## **PROJECT SPECIFICATION**

### **Build a Game Playing Agent**

**Game Agent Implementation**

| CRITERIA | MEETS SPECIFICATIONS | Student Comments |
| --- | --- | --- |
| .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 | Implemented in my\_custom\_player.py (lines 53-54)  (Note: Commenting the line depending on whether im running alpha-beta or principal variation) |
| 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) | Implemented in my\_custom\_player.py  All unit tests passing. |

**Experimental Results**

****

**Report**

| CRITERIA | MEETS SPECIFICATIONS | Student Comments |
| --- | --- | --- |
| 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.) | I have decided to implement the principle variation search with minimax  Implemented in my\_custom\_player.py  I have referred the following pages for implementation along with the lectures and AIMA book  [https://en.wikipedia.org/wiki/Alpha%E2%80%93beta\_pruning](https://en.wikipedia.org/wiki/Alpha–beta_pruning)  <https://en.wikipedia.org/wiki/Principal_variation_search> |
| 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**   * Baseline: student must specify an appropriate baseline for comparison (student must decide whether or not fair\_matches flag should be used) | The results are provided in page 2 of this report and also a chart comparing the baseline (alpha-beta) and the implementation algorithm(principal component search with minimax) is provided. As we can see in the chart, the the agent plays with itself, the results are around 50% and in all the remaining scenarios, the principal variation search works better. |
| 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? | * I observe 5.24%, 5.5% and 6.52% improvement with principal variation search for GREEDY, RANDOM and MINIMAX respectively. Please take a look at the table in page 2 of this report. * From <https://en.wikipedia.org/wiki/Principal_variation_search>, we know that principal variation search dominates alpha-beta pruning in the sense that it will never examine a node that can be pruned by alpha-beta; however, it relies on accurate node ordering to capitalize on this advantage |