CONNECT-K FINAL REPORT

Partner Names and IDs:

Dennis Tran #57725022

Andrew Kim #60499432

Team Name: Best Connect K- NA

1)

The heuristic function or the Eval function, was made of many different heuristics combined to find the best move at any given game state; this was done without a dot product of the weights and features. The Eval function was made from four different heuristics: a Row heuristic, a column heuristic, and two diagonal heuristics that find both diagonals. What each of these does can largely be guessed from the names they are assigned; each of these check the heuristic value or how good a state it is for the AI for a condition in that direction. The diagonals were split for simplicity as accounting for both in one loop proved to be problematic. How each of the heuristics works is largely the same.

What each one does is go through the rows, columns or diagonals and finds and counts any unblocked paths; it then adds them all up to get the total heuristic value which is done with K^N where K is the length preset by the board and n is the number of valid win states of either the row, column or diagonal based on what heuristic value is being calculated. A valid win state is one that the opposing agent, whether a human or AI, is not blocked by a piece from the opposing force.

A power function was chosen as made board states with a lot of possible conditions quickly become more desired than a linear relation; the AI would at times attempt to go for more apparent wins due to the horizon effect placing way too much emphasis on a weak agent. This seemed to be more like how the Poor AI thought where it could not add complexity to its moves and instead went for just making quick wins that were easily counter- able.

2)

The Alpha- Beta pruning was done with multiple switches in that it would stop working after a time limit. This was extremely helpful as there are a lot of branches that could not be checked with only the 5 seconds allocated per turn. While one could search at very deep depths in the tree, often the case was that the tree was so large that finding the most optimal move on large game boards was not feasible in the 5 seconds allocated.

In order to make each calculation valuable, it made sense to use pruning as it got rid of bad moves that would only expand the tree to an eventual loss at worst or a slower, and much more counter- able move at best. Some moves that a human would clearly not make as it put them in a terrible position would be considered and looked into by an AI if the pruning did not exist. There are many of these moves and as a consequence, there are many branches that could be immediately pruned before they were explored when there is already a better move found previously.

3)

The IDS was implemented so that it was done before any search of a lower depth and was done to help find good states quicker; while the game tree could have been traversed every node,

IDS would usually find the shorter of paths much faster which proved to be invaluable given the time constraints. How much “smarter” the AI got from implementing the IDS, or rather how much deeper the AI could search, was pretty impressive. One problem early on in the development of the AI was the fact that the depth of the tree was not that great. While the Alpha- beta pruning did help get rid of many terrible moves, the IDS helped find good moves and thus achieved more of the inverse of the pruning.

While there are distinguishably bad or terrible options, the same can be said about the good moves; there are also many states that are guaranteed to be decent. With the IDS, these will be checked before any “worse” moves and will likely also make the pruning a bit quicker and useful.

4)

The values associated with each node in the game tree at the previous IDS depth limit was not used in this project. After much debate, it seemed like a pricey operation for what seemed to not be of much benefit. This was not important when the turn count was low; however as the amount of turns taken increased linearly, saving values did seem useful when searching the tree again. Saving the values seemed to actually make the AI slower when it went first.

However the AI at the current state does not seem to have very good second turn moves with gravity on as is does not seem able to search as deep as in the other cases. While this is almost a non- factor on a more powerful computer, this could be a problem in a tournament environment where there is not as much computing power available to one java program.

5)

The quiescence test was a very simple test that was used when there was time left; while there may have been something that objectively looked good based on the collected heuristics value, there is something known as the Horizon effect. While a move may look good on paper, it could lead to a loss much deeper down the tree that was not explored due to the amount of heuristics that must be calculated as a consequence of such a large game tree.

What the quiescence did was a simple and quick check one move ahead to make sure that the AI missed a tragically obvious mistake it would have made otherwise from not thinking too far ahead. The quiescence test did not seem to help too much against the “dumber” AIs who lacked any strategy; however against much smarter AIs this helped greatly as smarter AIs seemed to think more ahead than the AI without a quiescence test. In fact when the AI was playing against itself, it seemed to be unable to defend itself against its own tactics without this test. This help make the AI much better for when it went second.

6)

The alternating of who gets the first turn seems to be the only way to give a fair game to both players; placing two marks does not seem to be good for the AI and would probably mess up some people’s trees. One thing that could be tried is to make the it so that the second player needs less connections by 1; this would mean that while the first player would not have as big of a lead but this may in fact give the second player way too much of an advantage.