Heuristic Report- Building An Isolation Game Playing Agent

Introduction

In this project, we were required to build an agent for the Isolation game. Isolation is a two player game played on a rectangular grid. Each player takes turn to fill up one of the squares on the grid, the player who runs out of moves first, loses the game. In this game, the players are only allowed to move in L shaped steps, and the steps that were once held cannot be occupied again.
In order to build this agent, we were required to:

- Implement the Minimax algorithm.
- Implement the Alpha-beta algorithm
- Implement the get_move algorithm for iterative deepening search.
- Implement three heuristic evaluations of our own.

Heuristic Evaluations

We were provided with four heuristic evaluations

- Null_score :this function returns a zero for all the game states which are non-terminal states.
- Open_move_score: this function returns the number of moves available for the computer player.
- Improved_score: this function returns the score equal to the difference in the number of moves available to the players.
- Center_score: this function returns a score that is equal to the distance of the player from the center of the board.

My heuristic evaluations are:

- Custom_score : this function focuses on limiting future opponent moves by assigning a higher evaluation function to a game state, if the possibility of future moves is scarce, and a lower evaluation function if the possibility of future moves is plenty. In the lectures, this was mentioned as #myMoves-2* #opponentMoves.
  It returns a float of the number of moves.
- Custom_score_2: this function finds the difference in player moves and an opponent's moves based on the distance between both players. The evaluation will help decide which moves are least favorable (penalty moves). It returns a float of the number of moves.
- Custom_score_3: this function focuses on aggressively chasing the opponent. It returns a float of the number of moves.

Evaluation And Results

In order to test the implemented functions, we were given a set up tournament that contained different players. The players included a random player, a minimax agent using open_move_score heuristic, a minimax agent using center_score heuristic, a minimax agent ayer using improved_score heuristic, an alphabeta agent using IDS and the open_move_score heuristic , an alphabeta agent using IDS and the open_move_score heuristic, and an alphabeta agent using IDS and the improved_score heuristic.

To create a fair estimate I have simulated 10 tournaments. The table below shows a summarized version of the results.

	Win Rate %			
Tournament	**AB_Improved**	**AB_Custom**	**AB_Custom_2**	**AB_Custom_3**
1	70.0%	64.3%	65.7%	61.4%
2	70.0%	58.6%	75.7%	61.4%
3	67.1%	61.4%	68.6%	60.0%
4	62.9%	67.1%	62.9%	64.3%
5	62.9%	67.1%	62.9%	64.3%
6	64.3%	61.4%	55.7%	55.7%
7	70.0%	58.6%	68.6%	55.7%
8	70.0%	61.4%	70.0%	60.0%
9	67.1%	54.3%	55.7%	55.7%
10	74.3%	68.6%	60.0%	57.1%
Mean Wins	**67.86%**	**62.28%**	**64.58%**	**59.56%**

The table below shows the total average wins against each player

	AB_Improved	**AB_Custom**	**AB_Custom_2**	**AB_Custom_3**
Random	8.6	8.6	7.6	8.3
MM_open	7.8	6.2	6.8	6.1
MM_center	8.2	8.6	7.6	7.8
MM_improved	6.8	6.6	6.5	6.3
AB_open	6.1	4.6	5.4	4.5
AB_center	5.1	5.2	6.3	5.2
AB_improved	4.9	3.8	5.0	3.5

According to the data the Alpha-beta outperforms all the heuristics. However, it can be deduced that the second heuristic **(AB_Custom_2)** should be the one used for the following reasons:

- The function finds the difference in player moves and an opponent's moves based on the distance between both players. This is important because it helps narrow down the available moves and which of these respective moves are least favorable (penalty moves).
- The function performs better when compared to the other heuristics at a 64.58% win rate.
- When going against other opponents, the average wins are more than half as to when compared to the other heuristics.