队伍编号	201748
题号	C

货架式仓储拣货优化问题研究

【摘 要】

仓储中心是物流行业中的重要组成元素，研究仓储效率问题就是要对拣货路径选择与货位分配进行分析，在保证有序完成任务订单的同时尽可能缩短出货时间。本文基于**最短路径方法**，利用遗传算法、贪心算法进行求解，对拣货路径选择做出了合理的设计，同时对货位分配也做了深入的探索性研究。

针对问题一，需要计算货格与复核台共 3013 个元素之间的距离。考虑到不同情况下最短距离的计算方法不同，本文分了三个部分依次计算：货格与货格之间、货格与复核台之间、复核台与复核台之间。进一步通过分析不同排及不同巷道的情况得到了各种情形下的距离计算方法。

针对问题二，由于终点待定，本文应用图论知识提出了**替换复核台**的概念，固定了路径的起点和终点后，基于问题一求得的距离矩阵建立最短路径模型，这属于 **NP 难**问题，通过**遗传算法**进行求解，计算得到完成出库花费的总时间为 725.8 s 。

问题三中任务单的数量增多，复核台有两个，应用替换复核台的方法将其转换为**旅行商问题**，同样属于 NP 难问题，通过遗传算法求解得到每个任务单的完成路线，通过调整方向使五条路径首尾相连并从指定的初始复核台 FH03 出发，拣货员可以连续工作，从而有最高的效率。计算得到出库需要花费的时间为 2531.33 s ，FH03、 FH11 的利用率分别为 43.85% 、33.18%。

问题四进一步升级，需要对 4 个复核台、49 个任务单、9 个拣货员进行规划，同样先利用替换复核台方法及遗传算法算得每个任务单的最短路径，通过计算及理论分析发现在此种情形下，影响出货时间的主要因素为复核时间，因此只需调整 49 个任务单的终点复核台使得各复核台的复核时间达到平衡，运用**贪心算法**进行调整，从而得到最短的出货时间。计算得到出库需要花费的时间为 5699.4 s ，FH01、FH03、FH10、FH12 的利用率分别为 91.88%、92.30%、96.03%、92.58%。

问题五在问题四的基础上增加了一个复核台，由于第四问验证后发现复核台一直处于**满载**状态，本文对五个复核台的情况进行近似计算发现复核台仍处于满载状态，因此按照第四问的方法，增加第五个复核台后出库时间理论上大致会变为之前的五分之四，经过计算发现用时为 4605s，是四个复核台情况下的 80.8%，与理论分析相符。进一步探究发现，当复核台数量增加到六个后会出现复核台**空载**的情况，即无法继续通过增加复核台数量缩短出货时间，需要考虑对线路的进一步优化。

问题六提供了另一个侧面来提高仓库拣货的工作效率，通过调整商品的摆放位置来影响拣货效率。本文提出了仓库分区与商品分级的策略，在此基础上通过将二者耦合，构建了货物分配的优化模型并给出了相关建议。

关键词：最短路径　遗传算法　贪心算法　替换复核台　NP 难　旅行商问题

目 录

1 问题重述

随着互联网的日益普及，网上购物已收到许多消费者的青睐。但随着订单数目的日益增多，电商公司如何进行商品的拣货、复核、打包等工作成为了物流管理中最重要的问题。某电商公司客户订单下达仓库后，货物出库的流程包含：定位、组单、拣货、复核、打包等。因此，如何优化商品拣货的路径，保证订单出库耗时最短成为了提升物流效率的关键所在。

基于此，结合该电商公司仓库的实际情况，建立数学模型研究下列问题：

(1) 结合所给出的仓库中货格、复核台的具体信息，按照图中标示，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。

(2) 假设所有复核台正常工作，拣货员 P 在复核台 FH10 领取了任务 T0001。请给 P 规划理想的拣货路线，包括货格访问顺序、返回的复核台，计算完成出库花费的时间。

(3) 假设 2 个复核台(FH03，FH11)正常工作，5 个任务单(T0002-T0006)等待拣货，继续由拣货员 P 负责拣货，P 初始位置为 FH03。请给 P 指定任务领取顺序，规划理想的拣货路线，使得这些任务尽快出库。

(4) 假设 4 个复核台(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，请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。

(5) 在上述(4)的基础上，请评估增加一个正常工作的复核台对出库时间的影响。

(6) 商品在货架中的摆放位置，会在很大程度上影响拣货效率。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。对于仓内商品摆放问题，提出合理的建议。

2 问题分析

随着全球贸易的急速发展，物流效率的提高得到越来越多的关注。仓储在物流工作中担任着举足轻重的作用，提高仓储中心的作业效率成为相关领域的研究重点。本文主要针对某电商公司的货架式仓储中心的拣货路径进行研究，通过计算货格与复核台之间的距离建立最短路径模型，并利用遗传算法进行求解，对各种情况下的拣货路径做出了合理的设计。

2.1 问题一分析

问题一需要求出货格与复核台总共 3013 个元素之间的距离，即需要在考虑绕开障碍的情况下找到任意两个对象之间的最短距离。本文通过观察货格与复核

台的分布规律，分为三个部分分别计算：货格之间、货格与复核台之间、复核台与复核台之间。

对于货格之间，考虑到货架所在的排及巷道会影响路径的选择，本文对不同情况分别进行了分析；对于货格与复核台之间，处于仓库下方的复核台可直接计算，处于仓库左方的复核台则同样需要分多种情况进行分析；对于复核台与复核台之间则直接简化为坐标差的绝对值之和。

在计算过程中，本文提出了一种将货格坐标迁移到巷道相应位置的策略，极大程度上简化了计算的冗杂程度。

2.2 问题二分析

问题二给定了一个具体的任务单，需要给拣货员规划合理路径使得出库花费的时间最短。考虑到拣货员下架商品的时间及复核任务单的时间固定，只需求得最短拣货路径即可使得总用时最少。

此处已固定了初始复核台为 T0001，终点复核台可从 13 个复核台中任选，为了便于计算，本文提出了替换复核台的概念来固定终点，基于问题一得到的各元素之间的距离矩阵建立了问题二最短路径模型。由于该模型属于 NP 难问题，本文采用了遗传算法对模型进行求解，结合拣货员的行走速度即可计算出拣货路途中所花时间。

2.3 问题三分析

问题三将复核台数量限制在了两个，任务单数量变为五个，其余条件与第二问相同，需要求出出库最少花费的时间及复核台的利用率。本文将分为两个步骤进行分析，首先考虑先求出每个任务单各自的最短路径，其次再将这些路径按照合理的顺序与方向衔接起来。

在求解每个任务单的最佳路径时同样引入替换复核台来代替起点与终点，从而均可转化为旅行商问题，这依旧是一个 NP 难问题，求解方法与第二问相同，利用遗传算法寻求最短路径。在得到各个任务的最短路径后对它们调整顺序、改变路径方向使得第一项任务的起点为指定的 FH03、任务之间首尾相连、最后一项任务的复核时间尽可能短，即可得到耗时最短的方案。

2.4 问题四分析

问题四的复核台变为 4 个，拣货员增加到 9 个，任务单增加为 49 个，要求使得出库时间尽量少。考虑首先对每个任务单寻求最佳路径，再将这些路径进行合理的分配使得各个复核台需要复核的订单数量大致相等，以免造成需要复核的订单堆积而耗去大量时间，最后再通过贪心算法将这些任务单分配给拣货员。

2.5 问题五分析

问题五在问题四的基础上提供了一个额外的复核台，需要去评估对出库时间

的影响。通过对问题四的分析可以发现当拣货员较多时拣货的效率得以提高，从而复核时间对出货时间的影响有所上升。本文通过分析增加一个复核台后复核时间与拣货时间的关系来进行评估，当增加的一个复核台能够有效缓解复核台待复核订单堆积的情况时认为它在一定程度上提高了出货效率。

2.6 问题六分析

在自动化仓库进行出入库操作时，商品在货架中的摆放位置，会在很大程度上影响拣货效率。在货物摆放中，需要考虑的原则有很多，如货架应承载均匀，上轻下重；货物应加快周转，采取先入先出策略；提高拣货效率，应就近入库/出库等。然而，倘若一味地将畅销品放置在离复核台较近的位置，拣货员行走的距离固然会减少，但畅销品所在货架可能会造成拥挤，反而降低拣货效率。因此，提出了仓库分区与商品分级策略，以货格到复核台的距离对仓库库区进行分区，以商品的热销程度、相关性大小进行商品的分区。在此基础上，通过两者的耦合，构建货物分配的优化模型，使得总拣货达到最优。并在此基础上，给出货物存储的原则与货位分配的相关建议。

3 模型假设

1、假设拣货员完成一次任务即行走至复核台进行复核和打包；

2、假设不考虑商品与拣货车的长宽，即不考虑一任务单多车或多任务单一车的情况；

3、假设不考虑多储位货物拣货的情况；

4、多人同时在一个货格拣货，不考虑等待的时间；而多人同时到达一个复核台时，需要考虑货物等待的时间。

4 符号说明

数学符号	代表意义及说明	单位
w_1	货格长度	mm
l_1	货格宽度	mm
w_2	复核台长度	mm
l_2	复核台宽度	mm
w_{road}	竖直方向每组货架之间的距离	mm
l_{road}	水平方向相邻两排货架纵向距离	mm
$D(i,j)$	任意两个元素之间的距离	mm
x_{ij}	拣货所经过的路径	/
T_i	每个任务单出货时间	s

5　问题一模型建立与求解

基于对仓内路线分析，下列给出拣货员在货格之间、货格与复核台之间、复核台与复核台之间的最短路线的行走方式；并在此基础上计算 3000 个货格和 13 个复核台，总共 3013 个元素之间距离，给出最佳的仓内拣货路线方案。

5.1　模型准备——构建元素信息矩阵

通过观察数据文件中的序号顺序可知，S00102 表示第一个货架中第二个货格所处的位置、FH01 表示第一个复合台标号；这样的表示方式简洁明了，但其编号的不连续性会给结果的表示与输出造成一定困难。

因此，本文先提取出数据文件中所给货格名称中的行列号，并在此基础上通过一定映射关系将 3000 个货格按照 1-3000 顺序进行编号；同时考虑将 13 个复核台按照 3001-3013 的顺序编号，从而共有 3013 个元素参与排序。

基于此，下列通过构建元素信息矩阵 $L_{3013\times6}$ 来对每个元素进行坐标 (x_i,y_i)、其所在巷道位置以及货格所处位置等信息进行存储，方便进行各元素位置进行读取与调用。同时，考虑到货架与复核台距离计算方式的差异性，下列分别阐述货格信息矩阵（$L_{3013\times6}$ 中 1-3000 行）与复核台信息矩阵（$L_{3013\times6}$ 中 3001-3013 行）构建方法，以此得到总元素信息矩阵 $L_{3013\times6}$。

5.1.1　货格信息矩阵的构建

对于 L 矩阵中 6 列的数据存储内容如下 $(i\in[1,3000])$：

$L(i,1)$ 与 $L(i,2)$ 分别存储该货格的坐标信息 (x_i,y_i)；

$L(i,3)$ 与 $L(i,4)$ 分别存储该货架紧邻巷道位置的行数与列数；且 $L(i,3)$ 取 1-4 的整数，$L(i,4)$ 取 1-26 的整数；

$L(i,5)$ 与 $L(i,6)$ 分别存储该货格所处的货柜号与货格号。且 $L(i,5)$ 取 001-200 的整数，$L(i,6)$ 取 01-15 的整数；

例如，L 矩阵第三行 $(12300,14200,1,4,025,15)$ 中：$(025,15)$ 表示该货格为第 25 个货架中的第 15 个货格；$(12300,14200)$ 表示该货格的 x 坐标为 12300，y 坐标为 14200；$(1,4)$ 表示该货架紧邻在第 1 排第 4 个巷道旁。

（1）货架紧邻巷道编号规则

考虑到本文需要对不同货格行走距离进行分类讨论，为方便简洁起见，采用二维数组 (a,b) 来对该货架所处的巷道位置进行编号。其中，a 表示该巷道所在的行编号，取值为 1-4 的正整数；b 表示该巷道所在的列编号，取值为 1-26 的正整数。

例如：在 $(1,4)$ 标号中表示该货架是在第 1 行第 4 个巷道。为方便理解，下面给出 L 矩阵中各元素所表示含义的示意图。

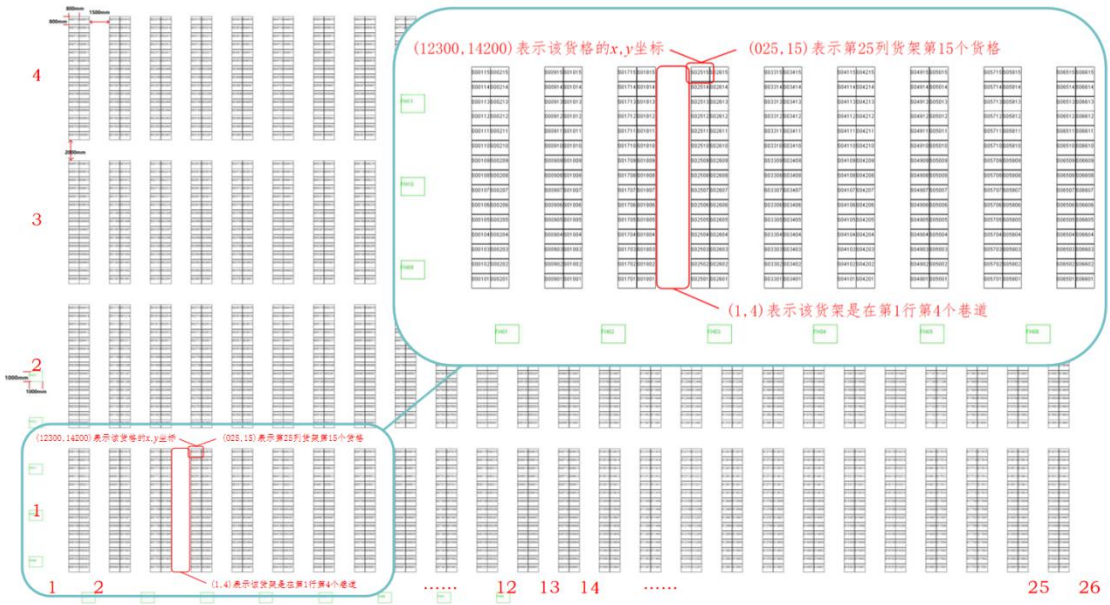

图 1　L 矩阵中各元素含义示意图

（2）构建货格信息矩阵

下列给出货格信息矩阵（元素信息矩阵 1-3000 行）各元素数值的计算求取方法：

$L(i,1)$ 与 $L(i,2)$ 表示的坐标信息 (x_i, y_i) 与 $L(i,5)$ 与 $L(i,6)$ 所表示的货格所处货柜号与货格号，均可以通过读取仓库数据文件获得；

$L(i,3)$ 与 $L(i,4)$ 分别存储该货架紧邻巷道位置的行数与列数，可以通过以下方式计算得到：

$$L(i,3) = floor\left(\frac{MOD\big(L(i,5),8\big)+1}{2}\right)$$

$$L(i,4) = floor\left(\frac{L(i,5)}{8}\right)+2-MOD\big(L(i,5),2\big)$$

此处，为了避免取整符号 $[x]$ 所引起的不必要的歧义，采用程序语言符号 $floor(x)$ 来表示对所得到的结果取整。

至此，已完成对于货格信息矩阵的构建。

5.1.2　复核台信息矩阵的构建

对于 L 矩阵中复核台部分的数据存储，仅需要对其坐标位置进行存储即可。因此，$L(i,1)$ 与 $L(i,2)$ 分别存储该货格的坐标信息 (x_i, y_i)；且将 $L(i,3) \sim L(i,6)$ 的元素置零 $(i \in [1,3000])$。

5.2　模型建立——计算各元素间距离

基于上述所构建的元素信息矩阵，结合仓库的实际情况，下列给出各元素之间的距离计算方法。

5.2.1　计算任意两货格间距离

（1）构建坐标偏移矩阵 C

在对仓库的布局进行细致分析后发现，同一组中左右两货架的计算方式不同；为了避免对该情况进行复杂的分类讨论，本文在计算货格距离之前，构建了各货格坐标的偏移矩阵$C_{(3000,2)}$，将坐标值换算到该货格紧邻的巷道中心点位置，即拣货员在取货时所处位置的外偏移量，下列给出了左右两货架两种情况下坐标偏移的计算方法。

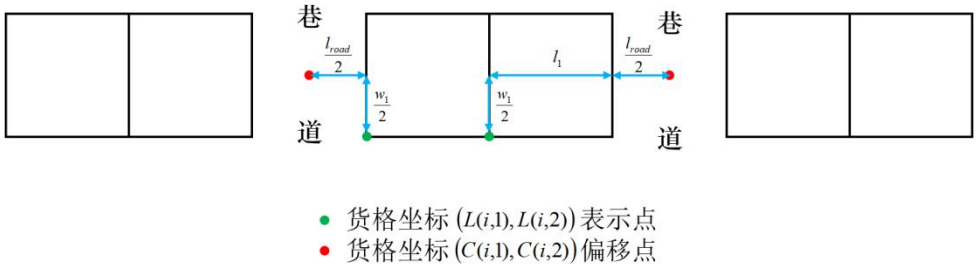

图 2　左右两货架坐标偏移示意图

转换公式如下：

如果该货格在该组货架的左侧$(if\ \ MOD(L(i,5),2)==1)$，坐标偏移矩阵C的横纵坐标分别为：

$$C(i,1)=L(i,1)-\frac{l_{road}}{2}$$

$$C(i,2)=L(i,2)+\frac{w_1}{2}$$

如果该货格在该组货架的右侧$(if\ \ MOD(L(i,5),2)==0)$，坐标偏移矩阵C的横纵坐标分别为：

$$C(i,1)=L(i,1)+l_1+\frac{l_{road}}{2}$$

$$C(i,2)=L(i,2)+\frac{w_1}{2}$$

至此，完成了偏移矩阵C的构建。

（2）各货格之间距离计算

考虑到仓库货格分布的实际情况，结合上述元素信息矩阵L中货格紧邻巷道的行列号，可以将货格距离的计算分成：同一排同一巷道、同一排不同巷道、不同排同一巷道、不同排不同巷道四种情况。下列分别对上述四种情况进行分类讨论：

- **情况一：两货格位于同一排同一巷道**

该情况下，元素信息矩阵L具有关系：$L(i,3)=L(j,3)$且$L(i,4)=L(j,4)$；此时，任意两货格之间的距离可以表示为：

$$D(i,j)=\left|C(i,2)-C(j,2)\right|+d\times 2$$

其中，d 为行走时的横向与竖向偏移，取 750㎜。

- **情况二：两货格位于同一排不同巷道**

该情况下，元素信息矩阵 L 具有关系：$L(i,3)=L(j,3)$且$L(i,4)\neq L(j,4)$；此时，拣货员最短路径的选取按照其所处货格的位置决定。

例如左图所示情况，当两件货物均处在货架偏上方时，选择从上方巷道行走距离较近；反之，拣货员应选择下方。因此，下列分两种情况进行讨论：

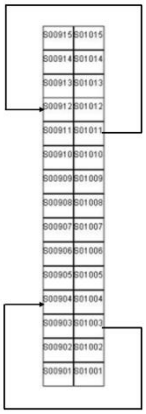

图 3　同一排不同巷道情况讨论示意图

我们引入变量 $P(i,j)=L(i,6)+L(j,6)$，以此来判定走上或者走下。

① 当 $P(i,j)\leq 15$ 时，选择从下方巷道行走距离较短；此时，任意两货格之间的距离可以表示为：

$$D(i,j)=\left|C(i,1)-C(j,1)\right|+w_1\times(P-1)+d\times 4$$

其中，w_1 表示货格的宽度。

② 当 $P(i,j)\geq 16$ 时，选择从上方巷道行走距离较短；此时，任意两货格之间的距离可以表示为：

$$D(i,j)=\left|C(i,1)-C(j,1)\right|+w_1\times(31-P)+d\times 4$$

- **情况三：两货格位于不同排同一巷道**

该情况下，元素信息矩阵 L 具有关系：$L(i,3)\neq L(j,3)$且$L(i,4)=L(j,4)$；此时，任意两货格之间的距离可以表示为：

$$D(i,j)=\left|C(i,2)-C(j,2)\right|+d\times 2$$

- **情况四：两货格位于不同排不同巷道**

该情况下，元素信息矩阵 L 具有关系：$L(i,3)\neq L(j,3)$且$L(i,4)\neq L(j,4)$；此时，任意两货格之间的距离可以表示为：

$$D(i,j)=\left|C(i,1)-C(j,1)\right|+\left|C(i,2)-C(j,2)\right|+d\times 2$$

5.2.2　计算货格到复核台之间距离

基于对仓库货架布局情况分析，结合“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”这一条件发现：

当复核台处于货架下方时（FH01-FH08），从上方进入复核台所耗距离一定≤其他方向到达复核台的距离；即对于下方的复核台，离最近一条边即为复核台上方；同理，对于左侧的复核台，最近方式即为从右侧到达复核台。

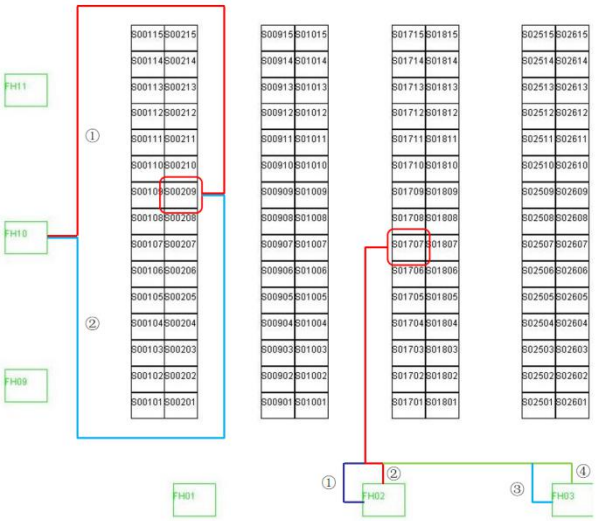

图 4　货架至复核台最短距离情况讨论示意图

因此，对于复核台最短距离的计算，分成 FH01-08 与 FH09-13 两种情况讨论。

（1）各货格至 FH01-08 复核台最短距离

该情况下，各货格至复核台之间的最短距离可以表示为：

$$D(i,j)=\left|C(i,1)-C(j,1)\right|+\left|C(i,2)-w_2\right|+d \qquad (i\in[1,3000],j\in[3001,3008])$$

其中，w_2 表示复核台的宽度，d 为行走时的横向与竖向偏移。

（2）各货格至 FH09-13 复核台最短距离

对于处在左侧的复核台，考虑到最短距离与拣货员所处的位置有关。例如，对于处在巷道数为第 3-4 排的拣货员来说，所有的复核台均在其下方，故向下无疑是最好的选择；

但对于处在巷道数为 1-2 排的拣货员来说，选择从上方/下方巷道行走直接影响着所耗距离。基于此，下列对拣货员从上方/下方行走至各复核台的距离进行分类讨论（如上图 4 中左侧①②两种行走方案），选择两者中较短距离作为最终的行走方案。

① 选择从上方巷道行走

该情况下，各货格至复核台之间的最短距离可以表示为：

$$D_{up}(i,j)=\left|C(i,1)-l_2\right|+\left|C(i,2)-L(j,2)\right|+\left(\left(15-L(i,6)+\frac{1}{2}\right)\times w_1\right)\times 2+3\times d$$

$$(i\in[1,3000],j\in[3009,3013])$$

其中，l_2 表示复核台的长度。

② 选择从下方巷道行走

该情况下，各货格至复核台之间的最短距离可以表示为：

$$D_{down}(i,j)=\left|C(i,1)-l_2\right|+\left|C(i,2)-L(j,2)\right|+\left(\left(L(i,6)-\frac{1}{2}\right)\times w_1\right)\times 2+3\times d$$

$$(i\in[1,3000],j\in[3009,3013])$$

其中，l_2 表示复核台的长度。

至此，对于处在巷道数为 1-2 排的拣货员来说，至复核台最佳的行走方案即为：$\min\left(D_{up}(i,j),D_{down}(i,j)\right)$。

此外，还需要考虑巷道在第 1 列，即处在复核台旁的特殊情况：

$$D(i,j)=\left|C(i,1)-l_2\right|+\left|C(i,2)-L(j,2)\right|+d\qquad(L(i,4)=1,j\in[3009,3013])$$

5.2.3 计算任意两个复核台之间距离

结合仓库的实际情况，两个复核台之间距离的计算应该分为在同一排（均在左侧/均在下方）以及不在同一侧两种情况。

- **情况一：复核台处于货架同一侧**

在此情况下，货架下方任意两复核台之间的距离可以表示为：

$$D(i,j)=\left|L(i,1)-L(j,1)\right|-l_2\qquad(i,j\in[3001,3008])$$

同样，货架左侧任意两复核台之间的距离可以表示为：

$$D(i,j)=\left|L(i,2)-L(j,2)\right|-w_2\qquad(i,j\in[3009,3013])$$

- **情况二：复核台处于货架不同侧**

在此情况下，任意两复核台之间的距离可以表示为：

$$D(i,j)=\left|L(i,1)-L(j,1)\right|-\frac{w_2}{2}+\left|L(i,2)-L(j,2)\right|-\frac{l_2}{2}$$

$$(i\in[3001,3008],j\in[3009,3013]\text{ 或 }i\in[3009,3013],j\in[3001,3008])$$

至此，完成了任意两个复核台之间的距离计算。

5.3 各元素之间距离计算结果

基于对上述货格与货格之间、货格与复核台之间、复核台与复核台之间的各种情况的详细讨论，得到了各元素之间的距离计算公式。

需要特别注意的是，在 3013×3013 的距离矩阵 D 中，由于 $i\to j$ 与 $j\to i$ 的距离相等，因此 $D(i,j)=D(j,i)$，且 $D(i,i)=0$。

至此，通过程序编写，最终得到的各元素之间的距离如附件结果 Q1 所示。限于篇幅原因，此处不再赘述。

6　问题二模型建立与求解

问题二需要对拣货员 P 完成任务单 T0001 的拣货路线进行规划，以复核台 FH10 为起点，终点可从 13 个复核台中任意选取。

基于上述分析，已经得到了任务单所有货格与复核台两两之间的距离，因此需要解决的是一个终点不确定的类旅行商问题。本问将引入一个替换终点的概念，将其转化为已知起点与终点的最短路径问题。

6.1　模型准备——替换复核台的引入

6.1.1　替换终点概述

下列先以一个简单的图论问题为例引入替换终点的概念。

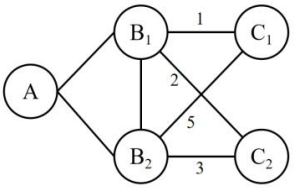

图 5　图论问题引入图

在上图中以 A 为起点，途径 B_1、B_2 两点，以 C_1 或 C_2 为终点的路径有两种，分别为 A-B_1-B_2-C_2、A-B_2-B_1-C_1，注意到当最后一个途径点为 B_1 时会选择更近的 C_1 为终点，当最后一个途径点为 B_2 时会选择更近的 C_2 为终点，故可引入一个替换终点 C，使得 $d(B_1,C)=d(B_1,C_1)$, $d(B_2,C)=d(B_2,C_2)$，即可确定一个终点 C，如下图所示：

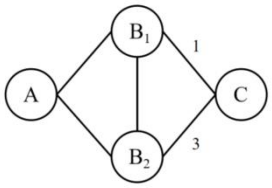

图 6　引入替换终点示意图

从而问题即被转化为以 A 为起点，途径 B1、B2 两点，以 C 为终点的最短路径规划问题。

6.1.2　引入替换复核台

假设该任务单中含有的货格数为 N，以 $H_i(i=1,\ldots,N)$ 表示这些货格点，FH10 为线路的起点，终点可在 FH01-FH13 中任意选择。

为了得到最短的路线，引入替换复核台 FH*，使得对于任意 i=1,…,N 满足

$$d(FH^*,H_i)=\min_{1\le j\le 13} d(FH_j,H_i),$$

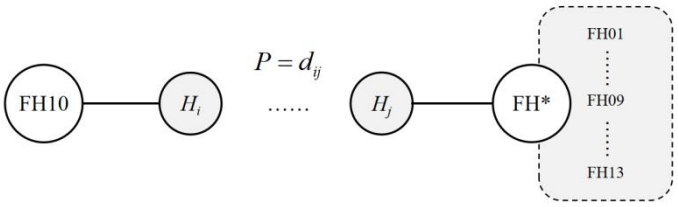

图 7　问题二最短路径转化示意图

于是得到含 FH10、FH*、$H_i(i=1,\ldots,N)$ 共 $N+2$ 个点在内的无向带权图，权值即为两点之间的距离，从而将问题转化为求解以 FH10 为起点、FH*为终点、途径 H_i 的最小距离问题。

6.2　仓内拣货最短路径模型

基于问题一的结果，我们可以得到货格与货格之间、货格与复核台之间、复核台与复核台之间的距离。因此，类似于旅行商问题（TSP）中最短路径的求解；但与之不同的是，本文所考虑的最短路径模型不需要拣货员返回起始出发点，是已知起点与终点的最短路径的规划选择问题。

6.2.1　最短路径模型建立

（1）优化目标函数

假设一个任务单中含有的货格数为 N，规定用 0 表示拣货员开始拣货的起点，通常是某一个复核台；用 N + 1 表示完成拣货后所到达进行复核与打包的终点，也是某一个复核台。

拣货路径优化问题的目标是使得拣货完成的总距离 D 最小，根据问题一中所得到的任意两个元素之间的距离 $D(i,j)$，则使得拣货路径最短的目标函数可以表示为[1]：

$$D = \min \sum_{i=0}^{N+1} \sum_{j=0}^{N+1} x_{ij} \cdot D(i,j)$$

其中，$D(i,j)$ 表示任意两个元素之间的距离；

x_{ij} 表示拣货所经过的路径，且 $x_{ij} = \begin{cases} 0, & \text{不存在元素} i,j \text{之间的路径} \\ 1, & \text{存在元素} i,j \text{之间的路径} \end{cases}$

（2）约束条件

考虑到仓库的实际情况，该问题中约束条件主要包含路径约束、货格约束。构建模型中，应保证在拣货过程中订单中的每个点访问数有且仅有一次，即货格约束条件可以表示为：

① 对于 $i=0,1,\ldots,N$ 来说，$\sum_{j=1}^{N} x_{ij} = 1 \quad (i \neq j)$，即对于起始复核台与中间所经

过的每个货格均会有下一个元素与之相连，其出度为 1；

② 对于 $j=1,2,\ldots,N+1$ 来说，$\sum_{i=1}^{N} x_{ij}=1 \quad (i\neq j)$，即对于中间经过的货格与终点复核台均会有上一个元素与之相连，其入度为 1。

至此，即可满足货格有进有出，且进出相等的条件。此外，对于起始复核台入度为 0、终点复核台出度为 0 的情况，给出以下规定：

$$\sum_{i=1}^{N+1} x_{i,0}=0 \qquad \sum_{j=0}^{N} x_{N+1,j}=0$$

6.2.2 最短路径模型汇总

基于 6.2.1 的分析，对于拣货最短路径 D 的优化问题，建立目标优化模型如下：

$$D=\min\sum_{i=0}^{N+1}\sum_{j=0}^{N+1} x_{ij}\cdot D(i,j)$$

$$s.t.\begin{cases}\sum_{j=1}^{N} x_{ij}=1 \quad (i=0,1,\ldots,N;i\neq j)\\ \sum_{i=1}^{N} x_{ij}=1 \quad (j=1,2,\ldots,N+1;i\neq j)\\ \sum_{i=1}^{N+1} x_{i,0}=0 \qquad \sum_{j=0}^{N} x_{N+1,j}=0\end{cases}$$

其中，$D(i,j)$ 表示任意两个元素之间的距离，求解公式基于问题一中的？？？式；

x_{ij} 表示拣货所经过的路径，且 $x_{ij}=\begin{cases}0, & \text{不存在元素}i,j\text{之间的路径}\\ 1, & \text{存在元素}i,j\text{之间的路径}\end{cases}$

6.3 基于遗传算法的最短路径求解策略

对于上述所构建的最短路径优化模型，本质上是一个典型的组合优化问题，属于 NP 难问题，即在多项式的时间内找不到确定的最优解。因此，针对这种问题目前采用最多的是启发式算法、遗传算法、蚁群算法等一系列智能算法。为此，本文采用遗传算法作为进行最短路径的求解。

6.3.1 遗传算法求解模型构建

（1）问题表示

对于待优化的最短路径求解问题，首先需要将其表示成适合于遗传算法操作的形式。用遗传算法解决最短路径的选取问题，该路线可以很自然地表示成为 N 个货格的排列；而路径表示是表示路程对应的基因编码的最自然、最简单的方法，在编码、解码、存储等过程中较为容易实现。

例如：最短路径(FH3→S00203→S00405→S01312→S01803→FH9)可以借助问题一所建立的元素信息矩阵 $L_{3013\times 6}$ 来进行编号，即表示为：(3003→18→50→192

→258→3009)，以方便路径结果的表示与输出。

（2）遗传算法模型构建

1）参数初始化，主要分为：

① 确定群体规模，即完成该任务单拣货员所经过的元素个数 n；

② 输入原始参数及给定参数，例如解求该问题时最大代数 genmax、交叉概率 pc、变异概率 pm 等；

③ 随机产生一组由初始个体构成的种群组成初始种群，令第一代为 gen=1。

2）适应度函数

对于每一个个体，计算其适应度函数；考虑到优化目标是使路径总长度最短，即适应度函数可以设置为路径长度的倒数：

$$f_k(1,2,\ldots,n)=\frac{1}{\sum_{i=1}^{n}D(i,i+1)}$$

其中，n 表示完成该任务单拣货员所经过的元素个数，k 表示任务单的序号。

3）选择算子

一般来说，在物竞天择的条件下，适应度较大（优良）的个体有较大的生存机会；而适应度较小（低劣）的个体继续存在的机会较小。换言之，在对于最短路径的选择问题中，适应度较高（路径长度较短）的行走方案继续保留的机会会更大。

简单遗传算法一般采用赌轮选择机制，令 $\sum_{k=1}^{num}f_k$ 表示群体的适应度值总和，f_k 表示种群中第 k 个染色体的适应度值，它产生后代的能力可以表示为：

$$P_k=\frac{f_k}{\sum_{k=1}^{num}f_k}$$

其中，num 表示仓内拣货的总订单数。

4）交叉算子

基于上述对编码方式的阐述，结合仓库实际情况，应要求每一个个体的染色体编码不允许有重复的基因码，即要满足任意一个货格必须而且只能访问一次的约束。

通过交叉操作不断地产生新的个体，本文采用部分匹配交叉（PMX）来实现交叉操作；先随机选取两个交叉点，定义这两点之间的区域为匹配交叉区域，并交换这两个父代的匹配交叉区域，生成新的子代染色体。

例如，选择两个父代染色体[3003, 18, 50, 192, 258, 388, 465, 732, 3009]与[3003, 786, 367, 398, 266, 287, 275, 135, 3009]，随机产生两个自然数 $r_1=4, r_2=6$，

将两个父本染色体 $r_1 \sim r_2$ 之间的基因片段进行交换，如下图 8 所示。

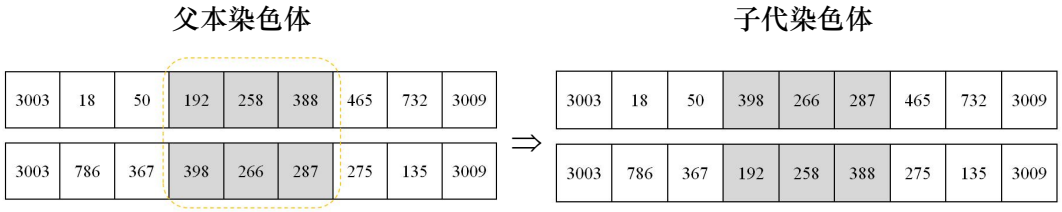

图 8　遗传算法染色体交叉示意图

5）变异算子

利用遗传算法解决最短路径的规划问题，传统的二进制编码的编译操作不太适用。基于问题表示中的分析，将个体编码成为一批货格的序列，随机的在该序列中抽取两个货格交换位置，从而实现个体编码中的变异操作。

6.3.2　基于遗传算法的最短路径选择算法流程图

基于上述所构建的遗传算法求解模型，结合最短路径的求解步骤，下列给出基于遗传算法的最短路径求解步骤。

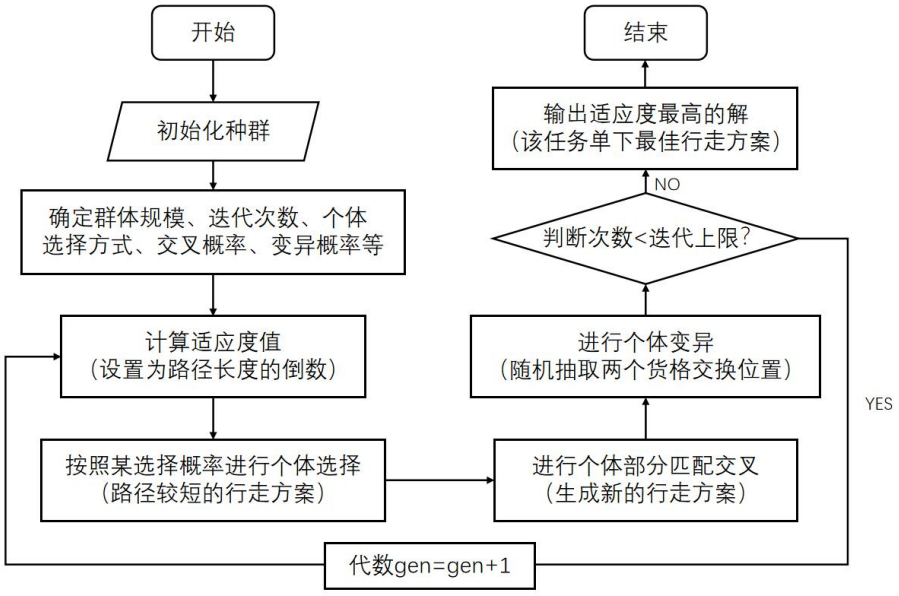

图 9　基于遗传算法的最短路径求解步骤

6.4　模型求解与结果分析

基于上述算法，通过 MATLAB 程序编写得到拣货员 P 完成任务单 T0001 最理想的拣货路线，包括货格访问顺序、待拣货商品件数、返回的复核台及各元素所对应的序号，具体结果如表 1 所示。

表 1　基于遗传算法的最短路径求解步骤

序号	元素原始名称	元素序号(1-3013)	商品件数
1	FH10	3010	0
2	S00107	7	2

3	S01713	253	2
4	S01308	188	1
5	S07515	1125	3
6	S08502	1262	1
7	S07212	1077	1
8	S06213	928	3
9	S10115	1515	1
10	S11106	1656	1
11	S11205	1670	1
12	S12608	1883	1
13	S13509	2019	1
14	S15911	2381	3
15	S14401	2146	1
16	S14908	2228	3
17	S13812	2067	3
18	S14510	2170	1
19	S13809	2064	1
20	S13004	1939	1
21	S12103	1803	2
22	S10508	1568	2
23	S10501	1561	1
24	S07305	1085	3
25	FH08	3008	0

在上述结果的基础上，计算得到理想条件下，拣货员完成任务单 T0001 的最短行走路线总长度为 373.200m。

此外，考虑到商品出货时间T_i包含拣货员行走时间t_{walk_i}、取货所需要的时间t_{pick_i}以及复核台复核打包耗时t_{check_i}三部分，即对于任务单 T0001 来说：

$$T_{0001} = t_{walk_{0001}} + t_{pick_{0001}} + t_{check_{0001}}$$

考虑到：拣货员的行走速度为 1.5m/s；对任意一个货格，若下架商品小于 3 件，每件完成下架花费 5s，否则每件花费 4s；每个订单复核和打包花费 30s。

因此，完成任务单 T0001 所耗总时间：

$$\begin{aligned} T_{0001} &= t_{walk_{0001}} + t_{pick_{0001}} + t_{check_{0001}} \\ &= 248.8\ \text{s} + 177\ \text{s} + 300\ \text{s} \\ &= 725.8\ \text{s} \end{aligned}$$

7 问题三模型建立与求解

问题三在问题二的基础上增加了任务单数量（T0002-T0006），同时将复核台的数量控制在了两个（FH03、FH11），通过设计任务的领取顺序并规划任务完

成的路线来使得任务出货的时间最短。

7.1 模型准备——替换复核台的引入

基于第二问的分析，可以采用类似的替换复核台方法，将两个复核台转化为一个充当起点和终点的替换复核台。因此对于每一个任务单而言，完成拣货即相当于从替换复核台出发，途径所有需要的货格点，最后回到替换复核台。把问题近似成起点与终点重合的最短路径策略求解，这是一个典型的旅行商问题。

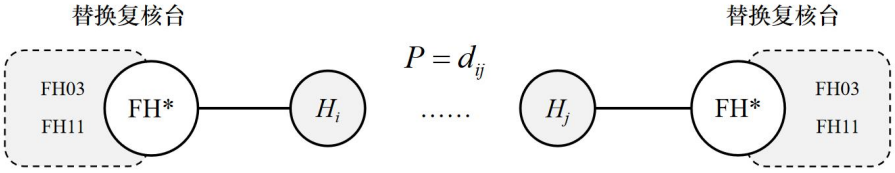

图 10　问题三最短路径转化示意图

7.2 多任务最短路径模型

考虑到对于拣货员 P 来说，完成任务单 T0002-T0006 只能一个一个任务完成，不能在完成一个任务过程中拣另一个任务的货；因此，可以看成完成各任务单相对独立。

此外，基于问题二中所构建的最短路径的优化模型，可以得到拣货员 P 完成每一任务单的最优路径选择。至此，单人、多任务最短路径问题可以转化为该拣货员完成单一任务的最优组合优化问题。

基于此，下文从以下两方面进行分析：

- 拣货员 P 完成每个独立任务单的最短路径选择；
- 各最短路径的组合优化问题。

7.2.1 单任务最短路径模型

针对每一个单独的任务单求解最短路径，此处的模型与问题二中的模型类似，只需改变起点与终点的限制条件。

（1）优化目标函数

假设一个任务单中含有的货格数为 N，规定用 0 表示替换复核台，即起点与终点。拣货路径优化问题的目标是使得拣货完成的总距离 D 最小，根据问题一中所得到的任意两个元素之间的距离 $D(i,j)$，则使得拣货路径最短的目标函数可以表示为：

$$D = \min \sum_{i=0}^{N} \sum_{j=0}^{N} x_{ij} \cdot D(i,j)$$

其中，x_{ij} 表示拣货所经过的路径，且 $x_{ij} = \begin{cases} 0, & \text{不存在元素} i,j \text{之间的路径} \\ 1, & \text{存在元素} i,j \text{之间的路径} \end{cases}$

（2）约束条件

与问题二不同的地方在于，此处的起点终点为同一点，它的出度与入度均为1，而对剩余的 N 个货格点而言，由于它们均需被访问一次，因此出度入度均为1，故约束条件可归纳如下：

$$\sum_{j=0}^{N} x_{ij} = 1 \quad (i \neq j)，\quad \sum_{i=0}^{N} x_{ij} = 1 \quad (i \neq j)$$

7.2.2 多任务衔接策略

在确定每个任务单的最短路径后，通过还原替换复核台可以确定每条路径的起点和终点。考虑到整项工作的初始起点已确定为 FH03 复核台，只需选择起点为 FH03 的任务单作为第一单，通过调整各任务单最短路径的方向或者端点的选择尽可能使得任务单之间首尾相连，即在完成一单任务到达复核台后，可以立即将此复核台作为下一个任务单的起点。

这样，通过对各任务单最短路径下起始点与终点的组合排序，可以使得拣货员 P 能够在选择最短路径的前提下，连续地完成所有任务单，这将最大限度地缩短所耗时间。

7.3 基于遗传算法的多任务最短路径模型求解

与问题二的最短路径优化模型类似，考虑到此处单任务最短路径的求解仍然属于 NP 难问题，故同样采用遗传算法进行求解。

7.3.1 遗传算法求解单任务最短路径

遗传算法求解的具体步骤与 6.3 类似，只需将编码部分染色体的首尾固定为替换复核台，转化成首尾相同的经典旅行商问题，其余部分不作赘述。通过编写 MATLAB 程序得到拣货员 P 完成 T0002-T0006 各任务单最理想的拣货路线：

T0002：（起点：FH03；终点：FH11）

T0003：（起点：FH03；终点：FH11）

T0004：（起点：FH03；终点：FH11）

T0005：（起点：FH03；终点：FH03）

T0006：（起点：FH03；终点：FH11）

7.3.2 多任务的衔接方案

基于 7.3.1 求解得到的结果，通过对各任务单下最短路径的组合优化（例如：变换任务单完成顺序、调整完成某一任务单的行走方向），可以使得拣货员 P 能够在选择最短路径的前提下，连续地完成所有任务单。

至此，考虑到拣货员 P 起始点为 FH03，且复核台进行复核和打包是以订单为基本单位，可以在拣货员 P 拣货的同时进行复核。因此，为了使得完成任务后

等待复核台复核和打包的时间最短，在同等情况下，优先考虑安排订单数较多的任务先完成，选择将订单数量最少的任务单 FH0003（共 11 个订单）作为最后一个任务，以减少拣货员完成所有任务后仍需等待最后任务单复核和打包的时间，从而使得所有任务尽快出货。

故，多任务单的最佳衔接方案为：

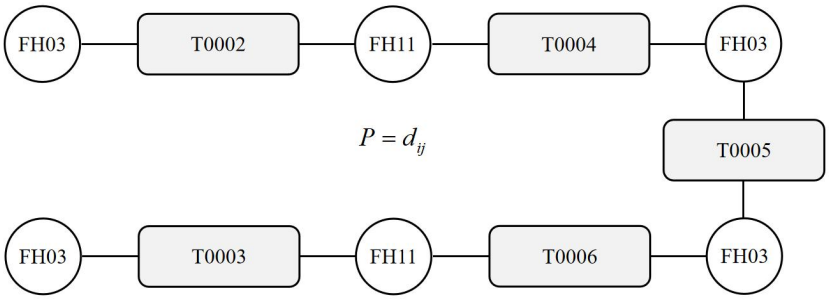

图 11　多任务单的最佳衔接方案示意图

7.3.3　多任务最短路径结果分析

（1）全部任务出库总耗时

基于上述所构建的多任务最短路径的组合优化模型，结合各任务单实际情况，计算得到拣货员完成各任务单的行走时间、取货时间以及复核台复核所耗时间等结果如表 2 所示。而各任务单具体的货格访问顺序、各元素所对应的序号等结果限于篇幅原因，见附件结果 Q1，此处不再赘述。

表 2　多任务单最短路径结果表

完成任务	拣货员完成任务时间				复核台复核打包时间		
顺序	行走距离(m)	行走耗时(s)	取货耗时(s)	拣货总时间(s)	订单数(个)	复核耗时(s)	复核台
T0002	423.9	282.60	182	464.60^{I}	15	$450^{①}$	FH11
T0004	392.2	261.47	186	$447.47^{\text{II}①}$	13	$390^{②}$	FH03
T0005	360.7	240.47	165	$405.47^{\text{II}②}$	13	$390^{③}$	FH03
T0006	441.6	294.40	168	$462.40^{\text{II}③}$	13	$390^{④}$	FH11
T0003	375.6	250.40	171	$421.40^{\text{II}④}$	11	330^{III}	FH03

然而，考虑到拣货员拣货与复核台复核的实际情况，如当拣货员完成 T0004 任务单的拣货工作、开始进行 T0005 任务单时，复核台可以同步进行 T0004 的复核与打包工作。

因此，完成所有任务的总耗时并不像问题二中“单任务问题”简单的相加，而是需要比较该复核台进行上一任务单复核时间与拣货员进行当前任务单拣货所耗时间。如果该复核台完成送至此处的前 N-1 任务单耗时 < 拣货员送至该处的后 N-1 个任务单的拣货总耗时，那么则不需要考虑最后一任务单送至该处的等

待问题；否则，当拣货员完成最后一单的拣货工作后，还需要等待一段时间τ后，才能开始最后一单的复核和打包工作。

换言之，如上表 2 中拣货员 P 完成 T0004 后，需将商品送至复核台 FH03；而当 P 开始进行 T0005 的拣货时，复核台 FH03 可以同步完成 T0004 的复核打包工作。也只有当复核台完成 T0004 复核的时间(上表标注复核耗时②) < P 完成 T0005 的拣货总时间时，才能够不产生货物的积压。否则，会出现当 P 连续送 N-1 个任务单的货物后，总复核时间$\sum_{i=1}^{N-1} t_{check_i}$ > P 完成后 N-1 单拣货时间$\sum_{i=2}^{N} t_{pick_i}$；那么当 P 送第 N 个任务单的货物来复核台时，需要等待一段时间τ后，才能开始第 N 个任务单的复核和打包工作。

为了方便进行各段时间的匹配与考虑，表 2 中需在同一复核台进行商品复核且对上标$(①\sim④)$相同的时间段进行比较。

至此，得到的最终总时间分成三部分：

- Ⅰ：拣货员 P 完成 T0002 时耗时，此时复核台空闲；
- Ⅱ：比较①～④各阶段拣货耗时与同一复核台复核耗时，最终完成 T0004~T0003 的时间；

需要特别提出的是，在进行①阶段时间比较时，考虑到拣货员 P 下一任务单(T0004)送至 FH03 复核台而不是同一复核台 FH11，因此，尽管 T0002 的复核时间较长，但与下一单不在同一复核台，故此处无需进行更改。

- Ⅲ：复核台 FH03 进行最后一单 T0003 的复核与打包，拣货员 P 空闲。

基于上述分析，所有任务出库的总耗时：

$$\sum_{i=2}^{6} T_{000i} = T_{\mathrm{I}} + T_{\mathrm{II}} + T_{\mathrm{III}} = 2531.33 \ \mathrm{s}$$

（2）各复核台利用率

基于上述对各阶段所耗时间的分析，在计算复核台利用率时做出如下假定：没有新任务前，复核台将处于空闲状态。开始时间设置为拣货员 P 开始行走拣货，并以最后一个订单商品出库时间最为结束。且从 0 时刻到 TOTAL_TIME 时刻，若一个复核台总空闲时间为 IDLE_TIME，则该复核台利用率η为：

$$\eta = \left(1 - \frac{\mathrm{IDLE\_TIME}}{\mathrm{TOTAL\_TIME}}\right) \times 100\%$$

那么，复核台 FH003 与 FH11 的利用率分别为：

$$\eta_{\mathrm{FH03}}=\left(1-\frac{\mathrm{IDLE\_TIME}_{\mathrm{FH03}}}{\mathrm{TOTAL\_TIME}}\right)\times 100\%=43.85\%$$

$$\eta_{\mathrm{FH11}}=\left(1-\frac{\mathrm{IDLE\_TIME}_{\mathrm{FH11}}}{\mathrm{TOTAL\_TIME}}\right)\times 100\%=33.18\%$$

8 问题四模型建立与求解

问题四的复核台数量、任务单数量、拣货员数量均有所增加，本文将首先计算每个任务单的最短路径，通过对这些任务单进行调整与分配使得各个复核台的订单数量大致相等从而避免造成待复核订单大量堆积的情况。

8.1 单任务最短路径求解

与 7.2.1 中求解方法相同，运用遗传算法对 49 个任务单分别求解最短路径，求解结果统计如下：

表 3 单任务单最短路径结果表

复核台	与之连接的路径数目(度)
FH01	15
FH03	45
FH10	4
FH12	34

并且，通过对结果的分析发现：49 个任务拣货时间平均值为 445.35s，复核时间的平均值为 419.39s。

8.2 基于贪心算法的任务单调整方案

由 8.1 中的结果可以发现，按照直接计算得到的路径，四个复核台作为端点的比例分配不均，其中以 FH10 作为端点的路径过少，以 FH03 作为端点的路径过多，这会造成 FH10 利用率过低，而 FH03 由于大量待复核任务单堆积而耗去大量时间。需要对部分路径的端点进行调整。

观察任务拣货时间与复核时间可以发现，这两个值相差较小，但由于拣货员多达九个，而复核台只有 4 个，拣货效率是复核效率的两倍多，可以推断影响出货时间的决定因素是复核时间。

因此为了使得出货总时间最小，首要目标是使得各个复核台审核的订单数量大致相等，从而避免出现某复核台需要复核的订单数量过多而耗去大量时间的情况。在调整过程中可以对任务的起点与终点进行改变，拣货员前往更远复核台造成的时间增量对总时间不造成影响。

为了使得复核台尽快开始工作，对于前四个任务单优先选择拣货时间最短的任务单，拣货员可用最少的时间完成任务并送至复核台处。

对于其他任务单，为了尽可能使得各复核台的复核时间大致相等，可采用贪心算法进行安排。每次领取任务优先选择复核时间最长的任务，完成任务后将其送至当前待复核时间最短的复核台处。通过不断选择当前情况下的最优解来达到全局最优。

下列给出任务单选择及终点复核台选择的算法流程图：

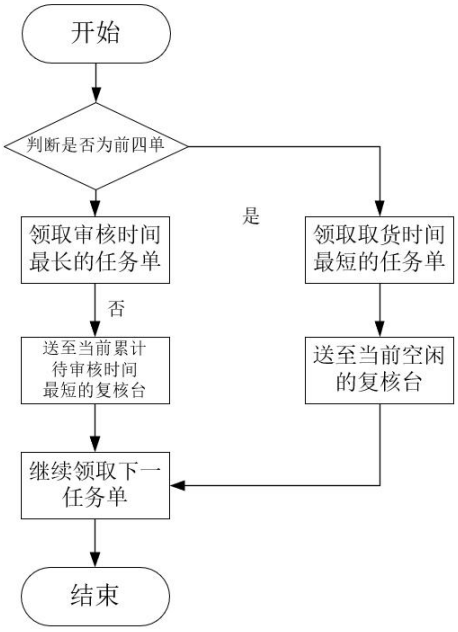

图 12　基于贪心算法的任务单及终点复核台选择方案示意图

同时为了确保不会出现复核台复核完成而没有新的任务单送达，本文特设计专门的程序对拣货员每次完成任务送至复核台处的时刻进行判断。

8.3　模型求解结果及分析

应用上述建立的基于贪心算法的任务单调整方案，求解得到了九个拣货员领取任务的顺序及路线，具体方案可见计算结果表格。

此处给出各个复核台运行的总时间、第一单任务到达该复核台所花费的时间及各复核台的利用率。

表 4　基于贪心算法的复核台时间表

复核台	第一单送达时间(s)	运行时间(s)	总时间(s)
FH01	228.00	5236.47	5464.47
FH03	204.40	5260.60	5465.00
FH10	226.13	5473.27	5699.40
FH12	197.47	5276.667	5474.13

由此可知，出货需要花费的时间为 5699.40s，计算得到四个复核台的利用效率如下表所示：

表 5　四个复核台的利用效率计算结果表

复核台	FH01	FH03	FH10	FH12
利用效率	91.88%	92.30%	96.03%	92.58%

值得说明的是，经过程序检验，在上述过程中并未出现复核台复核完成而没有新的任务单送达的情况。在一定程度上说明了此处结果的合理性。

9　问题五模型建立与求解

9.1　增加复核台的理论分析

问题五在问题四的基础上增加了一个复核台。由于在第四问中，考虑到复核台是一直处于满载的情况，即不存在复核台等着拣货员的情况，所以在进行分配拣货员线路是采用了先安排货物较多的任务这一贪心算法。但是当复核台逐渐增加时，是否还满足这一条件呢？本文进行了如下探讨：

首先考虑到如果复核台是一直处于满载，总的出货时间等于复核台等着拣货员送的第一单货物的时间加上复核货物的时间。

即如下式

$$T = t_1 + t_{复核}$$

这里用每个复核台的平均等待时间来近似等于复核台等着拣货员送的第一单货物的时间

$$t_1 \approx t_{ave} = \frac{(t_{T001} + t_{T003} + t_{T010} + t_{T012})}{4}$$

随后利用每个复核台平均的复核时间近似等于所需要计算的复核时间

$$t_{复核} \approx t_{复核\_ave} = \frac{num_{total} \times 30}{4}$$

需要注意的是，由于在计算时，总的出货时间应该取所有复核台之间耗时中的最大值，故此处利用的平均时间必然小于出货时间，但是由于对于每个复核台的第一单等待时间来说，由于均取最短耗时，同时每个复核台的复核时间也进行均匀分配，故两者差距并不大。利用第四问的模型，在分配路线的时候，依旧忽略复核台等待的时间。从而总时间应该与复核台数成反比，即（改进量用η表示）

$$\eta = \frac{T_{total5}}{T_{total4}} = \frac{4}{5}$$

9.2　增加复核台的实际结果

本文计算了 4、5、6、7 个复核台时，总的耗时与理论耗时，结果如下图所示：

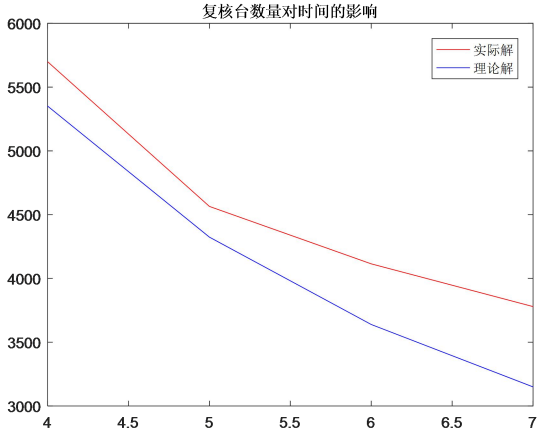

图 13　复核台数量对出货时间的影响分析图

若实际情况中出现了大量复核台的等待时间，则实际解与理论解之差必然会越来越大。由图可以看出当复核台数为 5 时，并未出现明显的复核台的等待时间。但是当复核台数大于 6 时，复核台的等待时间开始增加，此时如果还需要进行优化，就需要考虑对路线进行进一步的优化。

由于当复核台数为 5 时并未出现明显的复核台等待时间，故方法是可靠的。

实际计算结果为 4605 s，改进量为

$$\eta_{\text{实际}} = 0.808$$

实际情况与理论分析相符，出货时间缩短为四个复核台的 80.8%。

10　问题六模型建立与求解

在自动化仓库进行出入库操作时，商品在货架中的摆放位置，会在很大程度上影响拣货效率。在货物摆放中，需要考虑的原则有很多, 如货架应承载均匀，上轻下重；货物应加快周转，采取先入先出策略；提高拣货效率，应就近入库/出库等。然而，倘若一味地将畅销品放置在离复核台较近的位置，拣货员行走的距离固然会减少，但畅销品所在货架可能会造成拥挤，反而降低拣货效率。因此，下列提出了仓库分区与商品分级策略，通过两者的耦合，使得总拣货达到最优，并在此基础上给出了货位分配的相关建议。

10.1　仓库分区与商品分级策略

10.1.1　仓库分区优化问题

基于问题一可以得到任意两元素之间的距离 $D(i,j)$，并得到矩阵 3013×3013 中的最小距离 $D_{\min}=1.5\text{m}$ 与最大距离 $D_{\max}=132.2\text{m}$；考虑到仓库管理的实际情况，过多或者过少的分区都会造成管理的不变。因此，本文利用此极差确定分区

的个数为 6，并按照距离长短将仓库分为Ⅰ~Ⅵ共 6 个等级；其中，等级越高代表该货格到复核台的距离越短，拣货效率越高。

表 6　仓库分区等级表

库区等级	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ
距离范围(m)	1.5~20	20~40	40~60	60~80	80~100	100~132.2

10.1.2　商品分级优化问题

（1）根据热销度分级

基于市场营销学相关理论，人们对不同的商品总表现出不同程度的购买欲望，而人们对不同商品的需求程度也会呈现出差异。因此，基于附件中所给出的订单信息，我们可以统计得到订单数最多的 003413 货格被访问了 18 次，并且货格的访问次数大多成集群分布。

因此，在进行仓库的管理过程中，选取一段采样时间对仓库的订单商品出现的次数进行统计，即得到商品的种类和频率信息，并按照商品订购的次数进行热销程度的分级，以此分为火爆款、热销款、普通款和低频款。需要注意的是，在大型仓库的商品入库或出库情况一般会随着季节的变化而发生些许变动；在进行商品热销程度分级时应兼顾季节、温度、流行性等多类因素。

（2）商品聚类优化

倘若不考虑不同商品之间的相互关系，仅单纯地依赖访问频率来对所有商品进行货位的分配往往得不到较好的结果。例如：考虑螺钉与螺母、牙膏与牙刷等商品，如果仅依据上述规则，可能会被摆放在相距很远的货柜上，但是由于订单中两种物品常常会同时订购，经济学上成为互补品。因此，对不同商品进行聚类并尽可能将其摆放在附近位置，能够在很大程度上提升取货及出库的效率。本文简述两种方法用于聚类：

- 基于计算机的聚类分析。通过分析订单中货物的相关性，对各商品进行聚类分析，使关联性较大的商品储存在相近的储位；
- 基于经验的聚类分析。在计算机自动聚类的基础上，对自动聚类的结果进行补充与更正，从而能够更加满足实际情况及仓库的存储管理。

10.1.3　仓库分区与商品分级耦合

基于上述对于仓库分区与商品分级的讨论，能够确定各区以及各商品所对应的“热销程度”；基于此，将出货频率较高的商品与优先级较高的仓库库区进行对应，并使得距离较近的货格对应优先级较高的仓库库区，从而提高订单出库的效率，下列给出具体的分配策略。

Step 1　货物相关度计算

仓库中各货物的相关度主要体现在某一订单中，货物同时出现的概率的大

小。一般来说，两货物出现在同一个订单中的频率越大，则该两种商品之间的相关性就越强。因此，可通过 ρ 进行表示：

$$\rho_{ij}=\frac{n_{ij}}{n_i+n_j-n_{ij}}$$

其中，n_i 表示货物 i 出现在订单中的次数；n_{ij} 表示货物 i、j 同时出现在订单中的次数；用 ρ 来表示货物 i、j 之间的相关度。

Step 2　货物聚类

对各商品进行聚类分析，从而使得关联性大的商品能够被储存在相近的储位上；从而在拣货过程中，尽最大可能缩短拣货员的行走距离，以提高订单出库的效率[1]。

$$\max\sum_{g=1}^{G}\sum_{i=1}^{N-1}\sum_{j>i}^{N}\rho_{ij}X_{ig}X_{jg}$$

其中，$X_{ig}=\begin{cases}0, & \text{货物}i\text{未被分到}g\text{类中}\\ 1, & \text{货物}i\text{被分到}g\text{类中}\end{cases}$，且 $\sum_{g=1}^{G}X_{ig}=1,\sum_{g=1}^{G}X_{jg}=1\ (i,j=1,2,\ldots,N)$

Step 3　仓库分区与商品分级聚类耦合

基于上述聚类，可以得到不同类别的聚类中心以及该聚类下货物的热销程度（订购频率），结合 10.1.1 中的仓库分区，通过建立仓库分区与商品分级之间的耦合关系，使得订购次数越多的商品摆放在离复核台越近的地方。

$$\min\sum_{g=1}^{G}\sum_{h}^{M}F_gD_h$$

其中，F_g 表示聚类所得到的第 g 类货物平均订货频率；D_h 表示第 h 库存区域距离复核台的最短距离，并通过仓库分区来进行表示。G 代表聚类总数，M 表示分区 h 中的货位总数。

10.2　货物分配原则与建议

10.2.1　货物存储原则

货物存储是仓库管理中的核心问题，必须结合各仓库的管理情况、运作模式等采取较为合理的方式。基于此，下列概述了仓库存储的主要原则[1]：

- 稳定性原则。基于对货架受力的考虑，一般情况下，同一货架进行货格分配时，应该将质量较大的货物放置在货架的底层，以保证货架良好的稳定性。
- 先进先出原则。除了需要考虑上述热销品的存放问题，对于一般的商品，应合理布局仓库存储机制，并在出库时采用先进先出原则，以防止因存储时间变长而导致的货物变质。
- 效率优先原则。在货物出库的过程中，尽可能提高货物出库的效率，将出库/入库频率较高的货物存在离复核台较近的位置，最大可能提高出库/入库效

率。

- 货物相关原则。基于上述货物聚类的思想，尽量将相关性大的货物放在距离较近的位置，以方便相关性大的货物同时出库。
- ……

10.2.1 货物存储建议

在进行大型仓库的出入库以及商品拣货过程中，会出现许多订单中出现同一种商品的情况。倘若仍然采用“当有多个任务时，只能一个一个任务完成，不能在完成一个任务过程中拣另一个任务的货”的取货方式，无疑会大大降低拣货的效率，从而导致订单的挤压。

基于此，本文给出以下建议：

- 对一定时段内的订单商品进行统计，针对那些**存储距离较近**的商品，可以分配到同一任务中；即在拣货员进行拣货之间对订单中所涉及到的商品进行聚类统计。至此，能够很大程度上减少拣货员的行走距离，从而提高拣货的效率。
- 在离复核台较近的位置设置**畅销专区**，并分派专门拣货员进行热销品的取货。在进行订单的前期分析中，如果订购的商品仅包含畅销专区的商品，可安排该专门拣货员进行热销品的批量取货；在保证拣货效率的同时又兼顾了仓库取货巷道的畅通。
- 对于那些不仅**包含畅销品、而且还存在部分普通商品**的订单，仓库可以设置“分拣台”。指派拣货员进行热销品的批量取货，放置分拣台；而那些普通商品需等待其他拣货员完成取货后放置该处。待订单中所有商品均到达分拣台时，再统一送至复核台进行复核与打包。需要注意的是，“分拣台”的位置应设置在复核台周围，以降低路程运输消耗，提高出库效率。
- 此外，还可以根据仓库实际情况设置过度分区。这样可以防止某些货品的入库超过了该商品的库存大小，可以采取过度分区进行存放，以弥补了分区造成的不足，提高货格的通用性和利用率[2]。

11 模型评价与改进

11.1 模型评价

11.1.1 模型优点

1、本文对货格与复核台之间的不同位置关系分别研究，通过分类讨论得到不同情况下距离的计算方式，使得计算结果更加全面准确、贴近实际情况。

2、本文引入替换终点的概念，并借用图论中的知识，使得最短路径的模型简洁合理；同时通过遗传算法对模型进行求解，收敛速度较高。

3、本文从研究模型的内在机理出发到模型建立、求解、优化、验证、拓展，

整体框架较为完整，验证了模型较好的正确性，并给出了物流部门在仓库拣货的合理方案。

4、本文不仅仅局限于优化拣货路径，对货物摆放也做出了深入的探索，并提出了多种具有很强可操作性的建议，在实际问题中具有广泛应用。

11.1.2 模型缺点

1、在多人作业时没有考虑多人在同一巷道取货时可能产生的拥堵问题，与现实情况存在些许偏差。

2、由于 NP 难问题遍历复杂度较高，而本文所采用的基于遗传算法、贪心算法的求解策略只能寻找到该情况下的最优解，而不是全局最优。

11.2 模型改进

11.2.1 跨单取货

在模型中，我们假设每个人都只能一个一个任务完成，而实际情况中拣货员可根据自己后续任务中的商品灵活选择当前的取货策略。

我们可以统计一个拣货员所有任务单中涉及到的货物情况，若与后续订单中相同或是邻近的货物在当前订单中出现，在拣货车容量充足的情况下可以提前取货，这样可以缩短后续任务单的路程，大大提高了拣货员完成所有订单的总效率。

11.2.2 考虑多拣货员堵塞的情况

模型中我们假设拣货员在拣货过程中都能流畅作业，不存在拥堵情况。但在实际情况中，由于货架间的巷道可能过窄，当有拣货员正在巷道中下架商品或者从巷道中路过时，会因为堵塞影响到其他拣货员的工作，大大降低了拣货效率。

考虑可能发生的堵塞情况，在拣货员每次进入巷道前进行判断，若巷道中已有其他拣货员在工作则等待或者改变拣货路线。由此可以使得拣货的效率更高、拣货的方案更加合理。

11.2.3 货架的立体化设计

本文中研究的货架是单层的，这使得相同的面积能储存的货物较少，许多货物处于离复核台较远的位置，增加了拣货员的步行时间。

设计高层的立体仓库，能够提升仓储中心的储存量、提高了仓库空间的利用率，同时更多的货物能存放在离复核台较近的位置，使得拣货员的步行距离大为缩减，拣货效率得到大幅提高。

参考文献

[1] 杨景祥. 货架式仓储中心货位分配及拣货路径优化研究[D].北京交通大学,2016.

[2] 柳赛男,柯映林,李江雄,吕震.基于调度策略的自动化仓库系统优化问题研究[J].计算机集成制造系统,2006(09):1438-1443.

[3] 肖建,郑力.考虑需求相关性的多巷道仓库货位分配问题[J].计算机集成制造系统,2008,14(12):2447-2451.

附 录　MATLAB 源代码

```
clear
filename='C:\Users\MR.WEI\Desktop\附件 1：仓库数据.xlsx';
[data1]=importdata(filename);
counter=data1.data.x0x8D270x683C;
spotter=data1.data.x0x590D0x68380x53F0;
indent=data1.textdata.x0x4EFB0x52A10x5355(2:1201,:);
indent_num=data1.data.x0x4EFB0x52A10x5355;
n=size(counter,1);
for i=1:n
    str=char(data1.textdata.x0x8D270x683C(i+1,1));
    m1=str2num(str(2:4));
    m2=str2num(str(5:6));
    m3=mod(m1,8);
    m4=mod(m3,2);
    m5=floor(m1/8);
    m6=floor(m1/2);
    m7=ceil(m3/2);
    if(mod(m1,2)==1)
        L(i,1)=counter(i,1)-750;
    else
        L(i,1)=counter(i,1)+counter(i,3)+750;
    end
    L(i,2)=counter(i,2)+counter(i,4)/2;
    if(m3==0)
        L(i,3)=3*15+m2;
    else
        L(i,3)=(m7-1)*15+m2;
    end
    if(m4==1)
    L(i,4)=m5+m4;
    else
    L(i,4)=m5+2;
    end
    L(i,5)=m1;
    L(i,6)=m2;
    L(i,7)=counter(i,1);
    L(i,8)=counter(i,2);
end

for i=3001:3013
    if(i<=3008)
    L(i,1)=spotter(i-3000,1)+spotter(i-3000,3)/2;
```

```
    L(i,2)=spotter(i-3000,2)+spotter(i-3000,4);
    else
    L(i,1)=spotter(i-3000,1)+spotter(i-3000,3);
    L(i,2)=spotter(i-3000,2)+spotter(i-3000,4)/2;
    end
    L(i,7)=spotter(i-3000,1);
    L(i,8)=spotter(i-3000,2);
end
dis=zeros(3013);
for i=1:3013
    for j=1:3013
        if(i<j)
        if(i<=3000&&j<=3000)
            if(L(i,4)==L(j,4))
                dis(i,j)=abs(L(i,2)-L(j,2))+750*2;
            elseif(ceil(L(i,3)/15)~=ceil(L(j,3)/15))
                dis(i,j)=abs(L(i,1)-L(j,1))+abs(L(i,2)-L(j,2))+750*2;
            else
                p=L(i,6)+L(j,6);
                if(p<=15)
                    dis(i,j)=abs(L(i,1)-L(j,1))+800*(p-1)+750*4;
                else
                    dis(i,j)=abs(L(i,1)-L(j,1))+800*(31-p)+750*4;
                end
            end
        elseif(i<=3000||j<=3000)
            if(i<=3008&&j<=3008)
            dis(i,j)=abs(L(i,1)-L(j,1))+abs(L(i,2)-L(j,2))+750;
            else
                if(L(i,5)==1&&L(i,5)==3&&L(i,5)==5&&L(i,5)==7)
                dis(i,j)=abs(L(i,1)-L(j,1))+abs(L(i,2)-L(j,2))+750;
                else

d1=abs(L(i,1)-L(j,1))+L(i,6)*800-800/2+750+abs(L(i,2)-(L(i,6)*800-800/2+750)-L(
j,2))+750;

d2=abs(L(i,1)-L(j,1))+(15-L(i,6))*800+800/2+750+abs(L(i,2)-((15-L(i,6))*800+800
/2+750)-L(j,2))+750;
                if(d1<d2)
                    dis(i,j)=d1;
                else
                    dis(i,j)=d2;
                end
                end
```

```
            end
        else

dis(i,j)=abs(spotter(i-3000,1)-spotter(j-3000,1))+abs(spotter(i-3000,2)-spotter
(j-3000,2));
        end
        end
    end
end
for i=1:3013
    for j=1:3013
        if(i>j)
            dis(i,j)=dis(j,i);
        end
    end
end


%xlswrite('C:\Users\MR.WEI\Desktop\附件 4：计算结果.xlsx',dis,'Ques1','A1');
%-----------------------------------------------
% 第二问
% spotter_on=3001:3013;
% spotter_first=3001;
% spotter_last=0;
% [dislist, cityd] = tsp(indent,dis,spotter_on,spotter_first,spotter_last,1);
% [best,bestValue] =gaCircle(dislist, L,dis,cityd);
% best_list=name_1(best,dis,spotter_on,cityd);

%-----------------------------------------------
% 第三问
% spotter_on=[3003,3011];
% spotter_first=0;
% spotter_last=0;
% for i=1:5
%     [dislist, cityd,o_num] =
tsp(indent,dis,spotter_on,spotter_first,spotter_last,i+1,indent_num);
%     [best,bestValue] =gaCircle(dislist, L,dis,cityd);
%     b(i)=bestValue;
%     best_list=name_1(best,dis,spotter_on,cityd);
%     k=size(best_list,2);
%     for j=1:k
%         b_list{i,j}=best_list{j};
%     end
%     c(i,1:k)=cityd;
```

```
%     b_t(i,1:k)=best;
%     K_b(i)=k;
%     b_o(i,1:k-2)=o_num;
% end

%-------------------------------------------
% 第四问
spotter_on=[3001,3003,3010,3012];
spotter_first=0;
spotter_last=0;
c=zeros(100);
for i=49:49
    [dislist, cityd ,o_num] =
tsp(indent,dis,spotter_on,spotter_first,spotter_last,i,indent_num);
    [best,bestValue] =gaCircle(dislist, L,dis,cityd);
    b(i)=bestValue;
    best_list=name_1(best,dis,spotter_on,cityd);
    k=size(best_list,2);
    for j=1:k
        b_list{i,j}=best_list{j};
    end
    c(i,1:k)=cityd;
    b_t(i,1:k)=best;
    K_b(i)=k;
    b_o(i,1:k-2)=o_num;
end


j=1279;
for i=49:49
    clear b_2
    for k=1:K_b(i)
        if i<10
        b_2{k,1}=strcat('T000',num2str(i));  %第三问输出时这里是 i+1
        else
        b_2{k,1}=strcat('T00',num2str(i));
        end
        if(b_t(i,k)==1||b_t(i,k)==K_b(i))
        m=char(b_list{i,k});
        b_2{k,2}=str2num(m(3:4))+3000;
        else
        b_2{k,2}=c(i,b_t(i,k));
        end
        if(k==1||k==K_b(i))
```

```
            b_2{k,4}=0;
        else
            b_2{k,4}=b_o(i,b_t(i,k)-1);
        end
        b_2{k,3}= b_list{i,k};
    end
    b_3=strcat('N',num2str(j));
    xlswrite('C:\Users\MR.WEI\Desktop\xxx.xlsx',b_2,'Ques3',b_3);
    j=j+K_b(i);
end
```

```
function [dislist, cityd,o_num] =
tsp(indent,dis,spotter_on,spotter_first,spotter_last,s,indent_num)
[order,o_num]=fetch_order(indent,s,indent_num);
cityd=[spotter_first,order,spotter_last];
n=size(cityd,2);
for i=1:n
    for j=1:n
        if(cityd(i)~=0&&cityd(j)~=0)
            dislist(i,j)=dis(cityd(i),cityd(j));
        elseif(cityd(i)==0&&cityd(j)~=0)
            min=dis(spotter_on(1),cityd(j));
            for d=spotter_on
                if(dis(d,cityd(j))<min)
                    min=dis(d,cityd(j));
                end
            end
            dislist(i,j)=min;
        elseif(cityd(i)~=0&&cityd(j)==0)
            min=dis(cityd(i),spotter_on(1));
            for d=spotter_on
                if(dis(cityd(i),d)<min)
                    min=dis(cityd(i),d);
                end
            end
            dislist(i,j)=min;
        elseif(cityd(i)==0&&cityd(j)==0)
            dislist(i,j)=0;
        end
    end
end
end
```

```
function best_list=name_1(best,dis,spotter_on,cityd)
best_list=cell(1);
n=size(best,2);
for i=1:size(best,2)
    if(cityd(best(i))==0)
        if(i==1)
            min=dis(cityd(best(i+1)),spotter_on(1));
            for j=spotter_on
                if(dis(cityd(best(i+1)),j)<=min)
                    if(j<3010)
                    best_list{i}=strcat('FH0',num2str(j-3000));
                    else
                    best_list{i}=strcat('FH',num2str(j-3000));
                    end
                end
            end
        else
             min=dis(cityd(best(i-1)),spotter_on(1));
            for j=spotter_on
                if(dis(cityd(best(i-1)),j)<=min)
                    if(j<3010)
                    best_list{i}=strcat('FH0',num2str(j-3000));
                    else
                    best_list{i}=strcat('FH',num2str(j-3000));
                    end
                end
            end
        end
    else
        if(i==1||i==n)
             best_list{i}=cityd(best(i));
        else
            n_1=ceil(cityd(best(i))/15);
            if(n_1>=100)
                L_1=num2str(n_1);
            elseif(n_1>=10)
                L_1=strcat('0',num2str(n_1));
            else
                L_1=strcat('00',num2str(n_1));
            end
            n_2=cityd(best(i))-(n_1-1)*15;
            if(n_2>=10)
                L_2=num2str(n_2);
```

```
            else
                L_2=strcat('0',num2str(n_2));
            end
            best_list{i}=strcat('s',L_1,L_2);
        end
    end
end
end

function [order,o_num]=fetch_order(indent,r,indent_num)
i=1;j=1;d=1;
while i
    if(j>1200)
       break;
    else
    k=char(indent(j,1));
    end
    num=str2num(k(2:5));
    if(num==r)
        k1=char(indent(j,3));
        k2=str2num(k1(2:4));
        k3=str2num(k1(5:6));
        L_num=(k2-1)*15+k3;
        order(d)=L_num;
        o_num(d)=indent_num(j,1);
        d=d+1;
    elseif(num==r+1)
        i=0;
    end
    j=j+1;
end
end
%

%% 遗传算法求解最短路径
function [best,bestValue] =gaCircle(dislist, L,dis,cityd)

CityNum = size(dislist,1); % 数目，可以选 10, 30, 50, 75
%[dislist, Clist] = tsp(CityNum); % dislist 为之间相互的距离，Clist 为各的坐标

inn = 100; % 初始种群大小
gnMax = 20000;  % 最大代数
crossProb = 0.8; % 交叉概率
muteProb = 0.8; % 变异概率
```

```
% 随机产生初始种群
population = zeros(inn, CityNum); % population 为初始种群，包括多条染色体
for i = 1 : inn
    population(i,2:CityNum-1) = randperm(CityNum-2);
    for j=1:CityNum-1
        population(i,j)=population(i,j)+1;
    end
    population(i,CityNum)=CityNum;
end
[~, cumulativeProbs] = calPopulationValue(population, dislist); % 计算种群每条染色体的累计概率

generationNum = 1;
generationMeanValue = zeros(generationNum, 1); % 每一代的平均距离
generationMaxValue = zeros(generationNum, 1);  % 每一代的最短距离
bestRoute = zeros(inn, CityNum); % 最佳路径
newPopulation = zeros(inn, CityNum); % 新的种群
while generationNum < gnMax + 1
   for j = 1 : 2 : inn
      selectedChromos = select(cumulativeProbs);  % 选择操作，选出两条需要交叉编译的染色体，即父亲母亲
      crossedChromos = cross(population, selectedChromos, crossProb);  % 交叉操作，返回交叉后的染色体
      newPopulation(j, :) = mut(crossedChromos(1, :),muteProb);  % 对交叉后的染色体进行变异操作
      newPopulation(j + 1, :) = mut(crossedChromos(2, :), muteProb); % 对交叉后的染色体进行变异操作
   end
   population = newPopulation;  %产生了新的种群
   [populationValue, cumulativeProbs] = calPopulationValue(population, dislist);  % 计算新种群的适应度
   % 记录当前代最好和平均的适应度
   [fmax, nmax] = max(populationValue); % 因为计算适应度时取距离的倒数，这里面取最大的倒数，即最短的距离
   generationMeanValue(generationNum) = 1 / mean(populationValue);
   generationMaxValue(generationNum) = 1 / fmax;
   bestChromo = population(nmax, :);  % 前代最佳染色体，即对应的路径
   bestRoute(generationNum, :) = bestChromo; % 记录每一代的最佳染色体
   %Clist=coordinate(dislist,L,dis,bestChromo,cityd);
   %drawTSP(Clist, bestChromo, generationMaxValue(generationNum), generationNum, 0);
   generationNum = generationNum + 1;
end
```

```
[bestValue,index] = min(generationMaxValue);
%Clist=coordinate(dislist,L,dis,bestRoute(index, :),cityd);
%drawTSP(Clist, bestRoute(index, :), bestValue, index,1);
best=bestRoute(index, :);
% figure(2);
% plot(generationMaxValue, 'r');
% hold on;
% plot(generationMeanValue, 'b');
% grid;
% title('搜索过程');
% legend('最优解', '平均解');
% fprintf('遗传算法得到的最短距离: %.2f\n', bestValue);
% fprintf('遗传算法得到的最短路线');
% disp(bestRoute(index, :));
end

%-----------------------------------------------
% 计算获得所需坐标

function Clist=coordinate(dislist,L,dis,Route,cityd)
    n=size(Route,2);
    Clist=zeros(n,2);
    for i=1:n
        if(cityd(i)==0)  %%虚拟核算台坐标
            if(i==1)
            long=dislist(i,Route(2));
            for j=3001:3013
                k=dis(j,cityd(Route(2)));
                if(k==long)
                    tag=j;
                end
            end
            Clist(i,:)=L(tag,7:8);
            else
                long=dislist(i,Route(n-1));
            for j=3001:3013
                k=dis(j,cityd(Route(n-1)));
                if(k==long)
                    tag=j;
                end
            end
            Clist(i,:)=L(tag,7:8);
            end
        else   %其他
```

```
            Clist(i,:)=L(cityd(i),7:8);
        end
    end
end

%-----------------------------------------------
% 计算所有染色体的适应度
function [chromoValues, cumulativeProbs] = calPopulationValue(s, dislist)
inn = size(s, 1);  % 读取种群大小
chromoValues = zeros(inn, 1);
for i = 1 : inn
    chromoValues(i) = CalDist(dislist, s(i, :));  % 计算每条染色体的适应度
end
chromoValues = 1./chromoValues'; % 因为让距离越小，选取的概率越高，所以取距离倒数
% 根据个体的适应度计算其被选择的概率
fsum = 0;
for i = 1 : inn
    % 乘以 15 次方的原因是让好的个体被选取的概率更大（因为适应度取距离的倒数，若不乘次方，则个体相互之间的适应度差别不大），换成一个较大的数也行
    fsum = fsum + chromoValues(i)^15;
end
% 计算单个概率
probs = zeros(inn, 1);
for i = 1: inn
    probs(i) = chromoValues(i)^15 / fsum;
end
% 计算累积概率
cumulativeProbs = zeros(inn,1);
cumulativeProbs(1) = probs(1);
for i = 2 : inn
    cumulativeProbs(i) = cumulativeProbs(i - 1) + probs(i);
end
cumulativeProbs = cumulativeProbs';
end

%-----------------------------------------------
%“选择”操作，返回所选择染色体在种群中对应的位置
% cumulatedPro 所有染色体的累计概率
function selectedChromoNums = select(cumulatedPro)
selectedChromoNums = zeros(2, 1);
% 从种群中选择两个个体，最好不要两次选择同一个个体
for i = 1 : 2
   r = rand;  % 产生一个随机数
```

```
    prand = cumulatedPro - r;
    j = 1;
    while prand(j) < 0
        j = j + 1;
    end
    selectedChromoNums(i) = j; % 选中个体的序号
    if i == 2 && j == selectedChromoNums(i - 1)    % 若相同就再选一次
        r = rand;  % 产生一个随机数
        prand = cumulatedPro - r;
        j = 1;
        while prand(j) < 0
            j = j + 1;
        end
    end
end
end

%-----------------------------------------------
% “交叉”操作
function crossedChromos = cross(population, selectedChromoNums, crossProb)
length = size(population, 2); % 染色体的长度
crossProbc = crossMuteOrNot(crossProb);  %根据交叉概率决定是否进行交叉操作，1 则是，0 则否
crossedChromos(1,:) = population(selectedChromoNums(1), :);
crossedChromos(2,:) = population(selectedChromoNums(2), :);
if crossProbc == 1
    c1 = round(rand * (length - 2)) + 1;  %在[1,bn - 1]范围内随机产生一个交叉位 c1
    c2 = round(rand * (length - 2)) + 1;  %在[1,bn - 1]范围内随机产生一个交叉位 c2
    chb1 = min(c1, c2);
    chb2 = max(c1,c2);
    middle = crossedChromos(1,chb1+1:chb2); % 两条染色体 chb1 到 chb2 之间互换位置
    crossedChromos(1,chb1 + 1 : chb2)= crossedChromos(2, chb1 + 1 : chb2);
    crossedChromos(2,chb1 + 1 : chb2)= middle;
    for i = 1 : chb1 % 看交叉后，染色体上是否有相同编码的情况（路径上重复出现两个）。若有，则该编码不参与交叉
        while find(crossedChromos(1,chb1 + 1: chb2) == crossedChromos(1, i))
            location = find(crossedChromos(1,chb1 + 1: chb2) == crossedChromos(1, i));
            y = crossedChromos(2,chb1 + location);
            crossedChromos(1, i) = y;
        end
        while find(crossedChromos(2,chb1 + 1 : chb2) == crossedChromos(2, i))
            location = find(crossedChromos(2, chb1 + 1 : chb2) == crossedChromos(2,
```

```
i));
            y = crossedChromos(1, chb1 + location);
            crossedChromos(2, i) = y;
        end
    end
    for i = chb2 + 1 : length
        while find(crossedChromos(1, 1 : chb2) == crossedChromos(1, i))
            location = logical(crossedChromos(1, 1 : chb2) == crossedChromos(1, i));
            y = crossedChromos(2, location);
            crossedChromos(1, i) = y;
        end
        while find(crossedChromos(2, 1 : chb2) == crossedChromos(2, i))
            location = logical(crossedChromos(2, 1 : chb2) == crossedChromos(2, i));
            y = crossedChromos(1, location);
            crossedChromos(2, i) = y;
        end
    end
end
end

%-------------------------------------------------
%“变异”操作
% choromo 为一条染色体
function snnew = mut(chromo,muteProb)
length = size(chromo, 2); % 染色体的的长度
snnew = chromo;
muteProbm = crossMuteOrNot(muteProb);  % 根据变异概率决定是否进行变异操作，1 则是，0 则否
if muteProbm == 1
    c1 = round(rand*(length - 2)) + 1;  % 在 [1, bn - 1]范围内随机产生一个变异位
    c2 = round(rand*(length - 2)) + 1;  % 在 [1, bn - 1]范围内随机产生一个变异位
    chb1 = min(c1, c2);
    chb2 = max(c1, c2);
    x = chromo(chb1 + 1 : chb2);
    snnew(chb1 + 1 : chb2) = fliplr(x); % 变异，则将两个变异位置的染色体倒转
end
end

% 根据变异或交叉概率，返回一个 0 或 1 的数
function crossProbc = crossMuteOrNot(crossMuteProb)
test(1: 100) = 0;
l = round(100 * crossMuteProb);
test(1 : l) = 1;
n = round(rand * 99) + 1;
```

```
crossProbc = test(n);
end

%-------------------------------------------
% 计算一条染色体的适应度
% dislist 为所有相互之间的距离矩阵
% chromo 为一条染色体，即一条路径
function chromoValue = CalDist(dislist, chromo)
DistanV = 0;
n = size(chromo, 2); % 染色体的长度
for i = 1 : (n - 1)
    DistanV = DistanV + dislist(chromo(i), chromo(i + 1));
end
%DistanV = DistanV + dislist(chromo(n), chromo(1));
chromoValue = DistanV;
end

%-------------------------------------------
% 画图
% Clist 为坐标
% route 为一条路径
function drawTSP(Clist, route, generationValue, generationNum,isBestGeneration)
CityNum = size(Clist, 1);
for i = 1 : CityNum - 1
    plot([Clist(route(i), 1),Clist(route(i + 1), 1)],
[Clist(route(i),2),Clist(route(i+1),2)],'ms-','LineWidth',2,'MarkerEdgeColor','
k','MarkerFaceColor','g');
    text(Clist(route(i), 1),Clist(route(i), 2), ['  ', int2str(route(i))]);
    text(Clist(route(i+1), 1),Clist(route(i + 1), 2), ['  ',
int2str(route(i+1))]);
    hold on;
end
%plot([Clist(route(CityNum), 1), Clist(route(1), 1)], [Clist(route(CityNum), 2),
Clist(route(1),
2)],'ms-','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g');
title([num2str(CityNum),'TSP']);
if isBestGeneration == 0 && CityNum ~= 10
    text(5, 5, ['第 ',int2str(generationNum),' 代','  最短距离为 ',
num2str(generationValue)]);
else
    text(5, 5, ['最终搜索结果：最短距离 ',num2str(generationValue),',  在第
',num2str(generationNum),' 代达到']);
end
if CityNum == 10  % 因为文字显示位置不一样，所以将数目为 10 时单独编写
```

```
    if isBestGeneration == 0
        text(0, 0, ['第 ',int2str(generationNum),' 代','  最短距离为 ',
num2str(generationValue)]);
    else
        text(0, 0, ['最终搜索结果:最短距离 ',num2str(generationValue),', 在第 ',
num2str(generationNum),' 代达到']);
    end
end
hold off;
pause(0.005);
end


% for i=1:2
%     str=strcat(num2str(i),'.jpg');
%     A=imread(str);
%     [I,map]=rgb2ind(A,256);
%     if(i==1)
%         imwrite(I,map,'movefig.gif','DelayTime',0.1,'LoopCount',Inf)
%     else
%         imwrite(I,map,'movefig.gif','WriteMode','append','DelayTime',0.1)
%     end
% end

% clear
% filename='C:\Users\MR.WEI\Desktop\city.xls';
% [data1]=importdata(filename);
% data=data1.Sheet1(40:42,1:3);
% xishu=zeros(3);
% o1=data1.Sheet1(40:42,4);
% o2=data1.Sheet1(40:42,5);
% d1=data1.Sheet1(43,1:3);
% d2=data1.Sheet1(44,1:3);
%
% while(1)
%     x1=o2./o1;
%     x2=d2./d1;
%     for i=1:3
%         for j=1:3
%             xishu(i,j)=(x1(i,1)+x2(1,j))/2;
%             data(i,j)=data(i,j)*xishu(i,j);
%         end
%     end
%     for i=1:3
```

```
%         o1(i,1)=sum(data(i,:));
%         d1(1,i)=sum(data(:,i));
%     end
% end
j=1;num=zeros(49,2);jie=zeros(49,4);
dingdan=data1.textdata.x0x4EFB0x52A10x5355(2:1201,1:2);d_2=218;
for i=1:1299

    j1=ceil(j/2);
    j2=j-2*(j1-1);
    if(VarName15(i,1)>3000)
        jie(j1,j2)=VarName15(i,1)-3000;
        j=j+1;
    end
    num(j1,1)= num(j1,1)+xxx(i,1);
end
j1=0;d_12=0;d_2=0;
for i=1:1200
    ding=char(dingdan(i,2));
    d_1=str2num(ding(2:5));
    ding1=char(dingdan(i,1));
    d_11=str2num(ding1(2:5));
    if(d_12~=d_11)
        j1=j1+1;
        d_12=d_11;
    end
    if(d_1~=d_2)
        num(j1,2)= num(j1,2)+1;
        d_2=d_1;
    end
end

k1=zeros(1,2);k2=zeros(1,2);k3=zeros(1,2);k4=zeros(1,2);
for i=1:49
    for j=1:2
    if(jie(i,j)==1)
        k1(1,j)=k1(1,j)+1;
    elseif(jie(i,j)==3)
        k2(1,j)=k2(1,j)+1;
    elseif(jie(i,j)==10)
        k3(1,j)=k3(1,j)+1;
    elseif(jie(i,j)==12)
        k4(1,j)=k4(1,j)+1;
    end
```

```
    end
end
LK=data1.textdata.x0x4EFB0x52A10x5355;k_2=0;ji=0;task2=0;j=0;n_1=1;n_2=1;task=z
eros(49,1);
count=data1.data.x0x4EFB0x52A10x5355;
for i=1:1200
    TTT=char(LK(i+1,1));
    task1=str2num(TTT(2:5));
    if(task1~=task2)
        j=j+1;
        task2=task1;
    end
        if(count(i,1)>=3)
       task(j,1)= task(j,1)+count(i,1)*4;
        else
        task(j,1)= task(j,1)+count(i,1)*5;
        end
    k=char(LK(i+1,2));
    k_1=str2num(k(2:5));
    if(k_1~=k_2)
        ji=ji+1;
        k_2=k_1;
    else
    end
end
zuobiao=[4,0;
    13,0;
    0,8;
    0,17;
    8.5,0;
    31,0;
    0,21.5;
    ];
for i=1:5
    for j=1:5

long(i,j)=abs(zuobiao(i,1)-zuobiao(j,1))+abs(zuobiao(i,2)-zuobiao(j,2));
    end
end
long=long./1.5;
jie(:,3)=jie(:,3)+task;
jie(:,4)=zeros(49,1);
time=zeros(5,1);
people=zeros(9,49);
```

```
time_p=zeros(9,1);
[d1,d2]=sort(jie(:,3));
for i=1:49
    for j=1:2
        if(jie(i,j)==3)
            jie(i,j)=2;
        elseif(jie(i,j)==10)
            jie(i,j)=3;
        elseif(jie(i,j)==12)
            jie(i,j)=4;
        end
    end
end
jie(:,3)=jie(:,3)+t_ordz'/1500;

for i=1:49
    if(i<=5)
        [ord,t]=c_min(long,jie,0,i,1);
        [~,d2]=sort(time_p);
        time_p(d2(1,1))=time_p(d2(1,1))+t;
        for j=1:49
            if(people(d2(1,1),j)==0)
                people(d2(1,1),j)=jie(ord,2);
                break;
            end
        end
        time(i,1)=time(i,1)+t+num(ord,2)*30;
        jie(ord,4)=d2(1,1);
    else
        [~,d2]=sort(time);
        ttsp=d2(1,1);
        j_1=1;
        [~,d3]=sort(time_p);
        for j=1:49
            if(people(d3(1,1),j)==0)
                people(d3(1,1),j)=jie(ord,2);
                break;
            end
        end
        [ord1,t1]=c_max(long,jie,people(d3(1,1),j),ttsp,j_1,num);t=t1;

        while(1)

if(time(ttsp)>(time_p(d3(1,1))+t1)||time(ttsp)>(time_p(d3(1,1))+t)||j_1>49)
```

```
                break;
            else
                j_1=j_1+1;
                [ord,t]=c_max(long,jie,people(d3(1,1),j),ttsp,j_1,num);
                if t<t1
                    ord1=ord;
                    t1=t;
                end
            end
        end
        if(j_1==1)
            time_p(d3(1,1))=time_p(d3(1,1))+t1;
            time(d2(1,1),1)=time(d2(1,1),1)+num(ord1,2)*30;
            jie(ord1,4)=d3(1,1);
            people(d3(1,1),j)=ttsp;
            ord=ord1;
        else
            time_p(d3(1,1))=time_p(d3(1,1))+t1;
            time(d2(1,1),1)=time(d3(1,1),1)+num(ord1,2)*30;
            jie(ord1,4)=d3(1,1);
            people(d3(1,1),j)=ttsp;
        end
    end
end

function [ord,t]=c_min(long,jie,begin,end_1,k)
    for i=1:49
        if(begin==0)
        time(i)=jie(i,3)+long(jie(i,2),end_1);
        else
          time(i)=jie(i,3)+long(jie(i,1),begin)+long(jie(i,2),end_1);
        end
    end
    [d1,d2]=sort(time);k1=0;
    for i=1:49
        if(jie(d2(i),4)==0)
            k1=k1+1;
        end
        if(k==k1)
            break;
        end
    end
    t=d1(i);
    ord=d2(i);
```

```
end

function [ord,t]=c_max(long,jie,begin,end_1,k,num)
    for i=1:49
        if(begin==0)
        time(i)=jie(i,3)+long(jie(i,2),end_1);
        else
          time(i)=jie(i,3)+long(jie(i,1),begin)+long(jie(i,2),end_1);
        end
    end
    [~,d2]=sort(num(:,2));k1=0;
    for i=49:-1:1
        if(jie(d2(i),4)==0)
            k1=k1+1;
        end
        if(k==k1)
            break;
        end
    end
    t=time(d2(i));
    ord=d2(i);
End



j=1;num=zeros(49,2);jie=zeros(49,4);
dingdan=data1.textdata.x0x4EFB0x52A10x5355(2:1201,1:2);d_2=218;
for i=1:1299

    j1=ceil(j/2);
    j2=j-2*(j1-1);
    if(VarName15(i,1)>3000)
        jie(j1,j2)=VarName15(i,1)-3000;
        j=j+1;
    end
    num(j1,1)= num(j1,1)+xxx(i,1);
end
j1=0;d_12=0;d_2=0;
for i=1:1200
    ding=char(dingdan(i,2));
    d_1=str2num(ding(2:5));
    ding1=char(dingdan(i,1));
    d_11=str2num(ding1(2:5));
    if(d_12~=d_11)
```

```
        j1=j1+1;
        d_12=d_11;
    end
    if(d_1~=d_2)
        num(j1,2)= num(j1,2)+1;
        d_2=d_1;
    end
end



k1=zeros(1,2);k2=zeros(1,2);k3=zeros(1,2);k4=zeros(1,2);
for i=1:49
    for j=1:2
    if(jie(i,j)==1)
        k1(1,j)=k1(1,j)+1;
    elseif(jie(i,j)==3)
        k2(1,j)=k2(1,j)+1;
    elseif(jie(i,j)==10)
        k3(1,j)=k3(1,j)+1;
    elseif(jie(i,j)==12)
        k4(1,j)=k4(1,j)+1;
    end
    end
end
LK=data1.textdata.x0x4EFB0x52A10x5355;k_2=0;ji=0;task2=0;j=0;n_1=1;n_2=1;task=z
eros(49,1);
count=data1.data.x0x4EFB0x52A10x5355;
for i=1:1200
    TTT=char(LK(i+1,1));
    task1=str2num(TTT(2:5));
    if(task1~=task2)
        j=j+1;
        task2=task1;
    end
        if(count(i,1)>=3)
       task(j,1)= task(j,1)+count(i,1)*4;
        else
        task(j,1)= task(j,1)+count(i,1)*5;
        end
    k=char(LK(i+1,2));
    k_1=str2num(k(2:5));
    if(k_1~=k_2)
        ji=ji+1;
```

```
        k_2=k_1;
    else
    end
end
zuobiao=[4,0;
    13,0;
    0,8;
    0,17];
for i=1:4
    for j=1:4

long(i,j)=abs(zuobiao(i,1)-zuobiao(j,1))+abs(zuobiao(i,2)-zuobiao(j,2));
    end
end
long=long./1.5;
jie(:,3)=jie(:,3)+task;
jie(:,4)=zeros(49,1);
time=zeros(4,1);
people=zeros(9,49);
time_p=zeros(9,1);
[d1,d2]=sort(jie(:,3));
for i=1:49
    for j=1:2
        if(jie(i,j)==3)
            jie(i,j)=2;
        elseif(jie(i,j)==10)
            jie(i,j)=3;
        elseif(jie(i,j)==12)
            jie(i,j)=4;
        end
    end
end
jie(:,3)=jie(:,3)+t_ordz'/1500;

for i=1:49
    if(i<=4)
        [ord,t]=c_min(long,jie,0,i,1);
        [~,d2]=sort(time_p);
        time_p(d2(1,1))=time_p(d2(1,1))+t;
        for j=1:49
            if(people(d2(1,1),j)==0)
                people(d2(1,1),j)=jie(ord,2);
                break;
            end
```

```
        end
        time(i,1)=time(i,1)+t+num(ord,2)*30;
        jie(ord,4)=d2(1,1);
        kkkkk(i,2)=d2(1,1);
    else
        [~,d2]=sort(time);
        ttsp=d2(1,1);
        j_1=1;
        [~,d3]=sort(time_p);
        for j=1:49
            if(people(d3(1,1),j)==0)
                people(d3(1,1),j)=jie(ord,2);
                break;
            end
        end
        [ord1,t1]=c_max(long,jie,people(d3(1,1),j),ttsp,j_1,num);t=t1;

        while(1)
            if(time(ttsp)>(time_p(d3(1,1))+t1)||time(ttsp)>(time_p(d3(1,1))+t))
                break;
            else
                j_1=j_1+1;
                [ord,t]=c_max(long,jie,people(d3(1,1),j),ttsp,j_1,num);
            end
        end
        if(jie(ord,4)~=0)
            time_p(d3(1,1))=time_p(d3(1,1))+t1;
            time(d2(1,1),1)=time(d2(1,1),1)+num(ord1,2)*30;
            jie(ord1,4)=d3(1,1);
            people(d3(1,1),j)=ttsp;
            ord=ord1;
            kkkkk(i,2)=d3(1,1);
        else
            time_p(d3(1,1))=time_p(d3(1,1))+t;
            time(d2(1,1),1)=time(d2(1,1),1)+num(ord,2)*30;
            jie(ord,4)=d3(1,1);
            people(d3(1,1),j)=ttsp;
            kkkkk(i,2)=d3(1,1);
        end
    end
kkkkk(i,1)=ord;
end
for i=1:1298
    z{i,1}=(VarName14{i,1});
```

```
    z{i,2}=(VarName16(i,1));
    z{i,3}=VarName17{i,1};
    z{i,4}=VarName18(i,1);
end
result_100=cell(1);
bbb=1;
for i=1:9
    Location=1;val=0;
for j=1:49
    if(kkkkk(j,2)==i)
            end_1=people(Location,1);
            if (end_1==1)
                end_1=3001;
            elseif(end_1==2)
                end_1=3003;
            elseif(end_1==3)
                end_1=3010;
            elseif(end_1==4)
                end_1=3012;
            end
            if (i==9&&kkkkk(i,1)==8)
                kkkkkkkk=1;
            end
            zb=out_1(z,i,kkkkk(j,1),val,end_1);
            Location=Location+1;
            val=end_1;
            kk=size(zb,1);
    for b=bbb:bbb+kk-1
        result_100{b,1}=zb{b-bbb+1,1};
        result_100{b,2}=zb{b-bbb+1,2};
        result_100{b,3}=zb{b-bbb+1,3};
        result_100{b,4}=zb{b-bbb+1,4};
        result_100{b,5}=zb{b-bbb+1,5};
    end
    bbb=bbb+kk;
    end

end
end

function zb=out_1(z,n,j,begin,end_1)
j1=1;
for i=1:1298
    k1=char(z{i,1});
```

```
    if(size(k1,2)<5)
        kkk=1;
    end

    k2=str2num(k1(2:5));
    if k2==j
        zb{j1,1}=strcat('P',num2str(n));
        zb{j1,2}=z{i,1};
        zb{j1,3}=z{i,2};
        zb{j1,4}=z{i,3};
        zb{j1,5}=z{i,4};
        j1=j1+1;
    elseif(k2>j)
        break;
    end
end
if(begin~=0)
    zb{1,3}=begin;
    if(begin>=3010)
        zb{1,4}=strcat('FH',num2str(begin-3000));
    else
        zb{1,4}=strcat('FH0',num2str(begin-3000));
    end
end
if(end_1~=0)
    zb{j1-1,3}=end_1;
    if(end_1>=3010)
        zb{j1-1,4}=strcat('FH',num2str(end_1-3000));
    else
        zb{j1-1,4}=strcat('FH0',num2str(end_1-3000));
    end
end
end


function best_list=name_1(best,dis,spotter_on,cityd)
best_list=cell(1);
n=size(best,2);
for i=1:size(best,2)
    if(cityd(best(i))==0)
        if(i==1)
            min=dis(cityd(best(i+1)),spotter_on(1));
            for j=spotter_on
                if(dis(cityd(best(i+1)),j)<=min)
```

```
                    if(j<3010)
                    best_list{i}=strcat('FH0',num2str(j-3000));
                    else
                    best_list{i}=strcat('FH',num2str(j-3000));
                    end
                end
            end
        else
             min=dis(cityd(best(i-1)),spotter_on(1));
            for j=spotter_on
                if(dis(cityd(best(i-1)),j)<=min)
                    if(j<3010)
                    best_list{i}=strcat('FH0',num2str(j-3000));
                    else
                    best_list{i}=strcat('FH',num2str(j-3000));
                    end
                end
            end
        end
    else
        if(i==1||i==n)
             best_list{i}=cityd(best(i));
        else
            n_1=ceil(cityd(best(i))/15);
            if(n_1>=100)
                L_1=num2str(n_1);
            elseif(n_1>=10)
                L_1=strcat('0',num2str(n_1));
            else
                L_1=strcat('00',num2str(n_1));
            end
            n_2=cityd(best(i))-(n_1-1)*15;
            if(n_2>=10)
                L_2=num2str(n_2);
            else
                L_2=strcat('0',num2str(n_2));
            end
            best_list{i}=strcat('s',L_1,L_2);
        end
    end
end
end
```

```
function [ord,t]=c_min(long,jie,begin,end_1,k)
    for i=1:49
        if(begin==0)
        time(i)=jie(i,3)+long(jie(i,2),end_1);
        else
          time(i)=jie(i,3)+long(jie(i,1),begin)+long(jie(i,2),end_1);
        end
    end
    [d1,d2]=sort(time);k1=0;
    for i=1:49
        if(jie(d2(i),4)==0)
            k1=k1+1;
        end
        if(k==k1)
            break;
        end
    end
    t=d1(i);
    ord=d2(i);
end


function [ord,t]=c_max(long,jie,begin,end_1,k,num)
    for i=1:49
        if(begin==0)
        time(i)=jie(i,3)+long(jie(i,2),end_1);
        else
          time(i)=jie(i,3)+long(jie(i,1),begin)+long(jie(i,2),end_1);
        end
    end
    [~,d2]=sort(num(:,2));k1=0;
    for i=49:-1:1
        if(jie(d2(i),4)==0)
            k1=k1+1;
        end
        if(k==k1)
            break;
        end
    end
    t=time(d2(i));
    ord=d2(i);
end
```