Game Playing Assignment

Hrutvik Nagrale ;- IIT2018088

V Semester B.Tech, Department of Information Technology,

Indian Institute of Information Technology Allahabad, India

ABSTRACT

In this paper i have discussed the the checkers game using alpha- beta pruning and with it minimax search With the above puring and with that of developing heuristics.Also codes are provided into the formate of .java.MinMax is the an the standard technique to solving the problem.

INTRODUCTION

In this game of checkers mankind has been humbled by the machine However these programs are there can be lost as a game.Checker are dramatically with 8 x 8 the rule are simple yet can checker has high decision complexity complex than 9 x 9 Go and play of the bridge hands. On par with backgammon but less complex than class .

I have created the heuristic fusion and them implement them, and perform the experiment exploring their performance

The correction and i played with my own program and got defeated most of the game .of despite the game played the we got an awesome result. Yet on computing the required space for allocation of the memory are quite the increating the given apace.

Using the good heuristic will go a long way depth the last auction of the wrong way.

I have used a rather simple heuristic yet AI played with the GUI that has being played . with the mention of the of the pseudocode of the MInMax algorithm and added the heuristic function

ALGORITHM OVERVIEW

A formal proof of a game used in the MinMax in the technique is the standard way to solve the problem.For the large and the complex checkers problem with the heuristically guided . the etree manager maintain the composter of proof of the process . Given the position to the search this component of the program.It is backed by alpha-beta algorithm .

Pseudocode of the algorithm:

Pseudocode of the algorithm:

1. Max

```
State solveForMax (given,depth,currPlayer):

   if (depth == 0)
    this.beta = Heuristic(given.state,max_player)
        return null


    nextStates = given.getNextStates(currPlayer)

    nextBestState = null

    for each state in nextStates{
        // BETA PRUNING
      if(this.beta >= this.getParentAlpha())
            break

      MIN.solveForMin(state, depth-1, nextPlayer)


      if(MIN.alpha > this.beta)
            nextBestState = state
            this.beta = MIN.alpha
    }

   return nextBestState
```

2. Min

```
State solveForMin (given,depth,currPlayer):

  if (depth == 0)
   this.alpha = Heuristic(given.state,max_player)
        return null


    nextStates = given.getNextStates(currPlayer)
    nextBestState = null


    for each state in nextStates{
        // ALPHA PRUNING
        if(this.alpha <= this.getParentBeta())
            break

        MAX.solveForMax(state,depth-1,nextPlayer)

        if(MAX.beta < this.alpha)
            nextBestState = state
            this.alpha = MAX.beta
    }

  return nextBestState
```

DESCRIPTION OF HEURISTIC

The key factor for the safe result of the algorithm is Model is divided into using the heuristic algorithm The human player can manipulate the game even if the AI for the few squares .But since analyzing the entire game depth and evaluate how of devoluting the game in the given model of game.

Heuristic -1

1) The format for a state is as :

It is a list of lists. Each inner list represents a row of the
checkers board. Individual elements on the checker board are denoted by characters: "." (no piece), "b" (a black checker), "r" (a red
checker), "B" (a kinged black piece) and "R" (a kinged red piece).

7 : . r . r . r . r
6 : r . r . r . r .
5 : . r . r . r . r
4 : . . . . . .
3 : . . . . . .
2 : b . b . b . b .
1 : . b . b . b . b
0 : b . b . b . b .
0 1 2 3 4 5 6 7

Heuristic -2

It is a list of tuples, where each tuple represents a position (x,y)
counting from 0,0 in the lower left corner. The first tuple in the
list represents the initial position of the checker; the ith tuple
represents the ith move in the hop; and the last tuple represents the
final position of the checker. For example, if a move involves two
hops, the path might look like this:

[(1,1), (3,3), (5,5)]

Heuristic -3

The result based on the last id
Heuristic = (my total point) - (Enemy's total points)
This is actual heuristic that i have used in the model.

A very simple (and bad) heuristic function is already implemented in Checker.java
Definition:
def evalFun(node,space,player):

node: node.state.board: contains the current checker **board** -- refer to 1) above
node.parent: contains the parent state -- node.parent.state.board contains the previous checker **board**
node.gval: path cost from the initial state to itself (note that all edgecost = 1)
node.hval: (heuristic val) for your bookkeeping; not used in the base code.
space: used only for output
player: 'r', or 'b'

CONCLUSION

The alpha-beta pruning significantly improves the performance MinMax algorithm.

Connect Four:

- Minimax algorithm + alpha-beta pruning
- Heuristic that values moves which lead to more future 3-in-a-row and 4-in-a-row scenarios to optimize AI

Checkers:

- Minimax algorithm + alpha-beta pruning
- Heuristic that accounts for the value of each piece and its position on the board to optimize AI

REFERENCES

[1] **IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005**

[2]
https://www.researchgate.net/publication/281656081_Heuristic_Search