Lesson 9: Heuristic Search Algorithms

Reviewer

What Is Heuristic Search?

- ***Intelligent Shortcuts***: A ***heuristic*** is a technique to solve problems faster than classic methods or find approximate solutions when exact ones are too costly.
- ***Speed Over Perfection***: Heuristics prioritize speed, often trading optimality or completeness for a quicker, informed solution.
- ***Guided Estimation***: It uses a ***heuristic function*** to estimate how close a state is to the goal, intelligently guiding the search.

Why Do We Need Heuristics?

- ***Tackling Complexity***: Many problems boast ***exponentially large search spaces***, rendering exhaustive search methods impractical and time-consuming.
- ***Efficiency Boost***: Heuristics dramatically reduce search time by focusing on promising paths, enabling solutions within a reasonable timeframe.

Hill Climbing Algorithm: Climbing Towards the Goal

- ***Greedy Approach***: A greedy heuristic search that moves step-by-step to states with incrementally better scores.
- ***Potential Pitfalls***: While simple and fast, it can get stuck at ***local maxima***, ***plateaus***, or ***ridges***, preventing it from reaching the global optimum.
- ***Example***: Navigating a car through the streets of Washington, D.C., always moving closer to the monument, but potentially getting stuck in a traffic loop.

A* Search Algorithm

1. ***Combined Evaluation***: A^* combines the actual cost so far $g(n)$ and the heuristic estimate $h(n)$ into a single function:
$$f(n) = g(n) + h(n)$$
2. ***Optimality Guarantee***: It guarantees an optimal solution if the heuristic used is ***admissible*** (meaning it never overestimates the true cost to the goal).
3. ***Versatile Applications***: Widely employed in critical fields such as pathfinding, robotics, and advanced AI planning due to its effectiveness.

Example: A* Search in Action

Starting at node A and aiming for node H, the A^* algorithm systematically evaluates potential paths.

It calculates the total estimated cost $f(n)$ for each node by summing the cost already incurred $g(n)$ and the estimated cost to the goal $h(n)$. By always choosing the path with the lowest $f(n)$, A^* efficiently and effectively identifies the shortest path to the destination.

Characteristics of Good Heuristics

- ***Admissible***: A good heuristic should ***never overestimate*** the true cost to reach the goal.
- ***Consistent (Monotonic)***: The estimated cost from the current node to a neighbor, plus the neighbor's heuristic, must always be greater than or equal to the current node's heuristic.
- ***Domain-Specific***: Incorporating knowledge specific to the problem domain significantly enhances the quality of the heuristic and, consequently, the efficiency of the search.

Heuristic Search Powers Efficient AI

- ***Accelerated Problem Solving***: Heuristics empower AI to rapidly solve complex problems by strategically focusing on the most promising solutions.
- ***Balancing Act***: Algorithms like Hill Climbing, Best-First Search, and A^* skillfully balance speed and accuracy in their quest for solutions.
- ***Key to Advancement***: The understanding and masterful design of effective heuristics are fundamental to driving the progress of intelligent systems.

Greedy Algorithms

Greedy Algorithm: An algorithmic paradigm that follow the problem solving approach of making the ***locally optimal choice*** at each stage with the hope of finding a global optimum.

Note: The source material includes a diagram with various numerical values and nodes (e.g., gas 3 cost 42, 53, 43, 70, 8, etc.) illustrating a specific graph instance.

When to use greedy approach?

Problems with greedy approach work has two properties:

1. ***Greedy - choice property***: A global optimum can be arrived at by selecting a local optimum.
2. ***Optimal Substructure***: An optimal solution to the problem contains an optimal solution to subproblems.

Greedy Best-First Search

Formula:

$$f(n) = h(n)$$

Concept

Greedy Best-First Search always expands the node that is estimated to be closest to the goal.

Diagram Components

- Node[cost]
- Closed List
- Nodes represented: P, M, R, C, U, A
- Values associated: (10), [9], [8], [4], (6), (8), (11), (9), (4)

Pros & Cons

- ***Pros***: Very fast. Efficient in many simple cases.
- ***Cons***: Can get stuck in loops. Not optimal (may not find the shortest path).

Application of heuristic search algorithm

- ***GPS Navigation***: A^* is the backbone of finding the shortest route in maps (Google Maps, Waze).
- ***Robotics***: Path planning for autonomous robots in warehouses (Amazon Kiva).
- ***Video Games***: NPC movement and AI pathfinding.
- ***Network Routing***: Optimizing data packet paths across the internet.