Analyzing Bacteria Cell Evolution and the Effects of Horizontal Gene Transfer Using Cellular Automata and Genetic Algorithms

Thomas Guerena

Summary

This project involves a naive simulation of bacteria cells which evolve, potentially to become resistant to the simulation's antibiotic. The purpose of the simulation is to test the difference in performance between cells which only mutate and replicate, and those that also take part in horizontal gene transfer. This is a special type of "mating" in which two cells link and swap genetic information, immediately altering their DNA. Extending this concept, we can parameterize the act of selection during horizontal gene transfer, and create an AI which discovers the most advantageous methods of selection.

Description

Goal

Using genetic algorithms and cellular automata, demonstrate the advantages of horizontal gene transfer in bacteria cells over mutation-only evolution. Then, provide examples of different methods of selection during mating (*intelligent selection*). Ideally, create an AI which discovers new selection methods, and provide analysis of those methods.

Hypothesis

Horizontal gene transfer (*HGT*) leads to resistant bacteria cells much faster than mutation-only evolution. HGT creates a more diverse gene pool, faster. I believe this will lead to the emergence of a resistant bacteria cell before non-HGT evolution.

Design Requirements

There are a number of AI algorithms used in this project. Bacteria cells are represented as cellular automata. The actions of those automata result in a genetic algorithm to defeat the simulation's antibiotic. More accurately, the automata are discovering a weakness in the antibiotic and settling into it. This is more or less the simple hill-climbing problem.

The second layer of this problem, horizontal gene transfer, will be parameterized by an instruction set, over which an AI will search for advantageous selection methods. This is similar to Melanie Mitchell's approach to finding intelligent instruction sets for *Robby Robot*.
Source: http://web.cecs.pdx.edu/~york/cs441/Readings/MMevca.pdf

Implementation Plan

Currently, I've built a prototype using Javascript to visualize the evolution process. This prototype, so far, includes everything leading up to intelligent horizontal gene transfer. Meaning, the second AI has not yet been built. I am using this to refine my strategies and fine tune the math behind the evolution and selection. Once I feel confident with my solution, I will port the prototype to Python, leaving out the code to render a visual representation. If time allows, I may add this back in using the TkInter Python library.
Source: https://wiki.python.org/moin/TkInter

The rest of the project will be built using Python on Mac and Linux machines. So far, no code has been borrowed, and intend to continue building the project from scratch for the sake of education.

Representation (Representation, Representation)

Bacteria Cell-ular Automata *(just one pun)*: An NxM integer matrix of values from 0 to 4095.
Bacteria Cell DNA: Three-digit hex codes (0-4095), allowing for color visual representation.
Antibiotic: A fitness function applied to every cell, every generation. My max population is relatively small, which indicates my simulation is representative of a small environment. In such a small environment, the antibiotic would fully permeate the space. Therefore, there's no reason to represent it in the matrix.
HGT Instruction Set: UNKNOWN

Test Plan

My prototype has already begun testing my cellular automata and evolution processes. This testing will continue as I fine tune the simulation parameters such as: step-values (mutation variance), mutation rate, starting colony size, rate of annealing, and more.

Future tests, involving the selection method AI, will be run on the simulation. The results of each run will be recorded. Those records will, at a minimum, include the following data:

- **Test Number**: This is the *nth* test. I want to see if my AI is getting better at creating instruction sets, or simply stumbling upon ones that work.
- **Instruction**: What instruction set was used? Keep track of the contents.
- **Run Count**: How many times was the test run?
- **Success Count**: How many times did the simulation end due to an environment completely full of immune bacteria?

- **Failure Count**: How many times did the simulation end due to the death of all bacteria in the environment?
- **Max Cell Count**: During the simulation, what was the maximum number of cells alive at one time?
- **Min Cell Count**: During the simulation, what was the minimum number of cells alive at one time?
- **Avg Cell Count**: During the simulation, what was the average number of cells alive at one time?

This data will be stored per test run, in a CSV file named using the test number. I can then import the file contents into a spreadsheet software to run statistical analysis.

Project Timeline

	10.12 - 10.18	**10.19 - 10.25**	**10.26 - 11.01**	**11.02 - 11.08**	**11.09 - 11.15**	**11.16 - 11.22**	**11.23 - 11.29**
Finish Prototype	🟩						
Port to Python		🟩	🟩				
Refactor, Optimize		🟩	🟩	🟩			
Selection AI, create			🟩	🟩			
Selection AI, test				🟩			
Full Test Runs				🟩	🟩		
Data Analysis						🟩	🟩
Final Report							🟩

Anticipated Obstacles

Time. I'm predicting that the mate-selection AI will take longer to build than the simulation, and take a significant amount of time to discover viable instruction sets. Given the magnitude of its search space, I may not be able to achieve my larger goal without changing my cell representation. Luckily, AI feels more like fun than chore and I'm hoping my enthusiasm for the subject will trump the amount of work left to be done.