AU 332 Artificial Intelligence: Principles and Techniques

By: Xiaocheng Wang(517030910326)

HW#: 2

October 31, 2019

I. INTRODUCTION

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.

II. IMPLEMENTATION

1. Utility function desigh. In this part, I will explain my choice of heustic functions and how to evaluate a state.

   I mainly use three strategies to judge a state: vertical displacement, horizontal displacement and ending state.

   (a) Vertical displacement is quite intuitive, similar to given example using simple greedy method. Since our goal is reaching the opposite corner on the board in vertical direction, a larger vertical displacement should leads to a larger step to victory.

   (b) Horizontal displacement. According to the rule of chinese checker, we notice that proper sparsity of marbles is important. If too sparse, a marble has little chance to hop and thus approaching destination slowly; if too crowded, at least it's better than when too sparse. Therefore, we want the marbles mainly choose the center area of the board when going across the board. In this consideration, we take the horizontal displacement into consideration.

   (c) Ending state. This is easy to find, just use the sum of vertical position to represent whether some player has won the game. In (10,4) board, the player starting from the bottom has ending vertical sum 30, and the other player has 170.

2. Alpha-beta Pruning Algorithm During implementing alpha-beta pruning as described on book, I encountered a big problem: time limitation! In I want to search all possible actions in a moment, the search space is huge and unpredictable. Also, it's impossible to search as deep as we want within finite time. Therefore, I set a seemingly reasonable upper bound of actions to search each time and maximum depth to search.

   However, if only limited number of actions are searched, how to guarantee these actions "quality"? Using a priority queue to select actions owning better heuristic function makes sense, but computing a complex heuristic function needs computing the state after finishing action first, which is expensive. So I choose to judge the quality of an action by it's vertical displacement, as done in greedy strategy.

   Besides, by doing some tests, I notice that a better choice of depth setting is, at begining, a small number, say 1. After one iteration, if there's time left, we add 1 to depth and do search again. In the previous iteration, we also replace the initially simple heuristic function of actions with complex version, which will improve the choice's performance in the meantime.

III. DISCUSSION & CONCLUSION

In the competition with the simple greedy agent, I notice that the minimax algorithm cannot always win. By discussing with teammates, I redesign the heuristic function especially the horizontal displacement, which is to lower the score of the middle line, as shown in figure 1. Smaller number means better.

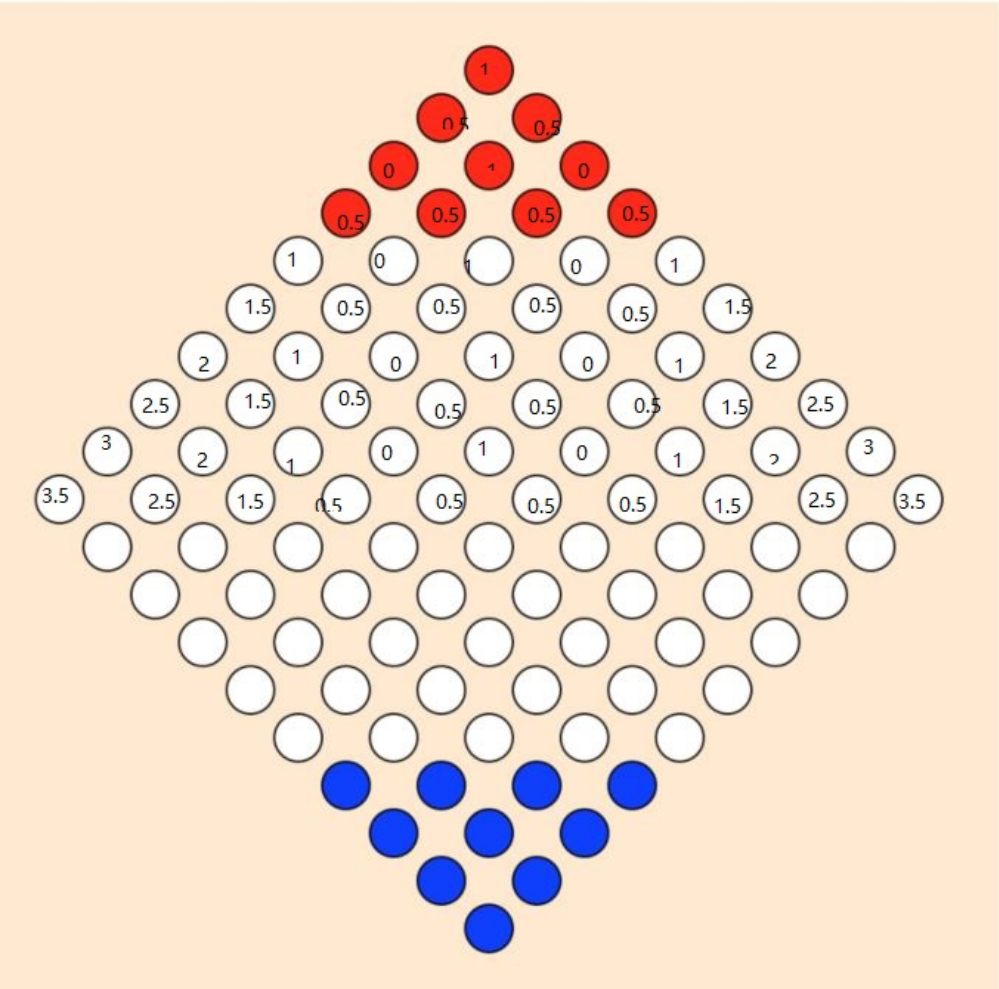

FIG. 1: Horizontal displacement

Coding minimax and alpha-beta pruning algorithm deepens my understanding, and designing such a agent to play chinese checker is very interesting.