MIT World Peace University
AIMLT

Assignment 2

Naman Soni Roll No. 06

Contents

1 Title

Implement A* Algorithm for any game search problem

2 Aim

Solve 8-puzzle problem using A* algorithm

3 Objective

To study A* algorithm

4 Theory

4.1 *Informed Search*

Informed search, also known as heuristic search, is a type of search algorithm used in computer science and artificial intelligence to efficiently find solutions to problems within a search space. Unlike uninformed search algorithms (such as depth-first search or breadth-first search), which explore the search space without any specific knowledge about the problem domain, informed search algorithms make use of additional information or heuristics to guide the search process.

4.2 *A* Algorithm*

The A* algorithm is a popular and widely used informed search algorithm in computer science and artificial intelligence. It is designed to efficiently find the shortest path or optimal solution in a search space while considering both the cost incurred to reach a particular state and a heuristic estimate of the remaining cost to reach the goal.

4.3 *A* Algorithm Steps*

The A* algorithm is a widely used informed search algorithm for finding the shortest path or optimal solution in a search space. Here are the steps of the A* algorithm in brief:

1. **Initialization**:

- Create an open list to store nodes to be explored.
- Initialize the open list with the starting node (initial state).
- Set the cost of the starting node to 0: 'g(start) = 0'.

2. **Main Loop**:

- While the open list is not empty:

3. **Select Node**:

- Choose the node with the lowest 'f(n)' value from the open list. 'f(n)' is calculated as 'f(n) = g(n) + h(n)' where:
- 'g(n)' is the actual cost to reach node 'n' from the starting node.
- 'h(n)' is the heuristic estimate of the cost from node 'n' to the goal.

4. **Goal Check**:

- If the selected node is the goal state, terminate the algorithm. You have found the optimal solution.

5. **Expand Node**:

- Generate successor nodes (states) from the selected node by applying possible actions or transitions.

6. **Evaluate Successors**:

- For each successor node:
- Calculate ‘g(n)‘ for the successor, which is the sum of the cost to reach the current node and the cost of the transition to the successor: ‘g(successor) = g(current) + cost(current, successor)‘.
- Calculate ‘h(n)‘ for the successor using the heuristic function.
- Calculate ‘f(n)‘ for the successor: ‘f(successor) = g(successor) + h(successor)‘.

7. **Update Open List**:

- Add the successors to the open list if they are not already in it.
- If a successor is already in the open list but has a higher ‘f(n)‘ value, update its ‘f(n)‘ value and parent pointer to the current node.

8. **Repeat**:

- Go back to the main loop and continue the search until the open list is empty or the goal state is found.

9. **Path Reconstruction (if goal is reached)**:

- Once the goal state is reached, you can reconstruct the optimal path by following the parent pointers from the goal node back to the initial node.

10. **Termination (if no solution)**:

- If the open list becomes empty and the goal state has not been reached, the algorithm terminates with the conclusion that there is no solution.

The A* algorithm combines the cost to reach a node from the start (‘g(n)‘) and the heuristic estimate of the remaining cost to the goal (‘h(n)‘) to prioritize the exploration of nodes. This combination of actual and estimated costs allows A* to efficiently find optimal solutions when appropriate heuristics are used.

4.4 *What is the 8-puzzle problem?*

The 8-puzzle problem is a classic sliding puzzle with a 3x3 grid containing eight numbered tiles and one empty space. The goal is to rearrange the tiles from a scrambled initial configuration to a specified target configuration by sliding tiles one at a time into the empty space. The challenge is to find the shortest sequence of moves to reach the goal configuration. This puzzle is used in AI and computer science as an example for various search and optimization algorithms, making it a popular problem for study and recreational play.

4.5 *Solve 8-puzzle problem*

The solution to the 8-puzzle problem involves rearranging the 3x3 grid of numbered tiles from an initial, scrambled configuration to a predefined goal configuration using the fewest possible moves. This is achieved by sliding the tiles one at a time into the empty space, following a sequence of steps guided by a search algorithm, such as A*. The algorithm searches for the optimal path, considering the cost of moves and a heuristic to estimate the remaining steps. Once the goal configuration is reached, the puzzle is solved.

5 Conclusion

Thus, learnt the A* Algorithm and it’s implementation.

6 FAQ's

1. What is a heuristic function? What is the advantage of using heuristic function? **Ans.** A heuristic function is an estimate used in problem-solving to guide decisions based on incomplete information. It advantages search algorithms by improving efficiency through informed decision-making.

   The advantage of using a heuristic function is that it improves the efficiency of search algorithms by guiding them toward more promising paths and reducing the search space, leading to faster and more effective solutions.

2. Explain different heuristic functions that can be used for the eight-puzzle problem. **Ans.** Several heuristic functions can be used for the eight-puzzle problem:

   - **Misplaced Tiles (Hamming Distance)**: This heuristic counts the number of tiles in the wrong position compared to the goal state.
   - **Manhattan Distance**: It calculates the total Manhattan distance (sum of horizontal and vertical distances) of each tile from its current position to its goal position.
   - **Linear Conflict**: It extends Manhattan Distance by considering conflicts between tiles in the same row or column. It adds extra cost for resolving conflicts.
   - **Pattern Database**: Divide the puzzle into smaller subproblems and create databases of optimal solutions for those subproblems. Combine the estimates from these databases.

   Each heuristic has its advantages and limitations in terms of accuracy and computation cost. The choice of heuristic can impact the efficiency and optimality of the A* search algorithm for the eight-puzzle problem.

3. Explain A* admissibile Property? **Ans.** The A* admissible property is a key characteristic of the A* search algorithm. In short, it means that the heuristic function used in A* must never overestimate the true cost to reach the goal from any given state. In other words, the heuristic should be optimistic but not overly optimistic.

4. What is the difference between the A* and AO* algorithm? **Ans.** A* and AO* are both search algorithms used for finding the shortest path or optimal solution in various problem-solving scenarios, but they have some differences in their approach and characteristics:

   1. **Basic Purpose**:

      - *A**: A* (pronounced "A star") is primarily used for search problems and pathfinding. It finds the optimal path from a start state to a goal state in a search space, considering both the actual cost to reach a state and a heuristic estimate of the remaining cost.
      - *AO**: AO* (Adaptive A*) is an extension of A* that's designed for optimization problems. It's used to find optimal solutions to problems where you want to minimize or maximize an objective function, not just find a path.

   2. **Objective Function**:

      - *A**: A* minimizes the total estimated cost, which is the sum of the actual cost to reach a state and the heuristic estimate of the remaining cost. It's focused on finding the most efficient path.
      - *AO**: AO* optimizes an objective function that you specify. While it can be used for pathfinding problems, it's more general and can handle various optimization problems by adjusting its search based on the specified objective function.

   3. **Search Space Exploration**:

      - *A**: A* explores the search space by expanding nodes in a way that minimizes the estimated total cost (f(n) = g(n) + h(n)), where g(n) is the actual cost and h(n) is the heuristic estimate. It maintains an open list of nodes to be explored.

- *AO**: AO* explores the search space by considering the values of the objective function. It adapts its search based on the changing values of the objective function and uses dynamic programming to keep track of the best solutions found so far.

4. **Use Cases**:

- *A**: A* is suitable for pathfinding in navigation systems, robotics, and puzzles like the 8-puzzle. It's used when you want to find the optimal path in a graph or grid.
- *AO**: AO* is more versatile and can be applied to various optimization problems, including resource allocation, scheduling, and game playing, where you need to optimize a certain objective while exploring a solution space.

In summary, while A* and AO* share similarities, they have different purposes and are suited for different types of problems. A* focuses on pathfinding and minimizing costs, while AO* is a more general optimization algorithm capable of solving a broader range of optimization problems.