeedTrans_new(j) = isNeedTrans(i);
    if(isNeedTrans(i) > 5)
        flag(j) = 1;
         j = j + 1;
    elseif(isNeedTrans(i) < 0)
        flag(j) = -1;
        j = j + 1;
    end
end

%调用 Kmeans 函数
%X 100*2 的数据矩阵
%Idx N*1 的向量,存储的是每个点的聚类标号
%Ctrs K*P 的矩阵,存储的是 K 个聚类质心位置
%SumD 1*K 的和向量,存储的是类间所有点与该类质心点距离之和
%D N*K 的矩阵，存储的是每个点与所有质心的距离;每个点到每个质心的距离。

[Idx,Ctrs,SumD,D] = kmeans(X_new,2,'Replicates',3,'Options',opts);
X_new1 = zeros(30,2);
X_new2 = zeros(30,2);
for i = 1: j-1
```

```
        if(Idx(i) == 1)
            X_new1(k,:) = X_new(i,:);
            isNeedTrans_new1(k) = isNeedTrans(i);
            k = k + 1;
        end
        if(Idx(i) == 2)
            X_new2(n,:) = X_new(i,:);
            isNeedTrans_new2(n) = isNeedTrans(i);
            n = n + 1;
        end
    end
    %画出聚类为 1 的点。X(Idx==1,1),为第一类的样本的第一个坐标；
X(Idx==1,2)为第二类的样本的第二个坐标
    plot(X_new(Idx==1,1),X_new(Idx==1,2),'r*','MarkerSize',5)
    hold on
    plot(X_new(Idx==2,1),X_new(Idx==2,2),'bo','MarkerSize',5)
    hold on
    %plot(X(Idx==3,1),X(Idx==3,2),'g.','MarkerSize',14)
    for i=1:j-1
        text(X_new(i,1)+18,X_new(i,2)+43,['(', num2str(i),')']);
        text(X_new(i,1)+23,X_new(i,2)-33,num2str(isNeedTrans_new(i)));
        hold on
    end
    %绘出聚类中心点,kx 表示是圆形
    plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)
    plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)
    %plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)

    legend('Cluster 1','Cluster 2','Centroids','Location','NW')=
```

附录五 模拟退火算法求解最短路程

```
clc
% 需求量
uj_1 = [15 23 38 38 17 32 13 40 26 18 18 35 7 12 38 17 21 23 28 23
15 35 15 34 21 23 35 18 13 18];
uj_2 = [22 24 28 32 23 12 35 22 19 31 24 20 27 19 16 22 39 38 20
21 33 7 20 9 9 21 30 21 22 30];
uj_3 = [30 35 31 22 28 10 34 16 19 37 27 33 28 13 12 6 23 32 20 20
34 6 38 10 13 20 35 20 17 38];
z = xlsread('data.xlsx'); % 距离矩阵
x = xlsread('dist.xlsx');
z_rev = 1./z;  % 距离的倒数
z_part = z_rev(1:30,1:30);
z_sum = 0;
ZSum =[];
M = []; % 概率
Back1=[];Back2=[];Back3=[];  % 归还量
p_1=[];p_2=[];p_3=[];  % 调度量
K_1=zeros(30, 1);K_2=zeros(30,1);K_3=zeros(30,1); % 0-1 矩阵
last_11 = []; last_12 = []; last_21 = []; last_22 = []; last_31=[];
last_32 = [];
p_11 = []; p_12 = []; p_21 = []; p_22 = []; p_31=[]; p_32 = [];
CMAX = 40; % 车辆最大数

% 求归还的概率矩阵（30*30 矩阵）
for i=1:1:30
z_sum = 0;
 for j=1:1:30
     if z_part(i,j) < 1
          z_sum = z_sum + z_part(i,j);
```

```
     end
end
ZSum = [ZSum z_sum];
M(i, :) = z_part(i, :) ./ZSum(:, i);  % 归还的概率矩阵（30*30 矩阵）
end

% 第一轮
 % 第一轮下来的归还量
for i=1:30
    r = 0;
    for j = 1:30
        if M(i,j) < 1
            r = r + M(i, j) * uj_1(j);  % 所有其他的 29 个租赁点 j 归还给租赁点 i 的单车数量
        end
    end
     Back1 =[Back1 r];  % 一轮下来的归还量
end
% 第一轮归还和骑出之后还剩下的量
x_1 = x' - uj_1 + Back1;
% 第一轮的调度（调度量以及是否需要调度）
for i = 1: 30
    p_1(i) = x_1(i) - uj_2(i)*1.1; % 调度量（与调度与否无关）
    % 现在车辆数目少于需求量的 110%
    if x_1(i) < (uj_2(i) * 1.1)
        K_1(i) = 1;
    end
    % 现在车辆数目超出需求量的 110%5 辆（5 辆车：预设的阈值）
    if uj_2(i)*1.1 - x_1(i) >= 1
        K_1(i) = 1;
```

```
        end
        % 现在车辆数目超出上下限的范围
        if (x_1(i) < CMAX * 0.2) || (x_1(i) > CMAX * 0.9)
            K_1(i) = 1;
        end
    end
    % 进行调度
    for i = 1: 30
        if K_1(i) == 1  % 需要进行调度，不需要进行调度的不做处理
            x_1(i) = uj_2(i) * 1.1;
            if x_1(i) > CMAX * 0.9  % 如果需求量的 110%超出了规定的上限
                x_1(i) = CMAX * 0.9;
            end
        end
    end

    %第二轮
    % 第二轮下来的归还量
    for i=1:30
        r = 0;
        for j = 1:30
            if M(i,j) < 1
                r = r + M(i, j) * uj_2(j);  % 所有其他的 29 个租赁点 j 归还
给租赁点 i 的单车数量
            end
        end
         Back2 =[Back2 r];  % 一轮下来的归还量
    end
    % 第二轮归还和骑出之后还剩下的量
    x_2 = x_1 - uj_2 + Back2;
```

```
% 第二轮的调度（调度量以及是否需要调度）
for i = 1: 30
    p_2(i) = x_2(i) - uj_3(i)*1.1; % 调度量（与调度与否无关）
    % 现在车辆数目少于需求量的 110%
    if x_2(i) < (uj_3(i) * 1.1)
        K_2(i) = 1;
    end
    % 现在车辆数目超出需求量的 110%5 辆（5 辆车：预设的阈值）
    if uj_3(i)*1.1 - x_2(i) >= 5
        K_2(i) = 1;
    end
    % 现在车辆数目超出上下限的范围
    if (x_2(i) < CMAX * 0.2) || (x_2(i) > CMAX * 0.9)
        K_2(i) = 1;
    end
end
% 进行调度
for i = 1: 30
    if K_2(i) == 1  % 需要进行调度，不需要进行调度的不做处理
        x_2(i) = uj_3(i) * 1.1;
    end
end

%第三轮
% 第三轮下来的归还量
for i=1:30
    r = 0;
    for j = 1:30
        if M(i,j) < 1
            r = r + M(i, j) * uj_3(j);  % 所有其他的 29 个租赁点 j 归还
```

给租赁点 i 的单车数量

```
            end
        end
         Back3 =[Back3 r];  % 一轮下来的归还量
    end
    % 第三轮归还和骑出之后还剩下的量
    x_3 = x_2 - uj_2 + Back3;
    % 第三轮的调度（调度量，全调度回到初始值进行迭代）
    for i = 1: 30
        x_new = x';
        p_3(i) = x_3(i) - uj_1(i)*1.1; % 调度量

        % 现在车辆数目少于需求量的 110%
        if x_3(i) < (uj_1(i) * 1.1)
            K_3(i) = 1;
        end
        % 现在车辆数目超出需求量的 110%5 辆（5 辆车：预设的阈值）
        if uj_1(i)*1.1 - x_3(i) >= 1
            K_3(i) = 1;
        end
        % 现在车辆数目超出上下限的范围
        if (x_3(i) < CMAX * 0.2) || (x_3(i) > CMAX * 0.9)
            K_3(i) = 1;
        end

    end
    % 进行调度
    for i = 1: 30
        x_3(i) = x_new(i);
       if K_3(i) == 1  % 需要进行调度，不需要进行调度的不做处理
```

```
        x_3(i) = uj_1(i) * 1.1;
        if x_3(i) > CMAX * 0.9  % 如果需求量的110%超出了规定的上限
            x_3(i) = CMAX * 0.9;
        end
    end
end

%带入30个点的实际距离值
%以第一个点为坐标原点，下列值为其横纵坐标，画图
X = xlsread('data_co.xlsx','data_co');
isNeedTrans = xlsread('firstTrans.xlsx','sheet1','F2:F31');
opts = statset('Display','final');
j = 1;
% 数据预处理
for i = 1: 30
    X_new(i,:) = X(i,:);
end

%调用Kmeans函数
%X 100*2的数据矩阵
%Idx N*1的向量,存储的是每个点的聚类标号
%Ctrs K*P的矩阵,存储的是K个聚类质心位置
%SumD 1*K的和向量,存储的是类间所有点与该类质心点距离之和
%D N*K的矩阵，存储的是每个点与所有质心的距离;每个点到每个质心的距离。

[Idx,Ctrs,SumD,D] = kmeans(X_new,2,'Replicates',3,'Options',opts);

%画出聚类为1的点。X(Idx==1,1),为第一类的样本的第一个坐标; X(Idx==1,2)为第二类的样本的第二个坐标
plot(X_new(Idx==1,1),X_new(Idx==1,2),'r*','MarkerSize',5)
```

```
hold on
plot(X_new(Idx==2,1),X_new(Idx==2,2),'bo','MarkerSize',5)
hold on
%plot(X(Idx==3,1),X(Idx==3,2),'g.','MarkerSize',14)
for i=1:30
    text(X_new(i,1)+18,X_new(i,2)+43,['(', num2str(i),')']);
%    text(X_new(i,1)+23,X_new(i,2)-33,num2str(X_new(i)));
    hold on
end
%绘出聚类中心点,kx 表示是圆形
plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)
plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)
%plot(Ctrs(:,1),Ctrs(:,2),'kx','MarkerSize',14,'LineWidth',4)

legend('Cluster 1','Cluster 2','Centroids','Location','NW')

for i = 1: 30
    if(Idx(i) == 1) && (K_1(i) == 1)
        last_11 = [last_11; X_new(i,:)];
        p_11 = [p_11, p_1(i)];
    end
    if(Idx(i) == 2) && (K_1(i) == 1)
        last_12 = [last_12; X_new(i,:)];
        p_12 = [p_12, p_1(i)];
    end
     if(Idx(i) == 1) && (K_2(i) == 1)
        last_21 = [last_21; X_new(i,:)];
        p_21 = [p_21, p_2(i)];
    end
    if(Idx(i) == 2) && (K_2(i) == 1)
```

```
            last_22 = [last_22; X_new(i,:)];
            p_22 = [p_22, p_2(i)];
        end
         if(Idx(i) == 1) && (K_3(i) == 1)
            last_31 = [last_31; X_new(i,:)];
            p_31 = [p_31, p_3(i)];
        end
        if(Idx(i) == 2) && (K_3(i) == 1)
            last_32 = [last_32; X_new(i,:)];
            p_32 = [p_32, p_3(i)];
        end
    end
    % 模拟退火算法求出最优解和最短路径
    cities11=last_11'; cities12=last_12'; cities21=last_21';
    cities22=last_22'; cities31=last_31'; cities32=last_32';
    load11=p_11; load12=p_12; load21=p_21;
    load22=p_22; load31=p_31; load32=p_32;
```

附录六 Lingo 程序 公平分配原则求解自行车未调整初始分配量

```
sets:
    Warehouse /1..30/: a;
    Customer /1..30/: b;
endsets
data:
b=15,23,38,38,17,32,13,40,26,18,18,35,7,12,38,17,21,23,28,23,15,35,15,34,21,2
3,35,18,13,18;
enddata
min = @sum( Warehouse(i):(a(i)-b(i))^2/a(i));
@sum(Warehouse(i):a(i))=850;
@for(Warehouse(i):a(i)-b(i)>0);
```

```
@for(Warehouse(i):a(i)<=40);
@for(Warehouse(i):@bnd(1,a,40));
!限制变量 a
```

附录七 求解 55 个租赁点的分配调度情况

```
sets:
    Warehouse /1..55/: a;
    Customer /1..55/: b;
endsets
data:
b=15,23,38,38,17,32,13,40,26,18,18,35,7,12,38,17,21,23,28,23,15,35,15,34,21,23,35,18,13,18,40,38,33,33,20,25,23,23,13,33,24,18,39,21,15,18,40,19,15,32,8,18,17,40,15;
enddata
min = @sum( Warehouse(i):(a(i)-b(i))^2/a(i));
@sum(Warehouse(i):a(i))=1600;
@for(Warehouse(i):a(i)-b(i)>0);
@for(Warehouse(i):a(i)<=40);
@for(Warehouse(i):@bnd(1,a,40));
!限制变量 a
```

附录八 0-1 规划程序，初步确定站点选址

```
clear,clc;
r=0.5;  %权重系数，不妨取 0.5
f=xlsread('data_ques2.xls',1);
F=r*f/2117+(1-r)*f/28/700/3;      %对两种指标进行按权重相加
A=ones(1,70);
b=25;    %给定约束条件
intcon = [1:70];
[x,fval]=intlinprog(-F,intcon,[],[],A,b,zeros(70,1),ones(70,1));
Y=[];
```

```
    for i=1:length(x)
        if x(i)~=0
            Y=[Y,i+30];
        end
    end
    factor=-fval
```

附录九 python 程序 经纬度转为欧式坐标横纵坐标，并求出邻接矩阵

```
from math import *
import csv
import pandas as pd


def get_distance(x1, y1, x2, y2):
    # 根据欧式坐标计算两个点距离
    dlon = x1 - x2
    dlat = y1 - y2
    dist = abs(dlon) + abs(dlat)
    return dist


with open('data.csv', 'r', encoding='utf-8') as csvfile:
    reader = csv.reader(csvfile)
    column1 = [row[4] for row in reader]
with open('data.csv', 'r', encoding='utf-8') as csvfile:
    reader = csv.reader(csvfile)
    column2 = [row[5] for row in reader]
```

```
print(column1)

print(column2)


g = open("data_done_55.csv", encoding='utf-8', mode='w')

g.write(" ,")

for _ in range(55):

    g.write("{},".format(_+1))

g.write("\n")

for i in range(55):

    g.write("{},".format(i+1))

    for j in range(55):

        dis = get_distance(float(column1[i]), float(

            column2[i]), float(column1[j]), float(column2[j]))

        g.write("{:0},".format(dis))

    g.write("\n")

g.close()
```

附录十 python 程序 55 个租赁点+3 辆调度车的聚类算法：

```
'''

@Author: your name

@Date: 2020-07-25 15:18:37

@LastEditTime: 2020-07-26 08:47:48

@LastEditors: Please set LastEditors
```

```
@Description: In User Settings Edit
@FilePath: \经纬-距离转换\kMeans58.py
'''
import xlrd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


#  从 Excel 中读取数据存入数组
data = xlrd.open_workbook('data_co.xlsx')
table = data.sheets()[0]    #  打开第一个工作表
data = []
for i in range(table.nrows):    #  第一个工作表的行数循环
    if i == 0:            #  第一行的表头名不读入
        continue
    else:
        data.append(table.row_values(i)[1:])


featureList = ['x', 'y']
mdl = pd.DataFrame.from_records(data, columns=featureList)
```

```
# 聚类
k = 3  # 需要进行的聚类类别数
mdl_new = np.array(mdl[['x','y']])  # 转化为数组
seed = 9  # 设置随机数
clf = KMeans(n_clusters=k, random_state=seed, n_jobs=4)  # 聚类, n_jobs 是并行数，一般等于 CPU 数较好
if __name__ == '__main__':
    clf.fit(mdl_new)  # 训练\拟合模型
    #print(clf.cluster_centers_)  # 查看 KMeans 聚类后的 3 个质心点的值。
    mdl['label'] = clf.labels_  # 对原数据表进行类别标记
    c = mdl['label'].value_counts()
    # print(mdl.values)
    # 一共聚类成 3 类，最后一列是类别数（0,1,2）

    r1 = pd.Series(clf.labels_).value_counts()  # 统计各个类别的数目
    r2 = pd.DataFrame(clf.cluster_centers_)  # 找出聚类中心
    r3 = pd.DataFrame(mdl.values)  # 找出聚类中心
    r = pd.concat([r2, r1], axis=1)  # 横向连接（0 是纵向），得到聚类中心对应的类别下的数目
    r.columns = ['聚类中心 x', '聚类中心 y', '类别数目']
    r3.columns = ['x', 'y', '聚类类别']
    print(r3[u'聚类类别'])
    print(r)
```

```
for i in range(k):
    print(r3[u'聚类类别'] == i)


numSamples = len(mdl_new)
centroids = clf.labels_
print (centroids, type(centroids))  # 显示中心点
print(clf.inertia_)  # 显示聚类效果
mark = ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r', 'pr']
mark_1 = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb']
# 画出质点，用特殊图型
centroids = clf.cluster_centers_
with plt.style.context(['science', 'scatter']):
    fig, ax = plt.subplots()
    ax.set(xlabel='Longitude x')
    ax.set(ylabel='Latitude y ')
    ax.fill_between([-2, 2], [-2.2, 1.8], [-1.8, 2.2],
                            color='dodgerblue', alpha=0.2, lw=0)
    for i in range(numSamples):
        #markIndex = int(clusterAssment[i, 0])
        # mark[markIndex])
        ax.plot(mdl_new[i][0], mdl_new[i][1], mark[clf.labels_[i]])
        # plt.plot(mdl_new[i][0], mdl_new[i][1], mark[clf.labels_[i]])
```

```
    for i in range(k):
        ax.plot(centroids[i][0], centroids[i][1], mark_1[i], markersize=8)
        # plt.plot(centroids[i][0], centroids[i][1], mark_1[i], markersize=12)

    ax.legend(title='Sample', loc=2)
    plt.show()
```

附录十一 python 程序 对于未优化的 K-means 聚类和优化后的 K-means 聚类求解的最短调度时间

```
'''
@Author: your name
@Date: 2020-07-26 10:53:29
@LastEditTime: 2020-07-26 13:15:33
@LastEditors: Please set LastEditors
@Description: In User Settings Edit
@FilePath: \经纬-距离转换\density.py
'''
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import xlrd
```

```
import warnings; warnings.filterwarnings(action='once')
from pylab import mpl
import matplotlib.lines as mlines


plt.style.use('seaborn-whitegrid')
mpl.rcParams['font.sans-serif'] = ['FangSong']    # 指定默认字体
mpl.rcParams['axes.unicode_minus'] = False          # 正常显示负号
# Import Data
df = pd.read_csv(
        "gdppercap.csv")


left_label = [str(c) + ', ' + str(round(y))
                        for c, y in zip(df.continent, df['优化前'])]
right_label = [str(c) + ', ' + str(round(y))
                         for c, y in zip(df.continent, df['优化后'])]
klass = ['red' if (y1-y2) < 0 else 'green' for y1,
                y2 in zip(df['优化前'], df['优化后'])]


# draw line
#                 https://stackoverflow.com/questions/36470343/how-to-draw-a-line-with-matplotlib/36479941


def newline(p1, p2, color='black'):
```

```
        ax = plt.gca()
        l = mlines.Line2D([p1[0], p2[0]], [p1[1], p2[1]], color='red' if p1[1] -
                                    p2[1] > 0 else 'green', marker='o', markersize=6)
        ax.add_line(l)
        return l

fig, ax = plt.subplots(1, 1, figsize=(14, 14), dpi=80)

# Vertical Lines
ax.vlines(x=1, ymin=0, ymax=250, color='black',
                alpha=0.7, linewidth=1, linestyles='dotted')
ax.vlines(x=3, ymin=0, ymax=250, color='black',
                alpha=0.7, linewidth=1, linestyles='dotted')

# Points
ax.scatter(y=df['优化前'], x=np.repeat(1, df.shape[0]),
                 s=10, color='black', alpha=0.7)
ax.scatter(y=df['优化后'], x=np.repeat(3, df.shape[0]),
                 s=10, color='black', alpha=0.7)

# Line Segmentsand Annotation
for p1, p2, c in zip(df['优化前'], df['优化后'], df['continent']):
```

```
        newline([1, p1], [3, p2])
        ax.text(1-0.05, p1, c + ', ' + str(round(p1)), horizontalalignment='right',
                verticalalignment='center', fontdict={'size': 18})
        ax.text(3+0.05, p2, c + ', ' + str(round(p2)), horizontalalignment='left',
                verticalalignment='center', fontdict={'size': 18})

# 'Before' and 'After' Annotations
ax.text(1-0.05, 240, 'Before', horizontalalignment='right',
            verticalalignment='center', fontdict={'size': 18, 'weight': 900})
ax.text(3+0.05, 240, 'After', horizontalalignment='left',
            verticalalignment='center', fontdict={'size': 18, 'weight': 900})

# Decoration
ax.set_title("调度需求量坡度图，优化前 vs 优化后",
                fontdict={'size': 22})
ax.set(xlim=(0, 4), ylim=(0, 120))
ax.set_xticks([1, 3])
ax.set_xticklabels(["Kmeans 优化前", "Kmeans 优化后"], fontsize=20)
plt.yticks(np.arange(0, 250, 20), fontsize=20)

# Lighten borders
plt.gca().spines["top"].set_alpha(.0)
```

```
plt.gca().spines["bottom"].set_alpha(.0)

plt.gca().spines["right"].set_alpha(.0)

plt.gca().spines["left"].set_alpha(.0)

plt.show()
```