Lab 4 Report: Multi-Agent Search in Pacman

Student ID: 210042175
Department of CSE
Islamic University of Technology

September 3, 2025

1 Introduction

In this lab, I worked on designing agents for the classic Pacman game with a focus on **multi-agent search algorithms**. The main tasks included implementing:

- A Reflex Agent with a custom evaluation function.
- Minimax Agent.
- Alpha-Beta Pruning Agent.
- Expectimax Agent.
- A stronger evaluation function for better performance.

The goal was to make Pacman play intelligently against ghosts and optimize decision-making under uncertainty.

2 Problem Analysis

The challenge was to adapt search algorithms like Minimax, Alpha-Beta, and Expectimax to a multi-agent setting where:

- Pacman acts as the **maximizer**.
- Ghosts act as the **minimizers** or stochastic agents.
- Depth is measured in terms of complete turns (Pacman + all ghosts).

Unlike single-agent search, here we had to handle multiple adversaries, probabilistic ghost behavior, and balance between food collection and ghost avoidance.

3 Solution Explanation

3.1 Reflex Agent

I improved the reflex agent by designing a better evaluation function that considers:

- Distance to the nearest food (closer food = higher score).
- Ghost positions (penalty for being too close).
- Capsule positions (encourage grabbing power pellets).
- Scared ghosts (bonus for chasing them when safe).

Listing 1: Snippet from ReflexAgent Evaluation Function

```
if action == Directions.STOP:
    score -= 5
if closest_food_dist:
    score += 10.0 / closest_food_dist
if ghost_dist <= 1 and not scared:
    score -= 500
```

3.2 Minimax Agent

The Minimax agent searches the game tree, alternating between Pacman (maximizer) and ghosts (minimizers). The terminal test occurs when:

- A win/loss state is reached.
- Maximum depth is reached.

3.3 Alpha-Beta Agent

The Alpha-Beta agent improves Minimax by pruning unnecessary branches using bounds (α, β). Key pruning rules:

- At MAX nodes: prune if $v > \beta$.
- At MIN nodes: prune if $v < \alpha$.

This greatly reduces the number of expanded states.

3.4 Expectimax Agent

Ghosts were modeled as **random agents**. Instead of minimizing, their value was the average of successor states. This models uncertainty and prevents over-optimistic assumptions about ghost moves.

3.5 Better Evaluation Function

The final evaluation function combined multiple features:

- Base game score.
- Inverse distance to nearest food and capsules.
- Ghost safety (avoid active ghosts, chase scared ghosts).
- Stronger weights on events like eating food or capsules.

4 Findings and Insights

- Reflex Agent performed decently on simple maps but failed against multiple ghosts.
- Minimax worked but was computationally heavy.
- Alpha-Beta pruning achieved the same results as Minimax with fewer expansions.
- Expectimax was more realistic against random ghosts, avoiding overly pessimistic moves.
- A well-designed evaluation function was crucial for winning consistently.

5 Challenges Faced

- Handling depth correctly with multiple ghosts.
- Implementing pruning without breaking the expanded state count.
- Balancing weights in the evaluation function.

6 Hyperparameter Exploration

- **Depth:** Increasing depth improved decisions but slowed down the game. Depth 2–3 was optimal.
- **Weights:** Larger penalties for ghost proximity reduced suicides, while higher rewards for capsules improved win rate.