Cmput 455 Assignment 2

CONTENTS

Due Mar 1, 11:55pm on eClass. Submit via eClass submission link on the main eClass course page.
Late submission (with 20% deduction) Mar 3, 11:55pm. Submit via separate eClass link for late submissions.

Assignment Overview

In this assignment, you develop a perfect endgame solver for the game of Gomoku. You also integrate your solver into a player, based on our starter code, which is an Assignment 1 sample solution. This player will play randomly at first, but will then play a perfect endgame as soon as it can solve the game.

To review the game Gomoku, see Assignment 1.

Setup

1. Review the Assignments page for general questions about assignments.
2. As in assignment 1, make sure you have Python 3, NumPy and GoGui 1.51 set up. You can review the procedures under Lecture 3 activities.
3. Also, check the Assignment 2 preview and "more preview" in the lecture slides.
4. Download assignment2.tgz and expand it. The directory `assignment2` contains:
   - Our starter code for assignment 2. This is based on a sample solution to assignment 1, and implements a random Gomoku player.
   - A file `assignment2-public-tests.gtp` with sample test cases for your program. See details below.
5. For reference, download the Lecture 8 python codes which include solvers for TicTacToe.

Goals of the Assignment

Implement the following functionality by adding or modifying existing GTP commands:

1. Implement a GTP command

   `timelimit seconds`

   The argument `seconds` is an integer in the range `1 <= seconds <= 100`. This command sets the maximum time to use for all following `genmove` or `solve` commands, until it is changed by another `timelimit` command.

   Before the first `timelimit` command is given, the default should be set to 1 second. The public test cases show examples.

2. Implement a new GTP command `solve` which attempts to compute the winner of the current position, assuming perfect play by both, within the current time limit.

Format:

```
solve
```

Your GTP response should be in the format:

```
= winner [move]
```

In the response, `winner` is either `b`, `w`, `draw`, or `unknown`. Write `unknown` if your solver cannot solve the game within the current time limit. Solving always starts with the current player (toPlay) going first. If the winner is toPlay or if its a draw, then also write a move that you found that achieves this best possible result. If there are several best moves, then write **any one** of them. If the winner is the opponent or unknown, then do not write any move in your GTP response. See the examples in public test cases.

3. `genmove color`

   Here, the argument `color` is either 'b' or 'w'. A version of the `genmove` command is already implemented in the starter code. You should change its behavior as follows:

   1. If the game is not over yet, the program should use the solver to try to play perfectly. The solver must respect the current time limit as described under `timelimit` above. Note that the time limit is per move, not for the whole game.
   2. If the game is not solved within the timelimit, or if toPlay is losing, then the `genmove` command should just play a random move.

Public Test Cases

We have grouped our `assignment2-public-tests.gtp` test cases into three categories.
Easy - our simple sample solver can do them within the time limit given.
Medium - our simple solver is too slow, but our better solver can do them.
Hard - even our better solver times out. These may be good test cases for checking your time limits and your GTP response in these cases. But if you can solve them, good for you!

There are comments in the gtp file which indicate these groups. To speed up your own testing, it may make sense to separate them into smaller groups and only run the most informative ones for you.

All the tests fail for the starter code, since it does not implement the new commands. Note that our given test cases only cover a small subset of possibilities. For full evaluation, we will use a more comprehensive set of tests.

Other GTP Commands

Many other GTP commands are already implemented in the starter code. Do not break their functionality, since we will test your program using those commands, such as:

```
boardsize, clear_board, cpu_time, name, version, play,...
```

You are free to add to the implementations of these commands if you need it, e.g. depending on your approach, you may need to do something extra in the play command. But do not break the existing functionality.

These commands are used by the test script. In this assignment, for convenience we will use multiple calls to `boardsize` and `clear_board` within the same gtp file, so make sure this use case still works in your solution.

Pre-submission Test and Submission

Follow the same general steps as in assignment 1 to create your `presubmission.log` file and your submission, but (of course) using your assignment2 directory, assignment2.tgz as file name, and `assignment2-public-tests.gtp` as test. Remember to include your new `presubmission.log` and `readme.txt`.

- Your file must be a valid tgz file named assignment2.tgz which can be uncompressed with tar -xf assignment2.tgz
- Do not change the name or location of your Gomoku.py main file, leave name and location the same as in the starter code.
- Do not introduce extra levels of directories or different directory names. Keep the same simple structure as in the starter code.
- You may add extra python files within the same directory.

Hints and Details - Read them All

- Your solve command must stay within the time limit. For some ideas of how to do that in Python 3, see Measuring Time. Whatever you do, make sure it works on the undergrad machines.

- We recommend that to ensure that your solver is working and keeping the time limits, you test your player by playing a few test matches between the original random player, and your player which includes your solver, with a modest time limit per move. You can use the gogui-twogtp tool to run the matches. See the gogui-twogtp notes.

  We think that such tests will help you to develop the program. However, you do not submit any test results with the assignment.

  Please be considerate in your use of shared resources such as lab machines, especially close to the deadline.

- No special error handling code is required in this assignment. We will only test with valid moves and valid GTP commands which have valid arguments.
- For examples of how to write new GTP commands, see the go and go2 codes, especially files gtp_connection_go1.py and gtp_connection_go2.py and how they are used in the main run() function of those players.
- We will only test with alternating play, so any `play` commands will be for the color toPlay. Clearing the board resets toPlay to Black.
- By default, we assume that teams will stay the same as in assignment 1. If you change your team, email the TAs the new information before the assignment 2 deadline.
- Note that being honest and returning "unknown" when your solver does not solve a position will lead to higher marks than randomly guessing the outcome.
- If you want to improve the efficiency of your solver, there is a large number of different approaches you can take. We have given some hints in class, but we will leave it up to you what kinds of improvements you try. In any case, make sure to retain correctness of your solver. You will lose marks if your program ``guesses'' the result, or does not compute it correctly because of bugs. Extensive testing is the key.

Marking

There will again be 5 marks for this assignment.

- 1 mark for your file `presubmission.log` which shows the log of your correct test execution, and your readme.txt file (as in Assignment 1).
- 2 marks for passing our public test cases
- 2 marks for passing our private test cases

Both sets will contain a mix of easy, medium and hard test cases. We will also test the player. The difficulty of these cases may vary, depending on which improvements you implement.

Code that does not run, or just hardcodes the public tests case by case, will receive a mark of 0. If your code needs fixes, you need to submit a revised version before the late submission deadline. TA will not attempt to fix your code under any circumstances. Use the discussion forum or consult the teaching team in case of questions.

Created: Jan 27, 2019 Last modified: Jan 8, 2021

Martin Müller
