CS4386 Assignment 1 (Semester B, 2021-2022)

Student name / Student no.: WANG Yinuo/55670000

March 19, 2022

1 Introduction

There are two players in this game. In each turn, a player can put an 'X' in the empty cell of the 6*6 board. Any player who can connect 3 or 6 adjacent 'X'(color does not matter) in a row/column (no diagonal) will get 3 or 6 points respectively. The game ends when all the cells are taken by players and the player who gets higher points is the winner.

This project aims to design an intelligent AI player to make the game a man versus machine or machine versus machine game.

2 Related work

2.1 Human versus machine game

The human-machine game is an essential branch of artificial intelligence, and a lot of research results have been produced to explore this field. The simplest case of a game is a zero-sum game. Zero-sum games are completely antagonistic and intensely competitive games. In a zero-sum game, the sum of the players' gains is zero (or some constant), and one player's gain is another player's loss. Based on the zero-sum game, the minimax algorithm is the most basic algorithm that Shannon formally proposed in 1950 [1].

2.2 Game Algorithm

The basic algorithm for two-player games is the alpha-beta pruning algorithm, which improves the MiniMax algorithm. In the human-machine game, the two sides play on a turn-based basis. One side considers the possible moves the other side will take if it makes a particular choice among all available moves to choose the most favorable move for it. This game process constitutes a game tree, in which both sides constantly search and choose the most favorable child node to move. In the process of searching, the party taking the maximum value is called Max, and the party taking the minimum value is called Min. Max will always choose the most valuable child node to move, while Min will do the opposite. That is the core idea of the minimization Max algorithm.

The most significant disadvantage of the minimax algorithm is that it will cause data redundancy, which can be divided into two types: (1) Maximum redundancy; (2) Minimum redundancy. Correspondingly, alpha pruning is used to solve the maximum redundancy problem, and beta pruning is used to solve the minimum redundancy problem, which constitutes the complete alpha-beta pruning algorithm.

In 1955, MacCarthy [2] proposed the alpha-beta pruning algorithm. In 1975, Knuth et al. [3] optimized the algorithm and proposed the negamax algorithm. The principle of negamax algorithm is essentially no different from the minimax algorithm, but it does not require the system to distinguish maximum and minimum, making the algorithm more unified. In addition, Knuth et al. [3] also carried out in-depth research on the search efficiency of the alpha-beta pruning algorithm, and Pearl [4] proved the optimality of the alpha-beta pruning principle in 1982.

In addition to the alpha-beta pruning algorithm, human-machine game algorithms include reinforcement learning, MCTS, dynamic programming, genetic programming, etc. In this project, the alpha-beta pruning algorithm is adopted.

3 Methodology

In order to improve efficiency, I replace minimax algorithm with negamax algorithm in implementation.The negamax algorithm is a variant of the minimax algorithm, and its basic principle is based on the following formula:

$$max(a, b) = -min(-a, -b) \tag{1}$$

Choosing the node with the lowest score is equivalent to multiplying the score of these nodes by -1 to get the node with the highest score. In this way, the negamax algorithm unified each step of the recursion process, and each recursion selected the maximum.

The alpha-beta pruning algorithm improves performance by cutting out redundant branches. As shown in the figure below, this is a game tree. The data under the node is the value of the node. The value of node B40 is 60, the value of node B41 is 63, the value of node B42 is 15, and the value of node B43 is 58. Node N30 is valued between -60 and -63, so the value of node N30 will be equal to -60; Node N20 is evaluated between the negative value of node N30 and the negative value of node N31. The value of node N30 will be equal to -60, while the value of node N31 will be greater than or equal to -15. Therefore, the value of node N20 will be greater than or equal to 60. Thus, the operation of subtracting the node B43 is called pruning.

The pseudo-code for the algorithm is attached to this example.

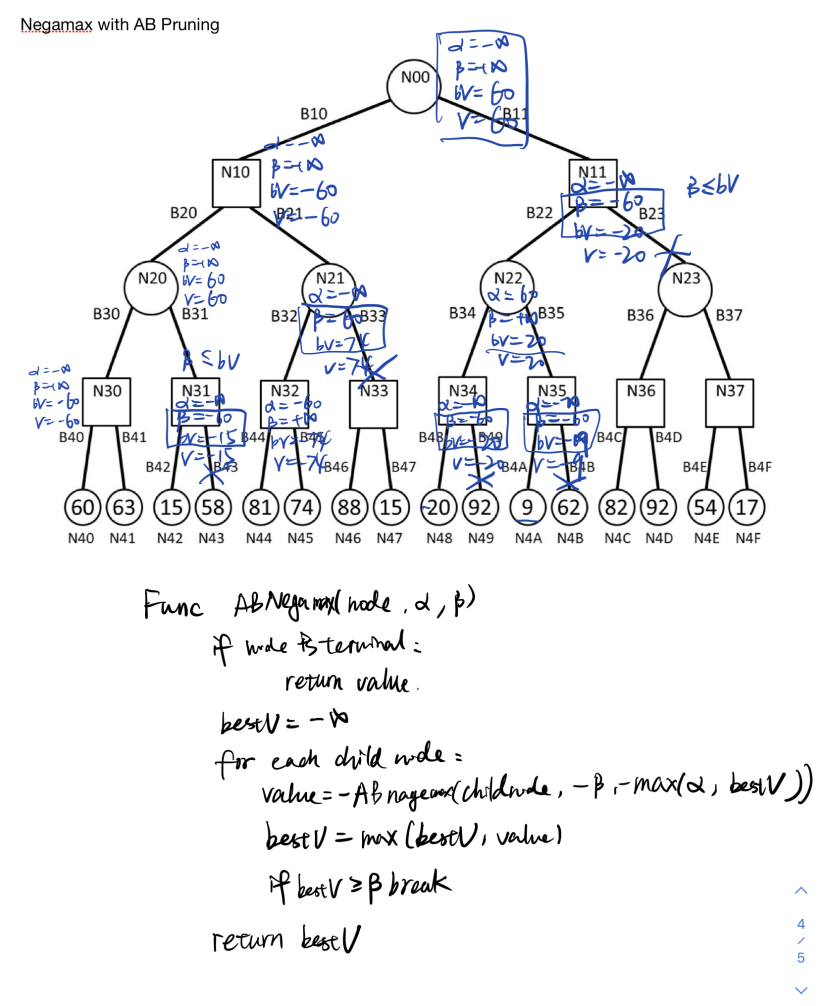

Figure 1: Negamax with alpha-beta pruning algorithm.

4 Implementation, Experiments, Analysis

In implementation, I created a "Broad" class to simulate two players moving. Broad contains a grid board and two player objects, using the take_move() function to simulate player moves.

The main function used to return a good action is abnegamax(board, maxDepth, currentDepth, alpha, beta), a recursive function.

Due to time constraints, I set the depth to 4 and the function ended up with good performance. Experiments have proved that in human versus machine or machine versus machine game, the latter player has a better chance of winning.

For code implementation, please refer to the code I submitted. Each function and step has clear and concise comments.

5 Conclusion and reflection

In summary, Negamax with alpha-beta pruning algorithm created AI players that did well in this game, but there might be a few problems:

The first is that Negamax with alpha-beta pruning algorithm is based on the prior knowledge that the other player will always choose the worst case for ourselves. However, this prior knowledge may not be positive in real situations, because it is impossible to determine whether the other party's game strategy is consistent with its own. If the other person's strategy is not optimal, then the assumption that my strategy is optimal ceases to exist.

Secondly, even if I prune with an alpha-beta pruning algorithm, this is so location-dependent that it might actually be as complex as a depth-first search.

Finally, it is difficult to determine whether the upper and lower bounds of the assessment are accurate, and it is easy to feel trapped in local optimality if the depth of backtracking is not deep enough. So probabilistic models (like Markov decision making and reinforcement learning) are worth a try in this case.

References

[1] C. E. Shannon, "Xxii. programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[2] J. McCarthy, "Human-level ai is harder than it seemed," 1955.

[3] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[4] J. Pearl, "The solution for the branching factor of the alpha-beta pruning algorithm and its optimality," *Communications of the ACM*, vol. 25, no. 8, pp. 559–564, 1982.