AI420 EVOLUTIONARY ALGORITHMS SPRING 2025

COURSE PROJECT INSTRUCTIONS

Instructions to Students:

- This is a group work project. Each group consists of **Four to Five students** *(the Teaching Assistant must approve group members through registration)*. Each group must develop the idea assigned to them using Python.
    - **Project Objectives:** The objectives of this project can be summarized as applying the main ideas, fundamental concepts, and basic algorithms in the field of Evolutionary Algorithms & Computing.
- **Submission:** Submission is done according to the following schedule:
    - **Saturday, May 11th 2024: Submission and Discussion of the** (1) **Project** and (2) **Documentation**. *The report should include the following: (1) Project idea in detail, (2) Main functionalities, (3) Similar applications in the market, (4) A literature review of Academic publications (papers) relevant to the idea (at least 4 to 6 papers, as per the number of team members), (5) the Dataset employed (preferably a publicly available dataset), (6) Details of the algorithm(s)/approach(es) used and the results of the experiments, and (7) Development platform.*
- **Assessment:** Assessment will be on the reports, code submitted, and discussions with team members. All the team members must contribute to all the phases, and the role of each member must be clearly stated in each report.
    - The Project will be assessed based on the following criteria:
        - The complexity of the problem, & the correctness of the algorithms employed.
        - The quality/comprehensiveness of your experiments & documentation.
        - The correctness of your analysis and design diagrams.
        - Implementation correctness.
- **Feedback:** If requested, further details and feedback could be provided for each group through discussions with the teaching assistant(s) during the weekly-labs/office-hours.
- You can only submit your work. Any student suspected of plagiarism will be subject to the procedures set out by the Faculty/University (including failing the course entirely).
    - **Academic Integrity:** The University's policies on academic integrity will be enforced on students who violate University standards of academic integrity. Examples of behaviour that is not allowed are:
        - Copying all or part of someone else's work and submitting it as your own;
        - Giving another student in the class a copy of your work and
        - Copying parts from the internet, textbooks, etc.
        - If you have any questions concerning what is allowed, please don't hesitate to discuss them with me.
    - **N.B., We understand that you might positively refer to AI tools to support your research. Please note that, in case there are any concerns about AI-generated submission content, you will be invited for an opportunity to verify and defend your work.**

AI420 Evolutionary Algorithms Spring 2025

Course Project Descriptions (14 Ideas)

Table of Contents

Important Guidelines for ALL Projects to follow:

a) Clearly define and formalize your problem as one of the problem types we've studied throughout this module: Optimization, Modelling, Simulation, Constraint Satisfaction, Free Optimization, Constrained Optimization, etc.

b) If your selected idea mandates Constraint Handling, clearly implement an approach to handling constraints (i.e., Penalty Functions, Repair Functions, Restricting Search to the Feasible Region, Decoder Functions, etc.).
c) If your selected idea mandates Coevolution, clearly implement a coevolutionary approach (cooperative or competitive).
d) Clearly define the Components of your Evolutionary Algorithm: For instance, in the case of a genetic algorithm, clearly define the Representation (Definition of Individuals), Evaluation Function (Fitness Function), Population, Parent Selection Mechanism, Variation Operators (Mutation and Recombination), Survivor Selection Mechanism (Replacement), Initialisation, and Termination Condition(s).
e) Select variation operators (mutation and recombination) suitable to the selected representation.
   - If possible, use at least 2 parent selection techniques (each independently) and report the results for each.
   - If possible, use at least 2 recombination techniques (each independently) and report the results for each.
   - If possible, use at least 2 mutation techniques (each independently) and report the results for each.
f) If possible, use at least 2 population-management-models/survivor-selection (each independently) and report the results for each.
g) Clearly describe and implement approaches to control/tune the parameters.
h) Describe and implement a suitable approach for preserving diversity (i.e., Fitness Sharing, Crowding, Automatic Speciation Using Mating Restrictions, Running Multiple Populations in Tandem such as the Island Model EAs, Spatial Distribution within One Population such as Cellular EAs, etc.).
i) Incorporate a functional user interface demonstrating the algorithm, parameters, inputs, and results.
j) The students may be awarded bonus marks in the following cases (only if the experiments are carried out properly and the results/performance were measured, reported, and analysed adequately:
   - Investigating the effect of multiple (at least 2) representations (when possible).
   - Investigating the effect of multiple (at least 2) initialization approaches (when possible).
   - Investigating the effect of over-selection for large populations (when possible).
   - An educational visual interface that illustrates (simulates) the changes in the evolutionary process and solutions when different parameters/options/approaches are selected. I.e., an interface/simulation that can be later used to teach students the effects of varying selected approaches/parameters, .. etc.
   - Contributions (experiments and results) with high publication potential.
   - Hybrid approaches (employing together more than a single EAs/SI approach).
   - Employing SOTA novel variants of the EAs/Si approaches rather than the traditional implementations.
k) Report the results (independently) for each setting (e.g., a choice of a mutation operator, a recombination operator, a representation, a parent selection approach, a survivor selection approach, an initialisation, an approach for preserving diversity, .. etc.).
   - The evolution should be carried out multiple times (optimally, 30 runs per setting).
   - The list of seeds (used to initialise the random number generator before each run) should be stored & provided.

[1] Adaptive University Timetabling Optimization using Particle Swarm Optimization (PSO) and/or Genetic Algorithms [Preferably in combination with other EAs/SI approaches of your choice]:

- Define the scheduling constraints (e.g., classroom availability, lecturer assignments, course times).
- Research basic PSO principles and map each "particle" to a candidate timetable (i.e., model the timetable as a set of particles where each particle represents a potential schedule).
- Apply PSO to evolve these schedules by updating particles' positions based on best-found configurations.
- Implement constraint handling (penalty functions, repair functions) to ensure no conflicts.
- Simulate several runs and document how parameters (swarm size, inertia weight, etc.) affect outcomes.
- Compare your approach with other SOTA approaches.
- Develop a simple user interface that visualises the timetable evolution and final schedule.
- Document all settings, parameters, and performance comparisons.

Detailed Description:

- **Context and Problem Statement:** University timetabling involves assigning courses to available time slots and classrooms while satisfying constraints such as room capacity, lecturer availability, and course conflicts. This problem is NP-hard and common in academic scheduling.
- **Key Terms and Concepts:**
  - *Particle Swarm Optimization (PSO):* A method that mimics the social behavior of birds/fish to find optimal regions in a search space.
  - *Timetabling:* Creating schedules that satisfy all necessary conditions.
  - *Constraints:* Conditions (like room availability) that must be met in the schedule.
- **Requirements/Deliverables:**
  - A working algorithm that produces valid timetables (a fully functional Python program that generates university timetables).
  - A report comparing parameter settings and performance over multiple runs (e.g., with at least 30 runs per setting). Include visualisations that show the evolution of timetables over iterations.
  - A user interface that displays evolving timetables.
  - Documented experimental data, results, and insights into the effect of parameter tuning.

[2] Exam Timetabling Optimization using Genetic Algorithms and/or Ant Colony Optimization (ACO) [Preferably in combination with other EAs/SI approaches of your choice]:

- Identify exam scheduling constraints (room capacities, student conflicts, exam durations).
- List exam scheduling requirements, such as avoiding student exam clashes and spreading out challenging subject combinations.
- Represent the scheduling problem as a graph where nodes represent exam slots and edges reflect transitions (or design a genetic algorithm where each individual represents an exam timetable).
- Define fitness functions based on the number of conflict-free schedules and balanced exam spacing. Or use ACO to simulate "ants" exploring possible scheduling solutions by laying pheromone trails.
- Experiment with different pheromone evaporation rates and heuristic functions, or Experiment with different parent selection, crossover, and mutation techniques.
- Build a basic visualisation to display the generated timetables.
- Deliver a final timetable, experimental performance data, and a detailed report.

Detailed Description:

Context and Problem Statement:

- The Exam Timetabling Problem requires scheduling examinations with overlapping student groups, limited exam venues, and time constraints. Finding an optimal schedule that minimises conflicts and resource overuse is challenging.

Key Terms and Concepts:

- *Ant Colony Optimization (ACO):* A swarm-based technique inspired by how ants find the shortest path using pheromone trails.
- *Pheromone:* A virtual marker that helps guide future searches toward promising solutions.
- *Heuristic:* A simple, practical method to assist decision-making (e.g., prioritising exams with high conflict rates).
- *Genetic Algorithm (GA):* A method mimicing natural selection where candidate solutions (individuals) evolve through selection, crossover, and mutation.
- *Fitness Function:* A measure that evaluates how well an exam timetable meets all constraints (e.g., no overlapping exams, balanced scheduling).
- *Population and Generations:* A collection of candidate schedules evolved over successive iterations.

Requirements/Deliverables:

- Implementation of an ACO/GA algorithm to generate valid exam timetables.
- A Python-based application to generate and refine exam timetables.
- A step-by-step report explaining the algorithm, choice of parameters (including representation and fitness function), and results.
- Simulations showing the effect of various parameters (e.g., different evaporation rates) and/or different GA operators (selection, mutation, crossover).
- A user-friendly interface that explains how the schedule evolves over time.
- A final report with visual results, a literature review section, and suggestions for practical extension.

[3] Green Delivery Route Optimization using Hybrid Genetic and Differential Evolution Algorithms (Preferably for Electric Vehicles) [Preferably in combination with other EAs/SI approaches of your choice]:

- Formulate the delivery routing problem with objectives (minimize travel distance and fuel consumption).
- Define the representation for routes (e.g., sequences of stops). Also include stops for battery recharge for electric vehicles.
- Implement both a genetic algorithm and differential evolution to solve the problem.
- Compare performance metrics such as cost reduction and time efficiency.
- Create a user interface to display routes over a map outline (simulated).
- You may employ other EAs/SI approaches instead of GAs and/or DE.

Detailed Description:

Context and Problem Statement: Efficient route planning plays a significant role in reducing emissions and costs in delivery services. This project applies hybrid evolutionary methods to optimize delivery routes in a "green" context.

Key Terms and Concepts:

- *Differential Evolution (DE):* An algorithm that creates new candidate solutions by combining existing ones using weighted differences.
- *Hybrid Approach:* Combining the strengths of two or more algorithms to improve performance.
- *Route Representation:* Encoding a series of stops as a candidate solution.

Requirements/Deliverables:

- Code that implements both GA and DE for a delivery routing problem with fuel efficiency considerations.
- Comparative analysis highlighting strengths and weaknesses of each approach.
- A detailed report including experiments, parameter studies, and visual maps of optimized routes.

[4] Traffic Flow Optimization on Urban Roads Using Differential Evolution [Preferably in combination with other EAs/SI approaches of your choice]:

- Model an urban traffic network with intersections and roads.
- Define optimization objectives such as minimizing overall travel time or congestion.
- Use Differential Evolution (DE) to optimize traffic signal timings at intersections.
- Experiment with variations in mutation strategies and crossover techniques.
- Validate the solution with simulated traffic scenarios and compare performance.
- You may employ other EAs/SI approaches instead of DE.

Detailed Description:

Context and Problem Statement: Urban traffic management is a significant challenge. Optimizing traffic light timings can reduce congestion and minimize travel delays, contributing to improved urban mobility.

Key Terms and Concepts:

- *Differential Evolution (DE):* An evolutionary algorithm that uses vector differences for mutation to explore the solution space effectively.
- *Mutation and Crossover:* Techniques used in evolutionary algorithms to create new candidate solutions.
- *Simulation:* Creating a virtual model to test and evaluate performance.

Requirements/Deliverables:

- A DE-based algorithm to optimize traffic light timings.
- Simulated traffic scenarios showing improvements in flow and reduced congestion.
- A report detailing algorithm design, experiments (with at least 30 runs per setting), and comparative analysis of parameter effects.
- A working simulation interface demonstrating before-and-after optimization performance.

[5] Feature Selection in Medical Image Analysis Using Genetic Algorithms [Preferably in combination with other EAs/SI approaches of your choice]:

- Identify a publicly available medical imaging dataset (e.g., for diagnosing diseases).
- Define an objective function for feature selection that maximizes classification accuracy while reducing feature count.
- Implement a Genetic Algorithm (GA) where each individual represents a feature subset.
- Run experiments comparing different selection and crossover strategies on the dataset.
- Summarize findings with charts, accuracy metrics, and a final report.
- You may employ other EAs/SI approaches instead of GAs.

Detailed Description:

Context and Problem Statement: In medical image analysis, reducing the number of features (or image characteristics) can simplify models and improve performance. Selecting the best features is critical for accurate diagnosis.

Key Terms and Concepts:

- *Feature Selection:* The process of choosing a subset of relevant features (variables) for model training.
- *Genetic Algorithm (GA):* An algorithm that evolves candidate solutions using techniques inspired by biological evolution.
- *Objective Function:* In this case, it balances high accuracy and lower complexity.

Requirements/Deliverables:

- A GA implementation for selecting a meaningful subset of image features.
- A detailed comparison of different GA operators (selection, crossover, mutation).
- Performance metrics (accuracy, reduction percentage) across multiple runs.
- A comprehensive report with visual aids (graphs and tables) showing algorithm performance.

[6] Optimized Waste Collection Routing in Cities Using Ant Colony Optimization [Preferably in combination with other EAs/SI approaches of your choice]:

- Model a city map with nodes representing waste collection points.
- Define an objective function that minimizes the distance traveled while covering all collection points.
- Apply Ant Colony Optimization (ACO) to simulate routes and update pheromone trails for efficient path discovery.
- Test your algorithm with various city layouts and constraints (e.g., time windows for waste collection).
- Present the best route planning solutions and report findings with performance metrics and visual maps.
- You may employ other EAs/SI approaches instead of the ACO.

Detailed Description:

Context and Problem Statement: Urban waste collection is a significant logistical challenge. Efficient routing can reduce operating costs and environmental impact.

Key Terms and Concepts:

- *Waste Collection Routing:* Planning routes to collect waste from multiple locations in an optimized manner.
- *Ant Colony Optimization (ACO):* A method where artificial "ants" traverse a graph to find optimal paths based on pheromone intensity.
- *Objective Function:* A function that quantifies the travel distance and route efficiency.

Requirements/Deliverables:

- Develop an ACO-based algorithm for planning efficient waste collection routes.
- Simulate several scenarios with different city maps and collection constraints.
- Generate a report including maps, route efficiency metrics, and comparisons across different parameter settings.
- Provide a user interface that visualizes the city map and the computed routes.

[7] Disaster Relief Resource Allocation Using Particle Swarm Optimization [Preferably in combination with other EAs/SI approaches of your choice]:

- Define a disaster relief scenario where resources (like food, water, medical supplies) need optimal distribution.
- Develop an objective function balancing speed of distribution and resource utilization efficiency.
- Use Particle Swarm Optimization (PSO) to decide on resource allocation among different regions.
- Simulate various disaster scenarios and document the efficiency of the allocation.
- Compare results across different PSO configurations and produce detailed performance reports.
- You may employ other EAs/SI approaches instead of the PSO.

Detailed Description:

Context and Problem Statement: In disaster relief, efficient allocation of resources can save lives. The challenge lies in distributing limited resources to where they are needed most.

Key Terms and Concepts:

- *Resource Allocation:* The process of distributing resources optimally in a constrained environment.
- *Particle Swarm Optimization (PSO):* Utilized here to explore optimal allocation strategies in a multidimensional space.
- *Simulation:* Creating virtual disaster scenarios to test resource allocation efficiency.

Requirements/Deliverables:

- A PSO-based solution to allocate relief resources efficiently.
- A set of simulations showing algorithm performance under different disaster conditions.
- A final report with performance metrics, visual representations of resource allocation, and parameter study results (e.g., convergence curves).
- A demonstration of the algorithm via a simple interface that allows manipulation of input parameters.

[8] Smart City IoT Sensor Placement Optimization via Hybrid Evolutionary Strategies [Preferably in combination with other EAs/SI approaches of your choice]:

- Define the problem of placing a limited number of IoT sensors in a city to maximize coverage and data collection.
- Develop a hybrid evolutionary algorithm combining Genetic Algorithms and Differential Evolution to select optimal sensor locations.
- Simulate sensor coverage using a model city grid.
- Compare different evolutionary strategies by running multiple trials and documenting improvements.
- Present visual maps showing sensor placements and summary statistics of area coverage.
- You may employ other EAs/SI approaches instead of GAs and/or DE.

Detailed Description:

Context and Problem Statement: In smart cities, the strategic placement of IoT sensors is crucial for effective monitoring of urban environments. The goal is to cover as much area as possible using limited sensors while minimizing redundancy.

Key Terms and Concepts:

- *IoT Sensors:* Devices that collect and transmit data on various urban metrics (e.g., air quality, traffic density).
- *Hybrid Evolutionary Strategy:* Combining features of different algorithms (GA and DE) to improve search efficiency.
- *Coverage:* The proportion of the city area effectively monitored by sensors.

Requirements/Deliverables:

- A hybrid algorithm that optimizes sensor placement across a city grid.
- Detailed simulations comparing different algorithm components (e.g., mutation strategies, crossover types).
- Visual outputs such as sensor distribution maps and coverage heatmaps.
- A final report discussing algorithm effectiveness and statistical analysis over multiple runs (ideally 30 runs per scenario).

[9] Cloud Resource Allocation using a Hybrid GA–ACO Approach [Preferably in combination with other EAs/SI approaches of your choice]:

- Model a simple cloud environment with a set of tasks and available computing resources.
- Define objectives like minimal cost or response time.
- Implement a hybrid approach: use GA for candidate solution generation and ACO for fine-tuning allocation paths.
- Test the approach on standard workload or simulated data.
- Evaluate the efficiency and robustness of the resource allocation strategy.

Detailed Description:

Context and Problem Statement: Cloud computing demands efficient resource allocation to balance load and reduce operational costs. This project uses evolutionary algorithms to devise a dynamic resource allocation strategy.

Key Terms and Concepts:

- *Resource Allocation:* Assigning computing resources to tasks in an optimal way.
- *Hybrid Algorithms:* Combining methods such as GA and ACO to exploit the benefits of both randomness and collective behavior.
- *Fitness Evaluation:* Assessing solutions based on cost and performance metrics.

Requirements/Deliverables:

- A Python simulation that models cloud tasks and resource constraints.
- An algorithm that integrates GA and ACO with clearly defined evaluation metrics.
- Experimental data and plots demonstrating how the hybrid model outperforms standard models.
- A final report detailing methods, experiments, and potential real-world applications.

[10] Smart Warehouse Layout Optimization using Ant Colony Optimization (ACO) [Preferably in combination with other EAs/SI approaches of your choice]:

- Simulate a warehouse with defined zones for storage, packing, and shipping.
- Represent warehouse layout options as potential solutions.
- Implement ACO to iteratively improve layout by mimicking ant foraging behavior.
- Define cost functions based on travel distance, picking times, and congestion.
- Include sensitivity analysis for different warehouse sizes or product mixes.
- You may employ other EAs/SI approaches instead of the ACO.

Detailed Description:

Context and Problem Statement: Optimizing warehouse layouts is essential for reducing operational costs and improving logistics efficiency. This project simulates the warehouse environment and applies ACO to find cost-effective layouts.

Key Terms and Concepts:

- *Ant Colony Optimization (ACO):* A method inspired by the foraging behavior of ants, where "pheromones" guide the search towards optimal paths (in this case, layouts).
- *Layout Optimization:* The process of arranging warehouse zones to reduce travel time and increase efficiency.
- *Cost Function:* A mathematical formulation that measures the quality of a warehouse layout based on key performance factors.

Requirements/Deliverables:

- A simulation model that defines a warehouse floor plan and evaluates layouts.
- Implementation of an ACO algorithm that iteratively refines layout designs.
- Comparative experiments showing performance improvements and scalability analysis.
- Documentation and visualizations (e.g., heatmaps and layout diagrams).

[11] Traffic Signal Timing Optimization using Particle Swarm Optimization (PSO) [Preferably in combination with other EAs/SI approaches of your choice]:

- Model a simplified urban traffic network with multiple intersections.
- Define control variables (signal timings) and objectives (minimizing wait times, congestion).
- Use PSO to explore the space of signal timing configurations.
- Simulate traffic flows using time-step updates and evaluate performance.
- Produce visual output demonstrating traffic improvements as signals adjust.
- You may employ other EAs/SI approaches instead of the PSO.

Detailed Description:

Context and Problem Statement: Optimizing traffic signals improves urban mobility by reducing delays and fuel consumption. This project leverages PSO to dynamically adjust signal timings in a simulated city environment.

Key Terms and Concepts:

- *Traffic Signal Optimization:* Adjusting the timing of traffic lights to improve flow.
- *Particle Swarm Optimization (PSO):* Uses simple "particles" (candidate solutions) that move within a search space influenced by personal and group bests.
- *Objective Functions:* Criteria such as average waiting time and number of stops that measure traffic performance.

Requirements/Deliverables:

- A Python simulation that represents an intersection network along with adjustable signals.
- An implementation of PSO to optimize the timing settings.
- Recorded results comparing baseline signal settings with optimized ones.
- A final report with charts, simulation videos/screenshots, and discussion of parameter effects.

[12] Neural Evolution for Handwritten Digit Recognition using Differential Evolution (DE) [Preferably in combination with other EAs/SI approaches of your choice]:

- Choose a simple neural network architecture for digit classification (e.g., on the MNIST dataset).
- Use Differential Evolution to optimize the network's weights and (optionally) its structure.
- Define performance metrics such as classification accuracy and loss reduction.
- Compare results with standard backpropagation methods.
- Create a visual dashboard showing learning curves and sample predictions.
- You may employ other EAs/SI approaches instead of DE.
- You may evolve Neural Networks for other applications/tasks (instead of Handwritten Digit Recognition).

Detailed Description:

Context and Problem Statement: Evolving neural network parameters using DE offers an alternative to traditional training methods. This project tests whether evolutionary methods can match or outperform standard algorithms on a well-known task—recognizing hand-written digits.

Key Terms and Concepts:

- *Differential Evolution (DE):* An optimization technique that improves candidate solutions by adding weighted differences between solutions.
- *Neural Network:* A computational model inspired by the human brain used for pattern recognition.
- *MNIST Dataset:* A standard dataset used for training and testing image recognition systems.

Requirements/Deliverables:

- A Python implementation that integrates DE to adjust neural network weights.
- Comparative analysis on performance against traditional training methods.
- Graphs and visualizations of model accuracy and error curves.
- A written report detailing methodology, experiments, and potential enhancements.

[13] Sports Tournament Scheduling via Genetic Algorithms (GA) [Preferably in combination with other EAs/SI approaches of your choice]:

- Represent tournament schedules (e.g., round-robin or knockout formats) as individuals in the GA population.
- Define objectives to minimize venue conflicts, ensure fair rest periods, and balance game times.
- Develop fitness functions to score schedules based on constraint violations.
- Implement genetic operators (crossover, mutation) tailored to scheduling order.
- Validate the approach by comparing generated schedules with a baseline solution.
- You may employ other EAs/SI approaches instead of GAs.

Detailed Description:

Context and Problem Statement: Organizing sports tournaments involves managing multiple constraints like venue availability, rest periods, and game order. Using GA, teams can evolve schedules that satisfy these constraints in an optimal manner.

Key Terms and Concepts:

- *Tournament Scheduling:* The process of planning matches such that all teams play under fair conditions.
- *Genetic Algorithm (GA):* An optimization method inspired by the process of natural selection.
- *Fitness Evaluation:* A way to quantify how well a given schedule meets requirements.

Requirements/Deliverables:

- A prototype Python tool that generates and refines tournament schedules using GA.
- Simulation results comparing different GA parameter settings.
- Visual scheduling charts (e.g., calendars, tables) to illustrate the final schedule.
- A report discussing challenges, solutions, and possible real-world applications.

[14] Adaptive Recommendation Engine using Coevolutionary Algorithms [Preferably in combination with other EAs/SI approaches of your choice]:

- Formulate a recommendation problem (e.g., movies, books, or music) using simple user–item rating data.
- Implement a coevolutionary algorithm where two sub-populations (e.g., users and items) evolve in parallel.
- Define fitness functions that measure recommendation accuracy or user satisfaction.
- Experiment with different coevolutionary strategies (competitive or cooperative).
- Build a simple interface to allow users to see recommendations and experiment with parameters.

Detailed Description:

Context and Problem Statement: Recommendation systems are used in many platforms to suggest products or media to users. In this project, students use coevolution—where two interacting populations evolve simultaneously—to design a dynamic, adaptive recommendation engine.

Key Terms and Concepts:

- *Coevolutionary Algorithms:* Techniques where multiple populations evolve together, influencing each other's evolution.
- *Recommendation Engine:* A system that suggests items to users based on various factors and learned patterns.
- *Fitness Function:* A measure that evaluates the accuracy and relevance of the recommendations generated.

Requirements/Deliverables:

- A Python-based recommendation system that implements coevolution of users and items.
- Experimentation with different parameter settings and strategies, including a clear comparison of outcomes.
- A user interface (even if basic) where recommendations can be visualized and tested.
- A final report detailing the methodology, experimental results, and potential for future improvements.