Checkers using Min Max Algorithm

IIT2018045, Ananya Aparajita Mohanty

V Semester, B.Tech, Department of Studies in Information Technology,

IIIT Allahabad , Prayagraj, India

Abstract--In this report, we have devised an algorithm to build checkers game with a set of rules, using Min Max Algorithm that helps player to determine possible moves of its opponent and play accordingly.

I. INTRODUCTION

Checkers is a two player game wherein each player starts with 12 pieces and applies strategies to either eliminate or restrict every piece of the opponent. The game, thus, comes to an end when either all the pieces of a player are captured or in such positions that the player is left with no possible further move.

In this paper, we have implemented American checkers which is a 8x8 checkerboard, played by two computers who form their strategy using an artificial intelligence algorithm – Min Max. Furthermore, we have implemented Computer vs Human and Human vs Human.

II. DEFINITIONS AND CONCEPTS USED

(A) Game Tree

A tree, in computer science, is a nested data structure which starts with a root and branches into other nodes, called its child nodes.

We can obtain all possible outcomes of a game by figuring out its game tree, and an almost correct prediction of these outcomes can help a player win over the other.

(B) Min-Max Algorithm

It is used to search through all possible game trees in order to determine the best possible move for a given player.
It is generally used for two player games, in which each player tries to stop the other from winning by trying to figure out possible moves by the other optimally.

It is applied in games in which the heuristic is such that one player tries to maximise the value whereas the other tries to minimize it.

It can be understood as a type of depth first search algorithm, since for every move it goes into the depth of all possible ways the game can be won/lost, and backtracks to the current state after deciding which possible path is most favourable.

Thus, it recursively tries to obtain the best possible move for said player with the assumption that the other is also playing equally optimally.

(C) Alpha - Beta Pruning

Since obtaining all possible nodes will require a lot of computational time and memory, pruning is used. In this, it eliminates all such nodes (before reaching the conclusion) that have no affect on the outcome.

By convention, Alpha is chosen for the player trying to obtain the maximum value and Beta is used for the player aiming for the minimum value.

III. DESIGN AND IMPLEMENTATION

In this paper, we have two players denoted by Black and Red pieces. They are allowed to move only diagonally unless they are capturing the piece of their opponent, in which case they may take a leap of twice the diagonal length (provided that block is empty).

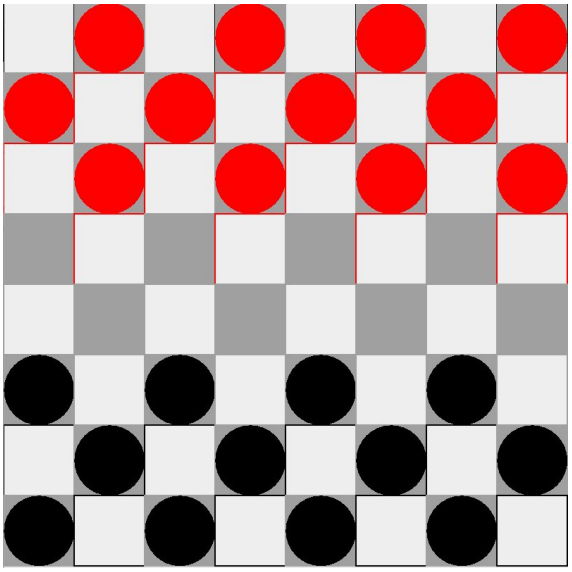

Fig.1 Checkers Board before the game begins.

Given below are the rules implemented in this version of the game:

i. Each player gets one turn at a time and it is Black that starts the game.

ii. For one's turn, the player checks for all possible combinations i.e. actions that it can take.

iii. A piece can travel diagonally but only forwards, i.e. it cannot go back.

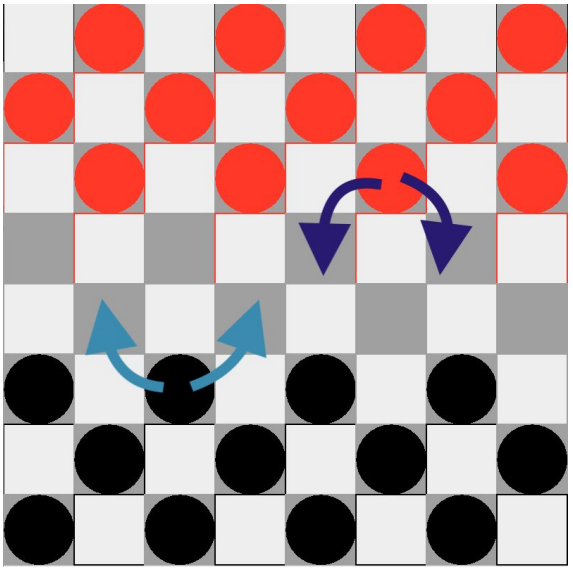

Fig.2: Possible moves for the pieces.

It may choose to jump over the piece of an opponent, if it leads to a capture, given the destination is empty.

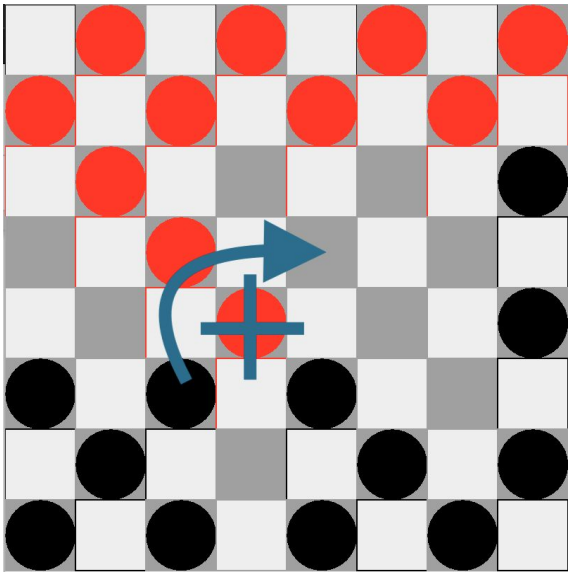

Fig.3: Black can jump over red since the next block is empty and it aims at killing the opponent.

A piece must go for multiple kills if it can. This means that after capturing one opponent, if it can capture another (and so on), it must do so, in the same trial.

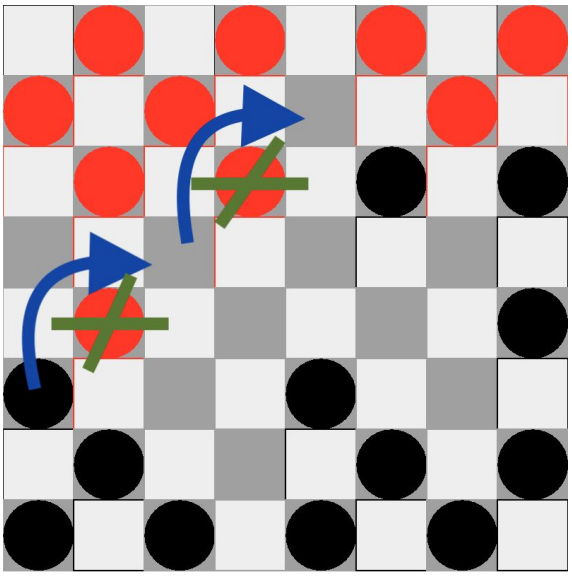

Fig.4: Black kills twice in same turn.

iv. A piece on reaching the opposite side i.e. the border of the opponent, becomes a King.

v. Every king has an additional power of moving backwards as well. So, it can move in four directions.

Moreover, a king may choose to go for double kill. In this, if two opponent pieces are lined up in same diagonal, a king may kill both by taking a loop of twice the size.

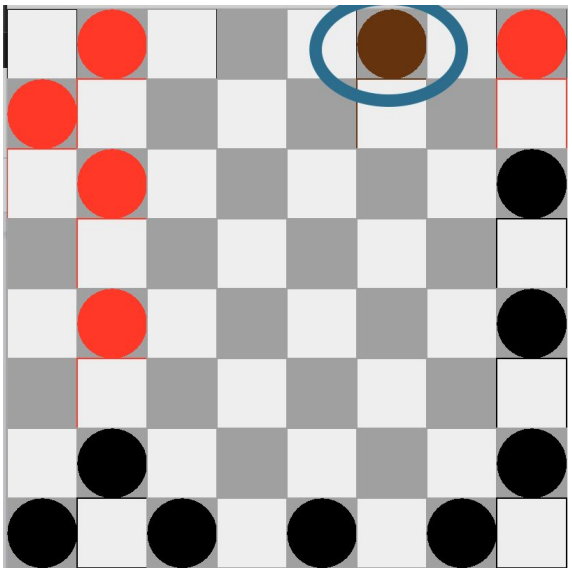

Fig.5: Black, on reaching the border of Red becomes a King, can now move four ways and is seen as brown to avoid confusion.

vi. Using the MIN-MAX algorithm, the player decides its one move and returns the configuration of the checkers board post its chosen action.

vii. Once a player is left with no move (either if its pieces are all blocked or eliminated), the game comes to an end with the other player being declared as the winner.

viii. Since we are implementing both the players to be computers, they both play optimally but even one mistake due to imperfect heuristic committed by a player determines its failure in the game.

We have implemented the checkers game in two modes –
i. Computer vs Computer
ii. Human vs Computer
iii. Human vs Human

The MIN-MAX algorithm combined with alpha-beta pruning is implemented for the computer while for humans the mouse is used for interaction.

(A) Computer vs Computer

Our heuristic considers the number of black pieces as positive and red as negative and returns the sum.

For black to win, we aim at maximising this value while for red we try to minimise this sum as much as possible.

Before we begin, let us start with the flow of implementation–

Step 1: Player 1 (black) starts the game and implements the best possible move for it.
In this, it tries to maximise the sum of the board (which considers black as positive unity and red as negative unity). So, it calls for the min function for every possible step it can take and opts for the checkerboard wherein this value is maximum. If it can kill an opponent piece, that is prioritised.

Step 2: Min function recursively calls for the Max function and the latter again calls the former until a given cutoff depth is reached. This is because to minimize red, in the next step it has to maximise itself and so on.

Step 3: Player 1, thus, implements the best move and returns the resultant matrix. Since turns alternate, Player 2 calls for its best possible move.

Step 4: In this, it tries to minimise the sum of the board and thus it calls for the Max function for every possible step it can take and opts for the checkerboard wherein this value is minimum. If it can kill an opponent piece, that is prioritised.

Step 5: Like Step 2, the Max function recursively calls for the Min function and the latter again calls the former until a given cutoff depth is reached. This is because to maximise value, in the next step it has to minimise and so on.

Step 6: Game comes to an end when either of the Players have no remaining moves.

The Min Max Algorithm functions are as follows –

Function: maxValue, inputs -> currBoard, cutoff, alpha, beta output->integer v
1: **if** cutoff > 5 **then**
2: return getScore(currBoard)
3: v<- Negative INF
4: **for** all possible actions when the cell is Red **do**
5: tempBoard <- board obtained after each action
6: tempScore <- minValue(tempBoard, curr+1)
7: if v < temp then
8: v <-temp, finalBoard <- tempBoard
9: **end for**
10: return v

Function: minValue, inputs -> currBoard, cutoff, alpha, beta output->integer v
1: **if** cutoff > 5 **then**
2: return getScore(currBoard)
3: v<- INF
4: **for** all possible actions when the cell is Red **do**
5: tempBoard <- board obtained after each action
6: tempScore <- maxValue(tempBoard, curr+1)
7: if v > temp then
8: v <-temp, finalBoard <- tempBoard
9: **end for**
10: return v

(B) Human vs Computer

This is the altered version of Computer vs Computer in which player 2 (red) is a human. The algorithm remains the same for player 1 who tries to maximise the heuristic value but for player 2, the human selects which red piece it wants to move and where. If the move is valid, the board is updated and player 1 plays and so on.

The Min Max algorithm remains the same as above for player 1.

(C) Human vs Human

This is the primitive version in which two humans play the game and if the move is legal, the board gets updated and the other player plays till it reaches the end.

No specific algorithm is applied here.

IV. RESULT AND CONCLUSION

The given Checkers game was implemented using JAVA swing for obtaining the GUI of the same. This is one of the cases wherein the game comes to an end with Red (Player 2) as the winner.

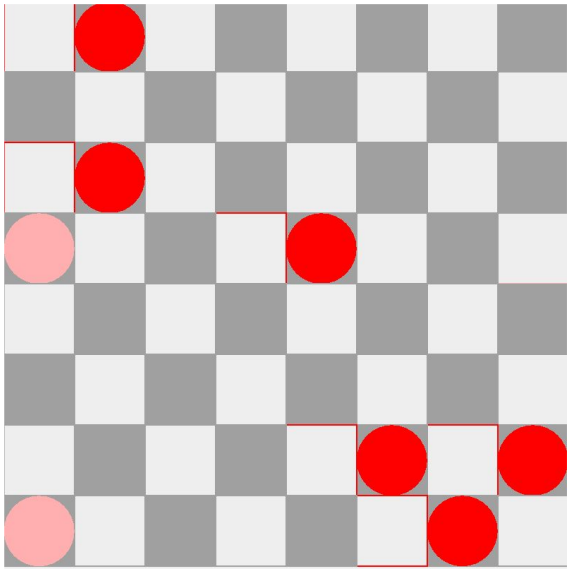

Fig.6: Red wins the game after eliminating every black piece on the checkers board

On changing the value of the cutoff, there may come a point in which both the players play such that the game doesn't proceed further. This is because both the players play optimally and do not wish to lose, and happens only when both have at least one king they can continuously oscillate. This is why we have added another calculation while determining the heuristic : more the red pieces near the middle of the board, more the negative heuristic and the same for black pieces (but positive).

V. REFERENCES

[1] Stuart Russel, Peter Norvig, "Artificial Intelligence, A Modern Approach", Third Edition

[2]https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-next-move-8f028bd0e7b1

[3]https://www.hackerearth.com/blog/developers/minimax-algorithm-alpha-beta-pruning/

[4]https://winning-moves.com/images/kingmerulesv2.pdf