针对分拣优化方案的研究

摘要

分拣系统的优化与设计对电商公司的工作效率有着极大的影响，本文将利用贪心算法，蒙特卡罗算法，遗传算法对分拣系统的优化以及工人的分配进行优化。

对于问题一，本文首先利用 python 将订单数据进行预处理，以订单为键，该订单包含的货品形成的集合为值构成字典。对于批次的划分本文采用贪心算法的思想，共提供了四种划分思路，得到的总批次分别是 95，68，63，38。其中最优的批次划分思路为：先将货品集合相互包含的订单进行合并，形成同类订单，再将订单对应的货品集合长度进行降序排序，将最长的集合作为母集合，合并次长，次次长集合等等，直到遍历完所有集合，保留下不超过货架总类 N 的集合，不断迭代上述过程，最终得到分批次的结果。当 N=200 时，分批结果为 38 批，每个批次的种类数平均为 196.71 种。

对于问题二，本文在问题一中的分批结果的基础上，将每个批次分开计算。为保证挑选距离最小，本文使相同订单的货品尽可能接近。本文以每个批次的挑选距离为目标函数，分别利用蒙特卡洛算法，以及遗传算法确定每次迭代的货物排放顺序，经过不断迭代，最终得到蒙特卡洛方法的结果为 147563，遗传算法的结果为 147462，可以遗传算法的结果比蒙特卡洛算法的结果稍微好一点，两种方法得到的结果相近，也可以说明结果的可靠性。

对于问题三，本文在前两问的基础上，在已知分批结果与货物排列顺序的条件下，对每个批次的订单进行分配，为做到均衡，首先将订单平均分配。由于时间与距离成正比，本文将距离视作时间进行分析。以每个批次的任务中，工人完成各自订单花费时间的最大值，以及工人花费时间的极差作为目标函数，利用蒙特卡洛的算法，不断迭代优化，最终得到最优的分配方案。完成所有批次所有订单的总时间为 62030，定义均衡度为每个批次中工人完成各自订单花费时间最小值与最大值的比值，所有批次的平均均衡度为 0.89。最后，本文变化 N 的值，对模型的稳定性进行了检验分析，随着 N 值的变化，批次数平缓下降，符合预期，曲线变化平缓说明模型的稳定性好，不会出现较大的突变。文末对模型优缺点的进行了分析，并提出了相应的改进措施。

关键字：分拣系统　贪心算法　遗传算法　蒙特卡罗法

目录

一、问题重述 ····· **1**

1.1 问题背景 ····· 1

1.2 问题提出 ····· 1

二、问题分析 ····· **1**

2.1 问题一分析 ····· 1

2.2 问题二分析 ····· 2

2.3 问题三分析 ····· 2

三、模型的假设与约定 ····· **2**

四、符号说明 ····· **3**

五、模型的建立与求解 ····· **3**

5.1 问题一模型的建立与求解 ····· 3

5.1.1 贪心算法正确性的证明 ····· 3

5.1.2 思路一 ····· 4

5.1.3 思路二 ····· 6

5.1.4 思路三 ····· 7

5.1.5 思路四 ····· 9

5.2 问题二模型的建立与求解 ····· 13

5.2.1 蒙特卡罗法 ····· 13

5.2.2 遗传算法 ····· 13

5.3 问题三模型的建立与求解 ····· 16

六、模型的检验与灵敏度分析 ····· **17**

七、模型评价与改进 ····· **18**

7.1 问题一模型的评价与改进 ····· 18

7.2 问题二模型的评价与改进 ····· 18

7.3 问题三模型的评价与改进 ····· 19

八、参考文献 ····· **19**

A 附录 ····· **19**

1.1 程序代码 19
1.2 支撑材料的文件列表 36

一、 问题重述

1.1 问题背景

通过配送中心周转货物是物流行业降低运输、库存、分拣和配送成本的重要手段，而分拣环节对某一个配送中心的作业效率起着决定性作用，优化分拣环节可以大幅提升配送中心的运转效率。分拣环节中有很多可以优化的点，譬如转运的批次、货品摆放位置、分拣任务的指派等都可以进行优化，需要采用数学思维建模求解合适的方案优化分拣系统。

1.2 问题提出

某电商公司配送中心的工作流程分为统计汇总、转运上架、按订单分拣、核对打包等步骤。其中，分拣环节耗时最长，其效率决定了配送中心的整体性能。转运工通过系统汇总出的订单信息将相应的货品由仓库转运至分拣处并放置到货架上等待分拣。上架时，一个货架中仅放置同一种货品。随着该公司业务量不断增长，未来有可能出现当天待配送订单所含货品种类数大于当前货架数量的情况。但是，由于受到场地和成本的限制，很难直接增加货架。本文希望解决如下问题:

(1) 将当日订单分批，保证每批订单所含货品种类不超过一定数量的条件下批次越少越好以期达到较高的转运效率。

(2) 优化货品摆放位置，使得一批分拣任务中分拣工移动距离即订单的拣选距离尽可能小。

(3) 对于某一批货品摆放位置已知的分拣任务，在考虑分拣工实际工作条件的前提下将任务分配给分拣工，要求分拣工能尽快完成任务且运动距离均衡。

二、 问题分析

2.1 问题一分析

本问要求设计分批算法，将一天的订单分为多个批次，在保证每批订单包含货品种类不超过 $N = 200$ 的条件下要求批次尽可能的少。对于该问题本文考虑采用优化后的贪心算法进行求解，并从多个思路出发不断对结果进行优化，最后给出贪心算法正确性的证明。由于题目中批次与订单间的相互关系和集合有着类似的特征，所以本文将它们视作集合从而进行分析，集合化的同时也方便之后使用 Python 语言进行编程与处理。

本文将每个订单看成一个集合，则订单中的货品种类就是集合中的元素。如此一来每一批次中的订单的合并就可以用集合取并集的运算方法来处理，从而达到简化计算，使解决算法更加直观实效的目的。

2.2 问题二分析

本问要求基于问题一的分批情况设计一种货品摆放算法来优化货品摆放位置，从而对于给定的某一批分拣任务分拣工拣选货品时移动距离能够尽量短，达到提高效率的目的。对于该问题本文考虑采用优化的遗传算法进行分析和求解。将每一批次中所有订单对应的拣选距离之和作为目标函数，并以第一问求得的批次订单数作为限制条件。由于货品摆放位置的选择非常多，我们用遗传算法随机模拟多次不同的商品摆放位置，并用 Python 字典查找的方法来计算距离，通过比较目标函数值即总订单拣选距离长短来进行多次优化迭代从而找出问题的最优解。

2.3 问题三分析

第三问要求在已知货品位置的前提下，加入分拣工人并对其进行合理分配，使得能够尽快完成任务并且运动距离尽可能均衡。本文采取蒙特卡罗法对订单分配情况进行随机处理，通过确定目标函数以及限制条件，对问题进行分析优化。对于任意一工人，可以得到当按照从每个订单对应的货品位置的两端开始分拣和结束时，工人在分拣这一订单所走过的路程最短。所以，可以将工人分拣不同订单的过程视为其在不同订单对应两端货品间往复运动的过程，如此便可得到目标函数，即每个工人处理自己分拣任务所走过的总距离。而对于所有工人，则存在总完成订单数一定这一限制条件。

但是，考虑到运动距离尽可能均衡，每位工人所分配到的订单数可能不同，使问题进一步趋于复杂。为简化目标函数的自由度，本文假设开始时每位工人分配到的订单数相同，如此，每位工人的独立性便被消除，从而达到简化算法的效果。在得到最终的最优分配方案以后，观察得到的分配结果，判断各工人间是否出现了总距离差距过大，即分配不均衡的情况发生。若有，则可以进一步通过调整每位工人的订单数或者订单分配情况从而进行优化修正。

三、 模型的假设与约定

1. 假设题目所有数据来源皆真实可靠，题目中假设皆真实合理。
2. 假设分拣工分拣货品时不出现失误，严格按照程序执行。
3. 假设每个订单中货品类别最高均不会超过货架数量。
4. 假设每位分拣工间不存在差异，考虑分配是否均衡只与路程有关。

四、 符号说明

表 1 符号说明

符号	说明	单位
N	货品种类数	/
w_i	每个订单的货品种类数	/
x_i	逻辑 0-1 变量	/
p	算子变异概率	/
d	每位工人完成拣选任务总路程	/
S	订单中货物所在位置序号	/

五、 模型的建立与求解

5.1 问题一模型的建立与求解

本文将问题转化为集合运算，则订单的大小指的就是订单中包含的商品的种类。在开始处理前本文首先进行了一个简单的预处理，即把 923 个订单中有明显包含关系的订单先进行合并，即将子集父集合并。经过这一步后剩余订单的数量是 876 个。本文采用贪心算法解决问题，希望每一批可以尽可能包含更多的货品种类。

在解决问题一的过程中，本文先证明贪心算法的正确性，之后一共尝试了四种思路，每次在前一种思路上进行进一步优化，最终得到最优化的解决思路。

5.1.1 贪心算法正确性的证明

本文参考最优装载问题, 主要通过数学归纳法给出贪心算法得到的结果是最优解的证明。

假设每个批次可以包含的最大的货品的种类数为 N，每个订单包含的货品的种类数为 w_i。现在要求每个批次尽可能多的包含订单。问题可以描述为:

$$max \sum_{i=1}^{n} x_i \tag{1}$$

$$\begin{cases} \sum_{i=1}^{n} w_i x_i \leq C \\ x_i \in \{0,1\}, 1 \leq i \leq n \end{cases} \tag{2}$$

要使用贪心算法解决问题，本文需要证明：(1) 该问题具备贪心选择性质；(2) 该问题具备最优子结构性质。

(1) 贪心选择性质的证明

设订单已依其包含的商品种类从多到少排序，$(x_1, x_2.....x_n)$ 是该问题的一个最优解。又设 $k = min\{i|x_i = 1\}(1 \leq i \leq n)$。易知如果给定的问题有解，则 $1 \leq k \leq n$；

1° 当 $k = 1$ 时，$(x_1, x_2.....x_n)$ 是满足贪心选择性质的最优解。

2° 当 $k > 1$ 时，取 $y_1 = 1, y_k = 0, y_i = x_i, 1 < i \leq n, i \neq k$，则 $\sum_{i=1}^{n} w_i y_i = w_1 - w_k + \sum_{i=1}^{n} w_i x_i \leq \sum_{i=1}^{n} w_i x_i \leq c \Rightarrow (y_1, y_2.....y_n)$ 是所给问题的可行解。又因为 $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i \Rightarrow (y_1, y_2.....y_n)$ 是满足贪心选择性质的最优解。

综上所述，得以证明该问题具备贪心选择性质。

(2) 最优子结构性质的证明

设 $(x_1, x_2.....x_n)$ 是满足贪心选择性质的最优解，易知，$x_1 = 1, (x_2, x_3.....x_n)$ 是一批货品种类为 $c - w_1$，订单为 $2, 3, ...n$ 时相应的问题的最优解。得以证明，最优问题具备最优子结构性质。

由上述，可以证明贪心算法得到的解是全局最优解。

5.1.2 思路一

首先本文从小的订单集合开始，每一步进行合并的是使相加后的集合最小的两个集合。由于集合与集合间可能有交集的情况发生，所以每次合并的情况要更加复杂。采用这种思路最终得到的批次数是 95 批。

批次	订单数	商品种类数	批次	订单数	商品种类数
1	11	110	49	9	146
2	7	111	50	14	147
3	10	111	51	8	147
4	5	115	52	10	148
5	16	116	53	14	149
6	5	116	54	9	151
7	7	117	55	7	153
8	5	117	56	8	153
9	8	119	57	8	158
10	8	120	58	10	160
11	13	121	59	11	160
12	8	122	60	11	162
13	14	122	61	11	163
14	9	125	62	14	166
15	7	125	63	11	166
16	4	125	64	10	167
17	9	126	65	9	169
18	7	126	66	10	171
19	12	127	67	12	171
20	7	128	68	11	172
21	9	128	69	10	172
22	9	130	70	7	172
23	20	130	71	8	173
24	8	130	72	11	173
25	5	130	73	11	177
26	9	131	74	24	178
27	5	132	75	11	178
28	7	132	76	15	179
29	10	133	77	14	179
30	8	133	78	10	182
31	9	133	79	6	183
32	10	134	80	10	184
33	6	135	81	13	185
34	6	136	82	14	186
35	7	136	83	10	186
36	11	137	84	9	186
37	9	138	85	12	188
38	9	138	86	13	188
39	9	140	87	10	190
40	7	140	88	12	192
41	6	140	89	10	193
42	6	140	90	8	194
43	8	141	91	14	195
44	6	142	92	14	196
45	4	142	93	5	196
46	7	143	94	24	197
47	2	143	95	16	199
48	11	146			

图 1　思路一结果

结果如图1所示，可以明显的看到每个批次的商品种类数都距离 200 有较大的差距，分析其原因可能是随着每步迭代次数的增加，每个集合的长度同时增长，导致最小的集合可能也容纳不下其他任意一个集合，导致循环提前结束，从而形成这样的结果。

5.1.3 思路二

从思路一的结果来看，有很多批次并没有充分利用 $N = 200$ 的数量上限这一条件，这与贪心算法的本质相违背。本文考虑了出现这种问题的原因，发现思路一中各批次是同步增长，当先处理小的订单集合时，容易导致小集合经过数次合并后都变成了不易进一步合并的大集合这一情况发生。为了解决这一问题本文又尝试了另一种思路，即先把所有集合从大到小排好序，对大集合先进行处理。

本文首先选取最大的集合，然后按照顺序从大到小遍历剩下的集合，如果与目前的集合合并后不超过 $N = 200$ 的数量上限则进行合并，否则考虑下一个更小的集合能否进行合并。这种思路比较符合贪心的想法的要求。采用这种思路最终得到的批次数是 68 批。

结果如图2所示，可以看到虽然几乎每一个批次的种类数都接近 200，但是有的批次订单数却很少，说明该批次中的订单重合度不是很高，并且最后一个批次的种类数只有 21，因此这种算法得到的结果仍有较大的改进空间。

批次	订单数	商品种类数	批次	订单数	商品种类数
1	38	200	35	9	200
2	24	200	36	9	200
3	26	200	37	9	200
4	24	200	38	9	200
5	24	200	39	10	199
6	28	199	40	9	200
7	24	200	41	10	200
8	26	200	42	12	200
9	25	200	43	10	200
10	23	200	44	10	200
11	7	200	45	11	200
12	5	200	46	11	200
13	6	200	47	12	200
14	9	200	48	11	200
15	27	200	49	11	200
16	8	200	50	14	200
17	7	200	51	12	200
18	8	200	52	12	199
19	6	200	53	12	200
20	9	200	54	13	199
21	9	200	55	12	200
22	7	200	56	13	200
23	9	200	57	14	199
24	8	200	58	15	200
25	12	200	59	13	199
26	8	200	60	16	200
27	9	200	61	16	200
28	9	200	62	16	200
29	11	200	63	16	200
30	8	200	64	19	200
31	9	200	65	18	199
32	8	200	66	22	200
33	10	200	67	32	200
34	9	200	68	5	21

图 2　思路二结果

5.1.4 思路三

观察思路二的结果发现结果较思路一明显更为理想，大部分批次都很接近 $N = 200$ 的数量上限，但仍存在空余。本文考虑进行进一步优化，将思路二和思路一结合，也就是每一步合并不是按简单的从大到小的顺序，而是像思路一一样先选一个最大的集合，

然后找能使合并后的集合最小的集合进行合并，这样因为每次合并的是小的集合，所以每一批订单都更可能接近 $N = 200$ 这一数量上限，更加符合贪心的想法。采用这种思路最终得到的批次数是 63 批。

思路三算法的流程图如图3所示：

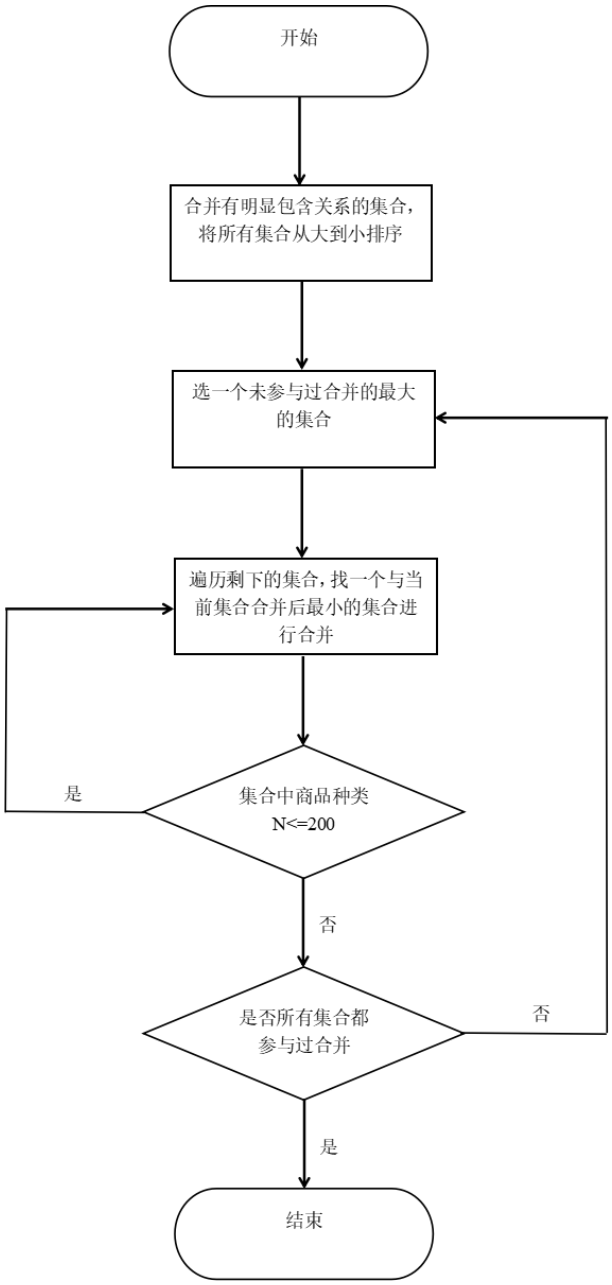

图 3　思路三流程图

结果如图4所示，可以看到大部分的批次最终都接近于 200，但是随着批次的增大订单数逐渐递减，考虑到后边批次合并不充分的问题，我们继续进行改进，寻找新的思路。

批次	订单数	商品种类数	批次	订单数	商品种类数
1	35	200	33	12	195
2	24	198	34	13	193
3	23	200	35	14	195
4	28	198	36	11	193
5	22	198	37	14	197
6	19	198	38	12	197
7	18	196	39	14	194
8	19	193	40	14	200
9	18	196	41	12	195
10	19	194	42	12	194
11	21	195	43	12	199
12	16	199	44	10	191
13	14	199	45	12	196
14	14	193	46	10	190
15	22	200	47	13	200
16	20	194	48	11	198
17	16	197	49	14	197
18	19	193	50	10	189
19	17	194	51	10	195
20	21	199	52	10	193
21	13	193	53	10	195
22	19	196	54	9	194
23	15	198	55	10	192
24	15	196	56	8	187
25	14	193	57	8	194
26	14	200	58	7	180
27	16	198	59	8	191
28	18	197	60	7	184
29	20	199	61	7	192
30	20	200	62	7	191
31	13	195	63	4	125
32	16	200			

图 4　思路三结果

5.1.5 思路四

尽管经过前三种思路的尝试和改进已经得到了一个比较理想的结果，本文还是尝试了一种更加创新的思路，并得到了更加优化的结果。这种思路最终得到的批次数是 38 批。其思路具体步骤为：

step1 对 876 个集合进行从大到小排序。

step2 找到最大的集合 M。

step3 按顺序遍历所有剩下的集合，每次选择与这个集合交集最大的集合进行合并，之后再次寻找与合并后的集合 $M^{'}$ 交集最大的集合与之进行合并。

step4 结合思路三，若集合合并后超过 $N = 200$ 的数量上限则不进行合并，而是寻找能使合并后的集合最小的集合进行合并，最后再找与之交集最多的集合进行合并，直到合并后最小的集合也超过 $N = 200$ 的数量上限为止。

step5 按照相同的思路进行下一批次的合并。

思路四算法的流程图如图5所示

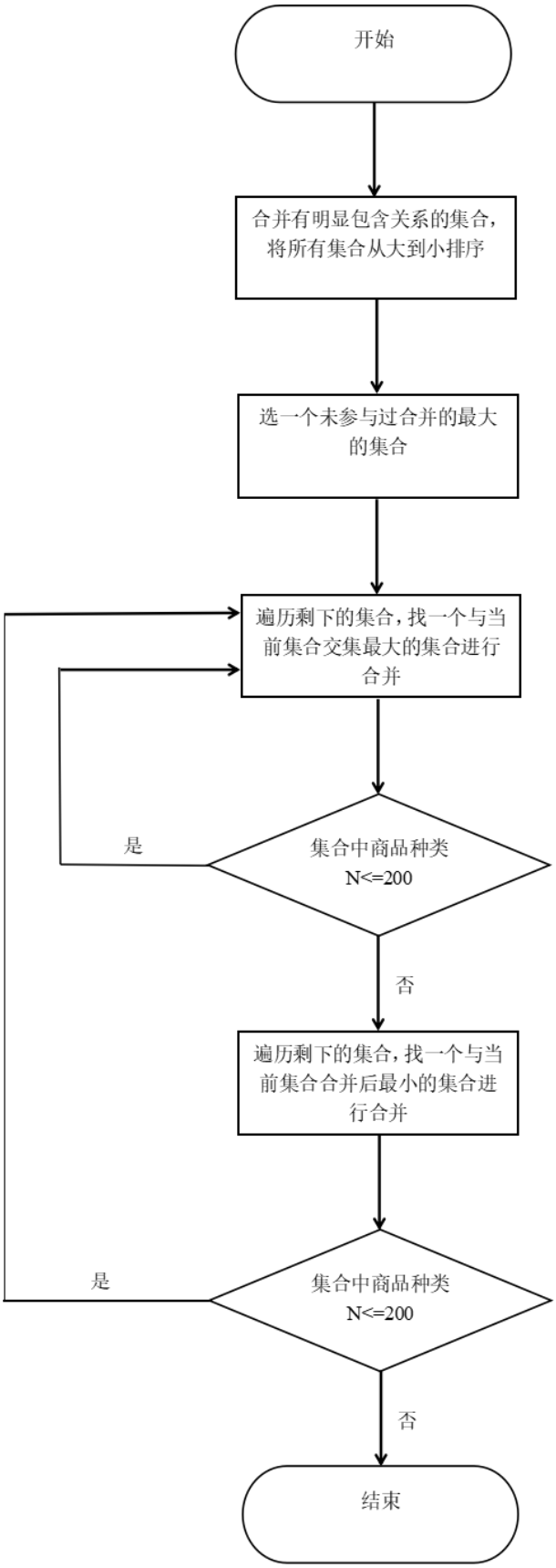
开始
合并有明显包含关系的集合，将所有集合从大到小排序
选一个未参与过合并的最大的集合
遍历剩下的集合，找一个与当前集合交集最大的集合进行合并
是
集合中商品种类N<=200
否
遍历剩下的集合，找一个与当前集合合并后最小的集合进行合并
是
集合中商品种类N<=200
否
结束

图 5　思路四流程图

结果如图6所示：

批次	订单数	商品种类数	批次	订单数	商品种类数
1	35	199	20	14	199
2	24	200	21	25	199
3	28	200	22	24	200
4	24	200	23	26	199
5	24	200	24	32	200
6	28	199	25	26	197
7	24	200	26	26	199
8	13	200	27	42	199
9	26	199	28	25	199
10	27	198	29	19	200
11	15	199	30	28	200
12	24	199	31	32	200
13	22	200	32	27	196
14	13	199	33	37	198
15	25	200	34	35	199
16	11	200	35	25	197
17	13	199	36	27	199
18	26	200	37	16	200
19	26	200	38	9	104

图 6　思路四结果

我们用柱状图反映每一批次订单数和货品种类数的关系，如图7所示，可以明显的看到每个批次的订单数都在 22 左右，分布较为均匀，且货品种类数均接近 200，因此可认为是一个较好的方案，整个分批方案见结果 result1.csv。

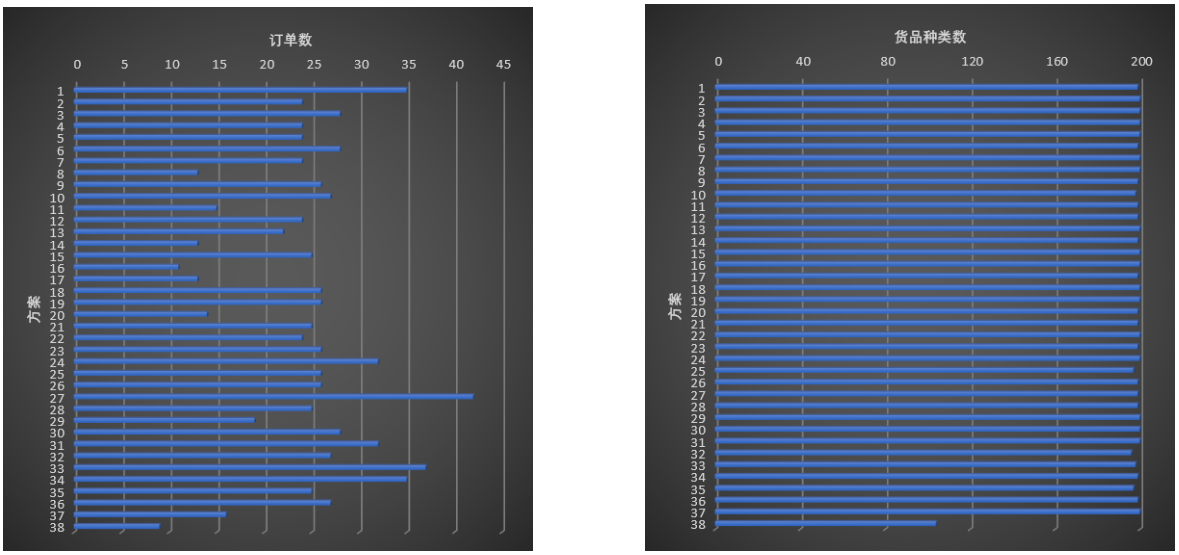

图 7　每一批订单数和货品种类数

5.2 问题二模型的建立与求解

本文通过蒙特卡罗法与遗传算法两种不同方法对货品最优摆放位置进行分析处理，通过比较结果得到更优解。

5.2.1 蒙特卡罗法

本文先采用蒙特卡罗法模拟货品随机摆放的位置，模拟多次得出拣选距离和的最小值。蒙特卡罗法每次模拟货品的摆放位置都是纯随机的，这种方法的问题在于即使模拟很多次也无法保证得到的最小拣选距离和为全局最优解，但是可行性在于通过增加模拟的次数结果也可以无限逼近全局最优解。

5.2.2 遗传算法

鉴于蒙特卡罗算法的局限性，本文考虑采用改进的遗传算法来寻找最优的货品摆放位置。不同于蒙特卡罗法，遗传算法是通过大量备选解的变换、迭代和变异，在解空间中并行动态地进行全局搜索的最优化方法，能够自我迭代寻找最优解。初始的最优拣选距离要尽可能设置的大，针对本问题本文设置为 10000。改进的遗传算法会将某次计算出的某一给定批次货在该摆放位置情况下的拣选距离与目前最短的拣选距离相比较，如该次的结果更优即这次的拣选距离更短则将变异概率 p 设置为 10%，并用这次的计算结果替代最优拣选距离，如果该次的拣选距离更大则用本次的拣选距离减去最优的拣选距离，若结果大于阈值 eps 则将变异概率 p 设置为 80%，若结果小于阈值 eps 则将变异概率 p 设置为 20%，而后让货品摆放位置进行变异，即比例 p 的货品位置发生随机变化，变化后进行下一次计算。重复这个比较过程，共重复 N_{max} 次。

比较两种方法可以发现改进的遗传算法的结果更优。如图8和9所示，展示了各批分拣任务拣选距离的总和和所有批次全部订单的拣选距离总和，分别为 147563 和 147462，具体货品的排列摆放位置见结果 result2.csv。

批次	蒙特卡罗法距离	遗传算法距离	批次	蒙特卡罗法距离	遗传算法距离
1	4015	3996	20	2349	2275
2	3825	3841	21	4106	4114
3	3814	3816	22	3927	3915
4	4001	4038	23	4314	4331
5	3751	3723	24	5567	5569
6	4202	4201	25	4360	4254
7	4096	4035	26	4256	4245
8	1478	1441	27	6779	6765
9	3920	3878	28	4364	4338
10	3800	3948	29	3306	3342
11	2083	2091	30	4613	4696
12	4140	4152	31	5558	5577
13	3537	3584	32	4469	4408
14	2119	2129	33	5215	5170
15	4255	4286	34	5712	5772
16	1925	1926	35	4214	4196
17	2133	2105	36	4667	4695
18	4243	4210	37	2616	2628
19	4537	4498	38	1297	1274
			总距离	147563	147462

图 8　蒙特卡罗法与遗传算法的结果

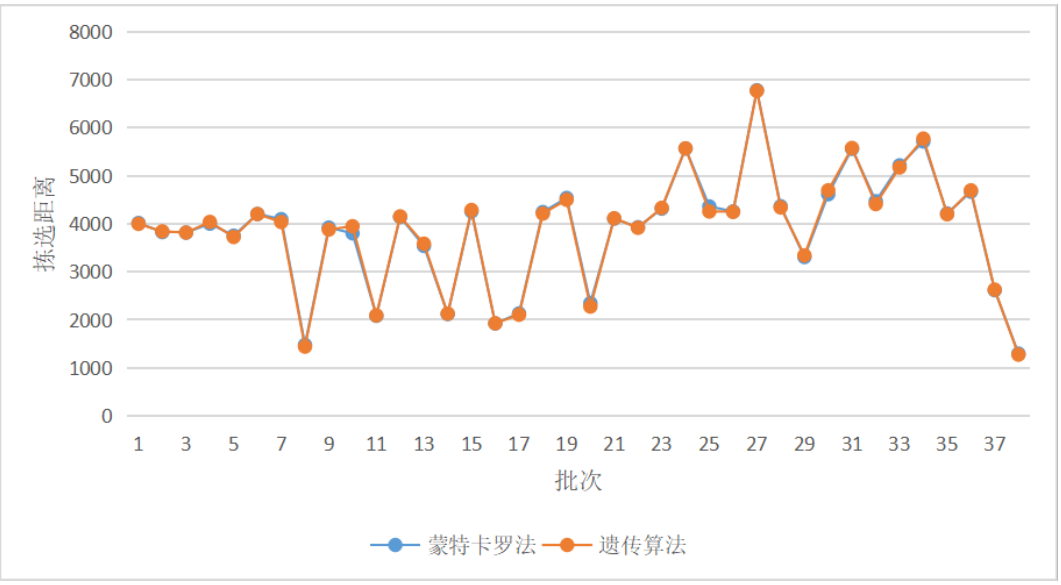

图 9　蒙特卡罗法与遗传算法结果的比较

从图中可以看出，两种方法得到的各批次拣选距离折线图大致重合，说明两种方法得到的结果基本一致，这也从另一个角度验证了本文算法的正确性和稳定性。

本文中使用的改进的遗传算法的流程图如图10所示：

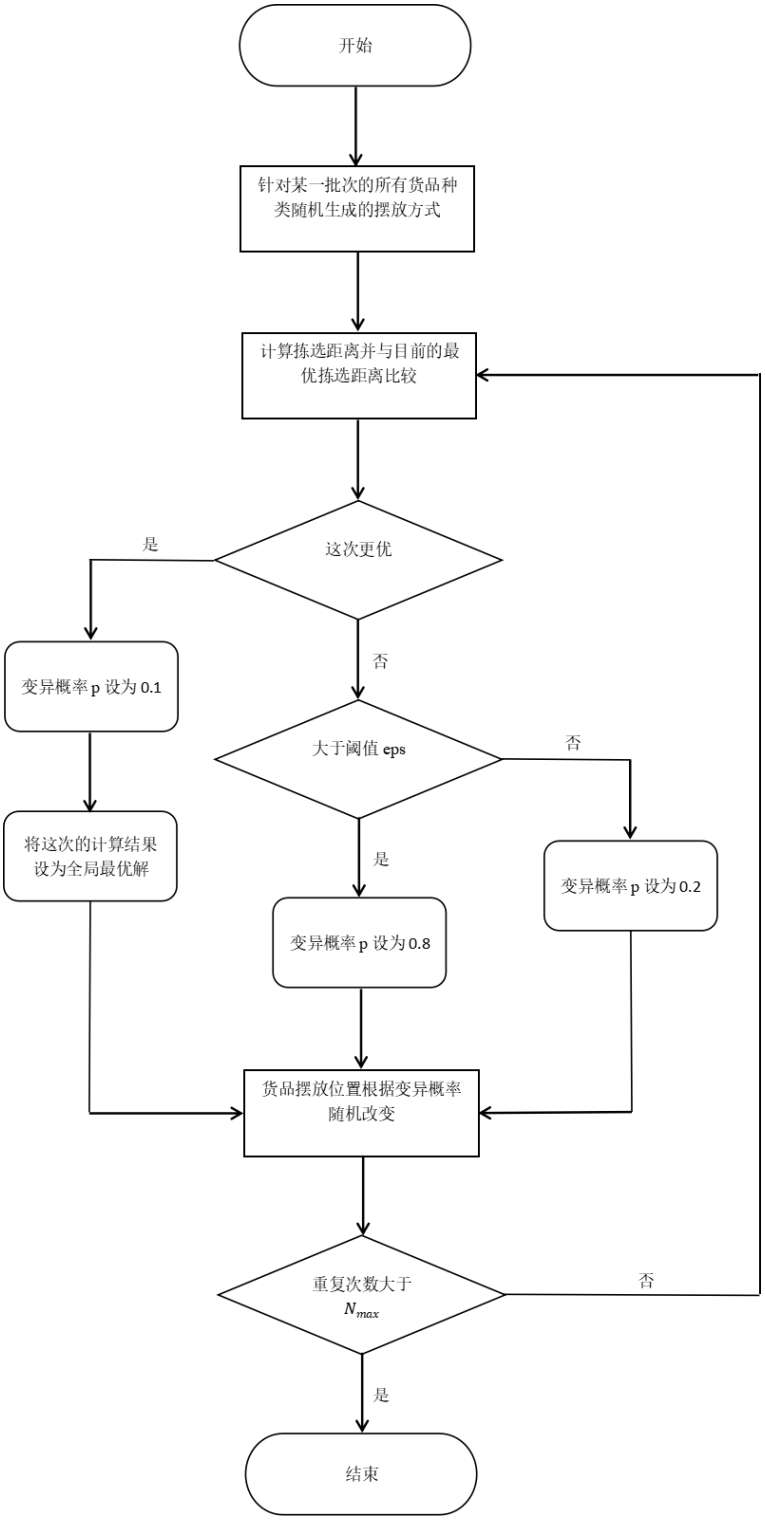

图 10　遗传算法流程图

从中可以看出该算法的自我迭代优化的功能更加强大，能够更大程度的寻找出全局最优解。

5.3 问题三模型的建立与求解

在使用蒙特卡罗随机数算法对各工人工作总距离进行最优化求解前，首先需要确立符合题设要求的自由变量以及目标函数。对于普遍情况，假设总工人数为 N，该批货物的订单总数为 n，并且每个订单对应的货品位置皆被问题二结果给出。

由题设要求，若要得到每位工人的最小拣选路程，且保证每位工人一次只能完成一笔订单，则每位工人在完成每一笔订单时需要尽可能保证拣选方向不变，从而减少回头路，达到路程最小的目的。故在考虑工人在完成每批任务时，每笔订单只需考虑两端货品位置即可。假设该批货物对应的每笔订单端点的位置序号为 $S_1, S_1^{'}, S_2, S_1^{'}...S_n, S_n^{'}$，脚标用于区分订单的两端序号，由问题二可知订单 k 两端序号满足

$$|S_k - S_k^{'}| = d(O_k) \tag{3}$$

同时，假设各工人分配到的订单数目分别为 $n_1, n_2, n_3, ...n_N$，且满足

$$n = \sum_{i=1}^{N} n_i \tag{4}$$

由上以及每位工人的初始位置，则可以得到每位工人的拣选总距离，即每位工人对应的目标函数

$$d_i = S_1 - 1 + |S_1 - S_2^{'}| + |S_2 - S_2^{'}| + |S_2 - S_3^{'}| + ... + |S_{n_i} - S_{n_i}^{'}| \tag{5}$$

如上可列出所有工人的拣选总距离，通过蒙特卡罗随机算法对所有订单顺序进行随机编排，则全局拣选时间最短对应各工人所需要耗费的最长时间，即最远路程；定义均衡度指标 Q，则全局最均衡分配对应工人所走路程的极差值，即

$$d_{ibest} = maxd_i \tag{6}$$

$$Q = 1 - \frac{maxd_i - mind_i}{maxd_i} \tag{7}$$

如此全局最优目标函数以及限制条件已全部确定，通过利用 Python 语言的蒙特卡罗随机编排以及调用问题二中批次货品摆放位置数据，得到最优化的工人任务分配结果。相应的极差，均衡度，平均均衡度以及工人路径最大优化值结果如图11所示，所有工人完成所有批次所需要的总时间为 62030，所有批次工人分配情况的均衡度平均为 0.89，因为均衡度越接近于 1，说明分配得越均衡，因此可以认为该结果是保证既均衡又耗时短的分配方案，具体分配方案见 result3.csv。

批次	时间	极差	均衡度	批次	时间	极差	均衡度
1	1756	240	0.86	20	1046	290	0.72
2	1626	158	0.9	21	1764	212	0.88
3	1612	138	0.91	22	1640	168	0.9
4	1678	136	0.92	23	1766	28	0.98
5	1560	176	0.89	24	2330	272	0.88
6	1754	148	0.92	25	1726	32	0.98
7	1718	216	0.87	26	1734	20	0.99
8	734	306	0.58	27	2778	124	0.96
9	1624	126	0.92	28	1784	148	0.92
10	1640	136	0.92	29	1408	238	0.83
11	880	112	0.87	30	1964	168	0.91
12	1736	232	0.87	31	2332	172	0.93
13	1504	88	0.94	32	1808	74	0.96
14	982	222	0.77	33	2158	146	0.93
15	1756	122	0.93	34	2350	58	0.98
16	878	142	0.84	35	1708	28	0.98
17	1008	296	0.71	36	1996	162	0.92
18	1746	56	0.97	37	1088	62	0.94
19	1840	80	0.96	38	618	222	0.64
				总用时	62030	平均均衡度	0.89

图 11　问题三结果

六、 模型的检验与灵敏度分析

为了检验本文核心结论批次分配的正确性，这里改变 N 的值对问题一的结果进行检验及灵敏度分析。为判断结果是否合理稳定，对题设限制 $N = 200$ 进行微扰处理，分别从 $N = 190$, 到 $N = 210$，使用与上文相同的算法对批次分配进行最优化求解，得到批次数随着种类限制数变化趋势结果如图12所示，可以发现批次数随着种类限制的增长呈下降趋势，但是变化的速度较为缓慢，没有较大的突变，说明该模型对 N 不敏感，比较稳定。

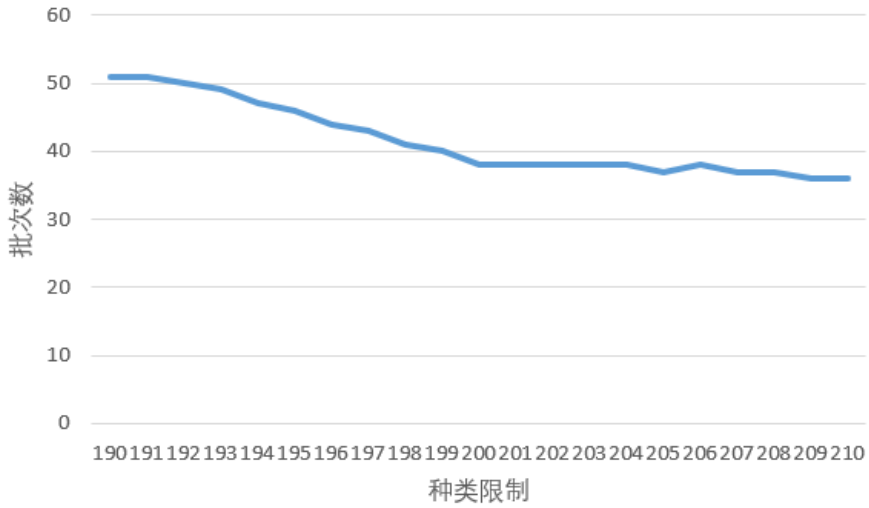

图 12　批次随 N 的变化

七、 模型评价与改进

7.1 问题一模型的评价与改进

本文多次改进的贪心算法模型最大程度上达到了“贪心”这一目的。从过程来看本文多次改进的思路越来越符合贪心的想法，从结果来看平均每一批订单所含货品种类达到了 196.71 种，如果剔除最后一批受订单数量限制的批次每一批订单所含货品种类更是高达 199.21 种，最大化地充分利用 $N = 200$ 这一条件。可见该算法模型比较好地契合了题目给出的条件，比较成功地完成了题目要求。

贪心算法作为一种简单的算法存在很大的修改与改进空间，贪心是很难有止境的。算法层面甚至可能有比这一算法更好的算法可以用来解决这道题目，这些都是可以进行进一步的改进与优化。

7.2 问题二模型的评价与改进

问题二本文一开始采用的蒙特卡罗模型，由于每次都是纯随机，在尝试很多次的情况下这种方法也只能做到无限接近最优解，而遗传算法则更有可能找出全局最优解，且不会陷入局部最优的情况。从结果来看遗传算法确实比蒙特卡罗法要更优。借助于 Python 的特性本文还将目标函数转化为另一种比较方便的求距离的方式。

对于遗传算法模型还可以进一步改进，可以通过更科学的方法来确定突变概率和阈值，比如使用函数，而不是采用定值，从而使结果更加精准可靠。

7.3 问题三模型的评价与改进

本文采用蒙特卡罗随机编排的方法对最优化情况进行求解，算法简单直观较易实现，同时进行合理假设员工等订单数均衡分配，从而进一步简化算法。但蒙特卡罗法仍然略逊色于遗传算法，对于模型的改进可选取更加准确优化的遗传算法进行求解。

除此之外，目标函数仍存在较大改进空间，对于每位工人分配到的订单数可以不加以限制，将其视为目标变量一并含入计算，虽然会导致代码量的巨幅增加，但在一定程度上也促进了优化结果的准确性和严密性。

八、 参考文献

[1] 肖际伟, 配送中心拣货系统优化, 山东大学,2010

[2] 卓成娣,ST 快递转运中心分拣流程的优化研究, 南京林业大学,2018

[3] 肖衡, 浅析贪心算法,《办公自动化》, 总第 164 期,25-26,2009.9

[4] 邹哲讷, 贪心算法及其应用,《计算机光盘软件与应用》,2015

[5] 葛继科, 邱玉辉, 吴春明, 蒲国林, 遗传算法研究综述,《计算机应用研究》, 第 25 卷第 10 期,2911-2916,2018.10

A 附录

1.1 程序代码

本文采用 Pycharm 编程

```
#1.问题一思路一
import csv

import matplotlib
import matplotlib.pyplot as plt
import numpy as np

N = 200
with open('dingdan.csv','r') as f:
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
for key in dict.keys():
dict[key] = set(dict[key])

val = []
```

```
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue
# print(dlist[0:2])
# print(len(dlist))



#分类集合
def listmin(lst,N):
Setlst = lst.copy()
leng = []
leng0 = [len(item0) for item0 in Setlst]
val_min = Setlst[leng0.index(min(leng0))]
Setlst.remove(val_min)
for set1 in Setlst:
a = val_min.union(set1)
leng.append(len(a))
index1 = leng.index(min(leng))
c = Setlst[index1]
b = c.union(val_min)
if len(b) <= N:
Setlst.append(b)
Setlst.remove(c)
else:
Setlst.append(val_min)
length = [len(item) for item in Setlst]
return Setlst,length

dlist1 = dlist.copy()
d = []
len_min = 1
while len(dlist1) >1:
```

```
val1 = dlist1.copy()
dlist1,length = listmin(val1,N)
if len(val1) == len(dlist1):
b = dlist1[length.index(min(length))]
d.append(b)
dlist1.remove(b)
d.append(dlist1[0])
le = [len(item) for item in d]
# print(len(d))
# print(le)

# 写入订单信息
dict2 = {'OrderNo':'GroupNo'}
for item in dict.items():
for i in range(len(d)):
if item[1].issubset(d[i]):
dict2[item[0]] = dict2.get(item[0],0)+i+1
break

#每批次订单数
dingdanshu = {}
del dict2['OrderNo']
for item in dict2.items():
dingdanshu[str(item[1])] = dingdanshu.get(str(item[1]),0)+1

d_order=sorted(dingdanshu.items(),key=lambda x:int(x[0]),reverse=False)
print(d_order)
lst1 = []
for i in range(len(le)):
a1 = list(d_order[i])
a1.append(le[i])
lst1.append(a1)
lst1.insert(0,['批次','订单数','商品种类数'])
# with open('pici95.csv','w',newline='',encoding='UTF-8') as f:
#     writer = csv.writer(f)
#     for row in lst1:
#         writer.writerow(row)
# #分批方案写入文件
# with open('result1.csv','w',newline='') as f:
#     writer = csv.writer(f)
#     for row in dict2.items():
#         writer.writerow(row)

#问题一思路二
N = 200
import csv
with open('dingdan.csv','r') as f:
```

```
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
for key in dict.keys():
dict[key] = set(dict[key])

val = []
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue

def finmax(lst,N):
ls = lst.copy()
ls1 = sorted(ls, key=lambda x: len(x), reverse=True)
b = ls1[0]
ls.remove(ls1[0])
for i in range(1,len(ls1)):
a = b.copy()
if len(ls1[i].union(a))>N:
b = a
continue
else:
b = ls1[i].union(a)
ls.remove(ls1[i])
return ls,b

d = []
dlist1 = dlist.copy()
while len(dlist1) >0:
val1 = dlist1.copy()
dlist1,b = finmax(val1,N)
```

```
d.append(b)
le = [len(item) for item in d]
print(len(d))
# print(leng)



# 写入订单信息
dict2 = {'OrderNo':'GroupNo'}
for item in dict.items():
for i in range(len(d)):
if item[1].issubset(d[i]):
dict2[item[0]] = dict2.get(item[0],0)+i+1
break

# #每批次订单数
# dingdanshu = {}
# del dict2['OrderNo']
# for item in dict2.items():
#     dingdanshu[str(item[1])] = dingdanshu.get(str(item[1]),0)+1
#
# d_order=sorted(dingdanshu.items(),key=lambda x:int(x[0]),reverse=False)
# # # print(d_order)
# lst1 = []
# for i in range(len(le)):
#     a1 = list(d_order[i])
#     a1.append(le[i])
#     lst1.append(a1)
# lst1.insert(0,['批次','订单数','商品种类数'])
#
# with open('pici68.csv','w',newline='',encoding='UTF-8') as f:
#     writer = csv.writer(f)
#     for row in lst1:
#         writer.writerow(row)



#分批方案写入文件
# with open('result1.csv','w',newline='') as f:
#     writer = csv.writer(f)
#     for row in dict2.items():
#         writer.writerow(row)
#问题一思路三
N = 200
with open('dingdan.csv','r') as f:
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
```

```
for key in dict.keys():
dict[key] = set(dict[key])

#获得只含有商品种类的集合
val = []
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue
# print(len(dlist))
def finmax(lst,N):
ls = lst.copy()
ls1 = sorted(ls, key=lambda x: len(x), reverse=True)
b = ls1[0]
ls.remove(ls1[0])
while len(b)<=N:
ls2 = [len(b.union(x)) for x in ls]
if len(ls2)>0:
if min(ls2)<=N:
index = ls2.index(min(ls2))
b = b.union(ls[index])
ls.remove(ls[index])
else:
break
else:
break
return ls,b

#d中包含分批次的商品种类信息
d = []
dlist1 = dlist.copy()
while len(dlist1) >0:
val1 = dlist1.copy()
```

```
dlist1,b = finmax(val1,N)
d.append(b)
le = [len(item) for item in d]
print(len(d))
print(le)

# 写入订单信息
# dict2 = {'OrderNo':'GroupNo'}
# for item in dict.items():
#     for i in range(len(d)):
#         if item[1].issubset(d[i]):
#             dict2[item[0]] = dict2.get(item[0],0)+i+1
#             break
#
#分批方案写入文件
# with open('result1.csv','w',newline='') as f:
#     writer = csv.writer(f)
#     for row in dict2.items():
#         writer.writerow(row)



# #每批次订单数
# dingdanshu = {}
# del dict2['OrderNo']
# for item in dict2.items():
#     dingdanshu[str(item[1])] = dingdanshu.get(str(item[1]),0)+1
#
# d_order=sorted(dingdanshu.items(),key=lambda x:int(x[0]),reverse=False)
# # print(d_order)
# lst1 = []
# for i in range(len(le)):
#     a1 = list(d_order[i])
#     a1.append(le[i])
#     lst1.append(a1)
# lst1.insert(0,['批次','订单数','商品种类数'])
# with open('pici38.csv','w',newline='',encoding='UTF-8') as f:
#     writer = csv.writer(f)
#     for row in lst1:
#         writer.writerow(row)
#
#问题一思路四
import numpy as np
import csv

N = 200
with open('dingdan.csv','r') as f:
```

```
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
for key in dict.keys():
dict[key] = set(dict[key])

#获得只含有商品种类的集合
val = []
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue
# print(len(dlist))
def finmax(lst,N):
ls = lst.copy()
ls1 = sorted(ls, key=lambda x: len(x), reverse=True)
b = ls1[0]
ls.remove(ls1[0])
while len(b)<=N:
ls2 = [len(b.intersection(x)) for x in ls]
if len(ls2)>0:
a = max(ls2)
index = ls2.index(a)
c = b.union(ls[index])
if len(c)<=N:
b = b.union(ls[index])
ls.remove(ls[index])
else:
ls3 = [len(b.union(x)) for x in ls]
if min(ls3)<=N:
index = ls3.index(min(ls3))
b = b.union(ls[index])
```

```
ls.remove(ls[index])
else:
break
else:
break
return ls,b

#d中包含分批次的商品种类信息
d = []
dlist1 = dlist.copy()
while len(dlist1) >0:
val1 = dlist1.copy()
dlist1,b = finmax(val1,N)
d.append(b)
le = [len(item) for item in d]
print(len(d))
print(le)

# 写入订单信息
dict2 = {'OrderNo':'GroupNo'}
for item in dict.items():
for i in range(len(d)):
if item[1].issubset(d[i]):
dict2[item[0]] = dict2.get(item[0],0)+i+1
break

# 分批方案写入文件
with open('result1.csv','w',newline='') as f:
writer = csv.writer(f)
for row in dict2.items():
writer.writerow(row)



#每批次订单数
dingdanshu = {}
del dict2['OrderNo']
for item in dict2.items():
dingdanshu[str(item[1])] = dingdanshu.get(str(item[1]),0)+1

d_order=sorted(dingdanshu.items(),key=lambda x:int(x[0]),reverse=False)
# print(d_order)
lst1 = []
for i in range(len(le)):
a1 = list(d_order[i])
a1.append(le[i])
lst1.append(a1)
```

```
lst1.insert(0,['批次','订单数','商品种类数'])
with open('pici38.csv','w',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
for row in lst1:
writer.writerow(row)

#2.问题二
import random
import numpy as np
import csv

N = 200
with open('dingdan.csv','r') as f:
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
for key in dict.keys():
dict[key] = set(dict[key])

#获得只含有商品种类的集合
val = []
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue
# print(len(dlist))
def finmax(lst,N):
ls = lst.copy()
ls1 = sorted(ls, key=lambda x: len(x), reverse=True)
b = ls1[0]
ls.remove(ls1[0])
while len(b)<=N:
```

```
ls2 = [len(b.intersection(x)) for x in ls]
if len(ls2)>0:
a = max(ls2)
index = ls2.index(a)
c = b.union(ls[index])
if len(c)<=N:
b = b.union(ls[index])
ls.remove(ls[index])
else:
ls3 = [len(b.union(x)) for x in ls]
if min(ls3)<=N:
index = ls3.index(min(ls3))
b = b.union(ls[index])
ls.remove(ls[index])
else:
break
else:
break
return ls,b

#d中包含分批次的商品种类信息
d = []
dlist1 = dlist.copy()
while len(dlist1) >0:
val1 = dlist1.copy()
dlist1,b = finmax(val1,N)
d.append(b)
le = [len(item) for item in d]
# print(len(d))
# print(le)

# 写入订单信息
dict2 = {'OrderNo':'GroupNo'}
for item in dict.items():
for i in range(len(d)):
if item[1].issubset(d[i]):
dict2[item[0]] = dict2.get(item[0],0)+i+1
break
#计算距离
def Distance(dict,d,dict2,n):
dis = 10000
setn = list(d[n-1])                          #找到第n批次货品种类,并转化为列表
rang = list(range(1,201))
set_res = []
for i in range(2000):
dis0 = 0
shunxu = random.sample(rang,len(setn))
```

```
set_current = list(zip(setn,shunxu))                #生成货物排列顺序
ddan = [k for k,v in dict2.items() if v==n] #找到第n批次对应的订单信息
for dd in ddan:
set1 = dict[dd]
index1 = [x[1] for x in set_current if x[0] in set1]
dis0 = dis0 + max(index1)-min(index1)
if dis0<dis:
dis = dis0
set_res = set_current
return set_res,dis
distance = []
for n in range(1,len(d)+1):
set_res,dis = Distance(dict,d,dict2,n)
distance.append([n,dis])
print(distance)
# with open('result2mengtekaluo.csv','w',newline='',encoding='UTF-8') as f:
#     writer = csv.writer(f)
#     writer.writerow(['批次','距离'])
#     for row in distance:
#         writer.writerow(row)
#遗传算法
import random
import numpy as np
import csv
N = 200
with open('dingdan.csv','r') as f:
lis = [item.strip().split(',') for item in f.readlines() ]
dict = {}
for item in lis[1:]:
dict[item[0]] = dict.get(item[0],[])+[item[1]]
for key in dict.keys():
dict[key] = set(dict[key])

#获得只含有商品种类的集合
val = []
for value in dict.values():
val.append(value)

#合并同类集合
dlist = val.copy()
for i in range(len(val)-1):
for j in range(i+1,len(val)):
if val[i].issubset(val[j]):
if val[i] in dlist:
dlist.remove(val[i])
else:
continue
```

```
continue
elif val[j].issubset(val[i]):
if val[j] in dlist:
dlist.remove(val[j])
else:
continue
# print(len(dlist))
def finmax(lst,N):
ls = lst.copy()
ls1 = sorted(ls, key=lambda x: len(x), reverse=True)
b = ls1[0]
ls.remove(ls1[0])
while len(b)<=N:
ls2 = [len(b.intersection(x)) for x in ls]
if len(ls2)>0:
a = max(ls2)
index = ls2.index(a)
c = b.union(ls[index])
if len(c)<=N:
b = b.union(ls[index])
ls.remove(ls[index])
else:
ls3 = [len(b.union(x)) for x in ls]
if min(ls3)<=N:
index = ls3.index(min(ls3))
b = b.union(ls[index])
ls.remove(ls[index])
else:
break
else:
break
return ls,b

#d中包含分批次的商品种类信息
d = []
dlist1 = dlist.copy()
while len(dlist1) >0:
val1 = dlist1.copy()
dlist1,b = finmax(val1,N)
d.append(b)
le = [len(item) for item in d]
# print(len(d))
# print(le)

# 写入订单信息
dict2 = {'OrderNo':'GroupNo'}
for item in dict.items():
```

```
for i in range(len(d)):
if item[1].issubset(d[i]):
dict2[item[0]] = dict2.get(item[0],0)+i+1
break
#计算距离
def Distance(dict,d,dict2,n):
alpha = -0.1
dis = 10000
setn = list(d[n-1])                          #找到第n批次货品种类,并转化为列表
rang = list(range(1,201))
set_res = []
for i in range(2000):
dis0 = 0
shunxu = random.sample(rang,len(setn))
set_current = list(zip(setn,shunxu))                #生成货物排列顺序
ddan = [k for k,v in dict2.items() if v==n] #找到第n批次对应的订单信息
for dd in ddan:
set1 = dict[dd]
index1 = [x[1] for x in set_current if x[0] in set1]
dis0 = dis0 + max(index1)-min(index1)
if dis0<=dis:
dis = dis0
set_res = set_current
p = 0.4*np.exp(alpha*i)
elif dis-dis0>30:
p = 0.8*(1-np.exp(alpha*i))
else:
p = 0.3*(1-np.exp(alpha*i))
a = int(np.ceil(p*len(shunxu)))
bb = random.sample(shunxu,a)
index11 = [shunxu.index(x) for x in bb]
chan = (set(rang)-set(shunxu))|set(bb)
cc = random.sample(list(chan),a)
for i in range(len(cc)):
shunxu[index11[i]] = cc[i]

return set_res,dis



distance = []
list_res2 = []
for n in range(1,1+len(d)):
set_res,dis = Distance(dict,d,dict2,n)
distance.append([n,dis])
list_res = [list(x) for x in set_res]
[x.insert(1,n) for x in list_res]
list_res2.extend(list_res)
```

```
# 写入文件
with open('result2.csv','w',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
writer.writerow(['ItemNo','GroupNo','ShelfNo'])
for row in list_res2:
writer.writerow(row)
with open('result2distanceyichaun.csv','w',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
writer.writerow(['批次','距离'])
for row in distance:
writer.writerow(row)
#3.问题三
import random
import numpy as np
import csv
#得到订单商品查询字典，key为订单，value为商品
with open('dingdan.csv','r',encoding='utf-8') as f:
lis = [item.strip().split(',') for item in f.readlines() ]
dict_dingdan = {}
for item in lis[1:]:
dict_dingdan[item[0]] = dict_dingdan.get(item[0],[])+[item[1]]

#得到批次货品及其位置信息查询字典，key为批次，value为[货物，位置]
with open('result2.csv','r') as f:
lis1 = [item.strip().split(',') for item in f.readlines() ]
dict_pici = {}
for item in lis1[1:]:
dict_pici[str(item[1])] = dict_pici.get(str(item[1]), []) + [[item[0],item[2]]]

#得到订单批次信息查询字典，key为批次，value为订单信息
with open('result1.csv','r') as f:
lis2 = [item.strip().split(',') for item in f.readlines() ]
dict_pidingdan = {}
for item in lis2[1:]:
dict_pidingdan[str(item[1])] = dict_pidingdan.get(str(item[1]),[])+[item[0]]

#写入文件
with open('result3.csv','a',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
writer.writerow(['OrderNo','GroupNo','WorkerNo','TaskNo'])
#n为批次
time = []
for n in range(1,39):
a = dict_pici[str(n)]
dict_place = dict(a) #该批次的位置字典信息,key为商品,value为位置
b_dingdan = dict_pidingdan[str(n)] #该批次的订单
le = len(b_dingdan)  #订单总数
```

```
rang = list(range(le))
d = 10000
for j in range(2000):
shunxu = random.sample(rang,le)
#第一个人
num_1 = list(range(0,le,5))
index_1 = [shunxu[i] for i in num_1]
dingdan_1 = [b_dingdan[i] for i in index_1]
place_1 = []
for item in dingdan_1:
pl = [ int(dict_place[x]) for x in dict_dingdan[item]]
pl.sort()
place_1.append(pl)
d_1 = place_1[0][-1]-1
for i in range(1,len(place_1)):
d_1 = d_1+np.abs(place_1[i][0]-place_1[i-1][-1])+place_1[i][-1]-place_1[i][0]
d_1 = d_1 + place_1[-1][-1]-1
#第二个人
num_2 = list(range(1,le,5))
index_2 = [shunxu[i] for i in num_2]
dingdan_2 = [b_dingdan[i] for i in index_2]
place_2 = []
for item in dingdan_2:
pl = [ int(dict_place[x]) for x in dict_dingdan[item]]
pl.sort()
place_2.append(pl)
d_2 = place_2[0][-1]-1
for i in range(1,len(place_2)):
d_2 = d_2+np.abs(place_2[i][0]-place_2[i-1][-1])+place_2[i][-1]-place_2[i][0]
d_2 = d_2 + place_2[-1][-1]-1
#第三个人
num_3 = list(range(2,le,5))
index_3 = [shunxu[i] for i in num_3]
dingdan_3 = [b_dingdan[i] for i in index_3]
place_3 = []
for item in dingdan_3:
pl = [ int(dict_place[x]) for x in dict_dingdan[item]]
pl.sort()
place_3.append(pl)
d_3 = place_3[0][-1]-1
for i in range(1,len(place_3)):
d_3 = d_3+np.abs(place_3[i][0]-place_3[i-1][-1])+place_3[i][-1]-place_3[i][0]
d_3 = d_3 + place_3[-1][-1]-1
#第四个人
num_4 = list(range(3,le,5))
index_4 = [shunxu[i] for i in num_4]
dingdan_4 = [b_dingdan[i] for i in index_4]
```

```
place_4 = []
for item in dingdan_4:
pl = [ int(dict_place[x]) for x in dict_dingdan[item]]
pl.sort()
place_4.append(pl)
d_4 = place_4[0][-1]-1
for i in range(1,len(place_4)):
d_4 = d_4+np.abs(place_4[i][0]-place_4[i-1][-1])+place_4[i][-1]-place_4[i][0]
d_4 = d_4 + place_4[-1][-1]-1
#第五个人
num_5 = list(range(4,le,5))
index_5 = [shunxu[i] for i in num_5]
dingdan_5 = [b_dingdan[i] for i in index_5]
place_5 = []
for item in dingdan_5:
pl = [ int(dict_place[x]) for x in dict_dingdan[item]]
pl.sort()
place_5.append(pl)
d_5 = place_5[0][-1]-1
for i in range(1,len(place_5)):
d_5 = d_5+np.abs(place_5[i][0]-place_5[i-1][-1])+place_5[i][-1]-place_5[i][0]
d_5 = d_5 + place_5[-1][-1]-1
d0 = max(d_1,d_2,d_3,d_4,d_5)
dmin = min(d_1, d_2, d_3, d_4, d_5)
if d0<d and d0-dmin<500:
d = d0
dmi = d0-dmin
fenpei = []
fenpei.append(dingdan_1)
fenpei.append(dingdan_2)
fenpei.append(dingdan_3)
fenpei.append(dingdan_4)
fenpei.append(dingdan_5)
res = []
for i in range(len(fenpei)):
for j in range(len(fenpei[i])):
res.append([fenpei[i][j],n,i+1,j+1])
time.append([n,d,dmi])
with open('result3.csv','a',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
for row in res:
writer.writerow(row)

with open('workertime.csv','w',newline='',encoding='UTF-8') as f:
writer = csv.writer(f)
writer.writerow(['批次','用时','均衡度'])
for row in time:
```

```
writer.writerow(row)
```

1.2 支撑材料的文件列表

1. 问题一分批方案 result1.csv
2. 问题二货品摆放数据 result2.csv
3. 问题三分配方案 result3.csv
4. 分批次 95 的方案 pici95.csv
5. 分批次 68 的方案 pici68.csv
6. 分批次 38 的方案 pici38.csv
7. 问题二蒙特卡洛每批次的距离 result2mengtekaluo.csv
8. 问题二遗传算法 result2yichuan.csv
9. 问题三工人每批次所花时间数据 workertime.csv
10. 检验数据 jianyan.xlsx