Ant trail problem

AGENDA

Problem Representation

Approach Solution

Result

PROBLEM REPRESENTATION

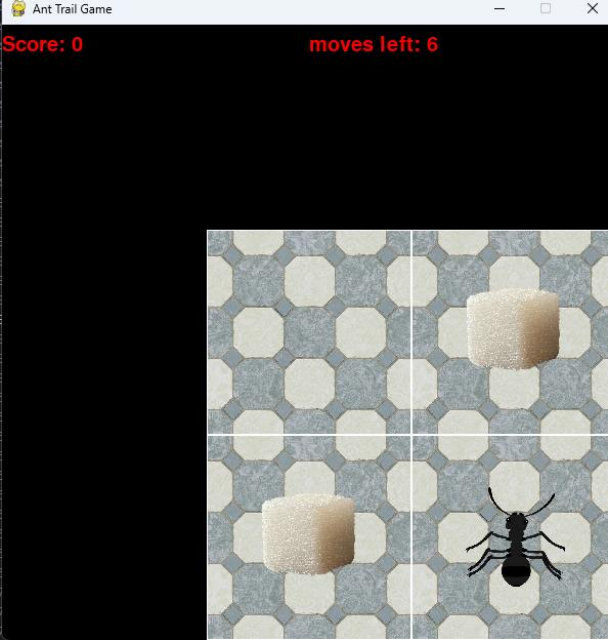
1. **Grid Setup**:
   - The problem environment is an N x N grid.
   - The game begins with the ant starting at a random position on the N x N grid.
   - The grid contains N randomly placed sugar cubes (rewards), and the ant must navigate the grid to collect as many as possible.

2. **Ant's View**:
   - The ant can only see a limited area around its current position, defined by a 2m + 1 square.
   - This restricts the ant's decision-making process as it can't see the entire grid.

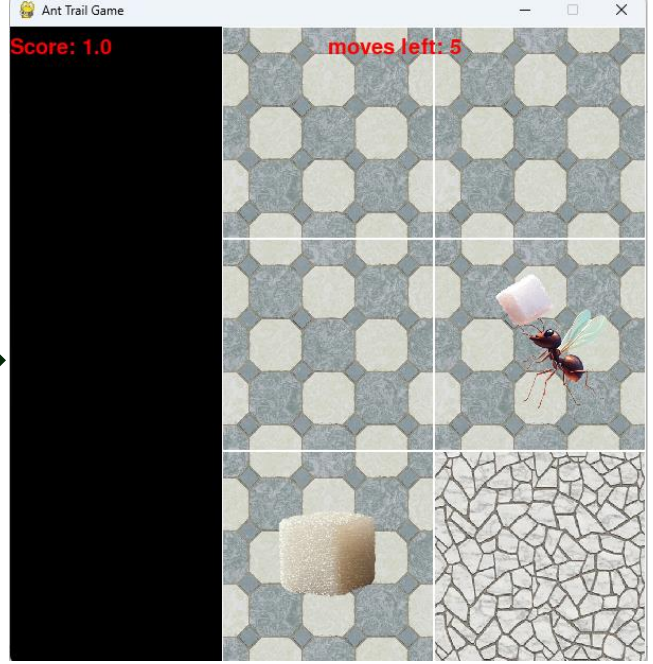
3. **Ant's Moves**:
   - The ant is allowed up to 2 * N moves to traverse the grid.
   - It can move up, down, left, or right.
   - After each move, the value of the cell the ant was previously on decreases by 1.

4. **Objective:**
   - The goal is to collect the maximum number of rewards while minimizing the number of moves.

5. **Game Ending Conditions:**
   - **Condition 1**: The game ends when the ant has made the maximum number of moves (2 * N).
   - **Condition 2**: The game finishes if the ant collects **all the sugar cubes** on the grid before using all of its moves.

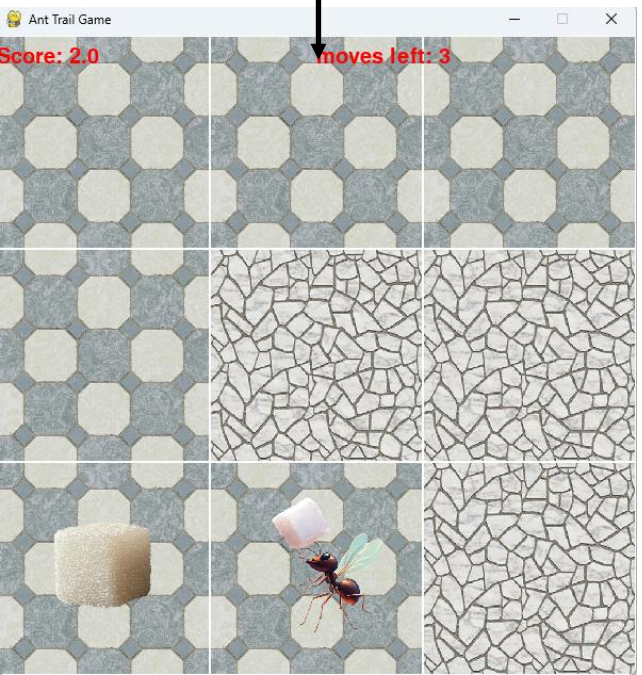


UP



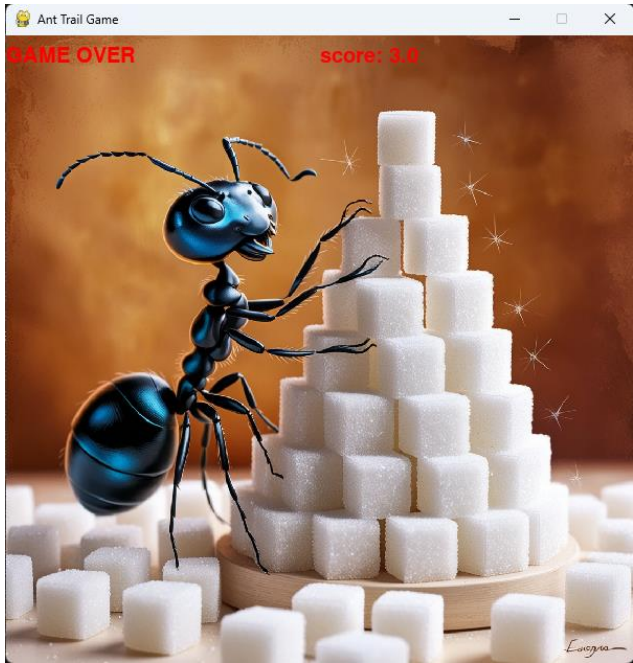
LEFT

DOWN



LEFT

GAME OVER

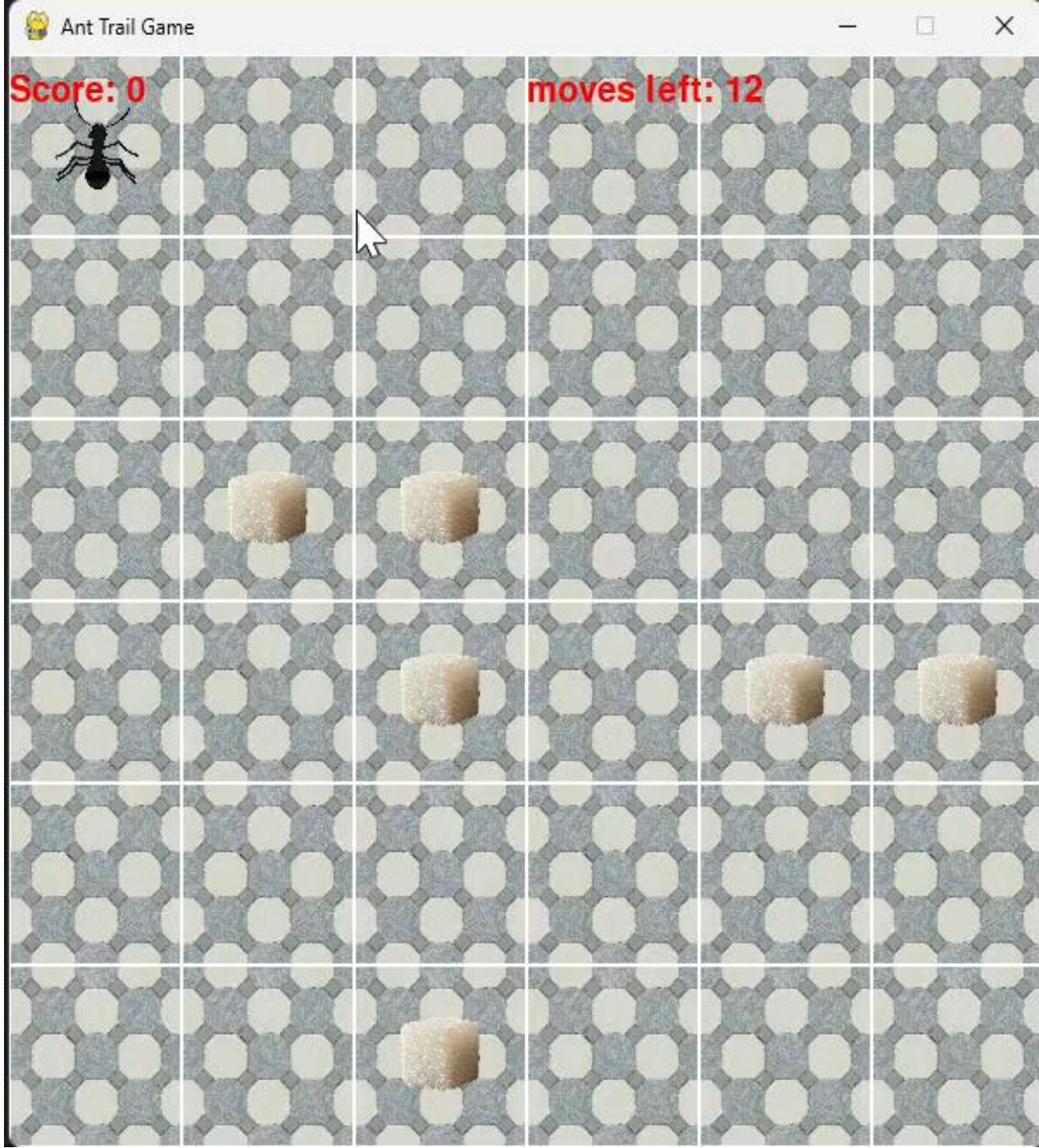


SOLUTION APPROACH

Recursive Backtracking

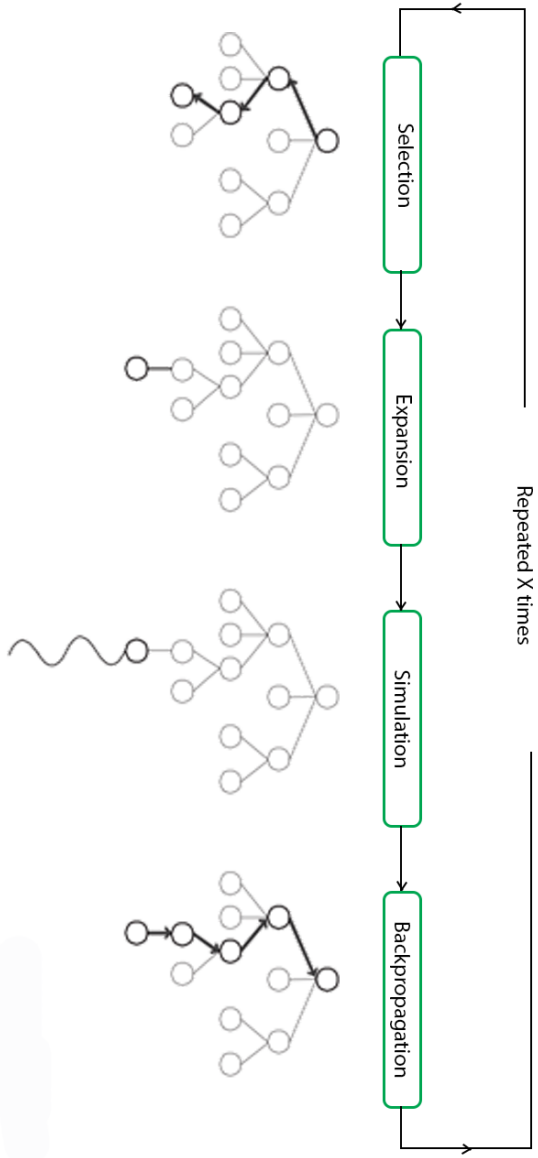
- **Overview** :This method helps the ant navigate an N x N grid, where it either tries to collect sugar (rewards) or explore the grid to discover new areas. The algorithm balances two main objectives: maximizing the rewards collected from sugar and exploring new cells to increase the ant’s knowledge of the grid.
- **Two Main Conditions**:
  - **Condition 1: When Sugar is Discovered Nearby:**
    - If sugar cubes are visible in the ant's limited view, the ant attempts to maximize its score (rewards) by choosing the path that gives access to the most undiscovered cells, while still trying to collect sugar efficiently.
  - **Condition 2: When No Sugar is Discovered Nearby:**
    - If there are no sugar cubes visible in the ant’s view**,** the ant attempts to maximize the number of new cells it can discover while minimizing unnecessary moves.

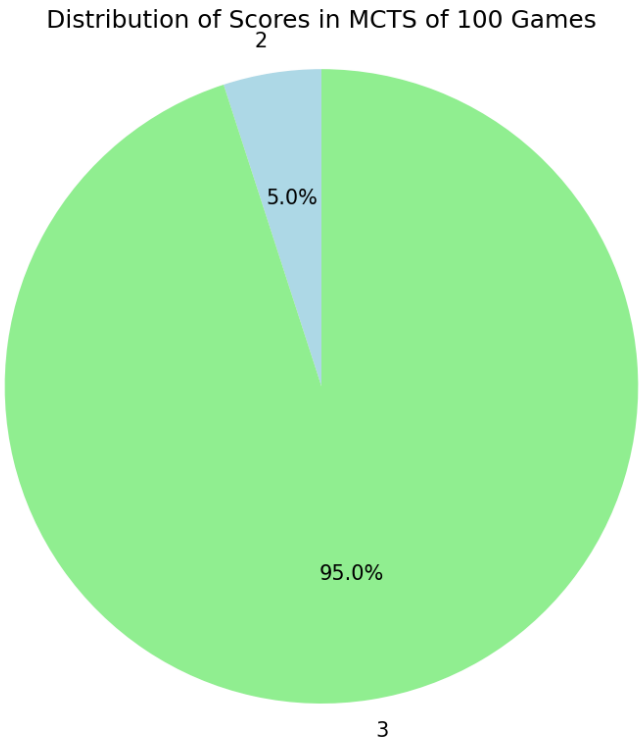
Example of game 6 X 6

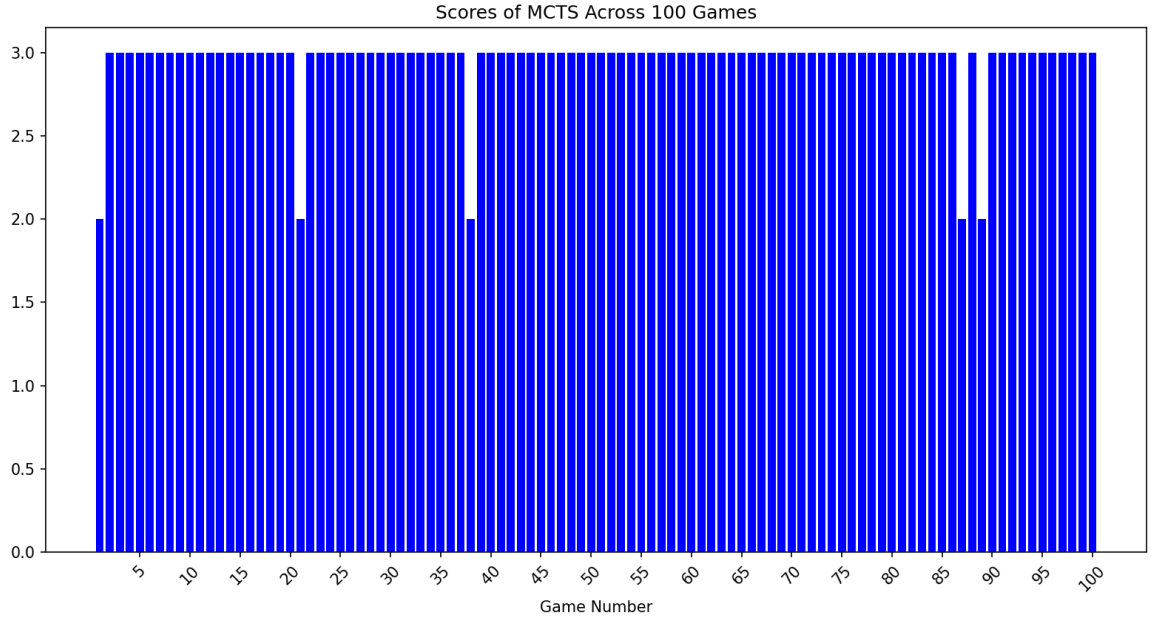


Monte Carlo Tree Search (MCTS)

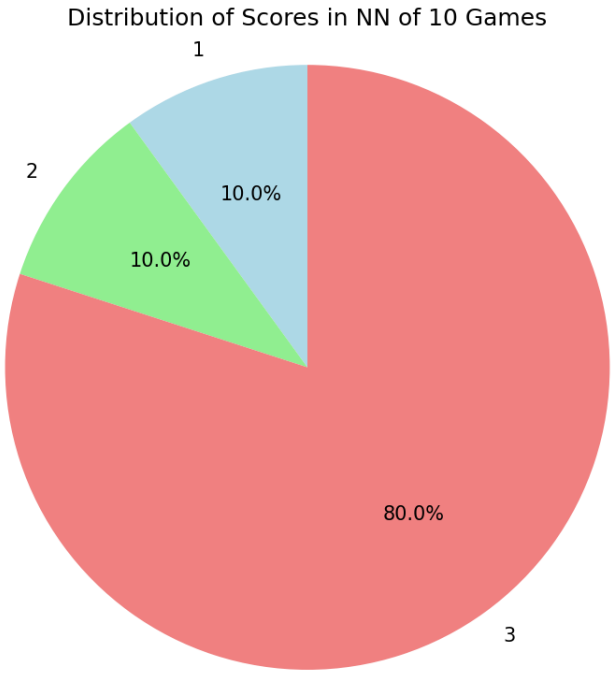
- MCTS is a search algorithm used for decision-making processes, particularly in games and simulations. It builds a search tree by running simulations to explore possible outcomes and find the best strategy:
    - Selection: The algorithm starts at the root node and traverses the existing search tree using a strategy based on the Upper Confidence Bound (UCB) formula. This formula helps balance the trade-off between exploration (trying new nodes) and exploitation (choosing the best-known option).
    - Expansion: A new child node is added to the tree by expanding the selected node from the Selection step, the new node represents a possible move or state that hasn't been fully explored yet.
    - Simulation: From the newly expanded node, a simulation is run by randomly or heuristically choosing moves until a final result is reached (e.g., win, loss, or draw in a game), The goal is to estimate the potential value of this node based on the simulated outcome.
    - Backpropagation**:** The results of the simulation are backpropagated up the tree to update the parent nodes, the algorithm updates: The number of simulations for each node visited during the traversal, The win count if the simulation led to a positive result.

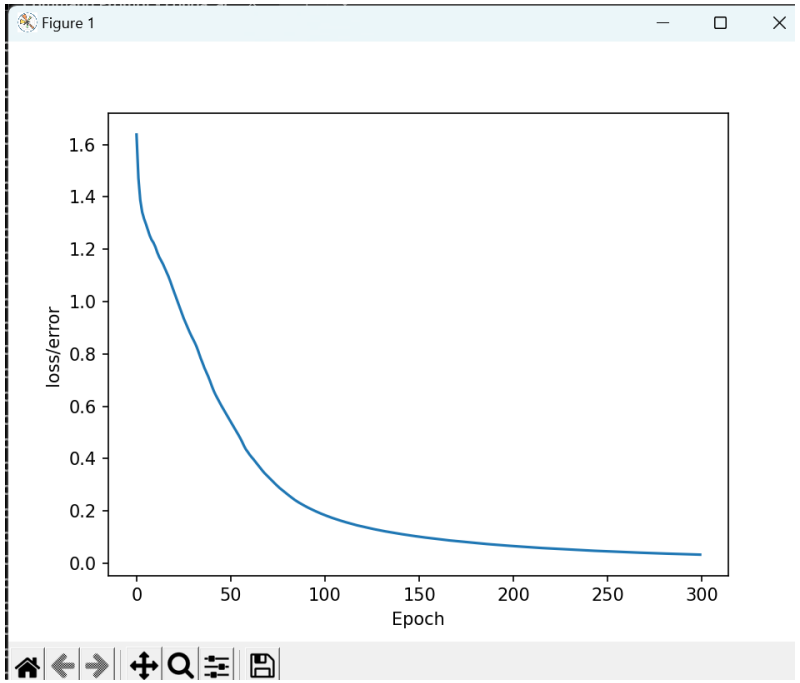


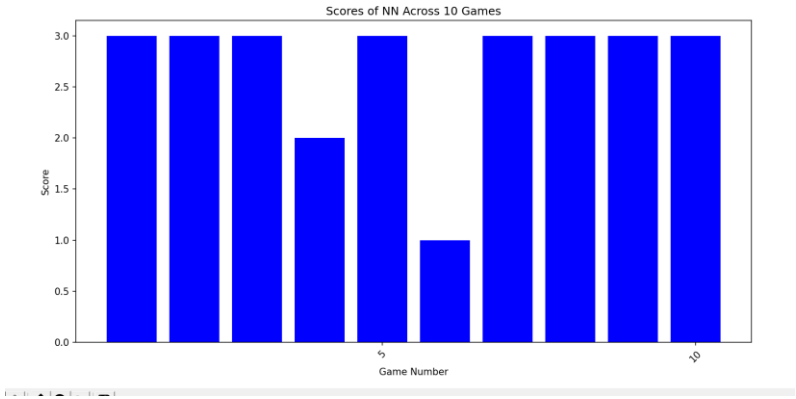




Results of Simulating MCTS

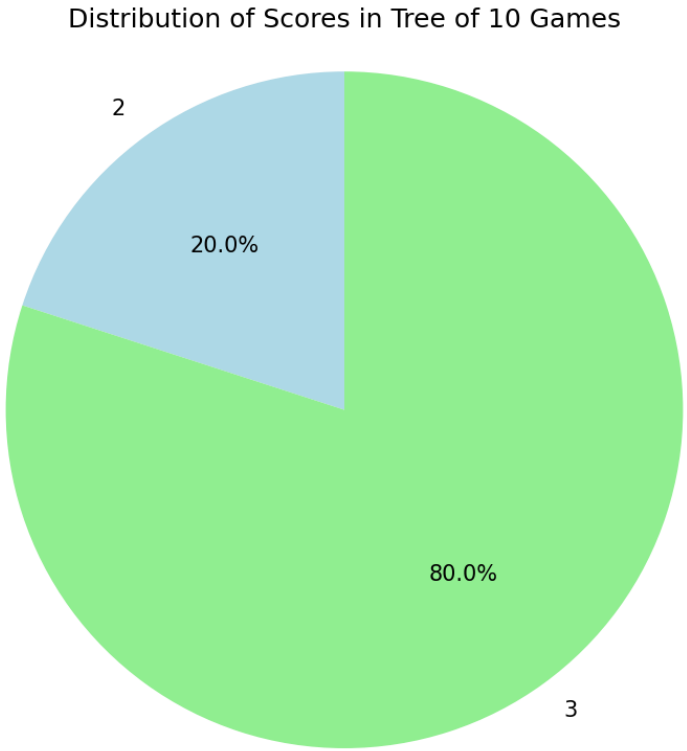


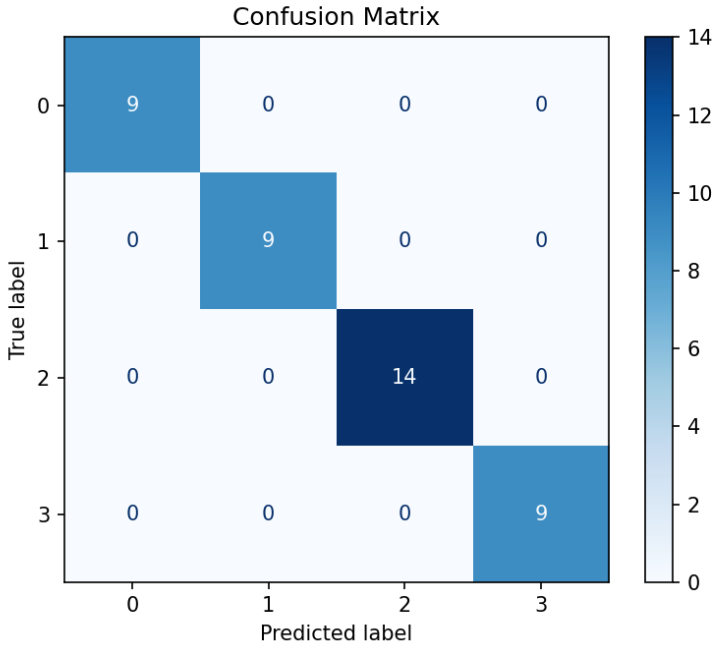


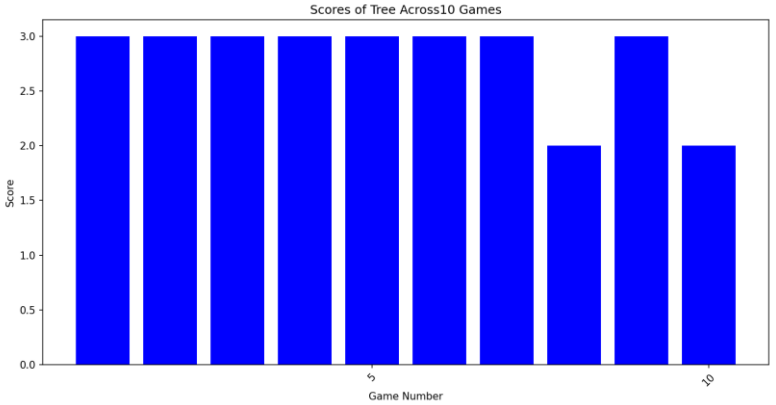


Neural Network

Grid length = 3, ant view = 2







Decision Tree

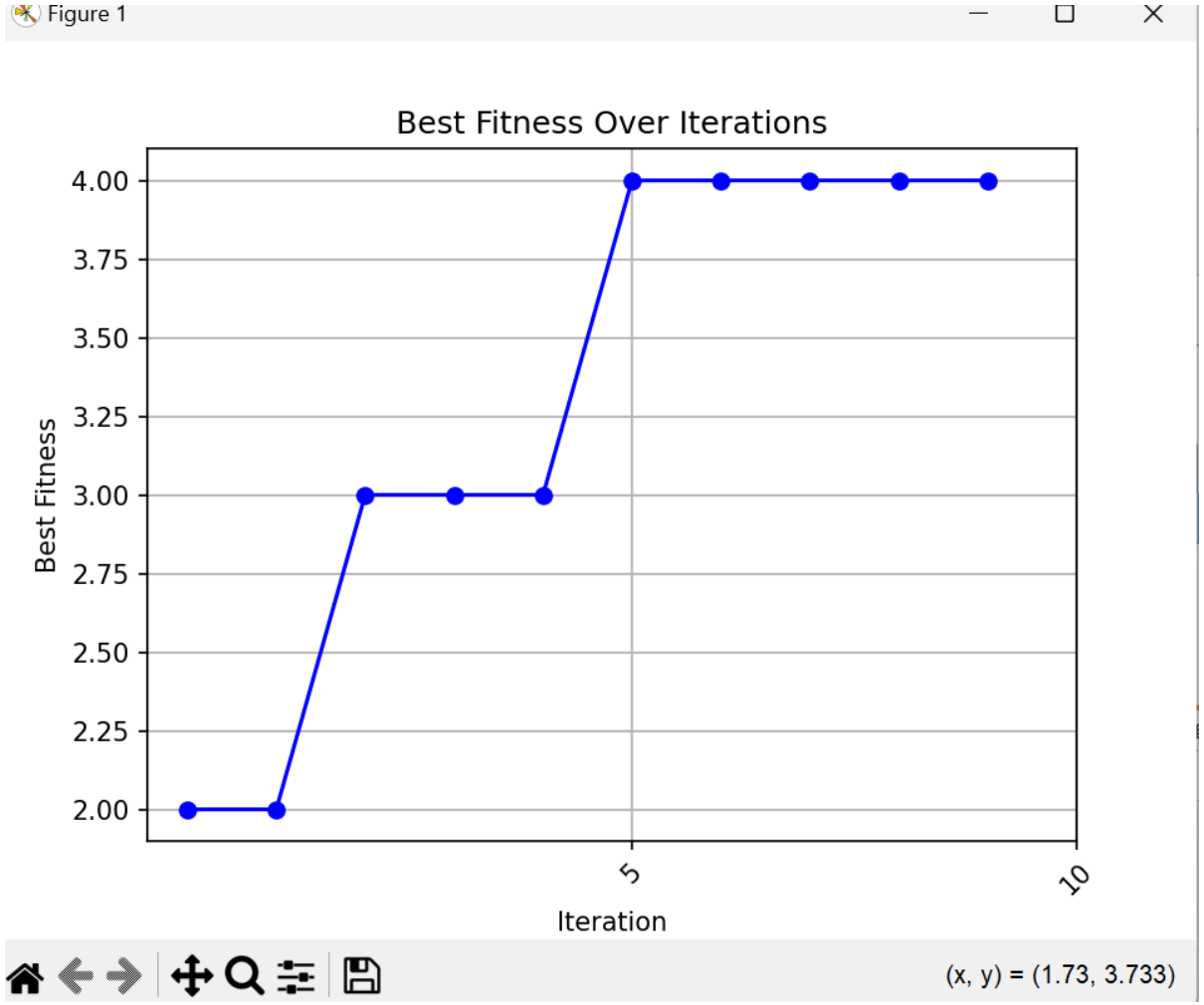
Grid length = 3, ant view = 2

Genetic algorithms (GAs)

Population size= 200, generation = 10,

Crossover rate = 0.7,

Mutation rate = 0.01



THANK YOU

Cognome : Kedis

Nome: Bassem Biskhayroun Bashir

Matricola: 356943