Lab Assignment 1: Missionaries and Cannibals Problem Rabbit Leap Problem

Reedam Choudhary
(Roll no.- 20251602020)
M.Tech(DS)
1st year

Tishu Verma
(Roll no.- 20251602024)
M.Tech(DS)
1st year

Mansi Surti
(Roll no.- 20251602014)
M.Tech(DS)
1st year

***Abstract*—This report presents the implementation of two classical state-space search problems — the Missionaries and Cannibals problem and the Rabbit Leap puzzle. Both problems are modeled as state-space search tasks and solved using Breadth-First Search (BFS) and Depth-First Search (DFS). The report discusses the problem definitions, state representations, search algorithms, and constraints, followed by an analysis of results.**

I. Introduction

Artificial Intelligence relies on search algorithms to solve constraint-based and combinatorial problems. In this experiment, we address two classic problems – the Missionaries and Cannibals crossing puzzle and the Rabbit Leap problem. These problems serve as examples of search strategies applied to well-defined state spaces.

II. Problem Definition

A. Missionaries and Cannibals: Three missionaries and three cannibals must cross a river using a boat constraint: At no point can missionaries be outnumbered by cannibals on other bank.

B. Rabbit Leap: Three rabbit facing east (E) and three facing west (E) must swap positions across stones witone empty space. Constraints: Rabbits can only move forward, either sliding into an empty spot or leaping over one rabbit.

III. State Space Modelling

Each problem is represented as a finite set of states connected by valid transitions. In the Missionaries and Cannibals problem, each move corresponds to transferring one or two individuals across the river while maintaining safety constraints. In the Rabbit Leap problem, moves are determined by available positions and directionality rules.

The Breadth-First Search (BFS) algorithm systematically explores the shallowest nodes first, ensuring the discovery of the shortest possible solution path. Conversely, the Depth-First Search (DFS) algorithm explores deeper states before backtracking, often using less memory but sacrificing guaranteed optimality.

IV. Algorithmic Framework

A. Breadth-First Search (BFS): BFS employs a queue structure. It systematically explores each level of the state tree before moving deeper, ensuring minimal-step solutions but consuming more memory.

B. Depth-First Search (DFS): DFS utilizes recursion or an explicit stack to explore deeper nodes first. While it requires less memory, it may revisit previously explored states and does not always find the shortest path.

V. Pseudo-code Implementation

A. BFS

```
function BFS(initial, goal):
    queue = [initial]
    visited = {initial}
    parent = {initial: null}
    while queue not empty:
        state = queue.pop_front()
        if state == goal:
            return path from parent
        for each next in
            generate_successors(state):
            if next not in visited:
                queue.append(next)
                visited.add(next)
                parent[next] = state
    return "No solution"
```

B. DFS

```
function BFS(initial, goal):
    function DFS(initial, goal):
    stack = [initial]
    visited = {initial}
    parent = {initial: null}
    while stack not empty:
        state = stack.pop()
        if state == goal:
            return path from parent
        for each next in
            generate_successors(state):
            if next not in visited:
                stack.push(next)
                visited.add(next)
                parent[next] = state
    return "No solution"
```

VI. Results and Discussion

A. Missionaries and Cannibals:

a) BFS: Guarantees a safe and optimal sequence of moves where constraints are never violated.

b) DFS: Finds a valid solution eventually but may traverse redundant or unsafe intermediate states.

B. Rabbit Leap:

a) BFS: Identifies the minimal number of valid slides and jumps to complete the puzzle efficiently.
b) DFS: Explores deeper configurations, sometimes leading to repeated states without progress if not properly checked.

VII. Conclusion

This experiment successfully demonstrates how classical search problems can be modeled using state-space representations and solved via BFS and DFS. BFS consistently produces optimal solutions but demands higher memory usage. DFS, while less memory-intensive, risks non-optimal paths and redundant exploration. These findings illustrate the fundamental trade-off between time complexity, memory efficiency, and optimality in classical AI search algorithms.

References

[1] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.
[2] Deepak Khemani, A First Course in Artificial Intelligence, McGraw-Hill Education.
[3] Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau. Simulated Annealing: From Basics to Applications.