State Spaces and Search

Saagar Sanghavi, Jin Seok Won, Kevin Zhu

Outline

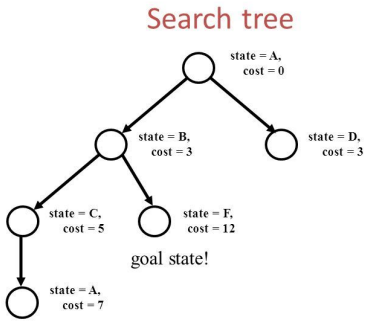
1. State Space
2. Types of Search Algorithms
3. Heuristic
4. Applications

Agent and Search Problems

Agent - entity capable of rational decision given the knowledge about its environment

World - Agent + Environment

Search Problem - how search through different states of the problem and take the best action to achieve its goal



search tree nodes and states are not the same thing!

Image source: https://slideplayer.com/slide/7883033/

State Space

State Space - The set of all possible states in the given world

Successor function - A function that takes in a state and an action and returns the cost of given action and the state after taking that action, which is called successor state

Start State - The initial state where the agent begins in

Goal test - A function that takes in a state and returns whether the goal is achieved

General Structure of Search Algorithms

1. Make a **fringe** that contains potential nodes to explore
2. Choose the next node to explore using **strategy**, remove that node from fringe
3. Add removed node's children to the fringe
4. Repeat until reaching goal state

Properties of Search Algorithms

Some nice properties you might want from search algorithms:

1. Completeness - whether the algorithm is guaranteed to find a solution within reasonable time
2. Optimality - whether the algorithm's output is an optimal solution if it ever finds one
3. Time and space complexity - upper bound on the runtime and space usage of the algorithm

Uninformed Searches

When given no information about where the solution is, we are forced to use uninformed searches, which are often much slower in the worst case than the informed searches. Here we will explore:

Depth First Search

Breadth First Search

Uniform Cost Search

Depth First Search (DFS)

Strategy - Select the deepest node in the fringe. There is usually a consistent tie-breaking scheme for multiple nodes with the same depth.

Fringe - first-out (LIFO) stack.

Completeness - No

Optimality - No

Time Complexity - O(b^m)

Space Complexity - O(bm)

Breadth First Search (BFS)

Strategy - Select the shallowest node in the fringe.

Fringe - first-out (FIFO) queue.

Completeness - Yes

Optimality - Only when edge weights are uniform

Time complexity - O(b^s) where s is depth of shallowest solution

Space complexity - O(b^s)

Uniform Cost Search (UCS)

Strategy - Select the lowest cost node in the fringe. Cost of the node is the sum of edge weights of the partial plan.

Fringe - Priority queue, ranked by backward cost

Completeness - Yes

Optimality - Yes

Time complexity - O(b^(C*/epsilon)) where C* is cost of optimal path and epsilon is minimum cost in optimal path

Space complexity - O(b^(C*/epsilon))

Informed Searches

When given information about where the solution might be, performance can be improved by focusing on specific direction first. This is the central idea of the **Informed Searches.** We will explore Greedy Search and A* search.

Heuristic

Heuristics refer to the techniques that allow us to make estimations very quickly while known algorithms can be very slow. Heuristics allow search algorithms to figure out quickly where to focus on.

Heuristics need to meet some conditions for the search algorithms to work as intended, such as admissibility and consistency.

Greedy Search

Strategy - Select the lowest heuristic node from the fringe.

Fringe - Priority queue ranked by heuristic

Completeness and Optimality - both can be varied depending on the chosen heuristic. In general greedy search is not complete or optimal

A* Search

Strategy - Select the lowest estimated total cost node in the fringe, where estimated total cost is the sum of cost of partial plan and heuristic

Fringe - Priority queue ranked by estimated total cost

Completeness and Optimality - Given a heuristic that underestimates the total cost, A* search is both complete and optimal

Applications