Abalone - P-Proposal

Connor Settle

For my CS 221 final project, I want to build a game playing agent for a generalized version of the popular strategy game Abalone. This game normally operates on a hex grid with an edge length of 5, containing 14 white balls and 14 black balls placed in rows on either side of the board. The game is won through "pushing" the opposing team's balls off the edge of the board. My game playing agent will be designed to handle a board of any size, with any starting configuration of balls. This will allow me to build and train the agent with a smaller board, and explore the different consequences of slight rule changes, while laying the foundations to scale up to the full board or beyond.

On a more detailed level, the agent will take as input a numerical representation of the current state of the board, and will output a single valid action that will be used as the agent's "move". The agent will either simulate play against a predefined sequence of moves or a predefined policy that will act as the agent's opponent, or will take input from a human player and attempt to defeat them. The agent will have succeeded in its task if it's able to beat a competent human player at least 50% of the time.

To elaborate further on the input and output the algorithm will use, this is an example of an input state (on a grid of edge width 3 rather than 5):

board = [**Actual board view:**
[None,-1,-1] **B B**
[0,0,0] **O O O**
[1,1,None] **W W**
]

In this example, black balls are -1, white balls are 1, and empty slots on the board are 0. Since this is a hex grid, there will need to be corners of the array representation that are None. These are out of bounds and are not considered part of the state.
An example of an output to this state (assuming that the game playing agent is manipulating the white balls) is:
format = ((row, col), movement direction, direction of attached balls, number of balls attached)
action = ((2, 1), 1, 2, 2)
This would move the two white balls at the bottom of the grid upward toward the middle, securing a safer position for the agent.

Move overview:

B B **B B**

O O O => O W W
W W O O

I've crafted a simple baseline implementation for this algorithm that simply tries to move all of the agent's balls generally away from the edges, and perform knockouts when the opportunity arises.
An oracle for this algorithm would take as input a game state, and would output the move that makes victory most likely. This makes the most accurate implementation of this oracle human input, as reaching a perfect oracle is computationally difficult.
There is a pretty fair gap between these two implementations, as the baseline is very simplistic and is highly vulnerable to multi-turn, thought out displacement attacks, which are core to the game. The oracle, however, is difficult to reach, since the state space is far too large to completely analyze.

The main challenges with this gap are:

- The only immediate measure of progress that the game provides is a knockout, which happens quite rarely
- Due to the combinations of balls that make up the game's possible moves, the game has a ridiculously large state space

Good potential solutions for these issues are:

- Heuristics: this provides an immediate measure of progress for each move
- Alpha-Beta Pruning: reduce state space
- Minimax: with heuristics, can be used to estimate most likely opponent move, which helps to choose a winning strategy
- TD Learning: good on-policy algorithm that can provide a path forward even without knowing whole state space
- Depth-Limited Search: again, helps to reduce state space

I've found an interesting research paper from Cornell University, linked here, in which students crafted a game playing agent that could play Abalone under normal circumstances with fair accuracy. [1] This paper could provide me with good ideas for heuristics and other algorithm-related ideas to help me generalize my agent to be able to effectively play any version of Abalone.

To this end, I've written an Abalone simulator in Python that can simulate a game of any size, read in board, as well as moves and actions from external files, and execute valid moves under varied rulesets. This simulator will be a critical tool during my development of my game agent.

References:

[1] Lee, Benson, and Hyun Joo Noh. “Abalone - Final Project Report.” www.cs.cornell.edu/~hn57/pdf/AbaloneFinalReport.pdf.