AIND - Isolation Project - Heuristic Analysis

by Marco Zorzi

In this project, I develop an adversarial search agent to play the game "Isolation". Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent is declared the winner.

This project uses a version of Isolation where each agent is restricted to L-shaped movements (like a knight in chess) on a rectangular grid (like a chess or checkerboard). The agents can move to any open cell on the board that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. Movements are blocked at the edges of the board (the board does not wrap around), however, the player can "jump" blocked or occupied spaces (just like a knight in chess).

Additionally, agents will have a fixed time limit each turn to search for the best move and respond. If the time limit expires during a player's turn, that player forfeits the match, and the opponent wins.

I implement 3 position evaluation heuristics:

- **custom_score() : heuristic that only considers player's moves**
  alpha * len(my available moves)^2
  alpha>0 and in this case alpha = 1.7 (empirical value)

- **custom_score_2(): heuristic that balancing maximizing player's moves and minimizing opponent's moves**
  alpha * len(my available moves)^2 - len(opponent moves)^2
  alpha>0 and in this case alpha = 1.7 (empirical value)

- **custom_score_3(): heuristic that tries to keep the center or the corners**
  alpha * central(game, game.get_player_location(player)) * (my_moves - opponent_moves)
  alpha>0 and in this case alpha = 1.7 (empirical value)

With tournament.py script I evaluate the effectiveness of my custom heuristics. The script measures relative performance of my agent in a round-robin tournament against several other pre-defined agents. The Student agent uses time-limited Iterative Deepening along with my custom heuristics.

The performance of time-limited iterative deepening search is hardware dependent (faster hardware is expected to search deeper than slower hardware in the same amount of time). The script controls for these effects by also measuring the baseline performance of an agent called "ID_Improved" that uses Iterative Deepening and the improved_score heuristic defined in sample_players.py. My goal is to develop a heuristic such that Student outperforms ID_Improved.

The tournament opponents are listed below:

- Random: An agent that randomly chooses a move each turn.
- MM_Open: MinimaxPlayer agent using the open_move_score heuristic with search depth 3

- MM_Center: MinimaxPlayer agent using the center_score heuristic with search depth 3
- MM_Improved: MinimaxPlayer agent using the improved_score heuristic with search depth 3
- AB_Open: AlphaBetaPlayer using iterative deepening alpha-beta search and the open_move_score heuristic
- AB_Center: AlphaBetaPlayer using iterative deepening alpha-beta search and the center_score heuristic
- AB_Improved: AlphaBetaPlayer using iterative deepening alpha-beta search and the improved_score heuristic

Results:

All my costum heristics outperforms ID_improved and "custom_score_2(): balancing maximizing player's moves and minimizing opponent's moves" is the best one with a 64.7 % rate of winning games.

This is the tournament result:

```
                    **************************
                         Playing Matches
                    **************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      87  |  13    92  |   8    91  |   9    90  |  10
   2       MM_Open     52  |  48    54  |  46    65  |  35    63  |  37
   3      MM_Center    69  |  31    79  |  21    81  |  19    81  |  19
   4     MM_Improved   50  |  50    48  |  52    55  |  45    53  |  47
   5       AB_Open     45  |  55    48  |  52    54  |  46    52  |  48
   6      AB_Center    52  |  48    44  |  56    52  |  48    55  |  45
   7     AB_Improved   50  |  50    55  |  45    55  |  45    51  |  49
--------------------------------------------------------------------------
          Win Rate:      57.9%        60.0%        64.7%        63.6%
```

I consider custom_score_2 the best heuristic because it has got a greater win rate than the others and without much complexity. This heuristic predicts the final outcome of the game looking at my available move.