AI Progress Report #1

Thomas Guerena

Last week was spent redesigning the prototype I had finished the week before. We met last week and determined that my prototype didn't show an emergent behavior - an issue attributed to the overall design and model of my simulation. Most of the code remains useable in the updated version of my project. This past week's work, however, took place on paper.

Definitions:

Cell: a matrix cell at some position (x,y) - not an organism!
Bacteria: one or more bacteria cells - organisms!

What I Thought I Was Doing

I thought my project was a study of simple genetic algorithms in cellular automata. What I was really trying to use was co-evolutionary programming. The simple genetic algorithm only came into play during horizontal gene transfer (mating). That confusion was part of what lead to my predictable simulation. What I ended up with was an example of a hill-climbing algorithm with an *bacteria vs antibiotic* theme. The model didn't allow for any emergent behavior or problem solving.

What I'm Doing Now

The **antibiotic**, rather than being a blanket fitness function, now has representation in the environment. Antibiotic will be placed at various locations in the matrix, and then it will spread based on a simple diffusion algorithm. The concept is:

a. antibiotic *sources* are place in the environment. From these cells, the antibiotic spreads radially. When antibiotic cells overlap, they combine to become a more effective antibiotic.
b. Each antibiotic cell has a *potency* associated with it. Its potency indicates how effective it is at killing bacteria. Potency will vary in one of two ways…
    i. As a function of time - the antibiotic weakens the longer it's in its environment. This would mean that all cells containing antibiotic would have the same potency.
    ii. As a function of density - the antibiotic weakens the farther it spreads. This would mean that it would cost an antibiotic cell *x* potency to spread to a neighboring cell.

The **bacteria** will remain similar, but their DNA representation, and its effect, will change completely. Before, all my bacteria behaved the same. Their DNA only served to defend them

against the antibiotic. My goal is for a bacteria to develop behaviors which help it survive. And most importantly, for there to be many different advantageous behaviors such that two bacteria can thrive in their environment while making different decisions.
I think I can achieve this by creating a simple set of actions that a single bacteria can take. So far, these are:

1. move
2. mate
3. replicate

A bacteria's DNA could then determine in what way those actions are taken for any given 3x3, bacteria-centered neighborhood. This is done by some *DetermineActions* function which takes a neighborhood state and a DNA, and returns *where to move to*, *who to mate with*, and *where to replicate to*. The only constraints of this function are population density. For example, assume a bacteria DNA was *0001 1000 0011* (move, mate, replicate), where *move* values mean 0, don't move; 1, move north; 2, move northeast; 3, east; etc. Then if there is a bacteria cell already occupying our acting bacteria's target position, it will shift its decision clockwise until it finds an empty cell, or decides it can't move.

Current Issues

I don't have a method for determining the chance of survival of a bacteria cell. Before, I used a fitness function. However, my plans do not include any real-time fitness. Instead, I had planned to use a genetic algorithm to discover advantageous DNAs by breeding a small population and testing each bacteria against every possible neighborhood state. Then, the best discovered DNAs would be used in the real-time simulation for testing and visualization.
My only ideas are:

a. P(survival) is the compliment of the contesting antibiotic's potency. The issue is that floating point potency creates a massive state space for neighborhoods, slowing my breeding process.
b. P(survival) is a value in a table based on the contesting antibiotic's potency, which would have to be an integer value between 1 and 5, for example. This makes it easy to value potency as a function of time, but it limits potency as a function of density.

All code can be found on github: https://github.com/thomasguerena/cs441-ai-project
I can start posting design notes to github if that seems useful, but I'll return to coding this week either way.