Report Assignment 1

Introduction to AI

Kadirov Saidaziz DSAI-02

30.10.2024

Algorithms Descriptions

A* Algorithm

The A* algorithm is an informed search strategy that combines features of uniform-cost search and pure heuristic search to efficiently find the shortest path to a goal. In my implementation, the agent uses the Manhattan distance as a heuristic to estimate the cost from the current node to the goal. The algorithm maintains two lists: an open set of nodes to be evaluated and a closed set of nodes already evaluated. At each iteration, the agent selects the node with the lowest total estimated cost (the sum of the cost to reach the node and the heuristic estimate to the goal) and explores its neighbors. This process continues until the goal is reached or no path is found.

Backtracking Algorithm

The Backtracking algorithm employs a depth-first search strategy, exploring as far as possible along each branch before backtracking. The agent recursively visits adjacent nodes, marking visited cells to prevent cycles. When the agent encounters a dangerous cell or a dead end, it backtracks to the previous cell and tries a different path. This uninformed search method does not utilize heuristics and may explore redundant paths, making it less efficient in complex environments. For searchign bfs method used.

PEAS Description

- **Performance Measure**: The agent's success is measured by its ability to reach the Keymaker in the shortest path possible while avoiding dangers. Metrics include the number of successful runs, average path length, and execution time.
- **Environment**: A 9×9 grid representing a partially observable, static environment with obstacles (dangers) and a goal (Keymaker). The agent receives limited perceptual information about its immediate surroundings.
- **Actuators**: The agent can move one cell at a time in four directions: up, down, left, and right.
- **Sensors**: The agent perceives the presence of dangers and the Keymaker in adjacent cells after each move.

https://github.com/inopolis/intro-to-AI-assignments

Statistical Analysis

Test Map Generation

Agents evaluated 1000 maps which is randomly generated. And the results is:

Data Collection

For each map, both algorithms were executed, and the following data were recorded:

- **Execution Time**: Time taken to find a path or determine that no path exists.
- **Path Length**: Number of steps in the path from the start to the Keymaker.
- **Success Flag**: Whether the agent successfully reached the Keymaker.

Results Summary

Metric	A* Algorithm	Backtracking Algorithm

Average Execution Time	0.002781	0.003828
Median Execution Time	0.002592	0.002697
Standard Deviation	0.000641	0.000599
Average Path Length	7.853458	7.973462
Median Path Length	8.000000	8.000000
Success Rate	85.385385	82.982983
Failure Rate	14.614615	17.017017

Statistical Observations

- **Execution Time**: Both algorithms demonstrated similar execution times, but the A* algorithm showed slightly lower mean and median times. This suggests that A* was generally faster, likely due to its heuristic guidance
- **Path Length**: Paths found by the A* algorithm were slightly shorter, indicating more optimal solutions
- **Success Rate**: had a slightly higher success rate than Backtracking, which aligns with A*'s efficiency in navigating complex obstacle maps.

Comparison of Algorithms

Execution Time

The A* algorithm demonstrated slightly faster execution times. The heuristic function effectively guided the agent towards the goal, minimizing the number of explored nodes. In contrast, the Backtracking algorithm often redundantly explored paths, increasing execution time.

Path Optimality

The A* algorithm found slightly shorter paths on average due to its heuristic's ability to estimate the remaining distance to the goal. The Backtracking algorithm, lacking heuristic guidance, sometimes found suboptimal paths or failed to find a path within reasonable time constraints. But they are very same.

Success Rate

The higher success rate of the A* algorithm can be attributed to its efficient exploration strategy. The Backtracking algorithm's exhaustive search approach led to timeouts in maps with complex obstacle configurations.

Agent Behavior Analysis

A* Algorithm Behavior

- **Efficient Exploration**: Prioritized nodes closer to the goal based on the heuristic, reducing unnecessary explorations.
- **Heuristic Limitations**: In maps where the heuristic underestimated obstacles, the agent still performed well due to the admissibility of the Manhattan distance heuristic.

Backtracking Algorithm Behavior

- **Exhaustive Search**: Explored all possible paths, which led to long execution times in dense maps.
- **Lack of Heuristics**: Without guidance, the agent often retraced steps or explored inefficient paths, reducing overall performance.

Outcomes

- **Unsolvable Maps**: Both algorithms failed in certain maps where the Keymaker was completely enclosed by dangers. These maps highlight the limitations of pathfinding algorithms in impossible scenarios.
- **Maps with Moving Dangers**: In maps simulated with dynamic dangers (not part of the initial test but worth mentioning), the A* algorithm adapted better due to its ability to re-evaluate paths, whereas Backtracking struggled significantly.

Graphical Representations

```
Unsolvable map:
```

```
N P . P P P . . .
P S P P A P . . .
. P . P P P . . .
. . . . . . . . .
. . . . K . . . .
. . . . . . . . .
. . . . . . . . .
. . . . . . . . .
. . . . . . . . .
```

Conclusion

The A* algorithm slightly outperforms the Backtracking algorithm in some scenarios, offering faster execution times, higher success rates, and more optimal paths. Its heuristic approach makes it more suitable for complex environments where efficiency is critical. The Backtracking algorithm is less practical for real-world applications due to its inefficiency in handling large search spaces.

(in github repo you can recheck the statistics with 1000 tests generation using the tests.py. Instructions are in README.md)