多目标规划思想下农作物最优种植策略研究

摘　要

为实现乡村经济可持续发展，本文基于**贪心算法、粒子群优化算法、蒙特卡罗模拟**，构建了**多目标优化模型、线性规划模型**，解决了未来七年内农作物最优种植策略。

针对问题一，首先对给定农作物和地块数据进行预处理，构建地块和作物间的映射关系，分析地块的种植潜力。接下来建立**线性规划**模型，以最大化乡村经济收益为目标，探讨不同作物在各类型地块上的最优种植面积分配。同时，依据农作物生长规律，在模型中引入了豆类作物种植频率限制等多个约束条件，并对超出销售量的部分进行滞销和降价两种情况处理，最终通过 Python 求解该线性规划模型，并对结果进行灵敏度分析，表明粮食类作物的合理分配对收益最大化有显著影响，而蔬菜类作物在不同地块上的种植则需平衡产量与销售风险。

针对问题二，进一步考虑了农作物亩产量、销售量、销售价格和种植成本的不确定性，引入**风险评估参数**，建立**多目标规划模型**，运用**贪心算法**，实现收益最大化和风险最小化。为了更好地使用正态分布模拟这些不确定性因素的随机变化，构建了随机规划模型，通过**蒙特卡洛模拟**生成多个场景，采用**粒子群优化算法**求解，得到最大年收益为 30812000、30196000、29413000、28832000、28487000、27948000、28383000 元。结果表明，在不确定条件下，合理规划小麦和玉米的种植比例有助于提高长期收益，而蔬菜类作物的种植策略需要根据其价格波动灵活调整。

针对问题三，为探究销售量、价格和种植成本之间的相关性，选取 **Pearson 相关系数**量化了这些变量的相互关系，又引入**需求交叉弹性理论和互补性参数**分别描述农作物之间的可替代性、互补性，结果发现粮食类和蔬菜类间具有可替代性，食用菌与蔬菜类和粮食类都具有互补性。此外构建多目标优化模型，引入了多农作物合作框架，以最大化收益为目标，制定高效稳健的种植策略。最后将问题三和问题二的结果进行对比，分别计算标准差和变异系数，得出问题三在引入可替代性和互补性的目标函数后，年总收益进一步增加。结果显示，合理利用作物间的互补性可有效降低市场波动带来的风险，而针对替代性作物的灵活调整能够提高种植策略的稳健性。通过合理的构建线性规划模型，本文制定了较为卓越的农作物种植方案为农作物的种植提供了经济性的方案。

关键词:线性规划　粒子群优化算法　蒙特卡罗模拟　多目标优化

一、 问题重述

1.1 问题背景

国务院印发的“十四五”推进农业农村现代化规划中指出，“三农”工作是全面建设社会主义现代化国家的重中之重。“十四五”时期，“三农”工作重心历史性地转向了全面推进乡村振兴，加快中国特色农业农村现代化进程上。

种植业作为农业的重要组成部分，对国家经济发展和人民生活水平的提高具有重要意义。根据乡村的实际情况，充分利用耕地资源，选择合适的农作物并优化种植策略，可以提高农产品的附加值，增加农产品的市场竞争力[1]。有助于促进农村经济的多元发展，提高农民的收入水平，推动乡村农业的可持续发展。

1.2 题目信息

某乡村位于华北山区，气候条件较为寒冷，导致大多数耕地每年只能种植一季作物。该乡村现有露天耕地 1201 亩，分为 34 块不同规模的地块，包括平旱地、梯田、山坡地和水浇地四种类型。平旱地、梯田和山坡地适合每年种植一季粮食作物；水浇地适合每年种植一季水稻或两季蔬菜。此外，乡村还拥有 16 个普通大棚和 4 个智慧大棚，每个大棚的耕地面积为 0.6 亩。普通大棚适合每年种植一季蔬菜和一季食用菌，而智慧大棚适合每年种植两季蔬菜。同一地块每季可以种植不同的作物。

根据作物的生长规律，同一地块不应连续种植相同作物，否则会导致减产。由于豆类作物的根菌有助于其他作物的生长，从 2023 年开始，每个地块必须在三年内至少种植一次豆类作物。此外，种植方案应考虑耕作和管理的便利性，每季的作物种植地应尽量集中，每种作物在单个地块种植的面积也不宜太小。

1.3 待求解问题

问题 1：假定未来各种农作物的预期销售量、种植成本、亩产量和销售价格保持与 2023 年相同，每季种植的作物当季销售。如果某种作物每季的总产量超过了预期销售量，超过的部分无法正常销售。请针对以下两种情况，分别给出该乡村 2024~2030 年的最优种植方案：

情况 1：超过部分滞销，造成浪费；

情况 2：超过部分按 2023 年销售价格的 50%降价出售。

问题 2：根据经验，小麦和玉米的预期销售量未来有增长趋势，年增长率在 5%~10%之间，而其他农作物的预期销售量相对于 2023 年大约有±5%的波动。农作物的亩产量会受气候等因素影响，每年可能有±10%的变化。种植成本每年平均增长 5%左右。粮食类作物的销售价格基本稳定；蔬菜类作物销售价格有增长趋势，年均增长约 5%；食用菌销售价格稳中有降，大约每年下降 1%~5%，其中羊肚菌的销售价格每年下降幅度为 5%。

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变动及潜在风险，给出该乡村 2024~2030 年的最优种植方案。

问题 3：在实际情况中，农作物之间可能存在一定的替代性和互补性，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。请综合考虑这些相关因素，给出该乡村 2024~2030 年的最优种植策略。通过模拟数据求解，并与问题 2 求得的结果进行比较分析。

二、 问题分析

2.1 问题一的分析

问题一涉及对销售量超出部分作物的两种处理情况，主要区别在于目标函数的不同。第一小问的目标是最大化乡村在 2024 年至 2030 年期间的净收益，同时最小化滞销损失，第二小问则关注在将超过部分的作物按 50%的价格出售的方案下，如何最大化乡村在此期间的总收益。首先，我们需要根据农作物的生长规律进行数据预处理，筛选适合不同季度种植的作物。接下来，设定决策变量和相对应的目标函数，并根据题目要求建立多个约束条件，如总面积约束，不可连续重茬种植约束等，利用线性规划模型进行求解。

2.2 问题二的分析

针对问题二，需要合理规划以最小化种植风险，并最大化未来的收益。首先对数据进行初步处理，建立地块与作物的映射关系。接着考虑销售量、亩产量、种植成本和销售价格的波动，可将波动视作一种风险，故引入风险评估参数。根据风险评估参数重新计算销售价、销售量和销售成本，在实现收益最大化和风险最小化的目标上，建立多目标规划模型，又因为波动具有随机性和不确定性，故使用蒙特卡罗模拟随机模拟不确定性因素，最后采用粒子群优化算法求解找到最优的种植方案。

2.3 问题三的分析

针对问题三，要求在问题二的基础上，综合考虑不同作物之间的相互关系，以及销售量、价格和成本等因素之间的相关性，构建一个更加复杂的优化模型。对于农作物之间的替代性和互补性，引入经济学中**交叉弹性和互补性系数**，对目标函数进行补充，约束条件与前面问题相同，构建多目标规划模型进行求解；而对于预期销售量与销售价格、种植成本之间的相关性，可用 Pearson 相关系数绘制热力图直观展现。最后比较问题二和问题三的结果，计算两者的标准差和变异系数。

三、 模型假设

(1)平旱地、山坡地和梯田每年都只种植一季作物。水浇地每年可以种植一季或者两季作物。大棚每年可以进行两季作物的种植。

(2)2023 年该乡村生产的农作物产品全部销售完毕。

(3)假定各种农作物未来的预期销售量、种植成本、亩产量和销售价格相对于 2023 年保持稳定。

(4)假设农民基于生产实践经验有计划性地种植，年产量与市场需求大致相同。

四、 符号说明

符号	说明	单位
p_i	第 i 种作物的销售价格	元/亩
c_i	第 i 种作物的种植成本	元/亩
q_i	第 i 种作物的亩产量	千克/亩
D_i	第 i 种作物的预期销售量	千克
A_j	第 j 个地块或大棚的总面积	亩
$y_{i,k,t}$	第 i 种作物在第 t 年第 k 季度的总产量	千克
Z_3	是 2024 年至 2030 年期间的总收益	元

五、 数据预处理

综合浏览附件所包含信息，可以发现其数据量大且覆盖范围广，贴近生活。进行数据整合，变量处理等基础工作，将为后续的问题分析和结果预测奠定坚实基础。

5.1 缺失值补充

观察附件二表格一“2023 年的农作物种植情况”，可知地块名称等信息存在缺失值。我们通过 Python 的 fillna 函数对缺失值进行补充。

5.2 数据整合

总览附件二表格信息，可以发现地块数据多样且分散。我们首先提取各地块数据信息，依照地块名称排序，并附上该地块对应的类型与面积，记作地块信息整合表。如下表所示意：

表格 1 地块信息整合示意表

地块名称	地块类型	地块面积/亩	地块名称	地块类型	地块面积/亩
A1	平旱地	80	B6	梯田	86
A2	平旱地	55	B7	梯田	55
A3	平旱地	35	B8	梯田	44
A4	平旱地	72	B9	梯田	50
A5	平旱地	68	B10	梯田	25
A6	平旱地	55	B11	梯田	60
B1	梯田	60	B12	梯田	45
B2	梯田	46	B13	梯田	35
B3	梯田	40	B14	梯田	20
B4	梯田	28	C1	山坡地	15
B5	梯田	25	C2	山坡地	13

接下来我们将“2023 年的农作物种植情况”表与“地块信息整合表”二表相聚合，统一代表相同意义但不同名称的变量名。如：“种植地块”与“地块名称”所在列现实意义完全相同，我们将其进行合并。接着，依照作物编号，我们将上表与附件二中表二“2023 年统计的相关数据”进行整合，得到一份排序清晰，条目全面的总表，方便后续数据提取。

5.3 核心变量计算

5.3.1 各作物销售量计算

农产品销售量的计算不可或缺，我们假设生产者基于经验有预期地进行种植，产量大致吻合市场需求。基于这个假设，我们计算生产量即可。由生产量=（种植面积/亩）*（亩产量/斤），我们提取各作物种植面积与亩产量信息，相乘得出 2023 年各作物生产量，也即销售量，如下表：

表格 2 各作物销售量表　　销售量单位：斤

作物名称	销售量	作物名称	销售量	作物名称	销售量	作物名称	销售量
刀豆	26880	油麦菜	4500	绿豆	33040	谷子	71400
包菜	3930	爬豆	9875	羊肚菌	4200	豇豆	36240
南瓜	35100	玉米	132750	芸豆	6240	辣椒	1200
土豆	30000	生菜	1350	芹菜	1800	青椒	2610
大白菜	150000	白灵菇	18000	茄子	45360	香菇	7200
大麦	10000	白萝卜	100000	荞麦	1500	高粱	30000
小青菜	35480	空心菜	3300	莜麦	14000	黄心菜	1620
小麦	170840	红萝卜	36000	菜花	3480	黄瓜	13050
榆黄菇	9000	红薯	36000	菠菜	900	黄豆	57000
水稻	21000	红豆	22400	西红柿	36210	黍子	12500
						黑豆	21850

将表格数据以柱状图形式可视化处理：

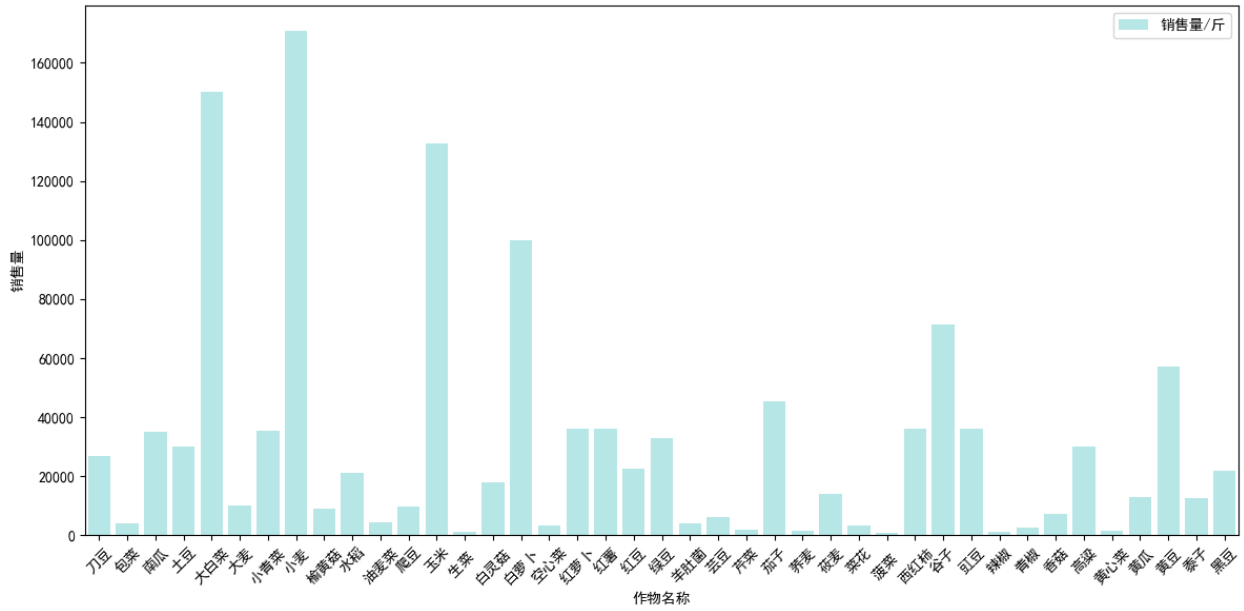


图 1 各作物销售量图

5.3.2 各作物销售价格计算

利润的计算同样需要销售价格数据。题目所给的销售单价是一个波动的范围，我们将其处理为具体的，便于计算的值。首先我们根据单价范围得出价格的上界与下界，分别记作 price_max 与 price_min。接下来我们运用 Python 的随机数代码生成 0 到 1 之间的随机数，记作系数 α。综合理性分析，我们规定定价公式为：

$$定价 = price\_min + \alpha * (price\_max - price\_min)$$

得到各产品定价如下表：

表格 3 各作物销售价格表　　销售单价单位：元/斤

作物名称	销售单价	作物名称	销售单价	作物名称	销售单价	作物名称	销售单价
刀豆	6.39	油麦菜	4.33	绿豆	6.52	谷子	6.75
包菜	7.50	爬豆	6.31	羊肚菌	103.17	豇豆	8.59
南瓜	1.45	玉米	3.03	芸豆	6.90	辣椒	7.63
土豆	3.99	生菜	5.64	芹菜	3.89	青椒	5.48
大白菜	2.83	白灵菇	16.66	茄子	5.81	香菇	19.23
大麦	3.78	白萝卜	2.85	荞麦	48.20	高粱	6.19
小青菜	6.08	空心菜	5.26	莜麦	5.04	黄心菜	5.83
小麦	3.36	红萝卜	3.33	菜花	5.71	黄瓜	7.82
榆黄菇	64.47	红薯	3.22	菠菜	5.81	黄豆	2.98
水稻	6.21	红豆	8.01	西红柿	6.83	黍子	7.50
						黑豆	7.71

六、 问题一模型的建立与求解

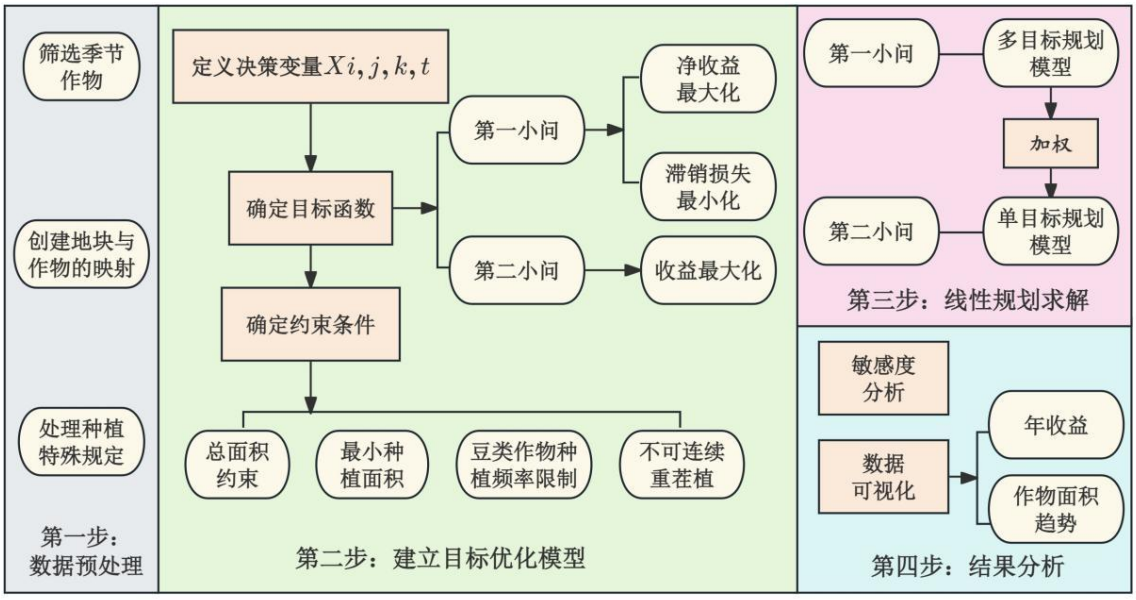


图 2 问题一思路流程图

6.1 数据处理与决策变量定义

6.1.1 筛选适合特定季度的作物

根据题目要求，乡村露天耕地包括平旱地、梯田、山坡地和水浇地四种类型，而室内大棚则分为普通大棚和智慧大棚两种类型。由于不同作物对种植季节的要求各异，为了简化计算，我们将只适合在一季种植的作物归类为适合第一季度种植的作物。

为实现这一目标，我们在 Python 中自定义函数，该函数根据输入的季节参数，从数据中筛选出适合该季节种植的作物名称列表。具体来说，若季节参数为第一季度，函数将考虑水浇地、普通大棚、智慧大棚、平旱地、梯田和山坡地等所有土地类型在第一季度的适宜作物。如果季节参数为第二季度，则函数仅考虑水浇地、普通大棚和智慧大棚在第二季度的适宜作物。

6.1.2 根据地块和季度综合因素进行作物筛选

①作物与地块类型的适应性

定义三元决策变量 $a_{i,j,k}$ ：

$$a_{i,j,k}=\begin{cases}1, & \text{表示作物 } i \text{ 在季度 } k \text{ 可以种植于地块 } j\\ 0, & \text{表示作物 } i \text{ 在季度 } k \text{ 不能种植于地块 } j\end{cases}$$

适应性约束可以表示为：

$$x_{i,j,k,t} \le a_{i,j,k} \cdot A_j,\ \forall i,j,k \in \{1,2\}, t \in \{2024,\dots,2030\}$$

其中：

地块类型 T_j ：代表地块 j 的类型。如平旱地、梯田、山坡地、水浇地、普通大棚、智慧大棚。

作物类型 S_i ：代表作物 i 的类型。如粮食、粮食（豆类）、蔬菜、蔬菜（豆类）、食用菌。

季度 k :表示种植季度，取值为 1 或 2，分别对应第一季度和第二季度。

②大棚内作物种植的季节性

对于“普通大棚”$(T_j = T_5)$，每年只能种植一季蔬菜和一季食用菌：

$$\sum_{i\in\{i|S_i=S_3\}} x_{i,j,1,t} + \sum_{i\in\{i|S_i=S_5\}} x_{i,j,2,t} \le A_j,\ \forall j \text{ with } T_j = T_5, t \in \{2024,\dots,2030\}$$

对于“智慧大棚”$(T_j = T_6)$，每年可以种植两季蔬菜：

$$\sum_{i\in\{i|S_i=S_3\}} x_{i,j,k,t} \le A_j,\ \forall j \text{ with } T_j = T_6, k \in \{1,2\}, t \in \{2024,\dots,2030\}$$

③特殊规定处理

如水浇地每年可单季种植水稻，不再种另一季作物。

为了处理这一特殊规定，我们在 Python 中自定义函数来调整土地与作物的映射关系，以确保如果某块水浇地在某年种植了水稻，则在同一年该地不再种植其他作物。

6.1.3 决策变量定义

定义$x_{i,j,k,t}$：表示在第 t 年第 k 个季度在第 j 个地块上种植第 i 种作物的面积。

其中，变量所属集合如下表所示：

表格 4 变量所属集合

$i \in I$	作物类型（如蔬菜、豆类、食用菌等）
$j \in J$	地块类型（如大棚、梯田、山坡地等）
$t \in \{2024,2025,\dots,2030\}$	种植年度
$k \in \{1,2\}$	种植季度

6.2 第一小问模型的建立与求解

在假设未来农作物的预期销售量、种植成本、亩产量和销售价格保持稳定的情况下，该问题要求制定一个从 2024 年到 2030 年的最优蔬菜种植方案。尤其需要考虑到当某种作物的总产量超过预期销售量时，超出部分将无法正常销售，从而导致浪费。因此，我们建立了一个基于多目标规划的种植策略模型，以实现乡村在 2024 年至 2030 年期间的净收益最大化，并尽量减少滞销损失。

6.2.1 目标函数确立

鉴于超过预期销售量生产的作物将面临滞销，导致浪费，我们的核心目标是在 2024 年至 2030 年的时间框架内，实现净收益的最大化，并同时最小化因滞销而带来

的损失。

(1)目标函数 1：净收益最大化

$$\text{Maximize: } Z_1 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_{i} \left(p_i \cdot \min\left(y_{i,k,t}, D_i\right) - c_i \cdot \sum_{j} x_{i,j,k,t} \right)$$

(2)目标函数 2：滞销损失最小化

$$\text{Minimize: } Z_2 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_{i} \left(p_i \cdot \max\left(y_{i,k,t} - D_i, 0\right) \right)$$

其中：

$$y_{i,k,t} = \sum_{j} x_{i,j,k,t} \cdot q_i$$

各项表达式定义如下表：

表格 5 表达式定义

符号	定义	单位
p_i	第 i 种作物的销售价格	元/亩
c_i	第 i 种作物的种植成本	元/亩
q_i	第 i 种作物的亩产量	千克/亩
D_i	第 i 种作物的预期销售量	千克
A_j	第 j 个地块或大棚的总面积	亩
$y_{i,k,t}$	第 i 种作物在第 t 年第 k 季度的总产量	千克
$\min(y_{i,k,t}, D_i)$	表示作物 i 在第 t 年第 k 季度的实际销售量	千克
$p_i \cdot \min(y_{i,k,t}, D_i)$	表示作物 i 的实际销售收入	元
$c_i \cdot \sum_j x_{i,j,k,t}$	表示作物 i 的种植成本	元
$\max(y_{i,k,t} - D_i, 0)$	表示超过预期销售量的部分	千克
Z_1	是 2024 年至 2030 年期间的总收益	元
Z_2	滞销造成的损失	元

6.2.2 约束条件确立

为了精确描述地块和作物的相关属性，我们采用一系列符号来表示。具体而言，对于地块名称，我们定义符号 P_j 来代表第 j 个地块，其中 j 是地块的编号。例如，P_{A1} 即表示地块 A_1。

同时，我们赋予地块类型特定的符号：平旱地对应 T_1，梯田对应 T_2，山坡地对应 T_3，水浇地对应 T_4，普通大棚对应 T_5，智慧大棚对应 T_6。

此外，地块的面积也是一个重要属性，我们用符号 A_j 来表示第 j 个地块的面积，单位为亩。例如，A_{A1} 表示地块 A1 的面积。

在讨论地块上种植的作物时，我们引入作物编号 i 来区分不同的作物种类。每种作物的名称由符号 C_i 表示。如 C_1 为黄豆，表示编号为 1 的作物是黄豆。

作物的类型则由符号 S_i 来标识，它描述了作物的种类归属。这些类型涵盖粮食、粮食（豆类）、蔬菜、蔬菜（豆类）以及食用菌。同样地，为了简化表达，我们可以将每种作物类型赋予一个特定的符号：粮食对应 S_1，粮食（豆类）对应 S_2，蔬菜对应 S_3，蔬菜（豆类）对应 S_4，食用菌对应 S_5。这样的符号体系有助于在数学建模过程中清晰地表达地块及其作物的各种属性。

整理上述思路得下表：

表格 6 约束条件符号及其说明

符号	说明
P_j	表示第 j 个地块，其中 j 为地块的编号
A_j	表示第 j 个地块的面积，其中 j 为地块的编号
i	第 i 种作物的编号
C_i	第 i 种作物的名称
S_i	第 i 种作物的类型
平旱地:T_1 梯田:T_2 山坡地:T_3 水浇地:T_4 普通大棚:T_5 智慧大棚:T_6 粮食:S_1 粮食 (豆类) :S_2 蔬菜:S_3 蔬菜 (豆类): S_4 食用菌:S_5	

现讨论约束条件的确立过程：

①约束条件 1：总面积约束

对于每个地块 P_j ，其每季度所有作物种植面积总和不能超过地块总面积 A_j：

$$\sum_i x_{i,j,k,t} \le A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,\dots,2030\}$$

②约束条件 2：最小种植面积约束

为确保每种作物在单个地块的种植面积不会过小，我们设定地块面积的 15%为每种作物在该地块的最小种植面积。表达式即：

$$x_{i,j,k,t} \ge 0.15 \cdot b_{i,j,k,t},\ \forall i,j,k \in \{1,2\}, t \in \{2024,\dots,2030\}$$

其中， $b_{i,j,k,t}$ 是一个二元变量，$b_{i,j,k,t} = \begin{cases} 1, & x_{i,j,k,t} > 0 \\ 0, & x_{i,j,k,t} < 0 \end{cases}$ 。

③约束条件 3：非连续重茬种植约束

为防止同一地块上同种作物连续重茬种植，需要确保对于每个地块 P_j，作物 i 不能在相邻的两年或两季连续种植：

$$x_{i,j,k,t} \cdot x_{i,j,k,t+1} = 0,\ \forall i,j,k \in \{1,2\}, t \in \{2024,\dots,2029\}$$

④约束条件 4：豆类种植频率约束

每个地块在三年内至少需要种植一次豆类作物。这可以通过计算在连续的三年内，地块上所有豆类作物种植面积的总和是否大于地块面积来实现。豆类作物集合由 S_2 和 S_4 表示。描述可以是：

$$\sum_{t'=t}^{t+2} \sum_{i \in \{i \mid S_i \in \{S_2, S_4\}\}} x_{i,j,k,t'} \ge A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,2025,2026,2027,2028\}$$

6.2.3 优化模型确立

综合以上目标函数与约束条件，我们可以得出这个基于多目标规划的种植策略模型：

$$\text{目标函数} \begin{cases} \max: Z_1 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_i \left(p_i \cdot \min\left(y_{i,k,t}, D_i\right) - c_i \cdot \sum_j x_{i,j,k,t} \right) \\ \min: Z_2 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_i \left(p_i \cdot \max\left(y_{i,k,t} - D_i, 0\right) \right) \end{cases}$$

$$
\text{约束条件}\begin{cases}
\sum_{i} x_{i,j,k,t} \le A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,\ldots,2030\} \\
x_{i,j,k,t} \ge 0.15 \cdot b_{i,j,k,t},\ \forall i,j,k \in \{1,2\}, t \in \{2024,\ldots,2030\} \\
x_{i,j,k,t} \cdot x_{i,j,k,t+1} = 0,\ \forall i,j,k \in \{1,2\}, t \in \{2024,\ldots,2029\} \\
\sum_{t'=t}^{t+2} \sum_{i \in \{i \mid S_i \in \{S_2,S_4\}\}} x_{i,j,k,t'} \ge A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,2025,2026,2027,2028\}
\end{cases}
$$

在原始的多目标规划问题中，我们有两个目标函数：最大化净收益（Z_1）和最小化滞销损失（Z_2）。为了简化问题求解，我们采用加权法将这两个目标函数合并为一个单目标函数。我们认为净收益最大化和滞销损失最小化同等重要，因此选取两个目标函数的权重均为 0.5。由于两个目标函数的单位相同，无需进行量纲的标准化处理，得到加权后的目标函数：。

$$
\text{Maximize: Z} = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_{i} \left(p_i \cdot \min\left(y_{i,k,t}, D_i\right) - c_i \cdot \sum_{j} x_{i,j,k,t} - p_i \cdot \max\left(y_{i,k,t} - D_i, 0\right) \right)
$$

利用 Python 执行线性规划求解，并将结果填入对应表格。以下是部分种植方案示例：

表格 7 种植方案示例

季别	地块名	黄豆	黑豆	红豆	绿豆	爬豆	小麦	玉米	谷子	高粱	黍子	荞麦
第一季	B1	6	0	6	6	6	0	6	0	6	0	12
第一季	B2	0	4.6	0	4.6	4.6	4.6	4.6	4.6	0	4.6	0
第一季	B3	0	0	4	4	0	4	4	0	4	0	8
第一季	B4	2.8	2.8	2.8	2.8	0	0	2.8	0	2.8	0	5.6
第一季	B5	0	2.5	2.5	2.5	2.5	2.5	0	0	0	0	2.5
第一季	B6	0	8.6	8.6	8.6	8.6	0	0	8.6	0	0	17.2
第一季	B7	0	0	5.5	5.5	5.5	0	5.5	5.5	0	5.5	5.5
第一季	B8	0	0	4.4	0	4.4	0	0	4.4	4.4	4.4	8.8

6.2.4 结果分析

(1)优化结果

求解该单目标规划模型，得到最佳种植方案，其结果部分展示如下：

表格 8 最佳种植方案部分展示

	地块名	黄豆	黑豆	红豆	绿豆	爬豆	小麦	玉米	谷子
2024 年第一季	A1	0	0	0	0	0	20	0	20
	A2	0	18.33	0	0	0	0	0	0
	A3	0	0	0	17.5	0	0	17.5	0
	A4	0	0	18	0	0	0	0	0
	A5	0	0	20.4	0	0	0	0	0
	A6	0	0	0	0	13.75	0	13.75	0
	B1	0	0	20	0	0	0	20	0
	B2	11.5	0	0	0	0	11.5	0	11.5

B3	0	7.2	0	0	0	7.2	0	0
B4	0	0	6.3	0	0	0	0	0
B5	0	0	0	0	5.625	0	0	5.625
B6	0	0	15.48	0	0	0	0	0

各年度总收益如下所示（单位：千元）：

表格 9 2024-2030 年度总收益

年份	总收益
2024	10988.72882
2025	9653.462671
2026	9953.494962
2027	9716.14296
2028	9875.225101
2029	11236.55598
2030	9041.98695

(2)灵敏度分析

从定量分析的角度研究加权后的目标函数 Z 发生某种变化对某一个或者一组关键指标影响程度的不确定分析，通过改变相关变量数值的方法来解释关键指标收到净收益和滞销损失影响大小的规律。

改变 Z1 和 Z2 权重，研究发现总收益的变化趋势。由下图可知，当权重之比为 1：1 时，总收益会达到最大值 10988.728（千元），其他比值时，总收益都会有所下降。

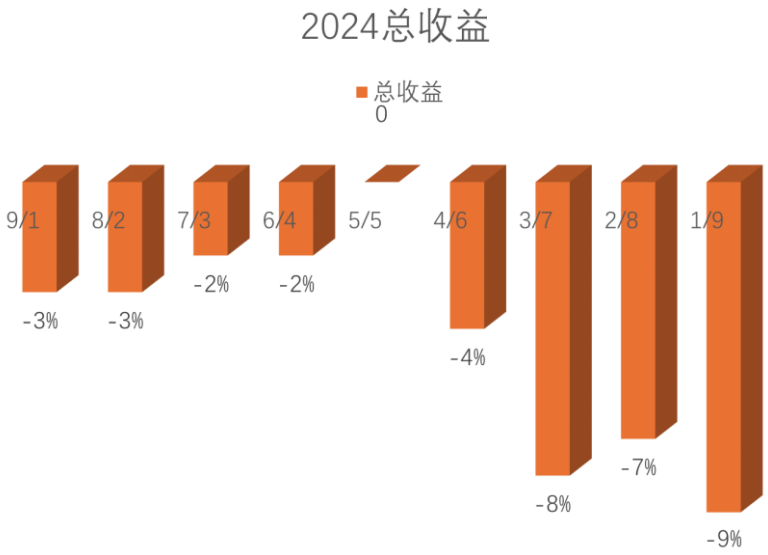


图 3 2024 年总收益灵敏度分析

(3)数据可视化

得到种植最优方案后绘制年度总收益折线图，直观显示了不同年份的年度总收益变化情况。时间跨度为 2024 年至 2030 年，总收益数值在一定范围内波动，可以看出在年度总收益均在较高水平。

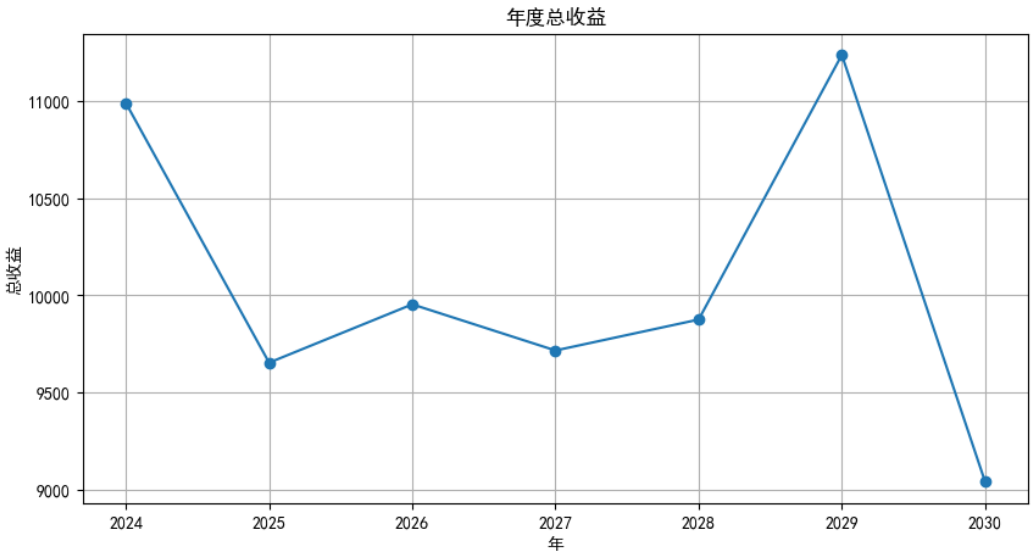


图 4 年度总收益折线图

6.3 第二小问模型的建立与求解

第二小问和第一小问的条件相同，只是对作物总产量超过预期销售量的部分处理方式不同。第二小问选择将超过部分折扣销售。因此利用线性规划模型求解时，只需变换目标函数，而决策变量和约束条件保持一致，建立以 2024 年至 2030 年期间的总收益目标规划模型求解即可。

6.3.1 目标函数确立

目标函数表示为 2024 年至 2030 年期间的总收益最大化，其中超过预计销售额部分作物按 2023 年销售价格的 50%降价出售：

$$\text{Maximize: } Z_3 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_{i} \left(p_i \cdot \min\left(y_{i,k,t}, D_i\right) + p_i' \cdot \max\left(y_{i,k,t} - D_i, 0\right) - c_i \cdot \sum_{j} x_{i,j,k,t} \right)$$

其中：$p_i' = 0.5 * p_i$，代表超过部分折扣处理。

对该函数的各表达式理解如下表：

表格 10 目标函数各表达式具体含义

表达式	含义
Z_3	是 2024 年至 2030 年期间的总收益
$\min\left(y_{i,k,t}, D_i\right)$	表示作物 i 在第 t 年第 k 季度的正常销售量
$\max\left(y_{i,k,t} - D_i, 0\right)$	表示超过预期销售量的部分
$p_i \cdot \min\left(y_{i,k,t}, D_i\right)$	是正常销售的部分收入
$p_i' \cdot \max\left(y_{i,k,t} - D_i, 0\right)$	是按折扣价销售的部分收入
$c_i \cdot \sum_j x_{i,j,k,t}$	表示种植成本

6.3.2 优化模型确立

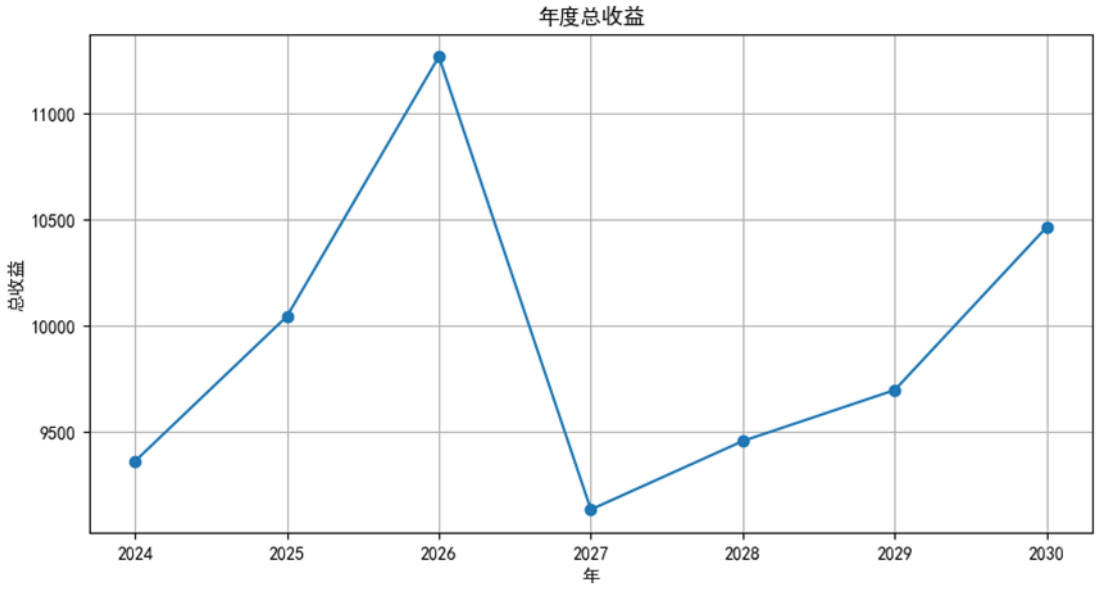
综合以上可以得出单目标规划模型：

目标函数：$\max: Z_3 = \sum_{t=2024}^{2030} \sum_{k=1}^{2} \sum_{i} \left(p_i \cdot \min(y_{i,k,t}, D_i) + p_i' \cdot \max(y_{i,k,t} - D_i, 0) - c_i \cdot \sum_{j} x_{i,j,k,t} \right)$

约束条件
$$\begin{cases} \sum_{i} x_{i,j,k,t} \le A_j, \ \forall j,k \in \{1,2\}, t \in \{2024, \dots, 2030\} \\ x_{i,j,k,t} \ge 0.15 \cdot b_{i,j,k,t}, \ \forall i,j,k \in \{1,2\}, t \in \{2024, \dots, 2030\} \\ x_{i,j,k,t} \cdot x_{i,j,k,t+1} = 0, \ \forall i,j,k \in \{1,2\}, t \in \{2024, \dots, 2029\} \\ \sum_{t'=t}^{t+2} \sum_{i \in \{i \mid S_i \in \{S_2, S_4\}\}} x_{i,j,k,t'} \ge A_j, \ \forall j,k \in \{1,2\}, t \in \{2024,2025,2026,2027,2028\} \end{cases}$$

6. 3. 3 结果分析

绘制年度总收益折线图，图像直观显示了不同年份的年度总收益变化情况。可以看出 2024 年至 2030 年年度总收益数值产生一定程度的波动，但总体维持较高水平。



七、 问题二模型的建立与求解

问题二要求我们在问题一的基础上考虑销售量、亩产量、种植成本和销售价格的波动，引入风险评估参数，并综合考虑其他隐性影响因素后，给出该乡村 2024~2030 年农作物的最优种植方案。

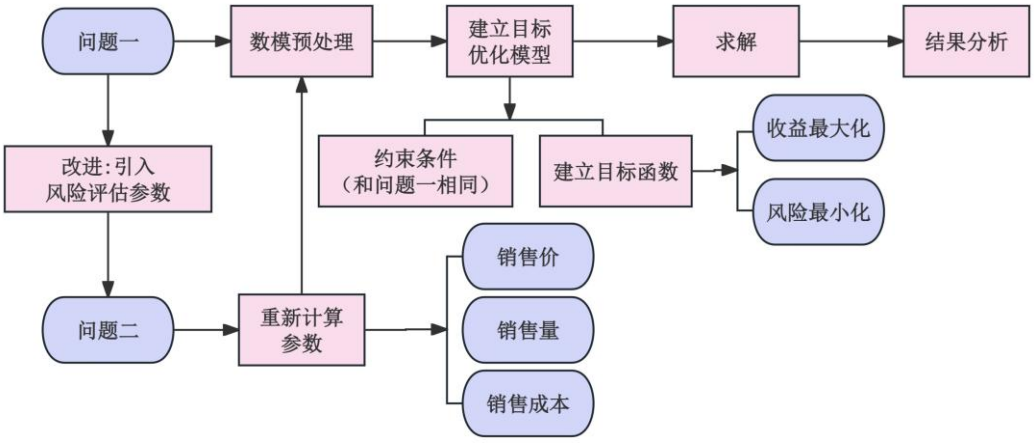


图 5 问题二思路流程图

7.1 模型的建立

7.1.1 建立作物—地块映射

我们根据地块类型和季节筛选作物，建立作物与地块的映射关系，如下表所示：

表格 6 作物—地块映射关系

作物	地块	作物	地块	作物	地块	作物	地块	作物	地块
黄豆 黑豆 红豆 绿豆 爬豆 小麦 玉米 谷子	平旱地 梯田 山坡地	高粱 黍子 荞麦 南瓜 红薯 莜麦 大麦	平旱地 梯田 山坡地	豇豆 刀豆 芸豆 土豆 西红柿 茄子 菠菜 青椒 菜花	水浇地 第一季 普通大棚 第一季 智慧大棚 第一季 第二季	包菜 油麦菜 小青菜 黄瓜 生菜 辣椒 空心菜 黄心菜 芹菜	水浇地 第一季 普通大棚 第一季 智慧大棚 第一季 第二季	水稻 大白菜 白萝卜 红萝卜 榆黄菇 香菇 白灵菇 羊肚菌	水浇地 水浇地 第二季 普通大棚 第二季

7.1.2 引入风险评估参数

农业生产，无论对于发达国家还是发展中国家而言，一直都是一种高风险的活动。农业风险的来源有许多，主要包括环境、气候和生物系统本身的内在不确定性以及农业投入产出品市场价格的外在不确定性[2]。

因此，我们引入风险评估参数Risk assessment $\in [-1,1]$，其中，当风险越大时，Risk assessment 的取值越大。以此更好地规避种植风险，制定最优种植方案。

不同作物的风险值不同，经济作物的生产风险要高于粮食作物。粮食作物适当降低风险，如设置为-0.05；蔬菜作物稍微增加风险，如设置为 0.05，食用菌类作物受环境等影响更大，如设置为 0.1。不同作物风险评估参数如下表：

表格 7 作物风险评估参数表

作物类型	风险评估参数
粮食	-0.05
粮食（豆类）	-0.05
蔬菜	0.05
蔬菜（豆类）	0.05
食用菌	0.1

7.1.3 根据风险评估参数重新计算主要参数

高风险可能会导致销售价适当下降，销售量适当减少，成本相应增加。故引入风险评估参数后要重新计算各个主要参数。现以蔬菜类作物为例，展示参数的计算过程。

(1)销售价格 $price_{after} = price_{befre} * (1 + 0.05) ** (year - 2023)$

(2)销售量 $sales_{volumn} = sales_{volumn_{lost}} * (1 - 0.05) ** (year - 2023)$

(3)销售成本 $sales\ cost_{after} = sales\ cost_{last} * (1 + 0.05) ** (year - 2023)$

7.1.4 目标函数和约束条件的选择

目标是在综合考虑各种农作物的销量、产量、成本等主要参数的不确定性以及潜在的种植风险的情况下，求解最优种植方案，可知目标函数可设置为同时满足收益最大化和风险最小化。

约束条件可大致分为四类，分别为总面积约束、最小种植面积约束、三年内至少种植一次豆类作物约束和不可连续重茬种植约束。

具体目标函数和约束表达式见模型求解部分。

7.2 模型的求解

7.2.1 目标函数确立

目标要求最大化收益和最小化风险，依此设置以下目标函数：

$$\text{目标函数}\begin{cases}\max\ Z = price_{after} * sales_{volumn} - sales\ cost_{after} \\ \min \sum_{j=1}^{5} \left| Risk\ assessment_j \right|\end{cases}$$

7.2.2 约束条件确立

问题二的约束条件与问题一相同，包括总面积约束，最小种植面积约束，非连续重茬种植约束和豆类种植频率约束，表达式分别如下所示：

$$\text{约束条件}\begin{cases}\sum_{i} x_{i,j,k,t} \le A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,\dots,2030\} \\ x_{i,j,k,t} \ge 0.15 \cdot b_{i,j,k,t},\ \forall i,j,k \in \{1,2\}, t \in \{2024,\dots,2030\} \\ x_{i,j,k,t} \cdot x_{i,j,k,t+1} = 0,\ \forall i,j,k \in \{1,2\}, t \in \{2024,\dots,2029\} \\ \sum_{t'=t}^{t+2} \sum_{i \in \{i \mid S_i \in \{S_2,S_4\}\}} x_{i,j,k,t'} \ge A_j,\ \forall j,k \in \{1,2\}, t \in \{2024,2025,2026,2027,2028\}\end{cases}$$

7.2.3 求解并可视化处理

我们利用 Python 求解该优化模型，将最优种植方法导出成 excel 文件，如下所示：

表格 8 2024-2030 总收益表

年份	总收益
2024	29401000
2025	29099000
2026	28756000
2027	28418000
2028	29132000
2029	27504000
2030	27704000

7.3 模型改进

问题二先前使用的方法为建立多目标优化模型。而多个目标同时优化可能存在局部最优解情况，无法准确找到全局最优解。全局收敛性较低。使用**粒子群算法**适用于求解非线性高维度函数优化问题，全局收敛性良好。

7.3.1 模型建立

我们引入随机性来模拟上述因素的波动，使用**蒙特卡罗模拟**处理不确定性因素的模拟。蒙特卡洛模拟通过重复随机抽样来模拟农作物预期销售量、亩产量、种植 成本和销售价格的不确定性。这种方法假设每个变量的波动符合某种概率分布 （例如，正态分布或均匀分布），通过随机生成不同的情景来分析决策的表现。

不确定性因素：

- 预期销售量：假设小麦和玉米的年增长率为 5%-10%，其他作物每年波动±5%。
- 亩产量：每年有±10%的波动。
- 种植成本：每年平均增长 5%。
- 销售价格：不同作物价格有不同的变化趋势（例如，蔬菜类增 长 5%，食用菌下降 1%-5%）。

每个变量都可以用随机变量来表示：

$$D_{t,j} = D_{2023.j} \times (1 + r_{D_{t,j}})$$

$$q_{t,j} = q_{2023.j} \times (1 + r_{q_{t,j}})$$

$$c_{t,j} = c_{2023.j} \times (1 + r_{c_{t,j}})$$

$$p_{t,j} = p_{2023.j} \times (1 + r_{p_{t,j}})$$

其中，$r_{D_{t,j}}$、$r_{q_{t,j}}$、$r_{c_{t,j}}$、$r_{p_{t,j}}$分别为每个变量的增长率或波动率，服从相应的分布。

7.3.2 模型求解

粒子群算法是一种基于群体智能的优化算法，具有收敛速度快、全局搜索能力强等特点，非常适合求解非线性多目标优化问题。

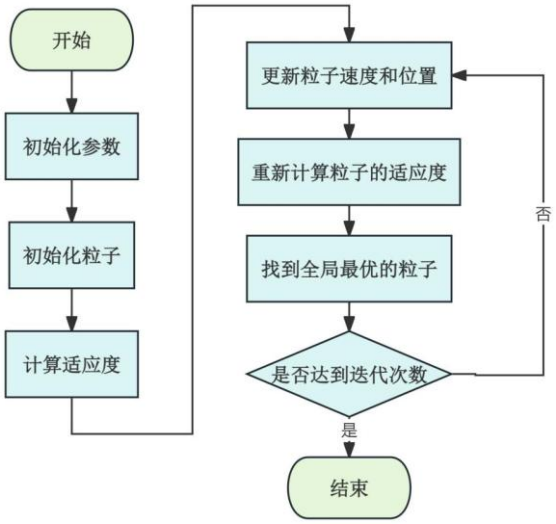


图 6 粒子群算法流程图

(1)初始化

①定义粒子群算法的参数

表格 14 定义粒子群算法参数及其含义

参数	含义
w	惯性权重，影响粒子的运动惯性
c1	个体学习因子，决定粒子向自身历史最优位置移动的步长
c2	社会学习因子，决定粒子向全局最优位置移动的步长
num_particles	粒子的数量
max_iter	最大迭代次数
num_simulations	蒙特卡罗模拟的次数

②定义问题参数

表格 15 问题参数及其含义

参数	含义
num_years	年份的数量
num_crops	作物种类的数量
num_plots	地块的数量
num_seasons	季节的数量

③初始化

首先设置最大迭代次数，目标函数的自变量个数，粒子的最大速度，位置信息为整个搜索空间，我们在速度区间和搜索空间上随机初始化速度和位置，设置粒子群规模为 M，每个粒子随机初始化一个运动速度。初始化每个粒子的最佳位置和全局最佳位置，创建列表用于记录最佳适应度和迭代次数。

(2)设定函数

①模拟参数的生成函数

函数用于模拟生成销售价格、种植成本、亩产量和预期销售量等参数。对于不同类型的作物，采用蒙特卡罗模拟，根据不同的正态分布进行参数的调整，例如小麦和玉米的预期销售量增长、亩产量变化、种植成本增长和销售价格变化等。

②目标函数

给定粒子位置和模拟参数下的目标函数值，计算销售收益减去种植成本的总和，实现收益最大化：

$$\max \quad Z = price_{after} * sales_{volumn} - sales\ cost_{after}$$

约束条件和问题一相同，这里不再展示。

(3)迭代过程

①调用 pso 函数

调用 pso 函数后得到最优解和对应的目标函数值

更新速度和位置公式：

$$V_{id} = \omega V_{id} + C_1 random(0,1)(P_{id} - X_{id}) + C_2 random(0,1)(P_{gd} - X_{id})$$

$$X_{id} = X_{id} + V_{id}$$

其中，ω 称为惯性因子其值为非负，较大时，全局寻优能力强，局部寻优能力强，较小时，全局寻优能力弱，局部寻优能力强。通过调整 ω 的大小，可以对全局寻优性

能和局部寻优性能进行调整。C_1和C_2称为加速常数，前者为每个粒子的个体学习因子，后者为每个粒子的社会学习因子。Suganthan 的实验表明:C_1和C_2,为常数时可以得到较好的解，通常设置$C_1 = C_2 = 2$，但不一定必须等于 2，一般取$C_1 = C_2 \in [0,4]$。$random(0,1)$表示区间[0,1]上的随机数，P_{id}表示第 i 个变量的个体极值的第 d 维，P_{gd}表示全局最优解的神经网络与数学建模第 d 维。

②终止条件

（1）达到设定迭代次数（2）代数之间的差值满足最小界限

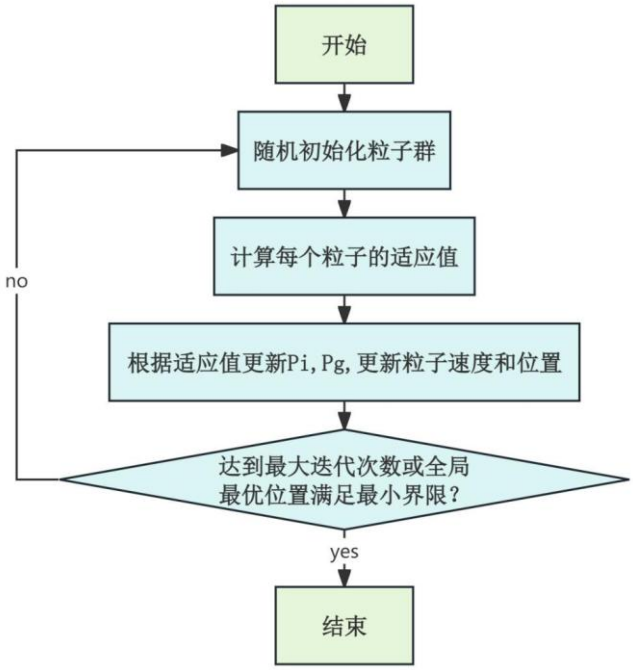


图 7 粒子群算法迭代流程图

7. 3. 3 结果分析

综合上述分析，最后利用 python 的 matplotlib 库绘制粒子群优化时的图像：

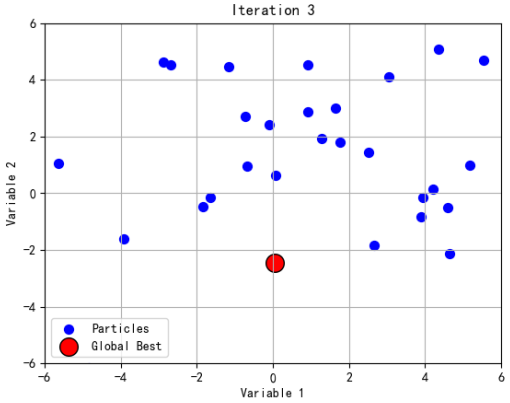


图 8 粒子群算法示意图 1

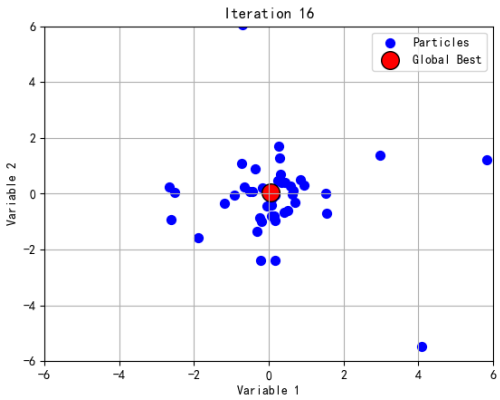


图 9 粒子群算法示意图 2

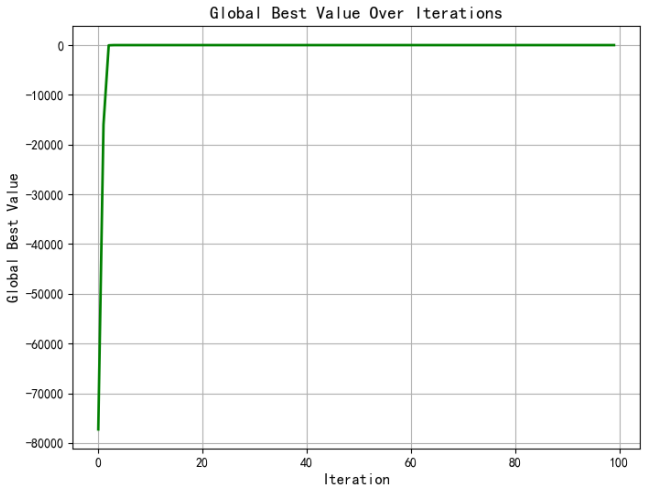


图 10 粒子群算法循环次数图

由图像可知，经过 10 次左右的循环之后，数据基本趋于稳定。最终结果如下表所示：

表格 9 基于粒子群算法得出的 2024-2030 总收益表

年份	总收益
2024	29914000
2025	29648000
2026	28920000
2027	28840000
2028	29085000
2029	28167000
2030	28561000

基于粒子群优化算法的求解过程，用于在给定的问题参数和数据下找到最优的种植方案，以最大化总收益，同时通过模拟参数的变化和多次蒙特卡罗模拟来提高结果的可靠性，方法优于简单地引入风险评估参数计算。

八、 问题三模型的建立与求解

8.1 相关性分析

问题三要求我们探讨预期销售量与销售价格、种植成本之间的关系，并且研究各农作物之间的可替代性和互补性。首先，附件将作物大致分为五类，分别为粮食类、粮食（豆类），蔬菜类，蔬菜（豆类）和食用菌类。故可使用皮尔逊相关性分析的方法探究五类作物预期销售量与销售价格、种植成本之间的相关性。

皮尔逊相关性是一种衡量两种变量相关性程度的方法，两个连续变量(X,Y)的 Pearson 相关性系数 P(X,Y)等于它们之间的协方差 COV(X,Y)除以它们各自标准差的乘积(σ_x,σ_y)。系数的取值总是在-1 到 1 之间，接近 0 的变量被称为无相关性，接近 1 或者-1 被称为具有强相关性。

$$\rho_{X,Y}=\frac{cov(X,Y)}{\sigma_x\sigma_y}=\frac{E\left((X-\mu_x)(Y-\mu_y)\right)}{\sigma_x\sigma_y}=\frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\sqrt{E(Y^2)-E^2(Y)}}$$

上式可以描述变量 X 和 Y 之间的相关性程度，其中$\rho_{X,Y}$代表总体相关系数，可以用于描述两个变量之间的总体关系。皮尔逊相关系数 r 可以用样本的协方差和标准差进行估算：

$$r=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}=\frac{1}{n-1}\sum_{i=1}^{n}(\frac{X_i-\bar{X}}{\sigma_X})(\frac{Y_i-\bar{Y}}{\sigma_Y})$$

其中，$\frac{X_i-\bar{X}}{\sigma_X}$代表变量的标准分数，$\bar{X}$代表变量的平均值、$\sigma$代表变量的标准差。

使用 Python 的 seaborn 库求解皮尔逊相关性问题，系数热力图如下所示：

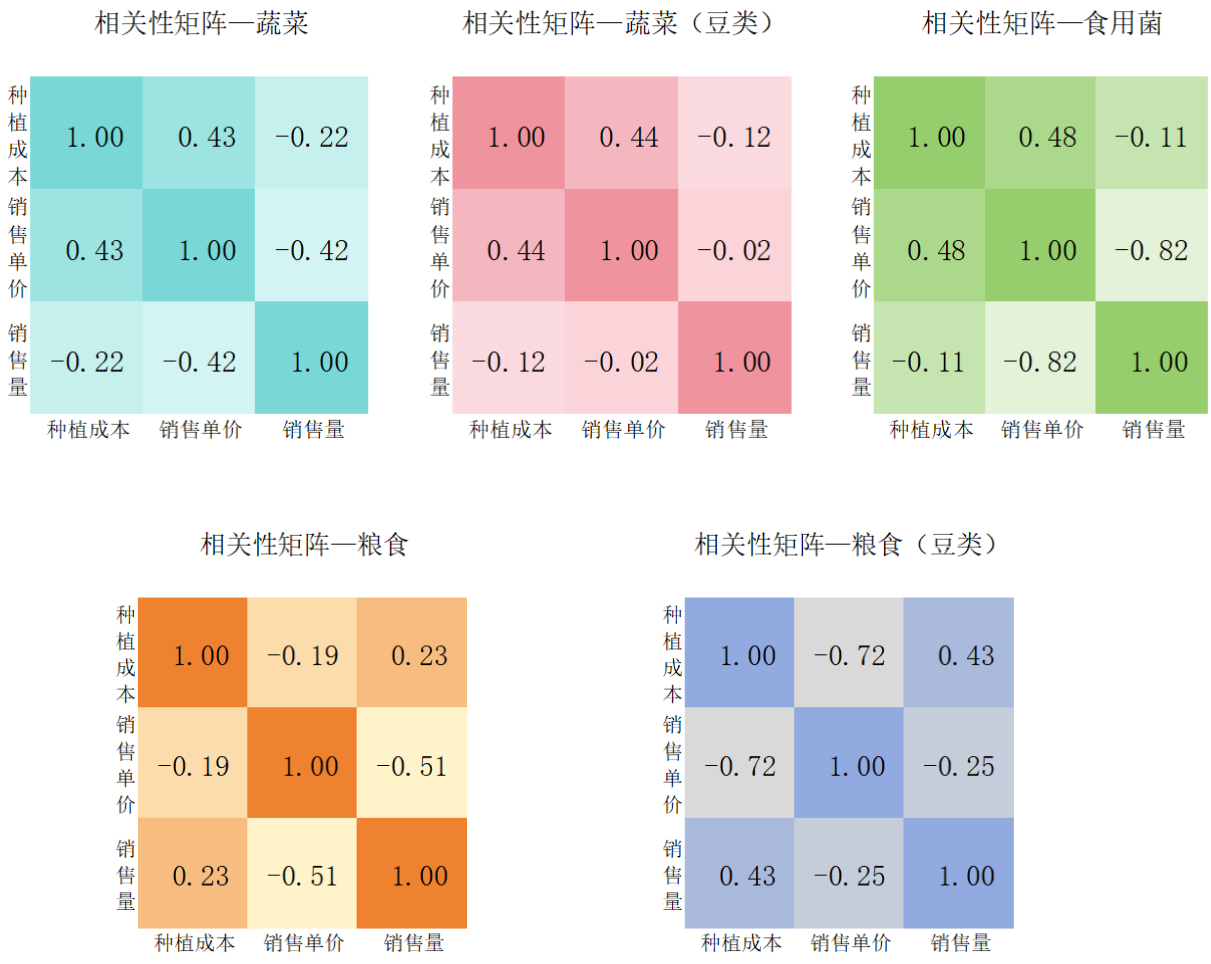


图 11 Pearson 相关系数热力图

对于粮食类，种植成本和销售单价相关性较小，销售单价和销售量相关性较大；对于粮食（豆类），种植成本和销售单价相关性较高；对于蔬菜类，三者之间相关性相似；对于蔬菜（豆类），销售单价和销售量之间相关性较低；对于食用菌类，销售单价和销售量之间相关性较高，销售量和种植成本之间相关性较低。

综合上述分析，针对粮食类，研究种植成本和销售单价之间的关系；针对粮食（豆

类），研究种植成本和销售单价之间的关系；针对蔬菜类，研究销量、种植成本和销售单价之间的关系；针对蔬菜（豆类），研究销售单价和销售量之间的关系；针对食用菌类，研究销售量和种植成本之间的关系。通过已有数据绘制散点图，再通过散点图拟合函数，进一步探究不同类作物的种植成本、销售单价和销售量之间的关系。拟合方法有很多种，可分为线性拟合、多项式拟合、多元线性拟合、高斯函数拟合和分段多项式拟合等。

通过上述数据分析，粮食类选用多项式拟合，粮食（豆类）选用线性拟合，蔬菜类选用多项式拟合，蔬菜（豆类）选用多项式拟合，食用菌类选用多项式拟合。

8.2 拟合函数分析

8.2.1 多项式回归

多项式回归是研究一个因变量与一个或多个自变量间的回归分析方法，其最大优点就是可以通过增加 x 的高次项对实测点进行逼近，直至满意为止。

多项式回归的数学原理是基于最小二乘法，再通过定义一个误差距离变量，使其最小化来推算多项式函数的系数，然后利用这些系数构建多项式函数，达到拟合数据的效果。

例如，假设多项式函数为：

$$\hat{Y} = a_k * x^k + a_{k-1} * x^{k-1} + \cdots + a_1 * x + a_0$$

定义一个误差距离变量，既是各点到这条曲线的距离之和，也是误差的平方和为

$$R^2 = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n} [Y_i - (a_k * x^k + a_{k-1} * x^{k-1} + \cdots + a_1 * x + a_0)]^2$$

再根据最小二乘原理，等式两边同时对a_i求偏导数，如下所示：

$$-2\sum_{i=1}^{n}\left[Y_i - \left(a_k * x_i^k + a_{k-1} * x_i^{k-1} + \cdots + a_1 * x_i + a_0\right)\right] = 0$$

$$-2\sum_{i=1}^{n}\left[Y_i - \left(a_k * x_i^k + a_{k-1} * x_i^{k-1} + \cdots + a_1 * x_i + a_0\right)\right] * x_i = 0$$

……

$$-2\sum_{i=1}^{n}\left[Y_i - \left(a_k * x_i^k + a_{k-1} * x_i^{k-1} + \cdots + a_1 * x_i + a_0\right)\right] * x_i^k = 0$$

将上述方程化简得：

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (a_0 + a_1 * x_i + \cdots + a_k * x_i^k)$$

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (a_0 * x_i + a_1 * x_i^2 + \cdots + a_k * x_i^{k+1})$$

……

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (a_0 * x_i^k + a_1 * x_i^{k+1} + \cdots + a_k * x_i^{2k})$$

通过线性代数的知识可以上述方程组转化为矩阵形式：

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^k \\ 1 & x_2 & \dots & x_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & \dots & x_k^k \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_k \end{bmatrix}$$

将已有数据点带入方程组求解之后，即可求解出拟合多项式的系数大小。

8.2.2 线性回归

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。其核心思想是找到一个最佳拟合直线，使得该直线能够最小化所有数据点到该直线的垂直距离，即残差平方和。通过最小化残差平方和，线性回归模型能够找到数据中的线性关系，并对新的数据进行预测。

多元线性回归的函数一般表达式为：

$$Y_i = \alpha_m * x_m + \alpha_{m-1} * x_{m-1} + \cdots + \alpha_1 * x_1 + \alpha_0$$

建立残差平方和参数，并使其最小化：

$$\min \sum_{i=1}^{n} (Y_i - \overline{Y_l})^2$$

代入散点图数据，使其残差平方和参数最小化，即可求解出线性回归的参数大小。

8.3 收益最大化

问题三要求我们考虑各个作物之间的可替代性和互补性，再综合问题一和问题二的目标函数和约束条件，求出该乡村 2024 年~2030 年农作物的最优种植策略。

8.3.1 作物之间替代性和互补性

(1)替代性：

若两种作物之间具有替代性关系，则认为两者之间可以相互替代，如当作物 1 亩产量减少时，可以通过种植作物 2 来弥补收益损失。我们可以用需求交叉弹性理论描述商品之间的替代性，其计算公式为：

$$E_{yx} = \frac{Y\text{商品需求量变动百分比}}{X\text{商品价格变动百分比}} = \frac{\Delta {Q_y}/{Q_y}}{\Delta {P_x}/{P_x}} = \frac{\Delta Q_y}{\Delta P_x} * \frac{P_x}{Q_y}$$

这个公式称为需求的点交叉弹性公式。更精确地说，需求的点交叉弹性公式为：

$$E_{yx} = \frac{dQ_y}{dP_x} * \frac{P_x}{Q_y}$$

为了进一步考察某一价格区间的需求交叉弹性，我们把

$$E_{yx} = \frac{\frac{\mathrm{Q_{Y2}} - \mathrm{Q_{Y1}}}{(\mathrm{Q_{Y2}} + \mathrm{Q_{Y1}})/2}}{\frac{\mathrm{P_{x2}} - \mathrm{P_{x1}}}{(\mathrm{P_{x2}} + \mathrm{P_{x1}})/2}} = \frac{\Delta Q_Y}{\Delta P_x} * \frac{P_{x2} + P_{x1}}{Q_{Y2} + Q_{Y1}}$$

叫做**需求的弧交叉弹性公式**[3]。

其中，若交叉弹性为正，表示作物 1 和作物 2 是替代品；如果为负，则表示他们

之间是互补品。

(2)互补性：

互补性是指两种作物之间种植存在互利共赢关系，当种植作物 1 时可以有助于提升作物 2 的产量和品质。可以引入一个互补性参数来描述这种关系：

$$0 \leq \text{互补性参数}_{12} \leq 1$$

8.3.2 目标函数的求解

综合上述有关相关性分析、替代性和互补性的讨论，目标函数的大致形式和问题一问题二相似：

$$\begin{cases} \max \quad Z = price_{after} * sales_{volumn} - sales\ cost_{after} \\ \min \sum_{j=1}^{5} \left| Risk\ assessment_j \right| \end{cases}$$

现考虑相关性、替代性和互补性，调整目标函数形式。

(1)考虑作物之间的替代性

我们可以引入交叉弹性来调整种植面积，如作物 1 和作物 2 互为替代品，当作物 2 的价格上升时，可以采取减少作物 1 并增加作物 2 种植面积的方法，此时可引入一个调整系数：

$$x_{ij}^{t} = x_{ij}^{t} + \beta_{12} \times \Delta P_2$$

其中，β_{12}是一个基于交叉弹性理论的系数，代表作物 2 产生价格变化时对作物 1 种植的影响。

(2)考虑作物之间的互补性

互补性代表两种作物之间存在互利共赢的关系，当两者同时种植时，两者的产量都会增加，故我们可以在产量公式中增加一个互补性系数即可：

$$q_i^t = q_i^{2023} \times (1 + \alpha_{12})$$

其中，a_{12}表示作物 1 和作物 2 之间的互补性系数。

(3)销售量、销售价与成本之间的相关性

$$D_i^t = D_i^{2023} \times (1 + \rho PS \times P_i^t)$$

其中，ρPS是销售量和价格之间的相关性系数。

其余约束条件和问题二相同，包括有耕地总面积限制、作物种植条件限制、不重茬限制和豆类作物三年内种植一次限制。通过求解该多目标优化模型，得出 2024-2030 总收益表如下：

表格 17 2024-2030 总收益表

年份	总收益
2024	30812000
2025	30196000
2026	29413000
2027	28832000
2028	28487000
2029	27948000
2030	28383000

8.4 与问题二的比较

问题三相较于问题二多考虑了不同作物之间的替代性和互补性，以及销售价、销售量和成本之间的相关性，分别计算问题二和问题三结果的标准差和变异系数。如下表所示：

表格 18 问题二三比较分析

比较分析	问题二	问题三
总收益标准差	456.59	453.65
总收益变异系数	0.051	0.046

由上表可知，问题三在引入替代性和互补性两个目标优化函数后，其稳定性和抗干扰能力有所提升。替代性说明当某种作物受到不良影响后，可选择栽培其可替代的作物来确保总收益的稳定；互补性说明不同种作物之间具有互利共赢的关系，若在农作物栽培过程中有意建立作物之间的生物联系，可以有效的提升总收益和抗干扰能力。

九、 模型的评价、改进与推广

9.1 模型的优点

(1)文章在求解的同时绘制了清晰的流程图，直观表达逻辑关系。

(2)问题一采用多目标规划模型，综合考量全部约束，同时进行灵敏度分析，确保模型全面性。

(3)问题二的求解引入蒙特卡洛模拟，同时也采用多种思路比较优劣。

9.2 模型的缺点

(1)问题二采用的粒子群算法具有局限性，粒子群优化算法在计算中虽然收敛速度较快，但容易陷入局部最优的情况。

(2)算法时间复杂度较高，未对所有作物进行统计分析。

9.3 模型的改进

(1)尝试先进的优化算法，如遗传算法、模拟退火等，并进行比较，以提高模型的优化效率。

(2)降低算法时间复杂度，比较多算法优劣。

9.4 模型的推广

(1)引入动态调整机制，根据天气和实时市场变化调整种植计划。同时可考虑作物间的互动效应，使得模型更加贴近现实。

(2)展示模型在不同地区和作物上的应用实例，证明其效果和可靠性。

(3)探索模型在其他相关领域的应用，如林业、水产养殖等，扩大其实用领域。

十、 参考文献

[1]顾运坤.加强农业技术推广应用，促进种植业发展[J].农业开发与装备,2024,(06):232-234.

[2]邢鹏.中国种植业生产风险与政策性农业保险研究[D].南京农业大学,2004.

[3]李秀兰.需求交叉弹性对市场经济的指导[J].电子科技大学学报(社科版),2000,(04):24-25+28.DOI:10.14071/j.1008-81052000.04.008.

附录

附录 1

问题一的多目标优化算法

```
%% 导入和处理土地数据
WWL_land_filename = '附件 1-1.xlsx';
land_data_WWL = readtable(WWL_land_filename, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
land_types_WWL = {'普通大棚', '智慧大棚', '平旱地', '梯田', '山坡地', '水浇地'};
WWL_names = [];
WWL_areas = [];
WWL_types = [];
for i = 1:length(land_types_WWL)
type_data = land_types_WWL{i};
names_data = land_data_WWL.('地块名称')(strcmp(land_data_WWL.('地块类型'),
type_data));
areas_data = land_data_WWL.('地块面积')(strcmp(land_data_WWL.('地块类型'),
type_data));
WWL_names = [WWL_names; names_data];
WWL_areas = [WWL_areas; areas_data];
WWL_types = [WWL_types; repmat({type_data}, length(names_data), 1)];
end
data_info = table(WWL_names, WWL_types, WWL_areas, 'VariableNames', {'种
植地块', '地块类型', '地块面积'});
%% 导入 2023 年不同作物种植数据
data_planting_filenames = '附件 2-1 导入.xlsx';
data_planting_2023 = readtable(data_planting_filenames, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
data_planting_2023 = join(data_planting_2023, data_info, 'Keys', '种植地块');
%% 导入不同种类作物数据
crop_filenames_data = '附件 2-2 清洗后数据.xlsx';
crop_data_WWL = readtable(crop_filenames_data, 'Sheet', 1, 'ReadVariableNames',
true, 'VariableNamingRule', 'preserve');
crop_names_WWL = crop_data_WWL.('作物名称');
WWL_types = crop_data_WWL.('地块类型');
planting_seasons_data = crop_data_WWL.('种植季次');
yield_per_mu_data = crop_data_WWL.('亩产量/斤');
planting_costs_data = crop_data_WWL.('种植成本/(元/亩)');
sale_prices_data = crop_data_WWL.('销售单价/(元/斤)');
merged_data = innerjoin(data_planting_2023, crop_data_WWL, 'Keys', {'作物名称
', '种植季次', '地块类型'});
%% 计算总产量和收益
yield_per_mu_data = merged_data.('亩产量/斤');
planting_areas_data = merged_data.('种植面积/亩');
```

```
sale_prices_data = merged_data.('销售单价/(元/斤)');
total_productions = yield_per_mu_data .* planting_areas_data;
expected_sales_WWL = 0.8 * total_productions;
revenue_case1 = zeros(size(total_productions));
revenue_case2 = zeros(size(total_productions));
for i = 1:length(total_productions)
production = total_productions(i);
expected_sale = expected_sales_WWL(i);
price = sale_prices_data(i);
% 情况 1: 滞销部分浪费
if production <= expected_sale
revenue_case1(i) = production * price;
else
revenue_case1(i) = expected_sale * price;
end
% 情况 2: 超出部分按 50%降价出售
if production <= expected_sale
revenue_case2(i) = production * price;
else
excess_production = production - expected_sale;
revenue_case2(i) = expected_sale * price + excess_production * price * 0.5;
end
end
disp('情况 1: 滞销部分浪费的总收益:');
disp(revenue_case1);
disp('情况 2: 超过部分降价出售的总收益:');
disp(revenue_case2);
crop_names_WWL = merged_data.('作物名称');
revenue_case1 = revenue_case1(:);
revenue_case2 = revenue_case2(:);
% 绘制柱状图进行对比
figure;
bar_width = 0.4;
b1 = bar(1:length(crop_names_WWL), revenue_case1, bar_width, 'FaceColor', [0.3,
0.6, 0.8]);
hold on;
b2 = bar(1:length(crop_names_WWL) + bar_width, revenue_case2, bar_width,
'FaceColor', [0.9, 0.5, 0.5]);
xticks(1:length(crop_names_WWL) + bar_width/2);
xticklabels(crop_names_WWL);
xtickangle(45);
legend([b1, b2], {'情况 1: 滞销浪费', '情况 2: 超出部分降价'}, 'Location',
'NorthWest');
title('2023 年两种情况下的收益对比', 'FontSize', 14, 'FontWeight', 'bold');
```

```
xlabel('作物名称', 'FontSize', 12);
ylabel('总收益（元）', 'FontSize', 12);
grid on;
set(gca, 'GridAlpha', 0.3);
set(gca, 'FontSize', 12);
set(gca, 'box', 'off');
hold off;
set(gcf, 'Position', [100, 100, 1200, 600]);
```

附录 2

问题二的贪心算法

```
%% 导入并且处理土地数据
WWL_lands_names = [];
WWL_lands_areas_data = [];
WWL_lands_types = [];
WWL_self_land_filenames = '附件 1-sheet1.xlsx';
WWL_self_land_database = readtable(WWL_self_land_filenames, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
WWL_choose_landtype = {'普通大棚', '智慧大棚', '平旱地', '梯田', '山坡地', '水浇
地'};
for i = 1:length(WWL_choose_landtype)
WWL_type = WWL_choose_landtype{i};
WWL_names = WWL_self_land_database.('地块名称
')(strcmp(WWL_self_land_database.('地块类型'), WWL_type));
WWL_areas = WWL_self_land_database.('地块面积
')(strcmp(WWL_self_land_database.('地块类型'), WWL_type));
WWL_lands_names = [WWL_lands_names; WWL_names];
WWL_lands_areas_data = [WWL_lands_areas_data; WWL_areas];
WWL_lands_types = [WWL_lands_types; repmat({WWL_type},
length(WWL_names), 1)];
end
WWL_plot_info = table(WWL_lands_names, WWL_lands_types,
WWL_lands_areas_data, 'VariableNames', {'种植地块', '地块类型', '地块面积'});
%% 导入 2023 年种植数据
planting_filenames_WWL = '附件 2-sheet1.xlsx';
WWL_planting_database_2023 = readtable(planting_filenames_WWL, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
WWL_planting_database_2023 = join(WWL_planting_database_2023,
WWL_plot_info, 'Keys', '种植地块');
%% 导入不同作物数据
crop_filenames_WWL = '附件 2-sheet2.xlsx';
WWL_crop_database = readtable(crop_filenames_WWL, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
```

```
WWL_crop_names = WWL_crop_database.('作物名称');
WWL_lands_types = WWL_crop_database.('地块类型');
WWL_planting_seasons = WWL_crop_database.('种植季次');
WWL_yield_per_mu = WWL_crop_database.('亩产量/斤');
WWL_planting_costs = WWL_crop_database.('种植成本/(元/亩)');
WWL_sale_prices = WWL_crop_database.('销售单价/(元/斤)');
%% 导入和处理作物和土地适用数据
WWL_filenames2 = '附件 1-sheet2.xlsx';
WWL_crop_land_data = readtable(WWL_filenames2, 'Sheet', 1,
'ReadVariableNames', true, 'VariableNamingRule', 'preserve');
WWL_crop_ids = WWL_crop_land_data.('作物编号');
WWL_crop_names = WWL_crop_land_data.('作物名称');
WWL_crop_types = WWL_crop_land_data.('作物类型');
WWL_crop_suitable_lands = WWL_crop_land_data.('种植耕地');
WWL_crop_names_all = {};
WWL_crop_types_all = {};
WWL_land_types_all = {};
WWL_seasons_all = {};
WWL_crop_ids_all = [];
for i = 1:height(WWL_crop_land_data)
WWL_suitable_lands_data = WWL_crop_suitable_lands{i};
WWL_suitable_lands_data = strrep(WWL_suitable_lands_data, '↵', '');
if isempty(WWL_suitable_lands_data)
continue;
end
tokens_WWL = regexp(WWL_suitable_lands_data,
'(?<land_type>\S+)\s(?<season>\S+)', 'names');
for j = 1:length(tokens_WWL)
WWL_land_type = tokens_WWL(j).land_type;
WWL_season = tokens_WWL(j).season;
WWL_seasons = strsplit(WWL_season, ' ');
for k = 1:length(WWL_seasons)
WWL_crop_ids_all(end+1, 1) = WWL_crop_ids(i);
WWL_crop_names_all{end+1, 1} = WWL_crop_names{i};
WWL_crop_types_all{end+1, 1} = WWL_crop_types{i};
WWL_land_types_all{end+1, 1} = WWL_land_type;
WWL_seasons_all{end+1, 1} = strtrim(WWL_seasons{k});
end
end
end
WWL_result_table = table(WWL_crop_ids_all, WWL_crop_names_all,
WWL_crop_types_all, WWL_land_types_all, WWL_seasons_all, ...
'VariableNames', {'作物编号', '作物名称', '作物类型', '地块类型', '季节'});
writetable(WWL_result_table, '分解后的作物地块和季节信息.xlsx');
```

```
    %% 各类参数的更新
    WWL_initial_yield_data = WWL_crop_database.('亩产量/斤');
    WWL_lands_areas_data = WWL_planting_database_2023.('地块面积');
    WWL_planted_crops = WWL_planting_database_2023.('作物名称');
    yield_per_plot = zeros(height(WWL_planting_database_2023), 1);
    for i = 1:height(WWL_planting_database_2023)
    WWL_crop_name = WWL_planted_crops{i};
    WWL_crop_index = find(strcmp(WWL_crop_database.('作物名称'),
WWL_crop_name));
    if ~isempty(WWL_crop_index)
    yield_per_plot(i) = WWL_crop_database.('亩产量/斤')(WWL_crop_index(1));
    else
    yield_per_plot(i) = 0;
    end
    end
    WWL_lands_areas_data = WWL_planting_database_2023.('地块面积');
    WWL_total_production_2023_data = yield_per_plot .* WWL_lands_areas_data;
    WWL_initial_sales_volume = WWL_total_production_2023_data * 0.8;
    WWL_initial_sales_volume = WWL_total_production_2023_data * 0.8;
    WWL_num_crops_data = height(WWL_crop_database);
    years = 2024:2030;
    data_number_years_WWL = length(years);
    grain_crops = {'小麦', '玉米'};
    vegetable_crops = {'西红柿', '黄瓜'};
    shiyongjun_crops = {'羊肚菌', '香菇'};
    grain_sales_growth_rate = @(year) (1 + rand*0.05 + 0.05); % 粮食类 5%~10%增
长
    other_sales_change = @(year) (1 + rand*0.05 - 0.025); % 其他作物±5%变化
    yield_change = @(year) (1 + rand*0.1 - 0.05); % 亩产量±10%
    cost_growth_rate = 1.05; % 成本每年增长 5%
    vegetable_price_growth_rate = 1.05; % 蔬菜类价格增长 5%
    shiyongjun_price_decline_rate = @(crop) (0.95 * strcmp(crop, "羊肚菌") + (0.99 -
rand*0.04) * ~strcmp(crop, "羊肚菌"));
    sales_volume_data = zeros(WWL_num_crops_data, data_number_years_WWL);
    WWL_yield_data = zeros(WWL_num_crops_data, data_number_years_WWL);
    WWL_cost_data = zeros(WWL_num_crops_data, data_number_years_WWL);
    WWL_price_data = zeros(WWL_num_crops_data, data_number_years_WWL);
    WWL_initial_sales_volume = zeros(WWL_num_crops_data, 1);
    for WWL_crop_index = 1:WWL_num_crops_data
    WWL_crop_name = WWL_crop_database.('作物名称'){WWL_crop_index};
    WWL_yield = WWL_crop_database.('亩产量/斤')(WWL_crop_index);
    WWL_total_area =
sum(WWL_lands_areas_data(strcmp(WWL_planting_database_2023.('作物名称'),
WWL_crop_name)));
```

```
        WWL_total_production = WWL_yield * WWL_total_area;
        WWL_initial_sales_volume(WWL_crop_index) = WWL_total_production * 0.8;
        end
        for WWL_year_index = 1:data_number_years_WWL
        for WWL_crop_index = 1:WWL_num_crops_data
        WWL_crop_name = WWL_crop_database.('作物名称'){WWL_crop_index};
        if ismember(WWL_crop_name, grain_crops)
        sales_volume_data(WWL_crop_index, WWL_year_index) =
WWL_initial_sales_volume(WWL_crop_index) *
grain_sales_growth_rate(years(WWL_year_index));
        else
        sales_volume_data(WWL_crop_index, WWL_year_index) =
WWL_initial_sales_volume(WWL_crop_index) *
other_sales_change(years(WWL_year_index));
        end
        % 更新亩产量，每年亩产量有 ±10% 的波动
        WWL_yield_data(WWL_crop_index, WWL_year_index) =
WWL_initial_yield_data(WWL_crop_index) * yield_change(years(WWL_year_index));
        % 更新种植成本，每年种植成本增加 5%
        WWL_cost_data(WWL_crop_index, WWL_year_index) =
WWL_planting_costs(WWL_crop_index) * cost_growth_rate^WWL_year_index;
        % 更新销售价格
        if ismember(WWL_crop_name, vegetable_crops)
        % 蔬菜类作物价格以 5% 的年增长率增长
        WWL_price_data(WWL_crop_index, WWL_year_index) =
WWL_sale_prices(WWL_crop_index) *
vegetable_price_growth_rate^WWL_year_index;
        elseif ismember(WWL_crop_name, shiyongjun_crops)
        % 食用菌类作物价格以 1%-5% 的下降率下降，羊肚菌下降 5%
        WWL_price_data(WWL_crop_index, WWL_year_index) =
WWL_sale_prices(WWL_crop_index) *
shiyongjun_price_decline_rate(WWL_crop_name)^WWL_year_index;
        else
        % 粮食类作物价格基本稳定
        WWL_price_data(WWL_crop_index, WWL_year_index) =
WWL_sale_prices(WWL_crop_index);
        end
        end
        end
        data_num_plots_WWL = height(WWL_plot_info);
        WWL_optimal_plan_first_season = zeros(data_num_plots_WWL,
WWL_num_crops_data, data_number_years_WWL);
        WWL_optimal_plan_second_season = zeros(data_num_plots_WWL,
WWL_num_crops_data, data_number_years_WWL);
```

```
    WWL_total_revenue_per_year = zeros(data_number_years_WWL, 1);
    for WWL_year_index = 1:data_number_years_WWL
    WWL_revenue_data = zeros(data_num_plots_WWL, WWL_num_crops_data);
    for WWL_plot_index = 1:data_num_plots_WWL
    WWL_plot_area = WWL_lands_areas_data(WWL_plot_index);
    for WWL_crop_index = 1:WWL_num_crops_data
    actual_yield_data_WWL = WWL_yield_data(WWL_crop_index,
WWL_year_index);
    expected_sales_data_WWL = sales_volume_data(WWL_crop_index,
WWL_year_index);
    price_database = WWL_price_data(WWL_crop_index, WWL_year_index);
    cost_data_WWL = WWL_cost_data(WWL_crop_index, WWL_year_index);
    WWL_total_production = actual_yield_data_WWL * WWL_plot_area;
    revenue_WWL = min(WWL_total_production, expected_sales_data_WWL) *
price_database - cost_data_WWL * WWL_plot_area;
    WWL_revenue_data(WWL_plot_index, WWL_crop_index) = revenue_WWL;
    end
    [max_revenue_first, best_crop_idx_first] =
max(WWL_revenue_data(WWL_plot_index, :));
    WWL_optimal_plan_first_season(WWL_plot_index, best_crop_idx_first,
WWL_year_index) = WWL_plot_area;
    [max_revenue_second, best_crop_idx_second] =
max(WWL_revenue_data(WWL_plot_index, :));
    WWL_optimal_plan_second_season(WWL_plot_index, best_crop_idx_second,
WWL_year_index) = WWL_plot_area;
    WWL_total_revenue_per_year(WWL_year_index) =
WWL_total_revenue_per_year(WWL_year_index) + max_revenue_first +
max_revenue_second;
    end
    end
    data_number_years_WWL = length(years);
    data_num_plots_WWL = height(WWL_plot_info);
    WWL_num_crops_data = height(WWL_crop_database);
    yearly_plans_WWL = struct();
    expected_sales_factor_WWL = 0.8;
    min_plot_area_WWL = 0.1;
    last_crop_planted_WWL_data = cell(data_num_plots_WWL,
data_number_years_WWL);
    for WWL_year_index = 1:data_number_years_WWL
    year_plan_first_season_data_WWL = zeros(data_num_plots_WWL,
WWL_num_crops_data);
    year_plan_second_season_data_WWL = zeros(data_num_plots_WWL,
WWL_num_crops_data);
    for WWL_plot_index = 1:data_num_plots_WWL
```

```
    data_plot_name_WWL = WWL_plot_info.('种植地块'){WWL_plot_index};
    WWL_plot_area = WWL_plot_info.('地块面积')(WWL_plot_index);
    applicable_crops_data = WWL_result_table(strcmp(WWL_result_table.('地块类型
'), WWL_plot_info.('地块类型'){WWL_plot_index}), :);
    if WWL_year_index > 1
    last_crop_data = last_crop_planted_WWL_data{WWL_plot_index,
WWL_year_index - 1};
    applicable_crops_data = applicable_crops_data(~strcmp(applicable_crops_data.('作
物名称'), last_crop_data), :);
    end
    season_1_crops_data = applicable_crops_data(strcmp(applicable_crops_data.('季节
'), '第一季'), :);
    if ~isempty(season_1_crops_data)
    for WWL_crop_index = 1:height(season_1_crops_data)
    [best_crop_idx, ~] = find_best_crop(season_1_crops_data(WWL_crop_index, :),
WWL_crop_database, WWL_plot_area, expected_sales_factor_WWL,
min_plot_area_WWL);
    year_plan_first_season_data_WWL(WWL_plot_index, best_crop_idx) =
WWL_plot_area;
    last_crop_planted_WWL_data{WWL_plot_index, WWL_year_index} =
WWL_crop_database.('作物名称'){best_crop_idx};
    end
    end
    season_2_crops_WWL = applicable_crops_data(strcmp(applicable_crops_data.('季
节'), '第二季'), :);
    if ~isempty(season_2_crops_WWL)
    for WWL_crop_index = 1:height(season_2_crops_WWL)
    [best_crop_idx, ~] = find_best_crop(season_2_crops_WWL(WWL_crop_index, :),
WWL_crop_database, WWL_plot_area, expected_sales_factor_WWL,
min_plot_area_WWL);
    year_plan_second_season_data_WWL(WWL_plot_index, best_crop_idx) =
WWL_plot_area;
    last_crop_planted_WWL_data{WWL_plot_index, WWL_year_index} =
WWL_crop_database.('作物名称'){best_crop_idx};
    end
    end
    end
    yearly_plans_WWL(WWL_year_index).first_season =
year_plan_first_season_data_WWL;
    yearly_plans_WWL(WWL_year_index).second_season =
year_plan_second_season_data_WWL;
    end
    %% 各类参数的初始化(豆类的约束条件)
    expected_sales_factor_WWL = 0.8;
```

```
    min_plot_area_WWL = 0.1;
    legume_crops_WWL = ["黄豆", "豇豆", "芸豆", "红豆", "黑豆", "绿豆", "爬豆", "刀豆"];
    last_crop_planted_WWL_data = cell(data_num_plots_WWL, data_number_years_WWL);
    last_legume_year = zeros(data_num_plots_WWL, 1);
    for WWL_year_index = 1:data_number_years_WWL
    year_plan_first_season_data_WWL = zeros(data_num_plots_WWL, WWL_num_crops_data);
    year_plan_second_season_data_WWL = zeros(data_num_plots_WWL, WWL_num_crops_data);
    for WWL_plot_index = 1:data_num_plots_WWL
    data_plot_name_WWL = WWL_lands_names{WWL_plot_index};
    WWL_plot_area = WWL_lands_areas_data(WWL_plot_index);
    applicable_crops_data = WWL_result_table(strcmp(WWL_result_table.('地块类型'), WWL_plot_info.('地块类型'){WWL_plot_index}), :);
    if WWL_year_index > 1
    last_crop_data = last_crop_planted_WWL_data{WWL_plot_index, WWL_year_index - 1};
    applicable_crops_data = applicable_crops_data(~strcmp(applicable_crops_data.('作物名称'), last_crop_data), :);
    end
    if (WWL_year_index - last_legume_year(WWL_plot_index)) >= 3
    WWL_crop_names = string(applicable_crops_data.('作物名称'));
    legume_crops_applicable = applicable_crops_data(ismember(WWL_crop_names, legume_crops_WWL), :);
    if ~isempty(legume_crops_applicable)
    applicable_crops_data = legume_crops_applicable;
    end
    end
    season_1_crops_data = applicable_crops_data(strcmp(applicable_crops_data.('季节'), '第一季'), :);
    total_planted_first_season = 0;
    if ~isempty(season_1_crops_data)
    for WWL_crop_index = 1:height(season_1_crops_data)
    [best_crop_idx, best_revenue] = find_best_crop(season_1_crops_data(WWL_crop_index, :), WWL_crop_database, WWL_plot_area, expected_sales_factor_WWL, min_plot_area_WWL);
    if all(best_crop_idx > 0) && all(total_planted_first_season < WWL_plot_area)
    planting_area = min(WWL_plot_area - total_planted_first_season, WWL_plot_area);
    year_plan_first_season_data_WWL(WWL_plot_index, best_crop_idx) = planting_area;
    total_planted_first_season = total_planted_first_season + planting_area;
```

```
        last_crop_planted_WWL_data{WWL_plot_index, WWL_year_index} =
WWL_crop_database.('作物名称'){best_crop_idx};
        if ismember(WWL_crop_database.('作物名称'){best_crop_idx},
legume_crops_WWL)
        last_legume_year(WWL_plot_index) = WWL_year_index;
        end
        end
        end
        end
        if total_planted_first_season == 0 && ~isempty(season_1_crops_data)
        random_crop_idx = randi(height(season_1_crops_data));
        year_plan_first_season_data_WWL(WWL_plot_index, random_crop_idx) =
WWL_plot_area;
        end
        season_2_crops_WWL = applicable_crops_data(strcmp(applicable_crops_data.('季
节'), '第二季'), :);
        total_planted_area_second_season = 0;
        if ~isempty(season_2_crops_WWL)
        for WWL_crop_index = 1:height(season_2_crops_WWL)
        [best_crop_idx, best_revenue] =
find_best_crop(season_2_crops_WWL(WWL_crop_index, :), WWL_crop_database,
WWL_plot_area, expected_sales_factor_WWL, min_plot_area_WWL);
        if all(best_crop_idx > 0) && all(total_planted_area_second_season <
WWL_plot_area)
        planting_area = min(WWL_plot_area - total_planted_area_second_season,
WWL_plot_area);
        year_plan_second_season_data_WWL(WWL_plot_index, best_crop_idx) =
planting_area;
        total_planted_area_second_season = total_planted_area_second_season +
planting_area;
        last_crop_planted_WWL_data{WWL_plot_index, WWL_year_index} =
WWL_crop_database.('作物名称'){best_crop_idx};
        if ismember(WWL_crop_database.('作物名称'){best_crop_idx},
legume_crops_WWL)
        last_legume_year(WWL_plot_index) = WWL_year_index;
        end
        end
        end
        end
        if total_planted_area_second_season == 0 && ~isempty(season_2_crops_WWL)
        random_crop_idx = randi(height(season_2_crops_WWL));
        year_plan_second_season_data_WWL(WWL_plot_index, random_crop_idx) =
WWL_plot_area;
        end
```

```
    end
    yearly_plans_WWL{WWL_year_index} = struct('year',
years(WWL_year_index), ...
    'first_season', year_plan_first_season_data_WWL, ...
    'second_season', year_plan_second_season_data_WWL);
    end
```

附录 3

问题二的粒子群算法

```
import numpy as np
import os
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
# 粒子群算法的参数设置
w = 0.5
c1 = 1.5
c2 = 1.5
number_particles = 30
max_data = 1000
WWL_number_years = 7
WWL_number_crops = 41
WWL_number_plots = 34
WWL_number_seasons = 2

file_path_data = '附件 2.xlsx'
WWL_data_stats = pd.read_excel(file_path, sheet_name='2023 年统计的相关数据')
WWL_data_stats['销售单价/(元/斤)'] = data_stats['销售单价/(元/斤)'].astype(str)
p = WWL_data_stats['销售单价/(元/斤)'].apply(lambda x: (float(x.split('-')[0]) +
float(x.split('-')[1])) / 2 if '-' in x else float(x)).values
c = WWL_data_stats['种植成本/(元/亩)'].values
q = (WWL_data_stats['亩产量/斤'].values / 2).astype(float)  # 1 斤 = 0.5 千克
data_crop_situation_WWL = pd.read_excel(file_path, sheet_name='2023 年的农作物种
植情况')
D = (data_crop_situation_WWL['种植面积/亩'].values * q[data_crop_situation_WWL['
作物编号'].values - 1])
particles = np.random.rand(num_particles, WWL_number_crops, WWL_number_plots,
WWL_number_seasons, WWL_number_years)
velocities_WWL=np.random.rand(num_particles,WWL_number_crops,
WWL_number_plots, WWL_number_seasons, WWL_number_years)
p_WWL = np.copy(particles)
g_WWL = np.copy(particles[0])
best_time_WWL = []
def ensure_tenth_multiples(x):
    return np.round(x * 10) / 10
# 目标函数 1
```

```
def objective_function1(x):
    Z1 = 0
    for t in range(WWL_number_years):
        for k in range(WWL_number_seasons):
            for i in range(WWL_number_crops):
                y_ikt = np.sum(x[i, :, k, t]) * q[i]
                Z1 += p[i] * min(y_ikt, D[i]) - c[i] * np.sum(x[i, :, k, t])
    return Z1
# 目标函数 2
def objective_function2(x):
    Z2 = 0
    for t in range(WWL_number_years):
        for k in range(WWL_number_seasons):
            for i in range(WWL_number_crops):
                y_ikt = np.sum(x[i, :, k, t]) * q[i]
                Z2 += p[i] * min(y_ikt, D[i]) + (0.5 * p[i]) * max(y_ikt - D[i], 0) - c[i]
* np.sum(x[i, :, k, t])
    return Z2
# 主函数
def pso():
    data_global_best, data_best, data_best_time
    for iter in range(data_max):
        if iter % 100 == 0 and iter>1:
            plt.figure(figsize=(10, 6))
            plt.plot(range(data_max),data_best_time,label='Best Data over Iterations')
            plt.xlabel('Iterations')
            plt.ylabel('Best Data')
            plt.title('PSO')
            plt.legend()
            plt.grid()
            plt.savefig(f"{folder_name}/Convergence_PSO.png")
            plt.show()
        for n in range(number_particles):
            fitness = objective_function1(particles[n])
            p_fitness = objective_function1(p_best[n])
            g_fitness = objective_function1(g_best)
            if fitness > p_fitness:
                p_best[n] = particles[n]
            if fitness > g_fitness:
                g_best = particles[n]
            velocities[n] = w * velocities[n] + c1 * np.random.rand() * (p_best[n] -
particles[n]) + c2 * np.random.rand() * (g_best - particles[n])
            particles[n] += velocities[n]
            particles[n] = np.maximum(particles[n], 0)
            particles[n] = ensure_tenth_multiples(particles[n])
        data_best_time.append(g_fitness)
        print("Iteration " + str(iter) + str(-g_fitness))
    return g_best
best_solution = pso()
best_value_data = objective_function1(best_solution)
```

```
# 绘制图表
plt.figure(figsize=(10, 6))
plt.plot(range(max_data), data_best_time, label='Best Data over Iterations')
plt.xlabel('Iterations')
plt.ylabel('Best Data')
plt.title('PSO')
plt.legend()
plt.grid()
plt.savefig(f"{folder_name}/_PSO.png")
plt.show()
```

附录 4

问题三的相关性分析代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

file_data = '附件 2-sheet2.xlsx'
Df_data = pd.read_excel(file_data)
columns_data = ['亩产量/斤', '种植成本/(元/亩)', '销售单价/(元/斤)']
WWL_selected = Df_data[columns_data]
WWL_selected = WWL_selected.dropna()
correlation_matrix = WWL_selected.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=1)
plt.title('亩产量，种植成本，销售单价')
plt.show()
```

附录 5

问题三的时间序列算法

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

data_WWL = pd.read_excel('附件 1.xslx')
```

```
X_data = data_WWL[['销售价格', '种植成本']].values
Y_data = data_WWL['预期销售量'].values
scaler_X_data = StandardScaler()
scaler_Y_data = StandardScaler()
X_data = scaler_X_data.fit_transform(X_data)
Y_data = scaler_Y_data.fit_transform(Y_data.reshape(-1, 1)).flatten()
X_data_train,X_data_test,Y_data_train,Y_data_test=train_test_split(X_data,Y_data,test_
size=0.2, random_state=42)

X_data_train_tensor = torch.tensor(X_data_train, dtype=torch.float32)
Y_data_train_tensor = torch.tensor(Y_data_train, dtype=torch.float32).view(-1, 1)
X_data_test_tensor = torch.tensor(X_data_test, dtype=torch.float32)
Y_data_test_tensor = torch.tensor(Y_data_test, dtype=torch.float32).view(-1, 1)
train_dataset = TensorDataset(X_data_train_tensor, Y_data_train_tensor)
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.hidden1 = nn.Linear(2, 64)
        self.hidden2 = nn.Linear(64, 32)
        self.output = nn.Linear(32, 1)
    def forward(self, x):
        x = torch.relu(self.hidden1(x))
        x = torch.relu(self.hidden2(x))
        x = self.output(x)
        return x
model = MLP()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
epochs = 500
for epoch in range(epochs):
    model.train()
    for X_data_batch, Y_data_batch in train_loader:
        optimizer.zero_grad()
        Y_data_pred = model(X_data_batch)
        loss = criterion(Y_data_pred, Y_data_batch)
        loss.backward()
        optimizer.step()
model.eval()
with torch.no_grad():
    Y_data_test_pred = model(X_data_test_tensor).numpy()
Y_data_test_pred = scaler_Y_data.inverse_transform(Y_data_test_pred)
Y_data_test_actual = scaler_Y_data.inverse_transform(Y_data_test_tensor.numpy())
# 评估模型的好坏
mse = mean_squared_error(Y_data_test_actual, Y_data_test_pred)
r2 = r2_score(Y_data_test_actual, Y_data_test_pred)
# 绘制散点图
plt.figure(figsize=(10, 6))
plt.scatter(Y_data_test_actual, Y_data_test_pred, color='red', label='预测值 vs 实际值')
```

```
plt.plot([Y_data_test_actual.min(),Y_data_test_actual.max()],[Y_data_test_actual.min(),
Y_data_test_actual.max()], color='green', linestyle='--', label='拟合线')
plt.xlabel('实际销售量')
plt.ylabel('预测销售量')
plt.title('实际值与预测值对比')
plt.legend()
plt.grid(True)
plt.show()
```

附录 6

问题三与问题二的对比

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df1 = pd.read_excel('结果 Q2.xlsx', sheet_name='年度总收益')
df2 = pd.read_excel('结果 Q3.xlsx', sheet_name='年度总收益')
std1 = df1['总收益'].std()
std2 = df2['总收益'].std()
cv1 = std1 / df1['总收益'].mean()
cv2 = std2 / df2['总收益'].mean()
```