Isolation Game Heuristic Analysis

by Ewa Tulodziecka

The report describes three different heuristic functions whose performance has been evaluated against the baseline 'ID_Improved' heuristic function computing a score based on a difference between the player legal moves and the opponent ones through iterative deepening.

Custom_heuristic 1

The heuristic builds on the "ID_Improved" taking into account the difference between the two players in legal moves, but it also factors in the difference observed one ply ahead in the future. Those two differences are summed up and used as the final score of the current game state. The future component added to the current legal moves difference ensures that the heuristic favors legal moves which creates more opportunities in terms of moves available in upcoming plies.

Implemenation:

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

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

own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

own_legal_moves = game.get_legal_moves(player)
for move in own_legal_moves:
    own_moves += len(game._Board__get_moves(move))

opp_legal_moves = game.get_legal_moves(game.get_opponent(player))
for move in opp_legal_moves:
    opp_moves += len(game._Board__get_moves(move))

return float(own_moves - opp_moves)
```

Custom_heuristic 2

This heuristic builds on the "ID_Improved" as well and on top of that, incorporates a bit of knowledge regarding the most promising moves or the areas which give the player the greatest chances of winning. As we know, not only number of available moves is important, but also, their locations, since the player closer to the board center is likely to perform better than the player whose remaining moves are closer to a border of the board where he can be easily cornered and will have gradually fewer options to move as the game advances. Thus, the heuristic adds the difference between player's distances (the Pythagorean Theorem based distance used) from the board center. The difference formula is the following: opponent distance from the center – player distance from the center, which in the best case

scenario gives 6 – 0 = 6 resulting in a score advantage for the player. In the worst case scenario, the result is negative (at most -6).

Implemenation:

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

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

own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
w, h = game.width / 2., game.height / 2.
player_y, player_x = game.get_player_location(player)
opp_y, opp_x = game.get_player_location(game.get_opponent(player))
player_dist = float((h - player_y)**2 + (w - player_x)**2)**(1/2)
opp_dist = float((h - opp_y)**2 + (w - opp_x)**2)**(1/2)

return float(own_moves - opp_moves + opp_dist - player_dist)
```

Custom_heuristic 3

The heuristic 3 is based on the same idea as the previous one but this one instead of favoring 'around center' positions penalizes the moves in the corners. Thus, the final score is computed as a difference between the players' numbers of available moves reduced by those which are in the corners.

Implemenation:

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

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

own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
corner_positions = [(0, 0), (0, game.height - 1), (game.width - 1, 0), (game.width - 1, game.height - 1)]

for move in game.get_legal_moves(player):
    if move in corner_positions:
        own_moves -= 1

for move in game.get_legal_moves(game.get_opponent(player)):
    if move in corner_positions:
        opp_moves -= 1
```

return float(own_moves - opp_moves)

Results

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	9	1	10	0	9	1
2	MM_Open	5	5	8	2	7	3	9	1
3	MM_Center	9	1	8	2	9	1	10	0
4	MM_Improved	9	1	10	0	7	3	6	4
5	AB_Open	6	4	6	4	9	1	5	5
6	AB_Center	5	5	8	2	3	7	8	2
7	AB_Improved	4	6	6	4	5	5	6	4
	Win Rate:	67.1%		78.6%		71.4%		75.7%	

The custom_heuristic 1 function has been proved to be the best and is recommended to be used what can be justified by the following:

- The custom_heuristic 1 builds on the "ID_Improved" which is quite efficient and simple, thus the time complexity of the custom heuristic 1 is comparable to the "ID_Improved" what should not impair the maximum depth searched by the algorithm.
- The function takes into account not only number of currently legal moves, but also available moves in the upcoming ply. This makes the algorithm less 'greedy' and more accurate as an evaluation function. It is because this version of the game where only L-shaped moves are allowed makes the value of the game state hard to predict by using only number of legal moves available in the current state.
- Tactics regarding favoring center moves and penalizing corner ones performed worse, thus the customer heuristic 1 function should be selected from all the considered functions.

Table 1Table presenting the results of all the heuristics in 7 tournaments consisting of 5 matches each per opponent.

	ID_Improved	Custom_heuristic 1	Custom_heuristic 2	Custom_heuristic 3
1	67.1%	78.6%	71.4%	75.7%
2	61.4%	70.0%	67.1%	72.9%
3	67.1%	78.6%	71.4%	75.7%
4	71.4%	68.6%	65.7%	67.1%
5	67.1%	74.3%	71.4%	65.7%
6	65.7%	65.7%	70.0%	71.4%
7	61.4%	85.7%	62.9%	77.1%
Average	65.9%	74.5%	68.6%	72.2%