Trevor Barnes

CMPS 140

KMN AI

Tournament Write Up

The goal of this project is to create an AI capable of defeating an opposing AI in a capture the flag style game. This problem can be described as two basic groups of sub problems. The first is that the AI must be capable of solving a maze and collecting pellets. The second sub problem is that the AI has no information about how its opponent solves the maze. To this end the AI has to be built with the ability to react to an opponent's movements in real time. These two problems make up the overarching problem. An AI must compete against another AI, and collect the pellets in a more efficient manner to maximize its score while keeping in mind that the other AI may try to stop it.

Since there are four agents total, two per team, and two main strategies, offense and defense, there are four types of strategies that make sense to look at. The first is making both agents stay on defense. This is obviously a bad strategy because if an agent is always on defense it can never collect pellets and will lose to any AI that has any offense at all.

The second strategy is sending both agents out on offense. This strategy has advantages and disadvantages mainly depending on the enemy's policy. If the enemy has two defensive AI then the offensive strategy works but is less effective due to having two agents to work against. If the enemy has also sent both of its agents on offense then whichever pair of agents collects pellets faster wins. If the enemy has one agent on offense and the other on defense then the two offensive agents just have to collect pellets faster than the enemy's single AI does while also avoiding the single defensive agent.

The third strategy is having one AI on defense and one on offense. The defensive agent would try to catch any enemies that enter its territory. The offensive agent would try to avoid enemy defenses while collecting pellets. This strategy has a weaker offense or defense when compared to strategies that solely focus on type of strategy. The idea of this would be that the defensive agent buys the offensive agent extra to collect pellets by slowing the rate at which the enemy can collect pellets.

The fourth strategy is the most complex and seemed intuitively the best at first. This strategy would make either one or both of the AI adaptive, meaning they would switch from offense to defense depending on the situation. Although this seemed the best at first it has a few inherent flaws. The first being that the travel time for an AI to move from offense to defense or vice versa would destroy both its ability to guard its side or collect enemy pellets. Additionally the added noise built into the problem makes detecting when to move from side to side much more difficult. These problems made this option much less appealing than I originally thought.

Ultimately I chose to go with having both AI on offense, because of how I anticipated the other AIs to work. No one was going to do two purely defensive AI, because it doesn't work very well. In the case that the enemy also did two offensive agents I just had to build my agents to collect pellets faster than the enemy did. I was most worried about the case where the opposing AI had one offensive agent and one defensive agent. If done well enough this scheme could potentially counter mine completely. In the worst case the one defensive agent would be able to effectively stop both of my offensive ones.

To combat this I originally planned to have a minimum distance the two agents had to keep away from each other. This would stop a single defensive agent from countering both. However, this created problems where the minimum distance would stop the AI from collecting pellets that were between the two. Ultimately in the worst case this distance caused a very big loss of efficiency when collecting pellets. Instead, I split the agents into top and bottom. I split the enemy's side in half horizontally and assigned one agent to collect pellets on the top and the other to collect pellets on the bottom. There were still some problems with this. For example, if both agents were near the dividing line at the same time then it was possible for an enemy defensive agents to easily get both my AI. The odds they were that close at any given time was low so it's an acceptable solution.

I used alpha beta pruning to pick moves. It seemed like the best choice since it is fundamentally an adversarial search, and therefore encodes the existence of an enemy automatically. Unfortunately the computational time limit on the server was a factor I had to take into account when deciding how deep on the tree I could search. I never really got my AI to work consistently or well so I never uploaded it onto the server, but when I could get it to work locally I searched to depth four on the tree which still may have been too slow for the sever.

My evaluation function placed highest priority on moving the agents to their respective sides on the enemy side. Then it prioritized moves that stopped it from being caught if it was being chased. If these conditions were met then the AI focused collecting pellets as best as it could. The agents tried to find the longest string of pellets it could find given its current location. If it didn't have any pellets in its immediate range then it moved in a random direction until it found something to eat.

I was unable to test how my agents worked, because at this point there was problem with my code I was unable to fix. This leaves me only able to speculate how well my agent would have done. I think many of things I did were good. There were some compromises I wasn't happy with such as the division between top and bottom agents. Preferably I would have found an effective implementation that allowed agents to stay away from each other while also effectively gathering pellets. From what I heard in class my agent shared many basic strategies with some of the agents that placed well. Although, these AI had many more refinements that made them much better than mine. If my AI had run it would have probably come around the middle, maybe worse, mainly because I'm not confident in my AI's ability to deal with defensive AI effectively. Additionally, I was unable to refine my AI because I couldn't get it to run consistently in the first place.

I learned a lot during this project. I spent more time than I thought I would just designing how I would implement my AI. It really helped to think about how other people would design their agents when I was making mine. Really for this project your AI just had to be good enough to beat other people in the class. There wasn't an actual test for if your AI was good or not which was interesting. Normally programming assignments in classes have a right or wrong answer. Working on an assignment where this wasn't the case was definitely more interesting then working on an assignment with a solution.

A teammate would have definitely been helpful. At the end of the assignment I was completely lost and was unable to find the problems causing my code to fail. When I ran the game it would crash. I wasn't able to determine what caused the crash, just watching the game run it seemed pretty random when it crashed and when it didn't. A partner would have definitely helped in the debugging process and I might have been able to fix my agent in time if I had one.