**Heuristic Analysis**

AIND - Feb Cohort
Peeradej Pete Tanruangporn
Due 16 march 2017

**Role of Heuristics**
In the context of game-playing AI, the role of heuristics (or evaluation functions) is to evaluate the input state of the game and help select moves. This is critical when the search space is too big. The AI agent would run out of time, and it would still have to make some move. An evaluation function helps it make those decisions, given the state of board. Put differently, heuristics serve as the best estimate of a complete search.

**Good Heuristics**
A good evaluation function should be accurate and fast. Being accurate means that the AI agent will make the correct move that gives it the highest chance of winning. Being fast allows the AI agent to search deeper.

**Accuracy vs Speed tradeoffs**
Designing an evaluation function that out performs intuitive basic types in both accuracy and speed is non-trivial. Frequently it requires a trade-off that may not be worth it. Higher accuracy evaluation functions usually require more computation, slowing it down, while faster functions are often less accurate. The tradeoffs are not easily overcome and a lot of high performing evaluation functions end up being hand-crafted, based on game opening books, or people's knowledge of different piece's worth in various situations in the game.

**My three custom heuristics (code in appendix)**
1. Inverse Open moves (aka: KILL MODE)
2. Aggressive Improved
3. *Improved + Distance/8 (Best performance)*

**Inverse Open moves**
How would an agent that solely seeks to minimize the opponents move compare to one that seeks to solely seeks to maximize? Would the result be even against MM_Open?

*Result*
The result is horrible, giving a lower performance against every type of player. I guess turning a blind eye to one's survival is never a good strategy, ever. Further, I imagine that knights have a much harder time trying to find path that it could block each other. When it can't, then it simply has no heuristics.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random          Result: 930 to 70
  Match 2: ID_Improved vs   MM_Null         Result: 821 to 179
  Match 3: ID_Improved vs   MM_Open         Result: 522 to 478
  Match 4: ID_Improved vs   MM_Improved     Result: 500 to 500
  Match 5: ID_Improved vs   AB_Null         Result: 634 to 366
  Match 6: ID_Improved vs   AB_Open         Result: 431 to 569
  Match 7: ID_Improved vs   AB_Improved     Result: 386 to 614
```

```
Results:
----------
ID_Improved        60.34%

*************************
  Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:  Student  vs  Random        Result: 805 to 195
  Match 2:  Student  vs  MM_Null       Result: 554 to 446
  Match 3:  Student  vs  MM_Open       Result: 271 to 729
  Match 4:  Student  vs  MM_Improved   Result: 264 to 736
  Match 5:  Student  vs  AB_Null       Result: 365 to 635
  Match 6:  Student  vs  AB_Open       Result: 228 to 772
  Match 7:  Student  vs  AB_Improved   Result: 217 to 783

Results:
----------
Student            38.63%
```

**Aggressive Improved**

The baseline algorithm balances the agent's move left with the opponent's. My hypothesis is that a more aggressive version of this should outperform the baseline because as a rule of thumb in games and sports, offense is the best defense. A multiplier of 1.1 is introduced to the code that counts the opponent's moves. The multiplier is small as I only want it to break ties, preferring to reduce the opponent's moves rather than increasing one's moves, when the score is equal.

*Result*

This heuristics gave a very slight improve performance at 1%. However, gains mostly came from playing against bad heuristics (MM_null and MM_open) and there was no change for MM_improved and the AB branch. I believe this is because against worse heuristics (or bad players), playing aggressive can actually help corner the opponent. Against a good heuristic or deeper search however, preferring a type of move with the same score as another made no difference.

Additional test runs with higher multiple of 1.5 and 2.0 performed quite worse than the baseline. Tampering with moves selected by basic move counts have had steep cost.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random        Result: 930 to 70
  Match 2: ID_Improved vs   MM_Null       Result: 830 to 170
  Match 3: ID_Improved vs   MM_Open       Result: 492 to 508
  Match 4: ID_Improved vs   MM_Improved   Result: 500 to 500
  Match 5: ID_Improved vs   AB_Null       Result: 620 to 380
  Match 6: ID_Improved vs   AB_Open       Result: 426 to 574
  Match 7: ID_Improved vs   AB_Improved   Result: 382 to 618
```

```
Results:
----------
ID_Improved        59.71%

*************************
  Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random          Result: 941 to 59
  Match 2:   Student   vs   MM_Null         Result: 854 to 146
  Match 3:   Student   vs   MM_Open         Result: 524 to 476
  Match 4:   Student   vs   MM_Improved     Result: 502 to 498
  Match 5:   Student   vs   AB_Null         Result: 618 to 382
  Match 6:   Student   vs   AB_Open         Result: 427 to 573
  Match 7:   Student   vs   AB_Improved     Result: 383 to 617


Results:
----------
Student            60.70%
```

**Improved + Distance/8**

This heuristics came from a the idea that there is probably different strategical value between being in the center of the board, and being in the fringe. It also derives insight from a “knight’s tour”, which is a puzzle regarding how a knight should move so that it eventually traverses all over the board without repeating any position. The solution to a knight’s tour mostly involves a knight traversing on the side of the board. From that, I reasoned that in the long term, being on the “side” of the board probably creates more chances of survival. To incorporate that insight into the heuristics, we added distance into the baseline score. The distance is divided by 8 because we only want the heuristics to help break ties, preferring the side over the center, without sacrificing better net-worth moves just to avoid the center.

*Result*

This algorithm produced good improvement at about 2% increased in win-chance over baseline. Against Random, it does slightly worse probably because searching deeper would be more useful, but the slight increase in win rate in the rest, especially against MM_open makes up for it. The fact that it does slightly better against decent opponents, MM_improved, and AB_improved, signifies that being on the side of the board has real strategic or survival values that may be found with a deeper search.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random          Result: 935 to 65
  Match 2: ID_Improved vs   MM_Null         Result: 829 to 171
  Match 3: ID_Improved vs   MM_Open         Result: 515 to 485
  Match 4: ID_Improved vs   MM_Improved     Result: 500 to 500
  Match 5: ID_Improved vs   AB_Null         Result: 634 to 366
  Match 6: ID_Improved vs   AB_Open         Result: 415 to 585
```

```
  Match 7: ID_Improved vs   AB_Improved    Result: 357 to 643


Results:
----------
ID_Improved        59.79%

*************************
  Evaluating: Student (Improved + Distance/8)
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random          Result: 923 to 77
  Match 2:   Student   vs   MM_Null         Result: 850 to 150
  Match 3:   Student   vs   MM_Open         Result: 599 to 401
  Match 4:   Student   vs   MM_Improved     Result: 512 to 488
  Match 5:   Student   vs   AB_Null         Result: 639 to 361
  Match 6:   Student   vs   AB_Open         Result: 433 to 567
  Match 7:   Student   vs   AB_Improved     Result: 367 to 633


Results:
----------
Student           61.76%
```

**Recommendation**

Of my three heuristics, I would recommend the third one. It gave the most improvement in performance, as seen in the graph, especially against decent heuristics such as Open and Improved. The slight drop in win against random can ignored because winning against a random agent is trivial.

**Appendix**

**1. Inverse Open_moves**

```
opp_moves = len(game.get_legal_moves(opponent))

return -(opp_moves)
```

**2. Aggressive improved**

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(opponent))
diff = float(own_moves - 1.1 * opp_moves)

return diff
```

**3. Improved + Distance/8 (Best performance)**

```
row, col = game.get_player_location(player)
dist_row = abs(2 - row)
```

```
dist_y = abs(2 - col)
dist = (dist_row + dist_y)/8

own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(opponent))
diff = float(own_moves - opp_moves)

return diff + dist
```