SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of AI &DS

Course Name – 23ADT201 ARTIFICIAL INTELLIGENCE

II Year / III Semester

Unit 1-INTELLIGENT AGENTS

UNINFORMED SEARCH STRATEGIES:

Case Study: Optimizing Delivery Routes Using Breadth-First Search (BFS)

Objective: To optimize the delivery routes for a local courier service in a small city by finding the shortest path between various delivery points using Breadth-First Search (BFS).

UNINFORMED SEARCH STRATEGIES:

Uninformed search strategies, also known as blind search strategies, explore the search space without any knowledge about the goal beyond what is provided in the problem definition. These strategies do not use domain-specific knowledge and work purely by exploring nodes in the search space systematically.

1. Breadth-First Search (BFS)

Description: Breadth-First Search explores all nodes at the present depth level before moving on to nodes at the next depth level. It is a level-order traversal of the search tree.

How it Works:

- Starts from the root node (initial state).
- Explores all child nodes at the present depth level before moving to nodes at the next depth level.
- Uses a queue (FIFO) to keep track of nodes to be explored.

UNINFORMED SEARCH STRATEGIES:

Properties:

- Completeness: Guarantees finding a solution if one exists, as long as the search space is finite.
- Optimality: Finds the shortest path in terms of the number of steps if all actions have the same cost.
- Time Complexity: O(b^d), where bbb is the branching factor and ddd is the depth of the solution.
- Space Complexity: O(b^d), which can be large due to storing all nodes in the queue.

Example: Finding the shortest path in an unweighted maze or graph.

2. Depth-First Search (DFS)

Description: Depth-First Search explores as far down a branch of the search tree as possible before backtracking to explore other branches. It uses a stack to keep track of nodes.

How it Works:

- Starts from the root node (initial state).
- Explores child nodes recursively, diving deep into each branch before backtracking.
- Uses a stack (LIFO) to keep track of nodes to be explored.

UNINFORMED SEARCH STRATEGIES:

Properties:

- Completeness: Not guaranteed, especially in infinite spaces or if there are loops.
- Optimality: Does not guarantee the shortest path.
- Time Complexity: O(b^m), where mmm is the maximum depth of the search tree.
- Space Complexity: O(bm), where bbb is the branching factor and mmm is the maximum depth.

Example: Solving puzzles or finding paths in a maze where paths are not equally weighted.

3. Uniform Cost Search (UCS)

Description: Uniform Cost Search is a variant of BFS that considers the cost of reaching a node. It expands nodes based on the lowest cumulative cost rather than the depth.

How it Works:

- Uses a priority queue (min-heap) to explore nodes with the lowest path cost.
- Starts from the root node, expands the node with the lowest path cost, and updates the costs of its neighbors.

UNINFORMED SEARCH STRATEGIES:

Properties:

- Completeness: Guarantees finding a solution if one exists.
- Optimality: Guarantees finding the least-cost path if costs are non-negative.
- Time Complexity: O(b^d) in the worst case.
- Space Complexity: O(b^d) due to storing all nodes in the priority queue.

Example: Finding the shortest path in a weighted graph where edges have different costs.

4. Depth-Limited Search (DLS)

Description: Depth-Limited Search is a variation of DFS that limits the depth of the search tree to a predetermined limit.

How it Works:

- Performs DFS up to a specified depth limit.
- If the goal is not found within the depth limit, it terminates.

Properties:

- Completeness: Not guaranteed; it might miss solutions beyond the depth limit.
- Optimality: Not guaranteed.
- Time Complexity: O(b^l), where lll is the depth limit.
- Space Complexity: O(bl), where bbb is the branching factor and lll is the depth limit.

Example: Searching for a solution in a tree where solutions are known to be within a certain depth.

5. Iterative Deepening Search (IDS)

Description: Iterative Deepening Search combines the benefits of BFS and DFS by performing a series of depth-limited searches, increasing the depth limit with each iteration.

How it Works:

- Performs a series of DFS with increasing depth limits.
- Each iteration is a depth-limited search with an increasing depth until the goal is found.

Properties:

- Completeness: Guarantees finding a solution if one exists.
- Optimality: Finds the shortest path if all actions have the same cost.
- Time Complexity: O(b^d), similar to BFS, but with less space complexity.
- Space Complexity: O(bd), due to storing nodes at a particular depth.

Example: Searching in a large or infinite space where the depth of the solution is unknown.

6. Bidirectional Search

Description: Bidirectional Search simultaneously searches from the start node and the goal node, hoping that the two searches meet in the middle.

How it Works:

- One search starts from the initial state and expands towards the goal.
- The other search starts from the goal and expands towards the initial state.
- Stops when the two searches intersect.

Properties:

- Completeness: Guaranteed if the solution exists.
- Optimality: Finds the shortest path if all actions have the same cost.
- Time Complexity: O(b^(d/2)), where ddd is the depth of the solution, since each search explores half of the depth.
- Space Complexity: O(b^(d/2)), as it stores nodes from both searches.

Example: Finding the shortest path between two cities on a map where both start and end points are known.

THANK YOU