COMP 424 – Artificial Intelligence – Final Project Report

Rayan Bouamrane

260788250

Introduction

Pentago is a two-player perfect information strategy game. The board is made up of a 6x6 grid divided into four 3x3 sized quadrants, and players take turns placing a marble onto a grid space and rotating a quadrant 90 degrees clockwise or anti-clockwise. Pentago-twist is a variant of Pentago where quadrants instead can be either rotated 90 degree clockwise or flipped horizontally. A player wins by linking 5 marbles together in row, vertically, horizontally, or diagonally.

Theory

Minimax aims to find the optimal play every level of the search tree until terminal nodes are found. The algorithm finds the best move for the opponent, assumes it is played, and proceeds to find the best move for the player, repeating this at every depth. At a max node, we store the best value of all the children of the node, and at a min node, we store the worst value of children nodes. Alpha-beta pruning is a modification of minimax aimed at traversing the search tree faster by discarding nodes we know we do not need to traverse. This returns the same move as minimax while traversing fewer nodes and spending less computation time. When we come across a possibility worse than an outcome we have already seen, we prune it.

Motivation

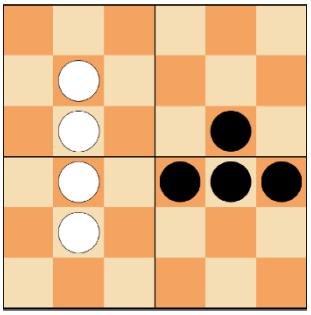
I chose this method as it has proved successful for AI mastering other complicated two-player strategy games such as chess. As a recreational chess player, I looked into the strongest chess artificial intelligences. The current strongest chess engine, Stockfish, was, until June 2020 implemented using alpha-beta pruning. This allowed the program to analyse to a greater depth than other chess engines, while taking comparable time to compute accurate moves. Pentago-twist has many similarities to chess: The games are deterministic, perfect-information, and zero-sum. They have rapidly growing but nonetheless finite search trees, meaning optimal solutions exist, and assuming perfect play for our opponents is reasonable. These similarities suggested to me that an approach that mastered chess, achieving a level 600 rating points higher than the strongest chess players, could be used for Pentago-twist. Furthermore, as Pentago-twist has an exponentially growing search tree, a method that limits the trees we need to traverse is not only ideal, but necessary.

Technical Approach

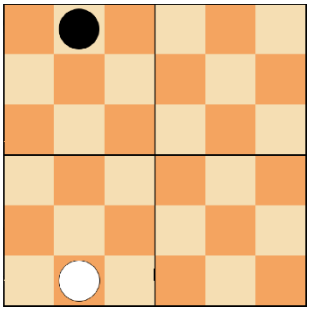
My AI used a depth limited minimax with alpha-beta pruning to play Pentago-twist. When the AI searches deeper in a tree, if it finds a move worse than a previously examined one, it discards those possibilities, saving computing time at no cost of move strength. If we have reached our maximum depth, found a game over state, or are nearly out of computing time, we run an evaluation algorithm on the board state. The evaluation function tells analyses how winning we deem the position to be for a given player.

Our evaluation works as follows: If a position is won or lost, we return $+\infty$ or $-\infty$, meaning the position is the best-case or worst-case respectively, and no further analysis is required. We will follow that variation or avoid it accordingly. If we were not depth-limited, and the game tree was feasible to traverse, this criterion would be sufficient to optimally play Pentago-twist.

If we do not see a winning combination, we look to maximize the number of marbles on open lines. An open line is what I call a row, column or diagonal where only our colour of marble is present. This represents a potential for 5 marbles in a row to be placed. If a line has marbles of both colours, it is either impossible or extremely unlikely that this line will have 5 marbles of a colour in a row, meaning the player has no real chance of victory in that line. The algorithm counts the number of marbles on the 6 rows, 6 columns and 6 diagonals of the board.



The 2nd is an open line for white.
The 4th row is a closed line for black.



By our definition, the column is open for black and white, but not by our implementation. This should not affect optimality, as if more marbles are filled in column 2, a directly winning line will be observed if one arises.

Another possible heuristic would have been to try to maximize the number of marbles on the single longest line we observe. For example, in figure 1, white's longest line is of length 4 and black's is of length 3. While this is useful to consider, and it is considered in the positional advantage portion of the evaluation function, it is too primitive and easy to counter to be the main evaluation criteria. Forming a single long line takes over 3 moves and can be stopped with 1 marble from the opponent. This strategy would allow an opponent to build advantages while using just a few turns to nullify our strategy. What I settled on was using to this metric alongside the counts of marbles on open lines. In practice, this means that if two board positions show white to have the same number of marbles on open lines, the position where white has a combination of horizontal, vertical, and diagonal open lines with 3 or marbles will be preferred.

Advantages

The implementation is quite flexible, the evaluation function can be tweaked, and the search depth can be increased if the time constraints are relaxed.

The program is quite fault tolerant. It will never return an illegal move and be disqualified, and is very unlikely to time out, as once 1800ms have elapsed, the only method that will execute is the evaluation, which is constant time.

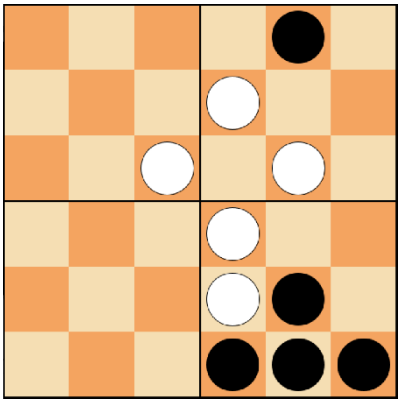
The evaluation function we have implemented values positional and non-overextending play, engages in strong attacks and stopping opponents' threats. The chess Grandmaster Lev Psakhis remarked that in chess you want to "prepare slowly, but attack fast." This AI follows a similar principle, when there are no threats on the board, the AI improves the position slowly, and prevents threats from arising, but can quickly switch to finishing the game swiftly if an opportunity arises.

Disadvantages

The program tends to occasionally take quite long to choose a move. While this wont result in a timeout, it can result in the program returning a suboptimal move, or sometimes returning a random if alphaBeta() never returned back to the top depth and updated the best move.

Given 2 seconds, the max depth the AI can consistently return its best move is 3. It seems that given that time the AI should have been able to analyse deeper. This is perhaps down to needing code optimizations or more selective pruning.

The evaluation function used can be improved. In an open line, we can consider it more advantageous if the marbles are all adjacent as opposed to spaced out, as they require more attention from the opponent. We also could have considered cases where a move can increase the score significantly such as the position below.



A white marble in the 3rd row, 4th column gives numerous strong threats.

Alternative Approaches

Monte Carlo tree search was the other main approach I explored, though I quickly settled on using alpha-beta pruning. Monte Carlo tree search does not assume an optimal AI opponent in the way alpha-beta does, though this is not a big drawback for the latter: During testing, my AI will be playing another strong AI, so assuming it plays near optimal is reasonable. If the opponent plays vastly different from my AI, I contend it would mean it is in fact playing suboptimally, and my AI would take advantage of that.

Another interesting approach, albeit one that proved infeasible immediately was solving the game. The original Pentago game was solved in 2014 by Geoffrey Irving. This was done through retrograde analysis, moving backwards from positions with 36 stones down to zero. Irving then stored all positions with 17 moves or less and their optimal moves forming the opening theory of Pentago, and all subsequent optimal moves can be feasibly calculated in reasonable times. The opening theory is 4TB and moves can require up to 20 seconds to calculate. Without the restrictions on this assignment the same approach could be used for Pentago-twist as the games are so similar and have the same number of possible moves and states.

Improvements

This AI could incorporate opening theory as well as endgame tablebases. Opening theory would either need to be developed by humans, based on heuristics and principles that have been shown to work in practice. A retrograde approach to optimizing opening moves such as what was undertaken to solve Pentago is simply unfeasible for us.

This approach could be used to develop endgame theory, however. This is analogous to the advancements in understanding chess endgames. In 2012, every single chess position with 7 pieces or less was solved by working backwards from every checkmate to every position that could force it. We could similarly work backwards from winning Pentago-twist positions and find the moves that lead to a win down the line. These would only need to be computed once, and allow for a greater search depth in certain forcing positions at little time cost.

We could also have improved the alpha-beta by improving the move order by which we traverse the tree. If we can ensure better moves are traversed first, there will be more cutoffs and fewer subtrees to traverse. The killer heuristic is an example of such an approach, where moves that previously were shown to be good are analysed first in future analysis. The combination of these approaches would create a more efficient AI, possibly one capable of analysing deeper in the same length of time.

References

[1] Wikipedia contributors. (2021, March 6). Alpha–beta pruning. Wikipedia. https://en.wikipedia.org/wiki/ Alpha–beta_pruning

[2] Russell, S. J., & Norvig, P. (2002). Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall.

[3] Kaufman, L. (2013, November 24). Stockfish depth vs. others; challenge. TalkChess. http://www.talkchess.com/forum3/viewtopic.php?start=0&t=50220

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484–489. https://doi.org/10.1038/nature16961

[5] Irving, G. (2014). Pentago is a first player win. Perfect Pentago. https://perfect-pentago.net/details.html

[6] Irving, G. (2014). Pentago is a First Player Win: Strongly Solving a Game Using Parallel In-Core Retrograde Analysis. ArXiv. https://arxiv.org/pdf/1404.0743.pdf

[7] Büscher, N. (2011). On Solving Pentago. Technische Universität Darmstadt. http://www.ke.tu-darmstadt.de/lehre/arbeiten/bachelor/2011/Buescher_Niklas.pdf

[8] Advanced Research Projects Agency, & Liskov, B. (1968, August). A program to play chess end games. https://apps.dtic.mil/dtic/tr/fulltext/u2/673971.pdf

[9] Makhnychev, V., & Zakharov, V. (2012, July). Lomonosov Endgame Tablebases. ChessOK. https://chessok.com/?page_id=27966