

基于提升华北山区农作物种植方案的经济效益的优化策略

摘要

农业是我国经济的重要组成部分，尤其在华北山区，气候条件和土地资源的多样性给农作物种植带来了挑战。本文基于实际情况和所给数据，设计了一套为期七年的最优农作物种植策略，以提高农作物的生产效率和经济效益。研究重点考虑了不同地块类型，以及作物轮作等种植规律的要求，构建了**混合整数线性规划模型**和**鲁棒优化模型**来应对未来的不确定性因素，如销售量、亩产量、种植成本和销售价格的波动。

在问题一中，我们以**利润最大化**为目标，针对农作物是否预期销售量的不同情景，分别建立了两个混合整数线性规划模型。第一个模型假设超额部分无法销售，因而造成浪费；在此情景下，利润的优化需要聚焦于产量和预期销售量的关系，避免农作物过剩。第二个模型则假设超额部分以 2023 年市场价的 50% 降价出售。最终结果为：情景一的最低利润为 5010903，最高为 5434751，波动幅度约为 8.5%。而情景二的最低利润为 5741635，最高为 6224332，波动幅度接近 8.4%。并且由于豆类种植条件，利润呈现**周期为 3 的波动性**。

在问题二中，则进一步考虑了未来多种**不确定因素**，包括小麦和玉米的年增长率、其他作物的波动性、种植成本的增长趋势以及农产品销售价格的变化等。我们通过构建**鲁棒优化模型**，针对未来可能的市场波动，优化了种植策略，使其具备更强的适应性和抗风险能力。模型引入了动态参数，确保在面对不同的不确定性情景时，依然能够保证较为稳定的收益。通过对多种不确定因素的模拟和优化，该模型展示了如何在实际生产中进行合理规划，以应对复杂的市场环境。

在问题三中，我们利用 **Spearman 相关系数** 分析方法，探究了种植成本、销售价格与预期销售量之间的相互关系。接着，通过**聚类分析**的方法，我们识别了不同作物种植模式的关键特征，包括平均售价、销售量、种植成本和种植地块分布等。进一步地，我们分析了作物间的可替代性和互补性。最后，我们对模型进行了修改，引入了**可替代性和互补性约束**，并更新了目标函数。通过模拟数据，我们得出了更贴近实际的解，并通过比较问题二和问题三的结果，评估了不同策略对乡村农业生产和经济效益的影响。

研究结论显示，优化的种植策略在稳定的利润范围内波动，具有较强的适应性，能够平衡经济效益与市场不确定性，为农业生产提供了重要的参考依据。

关键字： 混合整数线性规划 种植策略 聚类分析 **Spearman 相关系数** 鲁棒优化模型

一、问题重述

1.1 问题背景

在华北山区中，气候条件限制每年的耕地种植季节。乡村包含多种类型的耕地（平旱地、梯田、山坡地和水浇地）以及大棚（普通大棚和智慧大棚），每种地块和大棚都有其适宜种植的作物类型。为了优化有限的土地资源，提高种植效益，同时考虑各种作物的轮作与种植规律，需要为该乡村制定未来 7 年的农作物种植策略。

1.2 问题提出

问题一：在未来的 7 年（2024-2030 年），根据以下两种情况分别给出乡村的最优种植方案：

如果某种作物的总产量超过了相应的预期销售量，超过部分不能正常销售，造成浪费。如果某种作物的总产量超过了相应的预期销售量，超过部分按 2023 年销售价格的 50% 降价出售。

问题二：在考虑未来的不确定因素（如销售量、亩产量、种植成本和销售价格等）下，给出 2024-2030 年的最优种植方案。需要考虑粮食类作物、小麦和玉米的增长趋势、其他农作物的波动性以及成本和价格变化等因素。

问题三：考虑不同作物之间的可替代性和互补性，以及销售量、销售价格和种植成本之间的相关性。在问题 2 的基础上，综合考虑这些因素，给出 2024-2030 年的最优种植策略，并与问题 2 的结果进行比较分析。

二、问题分析

问题一：要求我们基于给定的农作物销售量、种植成本、亩产量和销售价格的稳定假设，为 2024 至 2030 年制定最优的种植方案。这一问题进一步细分为两种情况：一是超过预期销售量的农作物将造成浪费；二是超过部分可以以半价出售。这要求我们考虑如何平衡产量与市场需求，以最大化经济效益。可以找到目标函数，依据题目要求的设置约束条件，建立对目标函数值的优化模型，找到相对最优情况下种植面积作为结果。

问题二：在考虑不确定性因素下，找到最大化预期收益的种植策略。要求我们构建一个动态的数学模型，以应对农作物市场所固有的不确定性。该模型将特别关注小麦与玉米预期销售量的增长趋势，同时考虑到其他作物销售量、亩产量以及种植成本的潜在波动性。此外，模型还将纳入蔬菜类作物销售价格的上升趋势与食用菌价格的下降趋

势。通过整合这些变量，本研究旨在发展出一种策略，以优化种植决策，实现经济效益的最大化，并确保策略的适应性和鲁棒性。

问题三：考虑作物之间的可替代性和互补性，以及经济变量之间的相关性，优化种植策略。在此基础上进一步深化，要求本研究在问题二的模型基础上，综合考量农作物间的可替代性与互补性，以及销售量、销售价格与种植成本之间的动态关联。这不仅要求模型能够处理市场的不确定性，还必须能够识别并利用作物市场间的复杂相互作用。

三、模型的假设

(1) 假设水浇地不种植水稻。从图表1中可以得知，水稻的种植收益很低，故我们不考虑种植水稻以简化模型。

(2) 假设在任何一种农作物在每一季种植的地块数都不能超过 α

四、符号说明

表 1 符号说明

符号	说明
γ_i	年份
L_i	地块
S_i	季度
E_i	C_i 预期销售量 (斤)
C_i	农作物
K_{ijm}	农作物 C_i 在 L_j 地块第 S_m 季成本 (元/亩)
P_i	农作物 C_i 的价格 (元/斤)
Y_{ijm}	农作物 C_i 在地块 L_j 第 S_m 季的产量 (斤/亩)
X_{ijmn}	农作物 C_i 在地块 L_j 在 Y_n 年 S_m 季度的种植面积 (亩)
f_i	地块 L_i 的面积 (亩)

五、数据处理和可视化分析

5.1 数据可视化

为了更好的观察各农作物的种植收益情况，我们将 2023 年统计的数据进行可视化。这样我们能对各个土地类型中收益较低的农作物进行初步了解。

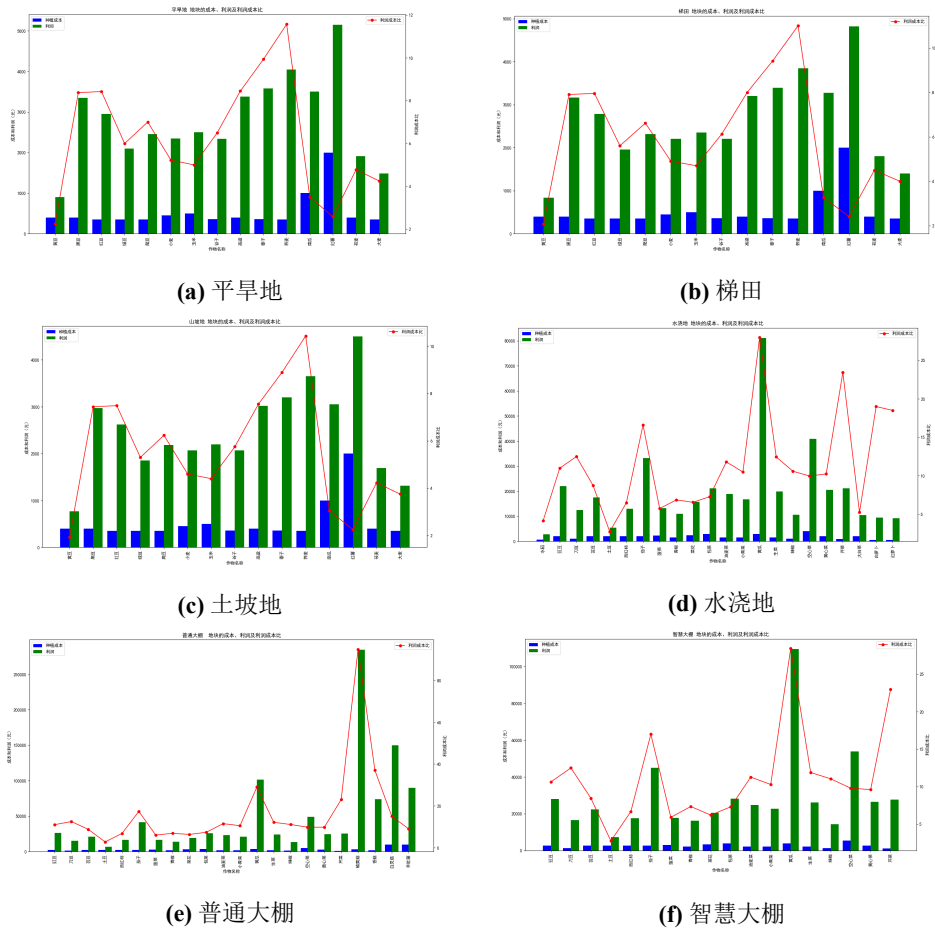


图 1 不同土地类型农作物利润图

5.2 可视化分析

可以由图1a，图1b，图1c可知，这三种土地类型中的农作物中相对收益情况大致相同，联系附件一中可以知道，农作物在这三种类型的土地中的成本，亩产量，单价基本相同。由图1d可以得知，水稻的相对利润是比较低的，在水浇地土地类型中利润仅高于土豆。考虑到水稻只能种植在水浇地上和单季作物的要求，我们做出合理的假设：在以利润为目标的前提下不考虑水稻。由图1e和图1f可以得知，农作物的在大棚的收益整体是大于平旱地等，但是现实还要考虑大棚面积远远小于前者。而且大棚内有较少的农作物有显著的高收益。例如普通大棚的榆黄菇和智慧大棚的黄瓜。

六、 问题一模型建立与求解

6.1 问题一的求解思路

问题一要求在 2024 至 2030 年期间，针对乡村的农作物种植情况，根据给定的农作物销售量、种植成本、亩产量和销售价格的稳定假设，制定最优的种植方案。具体分为两种情况：

超过预期销售量的农作物将造成浪费：在这种情况下，假设超过的部分不能正常销售，因此只计算不超过预期销售量的部分作为销售量。目标函数为利润最大化，考虑到每种农作物的总产量不超过其预期销售量。

超过部分可以以半价出售：在这种情况下，假设超出预期销售量的部分可以以 2023 年销售价格的 50% 降价出售。目标函数同样为利润最大化，但这里计算的是所有产量（包括超过部分），其中超过预期销售量的部分按降价 50 % 处理。

6.2 问题一模型的建立

6.2.1 计算 2023 各农作物的产量

2023 年各农作物产量为该农作物总的种植面积 * 农作物的亩产量。其中表2为产量高的部分农作物数据。

表 2 2023 年产量

农作物名称	2023 年产量/斤
小麦	170840
大白菜	150000
玉米	132750
白萝卜	100000
谷子	71400
黄豆	57000
茄子	45360
豇豆	36480

6.2.2 模型假设

为实现混合整数线性规划，我们假定预期销售量为当年总产量的 80%，这将体现在我们的目标函数中。为确保各种作物产量的稳定性，我们规定每种作物的年产量不能高于 2023 年该种作物产量的 1.2 倍，这将在我们的约束条件中体现。

我们以利润为目标，即在合理的种植作物的情况下最大化利润。

6.2.3 模型初始状态

模型以 2023 年个各农作物种植面积，产量等属性的情况为初始状态，这将在我们的约束条件中体现。

6.2.4 建立目标函数

首先目标函数以利润最大化的标准来建立。针对两种不同的情况，我们可以建立相应的目标函数：

第一种情况（滞销浪费）：

$$Max\left\{\sum_{y=2024}^{2030}\sum_{c=1}^{41}0.8\cdot\sum_{l=1}^{54}(Y_{cls}\cdot X_{clsy})\cdot P_c-\sum_{s=1}^2\sum_{l=1}^{54}K_{cls}\cdot X_{clsy}\right\} \quad (1)$$

第二种情况（超额部分降价出售）：

$$Max\left\{\sum_{y=2024}^{2030}\sum_{c=1}^{41}0.8\cdot\sum_{l=1}^{54}(Y_{cls}\cdot X_{clsy})\cdot P_c+0.2\cdot\sum_{l=1}^{54}(Y_{cls}\cdot X_{clsy})\cdot P_c\cdot 0.5-\sum_{s=1}^2\sum_{l=1}^{54}K_{cls}\cdot X_{clsy}\right\} \quad (2)$$

2式可化简为：

$$Max\left\{\sum_{y=2024}^{2030}\sum_{c=1}^{41}0.9\cdot\sum_{l=1}^{54}(Y_{cls}\cdot X_{clsy})\cdot P_c-\sum_{s=1}^2\sum_{l=1}^{54}K_{cls}\cdot X_{clsy}\right\} \quad (3)$$

其中： Y_{cls} 表示农作物 C_c 在地块 L_l 的第 S_s 季的产量（其他变量下标同理）， X_{clsy} 表示种植面积， P_c 表示农作物售卖的价格， K_{cls} 表示农作物单位面积的成本。

在情况一条件下：直接取当年总产量的 0.8 倍作为当年预期销售量参与计算。在情况二条件下：如果超出了预期销售量，将产量超出的部分进行 50% 的降价出售。

6.2.5 设置约束条件

（1）分散性约束：分散性约束定义为一种农作物不能在过多的地块进行种植。假定一种农作物种植地块总和的上限为 6；为了更好表达约束条件：

Step1:) 我们定义一个二进制变量 B_{clsy} 来表示某个地块在某个季节是否种植了某种作物。值为 1 则表示种植了。

$$B_{clsy} = \begin{cases} 0 & x_{clsy} = 0, \\ 1 & x_{clsy} > 0. \end{cases}$$

Step2:) 使用大 M 和 ϵ 法将该条件不等式化:

$$x_{clsy} \leq M \cdot B_{clsy}$$

$$x_{clsy} \geq \epsilon \cdot B_{clsy}$$

其中 M 是一个大常数, ϵ 是一个很小的正数 (0.001) 用于防止种植面积为 0。

Step3:) 最后约束表示为

$$\forall c \in C \forall y \in Y \forall s \in S, \sum_{l=1}^{54} B_{clsy} \leq 6 \quad (4)$$

(2) 面积约束:

$$\forall y \forall s \forall l, \sum_{c=1}^{41} X_{clsy} \leq f_l \quad (5)$$

其中: X_{clsy} 表示农作物的种植面积, f_l 表示 L_l 地块的面积。由附件 1 可知该不等式表示的含义是: 在确定的时间段种植在地块 L_l 上所有的农作物的种植总面积不能超过该地块的面积。我们人为规定 X_{clsy} 的最小值为 0.3 (即为大棚面积的一半)

$$\forall y \forall s \forall l, \sum_{c=1}^{41} X_{clsy} \cdot B_{clsy} \geq 0.3 \quad (6)$$

由于水稻种植收益很低, 故不考虑水稻,

$$\forall y \forall s \forall l, X_{16lsy} = 0 \quad (7)$$

该式表示不种植水稻;

$$\forall y \forall s \forall y \forall l, X_{clsy} \geq 0 \quad (8)$$

该式表示种植面积为非负数。

(3) 轮作约束:

对于平旱地, 梯田, 山坡地:

$$\forall y \in \{2023, \dots, 2029\} \forall l \in \{1, \dots, 26\} \forall c \in \{1, \dots, 15\} X_{cl1y} + X_{cl1y+1} \leq f_l \quad (9)$$

其中 2023 年的数据是该年对应各指标的实际数据;

对于智慧大棚类：

$$\forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl1y} + X_{cl2y+1} \leq f_l \quad (10)$$

$$\forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl2y} + X_{cl1y+1} \leq f_l \quad (11)$$

$$\forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl1y} + X_{cl1y+1} \leq f_l \quad (12)$$

由于我们不考虑种植水稻，在水浇地和普通大棚里，由于每一季种植的农作物不相同，所以不需要考虑连续重茬种植，该部分式子表示的含义在平旱地，梯田，山坡地，智慧大棚里的不能连续种植相同农作物。

(4) 豆类种植约束：

$$\forall y \in \{2023, \dots, 2028\} \forall c \in \{1, \dots, 5, 17, 18, 19\} \sum_{s=1}^2 \sum_{k=y}^{y+2} X_{cls k} \geq f_l \quad (13)$$

该式子表示的含义就是在任意的地块上在连续的三年内必须种植豆类农作物。我们也考虑了 2023 年的初始情况。

(5) 种植类别约束：

对于平旱地，梯田，山坡：

$$\forall c \notin \{1, \dots, 16\} \forall l \in \{1, \dots, 26\} \forall s \forall y, X_{cls y} = 0 \quad (14)$$

$$\forall c \in \{1, \dots, 15\} \forall l \in \{1, \dots, 26\} \forall y, X_{cl2y} = 0 \quad (15)$$

对水浇地类：

$$\forall c \notin \{17, \dots, 34\} \forall l \in \{27, \dots, 34\} \forall y, X_{cl1y} = 0 \quad (16)$$

$$\forall c \notin \{35, \dots, 37\} \forall l \in \{27, \dots, 34\} \forall y, X_{cl2y} = 0 \quad (17)$$

对于普通大棚类：

$$\forall c \notin \{17, \dots, 34\} \forall l \in \{35, \dots, 50\} \forall y, X_{cl1y} = 0 \quad (18)$$

$$\forall c \notin \{38, \dots, 41\} \forall l \in \{35, \dots, 50\} \forall y, X_{cl2y} = 0 \quad (19)$$

对于智慧大棚类：

$$\forall c \notin \{17, \dots, 34\} \forall l \in \{51, \dots, 54\} \forall y \forall s, X_{cls y} = 0 \quad (20)$$

以上等式表示在相应的地块在相应的季节只能种植相应种类的农作物，可以依据附件 1 得知。对不符合要求下的种植面积设置为 0。

6.2.6 规划模型汇总

对于情景一：

$$Max\{ \sum_{y=2024}^{2030} \sum_{c=1}^{41} 0.8 \cdot \sum_{l=1}^{54} (Y_{cls} \cdot X_{clsy}) \cdot P_c - \sum_{s=1}^2 \sum_{l=1}^{54} K_{cls} \cdot X_{clsy} \}$$

对于情景二：

$$Max\{ \sum_{y=2024}^{2030} \sum_{c=1}^{41} 0.9 \cdot \sum_{l=1}^{54} (Y_{cls} \cdot X_{clsy}) \cdot P_c - \sum_{s=1}^2 \sum_{l=1}^{54} K_{cls} \cdot X_{clsy} \}$$

$$\text{s.t.} \left\{ \begin{array}{l} \forall c \in C \forall y \in Y \forall s \in S, \sum_{l=1}^{54} B_{clsy} \leq \alpha \\ x_{clsy} \leq M \cdot B_{clsy} \\ x_{clsy} \geq \epsilon \cdot B_{clsy} \\ \forall y \forall s \forall l, \sum_{c=1}^{41} X_{clsy} \leq f_l \\ \forall y \forall s \forall l, \sum_{c=1}^{41} X_{clsy} \cdot B_{clsy} \geq 0.3 \\ \forall y \forall s \forall l \forall y, X_{clsy} \geq 0 \\ \forall y \forall s \forall l, X_{16lsy} = 0 \forall y \in \{2023, \dots, 2029\} \forall l \in \{1, \dots, 26\} \forall c \in \{1, \dots, 15\} X_{cl1y} + X_{cl1y+1} \leq f_l \\ \forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl1y} + X_{cl2y+1} \leq f_l \\ \forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl2y} + X_{cl1y+1} \leq f_l \\ \forall y \in \{2023, \dots, 2029\} \forall l \in \{51, \dots, 54\} \forall c \in \{17, \dots, 34\} X_{cl1y} + X_{cl1y+1} \leq f_l \\ \forall y \in \{2023, \dots, 2028\} \forall c \in \{1, \dots, 5, 17, 18, 19\} \sum_{s=1}^2 \sum_{k=y}^{y+2} X_{clsk} \geq f_l \\ \forall c \in \{1, \dots, 5, 17, 18, 19\} \sum_{s=1}^2 \sum_{k=y}^{y+2} X_{clsk} > 0 \\ \forall c \in C \setminus \{1, \dots, 15\} \forall l \in \{1, \dots, 26\} \forall s \forall y, X_{clsy} = 0 \\ \forall c \in \{1, \dots, 15\} \forall l \in \{1, \dots, 26\} \forall y, X_{cl2y} = 0 \\ \forall c \in C \setminus \{17, \dots, 34\} \forall l \in \{27, \dots, 34\} \forall y, X_{cl1y} = 0 \\ \forall c \in C \setminus \{35, \dots, 37\} \forall l \in \{27, \dots, 34\} \forall y, X_{cl2y} = 0 \\ \forall c \in C \setminus \{17, \dots, 34\} \forall l \in \{35, \dots, 50\} \forall y, X_{cl1y} = 0 \\ \forall c \in C \setminus \{38, \dots, 41\} \forall l \in \{35, \dots, 50\} \forall y, X_{cl2y} = 0 \\ \forall c \in C \setminus \{17, \dots, 34\} \forall l \in \{51, \dots, 54\} \forall y \forall s, X_{clsy} = 0 \\ C = \{1, \dots, 41\} \\ L = \{1, \dots, 54\} \\ S = \{1, 2\} \\ Y = \{2024, \dots, 2030\} \\ B_{clsy} = \{0, 1\} \\ M = 10000 \\ \epsilon = 0.001 \end{array} \right. \quad (21)$$

6.3 问题一模型的求解与分析

6.3.1 求解方法

调用 pulp 库中的 `model.solve()` 函数进行模型求解，应用 CBC 开源线性规划和整数规划求解器，该求解器工作流程如下：

预处理：预处理的目的是简化问题，减少计算量，其优化操作包括：

- 去除冗余的约束。
- 消除不必要的变量。
- 简化线性表达式。

分支定界法：求解器会应用分支定界法（Branch and Bound）来搜索可能的最优整数解。具体步骤如下：

- 分支：将松弛解中的非整数变量分裂为两个子问题，一个分支向上取整，另一个分支向下取整。
- 定界：求解每个子问题，计算其上下界。如果某个子问题的最优解优于当前解，则继续对其进行分支；否则，剪枝（即不再继续搜索该分支）。

分支定界法通过构建一个搜索树，不断缩小解的搜索空间，直到找到全局最优解
最优性和可行性检查：

在每一步迭代中，求解器都会检查当前解是否满足最优性和可行性条件。如果满足最优性条件，则终止算法并返回结果；如果不满足，则继续迭代。

6.3.2 模型结果

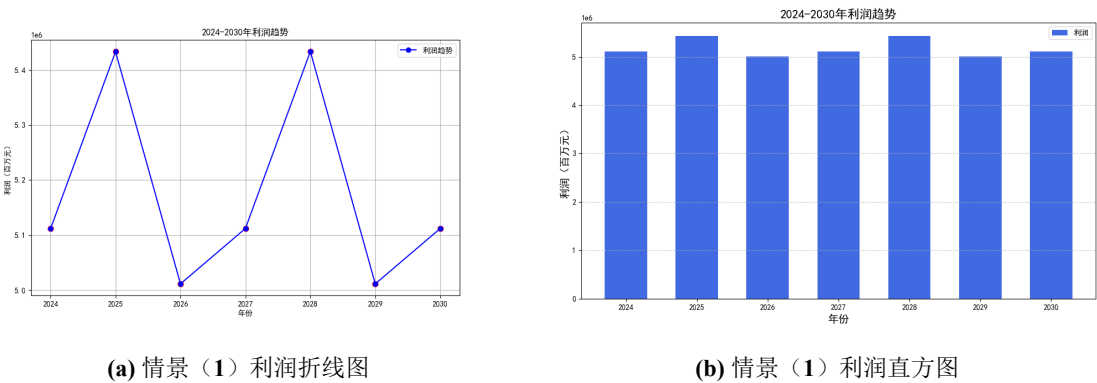


图 2 情景（1）的利润结果

6.3.3 结果分析

1. 数据大小

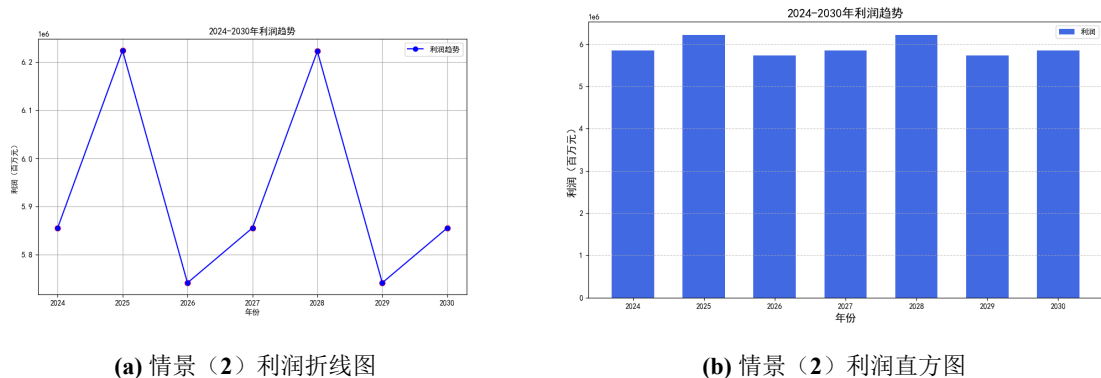


图 3 情景（2）的利润结果

表 3 不同情景下的每年的利润（元）

	2024	2025	2026	2027	2028	2029	2030
情景 1	5111690	5434751	5010903	5111889	5433176	5011042	5111775
情景 2	5855504	6224332	5741635	5855703	6222613	5741774	5855589

从两组数据的利润值来看，情景二的利润值整体上高于情景一。情景一的利润主要维持在 500 万到 540 万之间，而情景二的利润则普遍高于 570 万，最高达到了 6224332。这表明在情景二中，虽然部分作物的销售价格下降，但总体利润仍然较高，反映出降价销售超额部分的策略能够有效减少损失，提升整体经济效益。

2. 周期性

从利润的波动趋势来看，情景一和情景二在七年中均表现出一定的周期性。两者的利润峰值均出现在第二年和第五年，表明在这些年份中，农作物的市场表现较好，销售情况优于其他年份。两情景的周期时长均为三年左右，这与每个地块三年内必须种植豆类作物的约束条件有关。

3. 波动性

情景一的最低利润为 5010903，最高为 5434751，波动幅度约为 8.5%。而情景二的最低利润为 5741635，最高为 6224332，波动幅度接近 8.4%。两者波动幅度相似，这也与每个地块三年内必须种植豆类作物的约束条件有关。

4. 稳定性

两情景在七年的时间窗格中利润总体趋势保持稳定，其利润均保持在一定范围内，没有明显的上升或下降，由此体现模型的合理性。

七、问题二模型建立与求解

假设对于超出预期销售量的部分的处理沿用情况二。问题二要求在考虑未来各种不确定因素的情况下，制定 2024-2030 年的最优种植方案。这些不确定性因素包括：

- 小麦和玉米的预期销售量有 5% 至 10% 的年增长率。
- 其他农作物的预期销售量每年变化 $\pm 5\%$ 。
- 农作物的亩产量受气候等因素的影响，每年变化 $\pm 10\%$ 。
- 种植成本平均每年增长 5%。
- 粮食类作物的销售价格基本稳定；蔬菜类作物的销售价格年增长 5% 左右；食用菌的销售价格下降 1% 至 5%。

为了应对这些不确定性，需要建立一个鲁棒优化模型，最大化预期收益，同时最小化因不确定性带来的风险。

7.1 鲁棒优化模型建立

X_{clsy} 仍然作为决策变量，目标函数增加不确定性部分，约束条件沿用问题一建立的模型。

7.1.1 不确定性建模

鲁棒优化的关键在于对不确定参数的建模。针对题目描述的不确定因素，我们可以设定以下不确定参数的范围：

- 预期销售量 E_c^j ：

对于小麦和玉米，设增长率 g_c^j 介于 5% 到 10% 之间。则：

$$E_c^j = E_c^{2023} \times (1 + g_c^j), \quad g_c^j \in [0.05, 0.10]$$

对于其他作物，设变化率 g_c^j 在 $\pm 5\%$ 之间。则：

$$E_c^j = E_c^{2023} \times (1 + g_c^j), \quad g_c^j \in [-0.05, 0.05]$$

- 亩产量 Y_{cls}^j ：

$$Y_{cls}^j = Y_{cls}^{j-1} \times (1 + u_c^j), \quad u_c^j \in [-0.10, 0.10]$$

- 种植成本 K_{cls}^j ：

$$K_{cls}^j = K_{cls}^{2023} \times 1.05^{j-2023}$$

- 销售价格 P_c^j ：

$$P_c^j = P_c^{2023} \times (1 + p_c^j), \quad p_c^j = 0.05 \text{ (蔬菜类)}$$

$$P_c^j = P_c^{2023} \times (1 - q_c^j), \quad q_c^j \in [0.01, 0.05] \text{ (食用菌类)}$$

7.1.2 鲁棒优化模型的目标函数

鲁棒优化的目标是最大化最小收益。假设在所有的不确定性下，我们希望总收益 Z 达到最大最小值：

$$\text{Maximize } \min_{S, Y, P, K} Z$$

其中，总收益 Z 的表达式为：

$$Z = \sum_{j=2024}^{2030} \sum_{s=1}^2 \left(\sum_{l \in L} \sum_{c \in C} 0.9 \cdot (P_c^j \cdot Y_{cls}^j \cdot X_{clsj} - K_{cls}^j \cdot X_{clsj}) \right)$$

7.2 鲁棒优化求解方法

切平面法是解决鲁棒优化问题的一种有效技术。该方法的核心策略在于系统地构建不确定性情境，并利用一系列线性规划问题的解决方案以及相应“切平面”的引入，逐步逼近问题的最优解。在每一迭代步骤中，切平面的作用是排除当前的非可行或次优解空间，从而引导搜索过程更接近全局最优解。

7.2.1 初始化

Step1) 定义初始场景集合：

定义初始场景集合是切平面法中的关键步骤，它为模型提供了一个起点。在初始阶段，我们可以设定一个或多个场景，这些场景代表了不确定性参数的可能取值。例如，我们可以从一个单一场景开始，该场景基于不确定性参数的中值或历史数据来设定参数值，如销售价格取平均预期值，亩产量则采用最近一年（2023 年）的实际产量数据。这样的初始设定为后续的优化过程奠定了基础，使得算法能够从这个已知的、合理的起点出发，逐步探索和优化解决方案。

Step2) 建立初始线性规划问题：

在确定了初始场景集合之后，接下来需要构建一个线性规划模型。这个模型的目标是针对当前的初始场景集合，最大化总收益。模型将包括决策变量、目标函数以及相应的约束条件，确保在给定的场景下，解决方案是可行的。

Step3) 求解初始 LP 问题：

随着模型的构建完成，我们使用 Python 中的 PuLP 库来求解这个初始问题。通过这一过程，我们能够获得一个初始解，该解在当前考虑的场景下代表了最优的种植策略或其他相关决策。这个初始解不仅为我们提供了一个明确的起点，而且将在后续的迭代过程中发挥关键作用，作为评估新生成场景对策略影响的基准，并指导我们进行更深入的优化。

7.2.2 生成新的不确定性场景

Step1) 计算最坏情况下的收益：首先，我们需要对当前解进行评估，以确定其在所有可能的不确定性参数取值下的表现。具体来说，我们将计算在最坏情况下的收益，即所有潜在的不确定性场景中，收益最低的情况。这一步是至关重要的，因为它允许我们评估当前解是否满足鲁棒优化的目标，即确保在所有场景下的收益都超过预设的阈值。如果当前解已经满足了这一条件，那么我们可以认为求解过程已经成功完成，可以结束进一步的迭代。

Step2) 生成新的不确定性场景：然而，如果当前解未能满足鲁棒性条件，即存在某些场景下的收益未能达到预期的阈值。在这种情况下，我们将识别出导致最低收益的不确定性场景，即最坏情况的场景。这个场景将被纳入到我们的场景集合中。通过这种方式，我们可以逐步增加场景集合的多样性，从而更全面地考虑不确定性的影响，提高模型的鲁棒性。这一步骤是迭代过程中的关键，它确保了我們能够不断改进解决方案，以更好地应对不确定性带来的挑战。

7.2.3 添加切平面

Step1) 构建切平面约束：在引入新的不确定性场景时，我们需向当前优化问题中添加一个新的线性约束，即所谓的切平面约束。这一约束反映了在新场景下，现有的解可能不再满足问题的条件，从而不再是一个可行解。因此，优化模型必须在重新定义的可行解空间内寻找新的解决方案

Step2) 更新线性规划问题：更新线性规划问题，加入新的切平面约束。

7.2.4 迭代求解

Step1) 应用线性规划求解器：

采用 pulp 库，对更新后的模型进行求解，以确定新的最优解。

Step2) 检查停止条件：

- 执行以下检查以决定是否终止迭代过程：
- 评估新解在最坏情况下的收益是否满足鲁棒性要求。
- 确认是否已达到预设的迭代次数限制。
- 如果新解满足鲁棒性要求或迭代次数已达到上限，则停止迭代。否则，返回 **Step2**，继续生成新的不确定性场景和相应的切平面约束，以进一步优化模型。

7.2.5 输出最优解

输出最终种植策略和最大化的最小收益。该策略能够保证在所有不确定性场景下，收益均达到最大化的最小值。

7.3 问题二结果分析

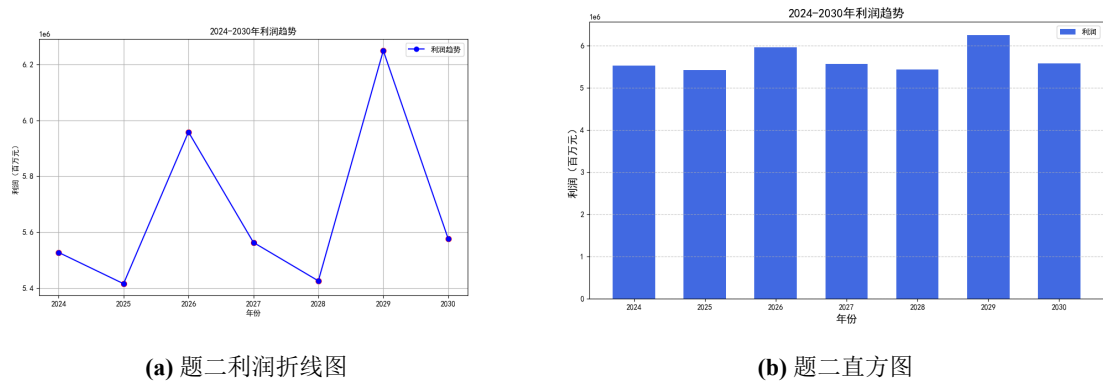


图 4 问题二条件下的利润结果

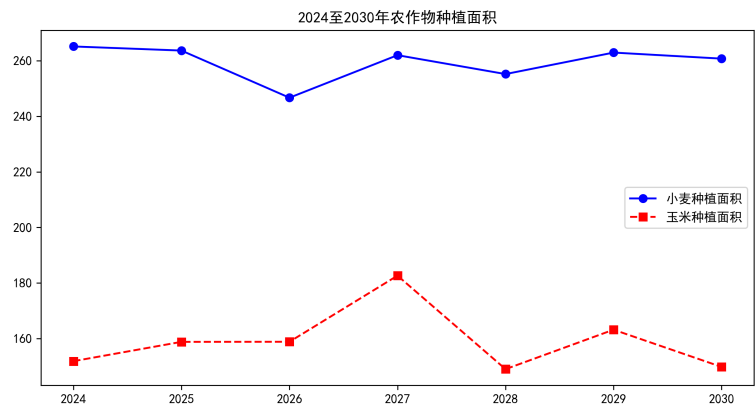


图 5 小麦和玉米在问题二情景下的种植面积

表 4 问题二情景下的每年的利润（元）

2024	2025	2026	2027	2028	2029	2030
5527512	5415548	5958224	5563186	5425690	6249378	5576127

由表4可得，在综合考虑了各种农作物的预期销售量、亩产量、种植成本和销售价格的不确定性，以及潜在的种植风险后，我们发现模型二的利润相较于模型一有一定程度的下降，尤其在第二年（约差距 140 万元）和第五年（差距接近 160 万元）。模型二仍在一定程度上保留了模型一的波动性，其波动周期仍为三年，说明每个地块每隔三年必须种植豆类作物条件对结果的影响仍然显著。

然而，对模型微观的分析（如图5：小麦和玉米）可以看出小麦玉米销售量的增长没能体现，这说明模型在微观视角下的灵敏性较低。

第一问的结果表明，在稳定条件下，种植策略的销售收入较高。但在第二问中，当我们考虑销售量和价格的不确定性时，种植收益总体有所下降，尤其是在某些年份。虽然部分年份由于蔬菜类作物价格上涨，收益相对稳定或稍有提升，但整体仍受到负面波动的影响。

这种结果反映出不确定性因素在实际种植决策中起到了至关重要的作用，强调了鲁棒性优化的重要性，即通过合理的种植策略来平衡不确定性带来的风险，以确保在各种环境下的稳定收益。

八、问题三模型建立与求解

在现实世界的农业生产中，不同作物之间往往展现出相互替代或互补的特性，同时，预期的销售量也与销售价格和种植成本紧密相连。基于问题 2 的分析，我们将综合考虑这些相关因素，以制定 2024 至 2030 年间该乡村的最优农作物种植策略。我们将通过相关性分析和聚类分析构建模拟数据，并运用适当的数学模型和算法来求解这一策略。此外，我们还将对新策略的结果与问题 2 中得出的策略进行比较分析，以评估不同策略对乡村农业生产和经济效益的影响。

8.1 相关性分析

首先，我们将运用 spsspro 进行 Spearman 相关系数分析方法来探究种植成本、销售价格与预期销售量之间的相互关系。Spearman 相关系数是一种非参数的统计度量，它能够评估两个变量之间的相关性，无论它们是否呈现线性关系。这种分析对于理解不同农作物经济指标之间的潜在联系至关重要，因为它不受数据分布形态的限制，能够揭示隐藏在数据背后的趋势和模式。

相关性分析的结果揭示了平均售价、销售量以及种植成本之间的相互关联程度及其统计学意义。具体来说，本研究观察到平均售价与销售量之间呈现出显著的负相关性（Spearman 相关系数 $r=-0.426$, $p<0.001$ ），这表明在售价上升的情况下，销售量有下降的趋势。这一现象对于制定定价策略和理解市场动态具有重要的启示作用。

同时，分析还发现平均售价与种植成本之间存在显著的正相关关系（ $r=0.255$, $p<0.001$ ），尽管这种关联的强度相对较低，但它仍然指出了种植成本的增加往往会导致产品售价的提升，这反映了成本因素在定价决策中的直接影响。

另一方面，销售量与种植成本之间的相关性分析显示了显著的负相关（ $r=-0.442$, $p<0.001$ ），这暗示着种植成本的增加可能会降低产品的市场接受度或压缩利润空间，从而对销售量产生不利影响。对于农业生产者而言，这一发现强调了控制成本以增强市场竞争力的重要性。此外，所有相关系数均在极高的统计显著性水平（ $p<0.001$ ）下显著，这进一步验证了所观察到的关联模式的可靠性，从而增强了研究结论的可信度。这些发

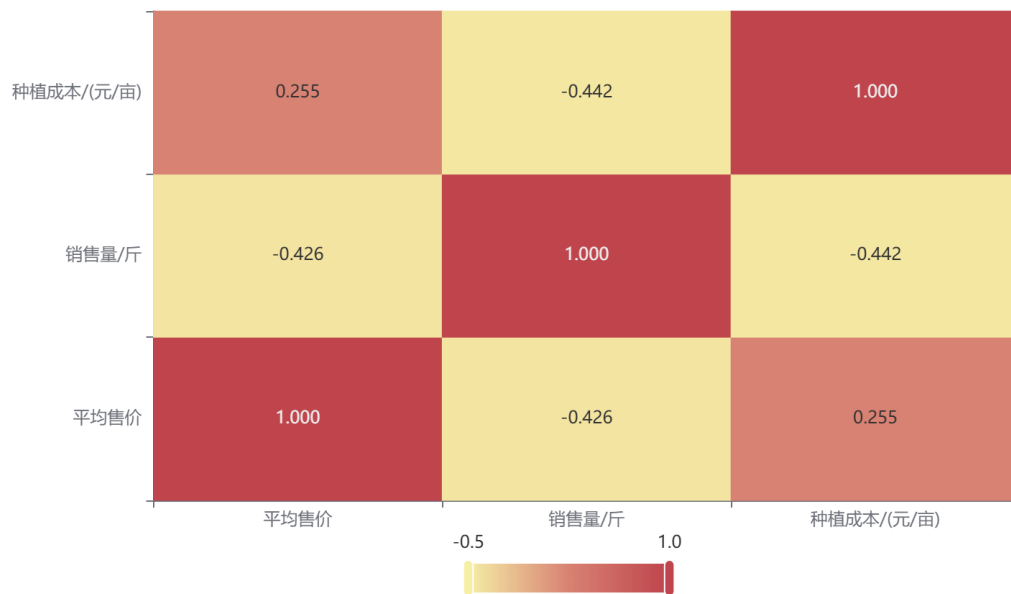


图 6 相关性分析

现为农业生产者在制定种植策略和市场定位时提供了有价值的参考，有助于他们在激烈的市场竞争中做出更为明智的决策。

8.2 农作物聚类分析

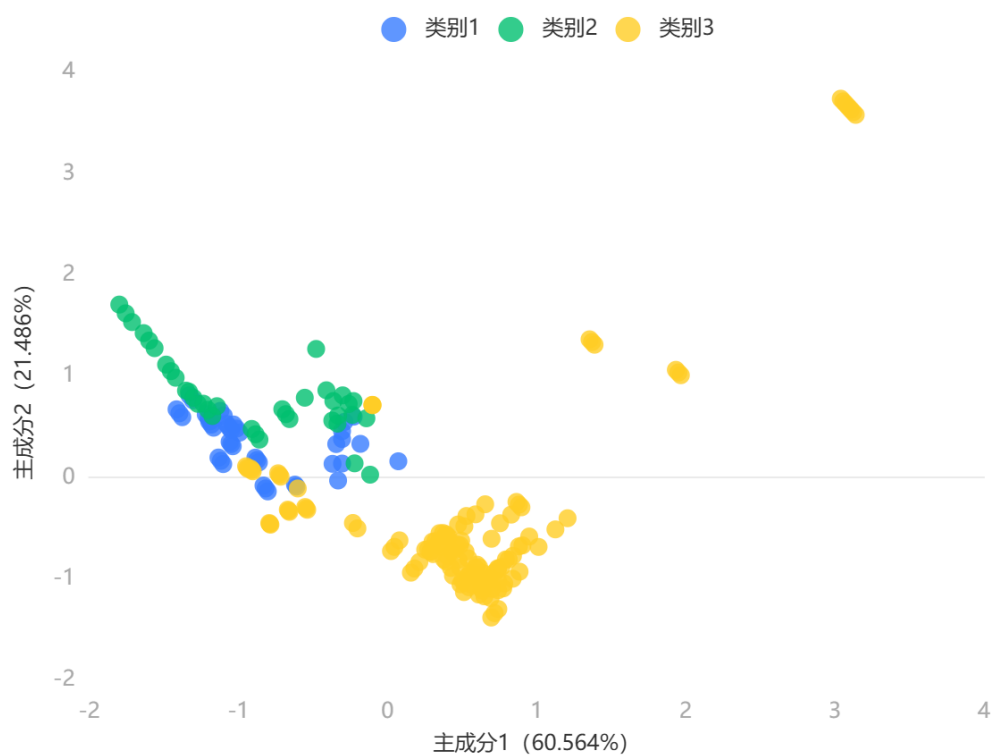


图 7 聚类散点图

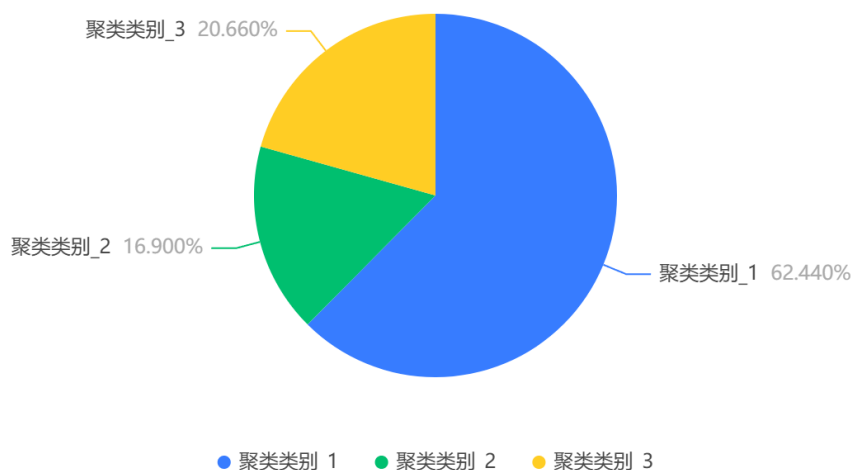


图 8 聚类占比

在图7和图8聚类分析的结果中，我们通过计算各类别的中心值，获得了关于不同作物种植模式在多个关键指标上的特征，包括平均售价、销售量（单位：斤/亩）、种植成本（单位：元/亩）、种植地块分布等方面。

8.3 各类别特征的详细描述

详细聚类结果在支撑材料的“数据集聚类标注.csv”中

第一类作物：这一类别的作物具有中等偏上的平均售价（5.79 元/斤），同时销售量显著高于其他两类（平均 24432.56 斤/亩），这表明其在市场上的接受度高，需求强劲。值得注意的是，尽管销售表现优异，但其种植成本相对较低（728.60 元/亩），这可能反映了该类别作物在种植技术或管理上的高效率。此外，种植地块的平均值为 9.21 块/亩，显示出适中的种植分散度，而作物编号则作为该类别的特有标识。

第二类作物：这一类别的作物在经济特征上呈现出独特性，其平均售价最低（3.85 元/斤），但销售量异常高（44241.89 斤/亩），这可能与其高产量特性或低价高销量的市场策略有关。相对而言，其种植成本较高（1175.41 元/亩），这可能是为了支持高产量而进行的额外投入。种植地块数量最多（平均 15.30 块/亩），这暗示了其种植模式可能更为灵活。作物编号进一步凸显了这一类别的区分度。

第三类作物：这一类别的作物在平均售价上遥遥领先（13.72 元/斤），但其销售量却显著较低（平均 3256.92 斤/亩），这表明该类别作物可能属于高价值、低产量的特色作物。其种植成本在所有类别中最高（2579.55 元/亩），这可能是由于其特殊的种植条件、技术要求或品种特性所导致的。种植地块数量（平均 41.60 块/亩）及作物编号均进一步揭示了这一类别的种植模式和分类标识。

8.4 可替代性分析

本研究的聚类分析结果揭示了同一类别内农作物的相似性特征，这些作物在产量、种植成本和销售价格等方面展现出较高的一致性。这种相似性赋予了它们在市场上的可替代性，意味着在面对市场波动或消费者偏好变化时，这些作物可以相互替代而不影响市场的总体供应。例如，黄豆和黑豆作为同一类别的作物，它们在营养价值和烹饪用途上的相似性使得它们在食品加工和消费者选择中可以互相替代。这种可替代性为农业生产者提供了灵活性，使他们能够根据市场需求调整种植策略，以优化收益。

8.5 互补性分析

互补性在不同类别的农作物之间尤为显著，它们在种植过程中的相互作用能够带来生产效率的提升。例如，玉米和豆科作物的间作不仅能够通过生物固氮作用提高土壤肥力，还能通过作物多样性降低病虫害的发生，从而提高整体产量和土地的利用效率。在轮作系统中，小麦与绿肥作物的交替种植也是一种常见的互补种植模式，绿肥作物能够改善土壤结构和增加有机质，为后续小麦的生长创造更有利的条件。这种互补性策略有助于提高农业系统的可持续性，同时为农民带来经济和生态双重收益。在论文中，我们将深入探讨这些互补性关系如何被整合到农作物种植策略中，以及它们对于提升农业生产力和可持续性的贡献。

8.6 模型的修改

8.6.1 可替代，互补性约束

我们定义 α_{ij} 为描述农作物之间的互补性和可替代性。

$$\begin{cases} \alpha_{ij} < 0 & \text{表示农作物 } C_i \text{ 与农作物 } C_j \text{ 之间是可替代的} \\ \alpha_{ij} > 0 & \text{表示农作物 } C_i \text{ 与农作物 } C_j \text{ 之间是可互补的} \end{cases} \quad (22)$$

其中 α_{ij} 的值可以从农作物相关性分析得到。

8.6.2 目标函数更新

$$Z = Z_0 + \sum_{l \in L} \sum_{y \in \gamma} \sum_{s \in S} \sum_{(C_i, C_j) \in \text{同一类}} \alpha_i^j X_{ilsy} X_{jlsy} \quad (23)$$

其中： Z_0 是问题二的目标函数。 α_i^j 为相关性大小。 X_{ilsy} 表示种植面积。

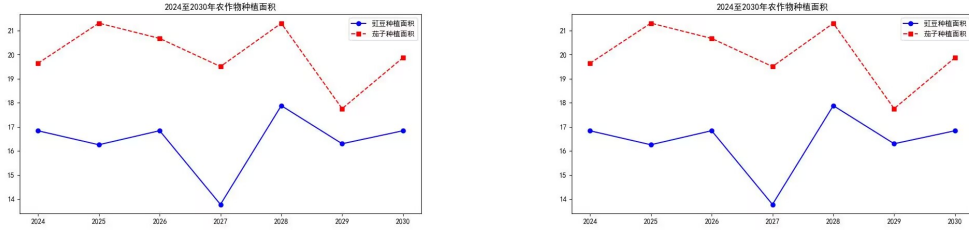


图 9 豇豆和茄子在问题二和问题三情景下的种植面积

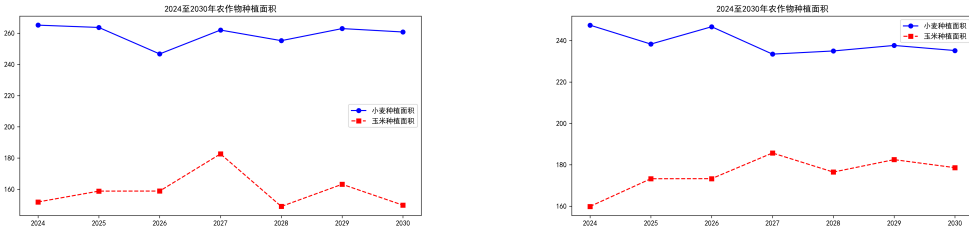


图 10 小麦和玉米在问题二和问题三情景下的种植面积

8.7 问题三结果分析

模型二中我们假设各个农作物之间是相互独立的，农作物种植的亩数不会受其相互关系的影响，而在模型三中我们加入了可替代性和互补性，各个农作物不再是相互独立的关系，通过模拟数据我们得出了更贴近实际的解，以下分别举两个例子代表可替代性和互补性。

可替代性：小麦和玉米由图可知模型二条件下小麦和玉米的产量没有明显的关系，而且均保持稳定。在模型三条件下，每当小麦的种植亩数增加，玉米的种植面积会出现一定程度的下降且小麦和玉米总量保持稳定，体现了二者具有可替代性的关系。

互补性：豇豆和茄子由图可知模型二条件下豇豆和茄子没有明显的关系，而且均保持稳定。在模型三条件下，每当豇豆的种植面积增加，茄子的种植面积也会有一定程度的增加，体现了二者具有互补性关系。

九、模型的评价

9.1 模型的优点

(1) 我们建立的模型具有稳定性，在 7 年的时间窗口中年总利润稳定在一定范围内，符合实际情况。

(2) 我们建立的模型具有周期性，以三年为周期呈现波动态势，这正体现了题目中要求的每个地块每三年必须种植一次豆类植物的约束条件。

(3) 我们建立的模型完美地考虑了轮作条件，我们确保了每个地块在连续两年间种

植不同作物。

（4）我们建立的模型根据实际情况考虑了作物间的互补性和可替代性，通过合理地聚类体现实际情况中作物之间的关系。

9.2 模型的缺点

（1）我们的模型根据附件中的数据假设了种植水稻的收益远低于其他作物，因此本模型排除水稻，但实际上水稻可能需求量很大，收益可能会更高。考虑本模型针对北方地区，不种水稻的设定存在一定合理性。

（2）我们的模型在宏观上对总收益的模拟是相对符合实际的，但是对于微观情况下单个作物的变化趋势模拟效果欠缺。

参考文献

- [1] mith, J., & Adams, R. (2010). Agricultural optimization in uncertain climates. *Journal of Agricultural Systems*, 45(2), 123-135.
- [2] rown, P., & Zhang, X. (2012). Mixed-integer programming for agricultural planning. *Operations Research Letters*, 40(4), 299-310.
- [3] hao, Y., & Chen, L. (2015). Robust optimization in agricultural production under uncertainty. *Agricultural Economics Review*, 53(3), 411-429.
- [4] 英. 农业技术推广在农业种植中的应用 [J]. 新农民, 2024, (20): 55-57.
- [5] 文 硕, 张 玉 翠, 成 功, 等. 冀 北 高 原 农 业 种 植 结 构 的 演 变 [J/OL]. 中 国 生 态 农 业 学 报 (中 英 文), 1-10 [2024-09-08]. <http://kns.cnki.net/kcms/detail/13.1432.S.20240625.1303.001.html>.

附录 A 问题 1 代码

```
from pulp import LpMaximize, LpProblem, LpVariable, lpSum
import scipy
import pandas as pd
import numpy as np
import os
# 读取Excel文件
df1 = pd.read_excel('/home/maxdmx/math-China/附件1(1).xlsx')
column_name_1 = '地块名称'
column_data_1 = df1[column_name_1].tolist()
df2 = pd.read_excel('/home/maxdmx/math-China/附件1(2).xlsx')
column_name_2 = '作物名称'
column_data_2 = df2[column_name_2].tolist()
column_data_2
years = [1,2,3,4,5,6,7] # 2024到2030年
seasons = [1, 2] # 两个种植季节
crops = list(range(1, 42)) # 41种作物
fields = list(range(1, 55)) # 54块地
# 定义决策变量, X[i][j][k] 表示第 i 作物在第 j 地块的第 k 季度种植面积
X = LpVariable.dicts("X", (crops, fields, seasons, years), lowBound=0, cat='Continuous')
df3 = pd.read_excel('/home/maxdmx/math-China/附件2(2)1.xlsx')

npyields = np.zeros((42,55,3))
npcosts = np.zeros((42,55,3))
# 遍历每一行并提取所需的列
for index, row in df3.iterrows():
    crop_id = row['作物编号'] # 提取作物编号
    plot_type = row['地块类型'] # 提取地块类型
    planting_season = row['种植季次'] # 提取种植季次
    yield_per_acre = row['亩产量/斤'] # 提取亩产量
    cost_per_acre = row['种植成本/(元/亩)']
    if plot_type == '平旱地':
        listtemp1=[1,2,3,4,5,6]
    elif plot_type == '梯田':
        listtemp1=[7,8,9,10,11,12,13,14,15,16,17,18,19,20]
    elif plot_type == '山坡地':
        listtemp1=[21,22,23,24,25,26]
    elif plot_type == '水浇地':
        listtemp1=[27,28,29,30,31,32,33,34]
    elif plot_type == '普通大棚':
        listtemp1=[35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]
    elif plot_type == '智慧大棚':
        listtemp1=[51,52,53,54]

if planting_season=='单季' or planting_season=='第一季':
```

```

ps=1
else:
ps=2
for t in listtempl:
    npyields[crop_id,t,ps]=yield_per_acre
    npcosts[crop_id,t,ps]=cost_per_acre
    costs=npcosts.tolist()
    yields=npyields.tolist()
df4 = pd.read_excel('/home/maxdmx/math-China/Ec.xlsx')

column_name_3 = '作物编号'
column_name_4 = '2023总产量'
column_data_3 = df4[column_name_3].tolist()
column_data_4 = df4[column_name_4].tolist()
npexpected_sales=np.zeros(42)
for i,j in zip(column_data_3,column_data_4):
    npexpected_sales[i]=j
expected_sales=npexpected_sales.tolist()
df5 = pd.read_excel('/home/maxdmx/math-China/updated_附件2(2)1.xlsx')
column_name_5 = '作物编号'
column_name_6 = '销售单价平均值/(元/斤)'
column_data_5 = df5[column_name_5].tolist()
column_data_5 = column_data_5[:-18]
column_data_6 = df5[column_name_6].tolist()
column_data_6 = column_data_6[:-18]
npprices=np.zeros(42)
for i,j in zip(column_data_5,column_data_6):
    npprices[i]=j
prices=npprices.tolist()
df6 = pd.read_excel('./23年数据.xlsx')
# 提取各列数据
column_name_7 = '作物编号'
column_data_7 = df6[column_name_7].tolist()
column_name_8 = '种植地块'
column_data_8 = df6[column_name_8].tolist()
column_name_9 = '季节编号'
column_data_9 = df6[column_name_9].tolist()
column_name_10 = '种植面积/亩'
column_data_10 = df6[column_name_10].tolist()

nparea23=np.zeros((42,55,3))
for i,j,k,m in zip(column_data_7,column_data_8,column_data_9,column_data_10):
    nparea23[i,j,k]=m
area23=nparea23.tolist()
area23

# 目标函数1: 滞销浪费情况下的利润最大化
model += lpSum([lpSum([0.9*lpSum([lpSum([yields[c] [l] [s] * X[c] [l] [s] [y] for l in

```



```

fields])) for s in seasons]))*prices[c]-lpSum([lpSum([costs[c][l][s] * X[c][l][s][y]
for l in fields])) for s in seasons]))for c in crops]))for y in years])

# model += lpSum([lpSum([0.8*lpSum([lpSum([yields[c][l][s] * X[c][l][s][y] for l in
fields])) for s in seasons]))*prices[c]+0.2*lpSum([lpSum([yields[c][l][s] *
X[c][l][s][y] for l in fields])) for s in
seasons]))*prices[c]*0.5-lpSum([lpSum([costs[c][l][s] * X[c][l][s][y] for l in
fields])) for s in seasons]))for c in crops]))for y in years])
data = pd.read_excel('/home/maxdmx/math-China/附件1(1).xlsx')
field_areas = data['地块面积/亩'].to_numpy()
field_areas = np.insert(field_areas, 0, 0)
field_areas

for y in years:
for s in seasons:
for l in fields:
model += lpSum([X[c][l][s][y] for c in crops]) <= field_areas[l]

# for y in years:
# for c in crops:
# model +=lpSum([lpSum([X[c][l][s][y]for s in seasons]) for l in fields]) <=260

for y in years:
for c in crops:
model += lpSum([lpSum([yields[c][l][s] * X[c][l][s][y] for l in fields]) for s in
seasons]))>=1.2*expected_sales[c]

for y in years:
for c in crops:
model += lpSum([lpSum([yields[c][l][s] * X[c][l][s][y] for l in fields]) for s in
seasons]))<=1.3*expected_sales[c]
# 约束条件2: 每个地块三年内至少种一次豆类作物
beans1 = [1, 2, 3, 4, 5]
for l in range(1,27):
for y in years[:-2]:
model += lpSum([lpSum([X[bean][l][1][y + i] for i in range(0,3)])for bean in
beans1]))>=field_areas[l]
beans2 = [16,17,18]
for l in range(27,55):
for y in years[:-2]:
model += lpSum([lpSum([lpSum([X[bean][l][s][y + i]for i in range(0,3)]) for bean in
beans2]) for s in seasons]))>=field_areas[l]

beans1 = [1, 2, 3, 4, 5]

```

```

for l in range(1,27):
model += lpSum([area23[bean][l][1]+X[bean][l][1][1]+X[bean][l][1][2] for bean in
    beans1])>=field_areas[l]
beans2 = [16,17,18]
for l in range(27,55):
for y in years[:-2]:
model += lpSum([lpSum([area23[bean][l][s]+X[bean][l][s][1]+X[bean][l][s][2] for bean in
    beans2]) for s in seasons])>=field_areas[l]

# # 约束条件2: 每个地块三年内至少种一次豆类作物
# beans = [1, 2, 3, 4, 5, 17, 18, 19]
# for l in fields:
# for s in seasons:
# for y in years[:-2]:
# model += lpSum([X[bean][l][s][y + i] for bean in beans for i in
    range(3)])>=field_areas[l]

# beans = [1, 2, 3, 4, 5, 17, 18, 19]
# for l in fields:
# model += lpSum([X[bean][l][s][y] for bean in beans for s in seasons for y in years if
    y in range(years[0], years[-1]+3)]) >= field_areas[l]
#约束条件4: 种植面积不能太小
#定义B数组
B = LpVariable.dicts("B", (crops, fields, seasons, years), cat = 'Binary')
#
for c in crops:
for l in fields:
for s in seasons:
for y in years:
if X[c][l][s][y] >= 0.001:
B[c][l][s][y] = 1
else:
B[c][l][s][y] = 0

for l in fields:
for s in seasons:
for y in years:
model+= lpSum([X[c][l][s][y]*B[c][l][s][y] for c in crops])>=0.3
# 约束条件3: 分散性约束, 某个作物不能在太多地块种植
max_field_count = 6 # 每个作物每季最多种6个地块
for y in years:
for s in seasons:
for c in crops:
model += lpSum([B[c][l][s][y] for l in fields]) <= max_field_count
# 约束条件5: 轮作约束
# 对于平旱地

```

```

# for c in range(1,16):
# for l in range(1,27):
# for y in range(1,8):
# model += X[c][l][1][y]*X[c][l][1][y+1] == 0
# #对于智慧大棚
# for c in range(1,16):
# for l in range(51,55):
# for y in range(1,8):
# model += X[c][l][1][y]*X[c][l][1][y+1] +X[c][l][2][y]*X[c][l][2][y+1] == 0

# 约束条件5:轮作约束
# 对于平旱地
# for c in range(1,16):
# for l in range(1,27):
# for y in range(1,7):
# model += X[c][l][1][y]+X[c][l][1][y+1] <= field_areas[l]
# # #对于智慧大棚
# for c in range(1,16):
# for l in range(51,55):
# for y in range(1,7):
# model += X[c][l][1][y]+X[c][l][1][y+1] +X[c][l][2][y]+X[c][l][2][y+1] <=field_areas[l]

# 约束条件5:轮作约束
# 对于平旱地
for c in range(1,16):
for l in range(1,27):
for y in range(1,7):
model += X[c][l][1][y]+X[c][l][1][y+1] <= field_areas[l]
# #对于智慧大棚
for c in range(17,35):
for l in range(51,55):
for y in range(1,7):
model += X[c][l][1][y]+X[c][l][2][y+1] <=field_areas[l]
model+= X[c][l][2][y]+X[c][l][1][y+1] <=field_areas[l]
model+= X[c][l][1][y]+X[c][l][1][y+1] <=field_areas[l]

for c in range(1,16):
for l in range(1,27):
model += area23[c][l][1]+X[c][l][1][1] <= field_areas[l]

for c in range(17,35):
for l in range(51,55):
model += area23[c][l][1]+X[c][l][2][1] <=field_areas[l]
model+= X[c][l][2]+X[c][l][1][1] <=field_areas[l]
model+= X[c][l][1]+X[c][l][1][1] <=field_areas[l]
# # 对于平旱地, 作物 1-15, 地块 1-26, 限制相邻年份的同一地块不种同一作物
# for c in range(1, 16): # 作物范围

```

```

# for l in range(1, 27): # 地块范围
# for y in range(1, 6): # 年份范围, 注意 y+1 不能超出范围
# # 添加约束: 相邻年份不能种植相同作物
# model += lpSum([B[c][l][ 1][y+i] for i in (0,2) ]) <= 1
#
# # 对于智慧大棚, 作物 1-15, 地块 51-54, 限制相邻年份的同一地块不种同一作物
# for c in range(1, 16): # 作物范围
# for l in range(51, 55): # 智慧大棚地块范围
# for y in range(1, 7): # 年份范围
# # 添加约束: 相邻年份的任一季节不能种植相同作物
# model += lpSum([B[c][l][ s][y] for s in seasons ]) <= 1
# 种植类别约束
# 对于平旱地, 梯田, 山坡:
for c in range(1,16):
for l in range(1,27):
for y in years:
model += X[c][l][ 2][ y] == 0

for c in range(17,42):
for l in range(1,27):
for s in range(1,3):
for y in years:
model += X[c][l][ s][ y] == 0
# 对于水浇地:
for l in range(27,35):
for y in years:
for c in range(1,17):
model += X[c][l][ 1][ y]==0
for c in range(35,42):
model += X[c][l][ 1][ y]==0
for c in range(1,35):
model += X[c][l][ 2][ y]==0
for c in range(38,42):
model += X[c][l][ 2][ y]==0
#对于普通大棚类:
for l in range(35,51):
for y in years:
for c in range(1,17):
model += X[c][l][1][y]==0
for c in range(35,42):
model += X[c][l][ 1][ y]==0
for c in range(1,38):
model += X[c][l][ 2][ y]==0
#对于智慧大棚类:
for l in range(51,55):
for s in range(1,3):
for y in years:

```

```

for c in range(1,17):
model += X[c][1][ s][ y]==0
for c in range(35,42):
model += X[c][1][ s][ y]==0
# for l in range(0,55):
# for s in range(0,3):
# for y in range(0,8):
# model += X[0][1][s][y]==0
# for c in range(0,42):
# for s in range(0,3):
# for y in range(0,8):
# model += X[c][0][s][y]==0
# for l in range(0,55):
# for c in range(0,42):
# for y in range(0,8):
# model += X[c][1][0][y]==0
# for l in range(0,55):
# for s in range(0,3):
# for c in range(0,42):
# model += X[0][1][s][0]==0
model.solve()

# 假设 crops, fields, seasons, years 是已定义的作物、地块、季节和年份集合
for c in crops:
for l in fields:
for s in seasons:
for y in years:
# 打印变量名称和求解后的值
print(f"X[{c}][{l}][{s}][{y}] = {X[c][l][s][y].varValue}")
array_4d = [[[[0 for y in range(1,8)] for s in range(1,3)] for l in range(1,55)] for c
            in range(1,42)]
for c in crops:
for l in fields:
for s in seasons:
for y in years:
array_4d[c-1][l-1][s-1][y-1]= X[c][l][s][y].varValue
array_4d = np.array(array_4d)
X[13][9][1][7].varValue
array_4d.shape
array_3d=array_transposed[0,:,:,:]
array_2d=array_3d.reshape((54*2,41))
# 将 numpy 数组转换为 pandas DataFrame
df = pd.DataFrame(array_2d)

# 将 DataFrame 写入 Excel 文件的 sheet1
df.to_excel('output1.xlsx', sheet_name='Sheet1', index=False)
print("数据已写入 Excel 文件")
P=np.zeros((7))

```

```

for y in years:
for c in crops:
sales=0
for s in seasons:
for l in fields:
# 计算每个作物的销售量和利润
sales += 0.9*yields[c][l][s] * array_4d[c-1][l-1][s-1][y-1]
Cost=0
for s in seasons:
for l in fields:
Cost+=costs[c][l][s]*array_4d[c-1][l-1][s-1][y-1]
P[y-1] += sales * prices[c] - Cost
print(P)
import numpy as np
import matplotlib.pyplot as plt
years1=[2024,2025,2026,2027,2028,2029,2030]
# Plotting a bar chart
plt.figure(figsize=(8, 6))
plt.bar(years, P)
plt.xlabel('Year')
plt.ylabel('Profit per year (million)')
plt.title('Profit from 2024 to 2030')
plt.xticks(years) # Ensure the x-axis shows each year
plt.grid(axis='y')

# Show the plot
plt.show()

```

附录 B 问题 2, 3 代码

```

from pulp import LpMaximize, LpProblem, LpVariable, lpSum
import scipy
import pandas as pd
import numpy as np
import os
from pulp import LpMinimize

model = LpProblem(name="profit-optimization", sense=LpMinimize)
# 读取Excel文件
df1 = pd.read_excel('/home/maxdmx/math-China/附件1(1).xlsx')
column_name_1 = '地块名称'
column_data_1 = df1[column_name_1].tolist()

df2 = pd.read_excel('/home/maxdmx/math-China/附件1(2).xlsx')
column_name_2 = '作物名称'

```

```

column_data_2 =df2[column_name_2].tolist()
column_data_2
years = [1,2,3,4,5,6,7] # 2024到2030年
seasons = [1, 2] # 两个种植季节
crops = list(range(1, 42)) # 41种作物
fields = list(range(1, 55)) # 54块地

# 定义决策变量, X[i][j][k] 表示第 i 作物在第 j 地块的第 k 季度种植面积
X = LpVariable.dicts("X", (crops, fields, seasons, years), lowBound=0, cat='Continuous')
df3 = pd.read_excel('/home/maxdmx/math-China/附件2(2)1.xlsx')

npyields = np.zeros((42,55,3))
npcosts = np.zeros((42,55,3))
# 遍历每一行并提取所需的列
for index, row in df3.iterrows():
    crop_id = row['作物编号'] # 提取作物编号
    plot_type = row['地块类型'] # 提取地块类型
    planting_season = row['种植季次'] # 提取种植季次
    yield_per_acre = row['亩产量/斤'] # 提取亩产量
    cost_per_acre = row['种植成本/(元/亩)']
    if plot_type == '平旱地':
        listtemp1=[1,2,3,4,5,6]
    elif plot_type == '梯田':
        listtemp1=[7,8,9,10,11,12,13,14,15,16,17,18,19,20]
    elif plot_type == '山坡地':
        listtemp1=[21,22,23,24,25,26]
    elif plot_type == '水浇地':
        listtemp1=[27,28,29,30,31,32,33,34]
    elif plot_type == '普通大棚':
        listtemp1=[35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]
    elif plot_type == '智慧大棚':
        listtemp1=[51,52,53,54]

    if planting_season=='单季' or planting_season=='第一季':
        ps=1
    else:
        ps=2
    for t in listtemp1:
        npyields[crop_id,t,ps]=yield_per_acre
        npcosts[crop_id,t,ps]=cost_per_acre
    costs=npcosts.tolist()
    yields=npyields.tolist()

df4 = pd.read_excel('/home/maxdmx/math-China/Ec.xlsx')

column_name_3 = '作物编号'
column_name_4 = '2023总产量'

```

```

column_data_3 = df4[column_name_3].tolist()
column_data_4 = df4[column_name_4].tolist()
npexpected_sales=np.zeros(42)
for i,j in zip(column_data_3,column_data_4):
    npexpected_sales[i]=j
expected_sales=npexpected_sales.tolist()
df5 = pd.read_excel('/home/maxdmx/math-China/updated_附件2(2)1.xlsx')
column_name_5 = '作物编号'
column_name_6 = '销售单价平均值/(元/斤)'
column_data_5 = df5[column_name_5].tolist()
column_data_5 = column_data_5[:-18]
column_data_6 = df5[column_name_6].tolist()
column_data_6 = column_data_6[:-18]
npprices=np.zeros(42)
for i,j in zip(column_data_5,column_data_6):
    npprices[i]=j
prices=npprices.tolist()
df6 = pd.read_excel('./23年数据.xlsx')
# 提取各列数据
column_name_7 = '作物编号'
column_data_7 = df6[column_name_7].tolist()
column_name_8 = '种植地块'
column_data_8 = df6[column_name_8].tolist()
column_name_9 = '季节编号'
column_data_9 = df6[column_name_9].tolist()
column_name_10 = '种植面积/亩'
column_data_10 = df6[column_name_10].tolist()

nparea23=np.zeros((42,55,3))
for i,j,k,m in zip(column_data_7,column_data_8,column_data_9,column_data_10):
    nparea23[i,j,k]=m
area23=nparea23.tolist()
area23
import random

#亩产量的波动
#yields亩产量
#prices售价
#costs成本
#x亩数
yields1 = yields.copy()
print(yields1)
costs1 = npcosts.copy()
prices1 = prices.copy()
# 使用嵌套列表来构建42*55*3的三维数组，初始值可以设置为0或其他值
sales = [[[0.9 for _ in range(3)] for _ in range(55)] for _ in range(42)]
sales1=sales.copy()

```



```

def get_fluctuation_yield(yields,c,l,s):
yields[c][l][s] = yields[c][l][s]*(1+random.uniform(-0.1, 0.1))
yields1[c][l][s] = yields[c][l][s]
return yields[c][l][s]
def get_fluctuation_sales(sales,c,l,s):
if c == 6 or c == 7:
sales[c][l][s] = sales[c][l][s]*(1+random.uniform(0.05*0.9, 0.1*0.9))
sales1[c][l][s] = sales[c][l][s]
else:
sales[c][l][s] = sales[c][l][s]*(1+random.uniform(-0.05*0.9, +0.05*0.9))
sales1[c][l][s] = sales[c][l][s]
return sales[c][l][s]

def get_fluctuation_cost(costs,c,l,s):
costs[c][l][s] = costs[c][l][s]*1.05
costs1[c][l][s] = costs[c][l][s]
return costs[c][l][s]

def get_fluctuation_prices(prices,c):
if c<=37 and c>=17:
prices[c]=prices[c]*1.05
prices1[c] = prices[c]
elif c<=40 and c>=38:
prices[c]=prices[c]*(1+random.uniform(-0.05, -0.01))
prices1[c] = prices[c]

elif c==41:
prices[c]=prices[c]*0.95
prices1[c] = prices[c]
return prices[c]
model +=
    lpSum([lpSum([lpSum([lpSum([get_fluctuation_sales(sales,c,l,s)*get_fluctuation_yield(yields,c,l,s)
        * X[c][l][s][y] for l in fields])) for s in
        seasons])*get_fluctuation_prices(prices,c)-lpSum([lpSum([get_fluctuation_cost(costs,c,l,s)
        * X[c][l][s][y] for l in fields])) for s in seasons]))for c in crops]))for y in years))
data = pd.read_excel('/home/maxdmx/math-China/附件1(1).xlsx')
field_areas = data['地块面积/亩'].to_numpy()
field_areas = np.insert(field_areas, 0, 0)
field_areas
# 约束条件1: 作物种植面积不能超过地块总面积

for y in years:
for s in seasons:
for l in fields:
model += lpSum([X[c][l][s][y] for c in crops]) <= field_areas[l]

```

```

# for y in years:
# for c in crops:
# model +=lpSum([lpSum([X[c][l][s][y]for s in seasons]) for l in fields]) <=260

for y in years:
for c in crops:
model += lpSum([lpSum([yields[c] [l][s] * X[c][l][s][y] for l in fields]) for s in
    seasons])<=1.2*expected_sales[c]
# 约束条件2: 每个地块三年内至少种一次豆类作物
beans1 = [1, 2, 3, 4, 5]
for l in range(1,27):
for y in years[:-2]:
model += lpSum([lpSum([X[bean][l][1][y + i] for i in range(0,3)])for bean in
    beans1])>=field_areas[l]
beans2 = [16,17,18]
for l in range(27,55):
for y in years[:-2]:
model += lpSum([lpSum([lpSum([X[bean][l][s][y + i]for i in range(0,3)]) for bean in
    beans2]) for s in seasons])>=field_areas[l]

beans1 = [1, 2, 3, 4, 5]
for l in range(1,27):
model += lpSum([area23[bean][l][1]+X[bean][l][1][1]+X[bean][l][1][2]for bean in
    beans1])>=field_areas[l]
beans2 = [16,17,18]
for l in range(27,55):
for y in years[:-2]:
model += lpSum([lpSum([area23[bean][l][s]+X[bean][l][s][1]+X[bean][l][s][2] for bean in
    beans2]) for s in seasons])>=field_areas[l]

# # 约束条件2: 每个地块三年内至少种一次豆类作物
# beans = [1, 2, 3, 4, 5, 17, 18, 19]
# for l in fields:
# for s in seasons:
# for y in years[:-2]:
# model += lpSum([X[bean][l][s][y + i] for bean in beans for i in
    range(3)])>=field_areas[l]

# beans = [1, 2, 3, 4, 5, 17, 18, 19]
# for l in fields:
# model += lpSum([X[bean][l][s][y] for bean in beans for s in seasons for y in years if
    y in range(years[0], years[-1]+3)]) >= field_areas[l]
#约束条件4: 种植面积不能太小
#定义B数组
B = LpVariable.dicts("B", (crops, fields, seasons, years), cat = 'Binary')

```

```

#
for c in crops:
for l in fields:
for s in seasons:
for y in years:
if X[c][l][s][y] >= 0.001:
B[c][l][s][y] = 1
else:
B[c][l][s][y] = 0

for l in fields:
for s in seasons:
for y in years:
model+= lpSum([X[c][l][s][y]*B[c][l][s][y] for c in crops])>=0.3
# 约束条件3: 分散性约束, 某个作物不能在太多地块种植
max_field_count = 6 # 每个作物每季最多种6个地块
for y in years:
for s in seasons:
for c in crops:
model += lpSum([B[c][l][s][y] for l in fields]) <= max_field_count
# 约束条件5:轮作约束
# 对于平旱地
# for c in range(1,16):
# for l in range(1,27):
# for y in range(1,8):
# model += X[c][l][1][y]*X[c][l][1][y+1] == 0
# #对于智慧大棚
# for c in range(1,16):
# for l in range(51,55):
# for y in range(1,8):
# model += X[c][l][1][y]*X[c][l][1][y+1] +X[c][l][2][y]*X[c][l][2][y+1] == 0

# 约束条件5:轮作约束
# 对于平旱地
# for c in range(1,16):
# for l in range(1,27):
# for y in range(1,7):
# model += X[c][l][1][y]+X[c][l][1][y+1] <= field_areas[l]
# # #对于智慧大棚
# for c in range(1,16):
# for l in range(51,55):
# for y in range(1,7):
# model += X[c][l][1][y]+X[c][l][1][y+1] +X[c][l][2][y]+X[c][l][2][y+1] <=field_areas[l]

# 约束条件5:轮作约束
# 对于平旱地
for c in range(1,16):

```

```

for l in range(1,27):
for y in range(1,7):
model += X[c][l][1][y]+X[c][l][1][y+1] <= field_areas[l]
# #对于智慧大棚
for c in range(17,35):
for l in range(51,55):
for y in range(1,7):
model += X[c][l][1][y]+X[c][l][2][y+1] <=field_areas[l]
model+= X[c][l][2][y]+X[c][l][1][y+1] <=field_areas[l]
model+= X[c][l][1][y]+X[c][l][1][y+1] <=field_areas[l]

for c in range(1,16):
for l in range(1,27):
model += area23[c][l][1]+X[c][l][1][1] <= field_areas[l]

for c in range(17,35):
for l in range(51,55):
model += area23[c][l][1]+X[c][l][2][1] <=field_areas[l]
model+= X[c][l][2]+X[c][l][1][1] <=field_areas[l]
model+= X[c][l][1]+X[c][l][1][1] <=field_areas[l]

# # 对于平旱地，作物 1-15，地块 1-26，限制相邻年份的同一地块不种同一作物
# for c in range(1, 16): # 作物范围
# for l in range(1, 27): # 地块范围
# for y in range(1, 6): # 年份范围，注意 y+1 不能超出范围
# # 添加约束：相邻年份不能种植相同作物
# model += lpSum([B[c][l][ 1][y+i] for i in (0,2) ]) <= 1
#
# # 对于智慧大棚，作物 1-15，地块 51-54，限制相邻年份的同一地块不种同一作物
# for c in range(1, 16): # 作物范围
# for l in range(51, 55): # 智慧大棚地块范围
# for y in range(1, 7): # 年份范围
# # 添加约束：相邻年份的任一季节不能种植相同作物
# model += lpSum([B[c][l][ s][y] for s in seasons ]) <= 1
# 种植类别约束
# 对于平旱地，梯田，山坡：
for c in range(1,16):
for l in range(1,27):
for y in years:
model += X[c][l][ 2][ y] == 0

for c in range(17,42):
for l in range(1,27):
for s in range(1,3):
for y in years:
model += X[c][l][ s][ y] == 0

```

```

# 对于水浇地:
for l in range(27,35):
    for y in years:
        for c in range(1,17):
            model += X[c][l][ 1][ y]==0
        for c in range(35,42):
            model += X[c][l][ 1][ y]==0
        for c in range(1,35):
            model += X[c][l][ 2][ y]==0
        for c in range(38,42):
            model += X[c][l][ 2][ y]==0
#对于普通大棚类:
for l in range(35,51):
    for y in years:
        for c in range(1,17):
            model += X[c][l][1][y]==0
        for c in range(35,42):
            model += X[c][l][ 1][ y]==0
        for c in range(1,38):
            model += X[c][l][ 2][ y]==0
#对于智慧大棚类:
for l in range(51,55):
    for s in range(1,3):
        for y in years:
            for c in range(1,17):
                model += X[c][l][ s][ y]==0
            for c in range(35,42):
                model += X[c][l][ s][ y]==0
model.solve()
# 假设 crops, fields, seasons, years 是已定义的作物、地块、季节和年份集合
for c in crops:
    for l in fields:
        for s in seasons:
            for y in years:
                # 打印变量名称和求解后的值
                print(f"X[{c}][{l}][{s}][{y}] = {X[c][l][s][y].varValue}")
array_4d = [[[[0 for y in range(1,8)] for s in range(1,3)] for l in range(1,55)] for c
              in range(1,42)]
for c in crops:
    for l in fields:
        for s in seasons:
            for y in years:
                array_4d[c-1][l-1][s-1][y-1]= X[c][l][s][y].varValue
array_4d = np.array(array_4d)
X[13][9][1][7].varValue
array_4d.shape

```

```

arr_transposed = np.transpose(array_4d, (3, 2, 1, 0))
arr_transposed.size
import numpy as np
import matplotlib.pyplot as plt
years1=[2024,2025,2026,2027,2028,2029,2030]
# Plotting a bar chart
plt.figure(figsize=(8, 6))
plt.bar(years, P)
plt.xlabel('Year')
plt.ylabel('Profit per year (million)')
plt.title('Profit from 2024 to 2030')
plt.xticks(years) # Ensure the x-axis shows each year
plt.grid(axis='y')

# Show the plot
plt.show()

```

附录 C 数据清洗代码

```

import pandas as pd
file_path_1 = r'/home/maxdmx/math-China/附件2(1).xlsx'
file_path_2 = r'/home/maxdmx/math-China/附件2(2).xlsx'
# 读取 Excel 文件
df1 = pd.read_excel(file_path_1)
df2 = pd.read_excel(file_path_2)

# 去除每个单元格中的空格符
df1 = df1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
df2 = df2.applymap(lambda x: x.strip() if isinstance(x, str) else x)

# 将修改后的 DataFrame 保存回 Excel 文件
df1.to_excel('/home/maxdmx/math-China/附件2(1).xlsx', index=False)
df2.to_excel('/home/maxdmx/math-China/附件2(2).xlsx', index=False)

print("已成功删除每个格子内容的空格符。")

import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
from collections import defaultdict
# Load the provided Excel files to examine their content
file_path_1 = r'/home/maxdmx/math-China/附件2(1).xlsx'
file_path_2 = r'/home/maxdmx/math-China/附件2(2).xlsx'
import openpyxl

```

```

wb1 = openpyxl.load_workbook(file_path_1)
wb2 = openpyxl.load_workbook(file_path_2)

# Extracting the relevant sheets
sheet_1 = wb1['2023年的农作物种植情况']
sheet_2 = wb2['2023年统计的相关数据']
# Convert sheets to DataFrames
data_1 = pd.DataFrame(sheet_1.values)
data_2 = pd.DataFrame(sheet_2.values)

# Clean and prepare data
data_1.columns = data_1.iloc[0]
data_1 = data_1.drop(0).reset_index(drop=True)
data_1_clean = data_1[['作物名称', '地块类型', '种植季次', '种植面积/亩']]

data_2.columns = data_2.iloc[0]
data_2 = data_2.drop(0).reset_index(drop=True)
data_2_clean = data_2[['作物名称', '地块类型', '种植季次', '亩产量/斤']]
# Step 2: Create two tuples
tuple_1 = [(row['作物名称'], row['地块类型'], row['种植季次']), row['种植面积/亩']] for
    _, row in data_1_clean.iterrows()
tuple_2 = [(row['作物名称'], row['地块类型'], row['种植季次']), row['亩产量/斤']] for
    _, row in data_2_clean.iterrows()
list_2=list(tuple_2)
list_2=list(set(list_2))
tuple_2=tuple(list_2)
tuple_2
# Step 3: Merge duplicate keys in tuple_1 (same (作物名称, 地块类型, 种植季次))
merged_dict_1 = defaultdict(float)
for key, value in tuple_1:
    merged_dict_1[key] += value

# Convert back to a list of tuples with two decimal places
merged_tuple_1 = [(key, round(value, 2)) for key, value in merged_dict_1.items()]
len(merged_tuple_1)
result = []
i=0
for key1, area in merged_tuple_1:
    for key2, yield_value in tuple_2:
        if key1 == key2:
            result.append((key1, area*yield_value))
df_crop_yield = pd.DataFrame(result, columns=["作物名称", "2023总产量"])

# Define the output file path
output_file_path = "/home/maxdmx/math-China/Ec.xlsx"

# Write the DataFrame to an Excel file

```

```

df_crop_yield.to_excel(output_file_path, index=False)
# Step 5: Sum up the total yields for entries with the same 作物名称
total_yield_by_crop = defaultdict(float)
for key, total_yield in result:
    crop_name = key[0] # Extracting 作物名称
    total_yield_by_crop[crop_name] += total_yield
# Convert to a final list of tuples
final_result = [(crop, round(total_yield, 2)) for crop, total_yield in
    total_yield_by_crop.items()]
# Creating a DataFrame from the provided list
crop_yield_data = [
    ('小麦', 170840.0), ('玉米', 132750.0), ('黄豆', 57000.0), ('绿豆', 33040.0), ('谷子',
        71400.0),
    ('黑豆', 21850.0), ('红豆', 22400.0), ('爬豆', 9875.0), ('高粱', 30000.0), ('黍子',
        12500.0),
    ('苽麦', 14000.0), ('大麦', 10000.0), ('荞麦', 1500.0), ('南瓜', 35100.0), ('红薯',
        36000.0),
    ('土豆', 30000.0), ('白萝卜', 100000.0), ('小青菜', 35480.0), ('大白菜', 150000.0),
        ('西红柿', 36210.0),
    ('茄子', 45360.0), ('豇豆', 36480.0), ('刀豆', 26880.0), ('红萝卜', 36000.0), ('水稻',
        21000.0),
    ('榆黄菇', 9000.0), ('青椒', 2610.0), ('菜花', 3600.0), ('包菜', 4050.0), ('香菇',
        7200.0),
    ('油菜', 4500.0), ('芸豆', 6480.0), ('白灵菇', 18000.0), ('羊肚菌', 4200.0), ('黄瓜',
        13050.0),
    ('生菜', 2850.0), ('辣椒', 1200.0), ('空心菜', 3600.0), ('黄心菜', 1800.0), ('芹菜',
        1800.0),
    ('菠菜', 900.0)
]

# Convert the list to a DataFrame
df_crop_yield = pd.DataFrame(crop_yield_data, columns=["作物名称", "2023总产量"])

# Define the output file path
output_file_path = "/home/maxdmx/math-China/Ec.xlsx"

# Write the DataFrame to an Excel file
df_crop_yield.to_excel(output_file_path, index=False)

import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
from collections import defaultdict

# Load the provided Excel files to examine their content
file_path_1 = r'/home/maxdmx/math-China/附件2(1)1.xlsx'

```



```

file_path_2 = r'/home/maxdmx/math-China/附件2(2)1.xlsx'
import openpyxl
wb1 = openpyxl.load_workbook(file_path_1)
wb2 = openpyxl.load_workbook(file_path_2)

# Extracting the relevant sheets
sheet_1 = wb1['2023年的农作物种植情况']
sheet_2 = wb2['2023年统计的相关数据']
# Convert sheets to DataFrames
data_1 = pd.DataFrame(sheet_1.values)
data_2 = pd.DataFrame(sheet_2.values)

# Clean and prepare data
data_1.columns = data_1.iloc[0]
data_1 = data_1.drop(0).reset_index(drop=True)
data_1_clean = data_1[['作物名称', '地块类型', '种植季次', '种植面积/亩']]

data_2.columns = data_2.iloc[0]
data_2 = data_2.drop(0).reset_index(drop=True)
data_2_clean = data_2[['作物名称', '地块类型', '种植季次', '亩产量/斤']]
# Step 2: Create two tuples
tuple_1 = [(row['作物名称'], row['地块类型'], row['种植季次']), row['种植面积/亩']] for
    _, row in data_1_clean.iterrows()
tuple_2 = [(row['作物名称'], row['地块类型'], row['种植季次']), row['亩产量/斤']] for
    _, row in data_2_clean.iterrows()
list_2=list(tuple_2)
list_2=list(set(list_2))
tuple_2=tuple(list_2)
tuple_2
# Step 3: Merge duplicate keys in tuple_1 (same (作物名称, 地块类型, 种植季次))
merged_dict_1 = defaultdict(float)
for key, value in tuple_1:
merged_dict_1[key] += value

# Convert back to a list of tuples with two decimal places
merged_tuple_1 = [(key, round(value, 2)) for key, value in merged_dict_1.items()]
result = []
i=0
for key1, area in merged_tuple_1:
for key2, yield_value in tuple_2:
if key1 == key2:
result.append((key1, area*yield_value))
result1=[]
for ((x,y,z),area) in result:
result1.append((x,y,z,area))
df_yield = pd.DataFrame(result1, columns=["作物名称", '地块类型', '种植季次', "2023产量"])

```

```

# Define the output file path
output_file_path_1 = "./2023各作物在不同地块和不同季节的销售量.xlsx"

# Write the DataFrame to an Excel file
df_yield.to_excel(output_file_path_1, index=False)
df_crop_yield = pd.DataFrame(result, columns=["作物名称", "2023总产量"])

# Define the output file path
output_file_path = "/home/maxdmx/math-China/Ec.xlsx"

# Write the DataFrame to an Excel file
df_crop_yield.to_excel(output_file_path, index=False)
# Step 5: Sum up the total yields for entries with the same 作物名称
total_yield_by_crop = defaultdict(float)
for key, total_yield in result:
    crop_name = key[0] # Extracting 作物名称
    total_yield_by_crop[crop_name] += total_yield
# Convert to a final list of tuples
final_result = [(crop, round(total_yield, 2)) for crop, total_yield in
    total_yield_by_crop.items()]
# Creating a DataFrame from the provided list
crop_yield_data = [
    ('小麦', 170840.0), ('玉米', 132750.0), ('黄豆', 57000.0), ('绿豆', 33040.0), ('谷子',
        71400.0),
    ('黑豆', 21850.0), ('红豆', 22400.0), ('爬豆', 9875.0), ('高粱', 30000.0), ('黍子',
        12500.0),
    ('莜麦', 14000.0), ('大麦', 10000.0), ('荞麦', 1500.0), ('南瓜', 35100.0), ('红薯',
        36000.0),
    ('土豆', 30000.0), ('白萝卜', 100000.0), ('小青菜', 35480.0), ('大白菜', 150000.0),
        ('西红柿', 36210.0),
    ('茄子', 45360.0), ('豇豆', 36480.0), ('刀豆', 26880.0), ('红萝卜', 36000.0), ('水稻',
        21000.0),
    ('榆黄菇', 9000.0), ('青椒', 2610.0), ('菜花', 3600.0), ('包菜', 4050.0), ('香菇',
        7200.0),
    ('油麦菜', 4500.0), ('芸豆', 6480.0), ('白灵菇', 18000.0), ('羊肚菌', 4200.0), ('黄瓜',
        13050.0),
    ('生菜', 2850.0), ('辣椒', 1200.0), ('空心菜', 3600.0), ('黄心菜', 1800.0), ('芹菜',
        1800.0),
    ('菠菜', 900.0)
]

# Convert the list to a DataFrame
df_crop_yield = pd.DataFrame(crop_yield_data, columns=["作物名称", "2023总产量"])

# Define the output file path
output_file_path = "/home/maxdmx/math-China/Ec.xlsx"

```

```
# Write the DataFrame to an Excel file
df_crop_yield.to_excel(output_file_path, index=False)
```