Build a Game-Playing Agent

Heuristic Analysis

By Nithin Devanand

Synopsis

Building a game playing agent project, aims at developing an adversarial search agent to play the game of 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 have defined three heuristics which seem to better describe a way to find solution to the problem described above.

Heuristic1: Center Score to Legal Move Heuristics

(centerscore_to_legalmove_heuristic)

In the heuristics, I have considered the fact that player occupying center position or location nearer to the center position has more open locations or legal locations to move to. As the player moves towards the edges of the board, their moves are limited for one of their axes (if they are at bottom edge, they cannot move further down).

Player at bottom edge – No movement further down

		X				

This will get more limited if the player moves to corner location.

Player at the corner.

X						

Here we calculate the ratio of center score for player to opponent. We also calculate the ratio of legal moves for player the opponent. The product of these two values determines the heuristic score.

Heuristic2: Open to blank spaces ratio

This heuristic is build on the idea that when the board opens for game there will be more blank spaces and so the player gets a chance to move freely. In other word there will be more legal open location for player to move to. As the game progresses, the blank/empty locations decrease. This will reduce a player's options to move to a new location.

Here I create a score which is the difference in ratio of the open to blank spaces between the player to choose who is better chance to move around.

High level computation details

blank_spaces = number of open blank/empty spaces

own_moves = the number of legal moves

opp_moves = the number of legal moves for opponent

own_ratio_open_to_blankspaces = own_moves/blank_spaces

opp_ratio_open_to_blankspaces = opp_moves/blank_spaces

score = own_ratio_open_to_blankspaces - opp_ratio_open_to_blankspaces

Heuristic3: Distance between players to legal moves ratio

This heuristic is build on the idea that as the players get near to each other they have less place to move into in next turn. As players are further apart they have more open locations to move into..

The ratio of legal move is a good indicator of which of the player has more options to move. In this heuristic, I combined the distance factor and legal moves ratio to arrive at a score to determine which player has a better chance of winning.

High level detail

own_moves = number of legal moves

opp_moves = number of legal moves for the opponent

p1x, p1y = Player location

p2x, p2y = Opponent location

distance_btw_players = |(p2y-p1y)|+ |(p2x-p1x)|

Score = distance_btw_players * (own_moves/opp_moves))

Results

After executing multiple times, I have summarized my finding by evaluating the performance of the agent using these three heuristics.

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    **************************
                        Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |   1     9  |   1     9  |   1     7  |   3
    2       MM_Open      5  |   5     8  |   2     9  |   1    10  |   0
    3      MM_Center    10  |   0     8  |   2     7  |   3    10  |   0
    4     MM_Improved    6  |   4     8  |   2     6  |   4     5  |   5
    5       AB_Open      5  |   5     7  |   3     6  |   4     3  |   7
    6      AB_Center     6  |   4     6  |   4     6  |   4     7  |   3
    7     AB_Improved    6  |   4     4  |   6     6  |   4     5  |   5
--------------------------------------------------------------------------
          Win Rate:      67.1%        71.4%        70.0%        67.1%
```

AB_Custom is Heuristic1: Center Score to Legal Move Heuristics

AB_Custom_2 is Heuristic2: Open to blank spaces ratio

AB_Custom_3 is Heuristic3: Distance between players to legal moves ratio

From the above results, we can conclude that the Center Score to Legal Move heuristic perform better. The proximity to the center, increases the legal moves. This in turn becomes favorable factor for a player to win. This also highlights the fact that occupying the edges and corner position reduces a player's chance of winning. I would suggest Heuristic1: Center Score to Legal Move Heuristics as a better option