Project Proposal: Heuristic Search for Puzzle Solving

Submitted by: Sumel Rattan, AI-550
Date: 09/27/2024

Introduction:

The objective of this project is to compare various search-based algorithms to determine the most efficient strategies for solving complex puzzles, specifically focusing on the A* and Greedy Best search algorithms. The project will explore the impact of different heuristics—both simple and domain-specific—on the performance of these algorithms. Metrics for comparison will include speed, solution quality (accuracy), and computational resource usage (memory and processing time).

Problem Statement:

The primary challenge in puzzle-solving using search-based algorithms is identifying heuristics that balance search speed and solution quality without over-consuming computational resources. While simpler heuristics, such as Manhattan distance, offer computational efficiency, they may not always yield the optimal solution in the shortest time. In contrast, more complex, domain-specific heuristics could lead to better results but might demand higher memory and processing power.

This project will compare the performance of search algorithms on four selected puzzles, each featuring a domain-specific heuristic, to analyze the trade-offs between simple and complex heuristics.

Puzzles and Heuristics:

I will focus on four distinct puzzles, each allowing for the design of domain-specific heuristics:

- **Sliding Puzzle (8-puzzle)**
    - **Heuristic:** Manhattan distance vs. Misplaced Tiles (more complex)
- **Rubik's Cube**
    - **Heuristic:** Quarter-turn distance vs. a more complex heuristic evaluating the cube's progress toward the solved state by considering block positions and orientations.
- **Tic-Tac-Toe**
    - **Heuristic**: Number of winning combinations still open to the player vs Minimax evaluation with depth-limited search.
- **Maze Solving (grid-based)**
    - **Heuristic:** Euclidean distance (simple) vs. a heuristic that accounts for obstacle density and path complexity (more domain-specific)

Algorithms:

- **A***: I will implement A* using different heuristics for each puzzle to evaluate its effectiveness in speed and solution accuracy.
- **Greedy Best-First Search**: This algorithm focuses purely on the heuristic function without considering the cost to reach the current node. It will provide a baseline for how heuristic complexity impacts performance.

Testing Hypothesis:

- More complex and domain-specific heuristics will improve search speed and solution quality compared to simpler heuristics.
- However, this improvement will come at the cost of increased computational resources, such as memory and processing time.

For instance, using a pattern-based heuristic in the Rubik’s Cube could provide better solutions in fewer moves but require more memory to store evaluation patterns.

Methodology:

- **Algorithm Implementation**: I will implement A* and Greedy Best-First Search for each puzzle and incorporate both simple and complex heuristics
- **Performance Metrics**: I will measure the following metrics for each algorithm-puzzle combination:
    - **Search Speed**: Time taken to solve the puzzle
    - **Solution Quality**: Number of moves to reach the goal (for Rubik’s Cube and sliding puzzles)
    - **Memory Usage**: Peak memory usage during the search process
- **Benchmarking Environment**: All algorithms will be run under the same computational setup for consistent benchmarking

Expected Outcomes:

- A* is expected to perform better than Greedy Best-First Search in terms of solution quality, as it takes into account both the heuristic and the cost function.
- Domain-specific heuristics will outperform simpler ones in puzzles like Rubik’s Cube and maze-solving, but they will demand more memory and processing power.
- Simple heuristics may perform well in more straightforward puzzles like sliding puzzles but may falter in more complex problem spaces.

Conclusion:

This project aims to deepen the understanding of how heuristic design impacts the performance of search algorithms in puzzle-solving tasks. By comparing simple and complex heuristics across four distinct puzzles, I will identify trends in speed, solution quality, and computational cost. These insights could be applicable to broader AI search problems, especially in fields where domain-specific knowledge is available for crafting optimized heuristics.