Artificial Intelligence - Othello Intelligence Improvement Log

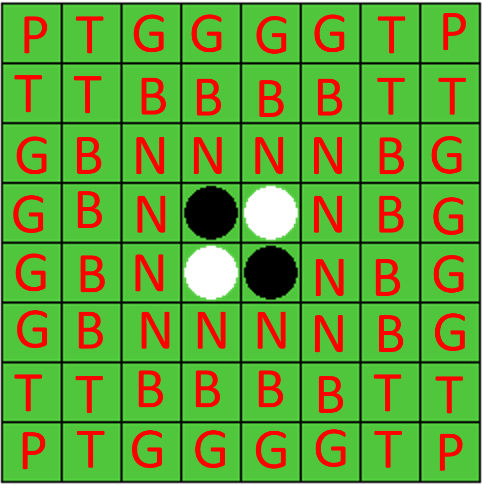
Ryan Helmlinger

Period 3

­January 8th, 2016:

I first made my computer so that it randomly picked from the possibilities list to pick an arbitrary spot to go for its turn. My first improvement was to make the computer pick the spot that would flip the most tokens over. I implemented it by returning it with my findPossibilities method in which it returned a set of tuples with the first value being the index and the second value being how many it would flip. This method won about 65% of the time.

January 15th, 2016

 My second improvement was making a 5 lists of spots where to play. The lists were ranked according to which spot should be played in first. If any of the possible indexes were in the first list, then it would chose the spot that would flip the most tokens from those indexes. Otherwise, it would keep going down the ranked lists until it found an index. The ranked lists are displayed in the picture below. The P’s, the corners, are the in the first list, followed by the G’s, then the N’s, then the B’s, and finally the T’s are in the last list.

However, my program still has some limitations. It doesn’t look ahead at what moves the player could make next. Also, labling lists makes it certain the computer doesn’t play in certain spots, but a number system might work better as it would allow more flexibilty. After these improvements, my computer wins about 80-85% of the time.

January 16th, 2016

My third improvement was to essentially combine both previous improvements. I made a dictionary that has higher values for the indexes that are worth the most. For example, corners are ranked the highest and the spots next to the corners are ranked the lowest. This is similar to the chess example of weighting pieces, which we talked about in class. I also made a function that takes the board and determines who is winning and by how much. My program currently tests each of the possibilites and determines which possibility gives the current player the biggest lead. This strategy wins about 80% of the time.

January 22nd, 2016

I added a helper method for the previous improvement I made. This method allows my program to look at the moves recursively. However, recuring all the way down takes too long for the program to run, so I have an if-statement in the method that stops recuring at a certain depth. At only a depth of 1, my program works the same as the previous method. But if I increse the depth, for example at a depth of 2, the program increses its record from about and averagae of 82 wins to about an average of 87 wins. I assume if I keep recuring further, the win percentage will go up, but it will take a lot longer. My next improvements need to be to make my program run faster.

