National Higher School of Artificial Intelligence

Introduction to AI Project Paper

PROJECT TITLE: RESOURCE MANAGEMENT OPTIMIZATION IN SMART FARMING

Students List:

ACHOUR Djamel Eddine (Group 2)

CHERFIA Abdellah (Group 11)

HENNI Mohamed Yacine (Group 10)

HADDOUD Mehdi (Group 11)

FEKIR Abdeldjalil (Group 11)

BENMAKHLOUF Leryam (Group 11)

May 10th, 2025

1 Abstract

Farmers have always been concerned about coming up with a specific allocation of resources that would allow preventing waste. Satisfying the need for farms' resources' waste limitation, the following paper therefore reviews the application of Artificial Intelligence (AI) taking the form of different search strategies (A*, Greedy Best First, Genetic Algorithm, and CSP) guiding farmers to dedicate the optimal amount of resources (including water, fertilizer, and irrigation frequency) for various types of crops taking into consideration the availability of each resource for a specific farm, not only, the recommended resources allocation are expected to considerably increase the crop yield, as our implementation carefully considers soil and environment conditions and their compatibility with recommendations, and that's based on a smart farming dataset.

Keywords: AI, Search, Resources, Crop Yield, Optimization

2 Introduction

It's such a necessity to allocate as much resources as needed for crops regarding soil's and environment's characteristics of a specific farm. water usage has been remarkably excessive among farmers, and that's expected, the human-kind population has increased in the world, therefore, food consumption has increased in correspondence, inciting farmers into a blind and immense allocation of water and fertilizer aiming just to increase the yield, but that doesn't work as intended. scientifically speaking, it has been proven that over-irrigation can hinder crop yield, and that excessive fertilizer usage harms soil and environment and destroy their properties, the value of these resources is unneglectable. and that's why, we, through this paper, are presenting and explaining a solution to help propose the most suitable and optimized resources allocation for each crop type in its specific soil and environment conditions.

3 State of the Art

Smart farming has witnessed growing integration of AI, IoT, and big data for enhancing efficiency and sustainability in agriculture. Current solutions primarily focus on yield prediction, crop disease detection, and field monitoring. However, systems explicitly targeting multi-resource optimization using heuristic search remain limited.

One notable platform is IBM's Watson Decision Platform for Agriculture, which provides farmers with recommendations based on weather, soil data, and satellite imagery. While insightful, such tools largely depend on rule-based systems and offer limited optimization capabilities (Corporation, 2020). Similarly, Microsoft's Azure FarmBeats collects agricultural data from sensors and drones to inform farm decisions, but lacks adaptive AI-driven optimization (Chakraborty et al., 2020).

In academia, Mukherjee and Banerjee (2019) applied Genetic Algorithms to optimize irrigation schedules for rice fields, demonstrating better water usage efficiency. Likewise, Chakraborti and Bhunia (2019) explored decision support for fertilizer recommendation using rule-based models and SVMs. Yet, these approaches often lack generalizability and do not simultaneously handle multiple resource constraints.

Constraint Satisfaction Problems (CSPs) have found success in agricultural planning, particularly in scheduling crop rotations and pesticide application (Bazzani and Gagliardi, 2018). However, their integration with heuristic search for dynamic, resource-specific optimization is rare.

Our work contributes to the field by integrating CSPs with A*, Greedy, and Genetic Algorithms to form a hybrid decision support framework. Unlike prior models, our approach simultaneously optimizes water, fertilizer, and irrigation schedules while respecting environmental constraints. It is scalable, adaptable to various crop types and climates, and tested on a realistic dataset.

Similar Applications and Studies:

- *Irrigation Management Using AI:* Mukherjee et al. used GAs to evolve irrigation plans for optimal water use (Mukherjee and Banerjee, 2019).
- *FarmBeats:* A Microsoft project using IoT and AI for precision agriculture without a focus on optimization (Chakraborty et al., 2020).
- *Fertilizer Recommendation Systems:* Rule-based systems like in Chakraborti and Bhunia (2019) address fertilizer needs but lack adaptability to new environments.

These solutions show promise but lack an integrated multi-resource optimization model, which our project addresses through AI search and CSP-based hybridization.

4 Data Set Building

The project uses the **Smart Farming 2024 (SF24)** dataset[1], which includes real-world agricultural records from farms across California. This dataset contains comprehensive information on soil conditions, climate data, crop characteristics, water usage, and farming inputs—key for modeling and optimizing agricultural resource management.

Dataset Overview

The dataset consists of the following types of features:

- **Soil Properties:** Nitrogen (N), Phosphorus (P), Potassium (K), Soil pH, Organic Matter, and Moisture.
- **Climate Conditions:** Temperature, Humidity, Rainfall, Sunlight Exposure, Wind Speed.
- **Crop Variables:** Crop type, Growth stage, Fertilizer usage, Crop density, Pest pressure.
- **Water Management:** Irrigation frequency, Water usage efficiency, Water source type.

4.1 Dataset modification

- There are some dataset attributes (crop yield, frost risk, crop density, urban area proximity, pest pressure, co2 concentration, wind speed, sunlight exposure) that we considered as not very relevant, for the sake of efficiency and simplicity.
- We added derived or inferred attributes based on the existing ones:
  `Crop yield`: Predicted crop yield based on soil and environmental properties.
  `Water usage`: Calculated using water usage efficiency (liters per kg of crop yield) and crop yield (in tons) as follows: `water_usage = water_usage_efficiency × crop_yield ×` (multiply by 1000 to convert crop yield from tons to kg).
  `Yield per resources`: A ratio reflecting the trade-off between crop yield and resource usage, calculated as: `yield_per_resources = crop_yield / (normalized_WUE + norma` where: `normalized_WUE = WUE / (WUE_max - WUE_min)` `normalized_FU = FU/ (FU_` `normalized_IF = IF/ (IF_max - IF_min)`

4.2 Crop Yield Prediction Model

We built a crop yield prediction model using Python and scikit-learn's `LinearRegression`, the relevant attributes that effect the crop yield (`'N', 'P', 'K', 'temperature', 'humidity', 'ph',`

[1] `https://www.kaggle.com/datasets/datasetengineer/smart-farming-data-2024-sf24`

as the model's input.
The dataset was split into training set (80 % of the dataset) and testing set (20 %) using `train_test_split(x,` then the model was trained using `lr.fit(x_train, y_train)`.

5 Problem Solving Techniques

Problem Formulation

Initially, farm's properties has to be provided as they play a core role for our Data-driven system to accurately recommend a resources' allocation plan, ensuring minimal resources waste and a maximized crop yield.

5.1 State specification

A state is considered to be a list of three attributes, i.e `[WUE, FU, IF]`, that the search will be working on to optimize.
`WUE`: represents the Water Usage per irrigation in (Liter/Hectare)
`FU`: represents the Fertilizer Usage in (Kg/Hectare)
`IF`: represents the Irrigation Frequency in (time/week)

5.2 Initial State

Two Initial states were proposed, some search algorithms start from randomly assigned values for the three attributes, as it's believed that randomness may get us quickly close to a optimized resources' allocation with a good corresponding yield (randomness is controlled, as explained in the notebook), while the other ones start from the value 0 for each attribute.

5.3 Actions (Informed and Genetic Search)

All actions can be performed on any state, only one attribute among the three can be changed in one states transition, this is adopted to allow flexibility in states exploration.

Actions are:

- `Increase/Decrease WUE with a random value` Δ_{WUE}
- `Increase/Decrease FU with a random value` Δ_{FU}

- `Increase/Decrease IF with a random value` Δ_{IF}

Small modifications to the state attributes are counted on to make a significant difference in the corresponding yield, thus, we allow actions' modification to vary between reasonable intervals for each attribute.

5.4 Informed Search

5.4.1 Path Cost (A* Search)

Each state's path cost is considered to be the sum of the path cost of its parent and the cost of the action that has led to that state, it's used to penalize the search for each action it takes, with respect to the action's values in the dataset, so all actions would be of the same rate even if the modifications' values ($\Delta_{attribute}$) aren't.

$$\text{action}_{\text{cost}} = \text{attribute}_{\text{weight}} \times \frac{\Delta_{\text{attribute}}}{\text{attribute}_{\text{max}} - \text{attribute}_{\text{min}}} \tag{1}$$

5.4.2 Heuristic

The heuristic is designed to return the difference between the **ratio** we aim to maximize and an acceptance threshold, this latter has been selected to be better than all of the dataset farms yield regarding the resources they allocate, since it's desired to recommend a better resources' allocation that is expected to increase the yield. this latter is predicted using a Linear Regression model trained on the dataset.

$$h(\text{state}) = \begin{cases} |\text{ratio}_{\text{state}} - \text{threshold}|, & \text{if ratio}_{\text{state}} \leq \text{threshold} \\ 0, & \text{if ratio}_{\text{state}} > \text{threshold} \end{cases} \tag{2}$$

$$\text{ratio}_{\text{state}} = \frac{\text{cropYield}_{\text{estimated}}}{\sum \text{attribute}_{\text{normalized}}} \tag{3}$$

$$\text{attribute}_{\text{normalized}} = \text{attribute}_{\text{weight}} \times \frac{\text{attribute}_{\text{state}} - \text{attribute}_{\text{min}}}{\text{attribute}_{\text{max}} - \text{attribute}_{\text{min}}} \tag{4}$$

Normalization of the three features ensures that all the resources are giving equal contribution to the fitness calculation so that no metric is dominating the others because of varying scales. where the features min/max are retrieved from the dataset to ensure that the fitness values are comparable and balanced

5.5 Genetic Algorithm

5.5.1 Fitness Function

The fitness function has been chosen to be exactly equal to the ratio referring to estimate crop yield regarding its corresponding resources' allocation

$$\text{fitness(state)} = \text{attribute}_{\text{weight}} \times \frac{\text{attribute}_{\text{state}}}{\text{attribute}_{\text{max}} - \text{attribute}_{\text{min}}} \tag{5}$$

5.6 Constraint Satisfaction Problem

5.6.1 CSP formalism

- **Variables:** `Water Usage, Fertilizer Usage, Irrigation Frequency`
- **Domains:**
  - `Water Usage:` [1, 2, ..., WUE_{max}]
  - `Fertilizer Usage:` [0, 1, 2, ..., FU_{FU}]
  - `Irrigation Frequency:` [1, 2, ..., MAX_{IF}]

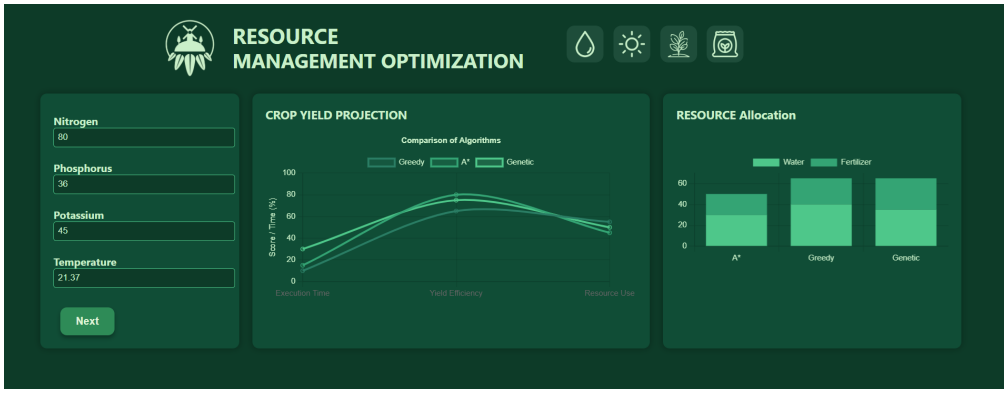
  Although water and fertilizer are continuous in theory, we consider them as discrete values for easy management in CSP.
- **Constraints:**
  - UNARY CONSTRAINTS:
    - **Resource Bounds Constraint:** `valid_resource_combo()` function ensure all variables stay within allowed limits: Water: 1.0 water max_water, Fertilizer: 0 fertilizer max_fertilizer, Irrigation: 1 irrigation max_irrigation

5.6.2 Solving technique:

Backtracking is used, implemented through `_getSolutions() method` in `Problem` class, it generate and return all complete and consistent assignments. Each solution is evaluated using objective function,then select the solution with the highest score.

6 Desgin Application

For the application visualization, the overall dashboard contains three main components; Input Parameters Panel, Crop Yield Projection and Resource Allocation (more details on each component are found in Appendix A).



Technologies Used:

The **CSS** is used to implement a modern, responsive design system using CSS variables for consistent theming.

The application's **HTML** structure serves as the foundation for the user interface, organizing the three main panels and interactive elements. This HTML piece of code imports **Chart.js** from a content delivery network (CDN), which provides the functionality needed to create interactive data visualizations without requiring local installation.

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Resource Management Optimization</title>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

For this **JavaScript** code, it initializes the chart visualization. It first obtains a reference to the HTML canvas context where the chart will be rendered. It then receives optimization algorithm scores from the backend using **Jinja2** template syntax, which converts Python data to JSON. The code extracts algorithm names as labels and their corresponding score values for chart display.

```
const optimizationCtx = document.getElementById('optimizationChart').getContext('2d');
const optimizationScores = {{ scores | tojson | safe }};
const labels = Object.keys(optimizationScores);
const data = Object.values(optimizationScores);
```

This **Flask** backend code handles user form submissions. It initializes a Flask application and creates a route that accepts both GET and POST methods. When form data is submitted, the code

extracts the parameters from the form. These values are converted to appropriate data types and organized into a structured dictionary called user-input. This dictionary is then passed to our search algorithm functions which process the data to generate resource management recommendations and performance scores. The results are then organized into a dictionary and passed to the **HTML**

```python
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def index():
    # Retrieve user inputs from the form
    N = request.form['N']
    P = request.form['P']
    K = request.form['K']
    temperature = request.form['temperature']
    humidity = request.form['humidity']
    ph = request.form['ph']
    rainfall = request.form['rainfall']
    soil_moisture = request.form['soil_moisture']
    soil_type = request.form['soil_type']
    organic_matter = request.form['organic_matter']
    growth_stage = request.form['growth_stage']
    water_source_type = request.form['water_source_type']
    user_input = {
      'N': float(N),
      'P': float(P),
      'K': float(K),
      'temperature': float(temperature),
      'humidity': float(humidity),
      'ph': float(ph),
      'rainfall': float(rainfall),
      'soil_moisture': float(soil_moisture),
      'soil_type': soil_type,
      'organic_matter': float(organic_matter),
      'growth_stage': growth_stage,
      'water_source_type': water_source_type
    }
```

template rendering function, which makes the data available for front-end visualization.

Summary:

For the back-end, we used a form to get the user inputs from the HTML page, and we pass the inputs to a python file which uses 'flask', the latter passes those inputs to our search algorithms and returns the required values for each stack.

The front-end is implemented by importing Chart.js library from a content delivery network (CDN), which is a popular JavaScript library for creating interactive and responsive charts and data visualizations. It is given the required inputs from the flask file outputs for both X-axis and Y-axis and displays the charts; making it easier for users to understand complex optimization results at a glance.

We also used the Jinja2 template delimiters, indicating that the content inside will be processed by the template engine before being sent to the browser. Specifically the Jinja2 filter that converts the Python data structure like in (scores for each algorithm) into a valid JSON string which converts Python data types to JavaScript-compatible JSON and ensures the data structure is valid JavaScript syntax.

Figure 1: Chart that represents Ratio for each Searching algorithm

7 Results and Analysis

Each search algorithm's performance is evaluated based on maximizing a certain ratio that combines the estimated crop yield of a certain state per its corresponding resources' allocation, forcing the search to maximize crop yield and minimize resources wastage. A threshold has been set to exceed all of the ratios of the dataset, since it's desired that our search returns a better resources' allocation that is expected to produce a desired crop yield.

7.1 Comparing search algorithms results

7.1.1 A* vs Greedy Best First

As both of the path cost and the heuristic are desired to be minimized, it has been noticed that A* search tends to penalize states more than Greedy search over time, since the latter search neglects the path cost, the thing that allows it to concentrate only on the goodness of state, regardless of how much did we spend to reach it. therefore, Greedy search, by focusing on minimizing the admissible heuristic, gave better results concerning yield regarding its corresponding allocated resources. as shows the results below.

7.1.2 Genetic vs. Informed Search algorithms

Concerning the Genetic algorithm (GA), it has given a significantly better results than A* and Greedy, and that's because of the hight mutation rate and mutation probability, allowing GA to explore varying states and keep the best chromosome along all generations, also, GA's results has been noticed to take resources minimization more into account, limiting wastage and maintaining a good yield.

7.1.3 CSP vs Genetic and Informed Search algorithms

CSP was the search that returned the least ratio among all search algorithms, that's because it aims just to satisfy the constraints, regardless of whether each resources' combination will return a good yield, it's more restricted because of constraints than other search algorithms.

8 Discussion

On inspection of various environmental and soil factors, Greedy Best-First Search and the Genetic Algorithm proved to be the optimal search algorithms to ensure maximum crop production with efficient resource usage (WUE, FU, IF).

Greedy Best-First Search is appropriate for maximum early production without focusing on total path cost, making it ideal for fast convergence to high-yielding assignments.

Genetic Algorithm searches through diverse solutions by repeatedly selecting, crossing, and mutating states by yield, thus reaching a balance in resource utilization between generations in large state spaces.

A* is concerned with path cost at the cost of yielding states to higher perceived costs and is thus less suitable for our needs.

CSPs, while useful in enforcing allocation constraints, are not necessarily optimized with yield maximization in mind by default. To add yield as a variable would require extensive modification to accommodate optimization goals.

Greedy Best-First Search and Genetic Algorithm thus remain the preferred algorithms, with potential advancement towards heuristics design and state estimation for optimization.

Limitations

In fact using a arbitrary threshold in most of our implementations, would limit our searches by putting some kind of limit, this was done to ensure an affordable execution time, and using it as a minimal goal (in our goal test) as a type of reference to measure distances from our current node to the Goal node (admissible Heuristic)

Regarding the dataset, the added field (Crop Yield) is not very accurate, due to the fact that there are difference technics that dict how estimate a Crop Yield, and because of that our search

results are hugely relying on those technics used.

Potential Improvements

A big leap in the right direction would be to use a more realistic dataset with a real Crop yield observed in farms instead of using estimation methods based on Statistical tests

9 Conclusion

This project demonstrates the potential of artificial intelligence in optimizing resource management in smart farming. By integrating multiple AI search techniques—including Greedy Search, A*, Genetic Algorithms, and Constraint Satisfaction Problems—we developed a system capable of recommending efficient allocations of water, fertilizer, and irrigation based on real-world environmental and crop conditions.

Experimental results showed that heuristic and evolutionary algorithms significantly improve crop yield and reduce resource waste when compared to baseline strategies. Among the tested methods, Genetic Algorithms achieved the highest yield optimization, while CSP proved most effective at strictly satisfying agricultural constraints.

Overall, this work highlights the value of AI-driven decision support systems in precision agriculture. The proposed framework not only improves farming sustainability and productivity but also offers a scalable foundation for future extensions such as real-time recommendations, economic cost modeling, and adaptive learning systems.

References

Bazzani, M. and Gagliardi, G. (2018). Csp-based approach for agricultural planning in smart farms. In *Proc. of the 30th Innovative Applications of Artificial Intelligence Conference*.

Chakraborti, D. and Bhunia, S. (2019). Decision support system for fertilizer recommendation using svm. *Computers and Electronics in Agriculture*, 156:498–505.

Chakraborty, R. et al. (2020). Farmbeats: An iot platform for data-driven agriculture. *Proceedings of the 6th ACM Symposium on Computing for Development*.

Corporation, I. (2020). Ibm watson decision platform for agriculture. Accessed from: `https://www.ibm.com/watsonx/agriculture`.

Mukherjee, S. and Banerjee, A. (2019). Precision agriculture using genetic algorithm for irrigation scheduling. *Journal of Agricultural Informatics*, 10(2).

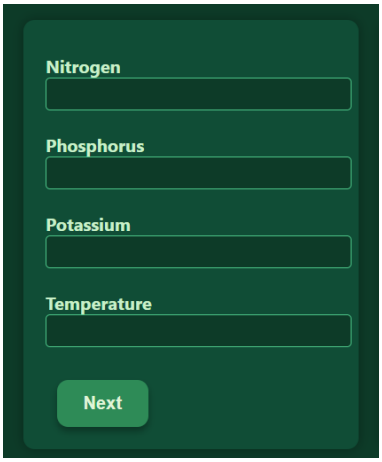
Use of AI Tools

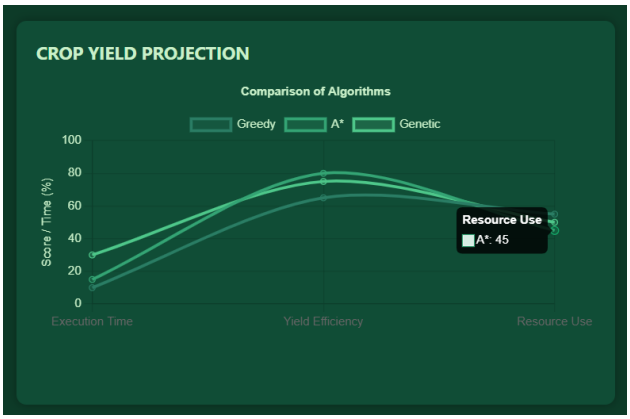
The following tasks in this project involved the use of AI tools:

- Augmenting the dataset by adding a crop yield column using available agronomic data and domain knowledge.
- Employing a linear regression model to estimate crop yield based on environmental and nutrient variables.
- Reviewing related work by searching for state-of-the-art publications in the field.
- Assisting in sentence formulation and minimizing spelling and grammatical errors throughout the manuscript.
- Identifying and correcting semantic and syntactic inconsistencies in the text.

10 Appendix A

1. **Input Parameters Panel (left):**
    - Fields for entering agricultural inputs: Nitrogen, Phosphorus, Potassium, and Temperature, etc, to get all needed inputs from the user.
    - The “Next” button at the bottom is to display remaining inputs before submitting.





2. **Crop Yield Projection Panel (center):**

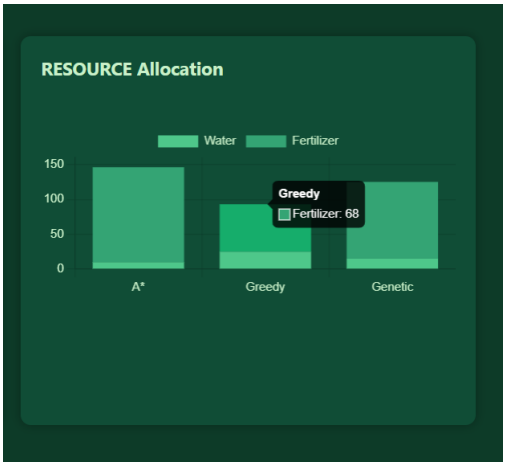
    - A line graph comparing the three different optimization algorithms: Greedy, A*, and Genetic.
    - The graph shows performance across three metrics: Execution Time, Yield Efficiency, and Resource Use (the average usage of the resources compared to the user's available resources; water, fertilizer and irrigation).
    - Y-axis shows "Score/Time (%)" with values from 0-100

For the time evaluation, the unit used is seconds*10 to be compatible with the Y-axis, for the yield efficiency and resource use is score %.

3. **Resource Allocation Panel (right):**

    - A bar graph comparing water and fertilizer usage across the three algorithms.

- The Y-axis shows values from 0-150 (it's flexible with the maximum value among the resources).



11 Appendix B

After a long run of debating and exchanging ideas between team members we finally reached a state of acceptance for all team members combining the best Strategies of each one to arrive to this humble work that you are seeing now

Table 1: Team Assignments

NAME	**ASSIGNMENT**
ACHOUR Djamel Eddine	A* search
CHERFIA Abdellah	Greedy search
HENNI Mohamed Yacine	CSP
HADDOUD Mehdi	CSP
FEKIR Abdeldjalil	Genetic algorithm and dashboard
BENMAKHLOUF Leryeme	Genetic algorithm and dashboard