Pacman Contest Report

Artificial Intelligence B

Remco van der Heijden 4474139
Dennis Verheijden 4455770

Search Algorithm

We used the Alpha Beta Pruning search agent. We wanted to create a search agent that calculates a few steps in advance. This way he would calculate which action has the best outcome, not only for one next state, but for multiple states. For example: if he would go into a corridor he could be fine, in a next step a ghost would come into the corridor and now pacman would be trapped. With this algorithm the agent is able to calculate more states, so the chance that pacman would be trapped is minimized.

We wanted to focus on staying alive and finishing the game, with the highest possible score. Using this algorithm would however put us in a disadvantage in the lower levels. This because the principals of this algorithm says that “Both players, pacman and the ghosts, should always pick the best option available to them”. This however is not the case in the lower levels, because there are mainly random ghosts and ghosts that walk a fixed pattern and that is not the best option available to them in most scenarios. What this means is that the algorithm plays it safe since it will not go to positions where the ghosts possibly could have a better score than pacman.

The search algorithm works with a dynamic depth. Since every level has different dimensions we could determine which level was running by requesting the wall height and width. We experimented with the depth per level and the time the pacman agent could maximally take to calculate a move. We found that for example level0 works best with depth 2, while level3 works best with depth 3.

Upping the depth means that another layer of searching is added. This means that you calculate another action of the ghosts and another action of pacman and pick the best outcome for pacman. Ultimately pacman does not calculate the best move for one state but for multiple states. We believe that this search agent has the most potential, that’s why we sticked with it. We just have to cope with the higher computational time and take it into account while making our heuristics.

We also made a “failsafe”, normally the agent would make a random action if the time to compute was exceeded. We made it that pacman would first look if his previous action is legal and if it is do that move. In most cases we found it better to do this than to do a random move. If the previous move wasn’t legal, then we would still use a random move.

Heuristic

We created our own heuristic based on: the distance to the nearest food, the amount of time remaining to eat near ghosts, positions that are already visited, whether the state is a losing state, if pacman is trapped, if power capsules and ghosts were close by, the score of pacman itself and some world specific heuristics.

Because we used the Alpha Beta Pruning agent, we already had relatively high computational times. So we had to make sure our heuristics were coded efficiently so that they will not require another huge load of computational time. We took a few measures to do this: First we calculated all the static properties of the world at the initial state, this saves some time because this doesn't have to be done anymore during pacman runtime. The second measure was that we made sure everything we used was only calculated once. For example: we made a list of properties for each ghost in the field. This allowed us to calculate all ghost data once for every state. These properties are the ghost index (0 to 3), the maze distance of pacman to the ghost and the scaredTime for the ghost.

The first two heuristic calculations are simply checking if the pacman state was a losing state, resulting in a really low heuristic, and adding the current score to the heuristic. The latter was really helpful for making pacman want to eat ghosts.

The next heuristic calculation was to calculate the minimum distance to the nearest food. We tried to do this as efficient as possible. We first tried to only take foods in account that were close to pacman (in a radius of 3 spaces away from pacman). We then used the maze distance, also known as the true distance in the maze, to calculate the distance to these foods. Lastly, the minimum distance of this list was chosen and used this as a negative heuristic since you want to be close to the nearest food.

If pacman was close to a power capsule and to a ghost, than pacman was determined to eat the power capsule. After eating the power capsule the heuristic gets: higher for being near scared ghosts and a bit lower for eating food (as compensation for the score heuristic increasing) to make sure pacman would chase ghosts and not keep eating foods. The heuristic score would not increase again for eating a power capsule if there are ghosts that are still scared. This results in an actively ghost chasing pacman.

If there is one food left and one or more power capsules, pacman would first eat the power capsule. This because it would result in a higher score if pacman eats as much ghosts as possible.

To make sure the agent doesn't get stuck in a loop, we added loop-detection. The agent has a list of the 15 previously visited positions. If the agent visits a position more than once, the heuristic is lowered for this place. When the same position is visited again in the last 15 positions, the heuristic is lowered even more. This way the agent will avoid situation where he would get stuck anyway because of pattern ghosts.

When the agent looks at possible actions from this state it also checks if there is a risk of getting trapped by ghosts. To do this, it looks at all the possible actions for both the pacman agent and the ghost agents. Then the agent removes all the possible ghost positions from the list of pacman positions. If there are no possible actions left, the state is a state in which pacman is trapped.

Pacman has the knowledge to determine in which level it is. He can use this knowledge to boost his score even more. In lower levels (level0 and level1) pacman would not finish the level if there are still scared ghosts. In lower levels this will result in a higher score because the chance of eating a ghost in these small levels is relatively high. Another thing that pacman will do is activate the desire to eat a power capsule in the higher levels (level2 and level3) as soon as he enters the world. We found that this greatly increases his chance of surviving the first moves in these worlds.