INFO8006: Project 1 - Report

Jad Akkawi - s216641
Christina Broche- s195292

October 28, 2021

1 Problem statement

a.

- $s = (M, P, G, t)$ where $M = \{0,1\}^{n*m}$ with $\{(i,j)|M_{ij} = 1\}$ is the set of position of foods dots, and$\{(i,j)|M_{ij} = 0\}$ is the set of position of no food dots.

$$P = (x,y)_{1\leq x\leq n, 1\leq y\leq m} \tag{1}$$

Legal positions of Pacman

$$G = (x',y')_{1\leq x'\leq n, 1\leq y'\leq m} \tag{2}$$

Legal positions of Ghost

$$t \in \mathbb{N} \tag{3}$$

M = Matrix, P = Pacman position, G = Ghost position, t = Time.
s_o= $(M_0, P_0, G_0, 0)$ we consider that t = 0 at initial state to avoid penalizing Pacman with an initial time step. Pacman starts the game.

- player(s=(M,P,G,t)) = (t+1)mod 2. This means that when it's not Pacman, it's the Ghost. player(s_0) = (0+1) mod 2 = 1
- action(s) = $\{(1,0),(-1,0),(0,1),(0,-1)\}$ except if player(s) is next to a wall. The possible actions are North, South, East, and West.
- transition model:

$$s' = result(s = (M,P,G,t), a) = (M', P', G', t+1) \tag{4}$$

$$with \begin{cases} P' = P + a, G' = G & if \quad player(s) = 1 \\ G' = G + a, P' = P & if \quad player(s) = 0 \end{cases} \tag{5}$$

$$and \quad M' \quad same \quad as \quad M, \quad but \quad M'_{ij} = 0 \quad if \quad P' = (i,j). \tag{6}$$

This means that state s' is the result of player(s) (the Ghost or Pacman) taking an action a in the state s.

- terminal state: $terminal(s = (M,P,G,T)) = \begin{cases} 1 & if \quad M = \{0\}^{mn} \quad or \quad P = G \\ 0 & if \quad M \neq \{0\}^{mn} \quad and \quad P \neq G \end{cases}$
The terminal state is achieved when there is no more food (Pacman wins) or if the ghost eats pacman (Ghost wins)
- $utility(s = (M,P,G,t), player(s) = 1) = -t + 10 * \sum_{ij} q_{ij} (with: Q = M_0 - M) + \begin{cases} 500 & if \quad M = \{0\}^{mn} \\ -500 & if \quad P = G \end{cases}$

b. $utility(s, ghost) = -utility(s, pacman)$ because $utility(s, Pacman) + utility(s, Ghost) = 0$ in a zero sum game. We say that the total payoff to all players is constant for all games.
In fact, for two-player games, agents share the same utility function, but one wants to maximize it while the other wants to minimize it. In our case, Pacman wants to eat all dots to maximize utility while Ghost wants to eat him before he finishes the food dots to minimize the utility value.

2 Implementation

a. In the game of Pacman, maps can be big enough and states may be visited many times over, specially with sub-optimal Min players. The tree of such situations is not finite and therefore it cannot guarantee completeness.
From Pacman's point of view, going in cycles would increase the amount of time steps without any more food dots eaten and thus will decrease the utility value so no advantage of course.
We must avoid actions that lead to states that were previously visited. This way utility function and terminal states are perserved.

b. ***code Minimax.py***

c. ***code Hminimax0.py, Hminimax1.py, Hminimax2.py***
Hminimax0 and Hminimax1's cutoff-Tests are as follows:
$$\text{cutoff-Test(state, depth)} = \begin{cases} True & if \quad depth = 4 \quad or \quad Terminal - State(state) = True \\ False & else \end{cases}$$

Cutoff-test initially set at depth = 4 because it represents 2 actions for MAX (Pacman) and 2 actions for MIN (Ghost). For Hminimax0 raising the depth to more than 4 did not have an effect on the score. While Raising Hminimax1 depth to 5(= Hminimax2) led to some progress in the score.

Therefore, Hminimax2's cutoff-Test is as follows:
$$\text{cutoff-Test(state, depth)} = \begin{cases} True & if \quad depth = 5 \quad or \quad Terminal - State(state) = True \\ False & else \end{cases}$$

For Hminimax0: We started with a straightforward yet very indicative heuristic. We considered that as long as Pacman is far from the ghost and close to the food, it's a good thing. The distance between Pacman and ghost is indeed something that Pacman should take into account but we nevertheless decided to lower the impact of this distance (to half) on the eval function because this distance made Pacman think that he is at an advantage and this resulted in useless movements (oscillations), especially infront of the dumby ghost.
Thus eval_0 is as follows: $eval\_0(state) = state.Score() - Distance(P - F) + Distance(P - G)/2$
Distance() being the manhattan distance function.

For Hminimax1 and Hminimax2: Taking into account only the closest food dot is not a sufficient estimation of the remaining cost for Pacman, this is why we added the estimated food path using our invented pathfinder *the Split Grid*:

From the food dot closest to Pacman we divide the grid into several zones, and the next food dot of choice will be the closest one of the zone that has the most amount of food dots. This goes on until there is no more food dots on the grid and the chain of distances is complete.

We obtain eval_1 and eval_2 as follows:

$$eval\_1or2(state) = eval\_0(state) + Split\_grid(state) \tag{7}$$

For all eval functions, we did not add +500 (at winning) because it's a constant so not worth adding when comparing expected utility of different actions.

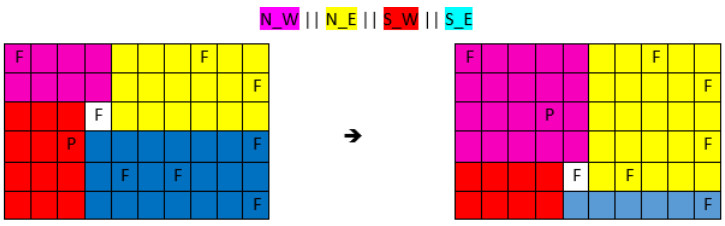

Figure 1: The split grid function

3 Experiment

a. ***figure 2, 3,and 4 page 3***

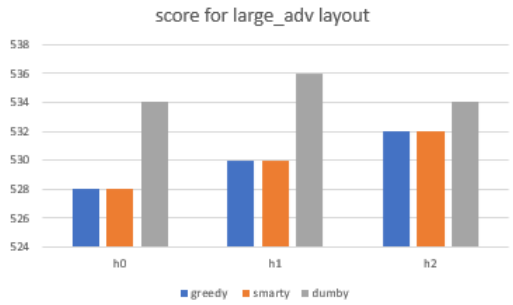

Figure 2: Graphic of the score for the large advanced maze

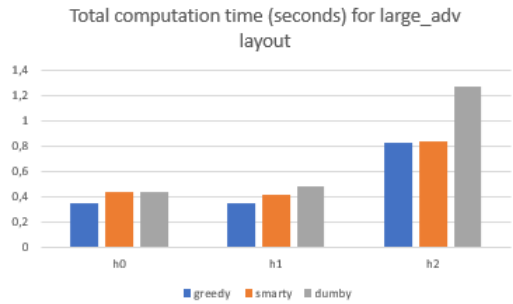

Figure 3: Graphic of the time for the large advanced maze

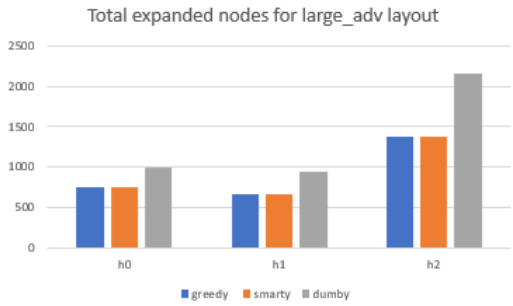

Figure 4: Graphic of the expanded nodes for the large advanced maze

b. ***figure 5 page 4.*** Regarding the evaluation functions, the results has shown that the quality of $eval_0$ is lesser than that of $eval_1$, this is because the expected utility of $eval_0$ does not reflect the actual prospect of winning for Pacman and this for all type of Ghost. Now for Hminimax1 and Hminimax2 that share the eval_1or2, we can see that the horizon effect was noticed for greedy and smarty ghosts, where increasing the depth of our lookup to 5 has led Pacman to get a better score than when it was 4. On the contrary for the dumby ghost, going to a depth of 5 has led to over-estimating the behavior of the ghost which did not reflect reality and Pacman got a lesser score.

Finally, dumby has more expanded nodes than smarty and greedy because in my opinion dumby ghost is far from Pacman so he has more available actions than when ghost is right next to him and following him.

large_adv layout

Total score :	*h0*	*h1*	*h2*
greedy	528	530	532
smarty	528	530	532
dumby	534	536	534

Total computation time (seconds):	*h0*	*h1*	*h2*
greedy	0,3426	0,351	0,827
smarty	0,433	0,413	0,838
dumby	0,438	0,475	1,266

Total expanded nodes :	*h0*	*h1*	*h2*
greedy	751	664	1380
smarty	751	664	1380
dumby	993	934	2152

Figure 5: Results of our Heuristic functions