队伍编号	201754
题号	C

仓内拣货优化问题

摘 要

本文通过建立不同情况下的出库总时间最小模型、非线性规划模型以及蒙特卡洛模拟算法，对仓内拣货的任务单分配、拣货顺序和复核台选择等问题进行了分析研究，并给出科学合理的解决办法。

针对问题一：首先，复核台之间距离计算方式在已知中已给出，而复核台靠近货架的三条边都可以接货，转化成一个求两个元素之间最短距离的问题。分别对货格之间距离、货格与复核台之间距离，通过画图归纳特征，进一步细分种类，从而得出准确且简便的距离计算方法。

针对问题二：影响该订单出库时间的因素主要有货格的访问顺序和选择交付的复核台，基于两个因素建立总目标最优的模型。用蒙特卡洛模拟随机生成货格访问顺序，再对该顺序中最后一个货格通过最短路的方法选择交付的复核台，以此循环，以达最优。

针对问题三：将任务尽快出库的目标，量化为 5 个任务单出库总时间最小化，分析方法同第一问，可以看出影响该时间的主要因素是任务安排次序、货格访问路线、各任务单交付复核台的选择。

针对问题四：针对该问题建立出库时间最小化模型，求出完成出库最短花费的时间和各复核台的利用率。考虑到等待时间的出现，对比从该点到各复核台直到开始复核打包的时间，然后选取时间最短的复核台进行交付，以此循环，直到所有任务单完成。

针对问题五：考虑到复核台根据位置不同，其带给拣货过程的影响效果也有积极和消极之分，所以想到计算问题四中任务单的平均坐标，按照到该平均坐标的距离将剩余的复核台排序，分别增加距离最大的和距离最小的复核台进行评估，并进行结果的对比。

针对问题六：在前几问的假设基础上增加假设：同一个货格前只能有一名拣货员进行拣货作业，其他人需要等待。筛选出任务单中出现次数最多的商品，将其随机安排到仓库某些位置，分别求其出库时间，通过结果比对，给出仓内商品摆放的合理性意见。

关键词：非线性规划　蒙特卡洛模拟算法　出库总时间最小化模型　最短路径

目录

一、问题重述

在电子商务仓库的各项内部作业中，拣货作业是一项繁琐但重要的工作，占据了大量人力和时间，仓内拣货优化成为企业亟待解决的问题。某电商公司客户订单下达仓库后，商品开始下架出库，出库主要包含定位、组单、拣货、复核和打包 5 个流程。

题目要求针对下面 5 种具体情况：

（1）仓库有多个货架，每个货架有多个货格，商品摆放在货格中，且每个货格最多摆放一种商品，商品可以摆放在多个货格。

（2）多个订单合并成一个任务单，拣货员每次只能拣一个任务单的商品，拣货完成后，拣货员将拣货车放到复核台采能领取下一个任务单。

（3）当绕障碍物折线行走时横向和竖向偏移都取 d=750mm。

（4）复核台之间距离简化为两复核台坐标差的绝对值之和。

（5）货格与复核台距离简化为货格中点到复核台最近一条边中点的距离。

完成以下任务：

问题 1：按照图中距离标示，设计一种计算 3000 个货格和 13 个复核台总共 3013 个元素之间距离的方法。

问题 2：所有复核台正常工作，任务单 T0001 等待拣货，拣货员 P 在复核台 FH10 领取了任务单 T0001 的情况下，给 P 规划理想的拣货路线并计算完成出库花费的时间。

问题 3：复核台 FH03、FH11 正常工作，任务单 T0002-T0006 等待拣货，拣货员 P 在 FH03 领取任务单的情况下，通过建模和优化，给出 P 理想的任务领取顺序和拣货路线，并计算完成出库需要花费的时间和每个复核台利用率。

问题 4：复核台 FH01，FH03，FH10，FH12 正常工作，49 个任务单（T0001-T0049）等待拣货，9 个拣货员（P1-P9）负责拣货，给每个拣货员分配任务单、起始拣货复核台，并分别规划理想的拣货路线，使得 49 个任务单尽快完成出库，并计算完成出库需要花费的时间和每个复核台利用率。

问题 5：在问题 4 的基础上，评估增加一个正常工作的复核台对出库时间的影响。

问题 6：商品在货架中的摆放位置，会影响拣货效率。针对仓内商品摆放问题，提出合理性建议。

二、模型的假设

假设 1：复核台在复核打包时拣货员可以同时领取任务单，且领取时间不计。

假设 2：拣货员绕障碍物折线行走时横向和竖向偏移都取 d=750mm。

假设 3：拣货员行走速度 v 保持不变，v= 1.5m/s 。

假设 4：拣货员负责多个任务单时，每次只拣一个任务单的货品。

假设 5：货格与复核台距离简化为货格中点到复核台最近一条边中点的距离，复核台之间距离简化为两复核台坐标差的绝对值之和。

假设 6：多人同时在一个货格拣货不考虑等待的时间，多人同时到达一个复核台需要考虑等待时间，且视为来到该复核台的订单等待前一订单完成的时间。

三、符号说明

表 1　　符号说明

符号	说明	单位
u	复核台编号	1
z	任务单编号	1
v	拣货员行走速度	m/s
Z′	任务单 z 拣货时的起始货格	1
Z′′	任务单 z 拣货时的最终货格	1
q_z	任务单 z 中包含的订单数量	个
t_{z_1}	任意任务单的等待时间	S
t_i	在货格 i 处的下架时间	S
x_i	货格 i 处需要拣货的数量	个
Z	需要完成的任务单集合	1
$i.j$	任意货格编号	1
Du	复核台 u 复核打包的任务单集合	1
H_Z	任务单 z 经过的复核台集合	1
$t'_{z_1,u}$	任意任务单 z_1 到达复核台 u 处的时刻	S
$t''_{z_2,u}$	任意任务单 z_2 在 u 复核台的复核完成时刻	S

四、问题分析

4.1 问题一分析

问题一要求计算 3000 个货格和 13 个复核台总共 3013 个元素之间的距离。

根据已知情况，这些距离可以分为三类来分别求解：复核台之间的距离、货格之间的距离以及货格与复核台之间的距离。首先，复核台之间距离计算方式在已知中已给出，而计算剩余两种距离时考虑到，每个复核台靠近货架的三条边都可以接货，所以该问题就转化成一个求两个元素之间最短距离的问题。随后，考虑到货格、复核台的位置关系复杂多样且数量众多，单纯选择遍历所有组合的方法来解决此问题，会导致算法效率过低甚至不能达到期望的目标。所以我们考虑分别对货格之间距离、货格与复核台之间距离，通过画图归纳特征，进一步细分种类，从而得出准确且简便的距离计算方法。解决问题一的思路流程图如下：

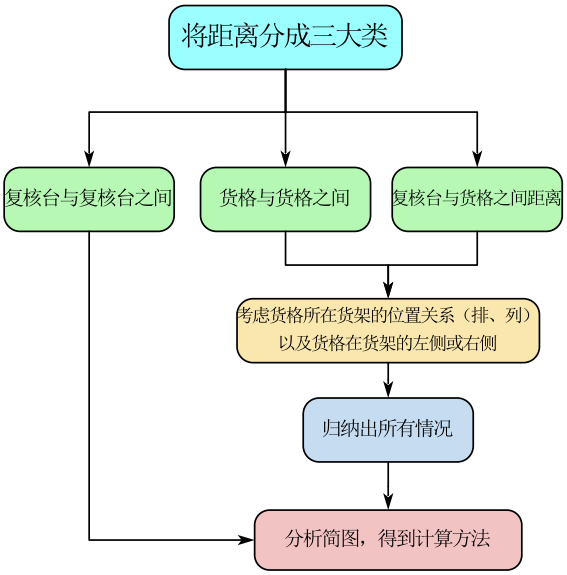

图 1 问题一的流程图

4.2 问题二的分析

问题二要求针对所有复核台正常工作、只有一个任务单、只有一个拣货员且出发拣货台确定的情况下，给该拣货员规划理想的拣货路线并计算完成出库花费的时间。

据问题分析可得，此问题针对的是拣货问题中的一般情况，因为初始复核台已给出，

又有任务单出库时间包含行走时间（货格间行走时间和复核台与货格间行走时间）、商品下架时间以及任务单复核时间，排除固定可求的时间外，可以看出影响该订单出库时间的因素主要有货格的访问顺序和选择交付的复核台。分别优化这两个因素，就可以达到总目标最优的目的。考虑到货格访问顺序繁杂度以及任务单一包含货格数量较少的情况，可以用蒙特卡洛模拟随机生成货格访问顺序，再对该顺序中最后一个货格通过最短路的方法选择交付的复核台，以此循环，以达最优。解决问题二的思路流程图如下：

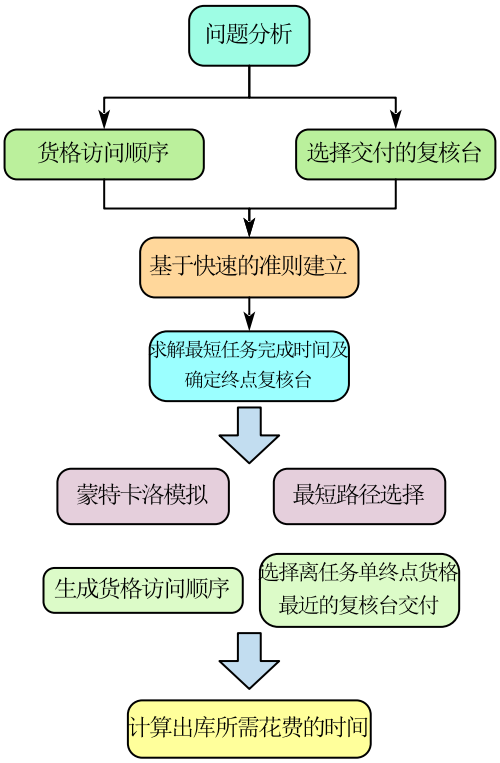

图 2 问题二的流程图

4.3 问题三的分析

问题三要求针对 2 个复核台正常工作、5 个任务单需要拣货、只有一个拣货员且出发拣货台确定的情况下，给该拣货员规划理想的任务领取顺序和拣货路线，使得这些任务尽快出库，并计算完成出库花费的时间和每个复核台利用率。

首先，将任务尽快出库的目标，量化为 5 个任务单出库总时间最小化，分析方法同第一问，可以看出影响该时间的主要因素是任务安排次序、货格访问路线、各任务单交付复核台的选择。通过表示出五个任务单行走时间、商品下架时间、任务单复核打包时间之和，建立出库时间最小化模型，从而求解出最短的时间和每个复核台利用率，反推

出任务领取顺序和拣货路线。将各部分时间分块，考虑到单独求解的简易程度，仍然可以选择复杂度低的蒙特卡洛模拟法。解决问题三的思路流程图如下：

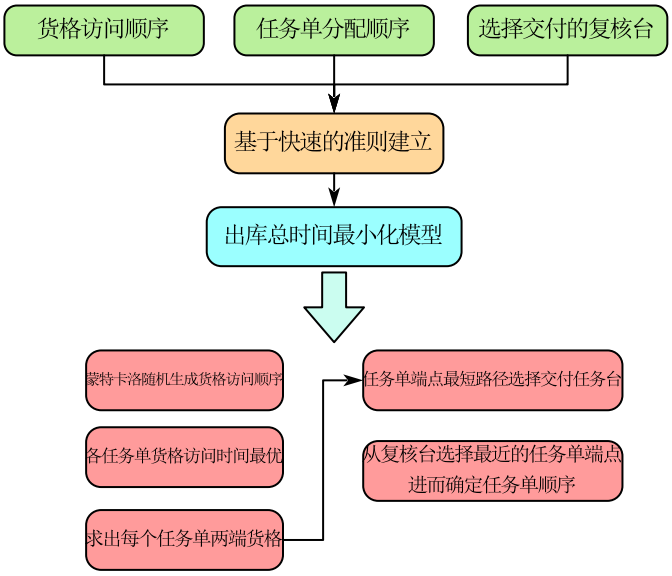

图 3 问题三的流程图

4.4 问题四的分析

问题四要求针对 4 个复核台正常工作、49 个任务单需要拣货、9 个拣货员同时工作且出发拣货台均不确定的情况下，给每位拣货员分配起始拣货台、任务单，并分别规划理想的拣货路线，使得这些任务尽快出库，并计算完成出库花费的时间和每个复核台利用率。

该问题的变化主要是拣货员的数量增加了，根据已知，拣货员在复核台交付完拣货车后立即领取新的任务单开始新的任务，但对于任务单来说，它的复核打包必须等到该复核台处于空闲状态才可以开始，即任务单的出库时间新增加了需要考虑的等待时间。同样，针对该问题建立出库时间最小化模型，求出完成出库最短花费的时间和各复核台的利用率。考虑到等待时间的出现，那么在某任务单终点货格处选择合适的复核台交付时，需要对比从该点到各复核台直到开始复核打包的时间，然后选取时间最短的复核台进行交付，以此循环，直到所有任务单完成。

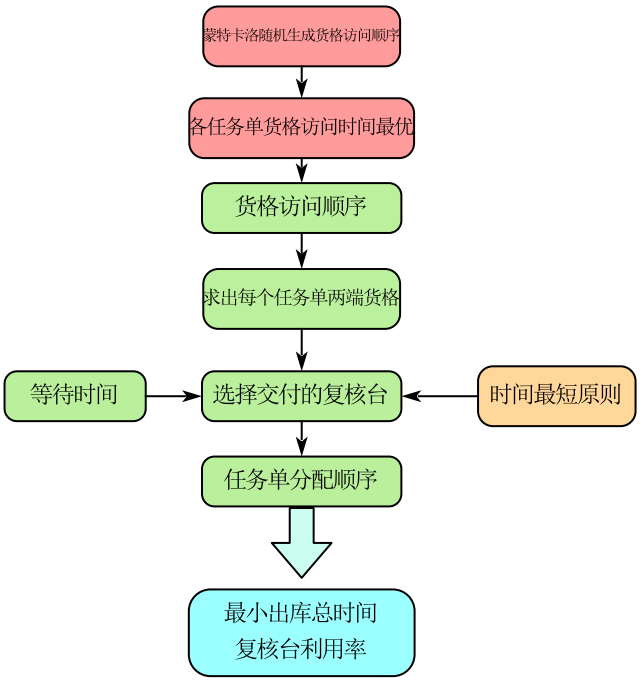

图 4 问题四的流程图

4.5 问题五的分析

问题五要求在问题 4 的基础上，评估增加一个正常工作的复核台对出库时间的影响。

考虑到复核台根据位置不同，其带给拣货过程的影响效果也有积极和消极之分，所以想到计算问题四中任务单的平均坐标，按照到该平均坐标的距离将剩余的复核台排序，分别增加距离最大的和距离最小的复核台进行评估，并进行结果的对比。

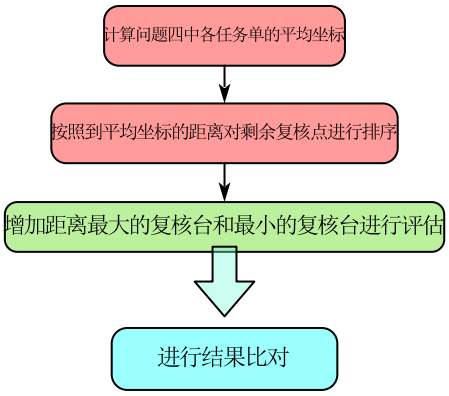

图 5 问题五的流程图

4.6 问题六的分析

问题六要求针对仓内商品摆放问题，提出合理性建议。

若将畅销品放置在离复核台较近的位置，拣货员行走距离相应减少，但畅销品所在货架可能拥挤，反而降低拣货效率。前五问都是建立在不考虑拣货时拥挤的前提下进行的，但实际生产生活中，拣货车的体积不能忽略而且多个拣货员不能同时在一个货格进行拣货作业，因此我们在前几问的假设基础上增加假设：同一个货格前只能有一名拣货员进行拣货作业，其他人需要等待。我们再筛选出任务单中出现次数最多的商品，将其随机安排到仓库某些位置，分别求其时间，通过结果比对，给出仓内商品摆放的合理性意见。

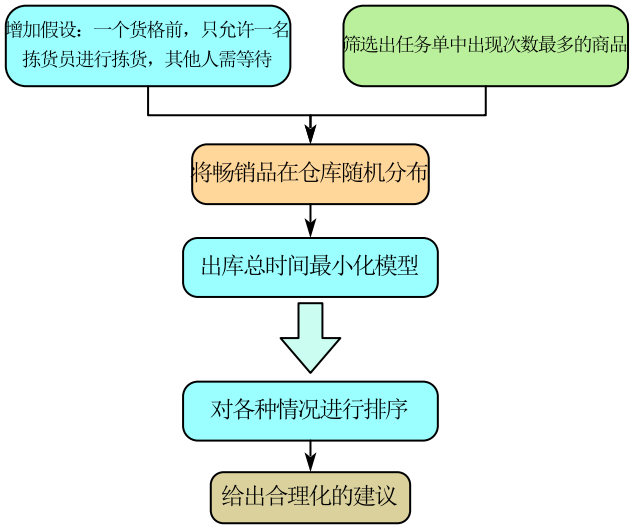

图 6 问题六的流程图

五、模型的建立与求解

5.1 问题一的求解

5.1.1 复核台与复核台之间距离

根据题目中的简化关系，我们可以直接给出复核台与复核台之间的计算方法和示意图：

$$L_{ij} = |x_j - x_i| + |y_j - y_i|$$

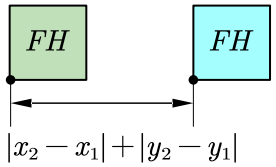

图 7 复核台与复核台之间距离示意

5.1.2 货格与货格之间距离

考虑到货架的位置关系（排和列）会对距离的求解产生不同的距离重复，所以将所有货格之间距离列出分类，经过归纳后得到四种分类方式，然后根据固定距离和距离与编号关系等确定简洁的距离计算公式。

①左左右右—货架同列（同排、不同排），货架不同列不同排

$$L_{ij}=1500+|x_j-x_i|+|y_j-y_i|$$

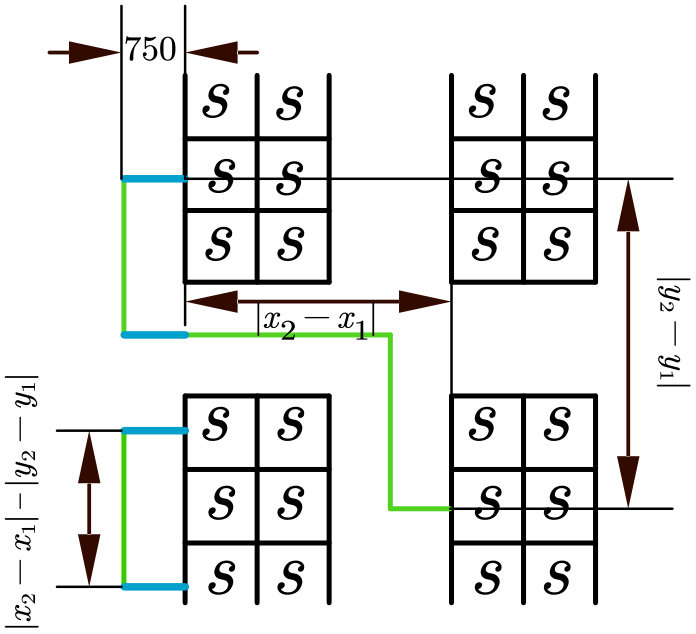

图 8 一号示意图

②左左右右—货架不同列同排

$$L_{ij}=|x_j-x_i|+|y_j-y_i|+3800+min[\ (i-1+j-1),(15-i+15-j)]*800$$

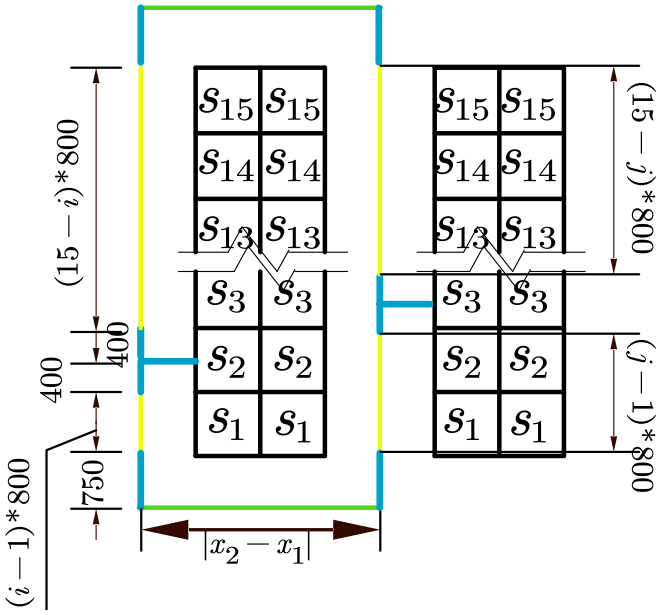

图 9 二号示意图

③左右右左—货架不同列（同排、不同排）

$$L_{ij}=|x_j-x_i|+|y_j-y_i|$$

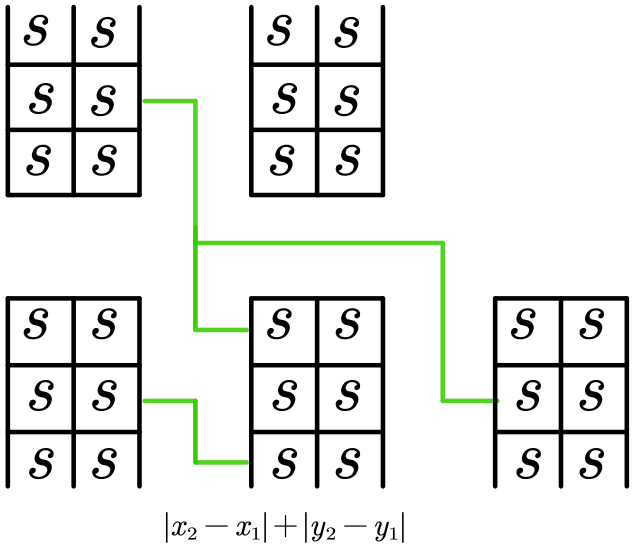

图 10 三号示意图

④左右右左—货架同列（同排、不同排）

$$L_{ij}=|x_j-x_i|+|y_j-y_i|+9800+min[\ (i-1+j-1),(15-i+15-j)]*800$$

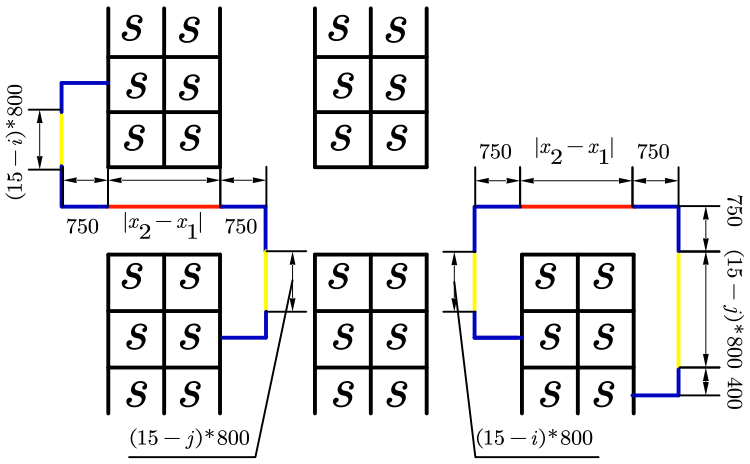

图 11 四号示意图

5.1.3 复核台与货格之间距离

考虑到复核台与货格的排与列位置关系会对距离重复造成一定的影响，综合复核台分为左侧和下侧两类，首先对货架进行分区，然后分区划进行分析和求解。

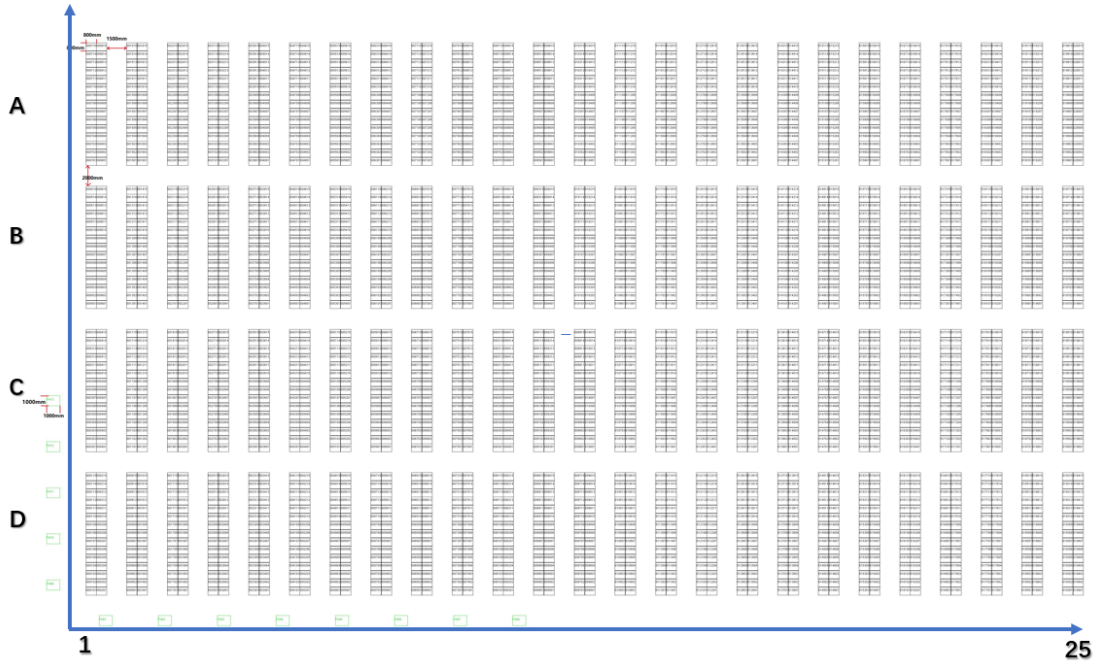

图 12 货架区划图

对于左侧货格，首先会发现有几个固定的距离，根据题目中给出的 L5 的含义，为左侧 FH09-13 每个复核台都定义一个固定长度的距离 2000mm：FH09-S00102、FH10-S00107、FH11-S00113、FH12-S00301 和 FH13-S00307。

代坐标原则：通过分析计算可以看出，无论行走路径，到左侧复核台 FH09-13 最近的点都在该复核台的左边中点处，我们把 FH09-13 左边中点坐标定义为该复核台的代坐标，对于 FH01-08 复核台的上边中点坐标同理。

对于下侧的复核台，要求各复核台到各货格的最短距离，需要辅助坐标从而帮助我们简便计算出复核台与货格之间距离，辅助坐标示意图如图：

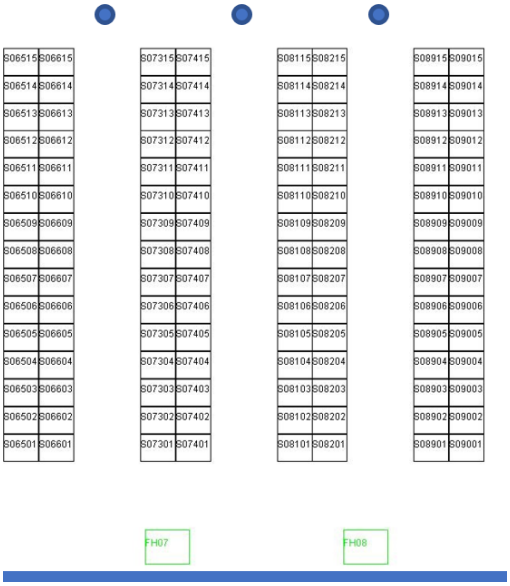

图 13 辅助坐标示意图

对于左侧的复核台来说，我们需要区分要计算距离的货格是否和该复核台属于同一列，据此我们再将具体计算方式给出。

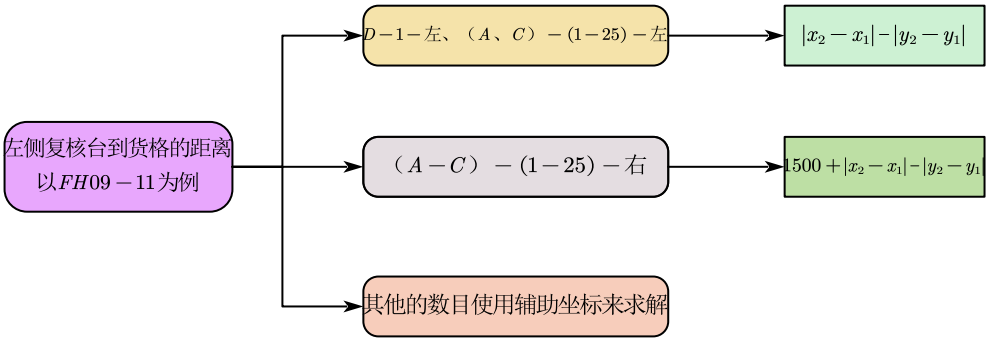

图 14 左侧符合台示意图

5.2 问题二的模型建立与求解

5.2.1 问题二的模型

$$\min T=\frac{\sum_{u\in H_z}\sum_{z=1}^{1} l_{u,z}+l_{z',z''}}{v}+\sum_{z=1}^{1}\sum_{i\in z} t_i+\sum_{z=1}^{1} 30*q_z$$

$$\begin{cases} \sum_{i,j\in Z}^{3000} l_{ij}=1,j=1,...,3000 \\ \sum_{i,j\in Z}^{3000} l_{ij}=1,i=1,...,3000 \\ \sum_{i,j\in Z} l_{ij}\leqslant q_z-1 \\ l_{ij}\in(0,1),i=1,...,3000,j=1,...,3000 \end{cases}$$

5.2.2 问题二的算法建立

STEP1：定义一个 result 变量记录蒙特卡洛模拟求得的最小时间，初始化为无穷大，定义蒙特卡洛模拟次数。

STEP2：根据 randperm 函数生成一个任务单货格的随机排列。

STEP3：计算从给定的 FH10 复核台领取订单后，当前订单访问序列所花费的总时间。

STEP4：找到当前排列的最后一个订单所在货格，计算距离当前货格最近的复核台完成交付。

STEP5：计算总的时间，判断当前结果是否大于 result,若为否，则返回 STEP2。

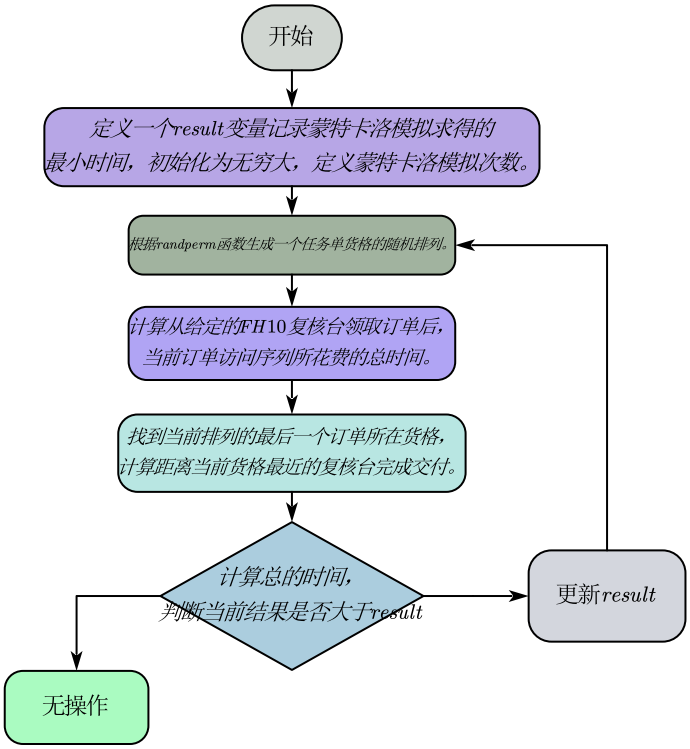

图 15 问题二算法流程图

5.3 问题三的模型建立与求解

5.3.1 问题三的模型

$$\min T=\frac{\sum_{u\in H_z}\sum_{z=1}^{5}l_{u,z}+l_{z',z''}}{v}+\sum_{z=1}^{5}\sum_{i\in z}t_i+\sum_{z=1}^{5}30*q_z$$

$$\begin{cases}\sum_{i,j\in Z}^{3000}l_{ij}=1,j=1,...,3000\\ \sum_{i,j\in Z}^{3000}l_{ij}=1,i=1,...,3000\\ \sum_{i,j\in Z}l_{ij}\leqslant q_z-1\\ l_{ij}\in(0,1),i=1,...,3000,j=1,...,3000\end{cases}$$

5.3.2 问题三的算法建立

STEP1：定义一个 result 记录拣货员完成所有订单花费的总时间，初始定义为无穷大，定义一个总够大的变量表示蒙特卡洛模拟的循环次数。

STEP2：根据 randperm 函数生成一个任务单之间的排列。

STEP3：对于第一个任务单，起始点是 FH03，此后的任务单的起始点都是上一个任务单的终点。

STEP4：遍历当前生成的任务单顺序，对于每一个任务单，利用 randperm 函数生成订单的排列。

STEP5：计算当前排列顺序所取得的时间。

STEP6：找到两个复核台中距离当前任务单的最后一个货格距离最近的一个作为当前任务单的终点。

STEP7：将当前的终点作为起点，继续下一个任务单的遍历，累计当前复核台的工作时间。

STEP8：是否为最后一个任务单，若是，则计算总的花费时间，更新 result，随后若判定到达迭代的边界，计算每个复核台的利用率，并输出 result 和每个复核台的利用率；若否，则返回 STEP3;若未达到边界，则返回 STEP1。

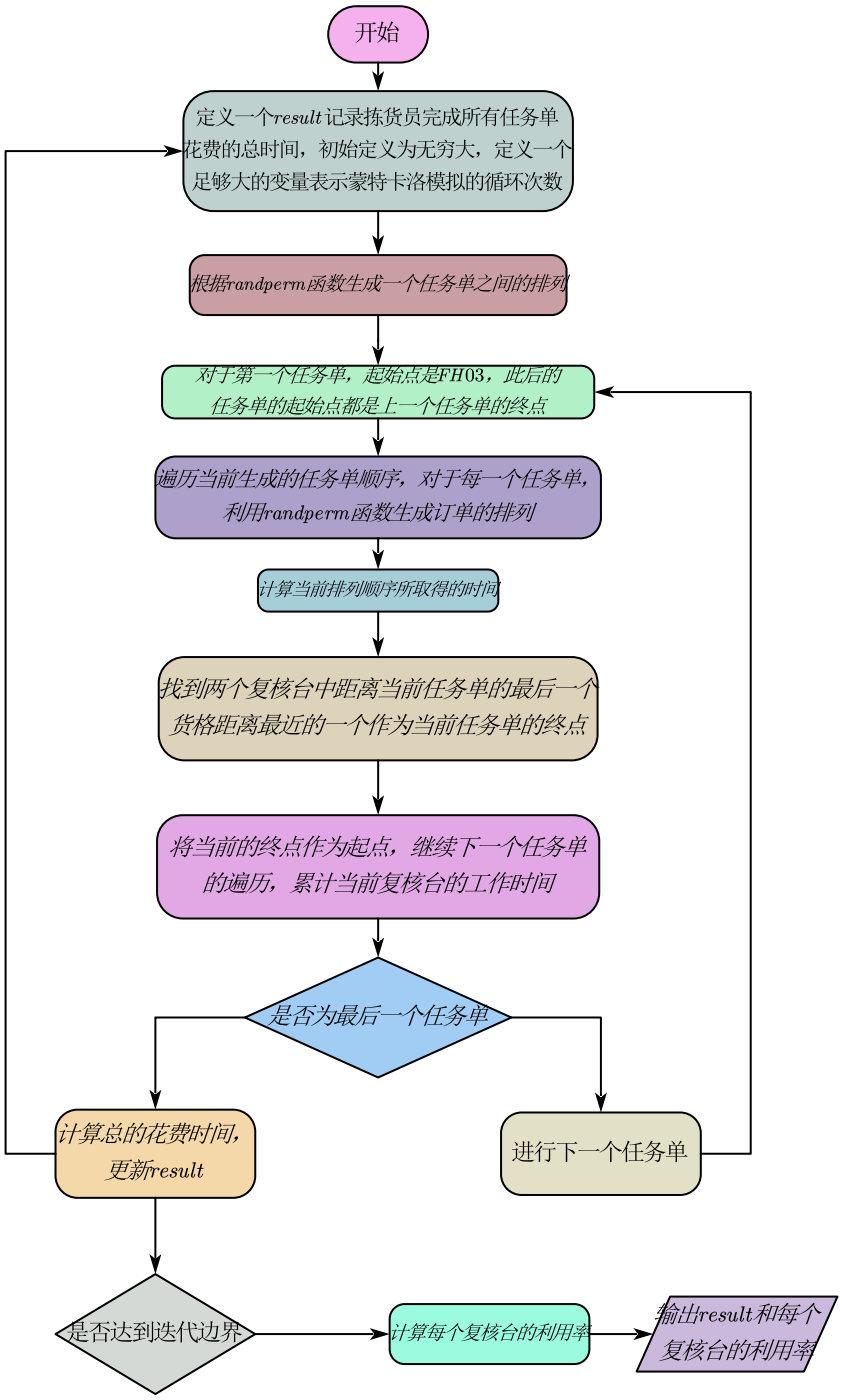

图 16 问题三算法流程图

5.4 问题四的模型

（1）行走时间 （复核台到货格的时间+货格之间行走的时间）

$$T_0 = \frac{\sum_{u \in H_z} \sum_{z=1}^{49} l_{u,z} + l_{z',z''}}{v}$$

其中，z 表示任务单的编号，H_z 表示 z 任务单经过的复核台集合，z', z'' 表示 z 任务单拣货时的起始货格和最终货格。

（2）下架时间 T_a

$T_a = \sum_{z=1}^{49} \sum_{i \in z} t_i$，在货格 i 处的下架时间 t_i 表达式如下：

$$t_i = \begin{cases} 5x_i, & x_i < 3 \\ 4x_i, & x_i \geq 3 \end{cases}$$ （此处，x_i 表示货格 i 处需要拣货的数量）

（3）复核打包时间 T_b

$T_b = \sum_{z=1}^{49} 30 * q_z$ 其中，q_z 表示 z 任务单中包含的订单数量。

（4）等待时间 T_d

$T_d = \sum_{u=1}^{4} \sum_{z_1 \in D_u} t_{z_1}$，其中 D_u 是复核台 u 复核打包的任务单集合。t_{z_1}（任意任务单的等待时间）表达式如下：

$$t_{z_1} = \begin{cases} 0, & t'_{z_1,u} \geq t''_{z_2,u} \\ s_{z_1 z_2}\left(t''_{z_2,u} - t'_{z_1,u}\right), & t'_{z_1,u} \leq t''_{z_2,u} \end{cases}, \quad z_1 \neq z_2$$

这里，$s_{z_1 z_2} = \begin{cases} 1, & \text{任务单} z_1 \text{在} z_2 \text{前面复核} \\ 0, & \text{否则为0} \end{cases}$，$t'$ 表示到达时刻，t'' 表示完成时刻，则 $t''_{z_2,u}$ 表示任意任务单 z_2 在 u 复核台的复核完成时刻，$t'_{z_1,u}$ 表示任意任务单 z_1 到达复核台 u 处的时刻，u 代表复核台编号，z_1 和 z_2 表示任意的任务单。

$$\min T = \frac{\sum_{u \in H_z} \sum_{z=1}^{49} l_{u,z} + l_{z',z''}}{v} + \sum_{z=1}^{49} \sum_{i \in z} t_i + \sum_{z=1}^{49} 30 * q_z + \sum_{u=1}^{4} \sum_{z_1 \in D_u} t_{z_1}$$

$$\begin{cases} \sum_{i,j\in Z}^{3000} l_{ij}=1,j=1,...,3000 \\ \sum_{i,j\in Z}^{3000} l_{ij}=1,i=1,...,3000 \\ \sum_{i,j\in Z} l_{ij}\leqslant q_z-1 \\ l_{ij}\in(0,1),i=1,...,3000,j=1,...,3000 \end{cases}$$

l_{ij}表示拣货员经过货格 i 到货格 j 为 1，反之为 0。式子（1）和（2）表示拣货过程中每个货格只能经过一次。式子（3）表示拣货路径不存在小回路。

通过上述的目标函数以及约束条件即可求出最优解。

六、模型的评价与推广

6.1 模型的优点

（一）基于优化理论，我们建立了非线性规划模型。从行走时间、下架时间、复核打包时间及任务单等待时间这四个时间来分析，基于合理假设使完成出库需要花费的时间最小化，使全文内在逻辑统一，保证了行文的连贯性。

（二）模型以完成出库需要花费的时间最小为目标，考虑了拣货过程中的一些约束条件，约束条件表达清晰简洁，整个线性规划表达式优美完整，具有较强观赏性。

（三）模型求解所得图表较多，全文直观性体验较好。

（四）利用蒙特卡罗模拟得到最终结果，具有一定的创新性。

（五）问题求解结果使用到 Matlab 软件，使得计算结果具有很好的精确性和可信力。

6.2 模型的缺点

（一）根据大数定理，蒙特卡洛模拟次数越多越逼近最优解，由于配置受限，模拟次数较少，可能时局部最优解。

（二）对模型假设较多，将拣货工作过程设置得较为理想化，未考虑到拣货员因拣货车体积、意外情况等产生的时间延迟对工作效率的扰动。

七、参考文献

[1]姜启源,谢金星,叶俊.数学模型(第四版).北京:高等教育出版社,2011.

[2]司守奎,孙兆亮.数学建模算法与应用(第二版).国防工业出版社.2017.01

附录

程序一：求货格中点坐标

```
clc
clear
 [~,~,a]=xlsread('货格.xlsx');

for i= 1:3000
   s=char(a(i,1));
   y=s(2:6);
   z=str2double(y);
   a(i,1)={z};
   b=char(a(i,9));
   c=b(2:4);
   d=str2double(c);
   a(i,9)={d};
   if(mod(d,2)==0)
        a(i,6)={1};
   else
        a(i,6)={0};
   end

end
a_=cell2mat(a);

for i=1:3000
    if(a_(i,6)==0)
        a_(i,3)=a_(i,3)+400;
    else
        a_(i,2)=a_(i,2)+800;
        a_(i,3)=a_(i,3)+400;
    end

end
 xlswrite('货格_.xlsx',a_);
```

程序二：分别求下面和左面复核台中点坐标

```
clc
clear
[~,~,a]=xlsread('复核台.xlsx');

```

```
for i= 1:13
    b=char(a(i,1));
    c=b(3:4);
    d=str2double(c);
    a(i,1)={d};
end

a_=cell2mat(a);

for i=1:13
    if(a_(i,2)~=0)
        a_(i,4)=a_(i,2)+500;%上边中点
        a_(i,5)=a_(i,3)+1000;
    else
         a_(i,4)=a_(i,2)+1000;%右边中点
        a_(i,5)=a_(i,3)+500;
    end
end
xlswrite('复核台_.xlsx',a_);


```

程序三：求货格之间的距离

```
clc
clear
a=xlsread('货格_.xlsx');

result=zeros(3000,3000);
% x= a(1,1)
% x1=mod(x,10)
% x=floor(x/10)
% x2=mod(x,10)
% x=x1+x2
for i =1 :3000
    for j=1:3000
        if(i~=j)
            if((abs(a(i,2)-a(j,2)) <=1600 && abs(a(i,3)-a(j,3))<=11200 )&&(a(i,4)==a(j,4)))
                result(i,j)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
                result(j,i)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
            elseif((abs(a(i,2)-a(j,2)) <=1600 && abs(a(i,3)-a(j,3))>11200 )&&(a(i,4)==a(j,4)))
```

```
                result(i,j)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
                result(j,i)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
            elseif((abs(a(i,2)-a(j,2)) >1600 && abs(a(i,3)-a(j,3))>11200 )&&(a(i,4)==a(j,4)))
                result(i,j)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
                result(j,i)=1500+abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
            elseif((abs(a(i,2)-a(j,2)) >1600 && abs(a(i,3)-a(j,3))<=11200 )&&(a(i,4)==a(j,4)))
                x= a(i,1);
                x1=mod(x,10);
                x=floor(x/10);
                x2=mod(x,10);
                x=x1+x2;

                y= a(j,1);
                y1=mod(j,10);
                y=floor(y/10);
                y2=mod(y,10);
                y=y1+y2;
                result(i,j)=3800+min((x+y-2),(30-x-y))*800;
                result(j,i)=3800+min((x+y-2),(30-x-y))*800;
            elseif(abs(a(i,2)-a(j,2)) >1600 &&(a(i,4)~=a(j,4)))
                result(i,j)=abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
                result(j,i)=abs(a(i,2)-a(j,2))+abs(a(i,3)-a(j,3));
            elseif(abs(a(i,2)-a(j,2)) <= 1600 &&(a(i,4)~=a(j,4)))
                x= a(i,1);
                x1=mod(x,10);
                x=floor(x/10);
                x2=mod(x,10);
                x=x1+x2;

                y= a(j,1);
                y1=mod(j,10);
                y=floor(y/10);
                y2=mod(y,10);
                y=y1+y2;
                result(i,j)=9800+min((x+y-2),(30-x-y))*800 + abs(a(i,2)-a(j,2));
                result(j,i)=9800+min((x+y-2),(30-x-y))*800 + abs(a(i,2)-a(j,2));
            end
        end
    end
end
 xlswrite('货格距离矩阵.xlsx',result);
```

程序四：求复核台到复核台以及复核台到货格之间的最短距离

```
clc
clear

a=xlsread('复核台_.xlsx');
b=xlsread('货格_.xlsx');

result=zeros(3013,3013);

for i=1:13
    for j=1:13
        if(i==j)
            result(i,j)=0;
            result(j,i)=0;
        else
            result(i,j)=abs((a(i,4)-a(j,4)))+abs(a(i,5)-a(j,5))-1000;
            result(j,i)=abs((a(i,4)-a(j,4)))+abs(a(i,5)-a(j,5))-1000;
        end
    end
end

for i=1:13
    for j=1:3000
        if( (a(i,1)==9 || a(i,1)==10 || a(i,1)==11) && (b(j,3)>=3400 && b(j,3)<=1460
0))
            if(b(j,4)==1)
                result(i,j+13)= 1500+abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-2250)+abs(b
(j,3)-2250),abs(a(i,5)-15750)+abs(b(j,3)-15750));
                result(j+13,i)= 1500+abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-2250)+abs(b
(j,3)-2250),abs(a(i,5)-15750)+abs(b(j,3)-15750));
            else
                result(i,j+13)= abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-2250)+abs(b(j,3)
-2250),abs(a(i,5)-15750)+abs(b(j,3)-15750));
                result(j+13,i)= abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-2250)+abs(b(j,3)
-2250),abs(a(i,5)-15750)+abs(b(j,3)-15750));
            end
        elseif ((a(i,1)==12 ||a(i,1)==13 ) && ( b(j,3)>=17400 && b(j,3) <= 28600))
            if(b(j,4)==1)
                result(i,j+13)= 1500+abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-16250)+abs(
b(j,3)-16250),abs(a(i,5)-29750)+abs(b(j,3)-29750));
                result(j+13,i)= 1500+abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-16250)+abs(
b(j,3)-16250),abs(a(i,5)-29750)+abs(b(j,3)-29750));
            else
```

```
                result(i,j+13)= abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-16250)+abs(b(j,3)-16250),abs(a(i,5)-29750)+abs(b(j,3)-29750));
                result(j+13,i)= abs(a(i,4)-b(j,2)) + min(abs(a(i,5)-16250)+abs(b(j,3)-16250),abs(a(i,5)-29750)+abs(b(j,3)-29750));
            end
        elseif(a(i,1)>=1 && a(i,1)<=8)
            if(b(j,4)==1)
                result(i,j+13)=abs(a(i,5)-b(j,3))+abs(a(i,4)-b(j,2)-750)+750;
                result(j+13,i)=abs(a(i,5)-b(j,3))+abs(a(i,4)-b(j,2)-750)+750;
            else
                result(i,j+13)=abs(a(i,5)-b(j,3))+abs(a(i,4)-b(j,2)+750)+750;
                result(j+13,i)=abs(a(i,5)-b(j,3))+abs(a(i,4)-b(j,2)+750)+750;
            end
        else
            if(b(j,4)==0)
                result(i,j+13)= abs(a(i,4)-b(j,2))+ abs(a(i,5)-b(j,3))+1500;
                result(j+13,i)= abs(a(i,4)-b(j,2))+ abs(a(i,5)-b(j,3))+1500;
            else
                result(i,j+13)= abs(a(i,4)-b(j,2))+ abs(a(i,5)-b(j,3));
                result(j+13,i)= abs(a(i,4)-b(j,2))+ abs(a(i,5)-b(j,3));
            end
        end
    end
end
xlswrite('台子到所有的距离.xlsx',result);
```

程序五：第二问求解方式

```
clc
clear

d = xlsread('距离矩阵.xlsx');
[~,order1] = xlsread('order1.xlsx');
[order2] = xlsread('order1.xlsx');


for i=1:23
    s=char(order1(i,1));
    y=s(2:6);
    z=str2double(y);
    order1(i,1)={z};
end
```

```
16.
17. order1_=cell2mat(order1);
18. order=[order1_,order2]
19.
20.
21.
22. for i=1:23
23.     x= order(i,1);
24.     x1=mod(x,10);
25.     x=floor(x/10);
26.     x2=mod(x,10);
27.     x_=x1+x2;
28.     x=floor(x/10);
29.
30.     order(i,1) = (x-1) * 15 +x_;
31. end
32.
33.
34. fhend=0;
35. min_result = +inf;
36. v=1.5;
37. N=10000;
38. fh=3010;
39. min_end=+inf;
40. min_path = [1:23];   % 初始化路径
41. for t=1:N
42.     result = 0;
43.     path = randperm(23);  % 生成一个 1-23 的随机打乱的序列
44.     if(order(path(1),2) < 3)
45.         result = result + order(path(1),2)*5;
46.     else
47.         result = result + order(path(1),2)*4;
48.     end
49.     result = result + v*d(fh,1);
50.     for i = 1:22
51.         result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
52.         if(order(path(i+1),2)<3)
53.             result = result + order(path(i+1),2)*5;
54.         else
55.             result = result + order(path(i+1),2)*4;
56.         end
57.     end
58.
59.     for k=3001:3013
60.         if(min_end>v * d(path(23),k))
```

```
            min_end =v * d(path(23),k);
            fhend=k;
        end
    end

    result = min_end + result;  % 加上返回复核台的时间
    if result < min_result  % 判断这次模拟走过的距离是否小于最短的距离，如果小于就更新最短距离
        min_path = path;
        min_result = result;
    end
end

min_result
min_path
fhend
```

程序六：第三问求解

```
clc
clear

d = xlsread('距离矩阵.xlsx');
[~,order1] = xlsread('order2.xlsx');
[order2] = xlsread('order2.xlsx');


for i=1:119
   s=char(order1(i,1));
   y=s(2:6);
   z=str2double(y);
   order1(i,1)={z};
end

order1_ = cell2mat(order1);
order = [order1_,order2];


for i=1:119
    x= order(i,1);
    x1=mod(x,10);
    x=floor(x/10);
    x2=mod(x,10);
```

```
    x_=x1+x2;
    x=floor(x/10);

    order(i,1) = (x-1) * 15 +x_;
end


% 开始求解


min_result = +inf;
v=1.5;
N=1;
fh1=3001;
fh2=3003;
fh3=3010;
fh4=3012;
final_path=[];
min_end=+inf;
fhend=0;
round=[];
for t=1:N
    orderrand=randperm(4);
    pathtmp=[];%当前的路径
    resulttmp = 0; %当前的结果
    start=fh3; %当前开始复核台
    endtmp=0; %当前终止复核台
    min_path=[];
    for p=1:5
        if(orderrand(p)==1)
            result = 0;
            path = randperm(26);  % 生成一个 1-n 的随机打乱的序列

            if(order(path(1),2) < 3)
                result = result + order(path(1),2)*5;
            else
                result = result + order(path(1),2)*4;
            end
            result = result + v*d(start,1); %起点到第一个货格的距离

            for i = 1:26
                result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
                if(order(path(i+1),2)<3)
                    result = result + order(path(i+1),2)*5;
                else
```

```
72.                     result = result + order(path(i+1),2)*4;
73.                 end
74.             end
75.
76.
77.             if(min_end<v * d(path(26),fh1))
78.                 min_end=v * d(path(26),fh1);
79.                 endtmp=fh1;
80.             end
81.             if(min_end<v * d(path(26),fh2))
82.                 min_end=v * d(path(26),fh2);
83.                 endtmp=fh2;
84.             end
85.             if(min_end<v * d(path(26),fh3))
86.                 min_end=v * d(path(26),fh3);
87.                 endtmp=fh3;
88.              end
89.             if(min_end<v * d(path(26),fh4))
90.                 min_end=v * d(path(26),fh4);
91.                 endtmp=fh4;
92.             end
93.             result = min_end + result;  % 加上返回复核台的时间
94.             resulttmp = resulttmp+result;
95.             min_path = [min_path,path];
96.             start=endtmp;
97.
98.
99.
100.         elseif(orderrand(p)==2)
101.             result = 0;
102.             path = randperm(20);  % 生成一个 1-n 的随机打乱的序列
103.             path=path+26;
104.             if(order(path(1),2) < 3)
105.                 result = result + order(path(1),2)*5;
106.             else
107.                 result = result + order(path(1),2)*4;
108.             end
109.             result = result + v*d(start,1); %起点到第一个货格的距离
110.
111.             for i = 1:20
112.                 result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
113.                 if(order(path(i+1),2)<3)
114.                     result = result + order(path(i+1),2)*5;
115.                 else
116.                     result = result + order(path(i+1),2)*4;
```

```
117.                 end
118.             end
119.
120.
121.             if(min_end<v * d(path(46),fh1))
122.                 min_end=min(min_end,v * d(path(46),fh1));
123.                 endtmp=fh1;
124.             end
125.             if(min_end<v * d(path(46),fh2))
126.                 min_end=min(min_end,v * d(path(46),fh2));
127.                 endtmp=fh2;
128.             end
129.             if(min_end<v * d(path(46),fh3))
130.                 min_end=min(min_end,v * d(path(46),fh3));
131.                 endtmp=fh3;
132.              end
133.             if(min_end<v * d(path(46),fh4))
134.                 min_end=min(min_end,v * d(path(46),fh4));
135.                 endtmp=fh4;
136.             end
137.             result = min_end + result;  % 加上返回复核台的时间
138.             resulttmp = resulttmp+result;
139.             min_path = [min_path,path];
140.             start=endtmp;
141.
142.
143.
144.         elseif(orderrand(p)==3)
145.             result = 0;
146.             path = randperm(26);  % 生成一个 1-n 的随机打乱的序列
147.             path=path+46;
148.             if(order(path(1),2) < 3)
149.                 result = result + order(path(1),2)*5;
150.             else
151.                 result = result + order(path(1),2)*4;
152.             end
153.             result = result + v*d(start,1); %起点到第一个货格的距离
154.
155.             for i = 1:26
156.                 result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
157.                 if(order(path(i+1),2)<3)
158.                     result = result + order(path(i+1),2)*5;
159.                 else
160.                     result = result + order(path(i+1),2)*4;
161.                 end
```

```
            end
            
            
            if(min_end<v * d(path(72),fh1))
                min_end=min(min_end,v * d(path(72),fh1));
                endtmp=fh1;
            end
            if(min_end<v * d(path(72),fh2))
                min_end=min(min_end,v * d(path(72),fh2));
                endtmp=fh2;
            end
            if(min_end<v * d(path(72),fh3))
                min_end=min(min_end,v * d(path(72),fh3));
                endtmp=fh3;
             end
            if(min_end<v * d(path(72),fh4))
                min_end=min(min_end,v * d(path(72),fh4));
                endtmp=fh4;
            end
            result = min_end + result;  % 加上返回复核台的时间
            resulttmp = resulttmp+result;
            min_path = [min_path,path];
            start=endtmp;
        elseif(orderrand(p)==4)
            result = 0;
            path = randperm(22);  % 生成一个 1-n 的随机打乱的序列
            path=path+72;
            if(order(path(1),2) < 3)
                result = result + order(path(1),2)*5;
            else
                result = result + order(path(1),2)*4;
            end
            result = result + v*d(start,1); %起点到第一个货格的距离
            
            for i = 1:22
                result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
                if(order(path(i+1),2)<3)
                    result = result + order(path(i+1),2)*5;
                else
                    result = result + order(path(i+1),2)*4;
                end
            end
            
            
            if(min_end<v * d(path(94),fh1))
```

```
207.                 min_end=min(min_end,v * d(path(94),fh1));
208.                 endtmp=fh1;
209.             end
210.             if(min_end<v * d(path(94),fh2))
211.                 min_end=min(min_end,v * d(path(94),fh2));
212.                 endtmp=fh2;
213.             end
214.             if(min_end<v * d(path(94),fh3))
215.                 min_end=min(min_end,v * d(path(94),fh3));
216.                 endtmp=fh3;
217.              end
218.             if(min_end<v * d(path(94),fh4))
219.                 min_end=min(min_end,v * d(path(94),fh4));
220.                 endtmp=fh4;
221.             end
222.             result = min_end + result;  % 加上返回复核台的时间
223.             resulttmp = resulttmp+result;
224.             min_path = [min_path,path];
225.             start=endtmp;
226.
227.
228.
229.
230.             else
231.             result = 0;
232.             path = randperm(25);  % 生成一个 1-n 的随机打乱的序列
233.             path=path+72;
234.             if(order(path(1),2) < 3)
235.                 result = result + order(path(1),2)*5;
236.             else
237.                 result = result + order(path(1),2)*4;
238.             end
239.             result = result + v*d(start,1); %起点到第一个货格的距离
240.
241.             for i = 1:25
242.                 result = v * d(order(path(i),1),order(path(i+1),1)) + result ;
243.                 if(order(path(i+1),2)<3)
244.                     result = result + order(path(i+1),2)*5;
245.                 else
246.                     result = result + order(path(i+1),2)*4;
247.                 end
248.             end
249.
250.
251.             if(min_end<v * d(path(119),fh1))
```

```
                min_end=min(min_end,v * d(path(119),fh1));
                endtmp=fh1;
            end
            if(min_end<v * d(path(119),fh2))
                min_end=min(min_end,v * d(path(119),fh2));
                endtmp=fh2;
            end
            if(min_end<v * d(path(119),fh3))
                min_end=min(min_end,v * d(path(119),fh3));
                endtmp=fh3;
             end
            if(min_end<v * d(path(119),fh4))
                min_end=min(min_end,v * d(path(119),fh4));
                endtmp=fh4;
            end
            result = min_end + result;  % 加上返回复核台的时间
            resulttmp = resulttmp+result;
            min_path = [min_path,path];
            start=endtmp;
        end


    end


      if resulttmp < min_result  % 判断这次模拟走过的距离是否小于最短的距离，如果小于就更新最短距离和最短的路径
        final_path=min_path;
        min_result = resulttmp;
        fhend=start;
        round=orderrand;
      end
end


min_result
final_path
fhend
round
```