Simulating spatial dynamics: cellular automata theory

Robert M. Itami

School of Environmental Planning and Centre for Geographic Information Systems and Modelling, University of Melbourne, Parkville, Vic. 3052, Australia

Abstract

A review of cellular automata (CA) theory is provided, with a discussion of its application in environmental modeling and simulation. CA theory shows potential for modeling landscape dynamics, so it is used to underpin a demonstration Geographic Information System (GIS) called SAGE. The grid-based GIS system is specified within the CA framework. The system is then demonstrated using an abstract simulation of vole population dynamics. A discussion of the implications of cellular automata theory on spatial analysis is provided, with directions for future research.

Keywords: Cellular automata theory; Environmental modeling; GIS

1. Spatial dynamics in planning, management and systems modeling

A great part of the challenge of modeling interactions between natural and social processes has to do with the fact that processes in these systems result in complex temporal–spatial behavior. Vegetation dynamics, surface and subsurface hydrology, soil erosion processes, wild fire behavior, population dynamics, disease epidemics, and urban growth are examples of processes that exhibit complex behavior through time. It can be a daunting task to model each component of the system much less simulate the interactions between them. Yet, with the rising awareness and concern for local effects on global climate and the need to understand how to maintain sustainable landscapes, it is more important than ever to develop theories and methods for understanding natural and human processes as complex dynamic systems. Many current techniques that planners use to characterize regions are inherently static or use mathematical formulations that, in the end, bear little resemblance to reality. This is not to say that all modeling efforts have failed in their attempts to understand real-world systems, but it is equally important to take stock of the large gap between the magnitude of environmental and social problems and the inadequacies in our current theories and methods for understanding these problems.

An important issue relates to how one conceptualizes complex natural and social systems. Systems theory tells us that complexity arises out of the simpler interactions of the components of the system. Early efforts were hampered by lack of data and primitive computing systems. However, with the advent of large-scale government data bases, geographic information systems, and remote sensing techniques, many of the limitations of earlier efforts are quickly becoming resolved. Although there are a large number of
choices in studying system behavior (see MacGill, 1986), systems theory has many advantages in the age of data-rich computing environments. MacGill (1986, pp. 1442-1443) stated: “Adoption of the systems paradigm as a framework for formal planning and analysis implies recognition that there is typically strong feedback and non-linear interaction between different components and strong dependence, possibly lagged or determined by critical or threshold values, of a system on its own history. Systems dynamics models (or dynamical systems models) are attempts to determine (predict or project ...) and understand ... the state of system components and their paths over time, by modeling explicitly the assumed dependence of particular variables on others—present and past values. The existence of successive interconnected feedback between different components means that there may be a complex web of interdependency to be picked up. The degree of refinement with which this is done distinguishes different categories of models within the systems dynamics style and, in turn, the degree to which they meet different utility criteria.”

2. Cellular automata

The spatial dynamics that typify problems in planning and management, pose a similar set of issues for the modeler. Couclelis (1985) has proposed a generic modeling framework based on discrete time cell spaced models referred to as cellular automata (CA). A cellular automaton (also called a tessellation automaton or iterative array when external input is allowed) arises by placing copies of the same sequential machine at each of the lattice points of a one-, two-, or three-dimensional array and connecting them together in a uniform fashion. As all the sequential machines undergo state transitions in parallel and simultaneously, this composition specifies a discrete time system (Zeigler, 1984).

Wolfram (1984b) listed five fundamental defining characteristics of cellular automata: (1) they consist of a discrete lattice of sites; (2) they evolve in discrete time steps; (3) each site takes on a finite set of possible values; (4) the value of each site evolves according to the same deterministic rules; (5) the rules for the evolution of a site depend only on a local neighborhood of sites around it. A typical cellular automaton begins with a one- or two-dimensional lattice of sites or cells. In a one-dimensional cellular automaton the sites are a single row of identical cells. In a two-dimensional cellular automaton the sites are composed of a matrix of identical cells regularly arranged in rows and columns. In either case each cell has a single value or state out of a finite range of values. In the simplest case the state will be either zero or one. The lattice of cell states at time zero ($t = 0$) is referred to as the initial state. In subsequent time steps ($t+1$, $t+2$, $t+3$, ..., $t+n$) the state of any cell in the lattice is a function of the cell’s current state and the state of local or neighboring cells. Often the function is expressed by summing the values of the neighboring cells and applying a deterministic rule based on the value of the cell’s current state and the neighborhood sum. This function is applied to the entire lattice of cells in the same time step (synchronously). The resulting configuration of cell values defines the state of the system in the next time step ($t+1$). In computing terms, this is referred to as a recursive algorithm.

The origins of cellular automata date back to the 1950s, when the mathematician John von Neumann, with suggestions from a colleague, S.M. Ulam, began to consider the possibility of a self-replicating machine. In an abstraction of the problem Von Neumann invented the first two-dimensional cellular automaton. Because of the enormous number of computations required in the study of these structures, cellular automata did not come under extensive study until the widespread availability of digital computers (see Burks, 1966, 1970).

A specific example of a CA model is found in John Horton Conway’s Game of Life. Martin Gardner (1970, 1971) introduced the Game of Life in the mathematical games section of Scientific American. Life is now a widely distributed public domain program available on personal computers. It is played on a two-dimensional mesh of cells. Each cell can exist in
either of two states (zero or one). The state of each cell in the next time period is dependent on the status of itself and its eight nearest neighbors in the current time period. There are three transition rules: (1) survivals—every live cell with two or three live neighbors survives into the next generation; (2) deaths—every live cell with four or more live neighbors, or less than two, dies; (3) births—every dead cell with exactly three live neighbors becomes live in the next generation.

To play the computer version of the game, the player ‘seeds’ the matrix with a pattern of live cells. When the game is run, the above rules are applied to each cell in the cellular space. After each update, the results are displayed graphically on the screen, and the entire process is repeated until the player stops the action. The compelling aspect of the Game of Life is the great variety and complexity of behavior that evolves from the three simple transition rules. The computer implementation of the game is fast enough to produce a fascinating animation as patterns of cells evolve through time and space. Colonies of cells may grow in regular or chaotic patterns, they may collapse into extinction, or they may produce the self-replicating structures that Von Neumann originally sought.

Wolfram drew a direct analogy between these four classes of behavior and attractors in continuous dynamical systems. Class 1 is analogous to a continuous system with a limit point, which brings the system to the same final state. Class 2 is like a limit cycle where a set of configurations are repeated indefinitely. Class 3 is analogous to chaotic or ‘strange’ attractors. These are the most common types of cellular automata behavior. Many Class 3 cellular automata evolve into regular fractal patterns, suggesting that cellular automata behavior in nature may generate the fractal patterns described by fractal geometry. Class 4 cellular automata are highly sensitive to initial configuration. Cellular automata in this class are irreducible; their behavior at any point in time in the future cannot be predicted using any more efficient means other than direct simulation. In addition, they tend to be irreversible. Previous configurations cannot be determined by running the rules ‘backwards’. Although the above four classes of cellular automata behavior have been deduced from study of one-dimensional cellular automata, it is highly possible that these ‘universal’ classes of behavior also apply to two- and three-dimensional cellular automata. In fact, the Game of Life has been shown to exhibit examples of all four classes of cellular automata behavior.

The question remains: do all physical and biological processes fall into these four universal classes or are there other (yet undiscovered) processes that will emerge with more extensive study of two- and three-dimensional cellular automata? In addition, are cellular automata really capable of computing and physical configuration? If so, how does one discover the rules that will generate these patterns? How are cellular automata used to simulate real physical systems?
These questions are explored in the following section.

3. Cellular automata and computer simulations of spatial dynamics in nature

Wolfram (1984a, p. 198) observed: “Much of physical science has traditionally focused on the study of computationally reducible phenomena, for which simple overall descriptions can be given. In real physical systems, however, computational reducibility may well be the exception than the rule.” He observed (p. 200): “In biological systems computational irreducibility may be even more widespread.”

Wolfram pointed out that the mathematical basis of most conventional models of natural phenomena is the differential equation. He stated (1984d, pp. vii-xii): “Whereas ordinary differential equations are suitable for systems with a small number of continuous degrees of freedom, evolving in a continuous manner, cellular automata describe the behavior of systems with large numbers of discrete degrees of freedom.”

According to Wolfram (1984a,b), the problem with complex mathematical models of natural phenomena (such as differential equations) is that in most cases, as exact quantities are not known, numerical approximations must be used. Often these approximations are used in a set of complex iterations with other equations, resulting in a loss of accuracy. In addition, these equations may provide information for overall properties of physical systems, but cannot account for the behavior of individual components of the system. Thus systems can only be described on the basis of average descriptions. Further, there are many cases where differential equations are not available and one must resort to direct simulation.

Among the advantages of cellular automata theory for modeling physical systems are the following: (1) the correspondence between physical and computation processes are clear; (2) Cellular automata models are often based on transition rules that are simpler than complex mathematical equations, but produce results which are more comprehensive; (3) Cellular automata can be modeled using computers with no loss of precision; (4) Cellular automata can mimic the action of any possible physical system; (5) Cellular automata are irreducible. There are no computations or algorithms that can speed up the cellular automata simulation. “Thus direct simulation is indeed the most efficient method for determining the behavior of some cellular automata. There is no way to predict their evolution; one must simply watch it happen” (Wolfram, 1984a, p. 198).

In other words, cellular automata models implemented on computers may serve as a framework for modeling complex natural phenomena in a way that is conceptually clearer, more accurate, and more complete than conventional mathematical systems. Application of cellular automata has been explored primarily in the physical and natural sciences at a wide range of scales. Maddox (1987) reported on cellular automata models of galaxy formation. At the other end of the scale, Wolfram (1984a) demonstrated the use of cellular automata simulations in the formation of snowflakes and turbulent flows in fluid dynamics (Wayner, 1988). Vichniac (1984) has shown that cellular automata may be used to simulate a wide range of processes in physics including percolation and nucleation. Yakowitz et al. (1989) reported on the development of cellular automata models of disease epidemics in human populations. More recent work has focused on the cellular automata within an information theory context, statistical properties of cellular automata and the study of lattice gases, turbulence and fluid dynamics (see Manneville et al., 1989).

Signorini (1989) made the important distinction between conventional use of computers for calculation and cellular automata computers as experimental frameworks. He stated (1989, pp. 62-63): “The use of CA in physics implies a reevaluation of the epistemological status of simulation. The cellular computer is not used uniquely for its computational capacity, as is typically the case for general purpose computer, but particularly for its capacity as an experimental environment for abstract or real phenomena.
of ordinary physics. Such an environment blurs the distinction between performing a calculation and performing an experiment. Moreover, even though the evolution of a simulation is deterministic, it is not predictable: new organizational devices or new possibilities for calculation present themselves during experimentation by simulation."

Cellular automata may therefore be seen not only as a framework for dynamic spatial modeling but as a paradigm for thinking about complex spatial–temporal phenomena and an experimental laboratory for testing ideas. When this concept is combined with the fact that cellular automata have also provided a framework for parallel computing machines, it is easy to see the enormous opportunities inherent in the concept. Fredkin (see Wright, 1988) has taken this concept to its extreme. In his information theory of physics, Fredkin believes that information is more fundamental than the particles of physics, and that cellular automata can describe the universe with precise precision because, at its most fundamental, Fredkin believes the universe is a cellular automaton. Though this view may be extreme, the general concept is that an algorithmic or computational description of nature (e.g. cellular automata) as compared with an analytic or equation-driven approach (e.g. differential equations) may yield a more realistic understanding of the mechanisms which drive change in natural systems.

The advantages of cellular automata to spatial dynamics may be evident; however, the implementation of the concept in a geographical sense remains problematic. The next section will discuss the implications of cellular automata theory on the development of geographic information systems (GISs).

4. Cellular automata in geographic information systems

To specify a geographical implementation of cellular automata, it is useful to review the formal definition of the framework from a systems theory viewpoint. A cellular automaton is a discrete time system in a spatial context. It is achieved by placing copies of the same sequential machine at regular intervals on a lattice. At the time interval \( t \), all of the sequential machines undergo a state transition simultaneously. This allows a model to achieve spatial uniformity as well as time uniformity (Zeigler, 1984). Formally, a cellular automaton is described as follows:

\[
Q = \langle S, N, T \rangle
\]

where \( Q \) is the global state of the system, \( S \) is a set of all possible states of the cellular automaton, \( N \) is a neighborhood of all cells that provide input values for the transition function \( T \), and \( T \) is a transition function that defines the change in state of the cellular automaton from its current state to the state in the next time step \( (t + 1) \).

Helen Couclelis (1985) has generalized the concept of two-dimensional cellular automata from the viewpoint of discrete systems theory following the framework described above for modeling and simulation (Zeigler, 1984). In the generalization, Couclelis showed that each of the limitations inherent in simple cellular automata models may be relaxed. This generalization is summarized below with the relationship of the modeling framework to existing GIS systems:

1. The space need not be infinite, nor do the spatial units need to be square cells. Spatial units may be hexagonal, irregular polygons, or a set of points. Couclelis referred to these spatial units as 'local components'. In a vector GIS local components may be points, lines or polygons. In a grid-based GIS local components may be composed of single cells or of regions of cells that may be either contiguous or fragmented.

2. The neighborhood (or 'influencer set') does not need to be defined in the same way for each local component. The neighborhood may be defined differently for every local component the array. In addition, the 'neighborhood' need not necessarily be composed of adjacent spatial units, but may be a set of units spatially detached from the local component. GISs allow for considering
three types of neighborhoods; (a) modeling context of single cells by overlay on corresponding cells on another map overlay; (b) modeling of adjacent cells through proximity functions; (c) single cells, effects on which may be modeled using cells on different map layers (inputs) or across the matrix (non-adjacent cells).

(3) The ‘transition functions’ may be different for each local component in the matrix and rules may be applied at different time intervals. The GIS has a set of spatial, mathematical, or logical operators, which are a part of the GIS analysis system. In a GIS model these operators (boolean, arithmetic, spatial proximity, etc.) are used to model or describe the relationship of the different map layers to a given condition or problem. In other words, transition functions could be implemented using existing GIS operators. In addition, the results of these transition functions are limited only by the numerical precision of the GIS package. Transition functions do not necessarily need to be deterministic. By introducing random functions in transition rules, one is able to produce a probabilistic cellular automata (Wolfram, 1984c).

(4) Both inputs to and output from the simulation model may be defined. Different map layers are all superimposed on the same grid system as the local components above. This allows links or ‘channels’ to corresponding cells on other map layers. If environmental monitoring systems are used to update spatial data bases, inputs to the model may be made in real time.

From the formalism specified above, it is possible to identify the characteristics in current GISs which support CA modeling:

(1) Global state—Q. The global state of the environment is represented in a GIS by a set of map overlays such as elevation, vegetation, soils, or land use which together describe the state of the environment at a given point in time. The problem is that different map overlays may represent the environment at different points in time. This poses difficulties for CA models, as the initial state is not defined consistently across map layers. As CA models are sensitive to initial state, data base construction in a GIS may require additional care when used with CA models.

(2) States—S. Many cell-based GISs and especially cell- or grid-based GISs are discrete, that is, they represent spatial attributes as a set of integers. This limits the analytical capability for modeling and simulation by limiting the number of possible states a cell can have. By increasing the numerical precision of the GIS by allowing floating point numbers, the functionality and utility can be greatly enhanced. Several cell-based GISs have added this capability, including IDRISI and ArcInfo’s GRID system. An example of where this additional numeric precision is useful is in digital elevation models. Integer elevations in meters may have too low a precision to depict accurately micro-terrain features. In addition, calculations such as slope and solar aspect cannot be made with the accuracy required by some applications.

(3) Transition functions—T. Although there are many GIS operators that allow for a large combination of transition rules, these are often ‘hard wired’ into the GIS, providing little flexibility for the user. A more open, programmable environment, which allows the user to design new transition functions is required. Tomlin’s (1990) ‘Map Algebra’ describes spatial programming language that is flexible and open ended. Map Algebra has been recently adopted by many commercial GISs. The algebraic syntax used by Map Algebra provides a great deal of flexibility for the user. By adding random functions, probabilistic cellular automata could be modeled.

(4) Neighborhood—N. The concept of a neighborhood of input regions is easily generalized in a GIS from immediately adjacent cells (as in the Game of Life) to three classes of neighbors already recognized in most GISs (although not explicitly within the context of discrete dynamic systems). The first class of neighbors, or what Couclelis referred to as ‘influencer sets’, is the same class of neighbors as specified in the Game of Life. In a cell-based GIS these are defined by the cell and its immediate neighbors. In Tomlin’s original MAP package, the operator SCAN implemented this type of neighborhood by allowing the user to specify the diameter and shape of the neighborhood (either square or round) and a limited set of operators (or tran-
tion functions) that were applied to these neighbors. In Map Algebra syntax these type of neighborhoods are implemented using the 'FOCAL' function. The second class of neighbors in a GIS are defined by membership in a region. In this type of neighborhood, the boundaries are not defined by adjacency, but by contiguous regions on the map. These regions may then be typified according to the Coudcelis (1985) generalization, which allows for influencer sets to be cells that are not adjacent. Current operators such as intervisibility, and distance functions allow for influence to extend across the matrix. However, new, more open methods of communicating information across the matrix should be explored. One method may be the introduction of networks to raster-based systems.

This analysis shows that cell-based GISs may indeed serve as a tool for implementing cellular automata models for the purposes of geographic analysis. Clearly, GISs have some important components that make them good candidates for use as non-autonomous sequential machines; however, current GISs have some limitations that make the concept hard to implement. A major limitation in most GISs is the lack of support for iteration. Iteration allows for repeating sequences of instructions in a program. With the exception of some primitive operators, such as distance, visibility and neighborhood functions, recursion is not a feature that is commonly available to the GIS modeler. This is an essential feature in cellular automata modeling. Current GISs are not designed for the rapid iterative command sequences of the type required in cellular automata modeling. However, even without iterative command processing capabilities, cellular automata models may be simulated using current GISs by creating batch files that contain the iterative command sequence. Clearly, existing GISs already have many of the characteristics needed to implement cellular automata models. In the discussion below, the development of a cell-based GIS cellular automata program is discussed.

5. Modeling language. Once the above components are implemented, an extension of the existing command structure of GIS systems needs to be enhanced. This should be in the form of a programming language that will support iteration, branching and looping, and callable functions.

The above discussion outlines a simple specification for the modification of existing cell-based GISs to support cellular automata simulations. The following discussion outlines the development of a prototype cellular automata based GIS that implements the above specification based on an existing GIS (Map Analysis Package, Tomlin, 1980).

5. Work toward incorporating spatial dynamic capabilities into cell-based GISs

SAGE (Itami and Rattlings, 1993) is a raster or grid GIS based on Tomlin's Map Analysis Package (MAP). SAGE uses the same command structure as MAP but, in many cases, improves the functionality of operators. In addition, better debugging sequences, more flexible means of file handling and better error trapping routines are provided for user input. SAGE is currently under development and will be implemented in three phases.

Phase 1 of SAGE has been to expand the capabilities for analysis by adding support for single precision floating point cell values. All the original operators in MAP had to be rewritten to support the higher precision. Second, additional operators have been implemented. The first is a complete algebraic equation processor (MATH) which works both on scalars (integer and floating point numbers) and maps as variables in algebraic expressions. This operator supports arithmetic operators (addition, subtraction, division, negation, integer division, modulo division, and exponentiation), trigonometric functions (arctangent, cosine, sine, and tangent), functions (absolute value, conversion to integer by rounding and by truncation, square root, and random number generation), constants (pi, row index, column index, number of columns and number of rows, and total number of cells), and user-defined variables. In addition, a BOOLEAN operator and ZONE operator have been
added to SAGE, which implement the full set of ZONAL functions described by Tomlin in his Map Algebra (Tomlin, 1990). Enhancements include improved support for color graphic output in plan and perspective, including user-defined color palettes, color tables for individual map overlays, and true hidden line removal. An improved user interface includes a command line parser which checks for correct syntax and a line editor which allows commands to be edited before execution. The viewshed operator (RADIATE) has been improved to produce more accurate results and greater flexibility in defining view direction. The proximity function (SPREAD) has been enhanced to handle floating point impedance maps, and region labeling for surfaces. The terrain operators have been enhanced. Slope may be calculated using three options: maximum slope, average slope, or maximum downhill slope. Analytical hillshading has been added. Output functions include file output for legends, ASCII cell values, and tabular reports of two or more maps showing corresponding cell values. A session log file is automatically created which records all transactions during the session. This facilitates debugging and provides a complete record of interactive sessions. Phase I is now complete.

Phase II will implement program control statements (iteration and branching) and a graphical user interface, and include some new global operators that will return global minima, maxima, means, standard deviations, and other summary values for an overlay. In addition, a new local operator which allows the user to specify calculations on neighborhood values will be implemented. This will provide a complete programming environment to implement CA models.

Phase III will implement an object-oriented modeling environment for SAGE within a graphical user interface.

6. A demonstration of cellular automata models: population dynamics of rodent populations

Two examples will serve to demonstrate the implementation of the cellular automata frame-
In the two experiments described here, the modeling assumptions are kept the same; however, only one of the three growth hypotheses is demonstrated, as shown in Fig. 1. The hypothesis is that vole populations do best at low (but not too low densities) and drop off at very low and increasingly high densities. In addition, the rule is applied for two-dimensional cellular automata rather than for one-dimensional cellular automata. To transform Couclelis' rule from a one-dimensional to a two-dimensional cellular automata simulation, the neighborhood is expanded from a three- or five-cell linear neighborhood to a nine-cell square neighborhood (the cell and its eight nearest neighbors). In Experiment 1, the totalistic rules are implemented for three-dimensional cellular automata with no environmental constraints. In Experiment 2, the same totalistic rules as in Experiment 1 are used; however, environmental constraints are added to the model to demonstrate, in a simple way, how environmental factors might be incorporated into CA models within the existing GIS framework.

6.1. Model 1: vole population dynamics without environmental constraints

State variables

In the first model, a nine-cell square neighborhood is arbitrarily selected. Each cell can take on one of four states: State 0 is a vacant cell; State 1 is low-density vole population; State 2 is medium-density vole population; State 3 is high-density vole population.

Transition rules

To implement the totalistic CA rule, each cell in the map matrix is visited and the state value of the cell and its eight adjacent neighbors are summed. The value of cell in the next iteration of the model is based on the rule shown in graph form in Fig. 1. Cells that have neighborhood sums less than four or greater than 22 are assigned a value of zero in the next iteration. If a cell has a neighborhood sum in the range of 4–5 or 18–22, the cell is assigned a density value of one. Neighborhood sums of 6–7 and 13–17 are assigned a density value of two. Neighborhood sums of 8–12 are assigned a density value of three.

Fig. 1. Graph showing totalistic rule for determining vole density levels (Level 1, 2 or 3) assigned according to the sum of the density levels for a cell and its eight neighbors.
The implementation of these rules for the first iteration in the SAGE package is shown in the sequence in Fig. 2, where, in Line [1], VOLEs is the map that represents the 'initial state' of the model. The operation SCAN moves through the VOLEs matrix one cell at a time and evaluates the eight nearest neighbors in the adjacent cells (the 'diagonally' option evaluates a square neighborhood of nine cells). The optional phrase SUM indicates that the operation should sum the values in the nine-cell neighborhood and assign the resulting value to the equivalent cell in the COUNT output map matrix. The resulting matrix will have values that range from zero (no 'live' neighbors) to 27 (all nine cells have density value three). In Lines [2]–[9] the transition rules are implemented using the RECODE operation, which allows the user to assign specific output values to specific values in the input map. In this example, the values in the COUNT map recoded according to the rules specified in Fig. 2. In Line [3], cells with neighbors that sum to values less than four are encoded with the value zero. In Line [4], cells with neighbors whose sum ranges from four to less than six are assigned a value of one. The ampersand (&) at the end of Lines [2]–[8] is a continuation marker indicating that the RECODE operation continues on the next line. When the operation is complete the results are stored in the new map VOLEs1. If this pair of commands is executed repeatedly, the configuration of vole densities varies in a similar fashion to the Game of Life.

**Initial state**

One of the advantages of the two-dimensional cellular automata is that it is possible to run more than one initial state at a time, by distributing different patterns across the map. The initial state for the model is shown in Fig. 3. Eight initial configurations of cells are tested within the map space of 3600 cells (60 rows and 60 columns). Fig. 3 shows the nine configurations, each centered in a 20 by 20 block of cells. Groups I–III (top row) in Fig. 3 are all randomly generated patterns of cells in a nine-cell neighborhood (three rows by three columns). This row will be an experiment of the rule as applied to random patterns of different density configurations. Group I is a low-density configuration composed of a pattern of zeros and ones (a binary state configuration). Group II is a pattern of zeros, ones and twos generated randomly, and Group III is a pattern of zeros, ones, twos and threes.

The second row shows three symmetrical configurations. The first patterns have bilateral symmetry, the third, radial symmetry. Group IV is a pattern of ones and twos. Group V is a pattern of zeros, ones and twos. Group VI is a pattern of zeros, ones, twos, and threes. This will allow experimentation on the effects of the deterministic rule on symmetrical configurations.

The last row of patterns shows two randomly generated patterns of ones, twos and threes in two different sized initial configurations. Group VII is composed of 16 cells (four rows by four columns) and Group VIII is composed of 15 cells (three rows by five columns).
**Model iterations**

The simulation for this experiment was run for 100 iterations. Fig. 4 shows the results for time steps 1–50 and every five iterations from 55 to 100. Fig. 5 shows a plot of the total number of cells in each density category for each time step.

6.2. **Observations of Model 1 simulation**

**Overall behavior of the model**

In two of the eight initial configurations—Groups I and IV—the simulations quickly evolved to a homogeneous state (extinction), in Time Step 1 and 2, respectively. In all other cases, the configurations evolved into Type 3 behavior in Wolfram's classification (evolution leads to chaotic aperiodic pattern). This compares well with the results of Couclelis' one-dimensional cellular automata model. The two-dimensional model has characteristics that are not evident in the one-dimensional model. These deal with the effects of symmetry, information transfer, edge effect, local cyclical density patterns and equilibrium in chaos. Each of these characteristics will be described individually.

**Symmetry**

In Groups V and VI two types of symmetry in the initial configuration were tested. Group V evolved in one time step \((t = 0\) to \(t = 1\)) from bilateral symmetry to radial symmetry. This symmetry was maintained until the configurations began to merge at Iteration 11 of the model \((t = 11)\). Because of the differences in initial configuration, however, both Group V and Group VI evolved in completely unique ways. In both cases, the symmetry was not maintained, as the patterns were 'swept over' by the influence of the random configurations. By Time Step 25 all signs of symmetry are gone.

**Information transfer**

In all configurations the maximum rate of spread across the uninhabited areas of the map is one cell per time step. This is referred to as the 'Speed of Light' in the Game of Life. It is an artifact of the nine-cell neighborhood. This rate of spread would increase in direct relation to the radius of the neighborhood. It is therefore possible to predict the exact time step when the evolution of one pattern will influence the evolution of another pattern by simply counting the number of cell spaces between the closest edges of each pattern and dividing by two. This is made particu-
Fig. 4. Results of Model 1 for 100 iterations. Results show examples of local extinctions, edge effects, equilibrium and chaotic behavior.
larly clear by studying the influence of evolution of random configurations on symmetrical configurations (study Groups VI and VIII). This has direct implications on spatial diffusion models and information models in geography.

**Edge effect**

In all eight groups, as the configuration expands to a critical mass \((t=3)\), an obvious 'edge effect' emerges. In each case, great diversity in density classes emerges along the edges of vacant areas. This is true both along the perimeter of the configuration, and within the configuration as the result of local extinctions.

**Local cyclical density patterns**

This effect is easiest to observe in the symmetrical patterns, but is evident in all configurations. Beginning with Time Step 4, a cyclical pattern of density emerges within the perimeter of each configuration. A fairly homogeneous region of low-density cells develops (see Group VI, \(t=4\), and Group V, \(t=5\)). This pattern quickly evolves (in one time step) to a core area of high-density cells surrounded by a field of medium-density cells. In the next time step, the high-density regions experience a population crash, resulting in local extinctions. This creates an opening which creates a strong edge effect in the next time step. The opening eventually is filled and the cycle repeats itself. This cyclical pattern has a striking resemblance to forest dynamics and especially 'Gap' models. It is easy to imagine the final stages of the model as a forest mosaic.

**Equilibrium in chaos**

By Time Step 39, the entire map is overcome by the growing configurations. From the graph in Fig. 5 one can see that even though the overall pattern of evolution is chaotic, the total population is in equilibrium. The three density levels also maintain orderly behavior in relation to each other, with medium-density cells being most frequent, followed by low-density, high-density, and finally vacant cells.

6.3. Model 2: Vole population dynamics with environmental constraints

**State variables**

The state variables for Model 2 are identical to those for Model 1.
**Fig. 6.** The HABITAT map showing four levels of habitat quality. This map was used to constrain Model 2.

<table>
<thead>
<tr>
<th>1</th>
<th>SCAN VOLES SUM DIAGONALLY FOR COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RECODE COUNT FOR VOLES1 &amp;</td>
</tr>
<tr>
<td>3</td>
<td>ASSIGN 0 TO 1 UPTO 4 &amp;</td>
</tr>
<tr>
<td>4</td>
<td>ASSIGN 1 TO 4 UPTO 6 &amp;</td>
</tr>
<tr>
<td>5</td>
<td>ASSIGN 2 TO 6 UPTO 8 &amp;</td>
</tr>
<tr>
<td>6</td>
<td>ASSIGN 3 TO 8 UPTO 13 &amp;</td>
</tr>
<tr>
<td>7</td>
<td>ASSIGN 2 TO 13 UPTO 18 &amp;</td>
</tr>
<tr>
<td>8</td>
<td>ASSIGN 1 TO 18 UPTO 23 &amp;</td>
</tr>
<tr>
<td>9</td>
<td>ASSIGN 0 TO 23 UPTO 28</td>
</tr>
<tr>
<td>10</td>
<td>MINIMIZE VOLES1 VERSUS HABITAT FOR VOLES1</td>
</tr>
</tbody>
</table>

**Fig. 7.** SAGE command sequence for first iteration of Vole Model 2.

**Transition rules**

The totalistic transition rules for Model 2 are the same as for Model 1 except that an additional constraint is added. In this model, a second map overlay called HABITAT is used to alter the population density by placing constraints on growth. Fig. 6 shows the HABITAT map. Four levels of quality are shown (zero is poor, one is low-quality, two is medium-quality and three is high-quality habitat). The environmental rule placed on the model follows the rationale of 'the most limiting factor'. In this case, the HABITAT overlay is interpreted thus: poor-quality habitat cannot support any voles; low-quality habitat can only support low-density vole populations; medium-quality habitat can support low- or medium-density populations; high-quality habitat can support low-, medium- or high-density populations.

To implement this rule the sequence in Lines [1]–[9] of Fig. 1 are implemented without change; however, an additional operator has been added to implement the constraints described for the HABITAT overlay.

The implementation of these rules for the first iteration in the SAGE package is shown in the sequence in Fig. 7, in which Lines [1]–[9] are identical to those in Fig. 2. Line [10] implements the environmental constraint rule. The MINIMIZE operator selects the minimum value for each corresponding cell in the VOLES1 map overlay and the HABITAT map overlay. This has desired effect of dampening population densities according to habitat constraints.

**Initial state**

The initial state for Model 2 is identical to that of Model 1.

**Model iterations**

The simulation for this experiment was run for 100 iterations. Figs. 8(a) and 8(b) show the results for Time Steps 1–50 and every five iterations from 55 to 100. Fig. 9 shows a plot of the total number of cells in each density category for each time step.
Fig. 8. Results of Model 2 for 100 iterations. Results show effects of the HABITAT quality overlay on population densities.
6.4. Results of Model 2

Model 2 has similar characteristics to Model 1, with the exception of the three areas where the HABITAT map constrains the density. As expected, the poor-quality habitat sites remain without voles for all 100 iterations. Less obvious is the result for the low-quality habitat area. In this case, the region filled uniformly with a low-density vole population by $t=20$ and then remained in steady state throughout the remaining 80 iterations. The third region, moderate-quality habitat, presents the most interesting behavior. This region filled with low- and medium-density sites by Iteration 19 and then maintained a constantly fluctuating pattern of these two densities for the remaining iterations. Once this region was filled there were never any cases where vacant cells emerged.

Model 2 shows the effect of environmental constraints. In the case of low- and medium-quality habitat, the diversity of density classes decreased. The system went into steady state early, but as a monoculture in the case of low-quality habitat, or as a dynamic equilibrium in the case of medium- and high-quality habitat.

6.5. Summary

This demonstration has not been a rigorous test of vole population dynamics. Instead, it shows the principle of cellular automata models in being able to explain complex behavior from the viewpoint of simpler deterministic rules that are applied recursively across time and space. The real world is not a flat homogeneous plain. Nor do voles, or people, live in a world without competition for space and food resources. However, these variables can easily be added using standard existing GIS operators. The cellular automata theory is refreshingly simple, yet the implications are incredibly profound. With the implementation of cellular automata theory into the existing GIS framework, a new paradigm for thinking about landscape evolution emerges. Some of the implications of this approach are discussed in the conclusion of this paper.

7. Implications: prediction vs. scenario testing

This paper has introduced cellular automata theory and has demonstrated its application within the context of geographic information
systems. It has described current efforts to enhance existing GIS software in a program called SAGE to facilitate the use of cellular automata theory in the context of applied planning and resource management systems. It is important, however, to mention and recognize the implications of the framework. As with any modeling technique, the utility of a simulation is limited by the reliability of its performance. Because of the deterministic nature of cellular automata models (i.e. the behavior of the system being the result of defined rules of the behavior of the components), there are clear challenges associated with operationalizing this framework for real-world problem solving. The most fundamental problem revolves around the issue of deterministic modeling and its relationship to evolution and chaos theory. By definition, in a chaotic system, behavior is deterministic and small changes at the micro levels can result in dramatic and unpredictable changes at the macro level.

Although cellular automata models may be ultimately capable of simulating any physical system, their great sensitivity to initial state and the great variation of possible outcomes make them perplexing to study. The exploration of even simple two-dimensional cellular automata with the interaction of only four right-angle neighbors (referred to as a Von Neumann neighborhood) is daunting without the aid of a computer. In a system of four neighbors and two states (zero and one) there are more than 65,000 possible rules. In a system of eight neighbors (referred to as the Moore neighborhood) there are $10^{77}$ possible rules. The Game of Life represents a binary state two-dimensional cellular automata with only one out of thousands of millions of possible rules, yet it has spurred years of study by mathematicians (see Wainwright, 1974; Berlekamp et al., 1982).

The application of cellular automata models to real-world environments represents a considerable research challenge. The nature of the theory has implications at both the technical and the conceptual level. At the technical level, this paper has already shown how cellular automata theory can guide development of GISs. Cellular automata theory has already been implemented in image processing, and there are rich possibilities for application of cellular automata models of environments on parallel processing computers.

At the conceptual level, many mathematically based models of spatial interaction can be recast within the simpler framework of cellular automata models, making these procedures more accessible to practitioners. Couclelis (1985) supported the idea of generic specification of discrete time systems. The idea that different models can be formulated within a common modeling framework leads to cross-fertilization between disciplines and the discovery of exciting linkages. MacGill (1986) has observed that models which generate global states from local states provide considerable improvement in model realism and greatly assist the modeler in understanding the mathematical nature and capability of their modeling tools. There is significant advantage in a modeling system where many different simulations can be run to identify critical values and gain insight into the functioning of the system. In other words, these models can be viewed very much like a flight simulator for training pilots. They do not predict changes in the real world, but have considerable validity and utility in learning about the real world. MacGill (1986, p. 1445) concluded that this type of modeling is "perhaps a more legitimate view of the nature of modeling than earlier, possibly more naive, predictive applications of ill-understood mathematics".

CA models have been explored by researchers primarily in geography mathematics and ecological modeling. Tobler (1970) proposed the use of cell space models for modeling geographic interactions. Green et al. (1989) have suggested the use of CA as a generic tool for modeling landscape dynamics. Hogeweg (1988) has developed a similar argument for use of CA theory in modeling landscape dynamics. Vicsek and Szalay (1987) demonstrated the use of three-dimensional CA models to describe the distribution of galaxies.

Discussion of cellular automata theory as applied to environmental issues is not complete.
without mention of closely related approaches that have been explored by others. Fractal geometry has been used in the study of urban growth and form (Batty et al., 1989) and in habitat design (Milne, 1991). These studies use differing techniques (diffusion limited aggregation and iterated function systems, respectively) to generate patterns of spatial evolution with fractal characteristics of self-similarity at a range of scales. Tobler (1970) has demonstrated the use of cellular geography to show the effects of proximity on simulation of urban population growth. Gardner et al. (1987) have used probabilistic cellular automata to study land use change in Georgia. Yakowitz et al. (1989) and Gani (1989) have demonstrated the use of cellular automata and percolation lattices in modeling of epidemics. A useful reference that shows the relationship of fractal geometry, percolation lattices and cellular automata may be found in Gould and Tobochnik (1988). From a conceptual point of view, all these approaches are used to study and simulate the behavior of complex spatial behavior based on the interaction of simpler elemental processes. The other aspect in common is that they are all sensitive to initial state and the transition functions that define their behavior, and because of this may not have the capacity of precise prediction. Instead, emphasis is placed on understanding of processes and a fundamental level. Engelen (1988, p. 45) expressed a common view to the utility of these simulation models: “The model ... should be seen as a tool to explore the different possible futures of the system. Such an explorative approach contributes to the design of actions to pilot the system so as to avoid the worst and hopefully take the most desirable course possible. This way catastrophes can perhaps be avoided and the impact of interventions tested in an interactive way between the planner and the modeled system.”

Cellular automata theory, because of its close relationship to existing GIS methods and its great potential for modeling complex spatial dynamics, presents a great challenge and opportunity for the environmental planning community. Although early explorations appear to be promising, much more work needs to be done to develop functional cellular automata models. The development of tools such as SAGE can contribute to progress in this effort.

References

Manneville, P., Boccara, N., Yichnaiac, G.Y. and Bidaux, R.,


