物流运输任务规划

摘要

双十一购物节让众多的商家及购物客户疯狂，物流业在这一天承担超负荷的商品运送任务。城市物流运输作为一种新型的现代物流模式，城市物流运输的需求日益增长，为减少企业的运输成本，制定一个好的车辆路线优化方案极为重要。

对于问题一：要完成商家的运送任务，制定最佳的行车路线和货车调度策略，使得货车在城内的运送时间总和最小。首先，首先我们对 68 个商家采用 k-means 聚类方法，初始聚类中心的个数从 1-18 进行遍历，其分为四类时城内总距离最短。根据商家在城内的位置坐标，我们采用直接法和交点位置法求解各个商家的最短出城距离，然后定义了适应度，改进传统交叉法为交换变异法，利用遗传算法求解行驶路线，中转站货车调度策略，其结果（见表 1），在城内运出最短总时间为 16.65 小时，运算结果较优。最终，我们对每个商铺的坐标在±5%的范围内随机仿真进行灵敏度分析，与原数据对比，完成任务总时间误差不超过 7%，说明模型比较稳定。

对于问题二：考虑装载容量与运输成本，完成任务制定最佳的行车方案，保证时间总和与总的运费尽可能的少。因货车装载量有限制，最少分为 10 类才可满足条件，我们对 68 个商家用 k-means 聚类分别求出 10-18 类的规划分布。然后我们对每一类用 lingo 求出货车到每个商家中心最优的调度方案，为花费最少的求解做准备。对每类沿用问题一的遗传算法，定义新的适应度函数，建立最少运费模型，求解出将商家分为 17 类，即派出 17 辆车时运费最少，再保留其优秀种群，以最少时间为目标，求出每一小类的最优路径，最短运送时间为 26.025 小时，最少运费为 3033.1 元，行车轨迹、装载清单和货车调度结果见表 7。最后，对商家坐标仿真，完成任务总时间误差不超过 1%，说明模型较稳定，得到的结果比较可靠。

对于问题三：购买量增加为平时的 5 倍，求出需准备车辆数，制定最佳行车与调度方案。根据问题二的算法，考虑有些车辆只前往了一个点就回中转站，结合实际考虑一辆车派往多个地点的情况，对时间进行约束，即所有出动的车辆最后一次入城时间在夜间 24：00 之后凌晨 4：00 之前，优化后可知该问可以由 7 辆车进行遍历。当运货量扩大 5 倍时，采用运费最小的方案，以局部最优代替整体最优，求得所需货车为 35 辆，其中 20 辆Ⅰ型车，15 辆Ⅱ型车，行车路线和调度策略见表 11 和 12。最后，对模型做了灵敏度分析，其误差结果不超过 5%，所以该模型比较稳定。

对于问题四：每个商场中心货物要求直接运送到 B03,B05 中转站，重新考虑二、三问。再重新讨论问题二，沿用第二问方法，改变遗传算法中求解适应度的函数。首先判断该路径中是否有中心商场，若有则选择离路径最后一个点最近的 B03 或 B05 中转站，即已知路径，程序动态选择该货车的去处，从而得到每条路径的适应度函数。最终选择出动 18 辆车全部出动；再重新讨论第三题，沿用问题四对问题二讨论的结论，类似于问题三的方法，对问题二结果的车辆进行优化，得出在规定时间内所有商场的货物进行装载需要 8 辆车，所以在考虑航空急件的情况下，第三问的结果为 40 辆车，其中Ⅰ型车 25 辆，Ⅱ型车 15 辆，运输总费用为 17112 元，城内总运行时间为 8076 分钟，行车轨迹等详见文中。最后，对数据进行仿真，其误差结果不超过 5%，说明该模型比较稳定。

最后，我们对建立的模型的优缺点进行了评价，并探讨了模型的推广前景和改进方向。

关键词：物流运输　　k-means 聚类　　遗传算法　　最短路　　动态规划

一、问题的背景与重述

1.1 问题的背景

双十一购物节让众多的商家及购物客户疯狂，物流业在这一天承担超负荷的商品运送任务。现代物流业是以运输业为重点，以信息技术为支撑，以现代制造业和商业为基础，集系统化，现代化，仓储现代化为一体的综合性产业。物流业是国民经济的基础产业，涉及相关领域广，物流业的发展可以推动产业结构调整升级，其发展程度是衡量国民经济的重要标志。

城市是区域经济中心，物流以城市为节点，随着城市的快速发展，越来越多的大型物流公司集结在城市周边，物流的结构、形态及布局与城市的发展息息相关。从这个意义上来说，城市发展与物流发展是密切相关的。我国经济增长阶段的转换期已经开始，物流增速有所趋缓，在这样的大环境下，物流业在国民经济中的地位和作用更加突出，进一步促进和推动其他产业的变革以及流通模式的转变，对城市经济发展和城市建设显得愈发重要，物流与城市发展的关系比以往任何时期都更加紧密，更为重要。

随着大数据、物联网等技术的研发，物流信息技术正在飞速发展。物流业的发展，与运输业紧密相关，车辆路径问题是公认的 NP-hard 问题，之前多数求解的目标最短路，本问题求解是最短时间和最短路，属于双目标问题，减少企业的运输成本，制定最优的车辆行驶路线和调度方案显得十分重要。

1.2 问题的重述

城市白天不允许货车入城，只能限定在夜间 24 点至凌晨 4 点入城。城区按各商业中心商场为圆心，7km 为半径的总体外包络为界（见附件 2 的图），城区外运行不受时间限制。某城市物流集团有 B01~B07 等 7 个物流分公司中转站，各中转站均配备一定数量的货车（各中转站具体坐标、配备的货车数量及货车容量见附件 1，附件 4，位置示意图见附件 2）。物流集团需要调配 7 个物流公司中转站的货车夜间进城收集需要运送的商品，每个收货点收货装载平均大约 10 分钟，货车执行完任务后需返回原物流公司中转站。

根据任务要求，需完成收货目标有 A01~A10 等 10 个商业区域，每个商业区域包含数量不等的网销商家，其中中心商城是该商业区域中网销规模较大综合性商场，所有商业区域的商家的具体坐标参数见附件 3，假设每个网销商家之间都有道路相连，路长简化为直线距离。每个商家按客户的订单需发货品数量及规格见附件 5。本文建立了合适的数学模型，并解决了如下的问题：

问题一：需要物流集团公司完成 10 个商业区域（共 68 个商家）的货车运送任务，若不考虑装载容量及运输成本，制定最佳的行车路线和货车调度策略，使得所有运货车辆在城内的运送工作时间总和最小。

问题二：在考虑装载容量及运输成本的情况下，结合所给数据，完成 10 个商业区域收运商品的需求，为物流集团货车运送任务拟制定最佳的行车路线和货车调度策略，保证所有运货车辆在城内的运送时间总和以及所有运货车辆总的运费尽可能的少。

问题三：双十一的购买量是平时的 5 倍,如果要完成当天的运输任务，给出需要准备两种类型货车的数量，货车分配到各个中转站的方案，并制定最佳行车路线与调度策略。

问题四：如果每个中心商场的货物都需要发航空急件，航空件要求直接运送到 B03 和 B05 中转站，如果允许车辆跨站运输，重新考虑第二，三问，制定最佳行车路线和货车调度模型。

二、模型的假设

（1）假设每个网销商家之间都有道路相连，路长简化为直线距离。
（2）假设货车城区内外平均车速 25km/h（转弯及等红绿灯计入在内）。
（3）假设没有重大交通意外事故的发生。
（4）假设忽略单行线的影响，即认为所有的路段均可双向行驶。
（5）假设已知各商场的坐标值与真实值的误差在可控范围之内。

三、主要符号说明

符号	符号说明
$I_{(i,j)}$	第 i 辆车的路程上第 j 个点的最短入城距离
$O_{(i,j)}$	第 i 辆车的路程上第 j 个点的最短出城距离
x_{ij}	第 i 辆车第 j 个点的横坐标
y_{ij}	第 i 辆车第 j 个点的纵坐标
h_i	表示每种分类方法中第 i 类商家中心的横坐标
H_j	表示第 j 辆车的横坐标
r_i	商家需要向客户发货的重量
R_i	货车的最大装载限度
d_i	第 i 个车在城内所行驶的距离
D_{ij}	第 i 个商家到第 j 个中转站距离
a	中转站发派车辆总数

注：其余符号详见文中说明。

四、模型的建立与求解

4.1 问题一：为运送任务制定最佳行车路线和调度策略

4.1.1 问题分析

需要物流公司完成 10 个商业区域的货车运送任务，拟制定最佳的行车路线和货车调度策略，使得所有运货车辆在城内的运送工作时间总和最小。因城市入城时间受限制，只能在夜间 24 点至凌晨 4 点入城，因此对 68 个商家所处位置进行预处理，采用 k-means 聚类方法进行分类，且至少分为四类才可满足时间的限制。然后我们求出每个商家出城的最短距离，规定了适应度，对划分的每一类用遗传算法求出最短距离，货车在城内运送时间最少以及货车行驶路线。

4.1.2 数据预处理

（1）对 68 个商家进行聚类

k-means 聚类是将数据集中在某些方面相似的数据成员进行分类组织的过程，基本方法是基于最小误差平方和准则通过迭代运算找出 k 个类别的划分方案。
准则函数：

$$SSE=\sum_{j=1}^{k}\sum_{i=1}^{n_j}\left\|x_i-z_j\right\|^2\left(x_i\in s_j\right)$$

k 为最终划分类类别；s_j：第 j 个聚类集；z_j：聚类中心；x_i：第 j 个聚类集 s_j 中包含的样本点。

聚类准则：

聚类中心 z_j 的选择应使 SSE 误差平方和极小，则

$$\frac{\partial SSE_i}{\partial z_j}=0,$$

解得 s_j 类的聚类中心应选为该类样本的均值。

算法步骤：

Step1：给定大小为 n 的数据集，先随机选取 k 个对象作为初始的聚类中心。

Step2：计算每个聚类的均值，求出每个样本数据对象与聚合中心的距离，并根据最小距离重新划分对象。

Step3：聚类中心以及分配给它们的对象就代表一个聚类。一旦全部对象都被分配了，每个聚类的聚类中心会根据聚类中现有的对象被重新计算。

Step4：这个过程将不断重复直到满足没有（或最小数目）聚类中心再发生变化，满足误差平方和局部最小。

采用 k-means 聚类方法对商区内 68 个商家进行聚类，采用以上算法步骤编写程序，使得各类样本的总体误差最小，计算样本与聚类中心的距离求出最好分类方法是四类，商家分类的分布情况如下图所示：

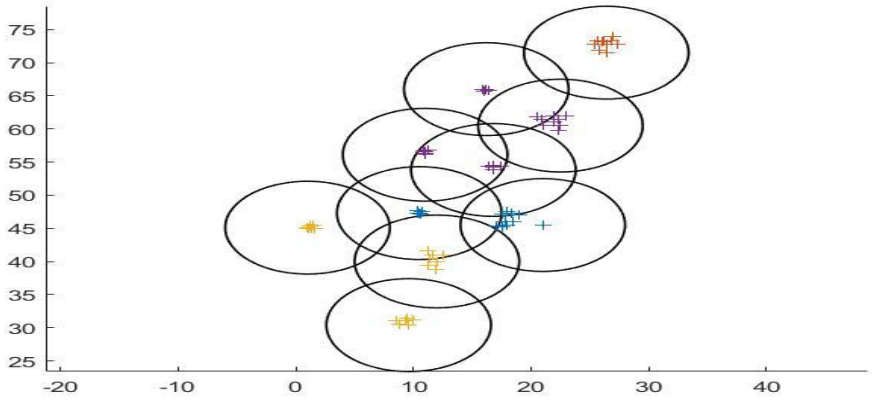

图 1：聚类方法分类效果图

（2）求出每个商家的最短出城距离

城区按各商业中心商场为圆心，7km 为半径的总体外包络为界，对出城最短距离的求解，商家出以圆包络为界的城区，简单来看，本商区的点从本商区出城，最短距离是圆心与该点的连线较短的路，但 10 个商业区之间存在相互重叠的地方，从本城区出来可能进入另一个城区，因此分成两种情况讨论。

①本商区出城点不经过其他商区。

根据圆的特点，将商家与本商区的中心商场连线，商区和圆的边界较短的路为最短路，当直线向弧两边延伸时，显然距离是递增的，因此沿半径方向较短路为最短距离。

②本商区出城点经过其他商区。

当第一种情况的出城点经过其他商区时，货车继续行驶会进入其他城区，如果从其它商区出城，距离会大大增加，因此从本商区出城距离才有最小值，经分析，从最近的出城点向两边搜索交点，知道找到从本商区出城的交点，求出交点距离进行比较，找到最短距离。

4.1.3 最佳行车路线模型的建立

遗传算法：通过群体搜索技术，根据适者生存的原则逐代进化，最终得到最优解。应用遗传算法求解需要完成四个主要步骤：

1. 确定表示方案
2. 确定适应值度量
3. 确定控制算法的参数和变量
4. 确定指定结果的方法和停止运行的准则

（1）决策变量的介绍：

$I_{(i,j)}$：第 i 辆车的路程上第 j 个点的最短入城距离；

$O_{(i,j)}$：第 i 辆车的路程上第 j 个点的最短出城距离；

x_{ij}：第 i 辆车第 j 个点的横坐标；

y_{ij}：第 i 辆车第 j 个点的纵坐标。

（2）目标函数：

适应度(F(x))=货车入城最短距离+经过每类商区的最短路程+货车出城最短距离；

$$F(x)=\frac{1}{\sum_{i=1}^{m} I_{(i,1)}+O_{(i,n_i)}+\sum_{j=1}^{n_{i-1}}\sqrt{(x_{ij}-x_{ij+1})^2+(y_{ij}-y_{ij+1})^2}}$$

将转弯及等红绿灯计入在内，假设货车城区内外的为平均车速 25km/h，每个收货点收货装载平均大约 10 分钟，68 个收获点大约停留 680 分钟，在城区内形式的最短时间为目标函数，如下：

$$\min=\frac{\sum_{i=1}^{m} I_{(i,1)}+O_{(i,n_i)}+\sum_{j=1}^{n_{i-1}}\sqrt{(x_{ij}-x_{ij+1})^2+(y_{ij}-y_{ij+1})^2}}{25}\times 60+680$$

（3）约束条件：

当派相同的车时，不同的商家位置的横坐标不同：

$$x_{ij}\neq x_{sl}, j\neq l$$

否则，商家的横坐标相同：

$$x_{ij}=x_{sl}, i=s, j=l$$

综上所述，模型总结为：

$$\min=\frac{\sum_{i=1}^{m} I_{(i,1)}+O_{(i,n_i)}+\sum_{j=1}^{n_{i-1}}\sqrt{(x_{ij}-x_{ij+1})^2+(y_{ij}-y_{ij+1})^2}}{25}\times 60+680$$

$$s.t.\begin{cases} x_{ij}\neq x_{sl}, j\neq l \\ x_{ij}=x_{sl}, i=s, j=l \end{cases}$$

（4）算法设计：

我们采用 k-means 聚类和遗传算法相结合的方法计算行车路线，运送所需要的最短时间。对于用车调度，行驶路线设计了流程图,其中对遗传算法的具体步骤做了详细的说明，主流程图如下所示：

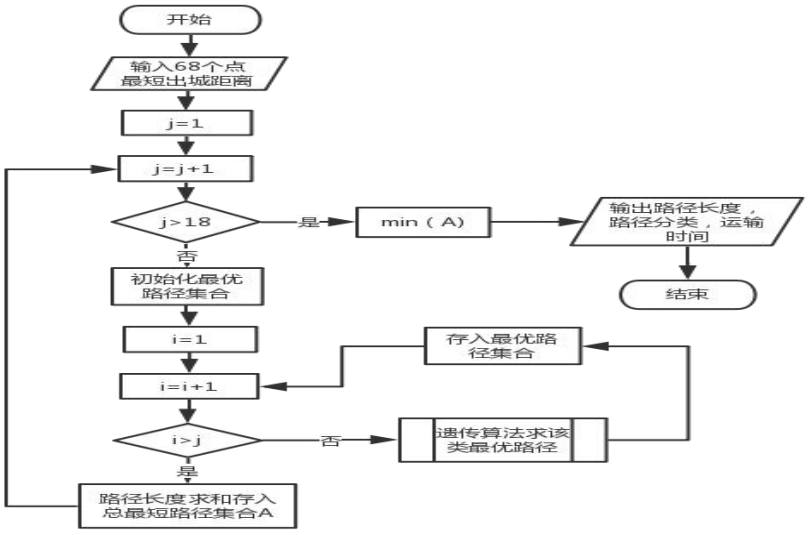

图 2：问题一的流程图算法

遗传算法求该类最优路径的算法设计为：

Step1：编码与初始化。我们将 68 个商区进行编号从 A0101-A1006，利用 randperm 函数产生不重复的随机自然数，我们直接使用数字编码表示路径的染色体。因解题方法是使用的是计算每一对点，则我们编码时将第一个节点单独放入，合并成完整编码。

Step2：计算适应度，因取最短路径值，采用 1/f(x)的方法，于是，可进一步计算相对适应度。

Step3：用轮盘选择法，产生随机值，顺序累计适应度的关系，选出优良个体进入下一代。

Step4：交叉与变异。染色体编码为从 1 到 num(该类的商家总数)的无重复编码，若采用传统的交叉方法，即上一行与下一行对位交叉，会产生无效路径，所以此处采用交换变异法。即随机产生两个数，交换两个节点的顺序，具体步骤如下：

（1）在表示路径的染色体 1 和 2 中，随机选取两个基因座（起点不能为基因座）i 和 j，将第 i，j 个基因座和之间的各个基因座定义为交叉域，并将交叉的内容分别记为 t1 和 t2。

（2）根据交叉区域中的映射关系，在个体 1 中找出所有与 t2 相同的元素，在个体 2 中找出所有与 t1 相同的元素，全部置为 0。

（3）将个体 1、2 进行循环左移，删除 0 元素，直到编码串中交叉区域的左端不再有 0；然后将所有空位集中到交叉区域，把交叉区域内原有的基因依次向后移动。因零元素可能较多，在程序实现时，我们将非零元素提出，然后再合成。

（4）将 t2 插入到个体 1 的交叉区域，t1 插入到个体 2 的交叉区域，形成新的染色体。

Step5：最后，调试种群数量、遗传代数、交叉概率等等，以达到较好的结果。

4.1.4 模型的求解

表 1：最佳行驶路线和货车调度方案

货车调度策略	最佳的行车路线	线路运行最少时间（单位：分钟）	线路运行最短路程 （单位：km）
B7	46-52-53-55-54-51-50-47-48-49-42-43-41-44-45	71.0771	29.6155

B1	3-1-2-5-6-7-8-4-9-10	39.0813	16.2839
B7	67-64-63-66-68-65-57-58-56-60-61-59-62-38-37-36-35-39-40	96.9017	40.3757
B1	29-28-26-27-25-32-33-31-30-34-18-17-19-16-22-20-23-21-24-15-13-11-12-14	106.0948	44.2062

我们将行车路线分为四段，货车派发了中转站 B1,B7 各两辆，各段的行车时间分别为 71.0771，39.0813，96.9017，106.0948 分钟，所有运货车辆在城内的总运送最短总路程为 130.4813km，所有运货车辆在城内的总运送工作的最短时间为 993.1549 分钟，即 16.56 小时，得到的最短时间结果较优。

各中转站发派的货车四种行驶路线如下：

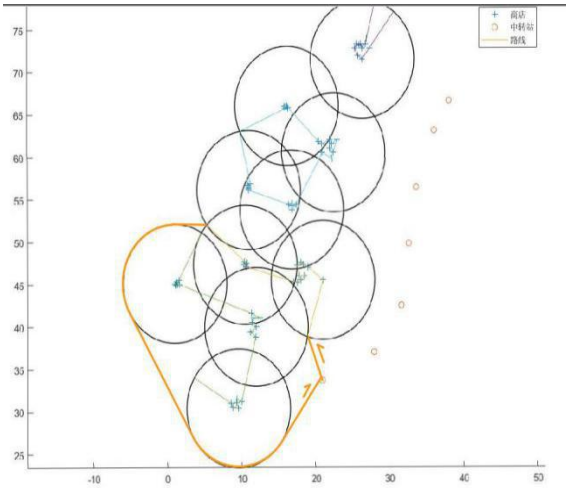

图 3：中转站 B7 货车的行车路线一

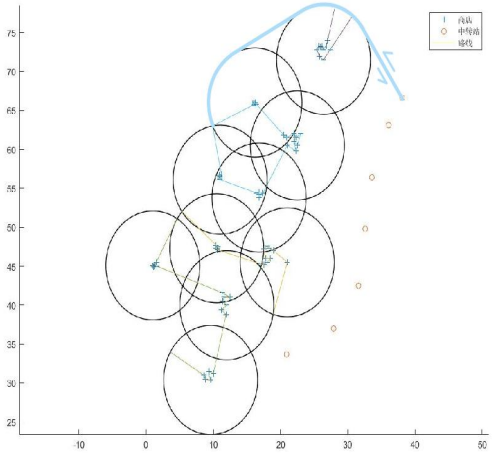

图 4：中转站 B1 货车的行驶路线二

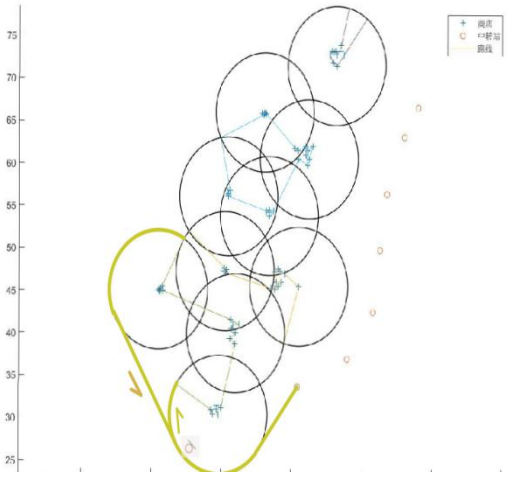

图 5：中转站 B7 货车的行车路线三

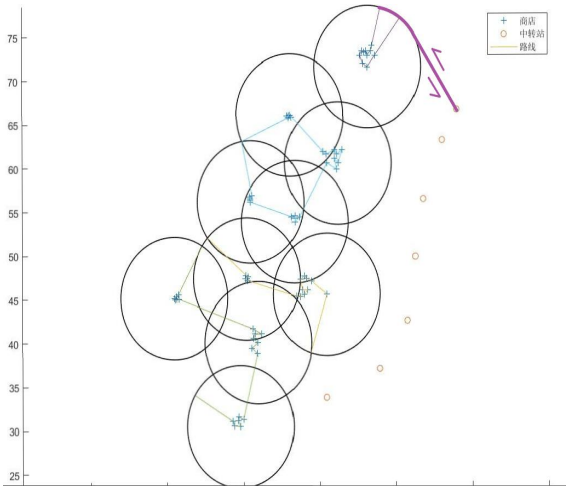

图 6：中转站 B1 货车的行驶路线四

由上面四个图，可以直观的知道各线路发派中转站的货车进城地点、出城地点以及在城外的行车轨迹，较清晰的描述了每类商家的最短行驶路线，对商场进行分类不仅仅化简了问题，而且求出来解的效果也比较好。

4.1.5 模型的灵敏度分析

考虑到现实情况中对商铺坐标的定位存在偏差，现对每个商铺的横纵坐标进行在±

5%的范围内随机仿真结果进行灵敏度分析，得到的完成任务总时间的预测结果和未改变坐标对应总时间的真实值比较，对稳定性进行分析：

表 2：数据调整±5%的结果

	横坐标之和	纵坐标之和	总时间
真实值	1059.9	3600.6	933.1547
随机结果	1062	3576.9	984.7722
百分比	99.8%	99.34%	93.96%

此表为所有坐标的变化与坐标的真实值的预测结果以及总时间真实值的对比，当横纵坐标变化±5%时，对完成任务总时间误差不超过 7%，说明该模型比较稳定性。

4.2 问题二：考虑装载与成本，为运送任务制定最佳方案

4.2.1 问题的分析

考虑装载容量及运输成本，完成商家按客户的订单需发商品数量，为运送制定最佳的行车路线和货车调度策略，使满足总的运费和车辆在城内的总运送工作时间最少。用 k 均值方法对 68 个商家进行分类，要使货车不超过装载容量，则至少分为 10 类，至多分为 18 类，因此问题得到了简化。我们对 18 辆车重新进行编号，对每一类用 lingo 算出中转站发派车辆到聚类商家中心行车方案，然后，选出 10-18 类中每一类最优的行驶路线，对每一类沿用问题一遗传算法，修改适应度为花费函数，求出花费最少需要货车的辆数，在最后的 17 各种种群中，求取在城内时间最少的每个种群的行驶路径以及花费清单。

4.2.2 商家分类处理

继续沿用问题一的 k-means 聚类方法，重新划分商家类别，根据货车装运的限制用此方法求出最少需要 10 辆车，因此对求出 10-18 类在城内的距离之和最小的分类方法。分类情况如下：

表 3：k-means 算法分为 10 类时商家分布情况

聚类类别	68 个商家所在类别											
1	46	47	48	49	50	51	52	53	54	55	0	0
2	1	2	3	0	0	0	0	0	0	0	0	0
3	63	64	65	66	67	68	0	0	0	0	0	0
4	35	36	37	38	39	40	0	0	0	0	0	0
5	25	26	27	28	29	0	0	0	0	0	0	0
6	16	17	18	19	20	21	22	23	24	0	0	0
7	4	5	6	7	8	9	10	0	0	0	0	0
8	30	31	32	33	34	0	0	0	0	0	0	0
9	11	12	13	14	15	0	0	0	0	0	0	0
10	41	42	43	44	45	56	57	58	59	60	61	62

注：此为 10 类时商家分布情况，11-18 类商家分布见附录 1。

4.2.3 模型的建立与求解

4.2.3.1 货车到每类商家路程最短的发车方案

（1）决策变量

h_i, g_i：分别表示每种分类方法中，第 i 类商家中心的横纵坐标；H_j, G_j：分别表示

第 j 辆车的横纵坐标；r_i：商家需要向客户发货的重量；R_i：货车的最大装载限度；

$$p_{ij}=\begin{cases}0,\text{第}j\text{辆车不向第}i\text{个商家中心发车}\\1,\text{第}j\text{辆车向第}i\text{个商家中心发车}\end{cases}$$

（2）目标函数

要求运货花费最少，先求出中转战发派的车辆到聚类出商家中心最短距离，目标函数如下：

$$\min=\sum_{j=1}^{18}\sum_{i=1}^{n}p_{ij}\times\sqrt{\left(h_i-H_j\right)^2+\left(g_i-G_j\right)^2},n=10,\cdots,18$$

（3）约束条件

根据聚类的方法，分出 n 类商家，使每类商家都有且只有一辆车进行运货：

$$\sum_{j=1}^{18}p_{ij}=1,i=1,\cdots,n$$

当中转站发车路线有两条或两条以上时，赋货车的发车轨迹只有一条，

即当$\sum_{i=1}^{n}p_{ij}\geq 2$时，

$$\sum_{i=1}^{n}p_{ij}=1,\ j=1,\cdots,18$$

每类商家货品重量之和不超过发派的货车的装载限制：

$$p_{ij}\times r_i\leq R_j,i=1,\cdots,n,j=1,\cdots,18$$

综上所述，装载模型的总结为：

$$\min=\sum_{j=1}^{18}\sum_{i=1}^{n}p_{ij}\times\sqrt{\left(h_i-H_j\right)^2+\left(g_i-G_j\right)^2},n=10,\cdots,18$$

$$s.t.\begin{cases}\sum_{j=1}^{18}p_{ij}=1,i=1,\cdots,n\\\sum_{i=1}^{n}p_{ij}=1,\ j=1,\cdots,18,\sum_{i=1}^{n}p_{ij}\geq 2\\p_{ij}\times r_i\leq R_j,i=1,\cdots,n,j=1,\cdots,18\\p_{ij}=0,1\end{cases}$$

（4）得出结果

根据聚类得出的聚类中心，求出 10-18 类的各中转站发车最佳策略，12-18 车的发车方案见附录 2，10，11 类的发车方案如下：

表 4：10 类的发车方案

商家中心	1	2	3	4	5	6	7	8	9	10
发车编号	15	2	4	17	5	13	12	14	3	16

表 5：11 类的发车方案

商家中心	1	2	3	4	5	6	7	8	9	10	11
发车编号	3	7	15	16	6	2	4	13	14	12	5

由以上两表可知，每个商家中心都派发一辆货车，Ⅰ、Ⅱ种类型的货车使用情况比

较均匀，能够较好地完成运输任务。

4.2.3.2 遗传算法求行程最短时间

（1）决策变量

n_j：出动的第 j 辆车的标号；b_i：第 i 辆车出发的中转站；num_i：第 i 辆车的路径上的商家个数；d_i：第 i 个车在城内所行驶的距离；w_{ij}：第 i 辆车第 j 步车内的剩余容量；D_{ij}：第 i 个商家到第 j 个中转站距离；a：中转站发派车辆总数；部分决策变量与问题二的 lingo 模型中相同，此处不再重复说明。

（2）目标函数

改进的适应度=

$$\frac{1}{\left(\text{正常商品总量}\times 0.03+\text{超出商品量}\times 0.02\right)\times\left(\text{车在城内行驶的路程}+\text{车城外较短路程}\right)}$$

所有运货车辆总的运费最少：

$$\min f1=\sum_{i=1}^{a}\sum_{s=1}^{m}\left\{\left(r_{(t_{(i,s)},1)}\times 0.03+r_{(t_{(i,s)},2)}\times 0.02\right)\times\left(\sum_{j=s}^{num_{(i-1)}}D_{(t_{(i,j)},t_{(i,j+1)})}+D_{(t_{(i,num_i)},b_i)}\right)\right\}$$

所有运货车辆在城内的运送工作时间总和最少：

$$\min f2=\sum_{i=1}^{18}d_i\times 60\div 25+680$$

其中，$r_{(i,1)}$表示第 i 个商场正常商品的需求量；$r(i,2)$表示第 i 个商场超出商品的需求量；$t_{(i.j)}$表示第 i 辆车第 j 步到达的位置。

（3）约束条件

第 i 辆车行驶路程上商家个数的约束：

$$num_i=\sum_{s=1}^{68}p_{si},1\le i\le a$$

商家只来一辆货车：

$$\sum_{j=1}^{a}p_{ij}=1,1\le i\le 68$$

货车行驶的路径不重复：

$$t_{(i,j)}\ne t_{(k,l)},l\ne j$$

每个商家运送商品量要小于火车剩余装载量，使每个商场达到运输需求：

$$r_{(t_{(i,j)},1)}+r_{(t_{(i,j)},2)}\le w_{ij},1\le i\le a,1\le j\le num_i$$

对出动的 a 辆车走遍 68 个商家：

$$\sum_{i=1}^{a}num_i=68,1\le i\le a$$

派出车辆的不重复：

$$n_i\ne n_j,i\ne j$$

综上所述，双目标优化的模型总结为：

$$\min f1=\sum_{i=1}^{a}\sum_{s=1}^{m}\left\{\left(r_{(t_{(i,s)},1)}\times 0.03+r_{(t_{(i,s)},2)}\times 0.02\right)\times\left(\sum_{j=s}^{num_{(i-1)}}D_{(t_{(i,j)},t_{(i,j+1)})}+D_{(t_{(i,num_i)},b_i)}\right)\right\}$$

$$\min f2=\sum_{i=1}^{18}d_i\times 60\div 25+680$$

$$s.t.\begin{cases}num_i=\sum_{s=1}^{68}p_{si},1\le i\le a\\ \sum_{j=1}^{a}p_{ij}=1,1\le i\le 68\\ t_{(i,j)}\ne t_{(k,l)},l\ne j\\ r_{(t_{(i,j)},1)}+r_{(t_{(i,j)},2)}\le w_{ij},1\le i\le a,1\le j\le num_i\\ \sum_{i=1}^{a}num_i=68,1\le i\le a\\ n_i\ne n_j,i\ne j\end{cases}$$

（4）算法设计

Step1：对 68 个商家采用 k-means 方法分类，首先根据车辆的运载量和数量，得出至少分为 10 类、最多 18 类，算出 10-18 类（即 10-18 辆车）商家的所在各类别的分布情况。

Step2：对已有的 18 辆车按运载量的大小依次进行编号。假如分为 10 类，对每一类的商区分配最佳的车辆调度情况，对车辆是否派送引入 0-1 变量，对每一类商区只分配一辆车和每辆车只规划一条路线作为约束条件，用 lingo 求解最优货车调度方案。其他类别的计算重复此步骤。

Step3：对分为 j 类的第 i 小类，运用问题一的遗传算法，规定运费适应度，以花费最少的目标函数，求出运费最少的分类类别，保留优秀种群，输出货车调动情况。

Step4：在优秀种群中，以在城内运送最短时间为目标，计算每一小类的路程最少的行驶路径，并更最优运费，保留其路径，输出优秀种群中城内距离最小的路径，行车路线并计算总运送时间。

（4）结果分析

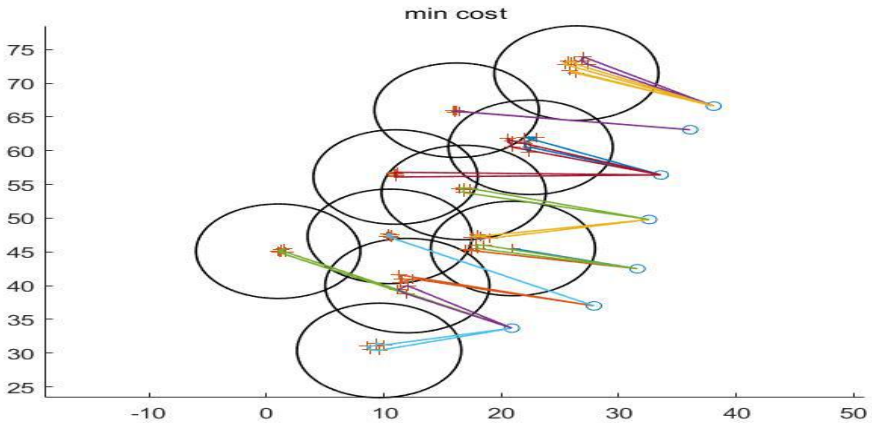

图 7:最小花费时车辆运输方案

由上图可以非常清楚的知道分类的大致情况，每一个中转站都派发出相应的车辆，各类的商家分布比较集中，且比较均匀。

表 6：寻求花费最小的方案

城内行驶总时间(单位:小时）	费用（单位：元）	车辆调度数量（单位：辆）
26.9736	2937.62	17

利用遗传算法，以花费最少为目标时，最优的方案为车辆使用 17 辆，最少费用为 2937.62 元，在城内行驶总时间为 26.98 小时。我们对保留的优秀种群做进一步筛选，求出运送最少时间（即求最短距离）的方案。

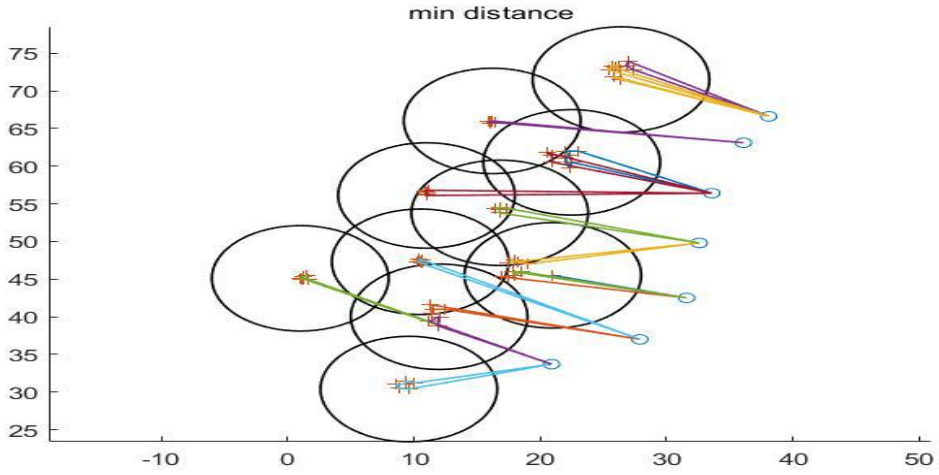

图 8：最小距离时货车调度方案

上图表示 17 辆车基于最小距离模型，绘制出的运输方案图，较直观的展现了个商场的分类情况，此图只是大致的说明一下分步情况，具体的行驶路线还需采用遗传算法。

与遗传算法产生的第一个方案相比此方案在车辆配送费用，配送目标商家没有改变，对车辆路径上商家装载的顺序发生了局部变动，产生结果如下：

表 7：在城内行驶距离最小距离时每辆车的路径

行驶路线	城内时间	装载重量（kg）	行驶路线	城内时间	装载重量（kg）
(1)52-55-53-54	89.38	227	(10)30-31-32-33-34	87.03	747
(2)10-9-3	67.04	260	(11)45-44-43-41-42	129.25	413
(3)37-36-35-39-40-38	152.95	626	(12)18-21-24	76.19	207
(4)63-64-67-66-68-65	99.68	583	(13)46	43.6	401
(5)25-26-27-28-29	134.24	366	(14)48-49	71.98	101
(6)16-17-19-20-23-22	103.71	620	(15)5-6-8-7-4	92.73	510
(7)59-61-60-62	96.41	223	(16)57-56-58	68.24	298
(8)2-1	57.03	424	(17)50-47-51	80.41	207
(9)11-14-13-12-15	111.56	569			

由上表展示了每辆车的行驶路线和装载的重量，第一辆车的行驶路线可知为 52-55-53-54，其他车辆的行驶路线同理。我们还求解了每辆车整段路运行的时间，总且城内总时间为 1561.5 分钟。

表 8：在城内的行驶最短时间、最少运费值

运费（单位：元）	在城内的行驶总路程	在城内的行驶总时间（单位：小时）

3033.0121	367.2710	26.025

Matlab 编程求出 17 辆车，在城内的最短总距离为 367.2710km，在城内的行驶最短时间为 26.025 小时，最少运费为 3033.0121 元，得出的结果较好。

4.2.4 模型灵敏度分析

为了验证模型的稳定性，现对每个商家的坐标进行在±5%的范围内随机仿真，得到以下结果：

表 9：城内运行时间最小模型数据调整结果

	未超重货物和	超重货物之和	运费	城内运行距离
真实值	4875	1907	3033.0	367.2710
随机结果	4892.9	1898.3	3022.9	366.4680
百分比	99.63%	99.54%	99.67%	99.78%

此表为所有商铺货物的变化与真实值的预测结果以及运费和距离真实值的对比，当货物变化±5%时，对完成任务总时间误差不超过 1%，说明该模型比较稳定，结果真实可靠。

4.3 问题三：增加购买量时，为完成运输任务制定新的调度策略

4.3.1 问题的分析

在问题二的基础上，购买量增加为原来的 5 倍，给出需要准备两种类型货车的数量，货车分配到各个中转站的方案以及最佳行车路线。因问题二中结果需派出 17 辆车，有些车辆只前往了一个点就回中转站，结合实际考虑一辆车派往多个地点的情况，我们求出最少运费模型和最少城内时间模型中 17 条线路的时间以及中转站到达该城区的距离，对两个模型中的时间进行考虑，若一中转站内车跑完自己的目标后有足够的时间前往其他类，便派该车前往该中转站的其他负责区域，且 0.6 吨载重的车不能派往 1 吨车的目标区域，1 吨的车可以前往 0.6 吨车的目标区域，同时考虑四点前入城即可出城时间任意，则前往的地点中所需运行时间最长的最后前往。当运货量扩大 5 倍时，采用运费最小的方案，求出货车数量等相关值。

4.3.2 模型的建立

本问使用问题的结论，建立在问题二模型的基础上，对运送方案进行优化。由于沿用问题二的模型，问题二中遗传算法模型（1）、（2）、（3）式继续加入本模型，作为决策变量，目标函数，约束条件。

4.3.3 算法设计

Step1：根据问题二求出 17 辆车城内运行时和中转站到城区的距离，对模型优化后，我们发现两种模型都只需要 7 辆货车即可完成任务，我们对从中转站发出的车赋初始值。

Step2：若一装车完成自己的任务后，载重大于限制条件时，循环另一车辆使用情况；若小于条件限制时，选取在前一辆货车完成任务的时间加上后一辆车进城所需要的时间小于 4 小时的车辆，将满足条件的车辆路程添加到前一辆车的路径上。

Step3：当分配车辆超过 7 时，输出货车最佳行驶路径，在考虑需求量增加 5 倍，局部最优决定着整体最优，将求出一类最优方案再乘以 5 即为整体最优策略。

4.3.4 结果展示

4.3.4.1 运费最少和时间最少模型的行驶时间的求解

根据问题二，求出运费最少和时间最少模型的行驶时间，派出车辆类别，结果如下：

表 10：运费最少的模型

入城点到中转站运行时间	城内运行时间	中转站到城区距离	派出车的中转站	派出车辆类别
84.51	33.65	13.84	1	1
72.41	35.30	13.64	1	1
110.41	38.26	13.75	1	1
132.38	59.71	18.47	2	1
149.37	81.98	13.16	3	1
128.87	36.02	11.60	3	2
92.13	41.42	11.61	3	1
105.70	43.63	13.16	4	1
113.88	32.54	22.32	4	2
53.24	33.60	9.64	5	1
82.89	50.76	9.69	5	1
92.27	46.66	9.65	5	1
116.25	50.30	14.88	6	1
142.59	75.89	10.57	6	1
168.36	89.40	9.42	7	2
118.69	32.16	11.45	7	1
80.85	32.18	9.37	7	1

对两个模型进行优化，最终发现都只需要 7 辆车即可完成任务，货车在城内运送时间最短模型运行时间值及中转站到商区距离见附录 3。派发七辆车分别为中转站 1-7 中车型Ⅰ、Ⅰ、Ⅱ、Ⅱ、Ⅰ、Ⅰ、Ⅱ。

4.3.4.2 车辆路线的规划

若一中转站内车跑完自己的目标后有足够的时间前往其他类，便派该车前往该中转站的其他负责区域，且 1 吨的车可以前往 0.6 吨车的目标区域，同时考虑四点前入城即可出城时间任意，则前往的地点中所需运行时间最长的最后前往。根据此原则我们对 7 辆车的路线重新规划，分别求出模型一、二的路线安排。模型二的结果见附录 4，模型一的结果如下：

表 11：模型Ⅰ的车辆线路安排

车辆安排	车辆路线	城内运行时间（分钟）（不包括装货）
第一辆	B1-3-9-10-B1-1-2-B1-4-8-7-6-5-B1	107.21
第二辆	B2-15-11-12-14-12-B2	59.71
第三辆	B3-21-24-18-B3-16-17-19-20-22-23-B3-25-26-27-28-29-B3	159.43
第四辆	B4-30-31-32-33-34-B4-52-53-55-54-B4	76.18
第五辆	B5-48-49-B5-46-B5-50-51-47-B5	131.02
第六辆	B6-59-60-61-62-B6-42-41-44-45-43-B6	126.19
第七辆	B7-56-58-57-B7-63-64-67-66-68-65-B7	153.74

-38-37-35-39-40-36-B7

由上表可知，城内总运行时间为 1493.3 分钟（不包括装货时间），各辆车从中转站的出发到各商家再回到该中转站，可清晰的得知行车路线。

当运货量扩大 5 倍时，采用运费最小的方案即模型一，按此路线派 5 辆车一起运行，最终需要 35 辆，运输总费用为 14865 元，城内总运行时间为 7467.5 分钟，即 124.46 小时。中转站货车调度策略如下：

表 12：中转站派车的类型以及数量

中转站	1	2	3	4	5	6	7
车型	1	1	2	2	1	1	2
数量	5	5	5	5	5	5	5

4.3.5 模型的灵敏度分析

问题二考虑到现实情况中对于货物重量的统计存在误差，现对每个商铺的进行在±5%的范围内随机仿真结果进行灵敏度分析，问题三采用问题二灵敏度分析预测的结果用于问题三的灵敏度分析，同时考虑实际情况中可能不是恰好 5 倍，对倍数进行±5%的波动：

表 13：倍数调整±5%的幅度对结果的影响

	I 型车数量	II 型车数量	总运费	城内运行总时间
真实值	20	15	14745	7467.5
随机结果	20	15	14063	7122.2
百分比	100%	100%	95.37%	95.38%

此表为所有商铺货物的变化与真实值的预测结果以及运费和距离真实值的对比，当倍数变化±5%时，对完成任务总时间误差不超过 5%，说明该模型比较稳定。

4.4 问题四：车辆允许跨站运输制定最佳运输方案

4.4.1 问题的分析

若每个中心商场的货物都需要发航空急件，要求直接运送到 B03 和 B05 中转站，如果允许车辆跨站运输，重新考虑第二，三问。问题二、三问直接固定了从中转站到商家中心或回到中转站，本问在中心商场的货物要直接运送到 B03 和 B05 中转站。对于考虑航空急件的问题二，只需重新规划路径的去处中转站即可，即改变原问题二的适应度（1/运费）计算方式，已知所有路径在遗传算法中随机生成，但最后一点到去处中转站的距离需要判断，即

1.若路径中存在中心商场，则在路径最后一个点到 B03 与 B05 之间选择距离最小的去处中转站。

2.若路径中不存在中心商场，则去处中转站与第二题相同，直接套用第二问结论即可，无需判断。

对于考虑航空急件的问题三，则利用考虑航空急件的问题二的结论，采用问题 3 的优化方式；对考虑航空急件的问题二的所发车辆数进行优化，考虑局部最优代替整体最优，将优化后的结果乘以 5 即为最终结果。

4.4.2 模型的建立

(1)符号说明：

本问建立在问题二、三的基础上，允许车辆跨站运送，重新考虑二、三问，我们沿

用二、三的模型，只是对适应度稍作改变，对中心商场运送货物的路线有了新的规划，而对二、三问的目标函数，约束条件，依旧作为问题四的目标函数，约束条件，此处不再重复说明。

（2）算法设计：

Step1：重新规定适应度，适应度=路径上各个商家的需求量*（路径上剩余路程之和+最后一个点到去处中转站的距离）。

Step2：判断路径中存在中心商场，则在路径最后一个点到 B03 与 B05 之间选择距离最小的去处中转站。否则去处中转站与第二题相同，直接套用第二问结论即可，无需判断。

Step3：对考虑航空急件的问题二的所发车辆数进行优化，求出中转站最优发车数，最小花费，行驶路线等。

Step4：重新考虑的问题三，利用考虑航空急件的问题二的结论，采用问题 3 的优化方式，将优化后的结果乘以 5 即为最终结果。

4.4.3 模型的求解

4.4.3.1 考虑跨站运输，重新求解问题二

利用遗传算法，并编写程序判断分类中是否有中心商场，有则将货物直接运送至 B03、B05 距离较近的中转站，其他问题二求解相同，计算结果如下：

表 14：最少运费模型的每类城内时间值

每辆车形式的路径	每类城内时间	运行全部时间
22-20-23-19	32.69	117.49
63	33.60	98.51
50-51-47-48-49	48.89	129.41
36-40-39-35-37-38	115.51	208.81
27-28-29	82.15	140.60
3-9-10	33.65	98.35
13-14-12-11-15	71.90	154.53
57-58-59-62-61-60-56	52.78	186.38
42-43-45-44-41	84.19	160.33
33-32-30-31-34	41.76	139.13
17-16	33.48	80.21
67-64	37.30	82.85
46	33.60	62.88
2-5-6-7-8-4-1	35.30	161.73
54-55-53-52	43.63	118.86
68-66-65	31.92	90.31
21-24-18	40.46	112.54
26-25	82.27	129.45

由上表可知，当 18 辆车全部派发（11 辆Ⅰ类型车，7 辆Ⅱ类型车）时，总花费最少，并求出总运输费用为 3422.3 元，城内行驶总距离：403.5km ，城内总时间：1615.1 分钟。进一步对城内时间最少为目标函数，其时间最少的每类城内时间值见附录 5，其计算出总运输费用为 3444.8 元，货车在城内行驶总距离为 401.8 km，在城内行驶总时间 1611.2 分钟，即 26.86 小时。

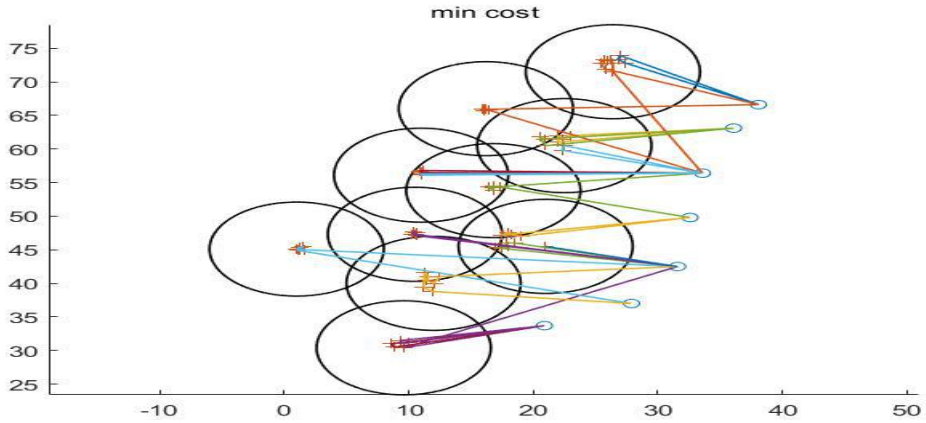

图 9：18 辆车运送路径

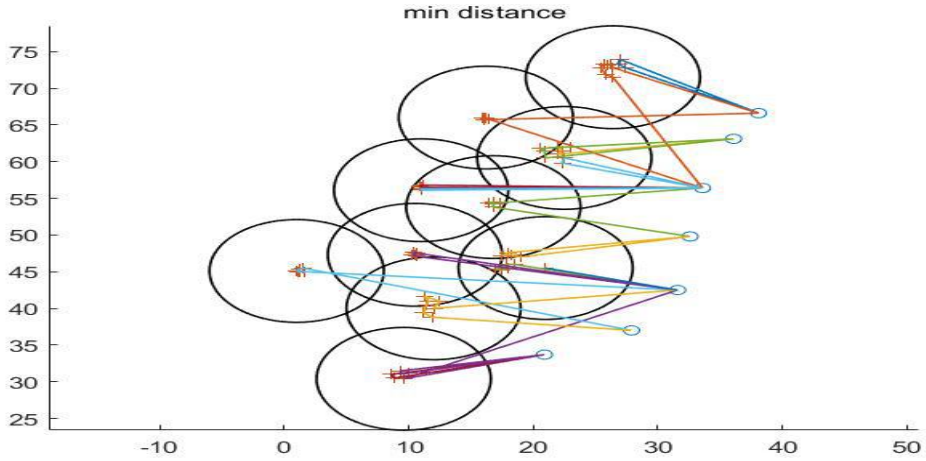

图 10：用最少时间模型得出货车运送轨迹

以上两图分别为以总运输的最小花费，最短时间为目标函数的模型，求出 18 辆车以及每辆车的行驶轨迹，编写程序，画出的轨迹图，非常清晰直观。

4.4.3.2 考虑跨站运输，重新求解问题三

因问题二中结果需派出 18 辆车，有些车辆只前往了一个点就回中转站，结合实际考虑一辆车派往多个地点的情况，我们求出最少运费模型和最少城内时间模型中 18 条线路的时间以及中转站到达该城区的距离，最少城内时间模型的货车调度方案见附录 6，运费最少模型的货车调度方案如下表所示：

表 15：运费最少模型的货车调度方案

入城点到中转站运行时间	城内运行时间（无装货）	中转站到城区距离	派出车的中转站	货车回到中转站名称	派出车辆类别
84.51	33.65	13.84	1	1	1
137.90	71.90	16.63	1	3	1
148.09	35.30	13.64	1	3	2
100.87	32.69	16.62	2	2	1
95.99	40.46	16.55	2	2	1
127.62	82.15	12.98	3	3	1
68.57	33.48	11.64	3	3	1
116.39	82.27	13.06	3	3	1

116.78	41.76	22.35	4	3	2
105.56	43.63	13.30	4	4	1
119.77	48.89	9.65	5	5	1
150.66	84.19	9.67	5	5	1
53.24	33.60	9.64	5	5	1
194.94	115.51	13.87	6	5	2
164.20	52.78	22.18	6	5	1
87.06	33.60	11.45	7	5	1
71.34	37.30	11.50	7	7	1
78.77	31.92	11.54	7	7	1

对两个模型进行优化，最终发现都只需要 8 辆车即可完成任务，8 辆车发别为：Ⅰ类型货车：B01、B02、B03、B06、B07 中转站各一辆，Ⅱ类型火车：B01、B04、B06 中转站各一辆。最少时间模型的行驶路线见附录 7，最少运费模型的行驶路线，车辆调度情况如下：

表 16：车型号即车辆行驶路线

1-8 辆车的行驶路线	城内运行时间（不包括装货）	车型
B1-3-9-10-B1-2-5-6-7-8-4-1-B3	68.95	2
B1-13-14-12-11-15-B3-17-16-B3-27-28-29-B3	187.53	1
B2-22-20-23-19-B2-21-24-18-B2	73.15	1
B3-27-28-29-B3	82.15	1
B4-54-55-53-52-B4-33-32-30-31-34-B3	85.39	2
B6-36-40-39-35-37-38-B5-42-43-45-44-41-B5	199.70	2
B6-57-58-59-62-61-60-56-B5-46-B5-50-51-47-48-49-B5	135.27	1
B7-68-66-65-B7-67-64-B7-63-B5	102.82	1

总城内运行时间 1615.1 分钟，当运货量扩大 5 倍时，采用运费最小的方案即模型一，按此路线派 5 辆车一起运行，局部最优代替整体最优，最终需要货车 40 辆（I 中类型货车 25 辆，II 种类型货车 15 辆），运输总费用为 17112 元，城内总运行时间为 8076 分钟。

4.4.4 模型的灵敏度分析

4.4.4.1 考虑跨站运输问题二的灵敏度分析

考虑到现实情况中对于货物重量的统计存在误差，现对每个商铺的进行在±5%的范围内随机仿真结果进行灵敏度分析，得到的完成任务总时间的预测结果和未改变坐标对应总时间的真实值比较，对稳定性进行分析：

表 17：对于运费最小

	未超重货物和	超重货物之和	运费	城内运行距离
真实值	4875	1907	3422.3	403.5
随机结果	4878.0	1899.0	3420.4	406.1
百分比	99.94%	99.58%	99.94%	99.36%

表 18：对于城内运行时间最小

	未超重货物和	超重货物之和	运费	城内运行距离
真实值	4875	1907	3444.8	401.8
随机结果	4908.3	1897.2	3453.1	401.5
百分比	99.32%	99.49%	99.76%	99.93%

此表为所有商铺货物的变化与真实值的预测结果以及运费和距离真实值的对比，当货物变化±5%时，对完成任务总时间误差不超过 1%，说明该模型具有很强的稳定性。

4.4.4.2 考虑跨站运输问题三的灵敏度分析

采用问题二灵敏度分析预测的结果用于问题三作灵敏度分析，同时考虑实际情况中可能不是恰好 5 倍，对倍数进行±5%的波动：

表 19：倍数波动对相关结果的影响

	I 型车数量	II 型车数量	总运费	城内运行总时间
真实值	25	15	17112	8076
随机结果	25	15	15411	7667.3
百分比	100%	100%	90.06%	94.94%

真实值与随机值对比，对其完成任务总时间误差不超过 5%，存在着一定的误差，但模型还是比较稳定的。

五、模型的评价与推广

5.1 模型的评价

5.1.1 模型的优点

（1）K-means 算法本身具有优化迭代的功能，针对部分小样本可以降低总的聚类时间复杂度。

（2）遗传算法从群体出发，具有潜在的并行性，可以进行多个个体的同时比较，加快求解速度，过程简单，容易与其他算法结合；具有良好的全局搜索能力，可以快速地将解空间中的全体解搜索出，而不会陷入局部最优解的快速下降陷阱。

（3）本文构建的模型较为简单，可以得到相对最优解，在解决实际生活中的需要时可较为快速的提供局部最优方案，在遗传算法中，当迭代数量与种群规模达到一定数量可以得出全局最优解。

5.1.2 模型的缺点

（1）遗传算法的局部搜索能力较差，在单纯的遗传算法比较费时，在进化后期搜索效率较低。采用何种选择方法既要使优良个体得以保留，又要维持群体的多样性，一直是遗传算法中较难解决的问题。

（2）需要预处理数据较多，且在选择需要排除多少辆车时采用了对每种情况进行遍历的方法，导致循环求解速度较慢。

（3）构建的模型只能输出局部最优解，对全局最优解的求法还需进一步的研究。

5.2 模型的推广与改进

5.2.1 模型的改进

（1）对第一问进行模拟退火与遗传混合求解算法，这样既可以提高模型的收敛速度，又不会轻易的落入局部最优解中。

（2）对第二问的两个目标进行去量纲的操作，放入符合实际生产需要的权值，进行智能算法求解，这样可考虑不同侧重下的方案调度问题。
（3）对第三问，第四问建立数学模型采用 lingo 求解，但通常建立的模型十分复杂，在实际操作中又一定的难度，但可求出调度问题的全局最优解。

5.2.2 模型的推广

本问采用 k-means 聚类与遗传算法混合两阶段算法，较好地解决了车辆路径问题，对商家进行分类，使复杂的问题更加简单化，运算更加方便。对于交通运输问题进行规划，有效帮助政府提高城市运输管理水平，完善服务体系，统筹交通资源配置，对促进城市经济发展、优化城市交通资源配置和促进节能减排等具有重要的现实意义。

六、参考文献

[1]郇鹏. 城市垃圾收运系统选址和选线优化研究[D].北京化工大学,2011.
[2]陈燕葵. 基于多车场双层车辆路径问题的城市物流共同配送体系[D].华南理工大学,2018.
[3]缪文清. 两类多起点多终点应急物流运输问题研究[D].沈阳工业大学,2011.
[4]潘新元,崔艳,钟秋平.物流运输中多车型配送车辆的调度模型[J].数学学习与研究,2015(13):127-128.
[5]孙小军,焦建民.一种求解最少时间最小费用路问题的算法[J].计算机工程与科学,2008(07):77-78+89.
[6]Mollaei,Mohammadi,Naderi. A bi-objective MILP model for blocking hybrid flexible flow shop scheduling problem: robust possibilistic programming approach[J]. International Journal of Management Science and Engineering Management,2019,14(2).
[7]孟祥飞,王瑛.基于灰色关联度的多目标规划解法[J].数学的实践与认识,2014,44(08):190-196.
[8]薛声家,林佳丽.关于线性多目标规划一种新方法的注记[J].暨南大学学报(自然科学版),2008(05):457-459.
[9]王彩玲,李忠范,刘庆怀.求解线性多目标规划的一种新方法[J].吉林大学学报(理学版),2005(03):282-286.
[10]彭煜. 基于多目标规划的 DEA 有效性研究[D].西南交通大学,2005.

七、附录

附录 1：问题二聚类为 11-18 类商家分布情况

由于此数据较大，将聚类为 11-18 类商家分布情况放入支撑材料中，见程序-题目二-t11 到 t18。

附录 2：问题二各分类种发车方案情况

12 车的发车方案

12 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12
发车编号	5	13	2	9	8	14	15	7	12	3	6	4

13 车的发车方案

13 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13
发车编号	16	3	9	8	5	2	14	6	4	15	13	12	7

14 车的发车方案

14 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13	14
发车编号	10	12	13	8	6	15	9	17	7	5	3	4	2	11

15 车的发车方案

15 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
发车编号	11	14	9	6	18	7	12	5	1	17	4	10	2	8	15

16 车的发车方案

16 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
发车编号	3	12	9	5	4	11	15	16	17	6	18	14	10	8	1	7

17 车的发车方案

17 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
发车编号	6	12	18	11	4	14	17	1	3	15	9	5	16	8	2	10	7

18 车的发车方案

18 个商家中心	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
发车编号	3	11	7	17	5	2	1	9	8	15	14	18	16	12	6	10	13	4

附录 3：城内时间最短模型运行时间求解

入城点到中转站运行时间	城内运行时间	中转站到城区距离	派出车的中转站	派出车辆类别
84.27	33.65	14.08	1	1
72.41	35.30	13.64	1	1
110.97	38.26	13.69	1	1
132.67	58.92	18.60	2	1
149.37	81.98	13.16	3	1
125.40	33.79	11.60	3	2
91.41	42.58	11.62	3	1
108.44	43.63	13.16	4	1
113.88	32.54	22.32	4	2
53.24	33.60	9.64	5	1
82.89	50.76	9.69	5	1
93.04	47.42	9.65	5	1
116.85	50.30	14.88	6	1
142.77	76.30	10.44	6	1
166.21	89.11	9.44	7	2
118.69	32.16	11.45	7	1
82.97	32.95	9.44	7	1

附录 4：模型 II

	车辆路线	城内运行时间（不包括装货）
第一辆	B1-10-9-3-B1-2-1-B1-5-6-8-7-4-B1	107.21
第二辆	B2-11-14-13-12-15-B2	58.92
第三辆	B3-18-21-24-B3-16-17-19-20-23-22-B3-25-26-27-28-29-B3	158.35
第四辆	B4-30-31-32-33-34-B4-52-55-53-54-B4	76.18
第五辆	B5-48-49-B5-46-B5-50-47-51-B5	131.78
第六辆	B6-59-61-60-62-B6-45-44-43-41-42-B6	126.60
第七辆	B7-57-56-58-B7-63-64-67-68-66-65-B7-37-36-35-39-40-38-B7	154.22

附录 5：问题四最少时间模型的每类城内时间值

每辆车形式的路径	每类城内时间	运行全部时间
22-23-20-19	32.69	116.16
63	33.60	98.51
47-49-48-51-50	46.66	130.77
40-39-35-36-37-38	115.63	205.49
27-28-29	82.15	140.60
3-9-10	33.65	98.35
14-13-12-11-15	71.52	154.20
57-58-62-61-60-59-56	52.78	188.03
44-45-43-41-42	83.86	158.91

30-32-33-31-34	41.35	138.21
17-16	33.48	80.21
67-64	37.30	82.85
46	33.60	62.88
2-5-6-7-8-4-1	35.30	161.73
55-54-53-52	42.19	119.36
68-66-65	31.92	90.31
18-21-24	41.28	111.57
26-25	82.27	129.45

附录 6：城内时间最短模型的货车调度方案

入城点到中转站运行时间	城内运行时间（无装货）	中转站到城区距离	派出车的中转站	货车回到中转站名称	派出车辆类别
84.51	33.65	13.84	1	1	1
137.40	71.52	16.80	1	3	1
148.09	35.30	13.64	1	3	2
99.54	32.69	16.62	2	2	1
95.12	41.28	16.45	2	2	1
127.62	82.15	12.98	3	3	1
68.57	33.48	11.64	3	3	1
116.39	82.27	13.06	3	3	1
115.89	41.35	22.32	4	3	2
105.96	42.19	13.40	4	4	1
121.11	46.66	9.66	5	5	1
149.27	83.86	9.65	5	5	1
53.24	33.60	9.64	5	5	1
192.23	115.63	13.26	6	5	2
165.85	52.78	22.18	6	5	1
87.06	33.60	11.45	7	5	1
71.34	37.30	11.50	7	7	1
78.77	31.92	11.54	7	7	1

附录 7：1-8 辆车的行驶路线

1-8 辆车的行驶路线	车型	城内运行时间（不包括装货）
B1-3-9-10-B1-1-2-4-5-6-7-B3	2	68.95
B1-11-12-13-14-15-B3-16-17-B3-27-28-29-B3	1	187.15
B2-19-20-22-23-B2-18-21-24-B2	1	73.97
B3-27-28-29-B3	1	82.15
B4-52-53-54-55-B4-30-31-32-33-34-B3	2	83.54
B6-35-36-37-38-39-40-B5-41-42-43-44-45-B5	2	199.49
B6-56-57-58-59-60-61-62-B5-46-B5-47-48-49-50-51-B5	1	133.04
B7-65-66-68-B7-64-67-B7-63-B5	1	102.82

附录 8：问题一轨迹模型所用 matlab 程序

```
%运行主程序
%此程序运行时间较长 约 10 分钟
clc,clear
% n-- 种群规模% ger-- 迭代次数% pc--- 交叉概率% pm-- 变异概率
% v-- 初始种群（规模为 n）% myfit-- 适应度向量
%num-- 需要跑的城市的数目(v 的列数)
load zuobiao.mat
load short_distance_outcity.mat
load chu_site.mat
%n=100;
ger=500;
pc=0.1;
pm=0.05;
%num
k=1;
min_zh=0;
for i=1:18
    step=i%记录运行到哪一步
    n=ceil(1000/i);
    leibie(:,k)=kmeans(zuobiao,i);
    k=k+1;
    zh=0;
    yuan_v=zeros(18,68);
    max_v=zeros(18,68);
    for j=1:i
        clear id distance
        [id,~]=find(leibie(:,k-1)==j);
        distance=count_distance(id);

[max_myfit(j),max_v(j,[1:length(id)])]=ga_zdl(n,ger,pc,pm,length(id),distan
ce,short_distance_outcity(id,1));
        %
zh=zh+sum_distance(id(max_v(j,[1:length(id)]),1),count_distance([1:68]'),sh
ort_distance_outcity(:));
        yuan_v(j,[1:length(id)])=id(max_v(j,[1:length(id)]),1)';%路径标号
    end
    zh=sum(1./max_myfit);
    num(i)=zh;
    if i==1
        global_max_v=yuan_v;
        min_zh=zh;
    else
        if zh<min_zh
```

```
            min_zh=zh;
            global_max_v=yuan_v;
        end
    end
end
min_zh*60/25+680%总时间

%%绘制图像

yx=[26.4    71.5;16.2    66;22.5    60.5;11    56.1;16.8    53.8;1    45.1;10.5
47.3;21  45.5;12  40;9.6  30.4];
B=[38.1    66.6;36.1  63.1;33.6  56.4;32.6  49.8;31.6  42.5;27.9  37;20.9
   33.7];
for i=1:10%绘制圆
    circle(yx(i,1),yx(i,2),7);
    hold on
end
plot(zuobiao(:,1),zuobiao(:,2),'+');legend();
scatter(B(:,1),B(:,2));
route=global_max_v;
route(all(route==0,2),:)=[];route(:,all(route==0,1))=[];%清除 0 行 0 列
[m,~]=size(route);
for i=1:m
    t=[];T=[];
    t=route(i,:);t(find(t==0))=[];
    T=[chu_site(t(1),:);zuobiao(t,:);chu_site(t(end),:)];
    plot(T(:,1),T(:,2),'linewidth',1);
end
legend('商店','中转站','路线');
```

附录 9：问题一灵敏度分析所用 matlab 程序

```
%用于灵敏度分析
clear
clc
load zuobiao.mat
load short_distance_outcity.mat
load chu_site.mat
r_zuobiao=zuobiao;
for i=1:68
    for j=1:2
        t=rand();%随机正负
        if t>0.5
            r_zuobiao(i,j)=zuobiao(i,j)+zuobiao(i,j)*rand*0.05;%5%以内波动
        else
            r_zuobiao(i,j)=zuobiao(i,j)-zuobiao(i,j)*rand*0.05;
```

```
        end
    end
end
x=sum(zuobiao(:,1));rx=sum(r_zuobiao(:,1));
y=sum(zuobiao(:,2));ry=sum(r_zuobiao(:,2));
Px=min(x,rx)/max(x,rx)
Py=min(y,ry)/max(y,ry)
% n-- 种群规模% ger-- 迭代次数% pc--- 交叉概率% pm-- 变异概率
% v-- 初始种群（规模为 n）% myfit-- 适应度向量
%num-- 需要跑的城市的数目(v 的列数)


%n=100;
ger=500;
pc=0.1;
pm=0.05;
%num
k=1;
min_zh=0;
for i=4:4%文中取了第四类最小
    n=ceil(1000/i);
    leibie(:,k)=kmeans(r_zuobiao,i);
    k=k+1;
    zh=0;
    yuan_v=zeros(18,68);
    max_v=zeros(18,68);
    for j=1:i
        clear id distance
        [id,~]=find(leibie(:,k-1)==j);
        distance=count_distance_linmindu(id,r_zuobiao);

[max_myfit(j),max_v(j,[1:length(id)])]=ga_zdl(n,ger,pc,pm,length(id),distan
ce,short_distance_outcity(id,1));
        %
zh=zh+sum_distance(id(max_v(j,[1:length(id)]),1),count_distance([1:68]'),sh
ort_distance_outcity(:));
        yuan_v(j,[1:length(id)])=id(max_v(j,[1:length(id)]),1)';%路径标号
    end
    zh=sum(1./max_myfit);
    num(i)=zh;
    global_max_v=yuan_v;
    min_zh=zh;
end
a=(min_zh*60/25)+680;%本次运行的最短时间
```

```
b=max(a,993.1547);
c=min(a,933.1547);
p=c/b%计算可信度
```

附录 10：问题二求解最小花费所用 matlab 程序

```
%此主函数目标为最小花费
clc,clear
% n-- 种群规模% ger-- 迭代次数% pc--- 交叉概率% pm-- 变异概率
% v-- 初始种群（规模为 n）% myfit-- 适应度向量
%num-- 需要跑的城市的数目(v 的列数)
load A_A_distance.mat
load A_B_distance.mat
load require.mat
load site.mat
global distance_io distance_in require
clc
%n=200;
ger=500;
pc=0.1;
pm=0.05;
%num
k=1;
for i=10:18
    step=i-9%计算运行到哪了
    n=ceil(1000/i);
    k=k+1;
    max_v=zeros(18,68);
    clear id
    if i==10
        load t10.mat
    end
    if i==11
        load t11.mat
    end
    if i==12
        load t12.mat
    end
    if i==13
        load t13.mat
    end
    if i==14
        load t14.mat
    end
    if i==15
        load t15.mat
```

```
    end
    if i==16
        load t16.mat
    end
    if i==17
        load t17.mat
    end
    if i==18
        load t18.mat
    end
    if i~=10
        id(:,1)=[];
    end
    for j=1:i
        clear id1
        id1=id(j,:);
        id1(find(id1==0))=[];

[max_myfit(j),max_v(j,[1:length(id1)])]=ga_zdl(n,ger,pc,pm,id1,site(i-9,j))
;
    end
    zh=sum(1./max_myfit);%总花费;
    if i==10
        global_max_v=max_v;
        min_zh=zh;
    else
        if zh<min_zh
            min_zh=zh;
            global_max_v=max_v;
        end
    end
end
global_max_v(all(global_max_v==0,2),:)=[];global_max_v(:,all(global_max_v==
0,1))=[];%清楚零行零列
changdu=[4 3 6 6 5 6 4 2 5 5 5 3 1 2 5 3 3];
zhf=0;
load distance_io_incity
for i=1:17

zhf=zhf+count_distance_in_gai(global_max_v(i,[1:changdu(i)]),site(17-9,i),d
istance_io_incity);
end
time=zhf*60/25+680;
%%绘制图像
```

```
huitu(global_max_v)
title('min cost')
```
附录 11：最小花费的灵敏度分析所用 matlab 程序
```
%用于灵敏度分析
%对于商品需求进行浮动变化
clc,clear
% n-- 种群规模% ger-- 迭代次数% pc--- 交叉概率% pm-- 变异概率
% v-- 初始种群（规模为 n）% myfit-- 适应度向量
%num-- 需要跑的城市的数目(v 的列数)
load A_A_distance.mat
load A_B_distance.mat
load require.mat
load site.mat
global distance_io distance_in r_require
clc
for i=1:68
    for j=1:2
        t=rand();%随机正负
        if t>0.5
            r_require(i,j)=require(i,j)+require(i,j)*rand*0.05;%5%以内波动
        else
            r_require(i,j)=require(i,j)-require(i,j)*rand*0.05;
        end
    end
end
%n=200;
ger=500;
pc=0.1;
pm=0.05;
%num
k=1;
for i=10:18
    step=i-9%确立程序运行到哪一步
    n=ceil(1000/i);
    k=k+1;
    max_v=zeros(18,68);
    clear id
    if i==10
        load t10.mat
    end
    if i==11
        load t11.mat
    end
    if i==12
```

```
        load t12.mat
    end
    if i==13
        load t13.mat
    end
    if i==14
        load t14.mat
    end
    if i==15
        load t15.mat
    end
    if i==16
        load t16.mat
    end
    if i==17
        load t17.mat
    end
    if i==18
        load t18.mat
    end
    if i~=10
        id(:,1)=[];
    end
    for j=1:i
        clear id1
        id1=id(j,:);
        id1(find(id1==0))=[];

[max_myfit(j),max_v(j,[1:length(id1)])]=ga_zdl_linmindu(n,ger,pc,pm,id1,sit
e(i-9,j));
    end
    zh=sum(1./max_myfit);%总花费;
    if i==10
        global_max_v=max_v;
        min_zh=zh;
    else
        if zh<min_zh
            min_zh=zh;
            global_max_v=max_v;
        end
    end
end
changdu=[4 3 6 6 5 6 4 2 5 5 5 3 1 2 5 3 3];
zhf=0;
```

```
load distance_io_incity
for i=1:17

zhf=zhf+count_distance_in_gai(global_max_v(i,[1:changdu(i)]),site(17-9,i),d
istance_io_incity)
end
sum(r_require(:,1))
sum(r_require(:,2))
```

附录 12：问题二求最短城内距离所用 matlab 程序

```
%% 此函数目标为在花费尽可能小的情况下找最短城内距离
clc,clear
load t17.mat
load site.mat
load A_A_distance.mat
load A_B_distance.mat
load require.mat
load distance_io_incity
global distance_io distance_in require
clc
id(:,1)=[];
i=17;
n=200;
ger=500;
pc=0.1;
pm=0.05;
best_d=zeros(1,17);
best_v=zeros(17,6);
%%
j=3;
clear id1
id1=id(3,:);
id1(find(id1==0))=[];
[v3,fit3]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(3,[1:length(v3(1,:))])]=count_distance_in_gai(v3,site(i-9,
j),distance_io_incity);
money(j)=count_fit(best_v(3,[1:length(v3(1,:))]),site(i-9,j));
%%
j=4;
clear id1
id1=id(4,:);
id1(find(id1==0))=[];
[v4,fit4]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v4(1,:))])]=count_distance_in_gai(v4,site(i-9,
j),distance_io_incity);
```

```
money(j)=count_fit(best_v(j,[1:length(v4(1,:))]),site(i-9,j));
%%
j=5;
clear id1
id1=id(5,:);
id1(find(id1==0))=[];
[v5,fit5]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v5(1,:))])]=count_distance_in_gai(v5,site(i-9,
j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v5(1,:))]),site(i-9,j));
%%
j=6;
clear id1
id1=id(6,:);
id1(find(id1==0))=[];
[v6,fit6]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v6(1,:))])]=count_distance_in_gai(v6,site(i-9,
j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v6(1,:))]),site(i-9,j));
%%
j=9;
clear id1
id1=id(9,:);
id1(find(id1==0))=[];
[v9,fit9]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v9(1,:))])]=count_distance_in_gai(v9,site(i-9,
j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v9(1,:))]),site(i-9,j));
%%
j=10;
clear id1
id1=id(10,:);
id1(find(id1==0))=[];
[v10,fit10]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v10(1,:))])]=count_distance_in_gai(v10,site(i
-9,j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v10(1,:))]),site(i-9,j));
%%
j=11;
clear id1
id1=id(11,:);
id1(find(id1==0))=[];
[v11,fit11]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v11(1,:))])]=count_distance_in_gai(v11,site(i
```

```
-9,j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v11(1,:))]),site(i-9,j));
%%
j=15;
clear id1
id1=id(15,:);
id1(find(id1==0))=[];
[v15,fit15]=ga_zdl_print(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v15(1,:))])]=count_distance_in_gai(v15,site(i
-9,j),distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v15(1,:))]),site(i-9,j));
%%
j=1;
v1=[52,55,53,54];
[best_d(j),best_v(j,[1:length(v1)])]=count_distance_in_gai(v1,site(i-9,j),d
istance_io_incity);
money(j)=count_fit(v1,site(i-9,j));
%%
j=2;
v2=[10,9,3];
[best_d(j),best_v(j,[1:length(v2)])]=count_distance_in_gai(v2,site(i-9,j),d
istance_io_incity);
money(j)=count_fit(v2,site(i-9,j));
%%
j=7;
v7=[59 61 60 62];
[best_d(j),best_v(j,[1:length(v7)])]=count_distance_in_gai(v7,site(i-9,j),d
istance_io_incity);
money(j)=count_fit(v7,site(i-9,j));
%%
j=8;
v8=[2 1];
[best_d(j),best_v(j,[1:length(v8)])]=count_distance_in_gai(v8,site(i-9,j),d
istance_io_incity);
money(j)=count_fit(v8,site(i-9,j));
%%
j=12;
v12=[18 21 24];
[best_d(j),best_v(j,[1:length(v12)])]=count_distance_in_gai(v12,site(i-9,j),
distance_io_incity);
money(j)=count_fit(v12,site(i-9,j));
%%
j=13;
v13=[46];
```

```
[best_d(j),best_v(j,[1:length(v13)])]=count_distance_in_gai(v13,site(i-9,j),
distance_io_incity);
money(j)=count_fit(v13,site(i-9,j));
%%
j=14;
v14=[48 49];
[best_d(j),best_v(j,[1:length(v14)])]=count_distance_in_gai(v14,site(i-9,j),
distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v14)]),site(i-9,j));
%%
j=16;
v16=[57 56 58];
[best_d(j),best_v(j,[1:length(v16)])]=count_distance_in_gai(v16,site(i-9,j),
distance_io_incity);
money(j)=count_fit(best_v(j,[1:length(v16)]),site(i-9,j));
%%
j=17;
v17=[50 47 51];
[best_d(j),best_v(j,[1:length(v17)])]=count_distance_in_gai(v17,site(i-9,j),
distance_io_incity);
money(j)=count_fit(v17,site(i-9,j));
%%
daan1=sum(best_d)%城内总行驶路程
daan2=sum(money)%花费的总费用
daan3=best_v%对应的路径
%% 绘制图像
huitu(daan3)
title('min distance')
%%
```

附录 13：最小路程灵敏度分析所用 matlab 程序

```
clc,clear
load t17.mat
load site.mat
load A_A_distance.mat
load A_B_distance.mat
load require.mat
load distance_io_incity
global distance_io distance_in r_require
for i=1:68
    for j=1:2
        t=rand();%随机正负
        if t>0.5
            r_require(i,j)=require(i,j)+require(i,j)*rand*0.05;%5%以内波动
        else
```

```
            r_require(i,j)=require(i,j)-require(i,j)*rand*0.05;
        end
    end
end
id(:,1)=[];
i=17;
n=200;
ger=500;
pc=0.1;
pm=0.05;
best_d=zeros(1,17);
best_v=zeros(17,6);
%%
j=3;
clear id1
id1=id(3,:);
id1(find(id1==0))=[];
[v3,fit3]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(3,[1:length(v3(1,:))])]=count_distance_in_gai(v3,site(i-9,
j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(3,[1:length(v3(1,:))]),site(i-9,j));
%%
j=4;
clear id1
id1=id(4,:);
id1(find(id1==0))=[];
[v4,fit4]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v4(1,:))])]=count_distance_in_gai(v4,site(i-9,
j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v4(1,:))]),site(i-9,j));
%%
j=5;
clear id1
id1=id(5,:);
id1(find(id1==0))=[];
[v5,fit5]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v5(1,:))])]=count_distance_in_gai(v5,site(i-9,
j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v5(1,:))]),site(i-9,j));
%%
j=6;
clear id1
id1=id(6,:);
id1(find(id1==0))=[];
```

```
[v6,fit6]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v6(1,:))])]=count_distance_in_gai(v6,site(i-9,
j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v6(1,:))]),site(i-9,j));
%%
j=9;
clear id1
id1=id(9,:);
id1(find(id1==0))=[];
[v9,fit9]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v9(1,:))])]=count_distance_in_gai(v9,site(i-9,
j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v9(1,:))]),site(i-9,j));
%%
j=10;
clear id1
id1=id(10,:);
id1(find(id1==0))=[];
[v10,fit10]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v10(1,:))])]=count_distance_in_gai(v10,site(i
-9,j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v10(1,:))]),site(i-9,j));
%%
j=11;
clear id1
id1=id(11,:);
id1(find(id1==0))=[];
[v11,fit11]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v11(1,:))])]=count_distance_in_gai(v11,site(i
-9,j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v11(1,:))]),site(i-9,j));
%%
j=15;
clear id1
id1=id(15,:);
id1(find(id1==0))=[];
[v15,fit15]=ga_zdl_print_linmindu(n,ger,pc,pm,id1,site(i-9,j));
[best_d(j),best_v(j,[1:length(v15(1,:))])]=count_distance_in_gai(v15,site(i
-9,j),distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v15(1,:))]),site(i-9,j));
%%
j=1;
v1=[52,55,53,54];
[best_d(j),best_v(j,[1:length(v1)])]=count_distance_in_gai(v1,site(i-9,j),d
```

```
istance_io_incity);
money(j)=count_fit_linmindu(v1,site(i-9,j));
%%
j=2;
v2=[10,9,3];
[best_d(j),best_v(j,[1:length(v2)])]=count_distance_in_gai(v2,site(i-9,j),d
istance_io_incity);
money(j)=count_fit_linmindu(v2,site(i-9,j));
%%
j=7;
v7=[59 61 60 62];
[best_d(j),best_v(j,[1:length(v7)])]=count_distance_in_gai(v7,site(i-9,j),d
istance_io_incity);
money(j)=count_fit_linmindu(v7,site(i-9,j));
%%
j=8;
v8=[2 1];
[best_d(j),best_v(j,[1:length(v8)])]=count_distance_in_gai(v8,site(i-9,j),d
istance_io_incity);
money(j)=count_fit_linmindu(v8,site(i-9,j));
%%
j=12;
v12=[18 21 24];
[best_d(j),best_v(j,[1:length(v12)])]=count_distance_in_gai(v12,site(i-9,j),
distance_io_incity);
money(j)=count_fit_linmindu(v12,site(i-9,j));
%%
j=13;
v13=[46];
[best_d(j),best_v(j,[1:length(v13)])]=count_distance_in_gai(v13,site(i-9,j),
distance_io_incity);
money(j)=count_fit_linmindu(v13,site(i-9,j));
%%
j=14;
v14=[48 49];
[best_d(j),best_v(j,[1:length(v14)])]=count_distance_in_gai(v14,site(i-9,j),
distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v14)]),site(i-9,j));
%%
j=16;
v16=[57 56 58];
[best_d(j),best_v(j,[1:length(v16)])]=count_distance_in_gai(v16,site(i-9,j),
distance_io_incity);
money(j)=count_fit_linmindu(best_v(j,[1:length(v16)]),site(i-9,j));
```

```
%%
j=17;
v17=[50 47 51];
[best_d(j),best_v(j,[1:length(v17)])]=count_distance_in_gai(v17,site(i-9,j),
distance_io_incity);
money(j)=count_fit_linmindu(v17,site(i-9,j));
%%
daan1=sum(best_d)%城内总行驶路程
daan2=sum(money)%花费的总费用
daan3=best_v%对应的路径
sum(r_require(:,1))
sum(r_require(:,2))
```

附录 14：问题三主程序所用 matlab 程序

```
clear
clc
load A_A_distance.mat
load A_B_distance.mat
load distance_io_incity.mat
load require.mat
load moxin_1_d.mat
%load moxin_2_d.mat%对于模型 II 的求解 求解时解除%并对 moxin_1_d 加上%
d1=[4,1,7,7,3,3,6,1,2,4,6,3,5,5,1,7,5];%中转站标号
for i=1:17
    t=d(i,:);
    t(find(t==0))=[];
    money(i,1)=count_fit(t,d1(i));
end%求出花费
beishu=5;
require=sum(require,2);
for i=1:17%求出载货量和派出货车种类
    t=d(i,:);
    t(find(t==0))=[];
    need(i,1)=sum(require(t));
    if need(i)>600
        lei(i)=2;
    else
        lei(i)=1;
    end
end
dist=zeros(17,1);
for i=1:17%求解运行第一趟运行的距离
    t=d(i,:);
    t(find(t==0))=[];
    dist(i)=dist(i)+distance_io_incity(t(1),d1(i));
```

```
    dist(i)=dist(i)+distance_io(t(end),d1(i));
    a=length(t);
    if a~=1
        for j=2:a
            dist(i)=dist(i)+distance_in(t(j-1),t(j));
        end
    end
end
Dist=zeros(17,1);
for i=1:17%求解城内运行距离
    t=d(i,:);
    t(find(t==0))=[];
    Dist(i)=Dist(i)+distance_io_incity(t(1),d1(i));
    Dist(i)=Dist(i)+distance_io_incity(t(end),d1(i));
    a=length(t);
    if a~=1
        for j=2:a
            dist(i)=dist(i)+distance_in(t(j-1),t(j));
        end
    end
end
for i=1:17%求解中转站到该类城区的距离
    t=d(i,:);
    t(find(t==0))=[];
    dist1(i,1)=distance_io(t(1),d1(i))-distance_io_incity(t(1),d1(i));
end
dist=[dist,Dist,dist1];
time=dist*60/25;%将距离转化为时间
for i=1:17
    t=d(i,:);
    t(find(t==0))=[];
    a=length(t);
    time(i,1)=time(i,1)+10*a;
end
D=[time,d1',lei']%输出三列时间 出发的中转站 需要的车型
%第一列为进城区到回到中转站的时间 第三列为中转站到城区距离 即第一列+第三列=
整个线路时间
%第二列为城区内运行时间
for i=1:7
    k(i,1)=sum(D(find(D(:,4)==i),2));
end
shijian=(sum(D(:,2))+680)*beishu%运行一趟城内时间
yunfei=sum(money)*beishu%运行一次总花费
```

附录 15：问题四新问题二最少费用所用 matlab 程序

```
%模型一 求解最少费用情况
clc,clear
% n-- 种群规模% ger-- 迭代次数% pc--- 交叉概率% pm-- 变异概率
% v-- 初始种群（规模为 n）% myfit-- 适应度向量
%num-- 需要跑的城市的数目(v 的列数)
load A_A_distance.mat
load A_B_distance.mat
load require.mat
load site.mat
global distance_io distance_in require
%n=200;
ger=500;
pc=0.1;
pm=0.05;
%num
k=1;
for i=10:18
    step=i-9%看输出到哪一步了
    n=ceil(1000/i);
    k=k+1;
    max_v=zeros(18,68);
    clear id
    if i==10
        load t10.mat
    end
    if i==11
        load t11.mat
    end
    if i==12
        load t12.mat
    end
    if i==13
        load t13.mat
    end
    if i==14
        load t14.mat
    end
    if i==15
        load t15.mat
    end
    if i==16
        load t16.mat
    end
```

```
    if i==17
        load t17.mat
    end
    if i==18
        load t18.mat
    end
    if i~=10
        id(:,1)=[];
    end
    for j=1:i
        clear id1
        id1=id(j,:);
        id1(find(id1==0))=[];

[max_myfit(j),max_v(j,[1:length(id1)])]=ga_zdl(n,ger,pc,pm,id1,site(i-9,j))
;
    end
    zh=sum(1./max_myfit);%总花费;
    if i==10
        global_max_v=max_v;
        min_zh=zh;
    else
        if zh<min_zh
            min_zh=zh;
            global_max_v=max_v;
        end
    end
end
global_max_v(all(global_max_v==0,2),:)=[];global_max_v(:,all(global_max_v==
0,1))=[];%清楚零行零列
zhf=0;%在城内的路程
load distance_io_incity
for i=1:18
    clear v
    v(i,:)=global_max_v(i,:);
    v(find(v==0))=[];
    zhf=zhf+count_distance_in_gai(v(1,:),site(end,i),distance_io_incity);
end
%%绘制图像
huitu(global_max_v)
title('min cost')
```

附录 16：问题四新问题三最少时间所用 matlab 程序

```
clear
clc
```

```
load A_A_distance.mat
load A_B_distance.mat
load distance_io_incity.mat
load require.mat
load moxin_1_d.mat
%load moxin_2_d.mat%对于模型 II 的求解 求解时解除%并对 moxin_1_d 加上%
d1=[2,2;7,5;5,5;6,5;3,3;1,1;1,3;6,5;5,5;4,3;3,3;7,7;5,5;1,3;4,4;7,7;2,2;3,3]
;%中转站标号
for i=1:18
    t=d(i,:);
    t(find(t==0))=[];
    money(i,1)=count_fit(t,d1(i,1));
end%求出花费
beishu=5;
require=sum(require,2);
for i=1:18%求出载货量和派出货车种类
    t=d(i,:);
    t(find(t==0))=[];
    need(i,1)=sum(require(t));
    if need(i)>600
        lei(i)=2;
    else
        lei(i)=1;
    end
end
dist=zeros(18,1);
for i=1:18%求解运行第一趟运行的距离
    t=d(i,:);
    t(find(t==0))=[];
    dist(i)=dist(i)+distance_io_incity(t(1),d1(i,1));
    dist(i)=dist(i)+distance_io(t(end),d1(i,2));
    a=length(t);
    if a~=1
        for j=2:a
            dist(i)=dist(i)+distance_in(t(j-1),t(j));
        end
    end
end
Dist=zeros(18,1);
for i=1:18%求解城内运行距离
    t=d(i,:);
    t(find(t==0))=[];
    Dist(i)=Dist(i)+distance_io_incity(t(1),d1(i,1));
```

```
    Dist(i)=Dist(i)+distance_io_incity(t(end),d1(i,2));
    a=length(t);
    if a~=1
        for j=2:a
            dist(i)=dist(i)+distance_in(t(j-1),t(j));
        end
    end
end
for i=1:18%求解中转站到该类城区的距离
    t=d(i,:);
    t(find(t==0))=[];
    dist1(i,1)=distance_io(t(1),d1(i,1))-distance_io_incity(t(1),d1(i,1));
end
dist=[dist,Dist,dist1];
time=dist*60/25;%将距离转化为时间
for i=1:18
    t=d(i,:);
    t(find(t==0))=[];
    a=length(t);
    time(i,1)=time(i,1)+10*a;
end
D=[time,d1,lei']%输出三列时间 出发的中转站 到达的中转站 需要的车型
%第一列为进城区到回到中转站的时间 第三列为中转站到城区距离 即第一列+第三列=
整个线路时间
%第二列为城区内运行时间
for i=1:8
    k(i,1)=sum(D(find(D(:,4)==i),2));
end
shijian=(sum(D(:,2))+680)*beishu%运行总城内时间
huafei=sum(money)*beishu%运行总花费
```

此为本题的主要程序，其完整程序见支撑材料。