基于历史数据的农作物种植策略

摘　要

随着农业生产向智能化发展，耕地资源的最大化利用和可持续发展成为当前种植业的关键问题。本文围绕华北山区某乡村的农作物种植策略展开，根据题目信息建立数学模型，采用**线性规划**等研究方法在避免重茬种植、保障土地质量的约束下，合理分配农作物分布及耕地面积以促进经济的可持续发展。

对于问题一，为在满足约束条件的前提下最大化利润，本文选择使用线性规划将耕地类型、耕地面积、预期销售量、销售价格等作为变量，以轮作、三年内每一地块必须种植一次豆类作物、耕地面积限制等为约束条件，建立目标函数进行求解。首先，根据相关论文，我们以 2023 年产量作为 2024 年农作物的预期销售量，然后分别采用**取平均值**和**映射**的方式对销售单价和亩产量进行了数据预处理。之后，分别在超过部分滞销和超过部分以 50%的价格降价销售的情况下拟合计算结果。

对于问题二，在满足所有第一题约束条件和第二题新增的约束条件的前提下最大化利润，继续使用线性规划模型添加每年每种植物的预期销售量，在每块地的种植成本和价格变化率等变量，再根据题目条件建立相应的预期销售量，种植成本和价格变化率的数组，在一定范围内随机调整相关作物每年的预期销售量，种植成本和价格，再在计算利润时根据这些变化率对原来的相应变量分别进行加权，之后按超过部分以 50%的价格降价销售的情况你和计算结果。

对于问题三，在满足第二题所有约束条件下添加新的约束条件来提现一些作物之间的替代性与互补性，实现利润最大化。根据相关论文资料，我们发现黄豆、黑豆、红豆、绿豆之间有替代性；小麦、玉米、谷子、高粱之间有替代性；菠菜和西红柿互补性较明显。以这些条件为根据，建立三个新的约束条件。对于替代性，约束每一年相关作物的预期销售量总和应保持稳定；对于互补性，约束每一年相关作物的价格增减率应保持一致。根据问题二中的数据，建立**相关系数矩阵**，用于探索预期销售量与销售价格、种植成本之间存在的一定的相关性。

关键词：线性规划　多目标优化　不确定性分析　农作物种植方案 相关系数矩阵

一、问题重述

1.1 问题背景

随着乡村振兴脚步的加速和农业种植趋向现代化、智能化，农作物种植分配优化问题逐渐成为举足轻重的话题。在耕地面积、土壤肥力有限的情况下，我们必须根据不同地形合理分配每一季农作物种植策略，以最大化利用有限土壤资源。

某华北山区乡村地形复杂，有多种耕地和大棚，适宜种植粮食、蔬菜、水稻、豆类多种不同的农作物。由于气候条件限制，每年大部分耕地只能种植一季作物，而大棚可以种植两季。根据作物轮作、田间聚集管理、每块地块三年内至少种植一次豆类作物以保障土地质量等要求，现需设计最优的农作物种植方案，帮助该地区实现农作物经济效益最大化，减少种植风险，并符合可持续发展的原则。

1.2 问题提出

问题是层层递进的，问题 2 和问题 3 在问题 1 的基础上做出了延伸拓展。

问题 1：在假设未来预期销售量、种植成本、亩产量和销售价格相对于 2023 年保持稳定的前提下，分别针对产量超过预期销售量的部份滞销处理和降价 50%出售两种情况，为该地区提供 2024～2030 年的农作物种植策略。

问题 2：在假设小麦和玉米的预期销售量每年以 5%～10%的年增长率上升、其他作物的预期销售量每年有±5%的波动、亩产量每年有±10%的波动的情况下，制定 2024～2030 年的最优农作物种植方案。

问题 3：在问题 2 的基础之上，考虑不同农作物之间替代性、互补性引起的相互影响，提出 2024～2030 年的最优种植方案，并将结果与问题 2 做出对比。

二、问题分析

2.1 数据分析

分析题目给出的信息以及附件中的数据，发现 2023 年农作物销售单价以价格范围的形式展现，故先将该类数据取平均值。将耕地进行编号后分类处理，便于更直观地把握数据特征。最后大量的数据以为着数据质量可能无法保证，对异常值和无关数据进行数据清洗。

2.2 问题一分析

问题一属于典型的**农作物种植分配优化问题**，要求在给定假设条件下，设计出 2024～2030 年最优的农作物种植策略。针对两种不同的销售策略，问题可以分为两种情形进行分析。

首先，我们可以建立一个**基于产量与销售量关系的数学模型 I**，该模型针对第一种情形，即作物产量超过预期销售量部分将滞销、造成浪费。通过该模型，优化种植面积、作物种类与耕地分配，以最大化销售收益并减少浪费带来的损失。

然后，建立一个**基于降价处理的数学模型 II**，该模型用于第二种情形，即作物产量超出部分以原价的 50%出售。模型 II 同样通过优化种植策略，考虑在降价处理情况下

的整体收益，并尽量避免不必要的降价情况。

通过对模型 I 和模型 II 的计算结果进行预测，可以分别获得在不同销售情形下的最优种植方案。随后，比较两种模型的结果，分析不同策略下对乡村经济效益的影响，并结合实际情况提出合理的种植优化建议。

2.3 问题二分析

对于问题二，首先题目推翻了问题一中亩产量、预期销售量、销售价格等与 2023 年保持一致的假设，引入了新的不确定性因素。将这些变化量化后加重新加载到公式中，其余的目标函数和约束条件与问题一保持一致，建立模型进行求解。

2.4 问题三分析

对于问题三，在问题二的基础上进一步复杂化，考虑了农作物之间的替代性和互补性，以及预期销售量、销售价格、种植成本之间的相关性。这种情形下，我们不仅要考虑种植面积的优化，还要进一步分析不同农作物的相互影响，确保种植方案在各种相关因素的作用下仍然保持最优。

首先，我们可以在问题二的模型基础上，构建一个新的模型 III，该模型引入了作物之间的替代性和互补性。例如，某些作物可能在特定条件下相互替代，从而影响种植决策，而其他作物的互补性则可能带来协同效应，提升整体收益。在这个模型中，销售量和价格的变化不仅受到市场因素的影响，还可能与其他作物的种植情况相关。

其次，我们要通过对相关性进行建模，确定销售量、价格、种植成本等因素之间的线性或非线性关系。此时可以借助相关性分析方法，来确定这些因素之间的相关性强弱，从而指导我们优化农作物的种植策略。

通过模拟数据进行求解，可以得到在考虑作物替代性和互补性情况下的最优种植方案。最后，将模型 III 的结果与问题二的结果进行比较，分析作物之间的相关性对整体种植策略和收益的影响，从而为乡村的长期可持续种植提供更为全面的优化建议。

三、模型假设

为了方便模型的建立与模型的可行性，我们这里首先对模型提出一些假设，使得模型更加完备，预测的结果更加合理。

1、任何未在数据集中明确记录的外部因素（如宏观经济状况、政策变动等）对预测结果的影响被忽略。

2、数据完整性：假设提供的数据集是完整的，没有缺失值或错误记录。

3、预期稳定性：假设以 2023 年农作物产量作为 2023 年的预期销售量。

4、均匀分布假设：每块农田上农作物均匀分配，以便于田间管理和农机作业。

四、符号说明

为了方便我们模型的建立与求解过程 ，我们这里对使用到的关键符号进行以下说明：

符号	符号说明
C_j	第 j 种作物的种植成本（元/亩）
$Q_{i,t}$	第 i 种作物在第 t 年的总产量（斤）
$E_{i,t}$	第 i 种作物在第 t 年的超出预期的销售量（斤）
p_j	第 j 种作物的销售价格（元/斤）
$S_{i,j}$	j 作物在 i 地上的亩产量（斤/亩）
$X_{i,j,s,t}$	第 i 块地在第 j 种作物第 s 年第 t 季度的面积（亩）
K_i	第 i 块土地的面积（亩）
$Y_{i,j,s,t}$	j 作物在 i 地块上 s 年第 t 季度是否种植
$E_{i,j}$	2023 年 j 作物在 i 地块上的生产量（斤）
$P_{i,t}$	第 i 种作物在第 t 年的销售价格（元/斤）
$C_{i,t}$	第 i 种作物在第 t 年的种植成本（元/亩）
$A_{i,t}$	第 i 种作物在第 t 年的种植总面积（亩）
TP	总利润（元）

（注：这里只列出论文各部分通用符号，个别模型单独使用的符号在首次引用时会进行说明。）

五、模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 问题一模型的建立

(a) 目标函数的数学表示

首先分析题目要求得到的数据是 2024～2030 年农作物种植的最优方案，那么我们可以根据已知的 2023 年农作物亩产量列出 2024～2030 年的**实际总产量（RP）**：

$$\sum_{j=1}^{41}\sum_{s=2024}^{2030}\sum_{i=A_1}^{F_4}\sum_{t=1}^{2} x\cdot S_{i,j}$$

由于我们假设以 2023 年的总产量为未来的预期总销售量，故列出未来的**实际销售量（RS）**为：

$$\sum_{j=1}^{41}\sum_{s=2024}^{2030}\sum_{i=A_1}^{F_4}\sum_{t=1}^{2} x\cdot S_{ij}$$

农作物**超出预期销售量的部份（Ex）**为：

$$RP - RS$$

根据历史数据可得出 2024～2030 年间种植农作物的**总成本（TC）**为：

$$\sum_{j=1}^{41} C_j\cdot\left(\sum_{s=2024}^{2030}\sum_{i=A_1}^{F_4}\sum_{t=1}^{2} x\right)$$

根据销售单价和产量可以得出**总收入（TR）**：

$$\sum_{j=1}^{41} P_j\cdot\left(\sum_{s=2024}^{2030}\min\left(\sum_{i=A_1}^{F_4}\sum_{t=1}^{2} x\cdot S_{i,j},\ \sum_{i=A_1}^{F_4} E_{i,j}\right)\right)$$

综合以上信息列出目标函数。在第一小问超过预期销售量的部份滞销的情况下，**总利润（TP）**为：

$$TR - TC$$

在第二小问超过预期销售量的部份以 50%的价格降价出售的情况下，**总利润（TP）**为：

$$TR - TC + \frac{1}{2}\sum_{j=1}^{41} P_j\cdot\sum_{s=2024}^{2030}\max\left(0,\ \sum_{i=A_1}^{F_4}\left(\sum_{t=1}^{2} x\cdot S_{i,j} - E_{i,j}\right)\right)$$

(b) 约束条件

规定所有$X_{i,j,s,t}$均满足大于等于 0。

1. 每块耕地上种植的作物面积不得超过该地块的面积：

$$K_i \geq \sum_{j=1}^{41} X_{i,j,s,t}$$

其中 s，t 为一确定的具体常数，s 为[2024，2030]之间的任意一个整数，t 为 1 或 2。

2. 种植面积不能太小，规定任意$X_{i,s,t,j}$都要大于等于 0.1 亩或者等于 0：

$$\begin{cases} X_{i,j,s,t} \geq 0.1, & X_{i,j,s,t} \neq 0 \\ X_{i,j,s,t} = 0, & X_{i,j,s,t} = 0 \end{cases}$$

3. 连续两季同一块耕地不能种植同一种植物：

$$\begin{cases} Y_{i,j,s-1,2} + Y_{i,j,s,1} \leq 1 \\ Y_{i,j,s,1} + Y_{i,j,s,2} \leq 1 \end{cases}$$

4. 平旱地、梯田、山坡地只能种植一季水稻之外的粮食：

①粮食之外的作物以及水稻不能种植

$$\sum_{j=16}^{41} X_{i,j,s,t} = 0$$

其中，$j \in (A_1 \sim C_6)$。

②只能种植单季粮食

$$Y_{i,j,s,1} + Y_{i,j,s,2} = 0$$

5. 水浇地可以种植一季水稻或者两季蔬菜

①水稻和蔬菜之外的农作物不能种植

$$\sum_{j=1}^{15} X_{i,j,s,t} + \sum_{j=38}^{41} X_{i,j,s,t} = 0$$

②若种植了水稻就不能种植蔬菜，种植了蔬菜就不能种植水稻

$$(Y_{i,16,s,1} + Y_{i,16,s,2})(Y_{i,j,s,1} + Y_{i,j,s,2}) = 0$$

其中，$j \in [17,37]$，$j \in \mathbb{N}$。

③若种植水稻，只能种植单季

$$Y_{i,16,s,1} + Y_{i,16,s,2} = 0$$

④若种植蔬菜则第一季必须是 17～34 号作物，第二季度必须是 35～37 号作物

$$\sum_{j=35}^{37} X_{i,j,s,1} = 0, \sum_{j=17}^{34} X_{i,j,s,2} = 0$$

6. 普通大棚第一季种 17～34 号作物，第二季种 38～41 号作物，并且 38～41 号作物只能在普通大棚第二季种

$$\sum_{j=1}^{16} X_{i,j,s,t} + \sum_{j=35}^{41} X_{i,j,s,t} = 0$$

其中，$i \in$ 普通大棚，$t = 1$。

$$\sum_{j=1}^{37} X_{i,j,s,t} = 0$$

其中，$i \in$ 普通大棚，$t = 2$。

$$\sum_{j=38}^{41} X_{i,j,s,t} = 0$$

其中，$t \neq 2$ 或 $i \notin$普通大棚。

7. 智能大棚可以种植两季 17～34 号作物

$$\sum_{j=1}^{16} X_{i,j,s,t} + \sum_{j=35}^{41} X_{i,j,s,t} = 0$$

其中，$i \in$ 智能大棚，$t \in \{1, 2\}$。

8. 三年内每块地至少种满豆子一次

$$\sum_{t=1}^{2} \sum_{j \in \{1,2,3,4,5,17,18,19\}} \sum_{s=2025}^{2029} X_{i,j,s,t} = K_i$$

5.1.2 问题一模型的求解

将预处理数据分别带入两种情景的数学模型，通过 Python 模拟计算得到最优种植农作物方案，编程代码详见附录。两种情景下的详细种植方案见附件表格，例图如下所示，此处不再赘述。两种情况下通过线性规划模型模拟得到的最优种植策略下的 7 年总利润分别为 38927719.66111925 元和 38929234.59988302 元。

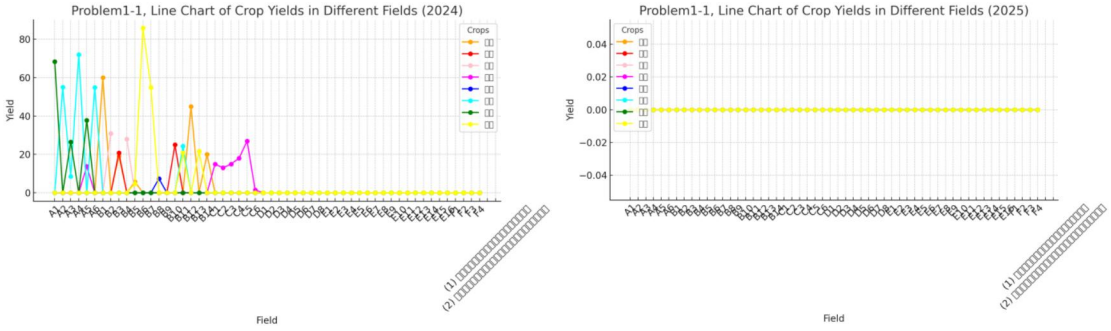

图 1 第一题第一小问的部分作物分配方案例图

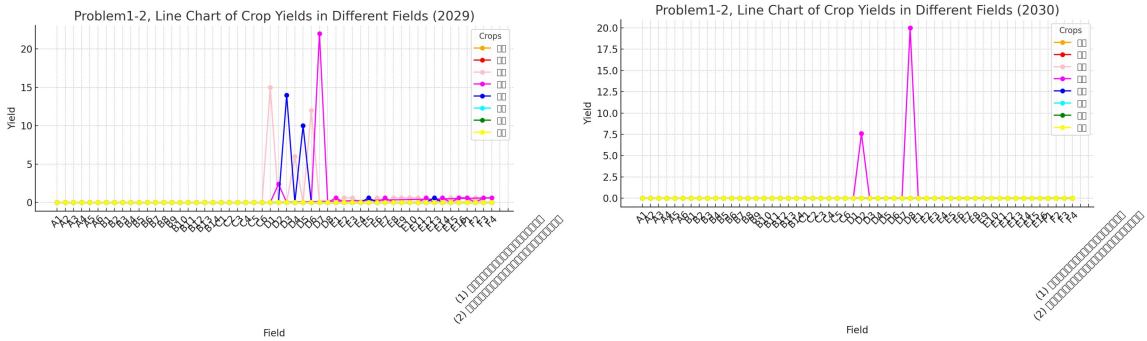

图 2 第一题第二小问的部份作物分配方案例图

5.2 问题二模型的建立与求解

5.2.1 问题二模型的建立

我们需要解决的问题是在问题一的基础上加入对农作物亩产量、种植价格、种植成本等变化因素的考虑，其余目标函数和约束条件不变，在上一题的基础之上随机调整相关作物每年的预期销售量、种植成本和销售价格，再在计算利润时对相应变量进行加权。

(a) 目标函数的数学表示

我们的模型目标函数是为了需大化 2024～2030 年间的总利润，该利润由每年的农作物产量、种植成本、销售价格等因素决定。目标函数**总利润（TP）**的数学表达式如下：

$$\text{Maximize} \sum_{t=2024}^{2030} \sum_{t=1}^{41} ((Q_{i,t} - E_{i,t}) \cdot P_{i,t} + E_{i,t} \cdot 0.5 \cdot P_{i,2023} - C_{i,t} \cdot A_{i,t})$$

(b) 不确定因素的确定方法

1. 小麦和玉米的年销售量增长率介于 5%～10%之间。

$$\text{ExpectedSales}_{\text{wheat,corn,t}} = \text{Sales}_{t-1} \cdot (1 + \text{wheat\_corn\_growth\_rate}_t)$$

2. 其他作物的年销售变化介于-5%～+5%之间。

$$\text{ExpectedSales}_{\text{other,crops,t}} = \text{Sales}_{t-1} \cdot (1 + \text{other\_crop\_sales\_variation}_t)$$

3. 每年亩产量可能会因为气候变化==等因素波动，变化范围为±10%。

$$\text{Yield}_{i,t} = \text{Yield}_{i,t-1} \cdot (1 + \text{yield\_variation}_t)$$

4. 种植成本年增长率为 5%。

$$\text{Cost}_{i,t} = \text{Cost}_{i,t-1} \cdot (1 + 0.05)$$

5. 蔬菜种植成本年增长率为 5%。

$$\text{Price}_{\text{vegetable,t}} = \text{Price}_{\text{vegetable,t-1}} \cdot (1 + 0.05)$$

6. 食用菌价格下降率为 1% ~ 5%。

$$\text{Price}_{\text{fungi,t}} = \text{Price}_{\text{fungi,t-1}} \cdot (1 - \text{fungi\_price\_decrease\_rate}_t)$$

7. 羊肚菌价格下降率为 5%。

$$\text{Price}_{\text{morchella,t}} = \text{Price}_{\text{morchella,t-1}} \cdot (1 - 0.05)$$

5.2.2 问题二模型的求解

将预处理数据按相关比例加权后带入问题一中的数学模型，约束不变，通过 Python 模拟计算得到最优种植农作物方案，编程代码详见附件 2。详细种植方案见附件表格，部份例图如下图，此处不再赘述。在问题二的背景下通过线性规划模型模拟得到的最优种植策略下的 7 年总利润为 58849.86488947 元。

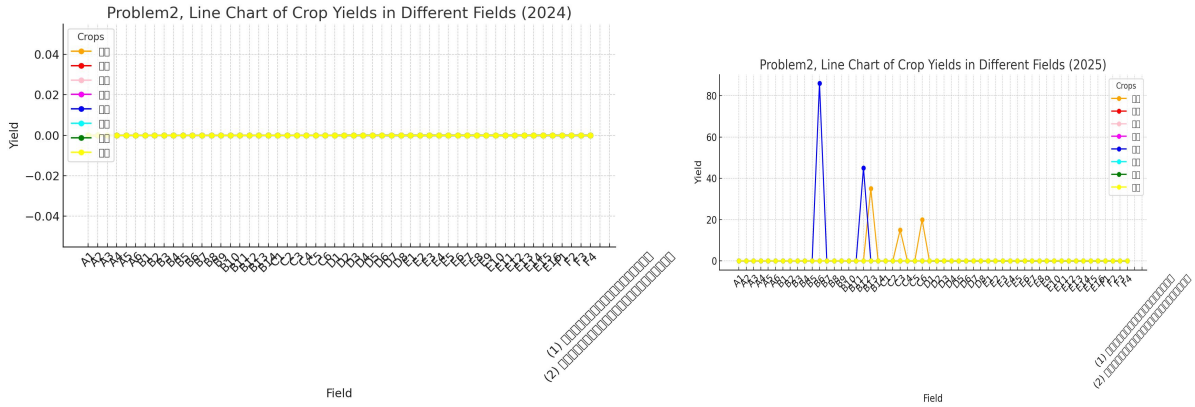

图 3 问题二的部分作物的分配方案例图

5.3 问题三模型的建立与求解

5.3.1 问题二模型的建立与求解

我们需要解决的问题是在问题二模型的基础上加上三条约束来体现作物之间的互补性与替代性并且利用问题二中算出的数据，对于预期销售量与销售价格、种植成本之间存在的一定的相关性进行探索。

(a) 约束条件

1. 黄豆，黑豆，红豆，绿豆有替代性，设置让这四种作物每年总产量保持不变

$$\sum_{t=1}^{2} X_{i,j,s,t} = \sum_{t=1}^{2} X_{i,j,2023,t}$$

其中 i 为任意一块土地，j∈{1,2,7}，s∈{2024,2025,2026,2027,2028,2029,2030}

2. 小麦、玉米、谷子、高粱替代性较高，设置让他们每年的总产量保持不变

$$\sum_{t=1}^{2} X_{i,j,s,t} = \sum_{t=1}^{2} X_{i,j,2023,t}$$

其中其中 i 为任意一块土地，j∈{6,7,8,9}，s∈{2024,2025,2026,2027,2028,2029,2030}

3. 菠菜和西红柿互补性较明显，使这两种植物的销售变化率相等

$$\text{crop21\_sales\_variation} = \text{crop23\_sales\_variation}$$

(b) 探索相关性

1. 数据提取与处理：首先，从 Excel 文件中提取了关于作物种植、产量、销售价格等信息，并在模型中设置了目标变量和约束条件，确保不同土地和作物的种植面积、成本等数据的合理性。

2. 模型求解：在对目标函数（利润最大化）进行求解后，你通过 varValue 获取了模型中决策变量的具体值，即每块土地在不同年份、季度种植的作物面积。

3. 数据填充：根据求解后的结果，计算并填充了每年的作物名称、作物类型、销售价格、种植成本和预期销售量的数据列表。这些数据构成了每年的 DataFrame，并在不同年份的数据表中分别存储。

4. 合并数据：你将所有年份的数据表合并为一个大的 DataFrame，以便进行整体分析。

5. 计算相关性：使用 corr() 函数计算了合并数据中“销售价格”、“种植成本”和“预期销售量”这三个变量之间的相关性矩阵。这一步通过计算各列数据之间的线性相关系数（通常为 Pearson 相关系数）来衡量变量之间的相关性。Person 相关系数的求解公式为：

$$r = \frac{\sum (X_i - X)(Y_i - Y)}{\sqrt{\sum (X_i - X)^2 \sum (Y_i - Y)^2}}$$

其中X_i，Y_i是两个变量的数据值，X 和 Y 是均值

6. 输出结果：最后，将相关性矩阵打印输出，显示了销售价格、种植成本和预期销售量三者之间的相关关系。

六、模型总结

6.1 模型优点

1. **综合性分析**：模型通过将不同类型的耕地、农作物品种和销售情景进行统一优化，综合考虑了产量、销售价格、耕作成本、地块资源等多方面因素，确保对问题的全面解决。

2. **数据处理**：在建立模型前，数据进行了充分的预处理，包括对不同类型耕地的分类、作物生长规律的约束等，确保了输入数据的准确性和合理性。

3. **模型简洁高效**：采用线性规划模型进行优化，结构简单，求解速度快，适合大规模问题求解，且能够提供直观的最优解。
4. **条件约束明确**：针对问题中的特定约束（如重茬限制、豆类作物轮作要求、不同地块种植限制等），通过线性约束条件进行精确描述，保证模型结果的可操作性。
5. **易于扩展**：线性规划模型具有较强的通用性，能够根据实际情况灵活调整约束条件和目标函数，使模型能够适应不同的农作物种植方案设计需求。
6. **直观的经济效益计算**：模型以经济效益最大化为目标，明确量化了种植收益，并能清楚反映不同种植方案下的经济影响，便于决策者直接使用。

6.2 模型缺点

1. **忽略了时间动态变化**：模型假设销售量、价格和成本在 2024-2030 年保持不变，而未考虑到这些变量随时间的动态变化，可能会导致结果在实际操作中不够精准。
2. **缺乏灵敏度分析**：模型未对输入参数（如产量、价格、成本等）进行灵敏度分析，无法衡量模型对这些参数变化的鲁棒性，降低了模型对现实复杂情况的适应能力。
3. **未考虑风险因素**：模型未对市场波动、气候变化等不确定因素进行建模，这可能使得实际种植中风险难以控制，种植方案的稳健性有待提升。

6.3 模型推广

1. **动态优化引入**：未来可以考虑引入时间维度，通过构建动态规划模型，结合农作物生长周期和市场价格的波动规律，优化长期种植策略，使模型能够适应动态变化的市场条件。
2. **风险管理扩展**：为应对市场波动和气候不确定性，可以在模型中加入风险约束或鲁棒优化，设计出对不同风险场景更加稳健的种植方案，增强方案在不确定环境下的适应性。
3. **多目标优化**：除了经济效益，还可以考虑环境保护、可持续发展等目标，构建多目标优化模型，通过平衡经济收益与其他社会、环境因素，制定更加全面的种植策略。
4. **非线性模型探索**：由于农作物之间存在互补性或替代性，可以采用非线性规划或混合整数规划模型，更加准确地描述作物之间复杂的相互作用，提高种植方案的科学性。
5. **灵敏度分析的引入**：未来应加入灵敏度分析，对模型中的关键参数（如种植面积、成本、市场价格等）进行波动分析，评估模型对参数变化的响应程度，增强模型的鲁棒性与可靠性。

七、参考文献

[1] https://www.johnnyseeds.com/growers-library/vegetables/beans/, 2024.
[2] Chunyan Liu. *Legume-based rotation enhances subsequent wheat yield and maintains soil carbon storage.* AGRONOMY FOR SUSTAINABLE DEVELOPMENT. Volume 43, Issue5, Pages -, 2023.
[3] 贺晓松，广义线性模型的理论与应用实例。
[4] Study Winter, 【数学建模】线性规划模型基本原理与案例分享, https://blog.csdn.net/Zhouzi_heng/article/details/113483537, 2024.
[5] Greenland, 线性规划模型的求解及应用, https://www.docin.com/p-988802409.html.

八、附录

附录 1

介绍：使用论文求解工具

论文写作软件 :Microsoft Word、LaTeX、WPS、EndNote、Grammarly
图表绘制软件:Excel
编程工具:VSCode
使用 Python 库:NumPy、Pandas、PULP

附录 2

介绍：支撑材料的文件列表

问题一：load_to_excel.py、第一题.py
问题二：load_to_excel.py、第二题.py
问题三：load_to_excel.py、第三题.py

附录 3

介绍：第一题.py

```
import pulp
import pandas as pd
import sys

#数据预处理
```

```
# 读取数据
attachment1_path = "C:/Users/Coco/Desktop/附件 1.xlsx"
attachment2_path = "C:/Users/Coco/Desktop/附件 2.xlsx"
attachment1_data = pd.read_excel(attachment1_path)
attachment2_data = pd.read_excel(attachment2_path)
all_sheets = pd.read_excel(attachment2_path, sheet_name=None)
sheets = pd.read_excel(attachment1_path, sheet_name='乡村的现
有耕地')
# 提取数据
crop_stats = all_sheets['2023 年统计的相关数据']
crop_production = all_sheets['2023 年的农作物种植情况']
```

```
#创建映射
plot_area_mapping = sheets[['地块名称', '地块面积/亩
']].drop_duplicates()
plot_yield_mapping = crop_stats[['作物编号','地块类型', '亩产量/
斤']].drop_duplicates()
plot_price_mapping = crop_stats[['作物编号','平均销售单价/(元/
斤)']].drop_duplicates()
plot_cost_mapping = crop_stats[['作物编号','种植成本/(元/
亩)']].drop_duplicates()
plot_type_mapping = sheets[['地块类型', '地块名称
']].drop_duplicates()
```

```
#初始化模型和变量
```

```
#创建模型
model = pulp.LpProblem("Maximize_Profit", pulp.LpMaximize)
# 创建目标函数中基础的变量：x[ijkt]：i 土地上作物编号为 j 的作物在 k 年第
t 季度的种植面积
variables = {}
for year in range(2024, 2031): # From 2024 to 2030
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1,42):
var_name = f"x_{plot}_{crop}_{season}_{year}"
variables[var_name] = pulp.LpVariable(var_name, lowBound=0,
cat='Continuous')
```

```
# 添加约束
```

```
# 约束 1：对于每块土地，每年每季度的种植在这块土地上的所有植物种植面积不
超过土地面积
for year in range(2024, 2031):
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_area = plot_area_mapping[plot_area_mapping['地块名称
']==plot]['地块面积/亩'].values[0]
```

```
total_area =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"] for
crop in range(1,42))
model += (total_area <= plot_area,
f"MAX_{plot}_{season}_{year}")
```

```
# 约束 2：对于每年每个季度每个植物在每块土地，如果种植，至少种植 0.1 亩
M = 86 # 定义一个大常数 M 表示地块最大面积
for year in range(2024, 2031): # 遍历年份
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
# 定义变量，表示某个作物在该地块和季度的种植面积
planting_area_var =
variables[f"x_{plot}_{crop}_{season}_{year}"]
# 定义二进制变量 y_crop 表示作物是否被种植
y_crop =
pulp.LpVariable(f"y_crop_{plot}_{crop}_{season}_{year}",
cat='Binary')
# 确保如果种植面积大于 0，则至少为 0.1
model += planting_area_var >= 0.1 * y_crop,
f"MinPlantingArea_{plot}_{crop}_{season}_{year}"
# 使用大 M 法确保当 y_crop 为 0 时，种植面积必须为 0
model += planting_area_var <= M * y_crop,
f"MaxPlantingArea_{plot}_{crop}_{season}_{year}"
```

```
# 约束 3：连续两个季度，在同一地块上不能种相同的植物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
if year > 2024:
# 定义二进制变量，分别表示第一季和前一年第二季种植该作物的情况
y_crop_first_season = pulp.LpVariable(f"y_{plot}_{crop}_第一
季_{year}", cat='Binary')
y_crop_second_season_last_year =
```

```
pulp.LpVariable(f"y_{plot}_{crop}_第二季_{year-1}",
cat='Binary')
y_crop_second_season = pulp.LpVariable(f"y_{plot}_{crop}_第二
季_{year}_crt", cat='Binary')
model += variables[f"x_{plot}_{crop}_第一季
_{year}"]<=M*y_crop_first_season
model += variables[f"x_{plot}_{crop}_第二季
_{year-1}"]<=M*y_crop_second_season_last_year
model += variables[f"x_{plot}_{crop}_第二季
_{year}"]<=M*y_crop_second_season
# 确保第一季种的作物与前一年第二季种的作物不相同：
# 如果作物在前一年第二季种植了（y_crop_second_season_last_year =
1），那么它不能在当前年的第一季种植（y_crop_first_season = 0）
model += y_crop_first_season + y_crop_second_season_last_year
<= 1, f"no_same_crop_{plot}_{crop}_{year}"
model += y_crop_first_season + y_crop_second_season <= 1,
f"no_same_crop_{plot}_{crop}_{year}_c"
```

```
# 约束 4：平旱地，梯田，山坡地，每年只能种一季（水稻以外的）粮食类作物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
if plot_type in ['平旱地', '梯田', '山坡地']:
# 计算第一季和第二季的总种植面积
total_area_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for
crop in range(1,16))
total_area_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"] for
crop in range(1,16))
non_crop_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一
季_{year}"] for crop in range(16,42))
non_crop_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二
季_{year}"] for crop in range(16,42))
# 定义二进制变量 y_first_season，控制是否种植第一季
y_first_season =
pulp.LpVariable(f"y_first_season_{plot}_{year}",
```

```
cat='Binary')
# 如果 y_first_season = 1，表示在第一季种植，第二季面积必须为 0
model += total_area_first_season <= M* y_first_season,
f"first_season_area_{plot}_{year}"
# 如果 y_first_season = 0，表示在第二季种植，第一季面积必须为 0
model += total_area_second_season <= M* (1 - y_first_season),
f"second_season_area_{plot}_{year}"
```

```
# 约束 5: 一年内水浇地可以种一季水稻或者两季蔬菜(第一季是白菜萝卜以外的，
第二季必须是白菜萝卜)
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
if plot_type == '水浇地':
# 水稻种植面积（第一季和第二季）
rice_area_season_1 = variables[f"x_{plot}_16_第一季_{year}"]
rice_area_season_2 = variables[f"x_{plot}_16_第二季_{year}"]
# 蔬菜种植面积（第一季和第二季）
vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for
crop in range(17, 35))
cabbage_family_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(35, 38))
# 第一季和第二季的作物总面积
total_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一季
_{year}"] for crop in range(1, 42))
total_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二季
_{year}"] for crop in range(1, 42))
# 定义两个二进制变量来控制是否种植水稻或蔬菜
y_rice = pulp.LpVariable(f"y_rice_{plot}_{year}",
cat='Binary')
y_vegetables = pulp.LpVariable(f"y_vegetables_{plot}_{year}",
cat='Binary')
# 不能同时种植水稻和蔬菜
model += y_rice + y_vegetables <= 1,
f"no_rice_and_vegetables_{plot}_{year}"
```

```
# 如果 y_rice = 1，则只种植水稻
model += rice_area_season_1 + rice_area_season_2 <= M * y_rice,
f"rice_planting_{plot}_{year}"
# 如果 y_vegetables = 1，则只种植蔬菜
model += vegetables_first_season +
cabbage_family_second_season <= M * y_vegetables,
f"vegetable_planting_{plot}_{year}"
# 如果种水稻只能种单季
total_plot_area = plot_area_mapping[plot_area_mapping['地块名
称'] == plot]['地块面积/亩'].values[0]
model += rice_area_season_1 + rice_area_season_2 <=
total_plot_area, f"rice_area_limit_{plot}_{year}"
```

```
# 约束 6：普通大棚第一季种蔬菜（白菜萝卜除外），第二季种可食用菌，可食用
菌只能种在普通大棚（在计算 profit 时已经有相应约束，但为了让代码约束与论
文公式相对应，再写一次）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '普通大棚':
non_vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"]for
crop in range(1,17)+variables[f"x_{plot}_{crop}_第一季
_{year}"]for crop in range(35,42))
model += (non_vegetables_first_season ==
0,f"only_vegetables_1_{plot}_{year}")
non_fungi_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(1,38))
model += (non_fungi_second_season ==
0,f"only_fungi_2_{plot}_{year}")
```

```
# 约束 7：智慧大棚每年种两季（除萝卜白菜之外的）蔬菜（在计算 profit 时已
经有相应约束，但为了让代码约束与论文公式相对应，再写一次）
for year in range(2024, 2031):
```

```
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '智慧大棚':
for season in ['第一季', '第二季']:
# Sum of all suitable vegetables in both seasons
non_vegetables_each_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"]for
crop in
range(1,17)+variables[f"x_{plot}_{crop}_{season}_{year}"]
for crop in range(35,42))
model += (non_vegetables_each_season == 0,
f"No_non_vegetables_{season}_{plot}_{year}")
```

```
# 约束 8: 每个地块每三年内一定要让所有土地上种过一次豆子
bean_crops = [1,2,3,4,5,17,18,19] # 豆类编号
for year in range(2025,2030):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
total_bean_planted =
pulp.lpSum([variables[f"x_{plot}_{crop}_{season}_{y}"]
for y in range(year-1, year + 2)
for crop in bean_crops]
for season in ['第一季', '第二季'])
model += total_bean_planted >=
plot_area_mapping[plot_area_mapping['地块名称']==plot]['地块面
积/亩'].values[0],
f"Plant_Beans_Once_Every_Three_Years_{plot}_{season}_{year}"
```

```
#定义目标函数
```

```
#初始化目标函数的值
total_profit = 0
```

```
# 从附件 2 的“2023 年农作物的种植情况”中获取作物编号为 1 的所有行
for i in range(1,42):
filtered_crop_data = crop_production[crop_production['作物编号
```

```
'] == i]
# 初始化函数中的变量
total_production_2023 = 0
total_production = 0
total_area = 0
cost = 0
price = plot_price_mapping[plot_price_mapping['作物编号'] ==
i]['平均销售单价/(元/斤)'].values[0]
# 计算该植物在每个地块的 2023 年实际产量之和（即该植物 2023 年总产量）
for index, row in filtered_crop_data.iterrows():
plot_type = row['种植地块']
plot_area = row['种植面积/亩']
plot_n = plot_type_mapping[plot_type_mapping['地块名称
']==plot_type]['地块类型'].values[0]
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作物编
号'] == i )&(plot_yield_mapping['地块类型'] == plot_n)]['亩产量
/斤'].values[0]
total_production_2023+= plot_area * yield_per_mu
# 计算基于变量 x 的该植物在 24 年到 30 年的总实际产量
for year in range(2024,2031):
for season in ['第一季', '第二季']:
for index, plot_row in plot_type_mapping.iterrows():
plot_name = plot_row['地块名称']
plot_type = plot_row['地块类型']
# 在附件 2 中查找该地块类型对应的亩产量,若没有说明该作物无法在该地块生长,
亩产量记为 0，因此在优化时，不会让这类植物种在该地块上，因为只会亏不会赚
if len(plot_yield_mapping[(plot_yield_mapping['作物编号']==i)
& (plot_yield_mapping['地块类型'] == plot_type)]['亩产量/斤
'].values) == 0:
yield_per_mu = 0
else:
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作物编
号']==i) & (plot_yield_mapping['地块类型'] == plot_type)]['亩产
量/斤'].values[0]
# 获取变量 x
var_name = f"x_{plot_name}_{i}_{season}_{year}"
planting_area_var = variables.get(var_name, None)
total_area += planting_area_var
```

```
cost +=
planting_area_var*plot_cost_mapping[plot_cost_mapping['作物编
号']==i]['种植成本/(元/亩)'].values[0]
# 计算每年每季该地块上的实际产量
actual_production = planting_area_var * yield_per_mu
# 将实际产量累加到总产量
total_production += actual_production
```

```
# 定义超出部分的产量（超过 2023 年产量的部分）
excess_production =
pulp.LpVariable(f"excess_production_{i}_{year}",
lowBound=0,upBound = total_production)
# 添加约束，确保 excess_production 表示超出的部分
model += (excess_production == total_production -
total_production_2023)
model += (excess_production >= 0)
```

```
# 利润只计算不超过 2023 年产量的部分，对于第一问为(total_production -
excess_production) * price - cost，以下是第二问
profit = (total_production - excess_production*0.5) * price -
cost
total_profit += profit
```

```
# 加入目标函数
model += total_profit
# 使用 CBC 求解器并设置时间限制
solver = pulp.PULP_CBC_CMD(msg=1, timeLimit=700)
model.solve(solver)
```

```
# 检查求解器状态
solution_status = pulp.LpStatus[model.status]
objective_value = pulp.value(model.objective)
```

```
# 将结果写入 txt 方便导入 xlsx 中
with open('C:/Users/Coco/Desktop/解 1.txt', 'w',
encoding='utf-8') as file:
sys.stdout = file
for variable in model.variables():
if variable.varValue != 0 and 'x_' in variable.name:
```

```
print(f"变量名: {variable.name}, 变量值: {variable.varValue}")
```

附录 4

介绍：第二题.py

```
import pulp
import pandas as pd
import numpy as np
import sys

#数据预处理
```

```
# 读取数据
attachment1_path = "C:/Users/Coco/Desktop/附件 1.xlsx"
attachment2_path = "C:/Users/Coco/Desktop/附件 2.xlsx"
attachment1_data = pd.read_excel(attachment1_path)
attachment2_data = pd.read_excel(attachment2_path)
all_sheets = pd.read_excel(attachment2_path, sheet_name=None)
sheets = pd.read_excel(attachment1_path, sheet_name='乡村的现
有耕地')
# 提取数据
crop_stats = all_sheets['2023 年统计的相关数据']
crop_production = all_sheets['2023 年的农作物种植情况']
#创建映射
plot_area_mapping = sheets[['地块名称', '地块面积/亩
']].drop_duplicates()
plot_yield_mapping = crop_stats[['作物编号','地块类型', '亩产量/
斤']].drop_duplicates()
plot_price_mapping = crop_stats[['作物编号','平均销售单价/(元/
斤)']].drop_duplicates()
plot_cost_mapping = crop_stats[['作物编号','种植成本/(元/
亩)']].drop_duplicates()
plot_type_mapping = sheets[['地块类型', '地块名称
']].drop_duplicates()
```

```
#初始化模型和变量
```

```
#创建模型
model = pulp.LpProblem("Maximize_Profit", pulp.LpMaximize)
# 创建目标函数中基础的变量：x[ijkt]：i 土地上作物编号为 j 的作物在 k 年第
t 季度的种植面积
variables = {}
for year in range(2024, 2031):
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1,42):
var_name = f"x_{plot}_{crop}_{season}_{year}"
variables[var_name] = pulp.LpVariable(var_name, lowBound=0,
cat='Continuous')
```

```
# 添加约束
```

```
# 约束 1：对于每块土地，每年每季度的种植在这块土地上的所有植物种植面积不
超过土地面积
for year in range(2024, 2031):
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_area = plot_area_mapping[plot_area_mapping['地块名称
']==plot]['地块面积/亩'].values[0]
total_area =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"] for
crop in range(1,42))
model += (total_area <= plot_area,
f"MAX_{plot}_{season}_{year}")
```

```
# 约束 2：对于每年每个季度每个植物在每块土地，如果种植，至少种植 0.1 亩
M = 86 # 定义一个大常数 M 表示地块最大面积
for year in range(2024, 2031): # 遍历年份
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
```

```
# 定义变量，表示某个作物在该地块和季度的种植面积
planting_area_var =
variables[f"x_{plot}_{crop}_{season}_{year}"]
# 定义二进制变量 y_crop 表示作物是否被种植
y_crop =
pulp.LpVariable(f"y_crop_{plot}_{crop}_{season}_{year}",
cat='Binary')
# 确保如果种植面积大于 0，则至少为 0.1
model += planting_area_var >= 0.1 * y_crop,
f"MinPlantingArea_{plot}_{crop}_{season}_{year}"
# 使用大 M 法确保当 y_crop 为 0 时，种植面积必须为 0
model += planting_area_var <= M * y_crop,
f"MaxPlantingArea_{plot}_{crop}_{season}_{year}"
```

```
# 约束 3：连续两个季度，在同一地块上不能种相同的植物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
if year > 2024:
# 定义二进制变量，分别表示第一季和前一年第二季种植该作物的情况
y_crop_first_season = pulp.LpVariable(f"y_{plot}_{crop}_第一
季_{year}", cat='Binary')
y_crop_second_season_last_year =
pulp.LpVariable(f"y_{plot}_{crop}_第二季_{year-1}",
cat='Binary')
y_crop_second_season = pulp.LpVariable(f"y_{plot}_{crop}_第二
季_{year}_crt", cat='Binary')
model += variables[f"x_{plot}_{crop}_第一季
_{year}"]<=M*y_crop_first_season
model += variables[f"x_{plot}_{crop}_第二季
_{year-1}"]<=M*y_crop_second_season_last_year
model += variables[f"x_{plot}_{crop}_第二季
_{year}"]<=M*y_crop_second_season
# 确保第一季种的作物与前一年第二季种的作物不相同：
# 如果作物在前一年第二季种植了（y_crop_second_season_last_year =
1），那么它不能在当前年的第一季种植（y_crop_first_season = 0）
model += y_crop_first_season + y_crop_second_season_last_year
```

```
<= 1, f"no_same_crop_{plot}_{crop}_{year}"
model += y_crop_first_season + y_crop_second_season <= 1,
f"no_same_crop_{plot}_{crop}_{year}_c"
```

```
# 约束 4: 平旱地，梯田，山坡地，每年只能种一季（水稻以外的）粮食类作物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
if plot_type in ['平旱地', '梯田', '山坡地']:
# 计算第一季和第二季的总种植面积
total_area_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for
crop in range(1,16))
total_area_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"] for
crop in range(1,16))
non_crop_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一
季_{year}"] for crop in range(16,42))
non_crop_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二
季_{year}"] for crop in range(16,42))
# 定义二进制变量 y_first_season，控制是否种植第一季
y_first_season =
pulp.LpVariable(f"y_first_season_{plot}_{year}",
cat='Binary')
# 如果 y_first_season = 1，表示在第一季种植，第二季面积必须为 0
model += total_area_first_season <= M* y_first_season,
f"first_season_area_{plot}_{year}"
# 如果 y_first_season = 0，表示在第二季种植，第一季面积必须为 0
model += total_area_second_season <= M* (1 - y_first_season),
f"second_season_area_{plot}_{year}"
```

```
# 约束 5: 一年内水浇地可以种一季水稻或者两季蔬菜(第一季是白菜萝卜以外的，
第二季必须是白菜萝卜)
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
```

```
if plot_type == '水浇地':
# 水稻种植面积（第一季和第二季）
rice_area_season_1 = variables[f"x_{plot}_16_第一季_{year}"]
rice_area_season_2 = variables[f"x_{plot}_16_第二季_{year}"]
# 蔬菜种植面积（第一季和第二季）
vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for
crop in range(17, 35))
cabbage_family_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(35, 38))
# 第一季和第二季的作物总面积
total_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一季
_{year}"] for crop in range(1, 42))
total_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二季
_{year}"] for crop in range(1, 42))
# 定义两个二进制变量来控制是否种植水稻或蔬菜
y_rice = pulp.LpVariable(f"y_rice_{plot}_{year}",
cat='Binary')
y_vegetables = pulp.LpVariable(f"y_vegetables_{plot}_{year}",
cat='Binary')
# 不能同时种植水稻和蔬菜
model += y_rice + y_vegetables <= 1,
f"no_rice_and_vegetables_{plot}_{year}"
# 如果 y_rice = 1，则只种植水稻
model += rice_area_season_1 + rice_area_season_2 <= M * y_rice,
f"rice_planting_{plot}_{year}"
# 如果 y_vegetables = 1，则只种植蔬菜
model += vegetables_first_season +
cabbage_family_second_season <= M * y_vegetables,
f"vegetable_planting_{plot}_{year}"
# 如果种水稻只能种单季
total_plot_area = plot_area_mapping[plot_area_mapping['地块名
称'] == plot]['地块面积/亩'].values[0]
model += rice_area_season_1 + rice_area_season_2 <=
total_plot_area, f"rice_area_limit_{plot}_{year}"


# 约束 6：普通大棚第一季种蔬菜（白菜萝卜除外），第二季种可食用菌，可食用
```

```
菌只能种在普通大棚（在计算 profit 时已经有相应约束，但为了让代码约束与论文公式相对应，再写一次）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '普通大棚':
non_vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"]for
crop in range(1,17)+variables[f"x_{plot}_{crop}_第一季
_{year}"]for crop in range(35,42))
model += (non_vegetables_first_season ==
0,f"only_vegetables_1_{plot}_{year}")
non_fungi_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(1,38))
model += (non_fungi_second_season ==
0,f"only_fungi_2_{plot}_{year}")
```

```
# 约束 7：智慧大棚每年种两季（除萝卜白菜之外的）蔬菜（在计算 profit 时已经有相应约束，但为了让代码约束与论文公式相对应，再写一次）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '智慧大棚':
for season in ['第一季', '第二季']:
non_vegetables_each_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"]for
crop in
range(1,17)+variables[f"x_{plot}_{crop}_{season}_{year}"]
for crop in range(35,42))
model += (non_vegetables_each_season == 0,
f"No_non_vegetables_{season}_{plot}_{year}")
```

```
# 约束 8: 每个地块每三年内一定要让所有土地上种过一次豆子
bean_crops = [1,2,3,4,5,17,18,19] # 豆类编号
for year in range(2025,2030):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
total_bean_planted =
pulp.lpSum([variables[f"x_{plot}_{crop}_{season}_{y}"]
for y in range(year-1, year + 2)
for crop in bean_crops]
for season in ['第一季', '第二季'])
model += total_bean_planted >=
plot_area_mapping[plot_area_mapping['地块名称']==plot]['地块面
积/亩'].values[0],
f"Plant_Beans_Once_Every_Three_Years_{plot}_{season}_{year}"
```

```
#定义目标函数
```

```
# 定义增长率下降率等
wheat_corn_growth_rate = np.random.uniform(0.05, 0.10,
size=(2031-2024)) # 小麦和玉米年增长率
other_crop_sales_variation = np.random.uniform(-0.05, 0.05,
size=(2031-2024)) # 其他作物的年销售变化
yield_variation = np.random.uniform(-0.10, 0.10,
size=(2031-2024)) # 每年亩产量变化
cost_growth_rate = 0.05 # 种植成本年增长率
vegetable_price_growth_rate = 0.05 # 蔬菜价格年增长率
fungi_price_decrease_rate = np.random.uniform(0.01, 0.05,
size=(2031-2024)) # 食用菌价格下降率
morchella_price_decrease_rate = 0.05 # 羊肚菌价格下降率
```

```
total_profit = 0
```

```
# 读取附件表格信息，以字典的形式得到 2023 年的作物数据（总产量，每亩成本，
销售价格）以及 2024-2030 年的每年每种作物的亩产量
crop_data_2023 = {}
crop_yield_per_year = {}
for i in range(1,42):
#对于每种作物，计算 2023 年的总产量，每亩成本，销售价格
```

```
filtered_crop_data = crop_production[crop_production['作物编号
'] == i]
crop_yield_per_year[i] = {}
total_production = 0
yield_per_mu = 0
cost_per_mu = 0
price = plot_price_mapping[plot_price_mapping['作物编号'] ==
i]['平均销售单价/(元/斤)'].values[0]
for index,row in crop_stats.iterrows():
plot_type = row['地块类型']
crop_yield_per_year[i][plot_type]={}
for year in range(2023, 2031):
crop_yield_per_year[i][plot_type][year] = 0
for index, row in filtered_crop_data.iterrows():
plot_name = row['种植地块']
plot_area = row['种植面积/亩']
#根据地块类型找到对应的亩产量
if len(plot_yield_mapping[(plot_yield_mapping['作物编号']==i)
& (plot_yield_mapping['地块类型'] == plot_type)]['亩产量/斤
'].values) == 0:
yield_per_mu = 0
else:
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作物编
号'] == i )&(plot_yield_mapping['地块类型'] == plot_type)]['亩
产量/斤'].values[0]
crop_yield_per_year[i][plot_type][2023] = yield_per_mu
```

```
for year in range(2024,2031):
crop_yield_per_year[i][plot_type][year] =
crop_yield_per_year[i][plot_type][year-1] * (1 +
yield_variation[year-2024])
# 计算 2023 年总产量
total_production += plot_area * yield_per_mu
cost_per_mu = plot_cost_mapping[plot_cost_mapping['作物编号
']==i]['种植成本/(元/亩)'].values[0]
crop_data_2023[i] = {'total_production': total_production,
'cost_per_mu': cost_per_mu, 'price': price}
```

```
#开始计算 2024-2030 年的总成本和总利润
```

```
for year in range(2024, 2031):
year_cost = 0
for i in range(1,42):
total_area = 0
expected_sales = 0
this_year_cost_per_mu = 0
this_year_price = 0
this_year_production = 0
excess_production = 0
if year == 2024:
last_year_production = crop_data_2023[i]['total_production']
last_year_cost_per_mu = crop_data_2023[i]['cost_per_mu']
last_year_price = crop_data_2023[i]['price']
else:
last_year_production = 0
last_year_cost_per_mu = 0
last_year_price = 0
#计算预期销售量
if i in range(6,8):
expected_sales = last_year_production * (1 +
wheat_corn_growth_rate[year - 2024])
else:
expected_sales = last_year_production * (1 +
other_crop_sales_variation[year - 2024])
#计算今年每亩的种植成本
this_year_cost_per_mu = last_year_cost_per_mu * (1 +
cost_growth_rate)
#计算今年的销售价格
if i in range(1,17):
this_year_price = last_year_price
elif i in range(17,38):
this_year_price = last_year_price * (1 +
vegetable_price_growth_rate)
elif i in range(38,41):
this_year_price = last_year_price * (1 -
fungi_price_decrease_rate[year - 2024])
elif i == 41:
this_year_price = last_year_price * (1 -
```

```
morchella_price_decrease_rate)
#计算今年某种植物的总产量
for season in ['第一季','第二季']:
for index, plot_row in plot_type_mapping.iterrows():
plot_name = plot_row['地块名称']
plot_type = plot_row['地块类型']
#获取实际种植面积
var_name = f"x_{plot_name}_{i}_{season}_{year}"
planting_area_var = variables.get(var_name, None)
total_area += planting_area_var
this_year_production += planting_area_var *
crop_yield_per_year[i][plot_type][year]
#计算今年的总成本
year_cost += total_area * this_year_cost_per_mu
#计算今年某种植物的利润
excess_production = this_year_production - expected_sales
total_profit += ((this_year_production - excess_production) *
this_year_price + excess_production * 0.5 * last_year_price -
year_cost)
#更新上一年的产量，成本，价格
last_year_production = this_year_production
last_year_cost_per_mu = this_year_cost_per_mu
last_year_price = this_year_price
```

```
# 加入目标函数
model += total_profit
#求解
model.solve()
```

```
# 检查求解器状态
solution_status = pulp.LpStatus[model.status]
objective_value = pulp.value(model.objective)
```

```
# 将结果写入 txt 方便导入 xlsx 中
with open('C:/Users/Coco/Desktop/解 2.txt', 'w',
encoding='utf-8') as file:
sys.stdout = file
for variable in model.variables():
if variable.varValue != 0 and 'x_' in variable.name:
```

```
print(f"变量名: {variable.name}, 变量值: {variable.varValue}")
```

附录 5

介绍：第三题.py

```
import pulp
import pandas as pd
import numpy as np
import sys

#数据预处理
```

```
# 读取数据
attachment1_path = "C:/Users/Coco/Desktop/附件 1.xlsx"
attachment2_path = "C:/Users/Coco/Desktop/附件 2.xlsx"
attachment1_data = pd.read_excel(attachment1_path)
attachment2_data = pd.read_excel(attachment2_path)
all_sheets = pd.read_excel(attachment2_path,
sheet_name=None)
sheets = pd.read_excel(attachment1_path, sheet_name='乡村的
现有耕地')
sheets1_1 = pd.read_excel(attachment1_path, sheet_name = '
乡村种植的农作物')
# 提取数据
crop_stats = all_sheets['2023 年统计的相关数据']
crop_production = all_sheets['2023 年的农作物种植情况']
#创建映射
plot_area_mapping = sheets[['地块名称', '地块面积/亩
']].drop_duplicates()
plot_yield_mapping = crop_stats[['作物编号','地块类型', '亩产
量/斤']].drop_duplicates()
plot_price_mapping = crop_stats[['作物编号','平均销售单价/(元/
斤)']].drop_duplicates()
plot_cost_mapping = crop_stats[['作物编号','种植成本/(元/
亩)']].drop_duplicates()
plot_type_mapping = sheets[['地块类型', '地块名称
']].drop_duplicates()
plot_crop_mapping = sheets1_1[['作物编号', '作物名称', '作物类
型']].drop_duplicates()
```

```
#初始化模型和变量
```

```
#创建模型
model = pulp.LpProblem("Maximize_Profit", pulp.LpMaximize)
# 创建目标函数中基础的变量：x[ijkt]：i 土地上作物编号为 j 的作物在 k 年第 t 季度的种植面积
variables = {}
for year in range(2024, 2031): # From 2024 to 2030
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1,42):
var_name = f"x_{plot}_{crop}_{season}_{year}"
variables[var_name] = pulp.LpVariable(var_name, lowBound=0, cat='Continuous')
```

```
# 添加约束
```

```
# 约束 1：对于每块土地，每年每季度的种植在这块土地上的所有植物种植面积不超过土地面积
for year in range(2024, 2031):
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_area = plot_area_mapping[plot_area_mapping['地块名称']==plot]['地块面积/亩'].values[0]
total_area = pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"]
for crop in range(1,42))
model += (total_area <= plot_area, f"MAX_{plot}_{season}_{year}")
```

```
# 约束 2：对于每年每个季度每个植物在每块土地，如果种植，至少种植 0.1 亩
M = 86 # 定义一个大常数 M 表示地块最大面积
for year in range(2024, 2031): # 遍历年份
```

```
for season in ['第一季', '第二季']:
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
# 定义变量，表示某个作物在该地块和季度的种植面积
planting_area_var =
variables[f"x_{plot}_{crop}_{season}_{year}"]
# 定义二进制变量 y_crop 表示作物是否被种植
y_crop =
pulp.LpVariable(f"y_crop_{plot}_{crop}_{season}_{year}",
cat='Binary')
# 确保如果种植面积大于 0，则至少为 0.1
model += planting_area_var >= 0.1 * y_crop,
f"MinPlantingArea_{plot}_{crop}_{season}_{year}"
# 使用大 M 法确保当 y_crop 为 0 时，种植面积必须为 0
model += planting_area_var <= M * y_crop,
f"MaxPlantingArea_{plot}_{crop}_{season}_{year}"
```

```
# 约束 3：连续两个季度，在同一地块上不能种相同的植物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
for crop in range(1, 42):
if year > 2024:
# 定义二进制变量，分别表示第一季和前一年第二季种植该作物的情况
y_crop_first_season = pulp.LpVariable(f"y_{plot}_{crop}_第
一季_{year}", cat='Binary')
y_crop_second_season_last_year =
pulp.LpVariable(f"y_{plot}_{crop}_第二季_{year-1}",
cat='Binary')
y_crop_second_season = pulp.LpVariable(f"y_{plot}_{crop}_
第二季_{year}_crt", cat='Binary')
model += variables[f"x_{plot}_{crop}_第一季
_{year}"]<=M*y_crop_first_season
model += variables[f"x_{plot}_{crop}_第二季
_{year-1}"]<=M*y_crop_second_season_last_year
model += variables[f"x_{plot}_{crop}_第二季
_{year}"]<=M*y_crop_second_season
```

```
# 确保第一季种的作物与前一年第二季种的作物不相同：
# 如果作物在前一年第二季种植了（y_crop_second_season_last_year = 1），那么它不能在当前年的第一季种植（y_crop_first_season = 0）
model += y_crop_first_season + y_crop_second_season_last_year <= 1, f"no_same_crop_{plot}_{crop}_{year}"
model += y_crop_first_season + y_crop_second_season <= 1, f"no_same_crop_{plot}_{crop}_{year}_c"
```

```
# 约束 4：平旱地，梯田，山坡地，每年只能种一季（水稻以外的）粮食类作物
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
if plot_type in ['平旱地', '梯田', '山坡地']:
# 计算第一季和第二季的总种植面积
total_area_first_season = pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for crop in range(1,16))
total_area_second_season = pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"] for crop in range(1,16))
non_crop_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for crop in range(16,42))
non_crop_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"] for crop in range(16,42))
# 定义二进制变量 y_first_season，控制是否种植第一季
y_first_season = pulp.LpVariable(f"y_first_season_{plot}_{year}", cat='Binary')
# 如果 y_first_season = 1，表示在第一季种植，第二季面积必须为 0
model += total_area_first_season <= M* y_first_season, f"first_season_area_{plot}_{year}"
# 如果 y_first_season = 0，表示在第二季种植，第一季面积必须为 0
model += total_area_second_season <= M* (1 - y_first_season), f"second_season_area_{plot}_{year}"
```

```
# 约束 5：一年内水浇地可以种一季水稻或者两季蔬菜（第一季是白菜萝卜以外
```

```
的，第二季必须是白菜萝卜）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
if plot_type == '水浇地':
# 水稻种植面积（第一季和第二季）
rice_area_season_1 = variables[f"x_{plot}_16_第一季
_{year}"]
rice_area_season_2 = variables[f"x_{plot}_16_第二季
_{year}"]
# 蔬菜种植面积（第一季和第二季）
vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"] for
crop in range(17, 35))
cabbage_family_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(35, 38))
# 第一季和第二季的作物总面积
total_area_1 = pulp.lpSum(variables[f"x_{plot}_{crop}_第一
季_{year}"] for crop in range(1, 42))
total_area_2 = pulp.lpSum(variables[f"x_{plot}_{crop}_第二
季_{year}"] for crop in range(1, 42))
# 定义两个二进制变量来控制是否种植水稻或蔬菜
y_rice = pulp.LpVariable(f"y_rice_{plot}_{year}",
cat='Binary')
y_vegetables =
pulp.LpVariable(f"y_vegetables_{plot}_{year}",
cat='Binary')
# 不能同时种植水稻和蔬菜
model += y_rice + y_vegetables <= 1,
f"no_rice_and_vegetables_{plot}_{year}"
# 如果 y_rice = 1，则只种植水稻
model += rice_area_season_1 + rice_area_season_2 <= M *
y_rice, f"rice_planting_{plot}_{year}"
# 如果 y_vegetables = 1，则只种植蔬菜
model += vegetables_first_season +
cabbage_family_second_season <= M * y_vegetables,
```

```
f"vegetable_planting_{plot}_{year}"
# 如果种水稻只能种单季
total_plot_area = plot_area_mapping[plot_area_mapping['地
块名称'] == plot]['地块面积/亩'].values[0]
model += rice_area_season_1 + rice_area_season_2 <=
total_plot_area, f"rice_area_limit_{plot}_{year}"
```

```
# 约束 6: 普通大棚第一季种蔬菜（白菜萝卜除外），第二季种可食用菌，可食
用菌只能种在普通大棚（在计算 profit 时已经有相应约束，但为了让代码约
束与论文公式相对应，再写一次）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '普通大棚':
non_vegetables_first_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第一季_{year}"]for
crop in range(1,17)+variables[f"x_{plot}_{crop}_第一季
_{year}"]for crop in range(35,42))
model += (non_vegetables_first_season ==
0,f"only_vegetables_1_{plot}_{year}")
non_fungi_second_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_第二季_{year}"]for
crop in range(1,38))
model += (non_fungi_second_season ==
0,f"only_fungi_2_{plot}_{year}")
```

```
# 约束 7: 智慧大棚每年种两季（除萝卜白菜之外的）蔬菜（在计算 profit
时已经有相应约束，但为了让代码约束与论文公式相对应，再写一次）
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
plot_type = row['地块类型']
plot_area = row['地块面积/亩']
if plot_type == '智慧大棚':
for season in ['第一季', '第二季']:
```

```
# Sum of all suitable vegetables in both seasons
non_vegetables_each_season =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"]f
or crop in
range(1,17)+variables[f"x_{plot}_{crop}_{season}_{year}"]
for crop in range(35,42))
model += (non_vegetables_each_season == 0,
f"No_non_vegetables_{season}_{plot}_{year}")
```

```
# 约束 8: 每个地块每三年内一定要让所有土地上种过一次豆子
bean_crops = [1,2,3,4,5,17,18,19] # 豆类编号
for year in range(2025,2030):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
total_bean_planted =
pulp.lpSum([variables[f"x_{plot}_{crop}_{season}_{y}"]
for y in range(year-1, year + 2)
for crop in bean_crops]
for season in ['第一季', '第二季'])
model += total_bean_planted >=
plot_area_mapping[plot_area_mapping['地块名称']==plot]['地
块面积/亩'].values[0],
f"Plant_Beans_Once_Every_Three_Years_{plot}_{season}_{yea
r}"
```

```
# 约束 9: 黄豆，黑豆和玉米之间的替代性
total_2023_beans = 0
for i in [1,2,7]:
filtered_crop_data = crop_production[crop_production['作物
编号'] == i]
for index, row in filtered_crop_data.iterrows():
plot_type = row['种植地块']
plot_area = row['种植面积/亩']
plot_n = plot_type_mapping[plot_type_mapping['地块名称
']==plot_type]['地块类型'].values[0]
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作
物编号'] == i )&(plot_yield_mapping['地块类型'] == plot_n)]['
亩产量/斤'].values[0]
```

```
total_2023_beans += plot_area * yield_per_mu
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
# 定义三个作物的种植面积变量
soybean_area =
pulp.lpSum(variables[f"x_{plot}_1_{season}_{year}"] for
season in ['第一季','第二季'])
blackbean_area =
pulp.lpSum(variables[f"x_{plot}_2_{season}_{year}"]for
season in ['第一季','第二季'])
corn_area =
pulp.lpSum(variables[f"x_{plot}_7_{season}_{year}"]for
season in ['第一季','第二季'])
model += soybean_area+blackbean_area+corn_area<=
total_2023_beans
```

```
# 约束 10: 小麦，谷子，高粱，玉米之间的替代性
total_2023_crop = 0
for i in range(6,10):
filtered_crop_data = crop_production[crop_production['作物
编号'] == i]
for index, row in filtered_crop_data.iterrows():
plot_type = row['种植地块']
plot_area = row['种植面积/亩']
plot_n = plot_type_mapping[plot_type_mapping['地块名称
']==plot_type]['地块类型'].values[0]
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作
物编号'] == i )&(plot_yield_mapping['地块类型'] == plot_n)]['
亩产量/斤'].values[0]
total_2023_crop += plot_area * yield_per_mu
for year in range(2024, 2031):
for index, row in attachment1_data.iterrows():
plot = row['地块名称']
# 定义三个作物的种植面积变量
total_area =
pulp.lpSum(variables[f"x_{plot}_{crop}_{season}_{year}"]
for crop in range(6,10) for season in ['第一季','第二季'])
```

```
model += soybean_area+blackbean_area+corn_area<=
total_2023_crop
```

```
#定义目标函数
```

```
# 定义增长率下降率等
wheat_corn_growth_rate = np.random.uniform(0.05, 0.10,
size=(2031-2024)) # 小麦和玉米年增长率
other_crop_sales_variation = np.random.uniform(-0.05, 0.05,
size=(2031-2024)) # 其他作物的年销售变化
yield_variation = np.random.uniform(-0.10, 0.10,
size=(2031-2024)) # 每年亩产量变化
cost_growth_rate = 0.05 # 种植成本年增长率
vegetable_price_growth_rate = 0.05 # 蔬菜价格年增长率
fungi_price_decrease_rate = np.random.uniform(0.01, 0.05,
size=(2031-2024)) # 食用菌价格下降率
morchella_price_decrease_rate = 0.05 # 羊肚菌价格下降率
```

```
total_profit = 0
```

```
# 读取附件表格信息，以字典的形式得到 2023 年的作物数据（总产量，每亩成
本，销售价格）以及 2024-2030 年的每年每种作物的亩产量
crop_data_2023 = {}
crop_yield_per_year = {}
for i in range(1,42):
#对于每种作物，计算 2023 年的总产量，每亩成本，销售价格
filtered_crop_data = crop_production[crop_production['作物
编号'] == i]
crop_yield_per_year[i] = {}
total_production = 0
yield_per_mu = 0
cost_per_mu = 0
price = plot_price_mapping[plot_price_mapping['作物编号'] ==
i]['平均销售单价/(元/斤)'].values[0]
for index,row in crop_stats.iterrows():
plot_type = row['地块类型']
crop_yield_per_year[i][plot_type]={}
```

```
for year in range(2023, 2031):
crop_yield_per_year[i][plot_type][year] = 0
for index, row in filtered_crop_data.iterrows():
plot_name = row['种植地块']
plot_area = row['种植面积/亩']
#根据地块类型找到对应的亩产量
if len(plot_yield_mapping[(plot_yield_mapping['作物编号
']==i) & (plot_yield_mapping['地块类型'] == plot_type)]['亩
产量/斤'].values) == 0:
yield_per_mu = 0
else:
yield_per_mu = plot_yield_mapping[(plot_yield_mapping['作
物编号'] == i )&(plot_yield_mapping['地块类型'] ==
plot_type)]['亩产量/斤'].values[0]
crop_yield_per_year[i][plot_type][2023] = yield_per_mu
```

```
for year in range(2024,2031):
crop_yield_per_year[i][plot_type][year] =
crop_yield_per_year[i][plot_type][year-1] * (1 +
yield_variation[year-2024])
# 计算 2023 年总产量
total_production += plot_area * yield_per_mu
cost_per_mu = plot_cost_mapping[plot_cost_mapping['作物编号
']==i]['种植成本/(元/亩)'].values[0]
crop_data_2023[i] = {'total_production': total_production,
'cost_per_mu': cost_per_mu, 'price': price}
```

```
#开始计算 2024-2030 年的总成本和总利润
for year in range(2024, 2031):
year_cost = 0
for i in range(1,42):
total_area = 0
expected_sales = 0
this_year_cost_per_mu = 0
this_year_price = 0
this_year_production = 0
excess_production = 0
if year == 2024:
```

```
last_year_production =
crop_data_2023[i]['total_production']
last_year_cost_per_mu = crop_data_2023[i]['cost_per_mu']
last_year_price = crop_data_2023[i]['price']
else:
last_year_production = 0
last_year_cost_per_mu = 0
last_year_price = 0
#计算预期销售量
if i in range(6,8):
expected_sales = last_year_production * (1 +
wheat_corn_growth_rate[year - 2024])
else:
expected_sales = last_year_production * (1 +
other_crop_sales_variation[year - 2024])
#计算今年每亩的种植成本
this_year_cost_per_mu = last_year_cost_per_mu * (1 +
cost_growth_rate)
#计算今年的销售价格
price_this_year_21 = 0
if i in range(1,17):
this_year_price = last_year_price
elif i in range(17,38):
#考虑到 21 号作物和 23 号作物的互补性，添加约束使每年 21 号作物和 23 号
作物的价格增长率相等
if i == 21:
this_year_price = last_year_price * (1 +
vegetable_price_growth_rate)
price_this_year_21 = last_year_price * (1 +
vegetable_price_growth_rate)
if i==23:
this_year_price = price_this_year_21
else :
this_year_price = last_year_price * (1 +
vegetable_price_growth_rate)
elif i in range(38,41):
this_year_price = last_year_price * (1 -
fungi_price_decrease_rate[year - 2024])
```

```
elif i == 41:
this_year_price = last_year_price * (1 -
morchella_price_decrease_rate)
#计算今年某种植物的总产量
for season in ['第一季','第二季']:
for index, plot_row in plot_type_mapping.iterrows():
plot_name = plot_row['地块名称']
plot_type = plot_row['地块类型']
#获取实际种植面积
var_name = f"x_{plot_name}_{i}_{season}_{year}"
planting_area_var = variables.get(var_name, None)
total_area += planting_area_var
this_year_production += planting_area_var *
crop_yield_per_year[i][plot_type][year]
#计算今年的总成本
year_cost += total_area * this_year_cost_per_mu
#计算今年某种植物的利润
excess_production = this_year_production - expected_sales
total_profit += ((this_year_production - excess_production)
* this_year_price + excess_production * 0.5 * last_year_price
- year_cost)
#更新上一年的产量，成本，价格
last_year_production = this_year_production
last_year_cost_per_mu = this_year_cost_per_mu
last_year_price = this_year_price
# 加入目标函数
model += total_profit
#求解
model.solve()
```

```
crop_type_list = []
crop_name_list = []
price_list = []
planting_cost_list = []
expected_sales_list = []
# 遍历求解后的变量
for year in range(2024, 2031):
for i in range(1, 42):
```

```
total_production = 0
this_year_price = 0
expected_sales = 0
# 根据作物编号和年份，获取相关的种植面积、价格、成本等
for season in ['第一季', '第二季']:
for index, plot_row in plot_type_mapping.iterrows():
plot_name = plot_row['地块名称']
plot_type = plot_row['地块类型']
```

```
# 获取变量名并获取其求解后的值
var_name = f"x_{plot_name}_{i}_{season}_{year}"
planting_area_var = variables.get(var_name, None)
if planting_area_var is not None and
planting_area_var.varValue is not None:
total_production += planting_area_var.varValue
```

```
# 更新作物的价格、种植成本和预期销售量
if i in range(1, 17):
this_year_price = last_year_price
elif i in range(17, 38):
this_year_price = last_year_price * (1 +
vegetable_price_growth_rate)
elif i in range(38, 41):
this_year_price = last_year_price * (1 -
fungi_price_decrease_rate[year - 2024])
elif i == 41:
this_year_price = last_year_price * (1 -
morchella_price_decrease_rate)
```

```
expected_sales = total_production * this_year_price # 这里
计算预期销售量
```

```
# 将数据填入对应列表
crop_name_list.append(plot_crop_mapping[plot_crop_mapping
['作物编号'] == i]['作物名称'].values[0])
crop_type_list.append(plot_crop_mapping[plot_crop_mapping
['作物编号'] == i]['作物类型'].values[0])
price_list.append(this_year_price)
planting_cost_list.append(last_year_cost_per_mu * (1 +
```

```
cost_growth_rate))
expected_sales_list.append(expected_sales)

#使用真实数据计算相关性矩阵
# 确保列表长度一致，并且每年有 41 个数据
group_size = 41

data_2024 = {
'年份': [2024] * group_size,
'作物名称': crop_name_list[:group_size],
'作物类型': crop_type_list[:group_size],
'销售价格': price_list[:group_size],
'种植成本': planting_cost_list[:group_size],
'预期销售量': expected_sales_list[:group_size]
}

data_2025 = {
'年份': [2025] * group_size,
'作物名称': crop_name_list[group_size:2*group_size],
'作物类型': crop_type_list[group_size:2*group_size],
'销售价格': price_list[group_size:2*group_size],
'种植成本': planting_cost_list[group_size:2*group_size],
'预期销售量': expected_sales_list[group_size:2*group_size]
}

data_2026 = {
'年份': [2026] * group_size,
'作物名称': crop_name_list[2*group_size:3*group_size],
'作物类型': crop_type_list[2*group_size:3*group_size],
'销售价格': price_list[2*group_size:3*group_size],
'种植成本': planting_cost_list[2*group_size:3*group_size],
'预期销售量':
expected_sales_list[2*group_size:3*group_size]
}

data_2027 = {
'年份': [2027] * group_size,
'作物名称': crop_name_list[3*group_size:4*group_size],
'作物类型': crop_type_list[3*group_size:4*group_size],
```

```
'销售价格': price_list[3*group_size:4*group_size],
'种植成本': planting_cost_list[3*group_size:4*group_size],
'预期销售量':
expected_sales_list[3*group_size:4*group_size]
}

data_2028 = {
'年份': [2028] * group_size,
'作物名称': crop_name_list[4*group_size:5*group_size],
'作物类型': crop_type_list[4*group_size:5*group_size],
'销售价格': price_list[4*group_size:5*group_size],
'种植成本': planting_cost_list[4*group_size:5*group_size],
'预期销售量':
expected_sales_list[4*group_size:5*group_size]
}

data_2029 = {
'年份': [2029] * group_size,
'作物名称': crop_name_list[5*group_size:6*group_size],
'作物类型': crop_type_list[5*group_size:6*group_size],
'销售价格': price_list[5*group_size:6*group_size],
'种植成本': planting_cost_list[5*group_size:6*group_size],
'预期销售量':
expected_sales_list[5*group_size:6*group_size]
}

data_2030 = {
'年份': [2030] * group_size,
'作物名称': crop_name_list[6*group_size:7*group_size],
'作物类型': crop_type_list[6*group_size:7*group_size],
'销售价格': price_list[6*group_size:7*group_size],
'种植成本': planting_cost_list[6*group_size:7*group_size],
'预期销售量':
expected_sales_list[6*group_size:7*group_size]
}

# 检查求解器状态
solution_status = pulp.LpStatus[model.status]
objective_value = pulp.value(model.objective)
```

```
# 将结果写入 txt 方便导入 xlsx 中
with open('C:/Users/Coco/Desktop/解 3.txt', 'w',
encoding='utf-8') as file:
sys.stdout = file
for variable in model.variables():
if variable.varValue != 0 and 'x_' in variable.name:
print(f"变量名: {variable.name}, 变量值:
{variable.varValue}")
```

附录 6

介绍：load_to_excel.py

```
import pandas as pd

# 文件路径
txt_file_path = 'C:/Users/Coco/Desktop/解 2.txt'
xlsx_file_path = 'C:/Users/Coco/Desktop/result1_2.xlsx'

# 读取 txt 文件
with open(txt_file_path, 'r', encoding='utf-8') as file:
lines = file.readlines()

# 解析 txt 文件中的数据
data = []
for line in lines:
parts = line.strip().split(", 变量值: ")
if len(parts) == 2:
variable_name, variable_value = parts[0].strip(),
float(parts[1].strip())
data.append((variable_name, variable_value))

# 加载 Excel 文件
excel_data = pd.ExcelFile(xlsx_file_path)

# 遍历每个年份的 sheet 进行修改
for variable_name, variable_value in data:
# 解析变量名，例如 x_A1_18_第一季_2029
```

```
parts = variable_name.split("_")
if len(parts) == 5:
plot_id, crop_id, season, year = parts[1], int(parts[2]),
parts[3], parts[4]
```

```
# 检查是否有对应的年份 sheet
if year in excel_data.sheet_names:
df = pd.read_excel(xlsx_file_path, sheet_name=year)
# 找到第一列为"第一季"的行
season_row = df[df.iloc[:, 0].str.contains(season,
na=False)].index
if not season_row.empty:
# 找到第二列为地块名（例如 A1）的行
plot_row = df[df.iloc[:, 1] == plot_id].index
if not plot_row.empty:
# 锁定行，并找到第 (2 + 作物编号) 列
target_row = plot_row[0] # 找到地块行
target_column = 1 + crop_id # 找到作物列
# 更新该单元格的值
if target_column < len(df.columns):
df.iloc[target_row, target_column] = variable_value
print(parts[1]+"_"+parts[2]+"_"+parts[3]+"_"+parts[4])
else:
print(f"列 {target_column} 超出范围，无法更新
{variable_name}")
else:
print(f"未找到季节 {season} 的行")
# 保存修改后的 DataFrame 到 Excel 文件
with pd.ExcelWriter(xlsx_file_path, mode='a',
if_sheet_exists='replace') as writer:
df.to_excel(writer, sheet_name=year, index=False)
```

附录 7

介绍：picture_painting.py

```
import matplotlib.pyplot as plt
crops = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆', '小麦', '玉米
```

```
', '谷子']
colors = ['orange', 'red', 'pink', 'magenta', 'blue', 'cyan',
'green', 'yellow']

file_path = './result1_1.xlsx'
excel_data = pd.ExcelFile(file_path)
sheet_names = excel_data.sheet_names
def plot_crop_yields_for_year(sheet_name):
data = excel_data.parse(sheet_name)
clean_data = data.dropna(subset=['地块名'])
```

```
plt.figure(figsize=(10, 6))
for idx, crop in enumerate(crops):
plt.plot(clean_data['地块名'], clean_data[crop],
marker='o', linestyle='-', color=colors[idx], label=crop)
plt.title(f"Problem1-1, Line Chart of Crop Yields in
Different Fields ({sheet_name})")
plt.xlabel("Field")
plt.ylabel("Yield")
plt.legend(title="Crops")
plt.xticks(rotation=45)
plt.tight_layout()
output_path = f"/mnt/data/problem1_1_{sheet_name}.png"
plt.savefig(output_path)
plt.close()
return output_path
```

```
output_paths = []
for sheet in sheet_names:
output_paths.append(plot_crop_yields_for_year(sheet))
```

```
output_paths
```