

基于贪心算法的农作物种植策略优化系统

摘要

本文根据 2023 年的农作物种植数据，在庞杂的约束条件限制下，综合考虑模型复杂度、模型运行速度以及模型的工程化与泛化能力，并结合数据本身蕴藏的序关系构建了基于贪心算法的农作物种植策略优化系统，并在求解过程中对农作物种植的相关问题进行探究。

针对问题一，我们首先对数据进行预处理，根据普通大棚第一季的可种植的蔬菜作物及其亩产量、种植成本和销售价格，补全智慧大棚第一季的相关数据，并实现数据的矩阵化。之后我们计算每一种作物的亩利润和降价后亩利润。为了增强对数据的直观理解我们通过绘制亩利润热力图来得到亩利润的分布，并通过绘制 2023 种植策略图来显示初始农作物的分布。在建模时，考虑到线性规划模型的复杂性及与后续问题的关联，我们采用速度更快、拓展能力更强的贪心算法模型。在（1）的条件下，我们的种植策略能够在 2024-2030 年获得总利润 37616883 元，在（2）的条件下，我们的种植策略能够在 2024-2030 年获得总利润 93599691 元，但年利润不稳定，波动性极大，并且种植策略出现了一些较为极端的情况。

针对问题二，我们拆解重构了问题一中代码的结构，以更好的适配作物各数据如预期销售量、亩产量、销售价格等随年份发生的变化。同时根据题目中的要求，采用随机数生成的方法在每两年更替时对相应的数据进行变化。此外，问题二中并没有给出对于超产作物的处理，因此我们结合实际生活，引入最大销售量的概念，在产量超过预期销售量但不超过最大销售量的作物按 50% 价格售出，超过最大销售量则滞销不再售出。经过这些处理我们求得 2024-2030 年总利润为 52249448 元，并成功解决了 1（2）中年利润波动大的问题，并且求解出的策略呈现出种植作物的多元化，避免了市场饱和现象的出现。

针对问题三，我们需要解决作物的可替代性、作物的互补性以及预期销售量与销售价格、种植成本之间的相关性这三个因素对模型求解的影响。在研究预期销售量与销售价格、种植成本之间的相关性时，我们认为可以引入价格弹性的概念进行预期销售量的处理，同时结合不同的作物具有不同程度的可替代性，我们认为每种作物的弹性应不同。在研究作物的互补性上，我们查阅到豆科植物与其他作物间种可显著提升其产量，于是引入了与豆科间作可以提升产量的机制。我们发现间作能够显著提升总利润，求得 2024-2030 年总利润为 64584904 元，相比 2 题提升大概 23.6%。本文根据 2023 年的总作物种植数据，在庞杂的约束条件限制下，综合决策模型复杂度、模型计算速度以及模型的可维护性与可拓展性，采用基于优先队列的贪心模型进行种植策略的优化，并在求解过程中研究了农作物种植的相关问题。

关键字：最优化 贪心算法 优先队列 价格弹性 间作

一、问题重述

1.1 背景

根据乡村的实际情况，充分利用有限的耕地资源，因地制宜，发展有机种植产业，对乡村经济的可持续发展具有重要的现实意义。选择适宜的农作物，优化种植策略，有利于方便田间管理，提高生产效益，减少各种不确定因素可能造成的种植风险。现有华北山区一乡村，存在大量露天耕地与大棚耕地，可以种植多种农作物，但作物的预期销量与亩利润不尽相同，这要求我们结合实际情况调整种植策略，以期达到最大收益。因此，建立一个或多个模型来具体化种植的各种限制，调整种植策略对于科学种植、最大化收益来说非常重要。

1.2 问题重述

本文的问题重述侧重于对问题的条件的解读，不同的解读方式可能对后续建模方式产生较大的影响。

问题一：要求针对两种潜在的滞销情况，分别给出 2024~2030 年农作物的最优种植方案。问题假定了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 年保持稳定以及每季种植的农作物在当季销售，因此我们需要将 2023 年的数据作为前提条件。同时，问题指出产量超出预期销量的部分不能正常销售，并给出了不能正常销售的两种可能情况：(1) 超过部分滞销，造成浪费；(2) 超出部分按 2023 年销售价格的 50% 降价出售。我们需要对两种情况，调整模型中价格方面的细节，得到不同的种植策略。

问题二：该问题对预期销量、亩产量、种植成本以及销售价格等因素进行了更细致的要求，期望我们在考虑以上的波动因素，给出新的 2024~2030 年间的种植策略。这要求我们首先对每年的预期销量、种植成本、亩产量等指标进行预测，同时采用修正的数据，得到更为精细的模型与合理的种植策略。

问题三：第三问是更为真实与全面的考虑，题目指出各种农作物之间也存在可替代性与互补性，且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。所以我们要对农作物间的相互作用，销量价格和预期销售量间的相互作用给出具体的刻画，在此基础上加入新的约束条件，延续问题二的建模，给出该乡村 2024~2030 年农作物的最优种植策略，并与问题二的结果进行比较。

二、问题分析

2.1 整体分析

该题目围绕 2023~2030 年间最佳农作物种植策略这一中心，共设置了三道问题，三道问题之间互不相同但又紧密相关，可以看作是同一个问题不断深入细化的过程。因此我们可以考虑建立一个统一的模型，并结合问题一、二和三各自的特点，对模型加以约束和优化，从而解决不同情形下的问题。

整体而言，问题求解的目标均为给出未来的最优种植策略，而“最优”二字，则指的是收益最高，即种植农作物获得利润最大。因此，该题可以建立最优化模型，目标函数即为种植获利，变量则是种植的方案，即在某块地上种植某种作物多少亩。同时，我们发现，如果先不考虑豆科植物每三年必须种植这一因素，每一年的种植策略实际上只与前一年的情况有关。因此我们可以把 2023 年作为初始状态，逐年推出下一年的最优策略，并在此基础上，结合各个问题的具体约束，给出符合题意的最优种植策略。因此，本题的核心在于相邻年份之间的状态转移，上一年得到的策略就是下一年的初始参数。

本题实际的困难点在于约束条件的设置与表达以及最优化的求解。约束条件上，不能连续重茬种植、三年内至少种植一次豆类作物、每种作物每季的种植地不能太分散、每种作物在单个地块（含大棚）种植的面积不宜太小等条件极大增加了约束的复杂度，需要我们在对数学化约束条件时进行更为细致的刻画与分析；最优化求解上，第二问中涉及到了求解边界的变化，第三问中涉及到了更为复杂的变量间的相互影响，使得最优化过程不再是简单的线性问题，需要我们考虑更加快捷与准确的优化算法。

2.2 问题一的分析

问题一：要求针对两种超产情况，分别给出该乡村 2024~2030 年农作物的最优种植方案

本题给出了多个边界假设，即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，每季种植的农作物在当季销售。实际上这些假设给出了我们模型中的求解边界与初始参数，即 2023 年的各个数据。因此我们可以通过直接调用 2023 年的数据，对最优化模型进行初始化。另外，题目要求我们针对“(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”这两大超产情况进行分析，实际上，这两超产情况只影响了最优化模型中目标函数的表达式，我们只需要对目标函数（农作物种植的总收益）进行修正即可。

2.3 问题二的分析

问题二：要求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，给出该乡村 2024~2030 年农作物的最优种植方案

问题二指出了各种农作物的预期销售量、亩产量、种植成本和销售价格具有不确定性以及潜在种植风险，实际上是告诉我们一些模型参数在不同年间的波动。在整体分析中，我们指出每一年的种植策略基本上只与上一年的情况有关，因此，第二问对模型的影响本质上在于两年之间的状态转移。我们需要结合模型参数的波动与改变，修正两年之间的状态转移，从而得到符合题设的种植策略。同时由于不确定性的存在，我们也可以考虑方差因素对于策略选取的影响。

2.4 问题三的分析

问题三：要求在问题二的基础上，考虑作物间的可替代性与互补性、销量与价格成本的关系，给出更为真实情况下的 2024~2030 年农作物的最优种植方案。同时模拟数据求解，并与问题二进行比较分析。

为了解决问题三，一方面，我们需要对作物间的可替代性与互补性做出准确的理解与定义，可以通过查阅文献并结合实际农业生产数据进行探究；另一方面，预期销量与价格、成本三者之间的关系需要有明确的数学关系，我们期望以此对得到模拟数据，并对模型增加约束，从而回到我们的最优化模型求解。求解过后，我们还需与问题二的结果进行比较分析。考虑到作物间的可替代性与互补性，我们可以着重关注问题二三求解结果中同类型作物品种等方面差异。

我们绘制如下框架图，以展示文章所建立的模型：

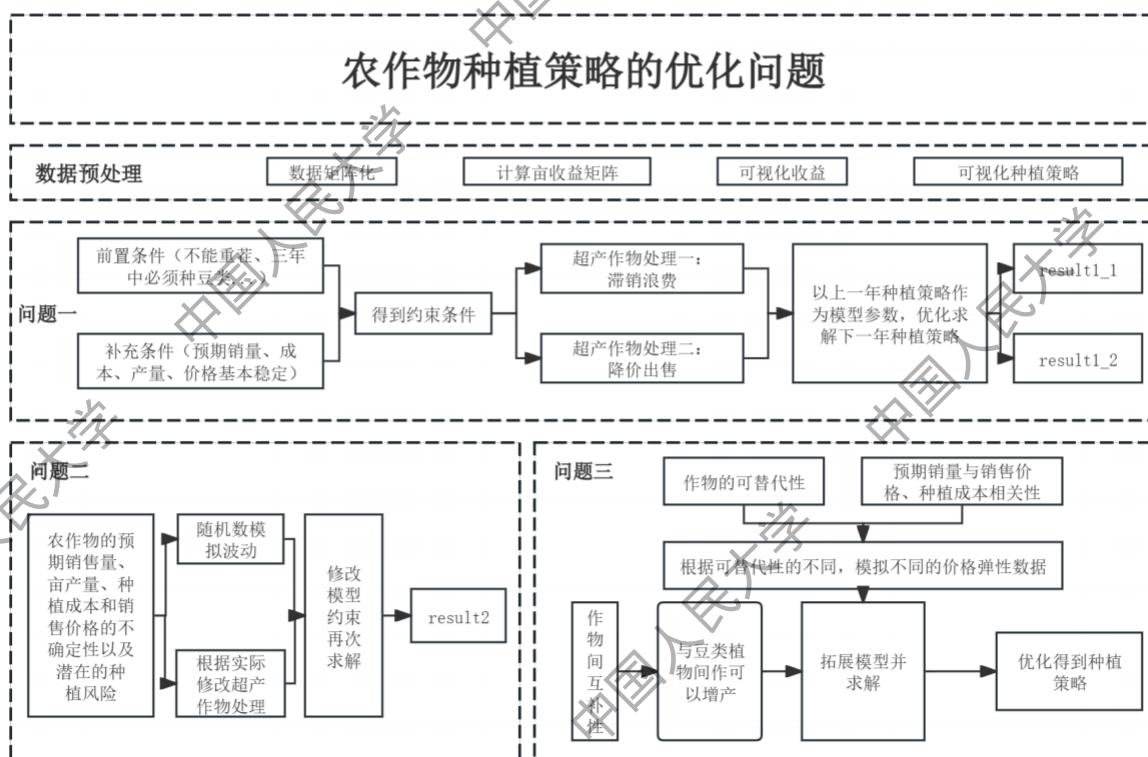


图1 全文模型框架图

三、模型假设

1. 假设同一地块种植的作物不会相互影响，即同一地块的合种组合在合法前提下是任意的。
2. 假设不同年份气候条件稳定，不会影响原本的农作物收成指标。
3. 假设市场环境稳定，不影响原本的农作物经济效益。
4. 假设所给的种植和收益数据包含了 2023 年期内的全部农作物信息，数据的记录和反应市场情况是准确和可靠的。
5. 假设农作物种植的目的是确定的，即最求经济效益的最大化。

四、符号说明

符号	说明
i	地块编码，对应地块 A_1, A_2, \dots, F_4 ，其中 $1 \leq i \leq 54$
j	农作物编号，其中 $1 \leq j \leq 41$
k	年份编码， $k = 0$ 对应 2023 年，以后依次增加
m	季编码， $m = 1, 2$ 分别对应第一季，第二季
s_i	第 i 地块的面积
$p_j^{(k,m)}$	第 k 年第 m 季时， j 作物的销售价格
$z_j^{(k)}$	第 k 年时， j 作物的预期销售量
$x_{ij}^{(k,m)}$	第 k 年第 m 季时，在 i 地块上种植 j 作物的亩数
$t_{ij}^{(k,m)}$	第 k 年第 m 季时，在 i 地块上种植 j 作物的亩产量
$c_{ij}^{(k,m)}$	第 k 年第 m 季时，在 i 地块上种植 j 作物的亩成本
$profit_{ij}^{(k,m)}$	第 k 年第 m 季时，在 i 地块上种植 j 作物的亩利润
$e_{ij}^{(k,m)}$	布尔型变量，表示第 k 年第 m 季，在 i 地块上是否能种植 j 作物
S	s_i 组成的地块面积列向量
$Z^{(k)}$	$z_j^{(k)}$ 组成的预期销售行向量
$P^{(k,m)}$	$p_j^{(k,m)}$ 组成的销售价格行向量
$X^{(k,m)}$	$x_{ij}^{(k,m)}$ 组成的种植策略矩阵
$T^{(k,m)}$	$t_{ij}^{(k,m)}$ 组成的亩产量矩阵
$C^{(k,m)}$	$c_{ij}^{(k,m)}$ 组成的亩成本矩阵
$Profit^{(k,m)}$	$profit_{ij}^{(k,m)}$ 组成的亩利润矩阵
$E^{(k,m)}$	$e_{ij}^{(k,m)}$ 组成的示性矩阵

五、模型的建立与求解

5.1 数据预处理

5.1.1 数据矩阵化

在解决农作物种植策略这一问题之前，我们首先对数据进行了预处理和标准化，以期简化复杂的数据结构，得到关键的数据信息，并提高算法的运行效率。我们将基于这个预处理过的数据集来建立针对一到三间的模型。

首先，我们对数据进行异常值与缺失值检测，发现数据保存良好，并未出现异常值，但缺失了 2023 年第一季智慧大棚可种植的蔬菜作物及其亩产量、种植成本和销售价格，我们使用普通大棚的对应数据进行补全。同时，我们观察了数据的基本结构，发现价格、销量、成本等参数都由种植地块和农作物种类共同决定，因此我们考虑将数据全部处理为矩阵形式（即矩阵 X, T, C, E ），横轴为农作物，纵轴为地块，以此来储存关于某一地块种植某一农作物的全部信息。

我们先对附件 1 的“乡村的现有耕地”表单进行处理，得到地块面积列向量，即 S ，之后我们结合附件 1 的表单“乡村种植的农作物”和附件 2 的两个表单，得到 2023 年的两季销售价格向量、2023 年的种植策略矩阵、亩产量矩阵、亩成本矩阵，即 $P^{(0,1)}$ 、 $P^{(0,2)}$ 、 $X^{(0,1)}$ 、 $X^{(0,2)}$ 、 $T^{(0,1)}$ 、 $T^{(0,2)}$ 、 $C^{(0,1)}$ 、 $C^{(0,2)}$ 。由于本题数据形式复杂但数据量大小适中，故直接使用 excel 即可完成初步的数据的矩阵化处理，我们直接在附录中给出整理好的数据并导入（见附录 A）。

值得注意的是，由于不同地块可种植的作物种类不同，且一些耕地可种两季而其它不可，我们无法直接填充上述矩阵。因此我们采取等效的方式，认为所有耕地均可种植两季，把实际上只能种一季的耕地在第二季中的亩产量均设置为零，亩成本设置为极大 (10^9)。通过这样的填充策略，我们得到了规范清晰的矩阵形式数据。

5.1.2 数据可视化分析

根据以上清洗得到的矩阵化数据，我们可以进行数据可视化分析。

5.1.2.1 耕地信息

该乡村现有露天耕地 1201 亩，分散为 34 个大小不同的地块，包括平旱地、梯田、山坡地和水浇地 4 种类型。我们结合数据，对耕地的基本信息进行分析（见附录 B）。

图 2 显示，地块 A、B、C、D、E、F 的平均面积呈递减，且大面积地块主要位于地块 A 与地块 B，即平旱地与梯田，这与乡村中优质耕地较少的实际情况相符合。

图 3 显示，总面积最大的地块是地块 B，占比 51.1%，其次是地块 A，占比 30.1%，而地块 C、D 总面积大致相当。这说明，在考虑后续种植问题时，由于所占面积巨大，地块 A、B 的种植方案会显著影响不同种植策略所带来的收益。同时由于总面积较大的

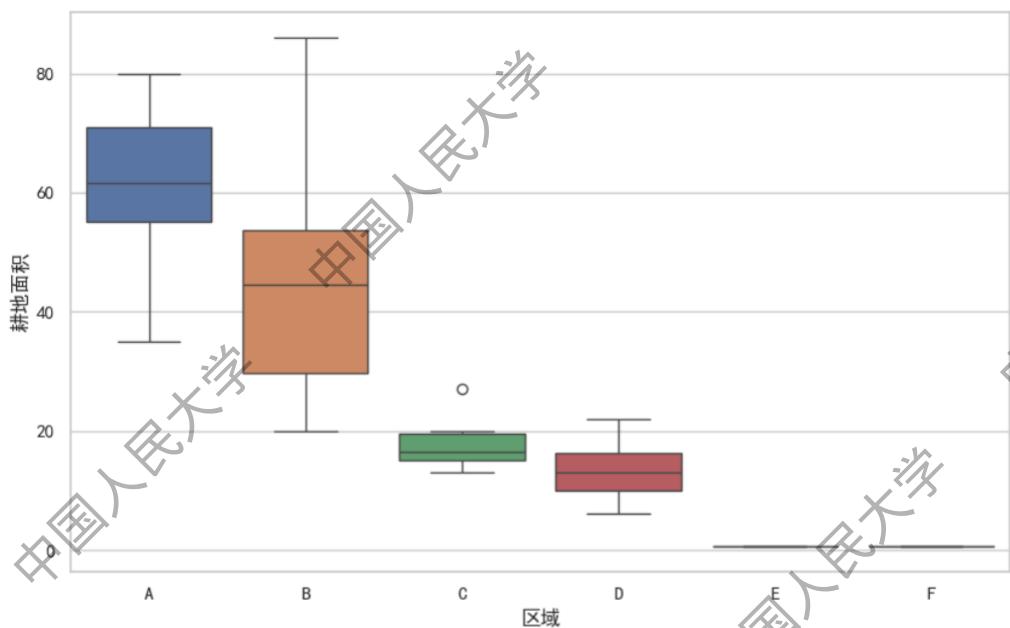


图 2 耕地面积分布

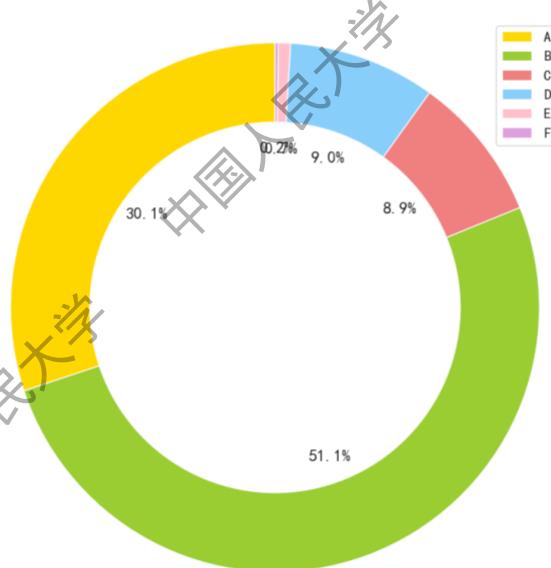


图 3 耕地面积占比

地块都是单季种植的地块，我们可以预见，第二季的收益对于整体而言可能会较小。

5.1.2.2 苗利润分析

首先，本题目关注的是农作物种植的收益问题，而收益取决于产量、销量、售价以及成本，这些因素由农作物与种植地块共同决定，因此我们有理由对农作物在不同地块的亩利润进行探究。

$$\text{亩利润} = \text{亩产量} \times \text{销售单价} - \text{亩成本}$$

即

$$profit_{ij}^{(k,m)} = t_{ij}^{(k,m)} \times p_j^{(k,m)} - c_{ij}^{(k,m)}$$

$$Profit^{(k,m)} = T^{(k,m)} \odot \begin{pmatrix} P^{(k,m)} \\ \vdots \\ P^{(k,m)} \end{pmatrix}_{54 \times 41} - C^{(k,m)} \quad (1)$$

我们得到了 $Profit^{(0,1)}$ 与 $Profit^{(0,2)}$ 并对其进行可视化 (见附录 C)

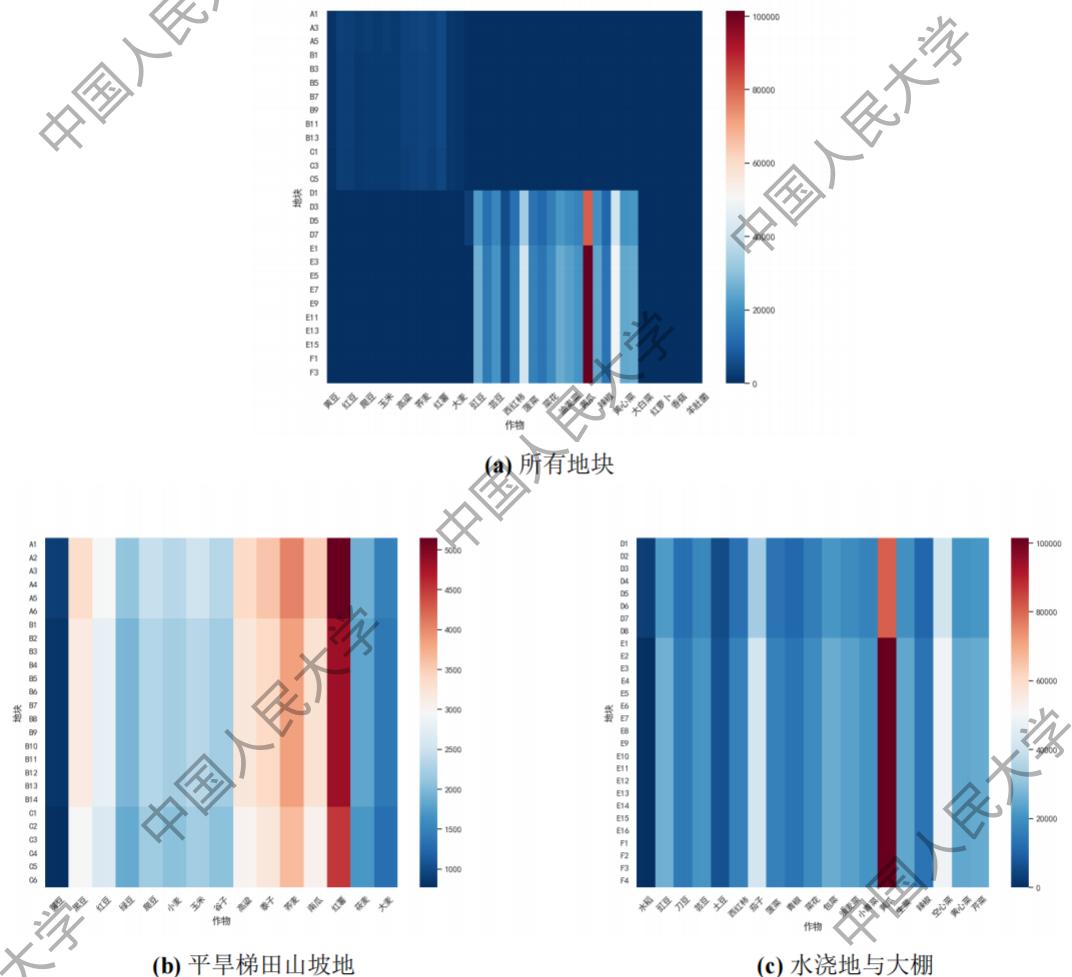


图 4 2023 年第一季亩利润热力图

通过 2023 年第一季亩利润热力图 (图 4)，我们可以看出，所有的地块在第一季都可以进行获益，但某些作物无法在第一季种植并取得利润。图像可以看出不同地块类型可种植的农作物种类具有显著的差异。同时 2023 年的第一季高亩利润区域主要在于水浇地与大棚，集中在黄瓜、茄子空心菜这些农作物上，且黄瓜的亩利润最高。而平旱地、梯田与山坡地，农作物亩利润普遍较低，这也符合现实中优质土地亩利润更高的生

活经验。这样的结果启发我们根据亩利润高低，对特别作物优先种植以期达到最优种植策略。

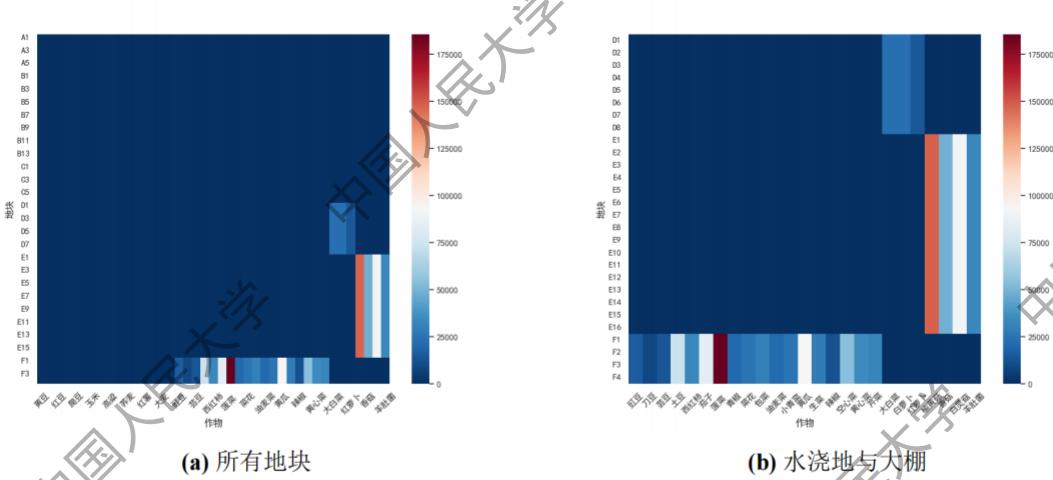


图 5 2023 年第二季亩利润热力图

通过 2023 年第二季亩利润热力图（图 5），我们可以看出，第二季中只有水浇地与大棚可以进行种植与获利，且并非所有农作物均可以在第二季种植与获利。图像中可以看出在第二季种水浇地、普通大棚、智慧大棚的目标农作物相互独立，因此在考虑第二季时，实际的情况是比较简单的。

5.1.2.3 2023 种植策略分析

除了亩利润，在每地块的种植策略也会显著影响最终的收益，且由于不能重茬种植等因素，2023 年的种植策略对后续几年的种植方案有着一定的影响，因此我们有必要对 2023 年的种植策略（即 $X^{(0,1)}$, $X^{(0,2)}$ ）进行可视化（见附录 D）。

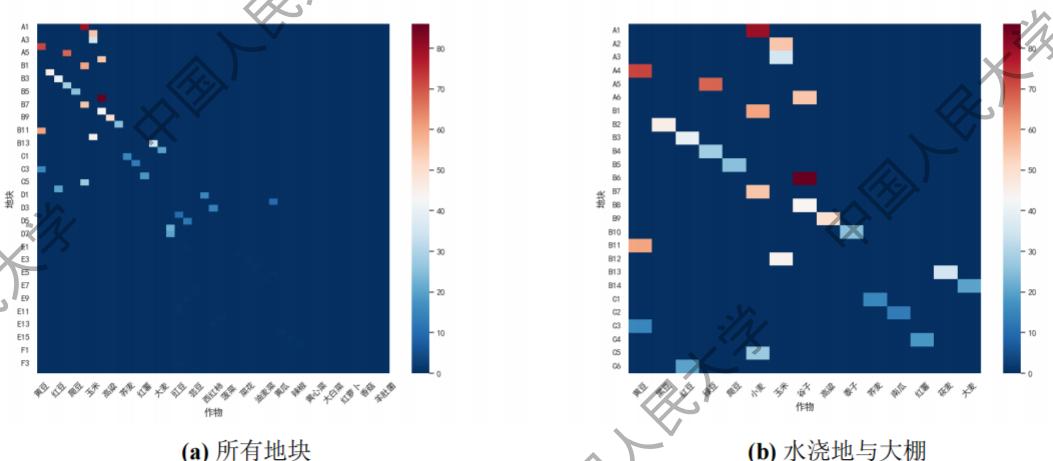


图 6 2023 年第一季种植策略

根据 2023 种植策略热力图（图 6、图 7），我们可以看出 2023 年的种植策略呈现分散的特点，即各种作物种植分散，较少存在连续多地块种植同一作物以及同一地块多种

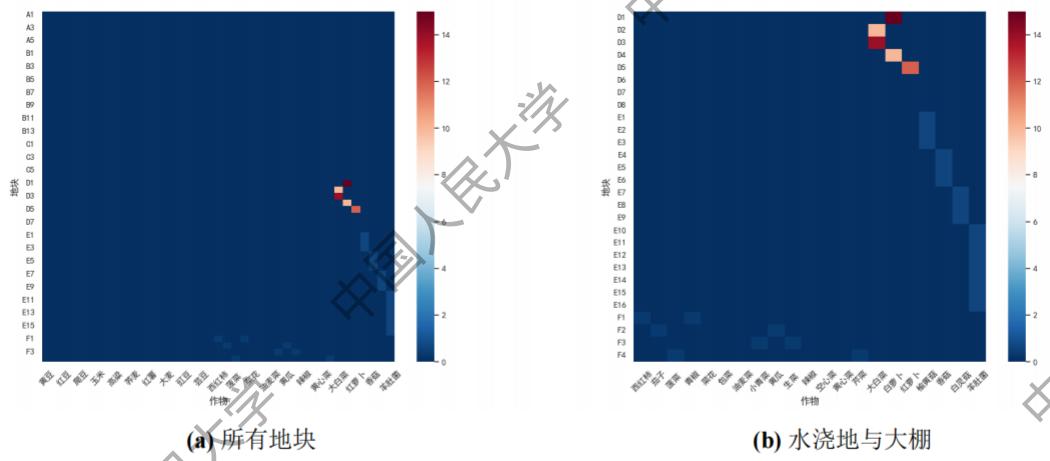


图 7 2023 年第二季种植策略

作物合种的情况，这为我们后续的逐年贪心规划提供了良好的初始条件。

5.2 问题一模型的建立与求解

5.2.1 模型初始化

5.2.1.1 模型假设说明

为了符合问题一的条件，同时使模型具有更好的泛化能力与良好的数学形式，我们做出如下约定与假设：

1. 认为所有地块与农作物均可种植两季，但将与实际情况不符的作物和地块的亩产量设置为 0，种植成本设置为极大，以此来等效地块和作物的种植季节差异。
2. 认为 2023 年的作物全部卖出，即各作物预期销售量等于 2023 年该作物总产量。
3. 认为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。即

$$\begin{cases} Z^{(k)} = Z^{(0)}, & 1 \leq k \leq 7 \\ C^{(k,m)} = C^{(0,m)}, & 1 \leq k \leq 7, m = 1, 2 \\ T^{(k,m)} = T^{(0,m)}, & 1 \leq k \leq 7, m = 1, 2 \\ P^{(k,m)} = P^{(0,m)}, & 1 \leq k \leq 7, m = 1, 2 \end{cases} \quad (2)$$

5.2.1.2 约束条件刻画

首先，我们给出模型的初始参数，即式(2)，我们记初始参数集合为 \mathcal{P}_0

确定初始参数后，我们对问题一所需要的约束条件进行数学刻画。

1. 由于地块的面积是一定的，我们在进行种植时不可能超出当前地块的总面积。这即

$$\sum_{j=1}^{41} x_{ij}^{(k,m)} \leq s_i, \quad 1 \leq i \leq 54 \quad \text{或者} \quad X^{(k,m)} V \leq S \quad (3)$$

其中 $V = (1, 1, \dots, 1)_{1 \times 54}^T$ 是一个列向量。

2. 由于每种作物在单个地块的种植面积不宜太小，我们设定在耕地中最小种植面积为 1 亩，大棚中最小种植面积为 0.1 亩。这即

$$\begin{cases} \text{当 } x_{ij}^{(k,m)} \neq 0 \text{ 时} & \begin{cases} x_{ij}^{(k,m)} \geq 1, & 1 \leq i \leq 34 \quad (S_i \text{ 为耕地}) \\ x_{ij}^{(k,m)} \geq 0.1, & 35 \leq i \leq 54 \quad (S_i \text{ 为大棚}) \end{cases} \end{cases} \quad (4)$$

3. 由于每种作物在同一地块（含大棚）都不能连续种植，我们这样考虑： $E^{(k,2)}$ 应该受到 $X^{(k,1)}$ 的影响，而 $E^{(k,1)}$ 会受到 $X^{(k-1,2)}$ 的影响，所以 $X^{(k,1)}$ 中不为零的位置对应 $E^{(k,2)}$ 中为 0，其中为零的位置对应 $E^{(k,2)}$ 中为 1， $E^{(k,1)}$ 情况类似。这即

$$\begin{cases} e_{ij}^{(k,2)} = 0, & \text{如果 } x_{ij}^{(k,1)} \neq 0 \\ e_{ij}^{(k,2)} = 1, & \text{如果 } x_{ij}^{(k,1)} = 0 \end{cases} \quad \begin{cases} e_{ij}^{(k,1)} = 0, & \text{如果 } x_{ij}^{(k-1,2)} \neq 0 \\ e_{ij}^{(k,1)} = 1, & \text{如果 } x_{ij}^{(k-1,2)} = 0 \end{cases} \quad (5)$$

我们可以写成矩阵的形式，如果引入示性函数 $I_0(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$ ，这即

$$\begin{aligned} E^{(k,2)} &= I_0(X^{(k,1)}) \\ E^{(k,1)} &= I_0(X^{(k-1,2)}) \end{aligned} \quad (6)$$

4. 由于要求每个地块（含大棚）的所有土地三年内至少种植一次豆类作物，我们需要根据前两年，即 $X^{(k-1,m)}, X^{(k-2,m)}$ 的数据来对 $E^{(k,m)}$ 限制。这即

$$\begin{cases} e_{ij}^{(k,m)} = 0, & \text{如果 } j \text{ 不是豆类且 } X^{(k-1)}, X^{(k-2)} \text{ 中对应地块未种植豆类} \\ e_{ij}^{(k,m)} = 1, & \text{如果不满足上面的情形} \end{cases} \quad (7)$$

以上即为第一问的全部约束条件，我们把公式(3),(4),(6),(7)组成的约束组记为 \mathcal{L}_k ，

以此来表示第 k 年时的约束。这即

$$\mathcal{L}_k : \begin{cases} X^{(k,m)}V \leq S \\ x_{ij}^{(k,m)} \geq 1, \quad 1 \leq i \leq 34 \quad (S_i \text{ 为耕地且 } x_{ij}^{(k,m)} \neq 0) \\ x_{ij}^{(k,m)} \geq 0.1, \quad 35 \leq i \leq 54 \quad (S_i \text{ 为大棚且 } x_{ij}^{(k,m)} \neq 0) \\ E^{(k,2)} = I_0(X^{(k,1)}) \quad E^{(k,1)} = I_0(X^{(k-1,2)}) \\ e_{ij}^{(k,m)} = 0, \text{ 如果 } j \text{ 不是豆类且 } X^{(k-1)}, X^{(k-1)} \text{ 中对应地块未种植豆类} \\ e_{ij}^{(k,m)} = 1, \text{ 如果不满足上面的情形} \end{cases} \quad (8)$$

5.2.1.3 目标函数建立

本题目目标是求解最优种植策略，我们认为最优是指经济效益的最大化，因此将每年的利润值作为目标函数。同时，由于本题提出两种情况，所以我们针对两个情况，分别给出目标函数表达式。

情况一：超过部分滞销，造成浪费

$$Target1 = \sum_{j=1}^{41} \left[\min \left(z_j^{(k)}, \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} \right) \cdot p_j^{(k,m)} - \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} c_{ij}^{(k,m)} \right] \quad (9)$$

其中， $\sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)}$ 表示的是第 j 种作物的总产量，这是由于 总产量 = \sum (是否种植 \times 每地块种植亩数 \times 地块亩产量)。

$\min \left(z_j^{(k)}, \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} \right)$ 表示总销量，这是由于超过预期销量的部分会产生滞销造成浪费，所以我们取总产量和预期销量的较小值作为真实销量。预期销量乘上预期售价，即可得到农作物 j 的总收入。

$\sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} c_{ij}^{(k,m)}$ 计算的是第 j 种作物种植的总成本。

最后我们用总收入减去总成本，得到第 j 种作物的净收益，并用 $\sum_{j=1}^{41}$ 遍历所有农作物，得到最终的总利润。

我们采用矩阵的语言重写上述目标：

$$Target1 = \left(\min \left[Z, U(E \odot X \odot T) \right] \odot P - U(E \odot X \odot C) \right) V \quad (10)$$

式中 $U = (1, 1, \dots, 1)_{1 \times 41}$ 是一 1×41 的行向量， $V = (1, 1, \dots, 1)_{54 \times 1}^T$ 是一 54×1 的列向量。

④ 是矩阵的哈达玛乘积，对矩阵相同位置的元素相乘并得到新矩阵。

情况二：超过部分按 2023 年销售价格的 50% 降价出售

$$Target2 = \sum_{j=1}^{41} \left[p_j^{(k,m)} \cdot \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - \max \left(0, \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - z_j^{(k)} \right) \cdot \frac{p_j^{(k,m)}}{2} \right. \\ \left. - \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} c_{ij}^{(k,m)} \right] \quad (11)$$

其中 $p_j^{(k,m)} \cdot \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - \max \left(0, \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - z_j^{(k)} \right) \cdot \frac{p_j^{(k,m)}}{2}$ 即为考虑 50% 折价出售后的总收入。其余部分与 Target1 类似，我们略去详细解释。

矩阵形式：

$$Target2 = \left[U(E \odot X \odot T) \odot P - \max(0, U(E \odot X \odot T) - Z) \odot \frac{P}{2} \right. \\ \left. - U(E \odot X \odot C) \right] V \quad (12)$$

5.2.2 求解器创建与年间状态转移

由 5.2.1 中的讨论，我们得到了初始参数集 \mathcal{P}_0 ，约束条件集 \mathcal{L}_k 与目标函数 $Target1, Target2$ ，下面我们根据这三者创建求解器

$$X^{(k+1,m)} = \mathcal{F}(X^{(k,m)}, \mathcal{P}_0, \mathcal{L}_k, Target) \quad (13)$$

求解器 $\mathcal{F}(X^{(k,m)}, \mathcal{P}_0, \mathcal{L}_k, Target)$ 的作用是：如果输入今年的种植情况 $X^{(k,m)}$ 与模型初始参数集 \mathcal{P}_0 ，那么求解器会返回满足约束条件集 \mathcal{L}_k ，且使得目标函数 $Target$ 最大的下一年的种植策略 $X^{(k+1,m)}$ 。

在初始参数、约束形式与目标函数不改变的情况下，我们可以简记为 $\mathcal{F}(X^{(k,m)})$ 。那么很自然地，年间的状态转移方程即为：

$$X^{(k,m)} = \mathcal{F}(X^{(k-1,m)}) = \mathcal{F}(\mathcal{F}(X^{(k-2,m)})) \\ = \dots = \mathcal{F}^k(X^{(0,m)}) \quad (14)$$

5.2.3 求解器实现——基于优先队列的贪心算法

由于模型求解器 \mathcal{F} 中的约束集 \mathcal{L}_k 十分复杂，且约束条件相互交叉，依赖传统的线性规划算法很难对约束条件进行良好的规范，同时种植策略矩阵 $X_{54 \times 41}$ 十分庞大，参

数量较多，直接使用常见的最优化算法可能面临内存溢出与代码超时的问题，因此我们尝试构建新的优化算法，以针对该问题实现模型的求解器。

我们注意到，目标函数是总利润的最大化，且超产造成的滞销和折价销售会限制农产品的种植亩数，所以我们应该考虑尽量优先种植利润高的作物。通过 5.1.2.2 中亩利润的分析，我们知道，在同一季中，每种农产品的亩利润，由其本身和所种植的地块类型唯一确定（见图 4）。综上，我们可以以亩利润的大小作为优先级的评判指标，构建优先队列。

优先队列（Priority Queue）是一种特殊的队列数据结构，每个元素都有一个与之相关的优先级。与普通队列不同，优先队列的出队顺序不是按照插入顺序，而是按照元素的优先级来决定的。优先级高的元素会比优先级低的元素先出队。

在本模型中，优先级即为亩利润的大小，亩利润越大，优先级越高。优先队列在此模型中的优势在于，本模型需要进行频繁的元素的插入、查找和删除操作，倘若直接使用数组，插入、查找和删除的时间复杂度为 $O(1), (n \log n), O(1)$ ，总时间复杂度为 $O(n \log n)$ ，而使用优先队列总时间复杂度为 $O(\log n)$ ，极大降低了算法运行的时间，提升了计算效率。

为了方便我们在取出优先队列中的最大利润时，获得该利润所对应的作物名称和地块类型，我们采取往优先队列里存入（亩利润，作物编号，地块类型编号）这种三元组的方式，此时的优先队列仍能以亩利润作为优先级的评判标准。

本求解器贪心算法的核心在于，根据我们所构建的优先队列，每次决策都优先选择亩利润最高的作物，种植在对应地块，同时在过程中判断是否满足约束条件，并以使总利润最大为目标，实现按照亩利润降序对农作物进行种植，从而达到充分扩大高亩利润作物的收益，并达到总利润的最大化（见附录 E、F）。

我们绘制如下代码流程图，以实现对算法的直观解释：

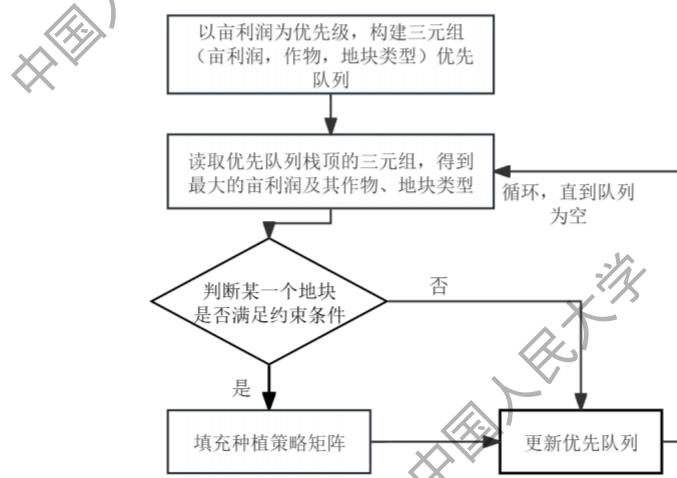


图 8 贪心算法程序框图

5.2.4 结果分析

情况一超过部分滞销，造成浪费

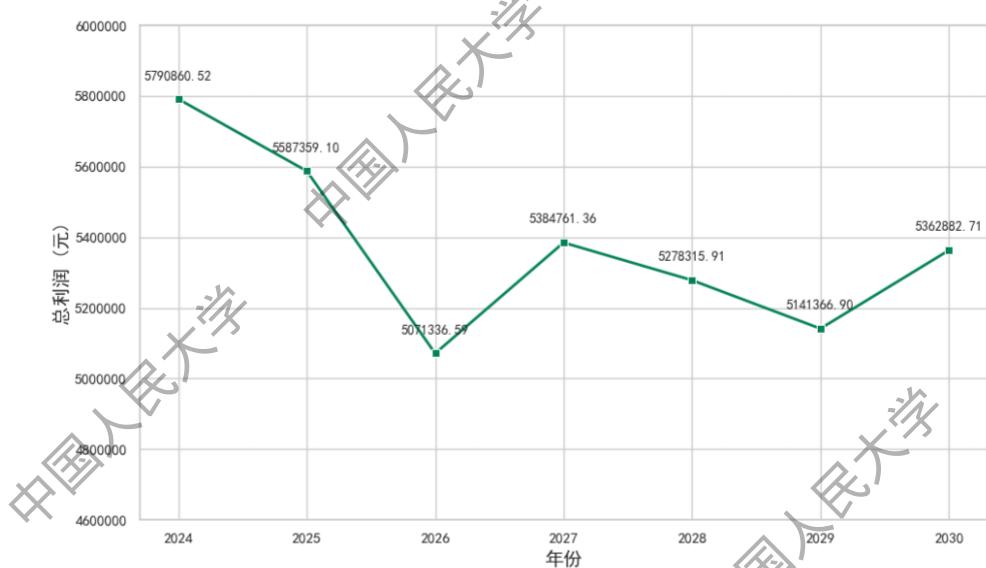


图 9 情况一利润折线图

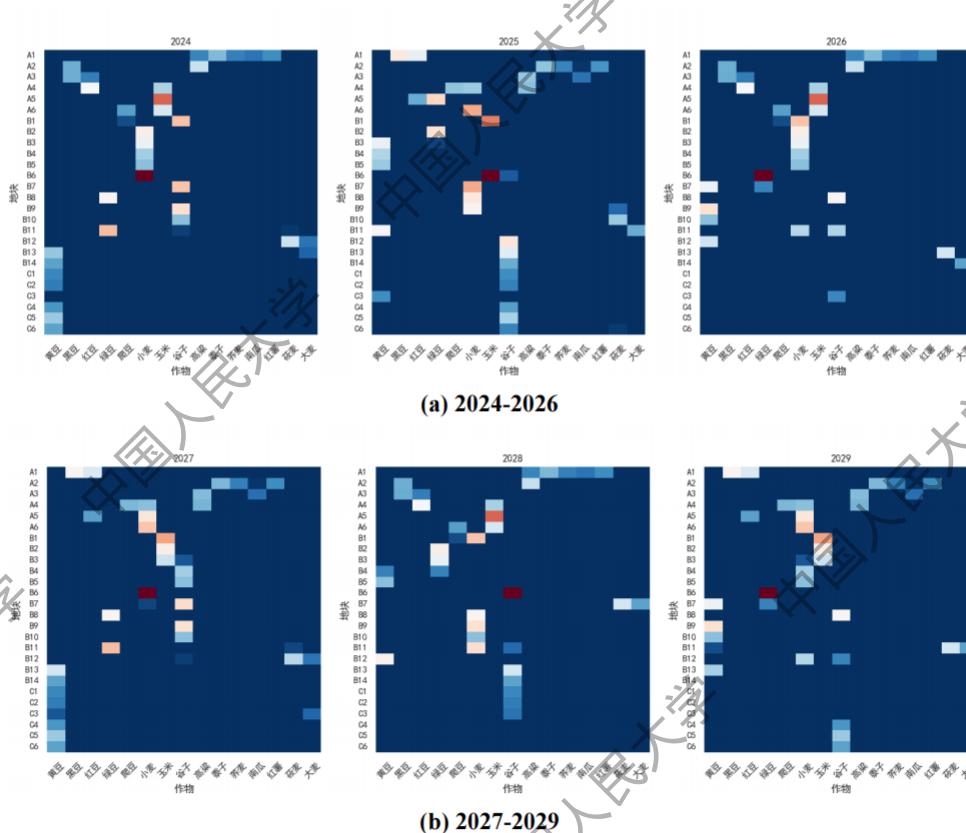


图 10 问题一情况一种植策略

情况一的年利润呈现出先波动后稳定的趋势（图 9），符合贪心算法的特征，在 *result1_1.xlsx* 中，种植策略整体较为分散，但也呈现出一定的周期性，例如 B13-C6 地

块在六年间的种植策略在黄豆和谷子之间徘徊，推测是因为在超产滞销的条件下，由于黄豆和谷子的预期销售量较高，更容易从优先队列中取出来种植更多的地块，同时由于黄豆为豆科植物，能够很好的符合三年种一次豆类的约束，从而形成了黄豆-谷子-黄豆的循环（图 10）。

情况二：超过部分按 2023 年销售价格的 50% 降价出售

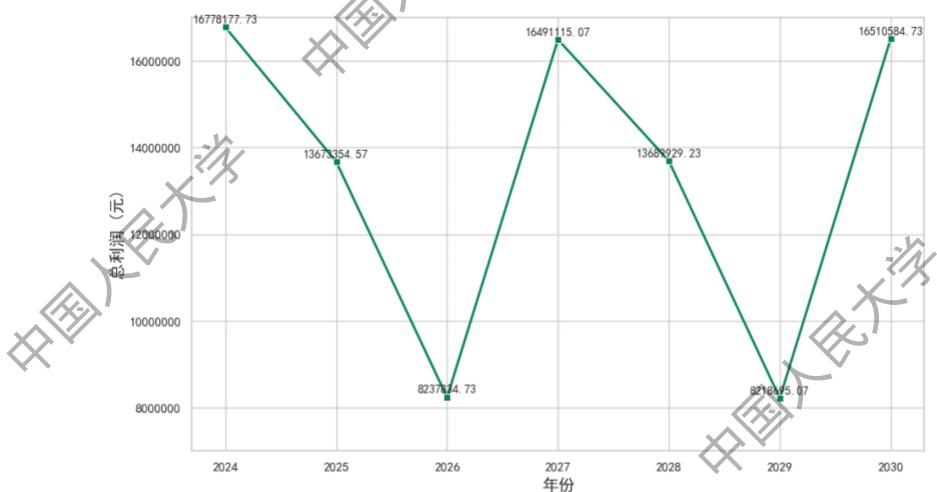


图 11 情况二利润折线图

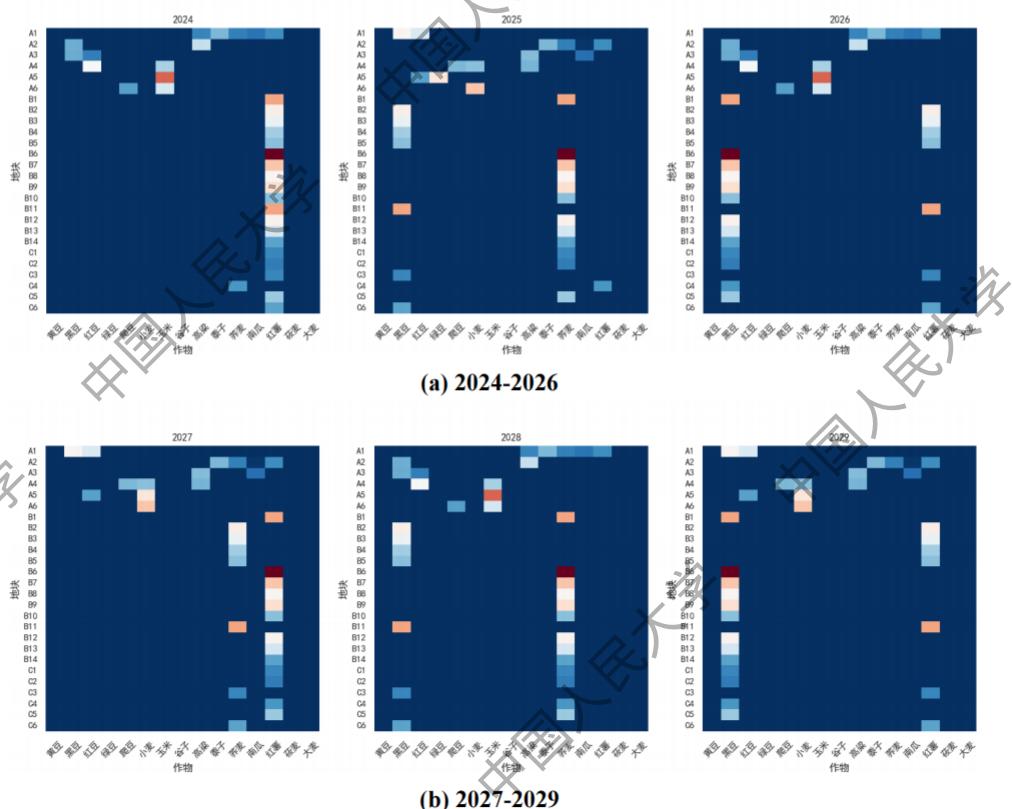


图 12 问题一情况二种植策略

情况二的年利润呈现出更为强烈周期性与波动性(图 11), 在 *result1_2* 中也明显体现出种植策略的周期性, 并且存在某几种作物的大量集中种植的现象。如在 A、B、C 地块中呈现出红薯-荞麦-黑豆的循环种植策略。经过查询附件可以得知, 红薯、荞麦为 A、B、C 地块中亩利润最高的作物, 而黑豆为豆科植物中亩利润最高的作物, 即使有超产降价 50% 的约束也是如此, 而无论超产多少均可售出的条件也使得这三种作物能够被反复大量种植, 从而在满足三年种一次豆科植物、不连续重茬种植的条件下实现了利益最大化(图 10)。

5.3 问题二模型的建立与求解

5.3.1 模型初始化

根据问题分析, 我们知道问题二紧承问题一, 二者共享相同的约束条件。我们只需要对问题一模型中的初始参数集以及目标函数 *Target* 进行修正即可。

5.3.1.1 初始参数集的修正

考虑到题目二给出的信息:

1. 小麦和玉米预期销售量平均年增长率介于 5%~10%, 其他农作物预期销售量相对于 2023 年大约有 $\pm 5\%$ 的变化;
2. 农作物的亩产量往往会每年会有 $\pm 10\%$ 的变化;
3. 农作物的种植成本平均每年增长 5% 左右;
4. 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, 食用菌的销售价格每年可下降 1%~5%, 羊肚菌的销售价格每年下降幅度为 5%。

由于上述条件都是范围或估计值, 所以我们以随机数 r_i 来模拟指标的波动, 得到以下参数序列的状态转移方程:

$$\begin{cases} z_j^{(k+1)} = z_j^{(k)} \cdot (1 + r_1), & j = 6, 7 \text{ 玉米和小麦的销售量变化} \\ z_j^{(k)} = z_j^{(0)} \cdot (1 + r_2), & j \neq 6, 7 \text{ 其他作物的销售量变化} \\ t_{ij}^{(k+1)} = t_{ij}^{(k)} \cdot (1 + r_3), & \text{农作物亩产量变化} \\ c_{ij}^{(k+1)} = c_{ij}^{(k)} \cdot (1 + r_4), & \text{种植成本变化} \\ p_j^{(k+1)} = p_j^{(k)} \cdot (1 + r_5), & 17 \leq j \leq 37 \text{ 蔬菜类作物销售价格变化} \\ p_j^{(k+1)} = p_j^{(k)} \cdot (1 + r_6), & 38 \leq j \leq 40 \text{ 食用菌销售价格变化} \\ p_{41}^{(k+1)} = p_{41}^{(k)} \cdot (1 + r_7), & \text{羊肚菌销售价格变化} \end{cases} \quad (15)$$

我们通过以上公式可以得到每一年的初始参数 P_k 。

5.3.1.2 目标函数的修正

在第一问中, 题目给出了两种情况, 即:

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

第二问中并没有对超产的情况做出规定，因此我们需要结合实际以及第一问的情况做出限制：实际销售中，超产部分折价销售是更为合理的。但同时我们注意到，在问题一里，折价销售情形的利润过于高，这是不正常的。经过分析，我们发现问题所在：由于一些作物利润很高，折价销售对其在优先队列中的影响不大，导致求解器在执行贪心算法时会过度种植某种作物。因此我们需要设定单一作物销量的最大上限，即超产到何种程度时，超出部分会产生滞销。

根据市场饱和理论与生活常识，销量上限通常在两倍预期销量左右。所以，预期产量以内的部分按照原价出售，超出预期产量但小于二倍预期产量的部分按照原价的 50% 出售，超出二倍预期产量的部分滞销。这即：

$$\text{售价} = \begin{cases} p_j^{(k,m)}, & \text{小于等于预期产量的部分} \\ p_j^{(k,m)}/2, & \text{预期产量与二倍预期产量之间的部分} \\ 0, & \text{超出二倍预期产量的部分} \end{cases}$$

根据上述讨论，我们给出问题二对应的目标函数 Target：

$$\left\{ \begin{array}{l} \text{Target} = \sum_{j \in I_1} \text{利润}_j + \sum_{j \in I_2} \text{利润}_j \\ \sum_{j \in I_1} \text{利润}_j = \sum_{j \in I_1} \left(p_j^{(k,m)} \cdot \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} c_{ij}^{(k,m)} \right) \\ \sum_{j \in I_2} \text{利润}_j = \sum_{j \in I_2} \left[p_j^{(k,m)} \cdot z_j^{(k)} + \min \left(z_j^{(k)}, \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} t_{ij}^{(k,m)} - z_j^{(k)} \right) \cdot \frac{p_j^{(k,m)}}{2} \right. \\ \quad \left. - \sum_{i=1}^{54} e_{ij}^{(k,m)} x_{ij}^{(k,m)} c_{ij}^{(k,m)} \right] \end{array} \right. \quad (16)$$

其中， I_1, I_2 为指标集，分别表示产量小于等于预期产量的作物 j 组成的集合和产量大于预期产量的作物 j 组成的集合。

5.3.2 模型求解与分析

第二问求解器可以沿用第一问，即： $X^{(k+1,m)} = \mathcal{F}(X^{(k,m)}, \mathcal{P}_k, \mathcal{L}_k, \text{Target})$ 只需要逐年对初始参数集 \mathcal{P}_k 以及约束集 \mathcal{L}_k 更新。年间的状态转移也是类似的，我们不再赘述。

通过简单调整第一问的求解器，我们求解出第二问的最优策略，并进行如下的分析：

问题二的年利润也呈现出一定的周期性（图 13），但相比于问题一的情形二波动幅度更低。在 result2 的种植矩阵中也体现出更加多元化的种植策略，能够有效避免市场饱和的情况出现，更符合实际生活（图 14）。

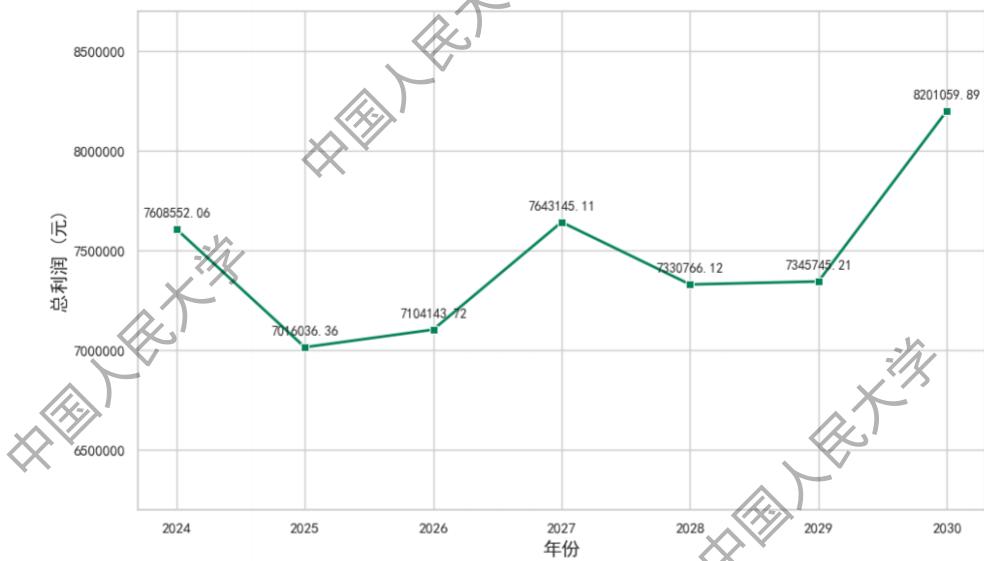


图 13 问题二利润折线图

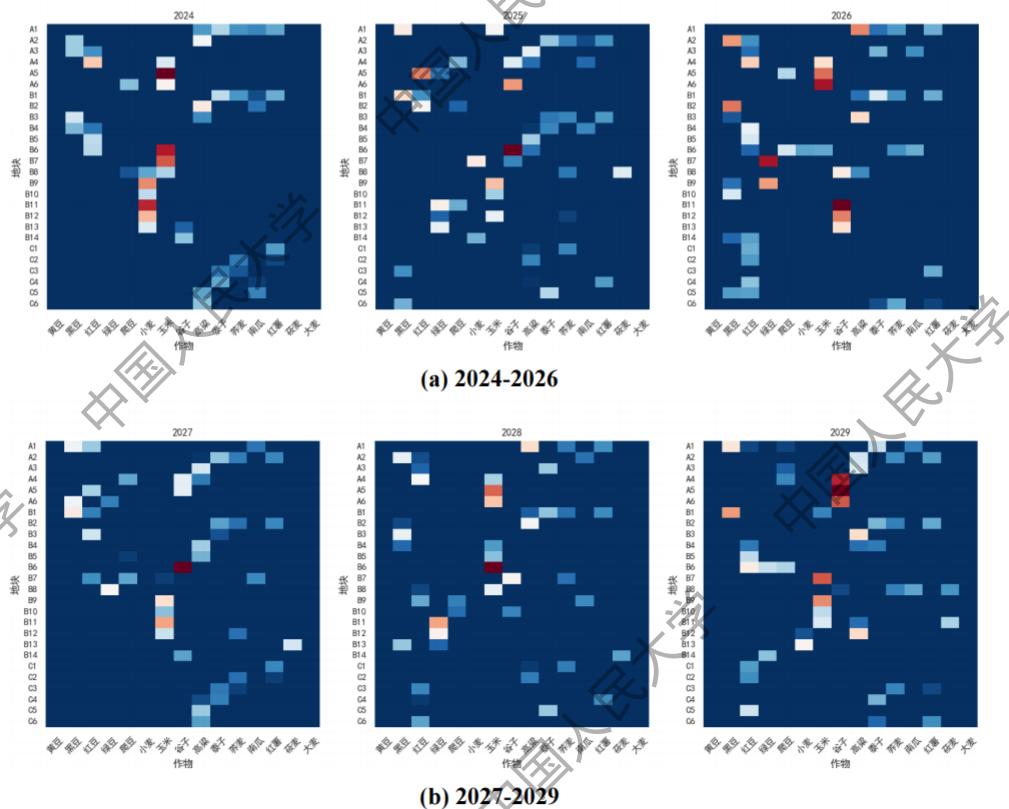


图 14 问题二种植策略

5.4 问题三模型的建立与求解

5.4.1 模型初始化

问题三指出在现实生活中，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。我们需要在第二问的基础上，考虑三种额外因素：农作物的可替代性、农作物的互补性以及预期销售量、销售价格、种植成本之间的关系。下面我们将首先对于这三种额外因素进行分析。

5.4.1.1 预期销售量、销售价格、种植成本之间的关系

首先，根据经济学原理，销售价格与种植成本应该显著正相关。同时我们使用斯皮尔曼相关系数对销售价格与预期销售量进行相关性检验，得到如下结果（图 15）

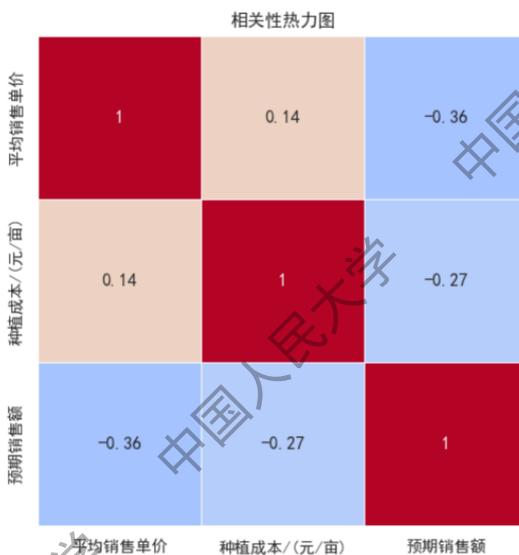


图 15 斯皮尔曼相关性检验

经相关性检验，在 95% 的置信度下，我们认为平均销售单价与预期销售额呈现负相关关系，从而可以构建需求弹性模型。

定义弹性：

$$E_d = \frac{\% \text{ 销售量变化}}{\% \text{ 价格变化}} \quad (17)$$

- 如果 $|E_d| > 0.7$ ，即销售量对价格的变化较为敏感，称为富有弹性，例如某些替代性较高的蔬菜。
- 如果 $|E_d| \geq 0.7$ ，即销售量对价格变化不敏感，称为缺乏弹性，例如主粮作物（如小麦、大米）

我们通过查阅文献，得到了作物的价格弹性表格（见附录 G）。

5.4.1.2 农作物的可替代性

农作物的可替代性是指在特定的农业生产环境中，一种作物可以被另一种作物替代的能力。这种替代性通常受到多种因素的影响，主要在于以下两方面：

- 市场需求：如果市场对某种作物的需求增加，农民可能会选择种植这种作物，而减少其他作物的种植。
- 价格弹性：作物的价格弹性影响农民的种植决策。例如，价格弹性较高的作物在价格上涨时，农民更有可能增加种植面积。

在实际考虑中，我们可以用 5.4.1.1 中的价格弹性的不同来对农作物的可替代性做出刻画。

5.4.1.3 农作物的互补性

农作物间的互补性是指在农业生产中，不同作物之间的种植可以相互促进、提高资源利用效率和产量的特性。具体来说，互补性体现在以下几个方面：

- 光照：高杆作物与低矮作物间作时，高杆作物可以遮挡阳光，减少土壤水分蒸发，而低矮作物可以在阴影下生长，充分利用光照资源；
- 水分：不同作物的根系深度和广度不同，深根作物可以有效吸收地下水，而浅根作物则利用表层水分，从而减少水分竞争；
- 氮固定：豆科作物具有固氮能力，可以提高土壤中的氮含量，促进与禾本科作物的间作，后者在生长过程中能更好地吸收这些养分；

在本题中，我们主要考虑豆科和其他作物的间作合种。根据参考文献，豆科的增产效果主要来源于其固氮作用，对氮需求高的作物影响显著，但对耐贫瘠、根系浅的受豆科增产效果差。由此我们根据作物的特性对其受到豆科增产效果的幅度分为三个档次（见附录 H），分别有 70%、30%、10% 的亩产量增幅。又经查阅文献可知，间作对豆科自身产量增幅不太明显，因此给予其 10% 的间作亩产量增幅。

5.4.1.4 模型修正

根据上述讨论，我们发现问题三的限制实际上只对每年的参数集 \mathcal{P}_k 添加了更为精细的结构，并没有改变约束条件集与目标函数。因此我们只需要在问题二的参数集上添加限制，即

1. 通过引入价格弹性来度量农作物间的可替代性，并刻画预期销售量与销售价格之间的关系；
2. 通过考虑与豆科植物的间作来刻画各种农作物之间的互补性；

故我们添加如下条件：

$$\begin{cases} \frac{\Delta z_j}{z_j} = \frac{\Delta p_j}{p_j} \times E_d \\ t_{ij} = t_{ij} \times (1 + \delta_n), \text{ 如果与豆科植物间种} \end{cases} \quad (18)$$

5.4.2 模型求解与分析

第三问我们仍使用求解器 $X^{(k+1,m)} = \mathcal{F}(X^{(k,m)}, \mathcal{P}_k, \mathcal{L}_k, Target)$ 来进行求解，并逐年维护参数集 \mathcal{P}_k 和约束集 \mathcal{L}_k ，得到 2024~2030 年间的种植策略。

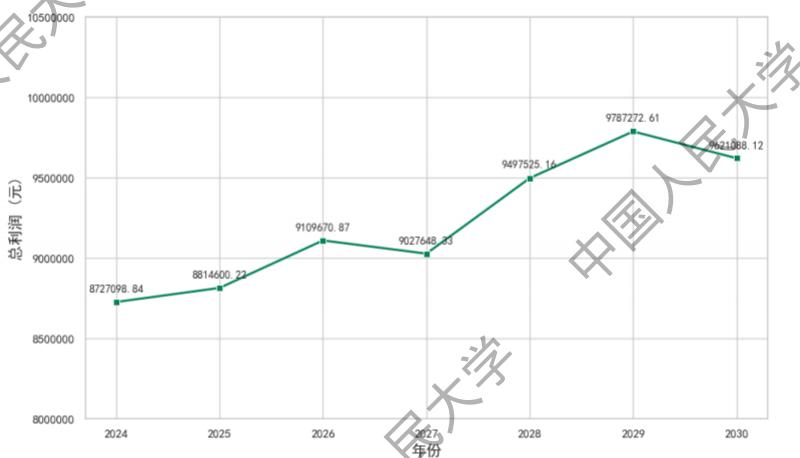


图 16 问题三利润折线图

第三问的总利润相比第二问有着显著的提升（图 16），猜测是由于间作带来的产量增加引起的，经过对比试验证实了我们的猜想，在 result3 中也大量体现出了豆科植物和其他作物间作的模式，说明考虑间作能够显著提升务农的收益（图 17）。

六、模型的评价与推广

6.1 模型评价

优点

1. 算法针对性强：采用优先队列实现求解器关注到了数据中所存在的序结构（即亩利润大小），从而对本题具有更好的针对性；
2. 模型轻便：采用优先队列驱动的贪心算法，大大加快了模型求解速度，同时也保证了结果的优秀程度；
3. 泛化能力强：模型求解器实现了参数、约束、目标的分离，仅需对某一板块做出限制即可解决新的问题，具有良好的泛化性

局限

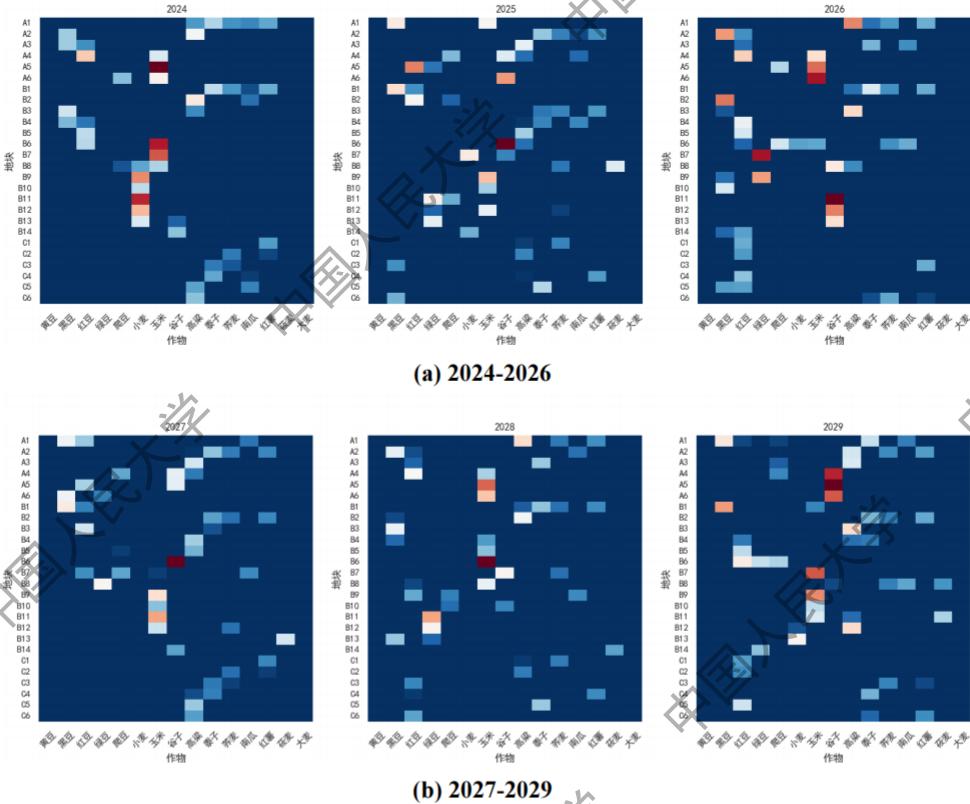


图 17 问题三种植策略

1. 局部最优：贪心算法只能保证每个步骤的局部最优，而未必能得到全局最优解，可能错失更好的长期收益；
2. 当约束条件复杂时，如种植密集程度要求，贪心算法可能无法充分满足所有约束；
3. 非线性收益或成本的挑战
4. 未考虑到市场竞争等其他经济学因素的影响收益/成本非线性：贪心算法适用于线性收益和成本场景，但当收益或成本随着投入的增加而发生非线性变化时，算法的效果会受到限制。

6.2 模型推广

本模型基于优先队列构建了最优化求解器，通过状态转移方程逐年贪心求解最优，且模型仅需要给出参数集与约束条件集以及目标函数集即可自动实现最优化目标函数并返回对应策略。因此该模型对于多种优化问题均可以实现快速求解。我们可以把模型推广至其他涉及时间序列或状态转移以及具有良好序结构的优化问题中，如劳动力布局、机械协同、航班调度等生产生活实际。

参考文献

- [1] Chunjie Li,Tjeerd-Jan Stomph,David Makowski, et al. The productive performance of intercropping[J/OL].PNAS,2023. <https://www.pnas.org/doi/pdf/10.1073/pnas.2201886120>. DOI:10.1073/pnas.2201886120.
- [2] 高阳, 段爱旺, 刘祖贵, 等. 单作和间作对玉米和大豆群体辐射利用率及产量的影响 [J]. 中国生态农业学报,2009,17(1):7-12. C. K.
- [3] Hiebsch, F. Tetio-Kagho, A. M. Chirembo, et al. Plant Density and Soybean Maturity in a Soybean-Maize Intercrop[J].Agronomy Journal,1995: 965-969.
- [4] Wentao Dong, Yayun Zhu, Huizhong Chang, et al.An SHR-SCR module specifies legume cortical cell fate to enable nodulation[J/OL].Nature,2020,589:586-590.<https://www.nature.com/articles/s41586-020-3016-z.pdf>.DOI:10.1038/s41586-020-3016-z.
- [5] Jiangli Dong,Tao Wang. Legume-specific SnRK1 promotes malate supply to bacteroids for symbiotic nitrogen fixation[J/OL].Molecular Plant,2023,16(9):1396-1412. [https://www.cell.com/molecular-plant/pdfExtended/S1674-2052\(23\)00244-7](https://www.cell.com/molecular-plant/pdfExtended/S1674-2052(23)00244-7). DOI:10.1016/j.molp.2023.08.009.
- [6] 陈军, 我国农产品滞销的疏通服务机制研究 [D/OL].<https://www.sinoss.net/uploadfile/2018/1031/20181031113529196.pdf>.
- [1] 中华人民共和国农业农村部,基于农产品属性分类的农业供给侧结构性改革对策研究 [R/OL]. (2017-01-07)https://www.jhs.moa.gov.cn/zlyj/201904/t20190418_6181045.htm.

附录 A 数据导入-pyton 源代码

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
%config InlineBackend.figure_format = 'svg'

T1 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="第一季亩产量", index_col=0)
T2 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="第二季亩产量", index_col=0)
C1 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="第一季种植成本", index_col=0)
C2 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="第二季种植成本", index_col=0)
price = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="销售单价", index_col=0)
X01 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="2023第一季种植情况", index_col=0)
X02 = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="2023第二季种植情况", index_col=0)
```

附录 B 耕地基本信息-jupyter 源代码

```
S = pd.read_excel("datacleaning_matrix.xlsx", sheet_name="耕地面积", index_col=0)
data = {
    'A': [80.0, 55.0, 35.0, 72.0, 68.0, 55.0],
    'B': [60.0, 46.0, 40.0, 28.0, 25.0, 86.0, 55.0, 44.0, 50.0, 25.0, 60.0, 45.0, 35.0, 20.0],
    'C': [15.0, 13.0, 15.0, 18.0, 27.0, 20.0],
    'D': [15.0, 10.0, 14.0, 6.0, 10.0, 12.0, 22.0, 20.0],
    'E': [0.6] * 15,
    'F': [0.6] * 4,
}

# 创建 DataFrame
df = pd.DataFrame(dict([(k, pd.Series(v)) for k, v in data.items()]))

# 计算描述性统计
stats = df.describe()
print(stats)

# 设置绘图风格
sns.set(style="whitegrid", font='SimHei')

# 绘制箱线图
plt.figure(figsize=(10, 6))
sns.boxplot(data=df)
plt.ylabel('耕地面积')
plt.xlabel('区域')
plt.savefig('六大地块各自面积分布.png')
```

```

plt.show()

sizes = df.sum()

# 设置颜色
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'pink', 'plum']

# 创建饼图
plt.figure(figsize=(8, 8))
patches, texts, autotexts = plt.pie(sizes, colors=colors, autopct='%.1f%%', startangle=90)

# 设置扇区为椭圆形
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# 设置标签字体大小
plt.setp(autotexts, size=16, weight="bold")
plt.setp(texts, size=16)

# 添加标题和图例
plt.legend(patches, df.columns, loc='upper right', fontsize=14)

# 显示图形
plt.axis('equal')
plt.tight_layout()
plt.savefig('耕地面积占比.png')
plt.show()

```

附录 C 苗利润的可视化-jupyter 源代码

```

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%config InlineBackend.figure_format = 'svg'

#替换极大值，便于绘图
C1.replace(999999999, 0, inplace=True)
C2.replace(999999999, 0, inplace=True)
#计算苗利润
Profit_perblock1=price.iloc[0]*T1-C1
Profit_perblock2=price.iloc[1]*T2-C2
#编写绘图函数
def plot_profit_heatmap(profit_matrix,cmap='YlGnBu',annot=False,name=0):

```

```

"""
绘制亩利润热图

参数:
profit_matrix: DataFrame, 表示每个地块上不同作物的亩利润
"""

plt.figure(figsize=(10,8)) # 设置图形大小
sns.set(font='SimHei')
sns.heatmap(profit_matrix, annot=annot, fmt=".0f", cmap=cmap, cbar=True,
            linewidths=0, linecolor='white') # 绘制热图
# plt.title("亩利润热图") # 设置标题
plt.xlabel("作物") # 设置x轴标签
plt.ylabel("地块") # 设置y轴标签
plt.xticks(rotation=45) # 设置x轴刻度旋转角度
plt.yticks(rotation=0) # 设置y轴刻度旋转角度
if(name!=0):
    plt.savefig(name,format='png')
plt.show() # 显示图形

plot_profit_heatmap(Profit_perblock1,cmap='RdBu_r',name='第一季亩利润热力图.png')
plot_profit_heatmap(Profit_perblock1.loc['A1':'C6','黄豆':'大麦'],cmap='RdBu_r',
                     name='第一季平旱梯田山坡地亩利润热力图.png')
plot_profit_heatmap(Profit_perblock1.loc['D1':'F4','水稻':'芹菜'],cmap='RdBu_r',
                     name='第一季水浇地与大棚亩利润热力图.png')
plot_profit_heatmap(Profit_perblock2,cmap='RdBu_r',name='第二季亩利润.png')
plot_profit_heatmap(Profit_perblock2.loc['D1':'F4','豇豆':'羊肚菌'],cmap='RdBu_r',
                     name='第二季水浇地与大棚亩利润热力图.png')

```

附录 D 2023 种植策略的可视化-jupyter 源代码

```

def plot_profit_heatmap(profit_matrix,cmap='YlGnBu',annot=False,name=0):
"""
绘制亩利润热图

参数:
profit_matrix: DataFrame, 表示每个地块上不同作物的亩利润
"""

plt.figure(figsize=(10,8)) # 设置图形大小
sns.set(font='SimHei')
sns.heatmap(profit_matrix, annot=annot, fmt=".0f", cmap=cmap, cbar=True,
            linewidths=0, linecolor='white') # 绘制热图
# plt.title("亩利润热图") # 设置标题
plt.xlabel("作物") # 设置x轴标签
plt.ylabel("地块") # 设置y轴标签
plt.xticks(rotation=45) # 设置x轴刻度旋转角度

```

```

plt.yticks(rotation=0) # 设置y轴刻度旋转角度
if(name!=0):
    plt.savefig(name,format='png')
plt.show() # 显示图形

plot_profit_heatmap(X01,cmap='RdBu_r',name='2023第一季策略.png')
plot_profit_heatmap(X01.loc['A1':'C6','黄豆':'大麦'],cmap='RdBu_r',
                     name='2023第一季干旱梯田山坡地策略.png')
plot_profit_heatmap(X02,cmap='RdBu_r',name='2023第二季策略.png')
plot_profit_heatmap(X02.loc['D1':'F4','西红柿':'羊肚菌'],cmap='RdBu_r',
                     name='2023第二季水浇地大棚策略.png')

```

附录 E 二次数据处理

```

# 第一季
# 降价后利润 (作物编号、地块编号): 降价后利润的字典
descenddict1 = {}
# 苗产量 (作物编号、地块编号): 每亩产量的字典
muchan1 = {}
# 优先队列, 用于存储作物编号、地块编号和全亩利润 (种植收益) 信息。队列中的元素按照全亩利润降
序排列。
pqall1 = []
# 遍历第一季的数据 data1_1, 从第二项开始 (跳过表头)
for data in data1_1[1:]:
    values = list(data.values()) # 提取该行数据的值
    # 计算降价后的利润, 并存入 descenddict1 字典, 键为 (作物编号, 地块编号)
    descenddict1[(values[0], values[1])] = values[2] / 2 * values[3] -
values[4]
    # 记录亩产量, 并存入 muchan1 字典, 键为 (作物编号, 地块编号)
    muchan1[(values[0], values[1])] = values[3]
    # 计算全亩利润, 并将其作为负数插入优先队列 pqall1, 以便最大值优先
    heappush(pqall1, (-(values[2] * values[3] - values[4]), values[0],
values[1]))
# 第二季
# 降价后利润 (作物编号、地块编号): 降价后利润的字典
descenddict2 = {}
# 苗产量 (作物编号、地块编号): 每亩产量的字典
muchan2 = {}
# 优先队列, 用于存储作物编号、地块编号和全亩利润 (种植收益) 信息, 按利润降序排列
pqall2 = []
# 遍历第二季的数据 data1_2, 从第二项开始 (跳过表头)
for data in data1_2[1:]:
    values = list(data.values()) # 提取该行数据的值
    # 计算降价后的利润, 并存入 descenddict2 字典, 键为 (作物编号, 地块编号)
    descenddict2[(values[0], values[1])] = values[2] / 2 * values[3] -

```

```

values[4]
    # 记录亩产量，并存入 muchan2 字典，键为（作物编号，地块编号）
    muchan2[(values[0], values[1])] = values[3]
    # 计算全亩利润，并将其作为负数插入优先队列 pqall2，以便最大值优先
    heapq.heappush(pqall2, (-(values[2] * values[3] - values[4]), values[0],
values[1]))
    # 豆类列表，用集合 pealist 存储所有豆类作物的编号
    pealist = set()
    # 遍历豆类作物的数据 data4，从第二项开始
    for data in data4[1:]:
        values = list(data.values()) # 提取该行数据的值
        # 将作物编号（豆类）加入集合 pealist
        pealist.add(values[0])
    # 预期销售量，创建字典 maxsold 存储每种作物的最大销售量
    maxsold = {}
    # 遍历所有作物的预期销售量数据 data2，从第二项开始
    for data in data2[1:]:
        values = list(data.values()) # 提取该行数据的值
        # 记录作物编号及其最大销售量，存入 maxsold 字典
        maxsold[values[0]] = values[1]
    # 地块类型和面积信息，创建字典 ground，键为地块类型，值为地块编号及其面积的列表
    ground = defaultdict(list)
    # 遍历地块信息数据 data3，从第二项开始
    for data in data3[1:]:
        values = list(data.values()) # 提取该行数据的值
        # 记录地块类型及其对应的地块编号和面积，并存入 ground 字典
        ground[values[1]].append([values[0], values[2]])
    # 如果当前年份是 2023
    if year == 2023:
        # 是否种豆类，创建字典 hvpea，用于记录地块是否种过豆类作物
        hvpea = {}
        # 遍历地块信息数据 data3，从第二项开始
        for data in data3[1:]:
            values = list(data.values()) # 提取该行数据的值
            # 记录地块编号及其是否种过豆类作物的状态，存入 hvpea 字典
            hvpea[values[0]] = values[3]
        # 上一季种植信息，创建字典 hvwhat，键为地块编号，值为种植的作物编号及其面积（初始为
        0）
        hvwhat = defaultdict(list)
        # 遍历上一季的种植信息数据 data5
        for data in data5:
            values = list(data.values()) # 提取该行数据的值
            # 记录地块编号及其种植的作物编号，并将面积初始化为 0，存入 hvwhat 字典
            hvwhat[values[0]].append([values[1], 0])
    else:
        # 如果当前年份不是 2023，则读取之前保存的 hvpea 和 finalhvwhat 文件
        # hvpea.json 存储了地块是否种植过豆类的信息

```

```

        with open(os.path.join(directory, str(year), "hvpea.json"), 'r',
encoding='utf-8') as f:
            hvpea = json.load(f)
            # finalhvwhat.json 存储了上一季的作物种植信息
            with open(os.path.join(directory, str(year), "finalhvwhat.json"), 'r',
encoding='utf-8') as f:
                hvwhat = json.load(f)

```

附录 F 贪心算法核心函数

```

import json
import os
import heapq
import copy
from collections import defaultdict
from openpyxl import load_workbook
from openpyxl.utils import get_column_letter

def greedysolve(muchan, pqall, descenddict, get_row_number, season):
    _ground = copy.deepcopy(ground) # 深拷贝地块信息，防止修改原始数据
    totalprofit = 0 # 初始化总利润
    if season == 1:
        _hvwhat = copy.deepcopy(hvwhat) # 如果是第一季，拷贝上一季种植情况
    else:
        _hvwhat = copy.deepcopy(finalhvwhat) # 如果是第二季，拷贝最终种植情况
    checkwhat = defaultdict(list) # 初始化检查上一季种植作物的字典
    for place, plantlist in checkhvwhat.items():
        for plant in plantlist:
            checkwhat[place].append(plant[0]) # 为每块地存储上一季种植的作物编号
    newhvwhat = defaultdict(list) # 初始化存储本季种植作物的字典
    while len(pqall) > 0: # 当优先队列还有作物待种植时
        profit, pldi, grid = pqall[0] # 从优先队列中取出利润最高的作物和地块
        if maxsold[pldi] / muchan[(pldi, grid)] <= limitminarea[grid]:
            heapq.heappop(pqall) # 如果作物的销售量不足以种植最小面积，跳过
            continue
        profit = -profit # 将负利润转换为正利润
        planted = False # 标记是否成功种植
        for i in range(len(_ground[grid])): # 遍历每个地块
            if maxsold[pldi] / muchan[(pldi, grid)] <= limitminarea[grid]:
                heapq.heappop(pqall) # 如果仍不满足条件，跳过
                break
            place, area = _ground[grid][i] # 获取地块和面积信息
            if hvpea[place] >= 2 and pldi not in pealist: # 检查豆类种植约束
                continue
            if pldi in checkwhat[place]: # 检查是否与上一季种植作物冲突

```

```

        continue
    if area <= limitminarea[grid]: # 如果面积小于最小种植面积, 跳过
        continue
    planted = True # 标记成功种植
    # 贪心, 尽量种最多
    plantarea = min(area, maxsold[pldi] / muchan[(pldi, grid)]) # 计算实
际种植面积
    totalprofit += plantarea * profit # 更新总利润
    _ground[grid][i][1] -= plantarea # 更新剩余地块面积
    maxsold[pldi] -= plantarea * muchan[(pldi, grid)] # 更新该作物的剩余销
售量
    newhvwhat[place].append([pldi, plantarea]) # 记录种植情况
    if not planted:
        heapq.heappop(pqall) # 如果没有成功种植, 跳过该作物
        continue
    # 更新豆类约束
    if season == 1:
        for place, plantlist in newhvwhat.items():
            p = 0
            for pldi, plantarea in plantlist:
                if pldi in pealist:
                    p = 1
                    break
            season1pea[place] = p # 更新第一季豆类种植情况
    else:
        for place, plantlist in newhvwhat.items():
            p = 0
            for pldi, plantarea in plantlist:
                if pldi in pealist:
                    p = 1
                    break
            season2pea[place] = p # 更新第二季豆类种植情况
    for place in hvpea.keys():
        if season1pea[place] or season2pea[place]:
            hvpea[place] = 0 # 如果有一季种豆, 重置计数
        else:
            hvpea[place] += 1 # 否则, 增加豆类计数
    # 将当前季节的种植信息更新到最终记录中
    for place in newhvwhat: # 遍历当前季节种植的新作物信息
        finalhvwhat[place] += newhvwhat[place] # 将当前季节的种植信息加入到最终种植记
录中
    checkhvwhat[place] = newhvwhat[place] # 更新检查字典, 记录本季种植的作物信息,
用于下一季的种植约束检查
    # 将种植结果写入Excel文件
    sheetname = str(year + 1)
    sheet = workbook[sheetname]
    for place, plantlist in newhvwhat.items():

```

```

for pldi, plantarea in plantlist:
    col = get_column_letter(pldi + 2) # 获取列字母
    row = get_row_number[place] # 获取行号
    cell_address = f'{col}{row}'
    sheet[cell_address] = plantarea # 将种植面积写入Excel单元格
return totalprofit # 返回总利润

```

附录 G 作物的价格弹性

表 1 作物及其价格弹性

作物编号	作物名称	价格弹性
1	黄豆	0.3
2	黑豆	0.3
3	红豆	0.3
4	绿豆	0.3
5	爬豆	0.4
6	小麦	0.2
7	玉米	0.2
8	谷子	0.2
9	高粱	0.2
10	黍子	0.2
11	荞麦	0.3
12	南瓜	0.4
13	红薯	0.3
14	莜麦	0.3
15	大麦	0.2
16	水稻	0.1
17	豇豆	0.5
18	刀豆	0.5
19	芸豆	0.4
20	土豆	0.3
21	西红柿	0.7
22	茄子	0.6
23	菠菜	0.5

续下页

作物编号	作物名称	价格弹性
24	青椒	0.6
25	菜花	0.5
26	包菜	0.5
27	油麦菜	0.6
28	小青菜	0.5
29	黄瓜	0.7
30	生菜	0.6
31	辣椒	0.7
32	空心菜	0.6
33	黄心菜	0.5
34	芹菜	0.5
35	大白菜	0.4
36	白萝卜	0.4
37	红萝卜	0.4
38	榆黄菇	0.8
39	香菇	0.7
40	白灵菇	0.8
41	羊肚菌	0.9

附录H 间作对作物产量的影响

表2 间作物及其产量提升量

间作物	间作物产量提升量（等级）
玉米	3
小麦	3
高粱	3
谷子	3
红薯	2
大麦	2
黍子	2
南瓜	1

续下页

间作物	间作物产量提升量(等级)
荞麦	1
莜麦	1
土豆	3
西红柿	3
茄子	3
辣椒	3
菜花	3
包菜	3
菠菜	2
油麦菜	2
小青菜	2
生菜	2
芹菜	2
黄瓜	2
空心菜	1
黄心菜	1

附录 I 支撑材料列表:

1_1代码及相关文件

2023

```
1_1.json
1_2.json
2.json
3.json
4.json
5.json
get_row_number1.json
get_row_number2.json
result.xlsx
script1.py
```

1_2代码及相关文件

2023

```
1_1.json
1_2.json
2.json
3.json
4.json
```

5.json
get_row_number1.json
get_row_number2.json
result.xlsx
script2.py
2题代码及相关文件
2023
1_1.json
1_2.json
2.json
3.json
4.json
5.json
get_row_number1.json
get_row_number2.json
result.xlsx
script.py
3题代码及相关文件
2023
1_1.json
1_2.json
2.json
3.json
4.json
5.json
elasticity.json
get_row_number1.json
get_row_number2.json
jzliang.json
jzshu.json
result.xlsx
script.py
数据处理文件
地块类型编号-地块.xlsx
二季作物.xlsx
各地块数据.xlsx
各地块在上一季种植作物.xlsx
价格弹性.xlsx
粮食类间作.xlsx
蔬菜类间作.xlsx
一季作物.xlsx
预期销售量.xlsx
作物编号-作物.xlsx
result1_1.xlsx
result1_2.xlsx
result2.xlsx