Heuristic analysis

Udacity Artificial Intelligence Nanodegree – Isolation game
Pierre Oberholzer – 15.11.2017

This report describes the three heuristics developed and tested as scoring function used to find the best move inside the alpha-beta search algorithm. The results obtained with these heuristics are reported in the result chapter below. The heuristic 1 "own moves only" performed the best with an average success rate of 54 % on 100 matches against "AB_Improved".

Heuristic 1 – "Own moves only"

The most promising heuristic obtained is inspired by the "improved score" provided. The idea is to calculate the difference of moves that players have and that their opponent has not. Based on the list of moves given for each player, we ignore the moves that intersect with the one of the opponent player, and calculate the difference of lengths of these own moves as our metric.

```
own_moves = game.get_legal_moves(player)
opp_moves = game.get_legal_moves(game.get_opponent(player))
only_own_moves = [c for c in own_moves if c not in opp_moves ]
only_opp_moves = [c for c in opp_moves if c not in own_moves ]
return float(len(only_own_moves) - len(only_opp_moves))
```

Heuristic 2 – "Free zone centroid"

The second strategy is inspired by the method "center score". Instead of keeping trying to move as far as possible from the center of the board, it can be advantageous as well to go to areas that offer more room for the next move. Such a strategy can be pursued by calculating the distance between the centroid of the free zone and the player location, and by trying to minimize this distance. Hence, we calculate the centroid of the free zone as follows:

```
blanks = np.array(game.get_blank_spaces())
centroid = blanks.mean(axis = 0)
center_h,center_w = centroid[0],centroid[1]
dist  = float((center_h - y)**2 + (center_w - x)**2)
return(1/(dist+0.0001))
```

In turns out that the "mean" method from the "numpy" library mean is actually quite slow. A faster implementation of centroid calculation was found in [1] and is used in the code.

Heuristic 3 – "Changing along game"

It can be assumed that the best strategy might change during the game. One way to introduce such change is to keep track of the coverage rate of the board (0 in the beginning and increasing until the end of the game). We can define a threshold on this coverage and change strategy accordingly.

```
board_size = game.width*game.height
blanks_threshold = 0.66

if len(blanks_array) < blanks_threshold*board_size:
      dist = float((center_h - x)**2 + (center_w - y)**2)
      return(1/(dist+0.0001))
else:
      dist = float((h - y)**2 + (w - x)**2)
      return(dist)
```

In the present case we alternate between the "center score" and the "free zone centroid". One possible strategy is to stay away from board center for most of the game duration, and try to approach the free zone centroid once some coverage is reached. Different threshold were tested, and the values 0.5 and 0.66 are used below.

Results

The following table shows the players chosen for the evaluation.

	AB_Custom "Own moves only"		**AB_Custom_2** "Free zone centroid"		**AB_Custom_3** "Changing along game"	
	Win	**Lost**	**Win**	**Lost**	**Win**	**Lost**
Opponent	6	4	5	5	3	7
Win Rate	60%		50%		30%	

Table 1 – Testing set-up

We reproduce this competition repeated 12 times, which represents a total of 120 matches for each player combination. The following results are obtained.

Opponent "AB_Improved"

	AB_Custom "Own moves only"	**AB_Custom_2** "Free zone centroid"	**AB_Custom_3** "Changing along game"[1]
Win Rate	50%	50%	40%
Win Rate	30%	40%	50%
Win Rate	60%	40%	40%
Win Rate	70%	50%	20%
Win Rate	70%	30%	50%
Win Rate	40%	40%	10%
Win Rate	60%	30%	30%
Win Rate	50%	70%	30%
Win Rate	50%	40%	50%
Win Rate	50%	60%	70%
Average Win Rate	**53%**	**45%**	**39%**

Table 2 – Testing results against " AB_Improved "

Opponent "AB_Center"

	AB_Custom "Own moves only"	**AB_Custom_2** "Free zone centroid"	**AB_Custom_3** "Changing along game" [2]
Win Rate	60%	60%	50%
Win Rate	40%	60%	50%
Win Rate	80%	30%	60%
Win Rate	70%	60%	50%
Win Rate	50%	50%	50%
Win Rate	50%	60%	50%
Win Rate	60%	80%	40%
Win Rate	50%	60%	40%
Win Rate	80%	60%	50%
Win Rate	90%	70%	70%
Average Win Rate	**63%**	**59%**	**51%**

Table 3 – Testing results against " AB_Center "

[1] blanks_threshold = 0.5
[2] blanks_threshold = 0.66

Recommendation

Based on the results obtained (Table 2), it looks like only the "own moves only" strategy might be able to perform slightly better than "AB_Improved" that runs the "own moves" strategy. The performances obtained against "AB_Center" (Table 3), which runs the "center score", are however illustrative of another potential. The "free zone centroid" appears to be slightly more efficient and could therefore serve as a candidate for a possible hybrid strategy: one could think of weighting the "own moves" by another metric, like the distance to the center of the board or the distance to the free zone centroid. Amongst all possible moves, the heuristic would guide the move toward a zone having more free spaces.
The "changing along the game" cannot be considered bringing any significant advantage.

Another improvement could be to effectively search for the local free zone having the more space. By dividing the board in four quadrants, the centroid and the free surface of the quadrant could be calculated. Another geometrical metric that could be used is the moment of area [2], which reflects the distribution of a surface in relation to an axis or to a point. One could for instance move the player so as to minimize the moment of the free area around his current position, which would reflect a potential wide choice of options. These strategies would however be computationally more extensive, which would counter-balance their possible benefit.

References

[1] https://stackoverflow.com/questions/23020659/fastest-way-to-calculate-the-centroid-of-a-set-of-coordinate-tuples-in-python-wi

[2] https://en.wikipedia.org/wiki/First_moment_of_area