2017

HEURISTIC ANALYSIS - AN ADVERSARIAL GAME PLAYING AGENT FOR ISOLATION

ARTIFICIAL INTELLIGENCE NANODEGREE, UDACITY

WALEED ALZOGHBY

waleed.alzoghby@outlook.com

Table of Contents

Introduction

In this project, I will 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. These rules are implemented in the isolation.Board class provided in the repository.

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.

Heuristics Description

I have implemented a three heuristic functions provided a long with “Null Score”, “Open Moves Score” and “Improved Open Moves Score” here is a description of each:

- Null Scope: This function returns 0 for all game board states which are non-terminal states.
- Open Moves Score: This function returns the number of moves available to player in a given board state as the score for non-terminal states.
- Difference of open moves score: This function returns the difference in the number of moves available to the player and the opponent player in a given board state as the score for non-terminal states.

Here is descriptions listing for the custom heuristic functions:

- Minimizing Opponent Moves: This heuristic logic goes in a way to minimize the opponent moves by applying this mathematical formula: Len (My Moves) – w * Len (Opponent Moves) where w is a calculated weight as follows w = 10 / ([current game move counts] + 1)

- Maximizing Player Moves: This heuristic logic goes in a way to maximize the player moves by applying this mathematical formula: x * Len (My Moves) – Len (Opponent Moves) where x belongs to (1, ∞) – x considered as 5 in the implemented project.

- Maximizing ratio of player to opponent: This heuristic logic goes in a way that the player should have more moves to the positions out of busiest corners and walls by applying the following mathematical formula on the number of available moves for each player:

  Player → score = Old Score + [1 - (busy spaces ratio * free space)] - for each free space
  Opponent → score = Old Score - [1 - (busy spaces ratio * free space)] - for each free space

Evaluation and Results of Heuristics

The tournament.py script is used to evaluate the effectiveness of your custom heuristics. The script measures relative performance of your agent (named "Student" in the tournament) in a round-robin tournament against several other pre-defined agents. The Student agent uses time-limited Iterative Deepening along with your 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. Your 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
- AB_Custom: AlphaBetaPlayer using iterative deepening alpha-beta search and the minimizing_opponent_moves heuristic

- AB_Custom_2: AlphaBetaPlayer using iterative deepening alpha-beta search and the maximizing_player_moves heuristic
- AB_Custom_3: AlphaBetaPlayer using iterative deepening alpha-beta search and the maximizing_ratio_of_player_to_opponent_move heuristic

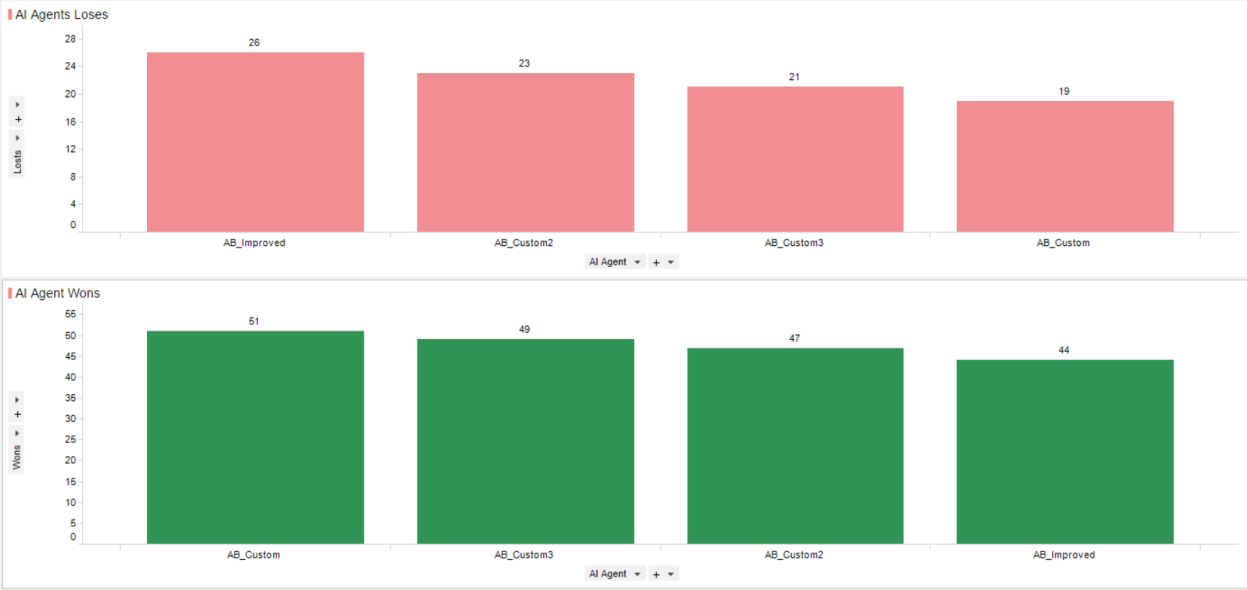
AI agents' performance tournament (10 matches for each opponent):

```
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
   8  |   2     7  |   3     8  |   2     9  |   1
    2       MM_Open
   7  |   3     8  |   2     8  |   2     8  |   2
    3      MM_Center
   8  |   2     9  |   1     8  |   2     8  |   2
    4     MM_Improved
   7  |   3     8  |   2     6  |   4     7  |   3
    5       AB_Open
   4  |   6     7  |   3     6  |   4     6  |   4
    6      AB_Center
   5  |   5     6  |   4     7  |   3     6  |   4
    7     AB_Improved
   5  |   5     6  |   4     4  |   6     5  |   5
--------------------------------------------------------------------------
          Win Rate:      62.9%        72.9%        67.1%        70.0%
```

AI Agents' evaluation chart:



As per the performance tournament results and visualizations we can observe that AB_Custom (Minimizing Opponent Moves) has a good performance over ID_Improved and remaining agents.

Recommendation

As per the evaluation and provided results, I'm recommending AB_Custom (Minimizing Opponent Moves) for the following reasons:

- It outperforms all other heuristics with win-rate 72.9%.
- It doesn't depend on historical states so no need for much memory.
- It is easy to implement with low processing time cost.