浙江大学
ZHEJIANG UNIVERSITY

运筹学

实验报告

问题名称 智能图书馆派送路径优化

成　　员 周意尧、樊铧纬、戴承江、杨逍宇

日　　期 2024 年 6 月 11 日

指导老师 何衍

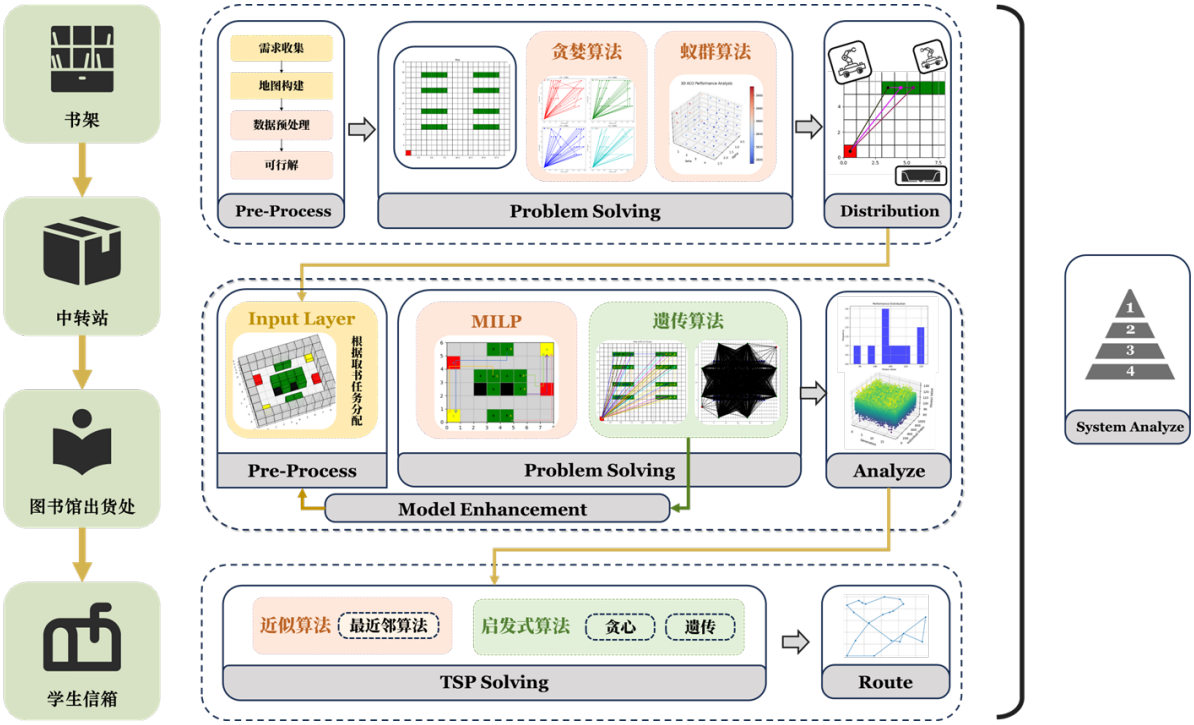
摘要

对于问题一，我们采用贪心算法来解决智能机器人在图书馆进行派送任务时的路径规划问题。目标是规划其运动路径，使其行驶路径最短。通过建立相应的数学模型，并考虑每个途径点只能访问一次等约束条件，我们直接利用贪心算法进行求解，得出最优路径，提升了 AGV 的取书效率。

对于问题二，针对多 AGV 智能机器人同时取书的场景，我们将此问题建模为 SD-VRP（Split Delivery Vehicle Routing Problem）问题。通过贪婪算法和蚁群算法的结合，解决了多辆 AGV 的协同调度问题，确保每个机器人在载满书籍或未取满情况下合理返回传送处，最终求解出总行驶路径最短的调度方案。

对于问题三，提出了一种基于混合整数规划（MILP）的方法，用于解决仓储环境中托盘与书籍的高效拣选和调度问题。我们建立了一个详细的数学模型，考虑了托盘内书本数量与预约订单匹配、托盘与书籍挑选节点的匹配、托盘放回空储位节点以及空托盘回收等多个约束条件。通过定义变量和约束条件，模型确保了从托盘中挑选出的书籍数量满足预约订单需求，同时最小化了托盘在仓库中的移动距离。模型的主要约束包括：托盘内书本数量与预约订单匹配约束、托盘与书籍挑选节点的匹配约束、托盘与回收处和储位的约束。

关键字：智能机器人　贪心算法　遗传算法　SDVRP　MILP



目录

一、 问题描述 …… 3
1.1 问题背景 …… 3
1.2 问题提出 …… 3
二、 问题分析 …… 4
2.1 问题一分析 …… 4
2.2 问题二分析 …… 4
2.3 问题三分析 …… 5
三、 模型假设 …… 5
四、 符号说明 …… 5
五、 问题一的模型的建立和求解 …… 7
5.1 模型建立 …… 7
5.2 模型求解 …… 7
六、 问题二的模型的建立和求解 …… 9
6.1 模型建立 …… 9
6.2 模型求解 …… 9
七、 问题三的模型的建立和求解 …… 15
7.1 模型建立 …… 15
7.2 模型求解 …… 18
八、 模型的分析与检验 …… 24
九、 模型的评价 …… 26
9.1 优点 …… 26
9.2 缺点 …… 26
9.3 模型改进 …… 26
参考文献 …… 27
A 附录 文件列表 …… 27
B 附录 人员分工 …… 27
C 附录 工作流程 …… 27

一、 问题描述

1.1 问题背景

在这样一个全民阅读的时代，国民人均阅读量不断增加，人们对便捷、高效的借阅服务需求也与日俱增。为了满足这一需求，浙江大学在去年又开了一座新的主图书馆，并引入先进的智能机器人技术，为读者提供更方便的借阅条件。该系统采用“网上预定——自动分拣——送书上门”的一条龙服务模式，读者只需在手机端预约心仪的书籍，系统便会根据预约时间将书籍分配到不同的托盘中。AGV 智能机器人会根据预设的路线，将装载书籍的托盘从储位搬运到拣选工位，完成书籍的打包，最后将包裹送至读者的指定地点。这种新型的借阅模式，不仅能有效地提高图书馆资源的利用率和服务效率，还能为读者提供更加便捷、人性化的借阅体验，并极大地提升图书馆的服务水平，为推动全民阅读做出积极贡献。

本任务总共分为三个步骤:

1. 书籍挑选

同学在手机通过浙大钉的图书馆应用可以进行线上借阅，在收到预约订单之后，我们规定，在每天的固定时间 11 点和 8 点对于该点之前的预约信息进行处理，即将预约书籍从货架上挑选出来，放在托盘中，然后送往传送处，最后送往书籍发货处。(注：在该取书情况下，不同时间处理的订单将会储存在不同的托盘中)

2. 书籍送货

在书籍通过托盘送到发货处的货架上时, 机器人会在每天的 13 点进行处理，订单一次处理 50 单 (考虑到每天预约书籍的量并没有那么多而设置的)，处理订单的步骤分为三步。

(1) AGV 搬运载有书本的托盘到空闲拣选工位;

(2) AGV 搬运拣选完成的托盘从工位回到仓库内空储位;

(3) AGV 搬运拣选完成的空托盘从工位到托盘回收处;

对于该问题来说，我们认为每一本书就是一个订单，不考虑多本书一起打包发送这件事情

3. 派送到户

小车将已经打包好的书籍送往每一位同学的信箱中。

1.2 问题提出

问题 1: 送书小车统筹调度问题

现在智能小车要进行派送任务，已知其要经过的途径点，请你规划其运动路径使其经过所有途径点并使行驶路径最短。

问题 2: 取书 AGV 统筹调度问题

考虑 4 辆 AGV 智能机器人同时取书，机器人从起始位置出发，然后前往书架取书了，每一个机器人一次最多承载 5 本书，一旦取了 5 本书之后，必须前往传送处将书放下，再回去取书 (机器人可以自行决定装载几本书后前往传送处，即未取满 8 本书也可放回传送处)，请你规划 4 个机器人的路径使得其总行驶路径最短。

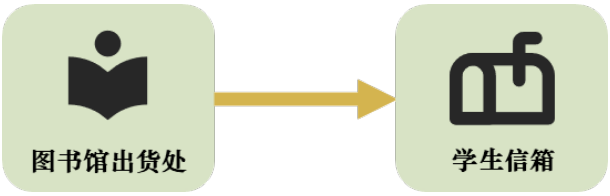
问题 3:AGV 指派调度问题

AGV 智能机器人从起始位置出发，先前往储位取下装有预定订单中书籍的托盘，然后送往挑选处将订单中需要的书籍挑选出去，挑选完成之后，如果托盘中没有书籍，则前往托盘回收处，如果仍然有书籍，则将托盘送回货架上，任务到此结束。不考虑小车完成任务后回到出发点的这一段距离，请你指派小车和托盘，使得 AGV 智能机器人取托盘以及托盘运行总路径长度最短。

二、 问题分析

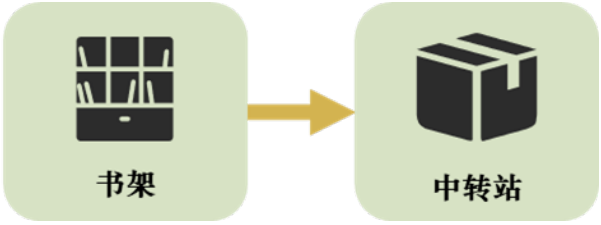
2.1 问题一分析

对于问题一，这是一个 TSP 旅行商问题，我们建立了相应的数学模型，考虑每个城市必须被访问仅一次、从起点出发最后回到起点等约束条件，考虑到图书馆的派送任务不算复杂我们直接利用贪心算法进行求解。



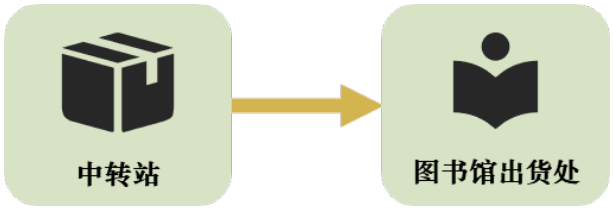
2.2 问题二分析

对于问题二，这个 SDVRP（Split Delivery Vehicle Routing Problem）属于组合优化问题（Combinatorial Optimization Problem），我们采用贪婪算法和蚁群算法来解决取书 AGV 统筹调度问题。我们建立了一个详细的数学模型，考虑了运输小车给书架的配送量和书架需求量的匹配，考虑运输小车容量的约束，考虑运输支路限制条件等多个约束条件。过定义变量和约束条件，模型确保了从运输小车的运输总量满足预约订单需求，同时最小化了运输小车在图书馆取数过程中的移动距离。



2.3 问题三分析

对于问题三，提出了一种基于混合整数规划（MILP）的方法，用于解决仓储环境中托盘与书籍的高效拣选和调度问题。我们建立了一个详细的数学模型，考虑了托盘内书本数量与预约订单匹配、托盘与书籍挑选节点的匹配、托盘放回空储位节点以及空托盘回收等多个约束条件。通过定义变量和约束条件，模型确保了从托盘中挑选出的书籍数量满足预约订单需求，同时最小化了托盘在仓库中的移动距离。可以先通过对小规模数据的实验，证明模型能够有效地最小化总路程，并确保所有书籍被正确拣选。再进一步扩展到大规模数据，并使用更大的地图进行实验。



三、 模型假设

为简化问题，本文做出以下假设:

- 假设 1: 对于所有问题，不考虑机器人的实际体积可能造成的碰撞问题
- 假设 2: 对于问题二，我们假设一个货架只能储存一个托盘来存放书籍
- 假设 3: 对于问题二，我们只考虑完成取书——挑拣——回收/放回这一任务的最短距离，不考虑 AGV 小车的调度问题，即不考虑多辆小车的调度问题

四、 符号说明

符号	说明
N	有需要取书的书架的数量
C	运输小车的数量
d_{ij}	书架 i 和书架 j 之间的距离
Q	运输小车的装载容量
x_{ijk}	运输小车 k 是否通过边 (i,j) 的二进制标识符
y_{ik}	运输小车 k 在书架 i 的配送数量
q_i	第 i 个书架的需求量
s	小车一次取书任务中经过书架的数量

符号	说明
H	书籍预约种类集合
I	托盘集合
Q	储位节点集合
M	拣选工作台节点集合
R	空盘回收节点集合
C_h	第 h 本书籍的订单需求
Q_i	第 i 个托盘的初始储位
C_h	第 h 本书籍的订单需求
Q_i	第 i 个托盘的初始储位
O_{ih}	第 i 个托盘里拥有第 h 种书的数量
J_{iq}	第 i 个托盘初始位于第 q 个储位节点，储存在 q 则为 1，否则为 0
D_{qm}	第 q 个储位至第 m 个拣选工作台的最短行走距离
D_{mq}	第 m 个拣选工作台到至第 q 个储位的最短行走距离
D_{mr}	第 m 个拣选工作台到至第 r 个空盘回收位的最短行走距离
x_i	如果托盘 i 被挑选去拣选工位则为 1，否则为 0
y_i	如果托盘 i 的书本全部被挑选则为 1，否则为 0
k_{ih}	托盘 i 在拣选工位应被挑选走的第 h 种书的数量
p_{im}	托盘 i 选择去第 m 个拣选工作台分拣书本则为 1，否则为 0
g_{iq}	托盘 i 选择回到第 q 个储位则为 1，否则为 0
f_{ir}	托盘 i 选择去第 r 个空盘回收位则为 1，否则为 0
d_{iqm}	托盘 i 从第 q 个储位去第 m 个拣选工作台的行走距离
d_{imq}	托盘 i 第 m 个拣选工作台回到第 q 个储位的行走距离
d_{imr}	托盘 i 从第 m 个拣选工作台到第 r 个空盘回收位的行走距离
z_q	第 q 个储位上的托盘被指派则为 1，否则为 0
d_{ij}	第 i 点到第 j 点的距离
x_{ij}	i 点和 j 点之间有线相连

五、 问题一的模型的建立和求解

5.1 模型建立

我们要使配送小车的运送总路径最短，所以定义目标函数为

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{1}$$

每个城市必须被访问一次且仅一次

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in n \tag{3}$$

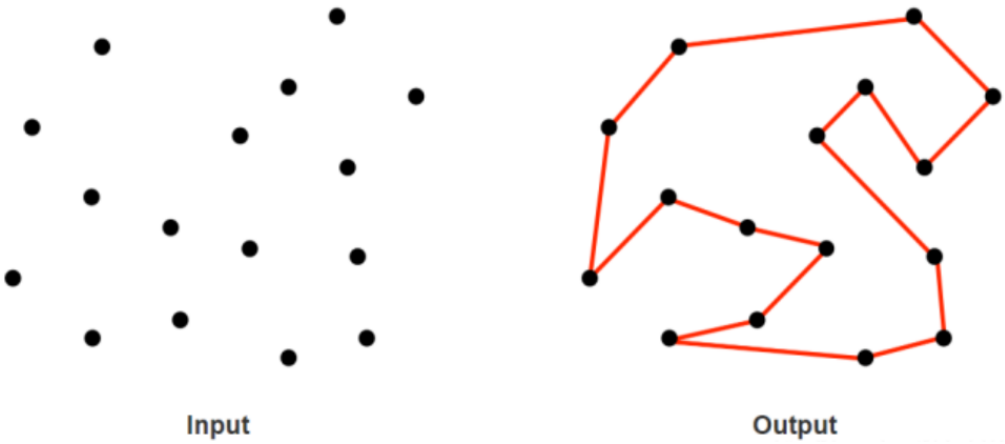
从起点城市出发，回到起点城市:

$$\sum_{i=1}^{n} x_{i1} = 1 \tag{4}$$

$$\sum_{j=1}^{n} x_{1j} = 1 \tag{5}$$

避免子回路

$$\sum_{i=1, i \neq j}^{n} \sum_{k=1, k \neq i, k \neq j}^{n} x_{ik} x_{kj} \leq n - 1, \forall j \in n \tag{6}$$



5.2 模型求解

对于这样一个旅行商问题，我们将上述的约束条件编写成 python 代码，利用贪心算法进行求解。

```
最佳路径:
(0, 0) (5, 13) (13, 28) (21, 35) (25, 37) (27, 38) (33, 44) (31, 48) (22, 50)
 (11, 44) (9, 45) (7, 47) (0, 43) (4, 37) (26, 13) (31, 12) (33, 19) (44, 12)
 (46, 33) (16, 0) (0, 0)
路径长度:  242.97992405372588
```

图 1　生成的 TSP 最佳路径

我们首先处理所有需要配送的书籍订单，将其与位置绑定生成一个字典。接着使用 TSP 旅行商问题的求解代码进行计算。

绘制出其路径如下图所示

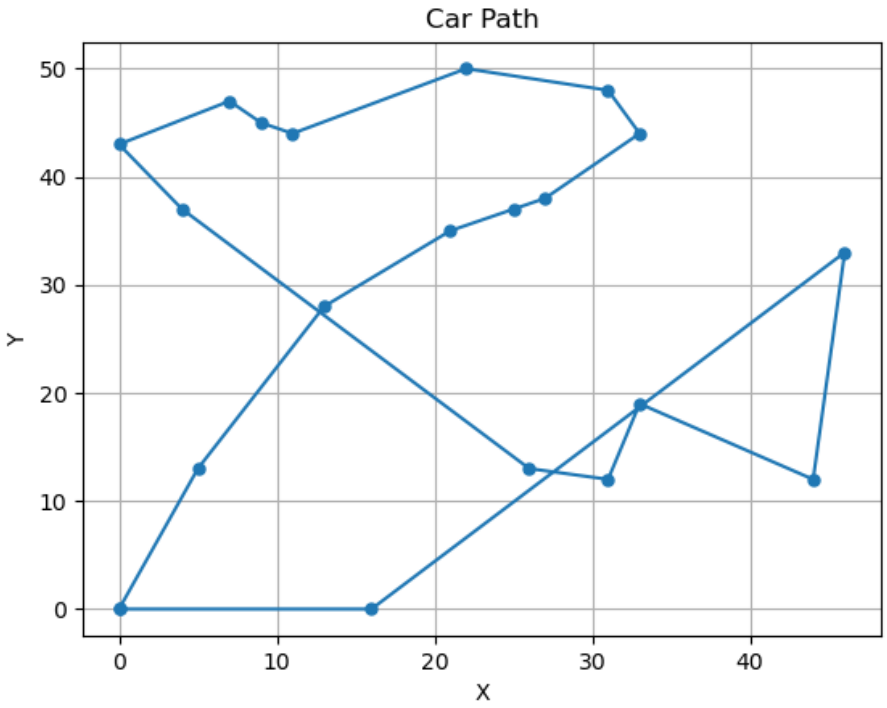


图 2　可视化 TSP 最佳路径

六、 问题二的模型的建立和求解

6.1 模型建立

目标函数

$$Z = \min \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{M} x_{ijk} \cdot d_{ij} \tag{7}$$

约束第 k 辆运输小车给书架的配送量不能超过书架的需求量，且只有在运输小车给顾客进行配送时 y_{ik} 才有意义

$$0 \leq y_{ik} \leq q_i, i \in N, k \in C \tag{8}$$

至少有一辆运输小车为书架进行配送服务

$$\sum_{i=0}^{N} \sum_{k=1}^{M} x_{ijk} \geq 1, j \in N \tag{9}$$

到达某点的车辆数应该等于离开该点的车辆数——不会停止不动

$$\sum_{i=0}^{N} x_{ipk} - \sum_{j=0}^{N} x_{pjk} = 0, p \in N, k \in C \tag{10}$$

所有书架的取书任务必须全部被完成；

$$\sum_{k=1}^{M} y_{ik} = q_i, i \in N \tag{11}$$

运输小车容量约束，即运输小车的容量必须大于等于运输小车在运输途中为顾客配送的总数量；

$$\sum_{i=1}^{N} y_{ik} \leq Q, k \in C, 0 < q_i < Q, i \in N \tag{12}$$

消除支路限制条件

$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} x_{ijk} \leq |s| - 1, 2 \leq |s| \leq n - 1 \tag{13}$$

6.2 模型求解

对于这样一个 SDVRP 问题，我们将上述的约束条件编写成 python 代码进行求解

Step1: 预处理：我们先对数据进行预处理，生成要图书馆书籍列表，生成书架列表，并且随机选出 500 本书生成借书列表。同时我们生成一层图书馆的地图

Step2: 求解过程

思路 1: 贪婪算法

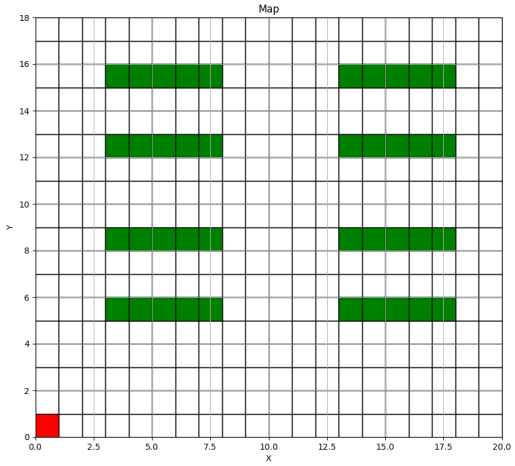


图 3　libaray map

贪婪算法实现：我们先对建立的数学模型采用贪婪算法，建立 SDVRP 模型进行求解，求解得到最佳路线和最佳距离

为了更好的阐释贪婪算法求解 SDVRP 问题的步骤，使用伪代码说明算法流程

Algorithm 1 贪婪算法求解最优路径

```
function generate_initial_solution
    初始化解决方案列表和剩余需求
    while 剩余需求大于 0 do
        初始化路线，从原点开始
        初始化剩余容量为车辆容量
        while 剩余容量大于 0 且剩余需求大于 0 do
            找到可访问的书架并随机访问一个书架
            if 剩余容量大于等于书架需求 then
                将书架添加到路线中
                更新剩余容量和需求
            else
                将书架添加到路线中
                更新书架需求和剩余容量为 0
            end if
        end while
        将路线添加到解决方案中
    end while
    返回解决方案
end function
```

```
function solve(max_iterations)
    生成初始解决方案, 设置初始解为最佳解
    计算初始解决方案的总距离
    for 最大迭代次数 do
        生成新解并评估其总距离
        if 新解的总距离小于当前最佳解 then
            更新最佳解和最佳距离
        end if
        记录当前最佳距离到距离历史记录
    end for
    根据路径长度将最佳解分配给小车
    返回最佳解和最佳距离
end function
```

结果如下图所示，我们可以看出其收敛速度很快，最终找到了总路径较为优秀的解

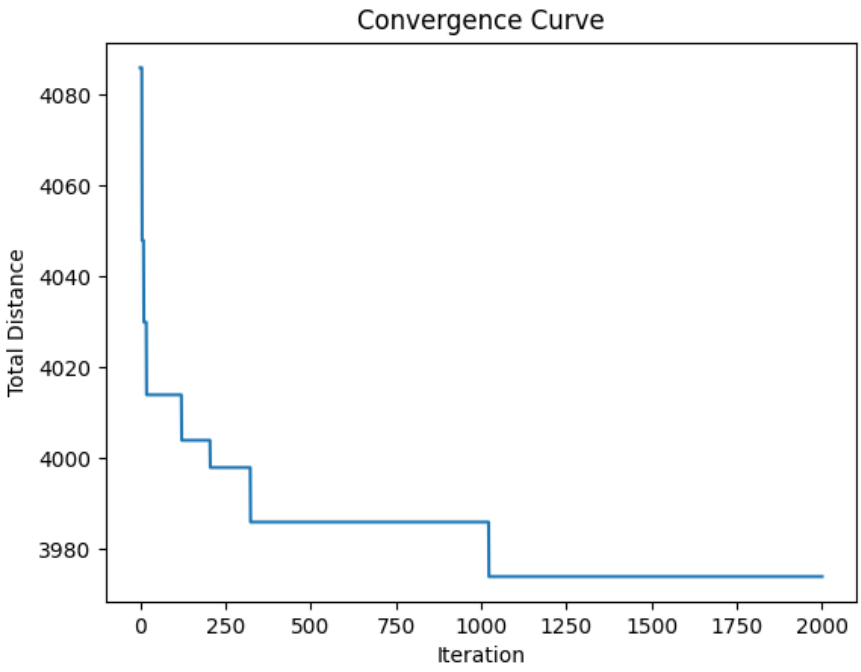


图 4　convergence curve

思路 2: 蚁群算法

对于这样一个 SDVRP 问题，我们采用 aco 蚁群算法来求解该问题

蚁群算法求解 SDVRP 问题的步骤，同样使用伪代码说明算法流程

Algorithm 2 蚁群算法求解最优路径

```
function ACO
    初始化蚁群
    初始化信息素矩阵
    for iteration = 1 to num_iterations do
        for 每只蚂蚁 do
            构建解决方案
            计算路径长度
            if 当前路径长度 < 最佳路径长度 then
                更新最佳路径和最佳路径长度
            end if
        end for
        更新信息素
        打印当前迭代的最佳路径长度
    end for
    return 最佳路径, 最佳路径长度
end function
function ConstructSolution(蚂蚁)
    for 每辆车 do
        while 存在未访问的书架 do
            选择下一个要访问的书架
            更新小车路径和书架的访问状态
        end while
        小车返回起点
    end for
end function
function UpdatePheromones(蚁群)
    依据路径长度更新信息素
end function
```

结果如下图所示，我们可以看出相对于贪婪算法其收敛速度更快，需要的迭代次数更少，并且能够得出更为优秀的最优路径

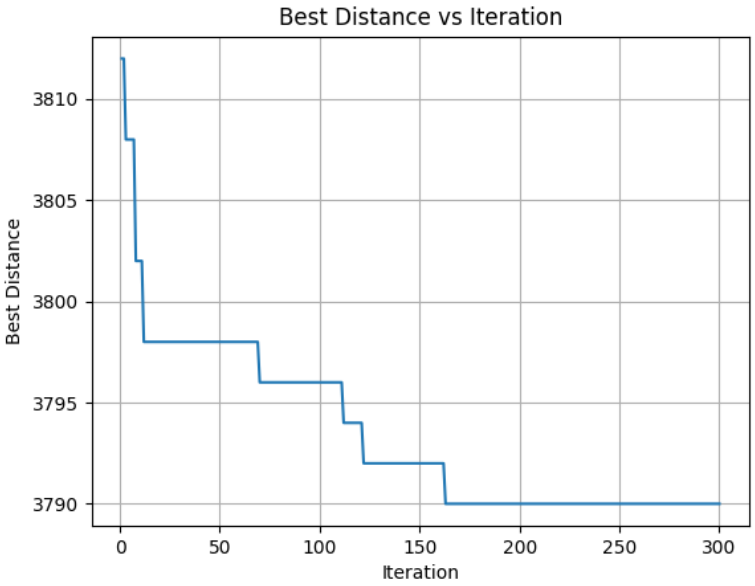


图 5　aco convergence

下图显示求得的小车总的最优路径

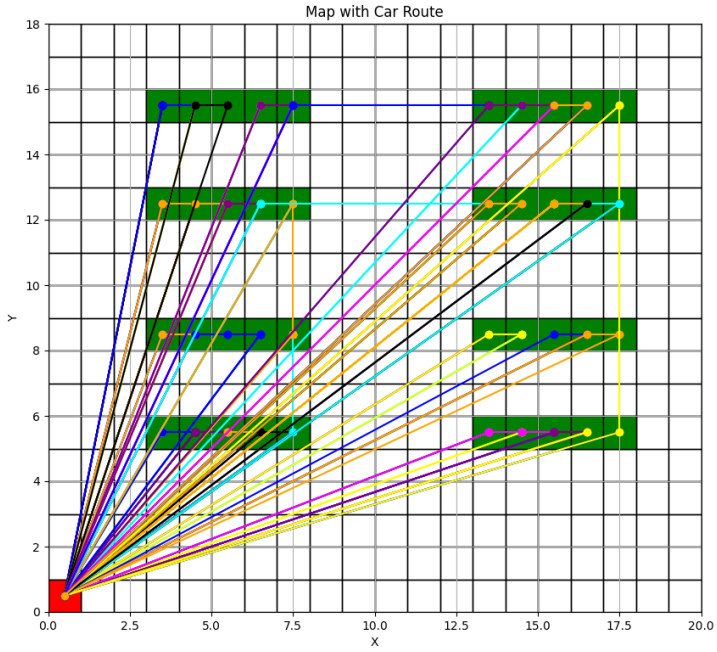


图 6　vehicle with route

在寻找到最佳路线后，我们尝试将路线分配给四个运输小车，采用贪心算法，使每辆小车的运输任务趋于一致。最后求得每辆运输小车各自的运输任务路径。

Algorithm 3 贪心算法分配小车任务

```
function split_routes(routes, num_vehicles)
    计算每条路线的总距离
    将路线按距离从大到小排序
    初始化车辆路线列表和车辆负载列表
    for 每条排序后的路线 do
        找到负载最小的小车
        将路线分配给该小车
        更新小车负载
    end for
    返回小车路线列表
end function
```

下图显示四辆小车分别的运输路径。

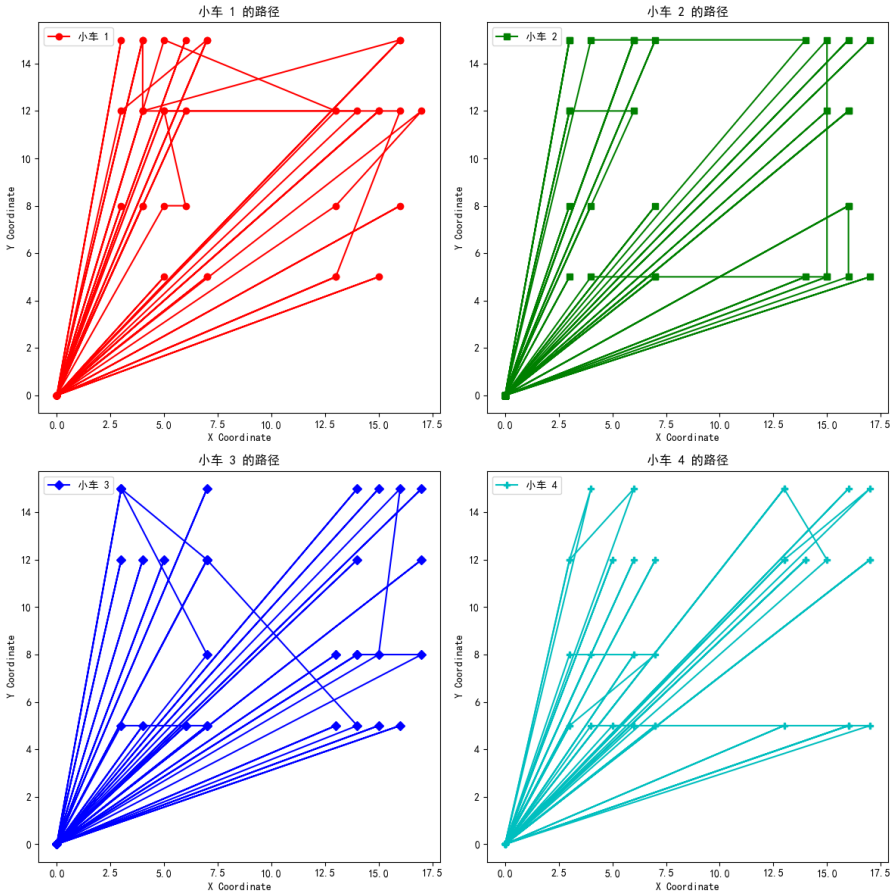


图 7　vehicle route

七、 问题三的模型的建立和求解

7.1 模型建立

目标函数

$$min\sum_{i\in I}(d_{iqm}+d_{imq}+d_{imr}) \tag{14}$$

1. 托盘内书本数量与预约订单匹配约束

虑托盘 i 在拣选工位应被挑选走的第 h 种书的数量小于第 i 个托盘里拥有第 h 种书得数量

$$0\leq k_{ih}\leq O_{ih},\forall i\in I,h\in H \tag{15}$$

考虑若托盘 i 被挑选出前往书本拣选节点，则从托盘 i 中挑选出来的书本数量总和应大于 0

$$\sum_{h\in H}k_{ih}+M(1-x_i)>0,\forall i\in I \tag{16}$$

考虑若托盘 i 没有被挑选出前往书本拣选节点，则从托盘 i 中挑选出来的书本数量总和应等于 0

$$\sum_{h\in H}k_{ih}-\sum_{h\in H}O_{ih}\times x_i\leqslant 0,\forall i\in I \tag{17}$$

限制从所有托盘中取出的第 h 种书本的数量应等于预约中需求的数量

$$\sum_{i\in I}k_{ih}=C_h,\forall i\in I,h\in H \tag{18}$$

托盘中的书籍全部被挑选走: 当且仅当托盘 i 中的书籍被完全挑选走时，y_i 才等于 1。

$$1\leqslant y_i+M\times\sum_{h\in H}(O_{ih}-k_{ih}) \tag{19}$$

如果没有被挑选完，y_i 等于 0

$$y_i\times\sum_{h\in H}O_{ih}\leqslant\sum_{h\in H}k_{ih} \tag{20}$$

2. 托盘与书籍挑选节点的匹配

考虑托盘 i 至多前往一个拣选工作台

$$\sum_{m\in M}p_{im}\leqslant 1,\forall i\in I \tag{21}$$

考虑若托盘 i 前往拣选工作台，限制其一定会被指派前往一个拣选工作台

$$\sum_{m\in M}p_{im}-x_i=0,\forall i\in I \tag{22}$$

考虑若托盘 i 从储位 q 被挑选出前往拣选工作台 m，则托盘 i 从储位 q 至拣选工作台 m 的行走路线长度 d_{iqm} 应大于从储位 q 至拣选工作台 *m* 的最短路径 D_{qm}

$$d_{iqm} + M(3 - x_i - p_{im} - J_{iq}) \geqslant D_{qm}, \forall i \in I, m \in M, q \in Q \tag{23}$$

考虑若托盘 i 从储位 q 没有被挑选出前往拣选工作台 m，则托盘 i 从储位 q 至拣选工作台 m 的行走路线长度 d_{iqm} 应等于 0

$$d_{iqm} - Mx_i \leqslant 0, \forall i \in I, m \in M, q \in Q \tag{24}$$

3. 如果托盘还有书籍，则需要放到空储位节点

考虑若托盘 i 没有被挑选出前往拣选工位，则托盘 i 中必有书籍存在

$$x_i - y_i \geqslant 0, \forall i \in I \tag{25}$$

考虑托盘 i 在被拣选完后，托盘可能为空也可能非空，至多前往一个储位

$$\sum_{q \in Q} g_{iq} \leqslant 1, \forall i \in I \tag{26}$$

考虑如果托盘 i 中的书籍未被全部挑选出来，即托盘 i 中被拣选之后还有剩余书籍，则为其分配一个储位

$$y_i + \sum_{q \in Q} g_{iq} = x_i, \forall i \in I \tag{27}$$

考虑若托盘 i 没有被挑选出前往拣选工作台，则不需要为其分配储位

$$x_i - \sum_{q \in Q} g_{iq} \geqslant 0, \forall i \in I \tag{28}$$

考虑任意一个托盘被挑选完后只能前往储位或空盘回收位中的一个

$$\sum_{q \in Q} g_{iq} + \sum_{r \in R} f_{ir} = x_i, \forall i \in I \tag{29}$$

考虑托盘 i 在拣选之后分配的储位仅能为空储位

$$g_{iq} - z_q \leqslant 0, \forall i \in I, q \in Q \tag{30}$$

约束一个储位只能有一个托盘

$$\sum_{i \in I} g_{iq} \leqslant 1, \forall q \in Q \tag{31}$$

考虑托盘 i 没有被挑选出前往拣选工作台，或托盘中的书籍被挑选空，则托盘从拣选工作台 m 至空储位 q 的行走路线长度应为 0

$$d_{imq} - M(x_i - y_i) \leqslant 0, \forall i \in I, m \in M, q \in Q \tag{32}$$

考虑托盘 i 被挑选出前往拣选工作台并且前往拣选工作台 m，挑选完成后托盘非空，则托盘从拣选工作台至空储位 q 的行走路线长度 d_{imq} 应大于从拣选工作台 m 至储位 q 的最短路径 D_{mq}

$$d_{imq} + M(4 - x_i + y_i - p_{im} - g_{iq} - z_q) \geqslant D_{mq}, \forall i \in I, m \in M, q \in Q \tag{33}$$

若托盘 i 被挑选走，则其所在的储位 q 是空储位

$$z_q + J_{iq} - x_i \leqslant 1, \forall i \in I, q \in Q \tag{34}$$

4. 若托盘无书籍，则考虑托盘与回收节点的匹配

考虑托盘 i 中的书籍全部被挑选走，则从托盘 i 中挑选出来的书籍数量总和应等于托盘 i 中拥有的书籍数量总和

$$\sum_{h \in H} k_{ih} - \sum_{h \in H} O_{ih} + M(1 - y_i) \geqslant 0, \forall i \in I \tag{35}$$

托盘 i 在被拣选完商品后，至多前往一个空盘回收位

$$\sum_{r \in R} f_{ir} \leq 1, \forall i \in I \tag{36}$$

考虑如果托盘 i 被挑选后没有书籍，则分配一个回收处，如果挑选后还有书籍，则不分配回收处

$$\sum_{r \in R} f_{ir} = y_i, \forall i \in I \tag{37}$$

托盘 i 没有被挑选出前往拣选工作台，则不需要为其分配空盘回收位

$$x_i - \sum_{r \in R} f_{ir} \geqslant 0, \forall i \in I \tag{38}$$

考虑托盘 i 被挑选出前往拣选工作台并且前往拣选工作台 m，挑选完成后托盘非空，则托盘从拣选工作台至空储位 q 的行走路线长度 d_{imr} 应大于从拣选工作台 m 至储位 q 的最短路径 D_{mr}

$$d_{imr} + M(4 - x_i - y_i - p_{im} - f_{ir}) \geqslant D_{mr}, \forall i \in I, m \in M, q \in Q \tag{39}$$

7.2 模型求解

思路 1:pulp 求解器

对于这样一个 MLP 问题，我们将上面的约束条件直接编写 python 代码进行求解，利用 Python 的 PULP 库即可。

但是对于大数据来说，求解为了验证其正确性，我们首先采用了小数据进行求解，来验证其正确性。

对于地图的颜色标注，我们有以下规定:

GREEN: 货架，用来存放装有书的托盘

YELLOW: 托盘回收处

BLACK: 障碍物，智能机器人不能通行

RED: 书籍挑拣点

Example1：

小地图，预定书籍 = 当前处理书籍

我们指定有 5 个托盘，分别存储在下图 2 的货架 1,3,5,7,9 中。每个托盘分别存储着 3 本书，该例中，当前处理书籍是托盘内的所有书，因此所有托盘都被指派且将到达回收处。根据 solve1.py 的求解结果 (见图一)，由 diqm，dimq，dimr 的值可以画出如下托盘路径，与期望的最短路径相同。

```
d_iqm solution:
d_iqm[0][1][0] = 5.0
d_iqm[1][3][0] = 4.0
d_iqm[2][5][1] = 3.0
d_iqm[3][7][1] = 2.0
d_iqm[4][9][1] = 5.0

d_imq solution:

d_imr solution:
d_imr[0][0][1] = 4.0
d_imr[1][0][1] = 4.0
d_imr[2][1][0] = 3.0
d_imr[3][1][0] = 3.0
d_imr[4][1][0] = 3.0
```

图 8　运行结果

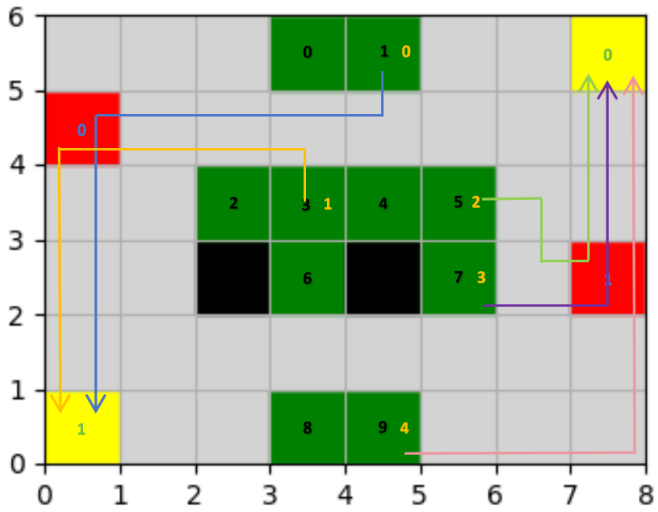


图 9　路径图示

Example2： 小地图，预定书籍 > 当前处理书籍

类似 Example1，不同点在于，当前处理书籍设置成部分库存。根据 solve2.py 的求解结果，由 diqm，dimq，dimr 的值可以画出如下托盘路径，与期望的最短路径相同。

下面详细说明 Example1 与 Example2 的区别以及测试目的:

Example2 在要处理的书籍中去除了托盘 2 的书籍 18, 和托盘 3 的书籍 4。直观上初步认为托盘 2 和托盘 3 将回到储位。但结果显示托盘 0 回到储位 2，托盘 2 去了回收处 0，托盘 3 回到了储位 7。这是由于托盘 0 内有书籍 18 和 4，而托盘 0-> 挑拣处 0-> 储位 2+ 托盘 2-> 挑拣处 1-> 回收处 0 的值小于托盘 0-> 挑拣处 0-> 回收处 1+ 托盘 2-> 挑拣处 1-> 储位 3。该例巧妙地说明了我们的程序确实是向着总路程最小的方向走。

```
d_iqm solution:
d_iqm[0][1][0] = 5.0
d_iqm[1][3][0] = 4.0
d_iqm[2][5][1] = 3.0
d_iqm[3][7][1] = 2.0
d_iqm[4][9][1] = 5.0

d_imq solution:
d_imq[0][0][2] = 3.0
d_imq[2][1][7] = 2.0

d_imr solution:
d_imr[1][0][1] = 4.0
d_imr[3][1][0] = 3.0
d_imr[4][1][0] = 3.0
```

图 10　运行结果

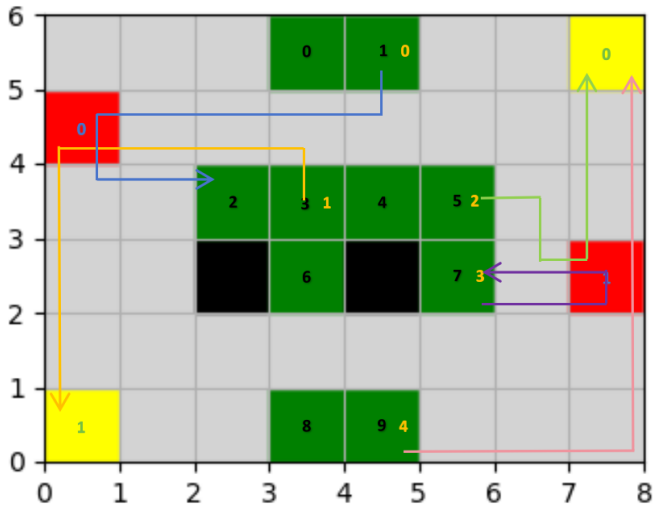


图 11　路径图示

对于大数据来说，原来的小地图并不能满足，因此我们构建新地图，如下图所示

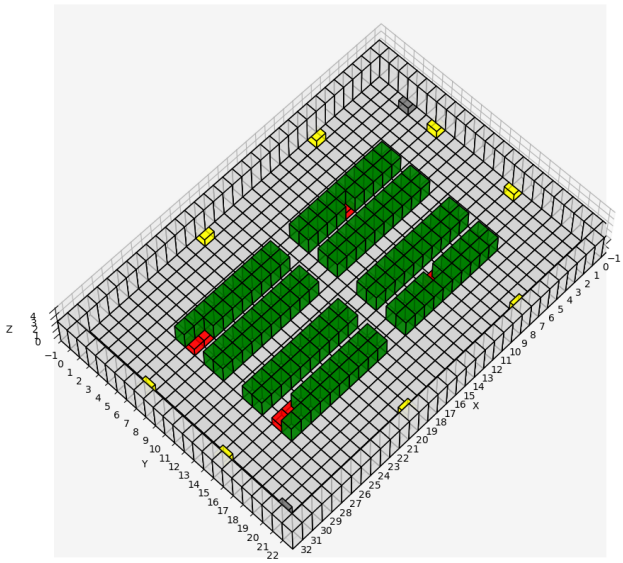


图 12　大数据地图

Example3：大地图，预定书籍 = 当前处理书籍

我们指定有 20 个托盘，存储在下图的 154 个货架中。每个托盘分别存储不同数量的书，该例中，当前处理书籍是托盘内的所有书，因此所有托盘都被指派且将到达回收处。根据 solve3.py 的求解结果，该例结果为，所有托盘被运到挑拣处并到达回收处的总路程为 370

```
Objective value:                370.00000000
Enumerated nodes:               15041
Total iterations:               196911
Time (CPU seconds):             21.52
Time (Wallclock seconds):       21.52

Option for printingOptions changed from normal to all
Total time (CPU seconds):       22.07   (Wallclock seconds):       22.07
```

图 13　最短路径

三步的距离如下图所示

```
x solution:   y solution:   d_iqm solution:          d_imq solution:

x[0] = 1.0    y[0] = 1.0    d_iqm[0][37][6] = 7.0
x[1] = 1.0    y[1] = 1.0    d_iqm[1][100][6] = 17.0   d_imr solution:
x[2] = 1.0    y[2] = 1.0    d_iqm[2][131][3] = 10.0   d_imr[0][6][1] = 3.0
x[3] = 1.0    y[3] = 1.0    d_iqm[3][97][6] = 12.0    d_imr[1][6][1] = 3.0
x[4] = 1.0    y[4] = 1.0    d_iqm[4][58][6] = 11.0    d_imr[2][3][0] = 6.0
x[5] = 1.0    y[5] = 1.0    d_iqm[5][101][6] = 18.0   d_imr[3][6][1] = 3.0
x[6] = 1.0    y[6] = 1.0    d_iqm[6][63][6] = 16.0    d_imr[4][6][1] = 3.0
x[7] = 1.0    y[7] = 1.0    d_iqm[7][141][5] = 12.0   d_imr[5][6][1] = 3.0
x[8] = 1.0    y[8] = 1.0    d_iqm[8][4][6] = 12.0     d_imr[6][6][1] = 3.0
x[9] = 1.0    y[9] = 1.0    d_iqm[9][8][6] = 16.0     d_imr[7][5][0] = 13.0
x[10] = 1.0   y[10] = 1.0   d_iqm[10][136][6] = 19.0  d_imr[8][6][1] = 3.0
x[11] = 1.0   y[11] = 1.0   d_iqm[11][40][6] = 10.0   d_imr[9][6][1] = 3.0
x[12] = 1.0   y[12] = 1.0   d_iqm[12][51][3] = 18.0   d_imr[10][6][1] = 3.0
x[13] = 1.0   y[13] = 1.0   d_iqm[13][142][5] = 11.0  d_imr[11][6][1] = 3.0
x[14] = 1.0   y[14] = 1.0   d_iqm[14][26][6] = 14.0   d_imr[12][3][0] = 6.0
x[15] = 1.0   y[15] = 1.0   d_iqm[15][98][6] = 15.0   d_imr[13][5][0] = 13.0
x[16] = 1.0   y[16] = 1.0   d_iqm[16][92][3] = 13.0   d_imr[14][6][1] = 3.0
x[17] = 1.0   y[17] = 1.0   d_iqm[17][113][3] = 9.0   d_imr[15][6][1] = 3.0
x[18] = 1.0   y[18] = 1.0   d_iqm[18][102][6] = 19.0  d_imr[16][3][0] = 6.0
x[19] = 1.0   y[19] = 1.0   d_iqm[19][35][3] = 16.0   d_imr[17][3][0] = 6.0
                                                      d_imr[18][6][1] = 3.0
                                                      d_imr[19][3][0] = 6.0
```

图 14　各步距离

Example4：大地图，预定书籍 > 当前处理书籍

由于大数据量带来的地图过大带来的维度过高，对于 MLP 的求解器负荷过高，故设置较少的托盘，现在设置 12 个托盘，运行结果如下图所示。

```
Objective value:                181.00000000
Enumerated nodes:               214
Total iterations:               2549
Time (CPU seconds):             1.41
Time (Wallclock seconds):       1.41

Option for printingOptions changed from normal to all
Total time (CPU seconds):       1.72   (Wallclock seconds):       1.72
```

图 15　最短路径

根据 solve3.py 的求解结果，该例结果为，所有托盘被运到挑拣处并到达回收处的总路程为 181

三步的距离如下图所示

```
x solution:
x[0] = 0.0
x[1] = 1.0
x[2] = 1.0
x[3] = 1.0
x[4] = 1.0
x[5] = 1.0
x[6] = 1.0
x[7] = 1.0
x[8] = 1.0
x[9] = 1.0
x[10] = 1.0
x[11] = 0.0

y solution:
y[0] = 0.0
y[1] = 1.0
y[2] = 1.0
y[3] = 1.0
y[4] = 1.0
y[5] = 1.0
y[6] = 1.0
y[7] = 1.0
y[8] = 1.0
y[9] = 1.0
y[10] = 1.0
y[11] = 0.0
```

```
d_iqm solution:
d_iqm[1][153][3] = 7.0
d_iqm[2][65][6] = 18.0
d_iqm[3][122][6] = 20.0
d_iqm[4][51][3] = 18.0
d_iqm[5][125][3] = 18.0
d_iqm[6][74][3] = 12.0
d_iqm[7][4][6] = 12.0
d_iqm[8][38][6] = 8.0
d_iqm[9][118][6] = 15.0
d_iqm[10][77][6] = 11.0

d_imq solution:

d_imr solution:
d_imr[1][3][0] = 6.0
d_imr[2][6][1] = 3.0
d_imr[3][6][1] = 3.0
d_imr[4][3][0] = 6.0
d_imr[5][3][0] = 6.0
d_imr[6][3][0] = 6.0
d_imr[7][6][1] = 3.0
d_imr[8][6][1] = 3.0
d_imr[9][6][1] = 3.0
d_imr[10][6][1] = 3.0
```

图 16　各步距离

思路 2: 遗传算法

对于这样一个 MLP 问题，我们用遗传算法与 deap 库来求解该问题

Example1：小地图，预定书籍 = 当前处理书籍

运行结果为最短路程：97, 如图17所示

```
gen     nevals
0       100
1       51
2       59
3       39
4       38
5       50
6       46
7       51
8       40
9       57
10      51
11      37
12      53
13      46
14      45
15      60
16      38
17      42
18      35
19      39
20      51
Multiple solutions found:
Best solution fitness: 97
```

图 17　迭代结果

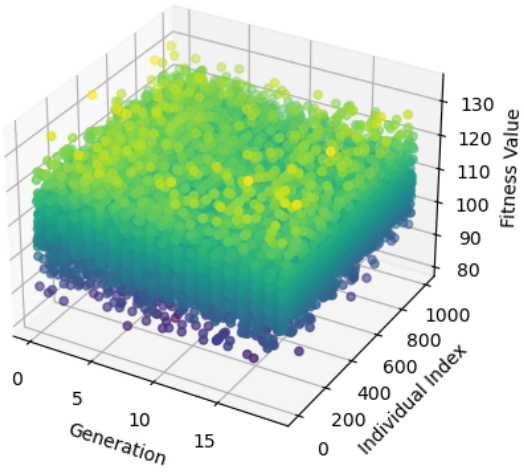


图 18　图形可视化

Example2：小地图，预定书籍 > 当前处理书籍

运行结果为最短路程：102, 如图19所示

```
gen     nevals
0       100
1       41
2       42
3       50
4       48
5       51
6       40
7       41
8       48
9       44
10      47
11      50
12      36
13      52
14      54
15      40
16      40
17      48
18      44
19      50
20      41
Multiple solutions found:
Best solution fitness: 102
```

图 19　迭代结果

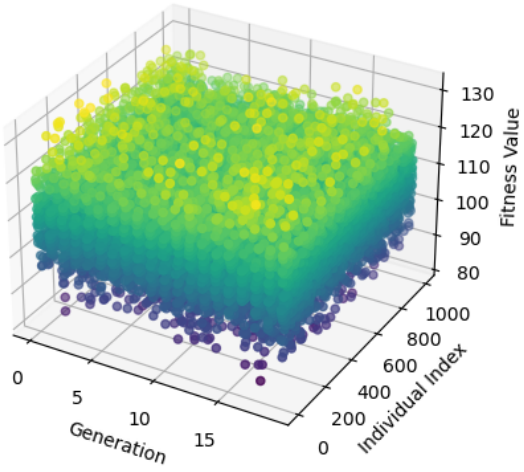


图 20　图形可视化

Example3：大地图，预定书籍 = 当前处理书籍

运行结果为最短路程：24596

```
gen     nevals
0       100
1       36
2       43
3       38
4       48
5       43
6       44
7       45
8       47
9       49
10      43
Multiple solutions found:
Best solution fitness: 24596
```

图 21　迭代结果

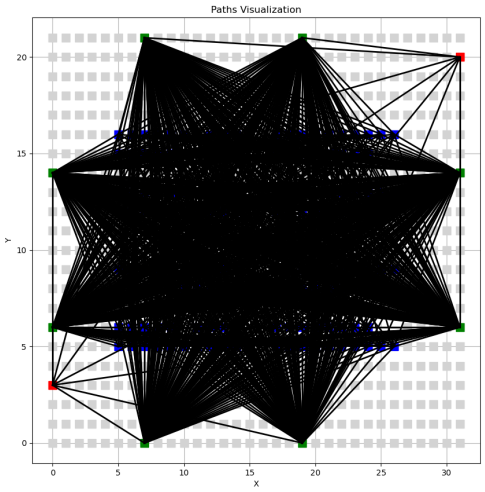


图 22　路径可视化

Example4：大地图，预定书籍 > 当前处理书籍

运行结果为最短路程：14715

```
gen     nevals
0       100
1       55
2       40
3       42
4       41
5       43
6       56
7       46
8       45
9       49
10      40
Multiple solutions found:
Best solution fitness: 14715
```

图 23　迭代结果

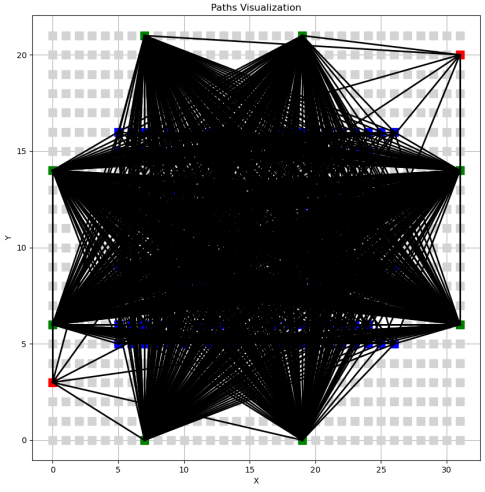


图 24　路径可视化

八、 模型的分析与检验

针对问题二

我们选择对蚁群算法进行参数敏感性分析，蚁群算法的参数指标主要是 alpha,beta 和 evaporation_rates 信息素浓度，以下图像是三个指标对结果数据的影响。

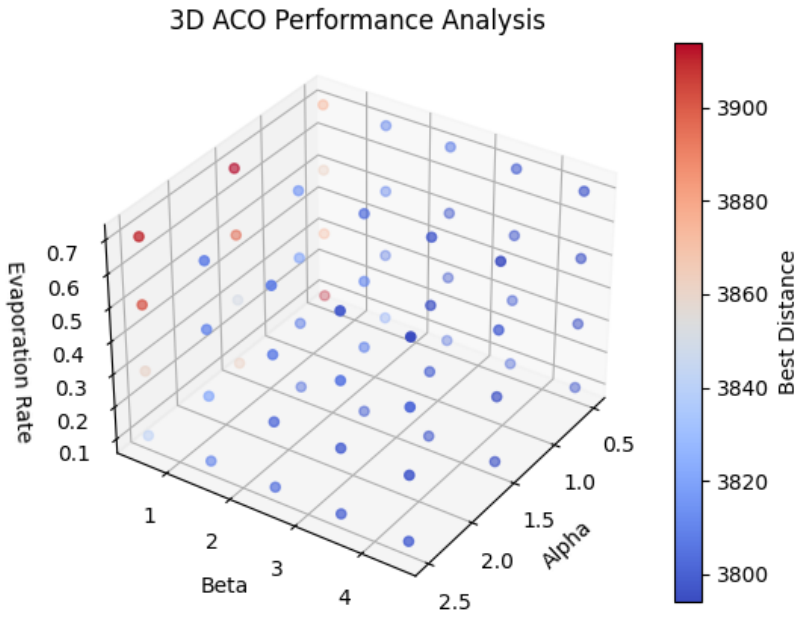


图 25　参数敏感性分析

1. 参数影响的直观观察: 从图形表示中可以观察到，代表蚁群优化 (ACO) 算法找到的最佳路径长度或成本的点的颜色随着参数 *alpha*、*beta* 和 *evaporation rate* 的变化而变化，这表明这些参数确实影响了算法的最终结果。红色点代表更高的成本，而蓝色点表示更低的成本，显示了在某些参数组合下算法找到了更短的路径。

2.Alpha 的影响: *Alpha* 参数控制信息素的重要性。从图像上看，当 *Alpha* 增加时，路径长度的变化似乎不是单调的，这可能意味着存在一个最优的 *Alpha* 值范围，而不是越大或越小越好。

3.Beta 的影响: *Beta* 参数控制启发式信息的影响力。图中显示，*Beta* 值的变化同样没有表现出明显的单调趋势，这表明启发式信息在平衡信息素影响力时起到关键作用。

4.Evaporation Rate 的影响: 信息素的蒸发率似乎对算法性能有显著影响。在较低的蒸发率（接近 0.1）时，算法找到的路径似乎更优，这可能表明较低的蒸发率有助于保留有价值的路径信息，从而指导搜索过程。

5. 参数间的交互效应: 图中数据点的分布可能还揭示了参数间的复杂交互关系。例如，在特定的 *alpha* 和 *beta* 组合下，某些 *evaporation rates* 可能更有效，这表明调整单

一参数可能不足以显著改善性能，而需要综合考虑多个参数的调整。

针对问题三：

对于遗传算法进行误差分析，我们进行了多次运算并对其进行误差分析

对于 Example1，其均值、标准差以及最坏最好值如下

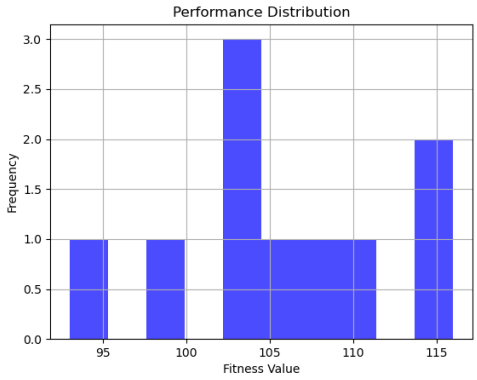


图 26　运行结果

```
Mean: 105.8
Standard Deviation: 6.896375859826668
Best Result: 93
Worst Result: 116
```

图 27　数学指标

对于 Example2，其均值、标准差以及最坏最好值如下

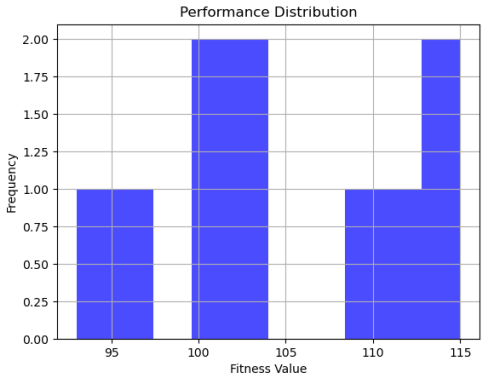


图 28　运行结果

```
Mean: 104.7
Standard Deviation: 7.430343195303969
Best Result: 93
Worst Result: 115
```

图 29　数学指标

九、 模型的评价

9.1 优点

- 问题二考虑了多辆 AGV 的协同工作，能够有效分配和利用资源。
- 问题二通过贪婪算法和蚁群算法求解，能够找到较优的路径，减少总行驶距离。
- 问题三基于混合整数规划的方法，能够精确优化托盘与书籍的拣选和调度问题。
- 问题三考虑了多种约束条件，如托盘数量、书籍匹配、储位等，保证了模型的准确性和实用性。

9.2 缺点

- 问题二中蚁群算法对参数设置敏感，参数选择不当可能影响求解效果。
- 问题二蚁群算法计算复杂度较高，求解时间较长，尤其在大规模问题中。
- 问题三模型复杂，变量和约束较多，实现和维护的难度较大。
- 问题三计算资源需求高：MILP 求解需要较高的计算资源，对于大规模问题，求解时间可能较长。

9.3 模型改进

- 对于问题二随着 AGV 数量增加，协调多辆 AGV 的路径规划和调度变得更加复杂。蚁群算法和贪婪算法在处理多 AGV 问题时可能需要较长的计算时间。可以考虑使用分布式算法或多智能体系统来提高计算效率。在处理大规模预约数据时，模型的计算复杂度和时间会显著增加。可以采用并行计算技术或分布式计算架构来加速求解过程。
- 对于问题二，考虑如果增加了多个仓库或存储区，需要考虑不同仓库之间的协调和调度问题。可以扩展模型以包含跨仓库调度的约束和变量。复杂约束处理：在实际应用中，可能会有更多的约束条件（如 AGV 的维护时间、不同类型书籍的处理优先级等）。模型需要具有灵活性，可以通过调整混合整数规划模型的约束条件来适应新的需求。

参考文献

[1] 杨文哲. 基于蚁群算法的需求可拆分车辆路径问题研究[M]. 北京: 吉林大学, 2020.

[2] 申栋栋. 智能仓库多移动机器人拣货路径规划研究[M]. 北京: 北京交通大学, 2019.

[3] 复杂无人仓搬运机器人协同调度机制与方法研究[M]. 武汉: 武汉大学, 2019.

[4] 熊昕霞. 智能仓库中多移动机器人路径规划研究[D]. 浙江: 浙江理工大学, 2021.

附录 A　文件列表

文件名	功能描述
Problem1	问题一代码
Problem2	问题二代码
Problem3	问题三代码
README	

附录 B　人员分工

成员	任务分工	成绩分配
周意尧	问题一、三，报告撰写	25%
戴承江	问题三，报告撰写	25%
樊铧纬	问题二，PPT 制作	25%
杨逍宇	问题二，报告撰写	25%

附录 C　工作流程