**INF 444 ARTIFICIAL INTELLIGENCE**

**THE PAC-MAN PROJECT**

Ayça Kiremitci Halit Fırtına Hazar Belge Muhammed Erdoğan

18401799 17401764 18401783 17401769

**About Expectimax Algorithm**

The “expectimax” function takes in the current game state, an action taken so far, the remaining depth of the search, and the index of the current agent. It returns an action and a value. If the game is over or the depth is 0, it returns the current action and the result of calling the “evaluationFunction” on the current game state. Otherwise, if the current agent is Pacman, it returns the result of calling the “max\_value” function on the current game state, action, depth, and agent index. If the current agent is a ghost, it returns the result of calling the “exp\_value” function on the same arguments.

The “max\_value” function returns the action and value that give the maximum value among all the possible actions Pacman can take at the current state. It does this by calling the “expectimax” function on each possible successor state that results from Pacman taking legal action and returning the action and value that resulted in the maximum value.

The “exp\_value” function returns the expected value of the current game state, given that the ghosts will act optimally. It does this by calling the “expectimax” function on each possible successor state that results from the ghosts taking legal action, averaging the values returned by “expectimax”, and returning the current action and this average value.

**About Better Evaluation Function**

The “betterEvaluationFunction” appears to be a heuristic function that is used to evaluate the utility of a given game state. It takes in a game state and returns a score indicating how good or bad the state is for Pacman. The score is computed based on several factors such as the distance to the nearest active ghost, the distance to the nearest scared ghost, the number of remaining capsules and food, and the distance to the nearest capsule or food.

To continue, the "getAction" function is the method that is called to determine the next action that Pacman should take. It does this by calling the “expectimax” function with the current game state, an empty action string, the depth of the search, and an agent index of 0 (corresponding to Pacman). The “expectimax” function is then called recursively to explore the game tree and evaluate the utility of each possible game state. The “expectimax” function returns the optimal action to take and the corresponding value of the resulting game state. The "getAction" function then returns this optimal action.

The “betterEvaluationFunction” is used to evaluate the utility of each game state that is reached during the search. It helps guide the search towards more promising paths by giving higher scores to game states that are more favorable for Pacman. The "expectimax" algorithm combines the use of this heuristic evaluation function with the idea of maximizing for Pacman and minimizing for the ghosts to determine the optimal action for Pacman to take.

**Our Strategy (Heuristic)**

First, we tried to find the weight multipliers in our evaluation function with a genetic algorithm, but we gave up thinking that this process would take too long despite all the optimizations we made due to the long game times. Then, we tried to create certain coefficient sets by considering different situations, and we tried to obtain the maximum scores by obtaining the necessary coefficient adjustments in our situations with a few trials and errors.

There are 2 main configurations:

* Multipliers
* Additional Conditions

**Multipliers:**

As it can be easily understood from the coefficients below, we have determined our PacMan's top priority job as capsule meals. The main reason for this is to both increasing points while be able to eat ghosts and creating a safer environment at the same time. PacMan's next main job is to eat ghosts. In addition to these 2 high coefficients, we determined the coefficients for eating food, approaching scared ghosts, and getting away from active ghosts, respectively.

* scoreMultiplier = 1
* numberOfFoodsLeftMultiplier = 10
* numberOfCapsulesLeftMultiplier = 100000
* distanceToClosestFoodMultiplier = 0.15
* distanceToClosestCapsuleMultiplier = 0.4
* distanceToClosestActiveGhostMultiplier = 1
* distanceToClosestScaredGhostMultiplier = 2
* numberOfScaredGhostMultiplier = 10000

Text

Description automatically generated

**Additional Conditions:**

* If there is an active ghost very close
* If there is a scared ghost too close

Text

Description automatically generated

* if no active ghost
* if no frightened ghost
* if there are no cookies left

Text

Description automatically generated

**Evaluation Score**

We’re setting current score with its multiplier and square as base for evalScore variable. Then respectively we’re taking squares of each coefficient to create better model.

Text

Description automatically generated