INFO8006: Project 2 – Report

Alyssa Di Matteo – s201486
Manon Gerard – s201354

October 27, 2022

1 Formalization

State space :
The state space contains the Pacman position, the ghost position, a boolean matrix that contains the position of the food, the directions of the ghost and the next player to play.

Initial state :
The initial state includes the initial position of Pacman, the initial position of the ghost, a boolean matrix that contains the initial position of the food, the initial directions of the ghost and the initial player (Pacman).

Player function :
This function tells us which player has the move. In our case, it returns (p+1)%2. If this is 0 Pacman has to move and if it is 1 the ghost has to move, p corresponds to the player.

Actions :
The two agents are allowed to go in 4 directions, which are North, South, West and East. Agents can also stop, but only if they are surrounded by 4 walls, which isn't the case here.

Transition model :
The transition model allows to return the state of the game after one of the two agents has performed a legal action.

Terminal test :
The terminal state check if the game is over, so when Pacman ate all the food, if he was eaten by the ghost and also if Pacman is doing a cycle.

Utility function :
The utility function assigns a numerical value at the end of the game, which corresponds to the state where either Pacman or the ghost has won. In our case, the utility function will be the score of the game which corresponds to :

$$\text{utility(s, p)} = \begin{cases} 10 * (\text{number of eaten food dots}) - 1 * (\text{time steps}) + 500, & \text{if p = Pacman} \\ 10 * (\text{number of eaten food dots}) - 1 * (\text{time steps}) - 500, & \text{otherwise} \end{cases}$$

2 Minimax

a. No, minimax isn't guaranteed to be complete as it can perform cycles. If we prevent cycles, then it can be complete.

b. A cycle is only advantageous if Pacman has no choice to do one. Indeed, in case Pacman does not have any path to follow, instead of losing he could go through a cycle, which would allow the game to continue.

c. To guarantee completeness, the minimax algorithm gives an extreme value to the cycle, which will allow Pacman to avoid it and keep an optimal solution unless it has no better choice. To best represent the reality, we chose ∞ and -∞ as extreme values because when Pacman will go through a cycle it will always decrease the score and therefore arrive at $-\infty$.

3 Heuristic

Cut off :

Our cutoff function decides to stop the expansion of states if Pacman has won or lost and if the depth of recursion is larger than our fixed value. We have chosen to fix this value to 5 as it was a good compromise between the time and the score. With lower values, the score was smaller but it took less time and fewer nodes. Therefore, we prioritized a better score with a reasonable time.

Heuristic :

Our heuristic function returns an estimate of the score of a state. To compute it, we started by finding the smallest rectangle surrounding the foods, which will allow us to compute the minimum distance Pacman must travel horizontally and vertically to get the farthest foods. With these distances, we have established the distance that Pacman still has to travel to go to all the sides of the rectangle. This distance is calculated under the assumption that no wall blocks Pacman's path. This calculation is what we called h. We subtracted it from g, which initially represented the score. To avoid cycles, we looked if the state we are in has already been traversed and if so we subtracted from g, 10 exponent the number of times we have visited a state. We have chosen to put a term with an exponent because it can become very large quickly, which will allow us to disadvantage the state and thus avoid cycles.