

Search Methods

Unit No 2

Dr. Praveen Barapatre

Search Methods

- Search methods are fundamental to artificial intelligence, providing a systematic way to explore a problem space and find solutions. They are often used in tasks such as game-playing, planning, and problem-solving.

Types of Search Methods

Uninformed Search: These methods do not have any additional information about the problem domain. They explore the search space blindly, relying solely on the search algorithm itself.

1. **Breadth-First Search (BFS):** Explores all nodes at a given depth before moving to the next level.
2. **Depth-First Search (DFS):** Explores a path as far as possible before backtracking.
3. **Uniform-Cost Search:** Expands nodes based on their total cost from the start state.

- **Informed Search:** These methods utilize domain-specific knowledge to guide the search towards promising solutions.
- *A Search:** Combines the heuristic function (estimated cost to the goal) with the actual cost to the node to prioritize the search.
- **Greedy Best-First Search:** Prioritizes nodes based solely on the heuristic function.
- **Hill Climbing:** Moves towards the highest-valued neighbor until a local maximum is reached.

- Search Space: The set of all possible states that an agent can reach.
- State: A representation of the current situation in the problem.
- Goal State: The desired final state.
- Action: A transformation that can be applied to a state to reach another state.
- Heuristic Function: Estimates the cost to reach the goal from a given state.

Applications

Search methods are used in a wide range of AI applications, including:

- **Game-Playing:** Finding optimal moves in games like chess and Go.
- **Planning:** Generating plans to achieve goals in complex environments.
- **Problem-Solving:** Solving puzzles and riddles.
- **Robotics:** Pathfinding for robots navigating in unknown environments.
- **Natural Language Processing:** Parsing sentences and understanding language.

Choosing the Right Search Method

The best search method depends on the specific problem and its characteristics. Factors to consider include:

- **Size of the search space:** Large search spaces may require more efficient algorithms.
- **Nature of the problem:** Some problems may have specific properties that can be exploited by certain search methods.
- **Availability of heuristic information:** Informed search methods are more effective when a good heuristic function is available.

State Space Search

- **State space search** is a fundamental technique in artificial intelligence that involves exploring a set of possible configurations (states) to find a solution to a problem. It's often used in problem-solving, planning, and game-playing.
- **State:** A representation of a particular configuration of the problem.
- **Action:** A transformation that can be applied to a state to reach another state.
- **Initial State:** The starting point of the search.
- **Goal State:** The desired final configuration.
- **Search Space:** The set of all possible states that can be reached from the initial state.

Example: The 8-Puzzle

- The 8-puzzle is a classic example of state space search. The goal is to rearrange a 3x3 grid of numbered tiles (1-8) to a specific target configuration. Each move involves sliding a tile into an adjacent empty space.

Generate and Test

- G&T is a simple yet effective search algorithm that involves generating potential solutions and testing them against a given criteria. It's particularly useful for problems where the solution space is relatively small and well-defined.

Steps Involved:

- 1.Generate:** Generate a potential solution candidate. This can be done randomly, systematically, or using domain-specific knowledge.
- 2.Test:** Evaluate the generated solution against the problem's constraints and goals. If the solution meets the criteria, it is considered a valid solution.
- 3.Repeat:** If no valid solution is found, repeat steps 1 and 2 until a solution is discovered or a predetermined termination condition is met.

Example

The 8-puzzle is a classic example of G&T search. The goal is to rearrange a 3x3 grid of numbered tiles (1-8) to a specific target configuration.

- 1.Generate:** Generate a random configuration of the tiles.
- 2.Test:** Check if the generated configuration matches the target configuration.
- 3.Repeat:** If not, generate a new random configuration and repeat until a solution is found or a maximum number of attempts is reached.

- Efficiency: G&T can be inefficient for large search spaces.
- Heuristics: Incorporating heuristics can improve efficiency by guiding the search towards promising solutions.
- Termination Conditions: Defining appropriate termination conditions is crucial to prevent infinite loops.

Simple Search

- **Simple search** is a fundamental technique in artificial intelligence (AI) used to explore a problem space and find solutions. It's often employed in tasks such as game-playing, planning, and problem-solving.

Depth-First Search (DFS)

- **Depth-First Search (DFS)** is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It's often used in situations where finding a solution quickly is more important than finding the optimal solution.

How DFS Works

- 1.Start:** Begin at a chosen starting node.
- 2.Explore:** Explore the current node's neighbors one by one, going deeper into the graph.
- 3.Backtrack:** If a dead-end or previously visited node is reached, backtrack to the previous node and explore its remaining unexplored neighbors.
- 4.Repeat:** Continue exploring and backtracking until the goal state is reached or all possible paths have been explored.

Pseudocode

```
function dfs(node):  
    if node is goal:  
        return node  
    mark node as visited  
    for each neighbor of node:  
        if neighbor is not visited:  
            result = dfs(neighbor)  
            if result is not None:  
                return result  
    return None
```


Advantages of DFS

- **Efficient memory usage:** DFS requires less memory than BFS as it only needs to store the current path.
- **Can find solutions quickly:** DFS may find a solution without exploring the entire search space, especially if the goal is deep in the graph.
- **Suitable for depth-limited search:** DFS can be used with depth limits to prevent infinite loops.

Disadvantages of DFS

- **May get stuck in infinite loops:** If the graph contains cycles, DFS can explore a path indefinitely.
- **May not find the optimal solution:** DFS doesn't guarantee finding the shortest path to the goal.

Applications

- **Maze solving:** Finding a path through a maze.
- **Graph traversal:** Exploring all nodes in a graph.
- **Game-playing:** Searching for possible moves in games like chess and Go.
- **Web crawling:** Discovering new web pages.

Video

<https://www.youtube.com/watch?v=7fujbpJ0LB4>