Simplifying spatial complexity in ecological models: Cellular automata

5.1 Introduction

5.1.1 Why cellular automata?

The models of classical, non-spatial dynamics (Pearl-Verhulst model of logistic growth or Lotka-Volterra model of population interactions for example) are based on three fundamental assumptions:

- populations consist of a large number of individuals (abundance assumption);
- all individuals are identical (uniformity assumption);
- the movement of the individuals is such that there is a perfect spatial mixing of the system (ergodicity assumption), meaning that each individual "feels" the same environment around him because individuals move fast and independently from one another.

Although these models yielded deep theoretical insights into many aspects of population processes, these assumptions do not often apply in biology. The ergodicity condition is violated by sessile individuals such as plants for example, which can interact with close neighbors only. They practically do not feel the presence of individuals outside a limited area around them. Even populations of individuals capable of very fast movement do not perfectly mix. Thus, an important question is: how to formalize the biological reality that individuals do not interact in the same way with all their conspecifics, but mainly interact with their nearest neighbors?

5.1.2 Basic properties of cellular automata

A simple and useful method for modelling population dynamics in a spatially explicit way is to use lattice or cellular automaton (CA) models. The principle is as follows. An ecological system is divided into a grid of cells. Each cell is in one of several particular states. At each time step,

the state of each cell is evaluated and compared to the states of its neighbors. Its state may then change from one to another according to a set of rules. Preliminary model elements to define are thus:

- the size of the cells (depending on the questions addressed by the model)
- the size of the whole system (the number of cells)
- the different possible states of the cells (quantitative, e.g. amount of biomass, or qualitative, e.g. presence or absence of individuals)
- the type of neighborhood (4 or 8 neighbors for example)
- the rules of transitions from one state to another
- the time step (depends on the biology of the system and on the questions addressed; 1 day or 1 year for example).

5.2.3 Brief history

Cellular automata as a theme in scientific computing cannot be fully understood without knowing anything about the history of CA within theoretical mathematics, computer science and artificial intelligence. Here, we give a brief overview. The history of CA more or less parallels the history of computers. The field of cellular automata was created by John von Neumann (without computers at this time!). In the 1940s-1950s John von Neumann wondered what exactly constituted life. According to him, a short answer to this question was: reproduction. Anything that lives is able to reproduce or to self-replicate, i.e. to make copies of itself. Therefore, if one succeeded in building a machine, or 'automaton', that was capable of self-reproduction, one could claim that the machine was alive. Von Neumann never succeeded in building an actual self-reproducing machine.

The mathematician Stanislav Ulam proposed to build a virtual self-reproducing automaton instead, using mathematical constructions rather than nuts and bolts. Consider an infinite grid, laid out like a chessboard. Each square of the grid can be seen as a 'cell'. Each cell on the grid is a separate finite state machine, acting on a shared set of rules. The configuration of the grid changes as discrete time steps tick off. Every cell holds information that is known as its state, and at every time step it looks at the cells around it and consults the rule table to determine its state in the next time step.

The machine defined in this way became known as a cellular automaton. In 1953, John von Neumann presented a series of lectures in Princeton on a self-reproducing CA that he designed. However, this CA was by no means simple: it had 200,000 cells and 29 possible states per cell. In 1984, Chris Langton succeeded in constructing a self-reproducing CA that requires only 8 different states. Cellular automata become very famous in 1970, when the mathematician John Conway invented the "game of life". In this game, cells can be in one of only two possible states: dead or alive. At each iteration, all eight neighbors of each cell are inspected. The rules of the game are very simple: a dead cell becomes alive if there are exactly three out of the possible eight neighbors alive; an alive cell dies if there are less than two alive neighbors (loneliness), or more than three (overpopulation). It is mainly due to the popularity of the game of life that people started working with CA, resulting in the numerous applications of CA and CA-derived models that are used in natural sciences today.

In summary, a cellular automaton (CA) is any set of individuals or cells with a finite number of characteristics - also called states - that change over time according to certain rules. These rules depend on the previous state of each cell and on the states of the cells with which each

cell interacts. Compared to the models presented in the previous chapters, space is subdivided into cells, which can represent one individual or one spatially homogeneous unit (bare soil, area occupied by vegetation...). Time is also described as a discrete variable. CA moreover allows to include stochasticity, or chance, in models.

5.2. Application: a forest fire model

In Mediterranean and boreal forests, fire is a regular factor that determines the composition and spatial structure of vegetation. After a fire in an area, different sequences of plant communities succeed one another. The occurrence of another fire sets the vegetation to the first stage of succession. Not all parts of a forest burn in every fire. Thus forests are composed of a mosaic of patches that are at different stages of succession. The purpose of this first exercise is to give a concrete example of a cellular automaton, and to learn how to implement it in MATLAB. At the end of the exercise, the notions stochastic/deterministic and synchronous/asynchronous should be clear.

5.2.1 The model

The size of the modelled area is 100×100 cells. Each cell corresponds to a few ha of forest. For the neighborhood, we consider the four nearest neighbors (above, below, right and left). The stage of a cell is described by its age since the last fire. There are three possible qualitative classes for the age: young vegetation, mature vegetation and old vegetation. At each time step, the vegetation goes from one category to the other until it reaches the old vegetation stage. If a square burns, the cell is considered to be empty at the next time step. An empty cell is automatically recolonized by young vegetation at the following time step. Fire is usually caused by lightning or other unpredictable factors. Therefore, the occurrence of lightning is described by a parameter set to $n = 5$ strikes per time step. The position of a lightning strike is chosen at random. If a square is hit by lightning, the probability that it starts burning is given by the parameter f. Flammability is supposed to increase with the age of the vegetation. The four cells adjoining a burning cell can also burn with a probability f.

5.1.2 Spatially explicit simulations

A global diagram of the program is provided in Appendix A (at the end of this chapter). There are 5 possible states in this model:

* empty
* young vegetation
* mature vegetation
* old vegetation
* burning

The model is initialized as follows. A 100×100 matrix is created. Each cell receives the initial state *empty* (value of the cell = 1) or *young vegetation* (value of the cell = 2) at random according to a defined proportion (p_v is the initial proportion of vegetated cells in the lattice).

A simulation is carried out by choosing $n = 5$ sites at random in the lattice and then determining whether they burn. The flammability f of the sites is calculated. For each of the $n = 5$ cells, a random number is drawn and then compared to f. If it is smaller than f, the cell burns (value of the cell = 5). This step is said to be **stochastic**. A process is stochastic when the next state in not fully determined by the previous state. Here, we could decide that if f is greater than a certain

value, the cells burns, otherwise it does not. In this case the process would be **deterministic**. But by first drawing a number at random and then comparing it to f to determine if the cell burns or not, we introduce chance in the model: even a cell with a very high f might not burn and vice versa. If at least one cell burns, the fire might propagate in the lattice. This step is explained in exercise 3.

After the fire has stopped, the state of the cells is updated: burned cells become empty, empty become young vegetation, young become mature and mature become old (this process is deterministic).

The first and last column and the first and last row of the lattice are supposed to be empty and are not updated (they remain empty).

Note that there are two different time scales considered in this model: the time scale of the fire and the time scale of the vegetation growth.

5.1.3 Exercises

The programs needed to analyze the model can be downloaded from Blackboard. Answers to the questions have to be provided in a file including figures (with legends: title and parameter values used to obtain the figure). For each question, draw a diagram explaining the structure of the MATLAB program you wrote (use Appendices A and B at the end of this chapter). Put the two final programs *ACForestFire1Year.m* and *ACForestFireSevYear.m* in appendices. Send also the corresponding .m files.

1. Let a be the age of the vegetation. Let a be 2 for young vegetation, 3 for mature vegetation and 4 for old vegetation. The flammability f is modelled as follows:

$$f = f_o + c * exp(a)$$

Give a biological interpretation of f_o and c. What are the possible ranges of values for f, f_o and c?

2. Consider that the size of a cell is 1 ha. If the probability of burning is f and there are n strikes per time step on an m×m area, what is the number of fires per year per ha? What do you think are realistic values (have a quick look on the internet)?

3. Run the main program ACForestFire1Year.m. This program is associated to two functions: *FireProgression.m* and *BurningNeighbors.m*. Run the program for different values of the parameters to understand how the system functions (see also Appendix A).

Have a closer look at the function *FireProgression.m*. Once a few cells are burning because of lightning, the function *FireProgression.m* works as follows:
- a cell is chosen at random in the lattice;
- if at least one of its neighbors is burning, f is calculated;
- a random number is picked. If it is smaller than f, the cells burns, otherwise it remains in the same state.

This way of updating the lattice is called **asynchronous**. Another way of updating the lattice would be to do the same but from the first to the last cell of the lattice, which is called **synchronous**. Write another function *FireProgressionSync.m* (use the diagram provided in Appendix B). Don't forget to change the name of the function in the main program *ACForestFire1Year.m* too. Run simulations. Is there any difference compared to the asynchronous update? Why? Show some figures. In the following, keep the asynchronous update.

4. Create a variable "Burning" that calculates the size of the fire at the end of a simulation (= the total number of burned cells).

5. For certain simulations, there is no fire (because the model is stochastic...). If you run several simulations you can thus calculate the probability of fire:

$$ProbaF = \frac{NumberOf\ SimulationsWhereAFireStarts}{TotalNumberOfSimulations}$$

You can also calculate the average size of the fire for the simulations where a fire occurred:

$$AvSizeF = \frac{SizeOfBurnedAreaAtTheEndOfTheFire}{NumberOfSimulationsWhereAFireStarted}$$

Create a loop that allows to calculate *ProbaF* and *AvSizeF* for 50 repetitions (total number of simulations = 50). Add error bars.

6. We can now follow *ProbaF* and *AvSizeF* for different values of f_o. Create a loop that allows to obtain *ProbaF* and *AvSizeF* for different values of f_o (*f* has to remain between 0 and 1!). Trace the corresponding graph. Do the same for different values of *n*.

At this step, you have the final version of the program *ACForestFire1Year.m* that you can put in the appendix of your document.

7. Go back to the initial program. We now wonder what happens in the same system when we consider several years:

Lightning(year t) -> Fire propagation(year t) -> Vegetation growth(year t) -> lightning(year t+1)...

The initial state of the system at t+1 is not random anymore, but is the final state of the system at time t. Create a loop that allows to see how the system behaves during several years (5 years for example). Show figures of the system at different time steps. How does the size of the fires vary during the years? Why is this the case, do you think? Which assumptions of the model are responsible for this behavior?

8. We are now going to add mortality of the vegetation. At each time step, each old vegetated site has a certain probability *m* of dying. It then becomes an empty site. See how it changes the behavior of the system. Show figures.

9. With the current version of the model, empty sites automatically become occupied at the next time step. Modify the model so that the probability *r* that an empty site becomes recolonized depends on the number of vegetated sites in its neighborhood (local seed dispersal, for example). Show figures. You now have the final version of the *ACForestFireSevYear.m* file that you can put in the appendix of your document.

References

[1] Czàràn T., 1998. Spatiotemporal models of population and community dynamics. Chapman & Hall, London, 284 pp.

[2] Harada Y. and Iwasa Y., 1994. Lattice population dynamics for plants with dispersing seeds ad vegetative propagation. Researches on Population Ecology, 36(2):237–249.

[3] Iwasa Y., 2000. Lattice models and pair approximation in ecology. In: Diekmann U., Law R. and Metz J.A.J. (eds.), The Geometry of Ecological Interactions. Simplifying Spatial Complexity, Cambridge University Press, Cambridge UK, pp. 227-251.

[4] Wissel C., 2000. Grid-based models as tools for ecological research. In: Diekmann U., Law R. and Metz J.A.J. (eds.), The Geometry of Ecological Interactions. Simplifying Spatial Complexity, Cambridge University Press, Cambridge UK, pp. 94-115.