CMPS 140

Froylan Valencia

Note updated code submitted along with pdf.

1) the fundamental problems you are trying to solve:

The problem we intend to solve is Capture the flag using Pacman as an agent. The game board is equally split amongst teams where there is red and blue teams. Each teams goal is to defend their pellets from being eaten while also attempting to capture the opposing teams pellets. Both sides begin with 2 pac men which on their side of the field will act as gate keepers i.e. ghosts, when crossing into the opposing teams field they become susceptible to being eaten by the opposing ghosts as they will now act as pac man. When eaten, the Pac men restart at the farthest corners of the map.

2) how you modeled these problems and your representations of the problems:

the problem is modeled as a 2 world grid with half the grid belonging to each team, containing two opposing agents we had a couple things that we had to account for in order to satisfy our problem, one is that we only have only a partly observable board so we must make a decision based on what we can observe, so the key here is to try to minimize uncertainty so while there where better approaches for this problem specifically I realized that in my attempts an agent that acted offensively until otherwise knowing of an enemies presence worked out the majority of the time 19/20 and 20/20 on average against the baseline agent. We also need to manage the offensive agent since his intention is to score but due to the distance of traveling across the board it would be preferable if he was to not get consumed by an opposing ghost.

I combined two approaches such as having a utility based agent that makes decisions based on the max expected utility from taking an action, however since I wanted to maximize offense without sacrificing easily reachable invaders, I also combined my agent with a part of a Model-Based Reflex Agent such that if in the case there was an intruder to cross over the path while we are close enough to notice the transition we would then act defensively for the time being. This allows us to not pass on defending the opposing pacmen when relatively close, and our other agent to focus on the main goal of consuming all pellets on our opponents side.

3) the computational strategy used to solve each problem:

Based on A(s), S(s,a), and U(s,a) we come up up with a an optimal choice based on our expected utility.

$A(s)$

A(s) function to generate best action based on utility;

$S(s,a)$

S(s,a) function that generates successor state when action taken a in state s

$U(s,a)$

U(s,a) is the utility of performing an action a in state s.

4) algorithmic choices you made in your implementation any obstacles you encountered while solving the problem:

There were two main issues that came up in my implementation:

Meeting in middle problem was another in which at the edge of the border line between the two sides, the attacker and defenders would get stuck preventing each other from going to the other side as each side's strategy was to not get eaten. Had to generate a way such that the agent would not end in a stale mate, this was done by eliminating Direction.Stop from our choices as well as adding a weight and feature such that it would move away from oppoising agents unless they are scared or they cross the bounds towards our side.

Another issue was dead ends adding functionality such that the agent would not be enclosed and lead to death however the dead end watch actually prevented the agent from proceeding to collect nearby pellets when in dead ends so it was a better choice for my implementation to forgo this and allow our agent to consume nearest food.