Taylon Holmes
Nomaan Khan

Final Report

Introduction

The Pacman CTF game is a contest between two teams to see which one can gather more of the other's pellets and then "Score" them by crossing back into their own half of the map. Each team is composed of 2 agents, which can then be used in any way the team sees fit. In our case we decided to maintain the simple Defensive/Offensive split with simple reflexive agents in order to make the project easier to split between 2 people as well as to allow the agent to run very fast and also not need to be trained a bunch.

Theoretical Foundation

Our solution is composed of two reflex agents, more specifically two simple reflex agents. An agent is simply something that makes decisions, and in this case those decisions are the direction that the pacman will move on its turn in order to gain the most benefit. The way an agent makes these decisions is by gathering data from its environment and then using that in some way to determine its desired course of action. In our case since we are using a simple reflex agent, all we care about is the current situation on the game board, which allows us to ignore previous moves by either our agent or by the enemy's as well as avoid needing to predict future movement of the enemy when making our decision. Since we only care about the current state of the board we determine the action that needs to be taken based upon a features and weights system that simply calculates a series of flags and distances and then multiplies those by various weights based upon the current situation the agent finds itself in. This means our agent is extremely adaptive to whatever strategy the enemy may be using at the time and, in theory, will help to get us more short-term gain for whatever enemy we face. Unfortunately, there are limitations to any implementation, and in the case of a simple reflex agent there are a few bigger problems. The inability for the agent to react to something that isn't explicitly added to its feature list is a big problem, and this means any part of the current game that that we didn't think to add is information that could be benefiting the pacman but isn't. Also, while not as big of a deal in pacman CTF, a simple reflex agent usually cannot be generated and stored for use in a decision tree for example since it would simply be too large to deal with. Lastly, since the bot relies on only information from the current state there is a lot of recalculation that it must do, some of which could be wasted computation time if that part of the state had not changed.

Taylon Holmes
Nomaan Khan

Final Agent Description

As stated above, our team is created using two simple reflex agents, one for offense, and one for defense. We decided to take this approach because not only did it create an easier divide in the work required for the project, but also because we feel that providing each agent with a specific job causes there to be less potential for problems caused by ties in their decision making.

- **Offensive Agent**

  The offensive agent contains the following features:
  - Successor score – used to tell the pacman to eat pellets near it.
  - Distance to food – used to track the distance to the closest pellet.
  - Stop(flag) – if the action involves pacman stopping his movement
  - Reverse(flag) – if the action involves going backwards
  - Enemy close(flag) – used to determine if an enemy is right next to pacman
  - Threat distance – used to measure distance to the nearest threat
  - Distance to capsule – used to get the distance to the nearest power pellet
  - Distance to score – used to find the distance to the nearest place to turn in pellets

  The weights for distance to score and enemy close are based on functions that take into account factors like how many pellets are being carried and if a power pellet has been recently eaten. All other values are hard coded and were tuned by hand.

  The agent takes all these factors and weights, multiplies the appropriate ones together and then the sum of all that will give each possible decision a score, the best score is the direction taken with ties being broken randomly.

- **Defensive Agent**

  The defensive agent contains the following features:
  - Number of Invaders – number of opponent's pacmen in home side
  - On Defense – used to determine whether agent is defense
  - Invader Distance – distance to opponent's pacmen
  - Stop – if the action involves pacman stopping his movement
  - Reverse – if the action involves going backwards

  The agent takes all these factors and weights, multiplies the appropriate ones together and then the sum of all that will give each possible decision a score, the best score is the direction taken with ties being broken randomly.

Taylon Holmes
Nomaan Khan

Observations

We tried many strategies including an offense only strategy where both agents would be on the offense. That strategy did not work well because it could defend against the pacmen and it also lost to the baseline team.

Finding the closest distance to the border was extremely useful in improving our offensive agent

Playing different iterations of our agents against each other was very helpful in gauging the improvement of latter iterations.

Depending on the circumstances sometimes our agents could not avoid dead ends because the offensive agent wasn't considering a path back home which does not involve getting eaten by a ghost.

Conclusion and Recommendations

The Pacman CTF contest is a game in which two teams, each with two agents, try to get as many pellets in opponents half as they can in the allotted time. The agents in their home half are ghosts who can eat opponent pacmen and they are pacmen in the opponent's half.

We are using a relatively simple reflex agent which only considers the current state, so that we can maximize our short-term gain. But it cannot react to any new strategy that isn't accounted for in the features list of the agents.

Our agents take various factors and weights, multiplies the appropriate ones together and then adds them all to give each possible decision a score, then the best score is the direction taken with ties being broken randomly.

Our agent can be improved by adding dead end avoidance, which would be that our offensive agent would first map a path back home such that it would only eat pellets such that it does not get dead ended by a ghost.

Another improvement would be implementing a machine learning technique like reinforcement learning which would learn from every move, every opponent and every game.