某华北山区农作物最优种植策略综合研究

摘要

根据乡村的实际情况，本文选择适宜的农作物、方便田间管理、提高生产效益、减少种植风险从而优化种植策略，具有重要的现实经济意义。为帮助某地处华北山区，常年温度偏低的乡村制定 2024~2030 年农作物的最优种植方案，对 2023 年农作物、耕地、种植数据整合与分析，兼顾提高生产效益与弱化种植风险，设定合理的基本假设，建立数学模型。

针对问题一，建立有约束条件的非线性规划模型，以最大化 2024~2030 总利润作为目标函数，以作物种植面积作为决策变量，以耕地面积、重茬种植等作为基本约束，并考虑到方便耕种作业，引入**香农多样性指数**和**冲突面积**约束。采用遗传算法求解整数规划问题，基于**贪心算法**的思想根据年份、耕地类型对 30996 个决策变量切片至最多仅有 780 个决策变量，显著降低算法参数量，由每步局部最优解，希望得到全局最优解。最终在第一种情形下解得总利润最大为 **3162 万元**，在第二种情形下解得总利润最大为 **3832 万元**。

针对问题二，首先**量化不确定性**，即明确本问中的预期销售量、亩产量、种植成本、销售单价中的变化量与确定量，进而计算得到利润的概率密度函数；其次考虑**种植风险指标**，由 Sharp Ratio 等经济学概念启发，选择可用于衡量单位风险的变异系数作为种植风险指标。基于问题一有约束条件的非线性规划模型，兼顾**提高生产效益与弱化种植风险**，以最大化利润期望与利润变异系数的比例作为目标函数，即**承担一单位风险获得的平均收益**。采用**蒙特卡洛**方法优化目标函数求解过程，即生成 100 组符合分布的随机数，得到确定的预期销售量等变化量，进而求解确定的利润样本，引入**无偏估计量**，由样本估计总体均值与方差，从而**兼顾运算速度与目标函数准确性**，实现双效益。最终解得承担一单位风险获得的平均最大收益为 **76.85 亿元**

针对问题三，首先引入**成本价格弹性系数**，融入成本对价格的影响，即成本与价格存在同向百分比变化；其次引入**需求价格弹性系数**，并考量农产品的必需性，即价格与需求存在反向百分比变化，且必需性越强相关性越弱；最后考虑农作物之间存在的替代性，由 **Shapiro-Wilk 检验**约束同类农作物价格的聚集性。延续问题二的蒙特卡洛方法优化与目标函数，最终解得承担一单位风险获得的平均最大收益为 **57.26 亿元**。并从波动性、收益性等角度对比分析问题二与问题三的结果。

关键词：遗传算法　贪心算法　蒙特卡洛　成本价格弹性　需求价格弹性

一、 问题重述

某地处华北山区的乡村，因常年温度偏低，导致大多数耕地每年只能种植一季农作物。该乡村拥有总计 1201 亩的露天耕地，这些耕地被分割成了 34 个不同大小的地块，它们根据地形和灌溉条件被分为四种类型：平旱地、梯田、山坡地和水浇地。其中，平旱地、梯田和山坡地适合每年种植一季粮食类作物；而水浇地可以选择每年种植一季水稻或者两季蔬菜。同时，该乡村另有 16 个普通大棚和 4 个智慧大棚，大棚耕地面积均为 0.6 亩，普通大棚和智慧大棚分别适合每年种植一季蔬菜和一季食用菌以及每年种植两季蔬菜。

根据农作物的生长规律，同一地块（包括大棚）**不能连续种植同一种作物**，否则会导致减产。由于豆类作物的根菌利于其他作物生长的特性，自 2023 年起，每个地块（包括大棚）在三年内必须至少种植一次豆类作物。此外，种植方案应考虑**便于耕作和田间管理**，例如：每季种植的作物地块不宜过于分散，单个地块（包括大棚）内的作物种植面积不宜过小等。

附件 1 给出了乡村现有耕地情况和乡村种植的农作物的情况；附件 2 则给出了 2023 年农作物的种植情况以及相关统计数据（亩产量、种植成本等）。现完成如下问题：

1. 在假定各种农作物未来的预期销售量、种植成本、亩产量和销售价格与 2023 年保持稳定且每季种植的农作物必须在当季销售的情况下，针对两种不同的处理产量大于相应的预期销售量的作物的方法：（1）超过部分**滞销**，造成浪费；（2）超过部分按 2023 年销售价格的 **50% 降价出售**，分别给出该乡村 2024 至 2030 年农作物的最优种植方案。
2. 在考虑到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与 2023 年相比大约有 ±5% 的变化，且农作物的亩产量每年有 ±10% 的变化、种植成本平均每年增长 5% 左右，以及粮食类作物的销售价格基本稳定、蔬菜类作物的销售价格平均每年增长 5% 左右、食用菌的销售价格大约每年可下降 1%~5%（羊肚菌的销售价格每年下降幅度为 5%）的情况下，给出该乡村 2024 至 2030 年农作物的最优种植方案。
3. 综合考虑由于各种农作物之间存在的**可替代性**和互补性而导致的它们**预期销售量与销售价格、种植成本之间也存在的相关性**，在问题 2 的基础上给出该乡村 2024 至 2030 年农作物的最优种植策略，通过模拟数据进行求解，并与问题 2 的结果作比较分析。

二、 问题分析

2.1 问题一的分析

针对问题一，首先基于附件 1 和附件 2 提取不同农作物在不同地块预期销售量、亩产、种植成本、销售价格等信息。考虑到题目存在目标函数以及限制条件，因此建立有约束条件的非线性规划模型，以作物种植面积作为决策变量，以最大化 2024~2030 总利润作为目标函数，通过决策变量与亩产、种植成本、预期销售量、销售价格等定量给出总利润计算公式，以每个地块的作物面积之和不大于耕地面积、不能重茬种植、三年内豆类种植等作为基本约束。考虑到方便耕种作业，分别对露天耕地和温室分别设置最小种植面积为 6 亩，0.3 亩作为约束防止耕地细碎化；并引入**香农多样性指数**作为约束，香农多样性指数较小，则说明农作物种类较少或某几种农作物占比较大；以及**冲突面积**约束防止作物布局细碎化。具体采用遗传算法求解整数规划问题，基于**贪心算法**的思想根据年份、耕地类型对 30996 个决策变量切片至最多仅有 780 个决策变量，显著降低算法参数量，由每步局部最优解，希望得到全局最优解。

2.2 问题二的分析

针对问题二，考虑到本问题设条件下预期销售量、亩产量、销售价格、种植成本不再是定值，而是与年份有关的变化量或确定量。因此首先应**量化不确定性**，即明确本问中的预期销售量、亩产量、种植成本、销售单价中的变化量与确定量，进而计算得到利润的概率密度函数；其次考虑**种植风险指标**，由 Sharp Ratio 等经济学概念启发，选择可用于衡量单位风险的变异系数作为种植风险指标。基于问题一中有约束条件的非线性规划模型，兼顾**提高生产效益与弱化种植风险**，以最大化利润期望与利润变异系数的比例作为目标函数，即**承担一单位风险获得的平均收益**。采用**蒙特卡洛**方法优化目标函数求解过程，即生成 100 组符合分布的随机数，将预期销售量、销售价格等变化量设定为符合分布的随机定值，进而求解确定的利润样本，引入**无偏估计量**，由样本估计总体均值与方差，从而**兼顾运算速度与目标函数准确性**，实现双效益。

2.3 问题三的分析

针对问题三，考虑各种农作物之间可能存在一定的可替代性和互补性，预期销售量与销售价格、种植成本之间也存在一定的相关性。对于种植成本和销售价格的相关性而言，引入**成本价格弹性系数**，通过查阅文献设定成本价格弹性系数符合均值为 0.427，标准差为 0.197 的正态分布从而融入成本对价格的影响，即成本与价格存在同向百分比变化；对于销售价格和预期销售量的相关性而言，引入**需求价格弹性系数**，并考量农产品的必需性，即价格与需求存在反向百分比变化，且必需性越强相关性越弱，通过查阅文献设定粮食、蔬菜需求价格弹性系数为-0.3，食用菌需求价格弹性系数为-0.1；最后考

虑农作物之间存在的替代性，由 **Shapiro-Wilk 检验**约束同类农作物价格的聚集性。就具体操作而言，在问题二的基础上，对成本价格弹性系数进行蒙特卡洛模拟，并加入成本价格弹性系数和需求价格弹性系数对价格和预期销售量进行修正，最后对蒙特卡洛生成的同类农作物价格数据进行 Shapiro-Wilk 检验，若不通过检验，则去除该组数据重新生成。

三、 基本假设

为了便于模型的计算与求解，我们定义了如下**基本假设**：

1. 2023 年的预期销售量与 2023 年的实际种植情况一致；
2. 取作物销售单价区间的平均值作为作物销售单价；
3. 水浇地、普通大棚、智慧大棚的同季度预期销售量统一；
4. 问题 2、问题 3 中波动的数据在相应的范围内呈均匀分布；
5. 问题 2、问题 3 中作物每季的总产量超过相应的预期销售量的部分滞销。

四、 符号说明

在这里，我们只说明符号本身的特定含义，即当符号含有到下标 *i,j,s,year* 时，默认为编号为 i 的作物在地 j 第 s 季度第 $year$ 年的情况（如 $W_{i,j,s,year}$ 表示编号为 i 的作物在地 j 第 s 季度第 $year$ 年的利润）。

表 1　符号说明

符号	含义	单位
W	相应情况下的利润	元
X	相应情况下的种植面积	亩
$Product$	相应情况下的总产量	斤
$\hat{Product}$	相应情况下的预期销售量	斤
$Sales$	相应情况下的实际销售量	元
$Yield$	相应情况下的亩产量	斤/亩
$Price$	相应情况下的销售单价	元/斤
$Cost$	相应情况下的种植成本	元/亩
μ_W	收益 W 的期望	元
σ_W	收益 W 的标准差	元

五、 数据预处理

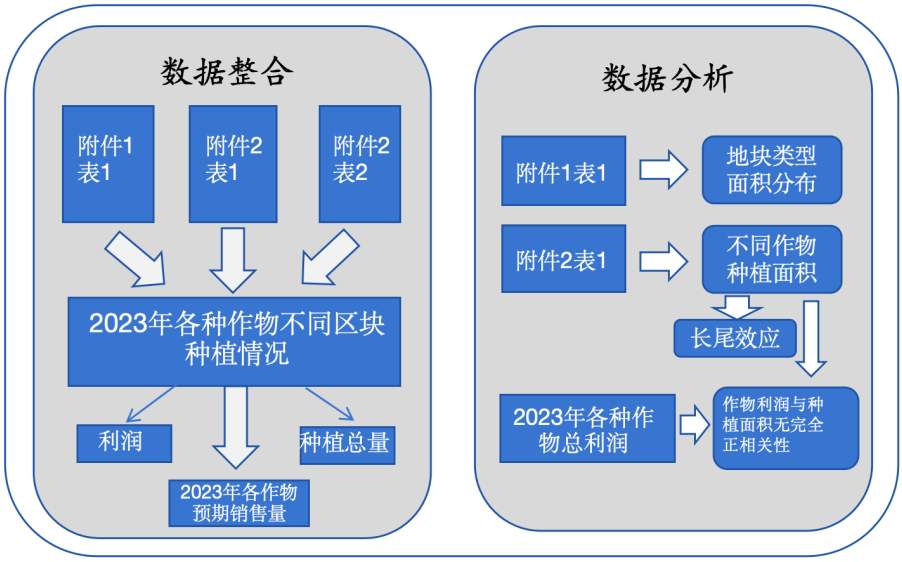


图 1　数据预处理思路

附件 1 给出了乡村现有耕地情况和乡村种植的农作物的情况，附件 2 则给出了 2023 年农作物的种植情况以及相关统计数据（亩产量、种植成本等）。我们首先依据数据之间的关联性整合数据，并用区间均值代替平均销售售价，再利用原始数据和整合数据对地块类型面积分布、不同作物种植面积分布和 2023 年各作物总利润情况进行数据对比分析等。

5.1 数据整合

首先，根据附件 2 中表“2023 年统计的相关数据”给出的各种作物在不同季与不同地块的销售单价区间，我们取区间上限与下限的平均值作为该作物的平均销售单价，即假设 2：

$$Price = \frac{Price_{max} + Price_{min}}{2} \tag{1}$$

这里的 $Price$ 表示各种作物的在**不同情况下平均销售单价**，鉴于其普适性，我们在公式（1）中省略了下标。

本题附件 2 中表“2023 年的农作物种植情况”给出了 2023 年每地块在不同季的作物种植情况，结合附件 1 中表“乡村的现有耕地”给出的地块编号与地块类型的对应关系、附件 2 中表“2023 年统计的相关数据”所给出的各种作物在不同季与不同地块的亩产量、种植成本和平均销售单价，可以得到 2023 年不同作物在不同地块类型中不同季度的种植面积与销售利润：

$$W_{i,j,s,2023} = X_{i,j,s,2023} Yield_{i,j,s,2023} Price_{i,j,s,2023} - X_{i,j,s,2023} Cost_{i,j,s,2023} \tag{2}$$

$$W^{J}_{i,s,2023} = \sum_{j \in J} W_{i,j,s,2023} \tag{3}$$

其中，下标 $i, j, s, 2023$ 表示 **2023 年编号为 i 的作物在第 j 地块第 s 季度的情况**，W 表示**利润**，X 表示**种植面积**，$Yield$ 表示**亩产量**，$Cost$ 表示**单位种植成本**。

而 $W^{J}_{i,s,2023}$ 表示 **2023 年编号为 i 的作物第 s 季度在第 J 大类获得的总利润**，其中 $J \in$ { 平旱地, 梯田, 山坡地, 水浇地, 普通大棚, 智慧大棚 }。

用上述得到的 2023 年不同作物在不同地块类型中不同季度的种植面积与销售利润，我们通过对同一作物的利润求和的方式，可以得到在 2023 年内不同作物所带来的总利润：

$$W_{i,2023} = \sum_{s=1}^{2} \sum_{J \in A} W^{J}_{i,s,2023} \tag{4}$$

其中，$W_{i,2023}$ 表示 **2023 年编号为 i 的作物获得的总利润**，A 即为地块类型大类的集合：{ 平旱地, 梯田, 山坡地, 水浇地, 普通大棚, 智慧大棚 }。

5.2 数据分析

本题附件 1 中表“乡村的现有耕地”给出了该乡村现有耕地情况，即地块类型与相应的面积。我们利用 Python 代码，将相同类型的地块面积求和：

$$S^J = \sum_{j \in J} S_j \tag{5}$$

其中，S^J 表示**第 J 大类地形的总面积**，S_j 表示**编号为 j 的地块的面积**。

得到了不同类型地块总面积的表格后，我们可视化了不同类型地块面积的分布。我们发现，地块**面积差异非常显著**，如：最大面积为梯田，有 619 亩；而最小面积的智慧大棚只有 2.4 亩。

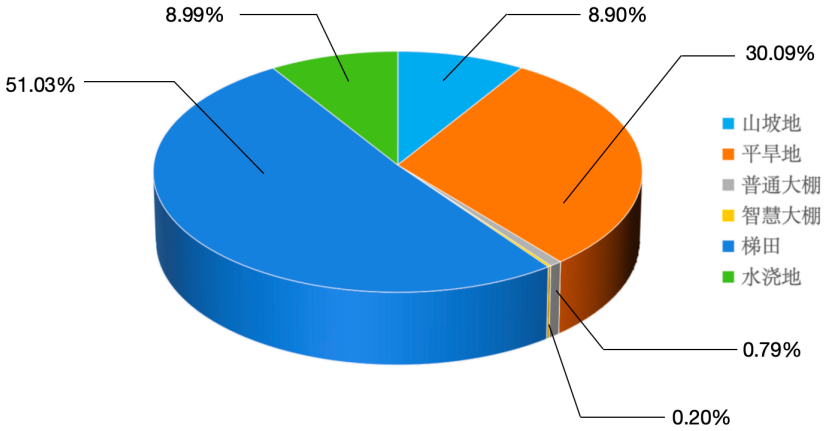
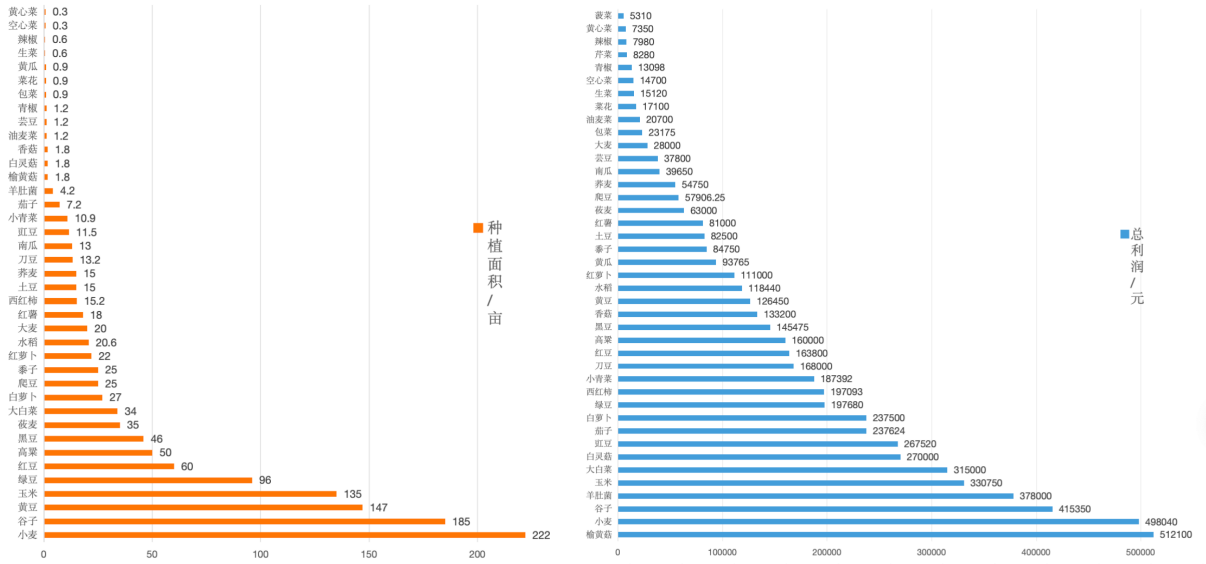


图 2　各地块类型面积分布

本题附件 2 中表“2023 年的农作物种植情况”给出了 2023 年各作物在各地块的种植情况，我们利用 Python 代码，求得了 41 种作物每种作物的总种植量。我们发现，**各作物种植面积的分布差异显著**。其具体表现为：粮食作物为**主要种植对象**，而经济作物种植**面积较小**。总的来说，每种作物的总种植量具有**长尾效应**：小麦（222 亩）、谷子（185 亩）、黄豆（147 亩）、玉米（135 亩）等粮食作物占据了种植面积的大部分。少部分作物面积非常大；相对而言，许多蔬菜、经济作物的种植面积非常小，例如羊肚菌（4.2 亩）、榆黄菇（1.8 亩）、油麦菜（1.2 亩）、生菜（0.6 亩）等。

同时，利用整合得到的“2023 每种作物种植利润”表格，观察到：作物的利润**分布范围从几千元到几十万元**，即不同作物之间的**种植效益差异显著**。具体来看表现为下面几个特征：高利润作物集中在蘑菇类；传统粮食作物表现稳健：小麦、谷子、玉米等传统粮食作物的利润中等偏高，表明它们具有相对稳定的市场需求；蔬菜类作物利润较低。



(a) 各作物种植总量　　(b) 各作物总利润

图 3　各作物种植与利润情况

通过对上图的对比分析，我们发现：种植面积和总利润之间**没有完全的正相关性**，这也反映了市场需求、作物种类及种植管理对最终利润的影响。

由于假设 1，求出 2023 年各作物每季度的种植情况对于后续预期销售量的确定至关重要。因此，我们利用 Python 代码处理了附件 2，得到了 2023 年各作物每季度的种植情况。下面以单季度地块为例对该指标进行具体分析：

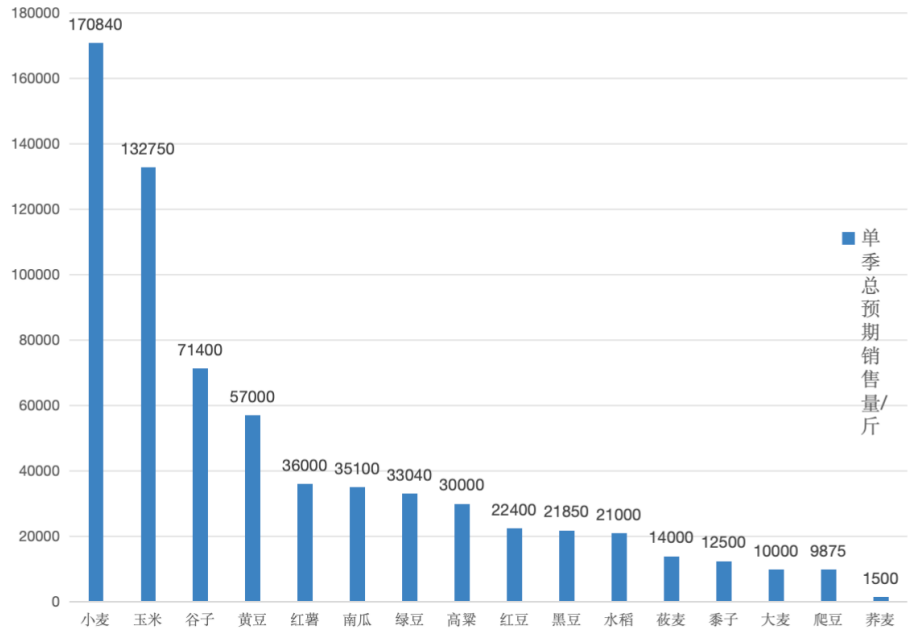


图 4　2023 年单季度作物预期销售量

在单季度作物预期销售量中，我们观察到：小麦的预期销售量远高于其他作物，而荞麦的销售量最低。总体来说，大多数作物的销售量集中在较低范围内，只有小麦和玉

米的销量大于十万斤，显示出销售量的分布是偏态的。另一方面，附件 1 给出了水浇地、普通大棚、智慧大棚的第一季、第二季时间分布差异较大，考虑到实际生活情形与模型建立，我们提出假设 3，即水浇地、普通大棚、智慧大棚的同季度预期销售量统一，为三种耕地的 2023 年同季度产量之和。

六、 问题一的模型建立与求解

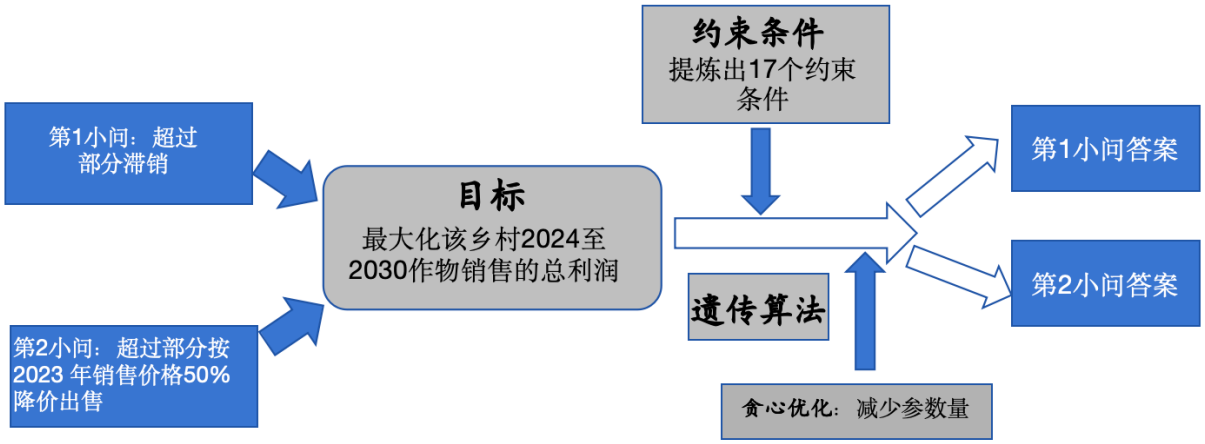


图 5　问题一思路流程图

6.1 确定决策变量

由题意知，本问中的决策变量为 2024 年至 2030 年不同地块的农作物种植情况，即：

$$X_{i,j,s,year} \quad s=1,2, j=1,2,\ldots,54, i=1,2,\ldots,41, year=2024,2025,\ldots,2030$$

6.2 确定优化目标

本文中需要选择最优的农作物种植方案来使该乡村 2024 至 2030 作物收益最高，故优化目标为**最大化该乡村 2024 至 2030 作物的总利润**，我们记编号为 i 的作物在 $year$ 年的第 s 季的销售量为 $Sales_{i,s,year}$：

$$\max \sum_{year=2024}^{2030} \sum_{s=1}^{2} \sum_{i=1}^{41} \left[Sales_{i,s,year} \cdot Price_{i,s,year} - \sum_{j=1}^{54} (X_{i,j,s,year} \cdot Cost_{i,j,s,year}) \right] \tag{6}$$

我们记第 $year$ 年在 s 季度第 i 种农作物的总产量为 $Product_{i,s,year}$，总预计销量为 $\hat{Product}_{i,s,year}$，因此根据第（1）小问中的要求：总产量超过预期销售量的部分滞销，造成浪费，得到第（1）问的 $Sales_{i,s,year}$：

$$\text{Sales}_{i,s,year} = \begin{cases} Product_{i,s,year} & Product_{i,s,year} \leq \hat{Product}_{i,s,year}, \\ \hat{Product}_{i,s,year} & Product_{i,s,year} > \hat{Product}_{i,s,year}. \end{cases} \tag{7}$$

再针对第二问，总产量超过预期销售量的部分按 2023 年销售价格的 50% 降价出售，得到第（2）问的 $Sales_{i,s,year}$：

$$Sales_{i,s,year} = \begin{cases} Product_{i,s,year} & Product_{i,s,year} \leq \hat{Product}_{i,s,year}, \\ \frac{\hat{Product}_{i,s,year} + Product_{i,s,year}}{2} & Product_{i,s,year} > \hat{Product}_{i,s,year}. \end{cases} \tag{8}$$

记第 $year$ 年在 s 季度第 i 种农作物在第 j 块耕地上种植的亩产量为 $Yield_{i,j,s,year}$，那么我们得到 $Product_{i,s,year}$ 的公式：

$$Product_{i,s,year} = \sum_{j=1}^{54} (X_{i,j,s,year} \cdot Yield_{i,j,s,year}) \tag{9}$$

6.3 确定约束条件

我们令 i 表示作物编号，j 表示耕地编号，s 表示季度（第一或第二季度），$year$ 表示 2024 至 2030 中的某一年。$X_{i,j,s,year}$ 表示在第 $year$ 年第 s 季度第 j 块耕地上种植的编号为 i 作物的面积，S_j 表示第 j 块耕地的总面积。

值得注意的是，如果下列约束条件后面没有给出 i、j、s、$year$ 的取值，那么默认：$i = 1, 2, \ldots, 41$，$j = 1, 2, \ldots, 54$，$s = 1, 2$，$year = 2024, 2025, \ldots, 2030$。

1. 平旱地、梯田和山坡地每年都只能种植一季作物：

$$X_{i,j,s,year} = 0, \quad s = 2, j = 12, \ldots, 26 \tag{10}$$

2. 平旱地、梯田和山坡地只能种植非水稻粮食作物：

$$X_{i,j,s,year} = 0, \quad j = 12, \ldots, 26, i \neq 12, \ldots, 15 \tag{11}$$

3. 水浇地水稻只能种植一季：

$$X_{i,j,s,year} = 0, \quad s = 2, j = 27, 28, \ldots, 34, i = 16 \tag{12}$$

4. 水浇地蔬菜第二季度只能种植大白菜、白萝卜、红萝卜：

$$X_{i,j,s,year} = 0, \quad s = 2, j = 27, 28, \ldots, 34, i \neq 35, 36, 37 \tag{13}$$

5. 水浇地蔬菜第一季只能种植非大白菜、白萝卜、红萝卜外的其他蔬菜：

$$X_{i,j,s,year} = 0, \quad s = 1, j = 27, 28, \ldots, 34, i \neq 17, 18, \ldots, 34 \tag{14}$$

6. 浇水地要么种一季水稻，要么种两季蔬菜：

$$\begin{cases} X_{16,j,1,year} \cdot X_{i_1,j,1,year} = 0 \quad i_1 = 17, 18, \ldots, 34 \\ X_{16,j,1,year} \cdot X_{i_2,j,2,year} = 0 \quad i_2 = 35, 36, 37 \\ j = 27, 28, \ldots, 34 \end{cases} \tag{15}$$

7. 普通大棚第一季度只能种植非大白菜、白萝卜、红萝卜外的其他蔬菜：

$$X_{i,j,s,year} = 0, \quad s = 1, j = 35, 36, \ldots, 50, i \neq 17, 18, \ldots, 34 \tag{16}$$

8. 普通大棚第二季度只能种植食用菌：

$$X_{i,j,s,year} = 0, \quad s = 2, j = 35, 36, \ldots, 50, i \neq 38, 39, 40, 41 \tag{17}$$

9. 智慧大棚两季度都只能种植非大白菜、白萝卜、红萝卜外的其他蔬菜：

$$X_{i,j,s,year} = 0, \quad j = 51, 52, 53, 54, i \neq 17, 18, \ldots, 34 \tag{18}$$

10. 每块耕地上种植面积应小于等于耕地面积:

$$X_{i,j,s,year} \leq S_j \tag{19}$$

11. 每块耕地三年内都至少种植一次豆类：

$$\sum_{t=year-1}^{year+1} \sum_{s=1}^{2} X_{i,j,s,year} > 0, \quad year = 2024, 2025, \ldots, 2029, i = 1, 2, 3, 4, 5, 17, 18, 19 \tag{20}$$

12. 每种作物在同一地块都不能连续两年种植：

$$X_{i,j,s,year} \cdot X_{i,j,s,year-1} = 0 \tag{21}$$

13. 智慧大棚中同一地块不能连续两年种同一蔬菜：

$$X_{i,j,s,year} \cdot X_{ijs,year+1} = 0, \quad s = 1, j = 51, 52, 53, 54, i \neq 17, 18, \ldots, 34 \tag{22}$$

14. 考虑到种植作物时，应**方便耕种作业和田间管理**，每种作物在单个地块种植的面积不宜太小。通过查阅文献，我们得到：在面积在约 6 亩以上的露天耕地的耕作效率随面积增大而保持恒定，而在约 2 亩至约 6 亩的露天耕地时耕作效率随面积增大而增大 [1]。同时，我们观察该乡村的所有露天耕地发现：最小的地块面积为 6 亩。因此，我们为**防止耕地细碎化、提高耕作效率，设定每种被种植作物在单一露天耕地中至少占 6 亩**：

$$X_{i,j,s,year} \geq 6 \quad or \quad X_{i,j,s,year} = 0 \quad j = 12, \ldots, 34 \tag{23}$$

15. 大棚种植作物体现了**精细化种植的优势**，同时，观察了 2023 年的作物种植数据后，我们发现：即使是大棚内种植作物，被种植作物在大棚中至少占 0.3 亩。因此，为了防止耕地细碎化所带来的影响，我们设定**每种被种植作物在大棚中至少占 0.3 亩**：

$$X_{i,j,s,year} \geq 0.3 \quad or \quad X_{i,j,s,year} = 0 \quad j = 35, \ldots, 54 \tag{24}$$

16. 为了刻画种植作物的 **“混乱程度”**，我们引入了**香农多样性指数（Shannon-Weaver Diversity Index）**，其可以反应作物的多样性与分布情况 [2]：

$$H' = -\sum_{i=1}^{S} p_i \ln p_i$$

其中，H' 即为香农多样性指数。S 为作物总数，p_i 是第 i 个物种的相对丰度。而为了提高耕作的效率和便于管理，我们希望作物的“混乱程度”更小，即耕种的作物种类更少或有少量作物耕种面积占比较大：**2024 年至 2030 年的香农多样性指数都应该小于 2023 年的香农多样性指数**：

$$H'_{year} = -\sum_{i=1}^{41}\left\{\sum_{s=1}^{2}\left[\frac{\sum_{j=1}^{54} X_{i,j,s,year}}{S_j} \ln\left(\frac{\sum_{j=1}^{54} X_{i,j,s,year}}{S_j}\right)\right]\right\} \leq H'_{2023} \tag{25}$$

17. 我们为防止每种作物每季的种植地太分散，即**作物布局的细碎化**对耕种作业和田间管理的影响，引入了**冲突面积**的概念：作物分散在不同地块时，除去作物最大面积地块的剩余面积和。我们设定，**2024 年至 2030 年的冲突面积都应该小于 2023 年的冲突面积**：

$$Conflict_{year} = \sum_{s=1}^{2}\sum_{i=1}^{41}\left(\sum_{j=1}^{54} X_{i,j,s,year} - \max\{X_{i,s,year}\}\right) \leq Conflict_{2023} \tag{26}$$

其中，$X_{i,s,year} = (X_{i,1,s,year}, X_{i,2,s,year}, \ldots, X_{i,54,s,year})$。

6.4 优化模型汇总

最终汇总如下所示：

$$max \sum_{year=2024}^{2030}\sum_{s=1}^{2}\sum_{i=1}^{41}\left[Sales_{i,s,year} \cdot Price_{i,s,year} - \sum_{j=1}^{54}(X_{i,j,s,year} \cdot Cost_{i,j,s,year})\right]$$

$$st.\text{式}(10) \sim (26)$$

6.5 模型求解

对种植方案进行确定时，我们使用 Python 的 scikit-opt 库编写**遗传算法**对模型进行求解。遗传算法（Genetic Algorithm）是一种启发式优化算法，基于自然选择和遗传学

中的进化原理。其核心思想是模拟生物进化中的" 适者生存"，通过选择、交叉（杂交）和变异等操作，逐步改进解决方案，以找到全局最优或接近全局最优的结果。

在本题求解最优种植方案，即最大化作物销售利润问题中，遗传算法的具体实施步骤如下：

- 第一步：初始化种群，每个个体代表一个种植方案（某种作物在某块土地上某个季次应当种多少亩）；
- 第二步：通过适应度函数计算每个方案的总利润和资源消耗，并添加惩罚项进行约束；
- 第三步：选择适应度较高的个体进行交叉和变异，生成新一代的种植方案；
- 第四步：不断迭代，最终找到最优种植方案。

在对模型的进行条件约束时，我们使用了**惩罚函数**：即让违背约束条件的个体加上一个**惩罚项**，这样就会使该个体适应度下降，在迭代过程中被淘汰。由于本题目标函数的值较大，我们修改 scikit-opt 库的源码，设定惩罚项 $Punish$ 为 10^{10}，以淘汰不符合限制条件的个体。

本题模型的决策变量为 2024~2030 年的种植方案中所有不同地块中种植不同作物的种植面积，根据我们数组 $X_{i,j,s,year}$ 的设置，参数共有 $41 \times 54 \times 2 \times 7 = 30996$ 个，参数量过于庞大，不便于求解。

因此，我们使用**贪心算法**的思想分步求解，以降低遗传算法的参数量，再逐步利用遗传算法求解最优种植方案。

1. 首先，我们根据种植季度以及可种植作物类型**分为五大类**（注：下面记平旱地为 A，梯田为 B，山坡地为 C，水浇地为 D，普通大棚的第一季为 E1，普通大棚的第二季为 E2，智慧大棚为 F）。由于 ABC 类型耕地只能单季种植且可种植作物相同，所以一起进行规划，并且只考虑单季；D 类型耕地既可以单季种植水稻也可以两季种植蔬菜，所以考虑两季，同时进行规划；E 类型耕地第一季和第二季可种植作物不同，因此分开成 E1 和 E2 两个季次进行规划。F 类型耕地每年两季可种植作物相同，因此放在一起进行规划。
2. 其次，限制条件中关于年份的限制最长是 “每个地块三年内至少种一次豆类作物”，所以我们可以从 2023 年开始，**以三年为周期**计算最优的种植情况，即只需通过 2023 年的数据计算 2024 年和 2025 年的最优种植方案，这样就可以求得最优解。

按照以上思路进行计算，我们每一次规划的参数可以缩减为 {A, B, C}:$15 \times 26 \times 2 = 780$，{D}:$22 \times 8 \times 2 \times 2 = 704$，{E1}:$18 \times 16 \times 2 = 576$，{E2}:$4 \times 16 \times 2 = 128$，{F}:$18 \times 4 \times 2 \times 2 = 288$。同时，我们的约束也只需要考虑耕地面积约束、连茬种植约束、种植豆类约束、耕地细碎化约束、香农多样性指数约束和冲突面积约束。很大程度上降低了参数量，极大提高了遗传算法的效率。通过让每一次规划都达到最优解，组合

起来的就是全局近似最优解。

需要注意的是，由于我们以三**年为一次种植循环**，因此需要添加 2023 年和 2025 年同一块土地不能种植同种作物的限制。这个约束会对我们的模型最优解产生一定影响，但是对于遗传算法以及更多的智能优化算法来讲，更低的参数量是更有利于求出最优解的，因此我们提出的基于贪心算法思想改进的模型是更有利于得到最优方案的。在此基础上，考虑到现实生活中土地面积不会过于细分，我们进行整形规划来提高效率并优化结果，对于大棚，则把决策变量精度限制在 0.1。

本模型其中一类的求解迭代过程如下图所示：

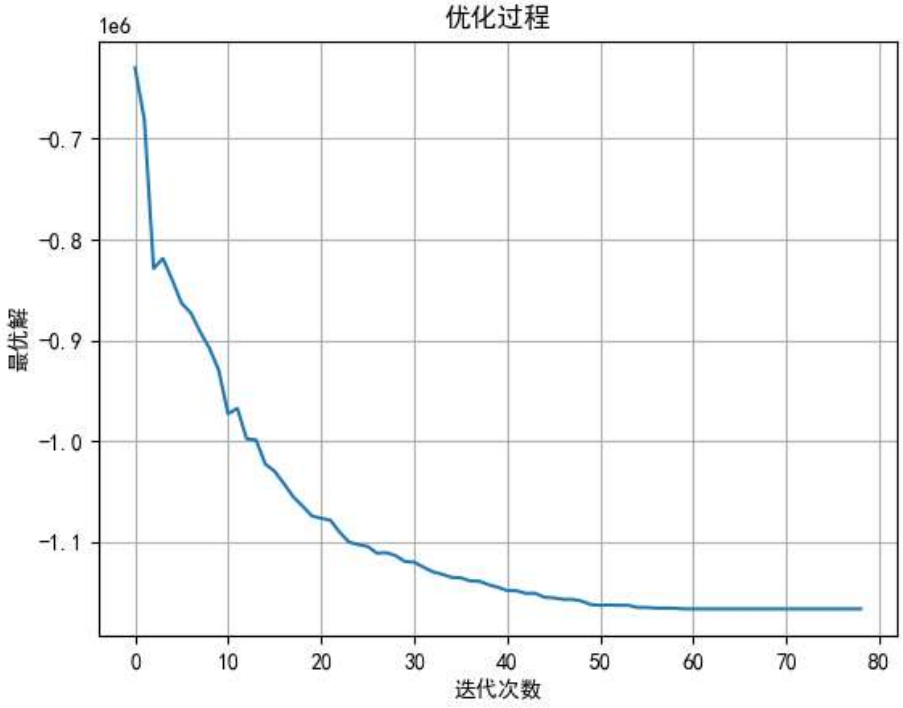


图 6　迭代过程中最优解的变化

由于 scikit-opt 库的源代码中是通过 argmin 找到最优解的，即求最小值，所以我们的适应度函数返回的是利润的相反数，可以看出，整个下降过程**近似符合双对数模型**，在迭代 80 轮左右时最优解已经收敛。

年份	情形 (1) 利润	情形 (2) 利润
2024 年	4103151.25	5196370.75
2025 年	3729940.25	5440108.75
2026 年	5926348.25	5926348.25
2027 年	4103151.25	5196370.75
2028 年	3729940.25	5440108.75
2029 年	5926348.25	5926348.25
2030 年	4103151.25	5196370.75
总计	31622030.75	38322026.25

七、 问题二的模型建立与求解

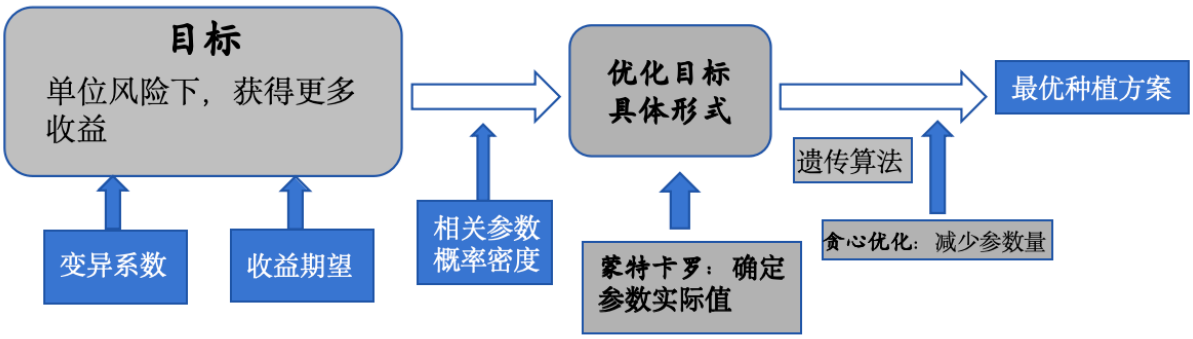


图 7　第二题思路流程图

7.1 确定决策变量

由题意知，本问中的决策变量为：在综合考虑各种农作物的预期销售量、亩产量、种植成本和销售价格的不确定性以及潜在的种植风险时，2024 年至 2030 年不同地块的农作物种植情况，即：

$$X_{i,j,s,year} \quad s = 1, 2, j = 1, 2, \ldots, 54, i = 1, 2, \ldots, 41, year = 2024, 2025, \ldots 2030$$

7.2 确定优化目标

7.2.1 明确变化量与确定量

在确定优化目标前，我们首先明确本问中的**变化量**与**确定量**，它们分别出现在预期销售量、亩产量、种植成本和销售单价中，下面将分别逐个讨论：

1. **销售量**：

（1）小麦和玉米未来的预期销售量有增长的趋势，下面公式中的 $i = 5, 6$：

$$\hat{Product}'_{i,s,year} = (1 + \alpha_1)^{(year-2023)} Product_{i,s,2023} \tag{27}$$

其中，小麦和玉米预期销售量的增长率 $\alpha_1 \sim U(5\%, 10\%)$。

概率密度函数为：

$$f_{P_1}(p) = \frac{20}{\hat{Product}_{i,s,2023} \cdot (year - 2023)} p^{\frac{1}{year-2023}-1} \tag{28}$$

概率分布函数为：

$$F_{P_1}(p) = \int_{p_{\min}}^{p_{\max}} f_{P_1}(p)dp = \frac{20}{\hat{Product}_{i,s,2023}} (p^{\frac{1}{year-2023}} - 1.05) \tag{29}$$

p 的取值范围如下：

$$1.05^{year-2023} \cdot \hat{Product}_{i,s,2023} = p_{\min} \leq p \leq 1.10^{year-2023} \cdot \hat{Product}_{i,s,2023} = p_{\max}$$

记作：$p_1 \leq p \leq p_2$。

（2）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，下面公式中 $i \neq 5, 6$：

$$\hat{Product}'_{i,s,year} = (1 + \alpha_2) \cdot \hat{Product}_{i,s,2023} \tag{30}$$

其中，其他农作物预期销售量的波动范围 $\alpha_2 \sim U(-5\%, 5\%)$。

概率密度函数为：

$$f_{P_2}(p) = \frac{10}{\hat{Product}_{i,s,2023}} \tag{31}$$

概率分布函数为：

$$F_{P_2}(p) = \int_{p_{\min}}^{p_{\max}} f_{P_2}(p)dp = \frac{10}{\hat{Product}_{i,s,2023}} (p - 0.9 \cdot Product_{i,s,2023}) \tag{32}$$

p 的取值范围如下：

$$0.95 Product_{i_2,s,2023} = p_{min} \leq p \leq 1.05 Product_{i_2,s,2023} = p_{max}$$

记作：$p_3 \leq p \leq p_4$。

2. **亩产量**：

一般来说，农作物的亩产量会受气候等因素的影响，每年会有 ±10% 的变化。考虑到现实情况，我们认为，每年 ±10% 的变化可以认为是与 2023 年的情况对比：

$$Yield'_{i,j,s,year} = (1 + \alpha_3) \cdot Yield_{i,j,s,2023} \tag{33}$$

其中，农作物亩产量的波动范围 $\alpha_3 \sim U(-10\%, 10\%)$。

概率密度函数为：

$$f_Y(y) = \frac{5}{Yield_{i,j,s,2023}} \tag{34}$$

y 的取值范围如下：

$$0.95Yield_{i,j,s,2023} \leq y \leq 1.05Yield_{i,j,s,2023}$$

总产量与亩产量之间存在如下关系：

$$Product'_{i,j,s,year} = \sum_{j=1}^{54}(X_{ij,s,year} \cdot Yield'_{i,j,s,year})$$

故，总产量 $Product'_{i,j,s,year}$ 的概率密度函数为：

$$f_Z(z) = 5 \cdot \sum_{j=1}^{54}(X_{ij,s,year} \cdot Yield_{i,j,s,year})' \tag{35}$$

z 的取值范围如下：

$$0.9\sum_{j=1}^{54}(X_{i,j,s,year} \cdot Yield'_{i,j,s,year}) \leq z \leq 1.1\sum_{j=1}^{54}(X_{i,j,s,year} \cdot Yield'_{i,j,s,year})$$

记为 $z_1 \leq z \leq z_2$。

3. **种植成本**：

受到市场条件影响，农作物的种植成本平均每年增长 5% 左右。因此：

$$Cost'_{i,j,s,year} = (1 + 5\%)^{(year-2023)} \cdot Cost_{i,j,s,2023} \tag{36}$$

4. **销售单价**：

（1）粮食类作物的销售价格基本稳定，下公式中的 $i = 1, 2, \ldots, 16$：

$$Price'_{i,s,year} = Price_{i,s,2023} \tag{37}$$

（2）蔬菜类作物的销售价格以平均每年增长 5% 左右的趋势上升，下公式中的 $i = 17, \ldots, 37$：

$$Price'_{i,s,year} = (1 + 5\%)^{(year-2023)} \cdot Price_{i,s,2023} \tag{38}$$

（3）羊肚菌的销售价格每年下降幅度为 5%：

$$Price'_{41,s,year} = (1-5\%)^{(year-2023)} \cdot Price_{41,s,2023} \tag{39}$$

（4）其他食用菌的销售价格稳中有降，大约每年可下降 1%~5%，公式（37）中的 $i = 38, 39, 40$：

$$Price'_{i,s,year} = (1+\alpha_4)^{(year-2023)} \cdot Price_{i,s,2023} \tag{40}$$

其中，$\alpha_4 \sim U(-5\%, -1\%)$，因此：$Price'_{i,s,year}$ 的概率密度函数如下：

$$f_X(x) = \frac{25}{Price_{i,s,2023}(year-2023)} x^{\frac{1}{year-2023}-1} \tag{41}$$

x 的取值范围为：

$$0.95^{year-2023} Price_{i,s,2023} \leq x \leq 0.99^{year-2023} Product_{i,s,2023}$$

7.2.2 定义优化目标

在明确了本文中的**变化量**和**确定量**的基本特征与分布情况后，我们开始确定优化目标。

首先，我们引入**变异系数（Coefficient of Variation, CV）**的概念：其通过标准差与均值的比值来衡量数据的相对变异程度，是一个用于衡量数据集离散程度的统计指标。

$$CV = \frac{\sigma}{\mu} \times 100\%$$

CV 为我们提供了一种更为普适的变异度量方式，可以帮助我们更直观地理解每个数据集的相对波动性和离散程度。其在金融领域、生物医学等领域都有应用。

特别地，在收益和风险的分析中，CV 可以用于**衡量单位风险**，在本题中具体体现为 CV 较小时，则说明它在获取一定收益时，其波动性或风险较小。反之，CV 较高则意味着在该种植策略下，需要承担更大的种植风险性和不确定性。

由此，我们考虑到，将 CV 引入我们的优化目标：一方面，我们希望总收益的均值 μ 尽可能大；另一方面，我们希望总收益的离散程度 CV 较小，这样，即可以实现在**较小的波动范围和风险程度下实现尽量大的收益**。

故，我们的优化目标如下：

$$\max \frac{\mu_w}{CV_w} \tag{42}$$

即最大化收益均值与收益变异系数的比值，其中：

$$CV_w = \frac{\sigma_w}{\mu_w} \times 100\% \tag{43}$$

从单位风险的角度，我们再次解释优化目标：它反映了**承担一单位风险所能获得的平均收益**，这与金融学中的**夏普比率（Sharpe Ratio）**有一定的相似性。这个优化目标可以帮助我们在风险与收益之间找到最佳平衡点，优化种植方案的选择。

7.2.3 计算相关函数

接下来，我们开始计算 $W_{i,j,s,year}$ 的**概率密度函数**，并以此计算 μ_w 和 σ_w。
某年某季度某种作物利润的表达式如下：

$$W_{i,s,year} = Sale'_{i,s,year} \cdot Price'_{i,s,year} - \sum_{j=1}^{54}(X_{i,j,s,year} \cdot Cost'_{i,j,s,year}) \tag{44}$$

收益均值 $\mu_{W_{i,s,year}}$ 的计算公式如下：

$$\mu_{W_{i,s,year}} = \int_{W_{min}}^{W_{max}} w \cdot f_{W_{i,s,year}}(w)\, dw \tag{45}$$

收益方差 $\sigma_{W_{i,s,year}}$ 的计算公式如下：

$$\sigma_{W_{i,s,year}} = \sqrt{\int_{W_{min}}^{W_{max}} w^2 \cdot f_{W_{i,s,year}}(w)\, dw - \left(\int_{W_{min}}^{W_{max}} w \cdot f_{W_{i,s,year}}(w)\, dw\right)^2} \tag{46}$$

根据上述公式，我们计算出总利润 W、总利润的均值 μ_W 和总利润的方差 σ_W：

$$\begin{cases} W = \sum_{year=2024}^{2030} \sum_{s=1}^{2} \sum_{i=1}^{41} W_{i,s,year} \\ \mu_W = \sum_{year=2024}^{2030} \sum_{s=1}^{2} \sum_{i=1}^{41} \mu_{W,i,s,year} \\ \sigma_W = \sqrt{\sum_{year=2024}^{2030} \sum_{s=1}^{2} \sum_{i=1}^{41} \sigma^2_{W,i,s,year}} \end{cases} \tag{47}$$

根据 $W_{i,s,year}$ 的计算公式，我们还需要求解 $Sales'_{i,s,year}$ 的概率密度函数 $f_{Sale'_{i,s,year}}(x)$：
首先，延续问题一的第一种情况：我们得到 $Sales'_{i,s,year}$ 的公式：

$$Sale'_{i,s,year} = \begin{cases} Product'_{i,s,year} & Product'_{i,s,year} \leq \hat{Product}'_{i,s,year} \\ \hat{Product}_{i,s,year} & Product'_{i,s,year} > \hat{Product}'_{i,s,year} \end{cases} \tag{48}$$

我们设定：

$$Product'_{i,s,year} \in (min_1, max_1), \quad \hat{Product}'_{i,s,year} \in (min_2, max_2)$$

于是有：

$$\begin{cases} min_1 = z_1, max_1 = z_2, min_2 = p_1, max_2 = p_2 & i = 6, 7 \\ min_1 = z_1, max_1 = z_2, min_2 = p_3, max_2 = p_4 & i \neq 6, 7 \end{cases} \tag{49}$$

因此，针对 min_1，min_2，max_1，max_2 的不同情况，分成以下六种情况讨论：

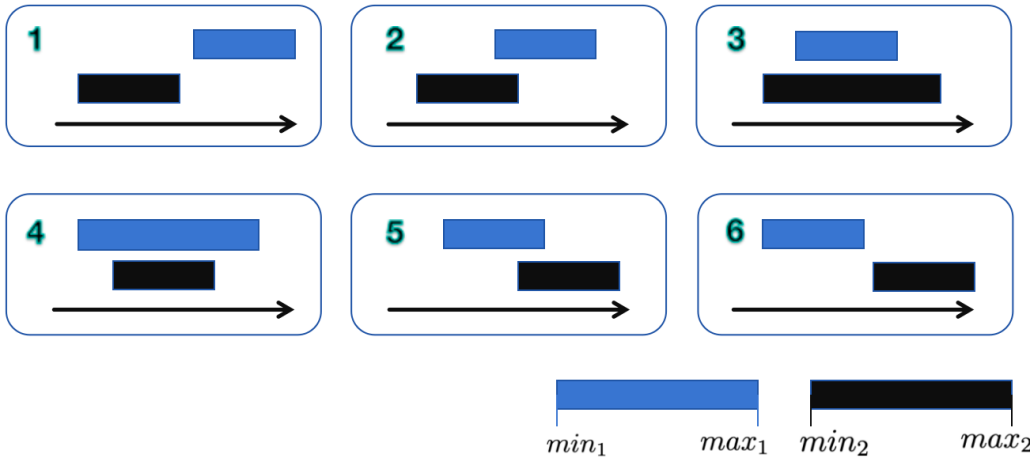


图 8　六种情况示意图

1. $min_1 \geq max_2$：

$$f_{Sale'_{i,s,year}}(x) = f_p(x), \quad min_2 \leq x \leq max_2 \tag{50}$$

2. $min_2 < min_1 < max_2 \leq max_1$：

$$f_{Sale'_{i,s,year}}(x) = \begin{cases} f_p(x) & min_2 \leq x \leq min_1 \\ f_z(x)[1 - F_p(x)] + f_z(x)[1 - F_p(x)] & min_1 < x \leq max_2 \end{cases} \tag{51}$$

3. $min_2 < min_1 < max_1 < max_2$：

$$f_{Sale'_{i,s,year}}(x) = \begin{cases} f_p(x), & min_2 \leq x \leq min_1 \\ f_z(x)[1 - F_p(x)] + f_z(x)[1 - F_p(x)], & min_1 < x \leq max_1 \end{cases} \tag{52}$$

4. $min_1 \leq min_2 \leq max_1 \leq max_2$：

$$f_{Sale'_{i,s,year}}(x) = \begin{cases} f_z(x), & min_1 \leq x \leq min_2 \\ f_z(x)[1 - F_p(x)] + f_z(x)[1 - F_p(x)], & min_2 < x \leq max_2 \end{cases} \tag{53}$$

5. $min_1 \leq min_2, min_2 < max_1 < max_2$：

$$f_{Sale'_{i,s,year}}(x) = \begin{cases} f_z(x), & min_1 \leq x \leq min_2 \\ f_z(x)[1 - F_p(x)] + f_z(x)[1 - F_p(x)], & min_2 < x \leq max_1 \end{cases} \tag{54}$$

6. $max_1 \leq min_2$：

$$f_{Sale'_{i,s,year}}(x) = f_z(x), \quad min_1 \leq x \leq max_1 \tag{55}$$

最后，我们利用求得的 $f_{Sale'_{i,s,year}}(x)$ 来计算利润 $W_{i,s,year}$ 的概率密度函数：

1. 当 $i = 1, 2, \ldots, 36, 37, 41$ 时：

$$f_{W_{i,s,year}}(w_{i,s,year}) = f_{Sale'_{i,s,year}}(Sale'_{i,s,year}) \cdot \frac{1}{Price'_{i,s,year}} \tag{56}$$

2. 当 $i=38,39,40$ 时，由 $Z=XY$ 时，$f_Z(z)=\int_{-\infty}^{+\infty} f_X(x)\cdot f_Y\left(\frac{z}{x}\right)\cdot\frac{1}{|x|}dx$ 可得：

$$\begin{cases} f_{W_{i,s,year}}(w_{i,s,year})=\int_{Sale'_{min}}^{Sale'_{max}} A\cdot B\cdot\frac{1}{Sale'_{i,s,year}}dSale'_{i,s,year} \\ A=f_{Sale'_{i,s,year}}(Sale'_{i,s,year}) \\ B=f_{Price'_{i,s,year}}(Price'_{i,s,year}-\frac{\sum_{j=1}^{54}(X_{i,j,s,year}\cdot Cost'_{i,j,s,year})}{Sale'_{i,s,year}}) \end{cases} \tag{57}$$

7.3 确定约束条件

由于问题二在种植方案要求方面**与问题一一致**，因此问题二的约束条件与问题一的约束条件（6.3 节）一致，即式（10）~（26）。

7.4 优化模型汇总

综上所述，种植方案优化模型最终汇总如下：

$$\begin{gathered} \max\frac{\mu_w}{CV_w} \\ st.\text{式}(10)\sim(26) \end{gathered} \tag{58}$$

7.5 模型求解

上述模型建立后，我们可以由确定的 $X_{i,1,s,year},X_{i,2,s,year},\ldots,X_{i,54,s,year}$，在预期销售量、亩产、种植成本、销售价格波动且未知具体数值的情况下，对 μ_w 与 σ_w 进行求解。

但在实际操作的过程中，我们发现：由于引入了大量积分与求导运算，无法快速地利用遗传算法得到较好的规划结果。因此，我们引入**蒙特卡洛**方法对上述模型的实际运算效率进行优化。

蒙特卡洛方法是一种统计学上的随机模拟技术，基于大量随机样本来估计未知量，利用随机数（或更常见的是伪随机数）来模拟复杂系统的行为或计算数学表达式的数值。其在解决复杂的多维问题上，比传统的数值方法更有效。

就具体操作而言：我们已知 $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ 都服从均匀分布，则我们分别取其区间中的**随机值**作为 $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ 的定值，再分别代入式（27）、（33）、（36）和（40）中求得 $Product'_{i,s,year}$，$Yield'_{i,j,s,year}$，$Cost'_{i,j,s,year}$ 和 $Price'_{i,s,year}$ 的具体数值。最后，再根据 $X_{i,j,s,year}$ 求得 W。

最终，考虑到实际运行效率与精度问题，我们对于 1 组 $X_{i,j,s,year}$ 设定 100 组随机的 $\alpha_1,\alpha_2,\alpha_3,\alpha_4$，求解得到 $W_1,W_2,\ldots,W_{100}$，最终得到目标函数的无偏估计值。

7.6 求解结果

问题二最终解得目标函数最大值为 76.85 亿元，即承担一单位风险获得的最大平均收益为 76.85 亿。最优种植方案对应的 100 次蒙特卡洛模拟得到的总利润，样本**均值为**

2932.48 万元，样本**标准差为 111893 元**。

八、 问题三的模型建立与求解

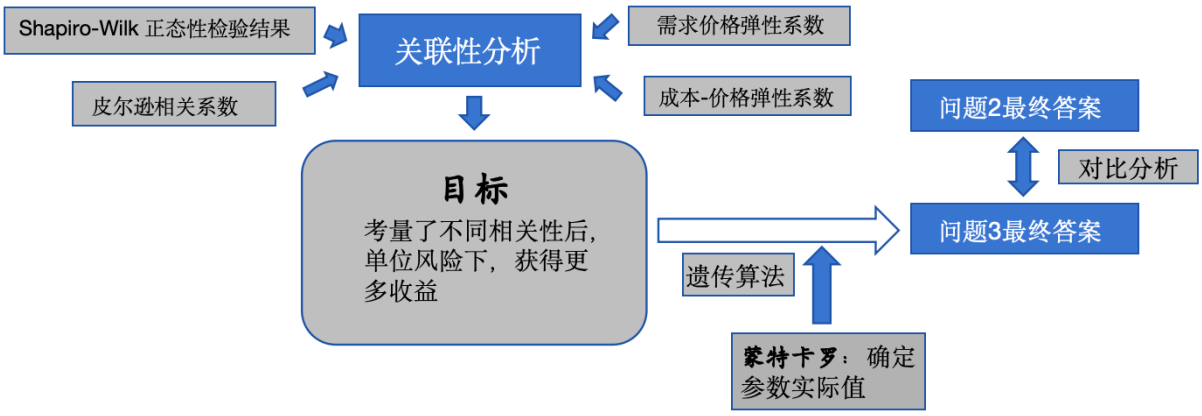


图 9　问题三思路流程图

8.1 关联性分析

通过查阅文献以及解读题设条件，我们引入各种农作物之间存在的**可替代性**和**互补性**、种植成本和销售价格之间的相关性以及种植成本和预期销售量之间的关联性，使模型更具有普适性以及现实意义。

8.1.1 种植成本和销售价格存在一定相关性

文献表明，研究者通过 2001 年 1 月-2013 年 6 月的数据测算出农业生产成本对农产品价格的弹性系数在 0.0001-0.884 之间变动，均值为 0.427，标准差为 0.197[3]。具体来说，**农业生产成本每增加 1%，会导致农产品价格平均增加 0.427%**。

根据上述内容，我们得出变异系数：$CV = \frac{\sigma}{\mu} \times 100\% = 46.13\%$，接近 50%，说明弹性系数的离散程度较高。故，我们假定：农业生产成本每增加 1%，会导致农产品价格平均增加 θ%。其中，$\theta \sim N(0.427, 0.197)$。

8.1.2 销售价格和预期销售量存在一定相关性

针对销售价格和预期销售量之间的关联性，研究者发现：以谷物蔬菜为例，它们是人们生活的必需品，其需求量并不会因为价格变化而产生明显的变化。他们通过 2005-2017 年的数据测算出：粮食和蔬菜自美国和加拿大的进口价格每增加 1%，则从美国和加拿大进口蔬菜的需求量均平均减少 0.3% 左右，而水果自美国和加拿大的进口价格每增加 1%，则从美国和加拿大进口蔬菜的需求量均平均减少 1.7% 左右。[4]

结合**需求价格弹性系数** $PED = \frac{\Delta Q/Q}{\Delta P/P}$，其中：$\Delta Q$ 是需求量的变化量，Q 是原始需求量，ΔP 是价格的变化量，P 是原始价格。即农产品价格增加 1%，会导致预期销售量平均变化 PED%。

我们设定：本题中粮食、蔬菜需求价格弹性系数为-0.3，又考虑到食用菌的必需性介于粮食、蔬菜与水果之间，故食用菌需求价格弹性系数为-1。

8.1.3 各种农作物之间可能存在一定的可替代性和互补性

研究者们发现：可替代农产品在双对数序列存在较强的相关性和同向变动关系。以猪肉和鸡蛋为例，二者的交叉弹性为 0.56，即猪肉价格上涨或下降 1 个百分点, 鸡蛋价格会相应变动 0.56% 左右 [5]。

基于以上研究，可得出可替代农产品在价格百分比变动上存在一定的相关性的结论。因此，我们类比猜测在农作物领域也存在类似的结论。故，我们假定同季度同类农作物的价格变化总倍数为 $x_1, x_2, \ldots, x_i$，它们是从同一正态总体 $N \sim (\mu, \sigma)$ 中抽取的。同时，我们采用 **Shapiro-Wilk 正态性检验**判断上述数列的正态性，以此来评价蒙特卡洛方法生成的数据是否符合同类农作物之间的可替代性和互补性。

Shapiro-Wilk 正态性检验是统计学中用于检验样本数据是否来自正态分布的一种方法, 尤其适用于小样本数据, 本例中样本数分别为 16, 18, 21, 4, 样本较小。Shapiro-Wilk 检验的原理是通过比较样本数据的排序值与正态分布理论下的期望排序值之间的差异, 来判断数据是否服从正态分布。

原假设 H0：数据服从正态分布。

备择假设 H1：数据不服从正态分布。

8.2 确定决策变量

由题意知，本文中的决策变量为：在问题二的基础上，综合考虑各种农作物之间的可替代性和互补性，预期销售量与销售价格、种植成本两两之间的相关性的情况下，种植成本和销售价格的不确定性以及潜在的种植风险时，即：

$$X_{i,j,s,year} \quad s = 1, 2, j = 1, 2, \ldots, 54, i = 1, 2, \ldots, 41, year = 2024, 2025, \ldots 2030$$

8.3 确定优化目标

与问题二类似，在确定优化目标前，我们首先明确本文中的**变化量**与**确定量**，它们也分别出现在预期销售量、亩产量、种植成本和销售单价中，下面将分别逐个讨论：

1. **销售量**

(1) 小麦和玉米未来的预期销售量有增长的趋势，并且存在需求价格弹性系数 PED，下面公式中的 $i = 5, 6$：

$$Product''_{i,s,year} = (1 + \alpha_1)^{(year-2023)}(1 + \lambda \cdot PED\%) \cdot \hat{Product}_{i,s,2023} \tag{59}$$

其中，小麦和玉米预期销售量的增长率 $\alpha_1 \sim U(5\%, 10\%)$。

（2）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，也需要考虑到需求价格弹性系数 PED 的影响，下面公式中 $i \neq 5, 6$：

$$\hat{Product}''_{i,s,year} = (1 + \alpha_2)(1 + \lambda \cdot PED) \cdot \hat{Product}_{i,s,2023} \tag{60}$$

其中，λ 表示价格的变化率，其他农作物预期销售量的波动范围 $\alpha_2 \sim U(-5\%, 5\%)$

2. **亩产量**：亩产量的计算与问题二中的亩产量计算一致：

$$Yield''_{i,j,s,year} = (1 + \alpha_3) \cdot Yield_{i,j,s,2023} \tag{61}$$

其中，农作物亩产量的波动范围 $\alpha_3 \sim U(-10\%, 10\%)$。

3. **种植成本**：

种植成本的计算与问题二中的种植成本计算一致：

$$Cost''_{i,j,s,year} = (1 + 5\%)^{(year-2023)} \cdot Cost_{i,j,s,2023} \tag{62}$$

4. **销售单价**：

实际上，销售单价的计算与问题二中的计算基本一致，但需要考虑 8.1.1 节中提出的种植成本和销售价格之间的相关性。

（1）粮食类作物，$i = 1, 2, \ldots, 16$：

$$Price''_{i,s,year} = (1 + 5\theta\%)Price_{i,s,2023} \tag{63}$$

（2）蔬菜类作物，$i = 17, \ldots, 37$：

$$Price''_{i,s,year} = (1 + 5\%)^{(year-2023)}(1 + 5\theta\%) \cdot Price_{i,s,2023} \tag{64}$$

（3）羊肚菌：

$$Price''_{41,s,year} = (1 - 5\%)^{(year-2023)}(1 + 5\theta\%) \cdot Price_{41,s,2023} \tag{65}$$

（4）其他食用菌，$i = 38, 39, 40$：

$$Price''_{i,s,year} = (1 + \alpha_4)^{(year-2023)}(1 + 5\theta\%) \cdot Price_{i,s,2023} \tag{66}$$

其中，$\alpha_4 \sim U(-5\%, -1\%)$。

在确定了需要的变化量以及确定量后，我们开始定义优化目标。回顾题意，问题三的优化目标应该与问题二一致，都希望在**尽可能小的风险程度**下实现**尽可能大的利润**。即，本文的优化目标为：

$$\max \frac{\mu_W}{CV_W} \tag{67}$$

8.4 确定约束条件

由于问题三在种植方案要求方面与问题一一致，因此问题三的约束条件与问题一的约束条件（6.3 节）一致，即式（10）~（26）。

8.5 优化模型汇总

综上所述，种植方案优化模型最终汇总如下：

$$\begin{aligned} &\max \frac{\mu_w}{CV_w} \\ &st.\text{式}(10) \sim (26) \end{aligned} \tag{68}$$

8.6 模型求解

延续问题二基于蒙特卡洛优化本模型的思路：已知 $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ 服从均匀分布，θ 服从正态分布，则分别根据其分布取区间中的随机值作为 $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta$ 的定值，代入式（60）（66），可得 $\hat{Product}''_{i,s,year}, Yield''_{i,j,s,year}, Cost''_{i,j,s,year}, Price''_{i,s,year}$ 的具体数值，进而由 $X_{i,j,s,year}$ 求解 W。考虑到实际运行效率与最终精度，我们设定对一组 $X_{i,j,s,year}$ 取一百组随机的 $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta$，最终求解得到 $W_1, W_2, \ldots, W_{100}$，最终得到目标函数的无偏估计值。

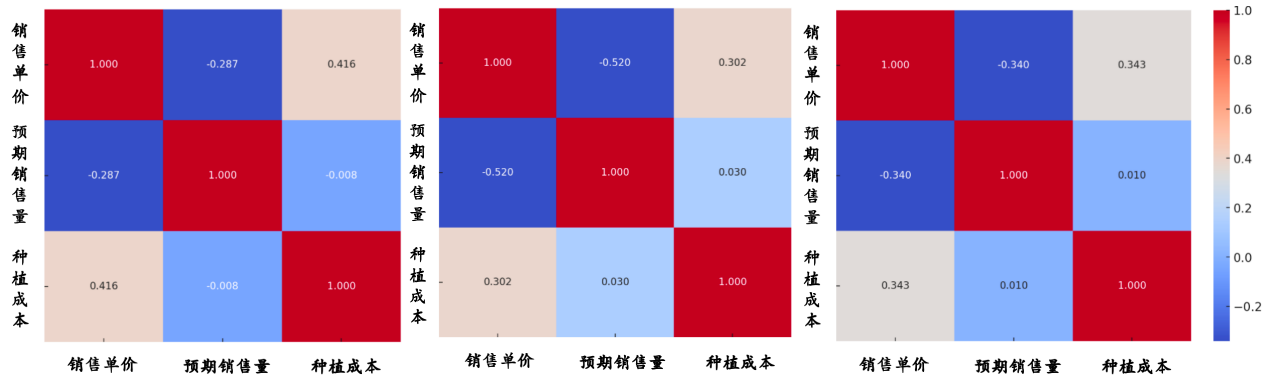
8.7 结果分析

8.7.1 相关性分析结果

表 2　各年份不同种作物 Shapiro-Wilk 正态性检验结果

年份	粮食 p 值	一季度蔬菜 p 值	二季度蔬菜 p 值	食用菌 p 值
2024	0.3951	0.7049	0.9352	0.8835
2025	0.4991	0.3802	0.6999	0.5484
2026	0.6214	0.6025	0.9973	0.5043
2027	0.5019	0.4414	0.6078	0.2555
2028	0.5298	0.8219	0.7192	0.4594
2029	0.7955	0.9603	0.7240	0.2359
2030	0.9222	0.8771	0.8071	0.5779

上表是一组蒙特卡洛模拟结果中粮食、一季度蔬菜、二季度蔬菜、食用菌的价格增长率 Shapiro-Wilk 正态性检验结果，所有年份所有类别的农作物在置信水平为 90%，95%，99% 的情况下，均能通过正态性检验，即在 90%，95%，99% 的可能性下，均不能拒绝原假设：**数据服从正态分布**。



(a) 粮食 **(b) 蔬菜** **(c) 菌类**

图 10　各作物预期销售量、销售单价以及种植成本 Pearson 相关系数

上图是一组蒙特卡洛模拟数据的相关性分析结果图，描述了粮食、蔬菜和食用菌类别中预期销售量、销售单价以及种植成本的线性相关性。从**指标角度**而言，种植成本和预期销售量之间的线性相关性极弱，销售单价与种植成本之间存在一定正向线性相关性，销售单价与预期销售量之间存在一定负向线性相关性。从**农作物角度**而言，粮食的销售单价、预期销售量、种植成本三者之间线性相关性最弱，蔬菜和菌类三者之间的线性相关性近似，符合现实情况，即粮食的需求量和价格的变化程度都较小，相较于蔬菜和菌类更加稳定。

需要注意的是，销售单价、预期销售量、种植成本均与时间等其他因素有关，且**时序的趋势不同**。因此 Pearson 相关系数较低并不能断定该模型对销售单价、预期销售量、种植成本之间相关性的模拟能力差，而可能是由于其他因素对三者的影响更大，且方向不同。

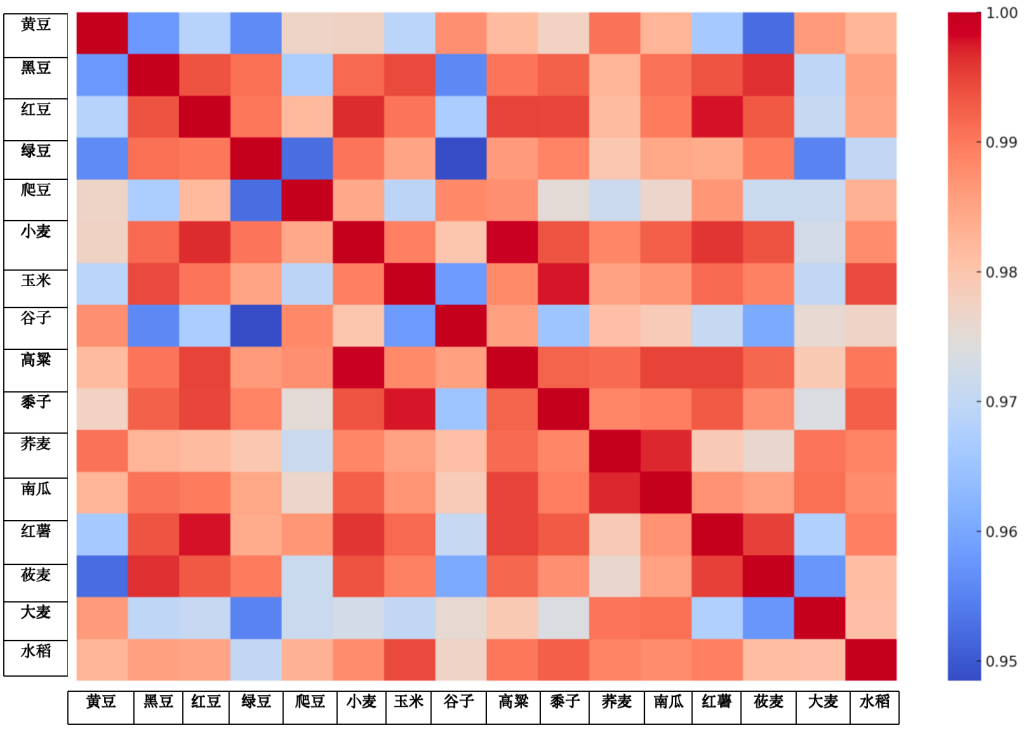


图 11　粮食价格之间 Pearson 相关系数热力图

上图是一组蒙特卡洛模拟数据的相关性分析结果图，描述了粮食类别中不同农作物价格之间的线性相关性。粮食价格之间的 Pearson 相关系数均大于 0.95，说明基于 Shapiro-Wilk 正态性检验对蒙特卡洛模拟数据进行筛选的方法，**充分模拟了各种农作物之间存在的可替代性和互补性**。

8.7.2 问题三与问题二答案对比分析

表 3　结果描述性统计

问题	极大值	极小值	均值	标准差	偏度	峰度
2	29665210.18	29039413.75	29324796.8	111893.0542	0.114469589	0.185146098
3	32970804.78	32072359.17	32479639	184230.4894	0.148931627	-0.129690522

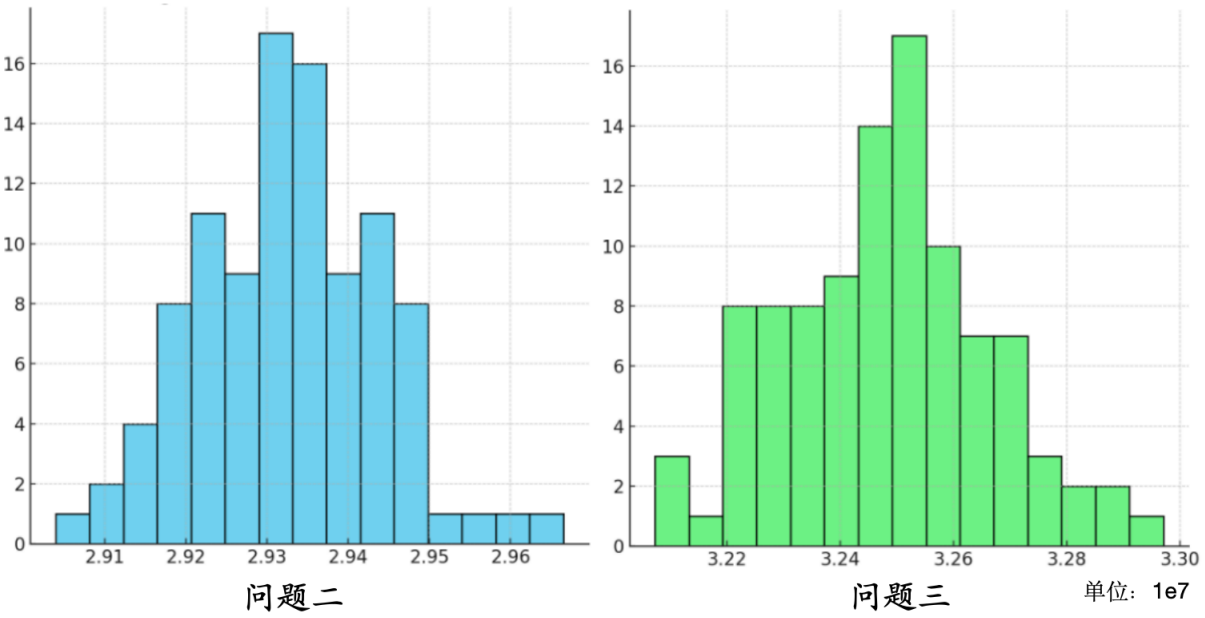


图 12　结果直方图

由结果描述性统计与直方图可看出，问题二与三蒙特卡洛模拟最优解的 100 个总利润均符合右偏分布，二者在峰度上都**近似于正态分布**。问题二结果的均值较低，但极差和标准差明显**小于**问题三的结果，波动性的影响大于收益性影响，因此目标函数 $\frac{\mu_w}{CV_w}$ 较大，即承担一单位风险所能获得的平均收益更大。

九、 模型评价

9.1 模型的优点

1. 问题一中使用贪心算法对传统的遗传算法进行改进，**缩减了参数量**，从而实现了遗传算法的**加速收敛，提升了效率**；
2. 问题一中罗列的约束条件考虑了便于耕种和田间管理的要求，避免了耕地细碎化等问题，使我们的模型更具有**现实意义**；
3. 问题二中我们利用**蒙特卡洛**方法优化了目标函数的求解过程，兼顾了**运算速度和目标函数准确性**；
4. 问题三的模型中，我们引入了**成本价格弹性系数**和**需求价格弹性系数**，使模型更贴合现实生活且具有**鲁棒性**。

9.2 模型的缺点

1. 模型假设了“取作物销售单价区间的平均值作为作物销售单价”，实际上应该考虑作物销售单价在对应的范围内呈现一定的分布。

参考文献

[1] 罗丹, 徐艳, 王跃朋, 等. 基于地块面积的土地整理耕作效率测算方法研究 [J]. 中国土地科学,2013(6):73-78.

[2] Ndip, Francis Ebai, Ernest L. Molua, Meyo-Elise Stephanie Mvodo, Robert Nkendah, Raoul Fani Djomo Choumbou, Rayner Tabetando, and Nina Fabinin Akem. Farmland Fragmentation, Crop Diversification and Incomes in Cameroon, a Congo Basin Country. Land use policy 130 (2023): 106663.

[3] 付莲莲. 国内农产品价格波动影响因素的结构及动态演变机制 [D]. 江西: 南昌大学,2014.

[4] 彭虹. 基于双对数模型的中国农产品进口需求价格弹性分析 [J]. 发展研究,2019(8):72-81.

[5] 赵一夫, 秦富. 我国鸡蛋价格变动特点及规律分析 [J]. 农业技术经济,2013(1):4-10.

附录 A　支撑材料列表

文件名	类型	简介
cal_profit1-1.py	python 代码文件	问题一第（1）小问计算利润代码
cal_profit1-2.py	python 代码文件	问题一第（2）小问计算利润代码
cal_profit2.py	python 代码文件	问题二计算利润代码
pre_deal.py	python 代码文件	预处理代码
problem1-1.py	python 代码文件	问题一第（1）小问求解代码
problem1-2.py	python 代码文件	问题一第（2）小问求解代码
problem2and3.py	python 代码文件	问题二和问题三求解代码
result1_.$xlsx$	xlsx 文件	问题一第（1）小问答案
result1_.$xlsx$	xlsx 文件	问题一第（2）小问答案
result2.xlsx	xlsx 文件	问题二小问答案
附件 1.xlsx	xlsx 文件	原附件 1 略微改动结果
附件 2.xlsx	xlsx 文件	原附件 2 略微改动结果
2023-2030 亩产量 + 种植成本.csv	csv 文件	亩产量与成本变化
2023-2030 预期销售量 + 销售单价 1.csv	csv 文件	预期销售量与单价变化
2023 蔬菜不同地块种植情况及利润.csv	csv 文件	种植情况与利润
Q2data	文件夹	问题二中蒙特卡洛模拟结果
Q3data	文件夹	问题三中蒙特卡洛模拟结果

附录 B　源程序代码

cal_profit1-1.py:

```
# %%
# coding=gbk
import pandas as pd
```

```
import numpy as np

file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])

# %%
land_plots = file1['地块名称'].unique()# 地块
crops = csv_data['作物编号'].unique()# 作物
land_type_mapping = file1[['地块名称', '地块类型']].set_index('地块名称').to_dict()['地块类型']

# 创建索引
land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
crop_indices = {crop: idx for idx, crop in enumerate(crops)}



# 四维数组 Cost_ijsyear（表示种植成本），形状为（作物编号数41，地块数54，2个季次，7年）
Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 7))
Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 7))

# 把2023年的作为初值
X_Init=np.zeros((len(crops), len(land_plots), 2, 7))
for idx, row in file2.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_plot = row['种植地块']
    if land_plot in land_plot_indices:
        plot_idx = land_plot_indices[land_plot]
        season = 1 if row['种植季次'] == '第二季' else 0
        area = row['种植面积/亩']
        X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变
# %%
# 三维数组 Except_Product_isyear（表示预期销售量），形状为（作物编号数41，2个季次，7年）
Except_Product_isyear = np.zeros((len(crops), 2, 7))
# 填充四维数组 Cost_ijsyear和Yield
for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if plot_type
        == land_type]
```

```
    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            cost = row['种植成本/(元/亩)']
            yield_=row['亩产量/斤']
            Cost_ijsyear[crop_idx, plot_idx, season, :] = cost # 成本每年不变
            Yield_ijsyear[crop_idx, plot_idx, season, :] = yield_# 每年亩产量不变
# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤': 'sum'}).reset_index()
grouped
# %%
for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    Except_Product = row['预期销售量/斤']
    Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变


Except_Product_isyear


# %%

# 第一维是地块个数54,第二维是季次数 2,第三维是7年
S_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)# 销售单价

# 按行遍历填充地块
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变


for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    if row['种植季次'] == '单季' or row['种植季次']== '第一季':
        Price_isyear[crop_idx, 0, :] = row['销售单价/(元/斤)']
    if row['种植季次']== '第二季':
        Price_isyear[crop_idx, 1, :] = row['销售单价/(元/斤)']


S_jsyear, Price_isyear
# %%
```

```
# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year

# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=7):
    return i * (s_dim * year_dim) + s * year_dim + year



# 得到索引
def getXindex(i,j, s , j_dim=54 , s_dim=2):
    return i * (j_dim*s_dim) + j * (s_dim) + s

# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year
# %%


i_dim=41
j_dim=54
s_dim=2
year_dim=7

S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量
#Yield_ijsyear[16,:,:,:]
# %%

def cal_each_year_profit(xi,yi=0):
    total_profit = 0
    total_sale=0
    total_cost=0
    yi=0
    for ii in range(i_dim):
        for si in range(s_dim):
            cost_total=0# 该作物该季总成本
            product_total=0# 该作物该季总产量(斤)
            for ji in range(j_dim):# 遍历所有地
                product_total+=xi[getXindex(ii,ji,si)]*Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                cost_total+=xi[getXindex(ii,ji,si)]*Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
```

```
                total_cost+=xi[getXindex(ii,ji,si)]*Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
            #求该作物该季度实际销售量（斤）
            if product_total<=sales:
                sales=product_total
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
    total_profit=total_sale-total_cost
    return total_profit

# %%


import pandas as pd
import numpy as np
# 读取Excel文件
file_path = './result1_2.xlsx'
excel_data = pd.ExcelFile(file_path)

for sum_i in range(7):

    # 读取表格数据
    df = pd.read_excel(excel_data, sheet_name=str(sum_i+2024))
    df = df.drop(df.columns[0:2], axis=1).fillna(0)
    df =df.drop(df.index[82:86]).fillna(0)
    df
    # %%
    # 去掉第一列，并将空值替换为0
    # 获取第一季次数据（前54行），并转换为三维数组
    X_ijs=np.zeros((41, 54, 2))
    for idx, row in df.iterrows():
        if idx>=54:
            for i in range(len(row)):
                X_ijs[i][idx-54+26][1]=row.iloc[i]
            continue
        for i in range(len(row)):
            X_ijs[i][idx][0]=row.iloc[i]
    # %%
    X_ijs_flat=X_ijs.flatten()
    profit=cal_each_year_profit(xi=X_ijs_flat)
    print(profit)
```

cal_profit1-2.py:

```
    # %%
# coding=gbk
import pandas as pd
```

```
import numpy as np
file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])

# %%
land_plots = file1['地块名称'].unique()# 地块
crops = csv_data['作物编号'].unique()# 作物
land_type_mapping = file1[['地块名称', '地块类型']].set_index('地块名称').to_dict()['地块类型']

# 创建索引
land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
crop_indices = {crop: idx for idx, crop in enumerate(crops)}



# 四维数组 Cost_ijsyear (表示种植成本), 形状为 (作物编号数41, 地块数54, 2个季次, 7年)
Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 7))
Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 7))

# 把2023年的作为初值
X_Init=np.zeros((len(crops), len(land_plots), 2, 7))

for idx, row in file2.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_plot = row['种植地块']
    if land_plot in land_plot_indices:
        plot_idx = land_plot_indices[land_plot]
        season = 1 if row['种植季次'] == '第二季' else 0
        area = row['种植面积/亩']
        X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变

# %%

# 三维数组 Except_Product_isyear (表示预期销售量), 形状为 (作物编号数41, 2个季次, 7年)
Except_Product_isyear = np.zeros((len(crops), 2, 7))

# 填充四维数组 Cost_ijsyear和Yield
for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
```

```
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if plot_type
        == land_type]

    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            cost = row['种植成本/(元/亩)']
            yield_=row['亩产量/斤']
            Cost_ijsyear[crop_idx, plot_idx, season, :] = cost # 成本每年不变
            Yield_ijsyear[crop_idx, plot_idx, season, :] = yield_# 每年亩产量不变

# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤': 'sum'}).reset_index()
grouped
# %%
for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    Except_Product = row['预期销售量/斤']
    Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变


Except_Product_isyear


# %%

# 第一维是地块个数54,第二维是季次数 2,第三维是7年
S_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)# 销售单价

# 按行遍历填充地块
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变


for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    if row['种植季次'] == '单季' or row['种植季次']== '第一季':
        Price_isyear[crop_idx, 0, :] = row['销售单价/(元/斤)']
    if row['种植季次']== '第二季':
        Price_isyear[crop_idx, 1, :] = row['销售单价/(元/斤)']
```

```
S_jsyear, Price_isyear
# %%


# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year

# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=7):
    return i * (s_dim * year_dim) + s * year_dim + year



# 得到索引
def getXindex(i,j, s , j_dim=54 , s_dim=2):
    return i * (j_dim*s_dim) + j * (s_dim) + s

# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year
# %%


i_dim=41
j_dim=54
s_dim=2
year_dim=7

S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量
#Yield_ijsyear[16,:,:,:]
# %%

def cal_each_year_profit(xi,yi=0):
    total_profit = 0
    total_sale=0
    total_cost=0
    yi=0
    for ii in range(i_dim):
        for si in range(s_dim):
```

```
            cost_total=0# 该作物该季总成本
            product_total=0# 该作物该季总产量(斤)
            for ji in range(j_dim):# 遍历所有地
                product_total+=xi[getXindex(ii,ji,si)]*
                Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]

                cost_total+=xi[getXindex(ii,ji,si)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                total_cost+=xi[getXindex(ii,ji,si)]*Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]

            #求该作物该季度实际销售量（斤）
            if product_total<=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]:
                sales=0.5*(product_total+Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)])
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
    total_profit=total_sale-total_cost
    return total_profit # 遗传算法求最小化问题，因此返回负值

# %%


import pandas as pd
import numpy as np
# 读取Excel文件
file_path = './result1_2.xlsx'
excel_data = pd.ExcelFile(file_path)

for sum_i in range(7):
    # 读取表格数据
    df = pd.read_excel(excel_data, sheet_name=str(sum_i+2024))
    df = df.drop(df.columns[0:2], axis=1).fillna(0)
    df =df.drop(df.index[82:86])
    df
    # %%
    # 去掉第一列，并将空值替换为0

    # 获取第一季次数据（前54行），并转换为三维数组
    X_ijs=np.zeros((41, 54, 2))
    for idx, row in df.iterrows():
        if idx>=54:
            for i in range(len(row)):
                X_ijs[i][idx-54+26][1]=row.iloc[i]
            continue
        for i in range(len(row)):
            X_ijs[i][idx][0]=row.iloc[i]
    # %%
```

```
    X_ijs_flat=X_ijs.flatten()
    profit=cal_each_year_profit(xi=X_ijs_flat)
    print(profit)
```

cal_profit2.py:

```
# %%
# coding=gbk
import pandas as pd
import numpy as np
i_dim=41
j_dim=54
s_dim=2
year_dim=7
file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])

# %%

# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year

# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=7):
    return i * (s_dim * year_dim) + s * year_dim + year



# 得到索引
def getXindex(i,j, s , j_dim=54 , s_dim=2):
    return i * (j_dim*s_dim) + j * (s_dim) + s

# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year
```

```
question='Q3' # 选择Q2还是Q3
total_w=0
# 读取Excel文件
file_path = './result2.xlsx'
result_w=pd.DataFrame(columns=['W'])
for sum_i in range(100):
    file_cost_path=question+'data/'+question+'_2023-2030预期销售量+销售单价_'+str(sum_i+1)+'.csv'
    file_product_path=question+'data/'+question+'_2023-2030亩产量+种植成本_'+str(sum_i+1)+'.csv'

    file_product = pd.read_csv(file_product_path,encoding='utf-8')
    file_product=file_product.applymap(lambda x: x.strip() if isinstance(x, str) else x)
    file_cost=pd.read_csv(file_cost_path,encoding='utf-8')
    file_cost=file_cost.applymap(lambda x: x.strip() if isinstance(x, str) else x)
    file_cost=file_cost.sort_values(by=['作物编号', '种植季次'])
    file_cost
    # %%
    land_plots = file1['地块名称'].unique()# 地块
    crops = csv_data['作物编号'].unique()# 作物
    land_type_mapping = file1[['地块名称',
        '地块类型']].set_index('地块名称').to_dict()['地块类型']

    # 创建索引
    land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
    crop_indices = {crop: idx for idx, crop in enumerate(crops)}

    # 四维数组 Cost_ijsyear (表示种植成本), 形状为 (作物编号数41, 地块数54, 2个季次, 7年)
    Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 7))
    Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 7))
    # 把2023年的作为初值
    X_Init=np.zeros((len(crops), len(land_plots), 2, 7))
    for idx, row in file2.iterrows():
        crop_idx = crop_indices[row['作物编号']]
        land_plot = row['种植地块']
        if land_plot in land_plot_indices:
            plot_idx = land_plot_indices[land_plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            area = row['种植面积/亩']
            X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变

    # %%

    # 三维数组 Except_Product_isyear (表示预期销售量), 形状为 (作物编号数41, 2个季次, 7年)
    Except_Product_isyear = np.zeros((len(crops), 2, 7))

    # 填充四维数组 Cost_ijsyear和Yield
    for idx, row in csv_data.iterrows():
```

```
    crop_idx = crop_indices[row['作物编号']]
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if
        plot_type == land_type]

    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            for yi in range(7):
                cost = file_product.iloc[idx][str(yi+2024)+'种植成本/(元/亩)']
                yield_=file_product.iloc[idx][str(yi+2024)+'亩产量/斤']
                Cost_ijsyear[crop_idx, plot_idx, season, yi] = cost
                Yield_ijsyear[crop_idx, plot_idx, season, yi] = yield_
# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤':
    'sum'}).reset_index()
grouped
# %%
# 第一维是地块个数54,第二维是季次数 2,第三维是7年
S_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)# 销售单价
for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    for yi in range(7):
        Except_Product = file_cost.iloc[idx][str(2024+yi)+'预期销售量/斤']
        Except_Product_isyear[crop_idx, season, yi] = Except_Product
        Except_Product = row['预期销售量/斤']
        Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变
        if row['种植季次'] == '单季' or row['种植季次']== '第一季':
            Price_isyear[crop_idx, 0, yi] =
                file_cost.iloc[idx][str(2024+yi)+'销售单价/(元/斤)']
        if row['种植季次']== '第二季':
            Price_isyear[crop_idx, 1, yi] =
                file_cost.iloc[idx][str(2024+yi)+'销售单价/(元/斤)']
# %%
def cal_each_year_profit(xi,yi=0):
    total_profit = 0
    total_sale=0
    total_cost=0
    yi=int(yi)-2024
    for ii in range(i_dim):
        for si in range(s_dim):
            cost_total=0# 该作物该季总成本
            product_total=0# 该作物该季总产量(斤)
```

```
            for ji in range(j_dim):# 遍历所有地
                product_total+=xi[getXindex(ii,ji,si)]*
                Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]

                cost_total+=xi[getXindex(ii,ji,si)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                total_cost+=xi[getXindex(ii,ji,si)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]

            #求该作物该季度实际销售量（斤）
            if product_total<=sales:
                sales=product_total
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
    total_profit=total_sale-total_cost
    return total_profit
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变
S_jsyear, Price_isyear
# %%
i_dim=41
j_dim=54
s_dim=2
year_dim=7
S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量
# %%
excel_data = pd.ExcelFile(file_path)
profit=0
for year in range(2024,2031):
    # 读取表格数据
    df = pd.read_excel(excel_data, sheet_name=str(year))
    df = df.drop(df.columns[0:2], axis=1).fillna(0)
    df =df.drop(df.index[82:86])
    X_ijs=np.zeros((41, 54, 2))
    for idx, row in df.iterrows():
        if idx>=54:
            for i in range(len(row)):
                X_ijs[i][idx-54+26][1]=row.iloc[i]
            continue
```

```
            for i in range(len(row)):
                X_ijs[i][idx][0]=row.iloc[i]
        X_ijs_flat=X_ijs.flatten()
        profit+=cal_each_year_profit(xi=X_ijs_flat,yi=year)
        result_w.loc[sum_i]=[profit]
    total_w+=profit

    print("第"+str(sum_i+1)+"个利润为"+str(profit)+"元")
result_w.to_csv(question+'w2_test.csv')
mu=total_w/100
sigma=0
for idx ,row in result_w.iterrows():
    sigma+=(row['W']-mu)*(row['W']-mu)
sigma=np.sqrt(sigma/(100-1))
object_fun_value=mu*mu/sigma
object_fun_value_yi=object_fun_value*1e-8
print("平均"+str(mu))
print("标准差"+str(sigma))
print("目标函数mu^2/sigma为%f 亿元" %object_fun_value_yi)
```

pre_deal.py:

```
    # coding=gbk
# %%
import pandas as pd
import numpy as np
data1=pd.read_excel('./附件2.xlsx',sheet_name='2023年统计的相关数据')
# 去除空格
data1 =data1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
data1=data1.drop(index = [125,126,127],axis=0)
# %%
data2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
data2 =data2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
# %%
data3=data2.groupby(['作物编号','作物名称','种植季次'])
data3_1=data3.count()
# %%
# 地块类型
data2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
data2 =data2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
data2['种植地块'] = data2['种植地块'].apply(lambda x: x[0] if isinstance(x, str) else x)
def sand_type(value):
    if value=='A':
        return '平旱地'
    if value=='B':
        return '梯田'
    if value=='C':
```

```
        return '山坡地'
    if value=='D':
        return '水浇地'
    if value=='E':
        return '普通大棚'
    if value=='F':
        return '智慧大棚'
data2['种植地块'] = data2['种植地块'].apply(sand_type)
data3=data2.groupby(['作物编号','作物名称','种植地块','种植季次'])
data4=data3['种植面积/亩'].agg(['sum']).sort_values('作物编号')
data4=data4.rename(columns={'sum':'种植总量/亩'})
data4.to_csv('每种作物在不同地块种植量.csv',float_format='%.1f',encoding='gbk')
data3_1=data2.groupby(['作物编号','作物名称','种植季次'])
data4_1=data3_1['种植面积/亩'].agg(['sum']).sort_values('作物编号')
data4_1=data4_1.rename(columns={'sum':'种植总量/亩'})
data4_1.to_csv('每种作物每季度种植量.csv',float_format='%.1f',encoding='gbk')
# %%
def calculate_range_average(value):
    if isinstance(value, str) and '-' in value:
        low, high = map(float, value.split('-'))
        return (low + high) / 2
    return value
# 以区间平均值作为销售单价
data1['销售单价/(元/斤)'] = data1['销售单价/(元/斤)'].apply(calculate_range_average)
# 去除DataFrame中每个元素末尾的空格
data1 = data1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
data1['每亩利润/元'] = data1['亩产量/斤'] * data1['销售单价/(元/斤)'] - data1['种植成本/(元/亩)']
data1['作物编号']=data1['作物编号'].astype(int)
data1.to_csv('蔬菜每亩利润.csv',index=False,float_format='%.2f',encoding='gbk')
data2=pd.read_csv('每种作物在不同地块种植量.csv',encoding='gbk')
#data2['种植地块'] = data2['种植地块'].apply(sand_type)
data2['作物编号']=data2['作物编号'].astype(int)
data2
# %%
data1_1=data1.drop(columns=['序号']).rename(columns={'地块类型': '种植地块'})
# %%
merged_df_ = pd.merge(data1_1, data2, on=['作物编号', '作物名称', '种植季次', '种植地块'],
    how='left')
merged_df_['2023种植总利润/元']=merged_df_['每亩利润/元']*merged_df_['种植总量/亩']
merged_df_['预期销售量/斤']=merged_df_['种植总量/亩']*merged_df_['亩产量/斤']
merged_df_.to_csv('2023蔬菜不同地块种植情况及利润.csv',index=False,encoding='gbk')
merged_df_1=merged_df_.groupby(['作物编号','作物名称'])['2023种植总利润/元'].agg(['sum'])
merged_df_1=merged_df_1.rename(columns={'sum': '总利润/元'})
merged_df_1.sort_values('总利润/元',ascending=False).to_csv('2023每种蔬菜种植利润.csv',encoding='gbk')
# %%
file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
```

```
file1.drop(columns=file1.keys()[-1])

# %%
land_plots = file1['地块名称'].unique()# 地块
crops = csv_data['作物编号'].unique()# 作物


# 创建索引
land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
crop_indices = {crop: idx for idx, crop in enumerate(crops)}
crop_indices
# %%

# 第一维是地块个数54,第二维是季次数 2,第三维是7年)
Y_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)
beta_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)


# 按行遍历填充地块
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    Y_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变


for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    if row['种植季次'] == '单季' or row['种植季次']== '第一季':
        beta_isyear[crop_idx, 0, :] = row['销售单价/(元/斤)']
    if row['种植季次'] == '单季' or row['种植季次']== '第二季':
        beta_isyear[crop_idx, 1, :] = row['销售单价/(元/斤)']


Y_jsyear, beta_isyear
# %%

# 将数组 Y_jsyear 展平成一维数组
Y_jsyear_flat = Y_jsyear.flatten()
# 定义函数：将三维索引转换为一维索引
def get_1d_index_3d(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year

# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year
```

problem1-1.py:

```
# %%
# coding=gbk
import pandas as pd
import numpy as np
from sko.DE import DE
from sko.GA import GA
from sko.tools import set_run_mode


file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])

# %%
land_plots = file1['地块名称'].unique()# 地块
crops = csv_data['作物编号'].unique()# 作物
land_type_mapping = file1[['地块名称', '地块类型']].set_index('地块名称').to_dict()['地块类型']

# 创建索引
land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
crop_indices = {crop: idx for idx, crop in enumerate(crops)}



# 四维数组 Cost_ijsyear (表示种植成本), 形状为 (作物编号数41, 地块数54, 2个季次, 2年)
Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 2))
Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 2))

# 把2023年的作为初值
X_Init=np.zeros((len(crops), len(land_plots), 2, 2))



for idx, row in file2.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_plot = row['种植地块']
    if land_plot in land_plot_indices:
        plot_idx = land_plot_indices[land_plot]
```

```
        season = 1 if row['种植季次'] == '第二季' else 0
        area = row['种植面积/亩']
        X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变

# %%

# 三维数组 Except_Product_isyear (表示预期销售量), 形状为 (作物编号数41, 2个季次, 2年)
Except_Product_isyear = np.zeros((len(crops), 2, 2))

# 填充四维数组 Cost_ijsyear和Yield
for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if plot_type
        == land_type]

    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            cost = row['种植成本/(元/亩)']
            yield_=row['亩产量/斤']
            Cost_ijsyear[crop_idx, plot_idx, season, :] = cost # 成本每年不变
            Yield_ijsyear[crop_idx, plot_idx, season, :] = yield_# 每年亩产量不变

# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤': 'sum'}).reset_index()
grouped
# %%
for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    Except_Product = row['预期销售量/斤']
    Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变


Except_Product_isyear


# %%

# 第一维是地块个数54,第二维是季次数 2,第三维是2年
S_jsyear = np.zeros((len(land_plots), 2, 2),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 2),dtype=np.float64)# 销售单价

# 按行遍历填充地块
```

```
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变


for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    if row['种植季次'] == '单季' or row['种植季次']== '第一季':
        Price_isyear[crop_idx, 0, :] = row['销售单价/(元/斤)']
    if row['种植季次']== '第二季':
        Price_isyear[crop_idx, 1, :] = row['销售单价/(元/斤)']


S_jsyear, Price_isyear
# %%
# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=2):
    return j * (s_dim * year_dim) + s * year_dim + year
# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=2):
    return i * (s_dim * year_dim) + s * year_dim + year
# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=2):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year

# %%


i_dim=41
j_dim=54
s_dim=2
year_dim=2

S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量


# 2023年豆类植物种植地块
bean_list=[3,4,7,8,9,10,16,22,25,30,31,34,40,41,42,43,44,52,53]
bean_js = np.zeros((len(land_plots), 2),dtype=np.float64)# 大豆种植情况
for i in [3,4,7,8,9,10,16,22,25,30,31,34,40,41,42,43,44,52,53]:
    bean_js[i][0]==1
```

```
    bean_js[i][1]==1
# 从4维数组变到X_type的索引
def X_4dToX_type(ii,ji,si,yi,type=4):
    if type==1:
        return ii*(26*2)+ji*2+yi
    if type==2:
        return ii*(8*2*2)+ji*(2*2)+si*2+yi
    if type==3:
        return ii*(16*2)+ji*2+yi
    if type==4:
        return ii*(16*2)+ji*2+yi
    if type==5:
        return ii*(4*2*2)+ji*(2*2)+si*2+yi
# %%
'''
ABC类地，编号j从0-25，种植编号0-15的作物，三年为一周期，从2023预估后两年
'''
X_abc=np.zeros((15, 26, 2))
n_dim1 = 15*26*2 # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim
i_dim1=15
j_dim1=26
# ABC类地，编号j从0-25，种植编号0-14的作物
def objective_function_landABC(xi):
    func_type=1
    total_profit = 0
    total_sale=0
    total_cost=0
    for yi in range(year_dim):
        for ii in range(i_dim1):
            for si in range(1):
                cost_total=0# 该作物该季总成本
                product_total=0# 该作物该季总产量(斤)
                for ji in range(j_dim1):# 遍历所有地
                    product_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                    cost_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                    total_cost+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
                #求该作物该季度实际销售量（斤）
                if product_total<=sales:
                    sales=product_total
                #销售量*单价-成本
                total_sale+=sales*
                Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
        total_profit=total_sale-total_cost
```

```
    # 惩罚项
    punish=0
        # 10
    for yi in range(year_dim):
        for si in range(1):
            for ji in range(j_dim1):
                if cons_ueq10_1(xi,yi=yi,si=si,ji=ji):
                    punish+=1e10
    # 11
    for ii in range(i_dim1):
        for ji in range(j_dim1):
            if cons_ueq11_1(xi,ii=ii,ji=ji)>0:
                punish+=1e10


    # 14 x*(6-x)<=0 每种被种植作物在单一露天耕地中至少占6亩
    for ji in range(j_dim1):# 0-26
        for ii in range(i_dim1):
            for si in range(1):
                for yi in range(year_dim):
                    if xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    (6-xi[X_4dToX_type(ii,ji,si,yi,func_type)])>0:
                        punish+=1e10
    # 12 每种作物在同一地块都不能连续两年种植
    for ii in range(i_dim1):
        for ji in range(j_dim1):
            if
                xi[X_4dToX_type(ii,ji,si,0,func_type)]*X_Init_flat[get_1d_index_4d(ii,ji,si,0)]!=0:
                punish+=1e10
            if xi[X_4dToX_type(ii,ji,si,0,func_type)]*xi[X_4dToX_type(ii,ji,si,1,func_type)]!=0:
                punish+=1e10
    return -total_profit+punish # 遗传算法求最小化问题，因此返回负值


n_dim1 = 15*26*2 # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim

i_dim1=15
j_dim1=26
# 定义等式约束
def constraint_eq_1():
    func_type=1 # 函数类型
    eq_constraints = []
    si=0 # 季次类型


    # 12 每种作物在同一地块都不能连续两年种植
```

```
    for ii in range(i_dim1):
        for ji in range(j_dim1):
                eq_constraints.append(lambda xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*
                X_Init_flat[get_1d_index_4d(ii,ji,si,0)])
                eq_constraints.append(lambda xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*
                xi[X_4dToX_type(ii,ji,si,1,func_type)])
    return eq_constraints

# -------不等式约束------




# 10 每块耕地上种植面积应不多于耕地面积
def cons_ueq10_1(xi,yi,si,ji):
    func_type=1
    total=0
    for ii in range(i_dim1):
        total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
    return total-S_jsyear_flat[get_1d_index_3d_s(ji,si,yi)]

# 11 每块耕地三年内都至少种植一次豆类
def cons_ueq11_1(xi,ii,ji):
    func_type=1
    si=0
    bean_js = np.zeros((len(land_plots), 2),dtype=np.float64)# 大豆种植情况
    for i in [3,4,7,8,9,10,16,22,25,30,31,34,40,41,42,43,44,52,53]:
        bean_js[i][0]==1
        bean_js[i][1]==1
    tmp=0
    tmp+=bean_js[ji][si]
    tmp+=xi[X_4dToX_type(ii,ji,si,0,func_type)]
    tmp+=xi[X_4dToX_type(ii,ji,si,1,func_type)]


    return 0.001-tmp

# 定义不等式约束 <=
def constraint_ueq_1():
    ueq_constraints = []
    func_type=1
    # 10
    for yi in range(year_dim):
        for si in range(1):
            for ji in range(j_dim1):
                ueq_constraints.append(lambda xi: cons_ueq10_1(xi,yi=yi,si=si,ji=ji))
```

```
    # 11
    for ii in range(i_dim1):
        for ji in range(j_dim1):
            ueq_constraints.append(lambda xi: cons_ueq11_1(xi,ii=ii,ji=ji))


    # 14 x*(6-x)<=0 每种被种植作物在单一露天耕地中至少占6亩
    for ji in range(j_dim1):# 0-26
        for ii in range(i_dim1):
            for si in range(1):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                                           (6-xi[X_4dToX_type(ii,ji,si,yi,func_type)]))

    return ueq_constraints # 返回所有的不等式约束


n_dim = i_dim*j_dim*s_dim*year_dim
n_dim1 = 26*15*2 # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim
precision=np.ones(n_dim1)
ub1=np.zeros(n_dim1)
for ji in range(j_dim1):
    for yi in range(year_dim):
        for ii in range(i_dim1):
            ub1[X_4dToX_type(ii,ji,0,yi)]=S_jsyear_flat[get_1d_index_3d_s(ji,0,yi)]

print(ub1)



ga1 = GA(func=objective_function_landABC,
         n_dim=n_dim1,
         size_pop=100000,
           max_iter=500,
           lb=[0] * n_dim1,
         ub=ub1,
         constraint_eq=constraint_eq_1(),
         constraint_ueq=constraint_ueq_1(),
         precision=precision,
         early_stop=20)
set_run_mode(objective_function_landABC, 'multithreading')


# %%
# import torch
# ga1.to(torch.device('cuda:0'))
```

```
best_x,best_y=ga1.run()

# # 输出结果
for yi in range(year_dim):
    with open('output_ga_landABC.'+str(yi)+'txt', 'w',encoding='gbk') as f:
        print(f"最佳总利润: {-best_y}",file=f)
        for ii in range(i_dim1):
            for si in range(1):
                for ji in range(j_dim1):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii)+
                          '个作物第'+str(si)+'个季度在第'+str(ji)+'块土地种植面积为',file=f)
                    print('%f亩' %best_x[X_4dToX_type(ii,ji,si,yi,1)],file=f)

        print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 显示负号
plt.plot(ga1.generation_best_Y)
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()
# %%
'''
D类地，种植编号15-36的作物，共22种，编号j从26-33,共8块，两个季次，三年为一周期，从2023预估后两年

'''# ii要+15，ji要+26
X_D=np.zeros((22, 8,2,2))


i_dim2=22
j_dim2=8
n_dim2 = i_dim2*j_dim2*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim



def objective_function_landD(xi):
    func_type=2
    total_profit = 0
    total_sale=0
    total_cost=0

    for yi in range(year_dim):
        for ii in range(i_dim2):
            for si in range(s_dim):
```

```
            product_total=0# 该作物该季总产量(斤)
            for ji in range(j_dim2):# 遍历所有地
                product_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*\Yield_ijsyear_flat[get_1d_index_4d(ii+
                total_cost+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii+15,ji+26,si,yi)]
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii+15,si,yi)]
            #求该作物该季度实际销售量（斤）
            if product_total<=sales:
                sales=product_total
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii+15,si,yi)]
    total_profit=total_sale-total_cost
#惩罚项
# 14 x*(6-x)<=0 每种被种植作物在单一露天耕地中至少占6亩
punish=0
for k in range(n_dim2):
    if xi[k]*(6-xi[k])>0:
        punish+=1e10
# 12 每种作物在同一地块都不能连续两年种植
for si in range(s_dim):
    for ii in range(i_dim2):
        for ji in range(j_dim2):
            if
                xi[X_4dToX_type(ii,ji,si,0,func_type)]*X_Init_flat[get_1d_index_4d(ii+15,ji+26,si,0)]!=0:
                punish+=1e10
            if
                xi[X_4dToX_type(ii,ji,si,0,func_type)]*xi[X_4dToX_type(ii,ji,si,1,func_type)]!=0:
                punish+=1e10
# 10 每块耕地上种植面积应不多于耕地面积
for yi in range(year_dim):
    for si in range(s_dim):
        for ji in range(j_dim2):
            if cons_ueq10_2(xi,yi=yi,si=si,ji=ji)>0:
                punish+=1e10
# 11 每块耕地三年内都至少种植一次豆类
for ii in range(i_dim2):
    for ji in range(j_dim2):
        if cons_ueq11_2(xi,ii=ii,ji=ji)>0:
            punish+=1e10
return -total_profit+punish # 遗传算法求最小化问题，因此返回负值

# 定义等式约束
def constraint_eq_2():
    func_type=2 # 函数类型
    eq_constraints = []
```

```
    # 12 每种作物在同一地块都不能连续两年种植
    for si in range(s_dim):
        for ii in range(i_dim2):
            for ji in range(j_dim2):
                    eq_constraints.append(lambda xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*

                    X_Init_flat[get_1d_index_4d(ii+15,ji+26,si,0)])
                    eq_constraints.append(lambda xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*

                    xi[X_4dToX_type(ii,ji,si,1,func_type)])

    return eq_constraints

# -------不等式约束------


# 10 每块耕地上种植面积应不多于耕地面积
def cons_ueq10_2(xi,yi,si,ji):
    func_type=2
    total=0
    for ii in range(i_dim2):
        total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
    return total-S_jsyear_flat[get_1d_index_3d_s(ji+26,si,yi)]

# 11 每块耕地三年内都至少种植一次豆类
def cons_ueq11_2(xi,ii,ji):
    func_type=2
    si=0
    tmp=0
    for si in range(s_dim):
        tmp+=bean_js[ji][si]
        tmp+=xi[X_4dToX_type(ii,ji,si,0,func_type)]
        tmp+=xi[X_4dToX_type(ii,ji,si,1,func_type)]
    return 0.001-tmp

# 定义不等式约束 <=
def constraint_ueq_2():
    ueq_constraints = []
    func_type=2
    # 10
    for yi in range(year_dim):
        for si in range(s_dim):
            for ji in range(j_dim2):
                ueq_constraints.append(lambda xi: cons_ueq10_2(xi,yi=yi,si=si,ji=ji))

    # 11
    for ii in range(i_dim2):
```

```
        for ji in range(j_dim2):
            ueq_constraints.append(lambda xi: cons_ueq11_2(xi,ii=ii,ji=ji))


    # 14 x*(6-x)<=0 每种被种植作物在单一露天耕地中至少占6亩
    for k in range(n_dim2):
        ueq_constraints.append(lambda xi:xi[k]*(6-xi[k]))

    return ueq_constraints # 返回所有的不等式约束
print(constraint_ueq_2())

precision=np.ones(n_dim2)
ub1=np.ones(n_dim2)
for ji in range(j_dim2):
    for si in range(s_dim):
        for yi in range(year_dim):
            for ii in range(i_dim2):
                ub1[X_4dToX_type(ii,ji,si,yi)]=S_jsyear_flat[get_1d_index_3d_s(ji+26,si,yi)]

print(ub1)


ga1 = GA(func=objective_function_landD,
         n_dim=n_dim2,
         size_pop=15000,
           max_iter=500,
           lb=[0] * n_dim2,
         ub=ub1,
        constraint_eq=constraint_eq_2(),
        constraint_ueq=constraint_ueq_2(),
         precision=precision,
         early_stop=10)


set_run_mode(objective_function_landD, 'multithreading')


# %%
# import torch

# ga1.to(torch.device('cuda:0'))

best_x,best_y=ga1.run()

# # 输出结果
for yi in range(year_dim):
    with open('output_ga_landD-test'+str(yi)+'.txt', 'w',encoding='gbk') as f:
```

```
        print(f"最佳总利润: {-best_y}",file=f)
        for ii in range(i_dim2):
            for si in range(s_dim):
                for ji in range(j_dim2):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii+15)+
                          '个作物第'+str(si)+'个季度在第'+str(ji+26)+'块土地种植面积为',file=f)
                    print('%f亩' %best_x[X_4dToX_type(ii,ji,si,yi,type=2)],file=f)

        print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 显示负号
plt.plot([-y for y in ga1.generation_best_Y])
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()

# %%
'''
E类地第一季度，可以种植16-33的作物，共18种，j编号34-49，有16块地，三年为一周期，从2023预估后两年
'''
# ii+16, ji+34
X_E1=np.zeros((18, 16, 2))
i_dim3=18
j_dim3=16
n_dim3 = i_dim3*j_dim3*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim

def objective_function_landE1(xi):
    func_type=3
    total_profit = 0
    total_sale=0
    total_cost=0
    for yi in range(year_dim):
        for ii in range(i_dim3):
            for si in range(1):
                cost_total=0# 该作物该季总成本
                product_total=0# 该作物该季总产量(斤)
                for ji in range(j_dim3):# 遍历所有地
                    product_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Yield_ijsyear_flat[get_1d_index_4d(ii+16,ji+34,si,yi)]
                    cost_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii+16,ji+34,si,yi)]
                    total_cost+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii+16,ji+34,si,yi)]
```

```
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii+16,si,yi)]
            #求该作物该季度实际销售量（斤）
            if product_total<=sales:
                sales=product_total
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii+16,si,yi)]
    total_profit=total_sale-total_cost
punish=0
    # 10
for yi in range(year_dim):
    for si in range(1):
        for ji in range(j_dim3):
            if cons_ueq10_3(xi,yi=yi,si=si,ji=ji)>0:
                punish+=1e10
# 11
for ii in range(i_dim3):
    for ji in range(j_dim3):
        if cons_ueq11_3(xi,ii=ii,ji=ji)>0:
            punish+=0
# 15 x*(0.3-x)<=0
for ji in range(j_dim3):
    for ii in range(i_dim3):
        for si in range(1):
            for yi in range(year_dim):
                if xi[X_4dToX_type(ii,ji,si,yi)]*(0.3-xi[X_4dToX_type(ii,ji,si,yi)])>0:
                    punish+=1e10

# 12 每种作物在同一地块都不能连续两年种植
for ii in range(i_dim3):
    for ji in range(j_dim3):
            if
                xi[X_4dToX_type(ii,ji,0,0,func_type)]*X_Init_flat[get_1d_index_4d(ii+16,ji+34,0,0)]!=0:
                punish+=1e10
            if xi[X_4dToX_type(ii,ji,0,0,func_type)]*xi[X_4dToX_type(ii,ji,0,1,func_type)]!=0:
                punish+=1e10
return -total_profit+punish



# 定义等式约束
def constraint_eq_3():
    func_type=3 # 函数类型
    eq_constraints = []
    si=0 # 季次类型
    # 12 每种作物在同一地块都不能连续两年种植
    for ii in range(i_dim3):
        for ji in range(j_dim3):
```

```
            eq_constraints.append(lambda
                xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*X_Init_flat[get_1d_index_4d(ii+16,ji+34,si,0)])
            eq_constraints.append(lambda
                xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*xi[X_4dToX_type(ii,ji,si,1,func_type)])


    return eq_constraints

# -------不等式约束------
# 10 每块耕地上种植面积应不多于耕地面积
def cons_ueq10_3(xi,yi,si,ji):
    func_type=3
    total=0
    for ii in range(i_dim3):
        total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
    return total-S_jsyear_flat[get_1d_index_3d_s(ji+34,si,yi)]

# 11 每块耕地三年内都至少种植一次豆类
def cons_ueq11_3(xi,ii,ji):
    func_type=3
    si=0

    tmp=0
    tmp+=bean_js[ji][si]
    tmp+=xi[X_4dToX_type(ii,ji,si,0,func_type)]
    tmp+=xi[X_4dToX_type(ii,ji,si,1,func_type)]
    return 0.001-tmp

# 定义不等式约束 <=
def constraint_ueq_3():
    ueq_constraints = []
    func_type=3
    # 10
    for yi in range(year_dim):
        for si in range(1):
            for ji in range(j_dim3):
                ueq_constraints.append(lambda xi: cons_ueq10_3(xi,yi=yi,si=si,ji=ji))

    # 11
    for ii in range(i_dim3):
        for ji in range(j_dim3):
            ueq_constraints.append(lambda xi: cons_ueq11_3(xi,ii=ii,ji=ji))


    # 15 x*(0.3-x)<=0
    for ji in range(j_dim3):
        for ii in range(i_dim3):
```

```
            for si in range(1):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[X_4dToX_type(ii,ji,si,yi)]*(0.3-xi[X_4dToX_type(ii,ji,si,yi)]))


    return ueq_constraints # 返回所有的不等式约束


precision=np.ones(n_dim3)
ub1=np.ones(n_dim3)
for ji in range(j_dim3):
    for yi in range(year_dim):
        for ii in range(i_dim3):
            ub1[X_4dToX_type(ii,ji,0,yi,type=3)]=S_jsyear_flat[get_1d_index_3d_s(ji+34,0,yi)]



ga1 = GA(func=objective_function_landE1,
         n_dim=n_dim3,
         size_pop=100000,
           max_iter=300,
           lb=[0] * n_dim3,
         ub=ub1,
         constraint_eq=constraint_eq_3(),
         constraint_ueq=constraint_ueq_3(),
         precision=precision,
         prob_mut=0.9,
         early_stop=20)


set_run_mode(objective_function_landE1, 'multithreading')


# %%
# import torch
# ga1.to(torch.device('cuda:0'))
best_x,best_y=ga1.run()
# # 输出结果
with open('output_ga_landE1.txt', 'w',encoding='gbk') as f:
    print(f"最佳总利润: {-best_y}",file=f)
    for yi in range(year_dim):
        for ii in range(i_dim3):
            for si in range(1):
                for ji in range(j_dim3):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii+16)+
                          '个作物第'+str(si)+'个季度在第'+str(ji+34)+'块土地种植面积为',file=f)
```

```
                print('%f亩' %best_x[X_4dToX_type(ii,ji,si,yi,type=3)],file=f)

    print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 显示负号

plt.plot(ga1.generation_best_Y)
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()
# %%

'''
E类地第二季度，可以种植37-40的作物，共4种，j编号34-49，有16块地，三年为一周期，从2023预估后两年
'''
# ii+37, ji+34
X_E2=np.zeros((4, 16, 2))
i_dim4=4
j_dim4=16
n_dim4 = i_dim4*j_dim4*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim

def objective_function_landE2(xi):
    func_type=4
    total_profit = 0
    total_sale=0
    total_cost=0
    for yi in range(year_dim):
        for ii in range(i_dim4):
            for si in range(1,2):
                cost_total=0# 该作物该季总成本
                product_total=0# 该作物该季总产量(斤)
                for ji in range(j_dim4):# 遍历所有地
                    product_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Yield_ijsyear_flat[get_1d_index_4d(ii+37,ji+34,si,yi)]
                    cost_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii+37,ji+34,si,yi)]
                    total_cost+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*
                    Cost_ijsyear_flat[get_1d_index_4d(ii+37,ji+34,si,yi)]
                sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii+37,si,yi)]
                #求该作物该季度实际销售量（斤）
                if product_total<=sales:
                    sales=product_total
                #销售量*单价-成本
```

```
                total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii+37,si,yi)]
        total_profit=total_sale-total_cost
    punish=0
        # 10
    for yi in range(year_dim):
        for si in range(1,2):
            for ji in range(j_dim4):
                if cons_ueq10_3(xi,yi=yi,si=si,ji=ji)>0:
                    punish+=1e10
    # 11
    for ii in range(i_dim4):
        for ji in range(j_dim4):
            if cons_ueq11_4(xi,ii=ii,ji=ji)> 0:
                punish+=1e10
    # 15 x*(0.3-x)<=0
    for ji in range(j_dim4):
        for ii in range(i_dim4):
            for si in range(1):
                for yi in range(year_dim):
                    if  xi[X_4dToX_type(ii,ji,si,yi)]*(0.3-X_4dToX_type(ii,ji,si,yi))>0:
                        punish+=1e10
    # 12 每种作物在同一地块都不能连续两年种植
    for ii in range(i_dim4):
        for ji in range(j_dim4):
                if xi[X_4dToX_type(ii,ji,si,0,func_type)]*
                X_Init_flat[get_1d_index_4d(ii+37,ji+34,si,0)]!=0:
                    punish+=1e10
    return -total_profit

# 定义等式约束
def constraint_eq_4():
    func_type=4 # 函数类型
    eq_constraints = []
    si=1 # 季次类型
    # 12 每种作物在同一地块都不能连续两年种植
    for ii in range(i_dim4):
        for ji in range(j_dim4):
                eq_constraints.append(lambda xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*
                X_Init_flat[get_1d_index_4d(ii+37,ji+34,si,0)])
    return eq_constraints

# -------不等式约束------

# 10 每块耕地上种植面积应不多于耕地面积
def cons_ueq10_4(xi,yi,si,ji):
    func_type=4
    total=0
```

```
    for ii in range(i_dim4):
        total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
    return total-S_jsyear_flat[get_1d_index_3d_s(ji+34,si,yi)]

# 11 每块耕地三年内都至少种植一次豆类
def cons_ueq11_4(xi,ii,ji):
    func_type=4
    si=1
    tmp=0
    tmp+=bean_js[ji][si]
    tmp+=xi[X_4dToX_type(ii,ji,si,0,func_type)]
    tmp+=xi[X_4dToX_type(ii,ji,si,1,func_type)]
    return 0.001-tmp

# 定义不等式约束 <=
def constraint_ueq_4():
    ueq_constraints = []
    func_type=4
    # 10
    for yi in range(year_dim):
        for si in range(1,2):
            for ji in range(j_dim4):
                ueq_constraints.append(lambda xi: cons_ueq10_3(xi,yi=yi,si=si,ji=ji))

    # 11
    for ii in range(i_dim4):
        for ji in range(j_dim4):
            ueq_constraints.append(lambda xi: cons_ueq11_4(xi,ii=ii,ji=ji))


    # 15 x*(0.3-x)<=0
    for ji in range(j_dim4):
        for ii in range(i_dim4):
            for si in range(1):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[X_4dToX_type(ii,ji,si,yi)]*(0.3-X_4dToX_type(ii,ji,si,yi)))


    return ueq_constraints # 返回所有的不等式约束


precision=np.ones(n_dim4)
ub1=np.zeros(n_dim4)
for ji in range(j_dim4):
    for yi in range(year_dim):
        for ii in range(i_dim4):
```

```
          ub1[X_4dToX_type(ii,ji,1,yi,type=5)]=S_jsyear_flat[get_1d_index_3d_s(ji+34,1,yi)]



ga1 = GA(func=objective_function_landE1,
        n_dim=n_dim3,
        size_pop=10000,
          max_iter=300,
          lb=[0] * n_dim3,
        ub=ub1,
        constraint_eq=constraint_eq_4(),
        constraint_ueq=constraint_ueq_4(),
        precision=precision,
        prob_mut=0.01,
        early_stop=20)


set_run_mode(objective_function_landE1, 'multithreading')


# %%
import torch

ga1.to(torch.device('cuda:0'))

best_x,best_y=ga1.run()

# # 输出结果
with open('output_ga_landABC.txt', 'w',encoding='gbk') as f:
    print(f"最佳总利润: {-best_y}",file=f)
    for yi in range(year_dim):
        for ii in range(i_dim4):
            for si in range(1,2):
                for ji in range(j_dim4):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii+37)+
                          '个作物第'+str(si)+'个季度在第'+str(ji+34)+'块土地种植面积为',file=f)
                    print('%f亩' %0.1*best_x[X_4dToX_type(ii,ji,si,yi,type=4)],file=f)

    print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 显示负号

plt.plot(ga1.generation_best_Y)
plt.title('优化过程')
plt.xlabel('迭代次数')
```

```
plt.ylabel('最优解')
plt.grid(True)
plt.show()
# %%
'''
F类地，种植编号16-33的作物，共18种，编号j从50-53,共4块，两个季次，三年为一周期，从2023预估后两年
'''
# ii要+16，ji要+50
X_F=np.zeros((18,4,2,2))


i_dim5=18
j_dim5=4
n_dim5 = i_dim5*j_dim5*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim



def objective_function_landD(xi):
    func_type=5
    total_profit = 0
    total_sale=0
    total_cost=0
    for yi in range(year_dim):
        for ii in range(i_dim5):
            for si in range(s_dim):
                cost_total=0# 该作物该季总成本
                product_total=0# 该作物该季总产量(斤)
                for ji in range(j_dim5):# 遍历所有地
                    product_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*Yield_ijsyear_flat[get_1d_index_4d(ii+1
                    cost_total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*Cost_ijsyear_flat[get_1d_index_4d(ii+16,ji
                    total_cost+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]*Cost_ijsyear_flat[get_1d_index_4d(ii+16,ji
                sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii+16,si,yi)]
                #求该作物该季度实际销售量（斤）
                if product_total<=sales:
                    sales=product_total
                #销售量*单价-成本
                total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii+16,si,yi)]
        total_profit=total_sale-total_cost
    # 惩罚项
    punish=0
    # 12
    for si in range(s_dim):
        for ii in range(i_dim5):
            for ji in range(j_dim5):
                if(xi[X_4dToX_type(ii,ji,si,0,func_type)]*X_Init_flat[get_1d_index_4d(ii+16,ji+50,si,0)]>0):
                    punish+=1e10
                if
```

```
                    xi[X_4dToX_type(ii,ji,si,0,func_type)]*xi[X_4dToX_type(ii,ji,si,1,func_type)]>0:
                    punish+=1e10


    # 10
    for yi in range(year_dim):
        for si in range(s_dim):
            for ji in range(j_dim5):
                total=0
                for ii in range(i_dim5):
                    total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
                    if total-S_jsyear_flat[get_1d_index_3d_s(ji+50,si,yi)]>0:
                        punish+=1e10
    # 11
    for ii in range(i_dim5):
        for ji in range(j_dim5):
            if cons_ueq11_5(xi,ii=ii,ji=ji)>0:
                punish+=1e10
    # 14 x*(0.3-x)<=0 每种被种植作物在单一露天耕地中至少占0.3亩
    for ji in range(j_dim5):
        for ii in range(i_dim5):
            for si in range(s_dim):
                for yi in range(year_dim):
                    if
                        xi[X_4dToX_type(ii,ji,si,yi,func_type)]*(0.3-xi[X_4dToX_type(ii,ji,si,yi,func_type)])>0:
                        punish+=1e10
    return -total_profit+punish # 遗传算法求最小化问题，因此返回负值
# 定义等式约束
def constraint_eq_5():
    func_type=5 # 函数类型
    eq_constraints = []
    # 12 每种作物在同一地块都不能连续两年种植
    for si in range(s_dim):
        for ii in range(i_dim5):
            for ji in range(j_dim5):
                eq_constraints.append(lambda
                    xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*X_Init_flat[get_1d_index_4d(ii+16,ji+50,si,0)])
                eq_constraints.append(lambda
                    xi:xi[X_4dToX_type(ii,ji,si,0,func_type)]*xi[X_4dToX_type(ii,ji,si,1,func_type)])
    return eq_constraints
# -------不等式约束------
# 10 每块耕地上种植面积应不多于耕地面积
def cons_ueq10_5(xi,yi,si,ji):
    func_type=5
    total=0
    for ii in range(i_dim5):
        total+=xi[X_4dToX_type(ii,ji,si,yi,func_type)]
```

```
    return total-S_jsyear_flat[get_1d_index_3d_s(ji+50,si,yi)]
# 11 每块耕地三年内都至少种植一次豆类
def cons_ueq11_5(xi,ii,ji):
    func_type=5
    si=0
    bean_js = np.zeros((len(land_plots), 2),dtype=np.float64)# 大豆种植情况
    for i in [3,4,7,8,9,10,16,22,25,30,31,34,40,41,42,43,44,52,53]:
        bean_js[i][0]==1
        bean_js[i][1]==1
    tmp=0
    tmp+=bean_js[ji][si]
    tmp+=xi[X_4dToX_type(ii,ji,si,0,func_type)]
    tmp+=xi[X_4dToX_type(ii,ji,si,1,func_type)]
    return 0.001-tmp
# 定义不等式约束 <=
def constraint_ueq_5():
    ueq_constraints = []
    func_type=5
    # 10
    for yi in range(year_dim):
        for si in range(s_dim):
            for ji in range(j_dim5):
                ueq_constraints.append(lambda xi: cons_ueq10_5(xi,yi=yi,si=si,ji=ji))
    # 11
    for ii in range(i_dim5):
        for ji in range(j_dim5):
            ueq_constraints.append(lambda xi: cons_ueq11_5(xi,ii=ii,ji=ji))


    # 14 x*(0.3-x)<=0 每种被种植作物在单一露天耕地中至少占0.3亩
    for ji in range(j_dim5):
        for ii in range(i_dim5):
            for si in range(s_dim):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[X_4dToX_type(ii,ji,si,yi,func_type)]*(0.3-xi[X_4dToX_type(ii,ji,si,yi

    return ueq_constraints # 返回所有的不等式约束


precision=np.ones(n_dim5)
ub1=np.ones(n_dim5)
for ji in range(j_dim5):
    for si in range(s_dim):
        for yi in range(year_dim):
            for ii in range(i_dim5):
                ub1[X_4dToX_type(ii,ji,si,yi,type=5)]=S_jsyear_flat[get_1d_index_3d_s(ji+50,si,yi)]
```

```
ga1 = GA(func=objective_function_landD,
        n_dim=n_dim5,
        size_pop=15000,
          max_iter=500,
          lb=[0] * n_dim5,
        ub=ub1,
        constraint_eq=constraint_eq_5(),
        constraint_ueq=constraint_ueq_5(),
        precision=precision,
        prob_mut=0.01,
        early_stop=20)

set_run_mode(objective_function_landD, 'multithreading')

# %%
# import torch
# ga1.to(torch.device('cuda:0'))
best_x,best_y=ga1.run()
# # 输出结果
for yi in range(year_dim):
    with open('output_ga_landF'+str(yi)+'.txt', 'w',encoding='gbk') as f:
        print(f"最佳总利润: {-best_y}",file=f)
        for ii in range(i_dim5):
            for si in range(s_dim):
                for ji in range(j_dim5):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii+16)+
                          '个作物第'+str(si)+'个季度在第'+str(ji+50)+'块土地种植面积为',file=f)
                    print('%f亩' %best_x[X_4dToX_type(ii,ji,si,yi,type=5)],file=f)

        print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 显示负号
plt.plot([-y for y in ga1.generation_best_Y])
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()
# %%
```

problem1-2.py：

```
# %%
# coding=gbk
import pandas as pd
```

```python
import numpy as np
from sko.GA import GA
from sko.tools import set_run_mode
file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])
# %%
land_plots = file1['地块名称'].unique()# 地块
crops = csv_data['作物编号'].unique()# 作物
land_type_mapping = file1[['地块名称', '地块类型']].set_index('地块名称').to_dict()['地块类型']

# 创建索引
land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
crop_indices = {crop: idx for idx, crop in enumerate(crops)}



# 四维数组 Cost_ijsyear (表示种植成本), 形状为 (作物编号数41, 地块数54, 2个季次, 7年)
Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 7))
Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 7))

# 把2023年的作为初值
X_Init=np.zeros((len(crops), len(land_plots), 2, 7))



for idx, row in file2.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_plot = row['种植地块']
    if land_plot in land_plot_indices:
        plot_idx = land_plot_indices[land_plot]
        season = 1 if row['种植季次'] == '第二季' else 0
        area = row['种植面积/亩']
        X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变

# %%

# 三维数组 Except_Product_isyear (表示预期销售量), 形状为 (作物编号数41, 2个季次, 7年)
Except_Product_isyear = np.zeros((len(crops), 2, 7))
```

```
# 填充四维数组 Cost_ijsyear和Yield
for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if plot_type
        == land_type]

    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            cost = row['种植成本/(元/亩)']
            yield_=row['亩产量/斤']
            Cost_ijsyear[crop_idx, plot_idx, season, :] = cost # 成本每年不变
            Yield_ijsyear[crop_idx, plot_idx, season, :] = yield_# 每年亩产量不变

# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤': 'sum'}).reset_index()
grouped
# %%
for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    Except_Product = row['预期销售量/斤']
    Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变


Except_Product_isyear


# %%

# 第一维是地块个数54,第二维是季次数 2,第三维是7年
S_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)# 销售单价

# 按行遍历填充地块
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变


for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    if row['种植季次'] == '单季' or row['种植季次']== '第一季':
```

```
        Price_isyear[crop_idx, 0, :] = row['销售单价/(元/斤)']
    if row['种植季次']== '第二季':
        Price_isyear[crop_idx, 1, :] = row['销售单价/(元/斤)']


S_jsyear, Price_isyear
# %%


# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year

# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=7):
    return i * (s_dim * year_dim) + s * year_dim + year



# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year

# %%

i_dim=41
j_dim=54
s_dim=2
year_dim=7

S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量
#Yield_ijsyear[16,:,:,:]
# %%

def objective_function(xi):
    total_profit = 0
    total_sale=0
    total_cost=0
    for yi in range(year_dim):
        for ii in range(i_dim):
            for si in range(s_dim):
                cost_total=0# 该作物该季总成本
```

```
            product_total=0# 该作物该季总产量(斤)
            for ji in range(j_dim):# 遍历所有地
                product_total+=xi[get_1d_index_4d(ii,ji,si,yi)]*Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi
                cost_total+=xi[get_1d_index_4d(ii,ji,si,yi)]*Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                total_cost+=xi[get_1d_index_4d(ii,ji,si,yi)]*Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
            sales=product_total

            #求该作物该季度实际销售量（斤）
            if product_total<=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]:
                sales=0.5*(product_total+Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)])
            #销售量*单价-成本
            total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]

        total_profit=total_sale-total_cost
    return -total_profit # 遗传算法求最小化问题，因此返回负值

set_run_mode(objective_function, 'multithreading')
n_dim = i_dim*j_dim*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim
# 定义等式约束
def constraint_eq():
    eq_constraints = []
    # 1
    for yi in range(year_dim):
        for ii in range(i_dim):
            for si in range(s_dim):
                if(si==2):
                    for ji in range(j_dim):
                        if(ji<=25):
                            eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,si,yi)])
    # 2
    for yi in range(year_dim):
        for ii in range(i_dim):
            if(ii>=15):
                continue
            for si in range(s_dim):
                for ji in range(j_dim):
                    if(ji<=25):
                        eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,si,yi)])

    # 3-4
    for yi in range(year_dim):
        for ii in range(i_dim):
            if(ii==34 and ii==35 and ii==36):
                continue
            for si in range(s_dim):
                if(si==1):
```

```
                for ji in range(j_dim):
                    if(ji<=33 and ji>=26):
                        eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,si,yi)])
# 5
for yi in range(year_dim):
    for ii in range(i_dim):
        if(ii<=33 and ii>=15):
            continue
        for si in range(s_dim):
            if(si==1):
                for ji in range(j_dim):
                    if(ji<=33 and ji>=26):
                        eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,si,yi)])
# 6
for yi in range(year_dim):
    for ii in range(i_dim):
        if(ii>=16 and ii <=33):
            for ji in range(26,34):# 26-33
                eq_constraints.append(lambda
                    xi:xi[get_1d_index_4d(ii,ji,0,yi)]*xi[get_1d_index_4d(15,ji,0,yi)])
        if(ii>=34 and ii<=36):
            for ji in range(26,34):
                eq_constraints.append(lambda
                    xi:xi[get_1d_index_4d(ii,ji,1,yi)]*xi[get_1d_index_4d(15,ji,0,yi)])

# 7
for yi in range(year_dim):
    for ii in range(i_dim):
        if(ii<=33 and ii>=16):
            continue
        for ji in range(34,50):# 34-49
            eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,0,yi)])

# 8
for yi in range(year_dim):
    for ii in range(i_dim):
        if(ii<=40 and ii>=37):
            continue
        for ji in range(34,50):# 34-49
            eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,1,yi)])
# 9
for yi in range(year_dim):
    for ii in range(i_dim):
        if(ii<=33 and ii>=16):
            continue
        for si in range(s_dim):
            for ji in range(50,54):
```

```
                eq_constraints.append(lambda xi:xi[get_1d_index_4d(ii,ji,si,yi)])

    # 12
    for yi in range(year_dim-1):
        for ii in range(i_dim):
            for si in range(s_dim):
                for ji in range(j_dim):
                    eq_constraints.append(lambda
                        xi:xi[get_1d_index_4d(ii,ji,si,yi)]*xi[get_1d_index_4d(ii,ji,si,yi+1)])

    # 13
    for yi in range(year_dim):
        for ii in range(i_dim):
            if(ii>=16 and ii<=33):
                continue
            for si in range(s_dim-1):
                for ji in range(50,54):
                    eq_constraints.append(lambda
                        xi:xi[get_1d_index_4d(ii,ji,si,yi)]*xi[get_1d_index_4d(ii,ji,si+1,yi)])

    return eq_constraints
# -------不等式约束------
# 10
def cons_ueq10(xi,yi,si,ji):
    total=0
    for ii in range(i_dim):
        total+=xi[get_1d_index_4d(ii,ji,si,yi)]
    return total-S_jsyear_flat[get_1d_index_3d_s(ji,si,yi)]

# 11
def cons_ueq11(xi,ii,ji):
    bean_js = np.zeros((len(land_plots), 2),dtype=np.float64)# 大豆种植情况
    for i in [3,4,7,8,9,10,16,22,25,30,31,34,40,41,42,43,44,52,53]:
        bean_js[i][0]==1
        bean_js[i][1]==1
    tmp=0
    for yi in range(year_dim-1):
        if yi==0:
            for si in range(s_dim):
                tmp+=bean_js[ji][si]
                tmp+=xi[get_1d_index_4d(ii,ji,si,yi)]
                tmp+=xi[get_1d_index_4d(ii,ji,si,yi+1)]
            return 0.001-tmp
        for si in range(s_dim):
            tmp+=xi[get_1d_index_4d(ii,ji,si,yi-1)]
            tmp+=xi[get_1d_index_4d(ii,ji,si,yi)]
            tmp+=xi[get_1d_index_4d(ii,ji,si,yi+1)]
```

```
    return 0.001-tmp
# 定义不等式约束 <=
def constraint_ueq():
    ueq_constraints = []
    # 10
    for yi in range(year_dim):
        for si in range(s_dim):
            for ji in range(j_dim):
                ueq_constraints.append(lambda xi: cons_ueq10(xi,yi=yi,si=si,ji=ji))
    # 11
    for ii in range(i_dim):
        for ji in range(j_dim):
            ueq_constraints.append(lambda xi: cons_ueq11(xi,ii=ii,ji=ji))
    # 14 x*(6-x)<=0
    for ji in range(40):# 0-39
        for ii in range(i_dim):
            for si in range(s_dim):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[get_1d_index_4d(ii,ji,si,yi)]*(6-xi[get_1d_index_4d(ii,ji,si,yi)]))
    # 15 x*(0.3-x)<=0
    for ji in range(40,53):# 40-53
        for ii in range(i_dim):
            for si in range(s_dim):
                for yi in range(year_dim):
                    ueq_constraints.append(lambda xi:
                                           xi[get_1d_index_4d(ii,ji,si,yi)]*(0.3-xi[get_1d_index_4d(ii,ji,si,yi)]))
    return ueq_constraints # 返回所有的不等式约束

n_dim = i_dim*j_dim*s_dim*year_dim
ga = GA(func=objective_function,
        n_dim=n_dim,
        size_pop=1000,
          max_iter=200,
          lb=[0] * n_dim,
        ub=[csv_data['种植总量/亩'].max()+50]* n_dim,
        constraint_eq=constraint_eq(),
        constraint_ueq=constraint_ueq(),
        precision=[1e-1]*n_dim,
        early_stop=20)
set_run_mode(objective_function, 'parallel')
# %%
# import torch
# ga.to(torch.device('cuda:0'))
best_x,best_y=ga.run()
# # 输出结果
with open('output_pb2_ga.txt', 'w',encoding='gbk') as f:
```

```
    print(f"最佳总利润: {-best_y}",file=f)
    for yi in range(year_dim):
        for ii in range(i_dim):
            for si in range(s_dim):
                for ji in range(j_dim):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii)+
                          '个作物第'+str(si)+'个季度在第'+str(ji)+'块土地种植面积为',file=f)
                    print('%f亩' %best_x[get_1d_index_4d(ii,ji,si,yi)],file=f)

                sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
    print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.plot(ga.generation_best_Y)
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()
```

problem2and3.py:

```
    # %%
# coding=gbk
import pandas as pd
import numpy as np
i_dim=41
j_dim=54
s_dim=2
year_dim=7
file1 = pd.read_excel('./附件1.xlsx',sheet_name='乡村的现有耕地')
csv_data=pd.read_csv('./2023蔬菜不同地块种植情况及利润.csv',encoding='gbk')
csv_data['种植总量/亩'].fillna(0, inplace=True)
file2=pd.read_excel('./附件2.xlsx',sheet_name='2023年的农作物种植情况')
# 去除空格
flie2 =file2.applymap(lambda x: x.strip() if isinstance(x, str) else x)
#去除空格
file1 =file1.applymap(lambda x: x.strip() if isinstance(x, str) else x)
csv_data=csv_data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
file1.drop(columns=file1.keys()[-1])
# %%
# 将三维每个地块每个季节每年的地块面积索引转换为一维索引
def get_1d_index_3d_s(j, s, year, s_dim=2, year_dim=7):
    return j * (s_dim * year_dim) + s * year_dim + year
# 将三维每种作物每个季节每年的 销售单价、预期销售量
def get_1d_index_3d_ep(i, s, year, s_dim=2, year_dim=7):
```

```
    return i * (s_dim * year_dim) + s * year_dim + year
# 得到索引
def getXindex(i,j, s , j_dim=54 , s_dim=2):
    return i * (j_dim*s_dim) + j * (s_dim) + s
# 定义函数：将四维索引转换为一维索引
def get_1d_index_4d(i,j, s, year , j_dim=54 , s_dim=2, year_dim=7):
    return i * (j_dim*s_dim * year_dim) + j * (s_dim*year_dim) + s * year_dim + year
question='Q3' # 选择Q2还是Q3
# 读取Excel文件
file_path = './result2.xlsx'
result_w=pd.DataFrame(columns=['W'])
def object_func_problem2(X):
    total_w=0
    for sum_i in range(100):
        file_cost_path=question+'data/'+question+'_2023-2030预期销售量+销售单价_'+str(sum_i+1)+'.csv'
        file_product_path=question+'data/'+question+'_2023-2030亩产量+种植成本_'+str(sum_i+1)+'.csv'
        file_product = pd.read_csv(file_product_path,encoding='utf-8')
        file_product=file_product.applymap(lambda x: x.strip() if isinstance(x, str) else x)
        file_cost=pd.read_csv(file_cost_path,encoding='utf-8')
        file_cost=file_cost.applymap(lambda x: x.strip() if isinstance(x, str) else x)
        file_cost=file_cost.sort_values(by=['作物编号', '种植季次'])
        file_cost
        # %%
        land_plots = file1['地块名称'].unique()# 地块
        crops = csv_data['作物编号'].unique()# 作物
        land_type_mapping = file1[['地块名称',
            '地块类型']].set_index('地块名称').to_dict()['地块类型']

        # 创建索引
        land_plot_indices = {plot: idx for idx, plot in enumerate(land_plots)}
        crop_indices = {crop: idx for idx, crop in enumerate(crops)}

        # 四维数组 Cost_ijsyear (表示种植成本)，形状为 (作物编号数41，地块数54，2个季次，7年)
        Cost_ijsyear = np.zeros((len(crops), len(land_plots), 2, 7))
        Yield_ijsyear= np.zeros((len(crops), len(land_plots), 2, 7))

        # 把2023年的作为初值
        X_Init=np.zeros((len(crops), len(land_plots), 2, 7))



        for idx, row in file2.iterrows():
            crop_idx = crop_indices[row['作物编号']]
            land_plot = row['种植地块']
            if land_plot in land_plot_indices:
                plot_idx = land_plot_indices[land_plot]
                season = 1 if row['种植季次'] == '第二季' else 0
```

```
        area = row['种植面积/亩']
        X_Init[crop_idx, plot_idx, season, :] = area # 每年种植面积保持不变

# %%

# 三维数组 Except_Product_isyear (表示预期销售量), 形状为 (作物编号数41, 2个季次, 7年)
Except_Product_isyear = np.zeros((len(crops), 2, 7))

# 填充四维数组 Cost_ijsyear和Yield
for idx, row in csv_data.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    land_type = row['种植地块'] # 从 CSV 中获取种植地块类型
    matching_land_plots = [plot for plot, plot_type in land_type_mapping.items() if
        plot_type == land_type]

    if matching_land_plots:
        for plot in matching_land_plots:
            plot_idx = land_plot_indices[plot]
            season = 1 if row['种植季次'] == '第二季' else 0
            for yi in range(7):
                # if yi==0:
                #     cost = row['种植成本/(元/亩)']
                #     yield_=row['亩产量/斤']
                # else:
                cost = file_product.iloc[idx][str(yi+2024)+'种植成本/(元/亩)']
                yield_=file_product.iloc[idx][str(yi+2024)+'亩产量/斤']
                Cost_ijsyear[crop_idx, plot_idx, season, yi] = cost
                Yield_ijsyear[crop_idx, plot_idx, season, yi] = yield_

# %%
# 填充三维数组 Except_Product_isyear, 按作物编号和种植季次进行groupby分组并求和
grouped = csv_data.groupby(['作物编号', '种植季次']).agg({'预期销售量/斤':
    'sum'}).reset_index()
grouped
# %%
# 第一维是地块个数54,第二维是季次数 2,第三维是7年
S_jsyear = np.zeros((len(land_plots), 2, 7),dtype=np.float64)# 地块面积
Price_isyear = np.zeros((len(crops), 2, 7),dtype=np.float64)# 销售单价

for idx, row in grouped.iterrows():
    crop_idx = crop_indices[row['作物编号']]
    season = 1 if row['种植季次'] == '第二季' else 0
    for yi in range(7):
        Except_Product = file_cost.iloc[idx][str(2024+yi)+'预期销售量/斤']
        Except_Product_isyear[crop_idx, season, yi] = Except_Product
        Except_Product = row['预期销售量/斤']
        Except_Product_isyear[crop_idx, season, :] = Except_Product # 预期销售量每年不变
```

```
        if row['种植季次'] == '单季' or row['种植季次']== '第一季':
            Price_isyear[crop_idx, 0, yi] =
                file_cost.iloc[idx][str(2024+yi)+'销售单价/(元/斤)']
        if row['种植季次']== '第二季':
            Price_isyear[crop_idx, 1, yi] =
                file_cost.iloc[idx][str(2024+yi)+'销售单价/(元/斤)']

# %%


# 按行遍历填充地块
for idx, row in file1.iterrows():
    plot_idx = land_plot_indices[row['地块名称']]
    area = row['地块面积/亩']
    S_jsyear[plot_idx, :, :] = area # 每季次、每年面积不变
S_jsyear, Price_isyear
# %%
i_dim=41
j_dim=54
s_dim=2
year_dim=7
S_jsyear_flat = S_jsyear.flatten()# 地块面积
Price_isyear_flat=Price_isyear.flatten()# 销售单价
Cost_ijsyear_flat=Cost_ijsyear.flatten()# 成本
Except_Product_isyear_flat=Except_Product_isyear.flatten()# 预期销售量
X_Init_flat=X_Init.flatten() # 初值
Yield_ijsyear_flat=Yield_ijsyear.flatten()# 亩产量
def cal_each_year_profit(xi,yi=0):
    total_profit = 0
    total_sale=0
    total_cost=0
    yi=int(yi)-2024
    for ii in range(i_dim):
        for si in range(s_dim):
            cost_total=0# 该作物该季总成本
            product_total=0# 该作物该季总产量(斤)
            for ji in range(j_dim):# 遍历所有地
                product_total+=xi[getXindex(ii,ji,si)]*
                Yield_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]

                cost_total+=xi[getXindex(ii,ji,si)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
                total_cost+=xi[getXindex(ii,ji,si)]*
                Cost_ijsyear_flat[get_1d_index_4d(ii,ji,si,yi)]
            sales=Except_Product_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]

            #求该作物该季度实际销售量（斤）
```

```
                if product_total<=sales:
                    sales=product_total
                #销售量*单价-成本
                total_sale+=sales*Price_isyear_flat[get_1d_index_3d_ep(ii,si,yi)]
            total_profit=total_sale-total_cost
            return total_profit
        profit=0
        for year in range(2024,2031):
            profit+=cal_each_year_profit(xi=X,yi=year)
        total_w+=profit
    mu=total_w/100
    sigma=0
    for idx ,row in result_w.iterrows():
        sigma+=(row['W']-mu)*(row['W']-mu)
    sigma=np.sqrt(sigma/(100-1))
    object_fun_value=mu*mu/sigma
    object_fun_value_yi=object_fun_value*1e-8 #单位转换成亿元
    return object_fun_value_yi
from sko.GA import GA
from sko.tools import set_run_mode
set_run_mode(object_func_problem2, 'multithreading')
n_dim = i_dim*j_dim*s_dim*year_dim # 因变量 xi 的维度 i_dim*j_dim*s_dim*year_dim
ga = GA(func=object_func_problem2,
        n_dim=n_dim,
        size_pop=1000,
          max_iter=200,
          lb=[0] * n_dim,
        ub=[csv_data['种植总量/亩'].max()+50]* n_dim,
        precision=[1e-1]*n_dim,
        early_stop=20)
# %%
# import torch
# ga.to(torch.device('cuda:0'))
best_x,best_y=ga.run()
# # 输出结果
with open('output_pb2_ga.txt', 'w',encoding='gbk') as f:
    print(f"最佳总利润: {-best_y}",file=f)
    for yi in range(year_dim):
        for ii in range(i_dim):
            for si in range(s_dim):
                for ji in range(j_dim):# 遍历所有地
                    print("第"+str(yi)+"年"+"第"+str(ii)+
                          '个作物第'+str(si)+'个季度在第'+str(ji)+'块土地种植面积为',file=f)
                    print('%f亩' %best_x[get_1d_index_4d(ii,ji,si,yi)],file=f)
    print(f"最佳总利润: {-best_y}",file=f)
import matplotlib.pyplot as plt
# 绘制最优解的变化图
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.plot(ga.generation_best_Y)
plt.title('优化过程')
plt.xlabel('迭代次数')
plt.ylabel('最优解')
plt.grid(True)
plt.show()
```