Luke Johnson, Chandler Capps

Saturday, December 3rd 2016

Mancala: Heuristic Research

We started off this research being given the aima-python repository of which it gave us the ability to build our Mancala program and run it with Alpha-Beta functions. The choice of of what game to make was quite difficult. We had a couple of different “simple” games lined up to review whether or not we could do it, but Mancala was the best fitting for us overall. It had simple moves, simple scoring techniques, and already had a good bit of research on it that we could either springboard ideas off of or we could simply take what other people have written as a heuristic value. Having all of this information made the idea of creating this program very feasible.

The first goal of the project was to be able to create, with the tools that we have already been given, a program that could intellectually play the game of Mancala. The second goal, or the main goal, was to find and/or create strong heuristics that would help the program make smart decisions and chose the best choice of possible states. We have a variety of sources that we got information from. These sources range from scholarly paper to just people that have spent quite a bit of time studying not only the heuristics for this game but also learning the best possible moves and the most effective way to play the game with winning in mind. All of the heuristics that we have did not come verbatim from any author. Rather all the heuristics were written by us just for this project, but most of them were inspired by the information that was given to us. For example, heuristic 1 and 2 came directly from what was said by the author, and heuristic 4 came directly from information of another scholarly paper. Heuristic 3 came from just inspiration from the other heuristics, this may also be why it is not quite as good as the others due to no information on it prior, just our own knowledge.

Heuristic one came from a paper by C. Gifford [2]. This heuristic focuses on looking at both your personal mancala and your opponents. After it would gather that information it would then “score” the two numbers together, giving a value for the utility function to pass to Alpha-Beta so that it can make an educated decision. He did not supply the code, rather he supplied the idea, information, research, and math behind the heuristic. We used this one due to it being rather simple but very effective at all point during the game. This one also had some of the most research done on it and was validated to be quite credible. He ran multiple test pitting other heuristics against each other and heuristic 1 came out on top majority of the time. Due to it having a high percentage win rate and very reliable sources of research, this was one of our best heuristics.

Heuristic 2 and 3 were sort of like the cousin that you don’t talk about. It’s not that they did not work, it is just that they did not work as well as their competitors. Heuristic 2 came from a blog, hackerrank.com [4], where they had given information on the research and studies behind this heuristic. This one focuses on ALL of the pieces that each person has on his side, it then gives values to moves that will keep your number of pieces high on your side, but low on the opponents, thus putting him in a position to make a bad move, or lose all his pieces and allowing us to keep all of our pieces and winning, hopefully. It had a lot of potential to be a valuable heuristic, though once we implemented it into our program, it did not work as well as they had made it out to be, or as other had made it out to be. We think that there was more to that heuristic that we just could not understand or that they did not supply other important information that would make it work better. Our implementation could have also been flawed, but we do not think that it could have been bad enough to not work as well as heuristic 1. Heuristic 3 on the other hand came from somewhere deep inside Luke’s brain. This one prioritizes empty pots and capturing your opponent’s pieces, this way he had fewer moves and you could end the game faster and giving you a similar ending outcome as heuristic 2. He felt that he could write something similar, if not better than the heuristics than we had prior. Though, after implementation it did not work anywhere near as expected. This may be due to human error, or just that we did not have it written correctly to be able to do what we wanted it to.

Heuristic 4, much like heuristic 1, was a very good heuristic. It did everything that we expected and more. This heuristic would highly prioritize capture moves. So it will look as far ahead as it can and make moves that will either get it a capture, keep the opponent from capturing, or make moves that will cause the opponent to be forced into a bad move that can open up a capture play. This is roughly just a much better version of heuristic 3 that we tried to create. This came from a scholarly paper written by C. Divilly [3]. In his paper he refers to it as H6. His explanation and research behind it are quite extensive, though the research is not so clearly devoted to each heuristic rather all of them equally.

From researching all of this, our conclusions were as follows: 1) H1 is slightly better than H4 all around, 2) H4 gets better and better the further you let it go down the Alpha-Beta tree, as do most heuristics but this one yields the best results from it, 3) We also found that a majority of the time, the person that gets to go first has a slight advantage as they can set up the game early how they would like, especially for a robot that can think much further ahead than a human. Looking back at everything that came from this project it really pushed the limits for both of us both in our coding abilities, but also cognitive as we had to be able to adapt to a new language, IDE, and paradigm of thinking. All of these things resulted in a program that we believe is acceptably smart at what it is meant to do.

Annotated Bibliography

[1] S. Russell and P. Norvig, Artificial intelligence, 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 2010, pp. 161-171.

This was our textbook for our Artificial Intelligence course. In the pages listed, the book describes Alpha-Beta pruning.

[2] C. Gifford et al., “Searching and Game Playing: An Artificial Intelligence Approach to Mancala” The University of Kansas, KS: July 2008, Available: https://fiasco.ittc.ku.edu/publications/documents/Gifford\_ITTC-FY2009-TR-03050-03.pdf

This paper analyzes seven different heuristic functions. It was determined that H1: (My Mancala – Opponent’s Mancala) was the most effective heuristic that they analyzed. For this reason, we chose this heuristic to compare with other heuristics.

[3] C. Divilly et al., “Exploration and Analysis of the Evolution of Strategies for Mancala Variants”, 2013, Available: http://eldar.mathstat.uoguelph.ca/dashlock/CIG2013/papers/paper\_45.pdf

This paper analyzes six different heuristic functions. The sixth heuristic function, which was our inspiration for our fourth and third heuristics, aimed to estimate the most stones that could be scored by the opponent after stones have been scored by the player.

[4] H. Karunanidhi, “Mancala”, February 2013, Available:http://blog.hackerrank.com/mancala/

This blog compiles the different methods that programmers used against Mancala bots. The first place winner used Alpha-Beta pruning along with a version of our second heuristic function.