CS485/612: Assignment 1

Fall 2012

Due: 04/20/2013 11:59 PM

Instructions

Please show all work on the written portion. The written portion MUST be submitted as a PDF, PS or text file. No other formats will be accepted. If you submit this assignment in LaTeX, I will give you back up to 10 points that you have lost on it.

This assignment will take time, especially if you are new to Python or if you have difficulties programming. Please start early, and ask questions or seek help if you are having problems.

List of things to submit:

1. The written portion of the assignment, in an acceptable format.
2. The python code for the programming portion.
3. A document in an acceptable format analyzing your program, as described in the programming section of these instructions.
4. Make sure your document includes the graphs specified in the instructions.

Please submit all of these documents in a tarball.

Written

Instructions If you are taking this course for undergraduate credit, you are not required to do any of the written problems. You may do the extra credit and / or extra extra credit parts to receive extra points.

If you are taking this course for graduate credit, you must do everything except the extra extra credit problem. You may do the extra extra credit parts to receive extra points.

Extra Credit (7.5 extra points) Consider a scenario with a 100x100 toroidal grid. There are a number of prey randomly distributed around the grid. There are four predators which can capture a prey by surrounding it (above, below, right, left). The prey can move in response to the predators. The prey have a strategy for avoiding predators, but you do not know what it is. Assume the predators and the prey move at the same speed. The goal of the predators is the capture the prey *as quickly as possible*.

- **(a)** What strategy would you assign the predators if they cannot communicate? That is, they can see where the prey and other predators are, but they cannot send messages to each other to coordinate a strategy.

- **(b)** What strategy would you assign the predators if they can communicate with each other?

Extra Extra Credit (15 extra points) Read the paper "Universality and Complexity in Cellular Automata" by Stephen Wolfram (Physica D 10 (1984) 1-35). Summarize this paper. How does Wolfram define the four complexity classes? What is your opinion of these definitions? Are they useful, useless, well defined, etc?

Programming

> **Instructions**: If you are taking this course for undergraduate credit, you must do everything except the extra credit and extra extra credit. You may do the extra credit and / or extra extra credit parts to receive extra points.
>
> If you are taking this course for graduate credit, you must do everything except the extra extra credit parts. You may do the extra extra credit parts to receive extra points.
>
> You must complete the programming assignment in Python. I used PyGame to create the GUI, and I recommend it. However, you can use any of the GUI frameworks listed here instead, if you'd like:
>
> `http://wiki.python.org/moin/UsefulModules#GUI`

In this programming assignment, you will study the behavior of a set of termite agents that live on a 100x100 grid. "Wood chips" are randomly distributed on this grid with a density of 0.3. The termites behave as follows:

- Each termite moves one grid square (up, down, left, right, or diagonal).

- If the termite is on a grid square containing a stick, it picks it up. This stick should now move with the termite.

- If the termite is on a grid square adjacent to another stick, it drops the stick. This stick should no longer move with the termite. You probably want to make sure the termite doesn't pick this stick up again immediately.

- Repeat until the experiment is over.

You must implement a function to evaluate the number of piles on the board. Two wood chips belong to the same pile if they have a contact along one of the four edges or one of the four corners. This must be called every *eval* iterations, where *eval* is specified at the command line with a "-e" option. For example, I should be able to run your program with the following command to evaluate the number of piles on the board every 100 iterations:

```
$ python2 termites.py -e 100
```

Your script should also take another command line parameter "-c" that runs it in command-line mode (*i.e.*, no GUI). You will want to use this when you are running a large number of experiments.

Here is an overview of the steps required:

1. Implement the code to simulate these termites, in python. Use the GUI to play with the behavior of the termites, until it looks correct. Output data (*e.g.*, number of piles) in a format that is easy to parse. Take a screenshot of your program executing to include with your documentation.

2. Implement code to count piles. Here is one way you can count the number of piles (assumes you color the chips white):

   - Go through every cell in the grid.
   - If the cell contains a white chip:
     - Increase the pile counter by one.
     - Use the flood fill algorithm[1] to color it and all the adjacent cells red.
   - Report the number of piles.
   - Reset all the chips back to white.

3. Write a shell script to perform 100 experiments that evaluate the number of piles every 1000 iterations, and run for 100,000 iterations, with a single termite. Generate a plot[2] of the average number of chip piles over the number of iterations. It should look something like the plot in Figure 1.

4. **Extra Credit**: Produce an experimental 90% confidence intervals for your graph of number of piles over time.

5. Use the same parameters as above, but vary the number of termites from 1 to 50. Plot the average number of piles at the end of x iterations depending on the number of termites. You can pick x to be any number where you think the number of piles is pretty stable. I used 100,000. Your plot should look similar to the plot in Figure 2.

[1] See `http://edge.cs.drexel.edu/GICL/people/peysakhov/Classes/Graphics1/lectures/L-05_Polygons.pdf` for a description of this algorithm.

[2] You are free to use any program you like to plot these results, but I recommend using `matplotlib` or `gnuplot`, both of which are freely available.

6. Explain these graphs and document your conclusions. Try documenting your results as though this were a real research paper. This means you should have one (or more) hypothesis, conduct experiments, and arrive at some conclusions about the behavior of the algorithm.

7. **Extra Credit**: Create two (or more!) different kinds of wood chips. See if you can design a behavior for the termites that causes them to be sorted them into different piles (that is, each pile would contain only one kind of chip). Here are some suggested measurements for the behavior of this modified termite scenario:

    - Measure the number of piles that touch.
    - Measure the length of the borders between piles.
    - Measure the porosity for each type of object. (Also, speculate on why they are porous.)

    Document how you measured the pile separation, and how you modified the termite behavior. Plot the algorithm's convergence and explain the behavior of the system. Include a screen shot of your program after it converges.

8. **Extra Extra Credit**: In addition to termites, add another agent we will call an entrosect. Entrosects move similarly to termites, except they only pick up chips that are in a pile of at least two chips, and they drop them in a location of their choosing that does not have any other chips adjacent to it. Run experiments and document how these two agents cause the system to behave as the number of entrosects and termites varies.

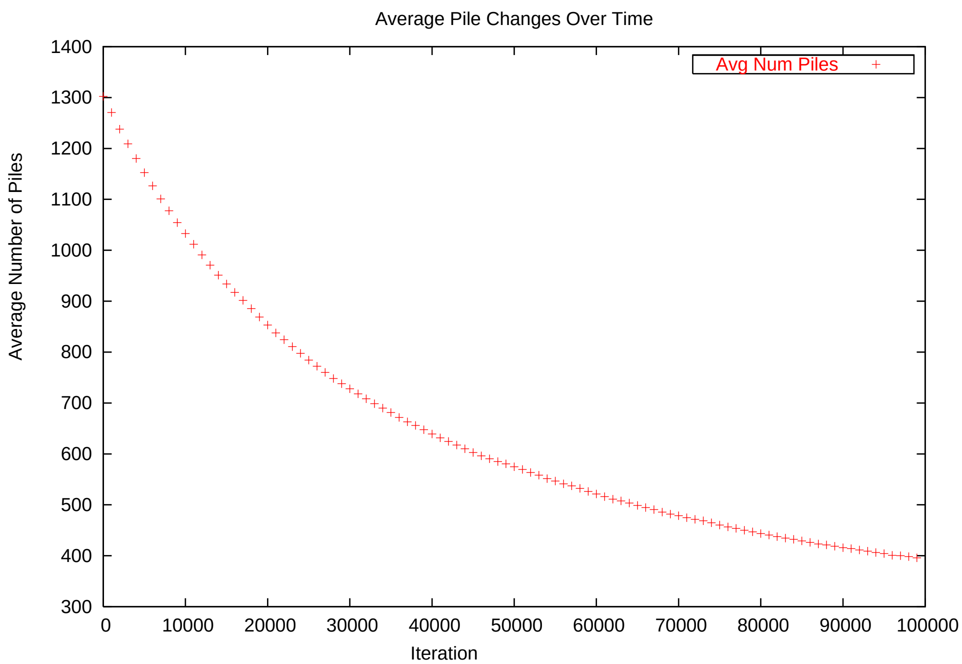

Figure 1: My plot of termite pile averages.

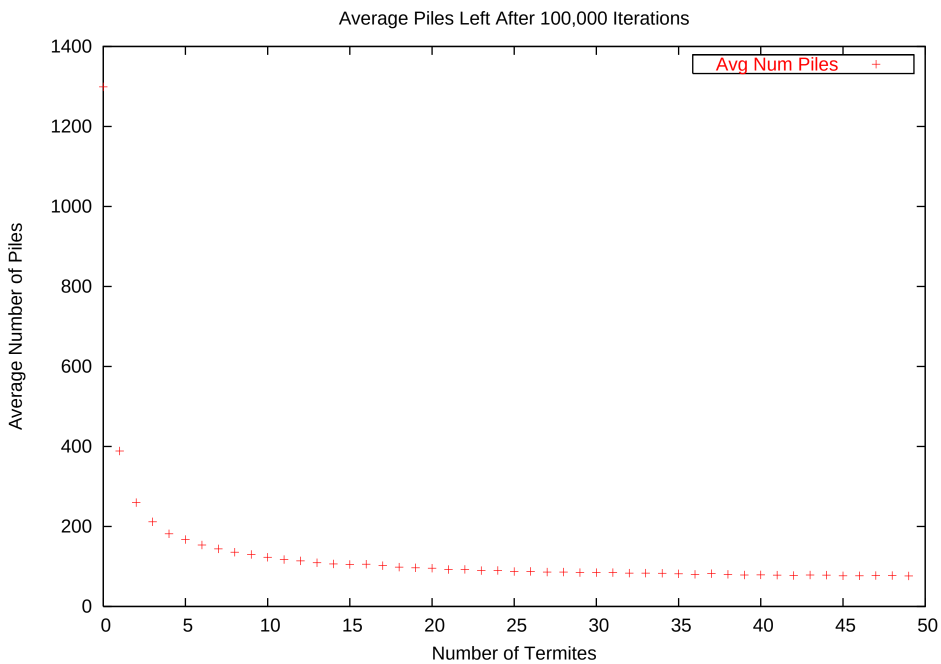

Figure 2: My plot of termite pile averages versus number of termites.