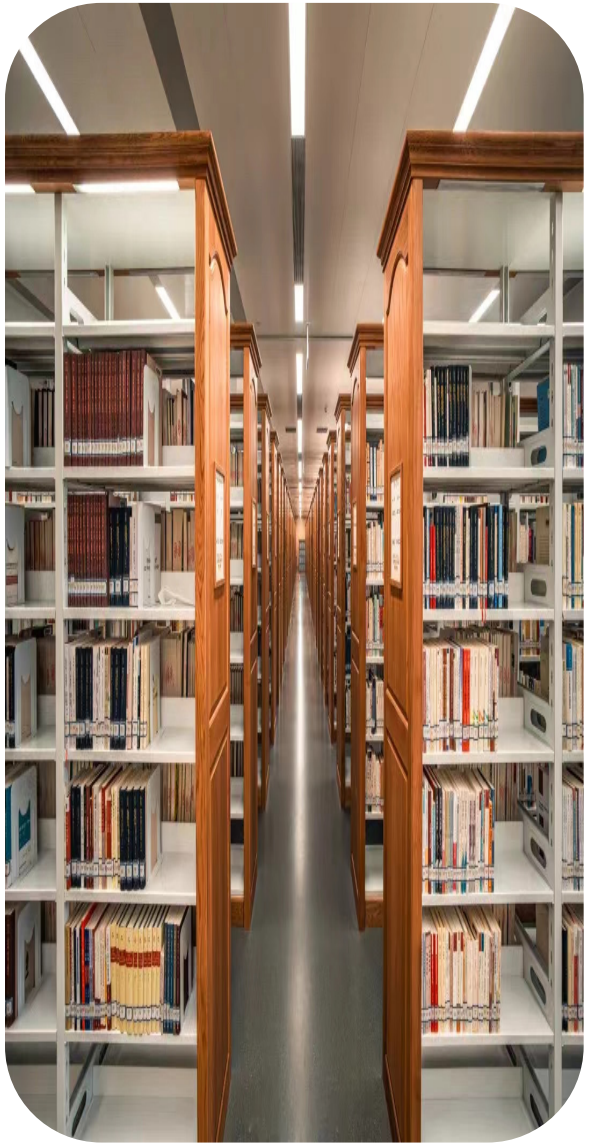
浙江大学
ZHEJIANG UNIVERSITY

智能图书馆派送路径优化

周意尧 戴承江 杨逍宇 樊铧纬

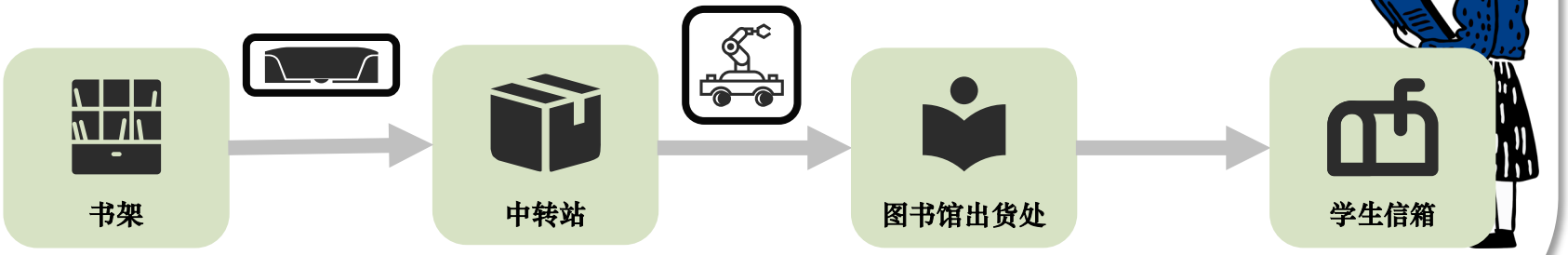
2024 年 6 月 11 日



背景导入

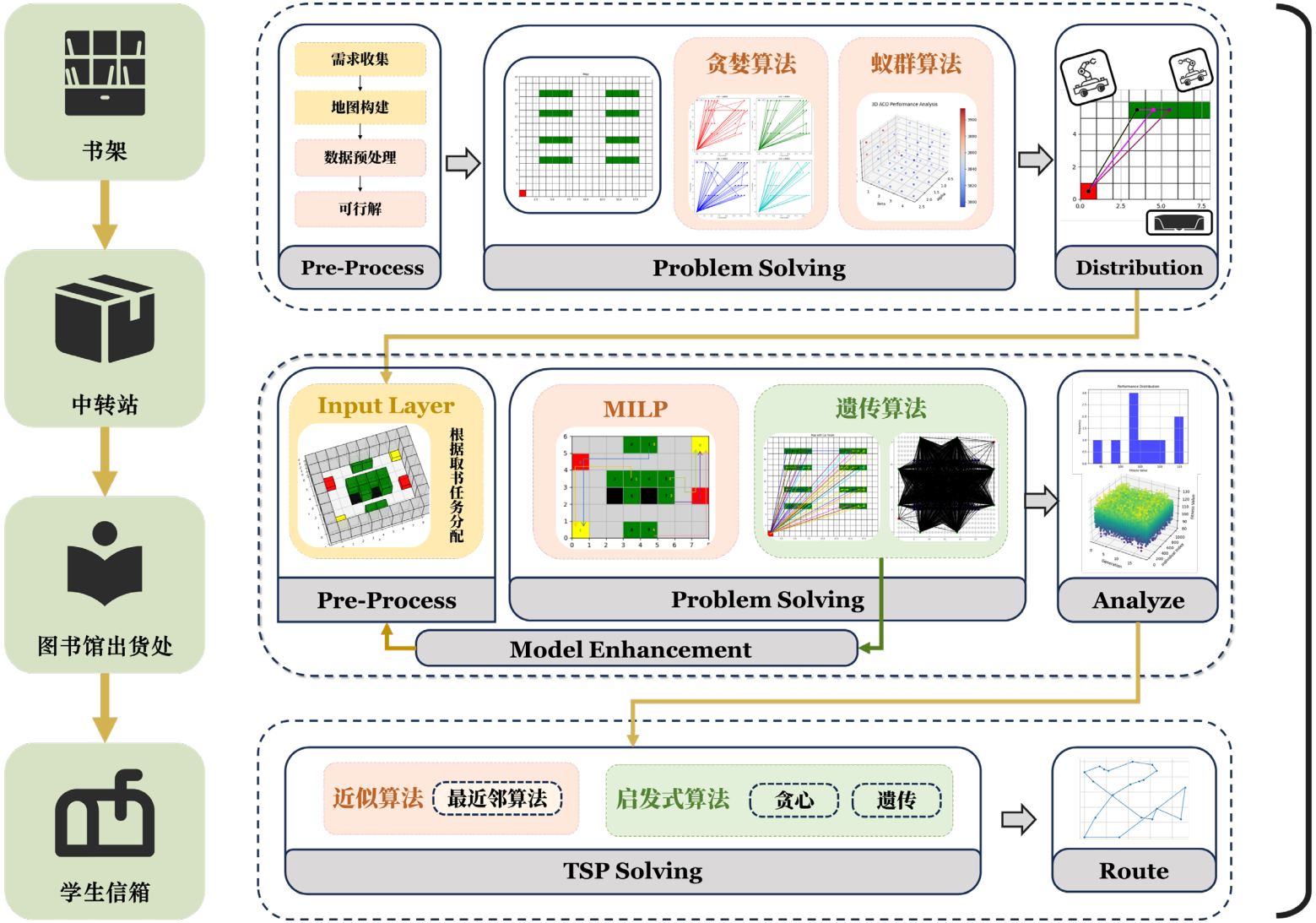
国民人均阅读量 ↑　　找不到书、忘记还书 ☹

便捷、高效的借阅服务需求 ↑



网上预定 — 自动分拣 — 送书上门

工作框架

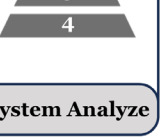



求解


代码同步


报告


展示



问题一定义

小车将已经打包好的书籍

送往每一位同学的信箱中

TSP 旅行商问题

模型构建

目标函数

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

每个城市必须仅被访问一次

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

从起点城市出发，回到起点城市

$$\sum_{i=1}^{n} x_{i1} = 1, \sum_{j=1}^{n} x_{1j} = 1$$

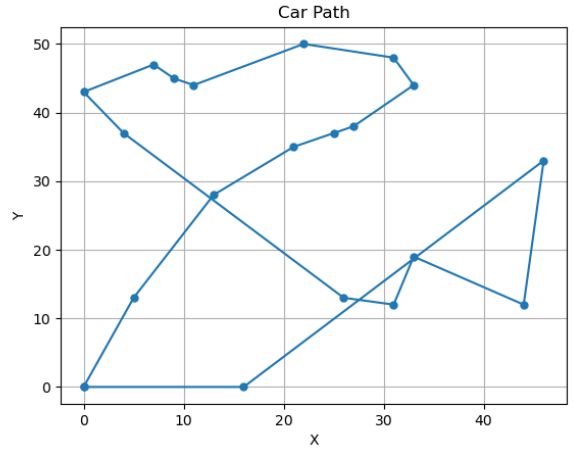
避免子回路

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

算法实现

```
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
```

可视化结果



书架

中转站

图书馆出货处

学生信箱

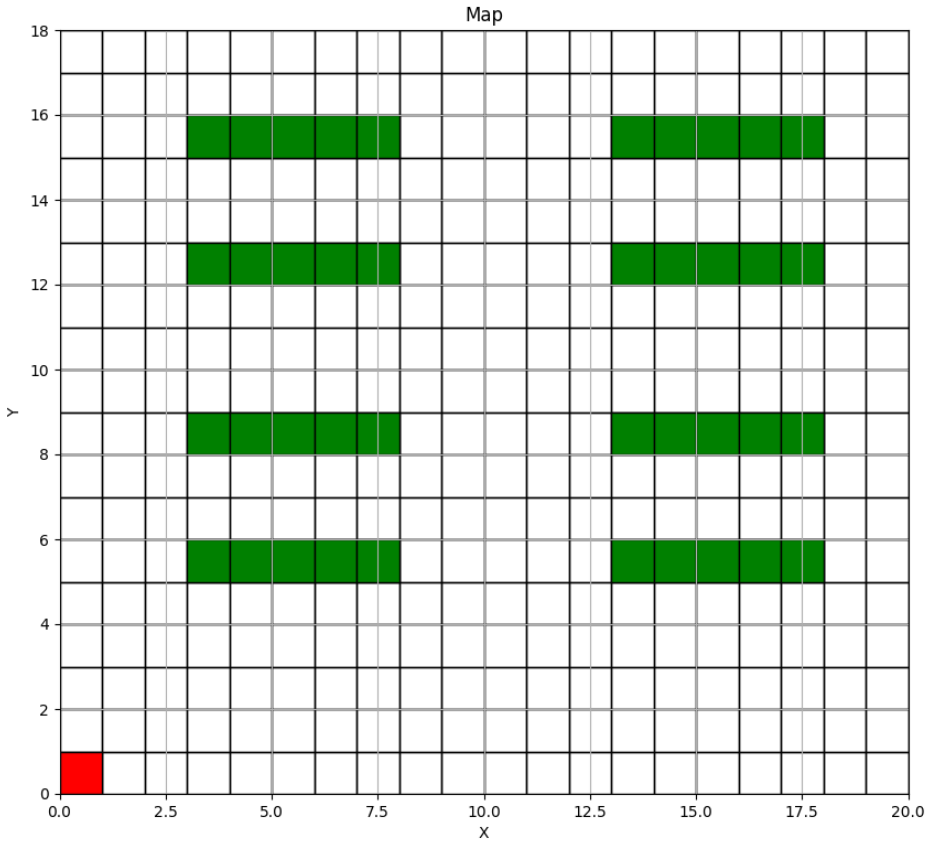
问题二定义

N 辆机器人同时取书

从起始位置出发往返书架取书

每个机器人一次最多M本书

规划机器人路径使总路径最短



模型构建

目标函数

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

配送量不能超过书架的需求量

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

至少有一辆运输小车为书架进行配送

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

到达某点的车辆数应该等于离开该点的车辆数

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

所有书架的取书任务必须全部被完成

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

运输小车不超载

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

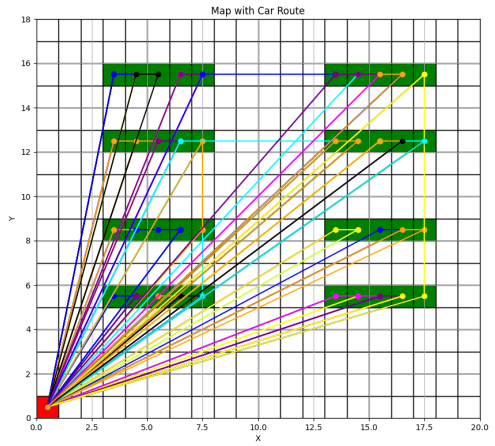
限制没有回路

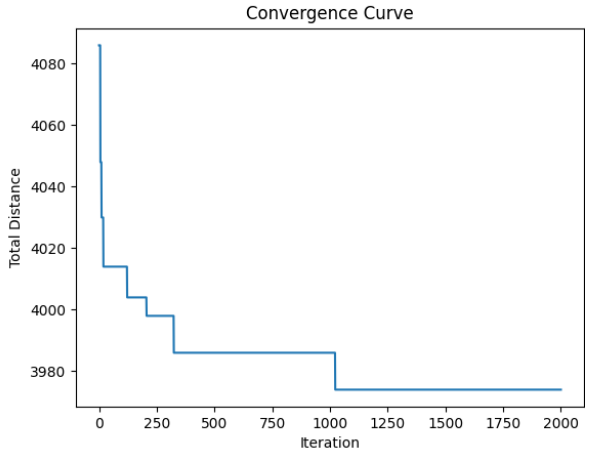
$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} x_{ijk} \leq |s| - 1, 2 \leq |s| \leq n - 1$$

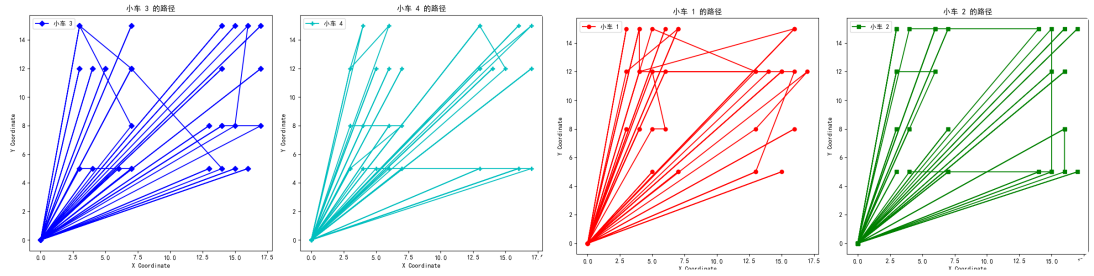
算法实现 —— 贪婪算法

```
1: function generate_initial_solution
2:     初始化解决方案列表和剩余需求
3:     while 剩余需求大于 0 do
4:         初始化路线，从原点开始
5:         初始化剩余容量为车辆容量
6:         while 剩余容量大于 0 且剩余需求大于 0 do
7:             找到可访问的书架并随机访问一个书架
8:             if 剩余容量大于等于书架需求 then
9:                 将书架添加到路线中
10:                更新剩余容量和需求
11:            else
12:                将书架添加到路线中
13:                更新书架需求和剩余容量为 0
14:            end if
15:        end while
16:        将路线添加到解决方案中
17:    end while
```

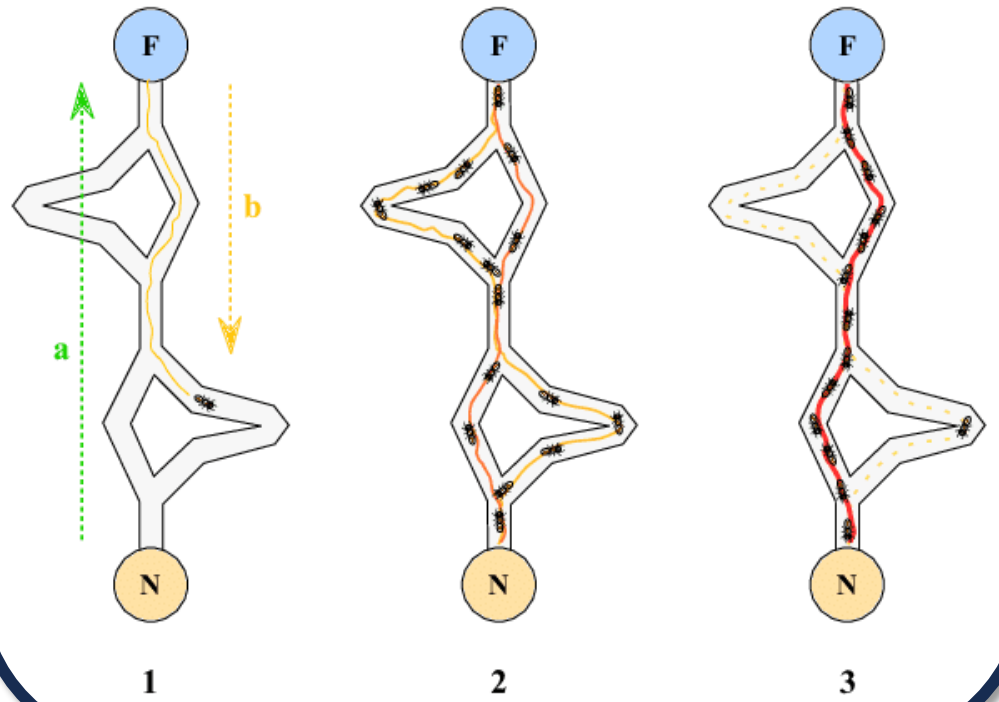
结果







蚁群算法



科学家发现，蚁群在可以在有障碍物的环境下，找到一条最短到达食物的路径。

蚂蚁在路径上释放**信息素**。
后到的蚂蚁沿**信息素浓度高**的路径行走。

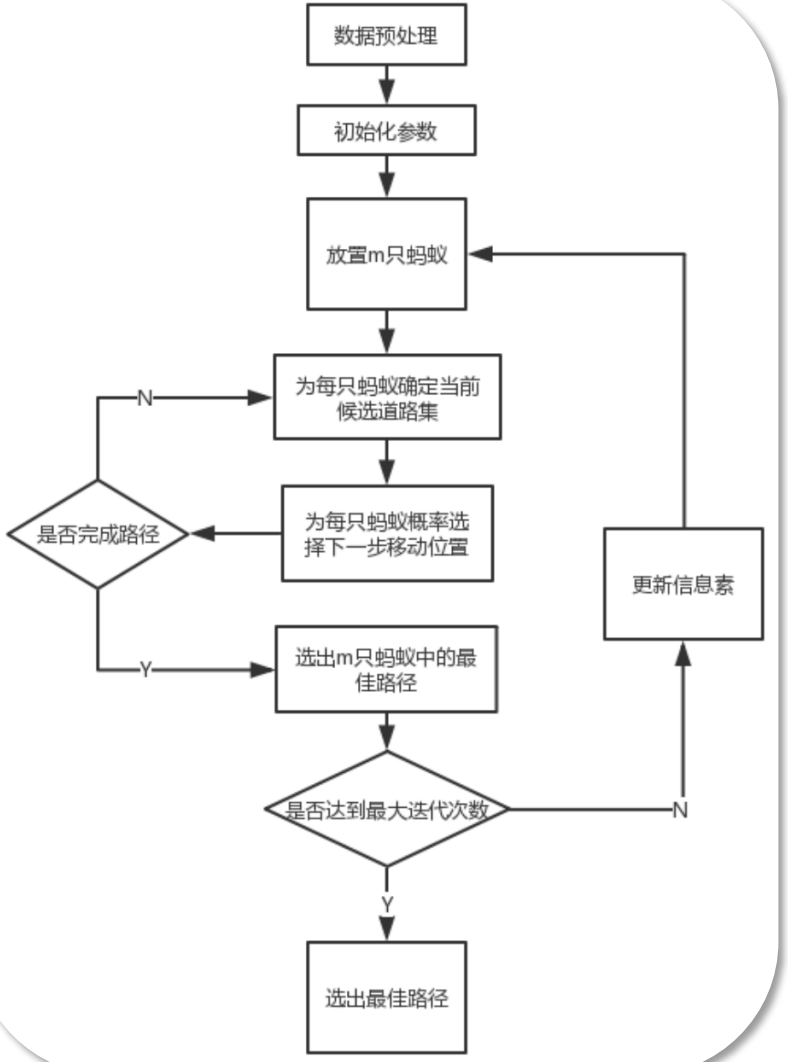
在所有可能的路径中，由于**往返最短路径**花的时间少，通过频率高，所以信息素浓度高。

这又会吸引更多蚂蚁走这条路径，
形成**正反馈**。

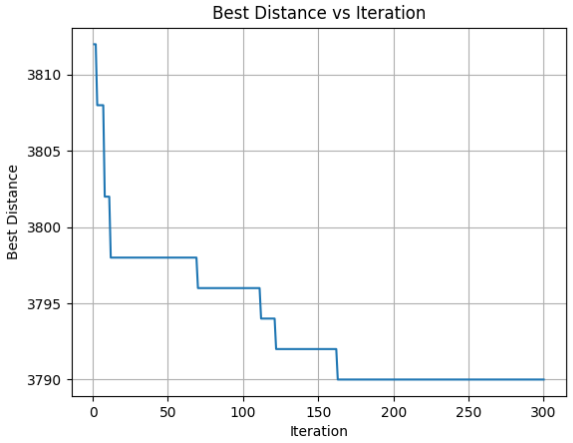
每只蚂蚁只关注局部信息 —— 群体的智能涌现

算法实现

蚁群觅食	蚁群优化算法
蚁群	搜索域内的一组有效解
信息素	信息素浓度变量
蚁巢到食物的一条路径	一个有效解
找到的最短路径	问题最优解



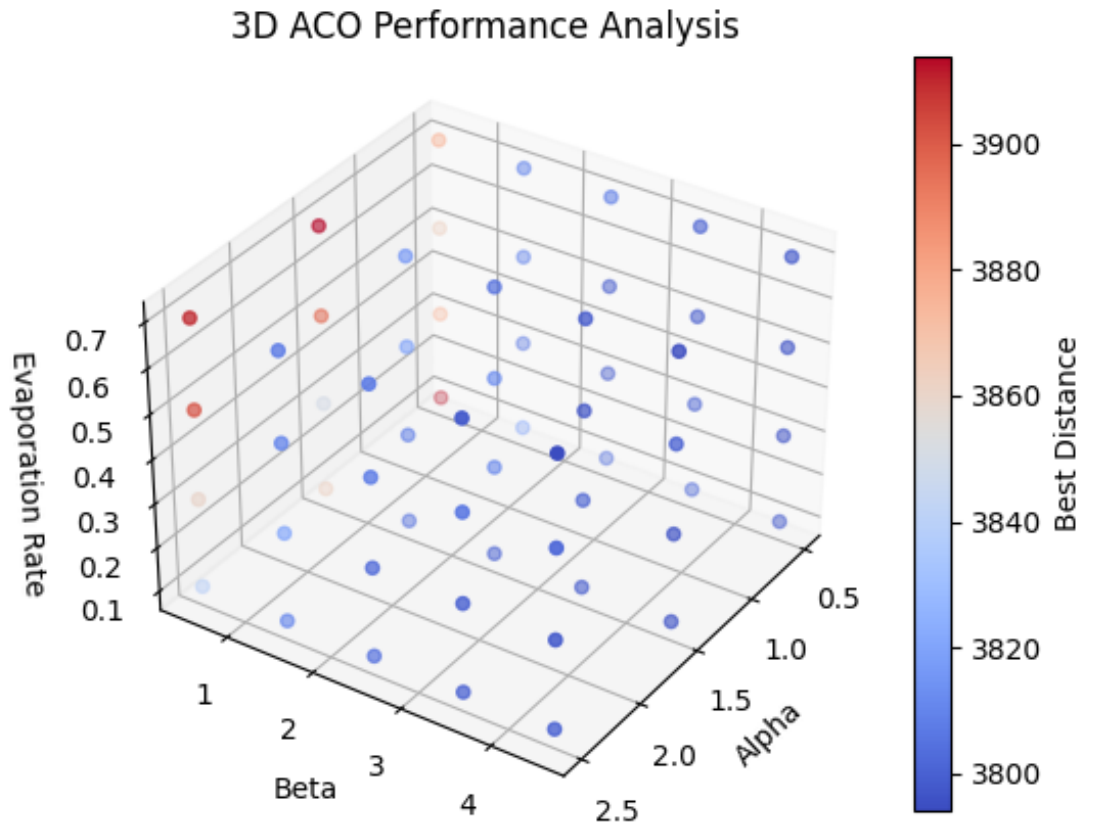
结果



相对于贪婪算法

其收敛速度更快，

需要的迭代次数更少



灵敏度分析

Alpha、Beta、信息素浓度

- Alpha 参数控制信息素的重要性
- Beta 参数控制启发式信息的影响力
- 较低的蒸发率有助于保留有价值的路径信

书架

中转站

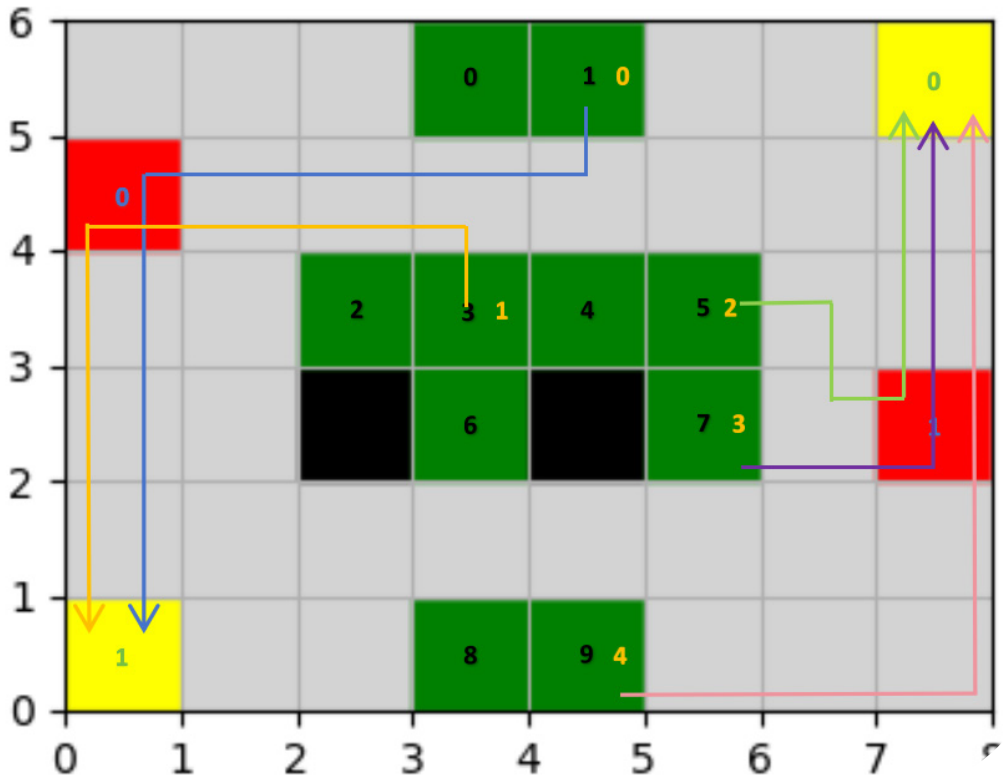
图书馆出货处

学生信箱

问题三定义

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

托盘运行总路径长度最短



模型构建——4类约束

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
得数量

考虑若托盘 i 被挑
应大于 0

托盘内书本数量与预约订单匹配约束

2. 托盘与书籍挑选节点

考虑托盘 i 至多前往一

虑若托盘 i 没有
等于 0

托盘 i 前往拣选

托盘与书籍挑选节点的匹配

从所有托盘中

托盘 i 从储位 q

路线长度 d_{iqm} 应

$d_{iqm} + M(3 -$

盘中的书籍全部

托盘 i 从储位 q

工作台 m 的行走路线长度 d

果没有被挑选完

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

如果托盘还有书籍，则需要放到空储位节点

盘 i 在被拣选

果托盘 i 中的
一个储位

托盘 i 没有被

意一个托盘被

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
托盘 i 中拥有的书籍数量总和

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

托盘 i 在被拣选完商品后，至多前往一个空盘回收位

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

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
不分配回收处

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

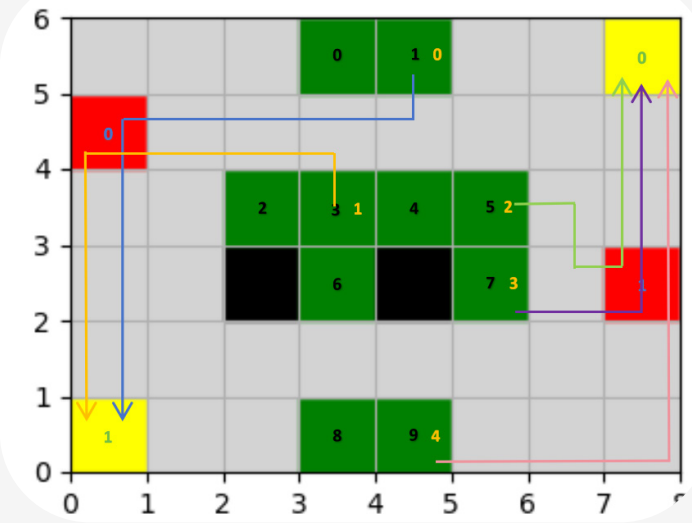
托盘 i 没有被挑选出前往拣选工作台，则不需要为其分配空盘回收位

$$x_i - \sum_{r\in R} f_{ir} \geqslant 0, \forall i \in I \quad (58)$$

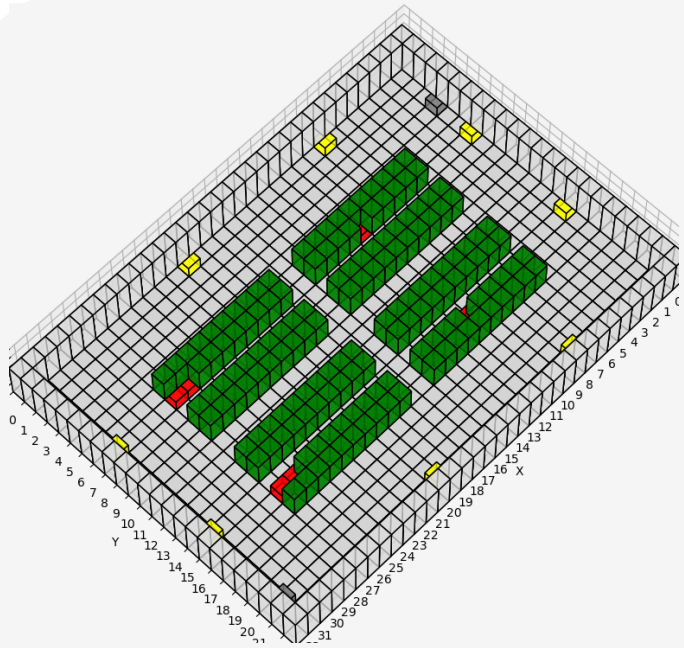
若托盘无书籍，则考虑托盘与回收节点的匹配

分类处理

预订书籍 ＝当前处理书籍
预订书籍 ＞当前处理书籍



小地图验证



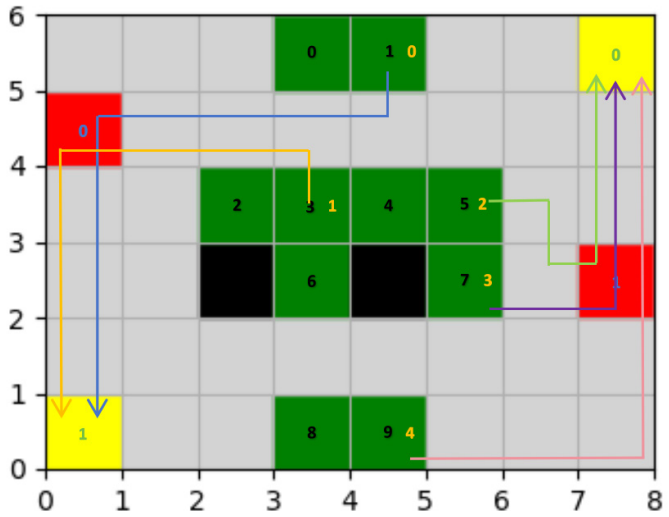
大地图测试

PULP 库结果

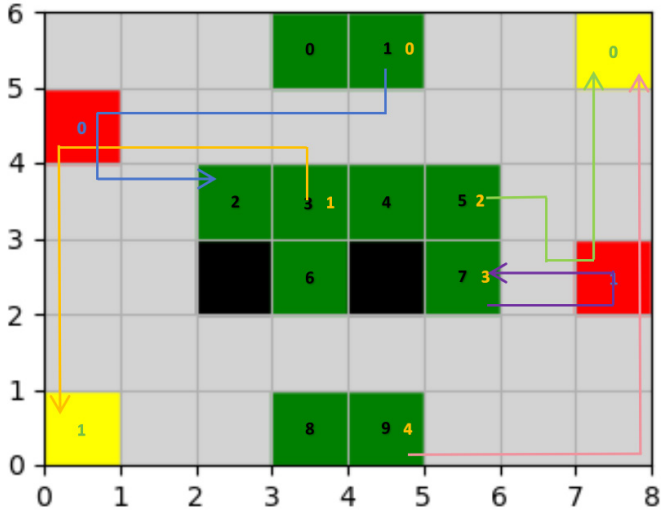
```
d_iqm solution:
d_iqm[0][1][0] = 5.0
d_iqm[1][3][0] = 4.0
d_iqm[2][5][1] = 3.0
d_iqm[3][7][1] = 2.0
d_iqm[4][9][1] = 5.0

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
```



Example 1



```
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
```

Example 2

PULP 库结果

```
Objective value:                370.00000
Enumerated nodes:               15041
Total iterations:               196911
Time (CPU seconds):             21.52
Time (Wallclock seconds):       21.52

Option for printingOptions changed from n
Total time (CPU seconds):       22.07   (
```

```
x solution:
x[0] = 1.0
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
```

```
y solution:
y[0] = 1.0
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
```

```
d_iqm solution:
d_iqm[0][37][6] = 7.0
d_iqm[1][100][6] = 17.0
d_iqm[2][131][3] = 10.0
d_iqm[3][97][6] = 12.0
d_iqm[4][58][6] = 11.0
d_iqm[5][101][6] = 18.0
d_iqm[6][63][6] = 16.0
d_iqm[7][141][5] = 12.0
d_iqm[8][4][6] = 12.0
d_iqm[9][8][6] = 16.0
```

```
d_imq solution:

d_imr solution:
d_imr[0][6][1] = 3.0
d_imr[1][6][1] = 3.0
d_imr[2][3][0] = 6.0
d_imr[3][6][1] = 3.0
d_imr[4][6][1] = 3.0
d_imr[5][6][1] = 3.0
d_imr[6][6][1] = 3.0
d_imr[7][5][0] = 13.0
d_imr[8][6][1] = 3.0
```

Example 3: 所有托盘被运到挑拣处并到达回收处的总路程为 370

```
Objective value:                181.0000
Enumerated nodes:               214
Total iterations:               2549
Time (CPU seconds):             1.41
Time (Wallclock seconds):       1.41

Option for printingOptions changed from
Total time (CPU seconds):       1.72    (
```

```
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
```

```
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
```

```
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
```

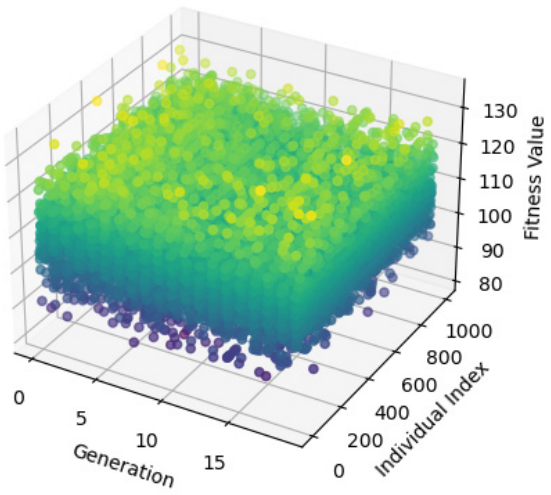
```
d_imq solution:

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
```

Example 4: 三步的距离如上图所示

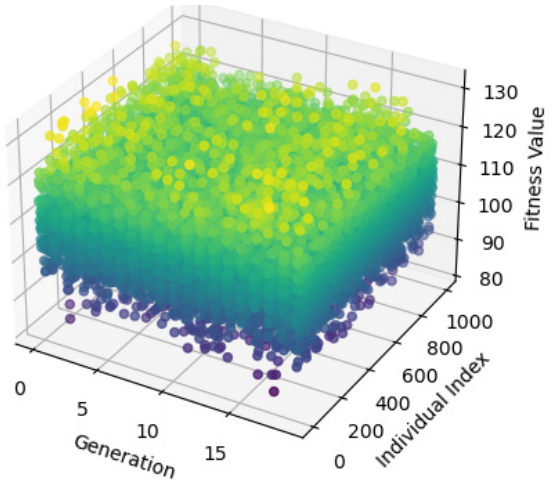
遗传算法 + Deap库

Example 1



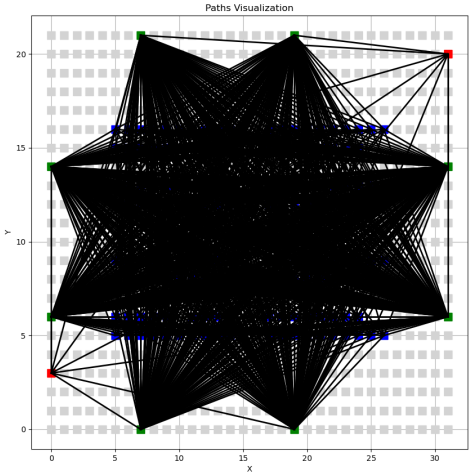
小地图
预订 = 处理

Example 2



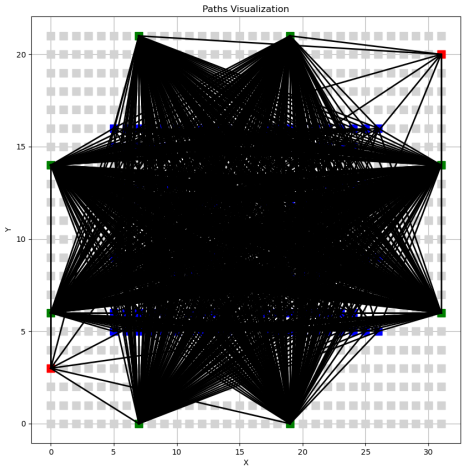
小地图
预订 > 处理

Example 3



大地图
预订 = 处理

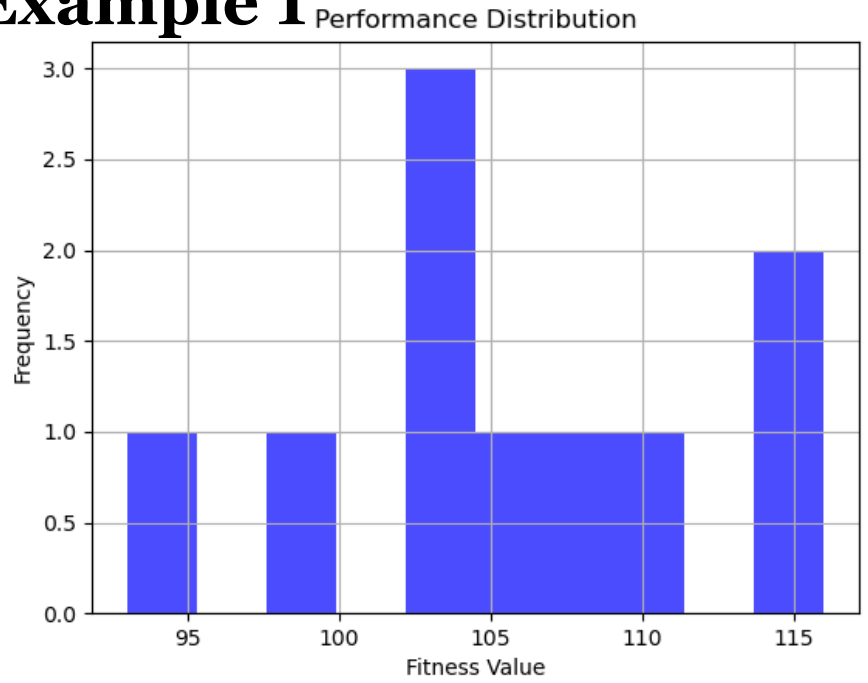
Example 4



大地图
预订> 处理

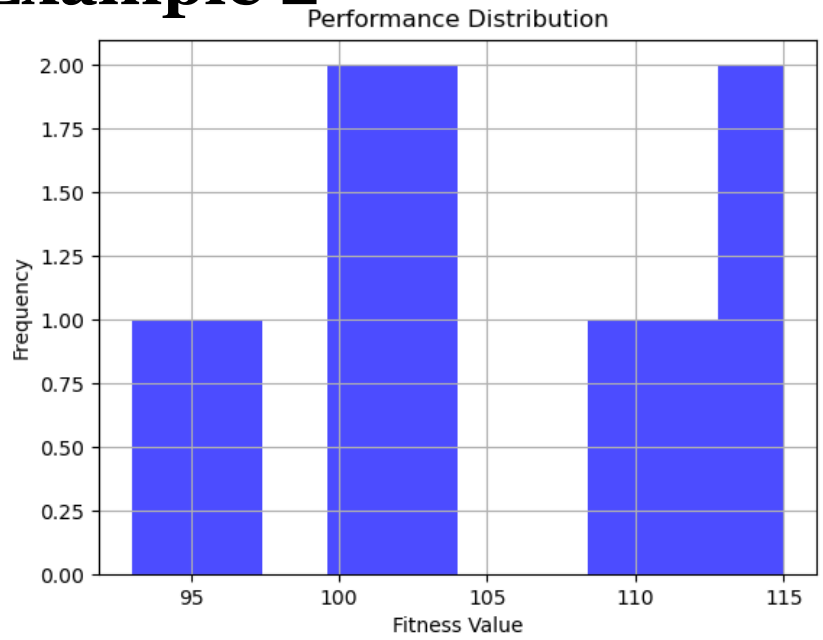
误差分析

Example 1




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

Example 2




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

团队与分工

周意尧

- 问题三
- 报告撰写

戴承江

- 问题三
- 报告撰写

樊铧纬

- 问题二
- PPT制作

杨逍宇

- 问题二
- 报告撰写