John Beardwood, Aaron Steele

CMPS 140

Tournament Report

Participating in a Pac-Man AI tournament was an interesting and unique experience. Brainstorming and programming possible strategies allowed us to be creative while implementing technical material. The independent nature of this assignment required us to think critically on our own, which will help our problem solving skills in the future. Being able to come up with creative solutions is incredibly important in AI design, and this tournament facilitated this practice.

The process of designing our AI was a journey, our initial ideas did not work out, and we ended up taking some unexpected routes. Initially we wanted to implement a minimax style algorithm so that we could predict our enemies actions, but this was not as easy as we had initially hoped. Not only did we run into an extreme number of frustrating errors, but the uncertainty of our opponents positions made these predictions unnecessarily difficult. At this point we decided that minimax was not worth the effort that would be required to make it useful.

Once our initial plan was scrapped, we examined the example agents more closely, and realized they were simple reflex agents. After ruminating on this, we postulated that we could make an effective reflex agent given that we could make a powerful evaluation function. In terms of general strategy we went for an all-out offensive approach. The idea being that if our two offensive pac-men were smart enough they could get all of the enemy food faster than they could get all of ours. The features that were already present in the baseline agent were the successor state's score, and the distance to food. We kept these features (eventually weighting them differently) as well as adding our own.

The first problem we noticed with the baseline code was that it was pretty much static: The agents performed essentially the same action every run, and died in approximately the same place. To help rectify this, we added a new feature: the distance to the nearest enemy. We weighted this so the agents attempted to avoid the area they thought the enemy agents were in. Once this was added and properly weighted, our pac-men performed more dynamically, and didn't take the exact same route every time. They did, however, follow one another almost exactly. This was very problematic for three reasons. Firstly, when two pac-men are next to each other, only one of them is gathering food. Secondly, one will have nothing to eat once it arrives at its goal. Finally, if one of them died the other would die as well in almost every case, given the position of the attacking agent. This obviously needed to be rectified in order to improve our game.

Our first attempt to eradicate this problem was to have one pac-man prefer food that is close, while the other tries to get food that is most far away. This did not behave as we had hoped. One agent would run forward and collect food while the other would sit pretty much perfectly still. We believe that this is because the way we coded it meant that the agent was reevaluating its target without actually moving towards it. Obviously because this technique rendered one of our players useless, we reverted the change and looked the problem over again.

Our next concept was to encourage the bots to avoid each other. The feature we implemented was simply the distance between our players. Penalizing the bots for being near each other forced them to take more interesting routes than they would have initially. It increased the speed at which food was eaten as they would get in each other's way less often. After adding this functionality we noticed that double-offense had an added advantage: Having two agents on the opposing side of the map can confuse the opponent's defense agent if they have one. It can cause the defender to be indecisive on its target.

While all of this seems very simple in hindsight, we were mostly satisfied with our performance in the tournament. The key for writing effective AI seems to be less in the complexity of the solution, but the analysis of the problem itself. Even just adding a single feature increased the effectiveness of our agent to a point that simply adjusting weighting can never accomplish. Overall we enjoyed this process and are happy with our 13th place finish.