Lucas Teltow LKT190000
Thiago Ciriaco TAC200002
Artificial Intelligence
Professor Ng
05/11/2023

Pacman Project Report

This report will cover our solution to the Pacman Capture the Flag game that our class is competing in. The game is similar to capture the flag, with each team's agents being able to kill the enemy agents only while defending, i.e. being on their own side of the map and the goal being to capture as many pellets from the enemies side as possible while stopping the enemy from taking pellets from your side. Each team was responsible for coding the intelligence powering their own Pacmen, and we were all provided the same base code used to generate visuals and allow us to compete. Each team has two agents, which can each be coded separately. These agents are allowed to do any role, and there are no restrictions on their positioning. Our strategy for winning revolves around three main pillars: our intelligent defense agent, our agents dynamically swapping between offense and defense, and our attack agent's greedy pathing algorithm.

Some important topics to understand that relate to our solution are reflex agents and greedy search. Each Pacman is controlled by a reflex agent, which analyzes the state of the game and performs an action based on it. In the context of this game, those actions are "North", "South", "East", "West", and "Stop". These reflex agents can use a number of complex strategies, but our approach is actually quite simple, and employs the Greedy algorithm methodology. Greedy algorithms look at the current situation, make a greedy choice (best local choice, without looking ahead), and repeat. The main advantage of greedy search is its speed and simplicity. In this challenge, we are limited to one second of computation per move to avoid disqualification, so speed is of utmost importance. The disadvantage though is that sometimes our agent will make a choice that leads to more danger down the line. However, as we will discuss in the next section, there are plenty of safeguards that make our greedy method safer than a simple score maximizing algorithm.

First of all, our OffenseAgent and DefenseAgent are really just one agent that makes use of the game state to determine whether they should be in offense or defense mode. At the start of the game we default to having one of our agents defending and one of them attacking, but as the game proceeds our strategy shifts. When we are winning by a lot, our attack agent returns to help defend, to hold our advantage. When we are losing by a lot, our defense agent attacks, giving us an increased offensive to regain the lead. Additionally, both agents toggle to attack when a power pellet is eaten, since they cannot defend anyways. Finally, in the end game, if the score is tied and the game is about to end, our defense agent will swap to offense to help us get that one final pellet to win.

Our offensive agent uses an algorithm to determine the safety rating of each pellet, based on how close it is to the pacman, to our side of the map, to any enemies, and to any power pellets. Using those values, it makes a greedy choice and sets its goal point to the safest pellet. This causes us to eat pellets that are easy to score, while also ensuring that we stay away from any defenders. Our agent also changes its behavior when it eats a power pellet, allowing it to eat more pellets due to no longer needing to worry about the defenders. When we have multiple

attackers, one attacker will prioritize pellets on the top of the map, while the other will seek lower ones, which ensures that we don't have them getting in each other's way or getting taken out at the same time. On another note, the offense agent favors returning home more and more as it collects pellets. We added a pellet threshold that serves as a maximum number of pellets we can be okay with holding before coming home. This makes sure that we score points quickly to establish an early lead, as well as keeps us from getting too excited and losing a lot of pellets.

As for our defense agent, it focuses on waiting on the centerline near the middle of the map to quickly intercept any enemy agents coming over from the other side. When an enemy attacks, our defense agent follows them and tries to eat them. In deciding where to wait, it tends toward the center of the field vertically, and it tries to defend power pellets in particular due to the massive impact the power pellets can have. When there are multiple defense agents, one of them will maintain the default defense, while the other waits for the enemies to pick up pellets before trying to stop it. This creates a pincer that is very effective at catching enemy agents, something that is often not expected with a greedy strategy.

All of these are accomplished by analyzing the game state and setting a goal point to path towards each turn. Then, we greedily path towards the chosen goal point. By being smart about the goal points we chose, we can gain the benefits of a greedy strategy while avoiding most of its downsides.

Throughout our development we found several unexpected bugs, mostly based around infinite loops caused by our agents swapping back and forth between offense and defense over and over. This bug was caused by overlapping conditions for switching to offense or defense. This was addressed by making sure that the conditions for switching to offensive or defensive mode are mutually exclusive. Another observation we found was that there are different overall strategies that can do better on different maps. For example, on very winding maps, having a solid defense guaranteed at least a tie. Finally, we noticed that teams that had the lowest number of wasted moves, steps like going back and forth or taking inefficient paths, tended to win more often.

Our agents adapted based on the game state, sort of like how a sports team can adapt when they are losing a game. This is our main advantage over other teams. However, a better performance could be achieved by combining our strategy with a similar adaptation based on the map, changing the various coefficients in our pathing algorithm based on how the pellets and walls are arranged. Another feature that might increase our performance would be to use a strategy that focuses on grabbing power pellets early and getting a quick lead before hunkering down and running a super defensive setup. Other non-greedy strategies may also do better than ours.

In conclusion, our offense and defense agents use greedy pathing to help them quickly determine the best routes, and this method of quickly generating great paths allowed us to devote time to calculating whether we needed more or less offense/defense for a given situation. Our agents also adapted to the clock, power pellets, and the map itself. We learned a lot from this project and had a lot of fun figuring out how to improve our agents.