EARIN Lab 2 Variant 4 Report

Dawid Wypych

Aayush Gupta

Implementation of Solution

In this lab, an AI for the game Gomoku was developed (as well as the game itself) using the min max algorithm with alpha beta pruning. The game is played on a 15 by 15 tile board by two players. Players take turns placing one stone at a time anywhere on the board that isn't already occupied, and the first player to make a chain of five with only their stones either horizontally, vertically, or diagonally wins the game. Unlike some other similar games, only a chain that is exactly five stones long wins the game. Chains that are six stones or longer do not count.

Ideally, an AI for such a game would be made by looking at all the possible moves that could be made until the end of the game and using the victory condition to score each move. Then, by working backwards, the best possible move could be determined. Unfortunately, Gomoku has upwards of 225! possible board states. So, strictly using victory conditions was not very feasible.

Instead, it was assumed that each piece on the board had the potential to result in a victory and that the more "potentials" a player had, the more likely they were to win. Reducing an opponent's "potentials" could be equally beneficial. Using this concept, a point system was developed by finding every possible victory state for each stone already on the board and assigning points based on the number of stones already on the board and inside of those victory states. A series of modifiers were chosen by hand so that creating longer chains of stones would be seen as more favorable than placing individual stones in unblocked regions, far from other stones. Applying this method allowed the AI to "see" the board so that it could know where to focus its attention.

While the alpha beta pruning method was an improvement over the vanilla min max algorithm, it still proved sluggish, even when the depth was set to two. As a result, an improvement was made to make it more efficient. Alpha beta pruning benefits greatly from having the first branches tested being the most optimal choices. Traditionally, traversing a 2D array from start to finish will have you start in the corner (at array[0][0]) and travel along the edge until you get to the other corner. Then you do the same one row down and repeat until you get to the end. In Gomoku, these are considered the worst possible moves, as the edges automatically block your pieces and give you fewer opportunities to make chains. The best region to place pieces is in and near the middle. Using this knowledge, a new algorithm was developed that traversed the array from the middle going outward in rings.

Analysis of Results

The first iteration of the algorithm was sluggish. It usually took around 20 to 40 seconds for the AI to make a move, but at times it took over a minute. The pruning wasn’t very effective and often failed to reach even 50 percent. While it made good moves, it was quite irritating to test, with the nonstop waiting.

The second iteration was vastly superior and cut down the waiting time significantly. In most cases, the AI it took between 3 and 10 seconds to make its move. Very occasionally it would reach 20 or even 30 seconds, but it was rather uncommon. It usually pruned at least 90% of branches and quite often got as high as 98%. Overall, it was a much more consistent (and less frustrating) experience testing it. Almost all of the tests that were performed on the effectiveness of the AI were done on the second iteration.

In terms of the skill of the AI, it troubles me to say that I have yet to beat it. While occasionally it does make some questionable offensive moves, it has yet to fail when it comes to defending itself. Overall, it is very impressive, especially since it can only predict two moves ahead.

Reflections

Overall, the algorithm was applied quite effectively. While the first iteration had its issues, they were able to be resolved with a bit of strategic thinking. The criteria used for assigning point values proved to be very functional and certainly exceeded all expectations that were had. Overall, it was a very interesting lab, and trying to come up with better ways to make the AI faster and more intelligent was quite an enjoyable experience.

There are definitely some improvements that could be made. The modifiers were chosen by hand, based on some experimentation. However, there was no real proper method to it other than choosing what felt right. Ideally, the values would be set and tweaked by performing automated testing with machine learning. Finding the values associated with the highest win rate would likely fix the occasional odd behavior of the AI.