Build an Adversarial Game Playing Agent

HanByul Yang (hanbyul.yang@gmail.com (mailto:hanbyul.yang@gmail.com))

Overview

This is report for a project (Build a Game Playing Agent) of Udacity AI Nanodegree program.

In this project, I developed a custom heuristic for Isolation game based on minimax alpha beta search algorithm with iterative deepening.

Heuristics

There are 3 heuristics for experiments. The first one is for performance baseline. It is from lecture and its formula can be written as below.

```
#my_moves - #opponent_moves
```

Based on baseline heuristic, I added weights to both player's # of moves.

```
weight_my * #my_moves - weight_opponent * #opponent_moves
```

then, I added to number of actions of games and board size ratio to weight. So that, weight can be expressed below.

```
new_weight = play_count / board_size * 2
```

`play_count` is cumulative count of the number of actions applied to the board. `board_size` is 9 x 11. And constant `2` is for changing behavior of heuristic when empty size of board is half. For example, if `weight_my = play_count / board_size * 2` and `weight_opponent = 1` , before half of board is filled, the heuristic prioritizes #opponent_moves at the beginning of the game and then as total number of actions are increased, it prioritizes #my_moves.

So, my two custom heuristics for this project are two.

Heuristic 1

```
play_count / board_size * 2 * #my_moves - #opponent_moves
```

Heuristic 2

```
#my_moves - play_count / board_size * 2 * #opponent_moves
```

Result of experiments

Comparison with baseline performance

Baseline heuristic is from lecture. Performance result are measured with flag `-f` (`fair_matches` flag enabled) and `-r 50` (100 games) for each given strategy. (`RANDOM` , `GREEDY` , `MINIMAX`)

Below table shows win percentage against each test agent with time limit 150ms

Heuristic	Random	Greedy	Minimax	Self
Baseline	97.0	90.0	79.5	45.5
Heuristic 1	96.5	92.0	79.0	51.0
Heuristic 2	98.0	95.5	84.5	44.5

Two tables of custom heuristics show win percentage against given test agents with various time limit. Search depth can be big when time limit is higher.

Heuristic 1

Time limit (ms)	Random	Greedy	Minimax	Self
50	97.0	94.0	81.5	52.0
100	97.0	92.0	87.0	49.0
150	96.5	92.0	79.0	51.0

Time limit (ms)	Random	Greedy	Minimax	Self
200	98.5	91.5	83.5	50.0
250	97.0	90.0	83.0	47.0

Heuristic 2

Time limit (ms)	Random	Greedy	Minimax	Self
50	99.0	95.0	81.5	50.0
100	97.0	92.0	83.0	48.5
150	98.0	95.5	84.5	44.5
200	96.0	94.0	85.5	48.5
250	99.0	94.5	86.0	46.5

Analysis

Based on the first table, both custom heuristics perform similar or better than baseline heuristic.

With heuristic 1, search time matters less with greedy and self agents but with other agents, search time seems no big relation.

The search speed of heuristic 2 matters more when opponent is minimax agent. But when opponent is itself, search speed matters less. With other agents, search time seems no relation.

Result Answers

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

  I used number of total actions over total board size and number of each players moves. I think ratio of total actions at certain moment and board size can give more information in time manner than baseline heuristic.

- **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?**

  I mentioned this in analysis.