Investigations of Cellular Automata Dynamics

Timothy Hunter-Kilmer
TJHSST Computer Systems Lab 2006-2007

March 29, 2007

Abstract

John Conway's Game of Life was the first cellular automaton, showing how simple rules can generate amazingly complex patterns. He designed a field filled with cells, each of which could be dead or alive, and devised three rules to govern how these cells changed from one step to the next. Any living cell that had two or three living neighbor cells, out of a total of eight, survived into the next generation; any dead cell that had three living neighbors was born. Conway chose these rules to provide some stability but also allow a great variety of changes; I will look at many different sets of rules, both in their own stability and in their interactions with other sets.

Keywords: cellular automaton, simulated evolution

1 Introduction

Conway's Game of Life provides a foundation for further examinations of cellular automata. In his simulation, one set of rules governed the whole field, and every cell in it; these rules were designed to limit stagnation and allow substantial growth. The trial field contains 14,400 cells, each of which stores its own private set of rules under which it operates; in the same spirit as Conway, I limited the individual cell to having three ways to survive, and three ways to be born. When a cell is dead, it inherits rules from its neighbors based on the numerical preponderance of each rule. It is then judged to live or die based on those new rules; when any cell is dead, its rules are cleared, and it begins the next generation fresh.

The main method of gathering information was at first simply testing the program; as I fiddled with the GUI, making sure everything worked properly, I noticed which rules came up most often. With this data, I ran systematic trials, testing two rulesets that I judged to be moderately successful, and began to assemble a list of what rules were the most prolific, survived the most, and had the most variations.

Each trial consisted of a simple, though lengthy, process; beginning with two previously selected sets of rules, approximately half the board was covered

manually with each, resulting in equally-sized semi-occupied sections of one ruleset each. Every few hundred generations, I judged based on the colors on the field and the statistics in a separate panel whether one set had dominated the board, or if the trial was inconclusive, or if it needed more time. Whatever results I got, I recorded under the beginning rulesets.

With data on which sets work the best, I will be able to run those rules against each other and form a rough hierarchy based on characteristics such as variability, individual progress, and overall success.

2 Background

Cellular automata have largely been overlooked, many researchers preferring to focus on more complex multi-agent based models; the most notable exceptions are John Conway [1], who to a large extent created the field, and Stephen Wolfram [2], who examined Conway's ideas in fewer dimensions. Wolfram did much the same thing that I am, conceptually; taking the Game of Life, making large changes, and seeing what comes out.

There are two essential algorithms for this project: processing the whole field to go through a generation, and handling dead cells. The former is simply an exercise in looping over rows and columns, performing appropriate actions on each cell; the second can be implemented in several different ways. I chose to have each dead cell regenerate its rules at the beginning of each generation; if most of its neighbors survive with two living cells around them, the new cell will probably do the same thing, unless it mutates. Mutation is expressed as a probability with each cell that it will go against the majority position in retaining or discarding a rule when it is born; like the rulesets, mutation probabilities are passed down to neighboring cells.

3 Structure of the Program

The GUI allows me two main options for creating a cell: either click the button, and alter the rules manually, or use the checkbox at the bottom to set the ruleset that all new cells will have until that set is changed again. There are menus at the top, both to condense the different functions of the GUI into a much smaller space and to allow keyboard shortcuts. On the right side, there is a table showing how much of the board possesses each rule, expressed as percentages, and judged both from all the cells on the field, and all the living cells. Along with the graphic representation of each cell, using different colors for different sets of rules, this is the method for determining what data I can gather from each trial.

3.1 Writing the GUI

I made extensive use of the Java API in creating this GUI, as little of what I created is taught in this school. Menus and Menuitems provided a way to

organize my different functions concisely and compactly, occupying little space on the panel but easily accessible. The various checkboxes form the heart of the rules-changing mechanism, without which running trials would be a great deal more difficult. The statistics table, whose data is gathered during the main generation process and put into a formal Table, is half of my basis for assessing each trial, and judging whether anything is going to change.

4 Results, Assessment

I don't yet have enough data to make any conclusions on the main goal of this project. However, there are a few things I have observed:

4.1 Observations

1. If one ruleset is a simple pair of triples (e.g. 123/234 or 234/123), and the other is a slight variation on it (e.g. 123/134 or 234/124), the original set will almost certainly dominate the entire field.

2. If one ruleset is a variation on a strong pair of triples, and the other is an original weaker pair of triples, there's no way to predict which will cover the board.

3. The preliminary data, coming from trials with weak paired rulesets, is not a good indicator of the strength of the resulting rulesets. That is, if a certain set was the result of only a couple of preliminary trials out of a couple dozen, it still may well beat many of its opponents in subsequent trials.

References

[1] Martin Gardner, "MATHEMATICAL GAMES",*Scientific American 223*, pp. 120-123, 1970.

[2] Stephen Wolfram, *Cellular Automata and Complexity*, 1994.

[3] ibid., *Theory and Applications of Cellular Automata*, World Scientific Publishing Co. Ltd., pp. 485-557, 1986.

[4] ibid., "Cellular Automata",*Los Alamos Science 9* pp.2-21, 1983.

[5] N.H. Packard and Stephen Wolfram, "Two-Dimensional Cellular Automata", *Journal of Statistical Physics 38*, pp.901-946 1985.

Evolution Simulator Eric Turner
TJHSST
Computer Systems Lab, 2006-2007

Abstract

The purpose of this project is to develop a computer system that accurately simulates natural selection among species in a given environment. If the simulation is accurate towards real-world situations, then observing natural behavior will be significantly easier to manipulate than a natural environment.

The Evolution Simulator will incorporate programming AGENT-based simulation techniques, and will allow a user to easily use and manipulate multiple environments.

5 Introduction

Computer simulation is best used for experiments that are otherwise difficult to perform. A field that certainly fits this example is that of natural selection among species. Conducting a large scale, real world trial of this would require huge isolated environments, endless amounts of critters to populate them, and the ability to observe not just the outward appearance, but also their genetic traits.

This goal of this project is to develop a computer simulation that accurately imitates observed data. It will incorporate the traits and complexities that most influence evolution and population growth. It will allow simulated organisms to grow and develop, and a user to observe their genetic traits, and their change over time. By changing the environment, organisms will adapt and evolve to fit their needs. This does not occur because they are coded to do so, but because those with more favorable traits will survive and reproduce. The hypothesis made is that the more complex the system, the more the observed trends will relate to what is observed in a natural environment.

6 Background - Biology

The basis of this project is Darwins Theory of Natural Selection, the idea that a group of organisms can mutate and adapt to their environment without sentient knowledge of the process. Since evolution requires a large group of individuals, natural selection is not the only trend that takes effect. There are also population trends that take effect, including one species filling a niche in an ecosystem, an equilibrium between the populations of predators and prey, and so on. It is important to examine these aspects of the simulation, as well as the intended evolution of genes, to so see that a population of organisms is behaving normally.

Population trends affect evolution trends. A lot of these trends can be applied to mathematical models, depending on the complexity of the system.

The idea of a simulation is to approach the complexity of a real-world situation, and a test of the validity of the system would to apply established mathematical models to the societys population within the system. One approach to modeling population systems is to separate an environment into two pieces: resources and population. If there is only a single species in an environment, say herbivores, and there are limited amount of resources, say plants, in an area the population will expand to cover more area as their numbers grow. If this environment is somehow bounded, then the population size is also bounded. The population trends of this species will approximate a logistical curve, which will approach a limit of the carrying capacity (i.e. the maximum population the environment can support). However, a logistical graph fits less perfectly to a more complex system. One of the most complicating features is the spread of ages of the individuals within a population, especially if these ages affect their probability of dying or their reproduction. In these more complex situations, a population is less likely to attain equilibrium, but rather oscillate (Levin 2002).

It can also be that the additional elements of a social group can be related to evolution. Most genetic evolution occurs when there is a pressure on a society to evolve, either because the population will be better off when they evolve (developing lungs to move onto land) or because they will be worse off if they dont evolve (developing immunity to a disease).

It is because of this that predation is considered to be a prime driver of evolutionary change. When a population of predators and a population of prey can both evolve, the result is often an arms race, where the predators adapt to the preys defenses and the prey adapts to the predators attacks. This has been seen in nature, notably in the development of the first shells and exoskeletons (Pratt 2005).

The idea of evolution depending on different aspects of an ecosystem leads to examining an ecosystem as a whole for evolution. Evolution is determined to be at the individual levels. Natural selection occurs, and thriving individuals reproduce. An ecosystems properties can change to become more stable and resilient, but is not considered to evolve. Individual species evolve, and their evolution makes the system more stable (under the same principle as natural selection, strong systems survive). Because a system becomes more stable, the actions of one individual should not affect the system drastically. One way this can occur is that the population grows to a large enough size that the individual would not affect the majority of the population. On the contrary, if an outside force affects an otherwise isolated system, it can have drastic results (Kareiva 2002).

So an accurate simulation of evolution should result in certain trends and properties for the population. If the population starts at a relatively small number, the growth should resemble a logistical curve. Given the complexity of the system, the population should oscillate around the carrying capacity of its environment. This population should be stable, assuming the environment is isolated.

7 Background - Computer Science

There are two ways to model a dynamic population. The first way is to keep track of the size and attributes of a population, but not of the individuals. The model would update the population based on its attributes and predetermined trends. The result would be an example of population growth over time, assuming the trends used were accurate. The model is much more efficient, and much faster, since a program would not need to iterate over every individual every time step, just the representation of the population as a whole. The downside to this is that the trends must already be known, and are the basis of the model. The model would confirm that the mathematical formula would work by showing that a population would not die out, but it doesnt confirm that the formula is actually representative of a real-world situation.

AGENT modeling is fairly common in computer simulations, and can very easily simulate a dynamic population. This has been shown in Sugarscape or Conways Game of Life. Oftentimes by simply controlling the trends of individuals the trends of populations emerge. For the purposes of evolution, this is very important. An individual can have set trends that allow it to reproduce and eat, but the idea of *natural* selection is defeated if the individual also has set trends to evolve better and not worse. The downside to this type of model is the efficiency of the program. The processing time would be proportional to the population size, and as a population becomes significant, the situation becomes harder to model. A cap can be put on the population size by bounding an environment, which is realistic of nature. However, this also limits the accuracy of observed trends, which would be smoother at higher populations.

This simulation does use AGENT modeling techniques, but does not use any previous models, or development tools such as MASON. Therefore the focus of this project is the biological implications, rather that the computer science implications.

8 Development - Environments

Creating a system where organisms can evolve requires two parts. First, there must be an organism that is able to not only pass on genetic traits to its offspring, but also allow those traits to mutate. Second, the organisms must reside in an environment that is dynamic and easily modified. This environment is not only geographic and weather features, but also other inhabitants of an area (food, predators, etc.).

To create an environment structure, a coordinate grid was used. Everything in the simulation takes place in the Environment. An environment is a two-dimensional grid that contains a Set of Items.

Every object in the system is an Item, including organisms, food, and trees. Items have several characteristics, including position, size, and which Environment they are in.

A Vector represents an Items position. The Environment provides an origin

for all the position Vectors, which allows them to relate to each other (position A is to the left of position B, etc.). This allows the Environment and the Items to detect and prevent collisions, which is critical for most simulations.

Using only a Set of all Items in an Environment, collision detection would have an efficiency of order N-squared, where every Item iterates through every other Item to check if they are overlapping. Since the Environments usually need to be fairly big, this would make the program's run-time far too slow. So, every Environment, along with a list of all Items, has a matrix of Sectors.

A Sector is a set of Items, representing an area on the terrain of an Environment. A sector can either be composed of water, or non-water (land). A sector can also be active or inactive. If a sector is inactive, then Items cannot enter it, so it acts as if the environment does not exist in that particular area. This allows Environments to have shapes that are more than rectangles. A sector can also contain varying amounts of disease, which can affect the health of an organism. If given a position, an Environment can return which Sector contains that position. This allows Items to know what is immediately around them, and provides O(n) efficient collision-detections. Usually, a grid of 9 Sectors are checked, just in case an Item lies on the edge of a Sector.

Environments have several other physical properties in addition to content, size, and shape. One of these is temperature. While the temperature does not affect the Environment itself, it can affect Items within the Environment. An Environment also has a timer, which can show an age, or a time of day (day-length is also a property of Environments). Because of this, an Environment has two states: daytime and nighttime. Temperature within an Environment varies by a specified amount during each day, reaching a maximum temperature in the middle of the day, and a minimum temperature in the middle of the night. The temperature change follows a sinusoidal curve. The Environment has properties for the average temperature, and the temperature range.

Every Environment also has an Atmosphere, which is predominantly used for the breathing of plants and animals. An Atmosphere contains Oxygen and Carbon Dioxide, the amounts of which are proportional to the size of the Environment. Although an Environment is displayed as a 2-D grid, it is in fact a torus. In other words, when an Item moves out-of-bounds on a side, they can reappear on the opposite side. This is a quality that can be toggled within all Environments. Since the Items of the Environment are continuously being iterated through, there is a special method for the adding and removing of Items from Environments. Each Environment has two public Sets: itemsToBeAdded and itemsToBeRemoved. If through the course of iteration it is decided for an Item to be added or removed, then they are added to the respective set. This provides more centralized control of the contents of the Environment.

9 Development - Food

The predominant item in an environment is food. Food acts as plant-life that herbivores eat. During the daytime, they continuously accumulate energy (en-

ergy is not conserved, the assumption is made that food gathers energy from sunlight, which is not actually part of the program). A piece of food has a randomly set maximum energy amount. A piece of food is only considered edible, or ripe, by organisms if it contains at least half this maximum. Food intakes carbon dioxide and exhales oxygen. Food grows twice as fast and twice as big in water than on land.

10 Development - Organisms

While Organisms are just one type of Item in the environment, they contain the code that this simulation is most dependant on. While every item has a quality isAlive(), which returns a boolean, Organism and its subclassses are the only ones to return true. An Organism is considered alive, because if certain conditions are not met, it dies (is removed from the environment). If an Organism's health is 0 or its oxygenStored is 0, it dies. Otherwise, a probability is checked at every timestep to see if it will die randomly. This probability is based on several genetic factors, including age, health, size, its comfort with the environments temperature, and a genetic variable called probability of dying. The key property of Organisms for this simulator is that they can evolve. They evolve by manipulating genetic variables that are represented as numbers. Organisms have genetic traits, and non-genetic traits. Their non-genetic traits include aspects such as gender or their current health. Most of these traits, like age or number of children, will change over time. Their genetic traits remain constant throughout their life, as in reality. Genetic traits are passed on to offspring, and have a probability of mutating. This mutation has an aspect of randomness, and a genetic variable has equal opportunity to increase or decrease. How often genes mutate, and by how much, are also genetic variables. Most of these genes affect an organisms health and behavior, like metabolism. A few genes, like a genetic image (freckles) are used just to easily differentiate families of organisms. There are 15 non-genetic variables, and 29 genetic variables.

There are three types of Organisms: Herbivores, Predators, and Omnivores. Herbivores eat only plants. Predators are carnivores, and eat non-Predator Organisms. Omnivores eat both plants and non-Omnivore Organisms.

The locomotive mechanism of the organisms is the use of a body structure, containing legs. Each leg is a triangular structure of muscles and body segments, with the head at one vertex. This allows a leg to rotate around the head, and pull the head forward in any direction. The number of legs an organism has is genetic. Requiring organisms to move solely on muscle contraction and expansion adds an additional layer of realism to the simulation. With this property, the body structure of an organism directly affects how suited it is to survive in its environment.

11 Development - AI

Part of this project is to show that non-coordinated individual organisms can satisfy personal goals, and by everyone doing what in their personal best interest, the species will thrive. Organisms have a basic AI, in which they determine their greatest need, and attempt to satisfy it. To do this, they have 5 states: Hungry, In Heat, Afraid, Curious, and Out of Breath. If an organism is hungry, then it finds the closest food; if its in heat, it finds the best mate. If all its needs are met, it is curious, and explores its environment. How the organisms do this is determined by what type they are. Organisms have a basic memory, and can return to an Item it has found before, such as food.

12 Development - Viewing Data

The purpose of the project is to be able to view the genetic traits and trends of the simulated organisms. The program constructs graphs of the population of each species of organisms, as well as average gene values. The user is able to determine which genes the program tracks, as well as whether it retains the maximum, minimum, or average of these genes.

13 Experiments - Population Trends

The first basic experiment is to put a small group of Herbivores in a food-filled environment, and see how their population develops. The population should follow a logistic curve. Figure 1 shows the population graph for this experiment. As shown, the herbivore population initially rises exponentially, and then as it nears its environments carrying capacity, the population oscillates near the limit.

Another basic test is to show how a population of predators and herbivores grow with each other. This program has shown three possible results for an environment with predators and herbivores. The first is that the populations are able to remain in equilibrium. Figure 2 shows this trend.

The predator and herbivore populations were able to grow together, and eventually form equilibrium. As the predator population rises, they eat more herbivores, so the herbivore population falls. As the herbivore population falls, the predator population falls. As the predator population falls, the herbivore population grows, and the predator population grows. This is a stable equilibrium. Much more often, however, one of the other two possibilities occurs. In both of the following cases, the predator population dies out. Either the predators eat so many herbivores that the herbivore population goes extinct (Figure 3), or they eat so little that the predator population goes extinct, but the herbivores survive (Figure 4). These two possibilities are far more likely than the first. An almost perfect balance is required for equilibrium, and oftentimes the predators simply eat themselves to death. This leads to the conclusion that while there are many of these relationships in nature, most ecosystems based on this become unstable the majority of the time.

14 Experiments - Genetic Trends

Evolution takes place under stress. One such situation occurs when half of the organisms environment is blanketed with disease. The organisms start and expand on the non-diseased side. The organisms initially dont have sufficient disease resistance to move to the diseased side, but as their population grows larger, there is more pressure to get the food that others cant reach. As shown by figures 5-8, the population fills the healthy area quickly. After a few days, there are colonies of resistant organisms on the diseased side, and eventually there are equal numbers of organisms on both sides of the environment. The organisms evolved disease resistance because there was a benefit to being resistant.

A second test to verify evolution of the species is to show the organisms in an environment that is mostly covered with water (Figure 9). Herbivores will quickly fill the land areas, and will venture out into the water, which not only contains more food, but also food that is twice as nutrient. Organisms have three genetic variables that affect their breathing: lung capacity, oxygen intake, and oxygen use. It was initially suspected that the lung capacity and oxygen intake would rise, and the oxygen use would fall. The result for the average genes was a slow rise in the lung capacity; a fall in the oxygen use to the smallest unit, 1, but the oxygen intake did not change significantly. The reason for this is because the oxygen intake does not affect the time that organisms can spend underwater, but how long they take to recover their breath. Since after being in the water and eating, they are not stressed for time (they have few immediate needs), the time it takes to recover their breath is not important. Lung capacity is the chief determinate of how long they can spend in the water, which is why it rises.

15 Results and Conclusions

Judging from the experimentation discussed above, the simulations generated from this program show similar trends to those observed in reality. While the program by no means incorporates all of the factors that affect an animal society, it does approximate them. It develops an isolated system, which shows the appropriate population growth. It shows that organisms evolve in mostly expected manners, for instance their metabolism and probability of dying consistently decrease over time.

A future step in the program would be to have genetic mutation become more than just the changing of numbers. The idea of organisms changing their structure by actually rewriting code is more appropriate. The next step to do this would be to have organisms control their movement by the flexing and contracting of muscles. How these muscles are arranged could be a genetic variable, and this could lead to more unique species that are able to fill their niche.

16 Application

As a system such as this program becomes more and more complex, it also becomes more and more realistic, incorporating more variables that affect real-world situations. The goal is to make a program that is realistic enough to accurately simulate a generalized real world. This would allow researchers to test theories on computers before using expensive experiments, animals, or the need to go into the field. Any research conducted with this program, however, is hard to apply to something outside of evolutionary biology. The experiments that were discussed in this article were meant more as verification than as actual research for the field.

References

[1] Simon A. Levin, "Mathematical ecology", in AccessScience@McGrawHill, http://www.accessscience.com, DOI 10.1036/10978542.409750, last modified: May 6, 2002.

This article refers to different situations populations encounter, including the population of a single species within a bound environment, or a predator-prey population. This deals mainly with finding mathematical formulas that accurately plot the populations in different situations such as these. I predict that this article will be the most helpful to my research, since I can compare my data to this observed data trends.

[2] P. M. Kareiva, "Systems ecology", in AccessScience@McGrawHill, http://www.accessscience.com, DOI 10.1036/10978542.675900, last modified: June 27, 2002.

While this article doesn't focus on evolution, it does provide a mathematical relationship to an ecosystem as a whole, specifically how different species react and interact. The main idea of the article is that since many different species are redundant, a single species loss won't affect the system immediately, but as more and more species die out, the function they served in the system no longer is performed, and species-death rates increase. If after a significant run-time, there is diversity in the herbivore community because of different evolution paths, and diversity in the communities of the other Species classes, one could relate the herbivores as similar speices and the different classes as distinctly separate species performing a different function. If Herbivores as a whole die out, then a similar trend as mentioned in this paper would occur.

[3] C. C. Li, "Biometrics", in AccessScience@McGrawHill, http://www.accessscience.com, DOI 10.1036/10978542.083700,last modified: August 23, 2002.

This article deals primarily with probability, staticstics, and combinational theory. I hope to use this type of modeling to see if I need to analyze every organism at every timestep, or if I can take a sample and depend on the accuracy of the randomness in selection. The article also deals with relating observed trends to an existing model. While this article deals mainly in the abstract and doesn't help my project directly, it can improve my ability to extract and analyze results.

[4] Brian R. Pratt, "Neoproterozoic predatorprey dynamics", in AccessScience@McGrawHill, http://www.accessscience.com, DOI 10.1036/10978542.YB051300, last modified: November 30, 2005.

While this article is not related to computer science, and does not offer mathematical trends like previous articles, it does give some characteristics of a predator-prey relationship, in the example of the Neoproterozoic time period. It almost immediately refers to an escalating "Arms race" that affected how marine predators evolved in the Mesozoic era. This refers to how both predators and prey will evolve to gain the advantage in an equilibrium situation, resulting in continued equilibrium that would spawn a large range of evolution. This is the purpose for my inclusion of a predator-prey relationship in the Evolution Simulator, and this article gives a documented case of such an event.

[5] Jonathan Roach, "Seeing Around Corners", The Atlantic Monthly, April 2002

This article examines various multi-agent computer models. Part of the challenge of first making my program was to develop a system that would use an efficient environment to contain the agents, and comparing it to existing models that use such engines as MASON would show how well my own design compares. It is important that the program runs fairly efficiently, since I am working with thousands of individual agents, each taking up a significant amount of memory, and running an AI protocol.