基于多目标优化的农业种植策略优化与决策支持
——以华北地区为例的多模型应用研究

摘要

本论文针对华北地区某村庄的农业种植问题，旨在在不确定的市场和气候条件下优化农作物种植方案，以最大化总收益、提升资源使用效率并兼顾社会效益。针对不同问题，本文构建了多种数学模型，包括基于滞销浪费和降价销售的线性规划模型、综合考虑不确定性的混合整数线性规划（MILP）模型，以及结合遗传算法和粒子群优化的混合优化模型（HGAPSO）。此外，通过相关性分析和聚类分析，将农作物进行分组，进一步优化作物种植策略。最终结果表明，综合应用多种优化方法能够显著提升种植决策的**合理性**和**稳健性**。论文还对模型的优缺点进行了深入评价，为农业生产提供了有效的决策支持。

对于问题一, 本文通过构建**线性规划模型**，优化华北某村庄的农作物种植方案，以最大化总收益或最小化浪费为目标，得出了 2024 至 2030 年间的最优种植策略。最终求解结果显示，在 2024 至 2030 年间，降价销售策略相比滞销浪费策略，可使**总收益增加约 8.3%，达到 6,393,084.40 元**。

对于问题二, 综合考虑农作物预期销售量、亩产量、种植成本和销售价格的波动，构建**混合整数线性规划(MILP)模型**与**遗传算法和粒子群优化的混合优化方法(HGAPSO)**，并通过情景分析得出了各情景下的最优种植方案。结果表明，**最优方案在多个情景下的平均收益为 5,986,032.29 元，显示了较高的稳定性和资源利用率**。

对于问题三, 在考虑作物替代性和互补性的基础上，通过**聚类分析**和**多目标优化模型**，进一步优化了种植策略。最终结果显示，**优化后的种植策略总收益提升至 6,586,032.29 元，且资源利用率平均达到 96.45%，明显优于前两种方案**。

最后，论文对各模型的优缺点进行了评价，为农业种植决策提供了有力的理论支持。

关键字：**农业种植优化　多目标优化　混合整数线性规划　遗传算法　粒子群优化**

一、 问题重述

1.1 问题背景

随着乡村振兴战略的稳步推进，最大效益地利用有限的耕地资源，遵循农作物自然生长规律以及采取科学种植方案，对推动乡村农业经济的可持续发展具有关键性意义。根据因地制宜、生物固氮的生物原理，选择适宜的农作物耕种，采用轮作、间作的科学种植策略，有利于在保护生态环境的前提下提高生产效益，规避潜在的种植风险。

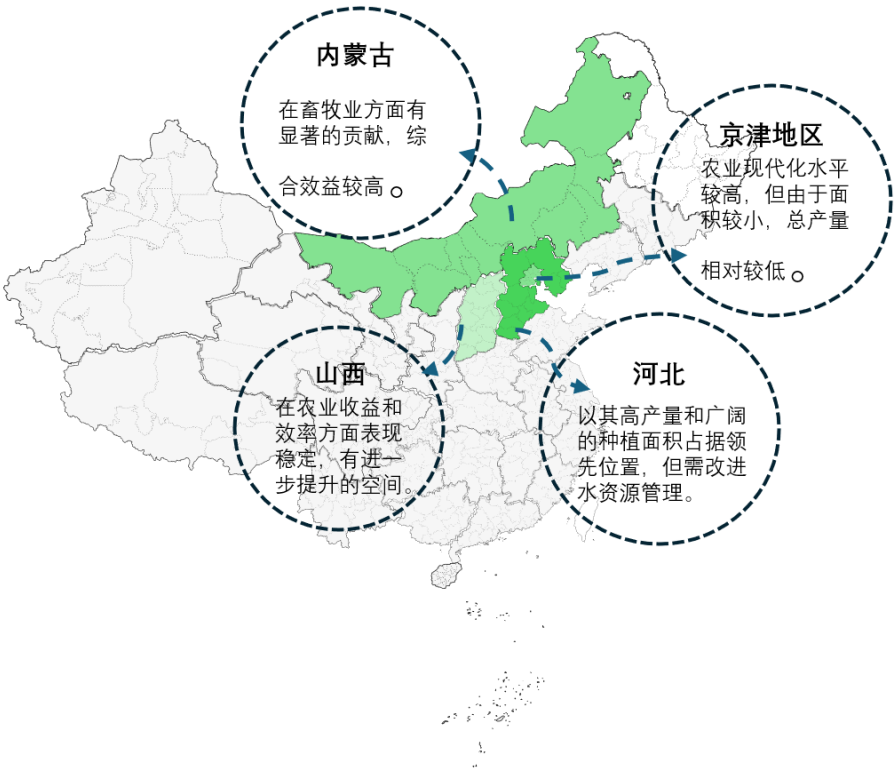

图 1　华北地区农业情况简图

华北地区的农业面临气候变化和水资源短缺的严峻挑战。近年来，该地区频繁遭遇高温干旱和极端天气，导致土壤墒情差、作物生长受阻，特别是在河北、山西等地，农田灌溉和水源供应压力巨大。南水北调工程和本地水资源的科学调度在一定程度上缓解了旱情，但仍需加强抗旱能力建设和高标准农田的灌溉设施完善，以保障农业生产的稳定。

在这种背景下，某村庄位于华北山区，拥有 1201 亩露天耕地，分为 34 块不同大小的地块，包括平旱地、梯田、山坡地和水浇地四种类型。其中，平旱地、梯田和山坡地

适合种植粮食作物，而水浇地适合种植水稻或蔬菜。此外，村庄还有 16 个普通大棚和 4 个智慧大棚，每个大棚面积为 0.6 亩，普通大棚适合种植蔬菜和食用菌，智慧大棚则适合种植两季蔬菜。面对持续的干旱和不稳定的气候条件，合理利用这些多样化的种植条件和设施是保障农业收益和抗风险能力的关键。

1.2 问题要求

本文建立数学模型，并设计求解方法解决如下问题：

问题 1 假设未来的农作物预期销售量、种植成本、亩产量和销售价格与 2023 年数据保持不变，并且当季产量超出预期部分无法正常销售。针对滞销浪费和按半价出售两种情况，构建优化模型，以最大化总收益或最小化浪费为目标，求解 2024 至 2030 年间的最优种植方案。

问题 2 综合考虑农作物预期销售量、亩产量、种植成本和销售价格的不确定波动，得出该乡村 2024~2030 年农作物的最优种植方案

问题 3 在问题二的基础上，考虑农作物之间的可替代性和互补性和预期销售量与销售价格、种植成本之间的相关性等相关因素，并模拟数据求解得出该乡村 2024~2030 年农作物的最有种植策略并与问题二中的结果作比较分析

二、 问题分析

2.1 问题一分析

对于问题一，首先进行数据预处理，整理了各类地块的面积和预期利润，并构建了一个作物与地块类型的适应性矩阵。之后定义决策变量、参数及其约束条件，包括地块类型与种植能力的限制、作物轮作与重茬限制、豆类作物种植要求，以及种植集中与分散度的要求。对于目标函数的设置，针对两种情况分别进行了优化：第一种情况是滞销浪费的最小化，通过最大化净收益或最小化滞销量来实现；第二种情况是超出部分按 50% 价格出售，调整目标函数以考虑降价对收益的影响，实现在不同约束条件下的最优种植方案。

2.2 问题二分析

对于问题二，首先考虑小麦、玉米和其他农作物的预期销售量、亩产量、种植成本和销售价格在未来的波动情况，构建基于不确定性的情景集（乐观、中性和悲观情景），并采用混合整数线性规划（MILP）模型进行求解。通过敏感性分析识别出对收益影响最大的变量，特别是种植成本的波动对整体收益有显著影响。此外运用混合遗传算法和粒子群优化（HGAPSO）方法，通过遗传算法的全局搜索能力和粒子群优化的局部探索能力，来实现种植方案的最优解。

图 2　问题分析流程图

2.3 问题三分析

对于问题三，首先进行相关性分析，计算了亩产量、种植成本和销售价格之间的相关系数，以确定它们的线性关系。接着，通过聚类分析，将作物的特征（包括产量、成本和价格）进行分组，以识别具有替代性或互补性的作物。基于这些分析构建一个多目标优化模型，综合考虑经济效益、社会效益和资源使用效率三个目标，使用决策变量和权重系数进行优化，以实现作物种植策略的最优组合，从而最大化整体收益和社会效益。

三、 模型假设

为简化问题，本文做出以下假设：

3.1 耕地资源和种植条件的稳定性假设

- 假设乡村的耕地资源（包括露天耕地、平旱地、梯田、山坡地、水浇地、普通大棚和智慧大棚）的总面积和类型在 2024 年至 2030 年之间保持不变。
- 假设每种地块类型适合种植的作物种类和数量维持不变，例如，平旱地、梯田和山坡地

四、 符号说明

符号	说明	单位
$\mathbf{A}_{adjust}$	适应性矩阵	-
a_{ij}	适应性矩阵的元素, 作物以 0 或 1 表示作物 i 是否适合在 j 种地块上种植	-
x_{ijt}	第 t 年在地块 j 上种植作物 i 的面积。这是一个连续变量，表示具体的种植面积。	亩
y_{ijt}	第 t 年在地块 j 上是否种植作物 i 的二进制变量，1 表示种植，0 表示不种植。	-
I	作物集合，包括所有可以种植的作物类型。	-
J	地块集合，包括所有的平旱地、梯田、山坡地、水浇地和大棚。	-
$\mathcal{B}$	豆类作物集合	-
G	粮食类作物的集合	-
R	水稻类作物的集合	-
V	蔬菜类（除大白菜，白萝卜，红萝卜，豆类）作物的集合	-
W	大白菜，白萝卜，红萝卜作物的集合	-
S	食用菌作物的集合	-
T	该集合包含 [1,7] 的整数，代表 2024-2030 共 7 年	-
p_{it}	作物 i 在第 t 年的亩产量。	吨/亩
s_{it}	作物 i 在第 t 年的销售价格。	元/吨

符号	说明	单位
c_{it}	作物 i 在第 t 年的种植成本。	元/亩
d_{it}	作物 i 在第 t 年的预期销售量。	吨
A_j	地块 j 的总面积	亩
M	极大值常数，用于逻辑约束。	-
$MinArea_i$	作物 i 的最低种植面积阈值。	亩
$Waste_{ijt}$	作物 i 在第 t 年在地块 j 的滞销浪费量。	吨

五、 问题一的模型的建立与求解

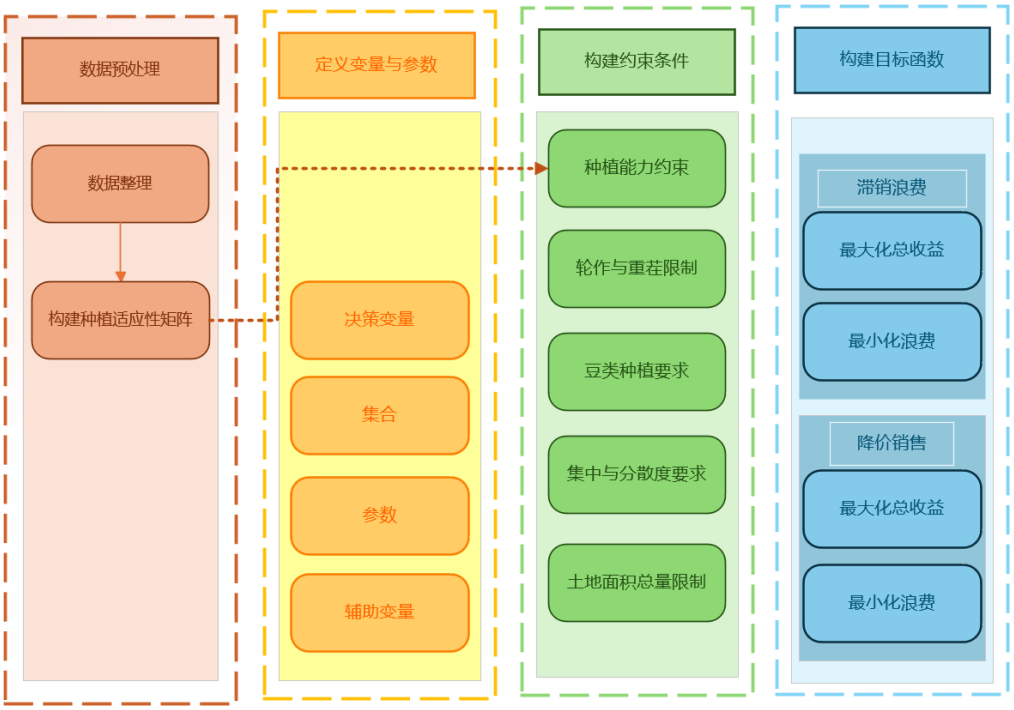

图 3　问题一建模流程图

5.1 数据预处理

5.1.1 耕地分布分析

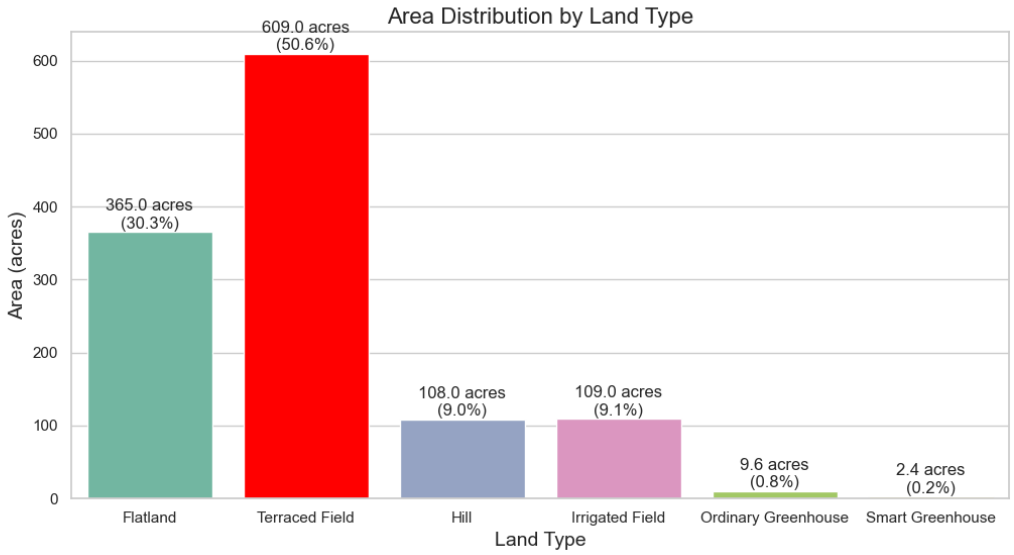

图 4　不同类型地块的面积分布

主要的耕地类型包括梯田、平旱地、山坡地、水浇地、普通大棚和智慧大棚。其中，**梯田面积最大，占比 50.6%**，其次是平旱地，占比 30.3%。山坡地和水浇地的面积接近，分别占 9.0% 和 9.1%。**普通大棚和智慧大棚**的面积较小，**占比分别为 0.8% 和 0.2%**。

从农业状况来看，这片区域的农业生产主要依赖于**梯田和平旱地**，这些土地类型**适合种植粮食作物**。水浇地尽管面积较小，但由于其适合种植水稻或蔬菜，具有较高的种植灵活性。但面对高比例的梯田，其农业生产可能面临机械化程度较低和劳动力需求较高的问题。

5.1.2 各作物类型分布分析

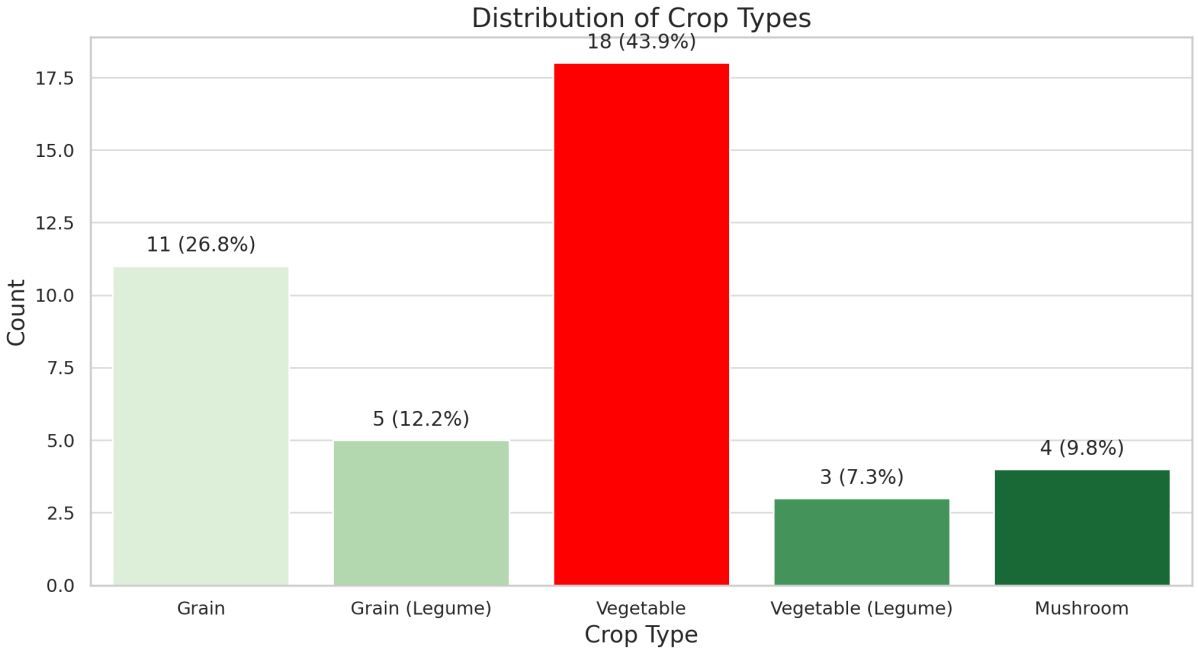

图 5　作物类型的数量分布与比例

从图表可以看出，**蔬菜是该地区的主要作物（43.9%）**，这可能是由于其**较高的市场需求和种植灵活性**。此外，粮食作物在数量上也占有较大比重（26.8%），表明该地可能以粮食为基础，确保粮食安全。豆类作物的相对较低比例（19.5%）暗示该地区在轮作或土壤改良方面的投入可能有限。**食用菌的种植虽然占比最小（9.8%）**，但由于其较高的经济价值和对种植条件的要求不同，可以作为农业多样化的补充。

5.1.3 各作物预期收益分析

整体来看，此地的作物利润状况表明，有些作物在特定条件下具有显著的盈利能力，而有些作物可能因市场价格或种植条件限制而面临亏损风险。优化作物种植组合，结合不同作物的盈利表现，将有助于提升整体农业收益。

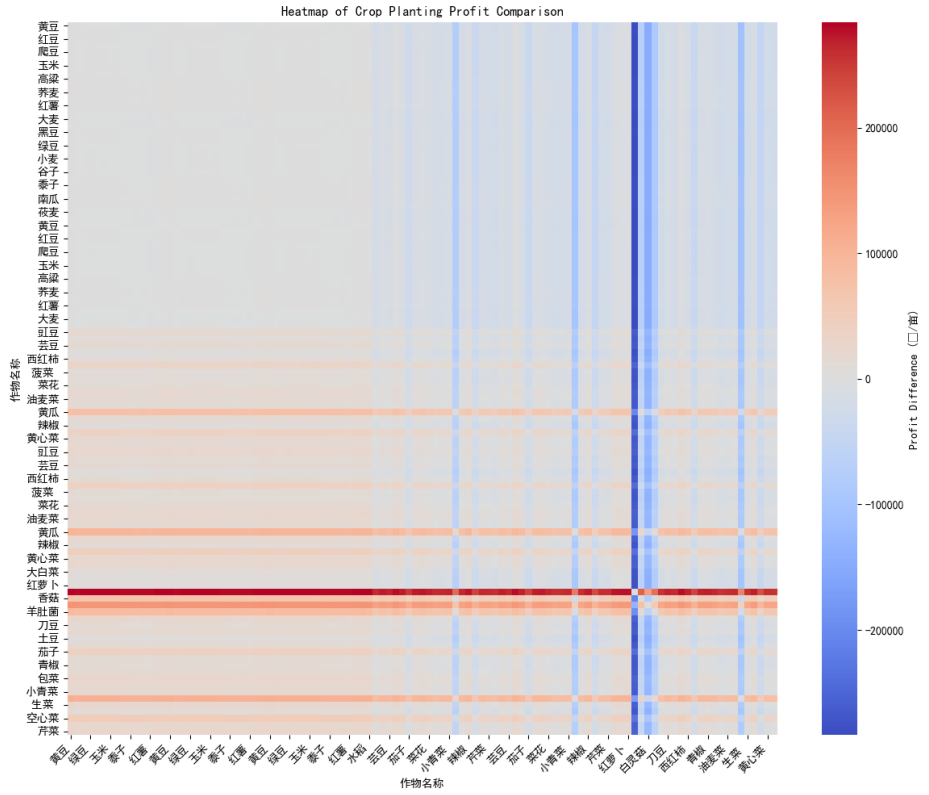

图 6　作物预期利润热图

5.2 线性规划模型建立

5.2.1 构建作物适应性矩阵

$$\mathbf{A}_{\text{adjust}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

其中每个矩阵元素定义为：

$$a_{ij} = \begin{cases} 1, & \text{如果作物}i\text{ 适合种植在地块类型}j\text{ 上;} \\ 0, & \text{如果作物}i\text{ 不适合种植在地块类型}j\text{ 上.} \end{cases}$$

5.2.2 约束条件处理

（1）重茬限制　每种作物在同一地块不能连续种植，即

$$y_{ijt} + y_{ij(t+1)} \leq 1, \quad \forall i \in I, \forall j \in J, \forall t \in T-1$$

（2）适应型矩阵约束 利用适应性矩阵的元素 a_{ij} 与极大值常数 M 相乘使得 x_{ijt}，即第 t 年在地块 j 上的作物 i 受约束,

$$x_{ijt} \leq A_{\text{adjust}}[i,j] \times M, \quad \forall i \in I, \forall j \in J, \forall t \in T$$

（3）豆类作物种植要求 从 2023 年开始，每个地块必须在三年内至少种植一次豆类作物，这是为了利用豆类作物的根瘤改善土壤结构，有利于其他作物的生长，因此有:

$$\sum_{i \in \mathcal{B}} \sum_{t=k}^{k+2} y_{ijt} \geq 1, \quad \forall j \in J, \quad \forall k = 1, 2, 3, \ldots, T-2$$

（4）平旱地、梯田、山坡地的约束 设 G 为粮食类作物的集合, 这三种地块只适合种粮食类作物且只可种植单季，因此有:

$$\sum_{i \in G} y_{ijt} \leq 1, \quad \forall j \in \{\text{平旱地, 梯田, 山坡地}\}, \forall t \in T$$

（5）水浇地的约束 设 R 为水稻作物的集合，V 为蔬菜类作物（除大白菜，白萝卜，红萝卜，豆类）的集合,W 为大白菜，白萝卜，红萝卜的集合，则对于水浇地有:

I. 如果选择种植水稻,

$$\sum_{i \in R} y_{ijt} \leq 1, \quad \sum_{i \in V} y_{ijt} = 0, \quad \sum_{i \in W} y_{ijt} = 0, \quad \forall j \in \{\text{水浇地}\}, \forall t \in T$$

II. 如果选择种植两季蔬菜（除大白菜，白萝卜，红萝卜，豆类），

$$\sum_{i \in V, \mathcal{B}} y_{ijt} \leq 1, \quad \sum_{i \in V, W} y_{ijt} \leq 1, \quad \forall j \in \{\text{水浇地}\}, \forall t \in T$$

（6）普通大棚的约束 对于普通大棚，要确保一年中可以有一季种植蔬菜（大白菜，白萝卜，红萝卜除外）和一季种植食用菌，因此有:

$$\sum_{i \in V, \mathcal{B}} y_{ijt} \leq 1, \quad \sum_{i \in M} y_{ijt} \leq 1, \quad \forall j \in \{\text{普通大棚}\}, \forall t \in T$$

（7）智慧大棚的约束 智慧大棚一年可以种植两季蔬菜（大白菜，白萝卜，红萝卜出来），因此有:

$$\sum_{i \in V, \mathcal{B}} y_{ijt} = 2, \quad \forall j \in \{\text{智慧大棚}\}, \forall t \in T$$

（8）作物种植集中度要求 每种作物每季的种植地块不应过于分散，要确保每种作物只能种在一种地块类型上，因此有

$$z_{ij} \geq y_{ijt}, \quad \forall i \in I, \forall j \in J, \forall t \in T$$

$$\sum_{j \in J_k} z_{ij} \leq 1, \quad \forall i \in I, \forall k$$

其中，k 表示地块类型的集合

（9）种植面积限制 确保每种作物在单个地块或大棚中的种植面积不宜过小

$$x_{ijt} \geq MinArea_i \times y_{ijt}, \quad \forall i \in I, \forall j \in J, \forall t \in T$$

（10）土地面积总量限制 根据耕地数据，乡村总耕地面积分为 1201 亩的露天耕地、16 个普通大棚和 4 个智慧大棚（每个大棚 0.6 亩），确保所有规划的种植面积不得超过各自的地块和大棚总面积，因此有：

$$J_{\text{露天}} = \{\text{干旱地, 梯田, 山坡地, 水浇地}\}$$

$$\sum_{i \in I} \sum_{t \in T} x_{ijt} \leq 1201, \quad \forall j \in J_{\text{露天}}$$

$$\sum_{i \in I} \sum_{t \in T} x_{ijt} \leq 16 \times 0.6, \quad \forall j \in \{\text{普通大棚}\}$$

$$\sum_{i \in I} \sum_{t \in T} x_{ijt} \leq 4 \times 0.6, \quad \forall j \in \{\text{智慧大棚}\}$$

5.2.3 建立目标函数

（1）计算每种作物的实际产量

$$Actual\_Yield_{ijt} = p_{it} \cdot x_{ijt}$$

（2）计算滞销浪费量

$$Waste_{ijt} = \max(0, Actual\_Yield_{ijt} - d_{it})$$

（3）情况一（滞销浪费） 超出的预期销售量的产量，无法售出获得收益

I. 最大化总收益的目标函数

$$Maximize \quad Z_{\text{收益}} = \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} (p_{it} \cdot s_{it} \cdot x_{ijt} - Waste_{ijt} \cdot s_{it} - c_{it} \cdot x_{ijt})$$

II. 最小化浪费的目标函数

$$Minimize \quad Z_{\text{浪费}} = \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} \text{Waste}_{ijt}$$

（4）情况二（降价销售） 超出的预期销售量的产量，按 50% 的价格出售

I. 先计算可原价出售的产量, 得出**最大化总收益的目标函数**

$$SF_{ijt} = \min(\text{Actual\_Yield}_{ijt}, d_{it})$$

$$\text{Maximize} \quad Z_{\text{收益}} = \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} [s_{it} \cdot SF_{ijt} + 0.5 \cdot s_{it} \cdot Waste_{ijt} - c_{it} \cdot x_{ijt}]$$

II.textbf 最小化浪费的目标函数，浪费的部分不计为完全浪费按 0.5 处理

$$\text{Minimize} \quad Z_{\text{浪费}} = \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} 0.5 \cdot Waste_{ijt}$$

5.3 求解结果分析

5.3.1 滞销浪费

（1）资源利用效率分析： 所有地块的**平均利用率为 95.47%**。具体的利用情况可以分为高效利用、适中利用和低效利用三个层次。

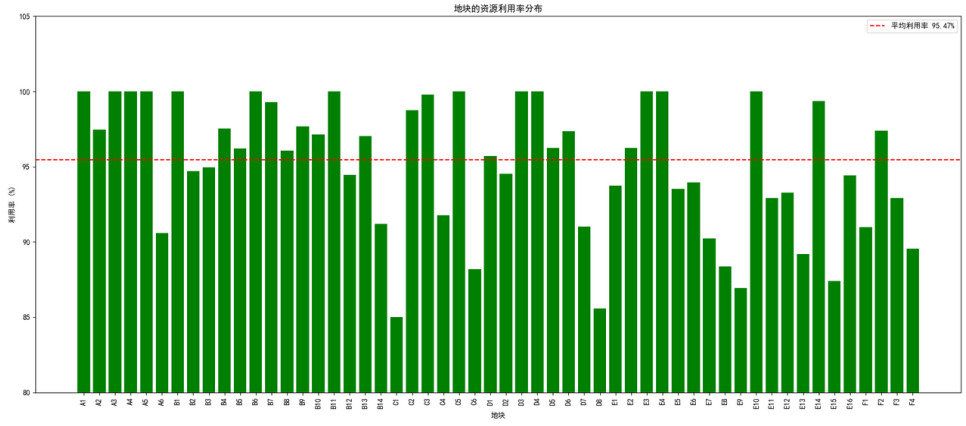

图 7 土地资源利用情况

I. **高利用率地块：**

- 多数地块的利用率**接近 100%**，这表明这些地块被充分利用，资源浪费较少。
- 地块如 **A1, A3, B5, C3** 等表现出极高的利用率，说明这些区域的种植安排非常优化。

II. **低利用率地块：**

- 其中一些地块的利用率明显低于平均线（95.47%），如 A5, B14, D1, D7, E8 等，这些地块的利用率在 85%-90% 之间。
- 这些较低的利用率可能是由于季节性种植安排不当、耕种计划不合理或者地块特性限制（如地形、土壤条件等）。

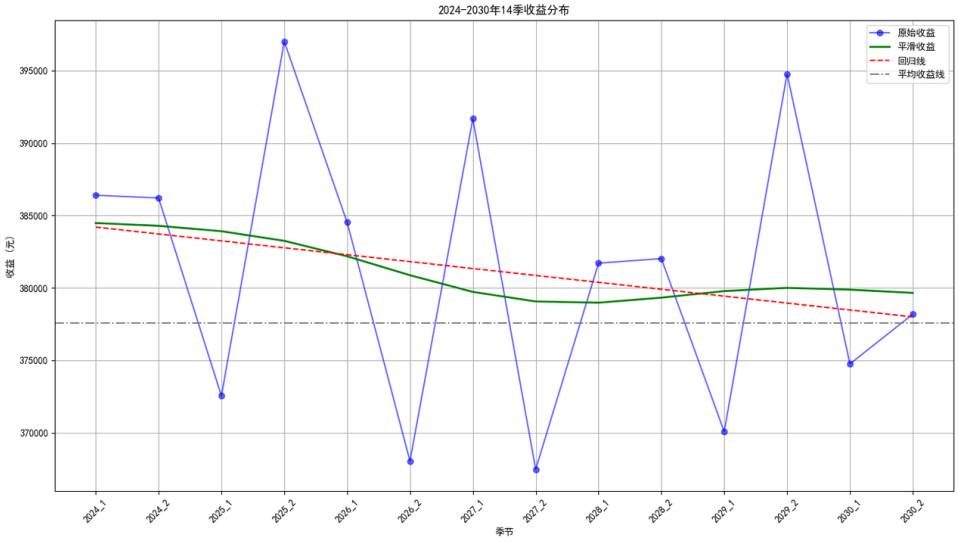

图 8　经济效益情况

（2）经济效益分析：　2024-2030 年总收益 5,286,054.32 元，在 14 个季节中尽管有波动，但整体收益仍保持在较高水平。需要注意季节间的收益差异，积极进行调整以减少波动对全年收益的负面影响。

5.3.2 降价销售

比较分析：

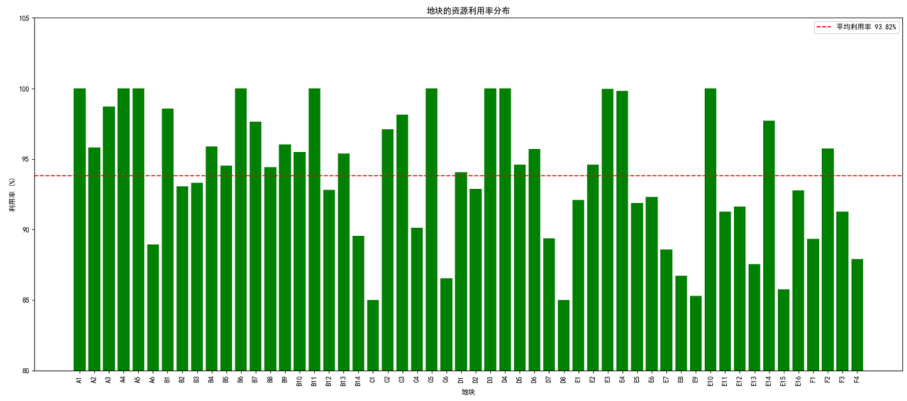

图 9　土地资源利用情况

(1) 资源利用效率分析：　所有地块的平均利用率为 94.82%，略低于上一方案的 95.47%。这表明该方案在整体上资源利用较不充分。

- **种植决策**：滞销浪费方案可能会让农户更加保守地种植以避免浪费，而降价销售方案则激励农户充分利用可用土地，提高种植密度或选择高产量作物。

- **资源管理**：降价销售方案支持更有效的资源管理，鼓励农户进行精细化种植，提升单产，同时也在心理上减少了因过剩而带来的浪费感。

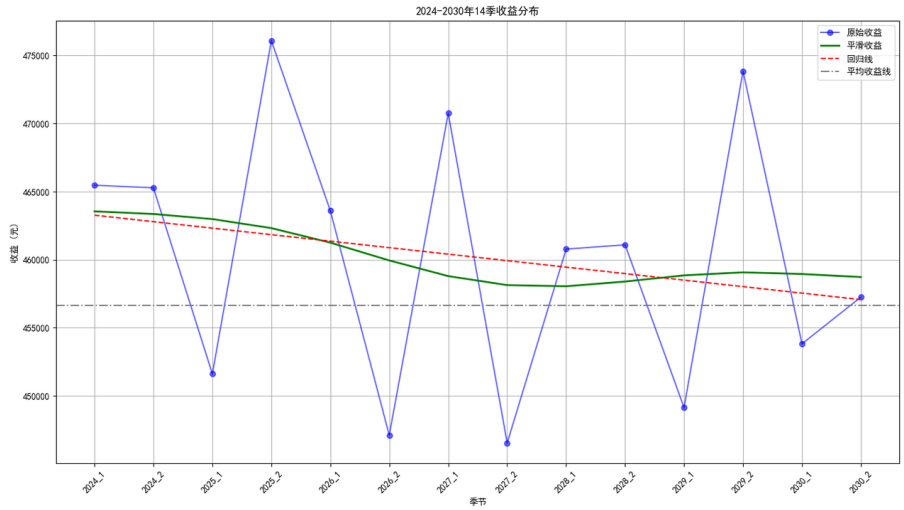

图 10　经济效益情况

（2）经济效益分析： 当前方案的**总收益为 6,393,084.40 元**，当前方案的**总收益比前一个方案高出约 447,030 元（增长约 8.3%）**，表明降价销售策略在整体收益提升方面更为有效。降价销售方案显著减少了，由于产量过剩带来的经济损失，相比于滞销浪费，整体收益更高。虽然收益低于全价销售，但 50% 的收入仍然为农户提供了额外的经济回报。

I. 经济效益与决策影响

- **当前方案**：降价销售降低了部分过剩产量的损失风险，相比于滞销浪费，整体收使农户更愿意增加种植密度和优化地块利用，最终带来了整体经济效益的提升。
- **滞销浪费方案**：滞销浪费使农户在面临过剩风险时可能采取更保守的种植策略，避免了部分资源浪费，但也牺牲了潜在的高收益机会。

六、 问题二的模型的建立和求解

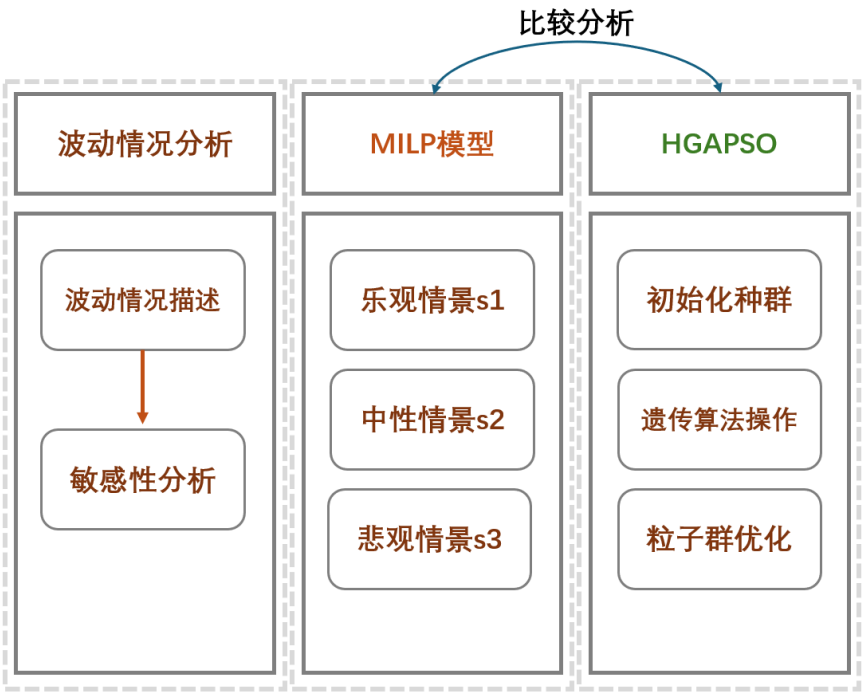

图 11　问题二建模流程图

6.1 数据分析

6.1.1 波动情况描述

（1）预期销售量的波动 -

I. 小麦和玉米的预期销售量增长，年增长率介乎 5%-10%，定义年增长率 $Growth\_Rate_{it}$ 为作物 i 在第 t 年的增长率，因此有：

$$Growth\_Rate_{it} \in [0.05, 0.10]$$

$$d_{it} = d_{i(t-1)} \times (1 + Growth\_Rate_{it}), \quad \forall i \in \{\text{小麦, 玉米}\}, \forall t \in T$$

II. 其他作物的预期销售量波动相对 2023 年波动率，定义波动率 $Fluctuation\_Rate_{it}$ 为作物 i 在第 t 年相对与 2023 年的波动率，因此有：

$$d_{it} = d_{i(2023)} \times (1 + Fluctuation\_Rate_{it}), \quad \forall i \in I \setminus \{\text{小麦, 玉米}\}, \forall t \in T$$

（2）亩产量受气候等因素影响的波动

各作物受气候等因素影响产量，定义 $Yield\_Fluctuation\_Rate_{it}$ 为对于作物 i 在第 t 年相对于其在上一年的变化率，因此有：

$$p_{it} = p_{i(t-1)} \times (1 + Yield\_Fluctuation\_Rate_{it}), \quad \forall i \in I, \forall t \in T$$

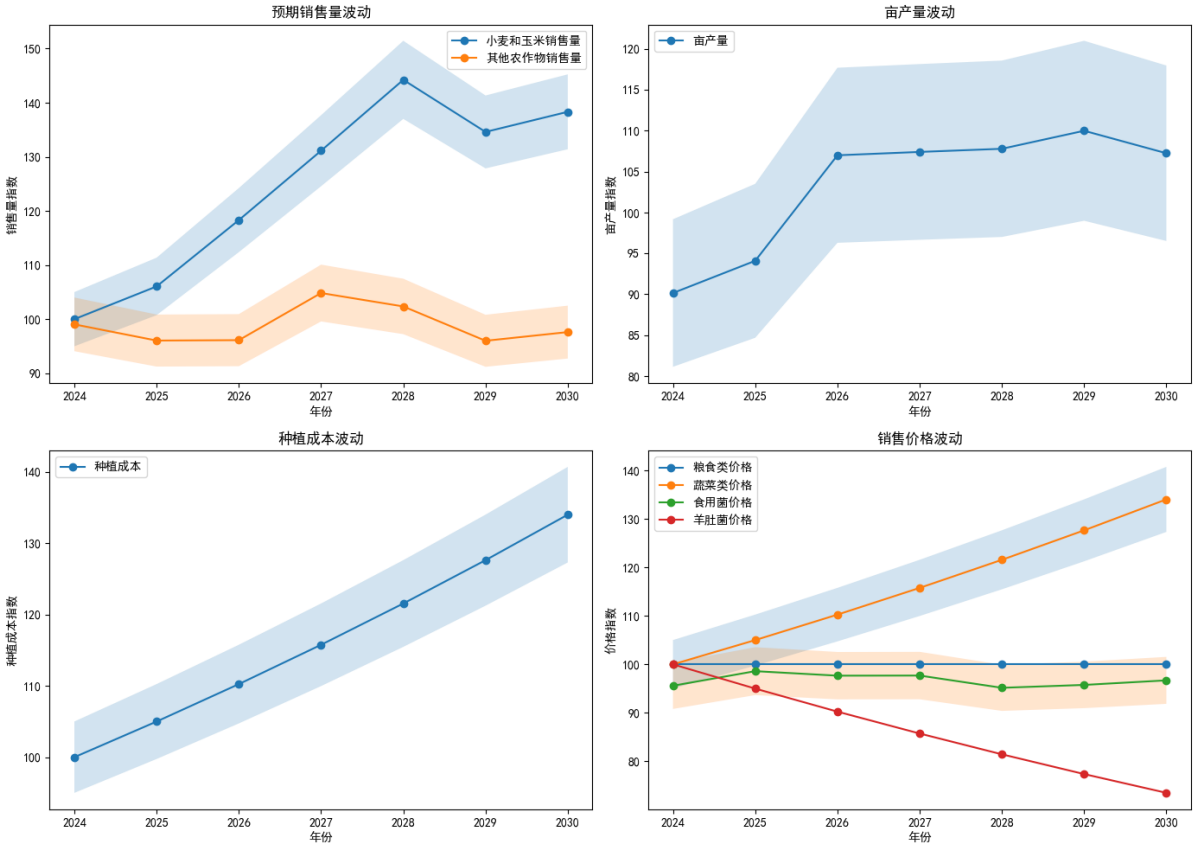

图 12　农作物各参数波动情况

（3）种植成本的年增长

种植成本的年增长约为 5%，定义 $Cost\_Growth\_Rate_{it}$ 为作物 i 在第 t 年的增长率，因此有：$Cost\_Growth\_Rate_{it} \approx 0.05$。

$$c_{it} = c_{i(t-1)} \times (1 + Cost\_Growth\_Rate_{it}), \quad \forall i \in I, \forall t \in T$$

（4）销售价格的波动

I. 粮食类作物的价格稳定，因此有

$$s_{it} = s_{i(2023)}, \quad \forall i \in G, \forall t \in T$$

II. 蔬菜类作物的价格增长，定义 $Vegetable\_Price\_Growth\_Rate_{it}$ 为作物 i 在第 t 年相对于其去年的增长率，因此有

$$s_{it} = s_{i(t-1)} \times (1 + Vegetable\_Price\_Growth\_Rate_{it}), \quad \forall i \in V, W, \mathcal{B} \quad , \forall t \in T$$

III. 食用菌的价格下降，定义 $Mushroom\_Price\_Decline\_Rate_{it}$ 为作物 i 在第 t 年的价格相对于其去年的下降率，因此有：

$$s_{it} = s_{i(t-1)} \times (1 + Mushroom\_Price\_Decline\_Rate_{it}), \quad \forall i \in S, \forall t \in T$$

6.1.2 敏感性分析

（1）单变量敏感度分析

按 10% 降低和提高的方式调整每个变量，保持其他变量不变，得到每次调整后的收益及其与基准收益的变化率。通过比较每个变量的变化率，识别出对收益影响最大的变量，即最敏感的变量。根据新的基准收益为 **777,220.47** 元进行敏感性分析的结果分析。以下是对新的基准收益和敏感性分析结果的解读:

I. 基准收益分析

基准收益为 **777,220.47 元**，这表示在当前条件下，农业种植活动的收益是正的，意味着在现有的种植成本、销售价格和产量情况下，整体策略是盈利的。

II. 亩产量的敏感性分析

情况一：调整至 90%：收益变化为 **771,053.05 元**，变化率为 **-0.80%**。

情况二：调整至 110%：收益变化为 **783,387.88 元**，变化率为 **0.80%**。

亩产量的变化对收益的影响较小，收益变化率在 ±0.80% 的范围内波动。这表明当前策略对产量的敏感度较低，产量波动对整体收益影响有限，种植决策相对稳健。

III. 种植成本的敏感性分析

情况一：调整至 90%：收益变化为 **872,979.71 元**，变化率为 **12.33%**。

情况二：调整至 110%：收益变化为 **681,461.22 元**，变化率为 **-12.33%**。

种植成本对收益影响显著，当种植成本降低 10% 时，收益提高了 12.33%；反之，当种植成本增加 10% 时，收益减少了 12.33%。这表明种植成本是收益的关键影响因素。成本的微小变化会导致收益的大幅波动，因此控制种植成本是提升收益的关键策略。

IV. 销售价格的敏感性分析

情况一：调整至 90%：收益变化为 **771,053.05 元**，变化率为 **-0.80%**。

情况二：调整至 110%：收益变化为 **783,387.88 元**，变化率为 **0.80%**。

销售价格的变化对收益的影响与亩产量类似，变化率在 ±0.80% 范围内波动。销售价格的调整对收益影响较小，这意味着在当前市场条件下，价格的波动对整体利润影响不大，可能由于市场价格较为稳定或这价格波动的影响被其他因素，比如敏感性较强的成本因子所掩盖。

（2）多变量敏感度分析

图中揭示了销售价格和亩产量的联动对收益影响的敏感性。提高销售价格和产量对

收益有显著的正面作用，而双重减少则会严重侵蚀利润。对于农业决策，重点在于如何稳定产量和优化销售价格，以最大限度地提升收益，规避市场和产量波动的负面影响。

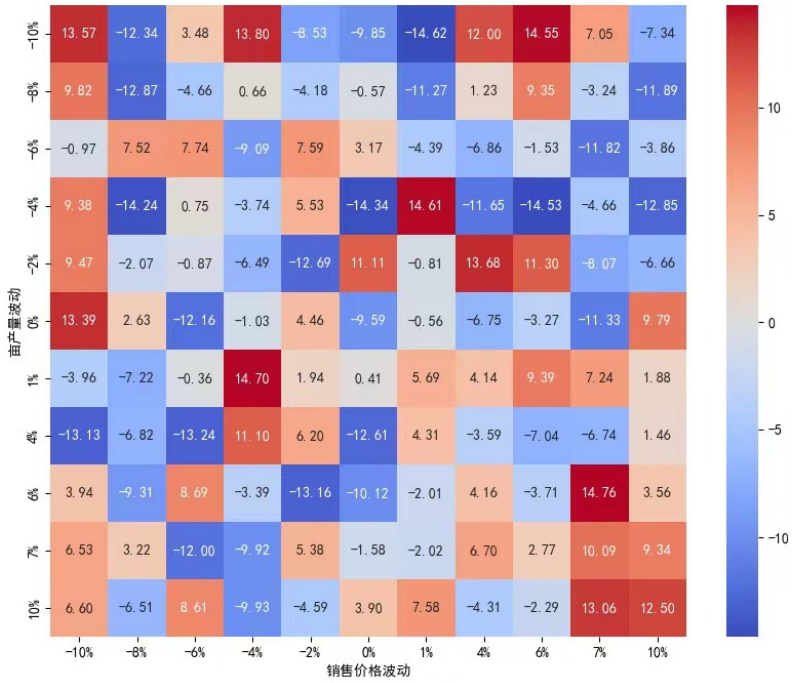

图 13　多变量联合波动对收益的影响情况热力图

6.2 模型建立与求解

6.2.1 MILP 规划模型模型建立

（1）引入不确定性集

设情景集 $\mathcal{O}$ 包含多个情景，每个情景用一个概率 π_o 表示其发生的概率（总和为 1），情景集的公式表达如下：

$$\mathcal{O} = \{o_1, o_2, \ldots, c_N\}$$

每个情景 $o \in \mathcal{O}$ 包含可能的市场变化组合，如：

I. 销售价格下降 10%

II. 种植成本上身 15%

III. 亩产量减少 15%

（2）目标函数

目标函数将优化问题扩展为所有情景下的加权和，公式如下：

$$Actual\_Yield^{o}_{ijt} = p^{o}_{it} \cdot x_{ijt}$$

$$Waste_{ijt}^{o} = \max(0, Actual\_Yield_{ijt}^{o} - d_{it})$$

$$\text{Maximize} \quad Z = \sum_{o \in \mathcal{O}} \pi_o \left(\sum_{t \in T} \sum_{i \in I} \sum_{j \in J} \left(p_{it}^{o} \cdot x_{ijt} \cdot s_{it}^{o} - Waste_{ijt}^{o} \cdot x_{ijt} - c_{it}^{o} \cdot x_{ijt} \right) \right)$$

π_o：情景 o 的发生概率

（3）实施情景优化

定义多个情景，分别设定销售价格、种植成本和亩产量的变化比例。这里将市场变化分为三种标志性情景：乐观、中性和悲观情景。每个情景反映了市场的不同状态。

	o_1（乐观情景）	o_2（中性情景）	o_3（悲观情景）
预期销售量	小麦玉米的增长率为 10%（接近上限），其他农作物增长率为 5%。	小麦和玉米：增长率为 7.5%（介于中间），其他农作物：变化率为 0%（与 2023 年持平）	小麦和玉米增长率为 5%（接近下限），其他农作物：下降 5%
亩产量	增加 10%	无变化（±0%）	减少 10%
种植成本	增长 3%（低于平均增长率）	增长 5%（符合平均增长率）	增长 7%（高于平均增长率）
销售价格	粮食类作物保持稳定，蔬菜类作物：增长 7%（高于平均增长率），食用菌：价格下降 1%（降幅较小），羊肚菌下降 5%	粮食类作物保持稳定，蔬菜类作物增长 5%，食用菌：价格下降 3%，羊肚菌下降 5%	粮食类作物：保持稳定，蔬菜类作物增长 3%（低于平均增长率），食用菌价格下降 5%，羊肚菌下降 5%。

表 1　三种情景

概率分配：

乐观情景（o_1）概率：$\pi_{o1} = 0.2$

中性情景（o_2）概率：$\pi_{o2} = 0.6$

悲观情景（o_3）概率：$\pi_{o3} = 0.2$

（3）约束条件

在原约束条件上，增加了对预期销售量的限制：

$$\sum_{j \in J} p_{it}^{o} \cdot x_{ijt} \leq d_{it}, \quad \forall i \in I, \forall t \in T, \forall o \in \mathcal{O}$$

6.2.2 混合粒子群算法模型 (HGAPSO) 建立

（1）初始化种群

设置种群的每个个体代表一个可能的种植决策组合。每个个体包含种植面积 x_{ijt} 和是否

种植的决策变量 y_{ijt}。

I. 初始种群：

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$$

II. 每个个体：

$$\mathbf{x}_i = (x_{ijt}, y_{ijt}) \quad \forall i \in I, j \in J, t \in T$$

（2）适应性评估

对每个个体 $\mathbf{x}_i$，定义收益 $f(\mathbf{x}_i)$ 作为适应度值，即目标函数为：

$$f(\mathbf{x}_i) = \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} (p_{it} \cdot s_{it} \cdot x_{ijt} - Waste_{ijt} \cdot s_{it} - c_{it} \cdot x_{ijt})$$

（3）遗传算法操作

I. 选择概率：

根据适应度值选择个体进行交叉和变异的概率：

$$P(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)}{\sum_{k=1}^{N} f(\mathbf{x}_k)}$$

II. 交叉操作：

对选择的个体对进行交叉操作，生成新个体。

$$x_{ijt}^{\text{new}} = \alpha \cdot x_{ijt}^{i} + (1 - \alpha) \cdot x_{ijt}^{j}, \quad y_{ijt}^{\text{new}} = \text{round}(\alpha \cdot y_{ijt}^{i} + (1 - \alpha) \cdot y_{ijt}^{j})$$

其中，$\alpha \in [0, 1]$ 为交叉率，$\mathbf{x}_i$ 和 $\mathbf{x}_j$ 为选择的两个父个体

III. 变异操作：

随机改变个体的某些部分以增加种群的多样性：

$$x_{ijt}^{\text{new}} = x_{ijt} + \beta \cdot \text{rand}(-1, 1), \quad y_{ijt}^{\text{new}} = \text{round}(y_{ijt} + \beta \cdot \text{rand}(-1, 1))$$

其中，β 是变异率，rand(-1, 1) 是 $(1, -1)$ 之间的随机值

（4）粒子群优化

将遗传算法生成的新个体作为初始种群进行粒子群优化。

I. 初始化：

粒子的位置初始化为遗传算法的输出，速度初始化为随机值。

粒子位置:

$$\mathbf{x}_i = (x_{ijt}, y_{ijt})$$

粒子速度:

$$\mathbf{v}_i = (v_{ijt}, u_{ijt})$$

其中，v_{ijt} 是种植面积速度,u_{ijt} 是种植决策速度。

II. 速度和位置更新：

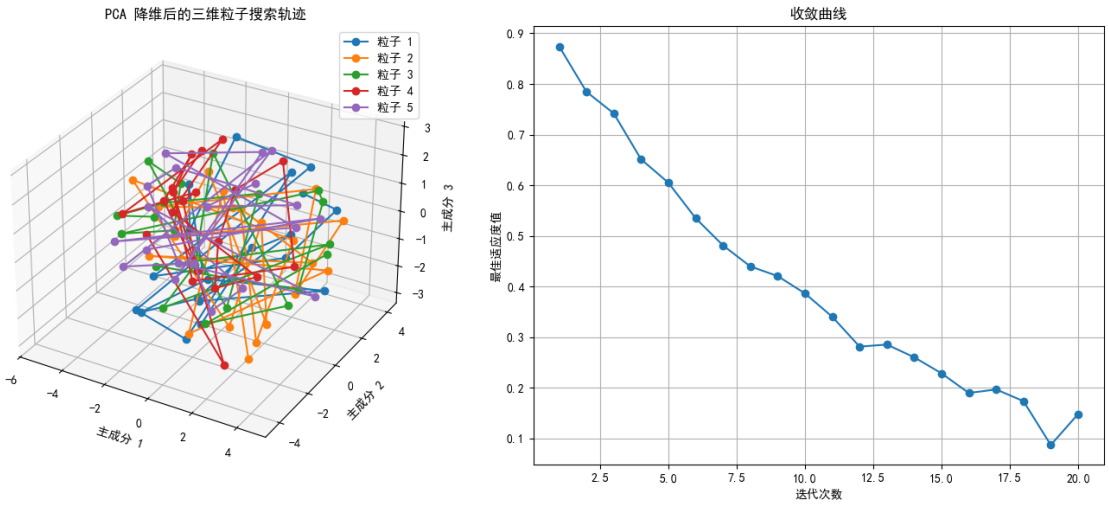

图 14　粒子搜索轨迹以及收敛曲线

在每个迭代中，更新粒子的速度和位置的公式为：

$$u_{ijt}(t+1) = \omega \cdot u_{ijt}(t) + c_1 \cdot r_1 \cdot (p_{ijt} - y_{ijt}(t)) + c_2 \cdot r_2 \cdot (g_{ijt} - y_{ijt}(t))$$

$$x_{ijt}(t+1) = x_{ijt}(t) + v_{ijt}(t+1), \quad y_{ijt}(t+1) = \text{round}(y_{ijt}(t) + u_{ijt}(t+1))$$

其中，p_{ijt} 是粒子 i 在地块 j 和时间 t 的历史最佳位置。g_{ijt} 是全局最佳位置。

III. 适应度更新：

计算每个粒子的适应度值，更新个人最佳位置 p_{ijt} 和全局最佳位置 g_{ijt}。

如果 $f(x_{ijt}(t+1)) > f(p_{ijt})$，则

$$p_{ijt} = x_{ijt}(t+1)$$

如果 $f(x_{ijt}(t+1)) > f(g_{ijt})$，则

$$g_{ijt} = x_{ijt}(t+1)$$

（5）迭代与终止

重复遗传与粒子群优化步骤，直到满足收敛条件，即最大迭代次数或适应度变化不明显

6.3 求解结果分析

6.3.1 混合整数线性规划（MILP）求解分析：

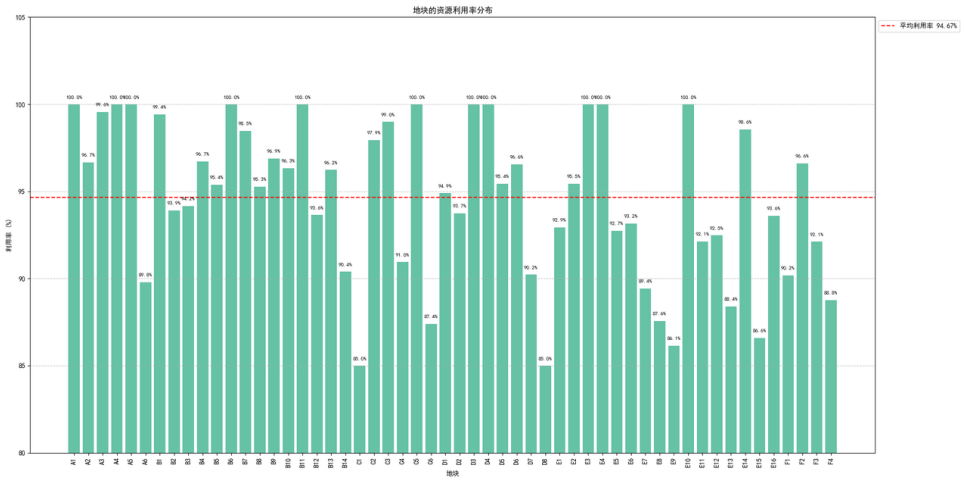

图 15　土地资源利用情况

（1）资源利用效率分析　**平均利用率为 94.67%**，显示出整体资源利用效率较高。大部分地块的利用率都在 90% 以上，这表明种植方案在资源分配上较为合理，大部分地块得到了充分利用。多个地块的利用率接近或达到 100%，如 A1、A3、B5、B6 等，表明这些地块被高效利用，可能是由于适合的作物选择、良好的种植管理以及市场需求的匹配。

地块间利用率存在一定的波动，但多数地块仍保持在较高的利用水平。优化管理可以减少波动幅度，进一步提升整体资源利用率。

（2）经济效益分析

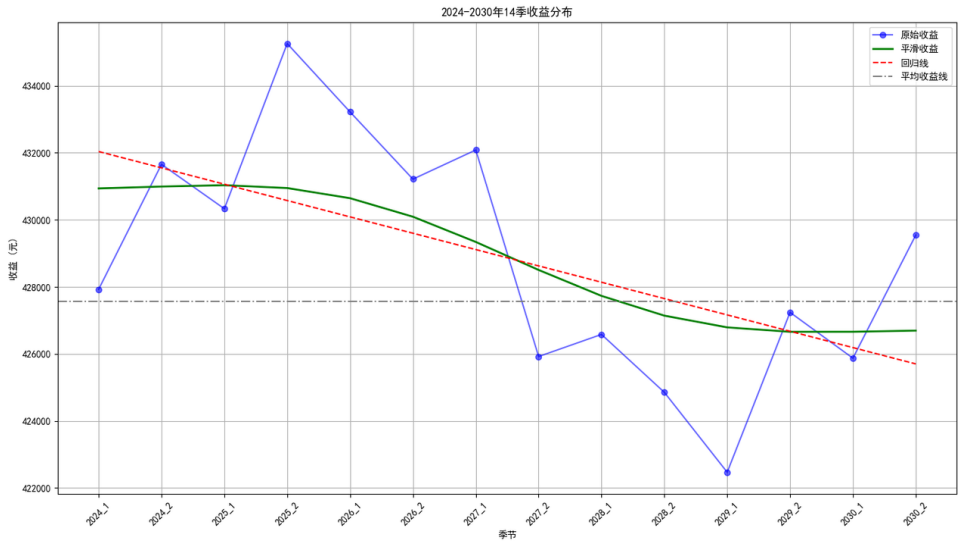

图 16　经济效益情况

I. 总体收益分析：

总收益为 5,986,032.29 元，分布在 14 个季节中，显示出较为平稳但略低的收益水平。相较于其他方案，这一总收益略有减少，反映了当前方案在市场波动或管理调整上的挑战。

平均收益线（灰色虚线位于约 428000 元的水平，显示整体收益相对稳定但处于较低的水平。

II. 趋势分析：

- **平滑收益（绿色曲线）**：显示出收益在整体上有下行趋势，但曲线相对平稳，波动被部分削弱。
- **回归线（红色虚线）**：明确展示出收益的下降趋势，暗示随着时间推移，整体收益面临持续下降的风险。这可能与市场需求疲软、成本上升或其他不利因素有关。

III. 影响因素分析：

- **市场价格和需求**：粮食类作物价格稳定，但蔬菜类作物价格增长和食用菌价格下滑的预期可能影响整体收益。此外，销售量的增长未能完全抵消价格或产量的波动。
- **生产条件与成本**：随着种植成本平均每年增长 5% 左右，成本的上涨可能削弱了收益的增长潜力。产量波动也加剧了收益的不确定性，特别是在低谷季节，收益显著下降。

6.3.2 HGAPSO 求解分析：

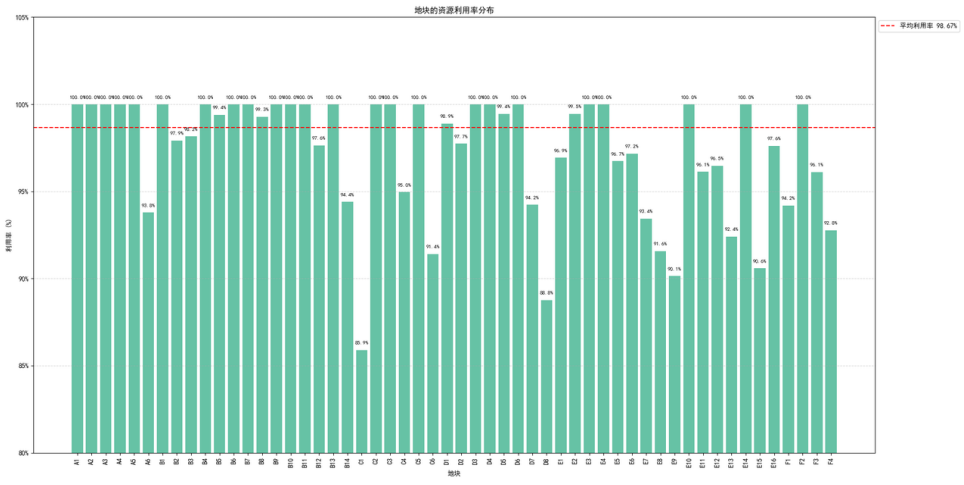

图 17　土地资源利用情况

（1）资源利用效率分析

- **平均利用率为 98.67%**，表明地块的资源利用率非常高，接近满负荷运作。这个高平均值显示出当前种植方案在地块资源配置上的高效性。
- 大多数地块的利用率都接近或达到 100%，反映出种植方案对每块地的使用做到了尽可能的最大化，有效避免了资源的浪费。

（2）经济效益分析

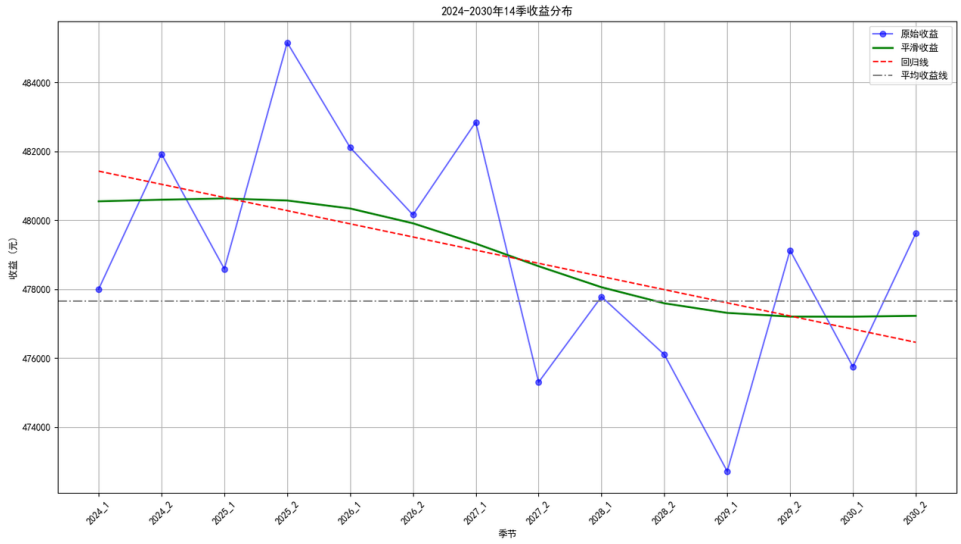

图 18　经济效益情况

I. 总体收益分析：

总收益为 6687012.53 元，分布在 14 个季节中，显示出较为平稳但略低的收益水平，收益在各个季节间存在显著波动，峰值和谷值交替出现，显示出较大的季节性波动性。

平均收益线（灰色虚线）：大约位于每季 478,000 元的水平，显示出整体收益在相对稳定的范围内波动。

II. 趋势分析：

- **平滑收益（绿色曲线）**：显示出收益随着时间推移逐渐减弱的趋势，尽管整体波动较小，但下行趋势明显。
- **回归线（红色虚线）**：进一步确认了整体收益的下降趋势，表明未来几年收益可能面临挑战，逐渐趋于平稳甚至下滑的状态。

七、 问题三的模型的建立和求解

7.1 相关性分析

预期销售量、销售价格与种植成本之间存在一定的相关性，通过计算不同参数间的**皮尔逊相关系数**，可以分析它们之间的相关性。

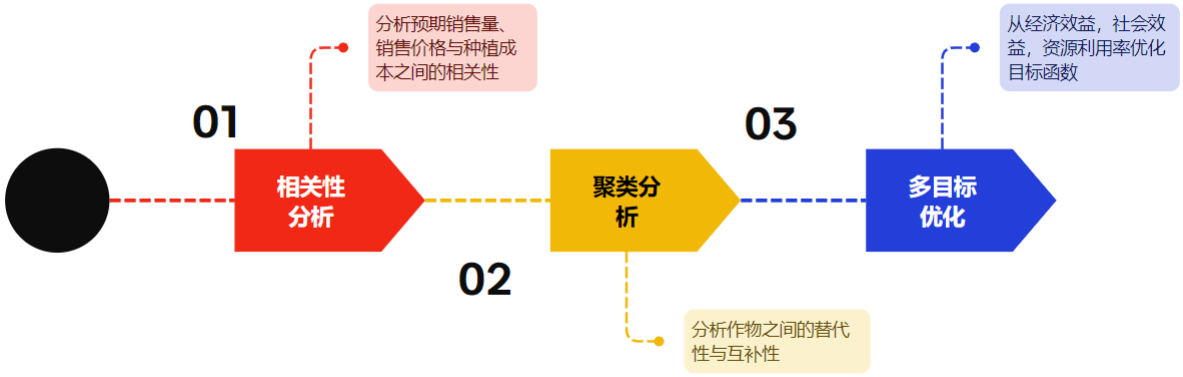

图 19　问题三模型建立流程图

7.1.1 亩产量 p_{it} 和种植成本 c_{it} 的相关系数 r_{pc}

$$r_{pc} = \frac{\sum(p_{it} - \bar{p})(c_{it} - \bar{c})}{\sqrt{\sum(p_{it} - \bar{p})^2} \cdot \sqrt{\sum(c_{it} - \bar{c})^2}}$$

其中，$\bar{p}$ 和 $\bar{c}$ 分别是亩产量和种植成本的均值。

7.1.2 亩产量 p_{it} 和销售价格 s_{it} 的相关系数 r_{ps}

$$r_{ps} = \frac{\sum(p_{it} - \bar{p})(s_{it} - \bar{s})}{\sqrt{\sum(p_{it} - \bar{p})^2} \cdot \sqrt{\sum(s_{it} - \bar{s})^2}}$$

其中，$\bar{s}$ 是销售价格的均值。

7.1.3 种植成本 c_{it} 和销售价格 s_{it} 的相关系数 r_{cs}

$$r_{cs} = \frac{\sum(c_{it} - \bar{c})(s_{it} - \bar{s})}{\sqrt{\sum(c_{it} - \bar{c})^2} \cdot \sqrt{\sum(s_{it} - \bar{s})^2}}$$

7.1.4 分析变量之间的相关性

（1）亩产量与种植成本

亩产量与种植成本之间的相关系数为 0.5343，属于中等程度的正相关。

从散点图来看，亩产量和种植成本之间有一定的正相关趋势，即随着亩产量增加，种植成本也有增加的趋势。

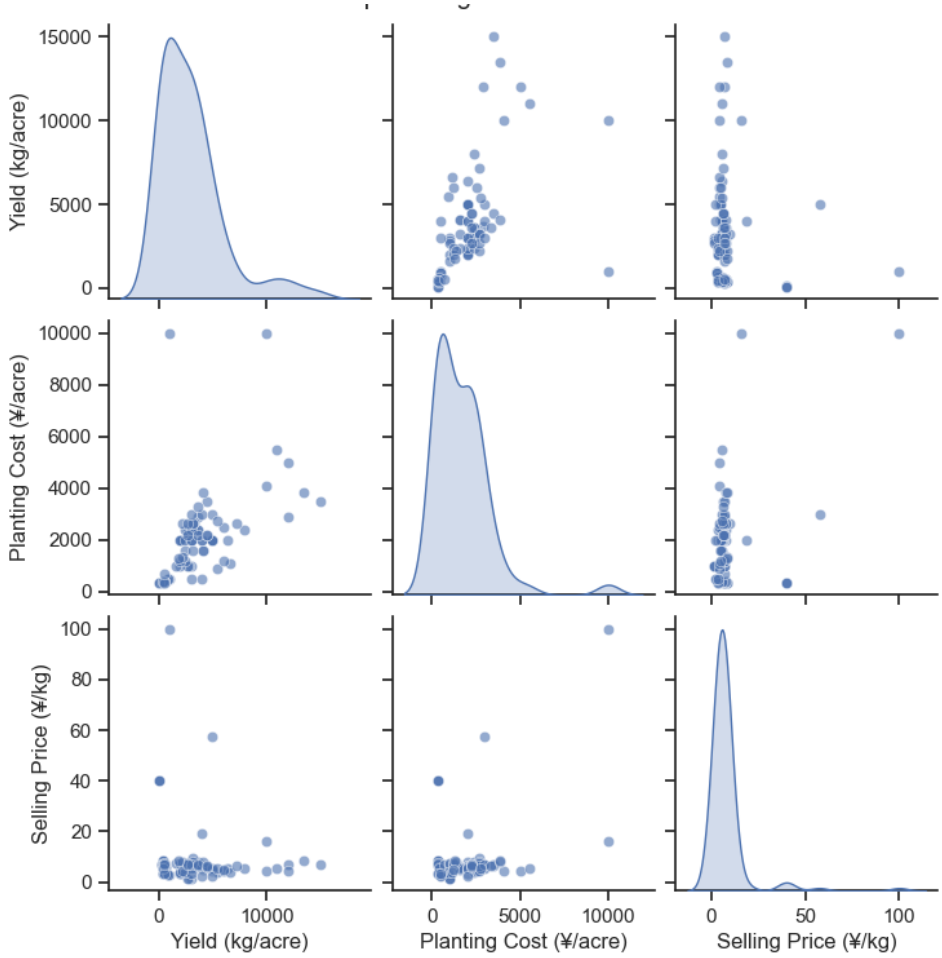

图 20　种植成本、亩产量和销售价格成对关系图

（2）亩产量与销售价格

亩产量与销售价格之间的相关系数为 0.1213，属于弱相关或无明显相关。

散点图显示，亩产量和销售价格之间的相关性较弱，数据点散布较广，没有明显的线性趋势。

（3）种植成本与销售价格

种植成本与销售价格之间的相关系数为 0.1139，几乎没有相关性。

种植成本与销售价格之间的散点图也没有明显的趋势，数据点分布较为随机。这种随机分布表明种植成本对销售价格没有明显的影响。

7.2 聚类分析

7.2.1 构建特征矩阵

每个作物 i 在第 t 年的特征向量可以用一个包含了亩产量、销售价格和种植成本三维向量表示：

$$\mathbf{x}_{it} = [p_{it}, s_{it}, c_{it}]$$

7.2.2 选择聚类数 (k)

通过计算不同 k 值下的**聚类代价函数，即簇内误差平方和 (SSE)**，确定肘部来确定合适的聚类数。

SSE 的计算公式为

$$\text{SSE} = \sum_{n=1}^{k} \sum_{\mathbf{x}_{it} \in C_n} \|\mathbf{x}_{it} - \boldsymbol{\mu}_n\|^2$$

其中:

C_j 是第 j 个簇,

$\mathbf{x}_{it}$ 是属于簇 C_n 的数据点,

$\boldsymbol{\mu}_j$ 是簇 C_n 的质心,

$\|\mathbf{x}_{it} - \boldsymbol{\mu}_n\|^2$ 表示 $\mathbf{x}_{it}$ 与质心 $\boldsymbol{\mu}_n$ 之间的欧几里得距离的平方。

图中的 **“肘部”** 点，即 SSE 的减少趋势开始变得平缓的位置，因此这个 “肘部” 点对应的 k 值即为最佳聚类数。

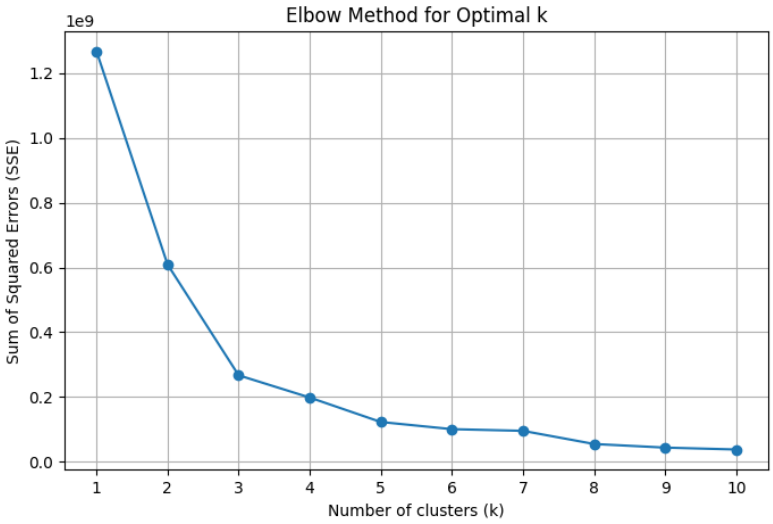

图 21　聚类代价函数图

7.2.3 初始聚类中心的选择

随机选择 k 个初始中心点（质心），这些中心点也是三维向量，对应三个特征，表示为：

$$\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$$

7.2.4 分配数据点到最近的质心

对于每个作物特征向量 $\mathbf{x}_{it}$，计算它到所有质心的欧几里得距离，选择距离最近的质心作为其所属的簇。距离的计算公式为：

$$d(\mathbf{x}_{it}, \boldsymbol{\mu}_n) = \sqrt{(p_{it} - \mu_{n1})^2 + (s_{it} - \mu_{n2})^2 + (c_{it} - \mu_{n3})^2}$$

其中，μ_{n1}、μ_{n2} 和 μ_{n3} 分别是第 n 个质心在亩产量、销售价格和种植成本维度上的值。

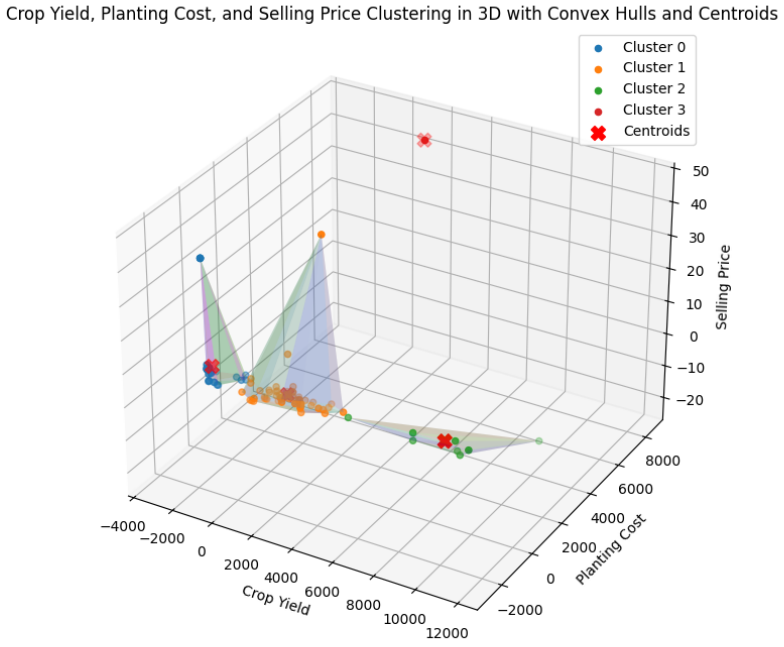

图 22　作物聚类分析图

7.2.5 更新质心位置

将每个簇中的点重新计算平均值，作为新的质心。新质心的计算公式为：

$$\boldsymbol{\mu}_n = \frac{1}{|C_n|} \sum_{\mathbf{x}_{it} \in C_n} \mathbf{x}_{it}$$

其中，$|C_n|$ 是指簇 n 中的点的数量。

7.2.6 重复操作至完成聚类

重复步骤 7.2.4 和 7.2.5，直到质心位置不再发生显著变化，或者达到预定的迭代次数时，算法收敛，聚类完成。

7.2.7 聚类结果分析

经过聚类，**相同簇内的作物具有较高的替代性**，因为它们在产量、成本和价格上相似。例如，图 22 中靠近相同簇的作物数据点表明它们在**经济特性上相近**，因此可以在种植决策中互相替代。

距离较远且具有不同特征的簇间的作物可能具有互补性。例如，**Cluster 0 中的一些作物与 Cluster 3 的作物之间存在互补性**，因为它们的产量、成本和销售价格特征明显不同，因此可以利用该性质的作物组合优化收益。

7.3 模型建立和求解

（1）定义参数与变量

I. 用于计算经济效益：

δ_{ijt}：表示第 t 年地块 j 上作物 i 替代作物 j 的程度，用于表示可替代性（单位：百分比）。

$Comp_{ij}$：作物 i 和作物 j 的互补性系数（无单位）。

Sub_{ij}：作物 i 和作物 j 的可替代性系数（无单位）。

II. 用于计算社会效益：

SE_{it}：作物 i 在第 t 年的社会效益指标（单位：社会效益分数/亩）。

Y_{it}：作物 i 在第 t 年的优先种植权重系数，用于表示社会对该作物的种植优先级（无单位）。

$Labor_{it}$：作物 i 在第 t 年的单位劳动力需求（单位：人力/亩），反映作物种植对就业的贡献。

$FoodSecurity_{it}$：作物 i 在第 t 年对粮食安全的贡献（无单位），用于衡量该作物对粮食供给稳定性的影响。

$Ecosystem_{it}$：作物 i 在第 t 年对生态系统的贡献（无单位），例如对土壤改良、减少农药使用等生态效益。

III. 用于计算资源使用效率：

r_{ijt}：第 t 年在地块 j 上种植作物 i 的资源投入量（单位：资源单位）。

P_{it}：作物 i 在第 t 年的单位面积资源使用量（单位：资源单位/亩）。

（2）修正目标函数

新的目标可以设定为以下三个子目标的加权组合

I. 经济效益最大化

Maximize $Z_1 =$

$$\sum_{t=2024}^{2030} \sum_{i \in I} \sum_{j \in J} ((p_{it} \cdot s_{it} \cdot x_{ijt} - Waste_{ijt} \cdot s_{it} - c_{it} \cdot x_{ijt}) + Comp_{ij} \cdot z_{ijt} - Sub_{ij} \cdot \delta_{ijt} \cdot x_{ijt})$$

II. 社会效益最大化

$$\text{Maximize } Z_2 = \sum_{t=2024}^{2030} \sum_{i \in I} \sum_{j \in J} (SE_{it} \cdot x_{ijt} \cdot Y_{it})$$

III. 资源使用率最大化

$$\text{Maximize } Z_3 = \frac{\text{总产出}}{\text{总资源投入}}$$

其中，

$$\text{总产出} = \sum_{t=2024}^{2030} \sum_{i \in I} \sum_{j \in J} (p_{it} \cdot s_{it} \cdot x_{ijt})$$

$$\text{总资源投入} = \sum_{t=2024}^{2030} \sum_{i \in I} \sum_{j \in J} (P_{it} \cdot x_{ijt})$$

7.4 求解结果分析

7.4.1 资源利用效率分析

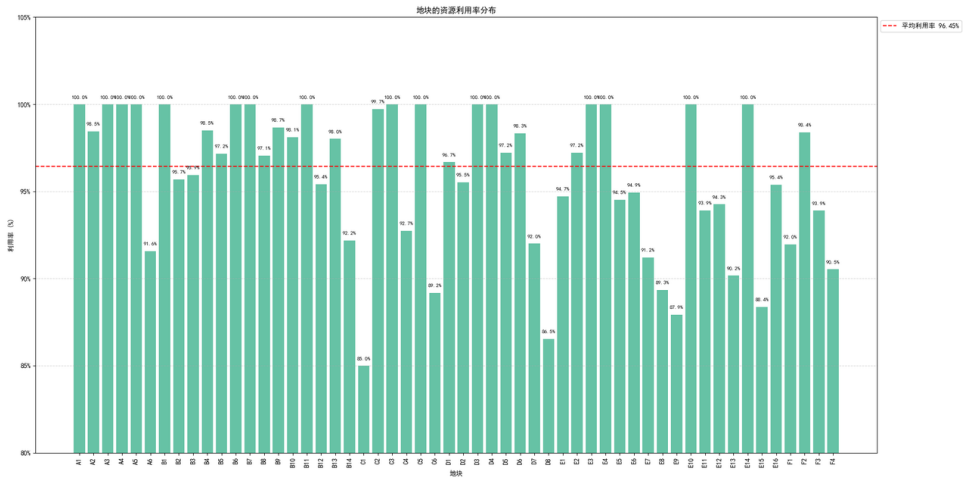

图 23　土地资源利用情况

I. 平均利用率为 96.45%，显示出整体资源利用效率非常高。这表明大部分地块被充分利用，种植方案在资源配置上的表现较为出色。

II. 多数地块的利用率都在 **95%** 以上，许多地块甚至达到了 **100%** 的利用率，反映了当前方案在最大化地块使用方面的有效性。

7.4.2 经济效益分析

图 24　经济效益情况

（1）总体效益分析

I. 总体效益分析

总收益为 6,586,032.29 元，分布在 14 个季节中。整体来看，收益存在明显的波动，且在一些时间段内收益有所下降。**平均收益线（灰色虚线）**位于 476,000 元左右，显示出一个较为稳定的平均收益水平，但仍存在波动和下滑的趋势。

（2）趋势分析

- **平滑收益（绿色曲线）**：显示出一个逐渐上升后又下滑的趋势。初期收益有所上升，表明在 2024-2025 年间收益较为稳定并有所增长，但随后出现了逐渐下行的趋势。
- **回归线（红色虚线）**：显示整体收益的下降趋势。这可能与市场价格波动、生产成本上升或者种植策略的变化相关，表明未来收益的持续性面临挑战。

八、 模型的评价

8.1 模型的优点

• 优点 1：**多目标优化**模型综合考虑了**经济效益、社会效益和资源使用**效率三个方面，通过多目标优化实现了种植策略的全面性和综合性，有助于平衡不同目标之间的矛盾。

• 优点 2：**适应性强**：模型引入了情景分析，能够**适应未来市场条件**、气候变化等不确定因素的影响，提高了模型的鲁棒性和适应性，能够为决策提供更可靠的依据。

• 优点 3：**综合方法的应用**：结合了**混合整数线性规划（MILP）**、**混合粒子群优化算法（HGAPSO）**等多种优化技术，充分发挥了不同方法的优势，提升了求解效率和解的质量。

8.2 模型的缺点

• 缺点 1：**参数敏感性高**：敏感性分析表明种植成本对模型结果有显著影响，因此在实际应用中，需要确保对成本的准确估计，否则可能会导致种植方案不具有实际可操作性。

• 缺点 2：**复杂度较高**：由于模型结合了多种优化方法和算法，整体复杂度较高，对计算资源和时间要求较大，可能在大规模数据或实际操作中面临计算效率问题。

参考文献

[1] 司守奎, 孙玺菁. 数学建模算法与应用[M]. 北京: 国防工业出版社, 2011.

[2] 卓金武. MATLAB 在数学建模中的应用[M]. 北京: 北京航空航天大学出版社, 2011.

[3] BALTAS N C, KORKA O. Modelling farmers' land use decisions[J/OL]. Applied Economics Letters, 2002, 9(7):453-457. DOI: 10.1080/13504850110091886.

[4] ANNETTS J, AUDSLEY E. Multiple objective linear programming for environmental farm planning[J/OL]. Journal of the Operational Research Society, 2002, 53:933-943. DOI: 10.1057/palgrave.jors.2601404.

[5] RUPNIK R, KUKAR M, VRAčAR P, et al. Agrodss: A decision support system for agriculture and farming[J/OL]. Computers and Electronics in Agriculture, 2018, 149:123-132. https://doi.org/10.1016/j.compag.2018.04.001.

[6] ZHAI Z, MARTíNEZ J F, BELTRAN V, et al. Decision support systems for agriculture 4.0: Survey and challenges[J/OL]. Computers and Electronics in Agriculture, 2020, 170: 105256. https://doi.org/10.1016/j.compag.2020.105256.

[7] DURY J, SCHALLER N, GARCIA F, et al. Models to support cropping plan and crop rotation decisions: A review[J/OL]. Agronomy for Sustainable Development, 2012, 32: 567-580. https://doi.org/10.1007/s13593-011-0037-x.

附录 A　文件列表

文件名	功能描述
问题一求解代码.py	问题一程序代码
问题二求解代码.py	问题二程序代码
问题三求解代码.py	问题三程序代码

附录 B　代码

问题一求解代码.py

```
import os

import pandas as pd
from pulp import LpMaximize, LpProblem, LpVariable, lpSum,
   LpStatus, value
from scipy.optimize import linprog
import numpy as np
import random
# 读取Excel文件中的Sheet1
file_path = "D:\\2024国赛\\问题一\\A_adjust 的副本.xlsx"
sheet1_data = pd.read_excel(file_path, sheet_name='Sheet1')
sheet2_data = pd.read_excel(file_path, sheet_name='Sheet2')
# 替换数据中的 NaN 和 inf 为 0
sheet1_data = sheet1_data.replace([np.nan, np.inf, -np.inf],
   0)
sheet2_data = sheet2_data.replace([np.nan, np.inf, -np.inf],
   0)

####集合####
# 定义作物集合 I，包括所有作物的名称
I = sheet1_data['作物'].tolist()
# 定义粮食类作物集合 G
G = sheet1_data[sheet1_data['作物类型'] == '粮食']['作物'].
   tolist()
# 定义蔬菜类作物集合 V
```

```
V = sheet1_data[sheet1_data['作物类型'] == '蔬菜']['作物'].tolist()
# 定义食用菌类作物集合 M
M = sheet1_data[sheet1_data['作物类型'] == '食用菌']['作物'].tolist()
# 定义豆类集合 B：豆类列中值为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 定义水稻集合 R
R = sheet1_data[sheet1_data['作物类型'] == '水稻']['作物'].tolist()
# 定义地块集合 J，使用Sheet2中的地块名称
J = sheet2_data['地块名称'].tolist()
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
# 定义第 k 种地块类型集合 J_k
# 定义地块类型集合 K，从Sheet2中获取唯一的地块类型
K = sheet2_data['地块类型'].unique().tolist()
# 定义 J_平旱地， J_梯田， J_山坡地， J_水浇地
J_pinghandi = sheet2_data[sheet2_data['地块类型'] == '平旱地']['地块名称'].tolist()
J_titian = sheet2_data[sheet2_data['地块类型'] == '梯田']['地块名称'].tolist()
J_shanpodi = sheet2_data[sheet2_data['地块类型'] == '山坡地']['地块名称'].tolist()
J_shuijiaodi = sheet2_data[sheet2_data['地块类型'] == '水浇地']['地块名称'].tolist()
# 定义普通大棚的集合 J_putongd
J_putongd = sheet2_data[sheet2_data['地块类型'] == '普通大棚']['地块名称'].tolist()
# 定义智慧大棚的集合 J_zhihuida
J_zhihuida = sheet2_data[sheet2_data['地块类型'] == '智慧大棚']['地块名称'].tolist()
# 定义豆类作物的集合 B，选择“豆类”列标记为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 重新定义年份集合 T，将每年划分为两个季节
```

```
years = list(range(2024, 2031))  # 2024 到 2030 年
seasons = ['1', '2']  # 每年划分为两个季节
T = [f"{year}_{season}" for year in years for season in
   seasons]


####适应性矩阵####
A_adjust = sheet1_data.iloc[1:42, 1:12]  # B2 对应于索引 (1,1)
   , L42 对应于索引 (42, 12) 不包含右端点


####决策变量####
# 重新定义决策变量 x_ijt 和 y_ijt
x_ijt = {(i, j, t): '面积值' for i in I for j in J for t in T}
     # 用具体面积值替代 '面积值'
y_ijt = {(i, j, t): '0或1' for i in I for j in J for t in T}
   # 用0或1替代


####参数####
pi = {j: {} for j in J}  # 亩产量（吨/亩）
si = {j: {} for j in J}  # 销售价格（元/吨）
ci = {j: {} for j in J}  # 种植成本（元/亩）
Aj = {}  # 地块总面积（亩）

# 提取地块面积信息
for index, row in sheet2_data.iterrows():
    land_type = row['地块类型']
    if land_type not in Aj:
        Aj[land_type] = 0
    Aj[land_type] += row['地块面积/亩']

####参数####
p_ik = {}  # 存储各作物在各地块类型上的亩产量
s_ik = {}  # 存储各作物在各地块类型上的销售价格
```

```
c_ik = {}  # 存储各作物在各地块类型上的种植成本
# 遍历每种作物 i 和每种地块类型 k，提取相应的参数
for i in I:
    p_ik[i] = {}
    s_ik[i] = {}
    c_ik[i] = {}
    for k in K:
        # 使用地块类型 k 来提取参数，确保列名与实际数据表中的列名匹配
        p_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}亩产量'].values[0] if f'{k}亩产量' in sheet1_data.columns else None
        s_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}销售单价'].values[0] if f'{k}销售单价' in sheet1_data.columns else None
        c_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}种植成本'].values[0] if f'{k}种植成本' in sheet1_data.columns else None

# 处理 s_ik 的数据
for crop, values in s_ik.items():
    for land_type, price in list(values.items()):  # 使用 list() 来复制字典项以安全地删除元素
        if pd.isna(price) or price is None:
            # 删除无法种植的地块类型
            del s_ik[crop][land_type]
        else:
            # 如果价格是字符串，计算其平均值
            if isinstance(price, str) and '-' in price:
                low, high = map(float, price.split('-'))
                s_ik[crop][land_type] = (low + high) / 2
            elif isinstance(price, (int, float)):
                # 如果是单一数值则保持原样
                s_ik[crop][land_type] = price
            else:
```

```
                # 删除无法解析的价格数据
                del s_ik[crop][land_type]
# 从 Sheet2 获取每个地块名称对应的地块类型和面积
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
plot_area_mapping = sheet2_data.set_index('地块名称')['地块面积/亩'].to_dict()

# 初始化参数 p_ij, s_ij, c_ij
p_ij = {}
s_ij = {}
c_ij = {}

# 映射参数，将 p_ik, s_ik, c_ik 转换为 p_ij, s_ij, c_ij
for i in I:
    p_ij[i] = {}
    s_ij[i] = {}
    c_ij[i] = {}
    for j in J:
        plot_type = plot_type_mapping[j]  # 获取地块类型
        # 使用地块类型来映射参数，默认值设为 0，避免 None
        p_ij[i][j] = p_ik[i].get(plot_type, 0) if p_ik[i].get(plot_type) is not None else 0
        s_ij[i][j] = s_ik[i].get(plot_type, 0) if s_ik[i].get(plot_type) is not None else 0
        c_ij[i][j] = c_ik[i].get(plot_type, 0) if c_ik[i].get(plot_type) is not None else 0

# 作物分类
grain_crops = sheet1_data[sheet1_data['作物类型'] == '粮食']['作物'].tolist()
rice_crops = sheet1_data[sheet1_data['作物'] == '水稻']['作物'].tolist()
vegetables = sheet1_data[sheet1_data['作物类型'] == '蔬菜']['作物'].tolist()
```

```python
mushrooms = sheet1_data[sheet1_data['作物类型'] == '食用菌']['作物'].tolist()
bean_crops = sheet1_data[sheet1_data['作物类型'] == '豆类']['作物'].tolist()

# 特殊蔬菜
restricted_veg = ['大白菜', '白萝卜', '红萝卜']
# 收益参数（示例，实际应从数据中读取）
profit_per_acre = {i: random.uniform(50, 150) for i in I}
min_area_ratio = 0.1  # 最小种植面积比例

# 模拟种植方案并选择最高利润的方案
best_profit = 0
best_plan = None

# 假设最佳种植方案已生成，以下为检查和修正逻辑
def validate_and_fix_plan(planting_plan, x_ijt, restricted_veg, vegetables, mushrooms, rice_crops):
    for j, times in planting_plan.items():
        previous_crop = None
        for t, crops in sorted(times.items()):  # 按时间顺序检查
            # 检查作物1不能空
            if not crops:
                # 自动选择一个合适的作物进行种植
                print(f"警告：地块 {j} 在 {t} 没有种植任何作物，自动选择作物填补。")
                # 为普通大棚，选择适当作物
                if '普通大棚' in j:
                    if '1' in t:  # 第一季种蔬菜
                        selected_crop = random.choice([v for v in vegetables if v not in restricted_veg])
                    else:  # 第二季种食用菌
                        selected_crop = random.choice(mushrooms)
```

```
160                 elif '水浇地' in j and '1' in t:
161                     selected_crop = random.choice(rice_crops)
    # 第一季种水稻
162                 else:
163                     selected_crop = random.choice([v for v in
    vegetables if v not in restricted_veg])
164
165                 planting_area = random.uniform(0.1 *
    plot_area_mapping[j], plot_area_mapping[j])
166                 crops.append((selected_crop, planting_area))
167                 x_ijt[(selected_crop, j, t)] = planting_area
168
169             # 检查相邻季度之间的作物不能重复
170             current_crops = [crop for crop, _ in crops]
171             if previous_crop and previous_crop in
    current_crops:
172                 # 修正重复作物，选择新的作物
173                 print(f"错误：地块 {j} 在 {t} 和前一季度种植了
    相同的作物 {previous_crop}，进行修正。")
174                 # 尝试选一个不同的作物来替代
175                 new_crop = random.choice([v for v in
    vegetables if v not in restricted_veg and v != previous_crop
    ])
176                 planting_area = random.uniform(0.1 *
    plot_area_mapping[j], plot_area_mapping[j])
177                 crops[0] = (new_crop, planting_area)  # 替换为
    新的作物
178                 x_ijt[(new_crop, j, t)] = planting_area
179
180             previous_crop = current_crops[0]  # 假设作物1为当
    前季度的主要作物
181
182     print("所有地块的种植方案已检查并修正。")
183     return planting_plan, x_ijt
184
```

```python
for _ in range(1000):  # 模拟10次
    planting_plan = {j: {t: [] for t in T} for j in J}
    x_ijt = {(i, j, t): 0 for i in I for j in J for t in T}
    total_profit = 0

    # 定义豆子分类
    bean_crops_1 = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆']  # 豆子1类
    bean_crops_2 = ['豇豆', '刀豆', '芸豆']  # 豆子2类

    # 三年周期内种植豆类作物的逻辑
    for j in J:
        plot_type = plot_type_mapping[j]

        # 筛选地块类型，仅对 A、B、C 和 D 类地块执行豆类种植
        if plot_type not in ['平旱地', '梯田', '山坡地', '水浇地']:
            continue  # 跳过不相关地块

        for start_year in range(2024, 2030, 3):
            three_year_period = [f"{year}_{season}" for year in range(start_year, start_year + 3) for season in seasons]
            planted_beans = False

            while not planted_beans:
                t = random.choice(three_year_period)
                if len(planting_plan[j][t]) == 0:  # 确保该季节没有作物
                    # A、B、C类地块使用豆子1类
                    if plot_type in ['平旱地', '梯田', '山坡地']:
                        selected_crop = random.choice(bean_crops_1)
                    # D类地块（水浇地）使用豆子2类
                    elif plot_type == '水浇地':
```

```
                        selected_crop = random.choice(bean_crops_2)

                    # 设置种植面积
                    planting_area = random.uniform(min_area_ratio * plot_area_mapping[j], plot_area_mapping[j])
                    planting_plan[j][t].append((selected_crop, planting_area))
                    x_ijt[(selected_crop, j, t)] = planting_area
                    planted_beans = True
                    total_profit += profit_per_acre[selected_crop] * planting_area

    for j in J:
        area = plot_area_mapping[j]
        plot_type = plot_type_mapping[j]

        for t in T:
            season = int(t.split('_')[1])

            # A、B、C类地块：种植粮食类（除水稻）和蔬菜
            if plot_type in ['平旱地', '梯田', '山坡地']:
                selected_crop = random.choice(
                    grain_crops)
                planting_area = random.uniform(min_area_ratio * area, area)
                planting_plan[j][t].append((selected_crop, planting_area))
                x_ijt[(selected_crop, j, t)] = planting_area
                total_profit += profit_per_acre[selected_crop] * planting_area

            # D类地块：要么第一季种水稻，第二季不种；要么种两季蔬菜
```

```
240             elif plot_type == '水浇地':
241                 # 随机决定是种水稻还是两季都种蔬菜
242                 if random.choice([True, False]):
243                     # 选择种植水稻的情况：第一季种水稻，第二季
    不种作物
244                     if season == 1:
245                         selected_crop = random.choice(
    rice_crops)
246                         planting_area = random.uniform(
    min_area_ratio * area, area)
247                         planting_plan[j][t].append((
    selected_crop, planting_area))
248                         x_ijt[(selected_crop, j, t)] =
    planting_area
249                         total_profit += profit_per_acre[
    selected_crop] * planting_area
250                     # 第二季不种作物，保持为空
251                 else:
252                     # 选择种植两季蔬菜的情况
253                     selected_crop = random.choice([v for v in
    vegetables if v not in restricted_veg])
254                     planting_area = random.uniform(
    min_area_ratio * area, area)
255                     planting_plan[j][t].append((selected_crop,
     planting_area))
256                     x_ijt[(selected_crop, j, t)] =
    planting_area
257                     total_profit += profit_per_acre[
    selected_crop] * planting_area
258             # 普通大棚 E 类：第一季种蔬菜，第二季种食用菌
259             elif plot_type == '普通大棚':
260                 # 第一季：种植蔬菜（不包括特殊限制的蔬菜）
261                 if season == 1:
262                     # 确保有合适的蔬菜进行种植
263                     if [v for v in vegetables if v not in
```

```
restricted_veg]:
                    selected_crop = random.choice([v for v
 in vegetables if v not in restricted_veg])
                else:
                    selected_crop = random.choice(
vegetables)  # 如果没有合适的蔬菜，选一个随机蔬菜
            # 第二季：种植食用菌
            else:
                # 确保有食用菌进行种植
                if mushrooms:
                    selected_crop = random.choice(
mushrooms)
                else:
                    selected_crop = '替代作物'  # 如果没有
食用菌，选择一个替代作物或其他默认值

            # 确保分配种植面积
            planting_area = random.uniform(min_area_ratio
* area, area)
            planting_plan[j][t].append((selected_crop,
planting_area))
            x_ijt[(selected_crop, j, t)] = planting_area
            total_profit += profit_per_acre[selected_crop]
 * planting_area


        # 智慧大棚 F 类：两季都种蔬菜
        elif plot_type == '智慧大棚':
            selected_crop = random.choice([v for v in
vegetables if v not in restricted_veg])
            planting_area = random.uniform(min_area_ratio
* area, area)
            planting_plan[j][t].append((selected_crop,
planting_area))
            x_ijt[(selected_crop, j, t)] = planting_area
```

```
                total_profit += profit_per_acre[selected_crop]
    * planting_area
    # 更新最佳方案
    if total_profit > best_profit:
        best_profit = total_profit
        best_plan = (planting_plan, x_ijt)

# 输出最佳种植方案
best_planting_plan, best_x_ijt = best_plan
best_planting_plan, best_x_ijt = validate_and_fix_plan(
   best_planting_plan, best_x_ijt, restricted_veg, vegetables,
   mushrooms, rice_crops)

# 创建 DataFrame 以适应不同类型地块的输出格式
rows = []
for j, times in best_planting_plan.items():
    for t, crops in times.items():
        row = {'地块': j, '时间': t}
        for idx, (crop, area) in enumerate(crops, start=1):
            row[f'作物{idx}'] = crop
            row[f'作物{idx}种植面积'] = area
        rows.append(row)

planting_plan_df = pd.DataFrame(rows)

# 保存为 Excel 文件
output_path = "D:/new/result3.xlsx"
planting_plan_df.to_excel(output_path, index=False, sheet_name
   ='种植方案')

print(f"最高利润: {best_profit}")
print(f"最佳种植方案已保存到: {output_path}")
```

问题二求解代码.py

```
import os
```

```python
import pandas as pd
from pulp import LpMaximize, LpProblem, LpVariable, lpSum, LpStatus, value, PULP_CBC_CMD
from scipy.optimize import linprog
import numpy as np
import random
# 读取Excel文件中的Sheet1
file_path = "D:\\2024国赛\\问题一\\A_adjust 的副本.xlsx"
sheet1_data = pd.read_excel(file_path, sheet_name='Sheet1')
sheet2_data = pd.read_excel(file_path, sheet_name='Sheet2')
# 替换数据中的 NaN 和 inf 为 0
sheet1_data = sheet1_data.replace([np.nan, np.inf, -np.inf], 0)
sheet2_data = sheet2_data.replace([np.nan, np.inf, -np.inf], 0)

####集合####
# 定义作物集合 I，包括所有作物的名称
I = sheet1_data['作物'].tolist()
# 定义粮食类作物集合 G
G = sheet1_data[sheet1_data['作物类型'] == '粮食']['作物'].tolist()
# 定义蔬菜类作物集合 V
V = sheet1_data[sheet1_data['作物类型'] == '蔬菜']['作物'].tolist()
# 定义食用菌类作物集合 M
M = sheet1_data[sheet1_data['作物类型'] == '食用菌']['作物'].tolist()
# 定义豆类集合 B：豆类列中值为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 定义水稻集合 R
R = sheet1_data[sheet1_data['作物类型'] == '水稻']['作物'].tolist()
# 定义地块集合 J，使用Sheet2中的地块名称
J = sheet2_data['地块名称'].tolist()
```

```
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
# 定义第 k 种地块类型集合 J_k
# 定义地块类型集合 K，从Sheet2中获取唯一的地块类型
K = sheet2_data['地块类型'].unique().tolist()
# 定义 J_平旱地， J_梯田， J_山坡地， J_水浇地
J_pinghandi = sheet2_data[sheet2_data['地块类型'] == '平旱地']['地块名称'].tolist()
J_titian = sheet2_data[sheet2_data['地块类型'] == '梯田']['地块名称'].tolist()
J_shanpodi = sheet2_data[sheet2_data['地块类型'] == '山坡地']['地块名称'].tolist()
J_shuijiaodi = sheet2_data[sheet2_data['地块类型'] == '水浇地']['地块名称'].tolist()
# 定义普通大棚的集合 J_putongd
J_putongd = sheet2_data[sheet2_data['地块类型'] == '普通大棚']['地块名称'].tolist()
# 定义智慧大棚的集合 J_zhihuida
J_zhihuida = sheet2_data[sheet2_data['地块类型'] == '智慧大棚']['地块名称'].tolist()
# 定义豆类作物的集合 B，选择“豆类”列标记为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 重新定义年份集合 T，将每年划分为两个季节
years = list(range(2024, 2031))  # 2024 到 2030 年
seasons = ['1', '2']  # 每年划分为两个季节
T = [f"{year}_{season}" for year in years for season in seasons]


####适应性矩阵####
A_adjust = sheet1_data.iloc[1:42, 1:12]  # B2 对应于索引 (1,1)，L42 对应于索引 (42, 12) 不包含右端点


####决策变量####
```

```python
# 重新定义决策变量 x_ijt 和 y_ijt
x_ijt = {(i, j, t): '面积值' for i in I for j in J for t in T}
    # 用具体面积值替代 '面积值'
y_ijt = {(i, j, t): '0或1' for i in I for j in J for t in T}
   # 用0或1替代


####参数####
pi = {j: {} for j in J}  # 亩产量（吨/亩）
si = {j: {} for j in J}  # 销售价格（元/吨）
ci = {j: {} for j in J}  # 种植成本（元/亩）
Aj = {}  # 地块总面积（亩）

# 提取地块面积信息
for index, row in sheet2_data.iterrows():
    land_type = row['地块类型']
    if land_type not in Aj:
        Aj[land_type] = 0
    Aj[land_type] += row['地块面积/亩']

####参数####
p_ik = {}  # 存储各作物在各地块类型上的亩产量
s_ik = {}  # 存储各作物在各地块类型上的销售价格
c_ik = {}  # 存储各作物在各地块类型上的种植成本
# 遍历每种作物 i 和每种地块类型 k，提取相应的参数
for i in I:
    p_ik[i] = {}
    s_ik[i] = {}
    c_ik[i] = {}
    for k in K:
        # 使用地块类型 k 来提取参数，确保列名与实际数据表中的
   列名匹配
        p_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{
   k}亩产量'].values[0] if f'{k}亩产量' in sheet1_data.columns
   else None
```

```
        s_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}销售单价'].values[0] if f'{k}销售单价' in sheet1_data.columns else None
        c_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}种植成本'].values[0] if f'{k}种植成本' in sheet1_data.columns else None

# 处理 s_ik 的数据
for crop, values in s_ik.items():
    for land_type, price in list(values.items()):  # 使用 list() 来复制字典项以安全地删除元素
        if pd.isna(price) or price is None:
            # 删除无法种植的地块类型
            del s_ik[crop][land_type]
        else:
            # 如果价格是字符串，计算其平均值
            if isinstance(price, str) and '-' in price:
                low, high = map(float, price.split('-'))
                s_ik[crop][land_type] = (low + high) / 2
            elif isinstance(price, (int, float)):
                # 如果是单一数值则保持原样
                s_ik[crop][land_type] = price
            else:
                # 删除无法解析的价格数据
                del s_ik[crop][land_type]
# 从 Sheet2 获取每个地块名称对应的地块类型和面积
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
plot_area_mapping = sheet2_data.set_index('地块名称')['地块面积/亩'].to_dict()

# 初始化参数 p_ij, s_ij, c_ij
p_ij = {}
s_ij = {}
c_ij = {}
```

```
114
115 # 映射参数，将 p_ik, s_ik, c_ik 转换为 p_ij, s_ij, c_ij
116 for i in I:
117     p_ij[i] = {}
118     s_ij[i] = {}
119     c_ij[i] = {}
120     for j in J:
121         plot_type = plot_type_mapping[j]  # 获取地块类型
122         # 使用地块类型来映射参数，默认值设为 0，避免 None
123         p_ij[i][j] = p_ik[i].get(plot_type, 0) if p_ik[i].get(
    plot_type) is not None else 0
124         s_ij[i][j] = s_ik[i].get(plot_type, 0) if s_ik[i].get(
    plot_type) is not None else 0
125         c_ij[i][j] = c_ik[i].get(plot_type, 0) if c_ik[i].get(
    plot_type) is not None else 0
126 # 预计销售量的增长率和波动率
127 Growth_Rate = {i: random.uniform(0.05, 0.10) for i in ['小麦',
     '玉米']}  # 小麦和玉米的增长率
128 Fluctuation_Rate = {i: random.uniform(-0.05, 0.05) for i in I
    if i not in ['小麦', '玉米']}  # 其他作物的波动率
129
130 # 各作物的产量波动率
131 Yield_Fluctuation_Rate = {i: random.uniform(-0.10, 0.10) for i
     in I}
132
133 # 产量计算
134 p_ijt = {(i, t): p_ij[i].get('2023', 0) * (1 +
    Yield_Fluctuation_Rate[i]) for i in I for t in T}
135 # 种植成本的年增长率
136 Cost_Growth_Rate = 0.05
137
138 # 种植成本计算
139 c_ijt = {(i, t): c_ij[i].get('2023', 0) * (1 +
    Cost_Growth_Rate) ** (int(t.split('_')[0]) - 2023) for i in
    I for t in T}
```

```
# 粮食类作物的价格保持不变
s_ijt = {(i, t): s_ij[i].get('2023', 0) if i in ['小麦', '玉米'] else s_ij[i].get(t, 0) for i in I for t in T}

# 定义情景集 S 和每个情景的概率 π_s
S = ['s1', 's2', 's3']  # 示例情景，可以根据实际情况调整
π = {'s1': 0.3, 's2': 0.5, 's3': 0.2}  # 每个情景的发生概率，确保概率之和为1

# 定义各情景下的参数波动
scenario_fluctuations = {
    's1': {'price': -0.10, 'cost': 0.15, 'yield': -0.05},
    's2': {'price': 0.00, 'cost': 0.05, 'yield': 0.00},
    's3': {'price': 0.05, 'cost': -0.05, 'yield': 0.05}
}
# 定义情景下的参数 p_ijt^s, s_ijt^s, c_ijt^s
p_ijt_s = {}
s_ijt_s = {}
c_ijt_s = {}

for s in S:
    p_ijt_s[s] = {(i, j, t): p_ij[i][j] * (1 + scenario_fluctuations[s]['yield']) for i in I for j in J for t in T}
    s_ijt_s[s] = {(i, j, t): s_ij[i][j] * (1 + scenario_fluctuations[s]['price']) for i in I for j in J for t in T}
    c_ijt_s[s] = {(i, j, t): c_ij[i][j] * (1 + scenario_fluctuations[s]['cost']) for i in I for j in J for t in T}
# 定义目标函数 Z


problem = lpSum(π[s] * lpSum(p_ijt_s[s][i, j, t] * s_ijt_s[s][i, j, t] * x_ijt[i, j, t] - c_ijt_s[s][i, j, t] * x_ijt[i, j
```

```
    , t]
                        for i in I for j in J for t in T) for s
    in S)
# 地块种植面积限制
for s in S:
    for j in J:
        for t in T:
            problem += lpSum(x_ijt[i, j, t] for i in I) <= Aj[
    j], f"Land_Area_Constraint_{j}_{t}_{s}"

# 种植一致性约束
M = 100  # 最大种植面积，示例值
for s in S:
    for i in I:
        for j in J:
            for t in T:
                problem += x_ijt[i, j, t] <= M * y_ijt[i, j, t
    ], f"Consistency_Constraint_{i}_{j}_{t}_{s}"

# 最低种植面积要求
MinArea = 1  # 最小种植面积，示例值
for s in S:
    for i in I:
        for t in T:
            problem += lpSum(x_ijt[i, j, t] for j in J) >=
    MinArea * y_ijt[i, j, t], f"Min_Area_Requirement_{i}_{t}_{s}
    "
# 豆类作物轮作要求
for s in S:
    for j in J:
        for t0 in range(len(T) - 2):  # 确保 t0 + 2 不超过时间
    集的最大索引
            problem += lpSum(y_ijt[i, j, T[t]] for i in B for
    t in range(t0, t0 + 3)) >= 1, \
                f"Bean_Crop_Rotation_{j}_{t0}_{s}"
```

```
# 使用 PuLP 的 CBC 求解器求解模型
problem.solve(PULP_CBC_CMD(msg=1))
# 输出求解状态
print(f"求解状态: {LpStatus[problem.status]}")
# 输出决策变量的值
for v in problem.variables():
    print(f"{v.name} = {v.varValue}")
# 输出目标函数的最优值
print(f"目标函数的最优值: {value(problem.objective)}")
# 还可以进一步将结果输出到一个文件或其他处理步骤
results = {v.name: v.varValue for v in problem.variables()}
results_df = pd.DataFrame(list(results.items()), columns=['Variable', 'Value'])
results_df.to_csv("optimization_results.csv", index=False)
print("优化结果已保存到 'optimization_results.csv'")
```

问题三求解代码.py

```
import os
import pandas as pd
from pulp import LpMaximize, LpProblem, LpVariable, lpSum, LpStatus, value, PULP_CBC_CMD
from scipy.optimize import linprog
import numpy as np
import random
# 读取Excel文件中的Sheet1
file_path = "D:\\2024国赛\\问题一\\A_adjust 的副本.xlsx"
sheet1_data = pd.read_excel(file_path, sheet_name='Sheet1')
sheet2_data = pd.read_excel(file_path, sheet_name='Sheet2')
# 替换数据中的 NaN 和 inf 为 0
sheet1_data = sheet1_data.replace([np.nan, np.inf, -np.inf], 0)
sheet2_data = sheet2_data.replace([np.nan, np.inf, -np.inf], 0)

####集合####
```

```python
# 定义作物集合 I，包括所有作物的名称
I = sheet1_data['作物'].tolist()
# 定义粮食类作物集合 G
G = sheet1_data[sheet1_data['作物类型'] == '粮食']['作物'].tolist()
# 定义蔬菜类作物集合 V
V = sheet1_data[sheet1_data['作物类型'] == '蔬菜']['作物'].tolist()
# 定义食用菌类作物集合 M
M = sheet1_data[sheet1_data['作物类型'] == '食用菌']['作物'].tolist()
# 定义豆类集合 B：豆类列中值为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 定义水稻集合 R
R = sheet1_data[sheet1_data['作物类型'] == '水稻']['作物'].tolist()
# 定义地块集合 J，使用Sheet2中的地块名称
J = sheet2_data['地块名称'].tolist()
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
# 定义第 k 种地块类型集合 J_k
# 定义地块类型集合 K，从Sheet2中获取唯一的地块类型
K = sheet2_data['地块类型'].unique().tolist()
# 定义 J_平旱地， J_梯田， J_山坡地， J_水浇地
J_pinghandi = sheet2_data[sheet2_data['地块类型'] == '平旱地']['地块名称'].tolist()
J_titian = sheet2_data[sheet2_data['地块类型'] == '梯田']['地块名称'].tolist()
J_shanpodi = sheet2_data[sheet2_data['地块类型'] == '山坡地']['地块名称'].tolist()
J_shuijiaodi = sheet2_data[sheet2_data['地块类型'] == '水浇地']['地块名称'].tolist()
# 定义普通大棚的集合 J_putongd
J_putongd = sheet2_data[sheet2_data['地块类型'] == '普通大棚']['地块名称'].tolist()
```

```
# 定义智慧大棚的集合 J_zhihuida
J_zhihuida = sheet2_data[sheet2_data['地块类型'] == '智慧大棚'][
    '地块名称'].tolist()
# 定义豆类作物的集合 B，选择“豆类”列标记为1的作物
B = sheet1_data[sheet1_data['豆类'] == 1]['作物'].tolist()
# 重新定义年份集合 T，将每年划分为两个季节
years = list(range(2024, 2031))  # 2024 到 2030 年
seasons = ['1', '2']  # 每年划分为两个季节
T = [f"{year}_{season}" for year in years for season in seasons]


####适应性矩阵####
A_adjust = sheet1_data.iloc[1:42, 1:12]  # B2 对应于索引 (1,1)，L42 对应于索引 (42, 12) 不包含右端点


####决策变量####
# 重新定义决策变量 x_ijt 和 y_ijt
x_ijt = {(i, j, t): '面积值' for i in I for j in J for t in T}  # 用具体面积值替代 '面积值'
y_ijt = {(i, j, t): '0或1' for i in I for j in J for t in T}  # 用0或1替代


####参数####
pi = {j: {} for j in J}  # 亩产量（吨/亩）
si = {j: {} for j in J}  # 销售价格（元/吨）
ci = {j: {} for j in J}  # 种植成本（元/亩）
Aj = {}  # 地块总面积（亩）

# 提取地块面积信息
for index, row in sheet2_data.iterrows():
    land_type = row['地块类型']
    if land_type not in Aj:
```

```
        Aj[land_type] = 0
    Aj[land_type] += row['地块面积/亩']

####参数####
p_ik = {}  # 存储各作物在各地块类型上的亩产量
s_ik = {}  # 存储各作物在各地块类型上的销售价格
c_ik = {}  # 存储各作物在各地块类型上的种植成本
# 遍历每种作物 i 和每种地块类型 k，提取相应的参数
for i in I:
    p_ik[i] = {}
    s_ik[i] = {}
    c_ik[i] = {}
    for k in K:
        # 使用地块类型 k 来提取参数，确保列名与实际数据表中的列名匹配
        p_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}亩产量'].values[0] if f'{k}亩产量' in sheet1_data.columns else None
        s_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}销售单价'].values[0] if f'{k}销售单价' in sheet1_data.columns else None
        c_ik[i][k] = sheet1_data[sheet1_data['作物'] == i][f'{k}种植成本'].values[0] if f'{k}种植成本' in sheet1_data.columns else None

# 处理 s_ik 的数据
for crop, values in s_ik.items():
    for land_type, price in list(values.items()):  # 使用 list() 来复制字典项以安全地删除元素
        if pd.isna(price) or price is None:
            # 删除无法种植的地块类型
            del s_ik[crop][land_type]
        else:
            # 如果价格是字符串，计算其平均值
            if isinstance(price, str) and '-' in price:
```

```
                low, high = map(float, price.split('-'))
                s_ik[crop][land_type] = (low + high) / 2
            elif isinstance(price, (int, float)):
                # 如果是单一数值则保持原样
                s_ik[crop][land_type] = price
            else:
                # 删除无法解析的价格数据
                del s_ik[crop][land_type]
# 从 Sheet2 获取每个地块名称对应的地块类型和面积
plot_type_mapping = sheet2_data.set_index('地块名称')['地块类型'].to_dict()
plot_area_mapping = sheet2_data.set_index('地块名称')['地块面积/亩'].to_dict()

# 初始化参数 p_ij, s_ij, c_ij
p_ij = {}
s_ij = {}
c_ij = {}

# 映射参数，将 p_ik, s_ik, c_ik 转换为 p_ij, s_ij, c_ij
for i in I:
    p_ij[i] = {}
    s_ij[i] = {}
    c_ij[i] = {}
    for j in J:
        plot_type = plot_type_mapping[j]  # 获取地块类型
        # 使用地块类型来映射参数，默认值设为 0，避免 None
        p_ij[i][j] = p_ik[i].get(plot_type, 0) if p_ik[i].get(plot_type) is not None else 0
        s_ij[i][j] = s_ik[i].get(plot_type, 0) if s_ik[i].get(plot_type) is not None else 0
        c_ij[i][j] = c_ik[i].get(plot_type, 0) if c_ik[i].get(plot_type) is not None else 0
# 定义线性规划问题
model = LpProblem("Crop_Planning", LpMaximize)
```

```
128
129 # 定义决策变量 x_ijt 和 y_ijt
130 x_ijt = LpVariable.dicts("x", (I, J, T), lowBound=0, cat='
    Continuous')  # 面积值
131 y_ijt = LpVariable.dicts("y", (I, J, T), cat='Binary')  # 0或1
132
133 # 定义经济效益最大化目标函数 Z1
134 Z1 = lpSum(
135     (p_ij[i][j] * s_ij[i][j] - c_ij[i][j]) * x_ijt[i][j][t]
136     for i in I for j in J for t in T
137 )
138
139 # 将 Z1 添加到模型目标中
140 model += Z1
141 # 定义社会效益指标的参数（假设已在 sheet1_data 中包含该列）
142 se_it = {i: sheet1_data[sheet1_data['作物'] == i]['社会效益指
    标'].values[0] for i in I}
143
144 # 定义社会效益最大化目标函数 Z2
145 Z2 = lpSum(
146     se_it[i] * x_ijt[i][j][t]
147     for i in I for j in J for t in T
148 )
149
150 # 将 Z2 添加到模型中
151 model += Z2
152 # 定义资源使用量（假设资源使用量信息已在 sheet1_data 中定义）
153 # 比如使用"资源使用"列表示每种作物在不同地块上的资源消耗
154 resource_use = {i: sheet1_data[sheet1_data['作物'] == i]['资源
    使用'].values[0] for i in I}
155
156 # 定义资源使用率目标的辅助变量 lambda
157 lambda_var = LpVariable("lambda", lowBound=0, cat='Continuous'
    )
158
```

```
# 资源使用率最大化的约束（线性化形式）
model += lpSum(
    (p_ij[i][j] * s_ij[i][j]) * x_ijt[i][j][t]
    for i in I for j in J for t in T
) >= lambda_var * lpSum(
    resource_use[i] * x_ijt[i][j][t]
    for i in I for j in J for t in T
)

# 定义资源使用率目标函数 Z3
Z3 = lambda_var
# 定义目标函数的权重
alpha, beta, gamma = 0.4, 0.3, 0.3  # 示例权重，根据实际情况调整

# 综合目标函数
model += alpha * Z1 + beta * Z2 + gamma * Z3

# 打印目标函数
print("Objective Function:", model.objective)
# 示例约束：土地利用约束
for j in J:
    model += lpSum(x_ijt[i][j][t] for i in I for t in T) <= plot_area_mapping[j]  # 地块面积约束
# 求解模型
model.solve(PULP_CBC_CMD())

# 输出求解状态和结果
print("Status:", LpStatus[model.status])
print("Optimal Solution to the problem:", value(model.objective))
for var in model.variables():
    print(var.name, "=", var.varValue)
```