International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT1228644

Analysis of Game Tree Search Algorithms Using Minimax Algorithm and Alpha-Beta Pruning

Prof. Sumit S Shevtekar, Mugdha Malpe, Mohammed Bhaila
Department of Computer Technology, Pune Institute of Computer Technology, Maharashtra, India

Article Info

Publication Issue :
Volume 8, Issue 6
November-December-2022

Page Number : 328-333

Article History
Accepted: 12 Nov 2022
Published: 28 Nov 2022

ABSTRACT

An important topic of research in computer systems is the optimization of finding the optimum course of action based on different variables, such as the environment's state, the system's goal, etc. The building of the entire state search space, also known as the minimax algorithm, can result from any search algorithm's attempt to find the best feasible solution from among all known possibilities. The recursive backtracking algorithm known as Minimax is used to select the next action in a game of strategy for two players. The algorithm works well because it anticipates that your adversary will play well as well. However, as the tree's depth increases, we observe that minimax frequently investigates repetitive and unlikely situations. We'll also take a look at the minimax extension known as alpha-beta pruning, which prohibits us from considering states that won't be chosen. We will also examine a number of established techniques for resolving two-player games, such as adversarial search and other machine learning-based techniques.

Keywords : Minimax algorithm, Alpha-beta pruning, Two-Player games, Game Theory, Game Tree Search Algorithms.

I. INTRODUCTION

The study of mathematical models of communication tactics between decision-makers is known as game theory. It is employed in numerous logic, computer science, and social science domains. The game theory, which now relies on a certain kind of behavioural interaction, has evolved into a catch-all phrase for the study of rational decision-making.

Scholars have noted the importance of game theory as a tool for comprehending a variety of fields. Game theory has been applied to create theories of ethical or conventional behaviour, further used to evaluate, anticipate, and define behaviour. Game theory-based principles are applicable to definition and modelling, business and economics, politics, project management, philosophy, computer science, and other fields. Recursive or backtracking algorithms include the mini-max algorithm. It is utilised in game theory and the decision-making process, as was previously mentioned. Assuming that the opponent is likewise playing really well, it provides the player a flawless move. Recursion is used by the minimax method to search across the game-tree. In 2-player games like tic tac toe, chess etc. the min-max algorithm is

frequently utilised [1]. The minimax choice for the current state is computed using this algorithm. When playing this game with two players, each player plays the game so that they gain the most from it and the opponent player gains the least. They select this strategy so that they receive the greatest benefit and their adversary receives the least benefit.

When exploring the game tree, the minimax method traverses till the tree's terminal vertex before going back up the tree using recursion. Minimax frequently explores duplicate states and states that are unlikely to be picked by the players, however, as the depth of the tree rises. This is where we present the idea of an additional optimization method that helps to avoid this: pruning by alpha-beta.

A more advanced variant of the minimax method, alpha-beta pruning has major advantages over the minimax algorithm. As previously established, the recursive backtracking of the game tree causes the number of outcomes it must evaluate to exponentially increase as the depth of the tree increases. By using a technique called pruning, we may compare the original result to the right minimax result and do so without having to examine every vertex in the game tree. Since it affects the two growth threshold factors "alpha" and "beta," it is known as "alpha-beta pruning."

II. LITERATURE SURVEY

The following table compares the 2-player game theory approaches and algorithms to demonstrate the literature review:

No.	Algorithm	Version of	Method	Examples
1	Negascout	Minimax	Reducing Calculation, that is, we do not thoroughly explore each node by excluding options that both players ignore.	zero-sum games tic-tac-toe checkers
2	Monte Carlo Tree search	Alpha-Beta Pruning	It is a simulation-based best first search algorithm that has been expanded to support pruning in the Alpha-Beta pattern [3].	zero-sum games like tic-tac-toe
3	Principle Variation Splitting (PVSplit)	Alpha-Beta Pruning	It is a parallel Alpha-Beta pruning algorithm that stipulates that before exploring more branches, one must first search the initial branch at a PV node.	zero-sum games like checkers
4	Young Brothers Wait Concept	Alpha-Beta Pruning	Here, before generating the other siblings in parallel, the first sibling node is searched.	-

III. METHODOLOGY

A. ALGORITHMS

In the context of a game, this section discusses the search algorithms Minimax, Alpha-Beta Pruning, and NegaScout. A game tree that includes all of the potential movements a player might make serves as the foundation for all algorithms.

1. Minimax

A prominent backtracking method in game theory is called Minimax. The Minimax algorithm iterates around the game tree to find the optimum move, returning it depending on the score at the leaf node. You may find a more thorough explanation and pseudocodes ahead. [1].

In two-person zero-sum games, each player attempts to minimise their prospective loss (penalty) while maximising the damage of their opponent. The backtracking technique known as Minimax is used in game theory to make decisions. It presumes that both players are performing at their highest level.

The terms "maximizer" and "minimizer" refer to two players. The former seeks the highest score possible, while the latter seeks the lowest score possible. The name "minimax" was created as a result of the fact that when one player wins, it automatically means that the other player loses.

Each zero-sum game has a score attached to it, and at any given time, if the maximizer is in the lead, this score will be in the positive range, and if the minimizer is ahead, it will be in the negative range. To further demonstrate the idea, consider game where, in the event that player 2 likewise plays optimally and chooses a course of action that results in a maximum payout of -V, player 1 can obtain V as the highest prize. Numerous zero-sum two-player games, including tic-tac-toe, chess, checkers, and others, may be played using this method. Here, with the aid of a diagram, we'll look at a tic-tac-toe example.

There are two participants in the game of tic tac toe: X and O. In this case, X is the maximizer and O is the minimizer; each have an equal probability of winning, losing, or drawing the game. They will pick that move if it puts them in a position where either of them has a strong probability of winning. Otherwise, if no move results in a win for the current player, the player will attempt to make a move that will result in a draw. With the aid of the graphic below, where a game has already been played up to a certain point and X must now participate, let's better comprehend this.

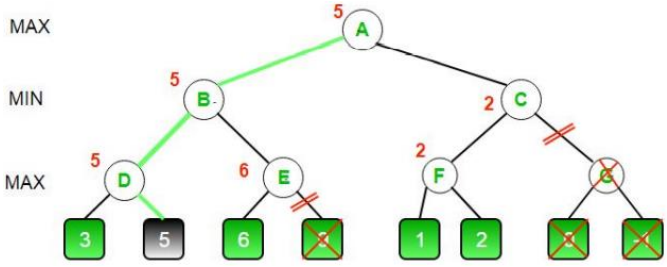

Figure 1 : Demonstration of tic-tac-toe using Minimax Algorithm

As can be observed, in the first level of the game tree, player X has a choice of 3 alternative nodes. However, closer examination shows that if player X chooses to play the left node, [2,0], player O will have two alternatives for the following move, and as both players aim to maximise them. In order to win the game, player O will select the first node since it yields a score of -10, whereas the other move would result in a tie and a score of zero. Therefore, adopting this action for player X is not ideal since it will result in Player O receiving the most advantage and the least amount of punishment.

Player O will have to select one of two nodes at the following level if player X takes the middle move, [2,1], however, player O cannot win in either of the two moves because the 1st node evaluates to +10 and the 2nd one results in net zero. Player O will try to maximize the maximum possible award for

themselves and minimize the maximum possible reward for their opponent as they play optimally as well, and as a result, they will select the second node to conclude the game in a tie. As a result, player X will have the highest chance of drawing by selecting the centre node.

However, if player X selects the correct node, [2, 2], then it will immediately result in player X's win with a value of +10, in order to maximise points, the player X will always play ideally.

2. Alpha-Beta Pruning

To find the best move, the Minimax algorithm must run through the whole game tree. The Minimax method is improved by the AlphaBeta Pruning technique, which prunes the tree nodes that have little probability of delivering a better move and does not assess them [7]. When pruning results in bypassing an entire sub-branch of the game tree, it saves a lot of time. However, the Alpha-Beta Pruning algorithm's worst-case performance is equivalent to that of the Minimax.

The minimax analyses every possible outcome of the game tree and that it grows exponentially as the depth of the tree increases. Since a more advantageous course of action has already been discovered, it prunes the unnecessary branches. Alpha-beta pruning was given this name because it does this by adding two more parameters to the minimax algorithm, specifically alpha and beta.

When used on a typical minimax tree, it produces the identical move that minimax would, but it also verifies its veracity by removing branches that cannot potentially affect the final choice. Alpha-beta pruning has the advantage of allowing the search tree's branches to be removed. In this manner, a deeper search may be done while still limiting the search time to the subtree that is "more promising."

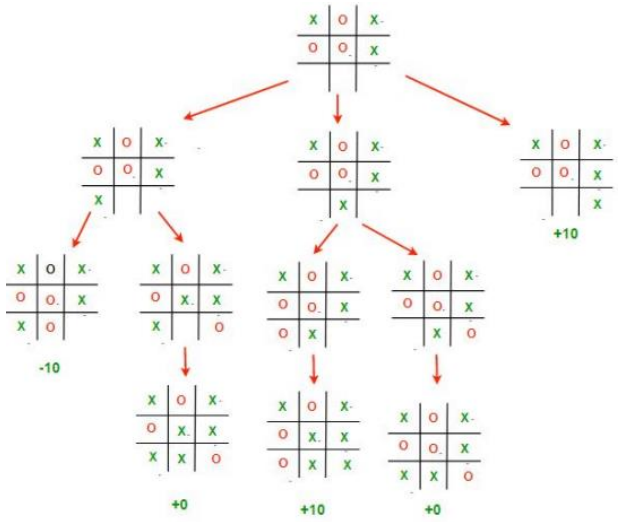

Figure 2: Alpha-beta visualization

Alpha-beta pruning has two parameters: alpha and beta. Largest value of maximizer at or above the stated level is alpha, whereas the greatest value of minimizer at or above the specified level is beta.

Let's see how these 2 parameters are used in practise. As we know, the score rises positively for the maximizer and negatively for the minimizer, therefore initially, alpha is set to minimum negative value and beta is set to maximum positive value, meaning that both players start with their lowest score. We shall now examine the circumstance in which a subtree may be removed. We can discard that subtree if a stage is reached when the highest score of the minimizer becomes lesser than the lowest score of the maximizer, that will never be considered.

3. NegaScout

The window (α, β), is where the Alpha-Beta Pruning algorithm starts the search, skipping any nodes that are outside of this window. By assuming that the first node found is the best node, the NegaScout algorithm seeks to raise the number of cut off nodes even further [5]. Using the exception of any nodes that violate the aforementioned presumption, the remaining nodes are only examined with a null window of (m, m+1) and a full window (α, β), research. The solver's performance is enhanced by the null window search's increased number of cut offs.

B. PSEUDOCODES

1. Minimax

As a result, we covered the minimax algorithm in the preceding part. Here, we'll try to implement the minimax pseudocode.

Algorithm 1: Minimax Algorithm

```
function miniMax(vertex,maxDepth, maxiPlayer)
 if maxDepth = 0 or vertex is a leaf node then
    return value of vertex
end
if maxiPlayer then
    score = -∞
    while every child of vertex do
        score=max(score, miniMax(child,
        maxDepth -1, FALSE))
    end
    return score
end
else
    score = +∞
    while every child of vertex do
        score=min(score, miniMax(child, maxDepth
        -1, TRUE))
    end
    return score
end
```

Algorithm 2: Sequential Alpha Beta Pruning Algorithm

```
function alpha_beta(vertex, maxDepth, α, β,
maxiPlayer)
if maxDepth = 0 or vertex is a leaf node then
    return value of vertex
end
if maxiPlayer then
    score = -∞
    while every child of vertex do
        score=max(score,
        alpha_beta(child,maxDepth-1,α, β, FALSE)
        α = max(α, score)
        if β ≤ α then
        break
        end
    end
    return score
end
else
    score = -∞
    while every child of vertex do
        score=max(score,
        alpha_beta(child,maxDepth-1,α, β, TRUE)
        β = min(β, score)
        if β ≤ α then
        break
        end
    end
    return score
end
```

Algorithm 3: Negascout Algorithm

```
function NegaScout(game
Position,depth,alpha,beta)
 if depth = 0 or game is over
    return Eval(gamePosition)
end
n=beta
score=∞
Generate(gamePosition)
for i=1 to sizeof(moves) do
    Make(moves[i])
    if curr>score then
        cur = -NegaScout(gamePosition, depth-1, -n,
-alpha)
    end
     if n=beta or d<=2 then
        score=cur
    end
    else
        score= -NegaScout (gamePosition, depth-1,
        -beta, - cur)
    end
    if (score > alpha)
        alpha = score
    end

    if (alpha >= beta)
        return alpha;
    end
    undo(moves[i]);

    n = alpha+1;
    return score;
end
```

IV. CONCLUSION

Game theory is a plain, easily understood idea that is also evident in everyday life. In two-player games, the Minimax algorithm always selects the best move for the player, supposing that the other player would always play strategically as well. Minimax is so named because, as can be seen, it seeks to maximise player profit while minimising that of the opponent. Numerous 2-player zero-sum games, including tic-tac-toe, chess, checkers, and others, have been taken into consideration.

We have attempted to make the method optimal by seeking the depth iteratively with alpha-beta pruning such that a successful move is still followed in the event of an interruption because the minimax algorithm is depth-first and its states expand exponentially. Furthermore, simultaneous alpha-beta pruning, which aims to accelerate the present alpha-beta pruning by an average of 3.03, has recently come up for discussion. If not, it would be necessary to investigate each of the tree's exponential game states, which would be incredibly expensive. Consequently, alpha-beta pruning improves the minimax method by preventing state exploration.

V. REFERENCES

[1]. Pranav G., Satvik M., Neeta P. “Realization of Game Tree Search Algorithms on FPGA: A Comparative Study”, 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2019, pp. 1-3.

[2]. Shubhendra P. S., M. Sridevi. “Comparative study of performance of parallel alpha Beta Pruning for different architectures” 2019 IEEE 9th International Conference on Advanced Computing (IACC), 2019, pp. 115-119.

[3]. Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, Jacek Mańdziuk. “Monte Carlo Tree Search: a review of recent modifcations and applications” Artif Intell Rev (2022) 10462-022-10228.

[4]. S. Ariyurek, A. Betin-Can and E. Surer, "Enhancing the Monte Carlo Tree Search Algorithm for Video Game Testing," 2020 IEEE Conference on Games (CoG), 2020, pp. 25-32.

Cite this article as :

Prof. Sumit S Shevtekar, Mugdha Malpe, Mohammed Bhaila, "Analysis of Game Tree Search Algorithms Using Minimax Algorithm and Alpha-Beta Pruning", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 6, pp. 328-333, November-December 2022. Available at doi :
https://doi.org/10.32628/CSEIT1228644
Journal URL : https://ijsrcseit.com/CSEIT1228644