Ain Shams University
Faculty of Engineering
Computer and Systems Engineering

CSE481: Artificial Intelligence

Project Name: Intelligent Mancala Game

June,2021

Contents

Team Members

Name	B.N
عبد الحميد خالد محمد احمد	1600724
عبد الله عمر ومحمد بدران	1600769
محمد رمضان سيد محمد	1601181
علي محمد علي علم الدين	1600841
جرجس وجيه ثابت	1600447

Game description

- The project was implemented using python and some external modules such as sys to handle recursion, time to calculate execution time and numpy .

- The game design is consisting of 3 classes and the main function. First class is the Mancala game which is responsible for the game rules, movement, show the board and static evaluation. Second class is Player class which is responsible for choosing the move for each player. Players have different types such as human and AI depending on the technique of searching. Third class is Game Handler which is basically take the players and the game as parameters to start and handle the game. Main function to initialize and define classes and to handle the sequence and the starting player.

- The Mancala board has two sides each side has 6 holes and a Mancala store to store the stones the human player always on the right side and the Ai on the left side of the board . any of the two can start the game

Detailed Description of Implementation

- ## Mancala Game Class:

Functions:

Initialization function() ➔ which takes initial state , mode and the level of the game .
Player_move() ➔ which take the hole number and the player as parameters and return if the player will play again or the other player has the turn. This decision is made by the game rules and stealing mode if stealing and last stone of player placed in his mancala then will play again otherwise the other player plays. this is done by initializing a round robin variable to detect the last stone position.

Is terminal() ➔ this function determine if the node is leaf node by calculating the total stones in each side of the mancala board and if one of them is empty then it puts the remaining stones in the mancala store of this player's side .

Who_won ()➔ to determine who won if the game is over by subtracting number of stones in each mancala store .

Static_evaluation() ➔ static evaluation for non-terminal node is determined by subtracting the mancala store stones the AI player is +ve and the human player is -ve . for Terminal node evaluations is determined by 4 scoring systems

```
def static_eval(self):
    if self.level == 2:
        if self.isterminal():
            if self.board[13] > self.board[6]:
                return 100
            elif self.board[13] == self.board[6]:
                return 0
            else:
                return -100
        else:
            score1 = self.another_turn_opportunities() * 0.1
            score2 = (self.board[13] - self.board[6]) * .45
            score3 = self.stealing_opportunities_for_opponent() * (-0.1)
            score4 = self.stealing_opportunities() * .05
            return score1+score2+score3+score4
    elif self.level == 1:
        if self.isterminal():
            if self.board[13] > self.board[6]:
                return 100
            elif self.board[13] == self.board[6]:
                return 0
            else:
                return -100
        else:
            score2 = (self.board[13] - self.board[6]) * .45
            return score2
    else:
        pass
```

1st is the difference between the stones in each mancala then times that with 0.45 .

2nd is check is Ai can play again and times that with 0.1

```
def another_turn_opportunities(self):
    count=0.0
    for i,a in enumerate(self.board[7:13]):
        if (7+i)+a == 13 or (a%13)+(i+7)-13 ==1 :
            count +=1
    return float(count)

def stealing_opportunities(self):
    count = 0
    if not self.stealing:
        return 0.0
    else:
        emptyholes=0
        list_of_empty=[]
        for i, a in enumerate(self.board[7:13],7):
            if a == 0:
                emptyholes +=1
                list_of_empty.append(int(i+a))
        if emptyholes == 0:
            return 0
        else:
            for i, a in enumerate(self.board[7:13],7):
                if  a +i in list_of_empty and  not a ==0:
                    count+=1
            return count
```

3rd is calculating stealing chances and times that with .05
4th is the chances for the opponent to play again and times that with (-0.1)

```
def stealing_opportunities_for_opponent(self):
    count = 0
    if not self.stealing:
        return 0.0
    else:
        emptyholes = 0
        list_of_empty = []
        for i, a in enumerate(self.board[0:6]):
            if a == 0:
                emptyholes += 1
                list_of_empty.append(int(i + a))
        if emptyholes == 0:
            return 0
        else:
            for i, a in enumerate(self.board[0:6]):
                if a + i in list_of_empty and not a == 0:
                    count += 1
            return count
```

Print_board()➔ to print the game board each time player plays

Validate _move()➔ to check the human's play is correct

- Player class:

  Player is abstract class has three functions __init__() and make_move() and turn()

Functions:

Initialization()➔ which takes player turn as parameter

Make_move()➔ for a human player he freely can choose any hole . for the AI player it use minmax with alpha beta pruning to choose the best move after searching the tree with depth first search algorithm the minmax consist of two parts one to maximize the Ai play to get higher score which means higher stones second part is to minimize the other

player expected move and repeat this process until reaching terminal node or the max depth .

Turn()➔ to print which player's turn

- Game handler

  Functions:

  Start_game()➔ inside the function there are two while loops to switch between each player turn and handle if player will play again . after each play the game board is displayed indication the AI move and the time consumed during the process of determine the best move .between each player turn there is 0.5 sec

- Main

  The player is asked for the mode , first player and the difficulty level

GitHub repo

Work distribution

Name	Contribution
عبد الحميد خالد محمد احمد	Implementing the main function and static evaluation function and its utilities Difficulty levels and verbose mode
عبد الله عمر ومحمد بدران	Implementing the handle_Game class
محمد رمضان سيد محمد	Implementing the print_board() and the player abstract classs
علي محمد علي علم الدين	Implementing some of the Game functions such as player_move() isterminal() possible_moves()
جرجس وجيه ثابت	Implementing the human player and ai

References

http://www.cs.toronto.edu/~hojjat/384f06/Lectures/Lecture06-4up.pdf
https://www.thesprucecrafts.com/how-to-play-mancala-409424
https://github.com/tmhdgsn/mancmancala