基于粒子群优化算法的华北农作物种植策略模型

摘　要

华北农村土地贫瘠，缺乏水分，不时还有气候灾害发生。华北民众凭着“地种百样不靠天，田种百处不靠地”的精神改造土地[1]，过上了富足的生活。而今新时代，随着乡村振兴政策的进一步落实与人民对于美好生活的向往日益提高，除了劳动人民的努力与奋斗之外，通过优化种植技术策略等使农作物产量的提高的手段成为了关键的问题。本文采用了**粒子群优化**[3]的数学模型来对种植策略进行建模，能帮助华北地区农村的优化种植策略，进一步的提高产量，增加产收。

针对问题 1，我们首先确定需要解决的两个问题类似，都需要在超过部分情况下求最佳的种植方案，属于优化类问题。先找优化问题的关键，我们设置了种植面积x_{ijk}作为决策变量。接着，在 7 年之内总产量等关键变量都不发生变化的假设下，我们按照题目中给的约束条件整理，结合附件中的表格，并且整理出来了约束条件，并且依照问题一的两个小问超出部分的利润计算差别列出目标函数。这之后，我们运用了 PSO **粒子群算法**迭代出来了最优耕种计划，以及最大的利润分别是 21894065.869285043 元与 28309501.921012778 元。

针对问题 2，我们沿用了第一题的思路，也用的是粒子群优化算法。问题二中的约束条件在一个区间变化，我们使用了**鲁棒优化**，保证了种植策略相对于每年不同行情都能表现良好。我们一共为鲁棒优化设置了 40 个场景，然后在这些场景中我们选出最坏的情况，并且将最坏的情况用 PSO 迭代处理，求出最坏场景的最大累计利润为 22093492.19518669 元。

针对问题 3，在处理数据的时候我们发现作物的可替代性和互补性与**聚类**的特征有相似之处，为了验证这一点，我们采用了两种聚类方式分别求解并进行比较，这两种聚类方式分别是通过区分地块和大棚手动划分，和用 K-means 方法为作物信息分出三类。然后，为了分析预期销售量和种植成本，销售价格之间的相关性，我们首先判断出不能用皮尔森相关系数，接着用**斯皮尔曼相关系数**进行了分析。最后我们比较了线性和非线性的回归，发现非线性的R^2值要优于线性的R^2值，由此我们认为非线性得到的两个非线性函数的拟合程度更高，于是采用这个**非线性回归**的模型预测预期销售量，并且运用 PSO 迭代，得到聚类的最大收益 37366915.74324817 元，大于手动划分的 24456926.29424125 元

关键词：种植策略　粒子群优化　鲁棒优化 相关性分析 非线性回归

一、 问题重述

1.1 问题背景

针对不同的乡村地域情况，合理的在耕种区域内发展种植产业，是乡村振兴和可持续发展中的重要一环。选择合适的农作物，制定合理的种植方法，能够提高农作物的产量。

1.2 问题提出

在华北山区中，一处常年偏低温的乡村种植的农产品是一年一季熟，现在有 34 个大小各异的露天耕地地块，拢共 1201 亩。有四类的耕地，其中有平旱地和梯田以及山坡地，水浇地。前三者每年能种植一季粮食类作物，而水浇地每年能种植一年熟的水稻或者一年收获两次的蔬菜。除了耕地之外，乡村还拥有 16 个普通大棚和 4 个智慧大棚。每个大棚拥有 0.6 亩的种植面积，普通大棚适合种植一年熟的蔬菜以及食用菌，而智慧大棚比较适合每年种植一年收获两次的蔬菜。一个地块在每一季都能种植一种或以上的作物。

在每一地块上同一作物不能够重茬种植，以避免减产。同时，豆类种植过的土壤有利于其他作物成长，从 2023 年开始，要求包括含大棚在内的地块在三年内至少种植一次豆类作物。同时，在一季内同一种作物的种植区域不能太分散，在单种地块同一种作物种植面积不能太小。现在建立数学模型，来帮助规划种植策略，解决下列问题：

（1）某种作物每季的总产量超过相应的预期销售量，超过部分不能正常销售。分别就超过部分滞销造成浪费，以及超过部分按 2023 年销售价格 50%降价销售两种情况给出种植方案。

（2）小麦和玉米的年产量会逐年拔高 5%至 10%，而其他的农产品则是正负 5%的变化区间。亩产量受天气的影响，粮食产量会有正负 10%的变化区间。因受市场条件影响，农作物的种植成本平均每年增长 5%左右。蔬菜类作物的销售价格平均每年增长 5% 左右。食用菌的销售价格大约每年可下降 1%~5%，其中羊肚菌的销售价格每年下降幅度为 5%。现综合各因素给出 2024~2030 年的最佳种植方案。

（3）各种农作物之间可能存在可替代性和互补性，预期销售量与销售价格、种植成本之间也存在一定的相关性。请综合考虑相关因素，给出该乡村 2024~2030 年农作物的最优种植策略。

二、 问题分析

2.1 问题一的分析

对于问题一，题目中给定了两种情况，我们将分别分析：

- 第一种情况：当超过部分不能正常销售时，我们需要将销售额减去成本价得到的函数，再结合表格做约束。找出决策变量、约束变量和约束条件，以及目标函数。接着，通过粒子群算法或遗传算法去更新迭代，以寻找种植策略的最优解。

- 第二种情况：类似于“分段收费”，在可能有超出剩余的情况下，乘以某个系数得到第二部分收益。将未超出的第一部分收益与第二部分收益相加，并进行约束。为分析供需关系，我们总结了表格，并绘制了如下图表：图表 1、图表 2 和图表 3。

2.2 问题二的分析

问题二与问题一有类似之处，本质上都是使用有约束的粒子群算法去寻找最优解。问题一中假定每年的售价、预期销售量、种植成本等与 2023 年相同，而问题二中增加了亩产量，销售量，销售价格等的限制条件。因为这些约束条件在一个区间内变化，我们想到了两种方法，第一种是考虑使用蒙特卡洛模拟，即在区间内随机多次采样亩产量，然后取平均值。另一种方法是鲁棒优化，即通过约束优化解，使得即使在这些不确定参数的最坏情况下，模型的性能仍然是可接受的。

2.3 问题三的分析

问题三考虑了作物之间的可替代性和互补性，以及销售量、价格与成本的相关性。优化种植方案以最大化收益，基于作物替代性、互补性及相关性。我们使用相关性分析来描述变量之间的相关性，其中我们可能会用到皮尔森相关或者斯米尔满相关进行分析，并进一步完善问题 2 中的模型。然后进行多元线性回归监测模型。问题三需要进行更多的模拟实验，以验证和比较不同情况下的最优方案。

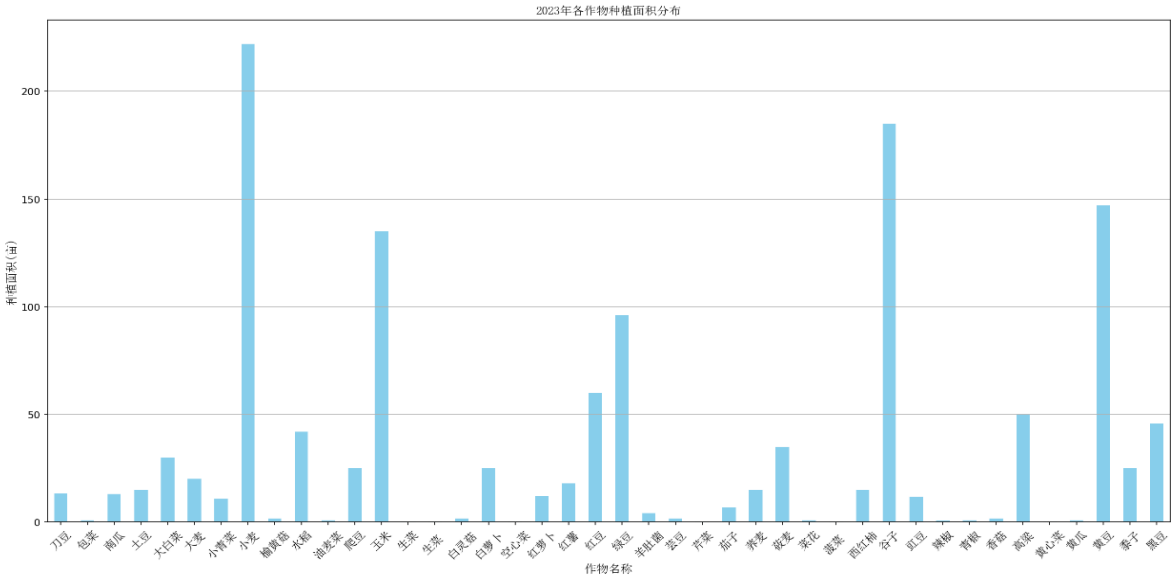


图 1 作物名称与种植面积

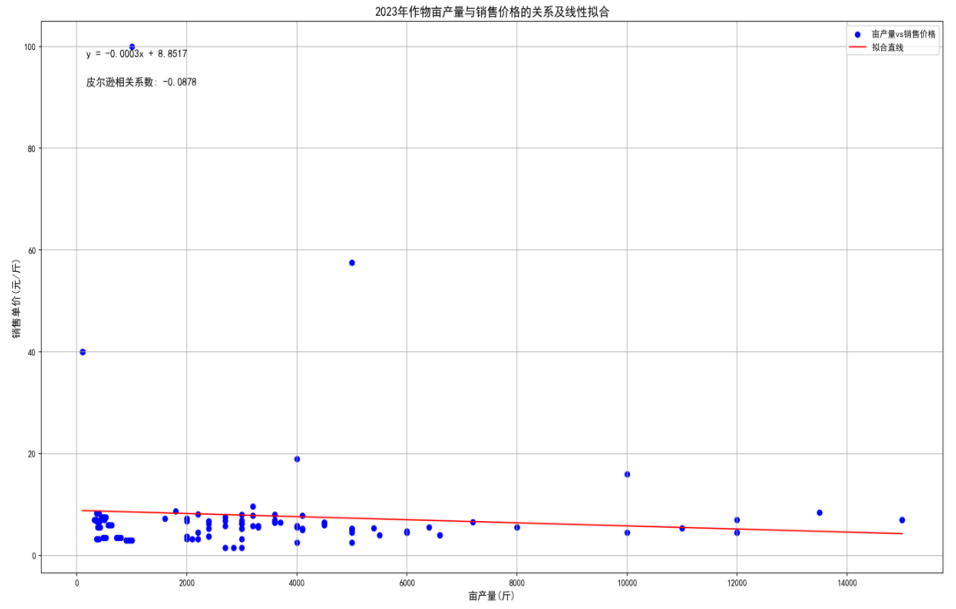


图 2 2023 年亩产量与销售价格的线性拟合

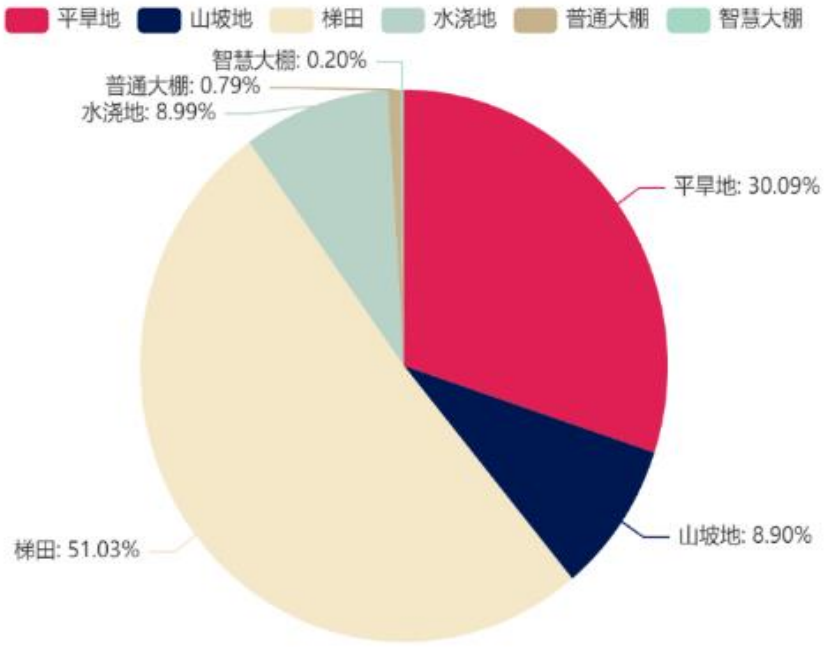


图 3 华北农村耕种地分布图

三、 模型假设

1. 各种地块和大棚的种植条件保持不变，适合种植的作物种类不变。
2. 每种作物不能连续重茬种植，三年内至少种植一次的豆类植物
3. 假设每一块耕地划分 i 是相等的
4. 假设 2023 年农产品的销售量等于 2023 年农产品的总产量

四、 符号说明

符号	说明	单位
i	地块	亩
j	作物	无
k	年份	无
C_{ijk}	单位面积种植成本	元/亩
x_{ijk}	种植面积	亩
P_{ijk}	单位面积收益（亩产量）	斤
S_{ijk}	作物销售量	元/亩

$\boldsymbol{B_{ijk}}$	作物销售单价	元/亩
$\boldsymbol{d_{ik}}$	种豆决定量	无

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的建立

I. 决策变量：

我们将这一题的决策变量设置为x_{ijk}，其中 i 表示种植的地块，j 表示种植植物的种类，k 表示种植植物年份。而x_{ijk}则表示了某一植物种植面积，我们以此为决策变量，去寻找最佳的解决方案。

II. 约束变量及约束条件

我们把原问题中在耕种时对于作物耕种，地块，对于年份的约束条件全部化为对于决策变量x_{ijk}的约束条件，下表是列出的对于题目条件的整理，含’的是考虑了两季的情况：

表 1　题目条件变量转化及其对应

地块 i	**作物 j**
$A_1 \sim A_6$ 平旱	1~15 粮食
$B_1 \sim B_{14}$梯田	16 水稻
$C_1 \sim C_6$山坡	17~34 蔬菜
$D_1 \sim D_8$水浇	35~37 特殊蔬菜
$E_1 \sim E_{16}$普通大棚	38~41 菌类
$F_1 \sim F_4$智慧大棚	17~19∪1~5 豆类
$D_1^{'} \sim D_8^{'}$ 水浇第二季	
$E_1^{'} \sim E_{16}^{'}$普通大棚第二季	
$F_1^{'} \sim F_4^{'}$ 智慧大棚第二季	

由此给出约束条件：

（1） **平旱地、梯田、山坡地约束：**

当地块属于平旱地，梯田以及山坡的时候，不能种植 7 号到 41 号农作物：

$$x_{ijk} = 0\,(\mathrm{i} \in \{A_1 \cdots A_6, B_1 \cdots B_4, C_1 \cdots C_6\},\quad \mathrm{j} \in (6,41])$$

（2） **水浇地约束：**

如果在水浇地种植水稻的地块大于 0 的话，在水浇地第二季种植地块就等于 0，而如果在水浇地种植植物的地块大于 0 的话，那么在水浇地第二季种植地块不能种植粮食，水稻，蔬菜以及菌类。粮食和菌类不能在水浇地或者水浇地第二季种植。数学形式化如下：

（1） $if \quad x_{ijk} > 0\,(\mathrm{i} \in \{D_1 \cdots D_8\}, \mathrm{j} = 16)$

$$x_{ijk} = 0\,(\mathrm{i} \in \{D_1' \cdots D_8'\})$$

$$else\ if \quad x_{ijk} > 0\,(\mathrm{i} \in \{D_1 \cdots D_8\}, \mathrm{j} \in [17,34])$$

$$x_{ijk} = 0\,(\mathrm{i} \in \{D_1' \cdots D_8'\}, \mathrm{j} \in [1,34] \cup [38,41])$$

（2） $x_{ijk} = 0\,(\mathrm{i} \in \{D_1' \cdots D_8' \cup D_1 \cdots D_8\},\ \mathrm{j} \in [1,15] \cup [38,41])$

（3） **普通大棚约束**：

当地块 i 是普通大棚的时候，不能种植粮食，水稻，特殊蔬菜以及菌类；当地块 i 是普通大棚的第二季种植时，不能种植粮食，水稻，蔬菜，以及特殊蔬菜：

（1） $x_{ijk} = 0\,(\mathrm{i} \in \{\mathrm{E}\},\ \mathrm{j} \in [1,16] \cup [35,41])$

（2） $x_{ijk} = 0\,(\mathrm{i} \in \{E_{\square}'\}, \mathrm{j} \in [1,37])$

（4） **智慧大棚约束**：

当地块 i 为智慧大棚或者是智慧大棚第二季的时候，不能种植粮食，水稻，以及菌类，特殊蔬菜：

$$x_{ijk} = 0\,(\mathrm{i} \in \{F \cup F_{\square}'\}, \mathrm{j} \in [1,16] \cup [35,41])$$

（5） **不重茬条件：**

$$x_{ijk}x_{ijk+1} = 0，(\mathrm{i} \in (\mathrm{A},\mathrm{B},\mathrm{C}))$$

$$x_{ijk}x_{i+28jk} = 0，（\mathrm{i} \in \mathrm{F}）$$

$$x_{ijk+1}x_{i+28jk} = 0，（\mathrm{i} \in \mathrm{F}）$$

（6） **三年至少需要种植一次豆类植物：**

$$\sum_{k=1}^{3} d_{ik} = 1$$

（7） **种植面积不能超过地块的面积：**

$$\sum_{j} x_{ijk} \leq A_i$$

（8） 各种农作物未来的预期销售量、种植成本、亩产量和销售价格相对于2023 年保持稳定，每季种植的农作物在当季销售。

（9） 对于种植来说，不能种的太分散，也不能一个地块同一个作物种的太少。此处的 0.2 考虑到了 2023 年的大棚中最多分种了 3 种作物，所以每一块作物最少也需要 0.2 亩的土地。公式化为：

$$x_{ijk} \geq minA_i \ \ when\ i \in (A,B,C,D)$$

$$x_{ijk} \geq 0.2\ when\ i \in (E,F)$$

III. **目标函数设计**

我们需要求的就是关于总利润的相关表达式，并且用以上条件约束使其在一个区域内达到最大值，结合第一题两种不同的策略，我们有如下两个目标函数：

剩余部分滞销造成浪费，我们就用相应面积的种植成本当作总成本，再定义总产量T_{ijk}为种植面积x_{ijk}与亩产量P_{ijk}相乘，然后与作物销售量比较，得到更小的那个便是能够全部得到收益的，再与作物销售单价相乘得到了总利润，最后再对每一块地，每一种农作物，每一年进行求和统计得出最后的目标函数:

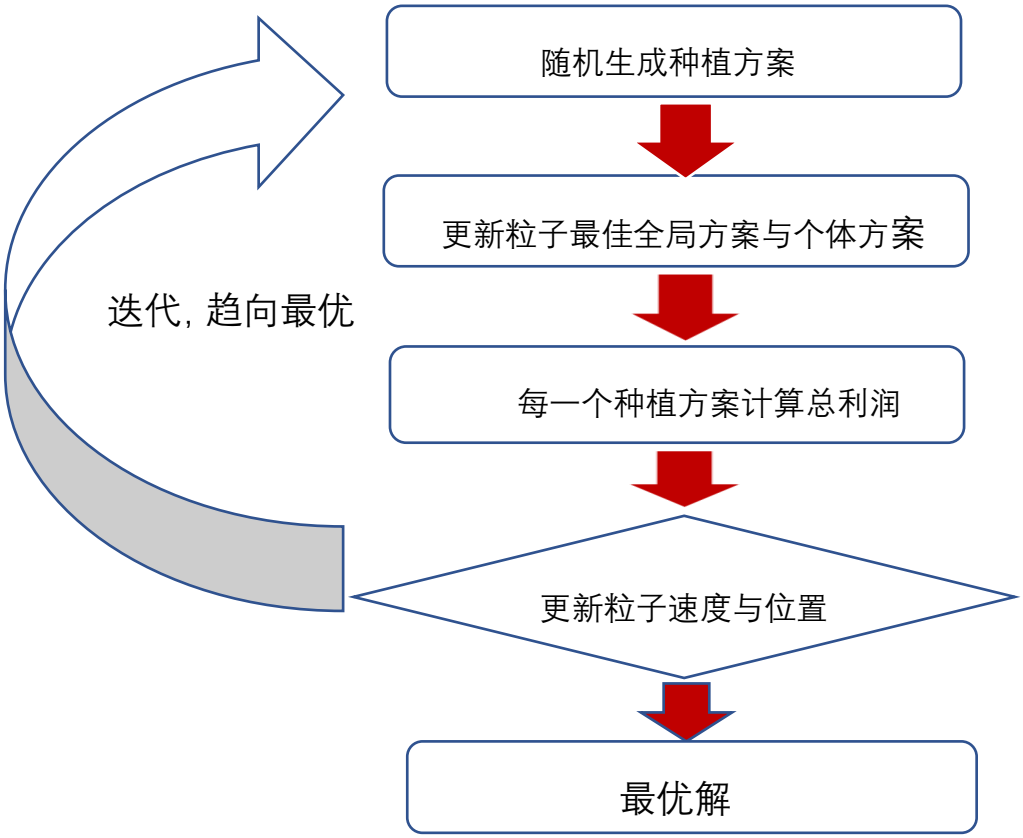
$$\sum_{k=1}^{7}\sum_{i,j} （B_{ijk} \cdot \min(T_{ijk}, S_{ijk}) - C_{ijk}x_{ijk}）$$

超过部分按 2023 年销售价格的 50%降价出售，与上一题的目标函数基本上是一致的，唯一区别就在于超出的部分，超出的部分就是用总产量减去总的销售量，如果得到的值小于 0 不符合现实生活，所以我们把其与 0 取了一个最大值，得到了第二小问的目标函数。

$$\sum_{k=1}^{7}\sum_{i,j}\left(B_{ijk}\cdot\min(T_{ijk},S_{ijk})-C_{ijk}x_{ijk}+\frac{1}{2}B_{ijk}\max\left(T_{ijk}-S_{ijk},0\right)\right)$$

IV. **算法设计**

我们在 PSO 和 GA 中选择了 PSO，相对于 PSO，GA 虽然具有多样性和灵活性，但是粒子群优化的更新规则简单[5][6]，能更快达到收敛，所以我们最终选择了 PSO。我们利用 PSO 优化算法去进行优化迭代，流程如下，迭代之后就能找到最优的种植方案，最大的种植利润。



5.1.2 模型的求解

将表格的数据整理出来并且提取完毕，通过粒子群优化（PSO）算法优化地块作物种植方案，以最大化总利润。接着进行约束函数的步骤，以及首先，定义多个约束函数以确保种植方案符合实际条件，如区域限制、作物种类、相邻地块冲突等；其次，目标函数计算总利润，包括产量、种植成本和销售收入，同时检查是否满足所有约束；最后，PSO 算法用于在定义的种植面积边界内寻找最佳方案。优化结果经过处理后保存到 Excel 文件中，供进一步分析和决策使用。值得注意的是，我们注意到表格的数据没给全，我们补充了智慧大棚第一季的数据。关键步骤如下：

- 定义约束函数
- water_crop_vege(x)`:确保水浇地作物种植合法
- ABC(x):确保平旱地，梯田，山坡地作物种植合法
- ordipeng(x)`:限制普通大棚的作物种植
- smartpeng(x)`:确保智慧大棚的种植要求
- is_bean(x)`:确保豆类作物的种植条件
- again(x)`:处理重茬的种植约束
- arealimit(x):处理种地面积的约束条件，同时保证种植不会过于分散
- profit_function(x)`:定义目标函数
- 计算每块地的总面积并调整种植面积，计算产量、种植成本和销售收入，确保满足所有约束条件，返回负的总净利润
- 定义优化边界 - 设置种植面积的上下界
- 执行粒子群优化(PSO)，使用 PSO 算法进行优化，输入目标函数、上下界、PSO 参数（如种群规模、最大迭代次数等）
- 获取最佳种植方案和最佳利润

完整代码已添加在附录。

整理完约束条件和代码之后，运行完毕，得到最大利润，与最佳的耕地分配方案，填入 result。需要注意的是，因为 PSO 算法最终迭代完毕得到的是最小值，所以我们将最大利润设置为负数。第一小问和第二小问运行结束之后用 python 绘制收敛曲线，得到结果如下图四，图五所示。

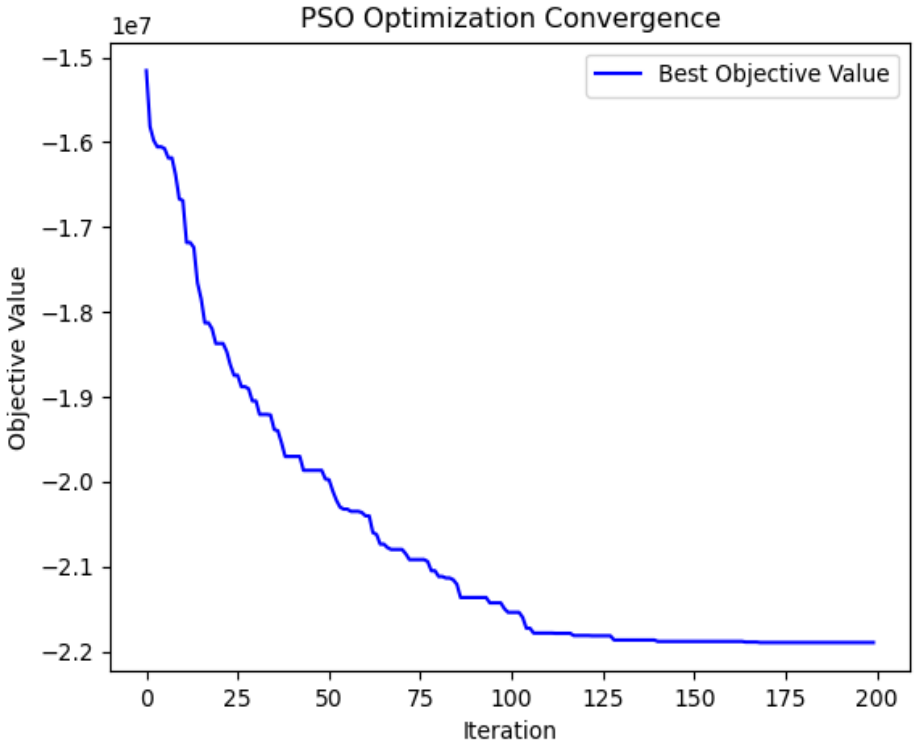


图 4 超过部分全部浪费策略 PSO 收敛图像

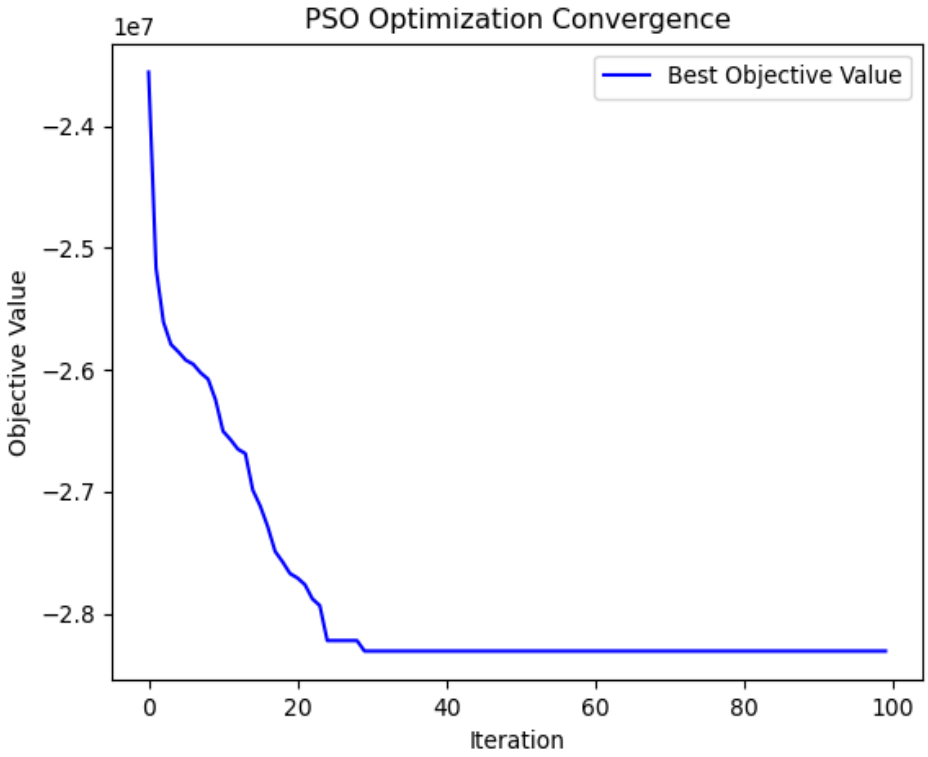


图 5 超过部分半价出售策略 PSO 收敛图像

最终，超过部分全部浪费的策略 PSO 在 200 次迭代时收敛，收敛成功，2024 年到 2030 年累计的最终最大利润:为 21894065.869285043 元。而超过部分半价出售策略 PSO 在 100 次左右迭代收敛，收敛成功，2024 年到 2030 年累计的最终最大利润:28309501.921012778 元。可以看到如果没有全部浪费超过预期部分的情况，最终最大利润在迭代次数更少的情况下仍然更大，这是符合预期与直觉的。

另外，观察种植方案可以发现，虽然确实满足了约束，但是很多地块和大棚仍有空置区域，我们认为这是因为目标函数仍未收敛到最小值，如果增加迭代次数 PSO 可能得到更优的解，但在算力有限的情况下，这是可以接受的。

5.2 问题二模型的建立与求解

5.2.1 模型的建立

对于问题二，我们选择的是鲁棒优化。我们认为鲁棒优化相较于蒙特卡罗方法，第一点优势是鲁棒优化的约束条件是严格成立的[2]。在问题一的基础上，问题二需要优化求解的依然是一个多约束目标，我们认为鲁棒优化会更符合实际效果。另外一点很重要的是，我们认为蒙特卡洛模拟虽然能够提供对于不确定性的较为全面的视角，但是鲁棒优化更具稳定性，考虑到华北农村地区未来无法预知的情况，可能天气恶劣，产量与销量不稳定，我们需要一种更能保证适用于多个年份，能够稳定产收的方案。

I. 决策变量

跟问题一一样，我们先确认决策变量仍然是种植面积x_{ijk}，同时每片土地的面积仍然是A_i。

II. 约束变量与约束条件

对于约束变量，除了问题一中的约束条件，问题二额外多出了几条，其中 r 为增长率：

（1）小麦玉米预期销售量 S 年均增长 5%~10%

$$S_j^k = S_j^{2023} * (1 + r_1)^{k-2023} \quad \text{其中 } r_1 \in [0.05,1]$$

（2）其他农作物预期销售量每年有 5%到 10%的变化，这个变化和年份 k 无关

$$S_j = S_j^{2023} * (1 + \varepsilon) \quad \text{其中 } \varepsilon \in [-0.05,0.05]$$

（3）亩产量 P 受天气影响，每年有$\pm 10\%$的变化

$$P_{ij}^{k} = P_{ij}^{2023} * (1 + r_2) \quad \text{其中 } r_2 \in [-0.1, 0.1]$$

（4）农作物的种植成本 C 平均每年增长 5%

$$C_{ij}^{k} = C_{ij}^{2023} * (1 + 0.05)^{k-2023}$$

（5）粮食价格 P 基本稳定

$$P_{j}^{k} = P_{j}^{2023}$$

（6）蔬菜价格每年增长 5%

$$P_{j}^{k} = P_{j}^{2023} * (1 + 0.05)^{k-2023}$$

（7）食用菌类价格每年下降 1%~5%，其中羊肚菌固定每年 5%

$$P_{j}^{k} = P_{j}^{2023} * (1 - r_3)^{k-2023} \quad \text{其中 } r_3 \in [0.01, 0.05]$$

III. **目标函数**

本题并未给出目标函数的约束，但为了保证公平性，我们将问题一的两种方案的目标函数取了一个平均值，得出来的就是本题的目标函数：

$$\sum_{k=1}^{7} \sum_{i,j} （B_{ijk} \cdot \min(T_{ijk}, S_{ijk}) - C_{ijk} x_{ijk} + \frac{1}{4} B_{ijk} \max(T_{ijk} - S_{ijk}, 0)）$$

5.2.2 模型的求解

我们将新的目标函数以及新的约束条件，加上旧的约束条件作为题目二的前提。接着我们编写鲁棒优化程序，程序随机产生场景，也就是在新约束条件的区间内随机产生场景，然后判断最坏的情景并记录，在设定的循环次数内找到一个最小值，并且将这个最小值作为最坏的情况来进行对目标函数求最优的 PSO 迭代。我们设定了 40 个循环，也就是 40 个场景，然后再从每一个场景去求最优的 PSO 迭代

鲁棒优化后，我们得到了最优结果，2024 年到 2030 年累计的最终最大利润:[22093492.19518669]元

100 次迭代后，可以看到还没完全收敛

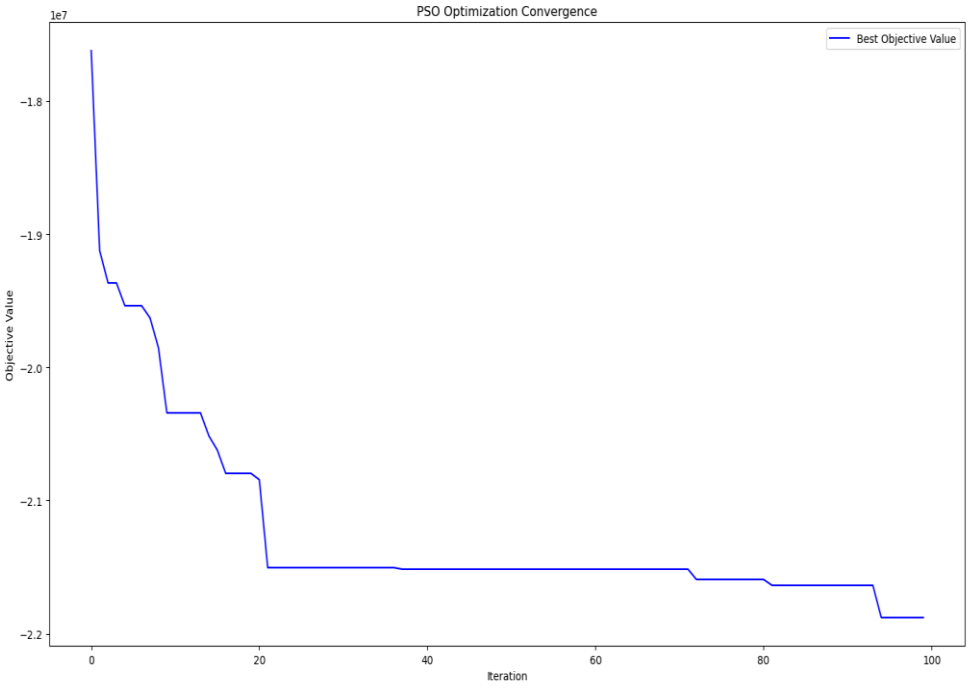


图 6 鲁棒优化策略 100 次迭代 PSO 收敛图像

200 次迭代后

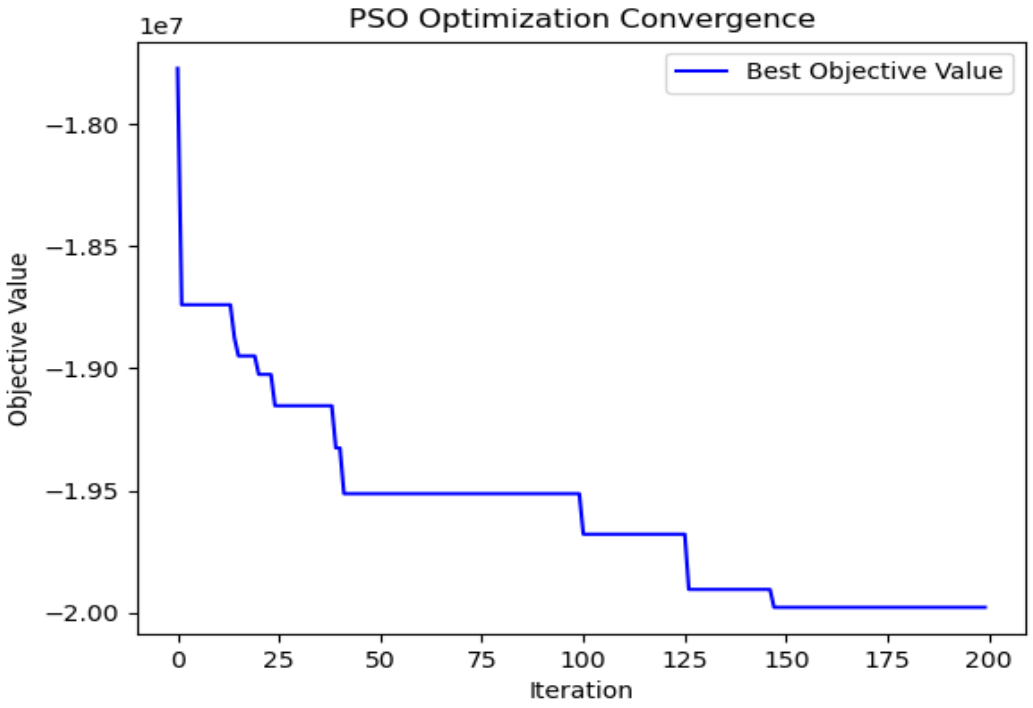


图 7 鲁棒优化策略 200 次迭代 PSO 收敛图像

5.3 问题三模型的建立与求解

5.3.1 模型的建立

题目中所提及的农作物之间的互补性和可替代性，我们想到了聚类。聚类是典型的无监督学习，相似的个体会成为一个簇，不相似的个体则会远离。我们在聚类方法中选择了 K-means 聚类方法。K-means 聚类伪代码如下

聚类簇数 k.

过程：从 D 中随机选择 k 个样本作为初始质心{u1,u2,…,uk}

repeat

令簇划分 Ci=空集（i=1~k）

for j=1,2,…,m do

计算样本 xj 与各初始质心的欧式距离:dji=||xj-ui||2;

根据距离最近的均值向量确定 xj 的簇标记;

end for

for i=1,2,…,k do

计算各簇的新质心（簇中元素的均值）;

end for

until 均值向量未更新

输出：簇划分 C={C1,C2,…,Ck}

运用 K-means，我们会得到几个聚类。我们认为，**作物的可替代性**指的是作物的数据信息相对比较相似，这里可以用聚类之内作物之间存在的关系来描述，因为聚类内的数据比较集中，显然存在着可替代性。**作物的互补性**我们认为是指作物的数据信息差异相对较大，而对于聚类之间来说，两个聚类存在差异是因为某种数据或者指标的区别（我们暂时先不考虑多个指标相同的情况，因为在本题中，随着数据量增大，数据类增多，多个指标都相同的概率会相应降低），聚类之间指标肯定有区别，我们就认为这样的两个聚类之间存在着互补的关系。

此外，为了验证我们使用聚类描述作物之间互补性和可替代性这种方法的有效性，我们又分别从种植在地块和种植在大棚两方面对作物进行了手动分类，并在后

续步骤中分别计算并比较了最终最大利润。

在聚类完成之后，我们首先需要对数据进行处理。对于预期销售量这一指标而言，需要利用 2023 年的总产量数据。但是在 2023 年种植面积为 0 的地块可能在之后的 7 年间并不会一直保持空置，因此我们需要对之前的表格进行相应的处理。具体来说，我们删去了 2023 总产量为 0 的数据，以防它们对后续模型的准确性产生影响。

另外，我们注意到题目中明确指出了需要分析预期销售量与销售价格、种植成本之间的关系。为了分析三者之间的相关性，我们首先想到使用皮尔森相关系数来进行分析。然而,使用皮尔森相关系数的前提是数据需要满足正态分布假设。因此,我们需要编写程序判断是否能符合正态分布，如果不符合的话，我们就需要用 Spearman 相关去分析数据了。我们使用了相关工具绘制热力图。这之后，我们运用了多元线性和非线性回归去建立模型[10]，以预期销售量为因变量，销售价格和种植成本为自变量，设定多元回归公式为：

$$S_i = \beta_0 + \beta_1 B_i + \beta_2 C_i + \cdots + \epsilon$$

其中β_0是截距，β_1和β_2是回归系数，ϵ是误差项，之后进行回归分析，再用 PSO 粒子群计算，并且分析模型 的正态性和独立性。

5.3.2 模型的求解

首先，我们进行了聚类。得到了下图：

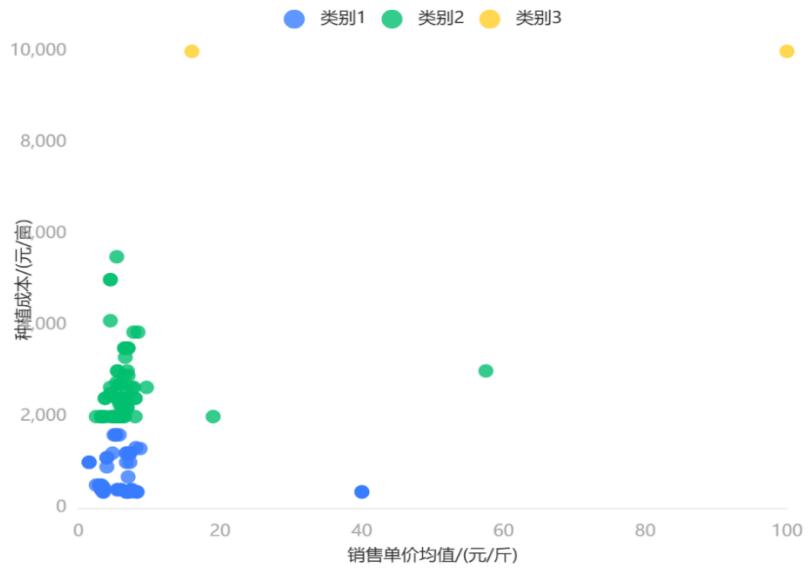


图 8 K-means 聚类

分析上图，一共是三个聚类，我们用图例标出，分别是类别 1，类别 2，以及类

别 3（具体分类情况在支撑材料中给出）。接下来，我们将这三个聚类分别进行相关性分析。

正如前文所叙，为了验证聚类体现作物互补性和可替代性行之有效，我们又进行了手动二分类，进行对照，下文所有的操作在二分类数据集上完全相同，最后我们进行了对比。

我们先编写了一段程序来判断是否能使用皮尔森系数，我们运用 Shapiro-Wilk 检验[7]和 Kolmogorov-Smirnov 检验来检验这三者是否符合正态分布。根据检验结果，我们绘制了相关的表格，为后续的分析工作奠定基础。

表 2　正态性检验分析结果

名称	样本量	平均值	标准差	偏度	峰度	Kolmogorov-Smirnov 检验		Shapiro-Wilk 检验	
						统计量 D	值 p	统计量 W	值 p
销售单价均值/(元/斤)	30	6.418	6.644	4.710	24.383	0.358	0.000**	0.438	0.000**
预期销售量/斤	30	28037.833	26395.894	1.524	1.794	0.194	0.005**	0.824	0.000**
种植成本/(元/亩)	30	572.000	339.405	1.743	2.032	0.351	0.000**	0.679	0.000**

*　$p<0.05$ **　$p<0.01$

P 值检验结果小于 0.5，不符合正态分布，说明并不能使用皮尔森系数。我们便转向了斯皮尔曼系数，

我们将处理过后的聚类数据分别带入斯皮尔曼相关的公式，这里以类别一为例子来带入。

表 3　　Spearman 相关

		销售单价均值/(元/斤)	种植成本/(元/亩)	预期销售量/斤
销售单价均值/(元/斤)	相关系数	1		

表 3　　Spearman 相关

		销售单价均值/(元/斤)	种植成本/(元/亩)	预期销售量/斤
	p 值	-		
	样本量	-		
	相关系数	-0.427*	1	
种植成本/(元/亩)	p 值	0.019	-	
	样本量	30	-	
	相关系数	-0.579**	0.247	1
预期销售量/斤	p 值	0.001	0.189	-
	样本量	30	30	-

*　p<0.05　**　p<0.01

并且使用了相关工具[8]绘制了类别 1 三者的热力图

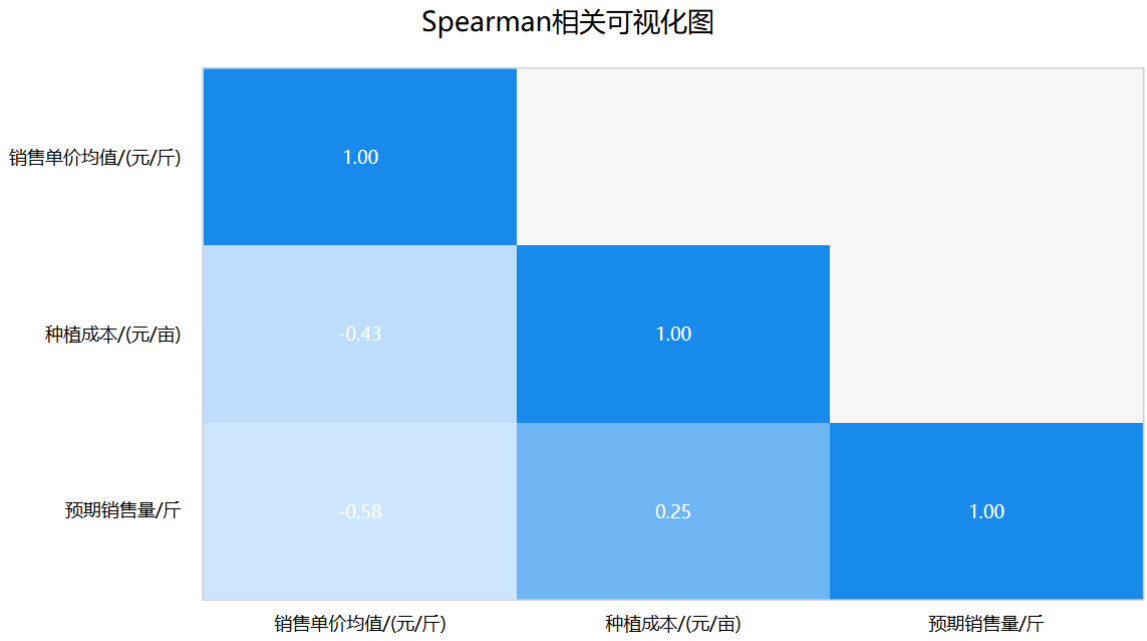


图 9 Spearman 热力图

从上表可知，利用相关分析去研究预期销售量/斤分别和销售单价均值/(元/斤)，种植成本/(元/亩)共 2 项之间的相关关系，使用 Spearman 相关系数去表示相关关系的强弱情况。具体分析可知预期销售量/斤和销售单价均值/(元/斤)之间的相关系数值

为-0.579，并且呈现出 0.01 水平的显著性，因而说明预期销售量/斤和销售单价均值/(元/斤)之间有着显著的负相关关系。预期销售量/斤和种植成本/(元/亩)之间的相关系数值为 0.247，接近于 0，并且 p 值为 0.189>0.05，因而说明预期销售量/斤和种植成本/(元/亩)之间并没有相关关系。

我们用程序(见附录)去进行线性和非线性回归分析，分别得到了如下结果（仍然以类别一为例）：

非线性回归结果：（使用高次多项式）

多项式回归的 R^2 值: 0.7802403848091567

线性回归结果：

R^2 值: 0.13716889016190736

分析以上数据，非线性回归的 R^2 达到了接近 80%的程度，远大于线性回归，可以算是较为优秀的拟合程度了。在此基础上，我们得出了聚类一的拟合函数：

Y = -146892.762+-73.923*x1+12513.405*x2+3.968*x1^2+-230.624*x1*x2+9751.42*x2^2+-0.01*x1^3+0.678*x1^2*x2+-15.959*x1*x2^2+-1049.686*x2^3+0.0*x1^4+-0.001*x1^3*x2+0.103*x1^2*x2^2+-5.269*x1*x2^3+171.25*x2^4+-0.0*x1^5+0.0*x1^4*x2+0.0*x1^3*x2^2+-0.004*x1^2*x2^3+0.233*x1*x2^4+-6.197*x2^5

我们用 matlab 和相关分析工具结合使用，得出拟合图像大致为：

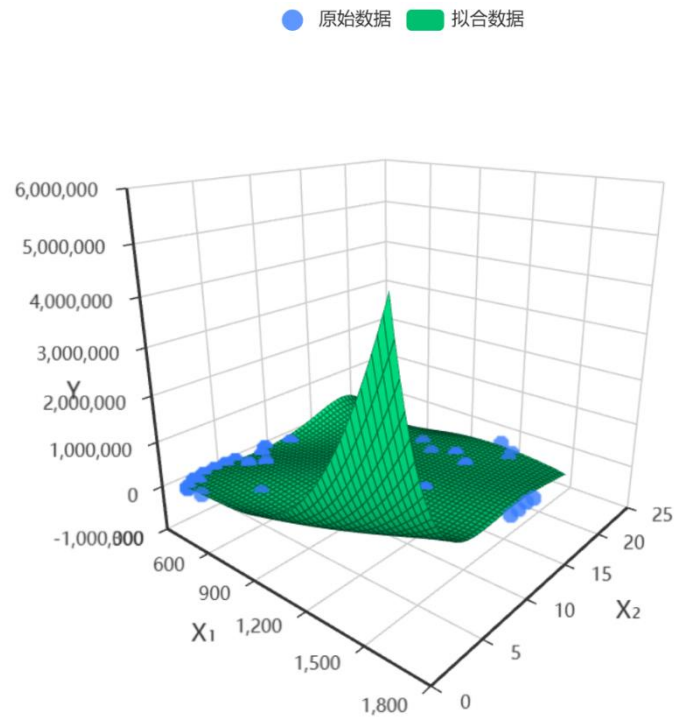


图 10 原始与拟合数据图像

得出拟合函数后，我们将其入为目标函数。结合问题二的约束变量以及决策变量，再运用 PSO 模型核算出三个聚类的累计最大收益为 37366915.74324817 元，同样的步骤，我们也对于手动划分的大棚和地块方案进行了求最大收益，求得的最大收益为 24456926.29424125 元。两个方案相对比，我们认为使用 K-means 聚类来分析可替代性和互补性是较为合理的。

在改变了类别与考虑了相关性之后得出的第三问收益，与第二问的累积最大收益相比多了将近 1000w，我们认为这是相对成功的策略。

六、 模型的分析与检验

对于模型的分析，我们采用的是敏度检验。通过绘制不同参数变化下的利润曲线，来观察哪些参数对于我们模型的影响大。首先确定分析目标，我们分析的目标是最终利润的影响，然后我们一个个的从种植成本，亩产量，市场价格等因素来分析，我们这里做的是局部灵敏度分析，因为我们模型较为复杂，采用全局灵敏度分析的话，可能会造成运行结果过长。对于模型的检验，我们参考了几种方法，其中一个是扰动参数检验，这一个我们能够参考上面的灵敏度分析。于是我们用了另外一种方法来佐证我们的模型是稳定的。我们以问题一来举例，我们总共运行了 8 次，每一次的数据分别是 21894065.869285043，22550886.954324392,

22460683.402293821，22348469.8279226134，21784569.686563911，22026106.777166397,22004080.670389224,21982076.324867335。我们将几次的图形汇表，得到下图

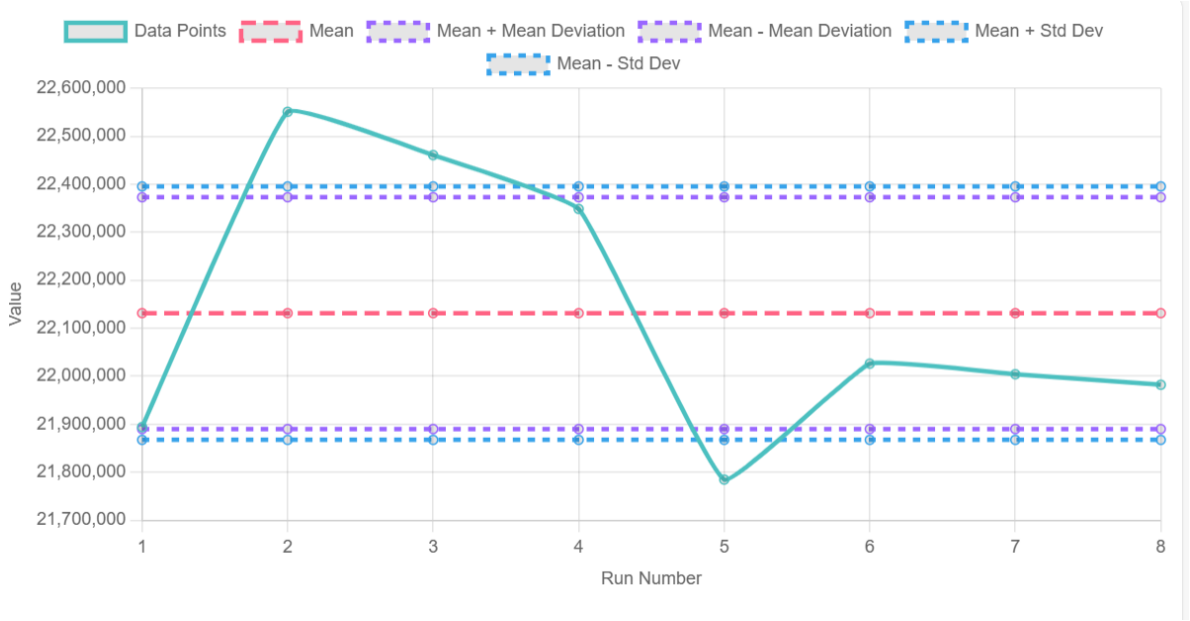


图 11 结果的折线图，标准差和平均值

经过计算，运行得到的每一个函数值都在 5%的绝对误差之内，观察图像，折现绝大部分时间都在一个标准差内，说明模型产生的结果较为稳定。

七、 模型的评价、改进与推广

7.1 模型的优点

1. 模型稳定，鲁棒，通用性较强：

我们考虑了华北地区产能与销量不稳定的特点，在算法选择上我们更偏向于稳定，不易崩溃的算法

2. 模型优化程度好，收敛快：

我们充分的考虑约束条件，由此得出的模型收敛度很高，而且收敛速度较快。

3. 模型简单，适用多种情况

我们认为我们的模型重点在于调整约束变量的关系，模型本身并没有非常复杂，将此模型推广到别的条件的耕地同样适用。

7.2 模型的缺点

受限于算力问题，迭代次数不能做的很高，导致并不能达到全局最优解。

7.3 模型的改进

提高 PSO 算法的效率，比如在适当的时机对粒子的速度或位置施加变异操作，以增加搜索空间的探索。或者结合其他算法，比如遗传，退火算法等。

7.4 模型的推广

模型推广程度高，可以根据不同地区不同的气候条件等给出不同的约束条件，能制定针对于具体地域的个性化种植服务。同时，该模型能推广到工业生产中，推广性很强。

参考文献

[1] 辛志统、小超 华北地区小农怎么应对多变的气候 [W] https://zh.fsnchina.info/post/交流会回顾 | 华北地区小农怎么应对多变的气候

[2] 苏向阳 优化|鲁棒优化基础[W] https://blog.csdn.net/weixin_53463894/article/details/128230010

[3] 马莉. MATLAB 数学实验与建模[M]. 清华大学出版社,2010.

[4] 郭文忠、陈国龙 离散粒子群优化算法及其应用[M] 清华大学出版社 2012

[5] Meetu jain、Vibha Saihjpal An Overview of Variants and Advancements of PSO Algorithm[J] *Appl. Sci.* 2022

[6] Kachitvichyanukul、Voratas. Comparison of Three Evolutionary Algorithms: GA, PSO, and DE[J] *Industrial Engineering and Management Systems* 2012

[7] Elizabeth Gonzalez、Acosta-Pech Shapiro-Wilk test for multivariate skew-normality[J] *Computaional Statistics* 37: 1985~2001

[8] 徐维超. 相关系数研究综述[J]. 广东工业大学学报,2012,29(3):12-17

[9] Scientific Platform Serving for Statistics Professional 2021. SPSSPRO. (Version 1.0.11).

[10] Kanti V.Mardia 、John T.Kent、Charles C.Taylor Multivariate Analysis[M] 2024

附录

本文使用了 spsspro 辅助绘图
所有程序和用到的表格以及结果都已放入支撑材料中

附录一（Q1_1.py 用于求解问题一第一问）

```
import pandas as pd
import numpy as np
from sko.PSO import PSO
import matplotlib.pyplot as plt


MIN_AREA_PENG = 0.2
MIN_AREA = 6
# %%
# 读取数据
land_data = pd.read_excel("地块信息.xlsx")  # 读取地块信息
crop_data = pd.read_excel("作物信息.xlsx")  # 读取作物基本信息
crop_supplement = pd.read_excel("汇总补充信息.xlsx")  # 读取作物补充信息
# %%
# 转换数据类型并清理数据
land_data["地块面积/亩"] = pd.to_numeric(
    land_data["地块面积/亩"], errors="coerce"
).fillna(0)
crop_supplement["亩产量/斤"] = pd.to_numeric(
    crop_supplement["亩产量/斤"], errors="coerce"
).fillna(0)
crop_supplement["种植成本/(元/亩)"] = pd.to_numeric(
    crop_supplement["种植成本/(元/亩)"], errors="coerce"
).fillna(0)
```

```
crop_supplement["销售单价均值/(元/斤)"] = pd.to_numeric(
    crop_supplement["销售单价均值/(元/斤)"], errors="coerce"
).fillna(0)

# %%
# 提取相关信息
areas = land_data["地块面积/亩"].tolist()  # 地块面积列表
land_names = land_data["地块名称"].tolist()  # 地块名称列表
crop_ids = crop_supplement["作物编号"].tolist()  # 作物编号列表
crop_names = crop_data["作物名称"].tolist()  # 作物名称列表
crop_yield = crop_supplement["亩产量/斤"].tolist()  # 每亩产量
crop_cost = crop_supplement["种植成本/(元/亩)"].tolist()  # 种植成本
crop_price = crop_supplement["销售单价均值/(元/斤)"].tolist()  # 销售单价
expected_demand = crop_supplement["总产量/斤"].tolist()  # 预期销售量
marks = crop_supplement["标记"].tolist()  # 标记
crop_ids = [x - 1 for x in crop_ids]
# %%
# 增加第二期地块
areas += areas[26:54]
land_names += land_names[26:54]
# %%
# 定义问题参数
num_blocks = len(areas)  # 地块数量
num_crops = len(crop_ids)  # 作物数量
num_rows = len(marks)



# %%
```

```
# def water_crop(x):  # 合法
#     for k in range(7):
#         for i in range(26, 34):
#             for j in range(0, 41):
#                 if j == 15:
#                     continue
#                 else:
#                     if x[i][j][k] != 0:
#                         return False
#         for i in range(54, 62):
#             for j in range(41):
#                 x[i][j][k] = 0
#     return True


# 约束函数


def ABC(x):  # 平旱地，梯田，山坡地
    for k in range(7):
        for block in range(0, num_blocks):
            # ABC 约束
            if block < 26:
                for j in range(15, 41):
                    x[block][j][k] = 0
    return True


def water_crop_vege(x, areas):  # 水浇地约束
```

```
    flag_crop = [False] * 8
    for k in range(7):
        for i in range(26, 34):  # 第一季
            for j in range(41):
                if j < 15 or j >= 34:
                    x[i][j][k] = 0  # 不能种水稻和蔬菜之外的
                if j == 15 and x[i][j][k] > 0:
                    for j_vege in range(16, 34):
                        x[i][j_vege][k] = 0  # 种水稻种不了蔬菜
                    flag_crop[i - 26] = True
                elif j >= 16 and j < 34 and x[i][j][k] > 0:
                    x[i][15][k] = 0
        for cnt in range(8):  # 第二季
            i = cnt + 54
            if flag_crop[cnt]:
                for j in range(41):
                    x[i][j][k] = 0
            else:
                for j in range(41):
                    if j >= 34 and j < 37:
                        continue  # 特殊蔬菜
                    else:
                        x[i][j][k] = 0
    return True


def ordipeng(x):  # 普通大棚
    for k in range(7):
        for i in range(34, 50):  # 第一季
```

```
            for j in range(41):
                if j >= 16 and j < 34:
                    continue  # 正常蔬菜
                else:
                    x[i][j][k] = 0
        for i in range(62, 78):  # 第二季
            for j in range(41):
                if j >= 37:
                    continue  # 菌类
                else:
                    x[i][j][k] = 0

    return True


def smartpeng(x):  # 智慧大棚
    for k in range(7):
        for i in range(50, 54):  # 第一季
            for j in range(41):
                if j >= 16 and j < 34:
                    continue  # 正常蔬菜
                else:
                    x[i][j][k] = 0
        for i in range(78, 82):  # 第二季
            for j in range(41):
                if j >= 16 and j < 34:
                    continue  # 正常蔬菜
                else:
                    x[i][j][k] = 0
```

```
    return True


# def seperate(x):
#     for k in range(7):
#         for i in range(82):
#             for j in range(41):
#                 if (i >= 34 and i < 54) or (i >= 62):  # 大棚
#                     if x[i][j][k] > 0 and x[i][j][k] < 0.2:
#                         x[i][j][k] = 0
#                 else:  # 普通地
#                     if x[i][j][k] > 0 and x[i][j][k] < 6:
#                         x[i][j][k] = 0
#     return True


def is_bean(x):  # 豆类约束，j 编号在 0-4，16-18 为豆
    for i in range(26):  # 粮食豆
        for j in range(0, 5):
            for k in range(0, 5):
                if x[i][j][k] + x[i][j][k + 1] + x[i][j][k + 2] == 0:
                    x[i][j][k] = 6
    for i in range(26, 54):  # 菜豆
        for j in range(16, 19):
            for k in range(0, 5):
                if x[i][j][k] + x[i][j][k + 1] + x[i][j][k + 2] == 0:
                    if i < 34:
                        x[i][j][k] = 6
                    else:
```

```
                        x[i][j][k] = 0.2
    for i in range(62, 82):
        for j in range(16, 19):
            for k in range(0, 5):
                if x[i][j][k] + x[i][j][k + 1] + x[i][j][k + 2] == 0:
                    x[i][j][k] = 0.2
    return True


def again(x):  # 重茬
    for i in range(54):
        for j in range(41):
            for k in range(6):
                if i < 26 and x[i][j][k] * x[i][j][k + 1] != 0:
                    x[i][j][k] = 0
                elif i >= 50 and (
                    x[i][j][k] * x[i + 28][j][k] != 0
                    or x[i + 28][j][k] * x[i][j][k + 1] != 0
                ):  # 含第二季的情况
                    x[i + 28][j][k] = 0
    return True


def arealimit(x):  # 种地面积约束
    for k in range(7):
        for i in range(0, 82):
            block_total_area = sum(
                x[i][j][k] for j in range(41)
            )  # 当前地块所有作物种植面积的总和
```

```
        if block_total_area > areas[i]:
            # 记录种的第一多和第二多的植物
            one = 0
            two = 0
            more = 0  # 不合格的累及面积
            for j in range(41):
                x[i][j][k] *= areas[i] / block_total_area
                if i < 34 or (i >= 54 and i < 62):  # 普通地块
                    if j > 5:
                        if x[i][j][k] > x[i][one][k]:
                            one = j
                        elif x[i][j][k] > x[i][two][k]:
                            two = j
                        if x[i][j][k] < 6:
                            more += x[i][j][k]  # 不满足条件的多余
的加起来
                            x[i][j][k] = 0
                    else:
                        continue
                else:  # 大棚
                    if j < 16 or j >= 19:  # 非豆类
                        if x[i][j][k] > x[i][one][k]:
                            one = j
                        elif x[i][j][k] > x[i][two][k]:
                            two = j
                        if x[i][j][k] < 0.2:
                            more += x[i][j][k]
                            x[i][j][k] = 0
            x[i][one][k] += more / 2
```

```
                x[i][two][k] += more / 2
    return True


# %%


def select_range(x, s, j, crop_areas, k):
    if s == "平旱地单季":
        for i in range(6):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "梯田单季":
        for i in range(6, 20):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "山坡地单季":
        for i in range(20, 26):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地单季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第一季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第二季":
```

```
        for i in range(54, 62):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "普通大棚第一季":
        for i in range(34, 50):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "智慧大棚第一季":
        for i in range(50, 54):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "普通大棚第二季":
        for i in range(62, 78):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "智慧大棚第二季":
        for i in range(78, 82):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    return crop_areas


# %%
# 定义目标函数
def profit_function(x):
    x = np.reshape(x, (82, 41, 7))  # 变三维数组
    total_profit = 0
    # 多个约束条件
    is_bean(x)
```

```
        again(x)
        ABC(x)
        water_crop_vege(x, areas)
        ordipeng(x)
        smartpeng(x)
        # seperate(x)
        arealimit(x)
        water_crop_vege(x, areas)
        ordipeng(x)
        smartpeng(x)

        for k in range(7):
            for row in range(num_rows):
                crop_areas = 0
                j = crop_ids[row]
                crop_areas = select_range(x, marks[row], j, crop_areas, k)
                production = crop_yield[row] * crop_areas  # 计算产量
                cost = crop_cost[row] * crop_areas  # 计算种植成本
                revenue = (
                    min(production, expected_demand[row]) * crop_price[row]
                )  # 计算有效收入(考虑滞销)
                total_profit += (
                    revenue
                    - cost
                    + max(0, production - expected_demand[row]) *
crop_price[row] / 2
                )  # 累加净收益
        return -total_profit  # 目标是最大化收益，因此返回负的净收益
    # 定义种植面积的上下界
```

```
lb = [0] * (num_blocks * 41 * 7)  # 下界为0
ub = [
    area for block_area in areas for area in [block_area] * 41 * 7
]  # 上界为各地块面积
# 使用PSO算法进行优化
pso = PSO(
    func=profit_function,
    dim=82 * 41 * 7,
    pop=50,
    max_iter=100,
    lb=lb,
    ub=ub,
    w=0.9,
    c1=2,
    c2=2,
)
pso.run()
best_pos = pso.gbest_x
best_profit = pso.gbest_y
# %%
# 处理优化结果
best_pos = [round(p, 2) for p in best_pos]  # 将结果四舍五入到小数点后两位
len(best_pos)
# 绘制全局最优目标函数值随迭代次数的变化
plt.plot(pso.gbest_y_hist, "b-", label="Best Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("PSO Optimization Convergence")
```

```
    plt.legend()
    plt.show()
    # %%
    # 打印最终的最大化的利润
    final_profit = -best_profit  # 由于目标函数返回的是负的净收益，这里取相反数
    print(f"2024 年到 2030 年累计的最终最大利润: {final_profit}元")
    # %%
    # 读取结果模板
    xls = pd.ExcelFile("result1_1.xlsx")
    best_pos_3d = np.reshape(best_pos, (82, 41, 7))
    # 读 sheet
    dfs = {sheet: xls.parse(sheet) for sheet in xls.sheet_names}

    for idx, sheet_name in enumerate(xls.sheet_names):
        # 第三维度是表的个数
        sheet_data = best_pos_3d[:, :, idx]

        # 逐行
        for row in range(sheet_data.shape[0]):
            for col in range(sheet_data.shape[1]):
                dfs[sheet_name].iloc[row, col] = sheet_data[row, col]

    # 保存
    output_path = "updated_result1_1.xlsx"
    with pd.ExcelWriter(output_path) as writer:
        for sheet_name, df in dfs.items():
            df.to_excel(writer, sheet_name=sheet_name, index=False)
```

附录二（Q1_2.py 用于求解问题一第二问）

```
# %%
import pandas as pd
import numpy as np
from sko.PSO import PSO
import matplotlib.pyplot as plt

MIN_AREA_PENG = 0.2
MIN_AREA = 6
# %%
# 读取数据
land_data = pd.read_excel("地块信息.xlsx")   # 读取地块信息
crop_data = pd.read_excel("作物信息.xlsx")   # 读取作物基本信息
crop_supplement = pd.read_excel("汇总补充信息.xlsx")   # 读取作物补充信息
# %%
# 转换数据类型并清理数据
land_data["地块面积/亩"] = pd.to_numeric(
      land_data["地块面积/亩"], errors="coerce"
).fillna(0)
crop_supplement["亩产量/斤"] = pd.to_numeric(
      crop_supplement["亩产量/斤"], errors="coerce"
).fillna(0)
crop_supplement["种植成本/(元/亩)"] = pd.to_numeric(
      crop_supplement["种植成本/(元/亩)"], errors="coerce"
).fillna(0)
crop_supplement["销售单价均值/(元/斤)"] = pd.to_numeric(
      crop_supplement["销售单价均值/(元/斤)"], errors="coerce"
).fillna(0)
```

```
# %%
# 提取相关信息
areas = land_data["地块面积/亩"].tolist()  # 地块面积列表
land_names = land_data["地块名称"].tolist()  # 地块名称列表
crop_ids = crop_supplement["作物编号"].tolist()  # 作物编号列表
crop_names = crop_data["作物名称"].tolist()  # 作物名称列表
crop_yield = crop_supplement["亩产量/斤"].tolist()  # 每亩产量
crop_cost = crop_supplement["种植成本/(元/亩)"].tolist()  # 种植成本
crop_price = crop_supplement["销售单价均值/(元/斤)"].tolist()  # 销售单价
expected_demand = crop_supplement["总产量/斤"].tolist()  # 预期销售量
marks = crop_supplement["标记"].tolist()  # 标记
crop_ids = [x - 1 for x in crop_ids]
# %%
# 增加第二期地块
areas += areas[26:54]
land_names += land_names[26:54]
# %%
# 定义问题参数
num_blocks = len(areas)  # 地块数量
num_crops = len(crop_ids)  # 作物数量
num_rows = len(marks)

# %%
# def water_crop(x):  # 合法
#       for k in range(7):
#             for i in range(26, 34):
#                   for j in range(0, 41):
```

```
#                         if j == 15:
#                               continue
#                         else:
#                               if x[i][j][k] != 0:
#                                     return False
#             for i in range(54, 62):
#                   for j in range(41):
#                         x[i][j][k] = 0
#       return True

# 约束函数

def ABC(x):   # 平旱地，梯田，山坡地
      for k in range(7):
            for block in range(0, num_blocks):
                  # ABC 约束
                  if block < 26:
                        for j in range(15, 41):
                              x[block][j][k] = 0
      return True

def water_crop_vege(x, areas):   # 水浇地约束
      flag_crop = [False] * 8
      for k in range(7):
            for i in range(26, 34):   # 第一季
                  for j in range(41):
                        if j < 15 or j >= 34:
                              x[i][j][k] = 0   # 不能种水稻和蔬菜之外
的
```

```
                        if j == 15 and x[i][j][k] > 0:
                              for j_vege in range(16, 34):
                                    x[i][j_vege][k] = 0   # 种水稻种
不了蔬菜
                              flag_crop[i - 26] = True
                        elif j >= 16 and j < 34 and x[i][j][k] > 0:
                              x[i][15][k] = 0
            for cnt in range(8):   # 第二季
                  i = cnt + 54
                  if flag_crop[cnt]:
                        for j in range(41):
                              x[i][j][k] = 0
                  else:
                        for j in range(41):
                              if j >= 34 and j < 37:
                                    continue   # 特殊蔬菜
                              else:
                                    x[i][j][k] = 0
      return True

def ordipeng(x):   # 普通大棚
      for k in range(7):
            for i in range(34, 50):   # 第一季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
            for i in range(62, 78):   # 第二季
```

```
                for j in range(41):
                    if j >= 37:
                        continue   # 菌类
                    else:
                        x[i][j][k] = 0

    return True

def smartpeng(x):   # 智慧大棚
    for k in range(7):
        for i in range(50, 54):   # 第一季
            for j in range(41):
                if j >= 16 and j < 34:
                    continue   # 正常蔬菜
                else:
                    x[i][j][k] = 0
        for i in range(78, 82):   # 第二季
            for j in range(41):
                if j >= 16 and j < 34:
                    continue   # 正常蔬菜
                else:
                    x[i][j][k] = 0
    return True

# def seperate(x):
#     for k in range(7):
#         for i in range(82):
#             for j in range(41):
#                 if (i >= 34 and i < 54) or (i >= 62):   # 大
```

```
棚
#                                   if x[i][j][k] > 0 and x[i][j][k] <
0.2:
#                                       x[i][j][k] = 0
#                           else:   # 普通地
#                                 if x[i][j][k] > 0 and x[i][j][k] < 6:
#                                       x[i][j][k] = 0
#       return True

def is_bean(x):   # 豆类约束，j 编号在 0-4，16-18 为豆
      for i in range(26):   # 粮食豆
            for j in range(0, 5):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              x[i][j][k] = 6
      for i in range(26, 54):   # 菜豆
            for j in range(16, 19):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              if i < 34:
                                    x[i][j][k] = 6
                              else:
                                    x[i][j][k] = 0.2
      for i in range(62, 82):
            for j in range(16, 19):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
```

```
2] == 0:
                                x[i][j][k] = 0.2
        return True


def again(x):   # 重茬
        for i in range(54):
                for j in range(41):
                        for k in range(6):
                                if i < 26 and x[i][j][k] * x[i][j][k + 1] !=
0:
                                        x[i][j][k] = 0
                                elif i >= 50 and (
                                        x[i][j][k] * x[i + 28][j][k] != 0
                                        or x[i + 28][j][k] * x[i][j][k + 1] != 0
                                ):   # 含第二季的情况
                                        x[i + 28][j][k] = 0
        return True


def arealimit(x):   # 种地面积约束
        for k in range(7):
                for i in range(0, 82):
                        block_total_area = sum(
                                x[i][j][k] for j in range(41)
                        )   # 当前地块所有作物种植面积的总和
                        if block_total_area > areas[i]:
                                # 记录种的第一多和第二多的植物
                                one = 0
                                two = 0
                                more = 0   # 不合格的累及面积
```

```
                for j in range(41):
                        x[i][j][k]  *=  areas[i]  /
block_total_area
                if i < 34 or (i >= 54 and i < 62):   #
普通地块
                    if j > 5:
                        if x[i][j][k] > x[i][one][k]:
                            one = j
                            elif  x[i][j][k]  >
x[i][two][k]:
                            two = j
                        if x[i][j][k] < 6:
                            more += x[i][j][k]   #
不满足条件的多余的加起来
                            x[i][j][k] = 0
                    else:
                        continue
                else:   # 大棚
                    if j < 16 or j >= 19:   # 非豆类
                        if x[i][j][k] > x[i][one][k]:
                            one = j
                            elif  x[i][j][k]  >
x[i][two][k]:
                            two = j
                        if x[i][j][k] < 0.2:
                            more += x[i][j][k]
                            x[i][j][k] = 0
                x[i][one][k] += more / 2
                x[i][two][k] += more / 2
```

```
        return True


# %%


def select_range(x, s, j, crop_areas, k):
        if s == "平旱地单季":
                for i in range(6):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "梯田单季":
                for i in range(6, 20):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "山坡地单季":
                for i in range(20, 26):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "水浇地单季":
                for i in range(26, 34):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "水浇地第一季":
                for i in range(26, 34):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "水浇地第二季":
                for i in range(54, 62):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
```

```
        elif s == "普通大棚第一季":
            for i in range(34, 50):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第一季":
            for i in range(50, 54):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "普通大棚第二季":
            for i in range(62, 78):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第二季":
            for i in range(78, 82):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        return crop_areas


# %%
# 定义目标函数
def profit_function(x):
        x = np.reshape(x, (82, 41, 7))   # 变三维数组
        total_profit = 0
        # 多个约束条件
        is_bean(x)
        again(x)
        ABC(x)
        water_crop_vege(x, areas)
        ordipeng(x)
```

```
    smartpeng(x)
    # seperate(x)
    arealimit(x)
    water_crop_vege(x, areas)
    ordipeng(x)
    smartpeng(x)

    for k in range(7):
        for row in range(num_rows):
            crop_areas = 0
            j = crop_ids[row]
                crop_areas = select_range(x, marks[row], j,
crop_areas, k)
             production = crop_yield[row] * crop_areas   # 计算
产量
            cost = crop_cost[row] * crop_areas   # 计算种植成本
            revenue = (
                        min(production, expected_demand[row]) *
crop_price[row]
            )   # 计算有效收入(考虑滞销)
            total_profit += (
                revenue
                - cost
                 + max(0, production - expected_demand[row]) *
crop_price[row] / 2
            )   # 累加净收益
    return -total_profit   # 目标是最大化收益，因此返回负的净收益

# %%
```

```
# 定义种植面积的上下界
lb = [0] * (num_blocks * 41 * 7)   # 下界为0
ub = [
      area for block_area in areas for area in [block_area] * 41 * 7
]   # 上界为各地块面积
# 使用PSO算法进行优化
pso = PSO(
      func=profit_function,
      dim=82 * 41 * 7,
      pop=50,
      max_iter=100,
      lb=lb,
      ub=ub,
      w=0.9,
      c1=2,
      c2=2,
)
pso.run()
best_pos = pso.gbest_x
best_profit = pso.gbest_y
# %%
# 处理优化结果
best_pos = [round(p, 2) for p in best_pos]   # 将结果四舍五入到小数点后两位
len(best_pos)
# 绘制全局最优目标函数值随迭代次数的变化
plt.plot(pso.gbest_y_hist, "b-", label="Best Objective Value")
plt.xlabel("Iteration")
```

```
plt.ylabel("Objective Value")
plt.title("PSO Optimization Convergence")
plt.legend()
plt.show()
# %%
# 打印最终的最大化的利润
final_profit = -best_profit   # 由于目标函数返回的是负的净收益，这里取相反数
print(f"2024 年到 2030 年累计的最终最大利润: {final_profit}元")
# %%
# 读取结果模板
xls = pd.ExcelFile("result1_1.xlsx")
best_pos_3d = np.reshape(best_pos, (82, 41, 7))
# 读 sheet
dfs = {sheet: xls.parse(sheet) for sheet in xls.sheet_names}

for idx, sheet_name in enumerate(xls.sheet_names):
      # 第三维度是表的个数
      sheet_data = best_pos_3d[:, :, idx]

      # 逐行
      for row in range(sheet_data.shape[0]):
            for col in range(sheet_data.shape[1]):
                  dfs[sheet_name].iloc[row, col] = sheet_data[row, col]

# 保存
output_path = "updated_result1_1.xlsx"
with pd.ExcelWriter(output_path) as writer:
      for sheet_name, df in dfs.items():
```

```
            df.to_excel(writer, sheet_name=sheet_name, index=False)
```

附录三（Q2.py 用于求解问题二）

```
# %%
import pandas as pd
import numpy as np
from sko.PSO import PSO
import matplotlib.pyplot as plt

MIN_AREA_PENG = 0.2
MIN_AREA = 6
# %%
# 读取数据
land_data = pd.read_excel("附件 1.xlsx")   # 读取地块信息
land_supplement = pd.read_excel("附件 1-补充.xlsx")   # 读取地块补充信息
crop_data = pd.read_excel("2023 总产量.xlsx")   # 读取作物基本信息
crop_supplement = pd.read_excel("牛逼文件.xlsx")   # 读取作物补充信息
# %%
# 转换数据类型并清理数据
land_data["地块面积/亩"] = pd.to_numeric(
      land_data["地块面积/亩"], errors="coerce"
).fillna(0)
crop_supplement["亩产量/斤"] = pd.to_numeric(
      crop_supplement["亩产量/斤"], errors="coerce"
).fillna(0)
crop_supplement["种植成本/(元/亩)"] = pd.to_numeric(
      crop_supplement["种植成本/(元/亩)"], errors="coerce"
).fillna(0)
```

```
crop_supplement["销售单价均值/(元/斤)"] = pd.to_numeric(
      crop_supplement["销售单价均值/(元/斤)"], errors="coerce"
).fillna(0)

# %%
# 提取相关信息
areas = land_data["地块面积/亩"].tolist()   # 地块面积列表
land_names = land_data["地块名称"].tolist()   # 地块名称列表
crop_ids = crop_supplement["作物编号"].tolist()   # 作物编号列表
crop_names = crop_data["作物名称"].tolist()   # 作物名称列表
crop_yield = crop_supplement["亩产量/斤"].tolist()   # 每亩产量
crop_cost = crop_supplement["种植成本/(元/亩)"].tolist()   # 种植成本
crop_price = crop_supplement["销售单价均值/(元/斤)"].tolist()   # 销售
单价
expected_demand = crop_supplement["总产量/斤"].tolist()   # 预期销售量
marks = crop_supplement["标记"].tolist()   # 标记
crop_ids = [x - 1 for x in crop_ids]
# %%
# 增加第二期地块
areas += areas[26:54]
land_names += land_names[26:54]
# %%
# 定义问题参数
num_blocks = len(areas)   # 地块数量
num_crops = len(crop_ids)   # 作物数量
num_rows = len(marks)

# %%
# def water_crop(x):   # 合法
```

```
#       for k in range(7):
#             for i in range(26, 34):
#                   for j in range(0, 41):
#                         if j == 15:
#                               continue
#                         else:
#                               if x[i][j][k] != 0:
#                                     return False
#             for i in range(54, 62):
#                   for j in range(41):
#                         x[i][j][k] = 0
#       return True

def water_crop_vege(x):
      flag_crop = False
      flag_vege = False
      for k in range(7):
            for i in range(26, 34):   # 第一季
                  for j in range(41):
                        if (j < 15 or j >= 34) and x[i][j][k] > 0:
                              x[i][j][k] = 0
                        if j == 15 and x[i][j][k] > 0:
                              flag_crop = True
                        elif j >= 16 and j < 34 and x[i][j][k] > 0:
                              flag_vege = True
            if flag_vege:
                  for i in range(54, 62):   # 第二季
                        for j in range(41):
                              if j >= 34 and j < 37:
```

```
                                    continue   # 特殊蔬菜
                              else:
                                    x[i][j][k] = 0
            if flag_crop:
                  for i in range(54, 62):
                        for j in range(41):
                              x[i][j][k] = 0
      return True

def ordipeng(x):
      for k in range(7):
            for i in range(34, 50):
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
            for i in range(62, 78):
                  for j in range(41):
                        if j >= 37:
                              continue   # 菌类
                        else:
                              x[i][j][k] = 0

      return True

def smartpeng(x):
      for k in range(7):
            for i in range(50, 54):
```

```
                for j in range(41):
                    if j >= 16 and j < 34:
                        continue   # 正常蔬菜
                    else:
                        x[i][j][k] = 0
            for i in range(78, 82):
                for j in range(41):
                    if j >= 16 and j < 34:
                        continue   # 正常蔬菜
                    else:
                        x[i][j][k] = 0
    return True

def seperate(x):
    for k in range(7):
        for i in range(82):
            for j in range(41):
                if (i >= 34 and i < 54) or (i >= 62):   # 大棚
                    if x[i][j][k] > 0 and x[i][j][k] < 0.2:
                        x[i][j][k] = 0
                else:   # 普通地
                    if x[i][j][k] > 0 and x[i][j][k] < 6:
                        x[i][j][k] = 0
    return True

def is_bean(x): #j 编号在 0-4，16-18 为豆
    for i in range(82):
        for j in range(0, 5):
            for k in range(0, 5):
```

```
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              if i < 34 or (i >= 54 and i < 62):
                                    x[i][j][k] = 6
                              else:
                                    x[i][j][k] = 0.2
            for j in range(16, 19):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              if i < 34 or (i >= 54 and i < 62):
                                    x[i][j][k] = 6
                              else:
                                    x[i][j][k] = 0.2
      return True


def again(x):
      for i in range(54):
            for j in range(41):
                  for k in range(6):
                        if i < 26 and x[i][j][k] * x[i][j][k + 1] !=
0:
                              x[i][j][k] = 0
                        elif i >= 50 and (
                              x[i][j][k] * x[i + 28][j][k] != 0
                              or x[i + 28][j][k] * x[i][j][k + 1] != 0
                        ):
                              x[i + 28][j][k] = 0
```

```
    return True


# %%


def select_range(x, s, j, crop_areas, k):
    if s == "平旱地单季":
        for i in range(6):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "梯田单季":
        for i in range(6, 20):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "山坡地单季":
        for i in range(20, 26):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地单季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第一季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第二季":
        for i in range(54, 62):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
```

```
        elif s == "普通大棚第一季":
            for i in range(34, 50):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第一季":
            for i in range(50, 54):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "普通大棚第二季":
            for i in range(62, 78):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第二季":
            for i in range(78, 82):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        return crop_areas


# %%
# 定义目标函数
def profit_function(x):
        x = np.reshape(x, (82, 41, 7))   # 变三维数组
        total_profit = 0
          # 多个约束条件
        water_crop_vege(x)
        ordipeng(x)
        smartpeng(x)
        is_bean(x)
        again(x)
```

```
for k in range(7):
    for block in range(0, num_blocks):
        # ABC 约束
        if block < 26:
            for j in range(16, 41):
                x[block][j][k] = 0
        # 第 7 项约束
        block_total_area = sum(x[block][0:41][k])   # 当前地块所有作物种植面积的总和
        if block_total_area > areas[block]:
            # over_area = block_total_area - areas[block] #多的面积
            # block_total_area -= over_area
            # cnt = sum(1 for j in range(41) if x[block][j][k] != 0)
            # over_area_average =over_area / cnt
            for j in range(41):
                x[block][j][k] *= (areas[block] / block_total_area)
#鲁棒优化
# 调整预期销售量
num_scenarios = 40
scenarios_profit = []

for _ in range(num_scenarios):
    for k in range(7):
        for row in range(num_rows):
            crop_areas = 0
            j = crop_ids[row]
```

```
                    crop_areas = select_range(x, marks[row], j,
crop_areas, k)
                # 鲁棒优化
                # 预期销售量
                if j == 6 or j == 7: # 小麦，玉米
                    demand_change = np.random.uniform(1.05,
1.10) ** k #按年增长
                else:
                    demand_change = np.random.uniform(0.95,
1.05) ** k
                # 亩产
                yield_change = np.random.uniform(0.90, 1.10)
                # 成本
                cost_change = np.random.uniform(1.04, 1.06) **
k
                # 蔬菜售价
                if j >=16 and j < 37:
                    price_change = np.random.uniform(1.04,
1.06) ** k
                #食用菌
                if j >= 37 and j < 40: #菌类（除羊肚菌）
                    price_change = np.random.uniform(0.95,
0.99) ** k
                elif j == 40: #羊肚菌
                    price_change = 0.95 ** k
                production = crop_yield[row] * yield_change *
crop_areas   # 计算产量
                cost = crop_cost[row] * cost_change *
crop_areas   #
```

```
                        expected_demand[row] *= demand_change
                        crop_price[row] *= price_change
                        revenue = (
                            min(production, expected_demand[row]) *
crop_price[row]
                        )   # 计算有效收入(考虑滞销)
                            total_profit += revenue - cost + max(0,
production - expected_demand[row]) * crop_price[row] / 2 # 累加净收益
            scenarios_profit.append(total_profit)
        total_profit = min(scenarios_profit)
        return -total_profit   # 目标是最大化收益，因此返回负的净收益
#%%

# 定义种植面积的上下界
lb = [0] * (num_blocks * 41 * 7)   # 下界为0
ub = [
        area for block_area in areas for area in [block_area] * 41 * 7
]   # 上界为各地块面积
# 使用PSO算法进行优化
pso = PSO(func=profit_function, dim= 82*41*7 ,pop=50, max_iter=50,lb=lb,
ub=ub, w=0.8,c1=2,c2=2)
pso.run()
best_pos = pso.gbest_x
best_profit = pso.gbest_y
# %%
# 处理优化结果
best_pos = [round(p, 2) for p in best_pos]   # 将结果四舍五入到小数点
后两位
len(best_pos)
```

```
# 绘制全局最优目标函数值随迭代次数的变化
plt.plot(pso.gbest_y_hist, 'b-', label='Best Objective Value')
plt.xlabel('Iteration')
plt.ylabel('Objective Value')
plt.title('PSO Optimization Convergence')
plt.legend()
plt.show()
# %%
# 打印最终的最大化的利润
final_profit = -best_profit   # 由于目标函数返回的是负的净收益，这里取相反数
print(f"2024 年到 2030 年累计的最终最大利润: {final_profit}元")

# 读取结果模板
result_df = pd.read_excel("result1_1.xlsx")
best_pos_3d = np.reshape(best_pos, (82, 41, 7))
for block_index in range(82):
      for crop_index in range(41):
            for season_index in range(7):
                                                crop_area       = best_pos_3d[block_index][crop_index][season_index]
                  if crop_area > 0:   # 只填充大于 0 的值
                        crop_name = crop_names[crop_index]   # 从作物名称列表中获取作物名称
                        block_name = land_names[block_index]   # 从地块名称列表中获取地块名称

                         # 检查地块名称和作物名称是否在 DataFrame 的列中
```

```
                        if (
                                block_name in result_df["地块名"].values
                                and crop_name in result_df.columns
                        ):
                                # 填充数据
                                    result_df.loc[result_df["地块名"] ==
block_name, crop_name] = (
                                       crop_area
                                )

# 保存结果到 Excel
result_df.to_excel("result_filled.xlsx", index=False)
print("最优种植方案已保存到'result_filled.xlsx'.")
```

附录四（Q3_cluster.py 用于求解第三问（kmeans 聚类））

```
import pandas as pd
import numpy as np
from sko.PSO import PSO
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from            statsmodels.stats.outliers_influence            import
variance_inflation_factor
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import curve_fit

def linear(file_path):
```

```
    # Step 1: 读取数据
    df = pd.read_excel(file_path, sheet_name="Sheet1")
    # Step 2: 数据预处理，选择相关的列
    X = df[["种植成本/(元/亩)", "销售单价均值/(元/斤)"]].copy()
    y = df["预期销售量/斤"]

    # Step 3: 填充缺失值（如有）
    X = X.fillna(0)
    y = y.fillna(0)
    # 转换为二次多项式特征
    poly = PolynomialFeatures(degree=4)
    X_poly = poly.fit_transform(X)

    # 拟合多项式回归模型
    poly_reg_model = LinearRegression()
    poly_reg_model.fit(X_poly, y)

    # 提取回归系数和截距
    coefficients = poly_reg_model.coef_
    intercept = poly_reg_model.intercept_
    return coefficients, intercept

# Step 1: 读取数据
file_path1 = "相关性信息_类别1.xlsx"
file_path2 = "相关性信息_类别2.xlsx"
file_path3 = "相关性信息_类别3.xlsx"

coefficients_1, intercept_1 = linear(file_path1)
coefficients_2, intercept_2 = linear(file_path2)
```

```
coefficients_3, intercept_3 = linear(file_path3)

def predict_sale(x1, x2, coefficients, intercept):
    """
    根据多项式回归模型计算 y 值。

    参数:
    x1 -- 种植成本/(元/亩)
    x2 -- 销售单价均值/(元/斤)
    coefficients -- 多项式回归模型的系数列表
    intercept -- 多项式回归模型的截距

    返回:
    y -- 计算得到的预期销售量/斤
    """
    # 构建特征列表，包括所有可能的四次项
    features = [
        1,
        x1,
        x2,
        x1**2,
        x1 * x2,
        x2**2,
        x1**3,
        x1**2 * x2,
        x1 * x2**2,
        x2**3,
        x1**4,
        x1**3 * x2,
```

```
            x1**2 * x2**2,
            x1 * x2**3,
            x2**4,
        ]
        # 计算 y 值
        y = intercept
        for coef, feature in zip(coefficients, features):
            y += coef * feature

        return y

MIN_AREA_PENG = 0.2
MIN_AREA = 6
# %%
# 读取数据
land_data = pd.read_excel("地块信息.xlsx")   # 读取地块信息
crop_data = pd.read_excel("作物信息.xlsx")   # 读取作物基本信息
crop_supplement = pd.read_excel("相关性信息.xlsx")   # 读取作物补充信息
# %%
# 转换数据类型并清理数据
land_data["地块面积/亩"] = pd.to_numeric(
    land_data["地块面积/亩"], errors="coerce"
).fillna(0)
crop_supplement["亩产量/斤"] = pd.to_numeric(
    crop_supplement["亩产量/斤"], errors="coerce"
).fillna(0)
crop_supplement["种植成本/(元/亩)"] = pd.to_numeric(
    crop_supplement["种植成本/(元/亩)"], errors="coerce"
```

```
).fillna(0)
crop_supplement["销售单价均值/(元/斤)"] = pd.to_numeric(
      crop_supplement["销售单价均值/(元/斤)"], errors="coerce"
).fillna(0)

# %%
# 提取相关信息
areas = land_data["地块面积/亩"].tolist()   # 地块面积列表
land_names = land_data["地块名称"].tolist()   # 地块名称列表
crop_ids = crop_supplement["作物编号"].tolist()   # 作物编号列表
crop_names = crop_data["作物名称"].tolist()   # 作物名称列表
crop_cluster_group = crop_supplement["聚类种类"].tolist()   # 作物聚类后的类别
crop_yield = crop_supplement["亩产量/斤"].tolist()   # 每亩产量
crop_cost = crop_supplement["种植成本/(元/亩)"].tolist()   # 种植成本
crop_price = crop_supplement["销售单价均值/(元/斤)"].tolist()   # 销售单价
expected_demand = crop_supplement["预期销售量/斤"].tolist()   # 预期销售量
marks = crop_supplement["标记"].tolist()   # 标记
crop_ids = [x - 1 for x in crop_ids]
# %%
# 增加第二期地块
areas += areas[26:54]
land_names += land_names[26:54]
# %%
# 定义问题参数
num_blocks = len(areas)   # 地块数量
num_crops = len(crop_ids)   # 作物数量
```

```
num_rows = len(marks)

# %%
# def water_crop(x):   # 合法
#       for k in range(7):
#             for i in range(26, 34):
#                   for j in range(0, 41):
#                         if j == 15:
#                               continue
#                         else:
#                               if x[i][j][k] != 0:
#                                     return False
#             for i in range(54, 62):
#                   for j in range(41):
#                         x[i][j][k] = 0
#       return True

# 约束函数

def ABC(x):   # 平旱地，梯田，山坡地
      for k in range(7):
            for block in range(0, num_blocks):
                  # ABC 约束
                  if block < 26:
                        for j in range(15, 41):
                              x[block][j][k] = 0
      return True

def water_crop_vege(x, areas):   # 水浇地约束
```

```
    flag_crop = [False] * 8
    for k in range(7):
        for i in range(26, 34):   # 第一季
            for j in range(41):
                if j < 15 or j >= 34:
                    x[i][j][k] = 0   # 不能种水稻和蔬菜之外的
                if j == 15 and x[i][j][k] > 0:
                    for j_vege in range(16, 34):
                        x[i][j_vege][k] = 0   # 种水稻种不了蔬菜
                    flag_crop[i - 26] = True
                elif j >= 16 and j < 34 and x[i][j][k] > 0:
                    x[i][15][k] = 0
        for cnt in range(8):   # 第二季
            i = cnt + 54
            if flag_crop[cnt]:
                for j in range(41):
                    x[i][j][k] = 0
            else:
                for j in range(41):
                    if j >= 34 and j < 37:
                        continue   # 特殊蔬菜
                    else:
                        x[i][j][k] = 0
    return True

def ordipeng(x):   # 普通大棚
    for k in range(7):
```

```
            for i in range(34, 50):   # 第一季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
            for i in range(62, 78):   # 第二季
                  for j in range(41):
                        if j >= 37:
                              continue   # 菌类
                        else:
                              x[i][j][k] = 0

      return True

def smartpeng(x):   # 智慧大棚
      for k in range(7):
            for i in range(50, 54):   # 第一季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
            for i in range(78, 82):   # 第二季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
```

```
    return True

# def seperate(x):
#     for k in range(7):
#         for i in range(82):
#             for j in range(41):
#                 if (i >= 34 and i < 54) or (i >= 62):  # 大棚
#                     if x[i][j][k] > 0 and x[i][j][k] < 0.2:
#                         x[i][j][k] = 0
#                 else:  # 普通地
#                     if x[i][j][k] > 0 and x[i][j][k] < 6:
#                         x[i][j][k] = 0
#     return True

def is_bean(x):  # 豆类约束，j编号在0-4，16-18为豆
    for i in range(26):  # 粮食豆
        for j in range(0, 5):
            for k in range(0, 5):
                if x[i][j][k] + x[i][j][k + 1] + x[i][j][k + 2] == 0:
                    x[i][j][k] = 6
    for i in range(26, 54):  # 菜豆
        for j in range(16, 19):
            for k in range(0, 5):
                if x[i][j][k] + x[i][j][k + 1] + x[i][j][k + 2] == 0:
                    if i < 34:
```

```
                                    x[i][j][k] = 6
                              else:
                                    x[i][j][k] = 0.2
        for i in range(62, 82):
              for j in range(16, 19):
                    for k in range(0, 5):
                          if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                                x[i][j][k] = 0.2
        return True

def again(x):   # 重茬
        for i in range(54):
              for j in range(41):
                    for k in range(6):
                          if i < 26 and x[i][j][k] * x[i][j][k + 1] !=
0:
                                x[i][j][k] = 0
                          elif i >= 50 and (
                                x[i][j][k] * x[i + 28][j][k] != 0
                                or x[i + 28][j][k] * x[i][j][k + 1] != 0
                          ):   # 含第二季的情况
                                x[i + 28][j][k] = 0
        return True

def arealimit(x):   # 种地面积约束
        for k in range(7):
              for i in range(0, 82):
                    block_total_area = sum(
```

```
                    x[i][j][k] for j in range(41)
                )   # 当前地块所有作物种植面积的总和
                if block_total_area > areas[i]:
                    # 记录种的第一多和第二多的植物
                    one = 0
                    two = 0
                    more = 0   # 不合格的累及面积
                    for j in range(41):
                                x[i][j][k] *= areas[i] /
block_total_area
                        if i < 34 or (i >= 54 and i < 62):   #
普通地块
                            if j > 5:
                                if x[i][j][k] > x[i][one][k]:
                                    one = j
                                    elif x[i][j][k] >
x[i][two][k]:
                                    two = j
                                if x[i][j][k] < 6:
                                    more += x[i][j][k]   #
不满足条件的多余的加起来
                                    x[i][j][k] = 0
                            else:
                                continue
                        else:   # 大棚
                            if j < 16 or j >= 19:   # 非豆类
                                if x[i][j][k] > x[i][one][k]:
                                    one = j
                                    elif x[i][j][k] >
```

```
x[i][two][k]:
                                                two = j
                                            if x[i][j][k] < 0.2:
                                                more += x[i][j][k]
                                                x[i][j][k] = 0
                            x[i][one][k] += more / 2
                            x[i][two][k] += more / 2
        return True


# %%


def select_range(x, s, j, crop_areas, k):
        if s == "平旱地单季":
                for i in range(6):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "梯田单季":
                for i in range(6, 20):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "山坡地单季":
                for i in range(20, 26):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "水浇地单季":
                for i in range(26, 34):
                        if x[i][j][k] != 0:
                                crop_areas += x[i][j][k]
        elif s == "水浇地第一季":
```

```
            for i in range(26, 34):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "水浇地第二季":
            for i in range(54, 62):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "普通大棚第一季":
            for i in range(34, 50):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第一季":
            for i in range(50, 54):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "普通大棚第二季":
            for i in range(62, 78):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第二季":
            for i in range(78, 82):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        return crop_areas

# %%
# 定义目标函数
def profit_function(x):
        x = np.reshape(x, (82, 41, 7))   # 变三维数组
```

```
total_profit = 0
# 多个约束条件
is_bean(x)
again(x)
ABC(x)
water_crop_vege(x, areas)
ordipeng(x)
smartpeng(x)
# seperate(x)
arealimit(x)
water_crop_vege(x, areas)
ordipeng(x)
smartpeng(x)
# para_his = []
for k in range(7):
      for row in range(num_rows):
            crop_areas = 0
            j = crop_ids[row]
            # 鲁棒优化
            # 预期销售量
            field_name = crop_cluster_group[row]   # 聚类种类
            price_change = 1
            # 亩产
            yield_change = np.random.uniform(0.90, 1.10) ** k
            # 成本
            cost_change = np.random.uniform(1.04, 1.06) ** k
            # 蔬菜售价
            if j >= 16 and j < 37:
                  price_change = np.random.uniform(1.04, 1.06)
```

```
** k
                # 食用菌
                if j >= 37 and j < 40:   # 菌类（除羊肚菌）
                        price_change = np.random.uniform(0.95, 0.99)
** k
                elif j == 40:   # 羊肚菌
                        price_change = 0.95
                # 改变后的成本和售价
                cost_new = crop_cost[row] * cost_change
                price_new = crop_price[row] * price_change
                if field_name == "类别1":
                        predict_sales = predict_sale(
                                cost_new, price_new, coefficients_1,
intercept_2
                        )
                        if predict_sales < 0:
                                predict_sales = 0
                elif field_name == "类别2":
                        predict_sales = predict_sale(
                                cost_new, price_new, coefficients_2,
intercept_2
                        )
                        if predict_sales < 0:
                                predict_sales = 0
                elif field_name == "类别3":
                        predict_sales = predict_sale(
                                cost_new, price_new, coefficients_2,
intercept_2
                        )
```

```
                        if predict_sales < 0:
                            predict_sales = 0
                        crop_areas = select_range(x, marks[row], j,
crop_areas, k)
                    production = crop_yield[row] * yield_change *
crop_areas  # 计算产量
                    cost = cost_new * crop_areas  # 计算种植成本
                    revenue = (
                        min(production, predict_sales) * price_new
                    )  # 计算有效收入(考虑滞销)
                    total_profit += (
                        revenue
                        - cost
                        + max(0, production - expected_demand[row]) *
crop_price[row] / 4
                    )  # 累加净收益
        return -total_profit  # 目标是最大化收益，因此返回负的净收益

# %%

# 定义种植面积的上下界
lb = [0] * (num_blocks * 41 * 7)  # 下界为0
ub = [
    area for block_area in areas for area in [block_area] * 41 * 7
]  # 上界为各地块面积
# 使用PSO算法进行优化
# 鲁棒优化，设置多个场景
pso = PSO(
    func=profit_function,
```

```
        dim=82 * 41 * 7,
        pop=50,
        max_iter=200,
        lb=lb,
        ub=ub,
        w=0.9,
        c1=2,
        c2=2,
)
pso.run()
best_pos = pso.gbest_x
best_profit = pso.gbest_y

# %%
# 处理优化结果
best_pos = [round(p, 2) for p in best_pos]  # 将结果四舍五入到小数点后两位
len(best_pos)
# 绘制全局最优目标函数值随迭代次数的变化
plt.plot(pso.gbest_y_hist, "b-", label="Best Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("PSO Optimization Convergence")
plt.legend()
plt.show()
# %%
# 打印最终的最大化的利润
final_profit = -best_profit  # 由于目标函数返回的是负的净收益，这里取相反数
```

```
print(f"2024 年到 2030 年累计的最终最大利润: {final_profit}元")
# %%
# 读取结果模板
xls = pd.ExcelFile("模板.xlsx")
best_pos_3d = np.reshape(best_pos, (82, 41, 7))
# 读 sheet
dfs = {sheet: xls.parse(sheet) for sheet in xls.sheet_names}

for idx, sheet_name in enumerate(xls.sheet_names):
    # 第三维度是表的个数
    sheet_data = best_pos_3d[:, :, idx]

    # 逐行
    for row in range(sheet_data.shape[0]):
        for col in range(sheet_data.shape[1]):
            dfs[sheet_name].iloc[row, col] = sheet_data[row, col]

# 保存
output_path = "updated_result3_2.xlsx"
with pd.ExcelWriter(output_path) as writer:
    for sheet_name, df in dfs.items():
        df.to_excel(writer, sheet_name=sheet_name, index=False)
```

附录五（Q3_ABCD_EF.py 用于求解第三问（手动二分类地块-大棚））

```
import pandas as pd
import numpy as np
from sko.PSO import PSO
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import curve_fit
def linear(file_path):
    # Step 1: 读取数据
    df = pd.read_excel(file_path, sheet_name="Sheet1")
    # Step 2: 数据预处理，选择相关的列
    X = df[["种植成本/(元/亩)", "销售单价均值/(元/斤)"]].copy()
    y = df["预期销售量/斤"]

    # Step 3: 填充缺失值（如有）
    X = X.fillna(0)
    y = y.fillna(0)
    # 转换为二次多项式特征
    poly = PolynomialFeatures(degree=4)
    X_poly = poly.fit_transform(X)

    # 拟合多项式回归模型
    poly_reg_model = LinearRegression()
    poly_reg_model.fit(X_poly, y)

    # 提取回归系数和截距
    coefficients = poly_reg_model.coef_
    intercept = poly_reg_model.intercept_
    return coefficients, intercept
```

```
# Step 1: 读取数据
file_path1 = "相关性信息_地块.xlsx"
file_path2 = "相关性信息_大棚.xlsx"

coefficients_field, intercept_field = linear(file_path1)
coefficients_peng, intercept_peng = linear(file_path2)

def predict_sale(x1, x2, coefficients, intercept):
      """
      根据多项式回归模型计算 y 值。

      参数:
      x1 -- 种植成本/(元/亩)
      x2 -- 销售单价均值/(元/斤)
      coefficients -- 多项式回归模型的系数列表
      intercept -- 多项式回归模型的截距

      返回:
      y -- 计算得到的预期销售量/斤
      """
      # 构建特征列表，包括所有可能的四次项
       features = [1, x1, x2, x1**2, x1*x2, x2**2, x1**3, x1**2*x2,
x1*x2**2, x2**3, x1**4, x1**3*x2,x1**2*x2**2, x1*x2**3, x2**4]
      # 计算 y 值
      y = intercept
      for coef, feature in zip(coefficients, features):
            y += coef * feature
```

```
        return y


MIN_AREA_PENG = 0.2
MIN_AREA = 6
# %%
# 读取数据
land_data = pd.read_excel("地块信息.xlsx")    # 读取地块信息
crop_data = pd.read_excel("作物信息.xlsx")    # 读取作物基本信息
crop_supplement = pd.read_excel("汇总补充信息.xlsx")    # 读取作物补充
信息
# %%
# 转换数据类型并清理数据
land_data["地块面积/亩"] = pd.to_numeric(
        land_data["地块面积/亩"], errors="coerce"
).fillna(0)
crop_supplement["亩产量/斤"] = pd.to_numeric(
        crop_supplement["亩产量/斤"], errors="coerce"
).fillna(0)
crop_supplement["种植成本/(元/亩)"] = pd.to_numeric(
        crop_supplement["种植成本/(元/亩)"], errors="coerce"
).fillna(0)
crop_supplement["销售单价均值/(元/斤)"] = pd.to_numeric(
        crop_supplement["销售单价均值/(元/斤)"], errors="coerce"
).fillna(0)


# %%
# 提取相关信息
areas = land_data["地块面积/亩"].tolist()    # 地块面积列表
land_names = land_data["地块名称"].tolist()    # 地块名称列表
```

```
crop_ids = crop_supplement["作物编号"].tolist()   # 作物编号列表
crop_names = crop_data["作物名称"].tolist()   # 作物名称列表
crop_yield_name = crop_supplement["地块类型"].tolist() #作物种植地块的名称
crop_yield = crop_supplement["亩产量/斤"].tolist()   # 每亩产量
crop_cost = crop_supplement["种植成本/(元/亩)"].tolist()   # 种植成本
crop_price = crop_supplement["销售单价均值/(元/斤)"].tolist()   # 销售单价
expected_demand = crop_supplement["总产量/斤"].tolist()   # 预期销售量
marks = crop_supplement["标记"].tolist()   # 标记
crop_ids = [x - 1 for x in crop_ids]
# %%
# 增加第二期地块
areas += areas[26:54]
land_names += land_names[26:54]
# %%
# 定义问题参数
num_blocks = len(areas)   # 地块数量
num_crops = len(crop_ids)   # 作物数量
num_rows = len(marks)

# %%
# def water_crop(x):   # 合法
#       for k in range(7):
#             for i in range(26, 34):
#                   for j in range(0, 41):
#                         if j == 15:
#                               continue
#                         else:
```

```
#                                 if x[i][j][k] != 0:
#                                     return False
#             for i in range(54, 62):
#                   for j in range(41):
#                         x[i][j][k] = 0
#       return True

# 约束函数

def ABC(x):   # 平旱地，梯田，山坡地
      for k in range(7):
            for block in range(0, num_blocks):
                  # ABC 约束
                  if block < 26:
                        for j in range(15, 41):
                              x[block][j][k] = 0
      return True

def water_crop_vege(x, areas):   # 水浇地约束
      flag_crop = [False] * 8
      for k in range(7):
            for i in range(26, 34):   # 第一季
                  for j in range(41):
                        if j < 15 or j >= 34:
                              x[i][j][k] = 0   # 不能种水稻和蔬菜之外
的
                        if j == 15 and x[i][j][k] > 0:
                              for j_vege in range(16, 34):
                                    x[i][j_vege][k] = 0   # 种水稻种
```

不了蔬菜

```
                        flag_crop[i - 26] = True
                elif j >= 16 and j < 34 and x[i][j][k] > 0:
                        x[i][15][k] = 0
        for cnt in range(8):   # 第二季
                i = cnt + 54
                if flag_crop[cnt]:
                        for j in range(41):
                                x[i][j][k] = 0
                else:
                        for j in range(41):
                                if j >= 34 and j < 37:
                                        continue   # 特殊蔬菜
                                else:
                                        x[i][j][k] = 0
    return True

def ordipeng(x):   # 普通大棚
    for k in range(7):
        for i in range(34, 50):   # 第一季
            for j in range(41):
                if j >= 16 and j < 34:
                    continue   # 正常蔬菜
                else:
                    x[i][j][k] = 0
        for i in range(62, 78):   # 第二季
            for j in range(41):
                if j >= 37:
                    continue   # 菌类
```

```
                        else:
                              x[i][j][k] = 0

      return True

def smartpeng(x):   # 智慧大棚
      for k in range(7):
            for i in range(50, 54):   # 第一季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
            for i in range(78, 82):   # 第二季
                  for j in range(41):
                        if j >= 16 and j < 34:
                              continue   # 正常蔬菜
                        else:
                              x[i][j][k] = 0
      return True

# def seperate(x):
#       for k in range(7):
#             for i in range(82):
#                   for j in range(41):
#                         if (i >= 34 and i < 54) or (i >= 62):   # 大
棚
#                               if x[i][j][k] > 0 and x[i][j][k] <
0.2:
```

```
#                                   x[i][j][k] = 0
#                           else:   # 普通地
#                                 if x[i][j][k] > 0 and x[i][j][k] < 6:
#                                   x[i][j][k] = 0
#       return True

def is_bean(x):   # 豆类约束，j 编号在 0-4，16-18 为豆
      for i in range(26):   # 粮食豆
            for j in range(0, 5):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              x[i][j][k] = 6
      for i in range(26, 54):   # 菜豆
            for j in range(16, 19):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              if i < 34:
                                    x[i][j][k] = 6
                              else:
                                    x[i][j][k] = 0.2
      for i in range(62, 82):
            for j in range(16, 19):
                  for k in range(0, 5):
                        if x[i][j][k] + x[i][j][k + 1] + x[i][j][k +
2] == 0:
                              x[i][j][k] = 0.2
      return True
```

```
def again(x):   # 重茬
      for i in range(54):
            for j in range(41):
                  for k in range(6):
                        if i < 26 and x[i][j][k] * x[i][j][k + 1] !=
0:
                              x[i][j][k] = 0
                        elif i >= 50 and (
                              x[i][j][k] * x[i + 28][j][k] != 0
                              or x[i + 28][j][k] * x[i][j][k + 1] != 0
                        ):   # 含第二季的情况
                              x[i + 28][j][k] = 0
      return True


def arealimit(x):   # 种地面积约束
      for k in range(7):
            for i in range(0, 82):
                  block_total_area = sum(
                        x[i][j][k] for j in range(41)
                  )   # 当前地块所有作物种植面积的总和
                  if block_total_area > areas[i]:
                        # 记录种的第一多和第二多的植物
                        one = 0
                        two = 0
                        more = 0   # 不合格的累及面积
                        for j in range(41):
                                    x[i][j][k]  *=  areas[i]  /
block_total_area
```

```
                    if i < 34 or (i >= 54 and i < 62):   #
普通地块
                        if j > 5:
                            if x[i][j][k] > x[i][one][k]:
                                one = j
                                elif  x[i][j][k]  >
x[i][two][k]:
                                two = j
                            if x[i][j][k] < 6:
                                more += x[i][j][k]   #
不满足条件的多余的加起来
                                x[i][j][k] = 0
                        else:
                            continue
                    else:   # 大棚
                        if j < 16 or j >= 19:   # 非豆类
                            if x[i][j][k] > x[i][one][k]:
                                one = j
                                elif  x[i][j][k]  >
x[i][two][k]:
                                two = j
                            if x[i][j][k] < 0.2:
                                more += x[i][j][k]
                                x[i][j][k] = 0
                    x[i][one][k] += more / 2
                    x[i][two][k] += more / 2
        return True

# %%
```

```
def select_range(x, s, j, crop_areas, k):
    if s == "平旱地单季":
        for i in range(6):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "梯田单季":
        for i in range(6, 20):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "山坡地单季":
        for i in range(20, 26):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地单季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第一季":
        for i in range(26, 34):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "水浇地第二季":
        for i in range(54, 62):
            if x[i][j][k] != 0:
                crop_areas += x[i][j][k]
    elif s == "普通大棚第一季":
        for i in range(34, 50):
            if x[i][j][k] != 0:
```

```
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第一季":
            for i in range(50, 54):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "普通大棚第二季":
            for i in range(62, 78):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        elif s == "智慧大棚第二季":
            for i in range(78, 82):
                if x[i][j][k] != 0:
                    crop_areas += x[i][j][k]
        return crop_areas


# %%
# 定义目标函数
def profit_function(x):
        x = np.reshape(x, (82, 41, 7))   # 变三维数组
        total_profit = 0
        # 多个约束条件
        is_bean(x)
        again(x)
        ABC(x)
        water_crop_vege(x, areas)
        ordipeng(x)
        smartpeng(x)
        # seperate(x)
        arealimit(x)
```

```
        water_crop_vege(x, areas)
        ordipeng(x)
        smartpeng(x)
        for k in range(7):
              for row in range(num_rows):
                    crop_areas = 0
                    j = crop_ids[row]
                    # 鲁棒优化
                    # 预期销售量
                    # flag 标记是地块还是大棚, 大鹏是真，地块是假
                    field_name = crop_yield_name[row] #地块名字
                    flag = "大棚" in field_name
                    predict_scale = 1
                    price_change = 1
                    # 亩产
                    yield_change = np.random.uniform(0.90, 1.10) ** k
                    # 成本
                    cost_change = np.random.uniform(1.04, 1.06) ** k
                    # 蔬菜售价
                    if j >= 16 and j < 37:
                          price_change = np.random.uniform(1.04, 1.06)
** k
                    # 食用菌
                    if j >= 37 and j < 40:   # 菌类（除羊肚菌）
                          price_change = np.random.uniform(0.95, 0.99)
** k
                    elif j == 40:   # 羊肚菌
                          price_change = 0.95
                    #记录参数
```

```
                cost_new = crop_cost[row] * cost_change
                price_new = crop_price[row] * price_change
                if flag: #大棚
                            predict_scale = predict_sale(cost_new,
price_new, coefficients_peng, intercept_peng)
                      if predict_scale < 0: predict_scale = 0
                       elif predict_scale > 20000: predict_scale =
18000
                else:
                            predict_scale = predict_sale(cost_new,
price_new, coefficients_field, intercept_field)
                      if predict_scale < 0: predict_scale = 0
                       elif predict_scale > 100000: predict_scale =
95000
                     crop_areas = select_range(x, marks[row], j,
crop_areas, k)
                    production = crop_yield[row] * yield_change*
crop_areas # 计算产量
                cost = crop_cost[row] * cost_change * crop_areas# 计
算种植成本
                revenue = (
                                min(production, predict_scale) *
crop_price[row] * price_change
                )  # 计算有效收入(考虑滞销)
                total_profit += (
                      revenue
                      - cost
                       + max(0, production - expected_demand[row]) *
crop_price[row] / 4
```

```
                    )    # 累加净收益
        return -total_profit    # 目标是最大化收益，因此返回负的净收益

# %%

# 定义种植面积的上下界
lb = [0] * (num_blocks * 41 * 7)    # 下界为0
ub = [
        area for block_area in areas for area in [block_area] * 41 * 7
]    # 上界为各地块面积
# 使用PSO算法进行优化
# 鲁棒优化，设置多个场景
pso = PSO(
        func=profit_function,
        dim=82 * 41 * 7,
        pop=50,
        max_iter=200,
        lb=lb,
        ub=ub,
        w=0.9,
        c1=2,
        c2=2,
)
pso.run()
best_pos = pso.gbest_x
best_profit = pso.gbest_y

# %%
# 处理优化结果
```

```
best_pos = [round(p, 2) for p in best_pos]   # 将结果四舍五入到小数点后两位
len(best_pos)
# 绘制全局最优目标函数值随迭代次数的变化
plt.plot(pso.gbest_y_hist, "b-", label="Best Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("PSO Optimization Convergence")
plt.legend()
plt.show()
# %%
# 打印最终的最大化的利润
final_profit = -best_profit   # 由于目标函数返回的是负的净收益，这里取相反数
print(f"2024 年到 2030 年累计的最终最大利润: {final_profit}元")
# %%
# 读取结果模板
xls = pd.ExcelFile("result1_1.xlsx")
best_pos_3d = np.reshape(best_pos, (82, 41, 7))
# 读 sheet
dfs = {sheet: xls.parse(sheet) for sheet in xls.sheet_names}

for idx, sheet_name in enumerate(xls.sheet_names):
    # 第三维度是表的个数
    sheet_data = best_pos_3d[:, :, idx]

    # 逐行
    for row in range(sheet_data.shape[0]):
        for col in range(sheet_data.shape[1]):
```

```
                dfs[sheet_name].iloc[row, col] = sheet_data[row, col]

# 保存
output_path = "updated_result3.xlsx"
with pd.ExcelWriter(output_path) as writer:
      for sheet_name, df in dfs.items():
            df.to_excel(writer, sheet_name=sheet_name, index=False)
```

附录六（pearson.py 用于问题三判断能否使用皮尔森并输出皮尔森系数）

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro, pearsonr, kstest

file_path_new = "汇总补充信息.xlsx"

# Load the Excel file
data_new = pd.read_excel(file_path_new)
# 选择相关列
columns_of_interest = ["总产量/斤", "销售单价均值/(元/斤)", "种植成本/(元/亩)"]
data_filtered = data_new[columns_of_interest].dropna()

# 绘制散点图，查看线性关系
sns.pairplot(data_filtered)
plt.show()
```

```
# 检查正态性 - 使用 Shapiro-Wilk 检验和 K-S 检验
for column in columns_of_interest:
    stat_shapiro, p_shapiro = shapiro(data_filtered[column])
    print(f"{column}的 Shapiro-Wilk P 值: {p_shapiro}")
    if p_shapiro > 0.05:
        print(f"{column} 满足正态分布（Shapiro-Wilk）")
    else:
        print(f"{column} 不满足正态分布（Shapiro-Wilk）")

    # K-S 检验假设数据是正态分布
    stat_ks, p_ks = kstest(data_filtered[column], 'norm',
args=(data_filtered[column].mean(), data_filtered[column].std()))
    print(f"{column}的 K-S P 值: {p_ks}")
    if p_ks > 0.05:
        print(f"{column} 满足正态分布（K-S）")
    else:
        print(f"{column} 不满足正态分布（K-S）")

# 计算皮尔森相关系数
for col1 in columns_of_interest:
    for col2 in columns_of_interest:
        if col1 != col2:
            corr, p_value = pearsonr(data_filtered[col1],
data_filtered[col2])
            print(f"{col1} 和 {col2} 的皮尔森相关系数: {corr}, P 值:
{p_value}")
```

附录 7（nonlinear_regression.py 用于问题三的预期销售量非线性回归）

```
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import curve_fit

# Step 1: 读取数据
file_path = "相关性信息_地块.xlsx"
df = pd.read_excel(file_path, sheet_name="Sheet1")

# Step 2: 数据预处理，选择相关的列
X = df[["种植成本/(元/亩)", "销售单价均值/(元/斤)"]].copy()
y = df["预期销售量/斤"]

# Step 3: 填充缺失值（如有）
X = X.fillna(0)
y = y.fillna(0)

# 转换为二次多项式特征
poly = PolynomialFeatures(degree=4)
X_poly = poly.fit_transform(X)

# 拟合多项式回归模型
poly_reg_model = LinearRegression()
poly_reg_model.fit(X_poly, y)
```

```
print("回归系数:", poly_reg_model.coef_)  # 输出系数
print("截距:", poly_reg_model.intercept_)  # 输出截距

# 预测值和R^2
y_poly_pred = poly_reg_model.predict(X_poly)
r2_poly = r2_score(y, y_poly_pred)
print(f"多项式回归的 R² 值: {r2_poly}")



# def nonlinear_model(X, a, b, c, d):
#     # X[0] 是种植成本，X[1] 是销售单价均值
#     return a * np.exp(b * X[0] + c * X[1]) + d



# # Step 4: 拟合非线性回归模型
# # 提取 X1 (种植成本) 和 X2 (销售单价均值)
# X_data = [X["种植成本/(元/亩)"], X["销售单价均值/(元/斤)"]]

# # 初始猜测的参数值
# initial_guess = [1, 0.01, 0.01, 0.01]

# # 使用 curve_fit 拟合模型
# params, covariance = curve_fit(nonlinear_model, X_data, y,
p0=initial_guess)

# # Step 5: 获取拟合参数 a, b, c
# a, b, c, d = params
# print(f"拟合的参数: a = {a}, b = {b}, c = {c},d={d}")
```

```
# y_pred = nonlinear_model(X_data, a, b, c, d)
# r2 = r2_score(y, y_pred)
# print(f"非线性回归的 R² 值: {r2}")
```

附录 8（Linear_Regression.py 用于问题三的预期销售量线性回归）

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures

# Step 1: 读取数据
file_path = "相关性信息_类别 1.xlsx"  # 替换为你的文件路径
df = pd.read_excel(file_path, sheet_name="Sheet1")

# Step 2: 数据预处理，选择相关的列
X = df[["种植成本/(元/亩)", "销售单价均值/(元/斤)"]].copy()
y = df["预期销售量/斤"]

# Step 3: 填充缺失值（如有）
X = X.fillna(0)
y = y.fillna(0)

# Step 4: 线性回归模型拟合
linear_reg = LinearRegression()
linear_reg.fit(X, y)
```

```
print("回归系数:", linear_reg.coef_)   # 输出系数
print("截距:", linear_reg.intercept_)   # 输出截距
# Step 5: 计算 R^2 值
y_pred = linear_reg.predict(X)
r2 = r2_score(y, y_pred)
print(f"R² 值: {r2}")

# Step 6: 计算 VIF（方差膨胀因子）
X_with_constant = sm.add_constant(X)   # 为 statsmodels 添加常数项
vif_data = pd.DataFrame()
vif_data["Feature"] = X_with_constant.columns
vif_data["VIF"] = [
      variance_inflation_factor(X_with_constant.values, i)
      for i in range(X_with_constant.shape[1])
]
print("VIF 数据:\n", vif_data)
```