AI Report

Elvric Trombert 260673394

April 2019

Introduction

The approach taken for the project was to use monte carlo along with simple "true" heuristic to get the best move possible. To keep up with the time limitation a flexible timer was included, timer that changes after every move to adapt to the increase or decrease in calculation speed between different board states. Finally depending on the turn number the depth at which all the possible moves are analyses increases.

Going back to the "true" heuristic, the intention was to use the simplest heuristic possible with as little human influence as possible. This led to my heuristic function to only identify board states that are for sure bad or good for me or the opponent such as 4 in a row for the opponent at its turn. I avoided writing heuristic functions that are solely base on the human perspective like making 3 in a row as a good state and so on as I believe that what can make an AI better than a human is for it to not think like one. Following the same thoughts this is why I chose to use monte carlo as it allows the AI to see for itself what might be best and put less importance of the heuristic function than alpha beta.

Going deeper into the algorithm the first step is the descent phase looking for the most interesting leaf once found, the children of that leaf are generated and trimmed by the heuristic to remove bad child or only keep the best child. Then the roll out and update phase are initiated on a random child. The process repeats until the think timer expires there the code finds the best move according to the monte carlo simulation and the think timer is modified based on the total time taken to retrieve the move.

Theoretical approach

The monte carlo approach used was very close to the one seen in class. First it undergoes the descent phase where while the node chosen has children, all of the children search scores are generated and the one with the largest search score becomes the next chosen node. Once a leaf node is reached (node with no children). The propagation phase is initiated, children of the leaf node are generated and the roll out phase is performed on one child at random. For the update phase if it is a win for me we increment the number of win by 1 if it is a draw by 0.5 otherwise 0.

Following what was thought in class, to be able to simulate as much game as possible and avoid a large branching factor simplifying the state of the game can be used. In this case this approach was taken by not considering every viable swap at certain depth.

To also allow for more meaningful monte carlo simulation when generating the children of a node these ones are trimmed through the heretic to only keep the most promising children and remove the bad ones.

The monte carlo search score was also change to better match alpha beta. In this case when reaching a node where it is the opponents turn, the win score was changed to be $numVisited - numWins$ as in theory the opponent should choose the path that benefits it the most which is not the case if it takes the child that has the greatest win score for me.

Advantages and Disadvantages

There are three advantages to the approach:
First since monte carlo is being used there is a random factor to the algorithm meaning that against the same opponent the same moves may not be made even if the board states are the same which leads to a more flexible algorithm, sometimes it may loose but if it did that does not mean that it would loose all the time against that same opponent. The second is the fact that as more and more moves are performed on the game the branching factor drastically decreases and since the monte carlo approach does not depend strongly on the heuristic it should perform better than alpha beta in certain cases as there heuristic may blow up and return distorted values. Finally the approach taken regarding the heuristic used remains very basic to decrease the influence of human perspective which leads to a monte carlo with more options to experiment with board states that may seem bad for us but could be seen as good in certain cases.

Disadvantages: The fact that I tried to make the heuristic as simple as possible is also one of the greatest weakness of my approach. In certain cases when testing with other people my monte carlo tended to want to avoid bad states by simply swapping quadrants rather than blocking which may be good in the short term as it allowed it to better position its own piece to get closer to a win but as the board gets more full such swaps become harder to always perform without any negative side effects which causes it to loose later in the game. Another weakness of the monte carlo lies during the early game phase, since the branching factor is so large the number of simulation it can make is fairly limited most of the time until the 7th move it had only done 20 to 30 simulation on the move it took which is not sufficient to deem ita as the absolute best move.

Other Approach

At the very begging I tried Min Max that quickly turned to alpha beta pruning. I was aiming to compare monte carlo vs my alpha beta pruning head to head but I quickly realized that it implied that I needed a hard heuristic for my alpha beta. The perspective of writing a hard heuristic based on my perspective of what is and is not a good state game went against the approach I originally intended. Thus I wrote a heuristic for the alpha beta that only look at absolute good and bad states meaning it identifies states that for sure will lead to a win or loss of the player in the future (4 in a row for opponent during his turn, winning move, loosing move, move that lead to 4 in a row with empty slots on both ends which is a guaranteed win unless the opponent can win next turn). As this was not enough for the alpha beta to be effective I dropped this approach to focus on monte carlo.

Improvement

One big improvement that can be made is to refine the heuristic function, towards the end of the project it appeared to me that there may exists other aspects of the board state that could lead to an unavoidable win or loss in the future, but these stages where quite hard to identify thus I did not have the time to implement them.

An other improvement is to make the heuristic harder in terms of identifying states that viewed by humans can be considered bad or good states like 3 in a row. Although this goes against the approach I took for this specific simple game it may actually be able to increase the performance of my monte carlo especially at the beginning where the branching factor is very large.