Mathieu Vachon

260745604

Comp 424 – Artificial Intelligence

Project write-up

1. Explanation of how the program works, and a motivation for the approach

My program uses a mixture of my own strategy and alpha-beta pruning to pick its next move. I will start by explaining my own strategy which I use for my first 4 moves. It is defensive based, and the approach is different depending if I am the White player or the Black player. If I am the White, my first two moves are in the middle of 2 different quadrants; the next two moves I block the third spot if the opponent put 2 pieces in a row, column or diagonal in the same quadrant. The reason I do that is that I found that one of the worst states is when your opponent has 3 consecutive pieces in a quadrant and the corresponding row/column/diagonal isn't blocked by your pieces, so I want to avoid that. Another very bad state is when the opponent has 4 pieces in 2 matching rows/columns/diagonals and you only have 0 or 1 pieces. Taking the example of a row, a very bad state would look something like this (I am White player in this example, E == Empty):

Examples of very bad states: B B B | E E E or W B B | B B E

However, I also do check if I have a piece in one of the middle spots of the 2 quadrants. If that is the case, the wins using the row, column, or diagonal passing through that middle piece will not work, so that consecutive 5 is dead. Finally, I also check the tricky diagonals to make sure the opponent is not trying to win using one of the diagonals that span 3 quadrants. If there isn't a move

to defend, I do an attacking move where I create a consecutive 2 in a quadrant or even a consecutive 3 if that is possible.

For the black player, the strategy is almost the same, the only difference is that for the second I do not put it in a random middle spot of a quadrant if the White player placed a consecutive 2, where I block it the same way as I described above. If that is not the case, I will put it in a random middle spot of a quadrant. After the first four moves, I use alpha beta pruning algorithm with depth 2, until my thirteenth move where I increase to depth 3. However, before calling my alpha beta, I do check first if any legal move can make me win (which I will take). If I can't win and if I have a very bad state as described above, I block it right away, which I will explain in following sections. Finally, my alpha beta, if it gets there, uses my heuristic. My heuristic gives very high points for states where you are really close to winning, and low to moderate points for states where you have a slight advantage. My heuristic checks every combination of rows (3 x 6 possible swaps), columns (3 x 6), and diagonals (2 x 6). It will also add a lot of points for a strong defensive block.

When it comes to the motivation of my approach, I have my strategy for my first four moves where I just check the current state because the branching factor is too high initially. My first four moves are defensive based so that I stay in the game and give myself a chance until my alpha beta takes over. Finally, my reasoning for using alpha beta pruning instead of Monte Carlo is that I feel that I really understood well the game states and what constitutes a bad one and a good one. That advantage manifests itself best in Alpha Beta which is heuristic based. The heuristic does not perform that well for defense though which is why I hard code defensive moves when facing a very bad state. I only increase my depth at level 13, since I do not want to risk a timeout, as I do not prune that many nodes off.

2. Theoretical basis of the approach

I used the alpha beta pruning algorithm, which we learned in class. The pseudocode I used as reference is from Wikipedia : https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning. The alpha beta algorithm runs a bit like Minimax with the tree structure and alternating turns, but it decreases the number of nodes to check by pruning branches. The part you need implement and think about is your heuristic which helps quantify and separate a good State generated from your Move and a bad State.

3. Advantages and disadvantages of the approach

First of all, an advantage with my approach is that it analyzes all the possible game States possible for my First Move and my Opponent's first Move, so I don't skip over a possible move (unless I can win of course, or if the state is very bad and I need to block). Another advantage is that with my first four moves, I sometimes beat the other player in 5 moves (happens almost always against random Player), and I never get beaten right away so I always have a chance. The last advantage is that my alpha beta heuristic easily sees good attacking moves, so my attack is very strong.

Now, when it comes to disadvantages, my alpha beta has very low depth. In fact, for most games it will only see depth 2, unless it gets to move #13. That means that I can't go very deep and I will miss plenty of good states or go into bad states that I would have seen had I been able to expand my tree deeper. Another disadvantage is that my defense is weak after the first 4 moves. Indeed, my heuristic always seems to favor offensive Moves so to counter that problem I had to implement a strategy on top of alpha beta that checks for very bad states, which in turn increases the computation time. Similarly, alpha beta is very heuristic dependant, so if your heuristic does not recognize a certain critical state, it will possibly miss a win or loss. That is a failure mode for me sometimes as my heuristic never checks for the tricky diagonal after the fourth move (win or loss

with a diagonal spanning 3 quadrants). Then, if the other player uses one, my algorithm risks to not detect it. Finally, compared to Monte Carlo which gets more powerful with time as they can simulate more games for every candidate move. The problem with my alpha beta as that I feel the heuristic gets less consistent with time, because it sees lots of bad states or lots of good states and it gets confused. Therefore, I am in trouble for longer games.

4. Other approaches

I spent around a week working on a Monte Carlo approach. I just coded a custom Monte Carlo algorithm using the course notes as a guide. I had a method for selection, expansion, simulation and backpropagation. For selection, using my scaling constant I would try to favor exploration over exploitation. That leads to my first problem of that approach, as during the start of the game the branching factor was too big so it would only visit many nodes once, so it would miss obvious good moves on many occasions. Another issue, which was arguably worse, was that my algorithm would often timeout. I would do a check at the top before every selection phase and if the time exceeded like 800ms (not a lot I know – another issue) I would cut it off and find the best value. The issue is it would sometimes get lost in the generate tree methods or the expansion for some reason. It would be even worse when I tested against people on a network. The Monte Carlo did have the big advantage of doing the work for you though as I did not have to worry about a heuristic. The "heuristic" was just a simple 1 for win, 0.5 for draw, and 0 for a loss. I also tried to pick the 5 best scores from alpha beta and run 25 games of Monte Carlo of all random moves with them (simulation phase) and pick the move with the best score. I did not pick that idea as the number of games were too small so it would not return consistently the best result. Also, it would approach too much the time limit.

5. Ideas for an improved AI

My first idea for an improved player is to prune with a custom algorithm a big part of the legal move. Every legal move that seems to go to a negative heuristic (a state that favors the opponent), should be pruned right away, which would allow me to go much deeper in my tree. A second idea would be to use learning. What I mean is that for many different states I could run plenty and plenty of games, and write the stats to a file (doing that before the competition). During the first 30 seconds, I could use that to read the data. Then, during my moves, if I recognize a certain good state, I would pick that move.