Heuristics Analysis

Adversarial Game Playing Agent for Isolation

Submitted by

Sunil Thakur

Artificial Intelligence Nanodegree, Udacity

The project focuses at developing an adversarial game agent for the game "Isolation". This report describes the heuristics used in game tree search for minimax and alpha-beta pruning. The argument is that number of my-moves is to be considered positive, and more likely to win with many available moves.

HEURISTICS USED

1. HEURISTIC 1: This heuristic is based on the idea that opponent's moves should be minimized. It is expressed as:

   length(my_moves) - aggression_weight_factor * length(opponent_moves))

   The value of aggression_weight_factor = 1.5(works best)

   Results
   Here's how our game-playing agent performs with this heuristic:

   ```
   *************************
   Evaluating: ID_Improved
   *************************
   Playing Matches:
   ----------
   Match 1: ID_Improved vs Random Result: 16 to 4
   Match 2: ID_Improved vs MM_Null Result: 15 to 5
   Match 3: ID_Improved vs MM_Open Result: 14 to 6
   Match 4: ID_Improved vs MM_Improved Result: 13 to 7
   Match 5: ID_Improved vs AB_Null Result: 19 to 1
   ```

2. HEURISTIC 2: This heuristic is based on the idea that position closer to the center position of the board is better as you have more spaces to moves i.e. catching the mobility factor. I have used the absolute distance of the current positon from the center and not taken the square root to make give more weightage to the absolute square difference. Also combined with the heuristic 1 and used the weightage factor of 10 for the same.

   It is expressed as:

(10 * (my_moves - opponent_moves) + (my_abs_distance - opponent_abs_distance))

3. HEURISTIC 3: This heuristic is based on the idea of Penalize/reward move count if some moves are against the wall.Here again we have used the heuristic 1.No weightage factors but penalize for moves close to the walls.

   It is expressed as:
   length(my_moves) – length(my_moves_close_vertical walls) -
   length(opponent_moves) – length(opponent _moves_close_vertical walls)

RESULTS:

Following results is obtained from above 3 heuristics where heuristics 1 is giving the best results. Results were obtained playing tournament of 5 matches

Following is the table of the of 5 matches played.

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

Match #	Opponent	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3
		Won | Lost	Won | Lost	Won | Lost	Won | Lost
1	Random	10 | 0	9 | 1	10 | 0	9 | 1
2	MM_Open	7 | 3	6 | 4	8 | 2	6 | 4
3	MM_Center	9 | 1	9 | 1	8 | 2	9 | 1
4	MM_Improved	8 | 2	8 | 2	9 | 1	8 | 2
5	AB_Open	6 | 4	7 | 3	5 | 5	3 | 7
6	AB_Center	3 | 7	6 | 4	6 | 4	5 | 5
7	AB_Improved	3 | 7	4 | 6	4 | 6	5 | 5
Win-rate		65.7%	71.7%	65.1%	64.3%

Recommendation:

We recommend using **Heuristic 1** because of the following factors:

1. Win-rate is 71.72%, which is higher than other heuristics being evaluated.

2. It works on the intuition that the player has higher changes of winning when he has higher number of available moves. In addition, the distance from the center should play a role but the performance of heuristics 1 compared to heuristics 2, which uses this extra information, is not adding much value where the performance is more than 5% down to 65.14%.

3. Heuristic 1 has lower complexity because it’s simpler to implement as it works only on the number of player moves and its opponents move.

4. As only moves are involved in this heuristic, this has lower memory overhead as well.