The National Higher School of Artificial Intelligence

Introduction to AI Course Project

Spring 2025

Introduction to AI Project Paper

Project Title: **Resource Management Optimization in Smart Farming**

Students List:

Full Name	**Group**
SMAIL Imadeddine	G12
GARAMIDA Abderraouf Adel	G5
BRAHIMI Aya	G12
AMEDDAH Mohamed	G12
CHERGUI Mohamed Bahae Eddine	G12
SAAD Mohamed Amine	G1

Abstract

Resource optimization in smart farming consists of minimizing the resources usage efficiently while ensuring the maximum yield, it aims to save up as much as possible while keeping the crops alive and healthy, avoiding any sort of unnecessary waste or crop loss. It uses artificial intelligence search techniques to precisely find the most optimal allocation of water, fertilizer, irrigation frequency and possibly other resources depending on the user's inputs of soil parameters, crop related data and environmental factors that possess direct impact on the growth process.

Keywords :

- Crop Yield Optimization - Smart Farming - Resource Optimization using search algorithms - Agriculture 4.0 - Sustainable Agriculture

Introduction

In an overpopulated world nutrition has become a major concern to people inhabiting it, it is estimated that by less than 30 years a large population may either suffer famine, malnutrition or countless diseases that resulted from poor diet and a lack of essential nutrients.

Hence, more attention is getting directed towards enhancing the source of healthy nourishment; agriculture. Scientists and researchers are called upon to act and rethink the efficiency of traditional farming techniques, to implement the latest technologies and rely on artificial intelligence in improving crop yields. And one of the most crucial aspects being addressed in this effort is resource optimization: the pursuit of producing the highest possible yield using the least amount of resources.

It's no longer just a matter of efficiency; it's a matter of survival. As time progresses and the effects of climate change and pollution continue to take their toll, resources such as clean water and agricultural fertilizers are slowly becoming less available and more valuable.

When these resources are managed poorly, the consequences are rather negative, they lead to failed crops, reduced food supply, economic instability, and hunger. However if we succeed in optimizing resource use, we open the door to feeding more people consistently. The greater the yield, the more families can be nourished, and the closer we come to fulfilling the objective: ensuring that no one is left behind in the struggle for food, health, and survival.

This report investigates AI driven algorithmic strategies for optimizing resource allocation in smart farming, with a particular focus on enhancing productivity while minimizing environmental impact.

State Of Art

In the last few years, research has focused on integrating artificial intelligence and data-driven decision-making into agriculture to create smart farming systems. These aim at optimizing yields in crops, automating irrigation and fertilization, and reducing their environmental impact. The majority of these works employed **machine learning models** to predict crop yield based on soil and weather data. For instance, decision tree classifiers and neural networks have been used to predict yield from previous data, with high accuracy according to the research conducted by *Pr P.Invayni, Pr J.Hong and Pr B.Topping* in their book **Advances in Engineering Software**:
https://www.sciencedirect.com/science/article/pii/S0965997822002277

These approaches, however, **lack real time adaptability and do not directly integrate optimization of resource usage in a dynamic setting.**

In the field of resource management, certain models have employed linear programming to determine optimal irrigation schedules or fertilizer application. These models most often rely on **pre-established thresholds** or static models that don't capture how environmental factors, crop type, or growth phases change. Some of the commercial platforms do possess crop recommendation or irrigation optimization features, but these usually **lack flexibility or transparency in decision logic.**

Pr L.Liu, Q.Fan https://ieeexplore.ieee.org/abstract/document/8349954

Genetic algorithms and other evolutionary methods have also been applied to agricultural optimisation problems, e.g., planting schedules. Although GAs are effective in searching large solution spaces, they are typically executed **independently and without environmental feedback mechanisms**. Similarly, conventional search algorithms like A* or Greedy are rarely used in this field due to the perceived **difficulty in representing states and transitions** in real farm conditions.

What distinguishes our system is that it applies **heuristic-based** informed search and genetic algorithms specifically for intelligent farming in hybrid mode. Our model has many environmental, crop specific, and soil related parameters using a real world dataset and adjusts decisions accordingly based on **constraints of availability**. The integration of domain-knowledge **heuristics and a cost function** inspired by real-world agricultural trade-offs (e.g., the worth of water or fertilizer) is novel, interpretable, and adaptable.

Dataset Building

For the data collection, the main dataset that was used is : **Smart Farming Data 2024 (SF24) ,** featuring most of **the required parameters needed to find an optimal solution to the problematic**, it is a comprehensive collection of agricultural data gathered from various farms across California, designed in a versatile manner, contributing to several applications like predicting crop health and yield, optimizing resource usage, and assessing the impact of environmental stressors on agricultural productivity.
(Dataset link :Smart Farming Data 2024 (SF24))

Dataset Description: The SF24 dataset contains a total of 28 features, including 23 original attributes and 5 derived features, each capturing a specific aspect of environmental, soil, or crop conditions. These columns serve as the foundation for the preprocessing and analysis stages of the project. Below is a description of the main features:

Crop type.

N (ppm), P (ppm), K (ppm): Represent the levels of nitrogen, phosphorus, and potassium in the soil, key nutrients affecting plant health and productivity.

Temperature (°C) & Humidity (%): Record atmospheric conditions that impact evapotranspiration and disease susceptibility.

pH: Indicates the soil's acidity or alkalinity, which affects nutrient availability.

Soil Moisture (%): Represents water retention in the soil, essential for crop water availability.

Soil Type: Categorical value (1 = Sandy, 2 = Loamy, 3 = Clay) that affects drainage and nutrient holding capacity.

Growth Stage: Categorized into 1 (Seedling), 2 (Vegetative), and 3 (Flowering), providing context for plant development.

Fertilizer Usage (kg/ha): Quantity of fertilizer applied, influencing nutrient supply.

Water Usage Efficiency (L/kg): Measures how efficiently water is converted into crop yield.

Crop Density (plants/m²): Indicates planting intensity per unit area.

Organic Matter (%): Amount of decomposed biological material in the soil, enhancing fertility.

Our AI system uses this dataset to extract optimal ranges of soil parameters and the thresholds that should not be exceeded, throughout the preprocessing stage we applied descriptive statistics in calculating the mean to construct ranges, and finding the minimal and maximal values of certain columns depending on other parameters, we grouped data based on three factors: **crop type, soil type, and growth stage ,**while giving less consideration to crop density.

External sources: In addition to the SF24 dataset, external sources were consulted to provide extra information. These include the amounts of Nitrogen, Phosphorus, and Potassium as well as the soil moisture's increases per liter of water added. We also researched priority factors that indicate the most pressing resource needs under specific environmental conditions, along crop water requirements of each growth stage. This information was drawn from academic research papers, blogs, and agricultural guidelines, which will be cited in the references section.

Problem Solving Techniques

It is important to note that our search immensely relied on interdependencies, as water naturally contains fixed amounts of nutrients, we had to take into account the alterations to the nutrient levels in the soil with each liter added, as well as the alterations to the pH levels. These alterations represent the transition model of the problem formulation,indeed; it defines exactly the interdependencies between the three parameters (water,fertilizer aànd pH level).

Problem Formulation :

State Representation : Each state in the problem space encapsulates the current soil condition and surrounding environment. It includes following attributes:
-Crop type ,Crop growth stage and Crop density.
-Soil type and water resource (Recycled,River or Groundwater) and Soil depth and Farm Area.
-Nutrients levels in the soil (NPK levels in ppm).
-PH level in the soil.
-Environmental conditions : (temperature in C°,humidity in %,sunlight exposure,wind speed in km/h)
-Water (added/reduced) during Optimization (in L), Fertilizer (added/reduced) during Optimization ,irrigation frequency .
-Maximum amount of water that Farmer can use per irrigation (in L), Maximum amount of fertilizer Farmer can use per application (in KG).
-Percentage of Nitrogen,Phosphorus and Potassium in the Fertilizer Used.

Initial State : The initial state consist of all attributes that a state contains (mentioned above in State representation), **in which the values are provided by the farmer** ,Except Water (added/reduced) during Optimization (in L), Fertilizer (added/reduced) during Optimization ,irrigation frequency which will be all 0 at the start and will be updated during the Search application .

Actions : Each action corresponds to applying adjustments to water and/or fertilizer. These actions are constrained to only modify parameters that fall outside their optimal ranges. The allowed values are predefined discrete steps:

- **Water**: [-3, -2, 0, 2, 3] (liters per m²)
- **Nutrients (N, P, K)**: [-3, 0, 3, 6] (units/application)

Only combinations that effectively bring deviating parameters closer to their optimal levels are retained as valid actions. The decision to consider negative amounts of both Water and Nutrients was taken after largely thinking about how to deal with the (Overirrigation/Overuse of Fertilizer) situations, in which the search will try to reduce the usage of water (fertilizer resp) and that is by considering the negative values.(Overall-> negative values tell the farmer to reduce the usage of water or fertilizer per X unit/m^2)

Transition model : The transition model defines how the soil responds to resource applications. It accounts for interdependencies between parameters:

- **Water** increases soil moisture based on soil type.
- As moisture increases, nutrient uptake also increases, improving N, P, K levels.
- Both **water and fertilizer** applications affect pH depending on the source or compound used.

The effect that application of resources does (for example : soil moisture increase) was found by doing some research on how each resource affects the visioned parameters that we are focusing on during our search (sometimes we used functions to calculate the impact, which were based on scientific formulas and research information).

Goal state : To find the optimal goal state for the search algorithms, we calculate the target soil parameters by grouping and filtering data. The dataset is filtered based on three discrete factors: crop label, soil type, and growth stage. We also apply a crop density constraint, selecting only the records where the crop density is within a fixed threshold of the input value. Once filtered, we calculate the average environmental conditions (temperature, humidity, and sunlight exposure) from the selected data, based solely on crop label, soil type, and growth stage. These average environmental conditions are treated as the optimal environment. Each soil moisture value is then individually adjusted by estimating how much it would differ if it were measured under the optimal environmental conditions instead of its original conditions. By doing this, we reduce the standard deviation among the soil moisture values, leading to more consistent and reliable data. After adjustment, we calculate a weighted average where each adjusted value is weighted inversely by its associated water use efficiency (WUE), assigning higher weight to less efficient records. For the other key soil properties (N, P, K, and pH), the mean is calculated directly from the filtered data. These final averaged and weighted values define the target soil conditions that the search algorithm aims to achieve during optimization.

Cost Function : The cost function was designed to penalize inefficient resource usage. It calculates the cost based on how many resources we have used till this moment during optimization, with respect to the availability of those resources , such that when the resource that was used was rare (low availability),then the penalty (cost) would be greater , and vice versa.

Heuristic function : (Admissible) For the heuristic, after a long discussion we have concluded to take as a heuristic **The Sum of the differences** of the key parameters (soil moisture, NPK values and ph level) **of goal state** and those of current state, while considering the priority of each key parameter based on the current Environment and Crop conditions (which are based on a scientific results) .
This heuristic can be easily proved to be Admissible as it is nearly close to the real cost of getting to the goal state which is the difference between the current parameters levels and target levels then it will never overestimate the cost.

Concerning irrigation frequency, after reaching a goal state, the system estimates how often irrigation is needed to maintain optimal soil moisture. It calculates an evaporation factor based on temperature, Then estimates how long the soil takes to dry out back to the initial state, and gets the recommended irrigation frequency per week.

Genetic Search: This method uses the following problem formulation: each chromosome represents a set of actions (amounts of water, nitrogen, phosphorus, and potassium added) applied to the

initial state , which is the same as that of general search. These genes allow for a wider range of values than previous algorithms, enabling more flexible exploration.The process involves three key steps:

Selection: Each individual is evaluated by applying its actions to a copy of the initial state. The cost is calculated using the heuristic function of the global search on the new state resulting from the application of the individual's actions on the copy of the initial state, and fitness is the inverse of this cost. Using roulette wheel selection, individuals with better fitness are more likely to be chosen for reproduction.
Crossover: Two parents are selected, and their genes are randomly combined to form a child—a new action set derived from both parents.
Mutation: One or more genes are randomly adjusted within safe ranges, avoiding overirrigation or overfertilization.

The goal state remains the same as the previous type of search.

Constraint Satisfaction Problem : The aim is to find the suitable amount of water, N, P, K to be added so that the soil moisture, the overall N, P and K fall into their optimal ranges that are calculated from the dataset, ensuring at the same time that we are not exceeding the maximum amount of water and fertilizer that the farmer is willing to provide.

Variables*:* Water, Nitrogen, Phosphorus, and Potassium to add, as well as the maximum amount of water and fertilizer provided by the user (farmer) at a time (in a single irrigation).
Domains: Each variable takes values from 0 to a derived maximum amount.
Constraints: We have unary constraints that ensure soil moisture and pH remain within optimal ranges. And binary constraints link water to each nutrient (N, P, K), requiring their combinations to keep soil moisture and nutrient levels ideal.

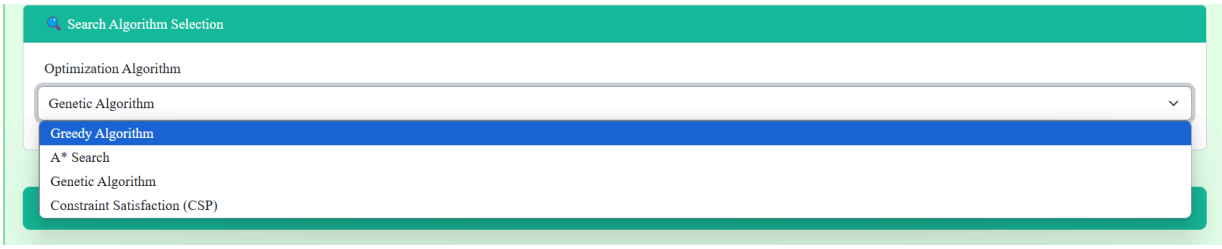
AC-3 Algorithm: We begin by processing arcs like (water_amount, soil_moisture) and (water_amount, Ph), pruning water values that do not satisfy the optimal soil moisture and pH ranges.
-Next, we process arcs such as (water_amount, N), (N, water_amount), (water_amount, P), (P, water_amount), (water_amount, K), and (K, water_amount), where for each variable, we check whether a value satisfies the constraint with at least one value from the connected variable domain, if not, it is pruned from the domain. Whenever pruning changes a domain, we re-add related arcs to the queue to recheck consistency.
-After AC-3, the domains of water_amount, N, P, and K are reduced so that each water value satisfies soil moisture , ph requirements and is compatible with at least one valid combination of nutrients such that we satisfy all the constraints .
Backtracking: After getting the reduced domains, we assign values to water_amount, N, P, and K, one at a time, if one value leads to a conflict, we discard it and move to another one, for example if p = 20 with the current water amount leads to a conflict for example soil moisture constraint not satisfied, we discard p = 20 and try another possible value for it.

- After getting a combination of (water amount, N, P, K) that satisfies all the constraints, we evaluate it based on closeness to the optimal values. The closer it is, the less the cost is. The one

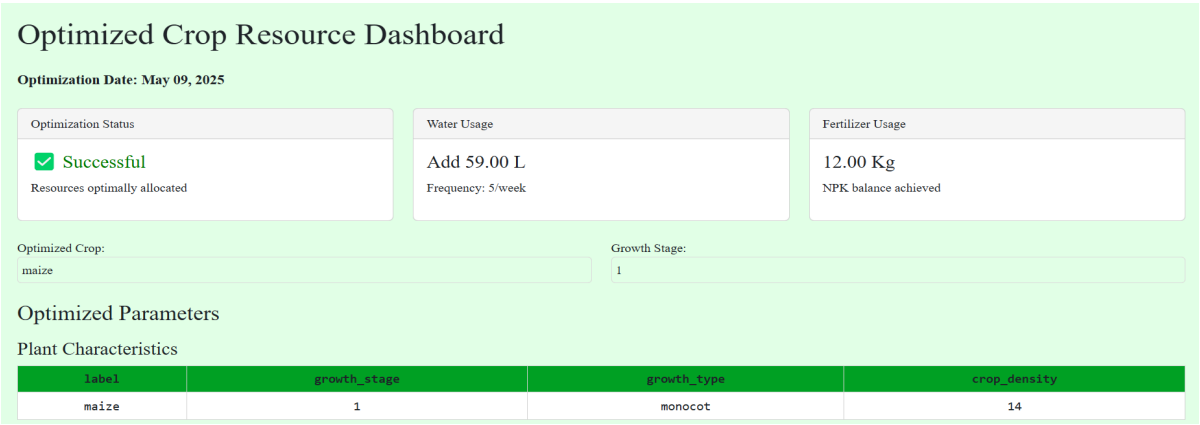
- with the least cost (in the function it is called total_error) we choose it as the best combination and return it.

Application Design

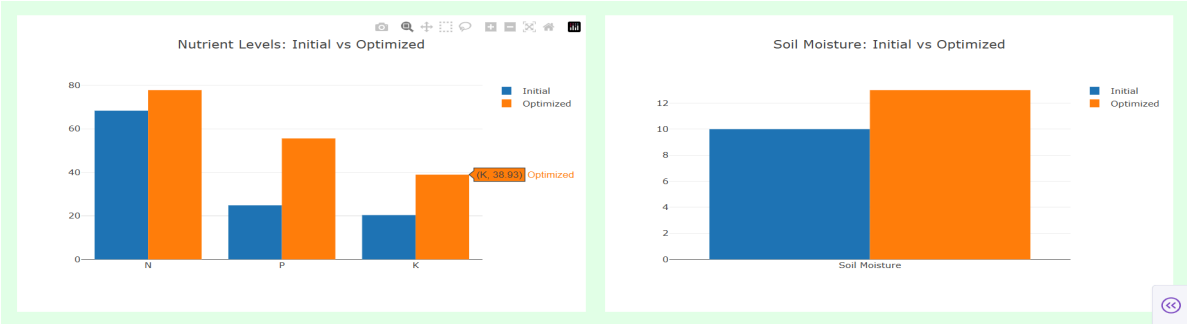
The Smart Farming System website is a user-friendly interface that allows filling a form containing crop parameters, environmental conditions and soil characteristics to recommend the best resource allocation of agricultural supplies efficiently. The user has the option to select a search algorithm (Greedy, A*, Genetic, CSP)



and generate the results by clicking “Run Optimization” which will run the chosen search by invoking the function *get_optimized_state()*, saving it in *optimized_values* and displaying a dashboard describing the results. The dashboard should show whether the algorithm executed with no issue by displaying a message “Successful. Resources optimally allocated”, the amount of each resource allocated and the irrigation frequency besides some plant characteristics obtained from the form.

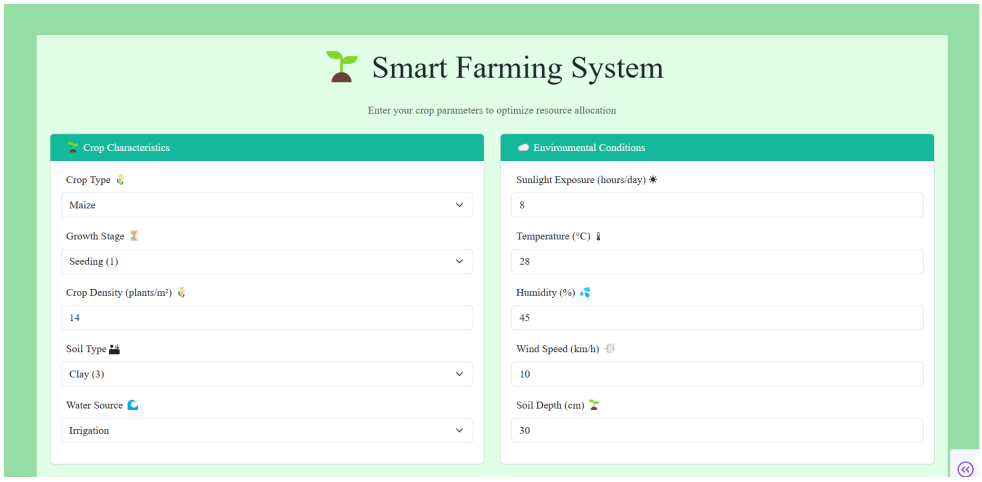


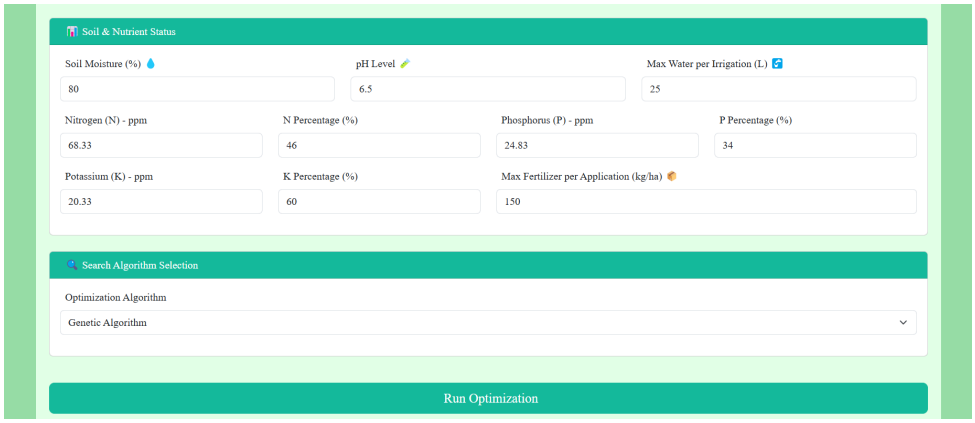
Comparison graphs are also appearing at the bottom of the dashboard to display the difference between the initial state of the crop while using supplies inefficiently and the state after applying the optimal amounts.



Application Modules:

- **Interface (Frontend):** Interactive visualization for the user to input crop parameters through filling a form and running the optimization which will then display a dashboard showing the best resources allocated and comparison charts between initial inputs and optimal states.





- **Data management (Backend):**The system takes user inputs through Dash callback handlers, which save form submissions as Python dictionaries and stores them in a *json* file. After running

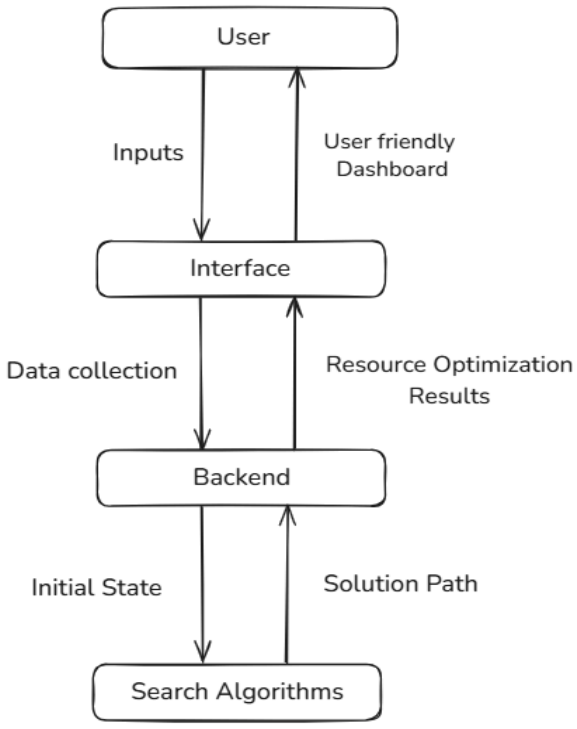
the optimization the user will be redirected to the dashboard webpage which then retrieves the dictionary from the json file and feeds it to the search algorithms and proceeds to display the output.

- **Optimization (Algorithms):** The state dictionary is passed directly to the algorithm. The algorithm then computes the optimal resource allocation by calling each function and returns the results as a state of adjusted inputs, then by calling the *get_optimized_state()* the results will be shaped into a dashboard using Dash and Plotly technologies.

Technologies:

- Dash + (Dash Bootstrap Component): Enables building web applications in python and neat styling while ensuring responsiveness.
- Plotly: Used to create interactive graphs and charts (supports +40 types).
- Python: The main programming language .
- Pandas: For data analysis, handles data processing during the optimization phase.
- CSV: The data storage format of the dataset.

Architectural Diagram of the System



Results and analysis

Evaluation Methods and Performance Metrics (Screenshots are in the appendix A)

To assess the effectiveness of the three search algorithms **Greedy**, **A***, and **Genetic Algorithm** the following evaluation methods and performance metrics were utilized:

1. **Success Rate**: Measures the percentage of runs where the algorithm successfully found a solution.
2. **Average Runtime**: The mean time taken by each algorithm to reach a solution, measured in seconds.
3. **Solution Quality**: The cost associated with the found solution (lower is better).
4. **Search Effort**: Quantified by the number of steps or iterations taken during the search process.
5. **Time vs. Solution Quality**: A scatter plot showing the trade-off between time taken and solution cost.
6. **Runtime Distribution**: A box plot showing the variance in runtime across multiple runs

Performance Evaluation and Comparisons

1. Success Rate

- All algorithms (Greedy, A*, Genetic) achieved a 100% success rate, which indicates high reliability in finding solutions,for this optimization problem, However for some specific inputs and because of the dataset provided to us to use, search may fail (lack of data, thus goal state is not so accurate)

2. Average Runtime

- Greedy was the fastest, with average runtime near 0.2 seconds.
- A* followed with approximately 3.6 seconds.
- Genetic Algorithm was the slowest, averaging 5 seconds.

3. Solution Quality

- Greedy delivered the lowest cost solutions (~78), indicating best quality.
- A* performed moderately (~85).
- Genetic had the worst solution quality (~103).

4. Search Effort

- Greedy and A* required significantly fewer steps (25 and 30, respectively).

- Genetic needed a much larger number of iterations (~90), showing higher computational effort.

5. Time vs. Solution Quality

- Greedy clustered in the lower-left quadrant, signifying low cost and fast time ideal performance.
- A* showed consistent cost and time but higher than Greedy.
- Genetic was scattered in higher cost and time regions, showing inconsistent and suboptimal results.

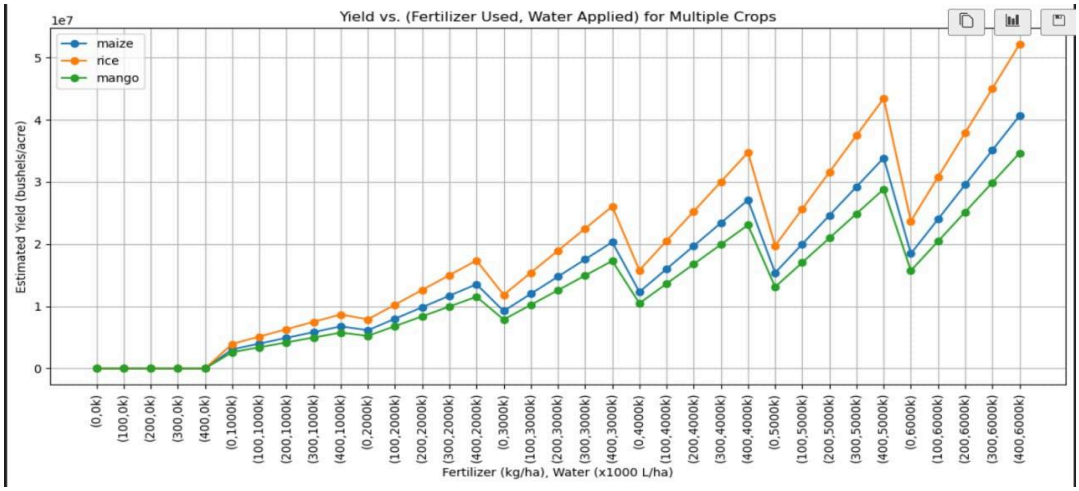
6. Runtime Distribution

- Greedy had the tightest distribution, confirming its speed and consistency.
- A* showed some spread but no extreme outliers.
- Genetic exhibited high consistency in being slow, with very narrow variance around 0.5s.

Key Insights and Conclusions

- **Greedy algorithms** outperform others in **speed**, **solution quality**, and **consistency**.
- **A*** provides a **balance between speed and quality**
- **Genetic Algorithm** performs the worst in **runtime, effort, and quality**, making it less suitable unless problem complexity demands global exploration or multiple optima

Correlation between Crop Yield and Resources applied



Discussion

The results of our system demonstrate the potential of combining AI search and optimization techniques, namely Greedy Search, A*, Genetic Algorithms, and Constraint Satisfaction Problems to manage agricultural resources efficiently and adaptively. Each approach contributed to the system's ability to reduce resource waste while maintaining soil parameters within optimal ranges.

Greedy Search helped with fast decision making, identifying the most pressing resource needs depending on environment and growth stage. However, its lack of cost awareness might lead to less optimal resource allocation. A* incorporated both a heuristic and a cost function that penalized excessive use of resources. This allowed it to balance urgency with cost, resulting in a more realistic and optimized outcome especially with the presence of constraints.

In the Genetic Algorithm, by encoding water and nutrient actions as genes, the system could explore a broader range of possible actions. However, tuning the mutation and crossover rates proved challenging, and the algorithm sometimes converged prematurely or got trapped in local optima. Despite this, it consistently produced valid and cost-effective states when given enough iterations and population diversity.

The CSP algorithm was used to ensure that actions did not violate constraints, such as crop specific thresholds, and. This helped reduce the search space by pruning invalid combinations early. While CSP offered precision, it also introduced complexity and increased computational overhead, considering the domain of actions large or heavily interdependent.

Several limitations remain in the current implementation. Our system operates on preprocessed data and does not yet support real time environmental feedback, which is important for application in dynamic field conditions. Additionally, although the SF24 dataset provides enough data to develop a functional AI system, it is region specific and may not generalize to different climates, soil types, or farming practices.

In future work, we plan to extend the system by integrating live sensor data. Ultimately, our vision is to develop a smart farming assistant capable of continuously optimizing inputs based on real-time conditions, seasonal changes, and evolving crop demands.

Conclusion

The results of our implementation demonstrate that search algorithms provide a promising approach to optimizing agricultural resources. However, the system is not without limitations, potential improvements may be applied to enhance its performance such as using real time sensors.

In future work, we envision developing a fully deployable smart farming assistant that combines real time sensing, smart planning, and a friendly interface for farmers. This would allow for continuous adjustment of resource usage.

References

Online Dataset:

Sikandar. (2024). Smart Farming Data 2024 (SF24) [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/9239304

Csv file link: https://drive.google.com/file/d/1QfG7AMlXATKMYWtwSlrpEG5OG0ShJnOc/view?usp=sharing

Academic Article:

C.Brouwer International Institute for Land Reclamation and Improvement and **M. Heibloem (1987)** Irrigation Water Management: Irrigation Water Needs

Link: https://www.fao.org/4/s2022e/s2022e02.htm

Website:

H. Cilliers, Dr C. Shmidt (2010) . Plant growth stages and nutrient requirements for Optimal yields. Kynoch

Link: https://www.kynoch.co.za/plant-growth-stages-and-nutrient-requirements-for-optimal-yields/

Online Blog:

JFH Horticultural team. (2024). Choosing the Right Fertiliser for Different Plant Growth Stages. JFH Horticultural Supplies

Link: https://www.jfhhorticultural.com/blog/post/choosing-the-right-fertiliser-for-different-plant-growth-stages-

Online Dataset:

STRUCT X TEAM. (2024). Density Ranges for Different Soil Types. STRUCT X

Link: https://www.structx.com/Soil_Properties_002.html

Academic Article:

Scott Gould, AG Technology Consultant (2024). A Guide to Soil Moisture

Link: https://connectedcrops.ca/the-ultimate-guide-to-soil-moisture/

Appendix A

Starting the system: a display of results of data preprocessing + a prompt to the user for input

```
Environmental Averages from Dataset:
  • Rainfall: 144.85
  • Humidity: 92.78
  • Temperature: 38.83
  • Sunlight_Exposure: 9.32

Average Original Soil Moisture: 19.62%

Standard Deviation of Original Soil Moisture: 5.70%

Adjusted Soil Moisture Values:
[17.54621451 29.44136256 16.92058607 19.00078485]

Weighted Optimal Soil Moisture (favoring low WUE): 21.63%
Std Dev (Adjusted): 5.87%
Estimated Optimal Soil Moisture (normal average): 20.73%
Estimated Optimal N    : 56.25
Estimated Optimal P    : 61.50
Estimated Optimal K    : 49.75
```

The Initial State used to test:

```
'label': 'rice', 'soil_type': 1, 'temperature': 16.0, 'crop_density': 12.0, 'humidity': 80.0, 'sunlight_exposure': 8.0, 'water_source': 1, 'wind
_speed': 10.0, 'growth_stage': 1, 'water_availability': 'medium', 'fertilizer_availability': 'medium', 'water_used': 0.0, 'fertilizer_used': 0.0
'max_water_per_irrigation': 100, 'max_fertilizer_per_application': 100, 'soil_depth': 28}
```

The output of the A* search:

```
=== Starting A* Search ===
Initial state: {'soil_moisture': 5.0, 'N': 20.0, 'P': 20.0, 'K': 20.0, 'N_percentage': 0.2, 'P_percentage': 0.2, 'K_percentage': 0.2, 'ph': 6.0
Goal ranges: {'soil_moisture': (13.559315969498371, 18.55931596949837), 'ph': (5.394098996, 7.394098996), 'N': (95.5, 100.5), 'P': (54.5, 59.5)
Solution found after 43113 steps!

Resource Optimization Results:
Optimal amount of water: Add 38.0 L
Optimal amount of fertilizer: Add 25.8 KG
Optimal frequency of irrigation: Irrigate 3 times per week

=== Running BEST_FIRST SEARCH

=== Starting Greedy Best-First Search ===
Initial state: {'soil_moisture': 5.0, 'N': 20.0, 'P': 20.0, 'K': 20.0, 'N_percentage': 0.2, 'P_percentage': 0.2, 'K_percentage': 0.2, 'ph': 6.0
Goal ranges: {'soil_moisture': (13.559315969498371, 18.55931596949837), 'ph': (5.394098996, 7.394098996), 'N': (95.5, 100.5), 'P': (54.5, 59.5)
Solution found after 14 steps.
```

Starting the Greedy Search:

```
=== Running BEST_FIRST SEARCH

=== Starting Greedy Best-First Search ===
Initial state: {'soil_moisture': 5.0, 'N': 40.0, 'P': 40.0, 'K': 40.0, 'N_percentage': 0.2, 'P_percentage': 0.2, 'K
Goal ranges: {'soil_moisture': (13.559315969498371, 18.55931596949837), 'ph': (5.394098996, 7.394098996), 'N': (95.
```

The output of the Greedy Search:

```
Resource Optimization Results:
Optimal amount of water: Add 38.0 L
Optimal amount of fertilizer: Add 25.8 KG
Optimal frequency of irrigation: Irrigate 3 times per week
```

The output of the GA Search:

```
--- OPTIMIZATION Test: Selection=tournament, Crossover=order, Mutation=swap ---
Generation 0: Best cost = 3.7903864326519257

Resource Optimization Results:
Optimal amount of water: Add 53.77 L
Optimal amount of fertilizer: Add 25.84 KG
Optimal frequency of irrigation: Irrigate 3 times per week
Result:  combination = {'K_added': 17.24472655950875, 'N_added': 75.47871622731225, 'water_added': 53.77434145544742, 'P_added': 36.49655298757

--- OPTIMIZATION Test: Selection=roulette, Crossover=order, Mutation=swap ---
Generation 0: Best cost = 6.583545635609951

Resource Optimization Results:
Optimal amount of water: Add 49.3 L
Optimal amount of fertilizer: Add 25.43 KG
Optimal frequency of irrigation: Irrigate 3 times per week
Result:  combination = {'P_added': 34.10518739996251, 'N_added': 75.84404036727547, 'K_added': 17.221251307612985, 'water_added': 49.3011274618
```

The output of the CSP:

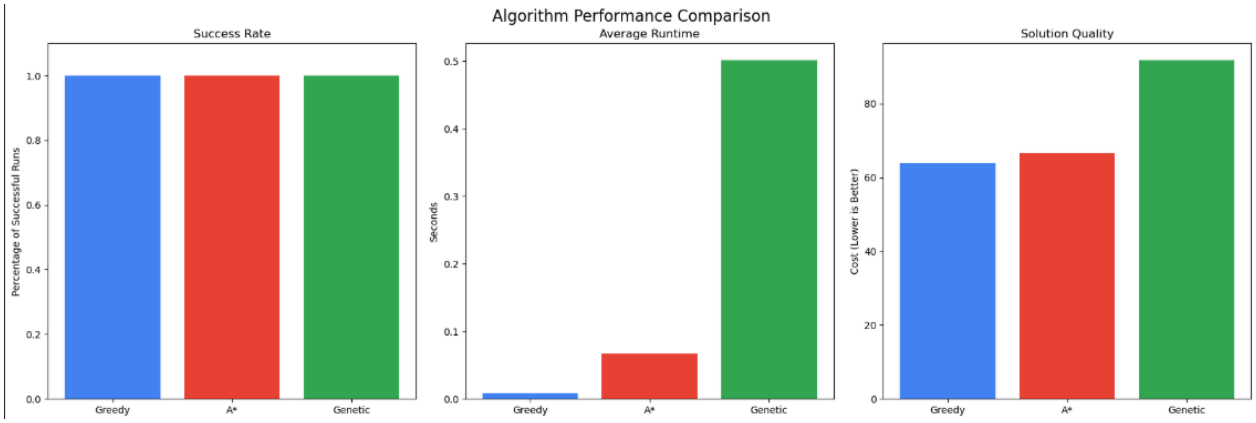
```
Starting AC-3 algorithm...
After pruning by soil moisture and pH constraints:
Water domain: [43.800000000000004, 43.900000000000006, 44.0, 44.1, 44.2, 44.300000000000004, 44.400000000000006, 44.5, 44.6, 44.7, 44.800000000


Final Domains after AC-3 pruning:
Water: [43.800000000000004, 43.900000000000006, 44.0, 44.1, 44.2, 44.300000000000004, 44.400000000000006, 44.5, 44.6, 44.7, 44.800000000000004,
N: [73.10000000000001, 73.2, 73.3, 73.4, 73.5, 73.60000000000001, 73.7, 73.8, 73.9, 74.0, 74.10000000000001, 74.2, 74.3, 74.4, 74.5, 74.6000000
P: [32.7, 32.800000000000004, 32.9, 33.0, 33.1, 33.2, 33.300000000000004, 33.4, 33.5, 33.6, 33.7, 33.800000000000004, 33.9, 34.0, 34.1, 34.2, 3
K: [14.4, 14.5, 14.600000000000001, 14.700000000000001, 14.8, 14.9, 15.0, 15.100000000000001, 15.200000000000001, 15.3, 15.4, 15.5, 15.60000000


Resource Optimization Results:
Optimal frequency of irrigation: Irrigate 3 times per week
Optimal amount of water: 43.800000000000004 L
Optimal amount of fertilizer : 27.18 Kg
result combination : {'Water': 43.800000000000004, 'N': 78.4, 'P': 37.9, 'K': 19.6}
```

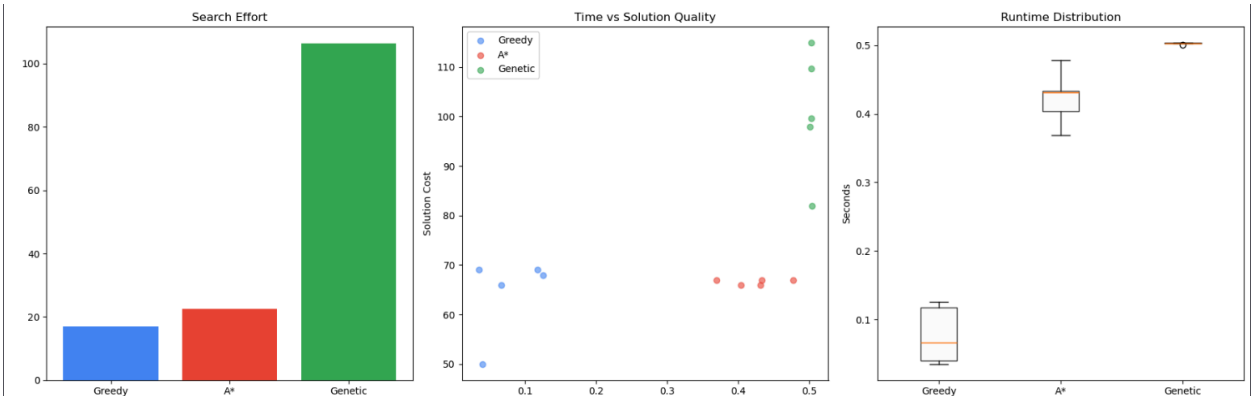
Displaying one trial of the performance testing:

```
Running performance comparison with 5 trials per algorithm...

Trial 1 of 5

=== Starting Greedy Best-First Search ===
Initial state: {'soil_moisture': 10.531246773627798, 'N': 40, 'P': 15, 'K': 100, 'N_percentage': 0.2, 'P_percentage': 0.5, 'K_percentage': 0.1,
Goal ranges: {'soil_moisture': (19.131246773627797, 24.131246773627797), 'ph': (5.60993806025, 7.60993806025), 'N': (53.75, 58.75), 'P': (59.0,
Solution found after 18 steps.
Greedy: Success (Time: 0.01s)

=== Starting A* Search ===
Initial state: {'soil_moisture': 10.531246773627798, 'N': 40, 'P': 15, 'K': 100, 'N_percentage': 0.2, 'P_percentage': 0.5, 'K_percentage': 0.1,
Goal ranges: {'soil_moisture': (19.131246773627797, 24.131246773627797), 'ph': (5.60993806025, 7.60993806025), 'N': (53.75, 58.75), 'P': (59.0,
Solution found after 111 steps!
A*: Success (Time: 0.05s)
Running genetic algorithm...
Genetic: Success (Time: 0.50s)
```

```
=== Final Results ===

Greedy Algorithm:
- Success rate: 5 out of 5 runs
- Average steps: 17
- Average time: 0.01 seconds
- Average solution cost: 56

A* Algorithm:
- Success rate: 5 out of 5 runs
- Average steps: 18
- Average time: 0.04 seconds
- Average solution cost: 56

Genetic Algorithm:
- Success rate: 5 out of 5 runs
- Average iterations: 101
- Average time: 0.50 seconds
- Average solution cost: 101

Generating performance graphs...
```

Some graphs displaying a comparison between performances:




Search Effort
0
20
40
60
80
100
Greedy
A*
Genetic
Time vs Solution Quality
Greedy
A*
Genetic
Solution Cost
50
60
70
80
90
100
110
0.1
0.2
0.3
0.4
0.5
Runtime Distribution
Seconds
0.1
0.2
0.3
0.4
0.5
Greedy
A*
Genetic

Appendix B

Member	Task completed
SMAIL Imadeddine	Problem formulation for global search - initial state definition - priorities determination - transition model implementation Problem formulation for genetic algorithm: - evaluate function implementation - initialize population function Implementation Problem formulation for csp - water_moisture_constraint function - ac3 function implementation Visual representation of water and fertilizer usage across different crops, showing the impact on crop yield. Report writing
GARAMIDA Abderraouf Adel	Problem formulation for global search Research for scientific formulas - heuristic function implementation - get_valid _actions function implementation - cost function implementation Problem formulation for genetic algorithm - selection function implementation (roulette wheel and tournament selection) Problem formulation for csp - water_and_nitrogean constraint function - ac3 function implementation Performance graphs comparing the efficiency of Greedy, A*, and GA-based methods for resource optimization. Report writing
CHERGUI Mohamed Bahae Eddine	Problem formulation for global search - goal state implementation (including data preprocessing) - goal check function implementation - calculate best_irrigation_frequency Function implementation Problem formulation for genetic - crossover function implementation (order_crossover) Problem formulation for csp

	- water and phosphore constraint function - ac3 function implementation Report writing
BRAHIMI Aya	Problem formulation for global search - apply_action function implementation - calculate drought_factor function Implementation - a* and greedy search algorithm Implementation Problem formulation for genetic - mutation function implementation Problem formulation for csp - water and potassium constraint function - backtrack function implementation A chart or dashboard showing the optimized resource allocation for each crop and growth stage. Report writing
AMMEDAH Mohamed	Problem formulation for global search - calculate_moisture increase function Implementation - get_availibility level function Implementation - reconstruct path function implementation Problem formulation for genetic - evolve population function implementation - solve function implementation Csp problem formulation - water and ph constraint function - backtrack function implementation Report writing
SAAD Mohammed El Amine	Implemented The platform