CSE 4618: Artificial Intelligence Lab

Lab 4 (Multi-agent Search)

Name: K.M.Tahlil Mahfuz Faruk

ID: 200042158

Islamic University of Technology, OIC

Gazipur, Bangladesh

Apr 9, 2024

Task-1(Reflex Agent)

Introduction

In this problem we have to design the evaluation function that will improve the reflex agent by considering the ghost location and the food locations.

Analysis of the problem

- Ghost locations and food locations of the successor state will provide some knowledge about how relevant it is to take that particular action.
- With that knowledge we can decide which action we should take and which should not.
- The higher the score, the better it is as it evaluates the successors for a particular action. Our target would be to evaluate the action.

Explanation of the solutions

- We can take the minimum Manhattan distance of the agent from the food. The closer the food the better it is. Because it would be optimal for the agent to eat some food rather than going to any other food palette that will penalize the agent as it sits idle rather than eating foods.
- We can take the minimum Manhattan distance of the agent from the ghost also to check how far the ghost is from the agent. The farther the ghost is, the better it is. Because the probability of getting eaten by the ghost will decrease.
- We need to check the proximity of the ghost also i.e. we need to ensure that the ghost is not too close to the agent. Then we need the agent to get away from the ghost by taking any other action.
- As a score we will use the reciprocals of the minFoodDistance and the minGhostDistance.
- We will add the reciprocal of minFoodDistance with the successorGameState's score as it is good to have the closest food for the agent. Also reciprocal is needed because smaller food distance will give us greater reciprocal value and that will provide us with greater score when added.
- We will subtract the minGhostDistance with the successorGameState's score as it is good to have the ghost further from the agent. Also reciprocal

is needed because greater ghost distance will give us smaller reciprocal value and that will provide us with greater score when we subtract with the score.

- Explanation with Example:
  - Let's say,The score was 500.
  - Now,
    - minFoodDistance=10
    - 1/minFoodDistance=0.1
    - Final Score =500+0.1=500.1……..(1)
  - Again,
    - minFoodDistance=100
    - 1/minFoodDistance=0.01
    - Final Score =500+0.1=500.01…….(2)
  - 500.1>500.01
  - So we can see that our score will increase more when we add reciprocal of smaller minFoodDistance
- Similarly, our score will increase more when we subtract reciprocal of greater minGhostDistance

Challenges and Solutions

No significant challenges were faced for this problem.

Behavior for Different Hyperparameters

- If we try to take euclidean distance from the agent to foods and the ghosts the autograder fails to pass the tests. Because we are not considering the walls between the agent and the ghost or food palettes.
- Manhattan Distance is comparatively good in this case and passes all the testcases.
- Without the proximity check of the ghost from the agent the autograder also fails some tests.

Interesting Findings

No interesting findings for this problem.

Task-2(Minimax)

Introduction

In this problem we have to design MiniMax agents and find the minimax action. The goal is to develop an agent that can select optimal actions to maximize Pacman's chances of winning while considering the moves of the opposing player (ghosts).

Analysis of the problem

- Pacman will try to maximize the score. So pacman is the maximizer.
- Ghosts will try to minimize the score. So ghosts are the minimizers.
- So we will need 2 different methods called minValue and maxValue and they will have different behaviors.
- Pacman signifies the agentIndex 0. So when agentindex is 0 we will call the maxValue function.
- Ghosts signifies the agentIndex 1 or higher. So when agentindex is 1 we will call the minValue function.
- Depth will signify how much depth the search should cover.

Explanation of the solutions

- We will solve the problem using recursion. Because we our agents are not limited to 2. We can have 1 pacman and more than 1 ghost. Each time the search will behave according to the agent recursively.
- Value function will perform minimization or maximization according to the agent,
- It calls the maxValue function when the agent is pacman. Pacman will try to maximize the score. So it will recursively call the value function for each legal action and try to get to the maximum depth or win state or lose state. These states will provide a score using the evaluation function. The maximized score between the successor and the current state will be taken. At the end an action will be selected based on the score.
- Same goes for ghosts. But this time minimization will take place instead.
- Basically it tries to find the best action based on the agent's behavior and tries to maximize the gain and minimize the loss in the game tree.

- In this way we are enforcing the pacman to maximize the score and the ghosts to minimize the score.

Challenges and Solutions

No Challenges faced for this problem.

Behavior for Different Hyperparameters

- Agent Index signifies the agent type in this case. If the agent index is 0 it's the pacman. Otherwise it's the ghost turn. This problem can have multiple ghosts at the same time.
- Depth signifies the maximum depth the game tree should be explored. It helps to reduce the time and space complexity of the search.

Interesting Findings

No interesting findings for this problem.

Task-3(Alpha-Beta Pruning)

Introduction

In this problem we need to design an Alpha-Beta agent that efficiently explores the minimax game tree.

Analysis of the problem

- The problem will be quite similar to MiniMax.
- In this case we won't explore the whole game tree. Rather we will try to optimize the search.
- To do so we will return from a node if its successor has greater(For Pacman) or lesser(For ghost) values than the current value.

Explanation of the solutions

- To solve this problem we will need to introduce 2 new parameters alpha and beta.
- Alpha represents the best (maximum) value that the maximizing player (e.g. Pacman) has found so far along the path of the current branch.
- Initially, alpha is set to negative infinity to ensure that any actual value encountered along the search path will be greater than alpha.
- During the search, if the algorithm finds a value greater than alpha, it updates alpha to that value. Alpha keeps track of the maximum value found so far.
- Beta represents the best (minimum) value that the minimizing player (e.g. ghosts) has found so far along the path of the current branch.
- Initially, beta is set to positive infinity to ensure that any actual value encountered along the search path will be less than beta.
- During the search, if the algorithm finds a value less than beta, it updates beta to that value. Beta keeps track of the minimum value found so far.
- Alpha and beta enables the program to prune some branches i.e. return without exploring some branches satisfying the relevant conditions.
- If at any point during the search, the algorithm finds a value for a node that exceeds the current beta (for a minimizing player) or is less than the current alpha (for a maximizing player), it indicates that the rest of the

nodes in that branch can be ignored because they will not affect the final decision.
- This pruning significantly reduces the number of nodes that need to be evaluated, leading to a dramatic improvement in the efficiency of the search.

Challenges and Solutions

No significant challenges are faced for this problem.

Behavior for Different Hyperparameters

- Same as Minimax hyperparameters.
- Alpha represents the highest value the maximizing player has found during the search in the game tree.
- Beta represents the lowest value the minimizing player has found during the search in the game tree.

Interesting Findings

No interesting findings for this problem.

Task-4(Expectimax)

Introduction

In this problem, we have to design expectimaxAgent that will take probabilistic modeling under consideration to make suboptimal decisions.

Analysis of the problem

- It's quite similar to the minimax problem.
- In problems like pacman, uncertainty is an issue. It's not guaranteed that all the actions of agents like ghosts would be predictable.
- We make the ghost actions dependent on a probabilistic model.

Explanation of the solutions

- In this case there would be no change in the maximizer. It would be the same as the minimax problem. Because pacman will always try to maximize the score.
- Ghosts won't be a minimizer in this case. Their actions will be uncertain or unpredictable. That can be introduced by probabilistic modeling.
- Ghosts are typically modeled as choosing uniformly at random from their legal moves. This means that each legal action has an equal probability of being chosen by the ghost. So,
    - Probability = 1/(Number of Legal Actions)
- By assigning equal probabilities to each legal action, the Expectimax algorithm effectively models the uncertainty in the ghost's behavior.
- Then we take the expected value of all possible outcomes weighted by their probabilities.
- This expected value represents the average or expected utility that the agent can achieve by taking a particular action, considering the uncertainty in the environment.

Challenges and Solutions

No challenges faced for this problem.

Behavior for Different Hyperparameters

- Similar to the minimax problem.
- We need to assign the currentValue to 0. Because we need to accumulate all the successor state scores and save it to the currentValue. That would represent the weighted average or the expected utility.

Interesting Findings

No interesting finding for this problem.

Task-5(Evaluation Function)

Introduction

In this problem we have to design a better evaluation function that will evaluate the current state.

Analysis of the problem

- Ghost locations and food locations of the current state will provide some knowledge about how relevant it is.
- With that knowledge we can evaluate the state.
- The higher the score, the better it is as it evaluates the currentGameState.
- We can take some hyperparameters on food distance and ghost distance such as foodValue, ghostValue and scaredGhostValue. They would have a significant effect based on the state discussed afterwards.

Explanation of the solutions

- In the solution we take some hyper parameters such as foodValue=10, ghostValue=10 and ScaredGhostValue=100.
- These hyper parameters will be multiplied with the reciprocal of the foodDistance and ghostDistance to give reward or penalize the score.
- For food distance we take the minimum manhattan distance and multiply the reciprocal with foodValue then add with the score. Because the minimal the food distance is it will be better for pacman score as not eating food pallets will penalize pacman score.
- For ghostDistance we need to keep in mind that ghosts can be scared. During the scary time, we can give more rewards to the score. That's why in this case we have taken 100 as ScaredGhostValue.
- But in normal cases i.e. when the ghost is not scared, we reward only 10.
- Now similarly we multiply the hyperParameters with the reciprocal of the Manhattan distance of each state with the score. This time we didn't take the minimal Manhattan distance. Because we have to evaluate two different situations.
    - When the ghost is scared.
    - When the ghost is not scared.

- If the ghost is scared we multiply the Manhattan distance reciprocal with scaredGhostValue and add it with the score as a symbol of rewarding the score.
- Else we multiply the Manhattan distance reciprocal with ghostValue and subtract it with the score as a penalizing the score.
- If the Manhattan distance is 0 then the ghost has eaten the pacman. So we will penalize that state the maximum.
- Likewise task 1, we will use reciprocal values here also for convenient reasons explained in task 1.

Challenges and Solutions

No significant challenges were faced for this problem.

Behavior for Different Hyperparameters

- Increasing the scaredGhostValue significantly might lead to riskier behavior as the agent becomes more inclined to pursue scared ghosts.
- This might sacrifice the motive to achieve higher scores. Because not eating food palettes will penalize the score.
- Increasing foodValue might lead the Pac-Man agent to prioritize consuming food pellets over other objectives, such as avoiding ghosts or maximizing overall score.
- A higher ghostValue would amplify the significance of ghost proximity in the agent's evaluation function. This could result in more risk-averse behavior, with the agent prioritizing evasion of ghosts to minimize the risk of losing a life.
- Finding the optimal balance between these hyperparameters is crucial for achieving desired gameplay outcomes.

Interesting Findings

No interesting findings for this problem.