AU 332 Artificial Intelligence: Principles and Techniques

Assignment 2 Due OCT 8th 11 : 59pm

> *Adhere to the Code of Academic Integrity.* You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section **??** of this document.

1 Minimax alpha-beta pruning

This homework consists of designing and implementing a program that plays Chinese Checker. It will exemplify the minimax algorithm, and alpha-beta pruning, and the use of heuristic (evaluation/static) functions to prune the adversarial search. Usage of any other existing AI techniques (e.g. machine learning, search strategies) and/or techniques that you learn or develop for this homework along with the Minimax algorithm will earn extra credit and potentially improve performance in the tournament.

1.1 Game Description

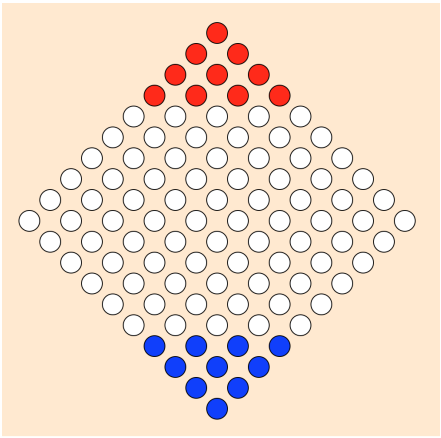

Figure 1: Two player Chinese Checker board

Chinese checkers is a perfect information game for 2 players. A Chinese Checkers board is shown in Figure **??**. The goal of the game is to get 10 pegs or marbles from one's starting position to one's ending position as quickly as possible. Starting and ending positions are always directly across from each other on the board, and players are placed as symmetrically as possible around the board. In a two-player game, the players would start at the top and bottom of the board. The goal of the game is moving all marbles of one color from starting point to the star point on the opposite side of the board.

1.2 Action

Pegs are moved by stepping to an adjacent position on the board or by jumping over adjacent pegs. One can jump over any player's pegs, or chain together several jumps, but pegs are not removed from the board after a jump. We demonstrate this with a set of consecutive moves in Figure **??**. The first move is a simple jump over a single piece. The second move is a double jump. The third move is a simple move to an adjacent

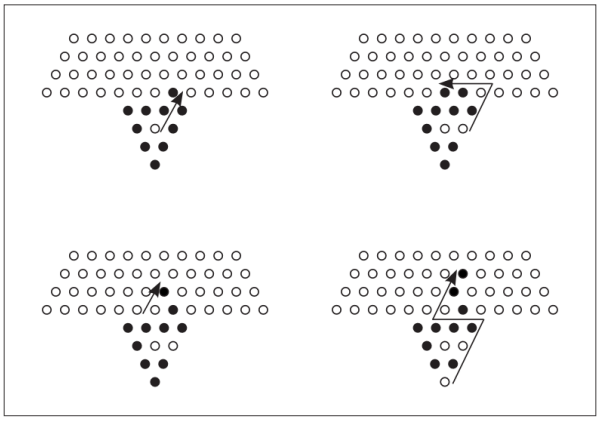

Figure 2: Move possiblities for Chinese Checker

space, and the last move involves 4 hops. Players can chain together as many jumps as they wish over both their own pieces and their opponents pieces, as long as every segment of the jump is from an empty space to another empty space over a single peg. There are no forced moves in Chinese Checkers.

1.3 Heuristics

Some possible heuristics you can choose from are:

(i) Random: the computer makes a move randomly without taking into consideration the current board configuration

(ii) Vertical Displacement generates all moves with minimax algorithm & sums up the vertical distance for its pieces and adds them. Same done for the opponent.

(iii) Vertical/Horizontal Displacement considers the horizontal pieces as well in deciding the next move. It is best to play in the middle of the board.

(iv) Vertical/Horizontal Displacement during the end of the game by considering moves of the pieces to the edges of the destination triangle once they have moved in. This is for creating space for others to come in.

1.4 Special Rules

In this homework, you are required to design an anytime algorithm. That means the program will give one second for each player to generate a move $timeout(agent.getAction, state)$. By the end of the one second, if your program did not assign a valid move to the *agent* action, you will lose the game immediately.
In order to prevent the situation that one of the players may not leave its triangle and prevent the opponent to occupy its space, after 100 turns, if any of the marbles of one player are still in its own triangle, the player loses its game immediately.

1.5 To Do

(i) This homework can be done in a group of two or one.

(ii) In order to play Chinese Checker, run *pythonrunGame.py*, we provided a random agent and a greedy agent.

(iii) Write your code in *agent.py* and only submit this python file. When you write your *getAction*() function, make sure you update $self.action$ for every iteration. When time is up, the program will retrieve this value only as the return value.

(iv) We provided a Windows version and a Linux version, we strongly recommend you use the Linux version to test your code. The major difference between these two versions is in Windows version, the time limit of 1 second is not set for each turn. If you are using Windows to test your code, make sure you check system time and how much time you are using. If you are using up the 1 second, please set a new value for $self.action$.

(v) Create a name for your agent. $class\ \ TeamNameMinimaxAgent(Agent)$ should be changed to your own team name.

(vi) Write a report of how you designed your alpha-beta pruning algorithm and utility function for the Chinese Checker game. Explain how you choose the parameters in your code.

2 Extra credit

For the tournament, the top 10% of the team will given extra credit based on their creativity and final ranking.

3 Submission instructions

1. Zip `agent.py` and `HW2.pdf` to a folder called $HW2\_name1\_name2.zip$
2. Send the zip file to TA 121103451@$qq.com$