多级约束条件下农作物种植最优化问题的分层求解模型

摘要

本文基于**正态分布和变异系数**，引入**随机化**、**概率化**的农作物产量、销售量、销售价格和预期销量的度量，并通过**线性模型**和**对数正态模型**限制其关联关系，在稳定性约束、土地约束、状态约束和经验等**多级约束条件**下，构建**混合整数规划模型 (MIP)**。以最大化利润和最小化**作物分布的离散程度**为联合目标，借助**马尔科夫链**思想，通过**分层优化**的方式对单一年份下的**多级约束**逐步求解，并利用预测结果在下一年份的多级约束下迭代求解，达到获取每年的最优种植策略的目的。

针对问题一，首先以**“3σ”原则**为销售价格的区间端点，以**变异系数**小于 10% 为衡量销售单价稳定性的依据构造正态分布律，使用**蒙特卡洛模拟**以正态概率密度随机化销售价格和销售单价。然后基于各种作物总产量及种类稳定和各农田种植数量和种类稳定构造稳定性约束；基于不同类型土地适宜生长的作物不同构造土地约束；基于轮作规律和豆类对其他作物的促进作用构造状态约束。最后，以作物在农田上种植矩阵的**平均曼哈顿距离**作为作物分布的离散程度的衡量（惩罚项），以最大化利润为目标构建 **MIP 模型**。借助**马尔科夫链**思想，利用 CBC 求解算法，**分层求解**各级约束条件，用预测结果迭代地求解下一年的方案，在不同销售策略下获得年平均 605 万和 633 万元的利润和最优种植方式。

针对问题二，利用**指数随机游走模型**将相关变量随时间的变化趋势引入决策，并适当放宽了问题一中的稳定性约束，以达到适应真实世界中数据变化的目的。利用相同的求解方法，得到了年平均 682 万元的利润和最优种植方式。

针对问题三，进一步将作物分为豆类粮食、豆类蔬菜、淀粉类、谷物类、叶菜类蔬菜、根茎类蔬菜、果菜类蔬菜和食用菌 7 个小类，使得每个小类之间的作物相互具有**可替代性**，同时从种植经验和营养成分两方面约束每个小类之间的**互补性**，使之符合**需求弹性**。利用**线性模型**拟合种植成本与销售价格，利用**对数正态模型**拟合具有**长尾分布**的销售价格与销量关系，以达到适应市场变化的目的。此外，考虑到消费者在全年对不同品种蔬菜的需求以及蔬菜**种植季度的时序重叠**关系，额外构造了一个时序约束，以保证全品种的蔬菜在全年的供应链。利用相同的求解方法，得到了年平均 695 万元的利润和最优种植方式。同问题二相比，互为可替代品的作物多元化程度降低，但获取的利润提高，同时保障了市场需求。

除此之外，对模型中引入的惩罚项超参数 λ 进行**灵敏度分析**并在对输入数据进行 10% 幅度的扰动下进行**鲁棒性检验**，结果显示，当 $\lambda \leq 10$ 时模型对 λ 敏感，当 $\lambda > 10$ 时模型稳定；模型在扰动输入数据下的结果变化不超过 10%，展现了一定的鲁棒性。

关键字：蒙特卡洛模拟，MIP，曼哈顿惩罚项，多级约束，分层求解，对数正态模型

一、 问题重述

1.1 问题背景

一直以来，我国致力于乡村振兴和可持续发展，并作出了持续努力。农业是乡村振兴的根本，根据乡村的实际情况，充分利用有限的耕地资源，因地制宜，发展有机种植产业，对乡村经济的可持续发展具有重要的现实意义。选择适宜的农作物，优化种植策略，有利于方便田间管理，提高生产效益，减少各种不确定因素可能造成的种植风险。然而，由于乡村地理位置的差异，温度、湿度、气候、人口等自然或人文因素，乡村耕种决策仍然存在挑战。

华北山区的气候条件艰苦，常年伴随着低气温。农作物生长规律与平原等地区不尽相同，针对农作物分区种植、定价销售方面存在的挑战。本研究旨在建立合理的数学模型，揭示成本、销量、产量之间蕴含关系的同时，帮助农民做出最优的耕种决策。

1.2 问题提出

农业问题的核心是**综合考虑自然条件和种植经验，在资源有限的约束条件下，获得利润最高的种植策略**。因此，本研究针对以下三个研究问题展开。

问题一：在自然条件和社会因素稳定的前提下，基于 2023 年的种植成本、亩产量和销售价格，在❶当产量大于销量时，超过部分滞销，造成浪费；❷超过部分按 2023 年销售价格的 50% 降价出售两种情况下，分别给出 2024~2030 年农作物的最优种植方案。

问题二：在问题一的基础上，引入气候因素和市场因素对农作物价格、产量和销量的影响，即：❶小麦和玉米未来的预期销售量平均年增长率介于 5%~10% 之间，其他农作物未来每年的预期销售量相对于 2023 年大约有 ±5% 的变化。❷农作物的亩产量每年会有 ±10% 的变化，种植成本平均每年增长 5% 左右。❸粮食类作物的销售价格基本稳定；蔬菜类作物的销售价格每年增长 5% 左右；食用菌的销售价每年可下降 1%~5%，羊肚菌的销售价格每年下降 5%。在此基础上，给出 2024~2030 年农作物的最优种植方案。

问题三：在问题二的基础上，考虑农作物之间的可替代性和互补性，引入先验知识和市场关系相关性等不确定因素对结果的影响，给出 2024~2030 年农作物的最优种植方案。并与问题二的结果进行对比。

二、 问题分析

2.1 总体分析

本题目是一个关于农作物种植与销售的**优化类问题**。图1展示了我们问题分析的流程。

从决策目的看，本研究旨在利用已有的种植方案和销售信息，根据经验构造目标函数，获得最大化的利润和最易于耕种和管理的种植方式。因此，需要完成四方面任务——其一，进行利润矩阵的度量和构建，寻找合适的定量指标描述作物种植的密集程度。其二，准备决策所需的变量，利用历史数据信息进行建模，概率化预期销售量、销售单价、亩产量和种植成本等决策变量，并寻找其中隐含的复杂相关关系。其三，构造约束条件，从已知信息中抽象出合理的约束条件表达式，并根据生产生活经验构造出合理的其他约束。其四，采用合适的优化模型求解目标函数的极值。

从数据特性看，本研究已知的数据少而精，具有明显的结构化特征。因此，无法通过大量的历史数据拟合变量间的关系，为了保证模型与现实世界的适配性，需引入随机变量的分布律进行信息概率化。

从模型选择看，本研究所构建的优化模型具有约束条件繁多，约束信息关联关系复杂，目标变量明确分为线性和二进制两类的特点。因此不适宜选择启发式搜索算法，宜选用混合整数规划 (MIP) 模型，采用 CBC 优化器，分层逐步约束的方法进行求解。

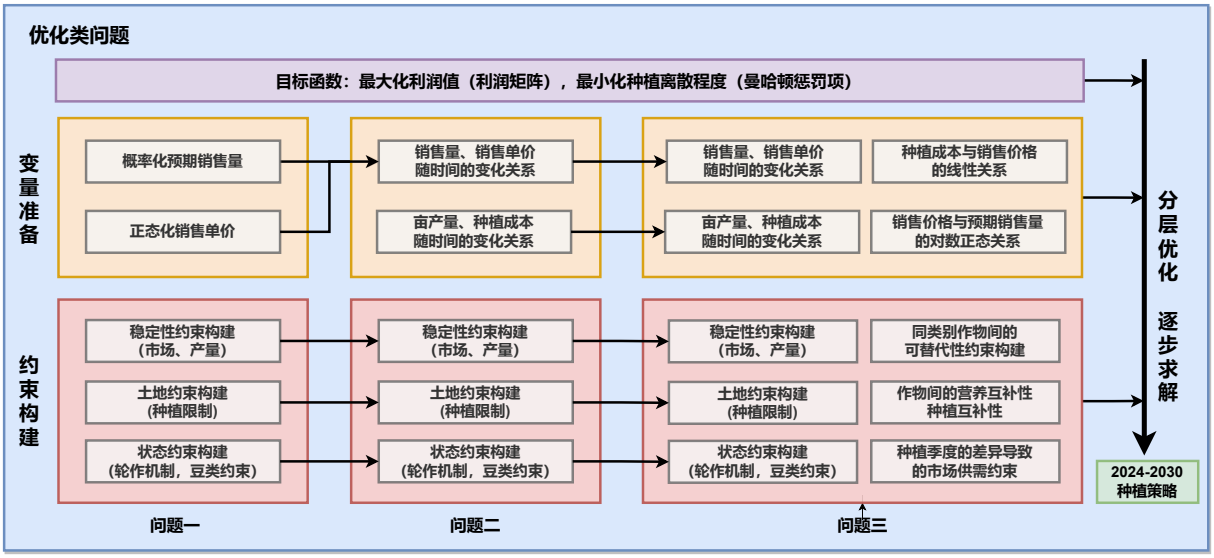

图 1　总体分析流程

2.2 问题一的分析

问题一的核心是从不同方面寻找最优化问题的约束条件，并构建合理目标函数，采用合适的优化算法进行求解。首先运用概率模型对数据进行决策相关信息处理，包括对

预期销售量矩阵 ($\boldsymbol{S}$) 采用变异系数进行概率化处理，以及对销售单价矩阵 ($\boldsymbol{P}$) 进行正态化处理。然后根据题目信息与现实需要，构建市场供需关系以及作物种植产量的稳定性约束，作物生产条件与种植限制的土地约束，交替种植与豆类固氮的状态约束三类约束条件。在此基础上建立模型，分别构建在两种销售情况下的利润矩阵计算公式，最大化利润目标函数。并引入曼哈顿距离惩罚项来减少地块种植的分散度，通过惩罚权重 λ 来决定种植分散度在总目标函数中的影响。最终综合目标函数，在多级约束条件下分层求解得到 2024~2030 年的种植规划以及最高收益。

2.3 问题二的分析

问题二的核心是将时间变化引入成本、销量、产量、销售价格等因素，需要修正问题一中的约束条件并采用合适的优化算法进行求解。为了进一步综合考量 2023 年之后市场条件的变化以及作物种植的不确定因素，我们首先对决策信息进行修正，根据小麦与玉米预期销售量 5%~10% 的增长趋势以及其他农作物 ±5% 的销售量浮动，将变化趋势引入到年销量中，获得概率化的修正年销售量。与预期销售量的修正类似，根据不同类别农作物决策信息的增减或浮动对销售单价，亩产量，种植成本均进行概率化修正，并根据市场供需的变化对作物种植总量的稳定性约束进行了修正。在此基础上使用 CBC 求解器对优化模型求解，最终得到考虑作物种植及市场条件变化后 2024~2030 年的每年总利润，并与第一问的模型结果进行比较分析，得出结论。

2.4 问题三的分析

问题三的核心是在问题二的基础上再引入有关不确定因素的复杂约束，如作物的可替代性和互补性，销售价格与种植成本、销量之间的关系等。在更加严格的约束条件下进行最优化问题的求解。首先，我们将所有作物细化为更详细的 7 个类别，利用类别之间的作物可替代性和作物的营养成分互补性构造更严格的稳定性约束。然后，利用线性模型拟合成本与销售价的正相关关系，利用对数正态模型拟合销售价与销量的关系。最后，分析各个季度之间时间期限的不同，考虑各个品种蔬菜的全年供应情况构造新的市场状态约束。在此基础上使用 CBC 求解器对优化模型求解，最终得到考虑作物种植及市场条件变化后 2024~2030 年的每年总利润，并与第二问的模型结果进行比较分析，得出结论。

三、 模型假设

- 假设一：2023 年销售单价区间数据来源于大数据统计，符合中心极限定理，服从正态分布。且区间上下限符合”3σ” 原则的约束。

- 假设二：2023 年各种作物的种植量和亩产量已经经过历史数据的预测，其预期销量的总体均值与预期产量的总体均值相等。
- 假设三：预期销售量、亩产量、种植成本、销售价格等指标的上涨或下跌为总体均值变动。
- 假设四：土地排序时，按照编号的字典序进行。
- 假设五：相互可替代性的作物，其市场需求也具有可替代性。

四、 符号说明

4.1 数学符号

符号	说明	单位
$Price_i^k$	在第 k 年作物 i 的年销售单价	元/斤
ES_i^k	在第年作物 i 的年预期销量	斤/亩
Y_i^k	在第 k 年作物 i 的年亩产量	斤/亩
$Cost_i^k$	在第 k 年作物 i 的年种植成本	斤/亩
$Plant_i^k$	在第 k 年作物 i 的年种植数量	亩
XP_i^k	在第 k 年是否种植作物 i	
A_i^k	在第 k 年种植作物 i 的总量	斤
$LandSize_j$	第 j 块土地的面积	斤
$\boldsymbol{P}$	单价矩阵	元/斤
$\boldsymbol{C}$	种植成本矩阵	元/亩
$\boldsymbol{Y}$	产量矩阵	斤/亩
$\boldsymbol{S}$	销量矩阵	斤/亩
$\boldsymbol{\Pi}$	利润矩阵	元
$\boldsymbol{P}^*$	修正后的单价矩阵	元/斤
$\boldsymbol{C}^*$	修正后的种植成本矩阵	元/亩
$\boldsymbol{Y}^*$	修正后的产量矩阵	斤/亩
$\boldsymbol{S}^*$	修正后的销量矩阵	斤/亩
$\boldsymbol{\Pi}^*$	修正后的利润矩阵	元
λ	惩罚项权重	

4.2 标号约定

为了简化后续处理，约定以表2中的符号代表不同地块在不同季度的种植情况，同时采用作物标号来辨识作物。特别地，由于水浇地的种植规律为每年种植一季水稻或两

表 2　地块地形的标号约定

标号	解释	标号	解释	标号	解释
A{n}	第 n 块平旱地的种植规划	D{n}-1	第 n 块水浇地第一季度的种植规划	D{n}-2	第 n 块水浇地第二季度的种植规划
B{n}	第 n 块梯田的种植规划	E{n}-1	第 n 块普通大棚第一季度的种植规划	E{n}-2	第 n 块普通大棚第二季度的种植规划
C{n}	第 n 块山坡地的种植规划	F{n}-1	第 n 块智慧大棚第一季度的种植规划	F{n}-2	第 n 块智慧大棚第二季度的种植规划

季蔬菜，因此当 D{n} 决定种植水稻时，D{n}-1 应该种植全部的面积，同时 D{n}-2 不应该种植任何作物。因此，在本研究中，存在一个地块标号之间的固有约束。

固有约束 1: 如果 D{n}-1 或 D{n}-2 种植了水稻，那么 D{n} 的总面积都必须种植水稻，且不能被拆分种植其他作物。如果 D{n}-1 或 D{n}-2 没有种植水稻，那么它们可以选择种植其他作物。

五、 问题一模型的建立与求解

5.1 决策信息准备

5.1.1 概率化预期销售量

由于各种农作物未来的预期销售量、种植成本、亩产量和销售价格相对于 2023 年保持稳定，每季种植的农作物在当季销售，采用变异系数 (CV) 对预期销售量进行稳定性约束 [4]。根据假设二，可以认为预期销量的总体均值与预期产量的总体均值相等 (即 $E(Y_i) = E(ES_i)$)，在变异系数为 10% 的条件下，认为作物 i 的年预期销售量 ES_i 服从正态分布:

$$CV = \sigma_i/\mu_i \times 100\% = 10\% \tag{1}$$

$$ES_i \sim N\left(E(Y_i), (CV \times E(Y_i))^2\right) \tag{2}$$

基于均匀分布的概率密度函数，随机取样一个样本 X，作为 ES_i 的观测值，代表作物 i 的年销售量。

5.1.2 正态化销售单价

观察到样本的 “销售单价/(元/斤)” 数据均处于某一区间 $[L, U]$ 内，基于假设一，将区间上下限分别作为 $\mu + 3\sigma$ 和 $\mu - 3\sigma$ 的边界。对作物 i 的年销售单价 $Price_i$ 进行估计:

$$\mu_i = \frac{L + U}{2} \tag{3}$$

$$\sigma_i = \frac{U - L}{6} \tag{4}$$

$$Price_i \sim N(\mu_i, \sigma_i^2) \tag{5}$$

其中,(μ_i, σ_i) 代表总体均值为 μ_i，总体标准差为 σ_i 的正态分布。基于正态分布的概率密度函数，随机取样一个样本 X，作为 $Price_i$ 的观测值，代表作物 i 的年销售单价。

5.2 约束条件构建

5.2.1 稳定性约束

1) 市场稳定性

由于市场供需关系相对于 2023 年保持稳定，在保证每种作物在单个地块上的种植面积同时，还要保证每种作物的种植总量维持相对稳定。在变异系数为 10% 的条件下,作物 i 的相邻两年的年种植数量 $Plant_i$ 服从均匀分布。

$$Plant_i^k \sim U[(1-CV) \times Plant_i^{2023}, (1+CV) \times Plant_i^{2023}] \tag{6}$$

其中，$Plant_i^k$ 表示第 k 年第 i 种作物的种植数量。

稳定性约束 1: 每种作物种植的总量在 0.9*(2023 年种植量)~1.1*(2023 年种植量)之间。

2) 产量稳定性

为了不同作物连续两年的产量稳定，避免土地闲置的资源浪费，每块地上的种植面积应等于这块地的面积。同时，为了方便耕种作业和田间管理，每块土地耕种的作物不应该过多。分析 2023 年的耕种数据可得，每块土地最多耕种作物不超过 2 种。因此，对于维持产量稳定性设置的约束条件如下:

稳定性约束 2: 每块地的种植面积必须等于该地的可用面积。
稳定性约束 3: 每块地上必须种植作物，且种植的作物种类至多两种。

5.2.2 土地约束

本研究中的村庄土地主要分为平旱地 (A)、梯田 (B)、山坡地 (C)、水浇地四类 (D)、普通大棚 (E) 和智慧大棚 (F)6 类，不同类型的土地具备不同的生产条件和作物种植限制。平旱地、梯田和山坡地每年适宜种植一季粮食作物 (除水稻)，而水浇地则适宜种植一季水稻或两季蔬菜。这表明了土地类型的差异在一定程度上限制了农业生产的多样性与频次。土地耕种时，不同的作物需要耕种在不同的土地和不同的季度，以保证作物的正常生长，具体约束如下:

土地约束 1: 平旱地、梯田、山坡地 (A,B,C) 只能种粮食类作物，即作物 1-作物 15。
土地约束 2: 第一季度的水浇地 (D{n}-1) 只能种植水稻或非大白菜、白萝卜和红萝卜的蔬菜，即作物 16-34。
土地约束 3: 第二季度的水浇地 (D{n}-2) 只能种植大白菜、白萝卜和红萝卜，即作物 35-37。
土地约束 4: 第一季度的普通大棚 (E{n}-1) 只能种植非大白菜、白萝卜和红萝卜的蔬菜，即作物 17-34。
土地约束 5: 第二季度的普通大棚 (E{n}-2) 只能种植食用菌，即作物 38-41。
土地约束 6: 智慧大棚 (F) 只能种植非大白菜、白萝卜和红萝卜的蔬菜，即作物 17-34。

5.2.3 状态约束

1) 轮作约束

根据农作物的生长规律，若同一地块（含大棚）连续种植同一种作物，会导致土壤中的特定养分耗竭、病虫害积累及土壤结构恶化，最终导致作物减产。因此，轮作制度成为农业生产中维持土壤健康和作物产量的关键手段，通过不同作物的交替种植，有效避免重茬对土壤和作物的不利影响，确保土地的长期生产力。

设二进制变量 XP_i^k 表示第 k 年是否种植第 i 种作物，则:

$$XP_i^k = \begin{cases} 1 & \text{if k 年种植作物 i} \\ 0 & \text{if k 年不种植作物 i} \end{cases} \tag{7}$$

轮作的约束条件可以表示为:

$$XP_i^k + XP_i^{k+1} \leq 1 \tag{8}$$

状态约束 1: 对于平旱地、梯田、山坡地 (A,B,C)，相邻两年间种植的作物不允许相同。对于水浇地、普通大棚和智慧大棚 (D,E,F)，下一年的第一季作物和本年份第二季作物不允许相同。

2) 豆类约束

从农业生态学的角度来看，豆类作物对土壤具有特殊的益处，因为其根部的根瘤菌能够与土壤中的氮气发生固氮作用，增加土壤中的氮含量，从而有效提高土壤肥力。这一过程不仅能够增强豆类自身的生长，还能为后续种植的其他作物提供更丰富的养分，促进作物健康生长。因此，出于提升土壤质量和保持农田长期生产力的考虑，自 2023 年起，规定每个地块（包括大棚在内）的所有土地在三年内必须至少轮作种植一次豆类作物。设二进制变量 XP_i^k 表示第 k 年是否种植第 i 种作物,L 表示所有豆类作物的集合，

则从 2023 年后的第三年开始种植需要满足：

$$\sum_{i \in L} \left(XP_i^k + XP_i^{k-1} + XP_i^{k-2} \right) \geq 1, \quad \forall k \geq 3 \tag{9}$$

状态约束 2: 从 2025 年起，每三年间所有土地上至少种植一次豆类作物。

5.3 目标函数建立

5.3.1 利润度量

为了最大化种植的收益，首先根据 2023 年的数据，构建 41 种作物在每块地上的单价矩阵 ($\boldsymbol{P}$)、成本矩阵 ($\boldsymbol{C}$)、产量矩阵 ($\boldsymbol{Y}$) 和销量矩阵 ($\boldsymbol{S}$)。具体构建如下：

单价矩阵 $\boldsymbol{P}$ 用来表示不同地块上每种作物的年销售单价，矩阵的元素 $P_{i,j}$ 表示在第 i 块地上种植第 j 种作物的单价，单位是元/斤。

$$P_{i,j} = \frac{P_{\text{max},j} + P_{\text{min},j}}{2} + \sigma_j \cdot Z \tag{10}$$

其中, $P_{\text{max},j}$ 和 $P_{\text{min},j}$ 分别为作物 j 的单价区间的最大值和最小值，$\sigma_j = \frac{P_{\text{max},j} - P_{\text{min},j}}{6}$ 表示作物 j 单价的标准差，Z 为从标准正态分布 [3] $\mathcal{N}(0,1)$ 中随机抽取的一个值，确保价格的随机性，$P_{i,j}$ 在区间 $[P_{\text{min},j}, P_{\text{max},j}]$ 之间。

成本矩阵 $\boldsymbol{C}$ 表示每块地种植每种作物的年种植成本，矩阵的元素 $C_{i,j}$ 表示第 i 块地上种植第 j 种作物的成本，单位是元/亩。

$$C_{i,j} = c_j \tag{11}$$

其中，c_j 表示作物 j 的固定种植成本（根据作物类型来确定）。

产量矩阵 $\boldsymbol{Y}$ 用来表示每块地上种植作物的亩产量，矩阵的元素 $Y_{i,j}$ 表示第 i 块地上第 j 种作物的亩产量，单位是斤/亩。

$$Y_{i,j} = y_j \tag{12}$$

其中，y_j 为作物 j 在标准条件下的固定亩产量，单位为斤/亩。

销量矩阵 $\boldsymbol{S}$ 用来表示每块地上种植每种作物的年预期销量，矩阵的元素 $S_{i,j}$ 表示第 i 块地上第 j 种作物的亩销售量，单位是斤/亩。基于变异系数为 10% 的随机销售量，公式表示如下：

$$S_{i,j} = Y_{i,j} \cdot (1 + \epsilon) \tag{13}$$

其中，$\epsilon \in [-0.1, 0.1]$ 为在区间内均匀分布的随机数，表示产量的波动范围（±10%），$Y_{i,j}$ 为前述的产量矩阵元素，表示标准条件下的亩产量。

根据这四个矩阵构建的两个利润矩阵目标函数的数学解析式如下。利润矩阵 $\mathbf{\Pi}$ 计算的是销售产量与种植成本的差异。在超过部分滞销，造成浪费的条件下，利润矩阵的计算方法为：

$$\Pi_{i,j}^{(A)} = P_{i,j} \cdot \min(Y_{i,j}, S_{i,j}) - C_{i,j} \tag{14}$$

在超过部分按 2023 年销售价格的 50% 降价出售的条件下，利润矩阵的计算方法为：

$$\Pi_{i,j}^{(B)} = P_{i,j} \cdot \min(Y_{i,j}, S_{i,j}) - C_{i,j} + 0.5 \cdot P_{i,j} \cdot \text{ReLU}(Y_{i,j} - S_{i,j}) \tag{15}$$

其中,$P_{i,j}$ 是第 i 块地种植作物 j 的销售单价; $Y_{i,j}$ 是第 i 块地种植作物 j 的亩产量; $S_{i,j}$ 是第 i 块地种植作物 j 的亩销售量; $C_{i,j}$ 是第 i 块地种植作物 j 的种植成本; $ReLU(Y_{i,j}-S_{i,j})$ 表示滞销部分的作物量，即实际产量超过销售量的部分。$ReLU$ 是一种常见的激活函数，经常被用于神经网络训练等问题，它的解析式如下：

$$ReLU(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{16}$$

在得到利润矩阵 $\mathbf{\Pi}$ 后，需要最大化 $\mathbf{\Pi}$ 中所有元素之和，以获得利润的最大值，利润部分目标函数如下：

$$\text{Maximize} \sum_i \sum_j \Pi_{i,j} \cdot Plant_{i,j} \tag{17}$$

5.3.2 作物密集程度度量

在获得最大利润的同时，还需要保证每种作物在每个季节的种植地块不能过于分散，以便于管理、灌溉、收割等农业操作的集中化和高效化。这可以通过限制每种作物的地块集中度来实现。

为实现这一目标，可以在目标函数中引入空间约束条件，确保它们集中在相邻地块上。曼哈顿距离可以用于衡量种植地块的分散程度。通过引入曼哈顿距离，可以控制同一作物在不同地块上的种植位置，以避免种植过于分散。

曼哈顿距离（Manhattan Distance）[1] 是度量两点间距离的一种方式，适用于网格状的布局。对于两个点 $A(x_1, y_1)$ 和 $B(x_2, y_2)$ 来说，曼哈顿距离定义为：

$$d(A, B) = |x_1 - x_2| + |y_1 - y_2| \tag{18}$$

该距离表示在网格中从一个点到另一个点需要经过的横向和纵向的步数之和。在作物种植问题中，曼哈顿距离用于衡量同一种作物在不同地块上的种植位置的分散程度。若同一种作物的种植地块相距较远，曼哈顿距离会较大，意味着种植布局较分散。通过惩罚这一距离，模型可以倾向于选择相对集中种植的方案。对于同一种作物 i 的所有地块 j，曼哈顿距离惩罚的计算可以表示为：

$$\text{ManhattanDensity}(\boldsymbol{XP}) = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k=j+1}^{n} (|XP_{j1} - XP_{k1}| + |XP_{j2} - XP_{k2}|) \quad (19)$$

其中,(XP_{j1}, XP_{j2}) 表示地块 j 在 $\boldsymbol{XP}$ 矩阵中的位置坐标,(XP_{k1}, XP_{k2}) 表示地块 k 在 $\boldsymbol{XP}$ 矩阵中的位置坐标,n 表示种植作物 i 的地块数量。$XP_{i,j}^{k}$ 是之前定义的二进制变量，它表示在第 k 年，第 j 块地是否种植作物 i。由于 $\boldsymbol{XP}$ 矩阵在构造时已经按照地块号和作物编号排序，故其曼哈顿距离密度能反映出种植的离散程度。

在目标函数中，曼哈顿距离惩罚项的作用是减少地块种植的分散度，具体形式为:

$$\lambda \cdot \text{ManhattanDensity}(\boldsymbol{XP}) \quad (20)$$

其中 λ 是惩罚权重，它控制了曼哈顿距离在总目标函数中的重要性。如果 λ 取值较大，模型会更加倾向于集中种植作物，减少地块的分散度。如果 λ 取值较小，利润最大化可能占据主导地位，分散种植的惩罚作用会减弱。

5.3.3 综合目标函数

最终目标函数是利润最大化与曼哈顿距离惩罚的结合:

$$\text{Maximize} \sum_{i} \sum_{j} \Pi_{i,j} \cdot Plant_{i,j} - \lambda \cdot \text{ManhattanDensity}(\boldsymbol{XP}) \quad (21)$$

通过计算种植地块的曼哈顿距离，可以有效控制种植布局的分散度，确保作物种植相对集中，便于管理、灌溉、施肥等操作，降低农业管理的复杂性和成本。引入曼哈顿距离作为目标函数的一部分，有助于平衡利润最大化与种植布局优化之间的关系，从而提高整体的种植效率和资源利用率。

5.4 多级约束条件下优化模型的分层求解

Phase I 目标函数确立

针对以下两种情况，采用不同的目标函数建立最优化模型，其中超参数 $\lambda = 10$。

❶ 超过部分滞销，造成浪费;

$$\text{Maximize} \quad [P_{i,j} \cdot \min(Y_{i,j}, S_{i,j}) - C_{i,j}] \cdot Plant_{i,j} - \lambda \cdot \text{ManhattanDensity}(\boldsymbol{XP}) \quad (22)$$

❷ 超过部分按 2023 年销售价格的 50% 降价出售。

$$\begin{aligned} \text{Maximize} \quad & [P_{i,j} \cdot \min(Y_{i,j}, S_{i,j}) - C_{i,j} + 0.5 \cdot P_{i,j} \cdot \text{ReLU}(Y_{i,j} - S_{i,j})] \cdot Plant_{i,j} \\ & - \lambda \cdot \text{ManhattanDensity}(\boldsymbol{XP}) \end{aligned} \quad (23)$$

Phase II 定量约束求解

基于5.2节中提到的稳定性约束、土地约束和状态约束 [2]，利用 CBC 求解器进行优化求解，具体约束条件如下：

$$
s.t.\begin{cases}
0.9 \cdot A_j^{2023} \leq \sum_i Plant_{i,j} \leq 1.1 \cdot A_j^{2023} & \text{稳定性约束1} \\
\sum_j Plant_{i,j} = LandSize_i \quad \forall i & \text{稳定性约束2} \\
\sum_j XP_{i,j} \leq 2 \quad \forall i & \text{稳定性约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{A,B,C\}, \quad \forall j > 15 & \text{土地约束1} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-1,...,D8-1\}, \quad \forall j \notin [16,34] & \text{土地约束2} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-2,...,D8-2\}, \quad \forall j \notin \{16,35,36,37\} & \text{土地约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-1,...,E16-1\}, \quad \forall j \notin [17,34] & \text{土地约束4} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-2,...,E16-2\}, \quad \forall j \notin [38,41] & \text{土地约束5} \\
Plant_{i,j} = 0 \quad \forall i \in \{F\}, \quad \forall j \notin [17,34] & \text{土地约束6} \\
Plant_{i,j}^k = 0 \quad \text{if } Plant_{i,j}^{k-1} > 0 & \text{状态约束1} \\
\sum_{k=k-2}^{k} XP_{i,j}^k \geq 1 \quad \forall j \in L & \text{状态约束2}
\end{cases} \tag{24}
$$

其中，$LandSize_i$ 是第 i 块地的面积，L 是豆类植物编号集合。

Phase III 固有约束修正

基于4.2节中提到的固有约束，需要对 Phase II 中得到的优化结果进行修正，设 Phase II 优化结果为 $Plant_{i,j}^{initial}$，最终优化结果的修正方程如下：

$$
Plant_{i,j}^{final} = \begin{cases}
LandSize_i, & \text{if } i \in \{D1-1,...,D8-1\} \text{ and } j = 16 \\
0, & \text{if } i \in \{D1-1,...,D8-1\} \text{ and } j \neq 16 \\
0, & \text{if } i \in \{D1-2,...,D8-2\} \text{ and } j \neq 16 \\
Plant_{i,j}^{initial}, & \text{otherwise}
\end{cases} \tag{25}
$$

Phase IV 迭代求解

借助了马尔科夫链的构造思想，独立为 2024-2030 年设立七个优化器，并采用 CBC 算法进行迭代求解。即，利用上一年优化出的种植方案，作为决策条件用于下一年的优化模型求解。通过迭代 7 次后，可以获得 2024-2030 每年的的最佳种植方案。

5.5 结果分析

本节分析了独立运行 10 次多级约束条件下优化模型后的求解结果均值。我们分别计算了 20242030 年的利润目标函数 1 和目标函数 2 的收益，并且将两组数据的结果进行了对比。2024-2030 年的具体种植规划详见附件B。两种目标函数的具体年利润如表3所示。

从表3中可以看出，利润目标函数 1 和目标函数 2 的年利润存在一定差异。在前 7 年的时间跨度中，目标函数 1 的平均利润为 6,051,223.81 元，而目标函数 2 的平均利润

表 3　优化模型 10 次求解后的利润均值

策略	2024	2025	2026	2027	2028	2029	2030	平均值
策略❶	5,911,519.99	6,080,888.65	6,226,563.84	5,737,603.47	6,472,374.85	5,947,270.12	5,992,341.75	6,051,223.81
策略❷	6,722,259.38	6,048,121.17	6,382,636.91	6,157,613.55	6,264,120.64	6,418,338.27	6,322,644.76	6,337,952.10

为 6,337,952.10 元。目标函数 2 由于引入了滞销部分以 50% 折扣出售的机制，整体利润略高于目标函数 1。目标函数 2 的平均利润比目标函数 1 高出约 4.7%，这表明引入滞销部分折扣出售机制有助于整体利润的提升。此外，目标函数 2 的年利润波动较小，而目标函数 1 在第 4 年和第 5 年之间的波动较大，这可能与引入的销售价格和预期销量的随机性有关。考虑到农业种植的复杂性和多变性，目标函数 2 的稳定性和利润提升在实际应用中更具有可操作性。

六、 问题二模型的建立与求解

6.1 决策信息修正

6.1.1 销售量修正

根据经验,小麦和玉米未来的预期销售量有增长的趋势,平均年增长率介于 5%~10% 之间,其他农作物未来每年的预期销售量相对于 2023 年大约有 ±5% 的变化。在5.1.1节的基础上，将变化趋势引入年销售量中，以获得概率化的修正年销售量。具体如下:

$$\begin{cases} ES_i^{k+1} = ES_i^k \times (1+\alpha), \quad i \in \{6,7\} \\ \alpha \sim U[0.05, 0.1] \end{cases} \tag{26}$$

$$\begin{cases} ES_i^{k+1} = ES_i^{2023} \times (1+\alpha), \quad i \in [1,5] \cup [8,16], \quad i \in \mathbb{Z} \\ \alpha \sim U[-0.05, 0.05] \end{cases} \tag{27}$$

为了研究年销售量的变化规律以及验证模拟的合理性，我们独立重复了 50 次销售量生成算法，并绘制出了 2024~2030 的变化趋势图，其中，单次模拟的结果如图2所示，50 次重复实验的结果如图3所示。

6.1.2 销售单价修正

粮食类作物的销售价格整体上较为稳定，通常不会出现大幅波动。相比之下，蔬菜类作物的销售价格则呈现出逐年增长的趋势，平均每年约上涨 5% 左右。食用菌的销售价格与蔬菜类作物相比则呈现出不同的趋势，整体上稳中有降，每年下降幅度大约在 1% 到 5% 之间。其中，羊肚菌作为一种高端食用菌，价格下降的幅度尤为显著，几乎

每年以 5% 的速度下滑。在5.1.2节的基础上，将变化趋势引入销售单价中，以获得概率化的修正销售单价。具体如下：

$$\begin{cases} Price_i^{k+1} = Price_i^k \times 1.05, \quad i \in [17, 37], \quad i \in \mathbb{Z} \\ Price_i^{k+1} = Price_i^k \times (1-\alpha), \quad i \in \{38, 39, 40\} \\ Price_i^{k+1} = Price_i^k \times 0.95, \quad i = 41 \\ \alpha \sim U[0.01, 0.05] \end{cases} \tag{28}$$

为了研究年销售单价的变化规律以及验证模拟的合理性，我们同样独立重复了 50 次销售量生成算法，并绘制出了 2024~2030 的变化趋势图，其中，单次模拟的结果如图2所示，50 次重复实验的结果如图3所示。

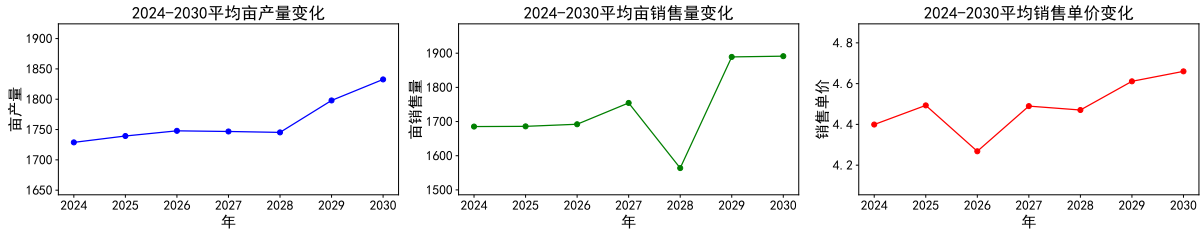

图 2　2024~2030 年的平均亩产量、销售量、销售单价变化模拟

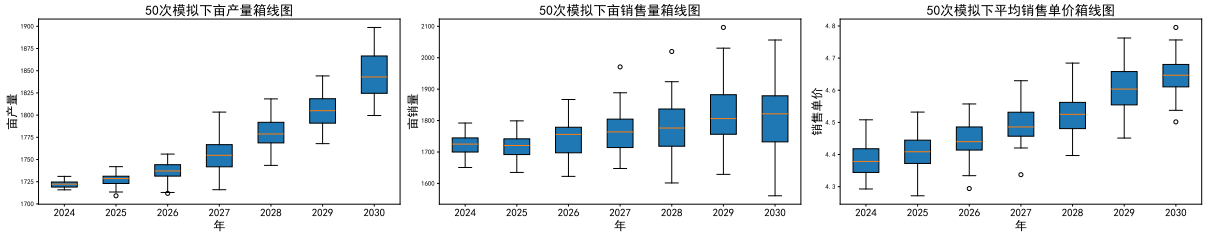

图 3　50 次独立重复模拟后，2024~2030 年的平均亩产量、销售量、销售单价变化趋势

6.1.3 亩产量修正

农作物的亩产量往往会受到多种因素的综合影响，其中气候条件是最为关键的因素之一。气温、降水、日照时长以及极端天气事件等，都会直接或间接地影响作物的生长情况。因此，农作物的亩产量每年都会出现一定程度的波动，通常在 ±10% 的范围内。在 2023 年亩产量的基础上，将变化趋势引入年亩产量中，以获得修正年亩产量。具体如下：

$$\begin{cases} Y_i^{k+1} = Y_i^k \times (1+\alpha) \\ \alpha \sim U[-0.1, 0.1] \end{cases} \tag{29}$$

为了研究年亩产量的变化规律以及验证模拟的合理性，我们同样独立重复了 50 次销售量生成算法，并绘制出了 2024~2030 的变化趋势图，其中，单次模拟的结果如图2所示，50 次重复实验的结果如图3所示。

6.1.4 种植成本修正

由于市场条件的变化，农作物的种植成本呈现出每年平均增长 5% 左右的趋势。这一现象主要受到多种因素的共同影响，包括农业生产资料价格的上涨、人工成本的增加以及能源价格的波动等。在 2023 年种植成本的基础上，将变化趋势引入种植成本中，以获得修正年种植成本。具体如下：

$$Cost_i^{k+1} = Cost_i^k \times 1.05 \tag{30}$$

6.2 约束条件修正

由于市场供需关系的变化，需要即使调整种植策略以保证每年耕地获得最高的收益值。因此5.2.1节中提出的稳定性约束需要被修正。在保证每种作物在单个地块上的种植面积同时，需要即使调整每种作物的种植总量，以适应市场需求和环境变化。因此，我们提高了变异系数为 30%，同时将变化趋势引入种植策略的约束，具体如下：

$$Plant_i^k \sim U[(1-CV) \times Plant_i^{k-1}, (1+CV) \times Plant_i^{k-1}] \tag{31}$$

其中，$Plant_i^k$ 表示第 k 年第 i 种作物的种植数量。

修正稳定性约束 1: 每种作物种植的总量在 0.7*(上一年种植量)~1.3*(上一年种植量) 之间。

6.3 优化模型的求解

我们延用了5.4节的求解步骤，在新的决策信息、约束条件和目标函数下进行求解。为了最大化收益，不妨在超出销售量的部分按 2023 年销售价格的 50% 降价出售的前提下最大化目标函数。利用修正后的信息构造目标函数，并保证作物种植的密集性，其中超参数 $\lambda = 10$。基于5.2和6.2节中提到的修正稳定性约束、土地约束和状态约束，利用 CBC 求解器进行优化求 [5]，具体目标函数约束条件如下：

$$\begin{aligned} \text{Maximize} \quad & [P_{i,j}^* \cdot \min(Y_{i,j}^*, S_{i,j}^*) - C_{i,j}^* + 0.5 \cdot P_{i,j}^* \cdot \text{ReLU}(Y_{i,j}^* - S_{i,j}^*)] \cdot Plant_{i,j} \\ & -\lambda \cdot \text{ManhattanDensity}(\boldsymbol{XP}) \end{aligned} \tag{32}$$

$$
s.t.\begin{cases}
0.7 \cdot A_j^{k-1} \le \sum_i Plant_{i,j} \le 1.3 \cdot A_j^{k-1} & \text{修正稳定性约束1} \\
\sum_j Plant_{i,j} = LandSize_i \quad \forall i & \text{稳定性约束2} \\
\sum_j XP_{i,j} \le 2 \quad \forall i & \text{稳定性约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{A,B,C\}, \quad \forall j > 15 & \text{土地约束1} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-1,...,D8-1\}, \quad \forall j \notin [16,34] & \text{土地约束2} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-2,...,D8-2\}, \quad \forall j \notin \{16,35,36,37\} & \text{土地约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-1,...,E16-1\}, \quad \forall j \notin [17,34] & \text{土地约束4} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-2,...,E16-2\}, \quad \forall j \notin [38,41] & \text{土地约束5} \\
Plant_{i,j} = 0 \quad \forall i \in \{F\}, \quad \forall j \notin [17,34] & \text{土地约束6} \\
Plant_{i,j}^k = 0 \quad \text{if } Plant_{i,j}^{k-1} > 0 & \text{状态约束1} \\
\sum_{k=k-2}^{k} XP_{i,j}^k \ge 1 \quad \forall j \in L & \text{状态约束2}
\end{cases} \tag{33}
$$

其中，$LandSize_i$ 是第 i 块地的面积，L 是豆类植物编号集合。求解出的初步结果经过5 .4节 Pahse III 的修正后获得最终的种植策略。

6 .4 结果分析

根据题目中预期销售量、亩产量、种植成本和销售价格的不确定性进行修正后，我们在超出销售量的部分按 2023 年销售价格的 50% 降价出售的前提下对优化模型进行求解，得出了独立运行 10 次多后的求解结果均值。具体的种植策略详见附录C。

表 4　考虑作物种植及市场条件变化后 10 次求解的年总利润

年份	2024	2025	2026	2027	2028	2029	2030	平均值
总利润	6,130,851.69	6,548,442.52	6,662,584.76	6,582,620.03	7,037,166.10	7,309,529.98	7,451,224.33	6,817,488.49

从表4可以看出，根据题目信息考虑作物种植及市场条件的逐年变化后，总利润得到了进一步提高，在同样策略下（超过部分按 2023 年销售价格的 50% 降价出售），总利润均值从第一问的 6,337,952.10 元增加到 6,817,488.49 元，共增长约 7.6%。这表明考虑 2023 往后七年农作物的预期销售量、亩产量、种植成本和销售价格的浮动变化之后，尽管存在作物成本浮动以及食用菌出现价格逐年下滑等现象，但总体上来看年利润仍是呈逐年增加的态势，这主要是因为：❶绝大部分粮食作物因其日常基础食品的特性，需求相对稳定，价格波动较小，出现销售量急剧下滑的可能性很低，保证了一大部分收入的稳定性。❷小麦与玉米两种重要的日常粮食作物销售量以 5%-10% 的增长率增加。❸蔬菜类作物销售价格则呈现出逐年增长的趋势，平均每年约上涨 5% 左右。可见市场对有机蔬菜领域的需求程度稳步提高，价格增幅最为明显。成为总利润上升的另一大因素。

七、 问题三模型的建立与求解

7.1 可替代性分析

可替代性的产生往往与作物的类别息息相关，且同类的作物一般具有可替代性，因为它们在营养成分、用途、种植条件等方面具有相似性。这种相似性使得在市场供应短缺或价格波动时，消费者和生产者能够在同类作物之间进行选择。

为了研究所给 41 种作物的可替代性，我们将作物进一步分为六类，表5展示了各类作物及其分类依据。

表 5　41 种作物的进一步分类方式和分类依据

类别	编号	名称	分类依据
豆类粮食	1~5	黄豆、黑豆、红豆、绿豆、爬豆	豆科植物的种子，主要作为粮食作物种植，富含蛋白质，用于主食或食品加工。
豆类蔬菜	17~19	豇豆、刀豆、芸豆	豆科植物，主要作为蔬菜食用，富含蛋白质和膳食纤维，常用于鲜食。
淀粉类	13, 20	红薯、土豆	富含淀粉，根茎类或块茎类作物，常用于主食、饲料及加工食品中。
谷物类	6~11, 14~16	小麦、玉米、谷子、高粱、黍子、荞麦、莜麦、大麦、水稻	禾本科或谷类植物，主要作为粮食作物种植，富含碳水化合物，是主要能量来源。
叶菜类蔬菜	23, 26~28, 30, 32~35	菠菜、包菜、油麦菜、小青菜、生菜、空心菜、黄心菜、芹菜、大白菜	主要食用叶子，富含维生素、矿物质和膳食纤维，常作为绿叶蔬菜。
根茎类蔬菜	36, 37	白萝卜、红萝卜	根茎类作物，主要食用根部，富含维生素和矿物质，常用于烹饪。
果菜类蔬菜	12, 21~22, 24~25, 29, 31	南瓜、西红柿、茄子、青椒、菜花、黄瓜、辣椒	主要食用果实，富含维生素、矿物质，常用于烹饪或生食。
食用菌	38~41	榆黄蘑、香菇、白灵菇	食用菌类，富含蛋白质、纤维及微量元素，具有较高营养价值。

基于假设五，需要适当放宽5.2.1节中的稳定性约束 1，同时，需要以每类作物的总销售量之和代替6.1.1节中的销售量估计，以保证同类作物之间的可替代性。对于每个作物类别 M，其中作物编号为 $i \in M$，它们的总种植面积需要限制在上一年年同类别作物种植面积的 0.9 倍到 1.1 倍之间。具体公式如下：

$$0.7 \times \sum_{i \in M} \text{Plant}_i^{k-1} \leq \sum_{i \in M} \text{Plant}_i^{k} \leq 1.3 \times \sum_{i \in M} \text{Plant}_i^{k-1} \tag{34}$$

其中，$\sum_{i \in M} \text{Plant}_i^{k}$ 表示类别 M 在第 k 年所有作物的总种植面积；$\sum_{i \in M} \text{Plant}_i^{k-1}$ 表示类别 M 在第 $k-1$ 年的总种植面积。

可替代稳定性约束 1: 每类作物种植的总量在 0.9*(上一年种植量)~1.1*(上一年种植量) 之间。

7.2 互补性分析

在作物的营养成分含量上，豆类和谷物的营养互补，豆类作物富含蛋白质，而谷物作物则主要提供碳水化合物，两者在膳食中可以互补。除此之外，蔬菜类和菌类作物的营养互补，蔬菜类作物（如菠菜、大白菜、包菜等）富含维生素和膳食纤维，而菌类作物（如香菇、白灵菇）富含蛋白质和微量元素，它们在营养上有良好的互补作用。烹饪中，菌类常与蔬菜一起炒制，既提升了风味，也丰富了营养。

与5.2.3节提到的状态约束相似，在耕种时，作物的选择不仅仅是为了提高产量，还需要考虑作物在营养成分上的互补性，以满足不同人群的饮食需求。通过合理的轮作和混合种植，可以优化作物的营养价值。基于这两种营养成分上的互补，我们提出了下面两个额外的状态约束：

状态约束 3: 如果计划种植的作物中存在豆类粮食，那么应该种植谷物类。
状态约束 4: 如果计划种植的作物中存在蔬菜类，那么应该种植食用菌。

7.3 种植成本与销售价格分析

种植成本与销售价格通常呈正相关关系，这种关系反映了生产者通过价格调整来应对成本变化的经济行为。当前常用的定价方法有基于竞争的定价法、成本加成定价法等，由于缺乏历史的价格数据和市场竞争数据，我们无法采用复杂的模型拟合成本与销售价格之间的关系。因此，基于成本加成定价法，采用线性模型来限制种植成本与销售价格的关系:

$$\boldsymbol{P} = \boldsymbol{W} \cdot \boldsymbol{C} + \Theta \tag{35}$$

其中 $\boldsymbol{W}$ 是待求解的权重，Θ 是偏置，$\boldsymbol{P}$ 是销售价格矩阵，$\boldsymbol{C}$ 是成本矩阵。

为了拟合线性方程，采用如5.1.1节提到的概率化方法。在变异系数为 10% 的条件下，利用正态分布将种植成本概率化，并基于5.1.2节提到的正态分布，构造一组数据来拟合线性方程。具体如下：

Step1) 计算分布

$$\begin{cases} Cost_i \sim N\left(E(Cost_i), (CV \times E(Cost_i))^2\right) \\ Price_i \sim N(\mu_i, \sigma_i^2) \end{cases} \tag{36}$$

其中，$Cost_i$ 为作物 i 的种植成本，$E(Cost_i)$ 为成本均值已经在数据表中给出，$Price_i$ 为作物 i 的预期定价。

Step2) 生成训练数据

为生成特定概率下的成本和价格值，我们使用正态分布的累积分布函数（CDF）及其逆累积分布函数（分位数函数）。累积分布函数给出了一个随机变量小于某个值的概率，而逆累积分布函数则根据给定的概率 p 反推出相应的值。设成对的 (x, y) 数据点，表示在同一累积概率下的成本 $Cost_i$ 和价格 $Price_i$，遵循以下步骤生成训练集：

- 设定累积概率 p：选择一个在区间 $(0, 1)$ 内的概率值 p。
- 计算成本值 x：根据成本分布的均值 $E(Cost_i)$ 和变异系数 CV，使用正态分布的逆累积分布函数计算成本值 x：

$$x = E(Cost_i) + CV \times E(Cost_i) \cdot \Phi^{-1}(p) \tag{37}$$

- 计算价格值 y：根据价格分布的均值 μ_i 和标准差 σ_i，使用正态分布的逆累积分布函数计算价格值 y：

$$y = \mu_i + \sigma_i \cdot \Phi^{-1}(p) \tag{38}$$

模拟生成的 100 条数据如图4所示。

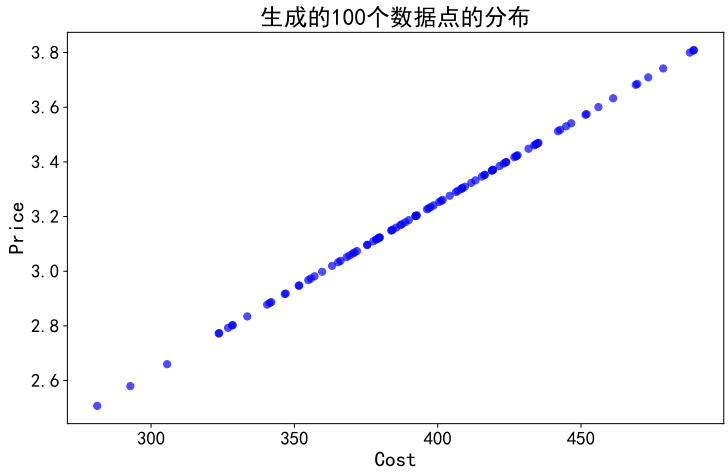

图 4　利用分布生成的 100 条模拟数据点

Step3) 拟合线性方程

在获得了线性数据后，利用最小二乘法拟合函数。最小二乘法是一种经典的优化方法，它通过最小化预测值与实际值之间的平方误差来求解最佳的模型参数。具体来说，给定一组观测数据 $(Cost_i, Price_i)$，最小二乘法优化的目标是最小化以下目标函数：

$$Loss(W, \Theta) = \sum_{i=1}^{n} (Price_i - (W \cdot Cost_i + \Theta))^2 \tag{39}$$

通过最小化该目标函数，最小二乘法优化器可以找到最佳的权重 W 和偏置 Θ，使模型的预测值尽可能接近实际数据。

7.4 销售价格与预期销售量分析

销售价格与预期销售量之间通常表现为反向关系，即价格上升时销售量下降，价格下降时销售量增加。然而，在某些情况下，过低的价格可能会引起消费者对产品质量的质疑，导致销售量减少，因此在较小的价格区间内，价格与销售量可能呈正向关系。整体上，销售价格与预期销售量的关系呈现左侧有峰、右侧有长尾的分布形式，因此选择用对数正态分布来拟合这种关系。

$$ES(Price) = \frac{ES_{\max}}{Price \cdot \sigma \cdot \sqrt{2\pi}} \exp\left(-\frac{(\ln(Price) - \mu)^2}{2\sigma^2}\right) \tag{40}$$

其中，$ES(Price)$ 是给定价格 $Price$ 时的预期销售量；$ES_{\max}$ 是最大预期销售量；σ 是价格的标准差，它控制销售量对价格变化的敏感性；μ 是对数正态分布的均值。

对数正态分布的均值 μ 可以通过 $Price_{\text{mean}}$ 计算：

$$\mu = \ln(Price_{\text{mean}}) \tag{41}$$

标准差 σ 反映了价格的分布范围。我们可以使用5.1.2中提到的标准差来近似代替。

$$\sigma \approx \frac{Price_{\max} - Price_{\min}}{6} \tag{42}$$

通过 $(Price_{\text{mean}}, ES_{\text{mean}}), (Price_{\max}, ES_{\min}), (Price_{\min}, ES_{\max})$ 三点，可以拟合出销售价格与预期销量的对数正态关系。图5展示了拟合后某作物的预期销售量随价格的变化。

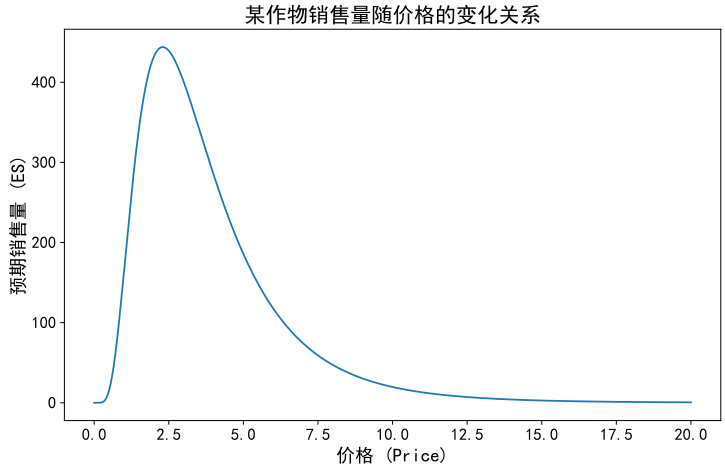

图 5　销售量与价格之间服从的对数正态分布（长尾分布）

7.5 种植季度分析

图6展示了水浇地、普通大棚和智慧大棚三类可以种植蔬菜的时序季度。由于蔬菜在各个季节都是人们的日常食用必需品，因此保证各个种类蔬菜的全年供应也具有重要意义。由于普通大棚第二季度只能种植食用菌，且水浇地的第二季度只能种植大白菜、红萝卜或白萝卜，因此需要保证水浇地第一季度和智慧大棚第一季度能覆盖全部的蔬菜种类，普通大棚第一季度和智慧大棚第二季度也能覆盖全部的蔬菜种类，即可保证全年都可以供应全部品种的蔬菜。我们引入以下两个状态约束来限定：

对于第 k 年，土地 $D\{n\}-1$ 和 $F\{n\}-1$ 的种植组合中必须至少种植一种属于蔬菜类别的作物。

$$XP^k_{v,a} + XP^k_{v,b} \geq 1, \quad \forall v \in V, \forall a \in \{D1-2, ..., D8-2\}, \forall b \in \{F1-1, ..., F4-1\} \tag{43}$$

其中 $XP^k_{v,a}$ 是第 k 年第 a 块土地上是否种植蔬菜作物 v，$XP^k_{v,b}$ 是第 k 年第 b 块土地上是否种植蔬菜作物 v，V 代表蔬菜类别的作物集合

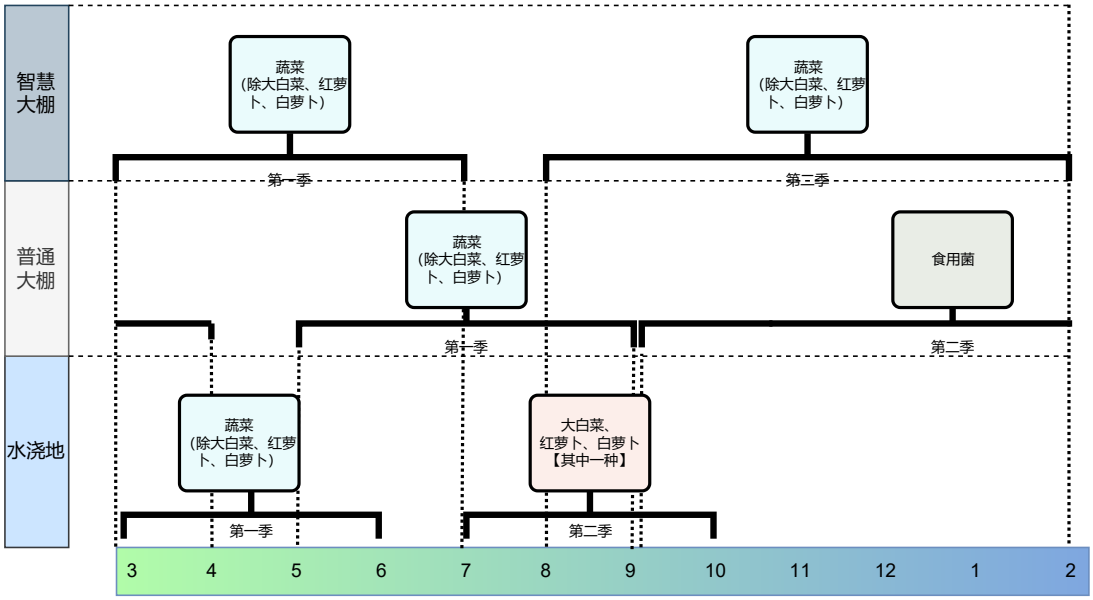

图 6　水浇地、普通大棚和智慧大棚三类可种蔬菜的季度时序

同理，对于第 k 年，土地 $En-1$ 和 $Fn-2$ 的种植组合中必须至少种植一种属于蔬菜类别的作物。

$$XP_{v,a}^{k}+XP_{v,b}^{k}\geq 1,\quad \forall v\in V,\forall a\in\{E1-1,...,E16-1\},\forall b\in\{F1-2,...,F4-2\} \tag{44}$$

其中，$XP_{v,a}^{k}$ 是第 k 年第 a 块土地上是否种植蔬菜作物 v，$XP_{v,b}^{k}$ 是第 k 年第 b 块土地上是否种植蔬菜作物 v，V 代表蔬菜类别的作物集合

状态约束 5: 水浇地第一季度和智慧大棚第一季度的种植策略能覆盖全部的蔬菜种类。

状态约束 6: 普通大棚第一季度和智慧大棚第二季度的种植策略也能覆盖全部的蔬菜种类。

7.6 优化模型的求解

我们延用了6.3节的求解步骤，在新的决策信息、约束条件和目标函数下进行求解。为了最大化收益，不妨在超出销售量的部分按 2023 年销售价格的 50% 降价出售的前提下最大化目标函数。利用修正后的信息构造目标函数，并保证作物种植的密集性，其中超参数 $\lambda=10$。基于5.2和6.2节中提到的修正稳定性约束、土地约束和状态约束，利用 CBC 求解器进行优化求解，具体目标函数约束条件如下：

$$\begin{aligned}\text{Maximize}\quad &[P_{i,j}^{*}\cdot\min(Y_{i,j}^{*},S_{i,j}^{*})-C_{i,j}^{*}+0.5\cdot P_{i,j}^{*}\cdot\text{ReLU}(Y_{i,j}^{*}-S_{i,j}^{*})]\cdot Plant_{i,j}\\&-\lambda\cdot\text{ManhattanDensity}(\boldsymbol{XP})\end{aligned} \tag{45}$$

$$
s.t.\begin{cases}
0.7 \times \sum_{i \in M} \text{Plant}_i^{k-1} \leq \sum_{i \in M} \text{Plant}_i^{k} \leq 1.3 \times \sum_{i \in M} \text{Plant}_i^{k-1} & \text{可替代稳定性约束1} \\
\sum_j Plant_{i,j} = LandSize_i \quad \forall i & \text{稳定性约束2} \\
\sum_j XP_{i,j} \leq 2 \quad \forall i & \text{稳定性约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{A,B,C\}, \quad \forall j > 15 & \text{土地约束1} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-1,...,D8-1\}, \quad \forall j \notin [16,34] & \text{土地约束2} \\
Plant_{i,j} = 0 \quad \forall i \in \{D1-2,...,D8-2\}, \quad \forall j \notin \{16,35,36,37\} & \text{土地约束3} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-1,...,E16-1\}, \quad \forall j \notin [17,34] & \text{土地约束4} \\
Plant_{i,j} = 0 \quad \forall i \in \{E1-2,...,E16-2\}, \quad \forall j \notin [38,41] & \text{土地约束5} \\
Plant_{i,j} = 0 \quad \forall i \in \{F\}, \quad \forall j \notin [17,34] & \text{土地约束6} \\
Plant_{i,j}^{k} = 0 \quad \text{if } Plant_{i,j}^{k-1} > 0 & \text{状态约束1} \\
\sum_{k=k-2}^{k} XP_{i,j}^{k} \geq 1 \quad \forall j \in L & \text{状态约束2} \\
\sum_{l \in L} XP_l^k \leq \sum_{g \in G} XP_g^k, \quad \forall j & \text{状态约束3} \\
\forall j, \quad \sum_{v \in V} XP_v^k \leq \sum_{b \in B} XP_b^k & \text{状态约束4} \\
XP_{v,a}^k + XP_{v,b}^k \geq 1, \quad \forall v \in V, & \\
\forall a \in \{D1-2,...,D8-2\},\ \forall b \in \{F1-1,...,F4-1\} & \text{状态约束5} \\
XP_{v,a}^k + XP_{v,b}^k \geq 1, \quad \forall v \in V, & \\
\forall a \in \{E1-1,...,E16-1\},\ \forall b \in \{F1-2,...,F4-2\} & \text{状态约束6}
\end{cases} \tag{46}
$$

其中，$LandSize_i$ 是第 i 块地的面积，M 是任意一类作物的编号集合，L 是豆类作物编号集合，G 是谷物类作物编号集合, V 是蔬菜类作物编号集合，B 是食用菌类的编号集合。求解出的初步结果经过5 .4节 Pahse III 的修正后获得最终的种植策略。

7.7 结果对比分析

表 6　综合考虑各种因素后的年总利润

年份	2024	2025	2026	2027	2028	2029	2030	平均值
总利润	10,044,860.31	6,320,560.54	7,954,518.26	6,124,002.29	6,532,072.44	5,337,196.53	6,352,258.64	6,952,209.85

表6展示了综合考虑各种因素后，优化得到的最大利润，具体种植策略详见附录D。通过对比表4和表6中的年总利润，可以看出，在综合考虑作物之间的可替代性、互补性以及种植成本对销售价格的影响等因素后，综合优化方案在大部分年份中提升了总利润。特别是 2024 年，总利润从 6,130,851.69 元大幅提升至 10,044,860.31 元，增幅显著。这主要归功于引入了作物可替代性和互补性策略，使得不同作物类别的种植决策更具灵活性和经济性。与此同时，蔬菜类和食用菌的互补性以及豆类和谷物类作物的组合种植策略，保证了不同作物之间的协同作用，从而提高了整体种植效益。此外，考虑到销

售价格与销售量之间的关系，以及种植成本的上升对价格的影响，进一步优化了种植结构，使得利润水平在多个年份中维持在较为稳定的范围内。最终，综合优化后的平均利润为 6,952,209.85 元，较基础方案的 6,817,488.49 元有所提升，表明在实际生产中，合理考虑作物之间的替代与互补性可以有效提升农业种植效益。

八、 模型评价

8.1 灵敏度分析

在本节中，我们分析5.3.2节中提出的曼哈顿距离惩罚项对于超参数 λ 的灵敏度。取 $\lambda = \{0.1, 1, 10, 10^3, 10^5, 10^7\}$ 时，在问题一的约束条件下，应用策略 1 的目标函数，独立求解 50 次，并做出 2024~2030 年平均最大利润的箱线图，如图7左所示。可以看出，随着 λ 值的增大，年利润均值先上升后趋于稳定，表明了 λ 较小时模型的敏感性。

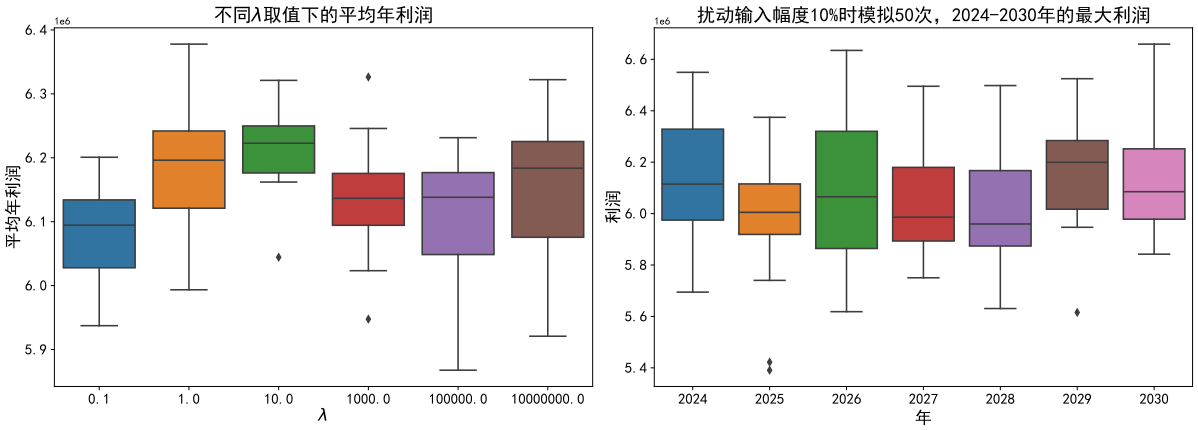

图 7　灵敏度分析与鲁棒性检验结果

8.2 鲁棒性检验

在本节中，我们为决策所依赖的历史数据随机添加 10% 的噪声扰动，然后重复 50 次5.1.2节的优化过程，并做出 2024~2030 年的平均最大利润箱线图，如图7右所示。可以看出，各年最大利润均值均在 10% 左右的范围内波动，属于合理波动。可以认为我们的优化模型具有一定了鲁棒性，对干扰变化不明显。

8.3 模型优缺点

8.3.1 模型优点

1. 利用销售量和销售单价的正态分布和指标浮动的均匀分布，将所有影响因素概率化，更加精准地模拟现实世界中可能存在的众多约束。

2. 曼哈顿距离惩罚项的引入，在最大化利润的同时引入了作物的空间分布密集程度，有利于耕种作业和田间管理。

3. 分层优化模型能够将不同类别的复杂约束逐级求解，避免大量计算的同时能够确保步步深入，获得全局最优解。

4. 种植成本与销售价格的线性模型简单易懂，可直观地反应二者的正相关关系，易于理解和解释。

5. 销售价格和销量之间的对数正态模型符合消费者的直观感受，更能模拟出真实世界中的复杂情况。

8.3.2 模型缺点

1. 惩罚项中超参数 λ 的设定会影响到最优解的状态，若 λ 选择不当，有可能陷入局部最优解。

2. 采用迭代的方法进行预测和最优化，下一年的结果会依赖于上一年的状态，若产生错误预测，则错误会逐级累计，对模型的精度提出了挑战。

参考文献

[1] 李璐, 李宝霖, 李丽红. 模糊曼哈顿距离加权最优粒度选择算法 [J]. 华北理工大学学报（自然科学版）2023,45(04):58-65.

[2] 陈梅, 马学艳, 钱罗雄, 张锦宏, 张弛. 基于多级自表示约束的不完备多视图聚类 [J/OL]. 控制与决策:1-9[2024-08-09 10:08].https://doi.org/10.13195/j.kzyjc.2024.0055..

[3] Fouzia Shile and El Hassan Ben Ahmed and Mohamed Sadik. Modeling groundwater flow with random hydraulic conductivity using radial basis function partition of unity method[J]. Engineering Analysis with Boundary Elements, 2024, 163 : 237-250.

[4] 仲格吉. 空间相关性和变异性对农作物面积空间抽样效率的影响研究 [D]. 中国农业科学院.2019.

[5] 吕志明, 王霖青, 王伟. 一种基于多代理模型的混合整数规划优化方法 [J]. 控制与决策,2019,34(02):362-368.10.13195/j.kzyjc.2017.1099.

附录 A　支撑材料文件列表

表 7　本研究所使用的代码文件结构

文件夹	文件名	描述
附件 3	result1$_1$.$xlsx$	要求提交的问题一结果
	result1$_2$.$xlsx$	要求提交的问题一结果
	result2.xlsx	要求提交的问题二结果
所有文件夹	2023 年的农作物种植情况.csv	格式化后的 2023 年农作物种植数据
	2023 年统计的相关数据.csv	格式化后的 2023 年销售数据统计
	乡村的现有耕地.csv	格式化后的土地编号和面积
	乡村种植的农作物.csv	格式化后的农作物编号和名称
RQ1	prprocessing.ipynb	数据预处理，格式化代码
	year7 规划.py	问题一的模型和求解代码
	目标函数 1_n.csv	在超出销量部分浪费时，从 2023 年起往后第 n 年的种植方式
	目标函数 2_n.csv	在超出销量部分降价一半时，从 2023 年起往后第 n 年的种植方式
RQ2	更新约束的 7 规划.py	问题二的模型和求解代码
	graph.ipynb	绘制指数随机游走折线图和箱型图的代码
	目标函数 2_n.csv	在新的约束条件下，从 2023 年起往后第 n 年的种植方式
RQ3	优化问题 3.py	问题三的模型和求解代码
	目标函数 2_n.csv	在相互作用的约束条件下，从 2023 年起往后第 n 年的种植方式
Sensitive	year7 规划.py	独立重复 50 次实验进行灵敏度和鲁棒性分析的代码
	graph.ipynb	绘制灵敏度和鲁棒性分析结果图的代码

附录 B　问题一的种植策略

表 8　策略 1 下各年份年的具体种植方案

2024																	
地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	63.1	A1	12	16.9	A2	2	55	A3	2	4.8	A3	8	30.2	A4	6	72
A5	6	68	A6	8	55	B1	10	25	B1	9	35	B2	7	46	B3	10	7.5
B3	5	32.5	B4	14	3.5	B4	3	24.5	B5	1	25	B6	9	6.6	B6	7	79.4
B7	11	19.5	B7	1	35.5	B8	4	20.6	B8	9	23.4	B9	13	23.4	B9	4	26.6
B10	3	25	B11	1	26.4	B11	7	33.6	B12	3	28.5	B12	7	16.5	B13	15	14
B13	14	21	B14	4	20	C1	8	15	C2	8	13	C3	8	15	C4	8	18
C5	8	27	C6	1	16	C6	8	4	D1-1	21	12.66	D1-1	19	2.34	D2-1	20	9.61

D2-1	33	0.39	D3-1	16	14	D4-1	24	1.17	D4-1	28	4.83	D5-1	20	8.83	D5-1	27	1.17
D6-1	17	2.66	D6-1	28	9.34	D7-1	16	22	D8-1	16	20	E1-1	22	0.6	E2-1	18	0.6
E3-1	22	0.6	E4-1	29	0.15	E4-1	22	0.45	E5-1	31	0.42	E5-1	22	0.18	E6-1	29	0.42
E7-1	18	0.03	E7-1	25	0.57	E8-1	25	0.6	E9-1	22	0.6	E10-1	22	0.6	E12-1	18	0.21
E13-1	22	0.54	E13-1	21	0.06	E14-1	22	0.6	E15-1	29	0.6	E16-1	22	0.6	F1-1	22	0.6
F2-1	22	0.6	F3-1	32	0.39	F3-1	26	0.21	F4-1	22	0.6	D1-2	36	4.4	D1-2	35	10.6
D2-2	36	10	D4-2	36	6	D5-2	35	6.4	D5-2	37	3.6	D6-2	37	12	E1-2	39	0.6
E2-2	39	0.6	E3-2	40	0.6	E4-2	40	0.54	E4-2	38	0.06	E5-2	41	0.6	E6-2	38	0.6
E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	38	0.6	E11-2	41	0.6	E12-2	39	0.06
E12-2	38	0.54	E13-2	40	0.6	E14-2	40	0.6	E15-2	41	0.06	E15-2	38	0.54	E16-2	41	0.6
F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6						

2025

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	8	80	A2	8	5.2	A2	6	49.8	A3	12	16.9	A3	6	18.1	A4	2	0.3
A4	8	71.7	A5	8	68	A6	2	55	B1	7	60	B2	1	26.5	B2	11	19.5
B3	7	40	B4	9	22.8	B4	4	5.2	B5	4	13	B5	14	12	B6	3	72
B6	15	14	B7	4	49	B7	3	6	B8	13	5.6	B8	1	38.4	B9	1	9.5
B9	7	40.5	B10	5	25	B11	13	17.8	B11	9	42.2	B12	10	32.5	B12	14	12.5
B13	7	35	B14	1	15.5	B14	2	4.5	C1	6	7.5	C1	5	7.5	C2	1	13
C3	6	15	C4	6	18	C5	6	27	C6	6	20	D1-1	16	15	D2-1	25	1.17
D2-1	18	8.83	D3-1	16	14	D4-1	16	6	D5-1	28	9.61	D6-1	16	12	D7-1	28	3.06
D7-1	20	18.94	D8-1	17	11.67	D8-1	18	8.33	E1-1	31	0.6	E2-1	22	0.6	E3-1	29	0.6
E4-1	27	0.57	E5-1	28	0.6	E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.54	E8-1	24	0.06
E9-1	34	0.39	E9-1	28	0.21	E10-1	33	0.39	E11-1	28	0.6	E12-1	22	0.6	E13-1	27	0.6
E14-1	29	0.57	E15-1	22	0.6	E16-1	28	0.09	F1-1	17	0.6	F2-1	32	0.39	F2-1	17	0.21
F3-1	22	0.6	F4-1	17	0.6	D2-2	35	10	D5-2	36	10	D7-2	36	21.4	D7-2	37	0.6
D8-2	35	20	E1-2	41	0.6	E2-2	41	0.6	E3-2	41	0.6	E4-2	41	0.6	E5-2	38	0.6
E6-2	39	0.6	E7-2	38	0.6	E8-2	40	0.6	E9-2	38	0.6	E10-2	41	0.6	E11-2	39	0.6
E12-2	41	0.06	E12-2	40	0.54	E13-2	38	0.54	E13-2	39	0.06	E14-2	41	0.6	E15-2	40	0.6
E16-2	40	0.6	F1-2	17	0.6	F2-2	17	0.6	F3-2	31	0.14	F3-2	17	0.46	F4-2	17	0.6

2026

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	80	A2	2	55	A3	8	35	A4	6	55.1	A4	12	16.9	A5	2	4.8
A5	6	63.2	A6	8	55	B1	14	24.5	B1	3	35.5	B2	7	46	B3	13	23.4
B3	1	16.6	B4	7	28	B5	1	15	B5	7	10	B6	9	65	B6	4	21
B7	10	7.5	B7	7	47.5	B8	7	44	B9	3	42.5	B9	5	7.5	B10	10	25
B11	5	5	B11	1	55	B12	11	19.5	B12	4	25.5	B13	4	21	B13	15	14
B14	5	20	C1	8	15	C2	8	13	C3	8	15	C4	8	1.7	C4	1	16.3
C5	8	27	C6	8	20	D1-1	17	13.83	D1-1	25	1.17	D2-1	16	10	D3-1	18	14
D4-1	28	4.83	D4-1	24	1.17	D5-1	21	7.66	D5-1	19	2.34	D6-1	16	12	D7-1	16	22
D8-1	20	8.95	D8-1	21	11.05	E1-1	26	0.6	E2-1	34	0.39	E3-1	22	0.5	E3-1	17	0.1
E4-1	22	0.6	E5-1	27	0.6	E6-1	29	0.27	E7-1	26	0.57	E7-1	31	0.03	E8-1	17	0.6
E9-1	29	0.3	E9-1	22	0.3	E10-1	29	0.6	E11-1	17	0.6	E12-1	27	0.54	E13-1	31	0.6
E14-1	22	0.6	E15-1	27	0.03	E16-1	18	0.6	F1-1	22	0.6	F2-1	22	0.6	F3-1	17	0.21
F3-1	32	0.39	F4-1	22	0.6	D1-2	36	12.5	D1-2	37	2.5	D3-2	35	11	D3-2	37	3
D4-2	35	6	D5-2	37	10	D8-2	36	20	E1-2	39	0.6	E2-2	39	0.6	E3-2	38	0.6
E4-2	38	0.6	E5-2	41	0.6	E6-2	40	0.6	E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6
E10-2	38	0.6	E11-2	40	0.6	E12-2	38	0.54	E12-2	39	0.06	E13-2	41	0.06	E13-2	40	0.54
E14-2	40	0.6	E15-2	41	0.6	E16-2	41	0.6	F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6
F4-2	22	0.6															

2027

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	8	80	A2	6	28.4	A2	8	26.6	A3	6	35	A4	2	4.8	A4	8	67.2
A5	12	16.9	A5	8	51.1	A6	2	55	B1	7	60	B2	1	27	B2	9	19
B3	14	24.5	B3	7	15.5	B4	1	28	B5	3	25	B6	5	10.9	B6	7	75.1
B7	1	4.6	B7	3	50.4	B8	11	19.5	B8	1	24.5	B9	4	47.2	B9	1	2.8
B10	7	22.4	B10	3	2.6	B11	15	14	B11	9	46	B12	13	23.4	B12	5	21.6
B13	7	2.5	B13	10	32.5	B14	4	20	C1	6	15	C2	6	13	C3	6	15
C4	6	18	C5	6	11	C5	1	16	C6	6	20	D1-1	16	15	D2-1	17	9.61
D2-1	33	0.39	D3-1	16	14	D4-1	18	6	D5-1	28	9.22	D5-1	17	0.78	D6-1	16	12
D7-1	21	4.76	D7-1	20	17.24	D8-1	16	20	E1-1	29	0.6	E2-1	24	0.6	E3-1	26	0.42
E4-1	28	0.6	E5-1	22	0.6	E6-1	27	0.6	E7-1	28	0.6	E8-1	28	0.6	E9-1	19	0.6
E11-1	26	0.21	E12-1	17	0.6	E13-1	27	0.57	E13-1	24	0.03	E14-1	28	0.03	E14-1	29	0.57
E15-1	19	0.6	E16-1	17	0.6	F1-1	17	0.6	F2-1	17	0.6	F3-1	26	0.54	F3-1	22	0.06
F4-1	17	0.6	D2-2	36	10	D4-2	36	6	D5-2	35	10	D7-2	35	19.5	D7-2	36	2.5
E1-2	41	0.6	E2-2	41	0.6	E3-2	40	0.6	E4-2	41	0.6	E5-2	38	0.6	E6-2	38	0.6
E7-2	38	0.6	E8-2	39	0.06	E8-2	38	0.54	E9-2	39	0.6	E10-2	41	0.6	E11-2	41	0.6
E12-2	41	0.06	E12-2	40	0.54	E13-2	39	0.6	E14-2	41	0.6	E15-2	40	0.6	E16-2	40	0.6
F1-2	17	0.6	F2-2	32	0.39	F2-2	17	0.21	F3-2	17	0.6	F4-2	19	0.06	F4-2	17	0.54

2028

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	12	16.9	A1	6	63.1	A2	2	55	A3	2	4.8	A3	8	30.2	A4	6	72
A5	6	68	A6	8	55	B1	9	36.6	B1	13	23.4	B2	14	24.5	B2	7	21.5
B3	4	17.2	B3	5	22.8	B4	7	28	B5	4	25	B6	10	32.5	B6	3	53.5
B7	9	28.4	B7	7	26.6	B8	7	37.3	B8	6	6.7	B9	15	3.5	B9	7	46.5
B10	4	25	B11	1	44.4	B11	7	15.6	B12	11	19.5	B12	1	25.5	B13	15	10.5
B13	3	24.5	B14	1	20	C1	8	15	C2	1	13	C3	8	15	C4	8	18
C5	8	17.3	C5	5	9.7	C6	8	20	D1-1	21	10.7	D1-1	17	4.3	D2-1	16	10
D3-1	21	8.67	D3-1	20	5.33	D4-1	16	6	D5-1	19	2.34	D5-1	18	7.66	D6-1	17	10.83
D6-1	24	1.17	D7-1	16	22	D8-1	28	10.5	D8-1	18	9.5	E1-1	28	0.6	E2-1	33	0.36
E2-1	22	0.24	E3-1	22	0.6	E4-1	22	0.15	E4-1	27	0.45	E5-1	33	0.03	E5-1	29	0.57
E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.6	E9-1	22	0.6	E10-1	29	0.6	E11-1	22	0.6
E12-1	31	0.42	E12-1	22	0.18	E14-1	25	0.21	E15-1	22	0.6	E16-1	27	0.18	E16-1	25	0.42
F1-1	22	0.6	F2-1	22	0.6	F3-1	17	0.21	F3-1	32	0.39	F4-1	22	0.6	D1-2	36	15
D3-2	36	7.5	D3-2	35	6.5	D5-2	36	10	D6-2	35	12	D8-2	35	20	E1-2	39	0.6
E2-2	39	0.6	E3-2	41	0.6	E4-2	38	0.6	E5-2	40	0.6	E6-2	40	0.54	E6-2	41	0.06
E7-2	40	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	40	0.6	E11-2	38	0.6	E12-2	38	0.54
E12-2	39	0.06	E13-2	41	0.6	E14-2	38	0.6	E15-2	41	0.6	E16-2	41	0.6	F1-2	22	0.6
F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6									

2029

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	8.1	A1	8	71.9	A2	12	16.9	A2	7	38.1	A3	6	35	A4	8	72
A5	8	68	A6	2	29.6	A6	6	25.4	B1	7	60	B2	1	45.6	B2	15	0.4
B3	13	10.5	B3	3	29.5	B4	11	19.5	B4	1	8.5	B5	7	25	B6	9	45
B6	7	41	B7	4	19.2	B7	1	35.8	B8	10	31	B8	1	13	B9	10	1.5
B9	3	48.5	B10	7	11.4	B10	15	13.6	B11	4	27.5	B11	5	32.5	B12	14	24.5
B12	4	20.5	B13	2	22.1	B13	13	12.9	B14	9	20	C1	6	15	C2	8	13
C3	6	15	C4	6	18	C5	6	27	C6	6	20	D1-1	16	15	D2-1	20	8.83
D2-1	24	1.17	D3-1	17	13.61	D3-1	34	0.39	D4-1	18	5.61	D5-1	25	1.17	D5-1	21	8.83
D6-1	16	12	D7-1	16	22	D8-1	20	10.24	D8-1	21	9.76	E1-1	29	0.15	E1-1	26	0.45
E3-1	28	0.6	E4-1	29	0.42	E5-1	22	0.6	E6-1	28	0.6	E7-1	28	0.21	E7-1	33	0.39
E8-1	26	0.12	E8-1	21	0.48	E9-1	28	0.6	E10-1	22	0.6	E11-1	21	0.09	E11-1	28	0.51
E12-1	27	0.6	E13-1	22	0.03	E13-1	27	0.57	E14-1	22	0.6	E15-1	28	0.6	E16-1	31	0.6
F1-1	28	0.6	F2-1	17	0.6	F3-1	22	0.6	F4-1	26	0.6	D2-2	35	7.9	D2-2	36	2.1
D3-2	37	14	D4-2	35	6	D5-2	35	10	D8-2	36	18.4	D8-2	37	1.6	E1-2	41	0.6
E2-2	41	0.6	E3-2	38	0.6	E4-2	41	0.06	E4-2	39	0.54	E5-2	41	0.6	E6-2	38	0.6
E7-2	41	0.6	E8-2	38	0.6	E9-2	40	0.6	E10-2	39	0.06	E10-2	38	0.54	E11-2	39	0.06
E11-2	40	0.54	E12-2	41	0.6	E13-2	39	0.6	E14-2	41	0.6	E15-2	40	0.6	E16-2	40	0.6
F1-2	17	0.53	F1-2	20	0.07	F2-2	21	0.21	F2-2	32	0.39	F3-2	29	0.6	F4-2	17	0.6

2030

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	12	4.8	A1	6	75.2	A2	8	55	A3	8	35	A4	2	59.8	A4	6	12.2
A5	6	68	A6	12	12.1	A6	8	42.9	B1	9	51.5	B1	1	8.5	B2	10	32.5
B2	9	13.5	B3	1	26	B3	15	14	B4	7	3.5	B4	14	24.5	B5	1	25
B6	4	67.2	B6	3	18.8	B7	5	4.4	B7	7	50.6	B8	7	41.4	B8	13	2.6
B9	5	28.1	B9	1	21.9	B10	11	19.5	B10	1	5.5	B11	3	39.2	B11	13	20.8
B12	7	45	B13	7	35	B14	3	20	C1	8	15	C2	1	13	C3	8	15
C4	1	3	C4	8	15	C5	8	27	C6	8	20	D1-1	16	15	D2-1	16	10
D3-1	16	14	D4-1	16	6	D5-1	16	10	D6-1	25	1.17	D6-1	21	10.83	D7-1	18	3.46
D7-1	20	18.54	D8-1	17	15.13	D8-1	18	4.87	E1-1	22	0.6	E2-1	29	0.6	E3-1	22	0.03
E3-1	27	0.57	E4-1	22	0.6	E5-1	34	0.39	E6-1	27	0.6	E7-1	22	0.6	E8-1	22	0.6
E9-1	22	0.6	E10-1	28	0.42	E11-1	22	0.6	E12-1	22	0.54	E12-1	28	0.06	E13-1	28	0.6
E15-1	22	0.6	E16-1	21	0.03	E16-1	29	0.57	F1-1	22	0.6	F2-1	22	0.6	F3-1	17	0.21
F3-1	32	0.39	F4-1	22	0.6	D6-2	36	12	D7-2	37	1.5	D7-2	36	20.5	D8-2	35	20
E1-2	39	0.6	E2-2	39	0.6	E3-2	41	0.6	E4-2	38	0.54	E4-2	40	0.06	E5-2	40	0.6
E6-2	41	0.6	E7-2	40	0.6	E8-2	41	0.6	E9-2	41	0.54	E9-2	39	0.06	E10-2	40	0.48
E10-2	41	0.12	E11-2	41	0.6	E12-2	40	0.6	E13-2	41	0.6	E14-2	38	0.6	E15-2	38	0.6
E16-2	38	0.6	F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6			

表 9　策略 2 下各年份的具体种植方案

2024

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	4.8	A1	6	75.2	A2	2	55	A3	8	26.8	A3	6	8.2	A4	6	72
A5	8	68	A6	8	55	B1	10	32.5	B1	8	27.5	B2	7	46	B3	15	2

B3	3	38	B4	3	28	B5	12	16.9	B5	3	8.1	B6	7	38.4	B6	8	47.6
B7	15	1.4	B7	7	53.6	B8	4	44	B9	1	46.8	B9	4	3.2	B10	5	25
B11	1	56.1	B11	3	3.9	B12	5	7.5	B12	7	37.5	B13	14	24.4	B13	15	10.6
B14	4	20	C1	9	15	C2	9	13	C3	13	14.9	C3	14	0.1	C4	9	18
C5	11	19.5	C5	13	7.5	C6	9	19	C6	13	1	D1-1	17	14.47	D1-1	31	0.53
D2-1	24	0.63	D2-1	21	9.37	D3-1	28	13.22	D4-1	25	1.17	D4-1	18	4.83	D5-1	16	10
D6-1	18	9.66	D6-1	19	2.34	D7-1	20	19.33	D7-1	18	2.67	D8-1	16	20	E1-1	22	0.6
E2-1	34	0.39	E2-1	27	0.21	E3-1	22	0.6	E4-1	22	0.6	E5-1	22	0.6	E6-1	17	0.27
E6-1	27	0.33	E7-1	26	0.6	E8-1	27	0.03	E8-1	26	0.57	E9-1	22	0.6	E10-1	22	0.6
E11-1	27	0.6	E12-1	17	0.6	E13-1	22	0.6	E14-1	22	0.33	E15-1	22	0.6	E16-1	22	0.6
F1-1	22	0.6	F2-1	29	0.6	F3-1	22	0.21	F3-1	32	0.39	F4-1	22	0.03	F4-1	29	0.57
D1-2	36	15	D2-2	36	10	D3-2	36	7.5	D3-2	35	6.5	D4-2	35	6	D6-2	37	12
D7-2	35	22	E1-2	39	0.6	E2-2	39	0.6	E3-2	38	0.6	E4-2	40	0.6	E5-2	41	0.6
E6-2	38	0.54	E6-2	41	0.06	E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	40	0.54
E10-2	41	0.06	E11-2	38	0.6	E12-2	38	0.6	E13-2	40	0.6	E14-2	40	0.6	E15-2	41	0.54
E15-2	39	0.06	E16-2	41	0.6	F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6

2025

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	8	80	A2	6	55	A3	2	35	A4	8	69.8	A4	2	2.2	A5	6	45.4
A5	2	22.6	A6	6	55	B1	7	45.1	B1	12	14.9	B2	1	13.5	B2	10	32.5
B3	4	40	B4	7	28	B5	15	5.4	B5	4	19.6	B6	3	78	B6	1	8
B7	5	32.5	B7	1	22.5	B8	1	38.9	B8	14	5.1	B9	7	50	B10	9	23
B10	12	2	B11	7	52.4	B11	4	7.6	B12	8	36.4	B12	15	8.6	B13	8	35
B14	1	20	C1	8	3.1	C1	11	11.9	C2	13	5.4	C2	11	7.6	C3	9	15
C4	13	18	C5	9	27	C6	14	20	D1-1	16	15	D2-1	28	9.22	D3-1	16	14
D4-1	28	3.66	D4-1	19	2.34	D5-1	16	10	D6-1	20	5.79	D6-1	21	6.21	D7-1	16	22
D8-1	22	5.94	D8-1	18	14.06	E1-1	27	0.24	E2-1	22	0.6	E3-1	17	0.6	E4-1	26	0.6
E5-1	26	0.57	E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.6	E9-1	27	0.6	E10-1	17	0.6
E11-1	31	0.6	E12-1	22	0.6	E13-1	17	0.03	E13-1	24	0.57	E14-1	25	0.6	E15-1	24	0.6
E16-1	17	0.21	E16-1	34	0.39	F1-1	17	0.27	F1-1	27	0.33	F2-1	22	0.03	F2-1	25	0.57
F3-1	29	0.42	F3-1	31	0.18	F4-1	17	0.6	D2-2	37	10	D4-2	36	6	D6-2	36	4.5
D6-2	35	7.5	D8-2	35	20	E1-2	41	0.6	E2-2	41	0.6	E3-2	40	0.6	E4-2	41	0.6
E5-2	38	0.54	E5-2	39	0.06	E6-2	40	0.6	E7-2	39	0.6	E8-2	38	0.6	E9-2	40	0.6
E10-2	39	0.6	E11-2	41	0.6	E12-2	41	0.06	E12-2	40	0.54	E13-2	41	0.6	E14-2	41	0.6
E15-2	38	0.6	E16-2	38	0.6	F1-2	32	0.39	F1-2	17	0.21	F2-2	17	0.6	F3-2	33	0.39
F3-2	29	0.21	F4-2	29	0.54	F4-2	17	0.06									

2026

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	4.8	A1	6	75.2	A2	2	55	A3	8	31.4	A3	6	3.6	A4	6	72
A5	8	55.1	A5	9	12.9	A6	8	55	B1	3	18	B1	8	42	B2	8	46
B3	7	40	B4	5	28	B5	3	25	B6	14	11.5	B6	7	74.5	B7	7	35.9
B7	4	19.1	B8	15	14	B8	10	30	B9	1	45.4	B9	6	4.6	B10	7	25
B11	1	57.5	B11	10	2.5	B12	4	28.1	B12	12	16.9	B13	3	35	B14	4	20
C1	9	15	C2	14	13	C3	11	15	C4	9	18	C5	13	23.4	C5	11	3.6
C6	9	19.1	C6	11	0.9	D1-1	21	1.71	D1-1	20	13.29	D2-1	16	10	D3-1	31	0.36
D3-1	28	13.64	D4-1	16	6	D5-1	33	0.39	D5-1	18	9.61	D6-1	16	12	D7-1	18	7.5
D7-1	17	14.5	D8-1	16	20	E1-1	22	0.6	E2-1	34	0.39	E2-1	27	0.21	E3-1	22	0.6
E4-1	22	0.6	E5-1	22	0.6	E6-1	26	0.39	E7-1	27	0.6	E8-1	17	0.24	E8-1	27	0.36
E9-1	22	0.6	E10-1	22	0.6	E11-1	22	0.6	E12-1	26	0.6	E13-1	22	0.54	E13-1	31	0.06
E14-1	17	0.6	E15-1	22	0.6	E16-1	22	0.6	F1-1	22	0.03	F1-1	29	0.57	F2-1	29	0.6
F3-1	22	0.6	F4-1	22	0.6	D1-2	35	15	D3-2	35	14	D5-2	37	5.5	D5-2	36	4.5
D7-2	35	4	D7-2	36	18	E1-2	39	0.6	E2-2	39	0.6	E3-2	41	0.6	E4-2	40	0.6
E5-2	40	0.6	E6-2	41	0.6	E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	40	0.54
E10-2	41	0.06	E11-2	38	0.6	E12-2	38	0.54	E12-2	39	0.06	E13-2	38	0.6	E14-2	38	0.6
E15-2	40	0.6	E16-2	41	0.6	F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	32	0.39
F3-2	29	0.21	F4-2	29	0.54	F4-2	17	0.06									

2027

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	63.1	A1	12	16.9	A2	2	55	A3	2	4.8	A3	8	30.2	A4	6	72
A5	6	68	A6	8	55	B1	10	25	B1	9	35	B2	7	46	B3	10	7.5
B3	5	32.5	B4	14	3.5	B4	3	24.5	B5	1	25	B6	9	6.6	B6	7	79.4
B7	11	19.5	B7	1	35.5	B8	4	20.6	B8	9	23.4	B9	13	23.4	B9	4	26.6
B10	3	25	B11	1	26.4	B11	7	33.6	B12	3	28.5	B12	7	16.5	B13	15	14
B13	14	21	B14	4	20	C1	8	15	C2	8	13	C3	8	15	C4	8	18
C5	8	27	C6	1	16	C6	8	4	D1-1	21	12.66	D1-1	19	2.34	D2-1	20	9.61
D2-1	33	0.39	D3-1	16	14	D4-1	24	1.17	D4-1	28	4.83	D5-1	20	8.83	D5-1	27	1.17
D6-1	17	2.66	D6-1	28	9.34	D7-1	16	22	D8-1	16	20	E1-1	22	0.6	E2-1	18	0.6
E3-1	22	0.6	E4-1	29	0.15	E4-1	22	0.45	E5-1	31	0.42	E5-1	22	0.18	E6-1	29	0.42
E7-1	18	0.03	E7-1	25	0.57	E8-1	25	0.6	E9-1	22	0.6	E10-1	22	0.6	E12-1	18	0.21
E13-1	22	0.54	E13-1	21	0.06	E14-1	22	0.6	E15-1	29	0.6	E16-1	22	0.6	F1-1	22	0.6
F2-1	22	0.6	F3-1	32	0.39	F3-1	26	0.21	F4-1	22	0.6	D1-2	36	4.4	D1-2	35	10.6

D2-2	36	10	D4-2	36	6	D5-2	35	6.4	D5-2	37	3.6	D6-2	37	12	E1-2	39	0.6
E2-2	39	0.6	E3-2	40	0.6	E4-2	40	0.54	E4-2	38	0.06	E5-2	41	0.6	E6-2	38	0.6
E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	38	0.6	E11-2	41	0.6	E12-2	39	0.06
E12-2	38	0.54	E13-2	40	0.6	E14-2	40	0.6	E15-2	41	0.06	E15-2	38	0.54	E16-2	41	0.6
F1-2	22	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6						

2028

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	80	A2	2	55	A3	8	31.6	A3	6	3.4	A4	6	72	A5	8	68
A6	8	50.2	A6	2	4.8	B1	8	48	B1	9	12	B2	1	46	B3	7	14.2
B3	3	25.8	B4	7	28	B5	3	25	B6	1	10.9	B6	7	75.1	B7	7	55
B8	5	32.5	B8	3	11.5	B9	4	46.8	B9	7	3.2	B10	14	9.3	B10	3	15.7
B11	1	46	B11	15	14	B12	8	27.1	B12	4	17.9	B13	10	32.5	B13	4	2.5
B14	12	16.9	B14	14	3.1	C1	9	15	C2	11	4.6	C2	13	8.4	C3	13	15
C4	9	18	C5	11	14.9	C5	14	12.1	C6	9	20	D1-1	28	12.66	D1-1	19	2.34
D2-1	17	9.22	D3-1	16	14	D4-1	20	0.09	D4-1	17	5.91	D5-1	16	10	D6-1	16	12
D7-1	16	22	D8-1	18	8.33	D8-1	21	11.67	E1-1	28	0.31	E1-1	24	0.29	E2-1	27	0.6
E3-1	31	0.6	E4-1	22	0.6	E5-1	22	0.6	E6-1	26	0.03	E6-1	27	0.57	E7-1	28	0.6
E8-1	26	0.6	E9-1	31	0.12	E9-1	22	0.48	E10-1	22	0.6	E11-1	24	0.21	E12-1	17	0.21
E12-1	34	0.39	E13-1	22	0.6	E14-1	26	0.54	E14-1	31	0.06	E15-1	24	0.6	E16-1	22	0.6
F1-1	32	0.39	F1-1	22	0.21	F2-1	28	0.6	F3-1	22	0.6	F4-1	22	0.21	F4-1	33	0.39
D1-2	36	10.4	D1-2	35	4.6	D2-2	37	9.6	D2-2	35	0.4	D4-2	37	6	D8-2	35	20
E1-2	39	0.6	E2-2	39	0.6	E3-2	41	0.06	E3-2	40	0.54	E4-2	38	0.54	E4-2	39	0.06
E5-2	40	0.6	E6-2	41	0.6	E7-2	41	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	40	0.6
E11-2	38	0.6	E12-2	38	0.6	E13-2	38	0.6	E14-2	40	0.6	E15-2	41	0.6	E16-2	41	0.6
F1-2	29	0.57	F1-2	22	0.03	F2-2	22	0.6	F3-2	22	0.6	F4-2	29	0.6			

2029

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	8	80	A2	6	55	A3	2	35	A4	8	72	A5	6	43.2	A5	2	24.8
A6	6	55	B1	10	32.5	B1	7	27.5	B2	6	10	B2	7	36	B3	5	32.5
B3	14	7.5	B4	8	28	B5	4	25	B6	3	78	B6	4	8	B7	8	2.1
B7	1	52.9	B8	12	16.9	B8	4	27.1	B9	1	50	B10	7	25	B11	7	36
B11	9	24	B12	15	14	B12	7	31	B13	8	35	B14	7	20	C1	11	15
C2	14	1.5	C2	9	11.5	C3	9	15	C4	4	7.1	C4	13	10.9	C5	13	12.5
C5	9	14.5	C6	11	4.5	C6	14	15.5	D1-1	16	15	D2-1	16	10	D3-1	19	2.34
D3-1	17	11.66	D4-1	16	6	D5-1	21	9.22	D6-1	21	10.02	D6-1	22	1.98	D7-1	28	14.17
D7-1	18	7.83	D8-1	16	20	E1-1	22	0.6	E2-1	22	0.6	E3-1	22	0.39	E4-1	26	0.57
E4-1	21	0.03	E5-1	17	0.6	E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.6	E9-1	27	0.6
E10-1	26	0.6	E11-1	22	0.6	E12-1	22	0.6	E13-1	27	0.57	E13-1	33	0.03	E14-1	22	0.6
E15-1	22	0.6	E16-1	17	0.21	E16-1	34	0.39	F1-1	17	0.6	F2-1	22	0.6	F3-1	33	0.15
F3-1	29	0.45	F4-1	29	0.6	D3-2	37	3	D3-2	35	11	D5-2	35	10	D6-2	35	12
D7-2	36	22	E1-2	41	0.6	E2-2	41	0.6	E3-2	38	0.6	E4-2	40	0.6	E5-2	41	0.6
E6-2	38	0.54	E6-2	39	0.06	E7-2	38	0.6	E8-2	39	0.6	E9-2	40	0.6	E10-2	41	0.6
E11-2	40	0.54	E11-2	41	0.06	E12-2	40	0.6	E13-2	41	0.6	E14-2	41	0.6	E15-2	38	0.6
E16-2	39	0.6	F1-2	33	0.21	F1-2	32	0.39	F2-2	17	0.6	F3-2	17	0.48	F3-2	29	0.12
F4-2	22	0.6															

2030

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	31.6	A1	6	48.4	A2	8	26.8	A2	2	28.2	A3	6	35	A4	6	72
A5	8	68	A6	8	55	B1	12	16.9	B1	8	43.1	B2	4	13.5	B2	5	32.5
B3	9	12	B3	1	28	B4	3	28	B5	14	12.4	B5	15	12.6	B6	1	0.5
B6	7	85.5	B7	7	55	B8	10	12.5	B8	1	31.5	B9	4	48.6	B9	15	1.4
B10	4	5.1	B10	3	19.9	B11	3	28	B11	8	32	B12	1	42.9	B12	3	2.1
B13	7	35	B14	10	20	C1	9	15	C2	13	13	C3	13	10.4	C3	14	4.6
C4	9	18	C5	14	7.5	C5	11	19.5	C6	9	20	D1-1	18	9.5	D1-1	20	5.5
D2-1	20	8.83	D2-1	25	1.17	D3-1	16	14	D4-1	17	1.81	D4-1	20	4.19	D5-1	16	10
D6-1	17	11.82	D7-1	16	22	D8-1	28	14.17	D8-1	21	5.83	E1-1	27	0.21	E1-1	34	0.39
E2-1	29	0.6	E3-1	26	0.6	E4-1	22	0.6	E5-1	22	0.6	E6-1	17	0.24	E6-1	27	0.36
E7-1	17	0.6	E8-1	17	0.03	E8-1	24	0.57	E9-1	22	0.6	E10-1	22	0.6	E11-1	17	0.21
E12-1	27	0.6	E14-1	26	0.57	E14-1	17	0.03	E15-1	17	0.6	E16-1	24	0.6	F1-1	22	0.18
F1-1	31	0.42	F2-1	29	0.21	F2-1	32	0.39	F3-1	22	0.6	F4-1	22	0.6	D1-2	36	15
D2-2	35	10	D4-2	37	6	D6-2	36	12	D8-2	35	20	E1-2	39	0.6	E2-2	39	0.6
E3-2	41	0.6	E4-2	41	0.6	E5-2	40	0.6	E6-2	40	0.6	E7-2	41	0.6	E8-2	41	0.06
E8-2	40	0.54	E9-2	38	0.6	E10-2	40	0.6	E11-2	38	0.54	E11-2	39	0.06	E12-2	41	0.6
E13-2	38	0.6	E14-2	38	0.6	E15-2	41	0.6	E16-2	41	0.6	F1-2	22	0.6	F2-2	22	0.6
F3-2	22	0.6	F4-2	33	0.24	F4-2	29	0.36									

附录 C　问题二的种植策略

表 10　问题 2 下各年份年的具体种植方案

2024

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	1	29.3	A1	2	50.7	A2	8	55	A3	10	32.5	A3	8	2.5	A4	1	27.6
A4	3	44.4	A5	8	58.9	A5	2	9.1	A6	3	33.6	A6	4	21.4	B1	7	60
B2	4	29.1	B2	12	16.9	B3	5	7.5	B3	6	32.5	B4	7	28	B5	6	25
B6	7	86	B7	8	55	B8	6	27.3	B8	4	16.7	B9	9	50	B10	5	25
B11	1	46	B11	15	14	B12	6	30	B12	9	15	B13	8	35	B14	8	20
C1	6	15	C2	6	6.5	C2	14	6.5	C3	11	15	C4	14	18	C5	11	3.6
C5	13	23.4	C6	6	19.1	C6	11	0.9	D1-1	17	13.83	D1-1	27	1.17	D2-1	18	8.64
D2-1	19	1.36	D3-1	16	14	D4-1	21	6	D5-1	17	1.51	D5-1	18	8.49	D6-1	21	11.22
D6-1	31	0.78	D7-1	16	22	D8-1	16	20	E1-1	22	0.6	E2-1	26	0.25	E2-1	21	0.35
E3-1	21	0.6	E4-1	22	0.6	E5-1	19	0.6	E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.6
E9-1	22	0.6	E10-1	21	0.6	E11-1	34	0.39	E11-1	24	0.21	E12-1	24	0.6	E13-1	26	0.22
E13-1	19	0.38	E14-1	22	0.6	E15-1	21	0.6	E16-1	28	0.17	E16-1	20	0.43	F1-1	33	0.39
F2-1	18	0.03	F3-1	22	0.21	F4-1	32	0.39	F4-1	22	0.21	D1-2	37	6.6	D1-2	36	8.4
D2-2	35	1	D2-2	37	9	D4-2	35	6	D5-2	35	10	D6-2	36	12	E1-2	39	0.6
E2-2	39	0.6	E3-2	41	0.6	E4-2	41	0.6	E5-2	41	0.6	E6-2	41	0.6	E7-2	40	0.6
E8-2	40	0.6	E9-2	38	0.6	E10-2	40	0.6	E11-2	38	0.54	E11-2	39	0.06	E12-2	41	0.6
E13-2	40	0.54	E13-2	41	0.06	E14-2	38	0.6	E15-2	41	0.6	E16-2	38	0.6	F1-2	22	0.5
F1-2	26	0.1	F2-2	24	0.03	F2-2	29	0.57	F3-2	26	0.6	F4-2	29	0.6			

2025

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	56.6	A1	13	23.4	A2	6	38.1	A2	12	16.9	A3	11	6.5	A3	6	28.5
A4	15	14	A4	8	58	A5	10	32.5	A5	6	35.5	A6	6	55	B1	6	39.9
B1	3	20.1	B2	3	7.3	B2	2	38.7	B3	9	37	B3	3	3	B4	9	28
B5	2	21.1	B5	4	3.9	B6	1	86	B7	5	32.5	B7	4	22.5	B8	1	16.9
B8	3	27.1	B9	4	50	B10	7	25	B11	7	60	B12	14	24.5	B12	3	20.5
B13	6	35	B14	7	20	C1	4	15	C2	11	13	C3	8	15	C4	8	9.5
C4	4	8.5	C5	8	27	C6	8	20	D1-1	16	15	D2-1	16	10	D3-1	27	1.17
D3-1	20	12.83	D4-1	24	0.84	D4-1	18	5.16	D5-1	16	10	D6-1	18	12	D7-1	21	18.74
D7-1	20	3.26	D8-1	22	7.15	D8-1	17	12.85	E1-1	19	0.6	E2-1	17	0.6	E3-1	34	0.39
E3-1	28	0.21	E4-1	17	0.6	E5-1	21	0.03	E5-1	25	0.57	E6-1	26	0.6	E7-1	19	0.51
E7-1	17	0.09	E8-1	25	0.6	E9-1	19	0.6	E10-1	33	0.39	E11-1	21	0.6	E12-1	17	0.6
E13-1	17	0.6	E14-1	26	0.57	E14-1	19	0.03	E15-1	28	0.6	E16-1	19	0.6	F1-1	22	0.03
F1-1	29	0.57	F2-1	22	0.21	F2-1	32	0.39	F3-1	28	0.6	F4-1	28	0.6	D3-2	35	14
D4-2	37	6	D6-2	37	7	D6-2	35	5	D7-2	36	19.4	D7-2	37	2.6	D8-2	35	20
E1-2	41	0.6	E2-2	41	0.6	E3-2	39	0.6	E4-2	38	0.6	E5-2	40	0.6	E6-2	38	0.54
E6-2	40	0.06	E7-2	38	0.6	E8-2	41	0.6	E9-2	40	0.6	E10-2	39	0.6	E11-2	41	0.12
E11-2	40	0.48	E12-2	40	0.6	E13-2	39	0.6	E14-2	39	0.18	E14-2	41	0.42	E15-2	38	0.6
E16-2	41	0.6	F1-2	29	0.6	F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.38			

2026

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	1	22.9	A1	4	57.1	A2	7	55	A3	15	14	A3	8	21	A4	14	24.5
A4	2	47.5	A5	12	16.9	A5	8	51.1	A6	7	55	B1	11	19.5	B1	8	40.5
B2	8	16.9	B2	7	29.1	B3	6	40	B4	3	28	B5	6	25	B6	5	2.5
B6	6	83.5	B7	9	18.6	B7	7	36.4	B8	2	12.3	B8	4	31.7	B9	13	23.4
B9	9	26.6	B10	6	25	B11	3	30	B11	5	30	B12	9	2.3	B12	6	42.7
B13	9	2.5	B13	10	32.5	B14	3	20	C1	1	15	C2	4	13	C3	9	15
C4	1	18	C5	1	27	C6	1	20	D1-1	18	10.84	D1-1	20	4.16	D2-1	28	2.52
D2-1	20	7.48	D3-1	17	11.66	D3-1	19	2.34	D4-1	26	1.17	D4-1	20	4.83	D5-1	17	3.68
D5-1	18	6.32	D6-1	20	3.03	D6-1	22	8.97	D7-1	16	22	D8-1	16	20	E1-1	34	0.39
E1-1	28	0.21	E2-1	28	0.6	E3-1	29	0.42	E3-1	32	0.18	E4-1	28	0.6	E5-1	28	0.6
E6-1	24	0.6	E7-1	28	0.6	E8-1	28	0.6	E9-1	28	0.6	E10-1	28	0.6	E11-1	29	0.15
E11-1	31	0.45	E12-1	21	0.03	E12-1	24	0.57	E13-1	32	0.21	E13-1	28	0.39	E14-1	28	0.6
E15-1	29	0.6	E16-1	28	0.6	F1-1	28	0.6	F2-1	25	0.21	F3-1	25	0.03	F4-1	27	0.6
D1-2	35	15	D2-2	37	1.6	D2-2	36	8.4	D3-2	37	14	D4-2	35	3.9	D4-2	36	2.1
D5-2	36	10	D6-2	36	12	E1-2	39	0.6	E2-2	39	0.6	E3-2	40	0.6	E4-2	40	0.6
E5-2	39	0.6	E6-2	41	0.6	E7-2	41	0.6	E8-2	40	0.6	E9-2	38	0.6	E10-2	41	0.6
E11-2	38	0.6	E12-2	38	0.6	E13-2	41	0.54	E13-2	40	0.06	E14-2	40	0.48	E14-2	38	0.12
E15-2	41	0.6	E16-2	38	0.42	E16-2	39	0.18	F1-2	28	0.6	F2-2	33	0.39	F3-2	28	0.03
F3-2	27	0.57	F4-2	21	0.27	F4-2	31	0.33									

2027

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	80	A2	10	31.9	A2	4	23.1	A3	7	35	A4	7	72	A5	7	3.5
A5	4	64.5	A6	10	0.6	A6	6	54.4	B1	9	27.5	B1	5	32.5	B2	1	46
B3	9	37.5	B3	1	2.5	B4	1	28	B5	8	25	B6	8	86	B7	3	53

B7	14	2	B8	8	44	B9	2	43.6	B9	1	6.4	B10	3	25	B11	8	43.8
B11	2	16.2	B12	13	23.4	B12	15	21.6	B13	14	22.5	B13	6	12.5	B14	1	20
C1	12	15	C2	11	13	C3	11	6.5	C3	6	8.5	C4	7	18	C5	7	27
C6	7	20	D1-1	16	15	D2-1	16	10	D3-1	28	13.79	D4-1	16	6	D5-1	16	10
D6-1	29	1.17	D6-1	18	10.83	D7-1	18	3.62	D7-1	21	18.38	D8-1	16	20	E1-1	22	0.6
E2-1	22	0.6	E3-1	22	0.6	E4-1	22	0.6	E5-1	22	0.6	E6-1	17	0.42	E7-1	22	0.6
E8-1	17	0.6	E9-1	22	0.6	E10-1	33	0.39	E10-1	24	0.21	E11-1	22	0.6	E12-1	22	0.6
E13-1	22	0.6	E14-1	22	0.57	E14-1	25	0.03	E15-1	22	0.6	F1-1	31	0.42	F1-1	21	0.18
F2-1	22	0.6	F3-1	32	0.39	F3-1	21	0.21	F4-1	22	0.6	D3-2	36	7	D3-2	35	7
D6-2	35	12	D7-2	36	22	E1-2	41	0.6	E2-2	41	0.6	E3-2	41	0.6	E4-2	39	0.6
E5-2	40	0.6	E6-2	38	0.6	E7-2	40	0.6	E8-2	41	0.6	E9-2	39	0.6	E10-2	38	0.6
E11-2	40	0.6	E12-2	40	0.48	E12-2	39	0.12	E13-2	38	0.6	E14-2	39	0.6	E15-2	38	0.54
E15-2	39	0.06	E16-2	40	0.06	E16-2	41	0.54	F1-2	21	0.6	F2-2	25	0.6	F3-2	22	0.6
F4-2	17	0.18	F4-2	24	0.42												

2028

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	12	16.9	A1	6	63.1	A2	2	55	A3	2	4.8	A3	8	30.2	A4	6	72
A5	6	68	A6	8	55	B1	9	36.6	B1	13	23.4	B2	14	24.5	B2	7	21.5
B3	4	17.2	B3	5	22.8	B4	7	28	B5	4	25	B6	10	32.5	B6	3	53.5
B7	9	28.4	B7	7	26.6	B8	7	37.3	B8	6	6.7	B9	15	3.5	B9	7	46.5
B10	4	25	B11	1	44.4	B11	7	15.6	B12	11	19.5	B12	1	25.5	B13	15	10.5
B13	3	24.5	B14	1	20	C1	8	15	C2	1	13	C3	8	15	C4	8	18
C5	8	17.3	C5	5	9.7	C6	8	20	D1-1	21	10.7	D1-1	17	4.3	D2-1	16	10
D3-1	21	8.67	D3-1	20	5.33	D4-1	16	6	D5-1	19	2.34	D5-1	18	7.66	D6-1	17	10.83
D6-1	24	1.17	D7-1	16	22	D8-1	28	10.5	D8-1	18	9.5	E1-1	28	0.6	E2-1	33	0.36
E2-1	22	0.24	E3-1	22	0.6	E4-1	22	0.15	E4-1	27	0.45	E5-1	33	0.03	E5-1	29	0.57
E6-1	22	0.6	E7-1	22	0.6	E8-1	22	0.6	E9-1	22	0.6	E10-1	29	0.6	E11-1	22	0.6
E12-1	31	0.42	E12-1	22	0.18	E14-1	25	0.21	E15-1	22	0.6	E16-1	27	0.18	E16-1	25	0.42
F1-1	22	0.6	F2-1	22	0.6	F3-1	17	0.21	F3-1	32	0.39	F4-1	22	0.6	D1-2	36	15
D3-2	36	7.5	D3-2	35	6.5	D5-2	36	10	D6-2	35	12	D8-2	35	20	E1-2	39	0.6
E2-2	39	0.6	E3-2	41	0.6	E4-2	38	0.6	E5-2	40	0.6	E6-2	40	0.54	E6-2	41	0.06
E7-2	40	0.6	E8-2	41	0.6	E9-2	41	0.6	E10-2	40	0.6	E11-2	38	0.6	E12-2	38	0.54
E12-2	39	0.06	E13-2	41	0.6	E14-2	38	0.6	E15-2	41	0.6	E16-2	41	0.6	F1-2	22	0.6
F2-2	22	0.6	F3-2	22	0.6	F4-2	22	0.6									

2029

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	8.1	A1	8	71.9	A2	12	16.9	A2	7	38.1	A3	6	35	A4	8	72
A5	8	68	A6	2	29.6	A6	6	25.4	B1	7	60	B2	1	45.6	B2	15	0.4
B3	13	10.5	B3	3	29.5	B4	11	19.5	B4	1	8.5	B5	7	25	B6	9	45
B6	7	41	B7	4	19.2	B7	1	35.8	B8	10	31	B8	1	13	B9	10	1.5
B9	3	48.5	B10	7	11.4	B10	15	13.6	B11	4	27.5	B11	5	32.5	B12	14	24.5
B12	4	20.5	B13	2	22.1	B13	13	12.9	B14	9	20	C1	6	15	C2	8	13
C3	6	15	C4	6	18	C5	6	27	C6	6	20	D1-1	16	15	D2-1	20	8.83
D2-1	24	1.17	D3-1	17	13.61	D3-1	34	0.39	D4-1	18	5.61	D5-1	25	1.17	D5-1	21	8.83
D6-1	16	12	D7-1	16	22	D8-1	20	10.24	D8-1	21	9.76	E1-1	29	0.15	E1-1	26	0.45
E3-1	28	0.6	E4-1	29	0.42	E5-1	22	0.6	E6-1	28	0.6	E7-1	28	0.21	E7-1	33	0.39
E8-1	26	0.12	E8-1	21	0.48	E9-1	28	0.6	E10-1	22	0.6	E11-1	21	0.09	E11-1	28	0.51
E12-1	27	0.6	E13-1	22	0.03	E13-1	27	0.57	E14-1	22	0.6	E15-1	28	0.6	E16-1	31	0.6
F1-1	28	0.6	F2-1	17	0.6	F3-1	22	0.6	F4-1	26	0.6	D2-2	35	7.9	D2-2	36	2.1
D3-2	37	14	D4-2	35	6	D5-2	35	10	D8-2	36	18.4	D8-2	37	1.6	E1-2	41	0.6
E2-2	41	0.6	E3-2	38	0.6	E4-2	41	0.06	E4-2	39	0.54	E5-2	41	0.6	E6-2	38	0.6
E7-2	41	0.6	E8-2	38	0.6	E9-2	40	0.6	E10-2	39	0.06	E10-2	38	0.54	E11-2	39	0.06
E11-2	40	0.54	E12-2	41	0.6	E13-2	39	0.6	E14-2	41	0.6	E15-2	40	0.6	E16-2	40	0.6
F1-2	17	0.53	F1-2	20	0.07	F2-2	21	0.21	F2-2	32	0.39	F3-2	29	0.6	F4-2	17	0.6

2030

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	6	80	A2	2	55	A3	9	25.5	A3	14	9.5	A4	9	39.5	A4	10	32.5
A5	7	35.5	A5	5	32.5	A6	1	35.5	A6	11	19.5	B1	8	60	B2	6	46
B3	7	40	B4	6	28	B5	4	25	B6	1	67.4	B6	8	18.6	B7	4	55
B8	6	18.1	B8	8	25.9	B9	15	9	B9	3	41	B10	8	25	B11	6	60
B12	7	45	B13	7	35	B14	6	20	C1	3	15	C2	12	13	C3	13	15
C4	12	3.9	C4	3	14.1	C5	6	18.6	C5	13	8.4	C6	15	5	C6	14	15
D1-1	18	5.91	D1-1	28	9.09	D2-1	18	5.22	D2-1	21	4.78	D3-1	16	14	D4-1	25	1.17
D4-1	18	4.83	D5-1	24	1.03	D5-1	22	8.97	D6-1	16	12	D7-1	16	22	D8-1	28	3.07
D8-1	20	16.93	E2-1	17	0.5	E2-1	26	0.1	E3-1	21	0.6	E4-1	17	0.6	E5-1	29	0.6
E6-1	18	0.6	E7-1	21	0.6	E8-1	18	0.6	E9-1	19	0.6	E10-1	21	0.03	E10-1	29	0.57
E11-1	21	0.6	E12-1	26	0.6	E13-1	32	0.39	E13-1	17	0.21	E14-1	17	0.6	E15-1	26	0.47
E16-1	17	0.21	F1-1	21	0.6	F2-1	28	0.6	F3-1	27	0.6	F4-1	21	0.55	D1-2	35	3.5
D1-2	37	11.5	D2-2	36	4.5	D2-2	35	5.5	D4-2	36	6	D5-2	35	10	D8-2	35	20
E1-2	39	0.6	E2-2	39	0.6	E3-2	39	0.06	E3-2	40	0.54	E4-2	41	0.6	E5-2	41	0.6
E6-2	38	0.6	E7-2	40	0.6	E8-2	38	0.6	E9-2	38	0.6	E10-2	41	0.6	E11-2	40	0.6

E12-2	41	0.6	E13-2	41	0.6	E14-2	41	0.06	E14-2	38	0.54	E15-2	40	0.6	E16-2	39	0.6
F1-2	33	0.39	F1-2	28	0.21	F2-2	28	0.6	F3-2	27	0.57	F3-2	24	0.03	F4-2	28	0.6

附录 D　问题三的种植策略

表 11　问题 3 下各年份年的具体种植方案

2024

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	11	80	A2	11	55	A3	11	35	A4	13	42.9	A4	10	29.1	A5	2	68
A6	11	55	B1	11	60	B2	2	46	B3	2	40	B4	11	28	B5	11	25
B6	11	63.2	B6	3	22.8	B7	11	55	B8	11	44	B9	11	50	B10	2	25
B11	2	60	B12	11	45	B13	11	35	B14	11	20	C1	11	15	C2	11	13
C3	11	15	C4	11	18	C5	10	27	C6	11	20	D1-1	16	15	D2-1	16	10
D3-1	32	14	D4-1	16	6	D5-1	32	10	D6-1	16	12	D7-1	29	22	D8-1	16	20
E1-1	29	0.6	E2-1	29	0.6	E3-1	29	0.6	E4-1	29	0.6	E5-1	29	0.6	E6-1	29	0.6
E7-1	29	0.6	E8-1	29	0.6	E9-1	29	0.6	E10-1	29	0.6	E11-1	29	0.6	E12-1	29	0.6
E13-1	29	0.6	E14-1	29	0.6	E15-1	29	0.6	E16-1	29	0.6	F1-1	29	0.6	F2-1	29	0.6
F3-1	29	0.6	F4-1	29	0.6	D3-2	37	14	D5-2	37	10	D7-2	37	22	E1-2	38	0.6
E2-2	38	0.6	E3-2	38	0.6	E4-2	38	0.6	E5-2	38	0.6	E6-2	38	0.6	E7-2	38	0.6
E8-2	38	0.6	E9-2	38	0.6	E10-2	38	0.6	E11-2	38	0.6	E12-2	38	0.6	E13-2	38	0.6
E14-2	38	0.6	E15-2	38	0.6	E16-2	38	0.6	F1-2	29	0.6	F2-2	29	0.6	F3-2	29	0.6
F4-2	29	0.6															

2025

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	10	80	A2	2	46.8	A2	9	8.2	A3	2	35	A4	11	72	A5	11	68
A6	2	55	B1	10	60	B2	10	46	B3	11	40	B4	10	28	B5	9	25
B6	13	23.1	B6	10	62.9	B7	9	55	B8	10	44	B9	10	50	B10	11	25
B11	11	60	B12	2	45	B13	10	35	B14	2	20	C1	2	15	C2	10	13
C3	9	15	C4	2	18	C5	2	27	C6	9	20	D1-1	16	15	D2-1	16	10
D3-1	16	14	D4-1	17	6	D5-1	16	10	D6-1	16	12	D7-1	16	22	D8-1	26	20
E1-1	26	0.6	E2-1	26	0.6	E3-1	17	0.6	E4-1	26	0.6	E5-1	26	0.6	E6-1	26	0.6
E7-1	26	0.6	E8-1	26	0.6	E9-1	26	0.6	E10-1	26	0.6	E11-1	26	0.6	E12-1	17	0.6
E13-1	26	0.6	E14-1	26	0.6	E15-1	26	0.6	E16-1	17	0.6	F1-1	17	0.6	F2-1	17	0.6
F3-1	17	0.6	F4-1	17	0.6	D4-2	37	6	D8-2	37	20	E1-2	40	0.6	E2-2	41	0.6
E3-2	40	0.6	E4-2	40	0.6	E5-2	40	0.6	E6-2	40	0.6	E7-2	40	0.6	E8-2	40	0.6
E9-2	40	0.6	E10-2	40	0.6	E11-2	40	0.6	E12-2	40	0.6	E13-2	40	0.6	E14-2	40	0.6
E15-2	40	0.6	E16-2	40	0.6	F1-2	17	0.6	F2-2	17	0.6	F3-2	17	0.6	F4-2	17	0.6

2026

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	11	80	A2	11	55	A3	11	35	A4	10	72	A5	9	68	A6	11	55
B1	2	37.8	B1	11	22.2	B2	5	1	B2	9	45	B3	9	40	B4	11	28
B5	11	25	B6	3	86	B7	11	55	B8	11	44	B9	2	50	B10	2	25
B11	2	60	B12	13	22.5	B12	11	22.5	B13	11	35	B14	11	20	C1	11	15
C2	11	13	C3	11	14	C3	1	1	C4	11	18	C5	11	27	C6	11	19
C6	4	1	D1-1	16	15	D2-1	16	10	D3-1	26	14	D4-1	16	6	D5-1	16	10
D6-1	17	12	D7-1	16	22	D8-1	16	20	E1-1	29	0.6	E2-1	29	0.6	E3-1	29	0.6
E4-1	29	0.6	E5-1	29	0.6	E6-1	18	0.6	E7-1	29	0.6	E8-1	19	0.6	E9-1	19	0.6
E10-1	29	0.6	E11-1	18	0.6	E12-1	29	0.6	E13-1	29	0.6	E14-1	29	0.6	E15-1	29	0.6
E16-1	29	0.6	F1-1	20	0.6	F2-1	29	0.6	F3-1	29	0.6	F4-1	29	0.6	D3-2	37	14
D6-2	37	12	E1-2	38	0.6	E2-2	38	0.6	E3-2	38	0.6	E4-2	38	0.6	E5-2	38	0.6
E6-2	38	0.6	E7-2	38	0.6	E8-2	38	0.6	E9-2	38	0.6	E10-2	38	0.6	E11-2	38	0.6
E12-2	38	0.6	E13-2	38	0.6	E14-2	38	0.6	E15-2	38	0.6	E16-2	38	0.6	F1-2	29	0.6
F2-2	29	0.6	F3-2	29	0.6	F4-2	29	0.6									

2027

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	2	80	A2	2	55	A3	2	35	A4	11	72	A5	11	68	A6	2	55
B1	9	60	B2	2	46	B3	2	40	B4	9	28	B5	2	25	B6	9	86
B7	9	55	B8	9	44	B9	11	50	B10	10	25	B11	9	60	B12	10	45
B13	9	35	B14	2	20	C1	9	15	C2	9	13	C3	3	15	C4	3	18
C5	13	23.1	C5	10	3.9	C6	9	20	D1-1	16	15	D2-1	16	10	D3-1	16	14
D4-1	29	6	D5-1	16	10	D6-1	16	12	D7-1	29	22	D8-1	16	20	E1-1	28	0.6
E2-1	28	0.6	E3-1	28	0.6	E4-1	28	0.6	E5-1	28	0.6	E6-1	29	0.6	E7-1	28	0.6
E8-1	17	0.6	E9-1	28	0.6	E10-1	17	0.6	E11-1	28	0.6	E12-1	28	0.6	E13-1	17	0.6

E14-1	28	0.6	E15-1	28	0.6	E16-1	28	0.6	F1-1	29	0.6	F2-1	17	0.6	F3-1	17	0.6
F4-1	17	0.6	D4-2	37	6	D7-2	37	22	E1-2	39	0.6	E2-2	40	0.6	E3-2	40	0.6
E4-2	40	0.6	E5-2	40	0.6	E6-2	40	0.6	E7-2	40	0.6	E8-2	40	0.6	E9-2	40	0.6
E10-2	40	0.6	E11-2	40	0.6	E12-2	40	0.6	E13-2	40	0.6	E14-2	40	0.6	E15-2	40	0.6
E16-2	40	0.6	F1-2	17	0.6	F2-2	17	0.6	F3-2	17	0.6	F4-2	17	0.6			

2028

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	3	80	A2	8	55	A3	11	35	A4	2	72	A5	2	68	A6	11	55
B1	10	60	B2	10	46	B3	10	40	B4	11	28	B5	10	25	B6	11	86
B7	10	55	B8	10	44	B9	5	50	B10	11	25	B11	11	60	B12	11	45
B13	11	35	B14	11	20	C1	10	15	C2	10	13	C3	10	15	C4	8	18
C5	6	27	C6	10	20	D1-1	16	15	D2-1	16	10	D3-1	16	14	D4-1	16	6
D5-1	29	10	D6-1	16	12	D7-1	16	22	D8-1	29	20	E1-1	17	0.6	E2-1	17	0.6
E3-1	34	0.6	E4-1	17	0.6	E5-1	17	0.6	E6-1	17	0.6	E7-1	17	0.6	E8-1	34	0.6
E9-1	17	0.6	E10-1	34	0.6	E11-1	34	0.6	E12-1	17	0.6	E13-1	34	0.6	E14-1	17	0.6
E15-1	17	0.6	E16-1	17	0.6	F1-1	17	0.6	F2-1	20	0.6	F3-1	34	0.6	F4-1	18	0.6
D5-2	37	10	D8-2	37	20	E1-2	38	0.6	E2-2	39	0.6	E3-2	38	0.6	E4-2	38	0.6
E5-2	38	0.6	E6-2	38	0.6	E7-2	38	0.6	E8-2	38	0.6	E9-2	38	0.6	E10-2	38	0.6
E11-2	38	0.6	E12-2	38	0.6	E13-2	38	0.6	E14-2	38	0.6	E15-2	38	0.6	E16-2	38	0.6
F1-2	20	0.6	F2-2	18	0.6	F3-2	34	0.6	F4-2	18	0.6						

2029

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	10	80	A2	4	1	A2	10	54	A3	10	35	A4	10	72	A5	10	68
A6	10	55	B1	3	60	B2	2	46	B3	2	40	B4	10	28	B5	2	25
B6	10	86	B7	5	55	B8	2	44	B9	10	50	B10	10	25	B11	10	60
B12	10	45	B13	10	35	B14	10	20	C1	2	15	C2	2	13	C3	2	15
C4	10	18	C5	10	26	C5	1	1	C6	2	20	D1-1	16	15	D2-1	34	10
D3-1	16	14	D4-1	16	6	D5-1	16	10	D6-1	16	12	D7-1	20	22	D8-1	16	20
E1-1	18	0.6	E2-1	34	0.6	E3-1	18	0.6	E4-1	18	0.6	E5-1	18	0.6	E6-1	18	0.6
E7-1	18	0.6	E8-1	18	0.6	E9-1	18	0.6	E10-1	19	0.6	E11-1	18	0.6	E12-1	20	0.6
E13-1	19	0.6	E14-1	31	0.6	E15-1	18	0.6	E16-1	20	0.6	F1-1	31	0.6	F2-1	31	0.6
F3-1	31	0.6	F4-1	31	0.6	D2-2	37	10	D7-2	37	22	E1-2	40	0.6	E2-2	38	0.6
E3-2	40	0.6	E4-2	40	0.6	E5-2	40	0.6	E6-2	40	0.6	E7-2	40	0.6	E8-2	40	0.6
E9-2	40	0.6	E10-2	41	0.6	E11-2	40	0.6	E12-2	40	0.6	E13-2	40	0.6	E14-2	40	0.6
E15-2	40	0.6	E16-2	40	0.6	F1-2	31	0.6	F2-2	31	0.6	F3-2	31	0.6	F4-2	34	0.6

2030

地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积	地块	作物	面积
A1	11	80	A2	11	55	A3	11	35	A4	3	72	A5	11	68	A6	3	43.8
A6	11	11.2	B1	10	60	B2	9	46	B3	9	40	B4	9	28	B5	9	25
B6	3	86	B7	9	55	B8	10	44	B9	9	50	B10	9	25	B11	4	60
B12	9	45	B13	9	35	B14	9	20	C1	13	0.1	C1	11	14.9	C2	11	13
C3	11	15	C4	11	18	C5	11	27	C6	11	20	D1-1	16	15	D2-1	16	10
D3-1	16	14	D4-1	34	6	D5-1	34	10	D6-1	31	12	D7-1	16	22	D8-1	16	20
E1-1	20	0.6	E2-1	20	0.6	E3-1	31	0.6	E4-1	31	0.6	E5-1	20	0.6	E6-1	20	0.6
E7-1	20	0.6	E8-1	31	0.6	E9-1	31	0.6	E10-1	18	0.6	E11-1	31	0.6	E12-1	31	0.6
E13-1	18	0.6	E14-1	18	0.6	E15-1	34	0.6	E16-1	31	0.6	F1-1	34	0.6	F2-1	18	0.6
F3-1	34	0.6	F4-1	34	0.6	D4-2	37	6	D5-2	37	10	D6-2	37	12	E1-2	38	0.6
E2-2	39	0.6	E3-2	38	0.6	E4-2	38	0.6	E5-2	38	0.6	E6-2	38	0.6	E7-2	38	0.6
E8-2	38	0.6	E9-2	38	0.6	E10-2	38	0.6	E11-2	38	0.6	E12-2	38	0.6	E13-2	38	0.6
E14-2	38	0.6	E15-2	38	0.6	E16-2	38	0.6	F1-2	18	0.6	F2-2	18	0.6	F3-2	18	0.6
F4-2	31	0.6															

附录 E　问题一的求解代码

```
import pandas as pd
import numpy as np
import pulp
import re
data = pd.read_csv("2023年统计的相关数据.csv")
# 定义函数，接收销售单价区间并返回随机价格
def get_random_price(price_range):
    # 提取区间的上下限
```

```
    lower, upper = map(float, re.findall(r"\d+\.\d+", price_range))
    # 计算均值 (mu) 和标准差 (sigma)
    mu = (lower + upper) / 2
    sigma = (upper - lower) / 6 # (upper - lower) = 6 * sigma
    # 从正态分布中生成随机价格
    random_price = np.random.normal(mu, sigma)
    # 确保生成的随机价格在区间范围内
    random_price = max(min(random_price, upper), lower)
    return random_price

# 定义函数，根据作物名称获取随机销售单价
def get_price_by_crop(dirt_type, crop_num, data):
    # 在数据中查找对应作物名称的销售单价区间
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        price_range = row['销售单价/(元/斤)'].values[0]
        return get_random_price(price_range)
    else:
        return f"作物未找到"

# 根据作物的亩产量生成随机亩销售量
def get_sales_per_mu(mu_production):
    # 计算上下限，变异系数 5% 意味着上下限为均值的 ±5%
    lower_bound = mu_production * (1 - 0.05)
    upper_bound = mu_production * (1 + 0.05)
    # 从均匀分布中随机生成一个值
    sales_per_mu = np.random.uniform(lower_bound, upper_bound)
    return sales_per_mu

# 根据作物名称查找亩产量，并返回随机生成的亩销售量
def get_sales_per_mu_by_crop(dirt_type, crop_num, data):
    # 查找对应作物名称的行
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        # 获取对应的亩产量
        mu_production = row['亩产量/斤'].values[0]
        return get_sales_per_mu(mu_production)
    else:
        return f"作物未找到"

import pandas as pd
import pulp
# 加载数据文件
crop_planting_info = pd.read_csv('2023年的农作物种植情况.csv')
land_data = pd.read_csv('乡村的现有耕地.csv')
crop_data = pd.read_csv('乡村种植的农作物.csv')
statistics_data = pd.read_csv('2023年统计的相关数据.csv')
```

```
# 假设 get_price_by_crop 是一个你定义好的函数
statistics_data['today_price'] = statistics_data.apply(lambda row:
    get_price_by_crop(row['地块类型'], row['作物编号'], statistics_data), axis=1)
statistics_data['sale_volume'] = statistics_data.apply(lambda row:
    get_sales_per_mu_by_crop(row['地块类型'], row['作物编号'], statistics_data), axis=1)

# 获取地块名称和作物编号
land_names = land_data['地块名称']
crop_ids = crop_data['作物编号']

# 初始化单价矩阵、成本矩阵和产量矩阵
price_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
cost_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
yield_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
# 销量矩阵
sale_matrix = pd.DataFrame(index=land_names, columns=crop_ids)

# 填充矩阵，基于地块类型匹配作物的单价、成本和产量信息
for _, land_row in land_data.iterrows():
    land_name = land_row['地块名称']
    land_type = land_row['地块类型']
    filtered_stats = statistics_data[statistics_data['地块类型'] == land_type]
    for _, crop_row in filtered_stats.iterrows():
        crop_id = crop_row['作物编号']
        price_matrix.at[land_name, crop_id] = crop_row['today_price']
        cost_matrix.at[land_name, crop_id] = crop_row['种植成本/(元/亩)']
        yield_matrix.at[land_name, crop_id] = crop_row['亩产量/斤']
        sale_matrix.at[land_name, crop_id] = crop_row['sale_volume']

# 确保所有矩阵都为数值类型
price_matrix = price_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
cost_matrix = cost_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
yield_matrix = yield_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
sale_matrix = sale_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)

# 单价矩阵、成本矩阵、产量矩阵和销量矩阵
price_matrix = price_matrix.astype(float)
cost_matrix = cost_matrix.astype(float)
yield_matrix = yield_matrix.astype(float)
sale_matrix = sale_matrix.astype(float)



# 计算每亩的利润矩阵：利润 = (单价 * 产量) - 成本
# profit_matrix = (price_matrix * np.minimum(yield_matrix, sale_matrix)) - cost_matrix
# 计算滞销部分，即 ReLU(亩产量 - 亩销售量)
unsold_amount = np.maximum(yield_matrix - sale_matrix, 0)
```

```
profit_matrix = (price_matrix * np.minimum(yield_matrix, sale_matrix)) - cost_matrix + (0.5 *
    price_matrix * unsold_amount)

land_data.set_index('地块名称', inplace=True) # 将 '地块名称' 设置为索引

# Step 1: 计算2023年每种作物的种植总量
# 获取2023年的种植面积信息（作物编号+地块的组合，种植面积）
total_area_2023 = crop_planting_info.groupby(['作物编号', '种植地块'])['种植面积/亩'].sum()
# 获取每种作物在不同地块的亩产量
yield_per_mu_2023 = statistics_data.set_index(['作物编号', '地块类型'])['亩产量/斤']
# 计算每个地块上每种作物的总产量
total_production_2023 = total_area_2023 * yield_per_mu_2023
# 定义豆类作物编号
legume_crops = [1, 2, 3, 4, 5, 17, 18, 19]
# 定义未来5年的规划
years = 7
optimal_areas_all_years = {}
profit_total_list = []
def manhattan_density(matrix):
    ones_positions = np.argwhere(matrix == 1)
    total_distance = 0
    total_pairs = 0
    # 遍历所有的1对，计算曼哈顿距离
    for i in range(len(ones_positions)):
        for j in range(i + 1, len(ones_positions)):
            total_distance += np.abs(ones_positions[i][0] - ones_positions[j][0]) +
                np.abs(ones_positions[i][1] - ones_positions[j][1])
            total_pairs += 1
    if total_pairs == 0:
        return 0 # 如果没有1对，返回0
    return total_distance / total_pairs # 平均曼哈顿距离


for year in range(1, years + 1):
    print(f"Year {year} planning...")
    # 创建一个新的线性规划问题
    prob = pulp.LpProblem(f"Crop_Optimization_Year_{year}", pulp.LpMaximize)
    # 重新定义种植面积和二进制变量（每年都是一个新的规划）
    planting_vars = pulp.LpVariable.dicts(f"Planting_Year_{year}", [(land, crop) for land in
        land_names for crop in crop_ids],lowBound=0, cat='Continuous')
    binary_vars = pulp.LpVariable.dicts(f"Binary_Year_{year}", [(land, crop) for land in
        land_names for crop in crop_ids],lowBound=0, upBound=1, cat='Binary')
    lambda_weight = 0
    # 定义目标函数（每年都最大化利润）
    profit = pulp.lpSum([profit_matrix.at[land, crop] * planting_vars[(land, crop)] for land in
        land_names for crop in crop_ids])
    prob += profit - lambda_weight * manhattan_density(binary_vars)
```

```
# 约束 1: 每块地最多只能种植两种作物
for land in land_names:
    prob += pulp.lpSum([binary_vars[(land, crop)] for crop in crop_ids]) <= 2,
        f"Max_2_Crops_{land}_Year_{year}"

# 约束 2: 种植面积和二进制变量的关系
for land in land_names:
    for crop in crop_ids:
        prob += planting_vars[(land, crop)] <= binary_vars[(land, crop)] *
            land_data.at[land, '地块面积/亩'], f"Binary_Area_{land}_{crop}_Year_{year}"

# 约束 3: 每块地的种植面积必须等于地块的可用面积
for land in land_names:
    prob += pulp.lpSum([planting_vars[(land, crop)] for crop in crop_ids]) ==
        land_data.at[land, '地块面积/亩'], f"Exact_Area_{land}_Year_{year}"

# 约束 4: 地块的作物范围限制
for land in land_names:
    if land.startswith(('A', 'B', 'C')):
        for crop in crop_ids:
            if int(crop) > 15:
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_ABC_{land}_{crop}_Year_{year}"
    elif land.startswith('D') and '-2' in land:
        for crop in crop_ids:
            if int(crop) not in [16, 35, 36, 37]:
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_Dn2_{land}_{crop}_Year_{year}"
    elif land.startswith('D') and '-1' in land:
        for crop in crop_ids:
            if int(crop) not in range(16, 35):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_Dn1_{land}_{crop}_Year_{year}"
    elif land.startswith('E') and '-1' in land:
        for crop in crop_ids:
            if int(crop) not in range(17, 35):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_En1_{land}_{crop}_Year_{year}"
    elif land.startswith('E') and '-2' in land:
        for crop in crop_ids:
            if int(crop) not in range(38, 42):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_En2_{land}_{crop}_Year_{year}"
    elif land.startswith('F'):
        for crop in crop_ids:
            if int(crop) not in range(17, 35):
```

```
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_F_{land}_{crop}_Year_{year}"

# 约束 5: 重茬种植限制 (上一年种植的作物在本年不能重茬种植)
if year > 1: # 从第二年开始应用重茬种植限制
    for land in land_names:
        for crop in crop_ids:
            if optimal_areas_all_years[year-1].at[land, crop] > 0:
                prob += planting_vars[(land, crop)] == 0,
                    f"No_Replant_{land}_{crop}_Year_{year}"

# 约束 6: 每种作物的总种植量在2023年总种植量的0.7倍到1.3倍之间
for crop in crop_ids:
    # 计算当前作物的总种植面积
    total_crop_area = pulp.lpSum([planting_vars[(land, crop)] for land in land_names])
    # 约束作物总种植面积不能超过2023年种植面积的1.05倍
    prob += total_crop_area <= 1.3 * total_area_2023.loc[crop],
        f"Max_Area_{crop}_Year_{year}"
    # 约束作物总种植面积不能低于2023年种植面积的0.95倍
    prob += total_crop_area >= 0.7 * total_area_2023.loc[crop],
        f"Min_Area_{crop}_Year_{year}"

# 约束: 豆类植物三年内至少种植一次 (从第三年开始)
if year >= 3:
    for crop in legume_crops:
        total_planting_legume = pulp.lpSum([planting_vars[(land, crop)] for land in
            land_names])
        planting_last_2_years = pulp.lpSum([optimal_areas_all_years[year-1].at[land, crop] +
            optimal_areas_all_years[year-2].at[land, crop] for land in land_names])
        prob += (total_planting_legume + planting_last_2_years) >= 1,
            f"Legume_Planting_Year_{year}_{crop}"

# 求解每年的优化问题
prob.solve(pulp.PULP_CBC_CMD(gapRel=0.05, gapAbs=20, threads=8))

# 获取初步解结果
initial_solution = {}
if pulp.LpStatus[prob.status] == "Optimal":
    for land in land_names:
        for crop in crop_ids:
            initial_solution[(land, crop)] = planting_vars[(land, crop)].varValue

# Step 2: 检查初步解中是否有 Dn-1 或 Dn-2 种植作物 16 的情况
for land in land_names:
    match = re.match(r'D(\d+)-1', land) # 匹配 Dn-1 格式的地块
    if match:
        dn1_land = land
```

```
            dn2_land = f"D{match.group(1)}-2" # 找到对应的 Dn-2 地块
            if dn2_land in list(land_names):
                # 检查是否种植作物 16
                dn1_has_16 = initial_solution[(dn1_land, 16)] > 0
                dn2_has_16 = initial_solution[(dn2_land, 16)] > 0
                # 如果 Dn-1 或 Dn-2 其中之一种植了作物 16，强制两个地块都只种作物 16
                if dn1_has_16 or dn2_has_16:
                    total_area = land_data.at[dn1_land, '地块面积/亩']
                    # 强制 Dn-1 和 Dn-2 都用于种植作物 16
                    initial_solution[(dn1_land, 16)] = total_area
                    initial_solution[(dn2_land, 16)] = 0 # 强制 Dn-2 不种植其他作物
                    # 清除其他作物的种植
                    for crop in crop_ids:
                        if crop != 16:
                            initial_solution[(dn1_land, crop)] = 0
                            initial_solution[(dn2_land, crop)] = 0

    # Step 3: 输出修正后的种植策略
    final_solution_df = pd.DataFrame(0, index=land_names, columns=crop_ids)
    for land in land_names:
        for crop in crop_ids:
            final_solution_df.at[land, crop] = initial_solution[(land, crop)]

    optimal_areas_all_years[year] = final_solution_df # 保存每年的种植结果
    # 初始化总利润为 0
    total_profit = 0

    # 遍历 final_solution_df 中的地块和作物，计算总利润
    for land in land_names:
        for crop in crop_ids:
            # 获取种植面积
            planting_area = final_solution_df.at[land, crop]
            # 获取对应的每亩利润（来自 profit_matrix）
            per_mu_profit = profit_matrix.at[land, crop]
            # 计算该地块作物的利润并累加到总利润
            total_profit += planting_area * per_mu_profit
    profit_total_list.append(total_profit)

# 如果所有年份都成功求解，则输出五年内的所有结果，并保存到CSV文件
if len(optimal_areas_all_years) == years:

    # 输出每年的最优种植结果，并保存为 CSV 文件
    for year in range(1, years + 1):
        optimal_areas_all_years[year].to_csv(f'目标函数2_{year}.csv', index=True)
    print("Results have been saved to CSV files.")
else:
    print("Failed to complete five years' optimization.")
```

附录 F　问题二的求解代码

```
import pandas as pd
data = pd.read_csv("2023年统计的相关数据.csv")
import re
import pulp
import numpy as np

# 定义函数，接收销售单价区间并返回随机价格
def get_random_price(price_range):
    # 提取区间的上下限
    lower, upper = map(float, re.findall(r"\d+\.\d+", price_range))
    # 计算均值 (mu) 和标准差 (sigma)
    mu = (lower + upper) / 2
    sigma = (upper - lower) / 6 # (upper - lower) = 6 * sigma
    # 从正态分布中生成随机价格
    random_price = np.random.normal(mu, sigma)
    # 确保生成的随机价格在区间范围内
    random_price = max(min(random_price, upper), lower)
    return random_price

# 定义函数，根据作物编号和年份获取价格调整系数
def get_price_adjustment(crop_num, year):
    if 17 <= crop_num <= 37:
        # 17-37号作物逐年增长5%
        return 1.05 ** year
    elif 38 <= crop_num <= 40:
        # 38-40号作物逐年下降1%-5%，随机选择一个下降百分比
        decline_rate = np.random.uniform(0.01, 0.05)
        return (1 - decline_rate) ** year
    elif crop_num == 41:
        # 41号作物逐年下降5%
        return (1 - 0.05) ** year
    else:
        return 1

# 定义函数，根据作物名称和年份获取随机销售单价
def get_price_by_crop(dirt_type, crop_num, year, data):
    # 在数据中查找对应作物编号的销售单价区间
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        # 获取价格区间
        price_range = row['销售单价/(元/斤)'].values[0]
        # 生成基础价格
```

```
        base_price = get_random_price(price_range)
        # 根据年份调整价格
        adjustment_factor = get_price_adjustment(crop_num, year)
        final_price = base_price * adjustment_factor
        return final_price
    else:
        return "作物未找到"

# 根据亩产量生成随机亩销售量
def get_sales_per_mu(mu_production):
    cv = 0.05
    # 计算标准差
    sigma = cv * mu_production
    # 从正态分布中随机生成一个值
    sales_per_mu = max(0, np.random.normal(mu_production, sigma))
    return sales_per_mu

# 根据年份调整亩产量
def adjust_production_by_year(mu_production, year):
    # 亩产量每年有+-10%的变化
    adjustment_factor = np.random.uniform(0.9, 1.1) ** year
    return mu_production * adjustment_factor

# 根据作物编号和年份调整销售量
def adjust_sales_by_year(sales_per_mu, crop_num, year):
    if 6 <= crop_num <= 7:
        # 6-7号作物每年销售量上涨5%-10%
        adjustment_factor = np.random.uniform(1.05, 1.10) ** year
    elif 1 <= crop_num <= 5 or 8 <= crop_num <= 16:
        # 1-5号和8-16号作物销售量每年变动在+-5%之间
        adjustment_factor = np.random.uniform(0.95, 1.05) ** year
    else:
        adjustment_factor = 1
    return sales_per_mu * adjustment_factor

# 根据作物名称查找亩产量，并返回随机生成的亩销售量
def get_sales_per_mu_by_crop(dirt_type, crop_num, year, data):
    # 查找对应作物编号的行
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        # 获取对应的亩产量
        mu_production = row['亩产量/斤'].values[0]
        # 根据年份调整亩产量
        adjusted_production = adjust_production_by_year(mu_production, year)
        # 获取基础亩销售量
        sales_per_mu = get_sales_per_mu(adjusted_production)
        # 根据作物编号和年份调整销售量
```

```
        final_sales_per_mu = adjust_sales_by_year(sales_per_mu, crop_num, year)
        return final_sales_per_mu
    else:
        return f"作物未找到"


# 加载数据文件
crop_planting_info = pd.read_csv('2023年的农作物种植情况.csv')
land_data = pd.read_csv('乡村的现有耕地.csv')
crop_data = pd.read_csv('乡村种植的农作物.csv')
statistics_data = pd.read_csv('2023年统计的相关数据.csv')

def construct_matrices(crop_planting_info, land_data, crop_data, statistics_data, year):
    # 更新统计数据中的价格和销量
    statistics_data['today_price'] = statistics_data.apply(
        lambda row: get_price_by_crop(row['地块类型'], row['作物编号'], year, statistics_data),
            axis=1
    )
    statistics_data['sale_volume'] = statistics_data.apply(
        lambda row: get_sales_per_mu_by_crop(row['地块类型'], row['作物编号'], year,
            statistics_data), axis=1
    )
    # 获取地块名称和作物编号
    land_names = land_data['地块名称']
    crop_ids = crop_data['作物编号']
    price_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    cost_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    yield_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    sale_matrix = pd.DataFrame(index=land_names, columns=crop_ids)

    # 填充矩阵
    for _, land_row in land_data.iterrows():
        land_name = land_row['地块名称']
        land_type = land_row['地块类型']
        # 筛选出与当前地块类型匹配的统计数据
        filtered_stats = statistics_data[statistics_data['地块类型'] == land_type]
        for _, crop_row in filtered_stats.iterrows():
            crop_id = crop_row['作物编号']
            original_yield = crop_row['亩产量/斤']
            adjusted_yield = adjust_production_by_year(original_yield, year)
            price_matrix.at[land_name, crop_id] = crop_row['today_price']
            cost_matrix.at[land_name, crop_id] = crop_row['种植成本/(元/亩)'] * (1.05 ** year)
            yield_matrix.at[land_name, crop_id] = adjusted_yield
            sale_matrix.at[land_name, crop_id] = crop_row['sale_volume']
    # 确保所有矩阵都为数值类型
    price_matrix = price_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
    cost_matrix = cost_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
```

```
    yield_matrix = yield_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
    sale_matrix = sale_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)

    # 确保所有矩阵数据类型为 float
    price_matrix = price_matrix.astype(float)
    cost_matrix = cost_matrix.astype(float)
    yield_matrix = yield_matrix.astype(float)
    sale_matrix = sale_matrix.astype(float)
    return price_matrix, cost_matrix, yield_matrix, sale_matrix

# 获取地块名称和作物编号
land_names = land_data['地块名称']
crop_ids = crop_data['作物编号']
land_data_cp = land_data.copy()
land_data.set_index('地块名称', inplace=True) # 将 '地块名称' 设置为索引
# Step 1: 计算2023年每种作物的种植总量
# 获取2023年的种植面积信息（作物编号+地块的组合，种植面积）
total_area_2023 = crop_planting_info.groupby(['作物编号', '种植地块'])['种植面积/亩'].sum()
# 获取每种作物在不同地块的亩产量
yield_per_mu_2023 = statistics_data.set_index(['作物编号', '地块类型'])['亩产量/斤']
# 计算每个地块上每种作物的总产量
total_production_2023 = total_area_2023 * yield_per_mu_2023
# 定义豆类作物编号
legume_crops = [1, 2, 3, 4, 5, 17, 18, 19]
# 定义未来7年的规划
years = 7
optimal_areas_all_years = {}
profit_total_list = []

def manhattan_density(matrix):
    ones_positions = np.argwhere(matrix == 1)
    total_distance = 0
    total_pairs = 0
    # 遍历所有的1对，计算曼哈顿距离
    for i in range(len(ones_positions)):
        for j in range(i + 1, len(ones_positions)):
            total_distance += np.abs(ones_positions[i][0] - ones_positions[j][0]) +
                np.abs(ones_positions[i][1] - ones_positions[j][1])
            total_pairs += 1
    if total_pairs == 0:
        return 0 # 如果没有1对，返回0
    return total_distance / total_pairs # 平均曼哈顿距离


for year in range(1, years + 1):
    print(f"Year {year} planning...")
    # 动态生成每年需要使用的矩阵
```

```
price_matrix, cost_matrix, yield_matrix, sale_matrix = construct_matrices(
    crop_planting_info, land_data_cp, crop_data, statistics_data, year=year
)
# 计算每亩的利润矩阵：利润 =（单价 * 产量）- 成本
profit_matrix = (price_matrix * np.minimum(yield_matrix, sale_matrix)) - cost_matrix

# 创建一个新的线性规划问题
prob = pulp.LpProblem(f"Crop_Optimization_Year_{year}", pulp.LpMaximize)
# 重新定义种植面积和二进制变量（每年都是一个新的规划）
planting_vars = pulp.LpVariable.dicts(f"Planting_Year_{year}", [(land, crop) for land in
    land_names for crop in crop_ids],lowBound=0, cat='Continuous')
binary_vars = pulp.LpVariable.dicts(f"Binary_Year_{year}", [(land, crop) for land in
    land_names for crop in crop_ids],lowBound=0, upBound=1, cat='Binary')

lambda_weight = 0
# 定义目标函数（每年都最大化利润）
profit = pulp.lpSum([profit_matrix.at[land, crop] * planting_vars[(land, crop)] for land in
    land_names for crop in crop_ids])
prob += profit - lambda_weight * manhattan_density(binary_vars)

# 约束 1：每块地最多只能种植两种作物
for land in land_names:
    prob += pulp.lpSum([binary_vars[(land, crop)] for crop in crop_ids]) <= 2,
        f"Max_2_Crops_{land}_Year_{year}"

# 约束 2：种植面积和二进制变量的关系
for land in land_names:
    for crop in crop_ids:
        prob += planting_vars[(land, crop)] <= binary_vars[(land, crop)] *
            land_data.at[land, '地块面积/亩'], f"Binary_Area_{land}_{crop}_Year_{year}"

# 约束 3：每块地的种植面积必须等于地块的可用面积
for land in land_names:
    prob += pulp.lpSum([planting_vars[(land, crop)] for crop in crop_ids]) ==
        land_data.at[land, '地块面积/亩'], f"Exact_Area_{land}_Year_{year}"

# 约束 4：地块的作物范围限制
for land in land_names:
    if land.startswith(('A', 'B', 'C')):
        for crop in crop_ids:
            if int(crop) > 15:
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_ABC_{land}_{crop}_Year_{year}"
    elif land.startswith('D') and '-2' in land:
        for crop in crop_ids:
            if int(crop) not in [16, 35, 36, 37]:
                prob += planting_vars[(land, crop)] == 0,
```

```
                    f"Crop_Range_Dn2_{land}_{crop}_Year_{year}"
        elif land.startswith('D') and '-1' in land:
            for crop in crop_ids:
                if int(crop) not in range(16, 35):
                    prob += planting_vars[(land, crop)] == 0,
                        f"Crop_Range_Dn1_{land}_{crop}_Year_{year}"
        elif land.startswith('E') and '-1' in land:
            for crop in crop_ids:
                if int(crop) not in range(17, 35):
                    prob += planting_vars[(land, crop)] == 0,
                        f"Crop_Range_En1_{land}_{crop}_Year_{year}"
        elif land.startswith('E') and '-2' in land:
            for crop in crop_ids:
                if int(crop) not in range(38, 42):
                    prob += planting_vars[(land, crop)] == 0,
                        f"Crop_Range_En2_{land}_{crop}_Year_{year}"
        elif land.startswith('F'):
            for crop in crop_ids:
                if int(crop) not in range(17, 35):
                    prob += planting_vars[(land, crop)] == 0,
                        f"Crop_Range_F_{land}_{crop}_Year_{year}"

    # 约束 5: 重茬种植限制 (上一年种植的作物在本年不能重茬种植)
    if year > 1: # 从第二年开始应用重茬种植限制
        for land in land_names:
            for crop in crop_ids:
                if optimal_areas_all_years[year-1].at[land, crop] > 0:
                    prob += planting_vars[(land, crop)] == 0,
                        f"No_Replant_{land}_{crop}_Year_{year}"

    # 约束 6: 每种作物的总种植量在2023年总种植量的0.7倍到1.3倍之间
    for crop in crop_ids:
        # 计算当前作物的总种植面积
        total_crop_area = pulp.lpSum([planting_vars[(land, crop)] for land in land_names])
        # 约束作物总种植面积不能超过2023年种植面积的1.3倍
        prob += total_crop_area <= 1.3 * total_area_2023.loc[crop],
            f"Max_Area_{crop}_Year_{year}"
        # 约束作物总种植面积不能低于2023年种植面积的0.7倍
        prob += total_crop_area >= 0.7 * total_area_2023.loc[crop],
            f"Min_Area_{crop}_Year_{year}"

    # 约束: 豆类植物三年内至少种植一次 (从第三年开始)
    if year >= 3:
        for crop in legume_crops:
            total_planting_legume = pulp.lpSum([planting_vars[(land, crop)] for land in
                land_names])
            planting_last_2_years = pulp.lpSum([optimal_areas_all_years[year-1].at[land, crop] +
```

```
            optimal_areas_all_years[year-2].at[land, crop]
                                    for land in land_names])
        prob += (total_planting_legume + planting_last_2_years) >= 1,
            f"Legume_Planting_Year_{year}_{crop}"

# 求解每年的优化问题
prob.solve(pulp.PULP_CBC_CMD(gapRel=0.05, gapAbs=20, threads=8))
# 获取初步解结果
initial_solution = {}
if pulp.LpStatus[prob.status] == "Optimal":
    for land in land_names:
        for crop in crop_ids:
            initial_solution[(land, crop)] = planting_vars[(land, crop)].varValue

# Step 2: 检查初步解中是否有 Dn-1 或 Dn-2 种植作物 16 的情况
for land in land_names:
    match = re.match(r'D(\d+)-1', land) # 匹配 Dn-1 格式的地块
    if match:
        dn1_land = land
        dn2_land = f"D{match.group(1)}-2" # 找到对应的 Dn-2 地块
        if dn2_land in list(land_names):
            # 检查是否种植作物 16
            dn1_has_16 = initial_solution[(dn1_land, 16)] > 0
            dn2_has_16 = initial_solution[(dn2_land, 16)] > 0
            if dn1_has_16 or dn2_has_16:
                total_area = land_data.at[dn1_land, '地块面积/亩']
                initial_solution[(dn1_land, 16)] = total_area
                initial_solution[(dn2_land, 16)] = 0 # 强制 Dn-2 不种植其他作物
                for crop in crop_ids:
                    if crop != 16:
                        initial_solution[(dn1_land, crop)] = 0
                        initial_solution[(dn2_land, crop)] = 0

# Step 3: 输出修正后的种植策略
final_solution_df = pd.DataFrame(0, index=land_names, columns=crop_ids)

for land in land_names:
    for crop in crop_ids:
        final_solution_df.at[land, crop] = initial_solution[(land, crop)]
optimal_areas_all_years[year] = final_solution_df # 保存每年的种植结果
total_profit = 0
for land in land_names:
    for crop in crop_ids:
        planting_area = final_solution_df.at[land, crop]
        per_mu_profit = profit_matrix.at[land, crop]
        total_profit += planting_area * per_mu_profit
profit_total_list.append(total_profit)
```

```
if len(optimal_areas_all_years) == years:
    for year in range(1, years + 1):
        optimal_areas_all_years[year].to_csv(f'目标函数2_{year}.csv', index=True)
    print("Results have been saved to CSV files.")
else:
    print("Failed to complete seven years' optimization.")
```

附录 G　问题三的求解代码

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from tqdm import tqdm
data = pd.read_csv("2023年统计的相关数据.csv")
import pulp
import re

import numpy as np
from scipy.stats import norm

def get_cost_for_p(cost_mean, p, cv=0.1):
    sigma = cv * cost_mean
    x = norm.ppf(p, loc=cost_mean, scale=sigma)
    return x

def get_price_for_p(price_range, p):
    lower, upper = map(float, re.findall(r"\d+\.\d+", price_range))
    mu = (lower + upper) / 2
    sigma = (upper - lower) / 6 # (upper - lower) = 6 * sigma
    y = norm.ppf(p, loc=mu, scale=sigma)
    y = max(min(y, upper), lower)
    return y

from scipy.optimize import curve_fit
def get_sales_per_mu_by_crop_predict(dirt_type, crop_num, year, data):
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        # 获取对应的亩产量
        mu_production = row['亩产量/斤'].values[0]
        price_range = row['销售单价/(元/斤)'].values[0]
        min_price, max_price = map(float, re.findall(r"\d+\.\d+", price_range))
        mean_price = (min_price + max_price) / 2
        # 定义对数正态分布模型
        def log_normal_model(Price, ES_max, Price_mean, Price_min, Price_max):
            mu = np.log(Price_mean)
            sigma = (Price_max - Price_min)/ 6 # 估计标准差
            return (ES_max / (Price * sigma * np.sqrt(2 * np.pi))) * np.exp(-(np.log(Price) -
```

```
                mu)**2 / (2 * sigma**2))

        # 根据年份调整亩产量
        adjusted_production = adjust_production_by_year(mu_production, year)
        max_sales, mean_sales, min_sales = adjusted_production * 1.1, adjusted_production,
            adjusted_production * 0.9
        popt, _ = curve_fit(lambda Price, ES_max: log_normal_model(Price, ES_max, mean_price,
            min_price, max_price),
                            [min_price, mean_price, max_price],
                            [min_sales, mean_sales, max_sales],
                            p0=[max_sales])
        ES_max_fitted = popt[0]
        base_sales_per_mu = max(0, log_normal_model(row['today_price'].values[0], ES_max_fitted,
            mean_price, min_price, max_price))
        final_sales_per_mu = adjust_sales_by_year(base_sales_per_mu, crop_num, year)
        return final_sales_per_mu
    else:
        return f"作物未找到"

# 加载数据文件
crop_planting_info = pd.read_csv('2023年的农作物种植情况.csv')
land_data = pd.read_csv('乡村的现有耕地.csv')
crop_data = pd.read_csv('乡村种植的农作物.csv')
statistics_data = pd.read_csv('2023年统计的相关数据.csv')

# 定义每个类别对应的作物编号
crop_categories = {
    '豆类粮食': [1, 2, 3, 4, 5],
    '豆类蔬菜': [17, 18, 19],
    '淀粉类': [13, 20],
    '谷物类': [6, 7, 8, 9, 10, 11, 14, 15, 16],
    '叶菜类蔬菜': [23, 26, 27, 28, 30, 32, 33, 34, 35],
    '根茎类蔬菜': [36, 37],
    '果菜类蔬菜': [12, 21, 22, 24, 25, 29, 31],
    '食用菌': [38, 39, 40, 41]
}

def get_price_by_predict(dirt_type, crop_num, year, data):
    # 在数据中查找对应作物编号的销售单价区间
    row = data[(data['作物编号'] == crop_num) & (data['地块类型'] == dirt_type)]
    if not row.empty:
        # 生成100个随机概率p值
        p_values = np.random.uniform(0, 1, 100)
        # 生成100个成对的成本和价格数据点
        cost_data = [get_cost_for_p(row['种植成本/(元/亩)'].values[0], p) for p in p_values]
        price_data = [get_price_for_p(row['销售单价/(元/斤)'].values[0], p) for p in p_values]
        # 转换为NumPy数组
```

```
        cost_data = np.array(cost_data).reshape(-1, 1) # 将成本数据转成二维数组
        price_data = np.array(price_data)
        # 使用线性回归模型拟合
        model = LinearRegression()
        model.fit(cost_data, price_data)

        # 预测价格数据
        price_pred = model.predict(np.array(row['种植成本/(元/亩)'].values[0] * (1.05 **
            year)).reshape(-1, 1))[0]

        # 根据年份调整价格
        adjustment_factor = get_price_adjustment(crop_num, year)
        final_price = price_pred * adjustment_factor

        return final_price
    else:
        return "作物未找到"

def construct_matrices(crop_planting_info, land_data, crop_data, statistics_data, year):
    # 更新统计数据中的价格和销量
    statistics_data['today_price'] = statistics_data.apply(
        lambda row: get_price_by_predict(row['地块类型'], row['作物编号'], year,
            statistics_data), axis=1
    )
    statistics_data['sale_volume'] = statistics_data.apply(
        lambda row: get_sales_per_mu_by_crop_predict(row['地块类型'], row['作物编号'], year,
            statistics_data), axis=1
    )
    # 获取地块名称和作物编号
    land_names = land_data['地块名称']
    crop_ids = crop_data['作物编号']
    # 初始化矩阵
    price_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    cost_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    yield_matrix = pd.DataFrame(index=land_names, columns=crop_ids)
    sale_matrix = pd.DataFrame(index=land_names, columns=crop_ids)

    # 填充单价、成本、产量和销量矩阵
    for _, land_row in tqdm(land_data.iterrows(), total=len(land_data), desc="Processing land
        data"):
        land_name = land_row['地块名称']
        land_type = land_row['地块类型']
        # 筛选出与当前地块类型匹配的统计数据
        filtered_stats = statistics_data[statistics_data['地块类型'] == land_type]
        for _, crop_row in filtered_stats.iterrows():
            crop_id = crop_row['作物编号']
            # 获取原始亩产量
```

```
            original_yield = crop_row['亩产量/斤']
            # 根据年份调整亩产量
            adjusted_yield = adjust_production_by_year(original_yield, year)
            # 填充单价、成本、产量矩阵
            price_matrix.at[land_name, crop_id] = crop_row['today_price']
            cost_matrix.at[land_name, crop_id] = crop_row['种植成本/(元/亩)'] * (1.05 ** year)
            yield_matrix.at[land_name, crop_id] = adjusted_yield
            sale_matrix.at[land_name, crop_id] = crop_row['sale_volume']

    price_matrix = price_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
    cost_matrix = cost_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
    yield_matrix = yield_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)
    sale_matrix = sale_matrix.apply(pd.to_numeric, errors='coerce').fillna(0)

    price_matrix = price_matrix.astype(float)
    cost_matrix = cost_matrix.astype(float)
    yield_matrix = yield_matrix.astype(float)
    sale_matrix = sale_matrix.astype(float)

    return price_matrix, cost_matrix, yield_matrix, sale_matrix

# 获取地块名称和作物编号
land_names = land_data['地块名称']
crop_ids = crop_data['作物编号']

land_data_cp = land_data.copy()
land_data.set_index('地块名称', inplace=True) # 将 '地块名称' 设置为索引
# Step 1: 计算2023年每种作物的种植总量
# 获取2023年的种植面积信息（作物编号+地块的组合，种植面积）
total_area_2023 = crop_planting_info.groupby(['作物编号', '种植地块'])['种植面积/亩'].sum()
# 获取每种作物在不同地块的亩产量
yield_per_mu_2023 = statistics_data.set_index(['作物编号', '地块类型'])['亩产量/斤']
# 计算每个地块上每种作物的总产量
total_production_2023 = total_area_2023 * yield_per_mu_2023
# 定义豆类作物编号
legume_crops = [1, 2, 3, 4, 5, 17, 18, 19]
# 定义未来7年的规划
years = 7
optimal_areas_all_years = {}
profit_total_list = []

def manhattan_density(matrix):
    ones_positions = np.argwhere(matrix == 1)
    total_distance = 0
    total_pairs = 0
    # 遍历所有的1对，计算曼哈顿距离
    for i in range(len(ones_positions)):
```

```
        for j in range(i + 1, len(ones_positions)):
            total_distance += np.abs(ones_positions[i][0] - ones_positions[j][0]) +
                np.abs(ones_positions[i][1] - ones_positions[j][1])
            total_pairs += 1

    if total_pairs == 0:
        return 0 # 如果没有1对，返回0
    return total_distance / total_pairs # 平均曼哈顿距离

for year in range(1, years + 1):
    print(f"Year {year} planning...")
    # 动态生成每年需要使用的矩阵
    price_matrix, cost_matrix, yield_matrix, sale_matrix = construct_matrices(
        crop_planting_info, land_data_cp, crop_data, statistics_data, year=year
    )
    # 计算每亩的利润矩阵：利润 = (单价 * 产量) - 成本
    profit_matrix = (price_matrix * np.minimum(yield_matrix, sale_matrix)) - cost_matrix

    # 创建一个新的线性规划问题
    prob = pulp.LpProblem(f"Crop_Optimization_Year_{year}", pulp.LpMaximize)
    planting_vars = pulp.LpVariable.dicts(f"Planting_Year_{year}", [(land, crop) for land in
        land_names for crop in crop_ids],lowBound=0, cat='Continuous')
    binary_vars = pulp.LpVariable.dicts(f"Binary_Year_{year}", [(land, crop) for land in
        land_names for crop in crop_ids],lowBound=0, upBound=1, cat='Binary')

    lambda_weight = 0
    # 定义目标函数（每年都最大化利润）
    profit = pulp.lpSum([profit_matrix.at[land, crop] * planting_vars[(land, crop)] for land in
        land_names for crop in crop_ids])
    prob += profit - lambda_weight * manhattan_density(binary_vars)

    # 约束 1：每块地最多只能种植两种作物
    for land in land_names:
        prob += pulp.lpSum([binary_vars[(land, crop)] for crop in crop_ids]) <= 2,
            f"Max_2_Crops_{land}_Year_{year}"

    # 约束 2：种植面积和二进制变量的关系
    for land in land_names:
        for crop in crop_ids:
            prob += planting_vars[(land, crop)] <= binary_vars[(land, crop)] *
                land_data.at[land, '地块面积/亩'], f"Binary_Area_{land}_{crop}_Year_{year}"

    # 约束 3：每块地的种植面积必须等于地块的可用面积
    for land in land_names:
        prob += pulp.lpSum([planting_vars[(land, crop)] for crop in crop_ids]) ==
            land_data.at[land, '地块面积/亩'], f"Exact_Area_{land}_Year_{year}"
```

```
# 约束 4: 地块的作物范围限制
for land in land_names:
    if land.startswith(('A', 'B', 'C')):
        for crop in crop_ids:
            if int(crop) > 15:
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_ABC_{land}_{crop}_Year_{year}"
    elif land.startswith('D') and '-2' in land:
        for crop in crop_ids:
            if int(crop) not in [16, 35, 36, 37]:
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_Dn2_{land}_{crop}_Year_{year}"
    elif land.startswith('D') and '-1' in land:
        for crop in crop_ids:
            if int(crop) not in range(16, 35):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_Dn1_{land}_{crop}_Year_{year}"
    elif land.startswith('E') and '-1' in land:
        for crop in crop_ids:
            if int(crop) not in range(17, 35):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_En1_{land}_{crop}_Year_{year}"
    elif land.startswith('E') and '-2' in land:
        for crop in crop_ids:
            if int(crop) not in range(38, 42):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_En2_{land}_{crop}_Year_{year}"
    elif land.startswith('F'):
        for crop in crop_ids:
            if int(crop) not in range(17, 35):
                prob += planting_vars[(land, crop)] == 0,
                    f"Crop_Range_F_{land}_{crop}_Year_{year}"

# 约束 5: 重茬种植限制 (上一年种植的作物在本年不能重茬种植)
if year > 1: # 从第二年开始应用重茬种植限制
    for land in land_names:
        for crop in crop_ids:
            if optimal_areas_all_years[year-1].at[land, crop] > 0:
                prob += planting_vars[(land, crop)] == 0,
                    f"No_Replant_{land}_{crop}_Year_{year}"

# 约束 6: 每类作物的总种植量在2023年总种植量的0.7倍到1.3倍之间
for category, crops in crop_categories.items():
    total_category_area = pulp.lpSum([planting_vars[(land, crop)] for land in land_names for
        crop in crops])
    total_category_area_2023 = total_area_2023.loc[crops].sum()
    prob += total_category_area <= 1.3 * total_category_area_2023,
```

```
            f"Max_Area_{category}_Year_{year}"
        prob += total_category_area >= 0.7 * total_category_area_2023,
            f"Min_Area_{category}_Year_{year}"

    # 约束：豆类植物三年内至少种植一次（从第三年开始）
    if year >= 3:
        for crop in legume_crops:
            total_planting_legume = pulp.lpSum([planting_vars[(land, crop)] for land in
                land_names])
            planting_last_2_years = pulp.lpSum([optimal_areas_all_years[year-1].at[land, crop] +
                optimal_areas_all_years[year-2].at[land, crop]
                                                for land in land_names])
            prob += (total_planting_legume + planting_last_2_years) >= 1,
                f"Legume_Planting_Year_{year}_{crop}"
    # 约束：如果种植了豆类粮食，必须种植谷物类
    legume_grains = crop_categories['豆类粮食']
    grains = crop_categories['谷物类']

    for land in land_names:
        # 引入二进制变量，表示是否种植豆类粮食和谷物类
        legume_grain_planted = pulp.lpSum([binary_vars[(land, crop)] for crop in legume_grains])
        grain_planted = pulp.lpSum([binary_vars[(land, crop)] for crop in grains])
        prob += legume_grain_planted <= grain_planted,
            f"Legume_Grain_Planting_{land}_Year_{year}"

    # 约束：如果种植了蔬菜类，必须种植食用菌
    vegetable_crops = crop_categories['豆类蔬菜'] + crop_categories['叶菜类蔬菜'] +
        crop_categories['根茎类蔬菜'] + crop_categories['果菜类蔬菜']
    mushroom_crops = crop_categories['食用菌']

    for land in land_names:
        # 引入二进制变量，表示是否种植蔬菜类和食用菌类
        vegetable_planted = pulp.lpSum([binary_vars[(land, crop)] for crop in vegetable_crops])
        mushroom_planted = pulp.lpSum([binary_vars[(land, crop)] for crop in mushroom_crops])
        # 如果种植了蔬菜类，则强制种植食用菌
        prob += vegetable_planted <= mushroom_planted,
            f"Vegetable_Mushroom_Planting_{land}_Year_{year}"

    # 蔬菜种类的作物编号
    vegetable_crops = crop_categories['豆类蔬菜'] + crop_categories['叶菜类蔬菜'] +
        crop_categories['果菜类蔬菜']

    # 状态约束5：Dn-1 和 Fn-1 必须覆盖全部的蔬菜种类
    for land in land_names:
        if land.startswith('D') and '-1' in land:
            dn1_land = land
            fn1_land = dn1_land.replace('D', 'F') # 假设 Fn-1 对应的是相同编号的 F 地块
```

```
        if fn1_land in land_names:
            for vegetable in vegetable_crops:
                # 至少在 Dn-1 或 Fn-1 上种植一种蔬菜作物
                prob += binary_vars[(dn1_land, vegetable)] + binary_vars[(fn1_land,
                    vegetable)] >= 1,
                    f"Vegetable_Coverage_{vegetable}_Dn1_Fn1_{dn1_land}_Year_{year}"

# 状态约束6: En-1 和 Fn-2 必须覆盖全部的蔬菜种类
for land in land_names:
    if land.startswith('E') and '-1' in land:
        en1_land = land
        fn2_land = en1_land.replace('E', 'F').replace('-1', '-2')
        if fn2_land in land_names:
            for vegetable in vegetable_crops:
                prob += binary_vars[(en1_land, vegetable)] + binary_vars[(fn2_land,
                    vegetable)] >= 1,
                    f"Vegetable_Coverage_{vegetable}_En1_Fn2_{en1_land}_Year_{year}"

# 求解每年的优化问题
prob.solve(pulp.PULP_CBC_CMD(gapRel=0.05, gapAbs=20, threads=8))

initial_solution = {}
if pulp.LpStatus[prob.status] == "Optimal":
    for land in land_names:
        for crop in crop_ids:
            initial_solution[(land, crop)] = planting_vars[(land, crop)].varValue

# Step 2: 检查初步解中是否有 Dn-1 或 Dn-2 种植作物 16 的情况
for land in land_names:
    match = re.match(r'D(\d+)-1', land) # 匹配 Dn-1 格式的地块
    if match:
        dn1_land = land
        dn2_land = f"D{match.group(1)}-2" # 找到对应的 Dn-2 地块
        if dn2_land in list(land_names):
            dn1_has_16 = initial_solution[(dn1_land, 16)] > 0
            dn2_has_16 = initial_solution[(dn2_land, 16)] > 0
            if dn1_has_16 or dn2_has_16:
                total_area = land_data.at[dn1_land, '地块面积/亩']
                initial_solution[(dn1_land, 16)] = total_area
                initial_solution[(dn2_land, 16)] = 0 # 强制 Dn-2 不种植其他作物
                for crop in crop_ids:
                    if crop != 16:
                        initial_solution[(dn1_land, crop)] = 0
                        initial_solution[(dn2_land, crop)] = 0
# Step 3: 输出修正后的种植策略
final_solution_df = pd.DataFrame(0, index=land_names, columns=crop_ids)
for land in land_names:
```

```
        for crop in crop_ids:
            final_solution_df.at[land, crop] = initial_solution[(land, crop)]
    optimal_areas_all_years[year] = final_solution_df # 保存每年的种植结果
    total_profit = 0
    for land in land_names:
        for crop in crop_ids:
            planting_area = final_solution_df.at[land, crop]
            per_mu_profit = profit_matrix.at[land, crop]
            total_profit += planting_area * per_mu_profit
    profit_total_list.append(total_profit)
# 如果所有年份都成功求解，则输出七年内的所有结果，并保存到CSV文件
if len(optimal_areas_all_years) == years:
    for year in range(1, years + 1):
        optimal_areas_all_years[year].to_csv(f'目标函数2_{year}.csv', index=True)
    print("Results have been saved to CSV files.")
else:
    print("Failed to complete seven years' optimization.")
```