Heuristic Evaluation

July 27, 2017

Overview

Isolation is a two-player, deterministic game. Each time a player occupies a square, that square is "erased" from the possible moves for both players. A player loses when he has no squares left to move into. Let's take these constrictions into consideration and come up with a viable heuristic to assess the position of the game.

Optimal Heuristic

Our purpose is to block the opposing player from oppurtunities. Therefore, our heuristic has to to consider the difference between the quantity of moves we have versus our opponent's. We also want to push our opponent closer tp the edge of the board, because that's where the amount of legal moves for L-shaped moves decreases. Hence, we chould compensate when the legal moves of the opponent are closer to the edge of the board. Also, we want our player to be closer to the opponent, so we could actively take moves that limit our opponent. In summary, we reward minimizing the opponent's options.

Data from tournament

```
                      *************************
                          Playing Matches
                      *************************
```

	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	10	0	7	3
2	MM_Open	6	4	7	3	10	0	7	3
3	MM_Center	9	1	9	1	8	2	8	2
4	MM_Improved	5	5	9	1	7	3	9	1
5	AB_Open	4	6	6	4	6	4	6	4
6	AB_Center	8	2	8	2	4	6	6	4
7	AB_Improved	5	5	7	3	3	7	7	3
	Win Rate:	67.1%		80.0%		68.6%		71.4%	

Code

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

```
    if game.is_winner(player):
        return float("inf")
    player_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))

    play_loc = game.get_player_location(player)
    opp_loc = game.get_player_location(game.get_opponent(player))

    advantage_factor = 1

    dist = sqrt( (play_loc[0]-opp_loc[0])**2 +\
                 (play_loc[1]-opp_loc[1])**2 )
    rel = exp( -dist )
    if opp_moves:
        rand=random.randint(0, len(opp_moves)-1)
        if sum([opp_moves[rand][0] == 1, opp_moves[rand][0] == 7]) == 1 \
          or sum([opp_moves[rand][1] == 1, opp_moves[rand][1] == 7]) == 1
            advantage_factor *=2.5
    own_score = float(len(player_moves)*advantage_factor)
    opp_score = float(len(opp_moves))
    return float(rel*(own_score - opp_score*2))
```

Highlights

Compensation : advantage factor is multiplied by the number of moves the player has, thereby increasing the evaluation of the position as we can "push" the opponent closer to the edge of the board. We are not stressing that we will not be pushed only that the opponent is.

Minimize opponent's moves : When take the difference between our moves versus twice the opponent's we are compensating positions that give the opponent less maneauverability rather than positions that maximize our maneauverability. This is done for custom 2 and custom 3 as well, albeit exclusively.

Relation between distance and the score : The less distance between the players the higher the score (using the negative x-values of Euler's function). This rewards sticking close to the opponent and minimizing his move options whilst keeping the player's oprions large.

Tournemant Analysis

AB custom heuristic outperformed AB Improved by about 13 percent. Additionally it outperformed the other two custom scores. Let's analyze:

1. AB custom performed worst against AB open. AB open simply rewards positions with as much legal moves for the player, whilst AB

custom tries to minimize the opponent's. This testifies that our function puts most of its focus on worstening our opponent's condition.

2. In terms of computation, AB custom is most expensive. This also led to a timeout. This is because we are calculating the players' distance and rewarding pushing the opponent to the edge of the board.

3. Ab custom performed segnificantly well against AB improved although they are both cencerned with the opponent's moves. This shows the importance of the minimizing the distance between players and the relative disadvantage of being at the edge of the board.

4. Custom 2 and 3 were better than Ab improved. Each one of the ses functions put more emphasis on the oppponent's legal moves than AB improved. We can see the relationship between restricting the opponent and a higher win rate.

5. Search Tree : In AB custom, we reward heavily pusing the player to the side of the board, thus we can prune other options, minimizing the search space. Also, minimax performed worst the AB in all categories versus AB custom showing the importance of pruning.