Assignment 3: Minimax and alpha-beta pruning

Xavier F.C. Sánchez-Díaz[*]

21st September 2022

Deadline: 14.10.2022, 23:59 hrs

1 Overview

In this assignment, you will implement the **Minimax algorithm** for adversarial search. For that, we recommend following the pseudocode provided in **Figure 6.3 (p. 196)** of the textbook (AIMA 4th Ed.) Afterwards, you will extend such implementation to use **alpha-beta pruning** to speed up the search. Pseudocode for alpha-beta pruning is provided in **Figure 6.7 (p. 200)** of the textbook.

For this assignment, we will use the *Pac-Man Projects*, developed at UC Berkeley by John DeNero, Dan Klein, Pieter Abbeel, and many others. These projects were developed for UC Berkeley's Intro to AI course, and later made available for other universities. You can read more about the projects following this link.

UC Berkeley has developed an extensive **Python** codebase to support these projects, which provides the framework in which you will write your Minimax and alpha-beta pruning implementations. This means that **for this assignment, you must use Python and the provided framework.** The codebase implements the rules of Pac-Man and the visualisation of the game so that you don't have to write this code yourselves.

The code also includes an essential program—`autograder.py`. This program will run various tests on your code in order to verify that you have correctly implemented all the subtleties of the algorithms.

2 Set up instructions

We will focus on Project 2: Multi-Agent Search. Go to `https://inst.eecs.berkeley.edu/~cs188/su21/project2/` and read the introduction. From now on, we will refer to this webpage as *the project link*.

1. Scroll down to the paragraph that reads "The code for this project contains the following files, available a **zip archive**". Click on the link and download the Python codebase.

[*]loosely adapted from the work of John DeNero et al.

2. Unzip the contents and verify that the code works. You can do this by opening a terminal and navigating to the extracted folder. Then run `python pacman.py`. If everything is OK, a new window will pop up and you can use the arrow keys to play.

3 Implementation

In order to pass this assignment, you will need to complete the following questions from *the project link*:

- Question 2: Minimax
- Question 3: Alpha-beta pruning

As specified in *the project link*, your code needs to be implemented in the `MinimaxAgent` and `AlphaBetaAgent` classes, in the `multiAgents.py` file. Your code must then be tested for correctness using the `autograder.py` as follows:

```
python autograder.py -q q2
python autograder.py -q q3
```

The `autograder` will require that your code passes **all** the tests, so you will always get either 0/5 or 5/5 as the output. In the output, you can see which specific tests are the one that need to be fixed to get 5/5 points.

4 Deliverables

Once you have achieved 5/5 points on both questions 2 and 3, you need to deliver the following:

1. **Code.** All your code changes need to be contained in `multiAgents.py`, so **this is the only code file** you need to submit. Your changes need to be well documented.
2. **Results.** Deliver a report (PDF) with your successful output from the `autograder`. The file should contain **the entire output** from running the `autograder.py` on both `q2` and `q3`.

If you use any additional Python libraries other than those provided in the codebase from Berkeley and the standard Python library, please specify what you used and how to install it so that TAs can do the same.

Recommendations

If by the deadline you still have some tests that do not pass, submit your assignment in Blackboard as it is. TAs will make a judgement as to whether your assignment can still be accepted based on your code and how many of your tests passed.

Make sure that your code is well documented and that you include your autograder results when you submit on Blackboard.