Noé Loyola
A01202681
Artificial Intelligence
“AI Challenge: Connect 4 AI”

The developed project is a Python implementation of a Connect 4 game, where human players can confront a computer controlled opponent. The interest of developing an AI for this particular game lies in that it is a very simple game to play, but still has many varied outcomes, thanks to its 7 X 6 sized board. This permitted me to focus on developing a useful algorithm that anticipated player moves, without bothering with difficult programming implementations that could arise when trying to mirror the rules of a much more complex game. However, the search space is still quite big, with $7 * 10^{13}$ possible configurations (Edelkamp, Kissman, 2011), making this an adequate challenge to develop a proper search algorithm.

The solution functions with the minimax algorithm as it's backbone. The algorithm creates a search tree of all the possible moves given a game state, with a predetermined depth. In the case of this implementation, the chosen depth was 4, because it allowed the computer to evaluate two of it's own moves and the response of the player to these. A bigger search depth resulted in the game running slowly, which isn't engaging to the user, and using a smaller depth resulted in the game being played with more short term consequences in mind. The algorithm then evaluates each level of the tree as turns of the game. If the given turn is that of the opponent, then the player will make the moves that are of his best interest. Therefore, the opponent maximizes his gain. To respond to this, the computer needs to minimize the possible damage caused by the opponent, therefore choosing the option that gives the opponent the least gain.

However, this algorithm does not provide ways of knowing which is a move that benefits a player and which is not. This is where the challenge comes into play. In order to determine good and bad moves, a heuristic must be developed according to the current problem being solved. In this case, I needed to develop a heuristic that determined if a certain state of the board was good or bad for the player.

My first approach tried to find wins and losses in the future states of the board. A win for the computer is the most desirable state, therefore it was labelled with positive infinity. Losses are the least desired state, so they are labeled with negative infinity. This approach works but is quite limited. It only finds if a board state is good or bad when wins and losses are present. But what about all other states? Knowing if a given board is desirable or not before a win or a loss is found is essential, as it allows the computer to generate more opportunities for victory, and to prevent situations where it could be defeated. With this in mind, the heuristic was improved to count all the possible win spaces for each player, by finding three in a rows that could become four in a rows and win the game. This allows the computer to know how many possibilities does it have to win given the current board, as well as how many to lose. The heuristic subtracts the possible wins for the opponent from the possible wins of the computer. This allows the computer to know that the boards with the highest value are the most beneficial.

Using the minimax algorithm, coupled with the heuristic, results in a very competitive AI that is difficult to beat. Personally, not being very talented in the Connect 4 game, I have not been able to defeat this computer opponent. The algorithm creates formidable opponent, and also runs in less than a second, preventing the player from getting bored or impatient.

How does this implementation compare to other solutions of problems regarding big search spaces? Connect 4 is a relatively simple game, with an approachable search space, which allows it to function with the developed algorithm. However, other games such as chess, have much more complex rules, were much more possible outcomes can be found in a single turn, as well as it has an enormous search space of 35^{100} possible states. (Russell, Norvig, 1995) Given this complexity, my solution would not be able to be efficient by any means. Minimax would evaluate all possible outcomes, making any computer run out of space very quickly.

To potentially solve this problem, pruning algorithms such as Alpha/Beta (Heineman, Pollice, Selkow, 2008) could be used, which prune branches from the tree and generate a much more approachable search space. Secondly, a lot of thought and time should be dedicated to develop a heuristic that simplifies the searching as much as possible. Even though my heuristic solves the problem, it probably isn't the best, primarily because the problem doesn't require the most optimal one to be solved, but also because I am a student and have limited to time to think of and develop an optimal heuristic.

Finding ways of traversing enormous search spaces is one of the main challenges of computer science and artificial intelligence. By developing this program, I believe I have introduced myself to this area of the field, allowing me to properly understand the main considerations that must be made when confronting these types of problems. I look forward to applying the knowledge I acquired to bigger challenges in the future.

References

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*(p. 38). Englewood Cliffs, N.J.: Prentice Hall.

Heineman, G., & Selkow, S. (2009). *Algorithms in a nutshell* (pp. 218-219). Sebastopol, California: O'Reilly.

Edelkamp, S., & Kissmann, P. (2011). On the Complexity of BDDs for State Space Search: A Case Study in Connect Four. Retrieved November 22, 2015, from https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/viewFile/3690/3827