CMPSCI 687: Homework 2
Fall 2020

Philip S. Thomas
University of Massachusetts
pthomas@cs.umass.edu

1 Instructions

For this assignment you will implement *any* algorithm to solve 687-Gridworld. The code is set up to make it easy for you to implement a black-box optimization algorithm like (first choice) hill-climbing, simulated annealing, a genetic algorithm, the cross-entropy method, or CMA-ES.

In the .zip containing this file, you will also find the source code that we have provided. You should start reading this code beginning with main.cpp. At a high level, the code creates TabularBBO and TabularRandomSearch agents and runs them each 1,000 times one 687-Gridworld.[1] The code then outputs the average returns for each episode (and each agent) to out.csv, along with information for plotting error bars (standard error).

You will be changing (and submitting) TabularBBO.hpp and TabularBBO.cpp. Currently, the code is set up to compare your TabularBBO algorithm to a naive random search (TabularRandomSearch.hpp and TabularRandomSearch.cpp). You are encouraged to look at how this competing algorithm works to get an idea for how to implement your own algorithm.

As described above, when you run the code it produces out.csv, which contains the average discounted returns for each algorithm and each episode (along with error bar information). You do not need to submit any plots, but while solving the assignment you may want to plot these results to see how your algorithm is behaving. You can write your own scripts to create plots from out.csv, or you can copy out.csv's contents into LearningCurve.xlsx, which will create a plot like the one shown in Figure 1.

This assignment is all or nothing. You will either receive full credit or no credit. To grade your submission, we will replace TabularBBO.hpp and TabularBBO.cpp in our project. No other files will be copied over, so your code must be contained entirely in these two files. Your code will be run for a maximum of ten minutes (the provided code takes roughly 30 seconds to run on my machine). If the average returns produced by your algorithm are above

[1]Note that 687-Gridworld is modified to terminate after 100 time steps, with a reward of -100 if this time limit is reached.

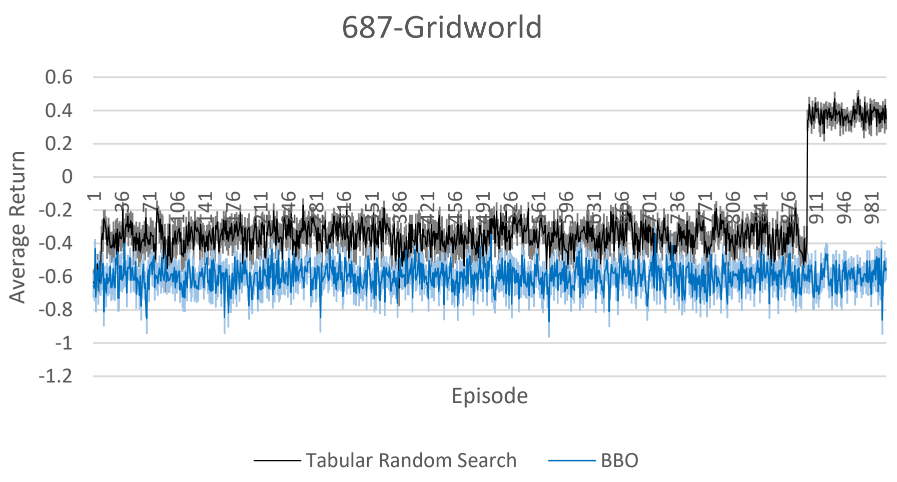

Figure 1: Learning curve plot for the out.csv file included with the assignment. Error bars indicate standard error over the 1,000 trials.

the average returns produced by TabularRandomSearch during the final 10% of episodes, you will get full credit. An example of results from a full-credit solution is provided in Figure 2. If your code does not compile, runs too long, or produces average returns below that of TabularRandomSearch, you will receive no credit for this assignment.

2 Due Date

The assignment is officially assigned on September 15, and due on September 22. You are welcome to get an early start. All students may have an extension until September 29, without penalty, as described for HW1 via email.

3 Eigen

The provided code relies on the Eigen library, which can be found here. To install this library, download the latest release. Inside of the main directory is a directory called “Eigen”. This is the only directory you need. Copy it into your project and ensure that it is in the include path. I usually put a directory called “lib” inside of my project, and place the “Eigen” directory inside of “lib”. I then add “lib” to the list of include directories (how this is done was covered in the C++ tutorial we provided).

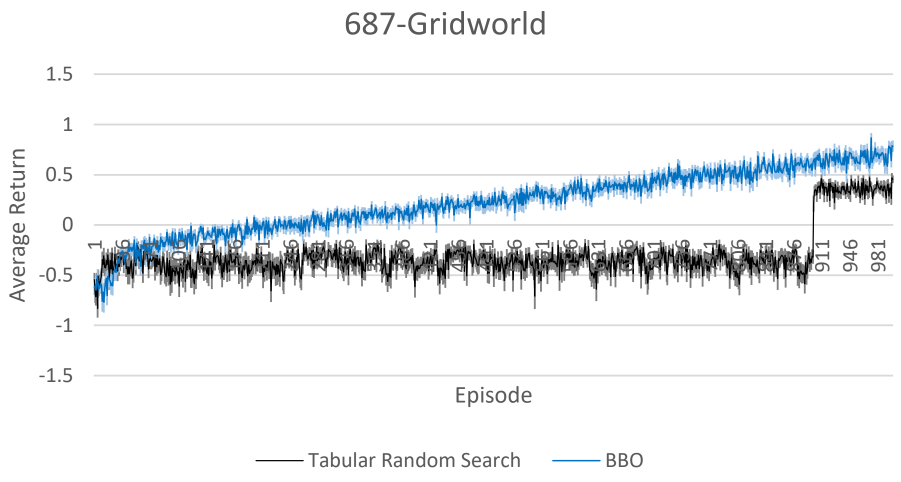

Figure 2: Example of a learning curve that would receive full credit.

4 Compilers

Your code must compile with either gcc (any recent version) or the compiler included with Microsoft Visual Studio. Note that Macs often pretend to be using gcc when they are really using CLang (which we will *not* be using in our grading pipeline).

5 Cheating

For this assignment, all code that you submit must either be code that you wrote, or code that you were provided with the assignment. Although you can discuss high-level topics like which BBO algorithm you plan to implement with others, all coding must be done individually. You may not use any additional libraries (other than C++ standard libraries and Eigen).

6 You Must Implement a BBO/Learning Algorithm

You might be thinking: I can solve this MDP by hand, and hand-code an agent that acts optimally. This will not receive credit: you must implement an algorithm that learns a policy on its own. You *can* tune hyperparameters like step sizes, distributions for random sampling, etc.

7 Hacking

This is not a security course. If the code that you submit attempts to compromise the machine it is running on (e.g., deleting or reading files outside of the project directory, downloading viruses or back-doors, etc.) it will be reported to the police (and depending on whether the machine was ever used for our DARPA or Army research, the FBI), and will result in your failing the course (via the "inappropriate behavior" clause in the syllabus).

8 Extra Credit

If there are any bugs in the assignment, the first person to point the bug out to me via email (pthomas@cs.umass.edu) will receive 5% extra credit on this assignment. Also, I will run your algorithm on a different, but similar MDP. The three students whose algorithms obtain the largest average returns during the final 10% of runs will receive 5% extra credit.

9 Assignment Changes

If any changes are made to the assignment after it is posted, this document will be updated and a description of changes included below.