Implementation of Automated Player For The Game Of Checkers

Abstract -- **we will implement the automated player for the game of checkers using AI algorithms and try to beat the human player and improve the performance of the automated player.**

I. INTRODUCTION

According to the wikipedia checkers can be described as following-

Draughts (British English) or checkers (American English) is a group of strategy board games for two players which involve diagonal moves of uniform game pieces and mandatory captures by jumping over opponent pieces. Draughts developed from alquerque. This game has many variations based on board size(Singaporean/Malaysian-12X12,Russian draughts-10X10,english draught -8x8) and rules(most variation on continuous kills,neccesity of killing option).
So here are the details of game which i implemeted

1.Board size- 8X8
2.Rules

- Any piece can move only diagonally.
- A piece can only move one squre
- If a piece reaches the furthest row from the player who is controlling the player,then the piece becomes king.
- A piece which is king can move backward and forward.
- A piece which is not king can move only forward.
- A piece can jump over the opponent piece if the opponent piece is diagonally neighbour of current piece and square next to the opponent neighbour is empty and all these three squares should be in straight line.
- If a player has no pieces left then he loses.
- If a player has no moves to play then loses.

Checkers is a very popular problem in AI.First computer program for playing was built by Arthur L. Samuel in 1952. Since then many optimizations have been done to this game and some of the implementation can beat the great players.

II.SOLUTION DESIGN

Before diving in the datails of the program I want to tell you the basic architecture of the program.

- Progaram has one 8X8 2d list of tuples (first item-color of piece(0,1,-1),   second item -king or not king(0,1)) which represent the 8X8 board.
- Program has a agent which takes the input(who is player) and output the optimal move for the player(utility of particular move till certain depth and old and new position of the piece)
- Program has an environment class which tells the validMove,Number of moves and Makechnage(to move the pieces).
- MakeChange takes 2 indexes old and new and changes the board accordingly.

GUI was implemented by using the canvas and input from the user was taken by the mouse clicks and handled by the event framework.
After every move of the human agent will be called and the computer will execute his move immediately.

III.AGENT

Agent wil take the input who is player and return the optimal move for the player.
Checkers is an adversarial search problem.So ,Agent is utilizing the minimax algorithms for the calculation of optimal move.
MINIMAX ALGORITHM- Minimax is used to calculate the optimal move for a player assuming that the other player is also playing optimal.
There are two players in this algorithm min and max ,min tries to minimize the value while max tries to maximize the value.
This algorithm tries all the possibilities using backtracking and returns the minimum or maximum value according to the player.
This is not very efficient because checkers have approximately 10^20 possible positions which is not feausible for our computer so we search only till limited depth.
AIPHA-BETA PRUNING-Since we are trying to optimize the value so we can cut the branches which will not be able to optimize the value.Alpha is the variable which is maximized by the max player and Beta is the player which is minimized by the min player.
So let's take a look at example-
Let's say we have alpha=10 and the player is min and a branch gives 8. We will simply cut all the branches of this level because we alread have alpha =10 and it will give a value less than 8.

HEURISTIC- Since we are not able to search all the states of the game so we have to decide which move (or branch) will be optimal in the future .
Here is the heuristic function which i have used

- We will be able to win the game if we kill the opponents piece.So we are giving the highest priority to the branches who will earn maximum effective kills(means the number of our pieces killed are less than the number of opponent pieces which are killed in the branch).we will give the utility 100 to the kill.

- When we become king, the number of moves are increased so we give the utility of 5.
- To avoid the draw ,we try to minimize the maximum minimum distance of our piece from opponent pieces.we will give extremely small utility value of 0.1 .In this case utility value value is(8-(max(minimum distance of piece from opponent piece)))*0.1
- When we lose the game we give the high penalty to the state so that it will not be chosen.utility of penalty is -10000
- When we win the games give the very good reward so that this branch will be chosen.utility of reward is 10000

All these utilities are for the max player .For min player utility values are just opposite to the max player.

IV.GUI

GUI is developed using the tkinter library.I have used the canvas for implementing the checkers board.To create the pieces i have used the circle object but berfore that i have drawn the rectangle on the canvas.I have created the text field in every circle object to specify the king when a piece becomes king.

I am handling the player turns by using the left click(button-1) event of the mouse (implemented using the event listener).

When a move is made we update the properties of objects(circle and text) according to the move.

1.if a piece become king highlight the piece

2.color the previous position of the piece with background color(function name-makeblank)

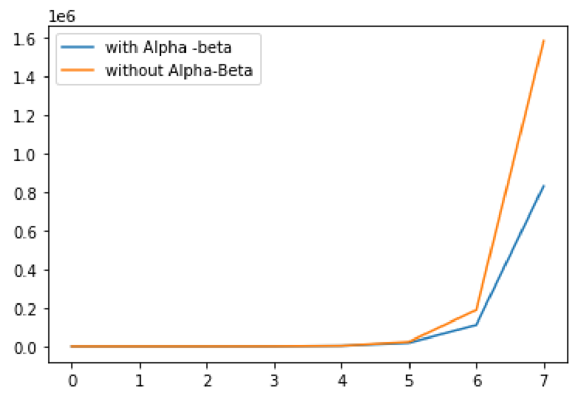
3.color the new position with appropriate color of piece.

When the result is found in the game then i pop up the result with an appropriate message.

When a human makes his moves then the computer makes his moves.

V.COMPLEXITY ANYALYSIS

Checkers have approximately 10^20 possible positions .Simple search was performing very bad then i have implemented the alpha-beta pruning which improved the overall complexity.Here is the comparative graph of growth -



As we can see the alpha-beta pruning improved the performance but it's still exponential so still you have to limit your search to a certain depth.

VI.CODE EXPLATIONATION

Here i will give the some information about code-
Variables-
Mat- 8x8 matrix(board) of tuples(color and king status)
Utility2-its a dictionary which contains the utilities.
maxDep- is the depth till you want to computer look while calculating optimal move
draw -this is the variable which stores the number of moves without kill if it's greater than 100 then game draw.
leftP -strores the ramining pieces of each player.

Function(class.functionName)-
agent.Efmove- it takes the input player(min or max) ,alpha ,beta ,depth(which is itially zero)
and returns the optimal move and utility of that move.
Whowins-does not take any input and returns the winning player(0,1) or match is still going on(-1).
circle.clicked1() -takes an circle object adn event and does the moves depending on event.
Others are just helper functions doing the small tasks.

VII.HOW TO PLAY

To play my implementation of the game just click(left click) on the piece which you want to move and you will see the possible moves(highlighted ring of light green color) and click on the one of the rings to move your piece to that position now your move is executed wait for the computer move(computer moves very fast ,so i have added a time lap of 3 second between human and computer move but effectiveness of timer depends on your computer's processor because sometimes it may take time to update the canvas and till then its possible that agent waiting time completed and just after your move and computer move will move).
If the clicked piece has no moves then no squares will be highlighted in that case you have to click on the other piece.
Once the result of the game is found you will see a nice pop -up stating the result.
New Game Button will start a new game

VIII.CONCLUSION

Very first implemnetation was for the computer vs computer and when I ran that program with the same depth (in searching) then most of the match was tied When I ran with the different depth player which had higher depth perfomed well.
Then I implemented the human vs computer version.
I have played many times with this automated player and most of the time it seems to perform well compared to me.