An Intuitive Implementation Of Alpha-Beta Pruning Using Tic-Tac-Toe

Conference Paper · July 2020

CITATIONS
0

READS
2,450

3 authors, including:

Asad Ali Jatoi
Mehran University of Engineering and Technology
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Isma Farah Siddiqui
Mehran University of Engineering and Technology
82 PUBLICATIONS **220** CITATIONS
SEE PROFILE

An Intuitive Implementation Of Alpha-Beta Pruning Using Tic-Tac-Toe

Asad Jatoi, Sarvech Ali and Isma Farah Siddiqui*
Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan
E-mail: 17sw45@students.muet.edu.pk , 17sw65@students.muet.edu.pk, isma.farah@teacher.muet.edu.pk
*Corresponding Author : Isma Farah Siddiqui

Abstract

This proposed work presents an intuitive method to implement the Alpha-Beta pruning followed by minimax algorithm, a back-pedal algorithm that is used in option choosing from combinations of several alternatives, to achieve better understating of how Artificial Intelligence (AI) works. By combining complex algorithms with regular paper-and-pencil game Tic-tac-toe leads toward a clear and intuitive learning. Such basic games are deeply melted in our instincts thus the concept of pruning can easily be approached. Minimax is used in game playing to find the best efficient move for a player assuming that the rival player will play to win too. Later Alpha-Beta pruning will equalize the minimax algorithm. It returns the same move but it removes all the branches that will not be affecting the final part of the game. For output, a standard grid of nine squares (3x3) is used, consisting of 3 rows and 3 columns. However, this grid can further be extended to 4x4 or 5x5 to increase the complexity of the algorithm.

Keywords: *Game playing, Minimax, Maximum, Artificial Intelligence, Tic-Tac-Toe, Alpha-Beta Prunning*

1. Introduction

Artificial Intelligence is no for a new concept for the world. However, the complexity of understanding the working principals has increased day to day. As new complex algorithms and techniques are being introduced, placing obstacles for beginners. The Alpha Beta pruning is one of the fundamental concepts to start with. By using regular paper-pen game concept such concepts can be taught easily.[1]

This paper proposed an intuitive implementation of the AlphaBeta followed by MINIMAX algorithm for tic-tac-toe game using 3x3 grid structure. The remaining paper is organized as, Section 2 gives information of the used algorithms, Section 3 discusses the proposed algorithms and implementation, Section 4 discuss the results obtained from these algorithm implementations and finally, Section 5 gives conclusion of this proposed work.

2. Background

2.1 Alpha-Beta Pruning

Alpha: It is the best choice for the player having max part. In this case we want to get the higher value.

Beta: It is the best choice for min part. In this case the low value is go-to-go.

AlphaBeta pruning is applied to equalize the Minimax. This will return the best available move from the rest in other words it will trim the rest of combinations for that particular move unaffecting the game.[2]

2.2 Minimax

Minimax is a kind of back pedal decision taking algorithm used in game trees to find the best and efficient option or the choice for the player considering that the counter player will also choose the best move. The authors of [3] have proposed a optimally modified MINIMAX

algorithm as Rminimax with improved results. This type of algorithms are used in board games such as TicTacToe, Chess etc. In Minimax algorithm the two players are known as maximizer(+1) and minimizer(-1). The maximizer tries to obtain the higher value while the minimizer tries to get the opposite value i-e low value.[4]
The pictorial form of the Alpha Beta Pruning is shown in **Fig. 1**.

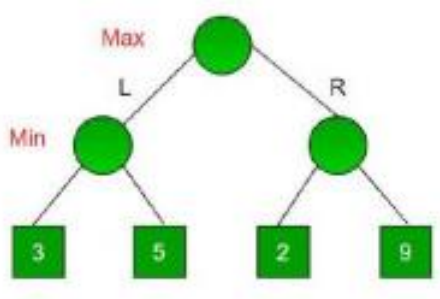


Fig. 1. Minimax algorithm in tree structure.

3. Implementation

Algorithm

Step#1: Begin / being assigned the move type.

Step#2: Find all the alternative moves If there are not any Go To Step#11.

Step#3: From the alternatives if a win move is found for the system then choose that and Go To Step#9 Otherwise follow Step#4.

Step#4: From all the alternatives if there is such a move with which the opponent can win then choose that and Go To Step#9 otherwise follow Step#5.

Step#5: From all the alternatives if there are any moves at the corners then choose them and Go To Step#9 otherwise follow Step#6.

Step#6:From all alternatives if the center is unoccupied then choose that move and Go To Step#9 otherwise follow Step#7.

Step#7: From all the alternatives if the sides are empty then choose side moves and Go To Step#9 otherwise follow Step#8.

Step#8: Check if there are empty moves then Take input from the opponent and Follow Step#2 .

Step#9: Check the condition if any of win-fit move is formed for the system then system won and GoTo Step#12 otherwise follow Step#10.

Step#10: Check the condition if any of the win-fit move for the user is formed then system lost, set the score (-1) and GoTo step#12 otherwise follow Step#11.

Step#11: Check if all the moves are occupied then the game is draw and follow Step#12.

Step#12: Terminate the game.

Pseudo code for Finding the Path:
minimax:

```
Body of the minimax()
Loop
        Get all the alternatives
        *if there is a win-move for the system
                assign that move to system
        move
                call checkResult() method
        *elseIf there is a move with which
        user can win
                assign that move to system
        move
                call checkResult() method
                get input move from user
                from user
        * elseIf there exists alternative in
        corners of the board
                assign that move to system
        move
                get input move from user
                from user
        * elseIf there exists an alternative for
        center position
                assign that move to system
        move
                get input move from user
                from user
        * elseIf there exists alternative for
        sides
                assign that move to system
        move
```

```
                get input move from user
                from user
        *else
                invoke resultCheck() function
        end of the Loop.
```

Checking Score:

```
Body of the checkScore method
Loop
        *if the system wins
                return 1
        *elseIf the user wins
                return  -1
        *else
                return 0 //(withdraw)

end of the loop
```

Checking Result:

```
Body of the checkResult method
Loop
        *if system move is
        diagonal
                assign result win to system
                invoke checkScore() method
                terminate;
        *elseIf user move is diagonal
                assign result lose to system
                invoke checkScore() method
                terminate;
        *elseIf there is no empty move
                assign result draw to system
                invoke checkScore() method
                terminate;
        *else
                return to the place from where
                called.
end of the Loop.
```

In above algorithm the function minimax() is used to calculate all the available alternatives for the player and the same method also returns the best move of all alternatives.

(i) Minimax:

The minimax() function will return the best available move for the current board-situation, any of the move can be verified via this function. The function will enlist all the alternatives and then results the best-of-all alternative. Guessing the other player will also play his/her best . Working of the maximizer and minimizer in the minimax() function is to set a best fit move.

(ii) Checking Result:

In order to verify if the game is over or not or the players are left with 0 moves (withdraw condition), the checkResult() function is invoked. The steady and simple function used to set the winning or losing state and to enhance and improve the effectiveness and progress of the intelligent system the function in last calls the checkScore() function. Which will keep the progress report of the system. The only motive of this function is the reward system used in MinMax (+1 for a win, -1 for losing and 0 for withdraw). [5]

Starting to the terminating of the game is drawn below in **Fig. 2**.

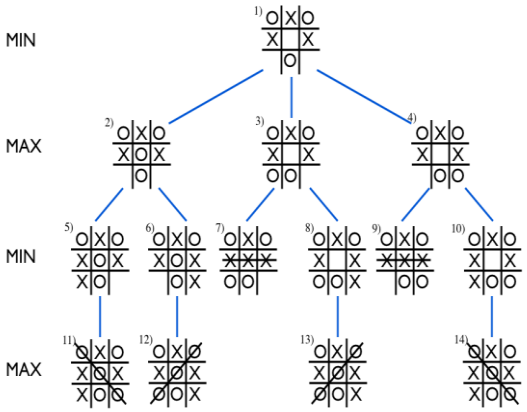


Fig. 2. Start-to-End working of algorithm.

4. Results and Discussion

The grid is a very important part of Tic-Tac-Toe. It has nine spaces. The player can draw anywhere they like. Here is the formation of a grid in below **Fig. 3**.

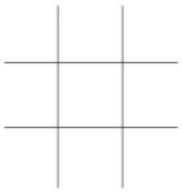


Fig. 3. Game Grid

The game starts with a blank board. In a continuous game playing mode the choice of X and O player and the fist move is alternative (changes from game tot game). For a single game the user can select any of the symbol in turns the algorithm expert system will be assigned the other symbol and the user will get the chance to play first.

From beginning to the termination of the game the moves are decremented. The algorithm understands the every combination in the form of a tree data structure. The algorithm iterates each and every combination and then select the best move from all alternatives (i-e Max or Min depending upon the player).[6]

After every move of the player(user) a tree structure is formed analysis of all the possible moves is carried and then selection of the best from all whether a min value or the max value depending upon the symbol it is assigned is made. This check-up of all moves makes the program expert unbeatable; however in game the draw is also possible when in last move the program is left with the choices of (-1) and (0) . A draw in this case is better than losing.

In such type of the games the minmax algorithm is best suitable.

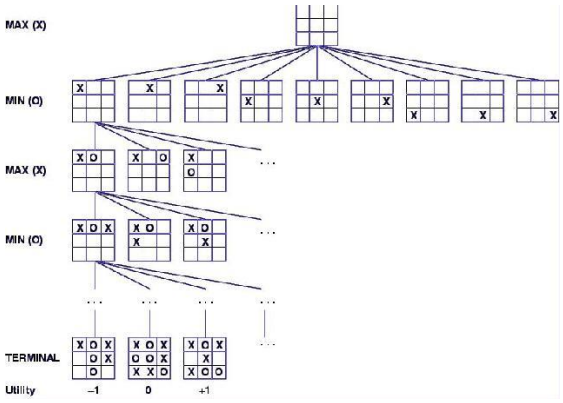


Fig. 4. Minimax applied from beginning

The **Fig. 4** shows the working of minmax algorithm from starting to the end of the game.

When all the moves are used i-e the board is full, then that is the ending of the game a draw condition however if a player has successfully made a fit (diagonal horizontal or vertical) then the results of the game are displayed at the time meaning that after each move the result function is invoked which checks the win condition if it is satisfied by X the X is winner otherwise the O.

The algorithm will always try not to give a easy win to the player although if a player tricks the algorithm then the algorithm will get a negative reward/score resulting to improve more next I'm time. On every game the algorithm either wins or it learns.

Following are some game iterations.At first the algorithm was tricked but after some iterations the algorithm blocked all the winning ways of the player.

In the **Fig.s 5-8**, demonstration of the algorithm is further seen as output to the user's moves.

Test1(Iteration 1): O be the player and X be the system

Fig. 5

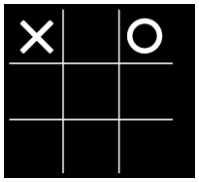
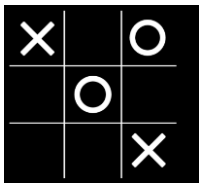


Fig. 6.



Test2 (Iteration n): X be the player and O be the system

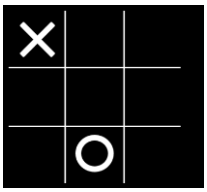


Fig. 7

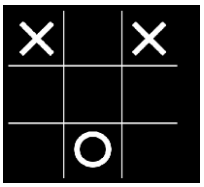
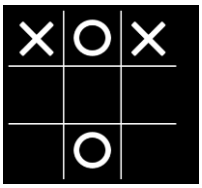
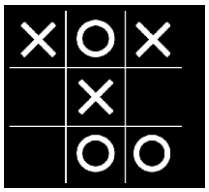


Fig. 8





5. Conclusion

This paper presents an intuitive implementation of Alpha-Beta pruning followed by minimax algorithm on a typical tic-tac-toe game. The back-pedal algorithm is used in option choosing from combinations of several alternatives, to achieve better understating of Artificial Intelligence (AI) work in game playing. Later Alpha-Beta pruning equalized the minimax algorithm. It returns the same moves, however,

removes all the branches that will not be affecting the final part of the game. The implementation is based on 3x3 grid for game.
In future this work can further be extended to 4x4 or 5x5 to increase the complexity of the algorithm.

6. References

[1] Ian Millington, John Funge Book: AI for games 2nd Edition (2009) 22-28.
[2] https://en.m.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning
(Last accessed on: 18 May 2020)
[3] Diez, Silvia Garcia, Jérôme Laforge, and Marco Saerens. IEEE Transactions on cybernetics 43.1 (2012): 385-393.
[4] https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/
(Last accessed on: 19 May 2020)
[5] https://www.codeproject.com/Articles/43622/Solve-Tic-Tac-Toe-with-the-MiniMax-algorithm
(Last accessed on: 20 May 2020)
[6] http://neverstopbuilding.com/minimax/
(Last accessed on: 20 May 2020)