UDACITY

Back to Artificial Intelligence Nanodegree

Build a Game Playing Agent

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Fantastic work buddy.

Game Agent Implementation

(AUTOGRADED) Game playing agent can return an action.

- `.get_action()` method calls `self.queue.put()` at least once before the time limit expires

Correct! (Note: this rubric item was graded automatically.)

(AUTOGRADED) Game playing agent can play a full game.

- `CustomPlayer` successfully plays as both player 1 and player 2 in a full game to a terminal state (i.e., the agent does not deadlock during search, return an invalid action, or raise an exception during a game)

Correct! (Note: this rubric item was graded automatically.)

Experimental Results & Report

`CustomAgent` class implements at least one of the following:

- Custom heuristic (must not be one of the heuristics from lectures, and cannot *only* be a combination of the number of liberties available to each agent)
- Opening book (must be at least 4 plies deep)
- Implements an advanced technique not covered in lecture (e.g., killer heuristic, principle variation search, Monte Carlo tree search, etc.)

Wow, fantastic implementation of `custom_agent`, I wrote you a `heuristic` you can try:

```
def heuristic2(game, player):

    #Calculating center position of the game board
    mid_w , mid_h = game.height // 2 + 1 , game.width // 2 + 1
    center_location  = (mid_w , mid_h)

    # getting players location
    player_location  = game.get_player_location(player)

    # checking if player is the center location
    if center_location == player_location:
        # returning heuristic1 with incentive
        return heuristic1(game, player)+100
    else:
        # returning heuristic1
        return heuristic1(game, player)
```

Submission includes a table or chart with data from an experiment to evaluate the performance of their agent. The experiment should include an appropriate performance baseline. (Suggested baselines shown below.)

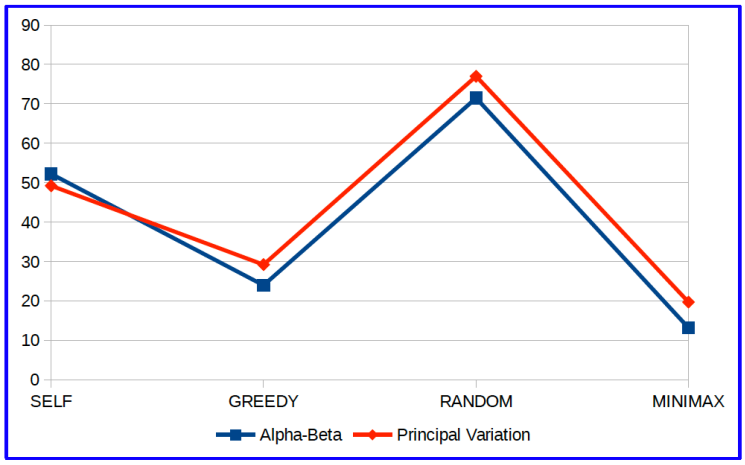
Advanced Heuristic

- Baseline: #my_moves - #opponent_moves heuristic from lecture (should use `fair_matches` flag in run_match.py)
  Opening book
- Baseline: randomly choosing an opening move (should *not* use `fair_matches` flag in run_match.py)
  Advanced Search Techniques
- Baseline: student must specify an appropriate baseline for comparison (student must decide whether or not `fair_matches` flag should be used)

Student produced several `results` and tabulated the performance of `heuristics` and the results of `matches` were also included:

Test Agent	Algotirhm	Run1	Run2	Run3	Run4	Run5	Average	#Games	Fair
SELF	Alpha-Beta	55	47.5	55	57.5	46.2	52.24	40	Yes
SELF	Principal Variation	45	55	46.2	48.8	51.2	49.24	40	Yes
GREEDY	Alpha-Beta	13.8	30	27.5	25	23.8	24.02	40	Yes
GREEDY	Principal Variation	32.5	45	25	28.8	15	29.26	40	Yes
RANDOM	Alpha-Beta	67.5	67.5	77.5	73.8	71.2	71.5	40	Yes
RANDOM	Principal Variation	81.2	77.5	75	72.5	78.8	77	40	Yes
MINIMAX	Alpha-Beta	10	15	17.5	12.5	11.2	13.24	40	Yes
MINIMAX	Principal Variation	15	23.8	20	20	20	19.76	40	Yes

Test Agent	Alpha-Beta	Principal Variation	%Improved
SELF	52.24	49.24	-3.00%
GREEDY	24.02	29.26	5.24%
RANDOM	71.5	77	5.50%
MINIMAX	13.24	19.76	6.52%



Submission includes a short answer to the applicable questions below. (A short answer should be at least 1-2 sentences at most a small paragraph.)

NOTE: students only need to answer the questions relevant to the techniques they implemented. They may choose *one* set of questions if their agent incorporates multiple techniques.

Advanced Heuristic

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

Opening book

- Describe your process for collecting statistics to build your opening book. How did you choose states to sample? And how did you perform rollouts to determine a winner?
- What opening moves does your book suggest are most effective on an empty board for player 1 and what is player 2's best reply?

Advanced Search Techniques

- Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?
- Why do you think the technique you chose was more (or less) effective than the baseline?

Student addresses many questions tendered above, and also they give their own opinion on the results and performance of algorithms:

CRITERIA	MEETS SPECIFICATIONS
CustomAgent search function uses an advanced search technique	`CustomAgent` class implements **at least one** of the following: • Custom heuristic (must **not** be one of the heuristics from lectures, and cannot *only* be a combination of the number of liberties available to each agent) • Opening book (must be at least 4 plies deep) • Implements an advanced technique not covered in lecture (e.g., killer heuristic, principle variation search, Monte Carlo tree search, etc.)
Report includes a table or chart documenting an experiment to evaluate the performance of their agent	Submission includes a table or chart with data from an experiment to evaluate the performance of their agent. The experiment should include an appropriate performance baseline. (Suggested baselines shown below.) **Advanced Search Techniques**

CRITERIA	MEETS SPECIFICATIONS
	• Baseline: student must specify an appropriate baseline for comparison (student must decide whether or not `fair_matches` flag should be used)
Report answers all required questions	Submission includes a short answer to the applicable questions below. (A short answer should be at least 1-2 sentences at most a small paragraph.) **Advanced Search Techniques** • Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline? • Why do you think the technique you chose was more (or less) effective than the baseline?

Game Agent Implementation

CRITERIA	MEETS SPECIFICATIONS
`.get_action()` method calls `self.queue.put()` at least once before the time limit expires	(AUTOGRADED) Game playing agent can return an action. • `.get_action()` method calls `self.queue.put()` at least once before the time limit expires
`CustomPlayer` successfully plays as both player 1 and player 2 in a full game to a terminal state (i.e., the agent does not deadlock during search, return an invalid action, or raise an exception during a game)	(AUTOGRADED) Game playing agent can play a full game. • `CustomPlayer` successfully plays as both player 1 and player 2 in a full game to a terminal state (i.e., the agent does not deadlock during search, return an invalid action, or raise an exception during a game)

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review