PROJECT 02: THE MULTI-AGENT PAC-MAN

ARTIFICIAL INTELLIGENCE

CSE 537

INTRODUCTION

The report is based on the The Multi-Agent Pac-Man – a game enjoyed by all! In this project, we have implemented the classic version of Pac-Man with both the pac-man agent and its opponents – ghosts. The task is to improve the performance of pac-man agent by evaluating the states rather than actions taken in a particular scenario. Opponents are stochastic and therefore, are considered to be random throughout the implementation.

Files utilized –

- ✓ multiAgents.py

Files provided –

- pacman.py
- game.py
- util.py

Authors/Project Owners – Rishabh Khot (SBU ID: 112682983) & Shubhi Nigam (SBU ID: 112672816)
Stony Brook University

1. **Reflex Agent**

   Reflex agent is the most basic type of intelligent agents and performs actions based on a current situation. With pre-determined rules, this agent performs actions by searching its Knowledge Base based on the situation at-hand. A successful implementation will consider both food and ghost's locations and act accordingly.

   Implementation Details:

   - Number of nodes expanded: 4832
   - Total Cost: 1238.4
   - Running Time Complexity: O(b^m)

2. **Minimax Agent**

   In the implementation of the Minimax adversarial game agent, we taken into the account one Pac-Man agent's move and all the ghosts' responses into a single ply. Also, the task involves random number of ghosts.

   Implementation Details:

   - Number of nodes explored: 4137
   - Total Cost: 84
   - Running Time Complexity: O(b^m)

3. **Alpha-Beta Pruning**

   Alpha-Beta pruning is an adversarial search algorithm that seeks to reduce the number of nodes evaluated by the minimax agent. Our implementation took care of the logic by extending the same to incorporate multiple minimizer agents.

Implementation Details:

- Number of nodes expanded: 3790
- Total Cost: 84
- Running Time Complexity: O(b^m)

4. **Expectimax**

   Expectimax is a variation of minimax algorithm in which the outcome depends on a combination of player's moves and chance elements. The structure involves both min/max nodes as well as chance nodes, which take the expected value of a random event occurring. Considering the suboptimal choices by the ghosts, we implemented the model by taking into account the probabilistic behavior of opponents.

Implementation Details:

- Number of nodes expanded: 3563
- Total Cost: 513
- Running Time Complexity: O(b*m)

Note: b is the number of legal moves at each point and m is the maximum depth of the tree.

For calculating the stats, we have used python library *profile*. A *profile* is a set of statistics that describes how often and for how long various parts of the program are executed. These statistics can be formatted into reports via the pstats module.

CONCLUSION

Minimax and Alpha-Beta Pruning algorithms are identical and resulted in same cost during their implementations, however due to multifarious tie-breaking behavior, the actions performed by the agents may vary. The time complexities for both are exponential. Reflex agent, being a very basic agent, incurs a good amount of cost due to exponential time complexity. Expectimax, on the other hand, took moderate cost to implement in linear time complexity.