A Cellular Automata Approach to Population Modeling

Alexa M. Silverman

May 21, 2009

Abstract

This project provides an agent-based model of the effects of temperature on population growth and change using a 2D cellular automata algorithm to predict behavior. The purpose of this project is to demonstrate how the behavior of a group is dependent on the interactions between individual group members, and to show that a cellular automata approach is valid in modeling ecosystem behavior. It models a system in which temperature and population changes have mutual effects on one another, and another in which a population's interactions cause a virus to be spread. The use of cellular automata as agents has not been thoroughly explored, and it is hoped that this research will be useful to researchers in the field of computer modeling as well as those of ecosystems science and biology.

Keywords: cellular automata, population modeling, 2D Life, agent-based modeling, temperature

1 Introduction

1.1 Cellular automata

Cellular automata (CA) are in fact a very basic form of artificial intelligence. In a CA program, a set of "cells" is created on a grid, and the behavior of each cell is determined by the states of its neighboring cells. Cells "know" their own states ("alive" or "dead"), the states of neighboring cells, and the rule that determines their behavior. These simple parameters, when applied over a large number of cells, create often surprising and complex patterns. In the past, CA have had several mathematical and computer science applications, but researchers have begun to see their viability in the field of agent-based modeling.

1.2 Life

The "Life" collection of automata rules is based on the idea that survival of individuals in a population requires an adequate number of neighbors; that an individual can die due either to loneliness or overcrowding, and that an individual is only born when a "family" of individuals is already present. In Life models, the appearance of the grid changes at the end of each turn. The best Life models use rules which create lifelike behavior.

1.3 Purpose

The purpose of this project is to model the behavior of a population using a cellular automata approach as well as to demonstrate that cellular automata can be used in population modeling. The model will observe the affects of temperature on the population and vice versa, in a manner similar to the Daisy-World simulation in ecosystems science.

For this model, the rule 14/3 was chosen because it causes cells to grow and move in a pattern resembling the spread of a species; cells begin localized and become more widespread. The use of the 14/3 rule suggests two types of individuals: "antisocial" (survives with 1 neighbor) and "social" (survives with 4 neighbors). This adds a degree of complexity to the model.

2 Background

2.1 CA Modeling

The field of cellular automata modeling is relatively new to computer science. In a study by R. Brukelaar and Th. Back of the Netherlands, it was determined that cellular automata, especially multi-dimensional cellular automata, provide efficient solutions to problems such as majority problem, where the cells are trying to 'figure out' what percentage of them are ones and what percentage are zeroes (alive or dead); the checkerboard problem, where the cells try to arrange themselves in a checkerboard pattern in a finite number of iterations; and the bitmap problem, which is similar to the checkerboard problem but uses a variety of images. According to Brukelaar and Back, this research supports the conclusion that cellular automata can be used in real world problems, including those that do not obviously relate to the idea of 'neighborhoods'.[3]

Some work is being done to determine the usefulness of cellular automata in modeling the growth of cities, the spread of a rumor, the spread of an invasive species[5], and the spread and suppression of forest fires[4]. Cellular automata are useful in modeling because they describe the behavior of groups based on the interactions of individuals, which can produce surprising, realistic, and unique results.

2.2 NetLogo

This project was written using NetLogo, a modeling environment created by Uri Wilensky which extends the Java language. Specifically, this project is written in NetLogo 4.0.3. In addition to the ability to view the interactions of cells in "real time" in NetLogo's graphics window, NetLogo also provides charts and graphs. By observing both the behavior of individual cells in the graphics window and the changes in population and temperature as shown by graphs and charts, the project's performance can be verified and results can be observed.

3 Development

3.1 First Quarter

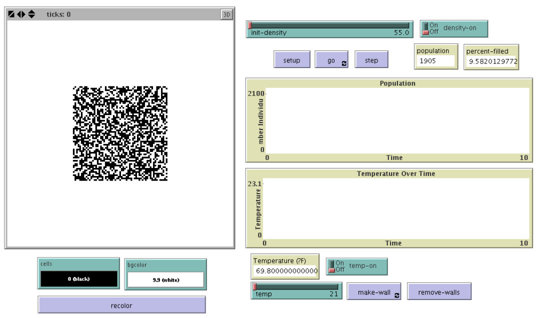
The first quarter version of the program correctly runs the cellular automata rule 14/3 (live cells with 1 or 4 neighbors survive the turn, dead cells with exactly 3 neighbors are born in the next turn) and allows the user to select the initial population density. It also graphs the percentage of live versus dead cells and provides population counts.

In test runs, the percentage and population of live cells are monitored. Variations occur due to the introduced variable of initial population density and the individual behavior of cells, which changes based on the random placement of cells on the grid. The nature of cellular modeling produces slightly different results with each run.

3.2 Second Quarter

The second quarter version of the program (Fig. 1) builds from the first quarter version and adds the factor of temperature. After an initial temperature is selected, temperature varies linearly with the changes in population (that is, the difference between the current population and the population at the end of the previous generation).

During a test run, monitors display the population count (number of live cells), temperature in degrees Fahrenheit, and percent-



ticks: 0
init-density 55.0
population 1905
percent-filled 9.5820129772
Temperature (?F) 69.800000000000
temp 21
cells 0 (black)
bgcolor 9.9 (white)

Figure 1: The NetLogo interface.

age filled of the grid (ratio of live to dead cells in the grid). Two graphs display the changes in temperature (in Celsius) and population over time. The user may also use the add and remove walls buttons to draw walls around the cells, separating a section of the population from the larger group. These walls cannot be crossed by cells and are considered 'dead' by the cells but are unable to become alive at the next generation.

3.3 Third Quarter

The third quarter version of the program begins to port the program to Java code. There are certain difficulties associated with programming cellular automata in Java; namely, due to Java's object-oriented nature, it is difficult to create an environment in which the individual cells are easily able to interact. In this program, cell interaction is accomplished via the cells being considered as belonging to a Grid class. The Grid contains a matrix in which each data point contains an individual cell, which knows its number of live neighbors and its own state.

The purpose of extending this program to Java is to allow for easier management of individual cells' attributes. In the Java program, a cell might, for instance, be able to have a virus. The spread of the virus from one cell to other cells could be observed, and testing could determine which environmental conditions best promote or best elimate the virus's spread.

The third quarter Java program is unable to run the 14/3 rule, but does allow for animation of cells on a grid. The interface, therefore, has been correctly implemented; fourth quarter development will be concerned with problems specific to cellular automata modeling.

3.4 Fourth Quarter

The fourth quarter version of the program correctly runs all "Life Like" rules, using a private variable to hold each cell's live neighbor count so that neighboring cells' state changes do not affect a cell before the end of the generation. Using the 23/3 Rule (commonly known as Conway's Game of Life), the program simulates the spread of a virus throughout the cell population. Each cell knows whether or not it carries the virus, and a cell has a 50/50 chance of passing the virus on to its neighbors. At the end of each generation, before cells change states, the program uses a PrintStream to write the percentage of live cells which are infected. This data can then be transferred to a spreadsheet to be graphed.

Two versions of the program were created. The first version displays the spread of the virus in a graphics environment and continuously updates until the user chooses to exit the program. It uses a CellDriver class which contains information about the size of the CellPanel. The CellPanel class contains the largest portion of program code; it creates a Grid which is populated with Cells and at each generation it contains instructions for the Cells to use their own class methods to check their neighbors' states and decide what their state will be at the next generation. One generation is measured as one hundred ticks on a Timer contained in the CellPanel. The CellPanel also has a Graphics class that draws the Cells onto a BufferedImage based on their position in the Grid and their states. This BufferedImage is also instructed to paint cells with the virus red so that virus spread can be easily observed (Fig. 2).

The second version uses the same Cell and Grid classes but does not use any graphics; instead results are viewed via the PrintStream which writes percentage infected data to a file. The program runs the simulation 100 times for 100 generations each, starting with zero percent of the cells initially infected and advancing one percent at a time until 100 percent of cells are infected initially. The program outputs data to a file using a PrintStream, and this data can be graphed to view the simulation's results.

3.5 Further Development

This model is very basic, having only two parameters, and could be shaped into part of a larger program using a cellular automata

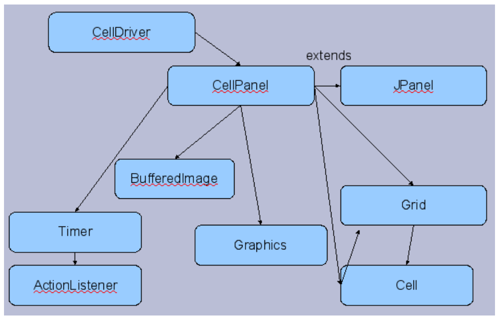


Figure 2: Java class hierarchy.

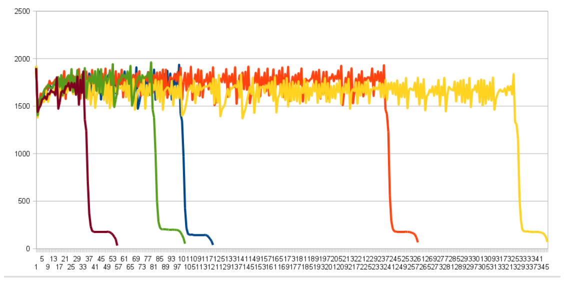


Figure 3: Results of temperature simulation showing five test runs.

population and showing its interactions with various factors in its environment such as nutrients and predators. Also, the model is not very specific; if modified to fit real-world data it could potentially model specific populations and their response to environmental changes.

Another possible extension would be trying the same model as a 3D cellular automata model using a "block" of cells which respond to temperature changes in their environment, showing diffusion through the block.

4 Results and Discussion

4.1 Results

The NetLogo program runs the cellular automata rule 14/3 (live cells with 1 or 4 neighbors survive the turn, dead cells with exactly 3 neighbors are born in the next turn) and allows the user to select the initial population density. It also graphs the percentage of live versus dead cells and provides population counts. Initial densities which are too low or too high result in small, isolated groups of individuals; the ideal initial density (shown above), in which the population grows more or less steadily outward, is about 55 percent.

With default settings (temperature of 21 degrees Celsius, population density of 55 percent), that the population eventually dies off. This may provide a (highly simplified) explanation for the phenomenon of global warming. This result is not always produced if the initial parameters are changed. The amount of time after which the population takes a sharp decline varies from run to run due to the nature of cellular automata to produce heterogeneous results. However, the trend remains the same (Fig. 3).

When walls are added, isolated groups that are very small immediately die out, whereas larger isolated groups live longer. However, when the walls are added and even after they are removed, the rest of the population of cells avoids the isolated segments of the population and grows away from the walls.

The Java program runs the cellular automata rule 23/3 (live cells with 2 or 3 neighbors survive the turn, dead cells with exactly

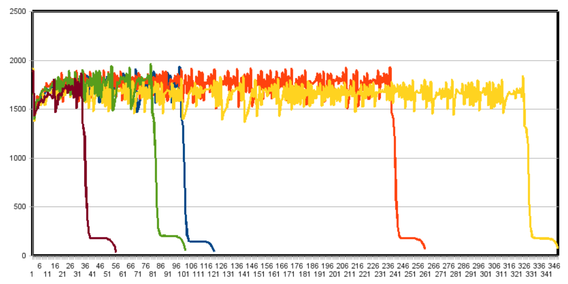


Figure 4: Results of virus spread simulation showing three test runs with different initial infected percentages.

3 neighbors are born in the next turn). This rule was chosen because it mimics the short-term behavior of a population and its movement throughout an area, so that the virus is spread based on the interactions between organisms. After 100 test runs, in which the initial percentage of infected cells was increased by 1 percent for each run, it appears that a general trend exists in which the percentage of infected cells decreases until it stabilizes at about 20 percent infected (Fig. 4). This seems to suggest that the simulation has a carrying capacity for infected cells, but it is clear that if even a small percentage of cells are infected, the virus will continue to infect cells.

4.2 Discussion and Conclusion

This project aims to model the behavior of a population based on the interactions between individuals using a cellular automata approach, in which the state of each individual at the end of a turn or generation depends on the states of its neighbors. By using the 14/3 rule, it creates a population which, under ideal conditions, grows and expands outward. This behavior realistically imitates the growth of a population of organisms. The results found in the NetLogo version of the program seem to show a situation similar to global warming, in which a population's expansion eventually creates a temperature that is too high to continue to sustain life, and the population rapidly decreases. The virus spread simulation shows that once a virus has been introduced to a population, it is very difficult to stop its spread even if only a small percentage of the population were initially infected.

References

[1] Robert Axelrod and Ross A. Hammond, "The Evolution of Ethnocentric Behavior", 2003.

[2] Lazaro Beltran-Sanchez and Doru M. Stefanescu, "Growth of Solutal Dendrites: A Cellular Automaton Model and Its Quantitative Capabilities", *Metallurgical and Minerals Transactions*, Volume 34A, February 2003.

[3] R. Brukelaar and Th. Back, "Using a Genetic Algorightm to Evolve Behavior in Multi Dimensional Cellular Automata", *GECCO '05*, June 25-29 2005.

[4] Xiaolin Hu, Alexandre Muzy, and Lewis Ntaimo, "A Hybrid Agent-Cellular Space Modeling Approach for

Fire Spread and Suppression Simulation", *Proceedings of the 2005 Winter Simulation Conference*, 2005.

[5] S. Clifton Parks, Maxim Garifullin, and Rainer Dronzek, "Argus Invasive Species Spread Model Constructed Using Agent-Based Modeling Approach and Cellular

[6] Jonathan Rauch, "Seeing Around Corners", /it The Atlantic Monthly, April 2002.

[7] "A Brief History of Cellular Automata", Palash Sarkar, Indian Statistical Institute, 2000.

[8] Srinivasa Shivakar Vulli and Sanjeev Agarwal, "Individual-Based Artificial Ecosystems for Design and Optimization", 2008.