Heuristic Analysis

Chansung, Park - 22 November 2017

Custom Evaluation Function

Former evaluation function measures superiority of a current player in each turn by calculating the difference between the number of moves that a current player could legally can make and the number of moves from the opponent player. It is quiet reasonable since the method represents more opportunities that a current player has.

However, the former method doesn't consider exceptional cases. For instance, when a current player starts placing on the center of the board, it has the maximum number of moves. On the other hand, on the next turn, the opponent wouldn't consider placing on the edge sides because considerable amount of moves couldn't be placeable. The opponent would rather tries to place near the previous player's site.

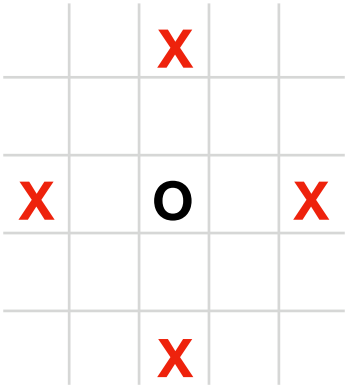

Fig1. Edge Moves

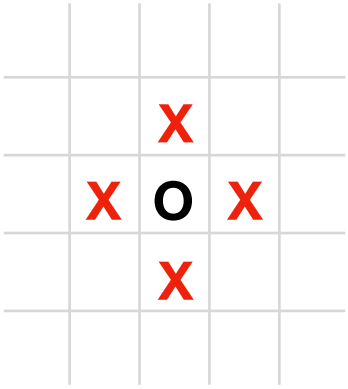

Fig2. Near Moves

Fig1. shows the maximum moves of four can be achieved based on the edges, and Fig2. depicts that there are two more opportunities of move achievable on the near sites at maximum. As a result, the opponent player, depicted as 'X', would choose the near moves over edge moves. However, on the second move, the situation get reversed like players started on the edges will get more chances of moves. It doesn't seem like the measurement of the number of moves has big influence since it can be fluctuated over turns.

Fig3. Start from near moves

Fig4. Start from Edge Moves

Fig3. and Fig4. depict situation about the start from the near and from the edge sites respectively. Even though the red color in the Fig3. had more opportunity than other red one in the Fig4., it lost the game. Especially, the red player in the Fig4. tried a special attempt to win the game.

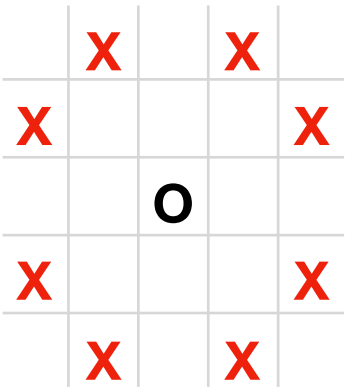

Fig5. Blocking the Opponent's Moves

Fig6. Exceptional Case

Fig5. Shows a brand new strategy to win the game. A player tries to block the opponent's next moves. This strategy expects to reduce the opponent's next possible moves and optimally maximize its next moves. However, there is an exceptional case as described in Fig6. The red player moved to block its opponent on the fourth move, but it failed because it is silly over the opponent. It needed to think about its next moves as well. In order to do that, the red player would choose the next move as in Fig4.. Like so, the red player could block the opponent along with a guarantee it still has at least an additional move to continue. Fig7. is the logical code which just does what I described above.

```
def custom_score(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    my_moves = game.get_legal_moves(player)
    my_pos = game.get_player_location(player)

    opponent = game.get_opponent(player)
    opponent_moves = game.get_legal_moves(opponent)

    base_score = float(len(my_moves) - len(opponent_moves))

    if my_pos in opponent_moves and len(my_moves) > 1:
        base_score = base_score + 2
        base_score *= len(my_moves) * 1.5

    return base_score
```

Fig7. Evaluation Function Code

```
**************************
    Playing Matches
**************************
Match #   Opponent    AB_Improved    AB_Custom
                       Won | Lost   Won | Lost
   1       Random       9  |   1     9  |   1
   2       MM_Open      9  |   1     9  |   1
   3      MM_Center     7  |   3     8  |   2
   4     MM_Improved    7  |   3     7  |   3
   5       AB_Open      5  |   5     5  |   5
   6      AB_Center     6  |   4     7  |   3
   7     AB_Improved    4  |   6     5  |   5
-----------------------------------------------
          Win Rate:      67.1%        71.4%
```

Fig8. Testment

Fig8. is the result from the testament comparing between the previous evaluation function(AB_Improved) and the new evaluation function proposed in this paper(AB_Custom). It shows a bit of improvement by 4.3% more winning chance.

Abbreviations

MM: Mini-Max

AB: Alpha-Beta