# Assignment 2. Multi-Agent Search Project Report

Mehmet Enes Erciyes, 68906
Computer Engineering

## I. Question 1: Features In Reflex Agent

### A. *What are the features you used in your evaluation function for your reflex agent?*

The features I used was:

- Game Score
- Reciprocal of the distance to nearest food
- Reciprocal of the distance to ghost
- The number of foods left

I used a linear combination of these features in the following way:

$$eval = gameScore + \frac{1.5}{\max(distanceToFood, 1)} - \frac{2}{\max(distanceToGhost, 5)} - 6 * numberOfFoodLeft$$

### B. *Why did you choose them?*

Game score is a good indicator of overall objectives of Pacman. It penalizes longer games, running into a ghost and rewards finishing the game and eating more dots.

Distance to nearest food is important for keeping Pacman moving. Unless we reward moving towards the food, Pacman does not move if there is no food reachable within the specified depth. However, lower distance should mean higher eval function. Therefore, I first tried using the negative of food distance, but using the positive reciprocal of it worked much better.

Thirdly, reciprocal of min distance to ghosts is used to make Pacman evade ghosts.

In the last two reciprocal features, I used a max function in denominator to get rid of the issue of dividing with 0 and huge results when distance gets close to 0.

Lastly, I used the number of foods left to encourage Pacman to pickup food whenever it can.

### C. *Do you think using the reciprocals or negatives of some values is a good idea and why?*

Yes. Because if some features are smaller, then the state is better. As I realized, it may sometimes also be preferable to choose one over the other. Distance to food and distance to ghost worked better with reciprocals because when distances get larger, they should have lesser effect while when they get closer, they should have greater effect.

## II. Question 2: Minimax vs Alpha-Beta Pruning

### A. *In your tests, were you able to see any speed difference between the MinimaxAgent and AlphaBetaAgent, between pacman actions?*

| Algorithm | Moves in 20 seconds |
|---|---|
| **Minimax** | About 12 |
| **Minimax with Alpha/Beta** | About 35 |

There was a clear speed difference as seen in the table.

### B. *If so, why and if not why not?*

Alpha/Beta Pruning greatly reduces the states that are expanded by pruning those that come after a value lower than the current max(alpha) in max layers and higher than the current low(beta) in min layers. This makes the computations much faster.

### C. *Is there any situation you came across that highlights this?*

I believe this question asks the situations/states with the most different decision time. The most different scenarios were where Pacman stands in an open area with lots of options to choose. In other places, Minimax is also reasonably fast. However, in open areas, Minimax has lots of states to expand, therefore becomes very slow.

## III. Question 3: Exactness With Minimax and Alpha-Beta

Yes, the moves were the same. Alpha-Beta Pruning does not change the optimal action because it only prunes the states that are impossible to be chosen. This way, actions remain optimal (with the defined depth limit of course) however, the number of states to expand decreases.

## IV. Question 4: Expectimax Comparison

This time, Pacman moved in a similar speed with the Minimax algorithm without Alpha-Beta Pruning. **This was expected, because Expectimax works the same way with Minimax expanding the same number of nodes**. Expectimax only takes the expected value of a node's children as its value instead of the minimum in adversary layers.

## V. Question 5: New Evaluation Function

My new evaluation function got 6/6 in Autograder. However, I did not change anything from the reflex agent logic. It worked quite fine for the Expectimax too. Only thing I changed was that the first function mapped State, Action pairs to evaluation values while this function maps only a state to a value.

## VI. Question 6: Tuning Weights

Yes, I tuned the weights as seen in the answer Question I – A. The tuning process was try-fail. I first thought I needed to scale each feature to the same range and then apply weights. However, it proved to be unnecessary for this project. I weighed food count and food distance higher to prevent Pacman from choosing to stop rather than eat food. However, this caused Pacman to not evade the ghost sometimes. So I increased its weight to two. I did not touch the game score.