Project Proposal

Solving the N-Queens Problem Using Search Algorithms

1. Project Title

Solving the N-Queens Problem Using Classical and Informed Search Algorithms

2. Project Overview

This project focuses on solving the N-Queens problem using multiple artificial intelligence search algorithms. The N-Queens problem is a classical constraint satisfaction problem where the goal is to place N queens on an N×N chessboard such that no two queens threaten each other.

The project aims to demonstrate how different search strategies explore the state space, handle constraints, and differ in terms of performance, efficiency, and optimality.

3. Problem Description

The N-Queens problem requires placing N queens on a chessboard in such a way that:

- No two queens share the same row
- No two queens share the same column
- No two queens share the same diagonal

In this project, N is fixed to 5 to allow consistent evaluation across all algorithms.

4. Why This Problem?

The N-Queens problem was selected because it is a well-known constraint satisfaction problem in artificial intelligence. It clearly demonstrates the strengths and weaknesses of different search algorithms and allows easy comparison between uninformed and informed search strategies.

5. State Representation

Each state is represented as a list of integers where the index represents the column number and the value represents the row number of the queen in that column.

Example:
[0, 2, 4]

This representation ensures that only one queen is placed in each column, automatically eliminating column conflicts.

6. Successor Function

Successor states are generated by placing a queen in the next column, trying all possible row positions, and accepting only positions that do not violate row or diagonal constraints.

7. Constraint Checking

A constraint-checking function is used to verify that no two queens are placed in the same row and no two queens attack each other diagonally. This function is shared across all implemented algorithms.

8. Algorithms Used

The following five search algorithms are implemented and compared:
- Depth-First Search (DFS)
- Breadth-First Search (BFS)
- Uniform Cost Search (UCS)
- A* Search Algorithm
- Hill Climbing Algorithm

The Hill Climbing algorithm is implemented collaboratively by the entire team.

9. Heuristic Function (A*)

For A*, a heuristic function based on the number of conflicting queen pairs is used. This heuristic estimates how close a given state is to a valid solution and improves search efficiency.

10. Performance Evaluation Criteria

- Execution Time
- Number of Expanded States
- Memory Usage (qualitative comparison)
- Solution Correctness
- Search Efficiency
- Completeness
- Optimality

These metrics are used to perform a fair comparison between all implemented algorithms.

Team Members and Contributions

The project work is distributed among eight team members to ensure balanced participation and clear task ownership. Each search algorithm is implemented by two members, while the Hill Climbing algorithm is developed collaboratively by the entire team. Additional responsibilities related to data handling and evaluation are also assigned.

Member Name	Responsibility Area	Assigned Task	Contribution Description
Karim	Algorithm Implementation	Depth-First Search (DFS)	Designed and implemented DFS using backtracking
Ahmed	Algorithm Implementation	Depth-First Search (DFS)	Tested DFS performance and verified correctness
Abdelrahman	Algorithm Implementation	Breadth-First Search (BFS)	Implemented BFS and analyzed state expansion
Mahmoud	Algorithm Implementation	Breadth-First Search (BFS)	Evaluated BFS memory usage and results
Michael	Algorithm Implementation	Uniform Cost Search (UCS)	Implemented UCS and handled priority queue logic
Mohamed	Algorithm Implementation	Uniform Cost Search (UCS)	Compared UCS behavior with BFS
Youssef	Algorithm Implementation	A* Search Algorithm	Implemented A* search and integrated heuristic function
Ziad	Algorithm Implementation	A* Search Algorithm	Analyzed A* performance and tuning heuristic

Collaborative and Supporting Tasks

Hill Climbing Algorithm:

The Hill Climbing algorithm is implemented collaboratively by all team members. This collective effort allows shared analysis of local search behavior and comparison with global search methods.

Data Collection and File Organization:

Experimental results, execution outputs, and performance measurements are systematically collected and organized into structured files to support reproducibility and clear analysis.

Comparison and Evaluation:

All team members participate in the final comparison and evaluation phase, where algorithm performance is analyzed based on predefined metrics such as execution time, expanded states, and solution quality.

12. Expected Outcomes

The expected outcomes of this project include:
- Correct implementation of multiple search algorithms
- Successful solution of the N-Queens problem
- A well-documented academic report and clean codebase

13. Tools and Technologies

- Programming Language: Python
- Development Environment: Visual Studio Code
- Version Control: GitHub

14. Conclusion

This project provides hands-on experience in applying artificial intelligence search algorithms to a classical problem. By comparing multiple approaches, the project highlights how different strategies impact performance and solution quality.