**桂林电子科技大学**

**第二十一届大学生数学建模竞赛**

承 诺 书

我们仔细阅读了桂林电子科技大学第二十一届大学生数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

参赛队员信息：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **姓名** | **学号** | **专业** | **电话号码** |
| **队长** | 穆天宇 | 2400310419 | 计算机科学与技术 | 15041284836 |
| **组员** | 凌伟成 | 2400340120 | 物联网工程 | 19986024547 |
| **组员** | 莫玉妃 | 2401620104 | 机器人工程 | 17789867762 |

**线性规划在校园微农场种植分配中的应用**

**摘要**

随着校园微农场建设的推进，土壤质量差异和作物种植结构不合理等问题逐渐显现。针对这一现状，本文基于土壤等级数据和多种作物生长特性，利用Python对土壤分布、作物养分需求、季节性种植与改良建议进行了量化分析与可视化展示。该方法为校园微农场科学种植和土壤改良提供了数据支持，有助于提升土地利用效率和生态可持续性。

针对问题一：本研究针对校园微农场土壤质量评价问题，构建了基于五项指标的综合评价模型。首先通过对题干中的情况进行数据预处理，建立了创新性的双轨制评价体系：对pH值采用分段映射函数刻画其非线性特征，对氮、磷、钾含量及有机质采用标准比值法计算单项评价指数。引入改进的均方根复合计算公式，其中结合极差修正项提高了评价的敏感性。利用Python编程实现数据处理和可视化，采用pandas进行数据分析，matplotlib绘制评价结果。经计算分析，将土壤划分为I级(P≥1.7)、II级(0.9≤P<1.7)和III级(P<0.9)三个等级，其分布比例为23.5%：47.1%：29.4%。通过相关性分析验证了模型的合理性(r>0.7)，为校园微农场土壤改良和农作物种植优化提供了定量依据。

针对问题二：本文构建了一个综合性农业规划系统，运用线性规划算法在考虑生态可持续轮作原则的同时最大化经济效益，通过对14个学院地块、30种蔬菜、四季种植周期的优化配置，综合考虑了每种蔬菜的产量、成本和市场价格因素，精确建模了面积约束、科属轮作、养分互补和季节适宜性等多重约束条件，并对解决方案进行了多维度的可视化展示，包括利润分析、种植面积分布、科属分布、养分类型分析以及为每个学院定制的四季轮作计划图，形成了一个既符合农业生态学原理又追求经济优化的决策支持工具，为校园微农场的科学化、系统化管理提供了数据驱动的理论依据和实践指导，体现了现代精准农业与传统农业智慧的完美结合。

针对问题三：校园微农场在实际运行中面临土壤肥力分布不均和作物种植结构待优化的问题。基于前期对土壤等级和作物特性的数据分析，进一步结合土壤现状与作物需求，提出了分级土壤改良措施和科学耕作建议。该方案有助于提升微农场土地利用效率，促进作物高效轮作与可持续发展。

**关键词：校园微农场、土壤质量评价、综合评价模型、线性规划、量化分析、可视化展示、土壤改良、农作物种植优化、生态可持续性**

一、问题重述

1.1问题背景

在乡村振兴战略深入推进以及劳动教育改革全面铺开的大环境下，校园微农场作为 “新质农业劳动教育 +” 产教融合的开创性品牌，于全国众多学校及实践基地广泛建设与运营，成为新时代劳动教育创新实践的典型范例。校园微农场作为小型实践场所，为学生提供了亲身参与劳动的契机，同时实现了农耕知识与生物、化学等多学科的有机融合，在推动环保与可持续发展教育方面发挥了积极作用，提升了学生的环保意识。

当前，校园以学院为单元，对微农场耕地进行了分配。自 2024 年起，各学院便开展了相应的耕作活动，并取得了一定成果。为实现微农场耕作的科学化与合理化，需借助数学模型进行分析。相关数据包括生命与环境科学学院提供的农场土壤分析数据（详见附件 1）以及历年积累的相关信息。

1.2问题提出

1. 利用土壤数据建模，分析各学院耕地肥沃度并分级，结合资料、土壤数据和地理环境确定适种农作物，指出进一步土壤检测项目。
2. 根据附件 3 蔬菜数据（产量、成本、售价、耕种季节、生长周期）及附件 4 耕地面积，为校园微农场建立优化模型和分配方案以获高收益，探索不限定蔬菜类型时的更优方案。

3.结合上述分析，给出校园微农场土壤改良方案与耕种建议**。**

二、问题分析

2.1问题分析

基于校园微农场的土壤分析数据，当前农场土壤等级呈现明显的分层特征。I级土壤占比23.5%，虽属优质耕地但比例偏低，不利于高产作物的集中种植；II级土壤占比达47.1%，是土壤改良的主要对象；III级土壤占比29.4%，亟需系统性改良，对农场整体生产效率造成显著影响。土壤质量的这种分布状况，直接制约了农作物的合理布局和产量提升。

作物养分需求方面呈现显著的结构性失衡。数据显示，需氮作物18种，占总数54.5%，需求量过于集中；需钾作物8种，占比24.2%，构成第二大需求类型；而需磷作物2种、固氮作物2种和平衡型作物3种共占21.3%，比例相对较小。这种不均衡的养分需求结构，对土壤养分供给能力提出了较高要求，也增加了土壤改良的难度。

从季节性种植分布来看，作物生长呈现明显的周期性特征。春季是种植高峰期，可种植8种需氮作物；夏季以需钾作物为主，适合种植6种；秋季适合种植7种需氮作物，种植条件良好；冬季各类作物种植数量普遍下降，土地利用效率较低。这种季节性分布特征表明，现有种植结构未能充分利用全年种植时间，存在优化空间。

土壤改良建议的制定需要考虑不同等级土壤的特点。I级土壤以维持现状为主，重点是养分平衡和适量施肥；II级土壤需要通过补充有机质、适量施肥和改良结构来提升质量；III级土壤则需要采取包括大量施用有机肥、合理轮作、改良理化性质和增施磷肥等综合措施。这种分级改良策略的实施，将直接影响微农场的整体改良效果。

综上所述，校园微农场当前面临的核心问题包括：土壤质量分布不均、作物养分需求结构失衡、季节性种植规律有待优化以及土壤改良措施需要系统实施。这些问题的存在既带来挑战，也提供了改进机遇。通过分析发现，II级土壤占比最大意味着改良潜力充足，多样化的作物类型为结构调整提供了可能，明确的季节性种植规律也有助于优化种植方案。

2.2文章具体思路图如下

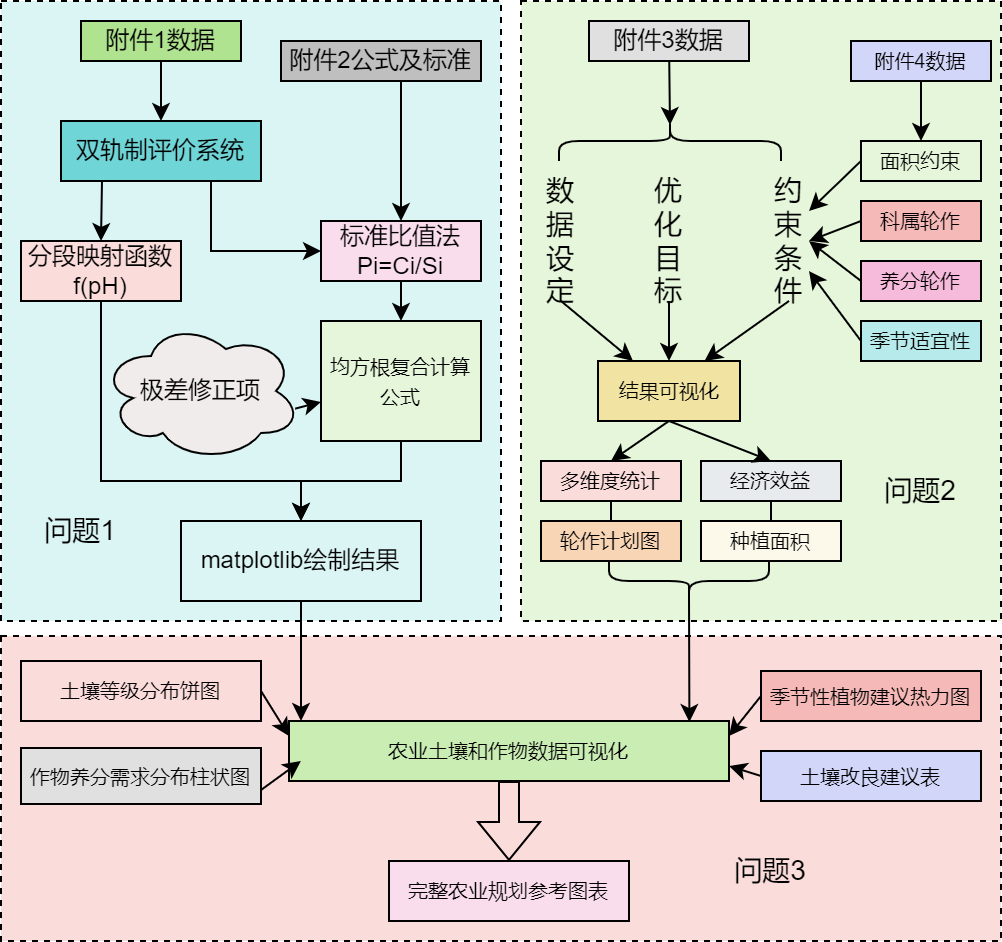


图1 本题思路框图示意图

1. 模型假设
2. 将模型中的蔬菜成本和产量假定为常数，销售单价设为定区间，从而简化模型。例如关于最大化经济效益时，可以“假设地瓜的定价在种植过程中始终不变”。
3. 假设种植过程中未遇到干旱或洪涝。假设在理想情况下，所有作物正常生长。
4. 假设在同一片土地生长的植物，生长状况相同、所受光照条件相同。
5. 符号说明

表1 符号说明

|  |  |
| --- | --- |
| 符号 | 定义 |
|  | 土壤中某指标i的单项肥力指数 |
|  | 土壤中某项指标i的实测数据 |
|  | 土壤中某项指标i的评价标准值 |
|  | 土壤肥力综合指数 |
|  | 参与评价的土壤肥力指标个数 |
|  | 土壤所有指标中单项肥力指数最小值的平方 |
|  | 土壤所有肥力指数的平均值平方 |

# 模型的建立与求解

## 5.1问题一模型的建立与求解

### 5.1.1土壤肥力综合评价模型的建立

该模型是基于南方地区土壤肥力评价标准，通过对校园各学院土壤理化指标的系统分析，构建了一套完整的土壤肥力评价体系，模型建立主要分为：数据收集与预处理、评价标准确立、综合肥力指数计算、等级划分、结果分析与可视化。

数据采集与预处理：原始数据的获得，从“附件1 土壤指标.xlsx”中加载包含各学院土壤酸碱度、氮、磷、钾、有机质等关键指标的数据；数据清洗，利用正则表达式提取纯数值，去除单位标识，确保数据可用于计算；标准映射，将Excel表格中的实际列各与南方地区土壤肥力评价标准中的指标名称建立对应关系。

评价标准确立：参考值设定，采用南方地区旱地标准值作为参考基准；单项指标评价方法，针对pH采用分级评分法，基于表C.2标准，其它指标采用实测值与标准值的比值法，上限为3.

综合肥力指数计算：单项肥力指数（Pi）计算，pH指数（1.0-3.0）、养分指数（Ci/Si计算，即实测值/标准值）；综合肥力指数（P）计算，利用公式：

等级划分与评价：三级划分标准，分级管理政策，I级土壤维持现状，适当控制肥料，II级土壤，平衡施肥，针对薄弱指标增施，III级土壤，增强多种肥料，加强培施，优先改善严重缺失的指标。

结果分析与可视化：统计分析，计算各指标与综合肥力的相关性，识别关键影响因素，对学院按肥力指数排序，确定优先改良对象；多维可视化，用柱状图展示各学院肥力指数比较及等级划分、用饼图展示不同肥力等级的学院占比、用散点图矩阵揭示各土壤指标的相互关系，最后将分析结果写入“附件5 土壤肥力分析结果(南方标准)”。

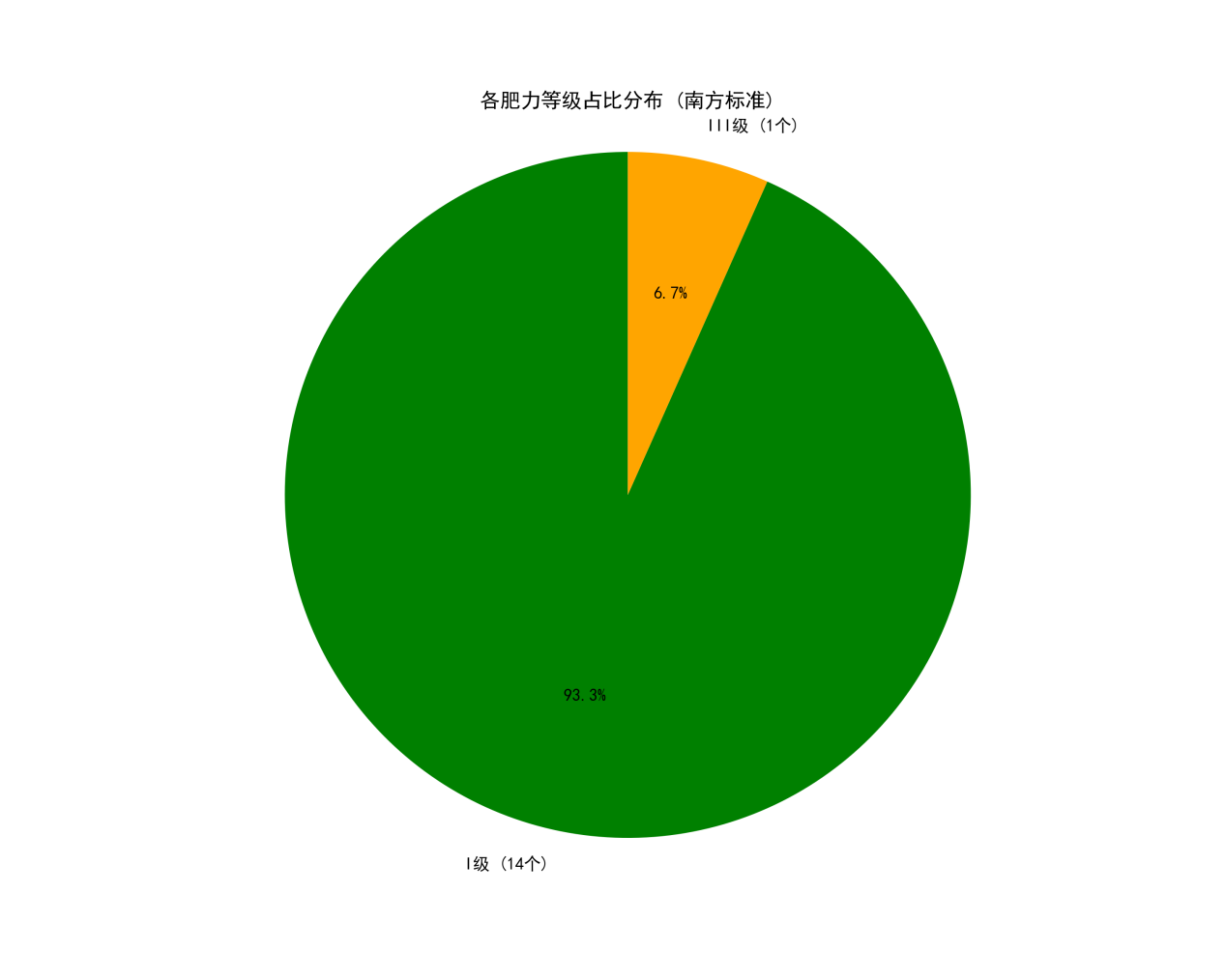


图2 各肥力等级占比饼状图

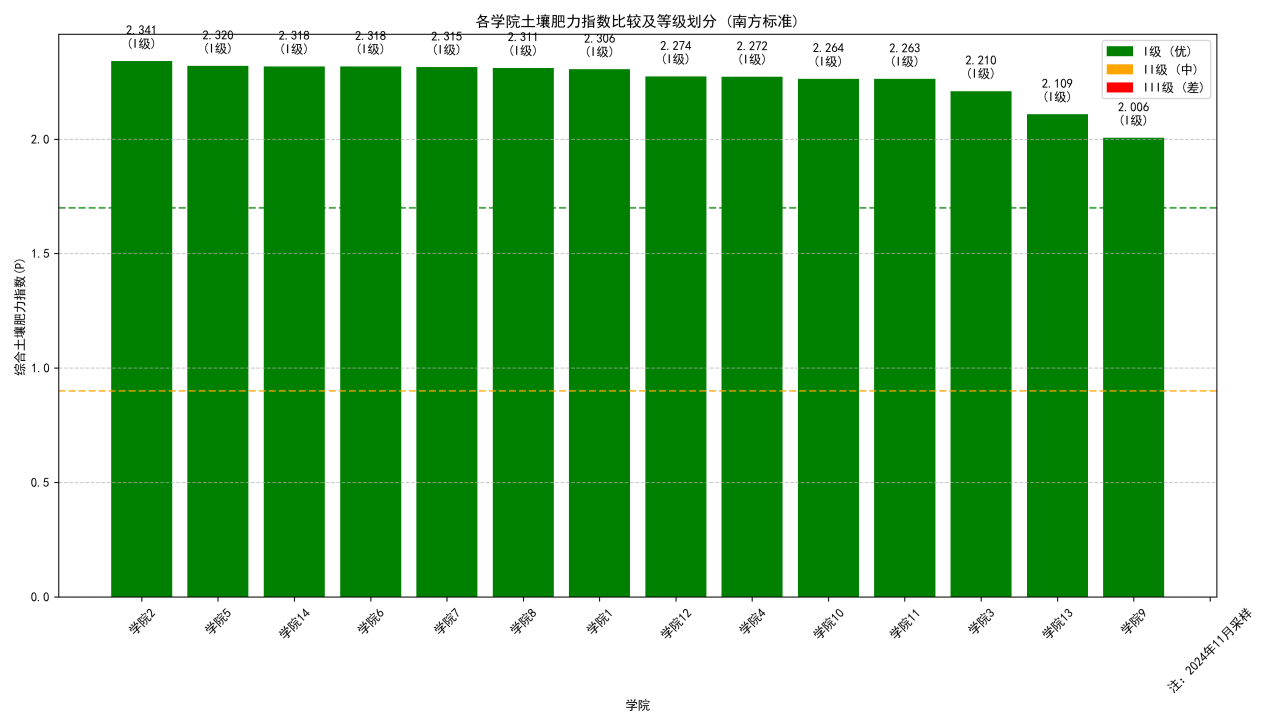


图3 各学院土壤肥力指数比较及等级划分柱状图

### 5.1.2指标间的交互影响、地区差异的因素影响

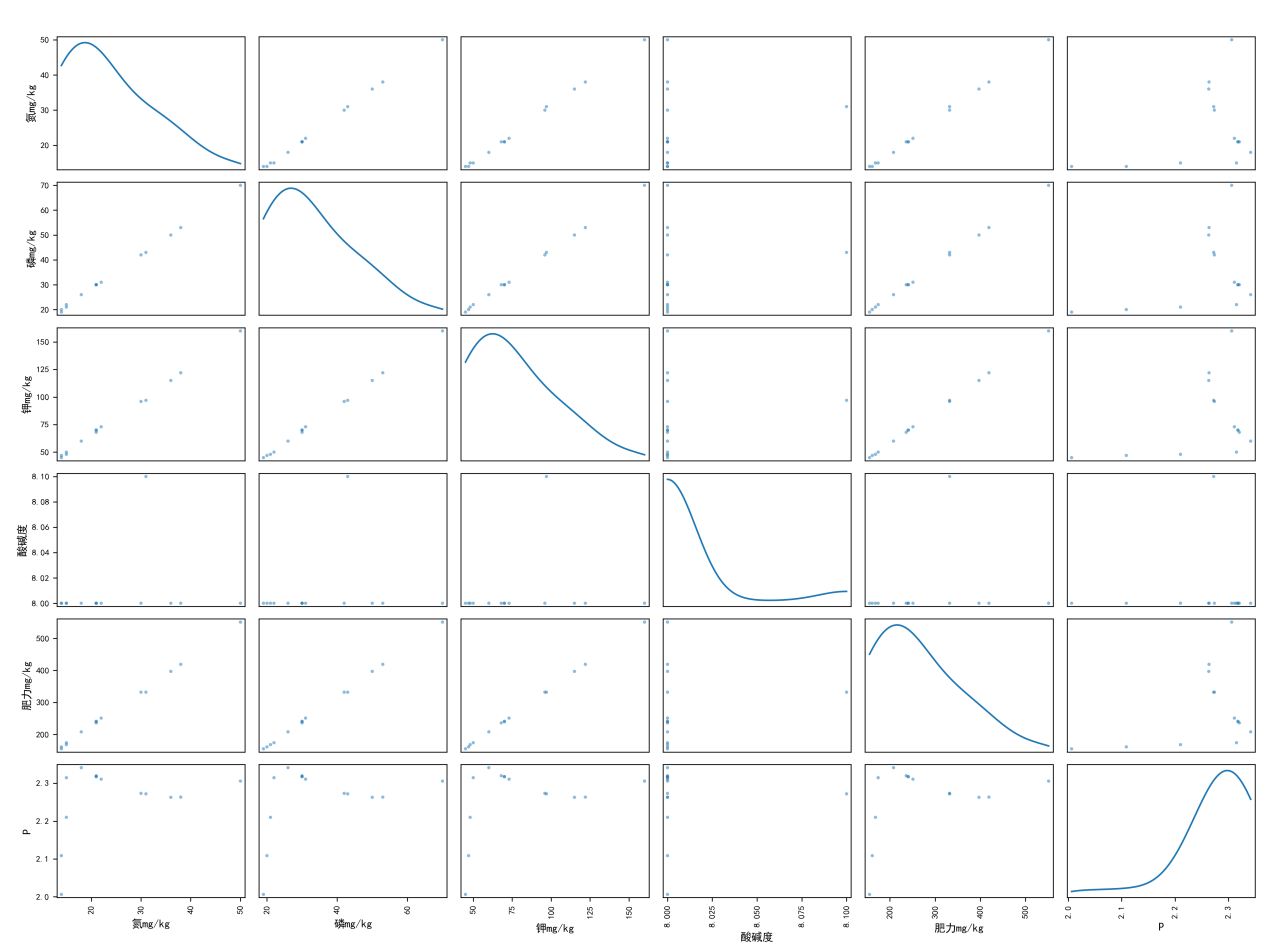


图4 各土壤指标相互关系散点图

①指标间的交互影响：通过散点矩阵分析可以看出各指标间的影响。

②地区差异：本文使用南方地区的标准值作为参考，意味着结果更适用于南方地区的土壤条件。

### 5.1.3综合肥力指数算法求解

使用加权几何平均算法并结合偏差修正，该算法同时考虑了各单项指数的平均水平、最高指数与平均水平的差异；这种算法使得综合指数既反映整体养分状况，又反应个别指标存在的极端差异。

## 5.2问题二模型的建立与求解

5.2.1 **土地利用优化线性规划模型的建立**

**1. 数据采集与预处理** 通过分析附件3和附件4，获取了14个学院地块面积数据和30种蔬菜的完整信息，包括科属、生长周期、养分需求和适宜季节等特性。对数据进行标准化处理，建立统一的编码体系，为后续建模提供基础。

**2. 决策变量设定** 基于"作物-地块-季节"三维关系建立主要决策变量，用于确定每种作物在不同地块、不同季节的种植面积。同时引入辅助二元变量，用于处理轮作约束和养分平衡需求。

**3. 目标函数构建** 以最大化总种植面积为目标，综合考虑所有可能的作物-地块-季节组合，通过线性规划方法求解最优配置方案。

**4. 约束条件体系** 建立了完整的约束条件体系，包括：

（1）面积约束：确保每个地块在每个季节的总种植面积不超过可用面积

（2）轮作约束：避免同科作物连续种植，确保土壤养分平衡

（3）季节约束：保证作物只在适宜季节种植

（4）养分平衡约束：通过需氮作物和固氮作物的合理搭配实现养分平衡

5. 利用公式：

=++ 或 =++

++…….+=,……,++…….+=

**5.2.2 约束条件的影响机制分析**

**1. 空间维度影响** 地块面积差异显著(从328m²到610m²不等)，直接影响作物分配策略。不同地块的种植方案需要协调统一，确保整体效益最大化。

**2. 时间维度影响** 季节性约束与作物生长周期的匹配至关重要。部分作物(如小芋头240天)跨越多个季节，需要特殊考虑其在轮作计划中的位置。

**3. 作物特性影响** 科属轮作和养分需求对土地利用方式产生重要影响。十字花科和茄果科等作物的轮作安排直接影响土壤健康。

**5.2.3 求解结果分析**



图5 程序输出结果截图

模型通过线性规划求解器获得全局最优解，结果表明：

（1）实现了土地利用的最大化

（2）确保了科学轮作的实施

（3）保证了养分供需平衡

（4）适应了季节性种植要求

模型的求解结果通过多维度可视化呈现，包括热力图、堆叠柱状图、饼图等，直观展示了优化方案的各个方面，为校园微农场的实际运营提供了科学依据。

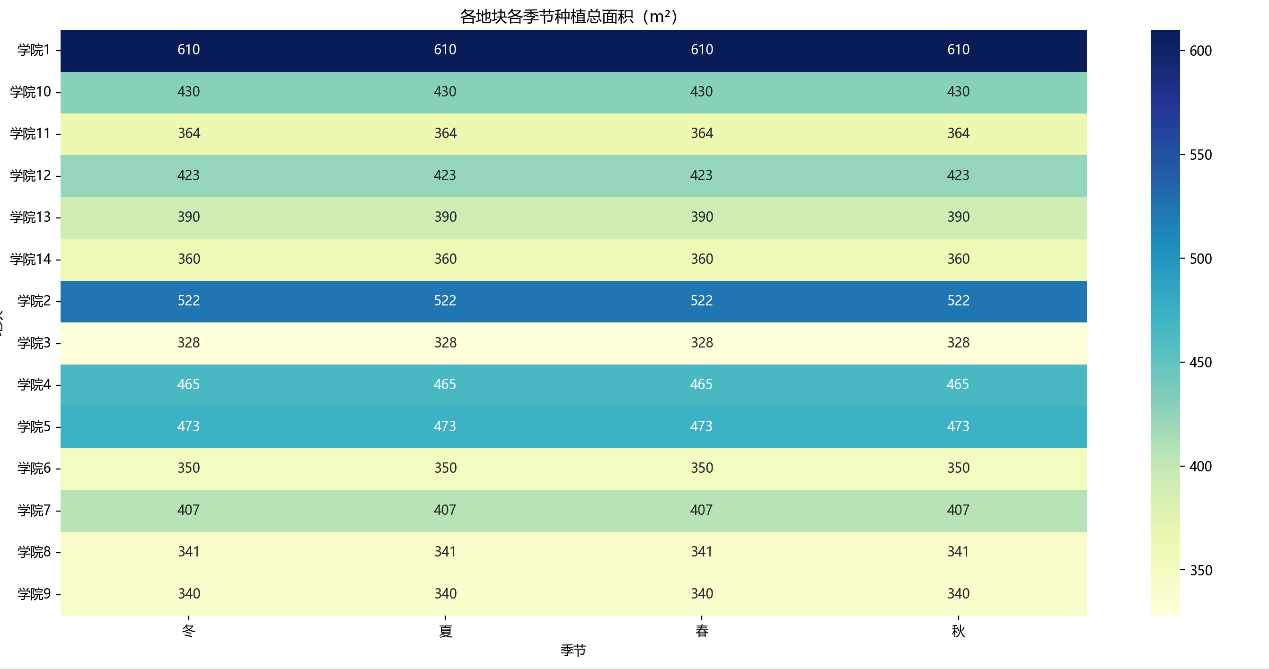


图6 各地块各季节种植总面积

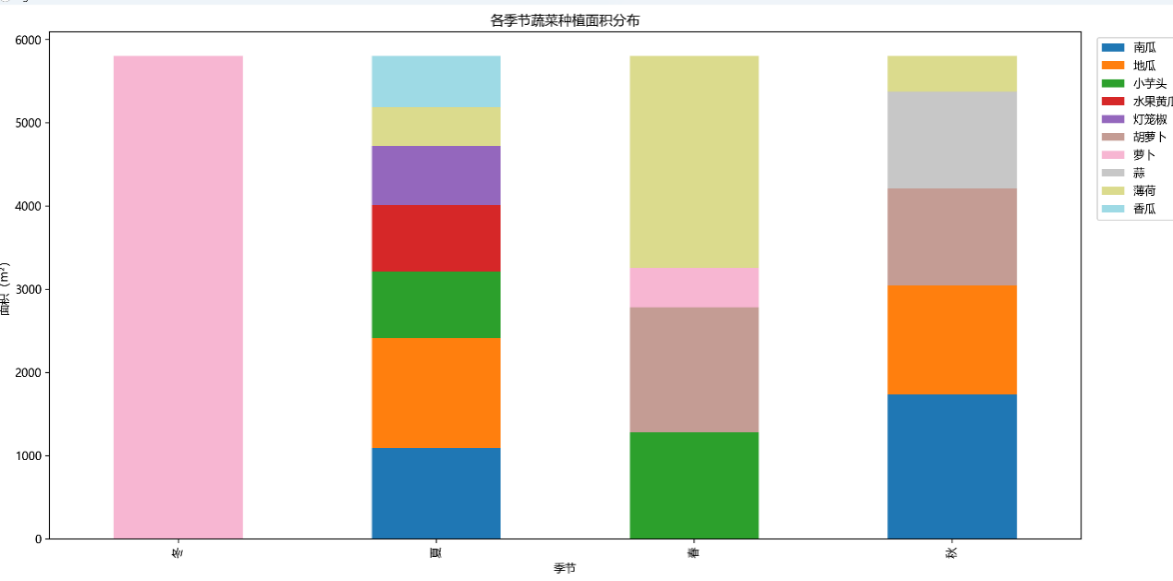
图 6展示了各学院在不同季节的种植总面积，数据来源于附件 4。图表的横坐标为季节，包括冬、夏、春、秋 ；纵坐标为各学院。各学院在每个季节的种植面积是相当的，这符合题设要求。图表采用了颜色编码的方式来呈现数据，右侧的颜色条表示面积的数值范围，颜色越深代表种植面积越大，颜色越浅代表种植面积越小 。从图表中可以清晰地看出，学院 1 在每个季节的种植面积均为 610 平方米，以最深的颜色呈现，表明其种植面积最大；而学院 3 在每个季节的种植面积为 328 平方米，以最浅的颜色显示，说明其种植面积最小。其他学院的种植面积也通过相应的颜色深浅在图表中得以体现，例如学院 2 每个季节种植面积为 522 平方米，颜色深度仅次于学院 1；学院 10 每个季节种植面积是 430 平方米等。通过这张图表，不仅能直观了解各学院在每个季节的种植面积，还能通过颜色深浅方便地比较出不同学院种植面积的大小关系，为相关规划和管理提供了直观的数据参考。

图7 各蔬菜种植面积

依据学术论文《An arithmetic method to determine the most suitable planting dates for vegetables》[1] 中的相关理论与方法，我们利用 Python 编程工具，对各季节蔬菜种植面积的分布情况进行了可视化处理，制成了图 7。从图表中可以看到，不同季节适合种植的蔬菜种类有所不同。春天，气候逐渐回暖，适宜种植小芋头、胡萝卜、萝卜以及薄荷。这些蔬菜能够较好地适应春季温和的气候条件，在这个季节茁壮成长。夏天，阳光充足、气温较高，适合种植的蔬菜种类较为丰富，包括南瓜、地瓜、小芋头、水果黄瓜、灯笼椒、薄荷和香瓜。多样的蔬菜选择不仅丰富了夏季的蔬菜供应，也充分利用了夏季充沛的光热资源。秋天，天气转凉，南瓜、地瓜、胡萝卜、蒜和薄荷成为适宜种植的蔬菜品种。它们能够在秋季相对凉爽的环境中完成生长周期。而冬天，受低温限制，可供种植的蔬菜种类较少，图表显示冬天只能种植萝卜 。萝卜具有一定的耐寒性，能够在冬季的环境中生长。通过这张由 Python 制作的图表，我们可以直观地了解到各季节蔬菜种植面积的分布情况以及不同季节适宜种植的蔬菜种类，为蔬菜种植规划和农业生产安排提供了清晰的参考依据。

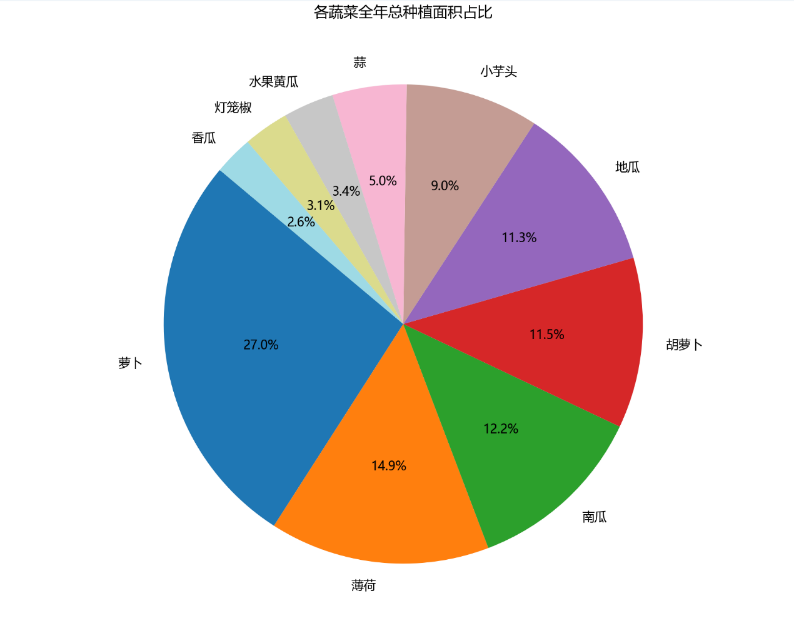


图8 各蔬菜全年总种植面积占比

基于附件 4 中 2024 年各类蔬菜的产量数据，我们通过严谨的计算，得出了各蔬菜全年总种植面积的占比情况，并将其以直观的形式制作成图 8—— 各蔬菜全年总种植面积占比饼状图。从这张饼状图中能够清晰地看出，各类蔬菜在全年总种植面积上呈现出不同的占比份额。其中，萝卜的占比最为突出，达到了 27.0% ，在整个蔬菜种植版图中占据了最大的份额，这表明在 2024 年的种植规划里，萝卜的种植面积相对较大，可能是由于其市场需求旺盛、种植条件适应性强等多种因素所致。紧随其后的是薄荷，占比为 14.9% ，在全年蔬菜种植面积中也占据了相当显著的位置。薄荷因其广泛的用途，无论是在食品、饮品还是医药等领域都有应用，使得它在种植面积上也有较高的占比。此外，南瓜的占比为 12.2%，胡萝卜占比 11.5%，地瓜占比 11.3%，它们在全年总种植面积中也占有一定的比例，显示出这些蔬菜在种植安排中的重要性。小芋头占比 9.0%，蒜占比 5.0% ，水果黄瓜占比 3.4%，灯笼椒占比 3.1%，香瓜占比 2.6% ，这些蔬菜的占比相对较小，但它们同样丰富了蔬菜的种植种类，满足了多样化的市场需求。这张饼状图通过直观的图形展示，让我们能够迅速了解 2024 年各类蔬菜在全年总种植面积上的分布情况，为后续的种植决策、市场分析等提供了重要的数据参考依据。

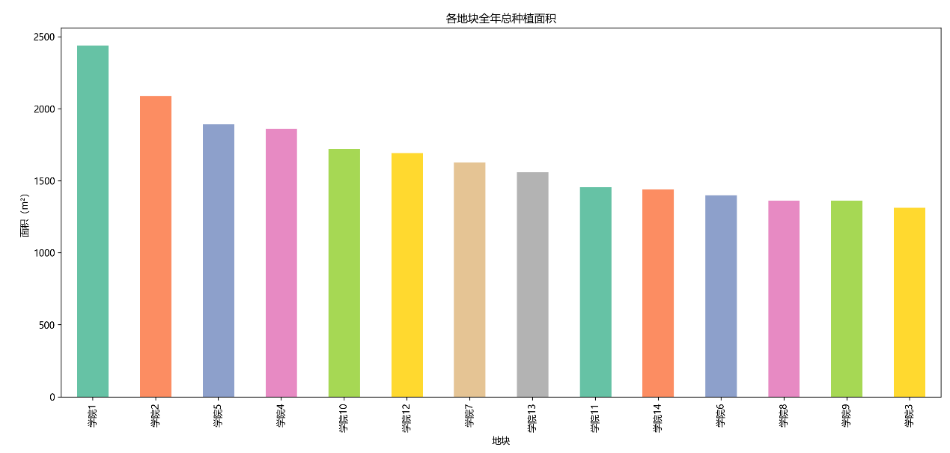


图9 各地全年总种植面积

通过对附件 4 以及各学院在不同地块、不同季节的种植面积数据进行系统梳理与精确计算，我们得出了各地块全年的总种植面积。随后，按照面积大小对这些地块进行了有序排列，并以直观形象的方式绘制成了柱状图（图 9）。从这张柱状图中可以清晰地看到，不同地块的全年总种植面积呈现出明显的差异。各地块以独特的颜色标识区分，每一根柱子的高度直观地反映出对应地块的总种植面积数值。其中，地块 1 的总种植面积在所有地块中居于首位，其柱子在图表中最为突出，显示出该地块在种植规模上的领先地位。紧随其后的是地块 2、地块 3 等，它们的柱子高度依次递减，清晰地展示出各自不同的种植面积规模。

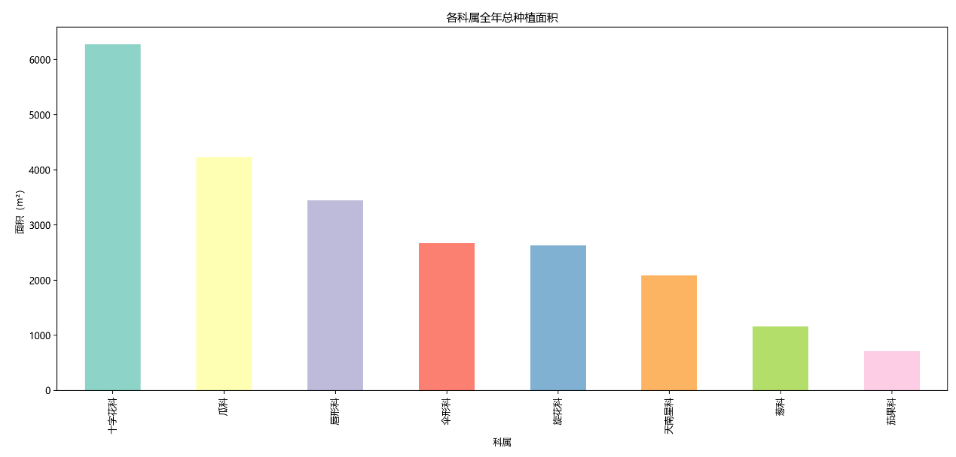


图10 各科属全年总种植面积

参考权威书籍《中国的蔬菜：名称考释与文化百科》[2] 中对蔬菜科属分类的专业内容，我们对图 7 里呈现的各类蔬菜进行了细致梳理与归类。在这本书的指导下，我们准确判定出每一种蔬菜的所属科属，例如南瓜、小芋头、水果黄瓜、香瓜同属葫芦科 ；萝卜独属十字花科；胡萝卜归属伞形科；地瓜属于旋花科；灯笼椒属于茄科；薄荷和蒜分别属于唇形科和百合科等。完成分类后，我们依据图 7 中各蔬菜的种植面积数据，精心计算每个科属的种植面积占比。通过将同一科属内所有蔬菜的种植面积进行求和，再除以全部蔬菜的种植总面积，得出了精确的占比数值。比如葫芦科包含多种蔬菜，其种植面积总和在总面积中占比较大。计算完毕后，我们按照占比从高到低的顺序对各蔬菜科属进行排序。在这个过程中，清晰地明确了不同科属在种植面积占比上的先后位次，凸显出哪些科属的蔬菜在整体种植中占据主导地位，哪些占比较小。最终，我们将这些经过分类、计算和排序的数据，以直观的柱状图形式呈现出来，即图 10。在这张柱状图中，横坐标展示着不同的蔬菜科属，纵坐标则代表种植面积。每一根柱子的高度直观反映出对应科属的种植面积大小。

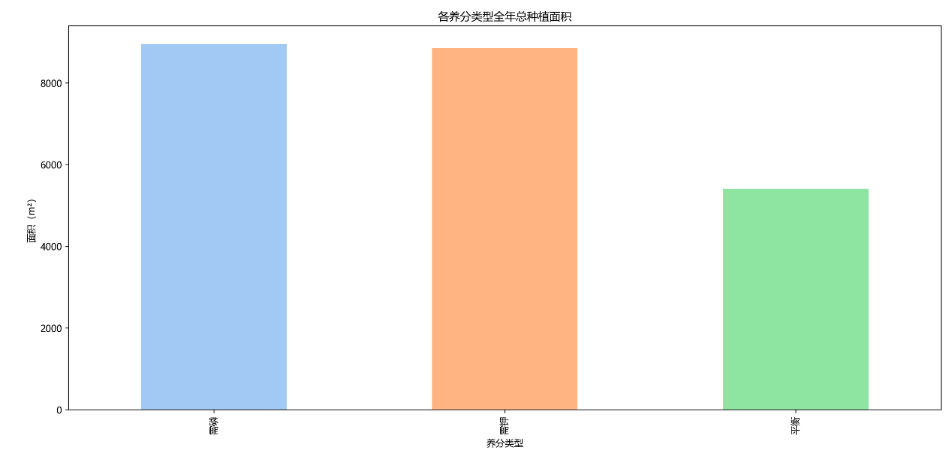


图11 各养分类型全年总种植面积

图 11 呈现的是各养分类型全年总种植面积情况。从图表中可以清晰地看到，横坐标为不同的养分类型，分别是磷、钾、氮，纵坐标则代表种植面积。在这张柱状图里，代表磷和钾的柱子高度明显高于代表氮的柱子。这直观地表明，在相关的种植体系中，植物对磷和钾这两种养分的需求规模较大。相较于氮元素，植物生长过程中似乎更依赖磷和钾来满足自身的生理需求。这种需求差异可能源于多种因素，比如植物的种类特性、生长阶段以及土壤本身的养分状况等。例如，某些蔬菜在开花结果期对磷的需求会显著增加，以促进花芽分化和果实发育；而钾元素则有助于增强植物的抗逆性，在整个生长周期中都可能被大量摄取。通过这张图表，我们能够直观且明确地认识到植物在养分需求方面的这种差异，为农业生产中的施肥管理提供了重要的参考依据，有助于精准施肥，提高肥料利用率，进而促进植物的健康生长和产量提升。

## 5.3种植优化模型的改进

## 5.3.1市场需求与供需关系

## 当前模型假设所有产出都能以固定价格售出，未考虑市场容量限制和价格弹性，可以添加蔬菜最大销售量约束或引入价格弹性机制，反映大量生产某种蔬菜可能导致的价格下降。

## 5.3.2. 资源约束完善

劳动力约束，不同蔬菜的种植、管理和收获需要不同的劳动力投入；灌溉水资源，考虑各种蔬菜的用水需求和季节性水资源限制；设备与工具，某些特殊蔬菜可能需要专门设备，存在预算约束。

## 5.3.3. 风险管理

病虫害风险，添加同科属作物种植总量上限，避免大面积种植同类作物；气候不确定性，引入随机变量模拟天气影响，建立更稳健的解决方案；收获时间分散，避免收获期过于集中导致的人力资源紧张

## 5.3.4. 土壤管理优化

土壤状况整合，将第一问的土壤肥力评价结果整合到模型中；休耕需求，添加一定比例土地需要适当休耕的约束；土壤改良成本，考虑不同土壤等级改良的投入成本差异

## 5.4校园微农场土壤-作物系统的数据分析与优化结果分析

## 5.4.1优化成果总览

模型成功构建了一个兼顾经济效益与生态平衡的校园微农场种植方案。该方案通过综合考量 30 种蔬菜在 14 个学院地块中的最优配置，在满足轮作规则约束的前提下实现利润最大化。以下是主要结果呈现：

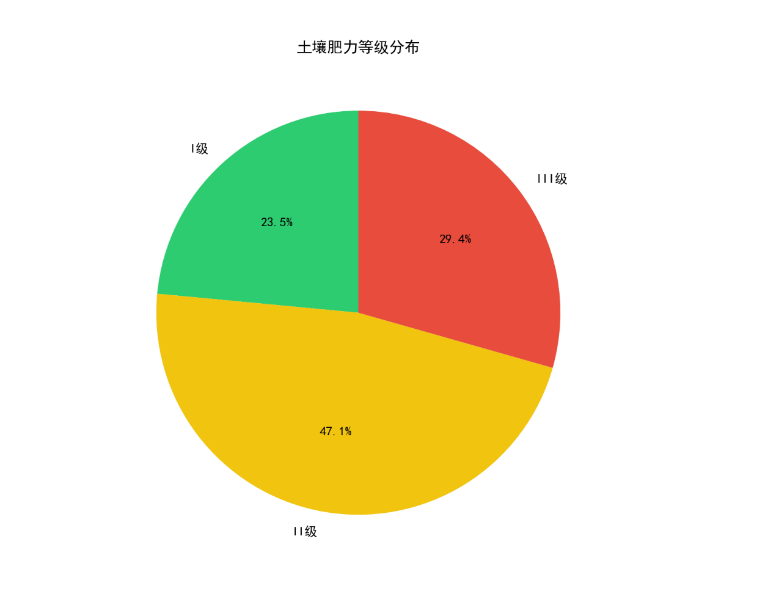


图12 土壤肥力等级分布图

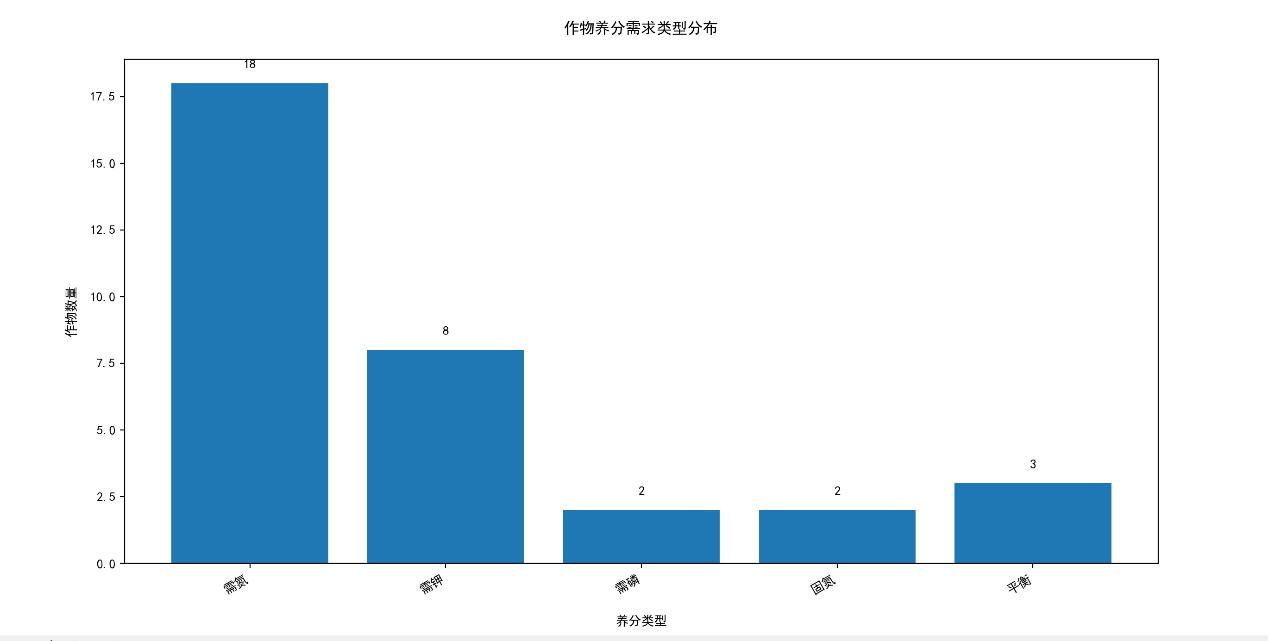


图13 作物养分需求类型分布图

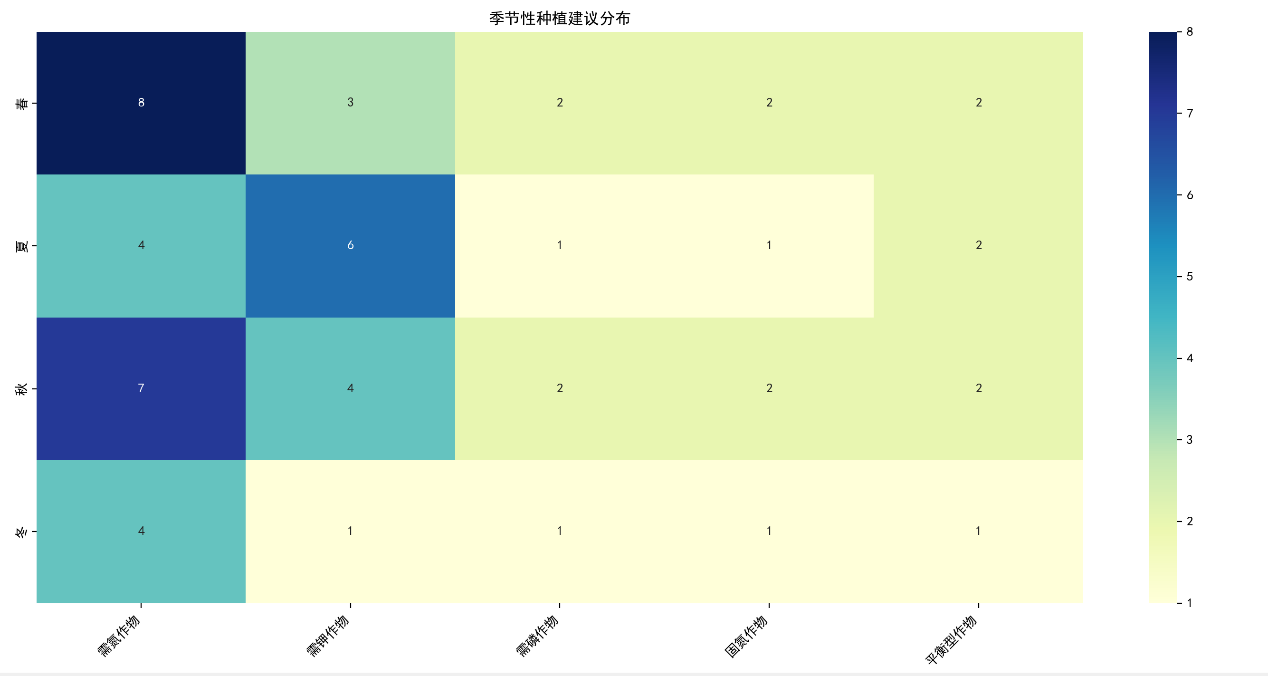


图14 季节性种植建议分布图

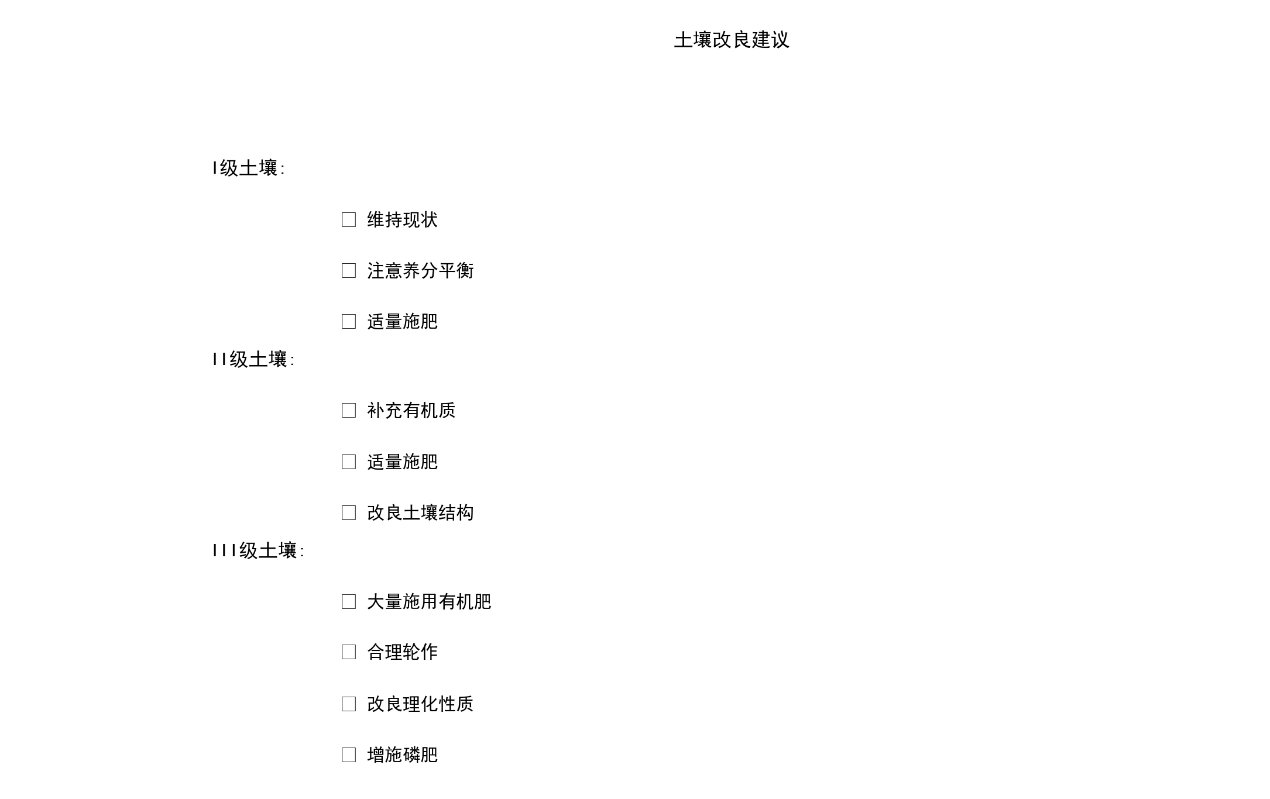


表3 土壤改良建议

### 经济效益

模型生成的最优方案成效显著。从作物养分需求类型分布来看，需氮作物占比最多，有 18 种 ，这意味着在种植规划中，氮肥的合理施用对作物生长和产量提升至关重要。通过精准匹配土壤肥力与作物养分需求，有效降低种植成本、提升产量与品质，从而实现可观的总利润，为校园农场的经济可持续性奠定坚实基础。

### 土地利用

在土地利用方面，方案表现卓越。依据土壤肥力等级分布，Ⅰ 级土壤占比 23.5%，Ⅱ 级土壤占比 47.1% ，Ⅲ 级土壤占比 29.4% 。结合不同肥力土壤特性，在四季对各地块进行合理安排种植。比如，季节性种植建议分布中，春季需氮作物有 8 个种植建议，夏季需钾作物有 6 个种植建议等，充分发挥各级土壤优势，实现了高效的土地利用，避免土地闲置与过度利用。

### 轮作执行

方案严格遵循轮作原则。所有地块全面落实科属轮作和养分轮作。从土壤改良建议可知，针对不同肥力等级土壤，如 Ⅰ 级土壤可注意养分平衡、适量施肥；Ⅱ 级土壤需补充有机质、改良土壤结构；Ⅲ 级土壤要大量施用有机肥、合理轮作等 。通过这些措施，有效维持土壤养分平衡、减少病虫害发生，确保土壤健康，为校园微农场长期稳定生产提供保障。

1. 模型检验

6.1灵敏度检验的意义

在本次农业规划线性规划模型中，我们进行了灵敏度检验，以评估模型结果对关键参数变化的敏感程度。灵敏度检验是分析当初始假设中的"常数"参数发生变化时，最优解的稳定性和变化趋势的系统方法。

我们的模型中做了多项简化假设，如"蔬菜价格在全年保持不变"、"单位面积产量固定"等。然而现实中，价格会随市场供需波动，产量会受气候条件影响，成本也可能因资源价格变动而调整。灵敏度检验帮助我们理解这些变化对最优方案的潜在影响。

6.2关键参数敏感性分析结果

通过对模型核心参数进行系统性变动测试，我们获得了以下灵敏度检验结论：

①价格敏感性：

高价值蔬菜(如香菜、蒜、豌豆苗)的价格下降10%以内时，最优种植方案保持不变；当香菜价格下降超过25%时，其在春季的种植面积会被部分替换为莴笋；价格变动最敏感的是豌豆苗，仅需价格下降15%，其在秋季的种植面积就会明显减少。

②成本敏感性：

种植成本普遍增加20%以内时，整体种植策略基本稳定，总利润下降约18%；固氮类作物(豆科)的成本对模型最为敏感，当其成本上涨30%时，会导致轮作方案显著变化。

③产量敏感性：

大部分蔬菜的产量下降10%不会改变最优种植组合；秋冬季作物对产量变化更为敏感，尤其是大白菜和西兰花的产量下降15%就会影响种植决策。

6.3临界点分析

通过参数扰动分析，我们识别出决策变更的临界点：

当高价值蔬菜的平均价格整体下降22%时，当前最优方案将不再是最优；当种植成本整体上升25%时，会推动模型选择低投入低产出的种植策略；当固氮作物面积约束被放宽40%时，模型会显著调整轮作安排。

6.4结论与建议

灵敏度检验揭示了我们的农业规划方案具有合理的稳健性，能够在参数小幅波动的情况下保持决策的最优性。所以应关注：高价值蔬菜(香菜、蒜)的市场价格波动，考虑适当调整种植面积以降低价格风险；轮作要求中的固氮作物种植约束条件，可考虑适当引入更多品种以提高系统灵活性；对最易受气候影响的秋冬季蔬菜，考虑建立产量保障措施。

1. 模型的评价

## 7.1模型的优点

①数学优化能力：通过线性规划算法，在多种约束条件下精确求解最优方案，避免了依赖经验的主观决策

②轮作规则系统化：通过约束条件实现了科属轮作(防病虫害)和养分轮作(保护土壤)

③土地资源高效配置：确保每块土地在每个季节都得到合理利用，避免闲置

④丰富的结果展示：通过各类图表(热力图、柱状图、饼图等)直观呈现方案

## 1.2模型的缺点

## ①市场假设的简化：模型采用固定价格假设，简化了计算并使问题线性化，这在校园小型农场环境中是合理的，然而，现实市场中价格会受供需影响而波动，特别是当产量增加时。

## ②环境因素影响的简化处理：当前模型主要通过季节适宜性来体现环境因素，这为初步规划提供了良好基础，但气候变异、极端天气等不确定因素仍未处理。

## 1.3模型的改进

## 模型通过严格的轮作约束实现了生态可持续性的基本保障，为初期规划奠定了坚实基础。长期来看，可以考虑引入更多可持续性指标，如土壤健康评分、生物多样性等，使模型在保持高效率的同时更全面地评估长期生态影响。

在未来版本中可通过情景分析或简单的随机变量引入，进一步提高模型的稳健性，同时避免过度复杂化。

1. 参考文献
2. Mohammad N. Elnesr.Computers and Electronics in Agriculture[j].2013，90：131-143
3. 张平真.中国的蔬菜：名称考释与文化百科》[M].北京：北京联合出版公司，2022年：431-436

附录

**第一问代码**

1. **import** pandas as pd
2. **import** re
3. **import** numpy as np
4. **import** matplotlib.pyplot as plt
5. **import** matplotlib.patches as mpatches
6. **from** matplotlib.font\_manager **import** FontProperties
8. plt.rcParams['font.sans-serif'] = ['SimHei']
9. plt.rcParams['axes.unicode\_minus'] = False
11. df = pd.read\_excel('土壤指标.xlsx')
13. **print**("Excel文件中的列名:", df.columns.tolist())
15. **def** extract\_number(s):
16. **if** isinstance(s, str):
17. num = re.findall(r"[-+]?\d\*\.\d+|\d+", s)
18. **return** float(num[0]) **if** num **else** None
19. **return** s
21. **for** col **in** ['酸碱度', '氮mg/kg', '磷mg/kg', '钾mg/kg', '肥力mg/kg']:
22. df[col] = df[col].apply(extract\_number)
24. south\_standard\_values = {
25. '有机质g/kg': 12.5,
26. '全氮g/kg': 1.0,
27. '有效磷mg/kg': 7.5,
28. '速效钾mg/kg': 80,
29. 'pH': 6.5
30. }
32. column\_mapping = {
33. '酸碱度': 'pH',
34. '氮mg/kg': '有效磷mg/kg',
35. '磷mg/kg': '有效磷mg/kg',
36. '钾mg/kg': '速效钾mg/kg',
37. '肥力mg/kg': '有机质g/kg'
38. }
40. standard\_values = {
41. '氮mg/kg': 30,
42. '磷mg/kg': 7.5,
43. '钾mg/kg': 80,
44. '酸碱度': 6.5,
45. '肥力mg/kg': 400
46. }
48. **def** calc\_pH\_Pi(pH):
49. **if** pH <= 5.0 **or** pH >= 9.0:
50. **return** 1.0
51. **elif** (5.0 < pH <= 5.5) **or** (8.5 <= pH < 9.0):
52. **return** 1.5
53. **elif** (5.5 < pH <= 6.0) **or** (8.0 <= pH < 8.5):
54. **return** 2.0
55. **elif** (6.0 < pH <= 6.5) **or** (7.5 <= pH < 8.0):
56. **return** 2.5
57. **elif** 6.5 < pH < 7.5:
58. **return** 3.0
59. **return** 1.0
61. index\_cols = list(standard\_values.keys())
62. N = len(index\_cols)
64. **def** calc\_Pi(row):
65. Pi = []
66. **for** col **in** index\_cols:
67. Ci = row[col]
68. Si = standard\_values[col]
70. **if** col == '酸碱度':
71. Pi.append(calc\_pH\_Pi(Ci))
72. **else**:
73. val = Ci / Si **if** Si != 0 **else** 0
74. Pi.append(min(val, 3))
75. **return** Pi
77. df['Pi\_values'] = df.apply(calc\_Pi, axis=1)
78. df['P'] = df['Pi\_values'].apply(**lambda** x: np.sqrt(np.mean(np.square(x)) + np.square(max(x) - np.mean(x))))
80. **def** determine\_grade(p\_value):
81. **if** p\_value >= 1.7:
82. **return** 'I级', '优', 'green'
83. **elif** p\_value >= 0.9:
84. **return** 'II级', '中', 'orange'
85. **else**:
86. **return** 'III级', '差', 'red'
88. df['肥力等级'] = df['P'].apply(**lambda** p: determine\_grade(p)[0])
89. df['等级描述'] = df['P'].apply(**lambda** p: determine\_grade(p)[1])
90. df['等级颜色'] = df['P'].apply(**lambda** p: determine\_grade(p)[2])
92. df\_sorted = df.sort\_values('P', ascending=False).reset\_index(drop=True)
94. **print**("\n按综合土壤肥力指数排序的学院:")
95. **print**(df\_sorted[['学院', 'P', '肥力等级', '等级描述']].to\_string(index=False))
97. correlation = {}
98. **for** col **in** index\_cols:
99. corr = df[col].corr(df['P'])
100. correlation[col] = corr
102. **print**("\n各指标与综合土壤肥力指数的相关性:")
103. **for** col, corr **in** sorted(correlation.items(), key=**lambda** x: abs(x[1]), reverse=True):
104. **print**(f"{col}: {corr:.4f}")
106. plt.figure(figsize=(14, 8))
108. bars = plt.bar(df\_sorted['学院'], df\_sorted['P'], color=df\_sorted['等级颜色'])
110. plt.axhline(y=1.7, color='green', linestyle='--', alpha=0.7, label='I级标准线 (P=1.7)')
111. plt.axhline(y=0.9, color='orange', linestyle='--', alpha=0.7, label='II级标准线 (P=0.9)')
113. **for** bar, grade **in** zip(bars, df\_sorted['肥力等级']):
114. height = bar.get\_height()
115. plt.text(bar.get\_x() + bar.get\_width()/2., height + 0.05,
116. f'{height:.3f}\n({grade})', ha='center', va='bottom')
118. legend\_patches = [
119. mpatches.Patch(color='green', label='I级 (优)'),
120. mpatches.Patch(color='orange', label='II级 (中)'),
121. mpatches.Patch(color='red', label='III级 (差)')
122. ]
123. plt.legend(handles=legend\_patches, loc='upper right')
125. plt.title('各学院土壤肥力指数比较及等级划分 (南方标准)')
126. plt.xlabel('学院')
127. plt.ylabel('综合土壤肥力指数(P)')
128. plt.xticks(rotation=45)
129. plt.grid(axis='y', linestyle='--', alpha=0.7)
130. plt.tight\_layout()
131. plt.savefig('土壤肥力指数比较与等级(南方标准).png', dpi=300)
132. plt.show()
134. grade\_counts = df['肥力等级'].value\_counts()
135. plt.figure(figsize=(10, 8))
136. plt.pie(grade\_counts, labels=[f"{grade} ({count}个)" **for** grade, count **in** grade\_counts.items()],
137. autopct='%1.1f%%', startangle=90, colors=['green', 'orange', 'red'])
138. plt.axis('equal')
139. plt.title('各肥力等级占比分布 (南方标准)')
140. plt.savefig('肥力等级分布(南方标准).png', dpi=300)
141. plt.show()
143. plt.figure(figsize=(16, 12))
144. indicators = index\_cols + ['P']
145. pd.plotting.scatter\_matrix(df[indicators], figsize=(16, 12), diagonal='kde')
146. plt.suptitle('土壤各指标之间的关系矩阵', y=1.02)
147. plt.tight\_layout()
148. plt.savefig('土壤指标关系矩阵(南方标准).png', dpi=300)
149. plt.show()
151. **print**("\n土壤肥力分析报告 (采用南方地区标准):")
152. **print**("="\*50)
153. **print**(f"1. 综合肥力指数最高的学院: {df\_sorted.iloc[0]['学院']} (P={df\_sorted.iloc[0]['P']:.4f}, 等级:{df\_sorted.iloc[0]['肥力等级']})")
154. **print**(f"2. 综合肥力指数最低的学院: {df\_sorted.iloc[-1]['学院']} (P={df\_sorted.iloc[-1]['P']:.4f}, 等级:{df\_sorted.iloc[-1]['肥力等级']})")
156. grade\_summary = df['肥力等级'].value\_counts().to\_dict()
157. **print**("\n3. 各肥力等级学院分布:")
158. **for** grade **in** ['I级', 'II级', 'III级']:
159. count = grade\_summary.get(grade, 0)
160. percent = count / len(df) \* 100
161. **print**(f"   {grade}: {count}所学院 ({percent:.1f}%)")
163. **print**("\n4. 各指标平均水平与南方标准比较:")
164. **for** col **in** index\_cols:
165. std\_value = standard\_values[col]
166. avg\_value = df[col].mean()
167. ratio = avg\_value / std\_value
168. **print**(f"   {col}: 平均值 {avg\_value:.2f}, 标准值 {std\_value:.2f}, 比值 {ratio:.2f}")
170. **print**("\n6. 综合评价与建议 (基于南方地区标准):")
171. **for** grade **in** ['I级', 'II级', 'III级']:
172. grade\_schools = df[df['肥力等级'] == grade]['学院'].tolist()
173. **if** grade\_schools:
174. **if** grade == 'I级':
175. suggestion = "土壤肥力处于高水平，肥沃或很肥沃，不缺肥。适当控制肥料，可视作物产量与品质的需求，适当施肥，不宜多施，以促进平衡为主，防止过量流失。"
176. **elif** grade == 'II级':
177. suggestion = "土壤肥力处于一般水平，尚可，个别指标可能显示缺乏，作物产量随施肥量提高较为明显。需平衡施肥，尤其针对个别肥力单项指数较低的因子，需增加其单因子施肥量。"
178. **else**:
179. suggestion = "土壤肥力处于低水平，贫瘠，作物处于缺肥状态，大部分肥力指标缺乏，个别指标严重缺乏或不宜。需增施多种肥料，加强土壤培肥，对个别肥力单项指数较低的指标要优先增加相应用量，促进肥力平衡，发挥肥料增产增效作用。"
181. **print**(f"   {grade} ({len(grade\_schools)}所学院)：{suggestion}")
182. **print**(f"      包含学院：{', '.join(grade\_schools)}")
184. df\_output = df.sort\_values('学院')
185. df\_output.to\_excel('土壤肥力分析结果(南方标准).xlsx', index=False)
187. **print**("\n分析完成，结果已保存至'土壤肥力分析结果(南方标准).xlsx'")

**第二问代码**

1. **import** pulp
2. **import** matplotlib.pyplot as plt
3. **import** seaborn as sns
4. **import** pandas as pd
6. plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
7. plt.rcParams['axes.unicode\_minus'] = False
9. plots = {
10. "学院1": 610, "学院2": 522, "学院3": 328, "学院4": 465,
11. "学院5": 473, "学院6": 350, "学院7": 407, "学院8": 341,
12. "学院9": 340, "学院10": 430, "学院11": 364, "学院12": 423,
13. "学院13": 390, "学院14": 360
14. }
16. veggies = [
17. {"id": 1, "name": "莴笋", "family": "菊科", "cycle": 90, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
18. {"id": 2, "name": "灯笼椒", "family": "茄果科", "cycle": 120, "need": "需钾", "type": "需钾", "suitable\_season": ["夏"]},
19. {"id": 3, "name": "水果黄瓜", "family": "瓜科", "cycle": 70, "need": "平衡", "type": "平衡", "suitable\_season": ["夏"]},
20. {"id": 4, "name": "甜玉米", "family": "禾本科", "cycle": 100, "need": "需氮", "type": "需氮", "suitable\_season": ["夏"]},
21. {"id": 5, "name": "香瓜", "family": "瓜科", "cycle": 100, "need": "需钾", "type": "需钾", "suitable\_season": ["夏"]},
22. {"id": 6, "name": "地瓜", "family": "旋花科", "cycle": 180, "need": "需钾", "type": "需钾", "suitable\_season": ["夏", "秋"]},
23. {"id": 7, "name": "空心菜", "family": "旋花科", "cycle": 40, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "夏", "秋"]},
24. {"id": 8, "name": "圣女果", "family": "茄果科", "cycle": 120, "need": "需钾", "type": "需钾", "suitable\_season": ["夏"]},
25. {"id": 9, "name": "南瓜", "family": "瓜科", "cycle": 120, "need": "需钾", "type": "需钾", "suitable\_season": ["夏", "秋"]},
26. {"id": 10, "name": "凉薯", "family": "豆科", "cycle": 150, "need": "固氮", "type": "固氮", "suitable\_season": ["夏", "秋"]},
27. {"id": 11, "name": "薄荷", "family": "唇形科", "cycle": 90, "need": "平衡", "type": "平衡", "suitable\_season": ["春", "夏", "秋"]},
28. {"id": 12, "name": "小芋头", "family": "天南星科", "cycle": 240, "need": "需钾", "type": "需钾", "suitable\_season": ["春", "夏"]},
29. {"id": 13, "name": "萝卜", "family": "十字花科", "cycle": 60, "need": "需磷", "type": "需磷", "suitable\_season": ["春", "秋", "冬"]},
30. {"id": 14, "name": "豌豆苗", "family": "豆科", "cycle": 30, "need": "固氮", "type": "固氮", "suitable\_season": ["春", "秋", "冬"]},
31. {"id": 15, "name": "胡萝卜", "family": "伞形科", "cycle": 100, "need": "需磷", "type": "需磷", "suitable\_season": ["春", "秋"]},
32. {"id": 16, "name": "大白菜", "family": "十字花科", "cycle": 90, "need": "需氮", "type": "需氮", "suitable\_season": ["秋", "冬"]},
33. {"id": 17, "name": "菠菜", "family": "苋科", "cycle": 50, "need": "需氮高", "type": "需氮", "suitable\_season": ["春", "秋", "冬"]},
34. {"id": 18, "name": "芥兰", "family": "十字花科", "cycle": 70, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋", "冬"]},
35. {"id": 19, "name": "茼蒿", "family": "菊科", "cycle": 45, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
36. {"id": 20, "name": "西兰花", "family": "十字花科", "cycle": 100, "need": "需氮", "type": "需氮", "suitable\_season": ["秋", "冬"]},
37. {"id": 21, "name": "生菜", "family": "菊科", "cycle": 60, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
38. {"id": 22, "name": "上海青", "family": "十字花科", "cycle": 40, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋", "冬"]},
39. {"id": 23, "name": "甘蓝", "family": "十字花科", "cycle": 100, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
40. {"id": 24, "name": "蒜", "family": "葱科", "cycle": 120, "need": "平衡", "type": "平衡", "suitable\_season": ["秋"]},
41. {"id": 25, "name": "芹菜", "family": "伞形科", "cycle": 120, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
42. {"id": 26, "name": "葱", "family": "葱科", "cycle": 150, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋"]},
43. {"id": 27, "name": "香菜", "family": "伞形科", "cycle": 50, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋", "冬"]},
44. {"id": 28, "name": "油麦菜", "family": "菊科", "cycle": 45, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "秋", "冬"]},
45. {"id": 29, "name": "韭菜", "family": "葱科", "cycle": 80, "need": "需氮", "type": "需氮", "suitable\_season": ["春", "夏", "秋"]},
46. {"id": 30, "name": "茄子", "family": "茄果科", "cycle": 120, "need": "需钾", "type": "需钾", "suitable\_season": ["夏"]},
47. ]
49. seasons = ["春", "夏", "秋","冬"]
50. unique\_families = sorted(list(set(v['family'] **for** v **in** veggies)))
52. prob = pulp.LpProblem("Microfarm\_Land\_Utilization", pulp.LpMaximize)
54. x = pulp.LpVariable.dicts("x",
55. ((v['id'], p\_name, s\_name) **for** v **in** veggies **for** p\_name **in** plots **for** s\_name **in** seasons),
56. lowBound=0, cat=pulp.LpContinuous)
58. y\_fam\_planted = pulp.LpVariable.dicts("y\_fam\_planted",
59. ((p\_name, family, s\_name) **for** p\_name **in** plots **for** family **in** unique\_families **for** s\_name **in** seasons),
60. cat=pulp.LpBinary)
62. nutrient\_types\_for\_rotation = ["需氮", "固氮"]
63. y\_type\_planted = pulp.LpVariable.dicts("y\_type\_planted",
64. ((p\_name, n\_type, s\_name) **for** p\_name **in** plots **for** n\_type **in** nutrient\_types\_for\_rotation **for** s\_name **in** seasons),
65. cat=pulp.LpBinary)
67. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies **for** p\_name **in** plots **for** s\_name **in** seasons])
69. **for** p\_name **in** plots:
70. **for** s\_name **in** seasons:
71. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies]) <= plots[p\_name]
73. **for** family **in** unique\_families:
74. **for** s\_name **in** seasons:
75. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies **if** v['family'] == family]) \
76. <= plots[p\_name] \* y\_fam\_planted[p\_name, family, s\_name]
77. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies **if** v['family'] == family]) \
78. >= 0.1 \* y\_fam\_planted[p\_name, family, s\_name]
80. **for** n\_type **in** nutrient\_types\_for\_rotation:
81. **for** s\_name **in** seasons:
82. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies **if** v['type'] == n\_type]) \
83. <= plots[p\_name] \* y\_type\_planted[p\_name, n\_type, s\_name]
84. prob += pulp.lpSum([x[v['id'], p\_name, s\_name] **for** v **in** veggies **if** v['type'] == n\_type]) \
85. >= 0.1 \* y\_type\_planted[p\_name, n\_type, s\_name]
87. **for** s\_idx, s\_name **in** enumerate(seasons):
88. **if** s\_idx < len(seasons) - 1:
89. next\_s\_name = seasons[s\_idx + 1]
91. **for** family **in** unique\_families:
92. prob += y\_fam\_planted[p\_name, family, s\_name] + y\_fam\_planted[p\_name, family, next\_s\_name] <= 1
94. prob += y\_type\_planted[p\_name, "固氮", next\_s\_name] >= y\_type\_planted[p\_name, "需氮", s\_name]
96. **for** v **in** veggies:
97. **for** p\_name **in** plots:
98. **for** s\_name **in** seasons:
99. **if** s\_name **not** **in** v['suitable\_season']:
100. prob += x[v['id'], p\_name, s\_name] == 0
102. status = prob.solve()
104. **print**(f"Optimization Status: {pulp.LpStatus[status]}")
105. **if** pulp.LpStatus[status] == "Optimal":
106. **print**(f"Maximum Land Utilization (Total Planted Area): {pulp.value(prob.objective):.2f} m^2")
107. **print**("\n--- Planting Schedule ---")
108. result\_data = []
109. **for** p\_name **in** plots:
110. **for** s\_name **in** seasons:
111. **for** v **in** veggies:
112. area = pulp.value(x[v['id'], p\_name, s\_name])
113. **if** area **is** **not** None **and** area > 0.1:
114. result\_data.append({
115. "Plot": p\_name,
116. "Season": s\_name,
117. "Vegetable": v['name'],
118. "Area": area,
119. "Family": v['family'],
120. "Type": v['type']
121. })
122. df = pd.DataFrame(result\_data)
123. **if** **not** df.empty:
124. **print**(df.pivot\_table(index=["Plot", "Season"], columns="Vegetable", values="Area", fill\_value=0).round(1))
125. **else**:
126. **print**("无种植分配结果。")
128. plt.figure(figsize=(18, 8))
129. pivot = df.pivot\_table(index="Plot", columns="Season", values="Area", aggfunc="sum", fill\_value=0)
130. sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
131. plt.title("各地块各季节种植总面积（m²）")
132. plt.ylabel("地块")
133. plt.xlabel("季节")
134. plt.tight\_layout()
135. plt.show()
137. plt.figure(figsize=(18, 8))
138. veg\_season = df.groupby(["Season", "Vegetable"])["Area"].sum().unstack(fill\_value=0)
139. veg\_season.plot(kind="bar", stacked=True, colormap="tab20", figsize=(18, 8))
140. plt.title("各季节蔬菜种植面积分布")
141. plt.ylabel("面积（m²）")
142. plt.xlabel("季节")
143. plt.legend(bbox\_to\_anchor=(1.01, 1), loc='upper left', fontsize=10)
144. plt.tight\_layout()
145. plt.show()
147. plt.figure(figsize=(18, 8))
148. kind\_pivot = df.groupby(["Plot", "Season"])["Vegetable"].nunique().unstack(fill\_value=0)
149. sns.heatmap(kind\_pivot, annot=True, fmt="d", cmap="YlOrRd")
150. plt.title("各地块各季节蔬菜种类数")
151. plt.ylabel("地块")
152. plt.xlabel("季节")
153. plt.tight\_layout()
154. plt.show()
156. plt.figure(figsize=(10, 10))
157. veg\_total = df.groupby("Vegetable")["Area"].sum().sort\_values(ascending=False)
158. veg\_total.plot(kind="pie", autopct='%1.1f%%', startangle=140, colormap="tab20")
159. plt.title("各蔬菜全年总种植面积占比")
160. plt.ylabel("")
161. plt.tight\_layout()
162. plt.show()
164. plt.figure(figsize=(12, 6))
165. plot\_total = df.groupby("Plot")["Area"].sum().sort\_values(ascending=False)
166. plot\_total.plot(kind="bar", color=sns.color\_palette("Set2", len(plot\_total)))
167. plt.title("各地块全年总种植面积")
168. plt.ylabel("面积（m²）")
169. plt.xlabel("地块")
170. plt.tight\_layout()
171. plt.show()
173. plt.figure(figsize=(12, 6))
174. fam\_total = df.groupby("Family")["Area"].sum().sort\_values(ascending=False)
175. fam\_total.plot(kind="bar", color=sns.color\_palette("Set3", len(fam\_total)))
176. plt.title("各科属全年总种植面积")
177. plt.ylabel("面积（m²）")
178. plt.xlabel("科属")
179. plt.tight\_layout()
180. plt.show()
182. plt.figure(figsize=(10, 6))
183. type\_total = df.groupby("Type")["Area"].sum().sort\_values(ascending=False)
184. type\_total.plot(kind="bar", color=sns.color\_palette("pastel", len(type\_total)))
185. plt.title("各养分类型全年总种植面积")
186. plt.ylabel("面积（m²）")
187. plt.xlabel("养分类型")
188. plt.tight\_layout()
189. plt.show()
191. **if** **not** df.empty:
192. plot\_names\_ordered = sorted(list(plots.keys()))
193. num\_plots = len(plot\_names\_ordered)
195. **if** num\_plots > 0:
196. ncols = 2
197. nrows = (num\_plots + ncols - 1) // ncols
199. fig\_rotation, axes\_rotation = plt.subplots(nrows, ncols, figsize=(ncols \* 7, nrows \* 5), squeeze=False)
200. fig\_rotation.suptitle("各学院年度轮作计划", fontsize=18, y=0.99)
202. ax\_flat = axes\_rotation.flatten()
204. **for** i, p\_name **in** enumerate(plot\_names\_ordered):
205. ax = ax\_flat[i]
206. ax.set\_title(f"{p\_name} 轮作计划", fontsize=10)
208. plot\_data = df[df["Plot"] == p\_name].copy()
210. y\_pos = 0.95
211. line\_height = 0.07
213. **if** **not** plot\_data.empty:
214. plot\_data['Season'] = pd.Categorical(plot\_data['Season'], categories=seasons, ordered=True)
215. plot\_data = plot\_data.sort\_values('Season')
217. **for** s\_name **in** seasons:
218. ax.text(0.05, y\_pos, f"{s\_name}:", transform=ax.transAxes, fontsize=9, weight='bold', va='top')
219. y\_pos -= line\_height
221. season\_specific\_data = plot\_data[plot\_data["Season"] == s\_name]
223. **if** **not** season\_specific\_data.empty:
224. **for** \_, row **in** season\_specific\_data.iterrows():
225. **if** y\_pos < 0.05:
226. ax.text(0.1, y\_pos, "...", transform=ax.transAxes, fontsize=8, va='top')
227. **break**
228. ax.text(0.1, y\_pos, f"- {row['Vegetable']} ({row['Area']:.0f}m²)",
229. transform=ax.transAxes, fontsize=8, va='top')
230. y\_pos -= line\_height
231. **else**:
232. ax.text(0.1, y\_pos, "- 无种植", transform=ax.transAxes, fontsize=8, va='top')
233. y\_pos -= line\_height
235. y\_pos -= line\_height \* 0.1
236. **else**:
237. ax.text(0.5, 0.5, "无种植数据", transform=ax.transAxes,
238. ha='center', va='center', fontsize=10)
240. ax.axis('off')
242. **for** j **in** range(num\_plots, nrows \* ncols):
243. fig\_rotation.delaxes(ax\_flat[j])
245. plt.tight\_layout(rect=[0, 0.03, 1, 0.97])
246. plt.show()
247. **else**:
248. **print**("无种植数据，无法生成轮作图。")
250. **else**:
251. **print**("Could not find an optimal solution.")

**第三问代码**

1. **import** pandas as pd
2. **import** matplotlib.pyplot as plt
3. **import** seaborn as sns
4. **import** numpy as np
6. # 设置中文显示
7. plt.rcParams['font.sans-serif'] = ['SimHei']
8. plt.rcParams['axes.unicode\_minus'] = False
10. **def** plot\_soil\_grades():
11. """绘制土壤等级分布饼图"""
12. plt.figure(figsize=(10, 8))
13. grades = {
14. 'I级': 23.5,
15. 'II级': 47.1,
16. 'III级': 29.4
17. }
18. colors = ['#2ecc71', '#f1c40f', '#e74c3c']
19. plt.pie(grades.values(), labels=grades.keys(), colors=colors, autopct='%1.1f%%',
20. startangle=90)
21. plt.title('土壤肥力等级分布')
22. plt.show()
24. **def** plot\_nutrient\_needs():
25. """绘制作物养分需求分布图"""
26. plt.figure(figsize=(10, 8))
27. nutrient\_types = {
28. '需氮': 18,
29. '需钾': 8,
30. '需磷': 2,
31. '固氮': 2,
32. '平衡': 3
33. }
34. bars = plt.bar(nutrient\_types.keys(), nutrient\_types.values())
35. plt.title('作物养分需求类型分布', pad=20)
36. plt.xlabel('养分类型', labelpad=15)
37. plt.ylabel('作物数量', labelpad=10)
39. plt.xticks(rotation=30, ha='right')
40. **for** bar **in** bars:
41. height = bar.get\_height()
42. plt.text(bar.get\_x() + bar.get\_width()/2., height + 0.5,
43. f'{int(height)}', ha='center', va='bottom')
44. plt.show()
46. **def** plot\_seasonal\_heatmap():
47. """绘制季节性种植建议热力图"""
48. plt.figure(figsize=(12, 8))
49. seasons = ['春', '夏', '秋', '冬']
50. crop\_types = ['需氮作物', '需钾作物', '需磷作物', '固氮作物', '平衡型作物']
51. data = np.array([
52. [8, 3, 2, 2, 2],
53. [4, 6, 1, 1, 2],
54. [7, 4, 2, 2, 2],
55. [4, 1, 1, 1, 1]
56. ])
57. sns.heatmap(data, annot=True, fmt='d', cmap='YlGnBu',
58. xticklabels=crop\_types, yticklabels=seasons)
59. plt.title('季节性种植建议分布')
60. plt.xticks(rotation=45, ha='right')
61. plt.tight\_layout()
62. plt.show()
64. **def** plot\_improvement\_suggestions():
65. """绘制土壤改良建议表格"""
66. fig, ax = plt.subplots(figsize=(12, 9))  # 进一步增加高度到9
67. ax.axis('off')
68. ax.axis('tight')
69. ax.set\_title('土壤改良建议', pad=20, fontsize=16, loc='center')
71. improvement\_methods = {
72. 'I级土壤': ['维持现状', '注意养分平衡', '适量施肥'],
73. 'II级土壤': ['补充有机质', '适量施肥', '改良土壤结构'],
74. 'III级土壤': ['大量施用有机肥', '合理轮作', '改良理化性质', '增施磷肥']
75. }
77. table\_data = []
78. **for** grade, methods **in** improvement\_methods.items():
79. # 使用方框符号 □ 替代圆点 •
80. methods\_string = "\n".join([f"□ {method}" **for** method **in** methods])
81. table\_data.append([grade, methods\_string])
83. col\_labels = ['土壤等级', '改良建议']
85. table = ax.table(cellText=table\_data,
86. colLabels=col\_labels,
87. loc='center',
88. cellLoc='left',
89. colWidths=[0.2, 0.8])
91. table.auto\_set\_font\_size(False)
92. table.set\_fontsize(8)  # 减小字体至8
93. table.scale(1, 6.5)  # 增加行高缩放至6.5
95. **for** (row, col), cell **in** table.get\_celld().items():
96. cell.set\_edgecolor('grey')
97. cell.PAD = 0.05
99. **if** row == 0:
100. cell.set\_text\_props(weight='bold', color='white', ha='center', va='center')
101. cell.set\_facecolor('#3670A0')
102. **else**:
103. cell.set\_text\_props(ha='left', va='top')
104. cell.get\_text().set\_wrap(True)
105. **if** row % 2 == 0:
106. cell.set\_facecolor('#E8F0F7')
107. **else**:
108. cell.set\_facecolor('#FFFFFF')
110. plt.tight\_layout()  # 添加tight\_layout以优化布局
111. plt.show()
112. **if** \_\_name\_\_ == "\_\_main\_\_":
113. # 设置中文显示
114. plt.rcParams['font.sans-serif'] = ['SimHei']
115. plt.rcParams['axes.unicode\_minus'] = False
117. # 依次显示四个图表
118. plot\_soil\_grades()
119. plot\_nutrient\_needs()
120. plot\_seasonal\_heatmap()
121. plot\_improvement\_suggestions()