Lab I: State Space Search Using BFS and DFS

Soham Naukudkar Roll No: 202351135
Shreyash Borkar Roll No: 202351136
Siddhikesh Gavit Roll No: 202351040

***Abstract*—Artificial Intelligence relies on problem-solving paradigms where states, actions, and transitions are formally defined. This report studies two classic problems — the Missionaries and Cannibals problem and the Rabbit Leap problem — as case studies in state space search. Both problems are modeled systematically, their search spaces are analyzed, and solutions are derived using Breadth-First Search (BFS) and Depth-First Search (DFS). BFS guarantees optimal solutions, while DFS provides a contrasting non-optimal approach. The report compares these algorithms in terms of solution optimality, time complexity, and space complexity, highlighting both theoretical and practical insights.**

Python implementations are available at: GitHub Repository.

***Index Terms*—State Space Search, BFS, DFS, Missionaries and Cannibals, Rabbit Leap, Artificial Intelligence**

I. Introduction

Artificial Intelligence (AI) often frames decision-making challenges as search problems in well-defined state spaces. Solving such problems requires traversing a graph of states from an initial configuration to a goal configuration using valid transitions. Classical problems such as Missionaries and Cannibals and Rabbit Leap have historically been used to benchmark search algorithms and analyze their properties. This report demonstrates how to model these problems as state-space search tasks, analyzes their complexity, and experimentally compares BFS and DFS in solving them.

II. Problem Formulation

A. Missionaries and Cannibals

Problem Definition: Three missionaries and three cannibals are on one bank of a river, along with a boat that can carry one or two people. The goal is to transport everyone to the opposite bank without ever leaving missionaries outnumbered by cannibals on any bank.

State Representation: (M, C, B), where M = missionaries on left bank, C = cannibals on left bank, B = boat position (0 = left, 1 = right).

Initial State: $(3, 3, 0)$

Goal State: $(0, 0, 1)$

Constraints: $0 \leq M, C \leq 3$ and missionaries are never outnumbered on either bank.

B. Rabbit Leap

Problem Definition: Three east-bound rabbits (E) and three west-bound rabbits (W) are arranged on a line of stones with one empty stone between them. Rabbits move forward into an adjacent empty stone or leap over exactly one rabbit into an empty stone.

State Representation: A configuration is represented as a 7-cell array, e.g., [E,E,E,0,W,W,W].

Initial State: [E,E,E,0,W,W,W]

Goal State: [W,W,W,0,E,E,E]

III. Search Space Analysis

Missionaries and Cannibals: Only states satisfying the safety constraint are valid. The theoretical maximum number of states is:

$$4 \times 4 \times 2 = 32 \tag{1}$$

After pruning unsafe states, fewer than 20 valid states remain.

Rabbit Leap: The number of unique permutations is:

$$\frac{7!}{3! \times 3! \times 1!} = 140 \tag{2}$$

Only a subset corresponds to legal transitions.

IV. Breadth-First Search (BFS)

A. Algorithm Overview

Breadth-First Search explores nodes level by level, ensuring the shortest solution path is discovered first. It guarantees completeness and optimality for uniform step-cost problems.

Algorithm 1: Breadth-First Search (BFS) for State-Space Search

```
1 [1] Initialize a queue with the initial state. while queue not empty do
2 Dequeue a state $s$. if $s$ is goal then
3 return solution path. forall valid unvisited successors $s'$ of $s$ do
4 Enqueue $s'$. return failure.
```

B. Results

- **Missionaries and Cannibals:** BFS produced the optimal solution in 11 moves.
- **Rabbit Leap:** BFS produced the shortest legal sequence of moves for all rabbits to swap positions.

V. Depth-First Search (DFS)

A. Algorithm Overview

Depth-First Search explores one path deeply before backtracking, which may lead to suboptimal or infinite paths but requires less memory.

Algorithm 2: Depth-First Search (DFS) for State-Space Search

```
1 [1] Initialize a stack with the initial state. while stack not empty do
2 Pop a state $s$. if $s$ is goal then
3 return solution path. forall valid unvisited successors $s'$ of $s$ do
4 Push $s'$ onto the stack. return failure.
```

B. Results

- **Missionaries and Cannibals:** DFS found valid but sometimes non-optimal solutions (more than 11 moves).
- **Rabbit Leap:** DFS produced variable-length solutions depending on exploration order.

VI. Comparison of BFS and DFS

A. Solution Optimality

BFS guarantees the shortest sequence of moves, while DFS does not guarantee optimality.

B. Complexity Analysis

Let b = branching factor, d = depth of the shallowest solution, and m = maximum depth of the search tree.

- **BFS:** Time complexity $O(b^d)$, Space complexity $O(b^d)$
- **DFS:** Time complexity $O(b^m)$, Space complexity $O(b \cdot m)$

C. Observations

- BFS is preferred when memory permits and optimality is essential.
- DFS is useful when memory is limited and a non-optimal solution is acceptable.

VII. Conclusion

This study modeled and solved two classical AI problems using state-space search. BFS consistently provided optimal solutions, albeit with higher memory usage, while DFS demonstrated trade-offs between efficiency and optimality. These findings reinforce the importance of algorithm selection based on problem size and constraints.

VIII. References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.