Michael Edelman

Professor Rosenfeld

Artificial Intelligence

COM 3760

Heuristic Description

My Agent was able to play agent the random agent and beat him 99.9%(999/1000 games) of the time. The heuristics used were some baseline ideas with added creativity when I would observe initially why my agent would lose sometimes. This is how the agent works: Whenever the Agent is playing a game and it is his turn he will find all the columns that a chip can be placed in using the

```
get_valid_locations(board)
```

function. Then for each possible slot the agent will evaluate putting our chip into that position with the

```
evalMove(board, color, row, col)
```

function. This function gathers heuristic data from four other functions as shown below:

```
 value += checkDown(board,color, row, col)
    value += checkHorizontal(board,color, row, col)
    value += checkDiagonal1(board,color, row, col)
    value += checkDiagonal2(board,color, row, col)
```

These check functions gather the heuristic data examining each direction. The heuristic evaluation can be described with this chart:

```
#####################Heuristic Value Chart#####################
# How many in a row*#    Same Color   #    Opposite Color   #
###############################################################
#        0          #       0         #         0           #
#        1          #       2         #         1           #
#        2          #       8         #         6           #
#        3          #      150        #        50           #
###############################################################
```

When we consider a position, we look at the chips adjacent to it to determine whether this would be a good offensive move (3, 4 in a row) or a good defensive move (block 3 in a row). The chart above says if there is 1 chip of the same color next to the position we are evaluating we give it a value of 2. If it's the opposite color, we give it a value of 1 (We therefore prioritize offense over defense). If there are 2 in a row next to this position, we will give it a value of 8 for the same color as our own chip and 6 for a value of the opposite chip. Again, the idea here is that the more in a row of either chip the better this position

is – either to get closer to winning or to block a potential win for the opponent. The reason the values jump from 2 to one in a row to 8 for two in a row is that we want to prioritize getting stuff in a longer row and not just put a bunch of chips close to each other. Same for the opposite color (but in that case blocking). For 3 in a row the value for our own color jumps to 150 – this is because if we have this situation, we win the game so there should be no sum of other heuristics that trump this one. 150 is way out of reach of other heuristics adding up to it. A similar Idea exists with blocking the opposite color when it has 3 in a row – it is impossible to have a position with a higher heuristic value than 50 unless it would result in us winning the game (150). This is the reason for this evaluation system. The heuristic algorithm takes other things into account as well. For one it actually defends against double trapping, because it recognizes when there are two empty spaces on either side of two (or more) consecutive chips and it gives it a defense value as if it were a 3 in a row situation. This is defending against a go to double trap in connect four and has a good level of creativity to it. Another thing the heuristic algorithm is creative about is it does not just consider each side individually but rather actually considers sides together. So, for example if on either side of a position are two of our color – then we could win if we put our chip there – but if we consider the sides individually, they would only generate heuristic values of 8 + 8. Our heuristic algorithm looks at the bigger picture and thinks about what would happen if we combined either side (whether horizontally or diagonally) for this position. The code has helpful comments if one were to look inside.