Build a Game Playing Agent

Meets Specifications

Commendable Learning Attitude.
To say, I was not impressed reviewing this work will be a lie. I enjoyed reading the report and also enjoyed the logic in the code. Creative!
I'll suggest you keep practicing to build up this amazing skill already gotten. As your reviewer, I think this project merits to be voted for excellence. Thanks for your time.

Please,take some few seconds to tell us the challenges you faced from each review and how we could make it even better. Reviewers are here for each student and we wish to give the highest quality possible. This is possible only if you help. Thanks.

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.)

✓ 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)

✓ 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?

Thank for your re submission.

DOWNLOAD PROJECT

RETURN TO PATH