INTRODUCTION

The goal of the project is to develop a AI game agent that plays the game of Isolation. Isolation is a two-player game played on a rectangular grid where both players take turns alternately to move their piece and the first player who runs out of available moves loses the game. The positions on the grid previously made are considered blocked for all future moves and in this specific version of Isolation game, the pieces are only allowed to move in L-shaped steps like a knight on a chess board.

OBJECTIVES

• Implement Minimax algorithm for searching the game tree.
• Implement Alpha-Beta pruning to improve the minimax algorithm
• Use Iterative Deepening (ID) approach along with the Alpha-Beta search to return the next best move in the time allotted.
• Develop new heuristic evaluation functions that perform better than the provided heuristic evaluation functions.

HEURISTIC EVALUATION FUNCTIONS

Three new heuristic evaluation functions are considered here:

Weighted Difference (aka AB_Custom): This function returns the difference between number of moves available to the player and number of moves available for the opponent with additional fact that the opponent move count is weighted by a factor of 2. The weighting factor is to ensure the separation between player move count and opponent move is magnified.

Squared Difference (aka AB_Custom2): This function squares the moves available to both. In scenarios where the difference in available moves is identical for two sets of moves (i.e. player_move_1 - opponent_move_1 == player_move_2 – opponent_move_2) on a given ply, this squared difference approach will give more weightage to the absolute value of available moves for the players.

Sum-Difference Ratio (aka AB_Custom3): This approach computes a ratio by considering the sum of moves available for each of the player and the difference in moves available to each player. By performing this operation, this heuristic will try to pick a value that is not solely based on difference between player and opponent moves alone. Instead this evaluation is based on separation in moves in relation to their absolute values as well. For an identical difference in moves across a given ply, this heuristic would pick that move which has more open positions to move on the board.

RESULTS

The following results were obtained by running the provided `tournament.py` script. These numbers are from 10 runs with each player starting first for a total of 20 rounds for every player versus his opponent.

#	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	8	2	6	4	9	1
2	MM_Open	6	4	6	4	6	4	8	2
3	MM_Center	10	0	8	2	6	4	6	4
4	MM_Improved	4	6	6	4	5	5	6	4
5	AB_Open	3	7	3	7	6	4	3	7
6	AB_Center	5	5	4	6	6	4	5	5
7	AB_Improved	6	4	6	4	3	7	4	6
	Win Rate	**61.4%**		**58.6%**		**54.3%**		**58.6%**	

OBSERVATIONS

- As can be seen from the above table, the winning percentage for *Weighted Difference* and *Sum-Difference Ratio* is identical with 58.6%. But in the match against "AB_Improved", *Weighted Difference* heuristic comes out on top.
- *Squared Difference* heuristic has a winning percentage against all opponents except "AB_Improved". But it did result in a couple of timeouts during the run.

To help narrow down the choices further, all three heuristic evaluation functions were tested against "AB_Improved" again by running another set of 10 matches with 20 rounds. But this time, max. number of moves in a given round and maximum depth searched in any round were measured. Please note max. depth and max. number of moves were not always for the same run.

Evaluation Function	Max. Depth Searched	Max. Moves To Pick Winner	Match Stats against AB_Improved		
			Wins	Losses	Win %
AB_Custom	53	25	15	5	75%
AB_Custom_2	64	29	9	11	45%
AB_Custom_3	46	26	11	9	55 %

CONCLUSION

After evaluating all the heuristics, it is recommended that *Weighted Difference* a.k.a *AB_Custom* evaluation function be used in building the game agent for the following reasons:

- **Performance:** Its performance in tournament of 75 % is far ahead of the other heuristic functions.
- **Computation Complexity:** Of the three heuristics considered, *Weighted Difference* is the least computationally intensive. The *Squared Difference* approach required two square operations per move, while the *Sum-Difference Ratio* method performs the more expensive division operation.
- **Depth/ Moves:** As seen in the above table, *Weighted Difference* function took no more than 25 moves in any round to generate a high winning percentage. The maximum depth searched was pretty comparable to other evaluation functions.