AIND Project #2
- Isolation Game -
Heuristic Analysis

Choice of my custom agents Agents

As the baseline agents are already using the « Random », « Number of moves open », « Square distance to the center of the board », « Difference in number of moves available to the players », I designed these three agents, which tend to usually overcome the baseline agents.

Custom Agent #1 (Better than baseline agent)

This agent tries to improve the agent based on the « Difference in number of moves available to the players » heuristic.

Returns a high rewards (« `inf` ») if the player has won the game, a high penalty (« `- inf` ») if the player has lost the game. When the game is not set, it returns a reward depending on a "weighted" difference between the 2 players number of legal moves available :

```
distance = (player_distance_to_board_center - 0 .707)/(4.301-0.701)
```

(Feature scaling is applied to the distance to the center of the board value, which ranges between 0.707 and 4.301)

```
score = nb_own_possible_moves - ( 2 * nb_opp_possible_moves * distance )
```

The idea behind this heuristics is somehow to combine the « Difference in number of moves available to the players » and « Square distance to the center of the board » : This heuristics tries to maximise solutions with much more playing moves available for the player compare to the opponent, while also pushing (reducing the penalty) for positions closer to the center of the board

---

Custom Agent #2

This agent is relying on a precomputed book of scores : Considering a blank 7x7 board, this static score book provides the number of maximum theoretical legal moves possible at each board positions when using the chess' knight moves (L shape).

```
score_book = [[2,3,4,4,4,3,2],
              [3,4,6,6,6,4,3],
              [4,6,8,8,8,6,4],
              [4,6,8,8,8,6,4],
              [4,6,8,8,8,6,4],
              [3,4,6,6,6,4,3],
              [2,3,4,4,4,3,2]]
```

n the accumulation of the number of theoretical maximum legal moves possible, for each of the legal moves currently available to each players. The final score consists in the difference between the 2 players scores

```
for each legal moves for the player:
  player_score <- player_score + score_book(move)

for each legal moves for the player's opponent:
  opp_core <- opp_score + score_book(move)

score <- player_score - opp_core
```

The idea behind this heuristics is to choose the positions offering the highest theoretical number of next possible moves. This heuristics produce low error at the beginning of the game (blank board), but the error increases as the game is getting ahead (number of blank positions is decreasing) : This is not a real

issue as this scoring technique is performed for both players, thus somehow « compensating » for the error.

Custom Agent #3

This is a variant of custom agent #2, and this agent is relying on the same precomputed book of scores.

Returns a high rewards (« `inf` ») if the player has oin the game, a high penalty (« `- inf` ») if the player has lost the game. When the game is not set, a mean score for the player is computed based on the accumulation of the number of theoretical maximum legal moves possible, but as if the player could play twice in a row.

It is a bit like « evaluating » a mean score for all the next moves available (depth+1), both for each players (but it is not like searching in the tree of possibilities, as it does not take into account the opponent next move)

```
for each move in player_legal_moves:
   for each next_move in player_next_legal_moves:
        player_score <- player_score + score_book(next_move)
   mean_score_sub_level <- mean(player_score)
mean_score <- mean(mean_score_sub_level)

for each move in opponent_legal_moves:
   for each next_move in opponent_next_legal_moves:
        opp_score <- opp_score + score_book(next_move)
   mean_opp_score_sub_level <- mean(opp_score)
mean_opp_score <- mean(mean_opp_score_sub_level)

score <- mean_score - mean_opp_score
```

Results

As this can be seen on a few tournaments results (next page), the first two custom agents tends to lead to better results than the baseline agents

Custom agent#1 is almost always the best one, while being challenged by Custom agent #2 in very few cases. Agents #3 is similar or better than baseline agent but roughly in only half the cases. Its lack of efficiency might be caused by the fact it is calculating a score considering one more next move (depth +1) but ignoring what action the opponent might perform, thus increasing its error (One possible idea to further improve such agent might be to try to exclude the opponent's legal moves from the options used to compute scores ?)

All agents lost a few games against the baseline agent using the « Random » heuristics. That's normal as our algorithms expect agents to always choose the best action.

Agents using Alpha-Beta pruning tends to usually perform better than agents using only minimax search. This might be partially explained by the fact the minimax algorithm, which is an exhaustive search, might not have to finish before timeout occurs, while the alpha-beta pruning allows to explore more positions that are relevant in the game tree.

The simple heuristic « Difference in number of moves available to the players » seems to be quite effective for this particular game, especially considering the chess' knight moves (L shape) combined with a limited board size, here 7x7, where the whole board can be crossed in 2-3 moves, making it hard to stay around the board center area or away from the board's border. Its simplicity might also lower the computation requirements. Building on top of this heuristics, as done in our custom agents (especially #1 and #2), seems a good foundation for solving this problem.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	10	0	8	2	8	2
2	MM_Open	9	1	7	3	9	1	10	0
3	MM_Center	8	2	8	2	7	3	9	1
4	MM_Improved	6	4	6	4	8	2	7	3
5	AB_Open	5	5	5	5	6	4	6	4
6	AB_Center	7	3	5	5	5	5	4	6
7	AB_Improved	3	7	6	4	5	5	3	7
	Win Rate:	65.7%		67.1%		**68.6%**		67.1%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	10	0	8	2
2	MM_Open	7	3	6	4	6	4	7	3
3	MM_Center	8	2	8	2	8	2	9	1
4	MM_Improved	8	2	9	1	7	3	6	4
5	AB_Open	4	6	5	5	6	4	5	5
6	AB_Center	6	4	5	5	5	5	3	7
7	AB_Improved	3	7	4	6	4	6	3	7
	Win Rate:	64.3%		**67.1%**		65.7%		58.6%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	9	1	8	2
2	MM_Open	8	2	8	2	7	3	9	1
3	MM_Center	7	3	9	1	10	0	9	1
4	MM_Improved	7	3	8	2	5	5	4	6
5	AB_Open	6	4	6	4	5	5	3	7
6	AB_Center	5	5	6	4	6	4	4	6
7	AB_Improved	6	4	6	4	5	5	4	6
	Win Rate:	68.6%		**75.7%**		67.1%		58.6%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	8	2	10	0	8	2
2	MM_Open	6	4	7	3	7	3	9	1
3	MM_Center	10	0	9	1	9	1	9	1
4	MM_Improved	6	4	7	3	6	4	7	3
5	AB_Open	4	6	5	5	5	5	3	7
6	AB_Center	4	6	5	5	8	2	6	4
7	AB_Improved	4	6	6	4	5	5	6	4
	Win Rate:	62.9%		67.1%		**71.4%**		68.6%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	9	1	9	1
2	MM_Open	7	3	7	3	5	5	7	3
3	MM_Center	7	3	10	0	10	0	9	1
4	MM_Improved	5	5	10	0	8	2	8	2
5	AB_Open	6	4	7	3	5	5	3	7
6	AB_Center	6	4	5	5	7	3	5	5
7	AB_Improved	6	4	7	3	5	5	5	5
	Win Rate:	65.7%		**80.0%**		70.0%		65.7%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	10	0	9	1
2	MM_Open	6	4	6	4	8	2	6	4
3	MM_Center	8	2	9	1	10	0	9	1
4	MM_Improved	8	2	6	4	5	5	6	4
5	AB_Open	5	5	6	4	6	4	4	6
6	AB_Center	6	4	7	3	6	4	4	6
7	AB_Improved	5	5	6	4	3	7	4	6
	Win Rate:	67.1%		**71.4%**		68.6%		60.0%	