Analysis of heuristic functions for the game of isolation

Abstract

The goal of this analysis is to evaluate the heuristic functions implemented for the Game agent. The purpose of these heuristics is to beat the baseline implementation, which is Alpha Beta with improved score.
I have three strategies that I will use:

1. Be defensive and try to block the opponent options as much as possible (AB_Custom)
2. Start out defensively and then switch to offense as the number of moves increases (AB_Custom_2)
3. Start out offensively and then switch to defense as the number of moves increases (AB_Custom_3)

The results of the heuristics is shown below

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

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      176 |  24    188 |  12    191 |   9    189 |  11
   2       MM_Open     147 |  53    153 |  47    161 |  39    147 |  53
   3      MM_Center    180 |  20    174 |  26    176 |  24    166 |  34
   4     MM_Improved   140 |  60    149 |  51    149 |  51    140 |  60
   5       AB_Open     99  | 101   99  | 101   107 |  93    101 |  99
   6      AB_Center    109 |  91    114 |  86    108 |  92    110 |  90
   7     AB_Improved   101 |  99    103 |  97    105 |  95    93  |  107
--------------------------------------------------------------------------
          Win Rate:      68.0%        70.0%        71.2%        67.6%
```

The results shows it might be most beneficial to be defensive in the beginning. The first two algorithms, AB_Custom and AB_Custom_2, uses this principle. As can be seen they both beat the AB_Improved score.

The results were achieved by setting the number of matches to 100 in the tournament.py script. 100 was chosen to achieve a broader foundation to better judge the algorithms.

AB_Custom: block the opponent

Implemented in: blocking_improved_score

In this function the player p will try to block the opponent o from the beginning. By setting ε to 2 and multiply it with the number of moves for the opponent $m(o)$ and then subtract it from number of player moves $m(p)$ we get a blocking behaviour from the start till the end

$$h(g,p) = m(p) - (\varepsilon * m(o))$$

The strategy is to get as close to the number of moves as twice the opponent. This will limit the opponent's options.

AB_Custom_2: start out defensively

Implemented in: decrease_blocking_improved_score

In this function the player will start out trying to block the opponent and then gradually become more offensive without considering the opponent. It is based on the number of legal moves for the player $m(p)$ minus the number of legal moves for the opponent $m(o)$. We have a start out blocking weight ε of 2 and an empirically tested decreasing value γ of 0.5, which is multiplied with the number of moves in the game gm so far.

$$h(g,p) = m(p) - (\varepsilon * m(o) - (\gamma * gm))$$

In this function we subtract the calculated increasing factor from the blocking factor. This gives a decreasing blocking effect as the game progresses.
It seems to be most beneficial and this is probably because blocking will have most effect in the beginning where there are most options, but as the game progresses there will be fewer options and trying to block the opponent will no longer be effective.

AB_Custom_3: start out offensively

Implemented in: increase_blocking_improved_score

This function will start out with a strategy that seeks a move that creates more options than the opponent. Besides this it does not care too much about trying to block the opponent. This will change as the game progresses, so eventually the player will try to block more and more.

A move count factor is created that consists of a weight γ which is empirically set to 0.5 and then multiplied with the number of game moves gm.
So we have the heuristic function $h(g,p)$ where g is the game and p is the player.

In the function we get the legal moves for the player p $m(p)$ and the legal moves for the opponent o $m(o)$.
The move count factor is added to the opponent's moves, creating an increasing blocking factor.

This creates the function

$$h(g,p) = m(p) - (m(o) + (\gamma * gm))$$

It seems not to be beneficial and this algorithm does not beat the AB_Improved score. Reason is probably that there will be fewer and fewer moves available as the game progresses and therefore a blocking strategy will not have the same effect at the end of the game as it will in the beginning of the game.

Conclusion

The algorithm that has achieved the best result is AB_Custom_2. This is the algorithm that starts out defensively and then switch to offense as the game progresses.
This algorithm beats the AB_Improved with a score of 71.2% winning, whereas the AB_Improved has a winning score of 68%. It means that AB_Custom_2 wins with a margin of 3.2 percentage points.
However the first algorithm, AB_Custom, also beats the AB_Improved score with 2 percentage points leading to the conclusion that blocking in the beginning of the game is very effective.

The recommendation is to use algorithm AB_Custom_2 as the preferred algorithm.