题 目： 基于聚类、贪心、模拟退火的分拣系统优化问题研究

关键词：分拣系统优化 最佳货品摆放模型 规划模型 聚类算法 模拟退火算法

摘 要：

货物分拣环节是配送中心的工作流程中影响工作效率的关键因素，分拣效率在很大程度上直接影响了整体物流供应链的供应效率。本文以分拣系统的优化问题作为研究对象，从分批策略、货品摆放位置以及分拣任务指派三个方面入手，建立对应的最优策略模型，研究如何提高公司分拣效率并设计出优化方案。

对于问题一，将订单按照邻近度分批，要求批次尽量少，建立最优分批模型，首先对比基于欧几里得距离聚类算法和余弦相似性聚类算法的分批策略对所分批次优化结果的优劣，之后选择采用基于余弦相似性求邻近度的凝聚型层次聚类算法的分批策略，采用 MATLAB 软件，计算当货架数量 N=200 时的最少分批批次为 38 批。对结果做了合理分析，所得结果很好地符合实际情况。

对于问题二，要求对问题一所分的各批次进行对不同货品种类摆放位置的优化，使得在拣选同一订单时移动距离最短。即在问题一的基础上建立最优货品摆放模型。该模型为针对某批订单中全部订单的拣选距离总和最小的单目标优化模型，用二维矩阵表示各订单与货物货架位置对应关系，首先将在各批订单中出现频率高的货品种类对应货架逐列交换至矩阵中心，简化了下一步采用贪心算法对订单处理的复杂度。通过贪心算法，求局部最优解，逐列判断、交换货品摆放的位置，直到没有更优的解。最终计算得到所有批次订单拣选距离总和为 89852。

对于问题三，在前两问的基础上，将各批订单分配给 n 名分拣工，要求尽快完成任务，且运动距离尽可能均衡。考虑到多旅行商问题（MTSP），建立以分拣用时最短为目标的规划模型，采用模拟退火算法，通过等概率地使用交换法、移位法、倒置法生成新分配方案的形式进行不断迭代，所得总距离随迭代次数的增加而逐渐减小达到优化的目的。在考虑均衡各分拣工运动距离的问题时，采用了惩罚函数算法，使得结果向均衡分配的方向靠近。

最后对模型进行了合理性分析、误差分析、优缺点评价以及模型的推广。

一、 问题重述

1.1 问题背景

近年来，商品的供应订单呈现出碎片化、零散化的趋势。越来越多的企业倾向于通过一个集订单处理、仓储管理、拣货配送于一体的物流配送中心来提高整体物流供应链的供应效率。而配送中心的工作流程中，分拣环节操作复杂，耗时较长，其效率是影响配送中心整体性能的关键因素，是制约整个物流配送流程效率的关键因素。当业务量加大时，如果每天全部订单的货品种类大于货架数量，在不增加货架的情况下，需要通过对拣货分拣系统进行优化，以提高分拣效率、缩短处理时间与降低物流成本。

分拣环节，首先统计汇总出当天全部待配送订单所包含的所有货品及相应数量。然后将这些货品由仓库转运至分拣处，并以一个货架对应一种货品的形式放置到货架上。本题不考虑货架容量，即假设每个货架放置的货品数量没有限制。最后，按照订单信息，每一个订单包含的货品被依次分拣。

1.2 问题一

基于给定的订单信息，设计分批算法，分析订单的分批问题，要求所分批次尽量少，且单批次订单所含货物种类小于 N，并根据算法，给出货架数量 N=200 时的分批方案，列出每批次的订单数量及货物总类数。

1.2 问题二

订单的拣选距离定义为该订单中货品所在货架序号的最大值与最小值之差，在问题一的分批基础上，建模确定货品摆放货架的编号，使得对于给定批次所有订单分拣任务的拣选距离最小，并计算出各批次拣选距离总和以及所有批次全部订单的拣选距离总和。

1.3 问题三

订单分批后，对于每一批次的订单分配给指定个数的分拣工完成分拣任务，基于对分拣取货过程以及货架等的理想化假设，将分拣效率（分拣完成任务单上所有订单所需时间）与每个分拣工的运动距离按正比关系对应。要求设计订单指派算法，将每一批订单合理分配给 n 位分拣工，目的是使完成一批订单分拣所需的总时间最短。并针对题目所给数据以及前两问的分批结果，计算当 n=5 时，各批次订单的分配结果以及每位分拣工处理订单的顺序。

二、 问题分析

2.1 问题一的分析

对于问题一，为了研究批次最少的分批方案，首先要考虑将特性相似的订单分在一起，也就是将所含货品种类相似的订单分在同一批，这时就要对所谓的相似度进行分析，找到定义邻近度[1]（相似度）的方法。

定义邻近度的算法，有基于欧氏距离的聚类算法和基于余弦相似性的聚类算法两种，先随机生成订单，分别将上述两种算法与先到先分批策略和不分批策略进行

比较，找到更优的分批策略，再将其应用于问题一所给数据，分析流程图如图 1。

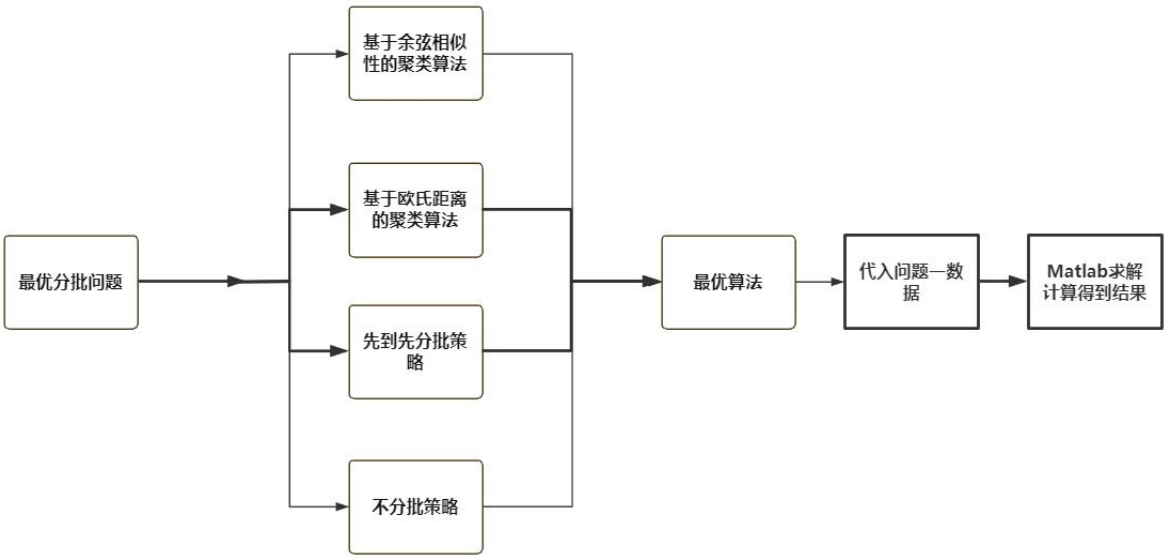

图 1 问题一流程图

2.2 问题二的分析

对于问题二，要研究总拣选距离最小的货品摆放方案，可以使用通过局部最优解逐步推出全局最优解的贪心算法。考虑将每一个订单的拣选距离减到最小，但根据单个订单对货品摆放进行调整时，会对总体的拣选距离产生影响。则考虑根据单个订单进行调整时，要始终保证调整后会使得总的拣选距离更小。

基于这种思路的贪心算法，可以考虑到通过减小单个订单的拣选距离来减小总拣选距离的全部可能，同时能够找出所有可能中使得总的拣选距离最小的解，因此能够通过局部最优解得到全局最优解。

2.3 问题三的分析

对于问题三，将给定的一批分拣任务分配给 n 位分拣工，要求耗时最短，即要尽量减少订单转换时的无效距离所消耗的时间。考虑到多旅行商问题（MTSP），每个分拣工从 1 号货架出发，分别前往各个订单，定义各订单间的距离，对各分拣工的订单排序时建立线性规划模型，据此采用模拟退火算法完成各分拣工的订单分配以及求解所用时间最少而且运动距离尽可能均衡的最优解。

三、 模型假设

1.假设每个货架能够无限制放置货品，不考虑货架的承载能力对货品放置的限制。

2.假设分拣工分拣时可将订单包含的某一种货品全部搬运。

3.假设货架等距排列在一条直线上，移动距离仅考虑货架序号最大值与最小值之差。

4.假设分拣过程理想化，分拣工之间互不干扰，分拣工做匀速运动，只要经过

该订单中某一货品所在货架至少一次，就视为已经拣选到该货品，忽略各种零碎时间，即用运动总距离便可表示一个拣货工完成所有订单所需时间。

5.对于问题三，假设不考虑货架几何尺寸，相邻两个货架间的距离均为单位 1。

四、 符号说明

符号	含义
s_{ij}	邻近度系数
$X_i(i=1,2,\cdots,n)$	订单编号
$V_i=\{v_{i1},v_{i2},\ldots,v_{im}\}$	订单X_i的特征向量
v_{ik}，v_{jk}	i、j 个订单所对应的特征向量之中第 k 维的特征值
$\mathrm{d}(\mathrm{O_k})$	第 k 个订单的拣选距离
p	每批订单的订单数量
c_{ij}	第 i 个订单到第 j 个订单间的距离
$z_k(k=1,2,\ldots,n)$	各分配工走过的总距离

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 基于凝聚型层次聚类算法的最优分批模型

通过对问题的分析，对订单进行分批，要求在每批订单所含货品种类总数不超过 N 的条件下使所分批次 T 最少。

采用凝聚型层次聚类，通过一定的邻近度计算，将所有订单分成不同的批次。

1）邻近度的计算通过特征向量来实现，利用订单中包含的货物种类来生成每个订单的特征向量，根据单个订单中是否包含该货品，定义总的货品种数目为 m，每种货物索引为$I_j(j=1,2,\cdots,m)$，Vi 为订单$X_i(i=1,2,\cdots,n)$的特征向量，则定义：

$$V_i=\{v_{i1},v_{i2},\ldots,v_{im}\}$$

其中：

$$v_{ij}=\begin{cases}1, & \mathrm{I}_j\in X_i\\ 0, & \mathrm{I}_j\notin X_i\end{cases}$$

求得通过欧几里得距离任意两订单的邻近度系数：

$$\frac{1}{s_{ij}}=\sqrt{\sum_{k=1}^{m}\left(v_{ik}-v_{jk}\right)^2}$$

其中，v_{ik}，v_{jk}分别表示第 i、j 个订单所对应的特征向量之中第 k 维的特征值。s_{ij}越大则两订单的邻近度越高。

通过余弦相似性求得任意两订单的邻近度系数：

$$s_{ij}=\frac{V_i\cdot V_j}{||V_i||V_j||}=\frac{\sum_{k=1}^{m}v_{ik}\times v_{jk}}{\sqrt{\sum_{k=1}^{m}(v_{ik})^2}\times\sqrt{\sum_{k=1}^{m}\left(v_{jk}\right)^2}}$$

其中，v_{ik}，v_{jk}分别表示第 i、j 个订单所对应的特征向量之中第 k 维的特征值。s_{ij}越接近 1，则两订单的邻近度越高。

2）基于聚类算法建立订单最优分批模型：

优化目标函数为：

$$min\,T$$

T 为所分批次。

约束条件：

$$\sum_{j=1}^{T}x_{ij}=1,i=1,2,\ldots,n$$

表示每个订单在分批后只允许在其中一个批次出现，既不能分割订单，也不能重复分配订单。

$$x_{ij}=\begin{cases}1,\ x<0\\0,\ x\geq 0\end{cases}$$

x_{ij}表示订单X_i的分批结果，若该订单分配到第 j 批中，取值为 1；若不存在，则取值为 0。

根据每个订单的特称向量通过欧几里得距离或余弦相似性计算两两之间的s_{ij}，将s_{ij}倒序排列，若存在相同的系数，则按照两个订单共同品项和总品项的数量进行排序。

选取s_{ij}最大的两个订单合并为同一订单，判断新订单是否满足约束条件，若不满足约束条件，选择下一个s_{ij}，直到满足所设定的约束条件，计算新订单的特征向

量。

如果再合并时，新订单包含的货品种类超过上限 N，则将合并前的该批订单输出，并重复对剩余订单进行聚类，直至所有订单均被分批。

5.1.2 分批算法的研究

在设计分批方案时，首先对各种分批策略进行研究，随机生成 900 个订单数据，总的货品种类定为 2000 个，每次实验分别抽取 100，200，…，900 个订单，分别通过基于欧氏距离的聚类算法、基于余弦相似性的聚类算法、先到先分批算法用 Matlab 进行分批计算，对比不同分批策略对所分批次结果的影响。所得结果如表 1 所示。

	100	200	300	400	500	600	700	800	900
基于欧几里得距离的聚类算法	14	28	41	50	58	60	66	67	67
基于余弦相似性的聚类算法	11	21	30	35	36	37	37	38	39
先到先分批算法	13	25	38	48	57	66	76	86	95

表 1　基于不同算法的分拣批次（单位：次）

不分批策略下，批次数就等于订单数，由此可得出基于不同算法的分批次数与不分批次数之比，如图 2。可以看出当订单数量较大时，先到先分批策略的优化效果较稳定，而两种聚类算法均对所分批次数有了较大优化效果。其中，采用余弦相似性计算邻近度的聚类算法在拣选效率的优化程度最高。

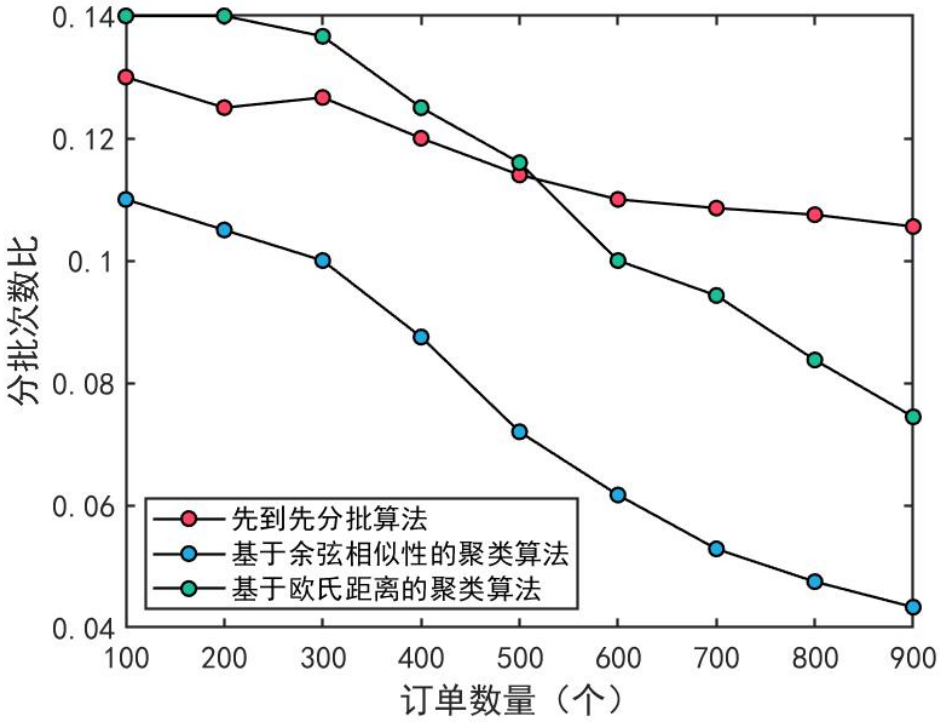

图 2　不同算法下所分批次与不分批批次的比值

再考虑算法复杂度，优先考虑算法的计算时间，对上述实验的计算时间进行统计，得到图 3。由图可知随着订单数量增加，聚类算法的运行时间明显增加，基于余弦相似性的聚类算法所消耗的时间略多于基于欧氏距离的聚类算法，但差别不大，可以不予考虑。

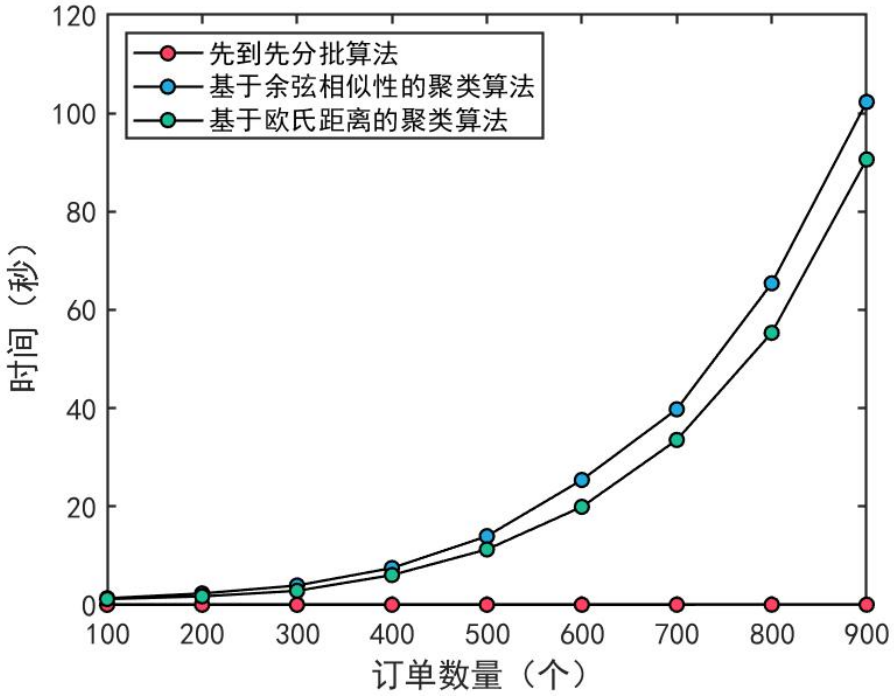

图 3　不同算法运算时间

5.1.3 基于凝聚型层次聚类算法的模型求解

通过上述对各类算法特点的分析，选择基于余弦相似性的聚类算法对本题进行求解。根据题干及附件 1 所给数据，将货架数量 N=200 代入，能够通过基于余弦相似性的聚类的算法计算出每批订单数量、货品种类的结果，并给出分批方案。

通过 MATLAB 软件计算，得到最少分批批次数为 38 批。每批订单数量如表 2。

批次	1	2	3	4	5	6	7	8	9
订单数量	30	15	21	44	59	23	23	18	25
货品种类	200	191	197	200	193	200	199	172	200
批次	10	11	12	13	14	15	16	17	18
订单数量	34	20	28	26	27	27	25	10	24
货品种类	199	192	200	200	194	195	197	199	199
批次	19	20	21	22	23	24	25	26	27
订单数量	21	17	23	27	23	25	27	28	21
货品种类	199	199	200	199	198	196	199	196	200
批次	28	29	30	31	32	33	34	35	36
订单数量	25	22	20	20	25	18	23	21	22
货品种类	185	193	200	192	200	196	200	199	198
批次	37	38							
订单数量	13	22							
货品种类	200	200							

表 2　每批的订单数量及货品种类

完整的原始分批方案数据存入文件 result1.csv 中。

5.2 问题二模型的建立与求解

5.2.1 基于贪心算法的最优货品摆放模型的建立

由上述分析，得到某批分拣任务全部订单（设共 T 个订单）的拣选距离总和最小的优化模型，优化目标为：

$$min\ \mathrm{d} = \sum_{\mathrm{k}=1}^{\mathrm{T}} \mathrm{d}(\mathrm{O_k})$$

根据定义，第 k 个订单的拣选距离定义为该订单中货品 i 所在货架序号的最大值与最小值之差，表示为：

$$\mathrm{d}(\mathrm{O_k}) = max_{i\in\ \mathrm{O_k}}\mathrm{S}(i) - \min_{i\in\ \mathrm{O_k}}\mathrm{S}(i) \qquad i = 1, \cdots, \mathrm{N}$$

x_i为第 i 种货品分配的货架编号，$1 \le x_i \le N$

为了使模型简化，考虑到若将涉及订单数多的货品种类排放在中间序号的货架上，能够有效地缩短拣选距离。再此基础上，采用贪心算法对每一批的订单编号与货品种类构成的 2 维矩阵进行逐列交换，在交换的过程中，只考虑从当前看来是最好的选择，也就是说，不从整体最优上加以考虑，具体策略如下：

总 $d(O_k)$是各订单的 $d(O_k)$之和，则考虑依次处理每一个订单。由于 $d(O_k)$只与最左边的货品与最右边的货品有关，对于每一个订单的处理，只需要考虑更改最左边货品和最右边货品即可。“处理订单” ，定义为当前正在处理的订单，初始为第一个订单，订单编号与货品种类构成的 2 维矩阵如表 3。

处理步骤：

1. 从左往右找到“处理订单”拥有的第一个货品种类。尝试将该货品与其右边的某一货品进行交换，使得所有订单的 $d(O_k)$和更小，即寻找更优解。如果找到了更优解，重复 1；如果找不到了，到步骤 2。

2. 从右往左找到该“处理订单”拥有的第一个货品。尝试将该货品与其左边的某一货品进行交换，使得所有订单的 $d(O_k)$和更小，即寻找更优解。如果找到了更优解，重复 2；如果找不到了，到步骤 3。

3. 该订单处理完毕，将“处理订单”变更为下一个订单。

	P0001	P0002	P0003	P0004	P0005	P0006	…	PN
D0001	0	0	1	1	1	1	…	1
D0002	1	0	0	0	0	0	…	1
D0003	0	1	1	0	0	0	…	1
…	…	…	…	…	…	…	…	…
DT	0	1	0	1	0	0	…	0

表 3　模拟的订单编号与货品种类 2 维矩阵

5.2.2 模型的求解

基于问题一的分批结果，列出每一批订单的订单编号与货品种类 2 维矩阵，根据上述步骤进行订单处理，采用贪心算法求局部最优解，逐列判断、交换货品摆放的位置，直到没有更优的解。流程图如图 4。

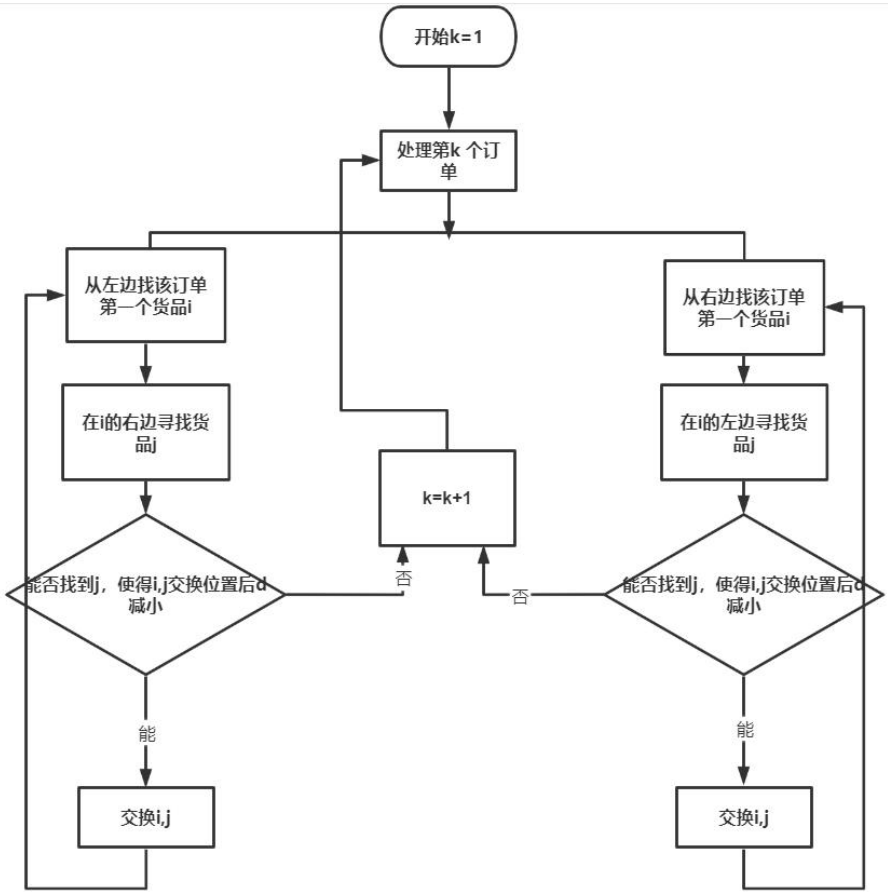

图 4　贪心算法求局部最优解流程图

计算出各批次分拣任务的拣选距离总和，如表 4 所示。

批次	1	2	3	4	5	6	7	8	9
拣选距离总和	3208	1220	956	3915	3339	2252	2322	1006	2633
批次	10	11	12	13	14	15	16	17	18
拣选距离总和	3050	1389	2988	3111	2535	2368	2552	1226	2941
批次	19	20	21	22	23	24	25	26	27
拣选距离总和	2268	1856	2664	2769	2144	2774	3081	3120	2145
批次	28	29	30	31	32	33	34	35	36
拣选距离总和	2421	2522	2067	1903	2831	1293	2791	2212	2370
批次	37	38							
拣选距离总和	1134	2376							

表 4 各批分拣任务的拣选距离总和

计算得到所有批次全部订单的拣选距离总和为89852。

最后将完整原始货品摆放方案按照指定格式输出到文件 result2.csv 中。

5.3 问题三模型的建立与求解

5.3.1 基于多旅行商问题的规划模型的建立

根据参考文献[5],以点a_1来表示分拣工的出发货架， A ={$a_2, a_3, \cdots, a_{p-1}, a_p$}表示 n 个分拣工 $b_1, b_2, \cdots, b_n$ 分配到的订单。定义变量：

$$x_{ij}^k = \begin{cases} 1, 分拣工 b_k 从订单\ i\ 到达订单\ j \\ 0, 否 \end{cases}$$

$$y_i^k = \begin{cases} 1, 分拣工 b_k 分拣订单\ i \\ 0, 否 \end{cases}$$

c_{ij}定义为两订单监的距离，具体描述为：执行完 i 订单的位置到执行 j 订单再到执行完 j 订单所走过的总路程。

目标函数为：

$$Z = min(\sum_{k=1}^{n} z_k)$$

其中：

$$z_k = \sum_{i=0}^{p} \sum_{j=0}^{p} c_{ij} x_{ij}^k \ ,\ k=1,2,\ldots,n$$

约束条件为：

$$\sum_{k=1}^{n} y_i^k = \begin{cases} n, i = 0 \\ 1, i = 1,2,3,\ldots p \end{cases}$$

$$\sum_{i=0}^{p} x_{ij}^k = y_j^k, \forall j = 1,2,\ldots,p; k = 1,2,\ldots,n$$

$$\sum_{j=0}^{p} x_{ij}^k = y_i^k, \forall i = 1,2,\ldots,p; k = 1,2,\ldots,n$$

$X = (x_{ij}^k) \in S$,*S 为支路消去约束，即消去构成不完整路线的解。S 可理解为消去未从给定起始点 a0 出发的闭回路 M1 ={a1,a2,a3} 或排除产生循环的支路（子集 M1 上的弧应小于等于 M1 - 1）等，具体含义可参见文献[6]*

5.3.2 基于模拟退火算法的模型求解

首先对于给定的某一批分拣任务，假设该批任务各订单编号为$a_1, a_3, \cdots, a_{\mathrm{p}-1}, a_{\mathrm{p}}$，且$a_0$为分拣工的起始位置。随机生成一个分配方案，将$a_0$作为分割点，将这个解进行划分，每一部分构成一个环来描述各分拣工的方案，通过动态规划算出该方案的总距离。等概率地使用交换法、移位法、倒置法生成新的随机序列。再求新序列的总距离。这样一直迭代下去，直至设定的迭代数。

其中，为了实现各分拣工的运动距离尽量均衡，在求序列总距离时采用了惩罚函数算法，使得结果趋向均衡。

所得总距离随迭代次数的增加而逐渐减小达到优化的目的，如图 5 所示，可以看出当最大迭代次数大于10^4时最少分拣时间已经趋于平稳，可以视为最优解。其中最大迭代次数=外层迭代次数×每个温度下的迭代次数。

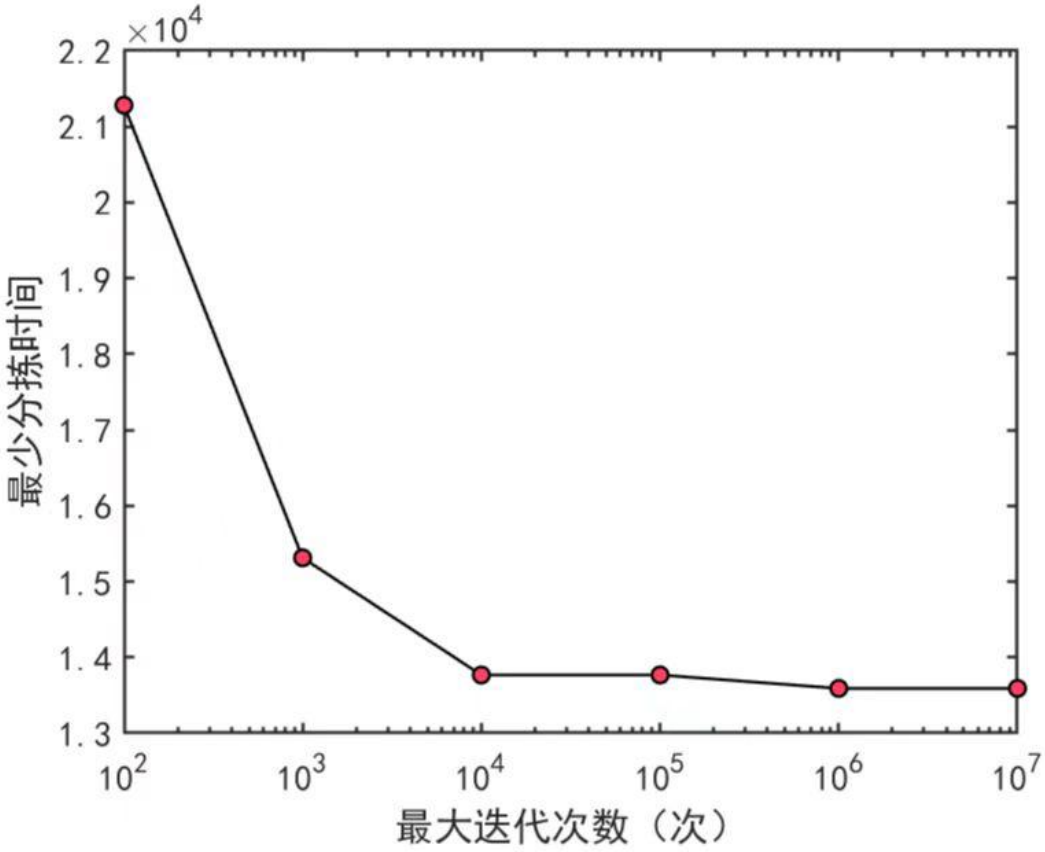

图 5 优化目标值随迭代次数变化图

最后，根据附件 1 的数据，并基于前两问的订单分批和货品摆放基础上，当分拣工人数 n= 5 时，对各个批次，计算出订单的指派结果及每一位分拣工处理订单的顺序，并将完整原始指派方案输出到了文件 result3.csv 中。

六、 模型的分析

6.1 对于问题一模型的合理性分析

对于问题一，针对货架数量 N 从 100 增加到 200，根据图 6 曲线可以发现，当货架数量增加时，分批批次数量逐渐减小，且趋势趋于平缓，这与实际情况是相吻合的，体现出模型的合理性。

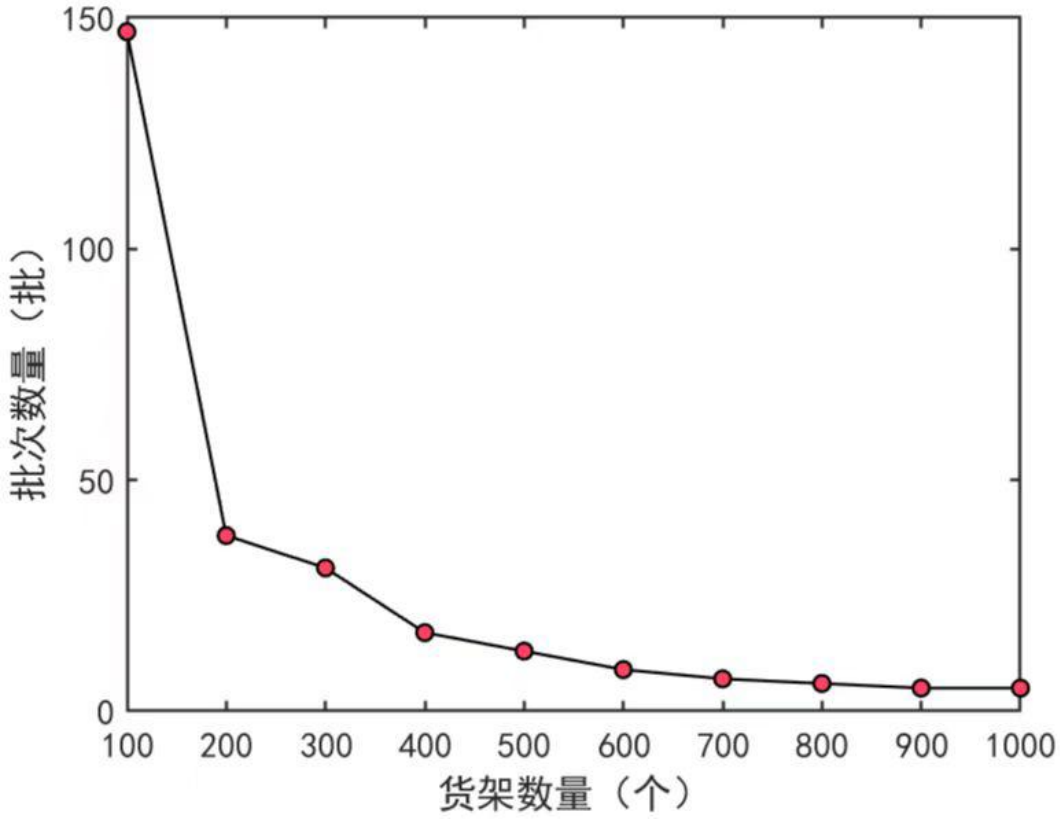

图 6　分批批次随货架数量的变化图

6.2 问题三模型的误差分析

1.模型误差，即简化问题所作假设带来的误差：问题三对分拣工分拣过程做了很多理想化处理，例如：分拣工可以同时拣选同一货架上的货品；不考虑拣货时间；不考虑货筐限制。因此，再将该模型应用于实际时，会因为拥堵等待等因素产生客观与实际之间的误差。

2.模拟退火算法自带的误差：模拟退火算法通过不断迭代而逐步优化结果，而迭代的次数是有限的，如果数据量很大，迭代次数不够的情况下，会产生较大误差。

6.3 问题三模型的复杂度分析

模拟退火算法，当要求提高结果的可靠性时，需要增加迭代次数。这种可靠性的提高，是以增大模型运算的复杂度为代价的，由图 7 可以看出，当最大迭代次数大于10^5时，运算时间迅速增加。

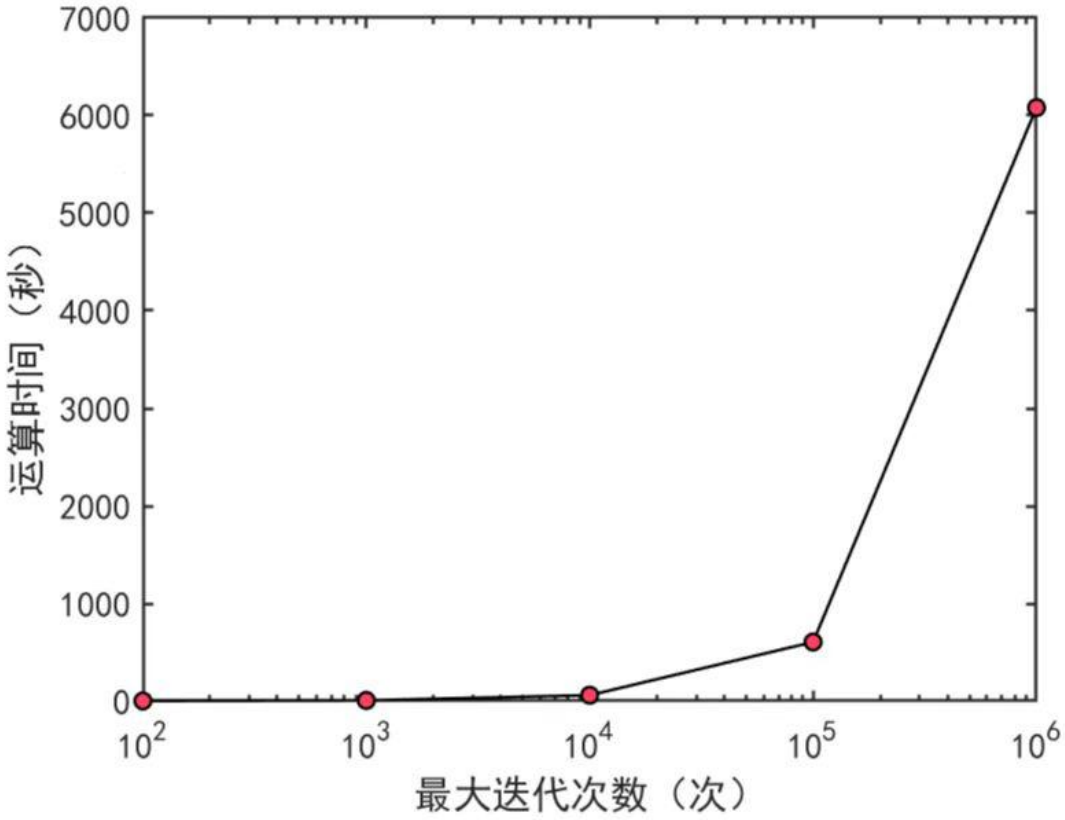

图 7　运算时间随迭代次数变化图

七、 模型的评价

7.1 模型的优点

1.在问题一中，对比了基于欧氏距离的聚类算法、基于余弦相似性的聚类算法、先到先分批算法对所分批次结果的影响，对下一步分批问题的优化奠定了算法基础。

2.在问题一中，对所建立的分批模型进行了合理性分析，研究了在不同货架数量下分批情况的趋势，与实际相符合，模型完整、适应性强。

3.问题二采用贪心算法，考虑通过局部最优解得到全局最优解，保证了每一步都是选择当前最优的，思路清晰，算法简洁实用。

4.问题四中采用了模拟退火算法，初始解可随机选取，具有良好的鲁棒性，具有抵御外界不稳定因素的能力。

7.2 模型的缺点

1.问题二、三模型对拣货员拣货过程和货架排列作了较大的简化，导致存在误差。

2.问题二中，基于余弦相似性的聚类算法所消耗的时间略多于基于欧氏距离的聚类算法。

3.模拟退火算法，如果追求最优解的可靠程度，迭代次数很大，运算时间将变

慢。

7.3 模型的推广

本文对配送中心的分拣环节过程的优化做了较为完整的分析，综合考虑了订单的分批问题、货品摆放问题以及分拣任务的指派问题，较为系统地对货品分拣的效率进行了全面的提高。通过参考相关资料，较为具体地考虑了分拣实际情况而建立的模型，因而对于实际分拣工作具有一定的借鉴意义。本文模型具有普适性，算法运算耗时短，且均基于大量数据建立，更贴合实际应用，能够很好地应用在电商公司的配送中心等场合，有效地提高货品分拣效率。

八、 参考文献

[1]陈志新，董瑞雪，郝宇楠.基于改进遗传算法的自动拣选系统拣选位分配建模与优化[J].工业工程,2019,第 22 卷(6)：40-44，56

[2]牛亚儒.配送中心动态拣货优化研究[D].陕西科技大学,2020

[3]姜启源，谢金星，叶俊.数学模型（第三版）[M].北京.高等教育出版社,2003;274-324.

[4]冯超玲.关于指派问题的数学建模及求解方法[J].广西职业技术学院学报,2013,第 6 卷(4)：25-28

[5]胡士娟，鲁海燕，黄洋，许凯波.求解工作量平衡多旅行商问题的改进遗传算法[J].计算机工程与应用,2019,第 55 卷(17)：150-155，231

[6] 李军，郭耀煌.物流配送车辆优化调度理论与方法[M].北京：中国物资出版社，2001：63-73

附录

附录 1

问题一：通过余弦相似性的聚类算法进行分批

```
%tic;
% O=O(1:2551,1);
% I=I(1:2551,1);
N=max(O);%n 为行数，含义为订 单编号
M=max(I)%M 为列数，含义为种类编号
L=size(I,1);
V=zeros(N,M);%V 是特征向量，V(x,y)表示 x 含不含有 y 种类
for i = 1 : L
    V(O(i,1),I(i,1))=1;
end
X=zeros(N,3);%1 是第一个，2 是第二个，3 是差异值
a=0;%a 是用于累计下标的，与 P 配套使用
for i = 1 : N
    for j = i+1 : N
        a=a+1;
        X(a,1)=i;
        X(a,2)=j;
```

```
X(a,3)=sum(V(i,:).*V(j,:))/(sqrt(sum(V(i,:).^2))*sqrt(sum(V(j,:).^2)));
    end
end
l=zeros(N,1);
for i=1:N
    l(i,1)=i;
end
while(1)
    X=sortrows(X,3,'descend');
    ok=0;
    for K = 1 : a
        i=X(K,1);
        j=X(K,2);
        if(size(find(V(i,:)|V(j,:)),2)>1000)
            continue;
        end
        ok=1;
        while(l(i,1)~=l(l(i,1),1))
            l(i,1)=l(l(i,1),1)
        end
        while(l(j,1)~=l(l(j,1),1))
            l(j,1)=l(l(j,1),1)
        end
        for kk = find(X(:,1)==i)
            X(kk,:)=[];
        end
        for kk = find(X(:,2)==i)
            X(kk,:)=[];
        end
        for kk = find(X(:,1)==j)
            X(kk,:)=[];
        end
        for kk = find(X(:,2)==j)
            X(kk,:)=[];
        end
        l(j,1)=l(i,1);
        V(i,:)=V(i,:)|V(j,:);%在维度中删去 j，并入 i
        V(j,1)=-1;
        a=size(X,1);
        for kk = 1 :size(V,1)
            if(kk==i || V(kk,1)==-1)
                continue;
            end
            a=a+1;
            X(a,1)=min([kk,i]);
            X(a,2)=max([kk,i]);

X(a,3)=sum(V(i,:).*V(kk,:))/(sqrt(sum(V(i,:).^2))*sqrt(sum(V(kk,:).^2)));
        end
        break;
    end
    if(ok==0)
```

```
        break;
    end
end
%先以第一个为例进行一次合并操作
for i=1:N
    while(l(i,1)~=l(l(i,1),1))
        l(i,1)=l(l(i,1),1)
    end
end
num=0;%用于存储有几个批次
index=ones(N,1).*(-1);%存储每一种标号在 cell 中哪个位置，还没放就是-1
as=cell(1,N);
for i = 1 : N
    if(index(l(i,1),1)==-1)
        num=num+1;
        index(l(i,1),1)=num;
        as{num}=[as{num} i];
        continue;
    end
    as{index(l(i,1),1)}=[as{index(l(i,1),1)} i];
    pp=index(l(i,1),1);
end
%runtime=toc;
S=zeros(N,M);%V 是特征向量，V(x,y)表示 x 含不含有 y 种类
for i = 1 : L
    S(O(i,1),I(i,1))=1;
end
gg=0;
for i = as{1}
   gg=gg+1;
   cg(gg,:)=S(i,:);
end
for i = 1 :100
    if(size(find(cg(:,i)),1)==0)
        cg(:,i)=[];
    end
end
writematrix(cg,'cg.csv');
result(1,1)="OrderNo";
result(1,2)="GroupNo";
for i = 1: N
    result(i+1,1)="D"+num2str(i, '%04d');
    result(i+1,2)=num2str(index(l(i,1),1));
end
writematrix(result,'result.csv');
```

附录 2

问题二：通过贪心算法求解最佳货品摆放位置

```
putin3=cell(1,38);
as=cell(1,size(pi,2));
sumz=0;
for k=1:38
    X=[];
    pi=fp{1,k};
    for i = 1 : size(pi,2)
        X(i,1)=i;
        X(i,2)=sum(pi(:,i));
    end
    X=sortrows(X,2,'descend');
    num=0;
    cg=[];%重构矩阵
    mark=ones(size(pi,2),1).*(-10000);%标记矩阵，每一种货品的位置在哪
    while(1)
        num=num+1;
        if(X(num,2)==0)
            break;
        end
        if(mod(num,2)~=0)
            cg=[cg pi(:,X(num,1))];
            mark(X(num,1),1)=size(cg,2);
        else
            mark(X(num,1),1)=1;
            mark(:,1)=mark(:,1)+1;
            cg=[pi(:,X(num,1)) cg];
        end
    end
    for ceng = 1 :size(cg,1)%根据第几层的那啥来换
        while(1)
            i=find(cg(ceng,:),1);%j 就是要往右送的这一列
            ok=0;
            for j = i+1 : size(cg,2);
                sum1=0;
                for z = 1 :size(cg,1)
                    de=find(cg(z,:));
                    sum1=sum1+de(size(de,2))-de(1);
                end
                t=cg(:,i);
                cg(:,i)=cg(:,j);
                cg(:,j)=t;
                sum2=0;
                for z = 1 :size(cg,1)
                    de=find(cg(z,:));
                    sum2=sum2+de(size(de,2))-de(1);
                end
                if(sum2<sum1)
                    st=mark(i,1);
                    mark(i,1)=mark(j,1);
                    mark(j,1)=st;
```

```
                    ok=1;
                    break;
                end
                t=cg(:,i);
                cg(:,i)=cg(:,j);
                cg(:,j)=t;
            end
            if(ok==0)
                break;
            end
        end
        while(1)
            i=find(cg(ceng,:),1,'last');%j 就是要往右送的这一列
            ok=0;
            for j = i-1 : -1:  1;
                sum1=0;
                for z = 1 :size(cg,1)
                    de=find(cg(z,:));
                    sum1=sum1+de(size(de,2))-de(1);
                end
                t=cg(:,i);
                cg(:,i)=cg(:,j);
                cg(:,j)=t;
                sum2=0;
                for z = 1 :size(cg,1)
                    de=find(cg(z,:));
                    sum2=sum2+de(size(de,2))-de(1);
                end
                if(sum2<sum1)
                    st=mark(i,1);
                    mark(i,1)=mark(j,1);
                    mark(j,1)=st;
                    ok=1;
                    break;
                end
                t=cg(:,i);
                cg(:,i)=cg(:,j);
                cg(:,j)=t;
            end
            if(ok==0)
                break;
            end
        end
    end
    sum3(k,1)=0;
    for z = 1 :size(cg,1)
    de=find(cg(z,:));
    sum3(k,1)=sum3(k,1)+de(size(de,2))-de(1);
    end
    sumz=sumz+sum3(k,1);
    for i = 1 : size(pi,2)
        if(mark(i,1)<0)
            continue;
```

```
        end
        as{i}=[as{i} [k;mark(i,1)]];
    end
    t=cg(:,4);
    cg(:,4)=cg(:,5);
    cg(:,5)=t;
    putin3{k}=cg;
end
    result(1,1)="ItemNo";
    result(1,2)="GroupNo";
    result(1,3)="ShelfNo";
    rnum=1;
    for i = 1:size(pi,2)
        kk=as{i};
        for j = 1 : size(kk,2)
           rnum=rnum+1;
           result(rnum,1)="P"+num2str(i, '%04d');
           result(rnum,2)=num2str(kk(1,j));
           result(rnum,3)=num2str(kk(2,j));
        end
    end
    sum3(39,1)=sumz;
    writematrix(result,'result2.csv');
    writematrix(sum3,'sum.csv');
```

附录 3

问题三：通过模拟退火算法分配订单任务

```
function [path1] = get_new_xl(path0)
    n = length(path0);
    % 随机选择两种产生新路径的方法
    p1 = 0.33;
    p2 = 0.33;
    r = rand(1);
    if  r< p1
        c1 = randi(n);
        c2 = randi(n);
        path1 = path0;
        path1(c1) = path0(c2);
        path1(c2) = path0(c1);
    elseif r < p1+p2
        c1 = randi(n);
        c2 = randi(n);
        c3 = randi(n);
        sort_c = sort([c1 c2 c3]);
        c1 = sort_c(1);  c2 = sort_c(2);  c3 = sort_c(3);  % c1 < = c2 <=
c3
        tem1 = path0(1:c1-1);
        tem2 = path0(c1:c2);
```

```
        tem3 = path0(c2+1:c3);
        tem4 = path0(c3+1:end);
        path1 = [tem1 tem3 tem2 tem4];
    else
        c1 = randi(n);
        c2 = randi(n);
        if c1>c2
            tem = c2;
            c2 = c1;
            c1 = tem;
        end
        tem1 = path0(1:c1-1);
        tem2 = path0(c1:c2);
        tem3 = path0(c2+1:end);
        path1 = [tem1 fliplr(tem2) tem3];
    end
end


function [F] = lenth(xl,S,dd)
%S 为路径,xl 为顺序的求法，S 能够提供每一个端点到另一个端点所需的距离
fg=find(xl(1,:)==80);%fg 即是分隔，找到分割的零；
gg=cell(1,5);
gg{1}=[80 xl(1,1:fg(1))];
gg{2}=xl(1,fg(1):fg(2));
gg{3}=xl(1,fg(2):fg(3));
gg{4}=xl(1,fg(3):fg(4));
gg{5}=[xl(1,fg(4):size(xl,2)) 80]
%得到了每一个人会走的序列
for k = 1: 5 %处理第 k 个人
    sum=0;
    dp=[];
    X=gg{k};
    dp(1,1)=0;
    dp(1,2)=0;%2 代表从右边出来的，1 代表 dp 到第一层
    for i = 1 : size(X(1,:),2)-1
        for j = 1 : 2      %讨论这一层从几进入
            a=X(1,i);
            b=X(1,i+1);
            if(j==1)
                dp(i+1,2)=min(dp(i,1)+S(a,1,b,1)+abs(dd(a,1)-
dd(a,2)),dp(i,2)+S(a,2,b,1)+abs(dd(a,1)-dd(a,2)));%从 1（左）进入，则该层就该
是从 2（出去）
            else%从 2 进入，则会使从 1 出去
                dp(i+1,1)=min(dp(i,1)+S(a,1,b,2)+abs(dd(a,1)-
dd(a,2)),dp(i,2)+S(a,2,b,2)+abs(dd(a,1)-dd(a,2)));
            end
        end
    end
    sum=sum+min(dp(size(X(1,:),2),1),dp(size(X(1,:),2),2));
end
F=sum;
end
```

```
tic;
result3(1,1)="OrderNo";
result3(1,2)="GroupNo";
result3(1,3)="WorkerNo";
result3(1,4)="TaskNo";
num=1;
for k = 1:38
rng('shuffle');  % 控制随机数的生成，否则每次打开 matlab 得到的结果都一样
load 'putin3.mat';
load '1out.mat';
pi=putin3{k};
shishei=as{k};
for i = 1:size(pi,1)%得到端点值
    v=find(pi(i,:));
    dd(i,1)=v(1,1);
    dd(i,2)=v(1,size(v,2));
end


for i=1:size(pi,1)
    for j = 1:size(pi,1)
        S(i,1,j,1)=abs(dd(i,1)-dd(j,1));
        S(j,1,i,1)=abs(dd(i,1)-dd(j,1));
        S(i,1,j,2)=abs(dd(i,1)-dd(j,2));
        S(j,2,i,1)=abs(dd(i,1)-dd(j,2));
        S(i,2,j,1)=abs(dd(i,2)-dd(j,1));
        S(j,1,i,2)=abs(dd(i,2)-dd(j,1));
        S(i,2,j,2)=abs(dd(i,2)-dd(j,2));
        S(j,2,i,2)=abs(dd(i,2)-dd(j,2));
    end
    S(i,1,80,1)=dd(i,1);
    S(i,1,80,2)=dd(i,1);
    S(i,2,80,1)=dd(i,2);
    S(i,2,80,2)=dd(i,2);
    S(80,1,i,1)=dd(i,1);
    S(80,1,i,2)=dd(i,1);
    S(80,2,i,1)=dd(i,2);
    S(80,2,i,2)=dd(i,2);
end
dd(80,1)=0;
dd(80,2)=0;
T0 = 10;   % 初始温度 1
T = T0; % 迭代中温度会发生改变，第一次迭代时温度就是 T0
maxgen = 10;  % 外层迭代次数
Lk = 1000;  % 每个温度下的迭代次数
alpfa = 0.95;  % 温度衰减系数


patht = randperm(size(pi,1));
path0=patht;
fen=floor(size(pi,1)/5);
path1 =[patht(1,1:fen*1) 80 patht(1,fen*1+1:fen*2) 80
patht(1,fen*2+1:fen*3) 80 patht(1,fen*3+1:fen*4) 80
```

```
patht(fen*4+1:size(pi,1))];
result0 = lenth(path1,S,dd); % 调用我们自己写的 calculate_tsp_d 函数计算当前路径的距离
min_result = result0;      % 初始化找到的最佳的解对应的距离为 result0
RESULT = zeros(maxgen,1); % 记录每一次外层循环结束后找到的 min_result (方便画图)


for iter = 1 : maxgen  % 外循环，我这里采用的是指定最大迭代次数
    for i = 1 : Lk  %  内循环，在每个温度下开始迭代
        patht=get_new_xl(path0);  % 调用我们自己写的 gen_new_path 函数生成新的路径
        path1 =[patht(1,1:fen*1) 80 patht(1,fen*1+1:fen*2) 80
patht(1,fen*2+1:fen*3) 80 patht(1,fen*3+1:fen*4) 80
patht(fen*4+1:size(pi,1))];
        result1 = lenth(path1,S,dd); % 计算新路径的距离
        %如果新解距离短，则直接把新解的值赋值给原来的解
        if result1 < result0
            path0 = patht; % 更新当前路径为新路径
            result0 = result1;
        else
            p = exp(-(result1 - result0)/T); % 根据 Metropolis 准则计算一个概率
            if rand(1) < p   % 生成一个随机数和这个概率比较，如果该随机数小于这个概率
                path0 = patht;  % 更新当前路径为新路径
                result0 = result1;
            end
        end
        % 判断是否要更新找到的最佳的解
        if result0 < min_result  % 如果当前解更好，则对其进行更新
            min_result = result0;  % 更新最小的距离
            best_path = path1;  % 更新找到的最短路径
        end
    end
    RESULT(iter) = min_result; % 保存本轮外循环结束后找到的最小距离
    T = alpfa*T;   % 温度下降
end
    minresult(1,k)=min_result;
    ss=find(best_path(1,:)==80);%fg 即是分隔，找到分割的零；
    gg=cell(1,5);
    gg{1}=best_path(1,1:ss(1)-1);
    gg{2}=best_path(1,ss(1)+1:ss(2)-1);
    gg{3}=best_path(1,ss(2)+1:ss(3)-1);
    gg{4}=best_path(1,ss(3)+1:ss(4)-1);
    gg{5}=best_path(1,ss(4)+1:size(best_path,2));
    for i =1:5
        x=gg{i};
        for j =1:size(x,2)
            num=num+1;
            result3(num,1)="D"+num2str(shishei(1,x(j)), '%04d');
            result3(num,2)=num2str(k);
```

```
            result3(num,3)=num2str(i);
            result3(num,4)=num2str(j);
        end
    end
    jy=bj(best_path,S,dd);
    cd{k}=jy;
    clearvars -except num result3 minresult cd;
end
time=toc;
writematrix(result3,'result3.csv');
save 'time.mat';
save 'minresult';
save 'cd';
```