Report of Mancala

Team Member: Sining Qu 31428972/squ8
Yuxin Liu 31425544/yliu198

Program Design

Firstly, we decided to divide the whole program into three modules -MancalaBoard, Play, MancalaGUI.

In module MancalaBoard, we displayed the rules of Mancala with the class MancalaBoard. And the class contains various variables and functions. We define the *reset* as the new game, each player has 6 pits and 4 stones in each pit, 0 stone in their stores to start with. The *legalMove* requires the players' move should comply with the rules of Mancala. *Makemove* and *makeMoveHelp* is the rule that how the number of stones change in the players' pits and stores, also determines whether the players get another turn to move and whether a capture of stones is happening. The function of *Gamover and hasWon* helps determine if the game is over and if a player has won. *hostGame* hosts the games between two players with the functions mentioned above.

In module Play, we define a class of Play containing different types of player - Random, Human, Minimax and AlphaBeta Prune(ABPrune). The function of *minimaxplays* defines how minimax algorithm plays showing its move. *Minval* calculates the minimax value for the next move at a state, also define an opponent who plays max value. Abpruneplay defines how the minimax with alphabeta pruning plays showing its move and scores. It will test the move on the copied board to see how the opponent would play and determine if it is a better move with a higher score and save the move. *Maxabprune* and *Minabprune* are finding the ABpruning value for the next move at a state and check if the game is over. *Selectmove* determines how to take the next move and if one player is human, let the player input the move. We defined our evaluation function with a score that consists of 2 parts, each counting for 50% of the total score. One is how many stones a player has in the store as a percentage of the total number of stones in both stores. The other is how many stones a player has on its own side as a percentage of the total number of stones on the board. Minimax and ABPrune will make moves based on the score.

The code of Module MancalaGUI is borrowed from Github, we modified some of the functions and variables to be consistent.This module helps us display the Game of Mancala better in a separate window. The resource of the code stated at the end of this report.

Member Contribution

Sining Qu was in charge of designing the Play module, adjusting the GUI, carrying out the experiments, writing the readme file and writing the experiments and discussion parts of the report.

Yuxin Liu was in charge of designing the Board module, coordinating the whole program/algorithm, writing the program design and discussion parts of the report.

Some tasks were shared and interchanged because the modules are correlated with each other.

Experiment 1

During this experiment, we first let random played against minimax player to see the effectiveness of the implementation of minimax. Out of a total of 30 runs, when setting random as player 1 and minimax as player 2, random and minimax each won 15 times. To control for the order of player, another 30 runs were done with minimax as player 1 and random as player 2. This time minimax won 17 times and random won 12 times with a draw. The overall performance of minimax is slightly better than random.

When letting minimax play against alphabeta pruning, no matter which player (minimax or alphabeta) played first, and how many times we played, player 2 has always won. The ending numbers of stones in each store were also exactly the same for each run, which was the result of the algorithm.

Experiment 2

The following is a table of 5 runs of Minimax and ABPrune played against ransom, where random is player 1 and Minimax/ABPrune is player 2. We can see that ABPrune has a slight edge on timing over minimax as expected. However, we need to take note that time used

here is very short in general so it may be affected by randomness. For example, it depends on how the random player played and how many turns it took to end the game.

Time used/s	Minimax	ABPrune
Trail 1	0.472829	0.234162
Trail 2	0.371528	0.609883
Trail 3	0.685483	0.55139
Trail 4	0.750411	0.672543
Trail 5	0.548017	0.466967
Average time	0.5656536	0.506989

Experiment 3

To investigate whether player 1 has an advantage over player 2 because it plays first, we ran random against random for 50 times, player 1 has won 15 times, player 2 has won 34 times and they drew once. This situation was further enhanced by letting minimax played against minimax, where player 2 has always won with the exact same resulting stores every time.

Therefore with our algorithm, the player who played second/last actually had the advantage. This is consistent with the result when we played minimax against alphabeta.

Discussion

From the project of Mancala, we found us had a deeper understanding of AI algorithms like Minimax and ABPrune, and learned how to apply them into real applications. The process of designing the game was really impressive, we met difficulties how to achieve some of the functions of Mancala, we made lots of research and finally made it.

During the experiments, one of the team members found it interesting that when random player is player 1 and minimax is player 2, random player seemed to have a much higher win rate, which is very different from our assumption. And it happened multiple times throughout our experiment at different times. However, another team member did not

encounter this issue. We spent lots of time checking the code, but it seems nothing went wrong. We would do further research on this situation.

Furthermore, We had a hard time determining the evaluation function. After some research, we decided to use the two most obvious measures- percent of stones in a player's store and on the player's side of the board. It made the most sense to us and was easy to implement.

Another interesting thing we noticed is that when investigating the first-choice bias, we were expecting the player who played first to win more, however, the result showed the opposite.

Reference:

https://github.com/sameersrivastava/mancala-player/blob/master/python/MancalaGUI.py