Lecture 3: Uninformed Search

Adam Hawley

January 23, 2019

Contents

1 Implementation of Search

1.1 State vs. Node

- A **state** is a representation of a physical configuration
- A **node** is a data structure which constitutes part of a search tree including state, parent node, action, path cost and depth.

1.2 Uninformed vs. Informed

- Uninformed (Blind) Search Algorithms: In making this decision, these look only at the structure of the search tree and not at the states inside the nodes.
- Informed (Heuristic) Search Algorithms: In making this decision these look at the states inside the nodes.

2 Evaluating Search Strategies

Strategies are evaluated along the following dimensions:

- Completeness: does it always find a solution if one exists
- Time complexity: number of nodes generated (not expanded)
- Space complexity: maximum number of nodes in memory
- Optimality: does it always find a least-cost solution?

Time and space complexity will use the following parameters:

- **b**: maximum branching factor of the search tree
- **d**: depth of the least-cost solution (root is of depth 0)
- **m**: maximum depth of the search tree (may be ∞)

3 Breadth-First Search

Expand the shallowest unexpanded node.

Implementation: *fringe* is a FIFO queue, so new nodes go at the end and nodes are selected from the front.

Properties:

- Complete: Yes (if b is finite)
- Time Complexity: $1 + b + b^2 + b^3 + ... + b^d = O(b^{d+1}) = O(b.b^d) =$ (If b is constant)$O(b^d)$

- Space Complexity: $O(b^{d+1})$ or $O(b^d)$ if only fringe is in memory
- Optimal: Yes, if all operators have the same cost

Space is a bigger problem than time. What if we want to find the optimal solution and operators have different costs?

4 Uniform-Cost Search

Expand the least-cost unexpended node. Implementation: finge is queue ordered by increasing path cost. Nodes are selected from the front.

Properties:

- Complete: Yes, if step cost $\leq \epsilon$
- Time Complexity: # of nodes with $g \leq C^*$, where C^* is the cost of the optimal solution, so $O(b^{\lceil \frac{C^*}{\epsilon} \rceil})$. Or $O(b^d)$.
- Space Complexity: # of nodes with g $\leq$ cost of optimal solution, $O(b^{\lceil \frac{C^*}{\epsilon} \rceil})$. Or $O(b^d)$.
- Optimal: Yes, since nodes are expanded in increasing order of $g(n)$.

5 Depth-First Search

Expand the deepest unexpanded node. Implementation: Fringe is a LIFO queue (or stack), so new nodes are put at front and nodes removed from the front.

Properties:

- Complete: No, could run forever if a tree has infinite depth. Complete in finite spaces though.
- Time Complexity: $O(b^m)$ (Size of the tree)
- Space Complexity: $O(bm)$ (Linear space)
- Optimal: No.

6 Depth-Limited Search

Depth-first search with depth limit l. That is, nodes whose depth is greater than l are ignored.

Implementation: same as depth-first search but if a node has depth l then it is not expanded.

Properties:

- Complete: No, solution not found if $d > l$

- Time complexity: $O(b^l)$
- Space complexity: $O(bl)$
- Optimal: No.

7 Iterative Deepening Search

To overcome the incompleteness of depth-limited search, if no solution is found, redo the search with an increased depth limit.

Properties:

- Complete: Yes
- Time complexity: $(d+1)b^0 + db^1 + (d-1)b^2 + ... + b^d = O(b^d)$
- Space complexity: $O(bd)$
- Optimal: Yes, if step cost = 1

See slide 54 for a table containing all of the properties.

8 Summary

- Problem represenation is important. Some representations can have much smaller search trees.
- Variety of uninformed search strategies.
- Iterative deepening search uses only linear space and not much more time than other uninformed algorithms.
- It is often important to eliminate repeated states.