Dakota Sliter

Austin Auckerman

Noah Betz

**AI Project 1**

**Introduction**:

Overall, this project was challenging from a programming aspect, but our group found it surprisingly fun to experiment with our agent behavior. Logistically, we had some issues getting together synchronously to work on the project, so we ended up working asynchronously most of the time and giving one another a summary of what we worked on. This included issues we encountered and solutions we came up with. We also threw around many ideas on how to approach agent behavior and discussed which ones we’d like to attempt implementation of. In the end, we settled on a few key behavioral features for our agents, building off of improvedTeam.py:

* One agent dedicated to defense, one for offense (in the beginning of the match)
* Fix issue with offensive agent cowering in dead end corners when enemy ghost was near
* Add points of interest for defensive agent to stay near (patrol)
* Optimize feature weights

We also established some nice-to-have agent behaviors, although we were not able to successfully implement these, either due to difficulty or time constraints:

* Defensive agent dynamically patrols or waits in areas of high-density food
* Strategic usage of the capsule to kill enemies
* Add feature to prioritize offense agent moving to areas of high-food density over lower densities. For example, if there is 1 food 3 moves away, but 3 food 5 moves away, attempt to prioritize pursuing the 3 food cluster
* Write genetic algorithm that finds the optimum weights of our agent features

This writeup will be broken into three main phases: *Exploration*, *Experimentation*, and *Optimization*.

**Exploration:**

To get started on the project, we first made a Discord server for all our communication purposes. We then started a GitHub repository with the base project structure and everyone set up their hosts for remote pulling and pushing. After running a few sample games to observe agent behavior, we took to analyzing the starter code line-by-line and commenting what each did. This served the purpose of familiarizing us all with the overall strategy employed by improvedTeam.py and getting an idea of what types of methods and properties were available to us. The commented code may seem a bit redundant in places, but it did help to understand the task better.

Next, we created a shared Google Document to hold some ideas we had about how to approach our agents’ behavior. We decided early on that we wanted one agent to remain on defense to avoid giving up any points in the beginning, so our other agent can focus on building up a lead. One idea was to implement a type of behavioral switch in our offense agent, such that when our point lead is greater than a set amount, the agent returns home and begins patrolling friendly food areas. Most of the ideas we came up with are summarized above in the introduction section of this document.

Observing how the games played out, it was clear that the improvedTeam.py offense agent “cowering” problem was a high priority fix for us. We proposed solutions of adding checks to prevent entering dead ends, even for food, as well as maintaining a state variable of last seen enemy positions. This would allow our agent to choose actions that avoid areas where the enemy is likely to be headed, and to stay away from fatal corridors. Additionally, we decided that the defensive agent could be improved if it prioritized certain areas of the map for patrol and didn’t stray too far from those points of interest, even to pursue an invading enemy. This way, an enemy agent would fail to lure the defense away from key chokepoints or food sources.

We now had a long list of ideas for our agent behavior, but the list was longer than we had time to experiment. As such, we rated the ideas based on priority:

1. Fixing the cowering behavior
2. Add points of interest for defense agent to stay close to
3. Offense agent preference for more food-dense areas
4. Switch offense to defense after a significant point lead
5. Target capsules strategically

Next, we began to experiment with implementation of these features and optimization of weightings.

**Experimentation:**

Ideas remaining:

* Switching offensive agent to defense centered around a different node than the other agent
* making it so when the offensive agent is in defense land it can go after guys, or will consider waiting breifly (around a corner or at edge?) for ghost to enter
* make it so when both agents are defending, the closer of the two goes after the guy and the other ignores him
* make it so offensive agent doesn’t take dead end paths when running away
* make it so offensive agent remembers what the last thing did
* make them pay attention to the white dot

Notes:

* adding a feature that checks for how close the nearest 3 food are on offense didn’t help the decision making at all, possibly even made it worse.
  + Idea was that it would prioritize a move that is closer to multiple food than 1 food so that it could spend less time, but in practice it didn’t do much.
  + I suspect issue was that it was throwing the weight scale out of whack. Initially it was a very significant skew because I did it based on each foods distance which made it start to care more about being between food than being on food or running away. After heavy tweaking I got it down to being less impactful than where the closest food is, but it still was still causing it was making it harder for me to prevent the pacman from entering into dead end paths that contain food while running away
  + could possibly have been better implemented using a genetic algorithm to determine weights, but this would have taken way too long as it already takes 15 or more minutes to run a single test (individual) 30 times against a single opponent.
* Removed the ability to stop in place on offense because it generally isn’t very useful
* removed the check for if pacman or ghost on offensive agent because it didn’t work
* added feature that weighs staying near the closest chokepoint on the default map highly so that the defensive agent hovers in the area. This combined with the 2 removal fixes already improved the capability of the AI from being very bad to being better than anything else, particularly on the red team. On the red team, the AI won 93% and tied 7% against the improvedteam agents, and it scored a median of +6 points and an average of 7.33 points. Against the baselineTeam agents, our AI still never and had the same median and average scores, but it had 2 more ties than against improvedTeams. Against itself the red side lost 3 games and tied 2, and it only scored an average of 3 points and a median of 1 point per game. This was a lot more even, but there seems to be a slight favoritism towards the red side for some reason. I believe this is because the position of the POIs might be slightly off on the blue side compared to the red.