AI Final implementation

By AJ and Kevin

We decided to make a reflex agent which would use some interesting strategies to ensure a win for our capture the flag project. Our original plan was to use minimax and expectimax agents for our team, however we ran into a lot of implementation problems when attempting those. Building off of the reflex agent provided helped us understand how the system worked and what we could do to pull a win. We stuck with one offensive and one defensive agent, as that seemed to be the best way to protect our pellets early while still maintaining the possibility of having an early lead. The strength of our agents rely on maintaining a small lead rather than accruing a larger lead over time.

Our offensive agent took more time to flesh out than the defensive agent. It operates by taking features that are extracted from the game data, such as the food list, the distance to food, distance to the enemy, and so on. This set of features were then weighted to inform the agent what action it should be taking. Tuning the weights was rather tricky when trying to get an implementation which balanced ghost avoidance and point maximization properly. One change we made was having a carry limit set for our pacman. When our pacman had two or more pellets on him we told him to ignore all other food and head directly back to his side of the map. This was an issue the baseline team had as well. The agent would pick up upwards of ten pellets but wouldn't return to their side of the map to redeem the points. We tried a bunch of different carrying limits. Carrying over two food pellets didn't allow enough time for our pacman to escape the other team's defensive ghost. Utilizing a carrying limit of two food pellets, the offensive pacman is normally able to cross into enemy territory and cross back before the other team could respond accordingly and eat our pacman. We think that given a better opponent it

would be increasingly difficult to get food pellets that are deeper into enemy territory which is why we utilize the strategy of maintaining a minor lead over attempting to increase the lead once it is established.

Our defensive agent is similar to the baseline agent with a few key modifications. It operates similarly to the offensive reflex agent because it also uses a set of features that are extracted from the game state data and weighted to inform the agent how to act. The difference being that the features and weights are different because its goals are different. The major problem with the baseline agent is the fact that they wander aimlessly unless there is someone within their range to attack. We added a new feature called dist_to_def to combat this issue, which has a relatively low weight. This way when the agent has no other features pulling it away from its spot, it constantly circles an area right in the center of the map. This “patrol” it does is excellent for catching anyone trying to cross the border, and also excellent if someone managed to make it past. The agent is good at cutting off exits as well, allowing it to chase opponent pacman agents easier, reducing the chance of the enemy capturing food and bringing the food back to their side of the map.

One thing we realized after much testing was the importance of maintaining a lead. We included an extra conditional statement for our offensive agent which would send him into defensive mode after we had a decent lead. Once we move our offensive agent into a secondary defense spot we were able to move our defensive agent to the south, leading to some complete coverage of the border. This means that there is only one way for the opposing team to safely cross the board and that is by using the path to the north, but because the agent is in the middle of the map it should in most scenarios notice that the enemy has crossed the border. This will trigger the agent to start chasing the enemy pacman and should theoretically

guarantee that the pacman will not be able to bring food back to their side of the board even if they capture some food.

While our reflex agent didn't have the more complex algorithms we originally planned, it did have some interesting strategies. We solidly beat the baseline team every game. Spending a lot of time tuning our weights was important because with previous iterations of our weights we would see our agents perform some erratic behaviour.