Udacity Artificial Intelligence Nanodegree
Heuristic Analysis

Fabio Quimio Pereira Fujii

March 15, 2018

1 Introduction

In this project, students 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.

2 Heuristic Analysis

In this analysis I will be comparing the performance of my best customized heuristic evaluation functions. During the course, the evaluation function proposed was the number of moves available for the player, *my_moves* minus the number of moves for the opponent player, *opp_moves*. This evaluation function captures the information of how a particular move could influence enemy's movement. However, other features can be extracted from the game, like the distance from the center.

Custom heuristic evaluation 1:

$$\omega \cdot future\_moves + \frac{my\_moves - opp\_moves}{center\_distance} \tag{1}$$

The future_moves is calculated by taking the maximum possible moves for all possible moves in the given the game state, where the heuristic evaluation function is calculated. The ω is the weight, with $\omega = 2$ (chose empirically). The features *my_moves* and *opp_moves* are the moves available for the player and the opponent. Finally, *dist_center* calculates the Euclidean distance of the player from the center.

Custom heuristic evaluation 2:

$$\frac{my\_moves - \omega \cdot opp\_moves}{1 + center\_distance} \tag{2}$$

The second heuristic is a variation of the first one, without the computation of the future moves. The ω weight here values 1.5. In this evaluation function, because of the weight, an aggressive play style is adopted by the agent.

Custom heuristic evaluation 3:

$$\frac{\omega \cdot my\_moves - opp\_moves}{1 + center\_distance} \tag{3}$$

This function is a safer variation of the second heuristic evaluation function. The function tries to maximize its value by giving emphasis to nodes that leads to more moves.

The table below shows the efficiency of each heuristic evaluation function. All agents but IDAB_Improved did not had used iterative deepening.

Table 1: Result table from the tournament.py

Opponent	IDAB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Random	9	1	10	0	9	1	8	2
MM_Open	5	5	8	2	10	0	5	5
MM_Center	8	2	8	2	9	1	9	1
MM_Improved	4	6	7	3	7	3	6	4
AB_Open	4	6	7	3	6	4	5	5
AB_Center	8	2	7	3	6	4	8	2
AB_Improved	5	5	6	4	7	3	7	3
IDAB_Improved	5	5	5	5	4	6	4	6
Total	60.5 %		72.5 %		72.5%		58.8%	