Build an Adversarial Game Playing Agent

Mohammad Al-Fetyani

April 2020

Synopsis

In this project, we will experiment with adversarial search techniques by building an agent to play knights Isolation. Unlike the examples in lecture where the players control tokens that move like chess queens, this version of Isolation gives each agent control over a single token that moves in L-shaped movements–like a knight in chess.

Isolation

In the game Isolation, two players each control their own single token and alternate taking turns moving the token from one cell to another on a rectangular grid. Whenever a token occupies a cell, that cell becomes blocked for the remainder of the game. An open cell available for a token to move into is called a "liberty". The first player with no remaining liberties for their token loses the game, and their opponent is declared the winner.

In knights Isolation, tokens can move to any open cell that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. On a blank board, this means that tokens have at most eight liberties surrounding their current location. Token movement is blocked at the edges of the board (the board does not wrap around the edges), however, tokens can "jump" blocked or occupied spaces (just like a knight in chess).

Finally, agents have a fixed time limit (150 milliseconds by default) to search for the best move and respond. The search will be automatically cut off after the time limit expires, and the active agent will forfeit the game if it has not chosen a move.

Results

I implemented a custom aggressive heuristics that looks as follows:

$$\#my\_moves - 3 \times \#opp\_moves + \#blank\_spaces/2 \tag{1}$$

I thought to try reducing the number of possible moves available for the opponent as well as solving it as fast as possible. I divided the number of blank spaces by 2 to reduce its effect a bit. The results I got after running 100 fair games are illustrated in Table 1.

Agent	Baseline (%)	Custom (%)
Random	93.2	97.0
Greedy	66.5	76.0
Minimax	50.0	51.5

Table 1: Win rate of multiple agents against my agent.

(Q1) What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?
ANS: I incorporated the number of available blank spaces into my heuristic. I think this matters because the agent moves to a black space so it might help to maximize the number of blank spaces. Also, I think it is better to win with the least possible number of moves that's why I made it aggressive.

(Q2) Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?
ANS: I chose it to be a depth of 3 which is good but it can be extended up to 7 before it gets slow. The search speed matter in this situation as no one is going to wait for you to play longer than 4 seconds. However, it also good to have a reliable agent that wins most of the time. This a trade off between time and accuracy and one should focus on either of them wisely without neglecting the other depending on the problem in hand.