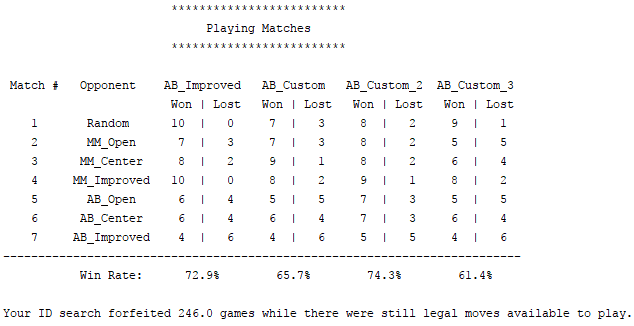
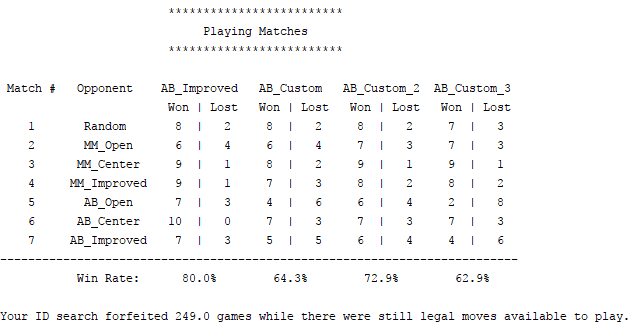
I tried 4 different heuristics –

* Custom Score #1: Return negative of number of opponent moves
  + Here, we are evaluating the best move by finding the move that leads to least number of opponent moves
* Custom Score #2: Return Number of Player moves – 2\*No of Opponent moves
  + Here, we are evaluating the best move by finding the move that leads to better moves for the current player than the opponent player
* Custom Score #3: Return the difference between current and opponent Player’s distance from the board center
  + Here, we are evaluating the best move by finding the move that brings the current player closer to the center than the opponent.
* Custom Score #4: Combination of score logic from #2 & #3
  + Here, we are evaluating the best move by using the combined logic from points 2 & 3 above.

Comparison between Custom Score logic 1, 2 & 3:



Comparison between Custom score logic 4, 2 & 3:



Observations & Analysis:

From all the scoring functions that I tried, custom scoring function #2 (“Number of Player moves – 2\*No of Opponent moves”) gives the best performance since it compares the available options of both the current player and the opponent. Custom Score logic #4 combines the best of #2 & #3 but is less effective (than #2) because its computation logic takes much longer resulting in exploration of fewer levels (and nodes by iterative deepening).

Recommendations:

Based on my analysis, I would like to recommend my evaluation function #2 which is “my\_moves – 2\*my\_opponent\_moves”. The main reasons are –

* Better performance (win rate) compared to the other evaluation functions that I tested
* This weighs the opponents\_moves higher than the current player’s moves for more aggressive game play and chases the opponent
* The timeouts/forfeits isn’t bad with this approach and it compares well with other approaches. I updated tournament.py to display the timeouts & forfeits for my evaluation functions 1, 2 & 3. Here are the observations (didn’t adjust headers & footer rows)

