Report of Adversarial Game Playing Agent

By Dwayne Elahie

The game playing agents are playing a modified isolation game, knights Isolation, where the allowed movements are L-shaped movements—like a knight in chess. The chosen experiment option was option 1, develop a custom heuristic. The game agent developed was alpha-beta search, with iterative deepening and depth limit. The baseline heuristic for the performance baseline is

#my_moves - #opponent_moves

The custom heuristic is

delta_liberty_difference / distance_between_current_and_opponent

where

delta_liberty_difference = (current_moves_remaining - penalty_multiplier*opponent_moves_remaining)

distance_between_current_and_opponent is the Manhattan distance. A side note on this, the board representation is using Bitboard Encoding for efficiency. The description of the setup is here

https://github.com/udacity/artificial-intelligence/blob/master/Projects/3_Adversarial%20Search/isolation/README.md

To decode the board distance back to row and column coordinate values for the custom heuristic to get the Manhattan, modulus and floor against the number 13.

The CustomPlayer (Alpha-beta search with iterative deepening and depth limit) agent had some hyper-parameters.

depth_limit # This is a hyper-parameter that needs to be tuned.

self.score_mode # 0: score baseline heuristic, 1: score custom heuristic A

self.penalty_coefficient # Custom heuristic penalty multiplier for having the opponent having moves.

Here is the log for playing CustomPlayer (Alpha-beta search with iterative deepening and depth limit) with both the baseline heuristic and the custom heuristic against other player agents found in sample_players.py.

#----------

Baseline Heuristic Score

10 Rounds

20 Games

(Start)

#----------

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o RANDOM -p 8

Running 20 games:

++++++++++++++++++++

Running 20 games:

++++++++++++++++++++

Your agent won 100.0% of matches against Random Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o GREEDY -p 8

Running 20 games:

++++++++++++++++++++

Running 20 games:

++++++++++++++++++++

Your agent won 100.0% of matches against Greedy Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o MINIMAX -p 8

Running 20 games:

+++++++++-++++++++-+-

Running 20 games:

+++++++++-+++++-+++++

Your agent won 87.5% of matches against Minimax Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o SELF -p 8

Running 20 games:

+-+-++++-++-+---++-+

Running 20 games:

-+-++++---+---+++++-

Your agent won 57.5% of matches against Custom TestAgent

#----------

Baseline Heuristic Score

10 Rounds

20 Games

(End)

#----------

#----------

Custom Heuristic Score

10 Rounds

20 Games

(Start)

#----------

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o RANDOM -p 8

Running 20 games:

++++++++++++++++++++

Running 20 games:

++++++++++++++++++++

Your agent won 100.0% of matches against Random Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o GREEDY -p 8

Running 20 games:

++++++++++++++++++++

Running 20 games:

++++++++++++++++++++

Your agent won 100.0% of matches against Greedy Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o MINIMAX -p 8

Running 20 games:

++++++++++++++-++++

Running 20 games:

+-++++---+++++-++++++

Your agent won 90.0% of matches against Minimax Agent

root@c5e11c24b1fd:/home/workspace# python run_match.py -f -r 10 -o SELF -p 8

Running 20 games:

+++++-----+++++--+++

Running 20 games:

-+++---+-++-++++++-

Your agent won 65.0% of matches against Custom TestAgent

#----------

Custom Heuristic Score

10 Rounds

20 Games

(End)

#----------

Here is the same information presented in a table.

Opponent	**Win %**	
	Agent - Baseline Heuristic	Agent - Custom Heuristic
Random	100.00%	100.00%
Greedy	100.00%	100.00%
Minimax	87.50%	90.00%
Self	57.50%	65.00%

Short answer to rubric questions.

Advanced Heuristic

- What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?

  The custom heuristic took the difference between the number of liberties the active player can make subtract the number of liberties the opponent can make multiplied by a penalty value. The penalty value is a hyper-parameter, meaning it is a value set by the programmer guided by intuition or empirically. That value is then divided by the Manhattan distance to get a common ratio to compare measures. I think getting the difference for the liberties of the active and opponent players and penalizing having the opponent player having moves helps promotes aiming to get better scores for the active player. By dividing by the Manhattan distance, it helps make a common reference frame to compare the values to.

  Just a side note, this heuristic did not take into account the unique characteristic movement of the knights (L-shape moves.) Nor does it take into consideration implicit environment conditions search as board symmetry. These properties could help reduce search times.

- Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?

  The search depth used for the experiment 100. Originally tested on lower depth such as 10 and 20 and the winning rate was significantly lower but it was much faster. I believe the accuracy matter more to the performance of the custom heuristic.