Przemysław JACEWICZ
Institute of Control and Computation Engineering
University of Zielona Góra
ul. Podgórna 50
65-246 Zielona Góra, Poland
e-mail: P.Jacewicz@issi.uz.zgora.pl

*Referees:*

- Adam KOWALEWSKI, Kraków University of Mining and Metalurgy
- Janusz HALAWA, Wrocław University of Technology

The text of this book was prepared based on the author's Ph.D. dissertation supervised by Professor Dariusz Uciński in close cooperation with professor Samira El Yacoubi.

Partially supported by the State Committee for Scientific Research (KBN) in Poland

ISBN 83-89321-67-X

Camera-ready copy prepared from the author's LaTeX2$_\varepsilon$ files.
Printed and bound by University of Zielona Góra Press, Poland.

# Contents

# Acknowledgments

It is a pleasure to express my sincere gratitude and thanks to:

- Professor Dariusz Uciński for suggesting the problem, and for his continuous support and supervision of this work,
- Professor Samira El Yacoubi and Abdelhaq El Jai, whose works introduced me to the ecological modelling problems.

In addition, I wish to thank all my friends and colleagues at the Institute of Control and Computation Engineering, who helped me in many, many ways while I was preparing this work.

# Chapter 1

## Introduction

The increasing prominence of computers has led to a new way of looking at the world. This view sees nature as a form of computation. That is, we treat objects as simple computers, each obeying its own set of laws. The notion of a *Cellular Automaton* (we will write CA for brevity) extends this analogy to provide a way of viewing whole populations of interacting *cells*, each of which is itself a computer (automaton). By building appropriate rules into a CA, we can simulate many kinds of complex behaviours, ranging from motion of fluids governed by the Navier-Stokes equations to outbreaks of starfish on a coral reef. Why has this transformation taken place? First, there is the simple matter of practicality: desktop computing power has reached a level at which it is quite feasible to simulate individuals as they move across a landscape, interact, reproduce and die. Second is the issue of the language: for many practitioners, rules encoded in computer algorithms are much more accessible than the formal mathematical language of dynamic systems. Third is the awareness that the simple models traditionally used in ecology, physics or process engineering have not always proved very successful in accounting for phenomena observed in real systems. The simulations have their own intrinsic interest; they can be a valuable aid in defining and characterizing the processes involved and can lead to the discovery of new and interesting phenomena.

CA's are dynamic systems. In a CA, a small set of values is defined. Space consists of a regular grid of cells, each of which contains a value from this set. Time consists of a sequence of discrete intervals. At each interval a 'snapshot' of the grid shows the distribution of values—the *population*—at that time. The population is transformed over time by the application of a small number of extremely simple rules. Surprisingly, it turns out that such models are capable of reproducing very complex phenomena. The application of the rules can lead to interesting patterns and behaviour even in the simplest case. The consequences of many of these ideas are raising some very fundamental questions for both theoreticians and practitioners.

From the theoretical point of view, the pioneer was beyond doubt John von Neumann who had been involved in the design of the first digital computers. Though his name is definitely associated with the architecture of today's sequential machines, his seminal idea of CA's outlined in the late 1940's [83] constitutes

also the first applicable model of massively parallel computation. Von Neumann thought about imitating the behaviour of a human brain in order to build a machine able to solve very complex problems [81, 82, 83, 84]. He postulated that the corresponding machine should thus contain self-control and self-repair mechanisms. What is more, the difference between processors and data should be removed, which led him to envisage a machine capable of building itself out of some available material. Following the suggestions of a great Polish mathematician Stanisław Ulam, von Neumann addressed this question in the framework of a fully discrete universe made up of cells. The first self-replicating CA was composed of a two-dimensional square lattice and the self-reproducing structure was made up of several thousand elementary cells. Each of these cells had up to 29 possible states. The system evolved in discrete time steps and the local rule determining this evolution was the same for all cells and updates of the internal state of each cell occurred synchronously. Due to its complexity, this first CA was only partially implemented on a computer at that time.

Von Neumann's work was completed and described by Arthur Burks who maintained an active interest in the field for several years afterwards [10, 77]. But the low computing power of the computers at that time has inhibited the development of the research for many years. From the more practical point of view, it was more or less in the late 1960's when John Horton Conway, a British mathematician, developed the *Game of Life* which drew scientists' attention to the subject again. It was the first such system calculated on computers. This game became that popular, that a scientific magazine published regularly articles about the 'behaviour' of this game. Contests were organized to prove certain problems. The Game of Life was supposedly the first program run on a parallel processing computer. In fact, it has been estimating that more computing time has been spent running the Game of Life program than any other computer program. While the Game of Life is an abstract 'toy' system that has not yet been found to directly represent any specific natural system, it has been the springboard for the study of so-called 'artificial life' systems because of the amazingly complex behaviours displayed by some of the patterns that occur during the running of the CA.

In the late 1980's the interest in CA's strengthened again as powerful computers became widely available. Today a set of accepted applications in simulation of dynamic systems are available. CA's have inspired several parallel computer architectures and several special-purpose machines for their simulation have been built. For description of CA's, several languages have been developed (Cellang, Cellsim [88]) and most CA simulations are still performed on general-purpose computers (for a survey, see e.g. [88, 12, 28, 27]), although specialized hardware is also accessible (CAM-8, CEPRA [88]).

At present, two main branches of research on CA's can be distinguished:

1. CA's as parallel models of computation and dynamic systems, which leads to setting up algorithms, language or pattern recognition, complexity theory, classifications, etc.

2. CA's as models of natural processes in physics, chemistry, biology, economy,

etc.; in this context, two lines of research are pursued:

- to simulate phenomena on CA's
- to try to predict phenomena in studying properties of relevant cellular models.

In this work attention will be focused on the second branch. But prior to a brief survey of CA applications and the statement of the main aims of the present contribution, some basic definitions necessary to understand what CA's are will be given.

## 1.1. CA Definition

A *cellular automaton* (CA) is a discrete dynamic system. Space and time in this system are also discrete. The basic element of a CA is the *cell*. A cell is a kind of a memory element which stores *states*. The state of each cell is a variable which takes a value from a given finite set of admissible values (these can be either numbers or properties). In the simplest case, each cell can have the binary states 1 or 0. In more complex situations the cells can have more different states. The cells are arranged in a regular spatial web called the *lattice*. Its topology usually corresponds to either a one-dimensional string of cells or a two-dimensional grid. In the latter case, the cells are most often arranged as a simple rectangular grid (like a sheet of squared notepaper), but other arrangements, such as honeycomb (with cells in form of hexagons) are sometimes used. A three-dimensional lattice can be made from cubes. Each cell has its *neighbourhood* being the set of nearby cells which interact with it. For example, on a grid these are normally the cells physically adjacent to the cell in question.

The CA system evolves over a succession of time steps. The cell states are updated synchronously in each time step using a set of *rules* which define how the state of a cell at a given time moment depends on its own state one time previously, and the states of its nearby neighbours at the previous time step. Thus the state of the entire lattice advances in discrete time steps.

The informal introduction above makes it possible to give the following general definition used subsequently in what follows:

**Definition 1.1** A $d$-dimensional cellular automaton (or $d$-CA) $\mathcal{A}$ is a quadruple $(\mathcal{L}, \mathcal{S}, N, f)$, where:

- $\mathcal{L}$ is a regular lattice which consists of cells arranged in a periodic paving of a $d$-dimensional spatial domain; the cells are indexed by tuples (multi-indices) $\underline{i} = (i_1, \ldots, i_d)$ of integers, i.e. the lattice is composed of cells $c_{\underline{i}} = c_{i_1,\ldots,i_d}$, where $i_1 = 1, \ldots, I_1; \ldots; i_d = 1, \ldots, I_d$;
- $\mathcal{S}$ is an $m$-element set of admissible state values ($m$ being finite), $\mathcal{S} = \{s^1, \ldots, s^m\}$;

- the neighbourhood $N$ of size $n$, is defined by a mapping

$$N : \begin{cases} \mathcal{L} & \longrightarrow & \mathcal{L}^n, \\ c_{\underline{i}} & \longmapsto & N(c_{\underline{i}}) = (c_{\underline{i_1}}, c_{\underline{i_2}}, \ldots, c_{\underline{i_n}}); \end{cases} \tag{1.1}$$

- $f$ is a function which determines the transition rule,

$$f : \begin{cases} \mathcal{S}^n & \longrightarrow & \mathcal{S}, \\ s_t(N(c)) & \longrightarrow & s_{t+1}(c), \end{cases} \tag{1.2}$$

  where $s_{t+1}(c)$ is the state of a cell $c$ at time $t+1$, and $s_t(N(c))$ stands for the state of the neighbourhood of $c$ at time $t$, i.e. $s_t(N(c_{\underline{i}})) = (s_t(c_{\underline{i_1}}), \ldots, s_t(c_{\underline{i_n}}))$.

**Remark 1.1** One of the crucial elements of Definition 1.1 is the notion of the transition function $f$ which governs the evolution of the CA system. This function can be given by an analytical function, a table of cell-state transitions or a set of transition rules. It depends, of course, on the lattice geometry, the neighbourhood and the state set, and can be deterministic or probabilistic.

**Remark 1.2** A *configuration* (global state) of a CA at time $t$ is a mapping $C_t : \mathcal{L} \longrightarrow \mathcal{S}$ which assigns every cell a state value. Then the global dynamics of a cellular automaton is defined as the function:

$$F : C_t \longrightarrow C_{t+1} \tag{1.3}$$

which changes the current configuration into a new one.

## 1.2. Lattice types

The most commonly used approach to define a CA lattice is to present it as a one or two-dimensional rectangular grid. This approach is very intuitive (the first simulations of CA's were performed by John von Neumann on the checkered paper). Moreover, it is very simple to implement it on computers where this type of lattices naturally translates itself onto vectors or arrays.

For simulations of natural phenomena it is very customary to place cells on a hexagonal lattice (cf. Fig. 1.2). This manner allows us to introduce an intermediate neighbourhood definition of size 7 between the von Neumann (of size 5) and Moore (of size 9) ones for the rectangular case. It is important because each increase in the neighbourhood size permits to improve the precision. Additionally, this choice is motivated by a wish to imitate a real biological tissue where for common species the most eager number of neighbours is also 6 or 7.

Because currently most of simulation tools define CA lattices as arrays of values or vectors, a geometrical transformation of the hexagonal lattice is com-

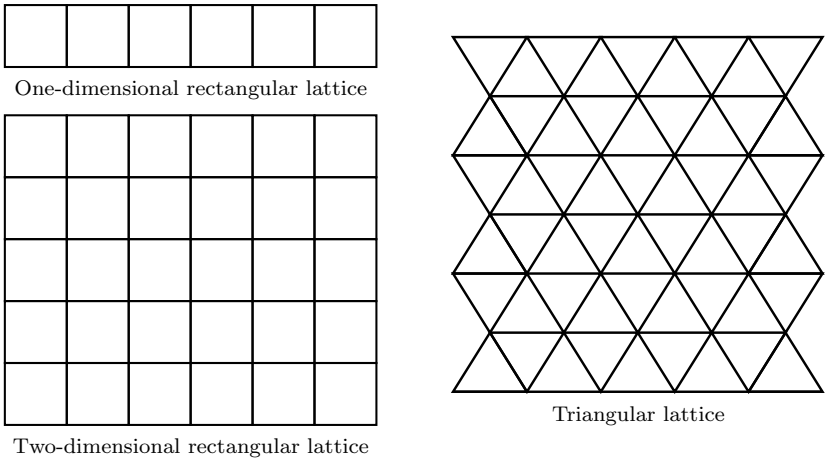



**Fig. 1.1.** Typical lattice shapes.

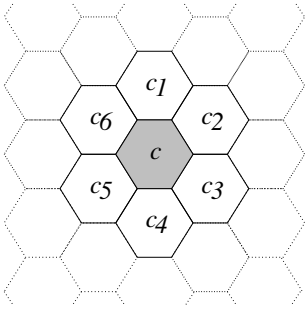



**Fig. 1.2.** Layout of cells on a hexagonal lattice.

**Tab. 1.1.** Relative coordinates of the hexagonal neighbourhood for the case shown in Fig.1.2.

| | Relative coordinates of neighbours for | |
|---|---|---|
| | $c \in$ even column | $c \in$ odd column |
| $c_1$ | $[0, 1]$ | $[0, 1]$ |
| $c_2$ | $[1, 1]$ | $[1, 0]$ |
| $c_3$ | $[1, 0]$ | $[1, -1]$ |
| $c_4$ | $[0, -1]$ | $[0, -1]$ |
| $c_5$ | $[-1, 0]$ | $[-1, -1]$ |
| $c_6$ | $[-1, 1]$ | $[-1, 0]$ |

monly used to a 2D rectangular lattice. For the case shown in Fig. 1.2, the hexagonal neighbourhood is defined as presented in Tab. 1.1 (cf. Fig. 1.3).

The values in the presented vectors mean columns and rows relative to the coordinates of the central cell $c$.

Some characteristic variety of a hexagonal lattice constitutes the triangular lattice used, among other things, in FHP-Gas [26].

## 1.3. Neighbourhood types

Let $\mathcal{A}$ be a cellular automaton $(\mathcal{L}, S, N, f)$. The neighbourhood of a cell $c$ is the set of all cells of the lattice $\mathcal{L}$ which will locally determine the evolution of $c$. It is finite and geometrically uniform, i.e. all cells share the same geometrical template of the neighbourhood. A neighbourhood may be punctured, i.e. $c \notin N(c)$, or it may include $c$.

Generally a neighbourhood can be any ordered finite set, but, actually, a few special types are mainly considered in applications. The most commonly used CA neighbourhoods are undoubtedly the von Neumann and Moore ones. They are known as the *nearest neighbour neighbourhoods*, and are mathematically defined in accordance with some associated distances. More precisely, if $\underline{i} = (i_1, \ldots, i_d)$ and $\underline{j} = (j_1, \ldots, j_d)$, then the following distances between cells $c_{\underline{i}}$ and $c_{\underline{j}}$ can be defined:

$$\rho_1(c_{\underline{i}}, c_{\underline{j}}) = \sum_{\alpha=1}^{d} |i_\alpha - j_\alpha|, \quad \rho_\infty(c_{\underline{i}}, c_{\underline{j}}) = \max\{|i_\alpha - j_\alpha| : \alpha = 1, \ldots, d\}. \tag{1.4}$$

Then we have

- the Von Neumann neighbourhood:

$$N_{\mathrm{VN}}(c_{\underline{i}}) = \{c_{\underline{j}} \in \mathcal{L} : \rho_1(c_{\underline{i}}, c_{\underline{j}}) \leq 1\}; \tag{1.5}$$

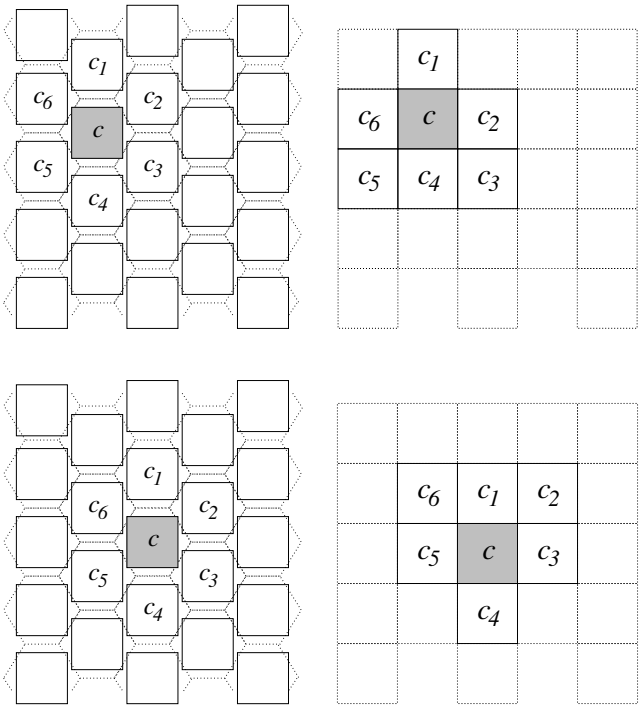



**Fig. 1.3.** A way to convert the hexagonal neighbourhood into the rectangular one.

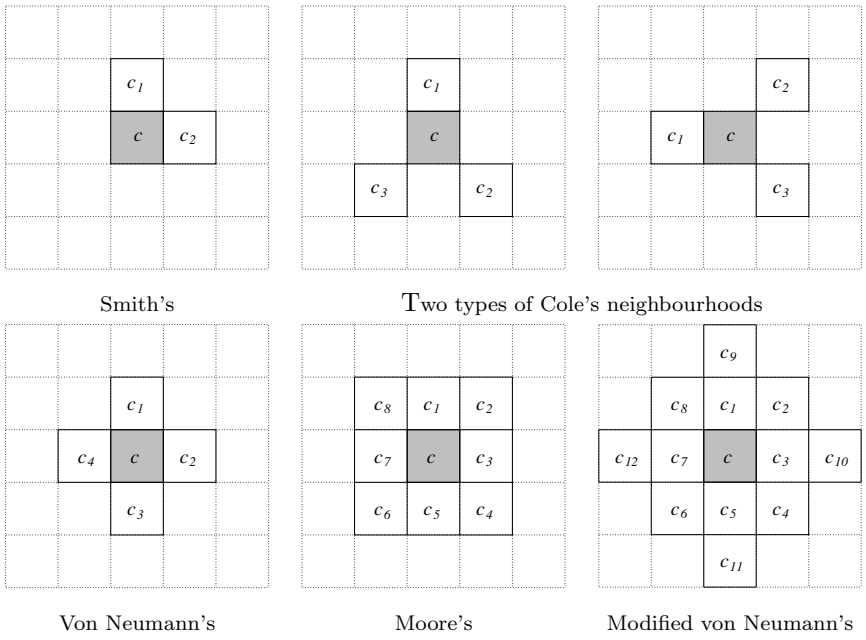



**Fig. 1.4.** Templates of the most common neighbourhoods.

sometimes the following extended version of the above is used:

$$N_{\mathrm{MVN}}(c_{\underline{i}}) = \{c_{\underline{j}} \in \mathcal{L} : \rho_1(c_{\underline{i}}, c_{\underline{j}}) \leq 2\}, \tag{1.6}$$

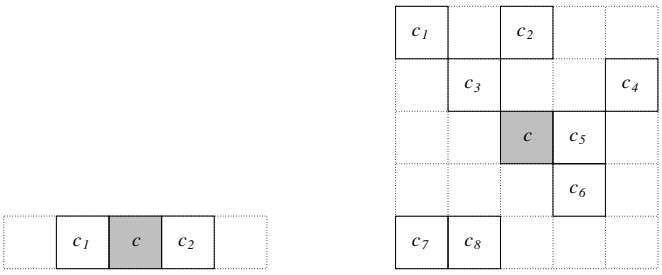
and it is called the *modified von Neumann neighbourhood*;

- the Moore neighbourhood:

$$N_{\mathrm{M}}(c_{\underline{i}}) = \{c_{\underline{j}} \in \mathcal{L} : \rho_\infty(c_{\underline{i}}, c_{\underline{j}}) \leq 1\}. \tag{1.7}$$

In the literature many other neighbourhood definitions can be found, e.g. Cole's [15] or Smith's [40] ones. The commonly used types of two-dimensional neighbourhoods are shown in Fig. 1.4. Note that in the one-dimensional (1D) case the Moore and von Neumann neighbourhoods coincide (see Fig. 1.5). Clearly, the possible choices are not restricted to the examples above and any other neighbourhoods can be constructed.

A slightly different approach to the CA neighbourhood definition consists in using the so-called *Margolus neighbourhood* [55, 77]. The rectangular lattice is there partitioned into a collection of $2 \times 2$ adjacent blocks which define a raster. Instead of a transition function, block rules are defined that look at the contents of a block and update the whole block (rather than a single cell as in an ordinary CA). The same set of block rules is applied to every block. Note that blocks do not




1D representation of von Neumann and Moore neighbourhoods.

Custom neighbourhood.

**Fig. 1.5.** The most popular 1D neighbourhood and an example of a 2D custom neighbourhood.

overlap, and no information is exchanged between adjacent blocks. The partition is changed from one step to the next, so as to have some overlap between the blocks used at one step and those used at the next step. This point is essential, as if we used the same partition at every step, the CA would be effectively subdivided into a collection of independent subsystems. At each time step, the lattice partitioning is shifted one cell down and one cell left. It is thus clear that the Margolus neighbourhood makes use of only two partitions (namely the even grid and the odd grid). An illustration of its use is given in Fig. 1.6.

## 1.4. Boundary conditions

For a complete spatial characterization of a CA, we need to define lattice boundary conditions. As the number of cells in the lattice has to be finite since one cannot deal with an infinite lattice, the question of what to do with cells at borders must be addressed. This is because the cells belonging to the lattice boundary do not possess the same neighbourhoods as other internal cells. Their influence depends on the size of the lattice. To give an example: in a $10 \times 10$ lattice about 40% of the cells are border cells, whereas in a $100 \times 100$ lattice only about 4% of the cells are of that kind. Anyway, this problem must be solved, which necessitates a proper determination of suitable neighbourhoods for the cells on the border of the lattice. For that purpose, the following three approaches are most commonly used:

1. **Periodic boundary conditions.** They have become default in the literature. In this solution the opposite borders of the lattice are 'sticked together'. A one-dimensional 'chain' of cells thus becomes a torus. In the case of a two-dimensional lattice, the left and right borders are treated as being connected, and so are the upper and lower borders. In a CA model

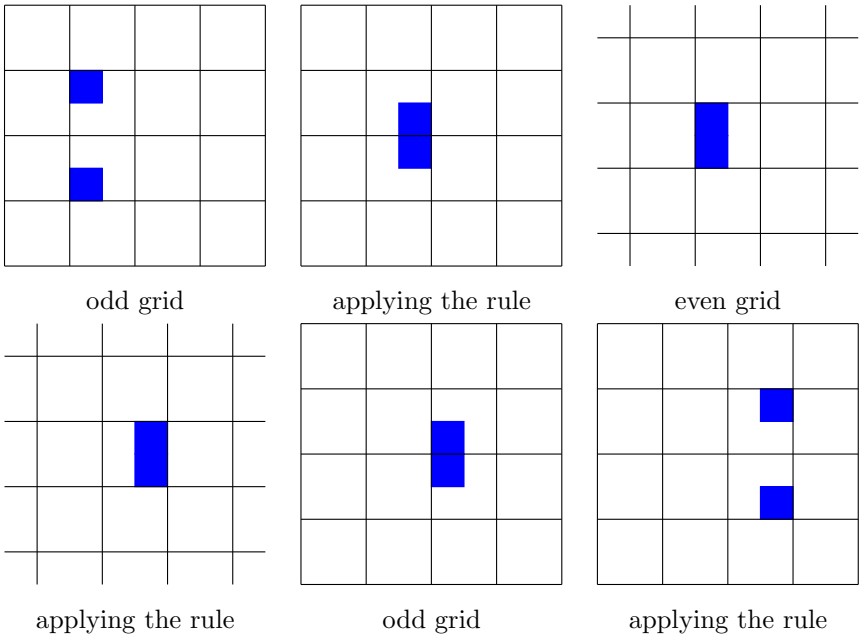



**Fig. 1.6.** Example of a CA evolution using the Margolus neighbourhood.

of mobile objects moving in space, the wrap-around boundary condition results in objects leaving the system at one border location and reentering at the opposing border location.

**Example 1.1**
Consider the von Neumann neighbourhood and a one-dimensional lattice of $\ell$ cells. Then we have

$$N(c_i) = \begin{cases} (c_\ell, c_1, c_2) & \text{if } i = 1, \\ (c_{i-1}, c_i, c_{i+1}) & \text{if } i = 2, \ldots, \ell - 1, \\ (c_{\ell-1}, c_\ell, c_1) & \text{if } i = \ell. \end{cases} \tag{1.8}$$

A more complex example of these boundary conditions is shown in Fig.1.7. It concerns a two-dimensional lattice with modified von Neumann neighbourhood.

2. **Fixed boundary conditions.** They correspond to Dirichlet boundary conditions encountered while solving partial differential equations. The solution is based on augmenting the lattice by a set of virtual cells beyond its limits. Then a constant state value is imposed on the additional cells. In some cases modellers apply different values to each lattice border. In Fig. 1.8 we can

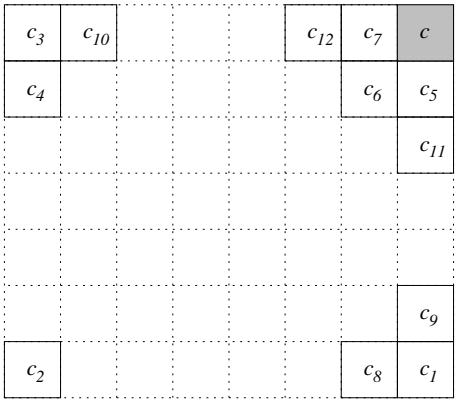



**Fig. 1.7.** Determination of the modified von Neumann neighbourhood for a border cell $c$ on a 2D lattice and periodic boundary conditions.

see the situation where the states of the missing cells $c_1$, $c_2$, $c_3$, $c_4$, $c_8$, $c_9$ and $c_{10}$ from the modified von Neumann neighbourhood definition of cell $c$ are to be defined in a sense. For fixed boundary conditions these values are permanently fixed constant for all time moments.

3. **Reflecting boundary conditions.** In this solution the lattice is also augmented by a suitable set of virtual cells beyond its limits. A reflecting boundary condition amounts to copying to the virtual cell the state value of the existing neighbour opposite.

**Example 1.2**
The reflecting boundary conditions for a one-dimensional lattice of $\ell$ cells and the von Neumann neighbourhood imply the following cases:

$$N(c_i) = \begin{cases} (c_2, c_1, c_2) & \text{if } i = 1, \\ (c_{i-1}, c_i, c_{i+1}) & \text{if } i = 2, \ldots, \ell - 1, \\ (c_{\ell-1}, c_\ell, c_{\ell-1}) & \text{if } i = \ell. \end{cases} \tag{1.9}$$

Completion of the modified von Neumann neighbourhood definition for a border cell $c$ on a two-dimensional lattice is illustrated in Fig. 1.9.

## 1.5. Types of transition functions

One of the most important points in CA modelling is an appropriate definition of the transition function which governs the evolution of the entire system. It can be given by an analytical function, a table of cell-state transitions, or a set of

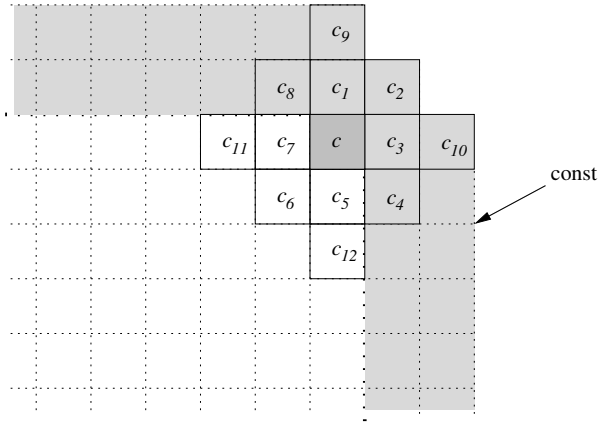



**Fig. 1.8.** Determination of the modified von Neumann neighbourhood for a border cell $c$ on a 2D lattice and fixed boundary conditions. The grey cells represent virtual cells whose states are set to a given fixed value.

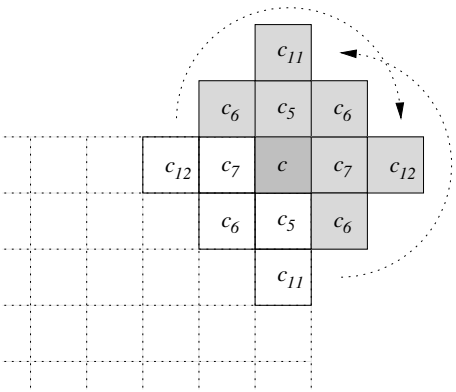



**Fig. 1.9.** Determination of the modified von Neumann neighbourhood for a border cell $c$ on a 2D lattice and reflecting boundary conditions.

transition rules. It depends, of course, on the lattice geometry, the neighbourhood and the state set, and can be deterministic or probabilistic.

Some characteristic transition functions encountered in real problems are as follows:

- A transition function $f$ is said to be *totalistic* if $f$ has the form

$$f(s_t(N(c))) = \varphi\Big(\sum_{c' \in N(c)} s_t(c')\Big), \tag{1.10}$$

  where $\varphi$ is a given mapping. Such functions give equal weight to all the cells in the neighbourhood and imply that the value of a cell state depends only on the sum of all previous state values of the neighbourhood cells.

  In the particular case when $c \notin N(c)$, a CA rule is said to be *outer-totalistic* (a classical example is the Game of Life).

- A probabilistic CA rule may be specified by a function.

$$f(s_t(N(c))) = \begin{cases} s^1 & \text{with probability } p(s^1|s_t(N(c))), \\ \vdots & \vdots \\ s^m & \text{with probability } p(s^m|s_t(N(c))), \end{cases} \tag{1.11}$$

  where the probabilities are non-negative and satisfy the normalization condition

$$\sum_{s \in \mathcal{S}} p(s|s_t(N(c))) = 1. \tag{1.12}$$

## 1.6. The most famous CA: Game of Life

The Game of Life (or simply Life) is not a game in the conventional sense. There are no players, and no winning or losing. Once the "pieces" are placed in the starting position, the rules determine everything that happens later. In most cases, it is impossible to look at a starting position (or pattern) and see what will happen in the future. The only way to find it out is to follow the rules of the game [11].

Life was invented by the mathematician John Conway in 1970. He chose the rules carefully after trying many other possibilities, some of which caused the cells to die too fast and others caused too many cells to be born. Life balances these tendencies, making it hard to tell whether a pattern will die out completely, form a stable population, or grow forever. To the moment this is still the most popular CA model [3].

The model is played on a two-dimensional binary lattice with periodic boundary conditions. A cell $c_{i,j}$ can be alive ($s_t(c_{i,j}) = 1$) or dead ($s_t(c_{i,j}) = 0$). To apply one step of the rules, we count the number of alive neighbours in the Moore sense for each cell. The life and death rules for updating a lattice site are as follows:

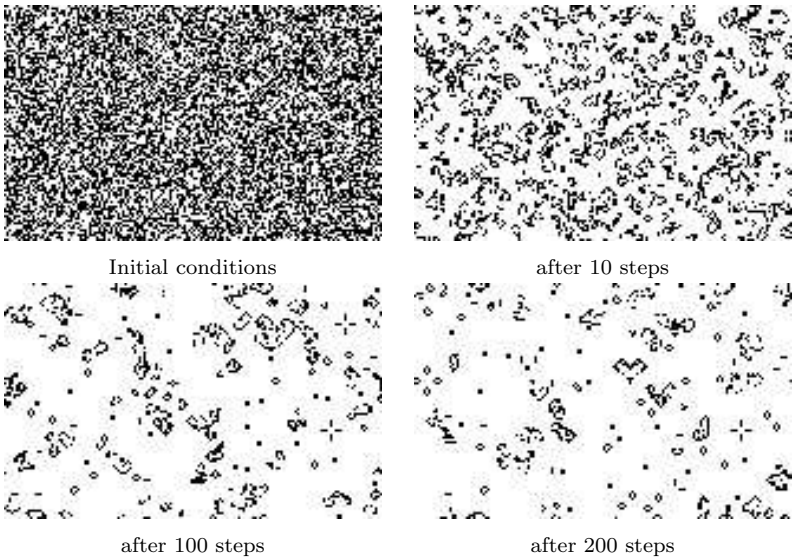



**Fig. 1.10.** Exemplary evolution for the game of life CA starting from random initial conditions.

- Any site (dead or alive) with three living nearest neighbour sites stays alive or is born.
- A living site ($s_t(c_{i,j}) = 1$) with two living neighbours remains alive.
- All other sites remain dead or die (for overcrowding or loneliness).

Figure 1.10 shows the evolution of the above two-dimensional CA starting from randomly generated initial conditions (the probabilities of the two states are equal to 0.5). White points represent dead or empty cells, and the black ones —alive cells. After the first 10 time steps the model reduces the number of alive sites so as to find an equilibrium of living conditions (cf. the last two pictures).

In Fig. 1.11 we can see characteristic states of this model working on the 11×11 lattice of cells, grouped by types [11]. The simplest are stochastic configurations. For a cell to remain alive, it must have 2 or 3 living neighbours, and every dead cell which is going to remain so may have any number of neighbours except 3. Oscillators are objects that change from step to step, but eventually repeat themselves. The simplest kind are period-2 oscillators, or those that repeat themselves after two steps. In this CA some of the objects can move. This was one of the most exciting early discoveries in Life. These common moving patterns, called gliders, may consist of just 5 cells. Finally, here are some simple starting patterns that develop into oscillators. A row of 10 alive cells becomes the period-15 oscillator called the pentadecathlon. Other patterns may become very pretty period-3 oscillators called the cross and the pulsar.

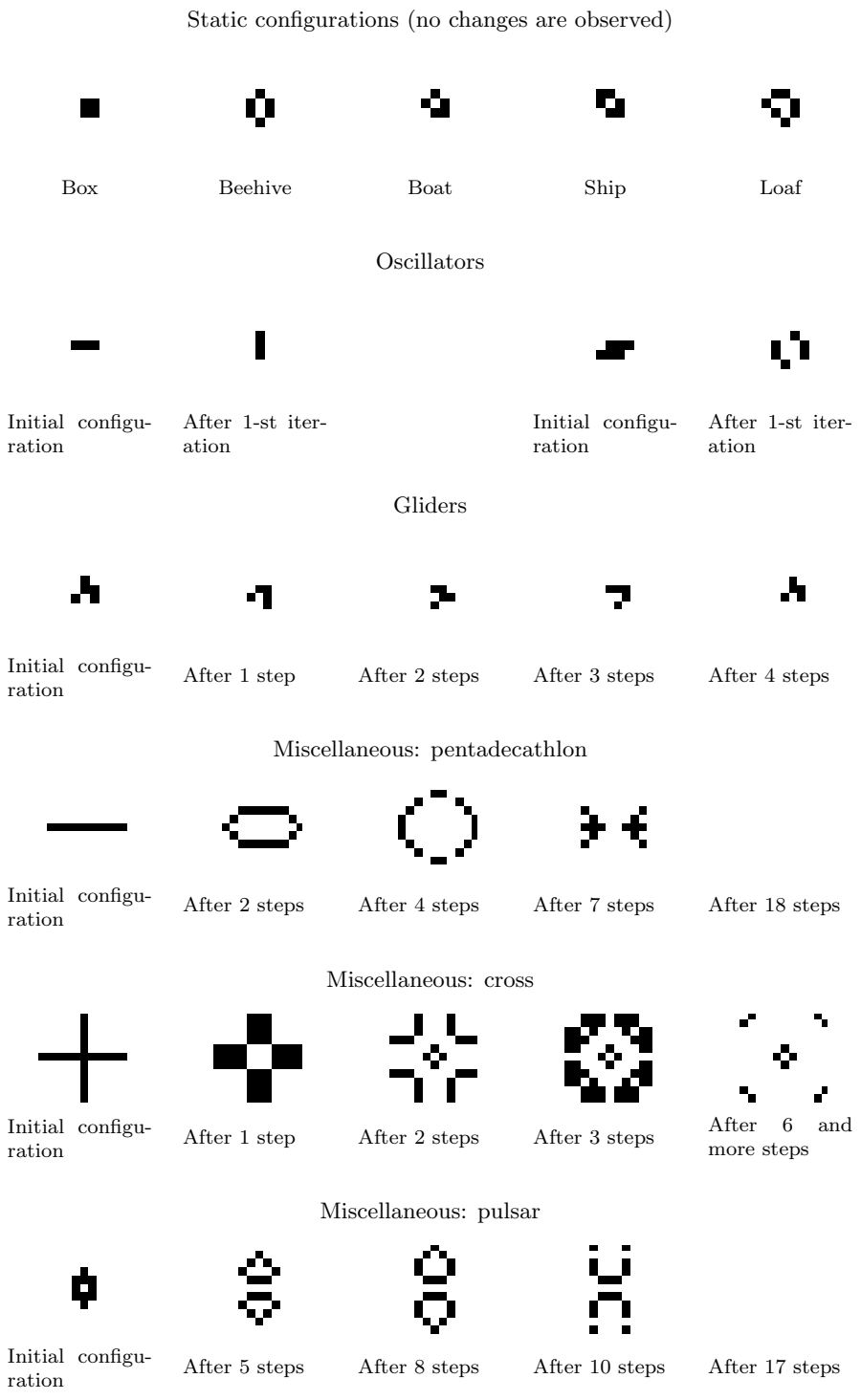



**Fig. 1.11.** Exemplary evolution for the Game of Live starting from some specific initial conditions.

## 1.7. Applications

As was already mentioned in the beginning of this chapter, at present there are two major branches of research on CA's. The first concentrates on setting-up algorithms, language or pattern recognition, complexity theory and classification. In this case CA's are used to solve some problems which are specified as their initial conditions. The solution is read from the final configuration, obtained after a number of time steps (the simplest example of such a CA is solving a maze, i.e. finding a path through a labyrinth). The other branch of CA research represents modelling issues which are in the heart of this dissertation. Such CA's are conceived as models for natural processes in physics (e.g. diffusion, fluid flow), chemistry, biology (e.g. spreadable and ecological models), economy, etc.

In most cases CA models are built on the basis of detailed knowledge of the phenomenon under consideration. It often happens that the global behaviour of a phenomenon can be described through local rules of state transitions, based on a moderately large neighbourhood. For example, the diffusion phenomena can be described using particles which move forward in the absence of collisions with other particles or change their directions if a collision takes place (cf. Section 1.7.2). Such a phenomenon is relatively simple to be described by a CA, and this constitutes a very interesting alternative to classical descriptions by partial differential equations. The situation becomes slightly worse when we strive to model e.g. the spread process of a computer virus via the Internet. As is well-known, each computer can be infected by another which is not necessarily connected to it, but by this one which has included its address in its e-mail address book. The problem is how to define proper rules of local transitions and a proper neighbourhood in this case, based on recorded observations. This will be the main subject of this work.

In the sequel, some basic issues involved in applications of CA's will be briefly discussed.

### 1.7.1. Universalities

The first who worked on *computational universality* and *construction universality* in CA's was John von Neumann. His works were formalized and more deeply investigated afterwards by Burks and people around him [10], as well as by Codd [14] and Banks [6]. Their main work was to construct two-dimensional CA's with these properties and as few states as possible, or to find conditions for CA's to satisfy them.

Roughly speaking, a system that is able to do universal computation is able to perform any finite algorithm. More precisely, a CA possesses the property of computational universality when it is able to compute any Turing-computable (or recursive) partial function. In this task we face some difficulties like: What does computing a function $\psi$ mean? How to input parameters? Where are the results? In order to translate the problem onto a CA architecture, we treat part of initial conditions as parameters of the studied function $\psi$. We obtain results after a number of time steps of the CA as part of its final configuration. Three problems then arise:

- How to encode the arguments?
- How to recognize the end of the computations?
- How to decode the results?

The main difficulty is to suitably encode infinite data into a finite set of cells in a lattice. A direct way was proposed by von Neumann and substantially extended by his main followers: Burks, Thatcher, Codd and Banks. Another predicament is the problem of directly comparing the computational power of CA's and Turing machines by means of simulations.

There are several notions of computational universality, depending on the fact whether or not they are set up by means of simulations, and in the former case, for example, depending on the fact that simulations are between computation models of different nature, essentially Turing machines and CA's (extrinsic simulations), or between cellular automata (intrinsic simulations), or depending on the configurations considered, on the encoding or decoding understandings, etc. [20]. Only a CA calculating for an infinite period of time can be universal.

*Intrinsic universality*, corresponding to some intrinsic simulations, was pointed out later in [2]. It also expresses the power of some CA's with respect to other (or a class of) CA's, and so it attests a sort of model (inner or auto-) coherence and potential expressive power. For that problem, the basic question is: do there exist CA's able to simulate all CA's, at least of a given class? It is not a problem to canonically coding a finite automaton underlying each CA, and to get a recursive enumeration of them. The problem arises with the configurations which are infinite [20]. Thus, if we restrict the devices to their finite configurations, we can imagine a solution analogous to the solution for Turing machines: Find a CA $\mathcal{U}$ such that, given the code of any automaton $\mathcal{A}$ of a given class and the code of a finite configuration $C$ of $\mathcal{A}$, we can get, in a configuration of $\mathcal{U}$, the halting configuration of $\mathcal{A}$ on $C$ when it exists (a serious problem will then be to satisfactory formulate the notion of the halting configuration).

Another problem is the so-called *construction universality*. In this task we look for an automaton to reproduce itself, without losing any of its properties (like computation abilities). In other words, we are interested in the self-reproduction ability of CA's.

The foregoing problems constitute the subject of numerous works of theoretical computer scientists. Since they are only of limited use in the context of this dissertation, we are not going to further dive into them.

### 1.7.2. Modelling

One of the primary applications of CA's is modelling complex physical processes. Owing to their discrete and parallel architecture, they seem to be especially suited for any spatial and discrete problems (like physical molecular models). An important step in the theory of CA's was made in the 1980's when the so-called HPP lattice gas model developed by Hardy, Pomeau and de Pazzis was recognized

as a CA [34]. This model consists of a simple and fully discrete dynamics of particles moving and colliding on a two-dimensional square lattice so as to conserve momentum and the number of particles. Researchers' hopes were raised that it would be possible to simulate the behaviour of a real system of particles (like a fluid or a gas) as a CA rule. CA's provided a new conceptual framework, as well as an effective numerical tool which retained important aspects of the microscopic laws of physics, such as simultaneity of the motion, locality of interactions and time reversibility. CA rules were viewed as an alternative form of the microscopic reality which bears the expected macroscopic behaviour. The first CA model which met these expectations was the famous FHP model proposed in 1986 by Frish, Hasslacher and Pomeau [26]. The authors showed that their model follows in some appropriate limits the behaviour prescribed by the Navier-Stokes equation of hydrodynamics. Similar models have been successful in modelling complex situations for which traditional computing techniques are not applicable. Flows in porous media, immiscible flows and instabilities, spreading of a liquid droplet and wetting phenomena, microemulsion, erosion and transport problems are some examples. An excellent survey of those and other applications can be found in [12].

CA models of spatio-temporal processes became immensely popular among biologists and researchers involved in ecology. This popularity is well deserved: as they are rule-based, biologists can readily turn them into algorithms for numerical simulation of individuals interacting in some spatial region. This makes an interesting alternative to common reaction-diffusion partial differential equations (PDE's) describing situations when several different species can interact with one another and diffuse in space. In PDE models, populations of each species are described by concentrations which vary in space and time. In CA models, populations are represented by units or individuals that move on a grid and interact with neighbouring units. While in some cases PDE models may best capture the dynamics or real reaction-diffusion systems, in other cases CA models are preferable [17, 18]: CA and PDE models represent two competing modelling paradigms, both of which capture, in a simplified way and to a certain extent, important aspects of real reaction-diffusion systems. In some cases it is even difficult to decide which alternative is to be preferred. Both modelling frameworks contain simplifications that result in a loss of details that may be important for the dynamics of real systems. While often it would be ideal to use a highly detailed, stochastic, individual-based model, such simulations of large (often three-dimensional) spatial systems require enormous computing power. Owing to limited computer resources, the model must then be highly simplified. Yet, CA and PDE models often can be derived as simplifications or limits of an underlying detailed, stochastic, individual-based model. In PDE models, which can be regarded as large-scale descriptions, populations of individuals are described by their concentrations and small-scale correlations are neglected. CA models, on the other hand, operate on a molecular level and focus on local correlations; however, they do not handle long-range correlations and large-scale patterns of concentrations as well as PDE's. For many real systems, both CA and PDE models can capture dynamics essentials. In some cases, however, simulations based on CA and PDE models

may give widely different results, even though each model is designed to describe a particular reaction-diffusion system as well as possible (within the limitations of each framework). CA models are to be preferred for microscopic simulations or for the study of other systems where numbers are low, and small-scale spatial correlations must be taken into account.

Below, some typical models which present capabilities of CA models are briefly presented.

#### 1.7.2.1. Modelling diffusion

When two materials are placed in contact with each other, an interface is formed between them. The nature of the interface is important in determining a variety of system properties, such as adhesion and mechanical strengths, thermal expansion, electrical conductivity, etc.

In real systems, diffusion derives from the global effect of many molecules which perform independent random walks. The random walk may come either from collisions between particles of the same species (self-diffusion) or from interactions with other species. In CA's, we can model diffusion in two different ways. The first method is essentially based on the mechanism which generates the diffusion phenomenon in real systems. The molecules are replaced by idealized particles and allowed to execute random movements on a regular lattice. A CA rule which involves discrete particles should conserve their total number.

The commonly used technique to model diffusion is the moving-particle approach. Two main elementary CA models used for modelling walking particles are:

- **Multiple random walkers** [27] being a model which operates on a square lattice with periodic boundary conditions. The state set defines a walk direction or a free space. Its transition function scans the modified von Neumann neighbourhood for free space to move a cell. Collisions are detected (particles then randomly choose new directions). A sort of this model is presented in Example 1.3.

- **Noise-driven diffusion** [77] being a deterministic CA model operating on the Margolus neighbourhood and on a square lattice. The transition function depends on a particular variant of the model, but in general it allows for particle movement and collision detection (molecules avoid one another, or collide elastically and change their movement directions). A diffusion model of that type is presented in Example 1.4.

**Example 1.3**
Consider a rectangular lattice of $100 \times 50$ cells. Each cell in the lattice is either empty or occupied by a particle. The cell state values are nonnegative integer values which correspond to the following situations [27]:

0 – an empty site,

1 – a site occupied by a north-facing particle,

2 – a site occupied by an east-facing particle,

3 – a site occupied by a south-facing particle,

4 – a site occupied by a west-facing particle.

Initially, all of the sites in the lattice are empty (i.e. the cells have value 0), except for the sites in the bottom row, which are occupied by particles facing randomly chosen directions (i.e. these sites have positive integer values). The left and right boundaries of the lattice are periodic. The bottom boundary acts like a source, so that a particle moving upward from the bottom row is replaced by another particle. The top boundary acts like a sink, so that a particle moving upward from the top vanishes.

The system evolves over a succession of time steps. In each time step, each particle moves to the nearest neighbour site lying in the direction the particle is facing (hence, the value of 1, 2, 3 or 4 indicates that the particle moves in the north, east, south or west direction, respectively) if the site is empty and if the movement does not result in a collision with another particle moving into the same site; otherwise, it remains in place. Whenever a particle moves out of the next-to-bottom row, it is replaced by another particle, and whenever a particle moves upward from the top row, it disappears.

The modified von Neumann neighbourhood is used here and then the movement of a particle in each cell depends on the state values of 12 neighbouring cells and therefore the update rules take 13 arguments. At each time step, a particle, regardless of whether it moves or remains in place, randomly chooses a direction to face. The following rules are used:

- A particle facing an empty site moves from the site it is occupying unless another walker faces the same empty site, in which case it remains in place and randomly chooses a direction to face.
- Any other particle remains in place and randomly selects a direction to face.
- An empty site remains empty if two or more particles face it.
- An empty site becomes occupied if it is faced by exactly one particle.
- Any other empty site remains empty.
- A particle moving out of the border bottom row is replaced by another particle.
- A particle moving upward from the top row disappears.

Figure 1.12 shows the evolution of the above two-dimensional CA (particles are marked in black). Various configurations are generated respectively after 250, 1000, 17000 time steps. As expected, the particles reveal the trend to occupy the whole space as time elapses.

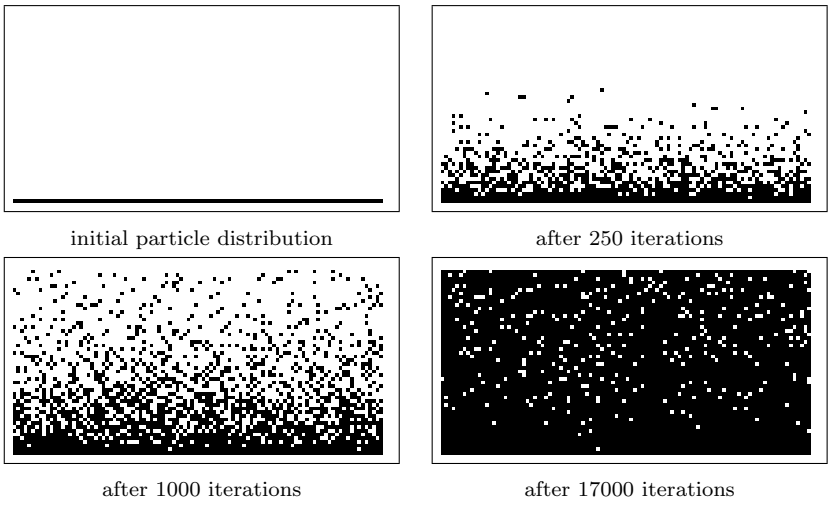



**Fig. 1.12.** Evolution of the CA modelling diffusion of Example 1.3.

**Example 1.4**
Let us model by a CA gas particles diffusing in a container consisting of two compartments with an aperture in the wall which separates these compartments. In this case, the CA in question is defined on a hexagonal lattice and von Neumann's neighbourhood definition is used. Each cell in the lattice is empty, occupied by a particle or constitutes part of the wall. The cell state values are nonnegative integers which correspond to the following situations:

0 – an empty site,

1 – a site occupied by a north-west facing particle,

2 – a site occupied by a north-east facing particle,

3 – a site occupied by an east facing particle,

4 – a site occupied by a north-east facing particle,

5 – a site occupied by a north-west facing particle,

6 – a site occupied by a west facing particle,

7 – a site representing a wall.

Initially, only the right compartment is filled with particles (see Fig. 1.13). This means that the cells in the left compartment have state values representing empty sites while the cells in the right compartment are randomly filled with values from 1 to 6 if a moving particle is to be present at this site (the probability of this presence is 1/3). Because the lattice is enclosed with cells representing the wall, the boundary conditions are not important.

The system evolves over a succession of time steps. At each of these discrete time moments, every particle strives to move to the nearest neighbour site lying

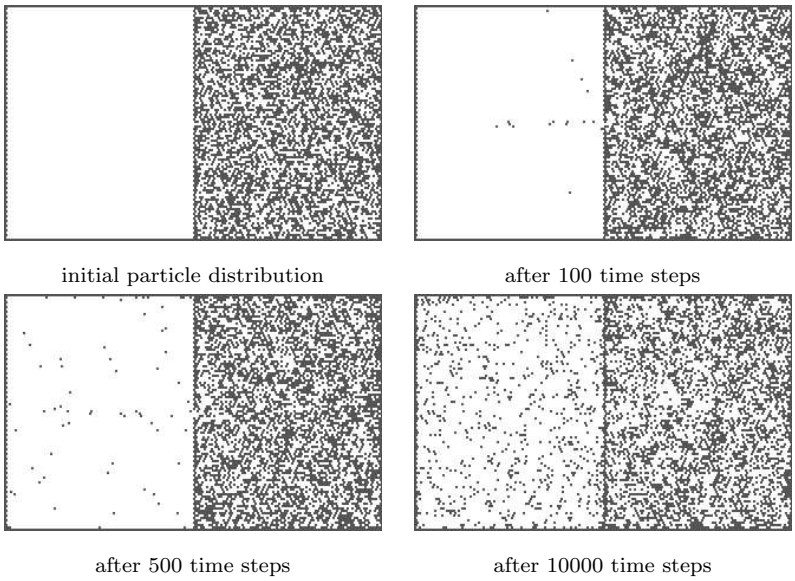



**Fig. 1.13.** Evolution of the CA modeling diffusion of Example 1.4.

in the direction it is facing (hence, a value of 1 indicates that the particle moves in the north-west direction, a value of 2 indicates the movement north east, etc.). If a site is vacant and the moving neighbours which face it do not induce a collision (this would be the case if two or more neighbouring particles tended towards this site), it takes over the incoming particle (if it does exist), otherwise it remains empty. If a collision takes place (a particle tries to move to a site occupied by another particle or a wall, or it faces an empty cell being a target for another particle) it changes the direction by turning left for west directions (state values 2, 3 and 4) or right for east directions (state values 1, 5 and 6).

Exemplary evolution of the described model is shown in Fig. 1.13. As can be seen, in spite of the barrier, particles from the right compartment aim at filling the left one so as to compensate for different particle densities.

#### 1.7.2.2. Hydrodynamics

One of the commonly applied approaches to describe fluid behaviour is to use the Navier-Stokes equation. It analytically expresses fluid behaviour under a wide range of conditions. If the general velocity of fluid is $v$, then the force acting on the fluid is of the form [12]

$$F = m\frac{\mathrm{D}v}{\mathrm{D}t} = \varrho V\frac{\mathrm{D}v}{\mathrm{D}t}, \tag{1.13}$$

where the operator $\mathrm{D}v/\mathrm{D}t$ is the material time derivative, $m$ is a fluid mass, $\varrho$ stands for the density and $V$ signifies the volume.

A continuous treatment starts by dividing space into cells, each of dimension $\Delta x$, $\Delta y$, $\Delta z$ and quantifying the overall effect in each direction. Thus the force in the $x$ direction is given by

$$\rho\Delta x\Delta y\Delta z\frac{\mathrm{D}v_x}{\mathrm{D}t} = F_x\rho\Delta x\Delta y\Delta z + P\Delta y\Delta z - (P + \frac{\partial P}{\partial x}\Delta x)\Delta y\Delta z, \tag{1.14}$$

where the first term on the right-hand side represents the external force, whereas the second term is the internal force acting in the $x$ direction ($P$ being pressure). The second and third terms together estimate the change in force over the $x$ length of the cell.

If this is simplified and if the differences are allowed to tend to zero, then we get

$$\frac{\mathrm{D}v_x}{\mathrm{D}t} = F_x - \frac{1}{\rho}\frac{\partial P}{\partial x},$$

or in general

$$\frac{\mathrm{D}v}{\mathrm{D}t} = F - \frac{1}{\rho}\nabla P. \tag{1.15}$$

A possible viscosity can be included by addition of some extra terms to the equation above. However, since these terms are generally nonlinear, the solution is often very difficult. Some of the problems can be quantified by the use of dimensionless parameters such as the Reynolds and Mach numbers. In this case the Reynolds number is a measure of the effects of nonlinear terms (due to viscosity), while the Mach number constitutes a ratio of velocities and indicates the effects of compressibility. Although additional effects can be included, this macroscopic treatment can quickly become intractable, particularly for realistic problems such as free convection and weather modelling.

Alternatively, the entire problem can be investigated from a microscopic point of view, where the behaviour of some of the constituent particles is monitored over time. It turns out that, on a macroscopic scale, the gases described by CA rules, such as the HPP model [34], obey the Navier-Stokes equation approximately [34] and a similar model on a hexagonal lattice, namely the FHP model, obeys this equation exactly [26]. Many researchers have recently become interested in such models of fluid dynamics, as they have a number of attractive conceptual features and show considerable practical promise [46].

**Example 1.5**
Let us consider a simple two-dimensional model of a fluid. Because the behaviour of fluids is determined by two factors: compensation of pressure and preservation of density (or volume), we can simplify the modelling problem and take account of only these two factors. In order to define local CA rules, we can decide that each cell containing a particle is in equilibrium state when all its neighbours are vacant. Any particles in the direct neighbourhood mean crush (it produces positive pressure $P$).

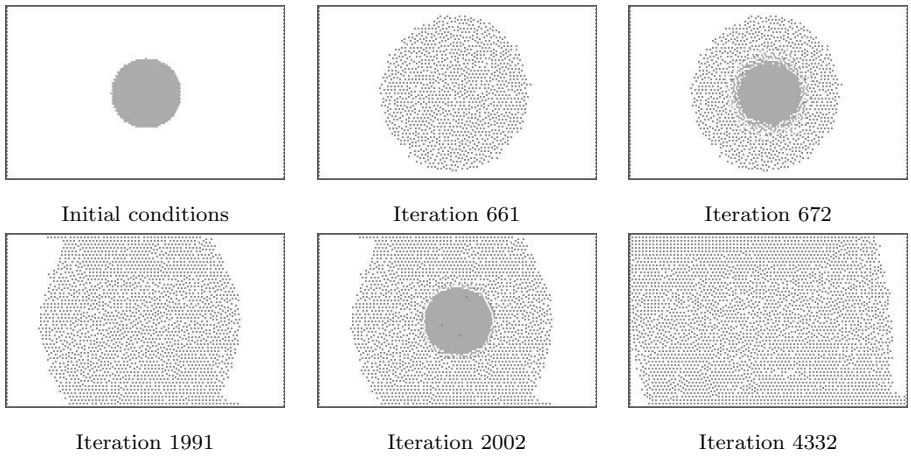



**Fig. 1.14.** Evolution of a density compensation model.

Each particle without empty neighbourhood moves in the direction opposite the position of the neighbours occupied by particles. To do that, we may think that each occupied neighbour of a cell $c$ has an effect of an external force acting on $c$ from the direction of this neighbour. To implement such an approach, we can apply the idea described in Section 3.2.2.

The evolution of an exemplary two-dimensional CA with hexagonal lattice of size 160×100 is shown in Fig. 1.5. The picture showing the initial lattice configurations corresponds to a drop falling on the plane. All cells are squeezed in a circle, and this represents a force of the drop (whose action on the occupied space constitutes the pressure). According to the outlined concept, each particle moves to compensate for the density (to eliminate pressure). After 660 iterations, the CA achieves the static configuration represented on the second panel. Then we put the next drop (in the 661-st iteration). The third panel shows the second drop after 10 iterations. The fourth panel shows the respective static configuration. The last drop is put in iteration 2002.

#### 1.7.2.3. Delayed models

Various processes in economics, biology, control, etc. are examples of systems in which the past time influences significantly the future behaviour of the system. Many situations in population dynamics are usual examples of such delayed processes. As regards systems with spatial interactions, CA models can also be used when we have to describe spatial phenomena with a state transition delay or a time delay inherent to the application of a specific control.

Consider a CA $\mathcal{A} = (\mathcal{L}, \mathcal{S}, N, f)$ where $\mathcal{L}$ is a regular lattice formed of the cells whose states are taken in a discrete set $\mathcal{S}$. Each cell interacts with its neighbours according to a given neighbourhood $N$ of size $n$, and subject to a

transition function $f$. We can extend this definition by introducing a generalized definition which allows the delay to be dependent on the neighbourhood (and then space-dependent). Let $D = \{T_1, T_2, \ldots, T_h\}$ be a discrete set of all possible delays which can affect the considered CA. This situation which occurs in some problems of ecology where the evolution may be regionally delayed depending on the soil and climate conditions, can hardly be described by partial differential equations, although recent advances in PDE's with delays are promising in this context [50].

**Definition 1.2** A cellular automaton $\mathcal{A}$ modelling a space-delayed system is defined by considering $\mathcal{A} = (\mathcal{L}, \mathcal{S}, N, f, \delta)$ where the transition function $f$ is defined as

$$s_{t+1}(c) = f(s_t(N(c)), s_{t-\Delta}(N(c))). \tag{1.16}$$

$\Delta = \delta(s_t(N(c))) \in D$ is the delay which affects the state of the neighbourhood $N(c)$ and where $\delta$ maps $\mathcal{S}^n$ into the set of delays $D$,

$$\delta : \begin{cases} \mathcal{S}^n & \longrightarrow \quad D \subset \mathbb{N} \cup \{0\}, \\ (s_1, \ldots, s_n) & \longmapsto \quad \Delta\ . \end{cases}$$

In the above definition, $\delta$ assigns a delay $\Delta \in D$ to each neighbourhood configuration ($D$ may or may not be finite). Some particular cases are:

- $D = \{0\}$, the cellular automaton is without delay,
- $D = \{\alpha\}$ corresponds to the case of a usual delayed system with a delay $\alpha$ (the same delay affects all the neighbours at any time),
- $\Delta = \delta(s_t(c))$ means that the delay depends only on the central cell of the neighbourhood.

A good example of this type of model constitutes the model of Green [31] presented in Sec. 2.2.1.

## 1.8. Identification of CA models and the problems addressed in this monograph

CA models have been extensively analysed mathematically for the last fifty years. With the availability of computers, the simulation of large CA models became feasible and abundant literature accumulated. This trend has recently been increased by the ubiquitous availability of fast personal computers. CA's exhibit three notable features, namely massive parallelism, the locality of cellular interactions, and simplicity of basic components (cells). As such they are naturally suited for hardware implementation, with the potential of exhibiting extremely fast and reliable computation that is robust to noisy input data and component failures.

On the other hand, CA modelling and simulation has been criticised by some practitioners, mostly on the ground of the lack of validity. It has been admitted

that nowadays the main difficulty in performing a realistic and convincing simulation of real processes is not the implementation of the model on a computer, but rather the incomplete information about the model structure and/or parameters. Often, it is difficult even to assess the correct order of magnitude of important parameters. A major impediment preventing ubiquitous computing with CA's stems from the difficulty of utilizing their complex behaviour to perform useful computations. Designing CA's to exhibit a specific behaviour or to perform a specific task is highly complicated, thus severely limiting their applications. This results from the local dynamics of the system, which renders the design of local rules to perform global computational tasks extremely arduous. Automating the design process would greatly enhance the viability of CA's. This involves the notions of structure and parameter identification which are widely used in the context of general dynamic systems [86].

Structure and parameter system identification are synonyms for statistical and numerical procedures to obtain reasonable model structures and model parameters, respectively, based on data such that the predicted response of the model is close, in some well-defined sense, to the process observations. The identification problem is also referred to as the inverse problem. Since the relationships encountered in the problem in context are nonlinear and linearization is of no use, the treatment of those problems is not straightforward and demands some prior information or reasonable guesses on the parameter values to get the numerical procedures started. In the context considered here, identification provides the link between data and model, between statistics and systems analysis and simulation.

The inverse problem for CA modelling is much more complex than the direct problem which consists in simulation of a CA model based on a given detailed description. As a result, the number of references is very limited. It can be said that the identification problem is understood in the literature as that of learning the underlying rules that govern the local behaviour of cells from temporal slices of the global evolution of the spatio-temporal pattern. Identification of CA models was thoroughly studied by Adamatzky in his monograph [1]. In that approach, it is understood as follows: Given a set of consecutive configurations (snapshots) of a completely unknown automaton, find an adequate CA model in the sense of producing the same observed evolution. In consequence, the components of the CA definition such as the state space, neighbourhood type and transition rules are to be discovered and, moreover, this description should be minimal, i.e. the size of the neighbourhood must be as small as possible. A number of algorithms are given and analysed for various classes of CA's, but those results are limited to finding appropriate transition tables and CA's are not treated there as parametric models. Adamatzky also discussed the complexity of identification of CA's and presented sequential and parallel algorithms for computing the local transition table. Another approach is by Yang and Billings [93] where a multiobjective genetic algorithm is introduced to identify both the neighbourhood and the rule set in the form of a parsimonious Boolean expression for both one- and two-dimensional CA's. The method is quite simple, but in contrast to Adamatzky's approach, the situation where the observed patterns are corrupted by static and dynamic

noise is considered. Unfortunately, only some abstract models are presented as applications and they are not related to any practical model of any use.

As for the structure identification, the number of the existing attempts to solve this problem is also very limited. To the best of the author's knowledge, only some works by Mitchell and the EVCA (evolving CA) group [58], Crutchfield and his co-workers [19], as well as by Koza [51], can be qualified as attempts to address this question. Evolutionary algorithms and genetic programming are respectively applied to find binary CA's performing non-trivial computational tasks. The input to the computation is encoded as an initial configuration, the output is the configuration after a certain number of time steps, and the intermediate steps that transform the input to the output are considered to be the steps in the computation. The "program" emerges through "execution" of the CA rule in each cell. The examples presented therein consider, however, computational problems which are far from applications. This idea was pursued in the monograph by Sipper [73], where genetic algorithms were use to find binary rules for non-uniform CA's (i.e. automata in which each cell possesses its own set of rules which may differ from those for other cells) so as to solve some non-trivial computational problems such as the synchronization, ordering, rectangle-boundary and thinning tasks.

The above brief overview of the state-of-the-art in inverse CA problems indicates that more attention should be paid to state and/or parameter estimation of CA models, as from an engineering point of view, the use of the existing scarce methods is restricted owing to excessively severe limitations imposed by those methods or involved computational difficulties. The central question addressed in this thesis is thus whether we can mimic physical and ecological processes by automatically creating CA's that exhibit characteristics such as those manifested by their natural counterparts. We prove this assertion by employing some tools of systems identification, optimization methods and genetic programming, and show that this task can be solved with relative ease and the modelling capabilities of CA models can be substantially improved. As a result, the limitations of the existing approaches, which seem to be the main impediments to persuade engineers to apply CA's in practice, can be circumvented.

This monograph constitutes an attempt to meet the needs created by practical applications through the development of new techniques and algorithms or adopting methods which have been successful in akin fields of dynamic systems identification. It is an outgrowth of original research conducted, among other things, in the framework of a Polish-French project Polonium with the Laboratory of Systems Analysis of the University of Perpignan (France) and a European project which gathered scientists from France, Spain, Italy and Poland. The author believes that the approach outlined here has significant practical and theoretical advantages which will make it, with sufficient development, a versatile tool in numerous CA design problems encountered in engineering practice.

# Chapter 2

# A CA approach to ecological modelling

## 2.1. Problem statement

The behaviour of many natural phenomena can be seen as a result of changes triggered by interaction of species living on the same territory over the same period of time. Therefore, we may use the word *succession* in the ecological modelling context. For most ecologists, succession involves changes in natural systems and the understanding of the causes and directions of such changes. The main problem here is: How to determine mechanisms that are associated with succession? Two aspects can be distinguished here: succession from individual attributes and an ecosystem succession.

The succession from individual attributes was studied in [56, 23, 39, 64, 16, 70, 71] and focuses on the following issues [70]:

- formulation of mathematical models that can be manipulated to explore the long-term theoretical implications of interactions of the dominant organisms (e.g. trees in the case of forests), and that can be used to test theory against data;
- an analysis of competition as a main mechanism in species composition dynamics;
- a recognition of species interactions in a community;
- a denial of the climax community concept and the recognition of the non-equilibrium nature of the vegetation that comprises most modern landscapes.

An ecosystem succession is an alternative view of succession that emphasises the dynamics of the ecological system as an integrity. In this view, the ecosystem (not a collection of changed populations) is viewed as the main object of study. The formulation of succession theory for ecologists who are concerned with ecosystem succession has several objectives and features [53, 62, 54, 57]:

- the recognition of regularities in ecosystem patterns, and an interest in the development of a theory that would allow succession to be viewed as a regular process in a number of different ecological systems;

- the recognition of processes involving more than plant-environment or plant-plant interactions for ecosystem dynamics;
- the incorporation of indices, methods and approaches from engineering, applied mathematics, cybernetics and general systems theory in studies of ecological systems;
- the influence of human activities on the ecosystem behaviour.

A success in forming ecological models lies in a correct perception of how to combine scales of phenomena, space and time. The investigation concentrates on the use of models to develop theory and to settle problems in our understanding of succession.

The theories of dynamical systems and partial differential equations are offered as various models for realistic problems that often exhibit spatial and temporal changes. However, one of the significant difficulties lies in both solving and implementing such equations. Indeed, most natural systems have discrete nature and their evolution generates very complex behaviours. A discrete mathematical idealisation, based on CA's, provides an alternative approach to modelling some part of population dynamics like spatial expansion phenomena, neighbour influence, etc. [89, 90].

For landscape ecology problems, as well as for various complex spatio-temporal systems, one may decouple modelling by considering on one hand the phenomena which are more specific to the time variable and, on the other hand, those related to space. Roughly speaking, we can globally consider this decoupling by assimilating biological layers to time dynamics (tree dynamics) and geographical layers to space dynamics (CA's). Specific simplifications will depend on the investigated phenomenon and associated variables.

### 2.1.1. Computer models of forest succession

It was in the beginning of the 20-th century that scientists started thinking about modelling forest ecosystems. Clements took a holistic view of ecosystem dynamics [13] whereas Gleason proposed an individualistic view of succession [29]. We can think of each modelling approach as a view of different facets of the same reality.

A commonly used modelling paradigm for populations would be a formulation of the following form:

$$\frac{\mathrm{d}N}{\mathrm{dt}} = f(N,t), \tag{2.1}$$

where $N$ is the number of individuals in the population and $t$ stands for time. Although this formulation ignores the effects of sex ration on reproduction or the effect of age structure on mortality, it can be successful as a basis for a general theory of population dynamics.

In building a mathematical model of forest succession, two fundamental problems must be overcome:

**Tab. 2.1.** Explanations of the growth of young forests at different scales [70, 71].

| Scale | Mechanism | Explanation |
|---|---|---|
| Very small | Photosynthesis | Within the leaf, the trees maintain a biochemical factory that converts $CO_2$ and $H_2O$ to sugar by using light energy to drive this synthesis. |
| Stomata | Energy balance | The stomata of the leaf must optimise the heat, water, and $CO_2$ balance of the forest. The resistance of the stomata to the inward diffusion of $CO_2$ tends (in some cases) to be a determining factor on the rate of photosynthesis. |
| Leaf | Leaf geometry | The shape and orientation of a leaf can have a pronounced effect on the ability of the plant to take in $CO_2$ to feed the photosynthesis machine while giving up little $H_2O$. Plants that do this well have a high water-use efficiency (ratio of $CO_2$ fixed/$H_2O$ lost). |
| Leaf layers | Light extinction in the vertical | The orientation and layering of the leaves of a tree can alter the rate at which the canopy captures light. Some arrangements of leaves can be quite inefficient relative to others. |
| Tree shape | Light extinction in the horizontal | Tall, thin trees are efficient in capturing light at high latitudes (where the sun angles are flat); they also capture light in the morning and evening (when moisture relations may be more favourable). The latter can improve the water-use efficiency at lower latitudes. Other geometries have other advantages. |

1. the appropriate scales for understanding the mechanisms of forest succession are not well-known;

2. new observations, either to elucidate ecological mechanisms or to test the models, most likely will be collected on time intervals that are short compared with the scale of the actual dynamics of forest ecosystems.

The problem is with the scales of the corresponding mechanisms for a young forest that is growing back after some distribuance to increase over time. Table 2.1 [70] summarizes several possible explanations (at different scales) for causes and variations in this increase. At a given place and for a given forest, any of these mechanisms may be important over some time scale. Unfortunately, all of these mechanisms cannot be included in a single model of forest growth (when one considers leaf energy balance and the higher order geometry of tree canopies in the same model, the consideration of even a few leaves and their interactions is difficult). Additionally, the number of interactions among the leaves increases as a function of the number of leaves squared. In practice, it is extremely hard to consider a large tree by simulating one leaf at a time.

In spite of the potential difficulties in bringing certain mechanisms into forest succession models, many of the models of forest dynamics have similar features. These features are the same in a general sense, but they often differ in their details

of formulations. Some models are restricted by their underlying assumptions of cases in which one or more of these featured mechanisms are constant or (at least) predictable. For example, a model that assumes the equal spacing of trees may be restricted in use to applications on plantations. The consistent features that are found across forest dynamics models are as follows [71]:

1. *Recruitment:* sprouting, seed production, seed dispersal, germination, and growth of seedlings until the young plants are large enough to be thought of as trees.

2. *Growth:* height and diameter increase of trees.

3. *Geometric competition:* spatial interactions of trees related to the actual geometry of the tree structures. Generally, larger individuals are favoured in the geometric competition.

4. *Resource competition:* growth-limiting factors that may limit the development of all the trees in a forest at a given site.

5. *Mortality:* the death of individual trees.

These features are treated with great detail in some forest dynamics models and are absorbed into the model parameters in others [71]. On the basis of these features, we can make a classification of models e.g. the one specified in Table 2.2. Phenomena typically included in the various models are listed there, with two asterisks (**) indicating strong emphasis, one asterisk (*) indicating some emphasis, and a blank indicating that there is little or no emphasis on the particular phenomena.

The first type of the presented models, developed from the late 1960s and modified by Hegyi [35] in 1974, simulates both the annual height growth and the diameter growth of each tree (the *tree* model). Competition decreased the maximum expected growth by using a geometric competition index. The model uses several regression equations for height and diameter increments at regular 25.4 cm intervals along the stem as a function of competition, age and size of the tree and the site index. Mortality is applied at 5-year intervals. The model operates on trees of one species (*mono* species) which grow in the same conditions (*even* age structure). Additionally, it belongs to *spatial* models, which means that each investigated tree does not have contact with another. This feature was improved by Sullivan and Clutter [75] who were looking for both a statistically reliable estimate of the volume of wood per unit area that is expected at a given time and a simultaneous estimate of the basal area of the stand. These two variables are both functions of the number of trees and individual tree geometry. The resulting approach is very useful for predicting the plantation response of species that were not well-known. In 1974 Solomon [74] extended this approach by introducing *mixed* species. His model also includes several options that involve harvesting and thinning.

Subsequent models introduce more features at the costs of complications:

- In 1975 Mitchell [58] provided a good example of how a detailed geometric model can be used to simulate monospecies forests.

- For prediction of the statistical distribution of tree diameters over time, Suzuki and Umemura introduced in 1974 [76] partial differential equations for the change in the mean (growth of an individual tree), its variance (a diffusion process about the moving mean), and a partial differential equation for the probability of mortality for trees as a function of the diameter and time.

- Ek and Monserud [24] proposed the most elaborate forest dynamic model that uses the individual tree growth as an underlying paradigm: the FOREST model. The model simulates growth and reproduction of mixed species and even or uneven-aged stands, and it includes natural regeneration and growth, as well as a variety of management applications.

- In 1975 Horn [38] viewed the species dynamics of a forest as a problem in determining the likelihood of whether a canopy tree of a given species would be replaced by the same or another species in the next generation. He modelled this view of nature as a first-order Markov process.

- In 1992 Antonovski [4] proposed a probabilistic model of forest fire dynamics.

The most promising approaches were outlined by Botkin [8, 7] and reiterated by Waggoner [85], Shugart and West [72] as the so-called gap models. Gaps are openings encountered in forest canopies. Basically, this model simulates the annual change on a small plot by calculating the growth increment of each tree, by tabulating the addition of new samplings to the stand (both from seeds and by sprouting) and by tabulating the death of trees. All of the processes are considered to be stochastic. The philosophy that underlies the model construction is to strive to represent dynamic phenomena by using general equations that can be parameterised from a knowledge of basic physiology, morphology or forestry; thus they do not require elaborate data sets for parameter estimation. The usual approach is to reserve data for independent tests on the models.

### 2.1.2. Gap models

The concept of a forest gap or a *gap phase* is attributed to Watt [87] who used the term to refer to a patch in a forest created by the death of a canopy tree. Gaps become localised sites of regeneration and subsequent growth. They range in size from small openings created by the death of a single branch to a large-scale blow-down caused by catastrophic disturbances such as storms; gaps may also be created by fires or aggregated insect outbreaks. Studies by Watt and many others indicated that a mature forest ecosystem could be seen as a relatively consistent average of the responses of the dynamics of such gaps.

The ordinary differential equation defining a gap constitutes a simplistic model of reality and it represents a growth process which does not take account of

**Tab. 2.2.** Classification of forest dynamics models [70].

| Model classification | | | | Phenomena | | | | | Example |
|---|---|---|---|---|---|---|---|---|---|
| Category | Age structure | Diversity | Space | Regeneration | Growth | Geometric competition | Resource competition | Mortality | |
| tree | even | mono | spatial | | ** | ** | | * | Hegyi [35] |
| tree | even | mono | nonspatial | | ** | | ** | | Sullivan [75] |
| tree | even | mixed | nonspatial | | * | * | | ** | Solomon [74] |
| tree | mixed | mono | spatial | | ** | ** | | * | Mitchell [58] |
| tree | mixed | mono | nonspatial | ** | * | | * | ** | Suzuki [76] |
| tree | mixed | mixed | spatial | * | ** | ** | * | ** | Ek [24] |
| tree | mixed | mixed | nonspatial | ** | | ** | | ** | Horn [38] |
| gap | mixed | mixed | nonspatial | ** | | | | ** | Waggoner [85] |
| gap | mixed | mixed | spatial | ** | ** | * | * | * | Shugart [72] |

very important parameters characterising vegetation such as water, light and temperature terms, or the crowding stress. The modifications aiming at circumventing this disadvantage involve, however, using non-linear partial differential equations (PDE's) for which any analysis (and even solution) becomes extremely hard [71]. The main difficulty lies in the fact that the ordinary differential equation in context does not reflect spatial phenomena which are crucial in accurately describing the crowding stress and its influence on light terms, which is related to the determination of the space occupied by particular individuals. Consequently, the main problem considered here is how to simplify the accurate (but cumbersome) gap model described by a PDE without deteriorating the quality of approximation. For that purpose, our idea here is to employ CA's, as they are completely discrete models and thus they are perfectly suited to describe natural phenomena like seed spread. Since the space is represented there by a regular lattice, we can simply define interactions with neighbourhood, etc.

As with most individual tree-based models, gap models require a basic equation to increment the size of each tree on the modelled stand. For all gap models proposed up to now, there have been two approaches to develop these equations. The initial approach (and the one used in most gap models) was set forth by Botkin [8, 7]. The growth equation is developed by assuming that the volume of a tree is a function of its diameter $D$ squared times its height $H$, and that the tree growth is based on the annual volume increment. The tree volume increment is governed by the following equation [70]:

$$\frac{\mathrm{d}[D^2H]}{\mathrm{d}t} = rL_a\left(1 - \frac{DH}{D_{\max}H_{\max}}\right) \tag{2.2}$$

where $r$ is a growth rate parameter, $L_a$ is a tree's leaf area, $D$ is the diameter at the breast height, $H$ is the tree height, and $D_{\max}$ and $H_{\max}$ are the maxima for the diameter and the height, respectively.

The basic growth equation (2.2) can be simplified by noting that the height $H$ is a function of the tree diameter $D$. One useful formulation for this relationship is [70]:

$$H = 137 + 2\left(\frac{H_{\max} - 137}{D_{\max}}\right)D - \left(\frac{H_{\max} - 137}{D_{\max}^2}\right)D^2, \tag{2.3}$$

where the constant 137 [cm] is set as the mean breast height of persons measuring the tree diameters. If it is further assumed that $L_a \sim cD^2$, where $c$ is a constant, then eqn. (2.2), on substitution and differentiation, becomes [71]

$$\frac{\mathrm{d}D}{\mathrm{d}t} = \frac{GD\left(\dfrac{1 - DH}{D_{\max}H_{\max}}\right)}{274 + 6\left(\dfrac{H_{\max} - 137}{D_{\max}}\right)D - 4\left(\dfrac{H_{\max} - 137}{D_{\max}^2}\right)D^2} \tag{2.4}$$

Equation (2.4) is used in most of the currently published gap models as the optimal growth equation.

When we know how to model the growth phenomenon, we are now in a position to build a population. To simplify the analysis, the working space is divided into smaller regular parts called *patches*. Like in nature, on each patch we examine all individuals living on it, e.g. if we operate on a patch of dimension 20m×20m and we know that on this selected patch we have 3 trees of one species and 8 of another, we must solve 3 equations tuned for the first species and 8 equations prepared for the second species. Finally, we can say that each individual is represented by one growth equation corresponding to this particular species.

All the currently used gap models incorporate some additional parameters like solar conditions, temperature effects, nutrient cycling, moisture outcomes, death and birth rules. The corresponding modifications in the model for these phenomena are known, cf. [70] for more details. The idea of gaps is currently developed by the National Gap Analysis Program. On the web page `http://www.gap.uidaho.edu/` many references, reports and publications are published.

A major problem with forest models is that both the foregoing original models and the others presented later, offer either a simple form for a very limited class of applications, or a very complex form for more detailed and universal models. A good example is a very complete approach proposed by Brufau [9] who shows how hard using detailed gap-like model described by PDE's can be. In her Ph.D. thesis she modelizes changes in the following variables: average density of the biomass of all species together, amount of water in the ground (humidity) and absolute humidity of the atmosphere. The model operates on a one-dimensional space domain composed of rectangles of the size representing the minimal distance which is covered by humidity (assuming that the velocity is known and constant). However, the resulting systems of equations are too cumbersome to be useful in practice.

In turn, the spatial model by Tongeren and Prentice [80] shows how to simplify the problem and to retain or even to improve the precision, but, unfortunately, only for shrubs (or another short vegetation). The FIRESUM [47] model indicates some possibilities to split the underlying model into a continuous part evaluating the growth process and three stochastic, discrete parts representing birth rules, the death process and fire dynamics. This approach partially simplifies the formulation of the model, but unfortunately, it still offers involved descriptions of the spatial interaction of trees.

## 2.2. Common CA models for forest dynamics

Most well-known CA forest models describe some characteristic phenomena occurring in ecosystem dynamics, but they cannot be checked through real data due to a strongly stochastic character. In what follows, two well-known CA models of forests will be presented. Both operate on two-dimensional lattices and similar state sets, and only the approach used to determine the transition functions clearly makes them different.

### 2.2.1. CA forest model by Green

A good example of modelling forest vegetation by CA's is represented by a known forest growth model originated from [31]. The applied approach is compatible with both pixel-based satellite imagery and with quadrat-based field observations. It also enables processes that involve a movement through space (e.g. fire, dispersal) to be modelled in "natural" fashion.

To see how easily the CA idea can be brought to bear on real systems, consider a system whose states are of the form $(s, \tau)$, where $\tau$ denotes *the time since fire burnt out the cell* and $s$ takes one of the values: *bare earth* (E), *grass* (G), *woodland* (W), and *closed forest* (F). The CA in context consists of a fixed array which represents a landscape and in which each cell represents an area of a land surface. The neighbourhood $N(c)$ of a cell $c$ is of von Neumann type (see Fig. 1.4). Define state transitions rules as follows:

$$
\begin{array}{lcll}
s_t(c)=(E,0) & \Longrightarrow & s_{t+1}(c)=(G,1), & \\
s_t(c)=(G,4) & \Longrightarrow & s_{t+1}(c)=(W,5), & \\
s_t(c)=(W,49) & \Longrightarrow & s_{t+1}(c)=(F,50), & \\
s_t(c)=(s,\tau) & \Longrightarrow & s_{t+1}(c)=(E,0) & \text{if a fire ignites nearby,} \\
s_t(c)=(s,\tau) & \Longrightarrow & s_{t+1}(c)=(s,\tau+1) & \text{otherwise.}
\end{array}
$$

Starting with an arbitrary "landscape" consisting of a two-dimensional grid of cells in random states, the rules quickly produce interesting patterns.

For a better representation of complexity in forest vegetation, this deterministic model can be changed into a probabilistic one (the model above always uses the transition probability $p$ equal to unity). The new probabilistic transition function can be defined as follows:

$$
\begin{array}{llcll}
s_t(c)=(E,\tau) & \text{and } \tau \geq 0 & \Longrightarrow & s_{t+1}(c)=(G,\tau+1), & \\
s_t(c)=(G,\tau) & \text{and } \tau \geq 3 & \Longrightarrow & s_{t+1}(c)=(W,\tau+1) & \text{with prob. } 1/2, \\
s_t(c)=(W,\tau) & \text{and } \tau \geq 44 & \Longrightarrow & s_{t+1}(c)=(F,\tau+1) & \text{with prob. } 1/2, \\
s_t(c)=(s,\tau) & & \Longrightarrow & s_{t+1}(c)=(E,0) & \text{with prob. } 1/2 \text{ if a fire ignites nearby,} \\
s_t(c)=(s,\tau) & & \Longrightarrow & s_{t+1}(c)=(s,\tau+1) & \text{otherwise.}
\end{array}
$$

These rules mean that cells with state values 1 and 2 remain intact for three and forty four time steps, respectively. The model is then a delayed CA as defined before cf. Definition 1.2, with $D = \{0, 3, 44\}$ and $\delta : \{0, 1, 2, 3\} \longrightarrow D$ which satisfies $\delta(0) = \delta(3) = 0$, $\delta(1) = 3$ and $\delta(2) = 44$.

On a lattice of 200×100 cells and starting with a given initial distribution, we can observe the CA evolution shown in Fig. 2.1. For this simulation, periodic boundary conditions were used. The light grey colour denotes *bare earth* ($s_t(c) = (E, \tau)$), grey colour stands for *grass* ($s_t(c) = (G, \tau)$), dark grey denotes *woodland* ($s_t(c) = (W, \tau)$), black denotes *closed forest* ($s_t(c) = (F, \tau)$), whereas the white

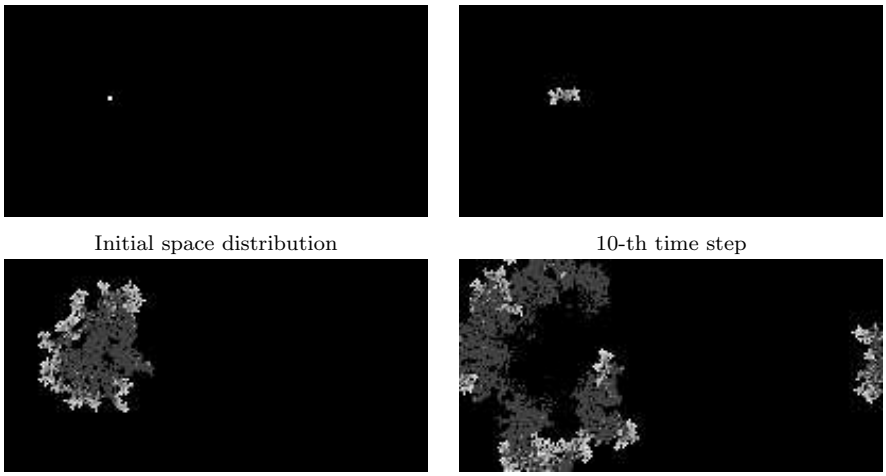



**Fig. 2.1.** Evolution of Green's forest-fire CA.

colour reflects fire centres.

As we can see, simulation results reveal some cyclic behaviour which represents regeneration of the forest after fire destruction. For instance, "forest zones" quickly develop, even though the model contains no assumptions about the environment or site preferences of the vegetation. Admittedly, this model is a mere caricature of a true forest succession, but it is only a short jump to management models taking (say) satellite data as inputs. It can be a good illustration in order to present some trends in ecosystem dynamics.

### 2.2.2. Stochastic forest fire model

The stochastic forest fire model was introduced by Dossel and Schwabl in 1994 [22]. In much the same way as in Green's CA forest model, the death of vegetation constitutes a cause of fire. The CA consists of a fixed square lattice which represents a landscape with periodic boundary conditions. The neighbourhood $N(c)$ of a cell $c$ is of von Neumann type (cf. Fig. 1.4). The states associated with each cell correspond to a vegetable area and take the following values:

0 – an empty site (cf. bare earth in Green's model);

1 – a site occupied by a tree;

2 – a site occupied by a burning tree.

The transition rules are defined as follows:

- an empty site sprouts a tree:

$$s_t(c) = 0 \Longrightarrow s_{t+1}(c) = 1$$

with tree growth probability $p_1$,

- a tree catches fire:

$$s_t(c) = 1 \Longrightarrow s_{t+1}(c) = 2$$

with probability $(1 - p_2)$ if at least one nearest neighbour tree is burning, $p_2$ being the immunity probability,

- a tree catches fire:

$$s_t(c) = 1 \Longrightarrow s_{t+1}(c) = 2$$

with probability $p_3(1 - p_2)$ if no nearest neighbour tree is burning, $p_3$ being the lightning probability,

- a burning tree burns down and becomes an empty site:

$$s_t(c) = 2 \Longrightarrow s_{t+1}(c) = 0$$

with probability 1.

Playing with parameter values, we can obtain qualitatively different behaviours of the "forest".

**Example 2.1**
On a 160×100 square lattice and starting with a given initial distribution, we can observe the evolution of the forest fire CA shown in Fig. 2.2. The light grey colour denotes empty cells ($s_t(c) = 0$), the dark grey colour signifies trees ($s_t(c) = 1$), whereas the white colour reflects fire ($s_t(c) = 2$).

Depending on parameter values, we can observe three regimes [22]:

- **spiral-shaped fire fronts**
When $p_2 = p_3 = 0$ and $p_1 \ll 1$, clusters of isolated forests of (non-burning) trees emerge within empty regions and these clusters persist until they grow into other clusters that are burning. The fire fronts separate the empty and forested areas. An example of that behaviour is shown by the first group of four pictures in Fig. 2.2. For simulations, the following values were used: $p_1 = 0.02$, $p_2 = 0$ and $p_3 = 0$.

- **self-organized critical (SOC) state**
When $p_2 = 0$, $p_1 \ll 1$ and $p_3 \ll p_1$, two time scale separations occur: tree growth occurs much more frequently than lightning strikes while forest clusters burn down faster than they grow. This results in an SOC state in which clusters of forests of all sizes are burning. An example of that behaviour is shown by the second group of four pictures in Fig. 2.2. For simulations, the following values were used: $p_1 = 0.05$, $p_2 = 0$ and $p_3 = 0.00025$.

- **percolation transition**
  When $p_3 = 0$, forests spread as $p_2$ increases and eventually there is a zero density and a percolation-like phase transition takes place at a critical value of $p_2$, which depends on $p_1$. An example of that behaviour is shown by the last group of four pictures in Fig. 2.2. For simulations, the following values were used: $p_1 = 0.2$, $p_2 = 0.52$ and $p_3 = 0$.

Despite very simple rules, we have obtained a complex behaviour (spiral-shaped fire fronts) known from observations of real forest fire and regeneration processes.

## 2.3. Coupling gaps with CA's: Hybrid models

Along with the increasing interest in the results coming from the work on gap models, researchers have begun to look for new manners of improving the quality of model approximation in known modelling methods. One of very promising ways is to apply CA's. Because the research on tree growth dynamics and forest dispersion is based on the analysis of interactions between trees and their neighbourhood, CA's seem especially suited as a modelling tool for this process. The simplest method to use CA's in ecological models is to represent each individual (tree) by one cell and to update vegetation parameters on the basis of gap equations. In the case considered, such a CA consists of a lattice of cells characterised by their state values representing a given parameter of vegetation. The CA evaluates the state value for each cell and passes this information to a gap model. Additionally, it evaluates the coefficients accounting for additional discrete phenomena.

The delineated approach was already applied in [33, 49] where it proved to be useful to improve simple and precise evaluation of space dynamics (seed transport, birth and death rules, etc.). The main problem in those works consists in appropriately combining a completely discrete CA with the continuous gap model. An additional problem is how to simplify the interface of the complex model for an average user who is not interested in understanding all mathematical problems involved, but he or she knows qualitative characteristics of the studied ecosystem.

In what follows, an original idea to solve both the above-mentioned problems is outlined.

### 2.3.1. Coupled model implementation

The coupling process for CA's and gap models leads to a more accurate problem description, but is also a source of many difficulties: What are we to describe by ordinary differential equations (ODE's) and what by CA's? What is the size of the part described by ODE's? How to combine the outputs of discrete and continuous models?

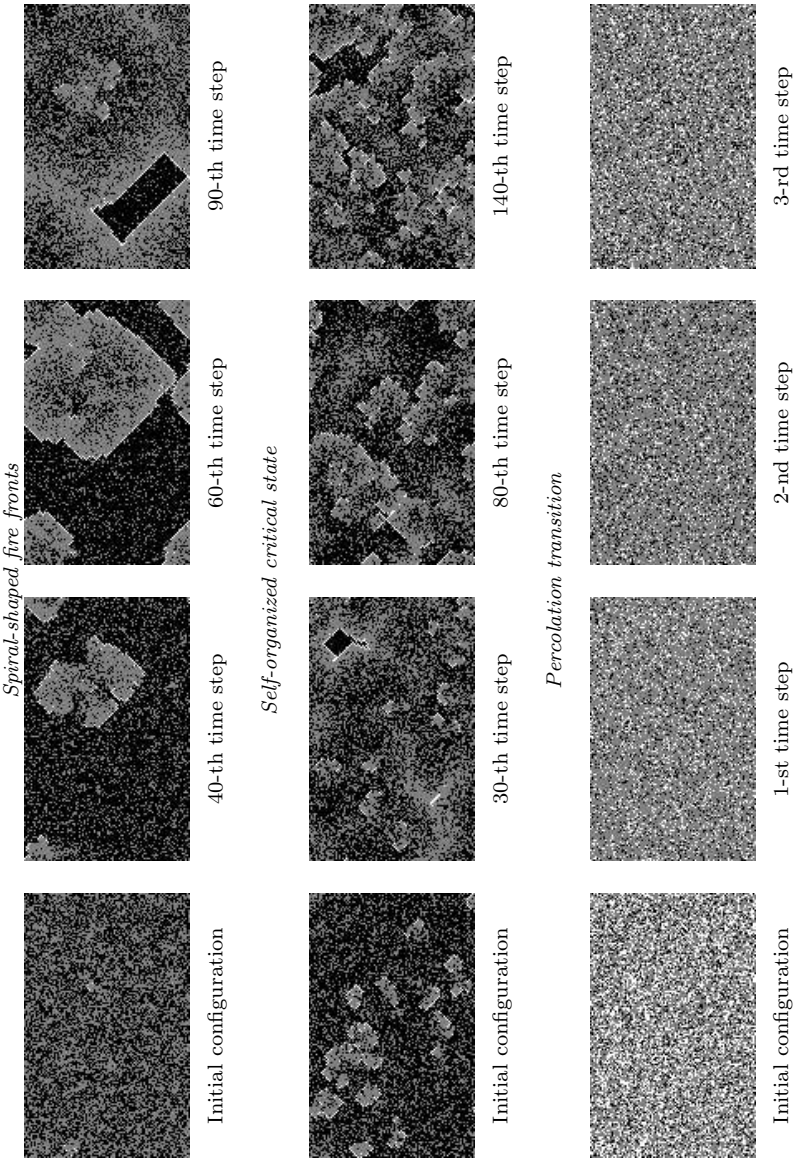



**Fig. 2.2.** Evolution of Dossel and Schwabl's stochastic forest fire CA.

**Tab. 2.3.** Comparison of classical and modified gaps ($n_S$ is the number of species which live on the studied terrain and $k_i$ denotes the number of individuals of species $i$ on a given space unit).

| Item | Typical gap | Modified gap |
|---|---|---|
| One space unit represents: | individual | patch |
| Number of ODE's per space unit: | $\sum_{i=1}^{n_S} k_i$ | $n_S$ |
| One ODE describes: | growth of an individual | mean growth of the population for selected species |

#### 2.3.1.1. ODE implementation

As we know (cf. Section 2.1.2) most of gaps are described by ODE's or PDE's. As regards our coupled model, we do not need to complicate the growth model usually described by first-order ODE's. A necessary complication is often used in order to implement spatial mechanics like seed transport, crowding effect, etc. If this is the case, this is a task for a CA.

Most gap models describe each individual by one equation, and this assumption is frequently used for coupling with CA's. Alternatively, some models, called *modified gap models*, operate on patches, and not on individuals. A patch is a given surface area used to specify a piece of space for determination of local evolution. Because the size of a patch is constant, we can say that the workplace of the modified gap is a regular lattice. Of course, equations which characterise the local evolution are not exactly the same as typical gap equations. In this case, we operate on $n_S$ equations per patch (where $n_S$ is the number of species), and not on one equation per individual. What is more, each equation describes the mean growth dynamics for selected species (see Table 2.3).

Application of the modified gap model allows us to simplify the model description, to improve the performance and constitutes some kind of simplification. Note, however, that we cannot increase the patch size too much, because this may be a source of unacceptable modelling errors. We are also faced with other problems for which solutions are not simple. Typical gap models are based on the description of individuals, but in order for them to be compared with real data, we need to compute mean values from some parts of the terrain. Furthermore, in typical gap models we make some errors regarding initial conditions when we change the measured data to the number of individuals and their sizes (the value of growth level), and secondly, when we compute mean values for comparison with measured data. And finally, the main problem is that we have to operate on too many ODE's per patch and we must tune their quantity adequately to birth and death rules steered by the CA in context.

A modified gap model proposed here comes from original works of the au-

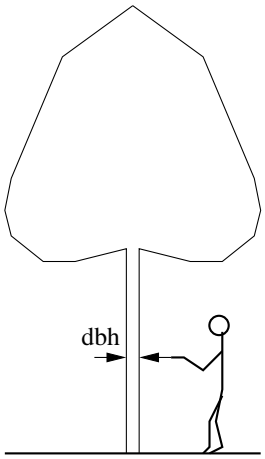



**Fig. 2.3.** Way of measuring the tree diameter (at the breast height).

thor within the framework of the Lucifer project*. It is developed and validated for some Mediterranean species of trees and shrubs. The growth process is represented by the evolution of the tree diameter, called the diameter at the breast height (or *dbh* for brevity, cf. Fig. 2.3) and denoted by $D$. Competition rules and climate characteristics are included in the formulation. The patch size is related to the size of a pixel on a map obtained from a satellite. The model describes vegetation changes by estimating periodically the diameter increments of each tree on a prescribed plot. The tree diameter is used to determine the tree height and leaf area profiles. The growth process is completed by considering four factors:

- the light effect,
- the crowding effect,
- the water stress, and
- the temperature reduction factor.

The **light effect** reflects the importance of light conditions for vegetation dynamics and is given as

$$r_L = \begin{cases} 1 - e^{-4.64(A_L - 0.05)} & \text{for shade tolerant species,} \\ 2.24(1 - e^{-1.136(A_L - 0.08)}) & \text{for shade intolerant species,} \end{cases} \tag{2.5}$$

where $A_L = \phi e^{-0.25 \sum_i L_{ai}}$ is the available light for a given class of trees, $\sum_i L_{ai}$ is the shading leaf area index defined as the sum of the leaf areas of all higher trees

*LUCIFER, this is the acronym of the European Programme CEE ENV 4-CT96-0320: ***L****and* ***U****se* ***C****hange* ***I****nteractions with* ***F****ire in Medit****e****rr****r****anean landscapes*, 1997–2000.

on the plot. In a typical gap, heights are obtained from equation (2.15) and then we can compute $\sum_i L_{ai}$ for highest trees. In the presented case we use a medium leaf area for specified species ((2.23) or (2.25)).

As for $\phi$, it is the annual solar illumination which is assumed to be of the form [70, 47, 71]

$$\phi = \frac{E}{E_H}, \tag{2.6}$$

where $E$ denotes the annual solar energy falling on the slope and then given by [47, 71]

$$E = \int\limits_{\text{year}} \int\limits_{\omega_s}^{\omega_r} I(\omega) \cos\Theta \, \mathrm{d}\omega \, \mathrm{d}j, \tag{2.7}$$

$E_H$ being the annual solar energy which would fall on the projected horizontal plane. $E$ is considered for $E_H$ when $\omega = 0$. In the previous expression, $I(\omega)$ is calculated from the following formula [47, 71]:

$$I(\omega) = 1230 \exp\left\{ \frac{-1}{3.8 \sin(h + 1.6(\pi/180))} \right\}, \tag{2.8}$$

where $h$ is an angle which indicates the solar height.

The angles $\Theta$ and $\omega$ involved in (2.7) correspond to the direct radiance incidence angle and the angle per hour, respectively. They are related to $h$ through the formula [47, 71]

$$\begin{aligned} \cos\Theta = {} & \sin\delta \sin\phi \cos\varphi - \sin\delta \cos\phi \sin\varphi \cos\gamma + \cos\delta \cos\phi \cos\varphi \cos\omega \\ & + \cos\delta \sin\phi \sin\varphi \cos\omega \cos\gamma + \cos\delta \sin\phi \sin\gamma \sin\omega, \end{aligned} \tag{2.9}$$

$$\sin h = \sin\delta \sin\varphi + \cos\delta \cos\phi \cos\omega, \tag{2.10}$$

where $\phi$, $\varphi$, $\delta$ and $\gamma$ are some needed angles which determine the position of the surface (slope, decline, etc.) with [47, 71]

$$\delta = 23.15 \frac{\pi}{180} \sin\left( 0.986 \frac{\pi}{180} (284 + N_j) \right), \tag{2.11}$$

where $N_j$ is the number of trees in the patch in the year $j$. The above values of $\phi$ should be computed before simulation for each patch and used as a constant topography map. For most cases, we can take $\phi = 1$ which corresponds to a flat horizontal terrain.

The **crowding effect** determines how the actual size of the biomass affects the growth process. It is given by [47]

$$r_N = 1 - \frac{B}{B_{\max}}, \tag{2.12}$$

where $B$ and $B_{\max}$ are the current biomass and the maximum reachable biomass,

respectively. This formula can be improved by considering, if possible, $r_N = 1 - (B_a/B_{a\,\max})$ where $B_a$ is the reachable basal area and $B_{a\,\max}$ denotes the maximum reachable basal area.

The **water stress** represents the influence of the water resource level on the growth process and it is given by [47]

$$r_\omega = 1 - \left(\frac{1 - A_p}{1 - W_s}\right)^2, \tag{2.13}$$

where $A_p = A_E/P_E$ is a ratio which indicates the ability of trees to be driest and $A_E$, $P_E$ are the actual and potential evapotranspirations, respectively. We have $A_p \in [0, 1]$. $W_s$ is the lower limit of the ratio $A_p$ for which the water stress is reduced to zero.

The **temperature reduction factor** describes the influence of the local temperature on the growth process and it is given by [47]

$$r_{\mathrm{Degd},i} = \frac{(D_5 - D_{5\,\min,i})(2D_{5\,\mathrm{opt},i} - D_{5\,\min,i} - D_5)}{(D_{5\,\min,i} - D_{5\,\mathrm{opt},i})^2} \tag{2.14}$$

where

$D_5$ – the heat sum of the site (a common value for all species interacting on the same patch),

$D_{5\,\min,i}$ – the minimum heat sum bearable by species $i$,

$D_{5\,\mathrm{opt},i}$ – the optimum heat sum bearable by species $i$.

To give the final model of growth, some preliminary calculations are needed. If $H_0$ denotes the initial *dbh* of the tree, the tree height $H$ is related to $D$ by the following expression [47]:

$$H = H_1 + b_2 D - b_3 D^2, \tag{2.15}$$

where

$$b_2 = 2\frac{H_{\max} - H_1}{D_{\max}} \tag{2.16}$$

and

$$b_3 = \frac{H_{\max} - H_1}{D_{\max}^2}, \tag{2.17}$$

$H_{\max}$ and $D_{\max}$ represent the maximum reachable height and diameter, respectively.

The generated model is very familiar to the typical gap equation (2.4) and depends on the kind of species. For *normal vegetation* (i.e. for species whose maximal height is greater than 3 m), the growth process is described by the following expression [92]:

$$\frac{\mathrm{d}D}{\mathrm{d}t} = r_L r_N r_\omega r_{\mathrm{Degd},i} \frac{GD\left(\dfrac{1 - DH}{D_{\max} H_{\max}}\right)}{2H_1 + 3b_2 - 4b_3 D^2}. \tag{2.18}$$

**Tab. 2.4.** Growth coefficients.

| Description | Symbol | Usual range | Unit |
|---|---|---|---|
| Maximum reachable biomass | $B_{\max}$ | $\in [5, 20]$ | kg/m$^2$ |
| Maximum reachable basal area | $B_{a\,\max}$ | $\simeq 0.004$ | m$^2$/m$^2$ |
| Ability of trees to be driest | $A_p$ | $\simeq 0.67$ | |
| Lower limit of tolerance in $A_p$ | $W_s$ | $\in [0, 1]$ | |
| Heat sum for a site | $D_5$ | $\in [2500, 4000]$ | d°/day |
| Minimum heat sum bearable by species | $D_{5\,\min}$ | $\in [2700, 3100]$ | d°/day |
| Optimum heat sum bearable by species | $D_{5\ \mathrm{opt}}$ | $\simeq 6500$ | d°/day |
| Maximum diameter of the tree | $D_{\max}$ | $\in [1, 200]$ | cm |
| Maximum height of the tree | $H_{\max}$ | $\in [20, 2000]$ | cm |
| Initial dbh of the tree | $H_0$ | $\simeq 0$ | cm |
| Maximum age of the tree | $A_{\max}$ | $\in [5, 800]$ | year |
| Breast height | $H_1$ | $= 137$ | cm |
| Solar illumination | $\phi$ | $\in [0, 1]$ | |

In the case where the species is considered as *short vegetation* (i.e. with maximal height $H_{\max} \leq 3$m), we have [92]

$$\frac{\mathrm{d}D}{\mathrm{d}t} = r_L r_N r_\omega r_{\mathrm{Degd},i} \frac{GD\left(\dfrac{1 - DH}{D_{\max} H_{\max}}\right)}{3b_2 - 4b_3 D}, \tag{2.19}$$

where $H_1$ denotes the breast height (a constant fixed at $H_1 = 137$) and the coefficient

$$G = f(D_{\max}, H_{\max}, A_{\max}) \tag{2.20}$$

which also depends on the maximum age of the tree $A_{\max}$ is to be calculated using the formula [92]

$$\begin{aligned} G = \frac{4H_{\max}}{A_{\max}} \Bigg[ & \ln\left(2(2D_{\max} - 1)\right) \\ & + \frac{\alpha}{2} \ln\left(\frac{9 + 2\alpha}{4(4D_{\max}^2 + 2\alpha D_{\max} - \alpha)}\right) \\ & - \left(\frac{2\alpha + \alpha^2}{2\sqrt{\alpha^2 + 4\alpha}} \ln\left(\frac{(3 + \alpha - \sqrt{\alpha^2 + 4\alpha})(4D_{\max} + \alpha + \sqrt{\alpha^2 + 4\alpha})}{(3 + \alpha + \sqrt{\alpha^2 + 4\alpha})(4D + \alpha - \sqrt{\alpha^2 + 4\alpha})}\right)\right) \Bigg] \end{aligned} \tag{2.21}$$

with $\alpha = 1 - (H_0/H_{\max})$.

For that law we assume $H_0 = 0$. The different coefficients introduced for describing the previous model are summarised with their units, normal variation ranges and numerical values for the above-mentioned species in Table 2.4.

The biomass $B$ (in kg) and leaf area $L_a$ (in $m^2$) are usually considered as given by a function of the form $aD^b$, where $D$ is given as a solution to one of the differential equations (2.18) or (2.19). Thus for medium and big size species trees we have

$$B = 0.1193D^{2.3933}, \tag{2.22}$$

$$L_a = 0.16D^{2.129}, \tag{2.23}$$

and for shrubs we get

$$B \simeq D^2H, \tag{2.24}$$

$$L_a \simeq D^2. \tag{2.25}$$

As a basis for tests, seven Mediterranean species known from ecological publications were considered:

- Quercus ilex (marked $Sp_1$),
- Pinus halepensis (marked $Sp_2$),
- Querqus coccifera (marked $Sp_3$),
- Ulex (marked $Sp_4$),
- Erica arborea (marked $Sp_5$),
- Cistus (marked $Sp_6$), and
- Brachypodium (marked $Sp_7$).

The corresponding growth coefficients characterizing the species are listed in Table 2.5.

#### 2.3.1.2. CA implementation

The modified gap model outlined above is well-suited for modelling growth phenomena. An acute problem in its use is that trees are not growing all the time. Sometimes they die and in their place other trees are born. This phenomenon should be included in the formulation in a sense if we aim at producing a model of any practical use. This is because a CA is proposed below in order to implement birth and death rules.

**Tab. 2.5.** Growth parameters for selected Mediterranean species.

| Species | $Sp_1$ | $Sp_2$ | $Sp_3$ | $Sp_4$ | $Sp_5$ | $Sp_6$ | $Sp_7$ |
|---|---|---|---|---|---|---|---|
| $B_{\max}$ | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| $B_{a\,\max}$ | 0.004 | 0.004 | 0.004 | 0.004 | 0.004 | 0.004 | 0.004 |
| $A_{\max}$ | 500 | 125 | 500 | 25 | 30 | 150 | 20 |
| $A_p$ | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 | 0.67 |
| $W_s$ | 0.45 | 0.2 | 0.1 | 0.1 | 0.2 | 0.2 | 0.1 |
| $D_5$ | 3200 | 3500 | 3500 | 3600 | 3500 | 3600 | 3200 |
| $D_{5\,\min}$ | 2700 | 3000 | 2000 | 3100 | 3000 | 3100 | 2700 |
| $D_{5\,\mathrm{opt}}$ | 6500 | 6500 | 6500 | 6500 | 6500 | 6500 | 6500 |
| $D_{\max}$ | 120 | 90 | 10 | 7 | 10 | 5 | 1 |
| $H_{\max}$ | 1500 | 1500 | 150 | 130 | 300 | 130 | 50 |

A formula which describes the **birth law** is based on the number of live seeds. New lives are born from seeds $S_{y,i}$ according to the dependence

$$S_{y,i} = P_i S_{y-1,i} + S_{T,i} - \frac{S_{y-1,i}}{\sum_i S_{y-1,i}} r_{\mathrm{ALS},i} S_P, \tag{2.26}$$

where

$y$ – the year number,

$i$ – the species index,

$P_i$ – the probability of seeds to survive for at least one year,

$S_{T,i}$ – the total number of alive seeds falling on the studied patch,

$S_P$ – the maximum number of possible seedlings including all species,

and

$$r_{\mathrm{ALS},i} = \begin{cases} \exp(-0.8 L_{a,i}) & \text{for shade intolerant species,} \\ \exp(-0.25 L_{a,i} - 1) & \text{for moderate shade tolerant species,} \\ 1 - \exp(-0.25 L_{a,i} - 0.2) & \text{for shade tolerant species.} \end{cases} \tag{2.27}$$

Additionally, we have to define seed transport rules. But to this end, we can simply say that the probability of encountering a seed is inversely proportional to the distance from the mother tree.

At this juncture, we know the appropriate formulae and mechanisms, but the question should be addressed how to describe this by CA rules. For that purpose, let us start with the transport problem. The rule which determines the

**Tab. 2.6.** Specific birth coefficients.

| | $Sp_1$ | $Sp_2$ | $Sp_3$ | $Sp_4$ | $Sp_5$ | $Sp_6$ | $Sp_7$ | Typical range | Units |
|---|---|---|---|---|---|---|---|---|---|
| $P_i$ | 0.01 | 0.4 | 0.01 | 0.86 | 0.01 | 0.73 | 0.1 | $\in [0,1]$ | 100%/year |
| $S_{T,i}$ | 1 | 10 | 1 | 10 | 10 | 20 | 5 | $\in [1,25]$ | Number/year |
| $S_P$ | 2 | 2 | 2 | 2 | 2 | 2 | 2 | $\simeq 2$ | $1/m^2$ |

existence of a seed can be written as follows:

$$\text{seed} = \begin{cases} \text{true} & \text{if rnd}(0\,..\,n) < N_m, \\ \text{false} & \text{otherwise,} \end{cases} \tag{2.28}$$

where $N_m$ is the number of mother trees in the neighbourhood $N(c)$ of size $n$. Of course, this is a good rule for the individual trees case. Because we operate on mean vegetation values per patch, we must take into consideration the intensity on the neighbourhood, and not $N_m$. If we denote by $B_i(c)$ the biomass of species $i$ on patch (cell) $c$, our rule takes on the following form:

$$\text{seed} = \begin{cases} \text{true} & \text{if rnd}(0\,..\,nB_{\max,i}) < \sum\limits_{\gamma \in N(c)} B_i(\gamma), \\ \text{false} & \text{otherwise.} \end{cases} \tag{2.29}$$

Now, when we know whether or not we have seeds, we can determine $S_{y,i}$ so as to decide about the birth of a new life. Detailed characteristics of the introduced species are contained in Table 2.6.

The **mortality** is described by the probability to get dead trees. The death may be due to the fact that the trees have reached their maximum age or a stress event. Thus $P_r$ will be the annual probability to survive for random mortality, and $P_s$ the annual probability to survive for stress mortality. For random mortality, we have

$$P_r = 0.01^{1/A_{\max}}. \tag{2.30}$$

For stress mortality, we have

$$P_s(y_s) = 0.82 P_s(y_s - 1), \tag{2.31}$$

where $y_s \geq 1$ is the number of stressful years. For high trees, if the plant annual growth $\Delta D$ is less than a threshold value $T_{\text{inc}}(i)$, then the surviving probability decreases according to the above law. For shrubs, there is a stress if $\Delta D/\Delta D_{\max} < T_{\text{inc}}(i)$.

The number of stressful years $y_s$ is determined from comparison of $\Delta D$ and $T_{\text{inc}}(i)$. A stressful year is to be considered if $\Delta D < T_{\text{inc}}(i)$. But for shrubs there is a stressful year if $\Delta D/\Delta D_{\max} < T_{\text{inc}}(i)$.

For the considered Mediterranean species, we have the values listed in Ta-

**Tab. 2.7.** Specific death coefficients.

| | $Sp_1$ | $Sp_2$ | $Sp_3$ | $Sp_4$ | $Sp_5$ | $Sp_6$ | $Sp_7$ | Typical range | Units |
|---|---|---|---|---|---|---|---|---|---|
| $T_{inc}$ | 0.001 | 0.01 | 0.001 | 0.001 | 0.005 | 0.001 | 0.001 | $\in [0, 0.05]$ | cm/year |

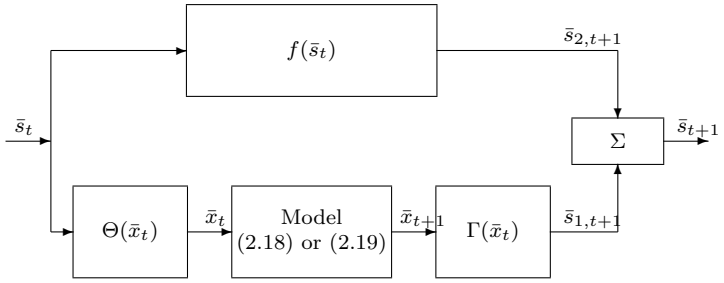



**Fig. 2.4.** Coupled system with discrete states.

ble 2.7.

#### 2.3.1.3. Coupling process

In contrast to the aspects discussed so far, the coupling process is not simple to introduce. The main problem lies in a proper cooperation of a discrete CA model and continuous modified gap ODE's. There are two possible ways of its solution:

1. The state vector should be compatible with discrete CA states. This scheme is very useful from the CA model's point of view. But numerous problems arise when we use too few CA states, or when the conversion from discrete states is badly scaled. In this case the growth process may run too quickly, or stop and cannot continue. This idea is illustrated in Fig. 2.4.

2. The state vector should include real values of vegetation. This idea gives preference to the part which calculates the modified gap equations. There are, however, difficulties with evaluating the CA transition function, as the CA operates on a discrete state set. The structure of the coupled model which implements a real state vector is illustrated in Fig. 2.5.

The choice of the appropriate solution depends on the designer. Both the methods possess their own disadvantages. The intuitive choice seems to be the first method. Since we use a discrete CA, we can discretise the problem. At this juncture, however, we must give a thought to what constitutes the essence of the model. The answer is simple: it is the modified gap model (it approximates the growth process, which is the main purpose of ecological modelling). Thus we must assure best conditions for this part of the model. Of course, the second approach conditions this in a better manner. An attempt at executing this part

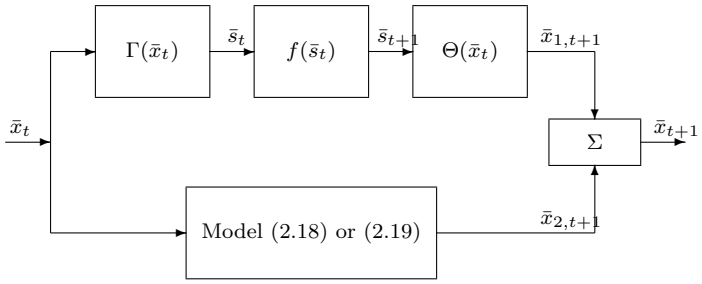



**Fig. 2.5.** Coupled system with continuous state vector.

with discrete state values necessitates an increased size of the state set for reduction of discretisation errors. But too many state values complicate the CA transition function. Finally, we pay a double price: first we introduce errors in the simulation of the growth process and second, we complicate the transition rules.

Consequently, in this work the model with continuous state set (cf. Fig. 2.5) has been chosen. Clearly, this approach is not free of problems, either—the CA must operate based on real values and, furthermore, it must affect a real system. In order to guarantee proper data for evaluation of the CA transition function, we must use a mapping

$$\Gamma : x \in \mathbb{R} \quad \mapsto \quad s \in \mathcal{S}, \tag{2.32}$$

where $\mathcal{S}$ is a discrete state set. The function $\Gamma$ is to be defined by partitioning the range of possible values of $x$ into a set of adjacent intervals and assigning to each of them a state value $s \in \mathcal{S}$.

Moreover, we have to define how to combine discrete CA states with the real values from the gap model. As can be seen from Fig. 2.5, the converted state vector of the CA $\bar{x}_{1,t+1}$ meets with the real vegetation state $\bar{x}_{2,t+1}$ in the block of a linear combination $\Sigma$. Let us define the block $\Sigma$ as follows:

$$\bar{x}_{t+1} = \bar{c}_1 \bar{x}_{1,t+1} + \bar{c}_2 \bar{x}_{2,t+1} \tag{2.33}$$

where $\bar{c}_1$ and $\bar{c}_2$ are constant coefficients. In this case, it is hard to generate values of $\bar{x}_{1,t+1}$ which do not disturb vegetation dynamics (the CA should not introduce errors).

As we know from the previous subsection, the CA is responsible for death and birth rules. Thus we must get a modification of the vegetation state only in this case. The proposed idea is to generate a positive impulse in the birth case, a negative value if a death takes place and a neutral value for normal vegetation:

$$\bar{x}_{1,t+1} = \begin{cases} +1 & \text{when a birth of a new vegetation is selected by the CA rules,} \\ -1 & \text{when a death is selected by the CA rules,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

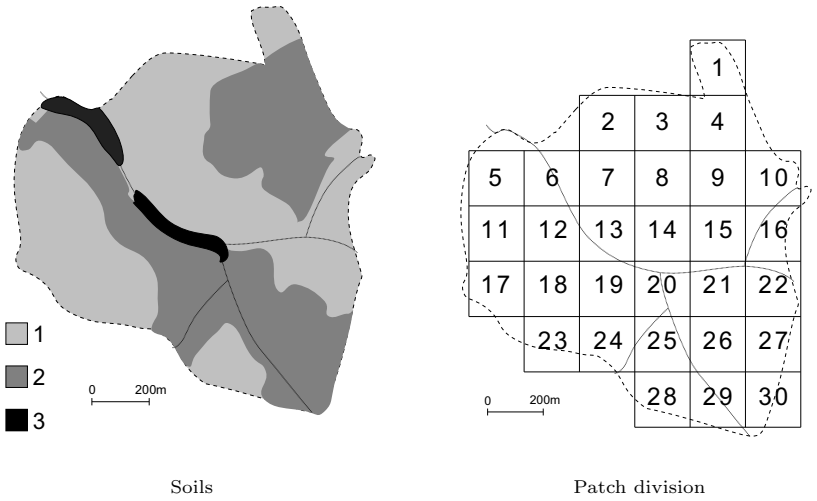



**Fig. 2.6.** Scheme of the Ratanica catchment.

For a proper implementation, we can tune death or birth intensities by appropriately changing the value of $\bar{c}_1$ and conversion function $\Theta$ must not change the values of $\bar{x}$, i.e.

$$\Theta(\bar{x}_t) = \bar{x}_t.$$

### 2.3.2. Adding the fire effect

For many ecologists it is extremely important to analyse the evolution of the landscape after an outbreak of fire. The simplest way to introduce this feature is to destroy (manually or by a special map) the biomass on selected patches. The level of destruction should be proportional to the fire intensity, e.g. if we model a poor element, we can destroy $\simeq 10\%$ of the total biomass on the patch (remove $\simeq 10\%$ of each species in the patch), in the strong fire case we can destroy $\simeq 95\%$ of the total biomass.

Many researchers may ask why a fire dynamics model is not proposed in this place. The answer is very simple: the proposed model operates with the constant time step equal to one year. As a result, it is impossible to introduce a fire dynamics model scaled on the same time range (it is not sensible to consider a fire lasting for one year or longer).

### 2.3.3. Simulation results

For implementation and extensive tests of the presented model, specialised software has been developed by the author in C++. The program LUCAS (**LU**cifer **C**ellular **A**utomata **S**imulator) produces the prediction of what a landscape would be in the future, taking into account fires or not [92]. It may evolve

**Tab. 2.8.** Growth parameters for the Ratanica catchment species.

| Species | Pinus sylvestris | Fagus sylvatica | Larix decidua |
|---|---|---|---|
| $B_{\max}$ | 13 | 13 | 13 |
| $B_{a\,\max}$ | 0.004 | 0.004 | 0.004 |
| $A_{\max}$ | 150 | 125 | 200 |
| $W_s$ | 0.2 | 0.2 | 0.1 |
| $D_5$ | 3200 | 3500 | 3500 |
| $D_{5\,\min,i}$ | 2700 | 3000 | 2000 |
| $D_{5\,\mathrm{opt},i}$ | 4000 | 4000 | 4500 |
| $D_{\max}$ | 80 | 90 | 50 |
| $H_{\max}$ | 1500 | 1500 | 1100 |

with, or without external forces exciting the landscape such as controls, and can be passive (as uncolonised zones) or active (fire, human activities).

The proposed approach has been validated on a real ecosystem. For that purpose, the ecosystem of the Ratanica catchment was adopted which had been thoroughly studied in the 1990s, cf. [32]. The Ratanica is a river whose catchment is located in Southern Poland (49°51'N, 20°02'E), 40 km south of the city of Cracow and in the vicinity of a big water reservoir providing the city with drinking water. The entire catchment covers an area of 241.5 ha. The vegetation of the catchment is typical of the Carpatian Foothills. Its upper part is covered with forests, while the lower one with meadowsand fields. The forests are of anthropogenic character. Coniferous species (*pinus sylvestris* and *larix decidua*) are dominant. The main deciduous tree is beech (*Fagus sylvatica*).

Beech-pine and pine-beech woods dominate in the catchment. The age of trees ranges from 40 to 80 years, the oldest classes being rarely found. The results of dendrometric studies in the Ratanica catchment indicate a large potential productivity. The most dynamic tree species in this region is beech. Its expansion and regression of Scots pine and larch might have been caused by changes in the forest management and increased atmospheric input of nitrogen.

In order to compare the behaviours of our model with nature, initial and final values for the volume occupied by the trees (i.e. the biomass volume) were taken (the results for the middle values are not included here due to limited space). For this comparison, the following information was used: soil configuration (only for the initial condition), species composition of tree stands and large timber volume measurements for the lattice composed of square patches with sizes $200 \times 200$ m (cf. Fig. 2.6). This makes it possible to compare only patches (not individual trees; this would be ideal, but very hard to interpret owing to the simplification and generalisation of the gap model).

Simulations using the proposed hybrid model with the lattice of 30 patches (they correspond to patches presented in Fig. 2.6) of $10 \times 10$ cells were stopped after the periods of one and two years. Simulation results in the form of the error

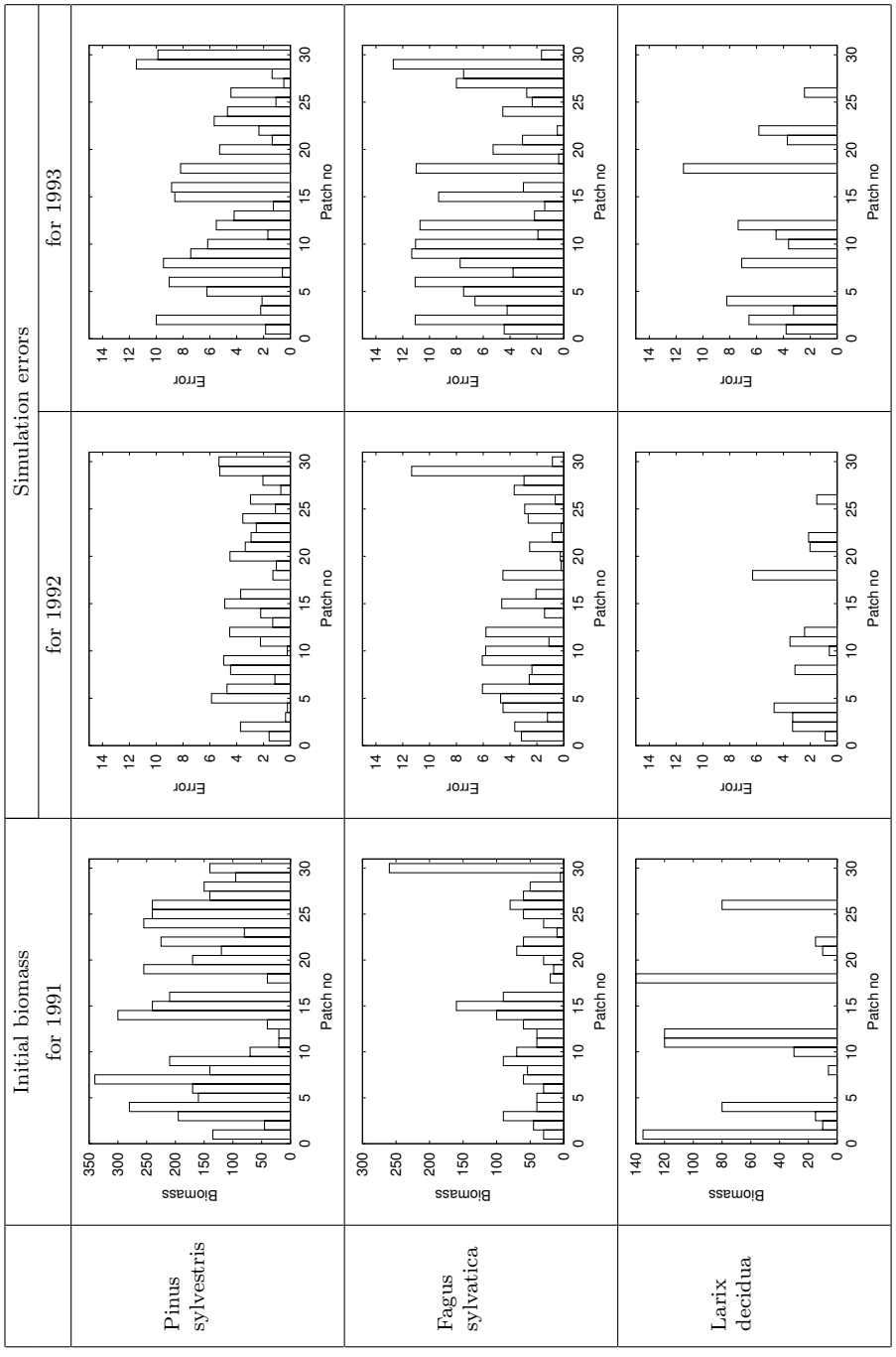



**Fig. 2.7.** Simulation results for the ecosystem of the Ratanica catchment (the errors are relative and given in per cents).

resulting from comparison with the corresponding measured values are given in Fig. 2.7. The initial configuration of the lattice was built on the basis of the decomposition map for the tree age and its biomasses. The values of biomasses in each patch are evaluated as the sums of component cells. The error values for the first and second year of simulations (max. 4.8% for 1992 and 8% for 1993) allow this model to be used for prediction of the growth tendency and general biomass states. Some errors may result from incorrectly tuned parameters of the gap model: the model relies on Polish species which are not so well documented in the gap context. In order to solve this problem, the unknown parameters of the gap model were estimated using the adaptive random search [86] (the algorithm and estimation process are described in detail in Chapter 4): the measurements at hand were used to run the hybrid model and to evaluate the respective descrepancy measure between observations and model outputs. The problem is not trivial because the model behaviour is discrete (due to the use of the CA) and stochastic (probabilistic rules of birth and death).

Unfortunately, our model does not take into account human and nature intervention (fire, cutting trees, pollution, etc.) which may additionally involve difficulties in obtaining good simulation quality.

## 2.4. Multi-cell specimen approach

### 2.4.1. Problem statement

As we know from Section 2.1.1, most models operate on modelling the forest using individual trees. Moreover, most gap models [70, 80, 47, 9] evaluate the growth equation for each tree. All those models must solve spatial problems like reproduction by seed transport, crowding stress, death and competition for solar energy. The foregoing model uses some specific formulae: (2.5) for defining the crowding stress and (2.3.1.2) for decisions about the crowding stress. For correction of these simplifications, we should define a more detailed model which would assure a better description of the phenomena. Using CA's, we could solve to some extent the specified problems if we found a method to describe each individual tree space by a cell on the lattice. The problem is not trivial, as in the ecosystem the behaviour of the growth process depends on many parameters and some of observed phenomena (information about the global body size and shape) can be modelled only for a huge neighbourhood (which may include a maximal body size). Additionally, tree behaviours characterise a variable growth intensity (greater in the beginning of the life and much slower at the end). The growth process cannot be modelled only by the spread of tree biomass, because tree crowns move up with time. Thus we have to tune the bottom space accordingly and not to disturb the growth process.

In this section the CA approach to modelling forests by each tree will be described. To attain this objective, each cell describes a smallest part of space (in the previous model one or more trees, e.g. an area of 20m×20m and now a cube of 10cm×10cm×10cm). Thus each tree is described by many cells (we obtain a

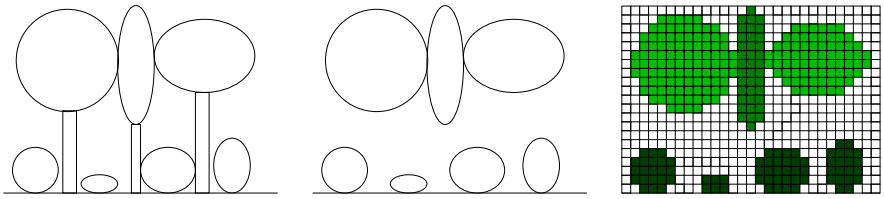


**Fig. 2.8.** Space discretisation process.

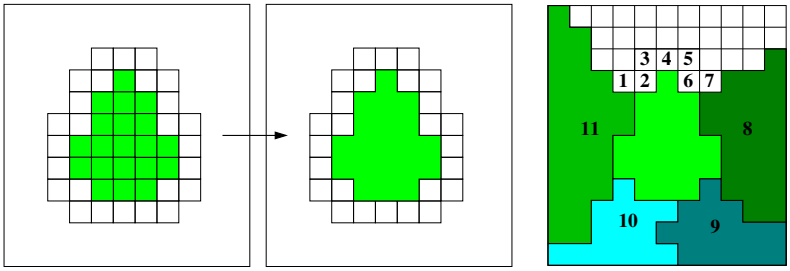



**Fig. 2.9.** Tree representation on the CA lattice, and the definition of the simplified neighbourhood in the 2D view.

tree-space discretisation) [41].

### 2.4.2. Proposed approach

Because the research on the tree growth dynamics and forest dispersion is based on the analysis of interactions between trees and their neighbourhoods, CA's seem especially suited as a modelling tool for this process. The simplest method to use CA's in ecological models is to represent each individual (tree) by one cell and to update vegetation parameters on the basis of gap equations. In this case, such a CA consists of a lattice of cells characterised by their state values representing a given parameter of vegetation. The CA evaluates the state value for each cell and passes this information to a gap model. Additionally, it evaluates the coefficients responsible for additional discrete phenomena.

The delineated approach was already employed [33, 49] and proved to improve simple and precise evaluation of space dynamics (seed transport, birth and death rules, etc.), but there still remains the problem that in a gap model we must take account of the crowding and solar effects depending on the current state of the neighbouring gaps. As a solution to this problem, we propose here to refine the description of one tree by enlarging the set of cells which represent this tree. In fact, this idea is implemented in nature, as all plants are built of many cells (cf. Fig. 2.8 and 2.9). Of course, the proposed model does not attempt to build trees from billions of cells like in nature. This methodology only provides a sim-

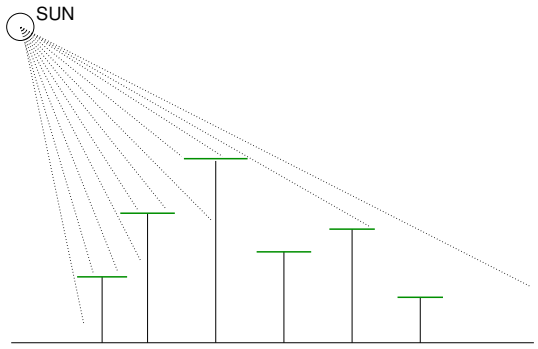



**Fig. 2.10.** A typical tree representation for gap models.

ple description of photosynthesis intensity and space stress. In the classical gap models, in order to determine the size of the green part which has direct contact with solar rays, we use a quite general formula which calculates this value from the diameter of the tree and the diameter of neighbourhood. Each tree is represented by a flat green part located perpendicularly to the trunk (Fig. 2.10). In the model proposed, we allow for a competition between all trees aiming at winning as wide space as possible, and we can easily detect how many cells are exposed to the sun light (top cells enumerated as 1–7).

The recommended CA consists of a regular three-dimensional lattice, in which each cell is characterised by a particular state taken in a discrete finite set of admissible states. The dynamics of each cell is defined by a set of transition rules which specify the future state of this cell as a function of its previous state and the states of neighbouring cells. The proposed neighbourhood definition is of von Neumann's type.

In the presented model, the state $\mathcal{S}$ is characterised by the vector, with the following components:

- **Species type** $S_t$**:** it has possible values $\mathsf{Nothing}$, $\mathsf{Species}_1$, $\mathsf{SpeciesSeed}_1$, ..., $\mathsf{Species}_n$, $\mathsf{SpeciesSeed}_n$, where $n$ stands for the number of species;
- **Specimen index** $S_i$**:** an integer label which identifies which tree a given cell belongs to;
- **Tree diameter** $D$ in the form of a float value to be calculated by the gap model;
- **Biomass** $B$**:** the number of cells constituting the tree.
- **Age** $A$**:** the age of the tree expressed in iterations.

In the context of the proposed approach, the most important component is the transition function $f$. Since it is very hard to describe growth phenomena by simple rules, a straightforward manner is to use known and already validated gap

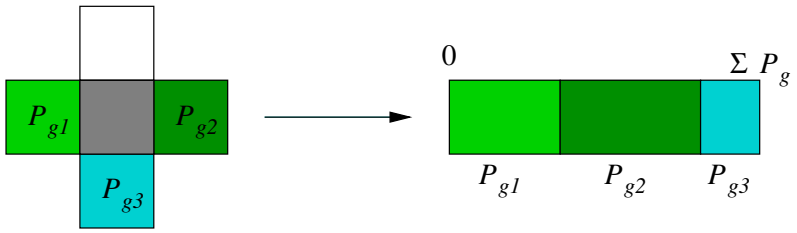



**Fig. 2.11.** Manner of choosing which tree grows to an empty cell.

models. The gap model (the same as that used for the previously proposed model, i.e. (2.18) for trees and (2.19) for shrubs) is thus used to estimate the growth tendency in the next iteration. This tendency can be used by empty cells which are in the neighbourhood of the tree for computing the probability of growth. The rules for the birth law can be defined by dispersal rules for seeds. The death rule simply evaluates the death probability equation [70], which depends on the current conditions and life length.

Summarising, the following situations may take place:

- *A cell is empty.* Then the neighbourhood must be analysed (four cells in the case of the von Neumann definition). If one of these neighbours belongs to an individual, then the probability of growth onto this cell is calculated according to the formula

$$P_{g,i} = \frac{f_B(D_i) - B_i}{N_{f,i}}, \tag{2.35}$$

  where $N_{f,i}$ is the number of cells which are in the neighbourhood of the individual $i$ and are not occupied by other individuals. In case a cell is located between two or more specimen (more than one neighbour belong to individuals), the probability of growth is calculated for each of those neighbours. Then the probabilities are written into the sequence (cf. Fig. 2.11) and a value from 0 to $\sum P_{g,i}$ is drawn at random. The membership of the drawn value into the range representing one of individuals decides about the winner (i.e. the individual chosen to grow in the site). For the case when $\sum P_{g_i} < 1$, and the upper part of neighbours represents a seed, we draw a real value from the range from 0 to 1 and values greater than $\sum P_{g,i}$ mean that this seed is captured.

- *A cell transports a seed and its bottom neighbour represents the ground.* Then a new individual must be born. This is achieved by generating a unique label $S_i$ to mark it and setting the corresponding type flag $S_t$.

- *A cell represents a tree's body.*

  – If a cell is on the bottom of the body (bottom neighbours do not be-

long to the same individual $s_{S_i}(c) \neq s_{S_i}(c_3)$), we compute the death probability,

$$P_d = (0.01)^{1/A} + \frac{|f_B(D) - B|}{B_{\max}}, \tag{2.36}$$

where $f_B(D)$ is a function which estimates the biomass value from the diameter $D$ and $B_{\max}$ is a maximal biomass value. Both the parameters depend on the considered species.

– Otherwise, the cell does not change its state value (except updating $D$ based on the gap model).

### 2.4.3. Simulation results

Since the proposed model operates on a precise description of the trees, it is a good idea to validate it on a small part of an ecosystem. Similarly to the simulation of the previous model, the Ratanica catchment was used as the real ecosystem.

For initial conditions the shapes of trees are generated from species composition of tree stands and large timber volume measurements for the lattice composed of 30 patches with the sizes $200 \times 200$ m (cf. Fig. 2.6). This makes it possible to compare only patches (not individual trees, as this would be ideal but very hard to interpret owing to simplification and generalisation of the gap model).

Simulation using our CA with the lattice of $1200 \times 1400 \times 200$ cells was stopped after the periods of one and two years. Simulation results together with the corresponding measured values are given in Fig. 2.12. The first column represents the initial biomasses of each species in each patch for the year 1991. The second and third columns present the biomasses obtained from simulation reduced by the real measurements, and thus they presents simulation errors. Similarly to the previous model, we can see small values of simulation errors. Unfortunately, owing to limited measurement data, we cannot carry out a more detailed analysis of the model behaviour. It can be concluded that the proposed model is fit to predict the growth tendency and general biomass states.

## 2.5. Conclusion

Simulation results confirm that the proposed models constitute interesting alternatives to single gap models. They allow us to describe many phenomena more intuitively and still obtain satisfactory results. These approaches make it possible to prepare universal tools which can be used by foresters and biologists who are not, in most cases, interested in a PDE description of all phenomena—a rule description is simpler to understand and develop. The results respect the same mechanics as that used by Brufau in [9], but the complexity of both the presented models is lower.

The first model offers a simplified model construction and still good modelling quality. The results obtained are comparable with the work of Tongeren [80]

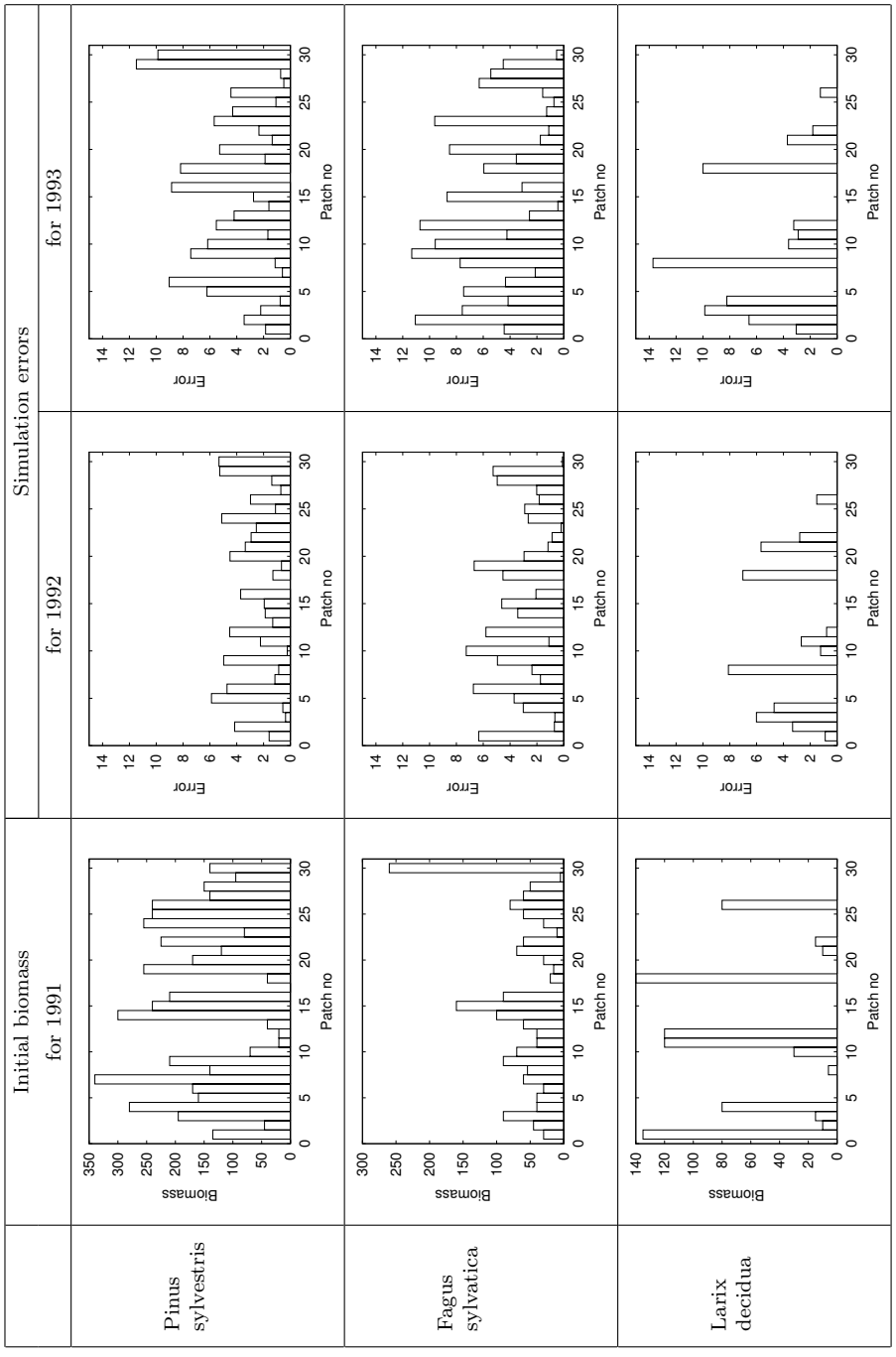



**Fig. 2.12.** Simulation results for the multi-cell specimen approach (the errors are relative and given in per cents).

for shrubs, except that the proposed model is more universal: it operates on shrubs and bigger trees and more than one tree can live on the same territory at the same time (cf. only one in [80]). Application of the modified gap model permits us to significantly reduce the number of differential equations (only one per species for each cell). Thus we obtain a considerable improvement in the performance: on a low-cost PC, the developed LUCAS simulator generates the next iteration (which often means one year) in two seconds for the terrain represented by 200×150 cells (each gap equation is interpreted from any form given by the user).

A model with detailed tree description offers a possibility of observation of only one or two simulated trees. This can give forest experts more information about the simulation quality than a table of figures resulting from integration of gap equations—it is a continuation of single-tree models presented at the beginning of this chapter. It better respects the influence of light terms (they are evaluated in more detail) and local disturbances of spatial configurations (the crowding stress).

A direction for future research is to improve the proposed model in order for it to better represent the shape of the trees and to modify parameters for better adaptation to the studied phenomena and validate them on bigger ecosystems for longer periods (for more than 10 years).

# Chapter 3

# Modelling spread phenomena

## 3.1. Introduction

So far, much of the research on CA's has been of an experimental type. In recent years, however, attempts have been made to insert it into general modelling theory [44, 45]. Moreover, it is possible to slightly modify the entire analysis and control theory of systems evolving in space and time so as to include CA models. As a consequence, the CA method could be considered as an alternative approach to modelling the so-called spreadable systems. For that purpose, in this chapter appropriate transition rules will be defined and characterised for spreadable CA's. Then the concepts of passive control in CA's will be introduced and some connections with spreadability will be indicated.

The problem of ecological modelling has a potential to take full advantage of the power of CA's. Spreadability is particularly motivated by vegetation dynamics problems, but its study involves various other fields of applications [61]. A wide variety of natural processes can be described by the spreading or kinetic growth (KG) model, see e.g. the fluid flow trough porous media, beverage accidentally spilled on the table, reproduction of yeast, fire on the flammable material, epidemic spread, etc. Like in any of model designing tasks, the main problem is a proper selection of a model. They are two ways to do that: to create model equations on the basis of detailed knowledge of the studied problem or to approximate its behaviour by a general model. In both the cases we obtain a model described by partial differential equations which may be complex and hard for analysis. Additionally, this kind of models may be hard to apply (known methods are rather numerically complex, and it goes without saying that they are time-consuming). The CA approach allow us to describe the problem by simple and comprehensible rules which do not necessarily cause a substantial loss of accuracy. Additionally, a cellular architecture considerably simplifies the visualisation task. In this case the discrete character of the CA state do not yield a discretisation error because the spreadable feature as well as space are not continuous.

There are different choices to construct CA's which concern the lattice geometry, neighbourhood type, boundary and initial conditions, state set and transition rules. As is shown below, a suitable choice of these parameters may lead to the

spreadability in CA models.

We recall that spreadability is a process in which an object or a feature extends itself over an increasingly larger area by incorporating neighbouring regions to itself. Let $\mathcal{A} = (\mathcal{L}, \mathcal{S}, N, f)$ be a CA (cf. Def. 1.1, p. 6) and $\mathcal{P}$ be a property to be spread or absorbed (a negative spread), defined by

$$\mathcal{P}s_t(c) \Leftrightarrow s_t(c) = s_D, \tag{3.1}$$

where $s_D \in \mathcal{S}$ corresponds to a desired state and is assumed to be reachable from initial time $t_0$. Consider now the family of subsets

$$\omega_t = \{c \in \mathcal{L} \mid \mathcal{P}s_t(c)\} = \{c \in \mathcal{L} \mid s_t(c) = s_D\}. \tag{3.2}$$

**Definition 3.1** A cellular automaton $\mathcal{A}$ is said to be $\mathcal{P}$-*spreadable* from $\omega_{t_0}$ if

$$\omega_t \subset \omega_{t+1}, \quad \forall t \geq t_0. \tag{3.3}$$

It is necessary to emphasise that $\mathcal{P}$ is a property (most often, it consists in attaining a given value of the state or part of the state vector), not a dimension. A good example is a group of cluster-spreadable models, where the spreading property $\mathcal{P}$ is represented by the *cluster membership*. In other words, we observe expansion of cells belonging to the cluster ($s_t(c) = s_D$, where $s_D$ means the cluster membership).

Good examples of cluster-spreadable CA's are implementations of an Eden model and a single percolation cluster model proposed here. The former one was introduced by biologists and, due to its simplicity, is excellent to illustrate spreadability phenomena and CA usefulness in spread modelling. The latter exemplifies how simply we can extend the previous model to make it useful in modelling complex phenomena by adding new features.

### 3.1.1. Eden model

The *Eden model* [27] is an original kinetic growth model which was introduced by a biologist (for whom it is named) to represent the growth of tumours. The Eden model operates on a two-dimensional square lattice system. It starts with a cluster of cells consisting of a spread origin and its neighbourhood. The state set is composed of three values:

- an empty cell,
- an occupied cell (the cell being a component of the cluster),
- a member of the cluster border (the empty cell in the direct neighbourhood of the cluster).

At any step we randomly choose a cell from the border and join it to the cluster. Then we need to define a new set of the cluster border (by adding the empty cells from the neighbourhood of the joined cell, which do not belong currently to the border).

#### 3.1.1.1. Eden algorithm

The original algorithm implementing the Eden model employs global rules with global variables. At the beginning, two lists of coordinates are created. The first list is composed of the coordinates of cluster cells

$$V_C = \big((x_1, y_1), \ldots, (x_{n_C}, y_{n_C})\big) \tag{3.4}$$

and the other of the coordinates of the border cells

$$V_B = \big((x_1, y_1), \ldots, (x_{n_B}, y_{n_B})\big), \tag{3.5}$$

where $n_C$ is the number of cells forming the cluster and $n_B$ is the number of cells in the direct neighbourhood of the cluster. The neighbourhood in original form is defined in the von Neumann style (1.5), but the Moore (1.7) or modified von Neumann neighbourhoods (1.6) can be also employed. Then we randomly pick a cell $b$ from $V_B$ for joining it to the cluster (the cell is thus moved from $V_B$ to $V_C$). Next, the nearest neighbour sites of $b$ that are not in $V_B$ yet are added to $V_B$. Another site is then randomly selected from $V_B$, and so on. This process continues until the cluster list reaches some maximum size $n_{C\,\max}$.

The scheme of this algorithm is shown in Fig. 3.1. Figure 3.2 presents an exemplary sequence of five steps of the Eden algorithm.

#### 3.1.1.2. CA implementation of the Eden algorithm

The Eden algorithm is very simple, which suggests that it must be also simple to apply it in the CA framework. It is not true, however, as in a CA each cell updates its state based on the neighbourhood configuration and not on global parameters like lists of cluster or border cells. To introduce the corresponding CA model, we can define the cluster border as all cells outside the cluster with one of their neighbours occupied by the cluster. Each border cell becomes a cluster cell with a fixed probability $p_C$.

On the analogy of the original Eden algorithm, the appropriate CA implementation should add new cells into the cluster only from among its actual neighbours. An essential difference is the state set which consist of only two values:

$$\mathcal{S} = \{\text{EMPTY}, \text{CLUSTER}\} \tag{3.6}$$

The third value BORDER is not necessary, because the detection of whether or not an EMPTY cell is a border cell can be performed using a simple rule like the following one:

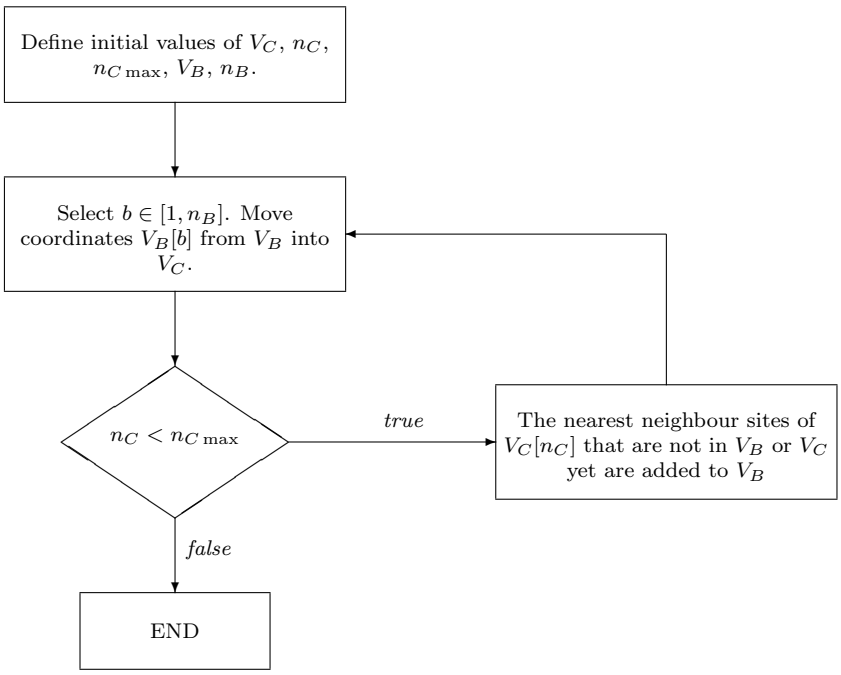



**Fig. 3.1.** Eden algorithm.

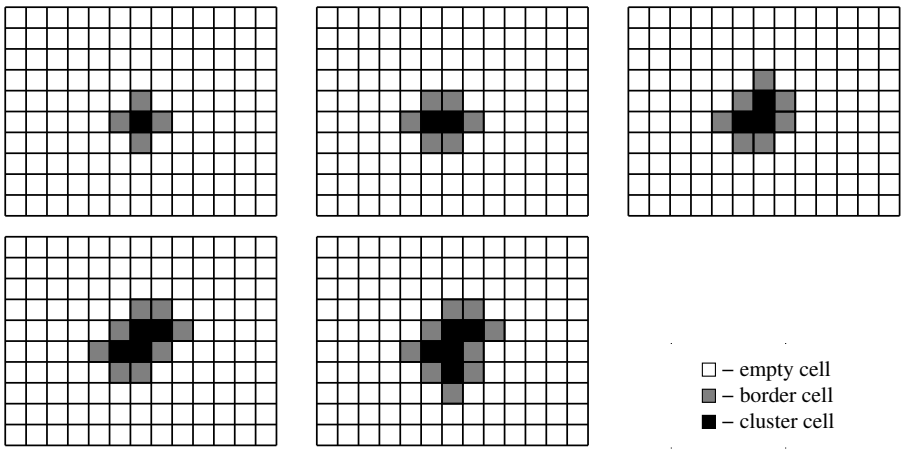



**Fig. 3.2.** An exemplary sequence configurations of the Eden model.

```
IsBorder:=false;
for i:=1 to n
  if (s(N(c)[i]) = CLUSTER)
  begin
    IsBorder:=true;
    break;
  end;
```

where the function $s(c)$ returns the state of a cell $c$, $N(c)$ stands for the neighbourhood and $n$ is the neighbourhood size (cf. Def. 1.1). The main difference lies in the cluster growth method. In the original model, at each time step we draw one of the border cells and move it into the cluster. Because the CA dynamics is based on local rules, we cannot operate on a global list of border cells and so we cannot simply pick one of its elements. A solution is to define some fixed probability $p_C$ of joining a border cell to the cluster. In this case we realize the main goal (the cluster spread), but at a given time step it may happen that no cells or more than one cell will be joined (this situation is impossible in the original model). For the correctness of this implementation, the value of $p_C$ is crucial: a small value makes the growth difficult, while a large one may produce an avalanche spread. To complete the description, we also need to define a neighbourhood $N(c)$: for most cases the von Neumann one is sufficient (see Section 1.3).

**Example 3.1**
Exemplary simulation results of a CA for a lattice of 130×100 cells, von Neumann neighbourhood and probability $p_C = 0.1$ are shown Fig. 3.3. Because in the CA case the growth intensity strongly depends on $p_C$, we can observe the exponential increase of the cluster set as time elapses. This behaviour perfectly fits to a real situation corresponding to the intensity of growth depending on the size of tumours. In nature each border cell of tumour strives to reproduce and does not care about the reproduction of other cells. There is no central mechanics which decides about the growth intensity. Thus in real situations the growth process is never linear as is suggested in the original Eden algorithm. Figure 3.4 presents the comparison of the growth intensity for three values of $p_C = \{0.01, 0.1, 0.5\}$.

The scale of the lattice presented here is enough to show a way of cluster growth. Its increase will change only the scale, but the character of results will be still retained.

The CA approach to the Eden model considerably simplifies the problem (we need neither to update the border at each time moment, nor to save lists of cluster and border cells). By joining several cells instead of one we do not disturb simulation results, as they are like the original Eden model played more quickly.

The presented results perfectly reflect spreadable phenomena. In our case the property to be spread is the membership to the cluster. Specific properties of CA's allow us to easily build a spreadable model and to observe its evolution. Additionally, the problem description by local rules makes this model more intuitive.

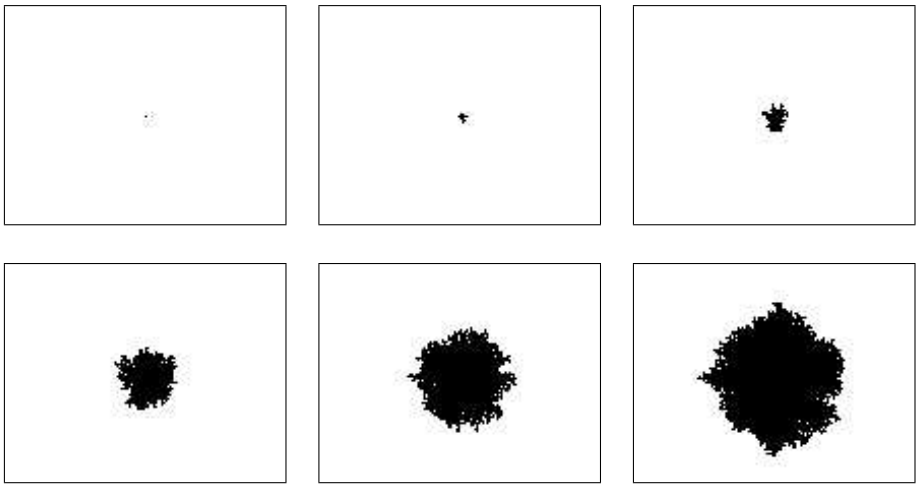


**Fig. 3.3.** Evolution of the CA implementing the Eden model.

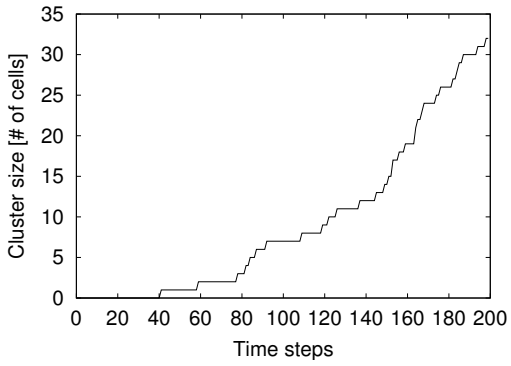
### 3.1.2. Single percolation cluster model

The single (or random) percolation cluster model constitutes a derivative of the Eden model and can be used to describe the epidemic spread of a disease [28]. Similarly to its ancestor, the discrete lattice is split into four sets:

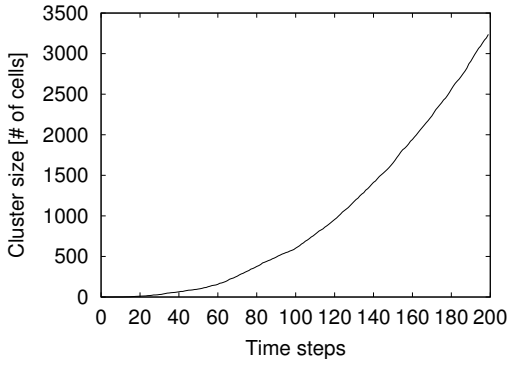
- the set of empty cells,
- the set of cells representing the spreading phenomenon (the cluster),
- the set of cells representing the border of the phenomenon, and
- the set of cells resistant to the spreading phenomenon.

In contrast to the Eden model, not all the cells chosen from the border set are included into the cluster. This process is controlled by the probability $p_j$ of joining a selected cell from the border to the cluster. In each time step we randomly choose a cell $b$ from the border set $V_B$ to possibly add it into the cluster. Then we draw a number from the interval $p \in (0, 1)$ and we compare it with probability $p_j$. If its value is less than $p_j$, we add $b$ to the cluster, otherwise it becomes *resistant*. No matter whether or not a cell is added to the cluster, the new border is determined (if a cell is *resistant* to the cluster spread, it will not be considered for joining to the cluster any more).

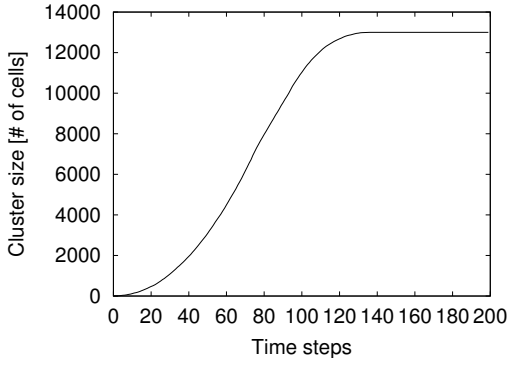
Because the border cells surround both the cluster cells and resistant ones, when we draw a cell $b$ from $V_B$, it can be a neighbour of a resistant cell. Thus we can say that in this model the cluster growth appears not only near the cluster cells. Unaffected cells are also a potential source for the cluster growth. It is much like in nature: very often unaffected individuals are a carrier of a disease. Thus we can say that they are also a spread medium like infected ones.




(a) $P_c = 0.01$




(b) $P_c = 0.1$




(c) $P_c = 0.5$

**Fig. 3.4.** The cluster growth for the CA implementation of the Eden model.

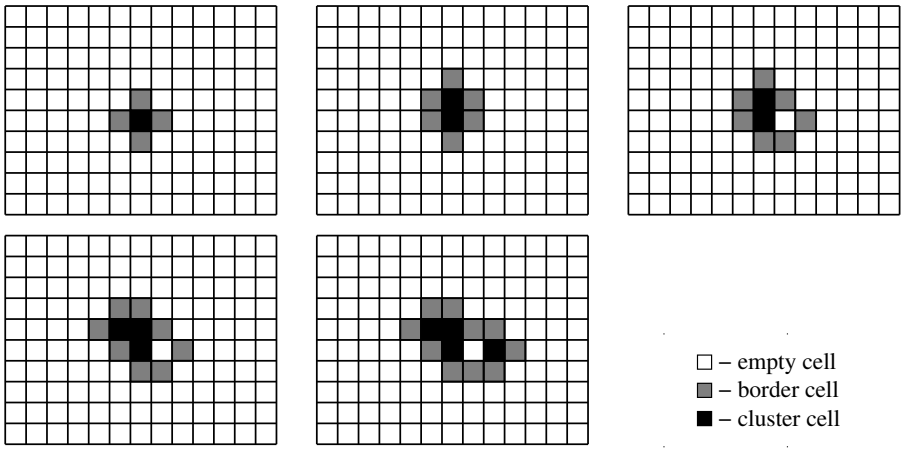



**Fig. 3.5.** Consecutive configurations of a single percolation cluster model.

An exemplary evolution sequence of the discussed model is presented in Fig. 3.5.

#### 3.1.2.1. CA implementation of the single percolation cluster model

In the CA implementation of the Eden model we have used a two-valued state set. Here we need to introduce a new state RESISTANT which corresponds to the state of an *unaffected* cell, i.e. we have

$$\mathcal{S} = \{\text{EMPTY}, \text{CLUSTER}, \text{RESISTANT}\}. \tag{3.7}$$

This state augmentation is necessary because, as opposed to border cells, we cannot detect the resistant cells arithmetically. Furthermore, this new state helps us to simply detect border cells. In the CA implementation of the Eden algorithm we mark an empty cell as the border if one or more of its neighbours are members of the cluster. As we see in Fig. 3.5, in this model sometimes a border cell has no neighbours belonging to the cluster (Configurations 3 and 4). The introduction of a new state RESISTANT which means a free cell resistant to the cluster expansion, solves this problem. Now we mark a cell as a border one if one or more cells in its neighbourhood are members of the cluster or a resistant one. The corresponding algorithm can be written as follows:

```
IsBorder:=false;
for i:=1 to n
  if ((s(N(c)[i])=CLUSTER) or (s(N(c)[i])=RESISTANT))
  begin
    IsBorder:=true;
    break;
```

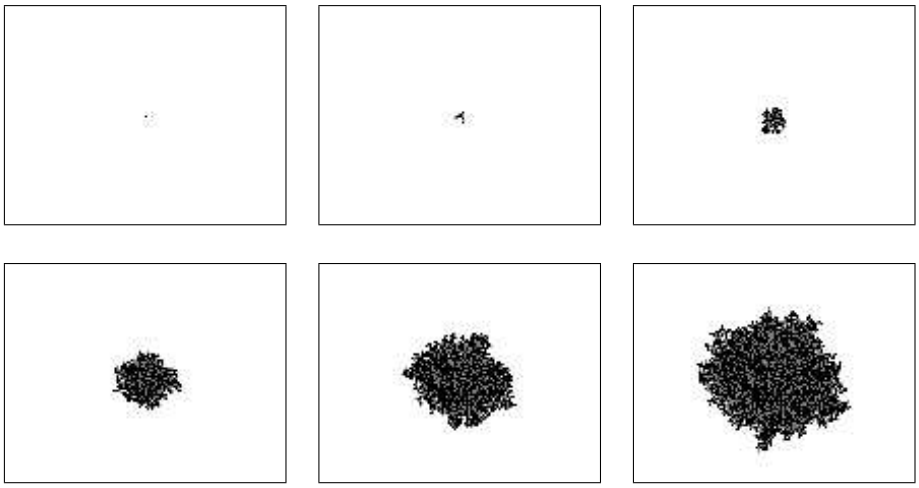


**Fig. 3.6.** Simulation results for the CA implementation of the single percolation cluster model.

**end**;

To complete our CA model description, we should add that at every iteration for cells with probability of joining in the cluster greater than $p_C$, we are to draw a number $p \in (0, 1)$ and compare it with $p_j$. The result of this comparison will permit us to qualify the cell for either the cluster, or the border.

**Example 3.2**
Some simulation results of a CA with a lattice of 130×100 cells, von Neumann neighbourhood, $p_C = 0.1$ and $p_j = 0.1$ are demonstrated in Fig. 3.6. Empty cells are presented as white squares, the grey ones correspond to resistant cells and black squares represent the cluster area.

At first glance the results are very close to those for the ancestor model and yet some cells marked with the grey colour mean empty and resistant cells. Its amount is directly proportional to $p_j$. In the model, the real probability of the cluster growth $p'_C$ is of the form

$$p'_C = p_C p_j. \tag{3.8}$$

The corresponding graph of the cluster growth graph is presented in Fig. 3.7. On this graph we clearly see the difference from the Eden model. For the simple percolation cluster model the growth process is slower (not all cells join in the spread phenomenon) because of the smaller value of $p'_C$.

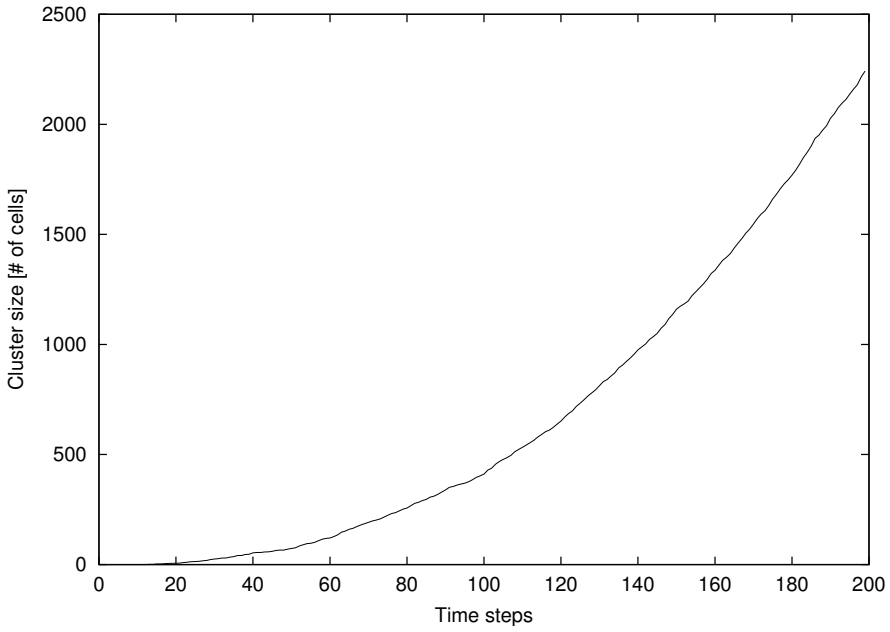



**Fig. 3.7.** The cluster growth process for the CA implementation of the single percolation model.

## 3.2. Spray Control

As developed in [44, 78, 45], it is possible to make time-continuous systems spreadable by means of appropriate controls, called spray controls. In the case of CA modelling, it is possible to extend this idea. To this end, we have first to define what a control can be. Let $\mathcal{A} = (\mathcal{L}, \mathcal{S}, N, f)$ be a CA and $\mathcal{L}_1$ be a subset of the lattice $\mathcal{L}$.

**Definition 3.2** A *control* $u : \mathcal{L}_1 \times T \to \mathcal{S}$ is a function which is defined on $\mathcal{L}_1 \subset \mathcal{L}$ ($\mathcal{L}_1$ can be reduced to a one-cell subset $\{c\}$ and varying in both time and space), where $T$ signifies the discrete time horizon. $\mathcal{A}$ becomes then a non-autonomous cellular automaton.

Below, two examples of the control influencing a CA are briefly discussed.

### 3.2.1. Invasion percolation model

The invasion percolation model is another derivative of the Eden model. It also operates on a two-dimensional rectangular lattice, with coordinates lists for cluster and border sites. In this model, every cell in the border set has an associated random number. The cluster spreads by incorporating a border site with the lowest associated number. If there are more than one cell with the lowest

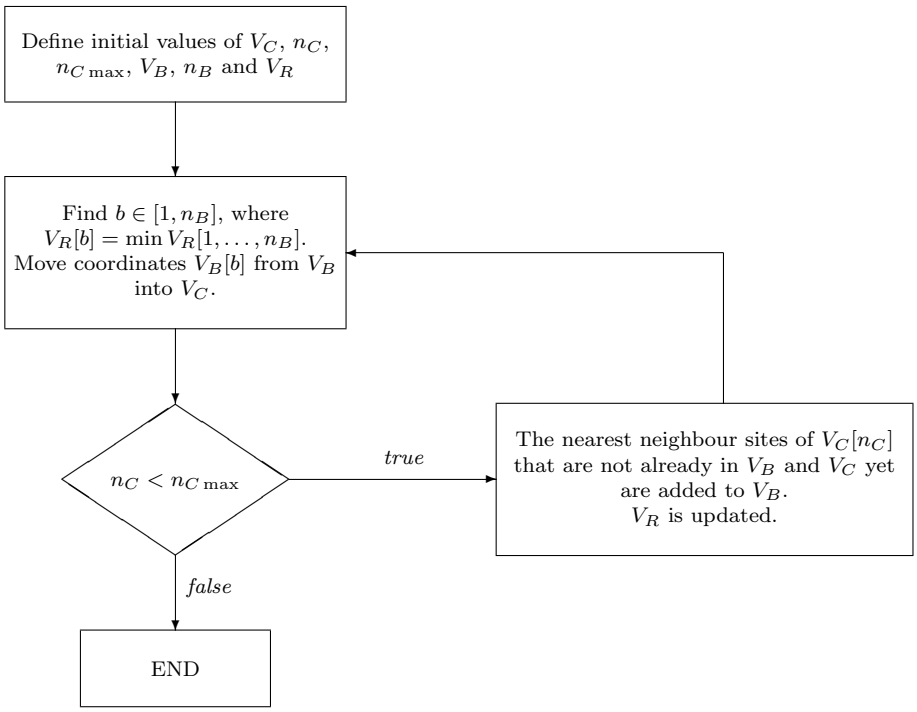



**Fig. 3.8.** Algorithm for the invasion percolation model.

value, a winner is randomly selected. Thus this model describes the process of spreading that "follows the path of the least resistance". It can be used to describe the fluid flow through porous media, which occurs e.g. in tertiary oil recovery.

#### 3.2.1.1. Algorithm

In much the same way as in the Eden model, at the beginning we create lists of cluster sites $V_C$ and the corresponding border sites $V_B$ (cf. (3.4) and (3.5)). Additionally, we have to define the list of resistance numbers $V_R$ corresponding to elements of $V_B$. Then we select a site from the $V_B$ list with the lowest resistance associated with the corresponding $V_R$ site. Next, we append this site to the cluster and update the border list $V_B$. Simultaneously, we have to randomly generate new resistance values of $V_R$ corresponding to the new sites added to $V_B$. If the cluster size is less than a maximum possible threshold $N_{C\,\max}$, we select a new site from $V_B$, and the process is repeated.

The block scheme of the algorithm is given in Fig. 3.8. To make the understanding of the invasion percolation algorithm easier, in Fig. 3.9 an exemplary sequence of five consecutive steps is presented.

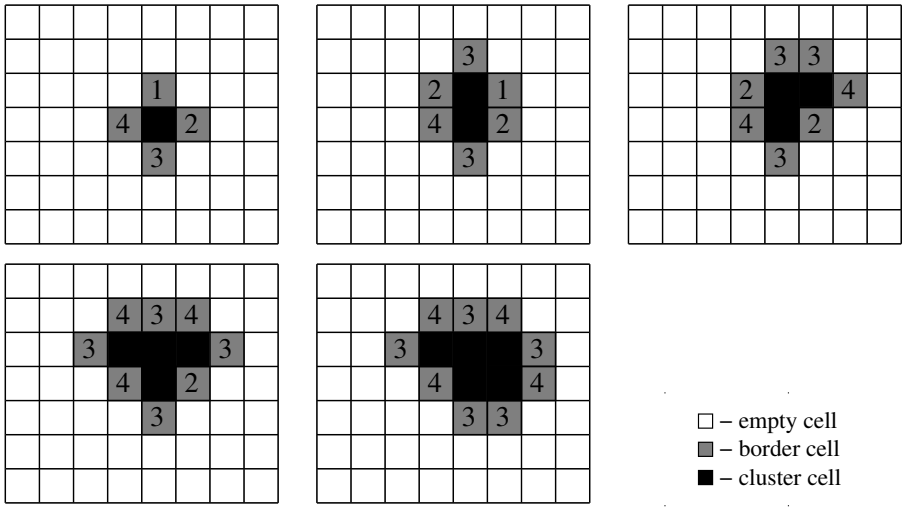



**Fig. 3.9.** An exemplary evolution of the invasion percolation model.

### 3.2.1.2. CA implementation of the invasion percolation model

As regards the CA implementation, the spread process in the invasion percolation model is driven by the local resistance. In the application of the model presented in [28], the resistance was drawn for new cells which were added to the border ($V_B$) and its value was stored in a vector of resistances $V_R$. Because $V_B$ is not needed (we can detect the border arithmetically, cf. Eden and single percolation cluster models), the state should include information about the type of the cell (an empty or a cluster component) and the local resistance for the empty case. We have two possibilities:

- Use one state variable, where the state set $\mathcal{S}$ is composed of the information about the cluster membership and the group of sets representing different resistances for empty cells:

$$\mathcal{S} = \{\text{CLUSTER}, \text{EMPTY}_{R_{\min}}, \ldots, \text{EMPTY}_{R_{\max}}\}, \tag{3.9}$$

- Extend the scalar state to the state vector:

$$\mathcal{S} = \{(v_1, v_2) \mid v_1 \in \{\text{EMPTY}, \text{CLUSTER}\}, v_2 \in [R_{\min}, R_{\max}]\}. \tag{3.10}$$

Both the approaches allow us to define a map of resistances which drives the spread effect (this is a more realistic situation because we can observe spreading phenomena on previously defined diversified media). This map depends on the initial conditions.

Once the state set has been defined, we should define mechanics of the cluster growth. Similarly to the Eden model, the inclusion of a cell into the cluster depends on the probability $p_j$. Thus for each border cell we draw a number $p_1 \in [0, 1]$ and compare it with $p_j$. The result of this comparison will determine the new state: as a candidate for the cluster member or still the border. A major novelty constitutes the condition for checking the resistance. If a cell has been selected by the rule of joining it as a candidate for the cluster member, we draw the second number $p_2 \in [R_{\min}, R_{\max}]$ and compare it with the local value of resistance. Only this rule finally will permit us to qualify a cell for either the cluster, or the border.

**Example 3.3**
Application of a randomly generated resistance map such as that in [28] causes some problems with the analysis of the influence of the resistant map on the cluster spread process. Thus it is better to use the regular one, e.g. like the one presented on the first panel in Fig. 3.10. The grey and black colours represent sites with $R = 98$ and $R = 36$, respectively. The remaining (white) region represents the zone which lies outside the cluster spread, i.e. the cells there correspond to $R = R_{\max} = 100$. The cells with values 98 and 100 can be interpreted as supports of the so-called passive control as their states do not change and they constitute a barrier for other cells which cannot thus colonize the region outside the cross on the top-left panel in Fig. 3.10.

Exemplary simulation results of a CA with a lattice of 130×100 cells and von Neumann neighbourhood are shown in Fig. 3.10. The first panel shows the resistance map. The next five panels present CA configurations after 1, 20, 50, 100 and 150 iterations, respectively. We can observe that cells with smaller resistances are attached to the cluster (the black colour on the resistance map) in the first place, and then those with larger ones are joined in. The cells with $R = R_{\max}$ are never joined to the cluster since the drawn value $p_2 \in [R_{\min}, R_{\max}]$ is always less than or equal to $R_{\max}$, i.e.

$$p_2 \leq R_{\max}, \quad \forall p_2 \in [R_{\min}, R_{\max}]. \tag{3.11}$$

The graph of the corresponding cluster growth is presented in Fig. 3.11. We clearly see a difference from the Eden and simple percolation cluster models. In the 87-th step we observe a deterioration of the cluster growth process after exhausting free places with smaller resistances. The results prove that the spread phenomenon in this model is controlled by the the local value of resistance because a change in the resistance of a cell will only change the value of itself.

### 3.2.2. Limited-energy walker model

Many physical and natural processes can be modelled as moving particles. They can be e.g. diffusion, fluid flow, tension distribution, etc. Two main elementary CA models used for modelling walking particles are the following:

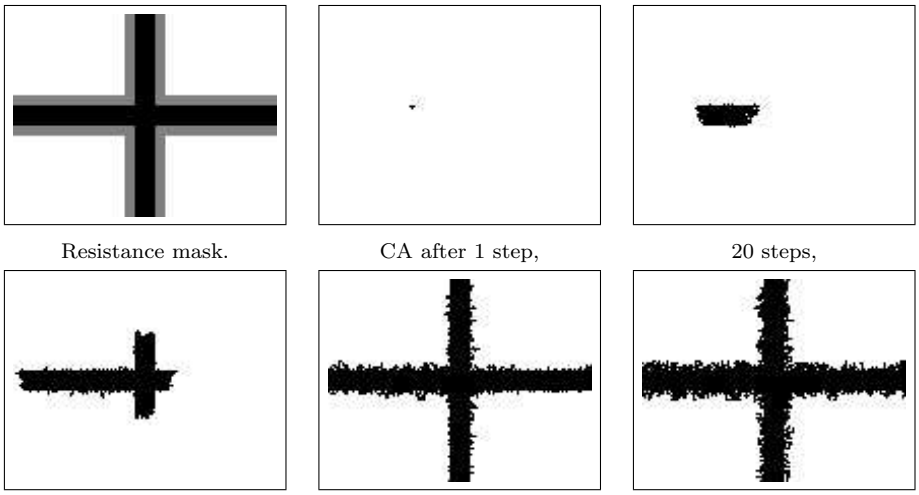



**Fig. 3.10.** Simulation results for the CA implementation of the invasion percolation model.

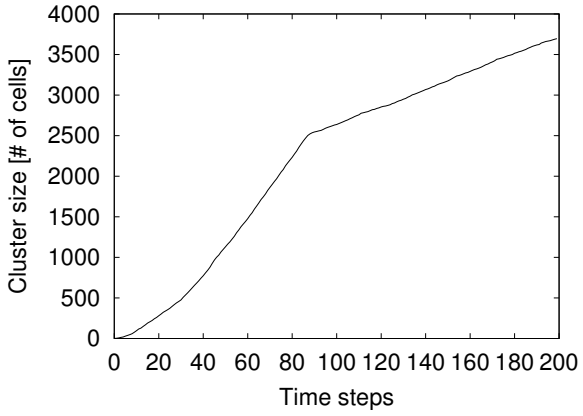



**Fig. 3.11.** Cluster growth for the CA implementing the invasion percolation model.

- *Multiple random walkers* [27] being a model which operates on a rectangular lattice with periodic boundary conditions. The state set defines a walk direction or the free space. Its transition function scans the modified von Neumann neighbourhood for free space to move a cell. The collisions are detected (particles randomly choose new directions).

- *Noise-driven diffusion* [77] being a deterministic CA model operating on the Margolus neighbourhood and on a rectangular lattice. The transition function depends on a particular variant of the model, but in general it allows for a particle movement and collision detection (molecules avoid one another, or collide elastically and change their direction of movement).

The idea to limit the movement energy was introduced so as to restrict the region of spreading particles. It is well-known that in some situations natural mixing of liquids is impossible (consider e.g. an oil spot on the water, water in petrol, etc.). Similarly, grains spilled on a table never cover it evenly. In this situations, typical models are of no use. An analysis of this fact and above-mentioned models (and others, cf. [77, 28, 27]) suggests a natural idea to expand the models by limiting the movement energy of the particles. These types of models are more appropriate for modelling liquids and solids.

In order to develop that model, we have to define a transport method for the property (a state set) which represents a particle. A problem arises however, when solving collision situations. There are three ways to overcome them:

- to check a neighbourhood of the target cell (the idea used in multiple random walkers [27]),

- to use the idea of the Margolus neighbourhood (using different cluster configuration for even and odd iterations),

- to use a hidden step, i.e. an iteration is made up of two steps:

  - a collision flag is evaluated according to the configuration (it is set to true if both the cell $c$ is empty and particles in two or more cells in the neighbourhood $N(c)$ point towards $c$);
  - if a cell is occupied by a particle and the particles in the neighbouring cells have activated the collision flag, their directions are changed; otherwise, if the cell is occupied, it passes the particle for the empty cell pointed by the particle.

A similar solution was used in [77] for tracing the flow.

The third approach above may look complicated, but it makes it possible to significantly simplify the transition function and makes it clearer (especially for larger neighbourhoods, when the first two solutions are more complicated).

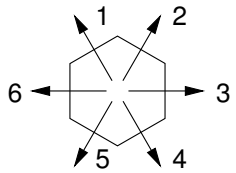
Then we have to define a lattice and a neighbourhood type. A large neighbourhood size involves a sophisticated transition function. A good solution seems

to be a hexagonal lattice with Moore neighbourhood. It ensures both a good accuracy (particles can move in six directions, whereas for the rectangular lattice we have only four directions) and a relatively simple transition function.

Finally, we have to define the state set. We know that each movement is related to a force with a defined direction, thus it is natural to define the state reflecting this value. If we assume that all possible direction values are as follows:

$$\mathcal{D} = \{0, 1, 2, 3, 4, 5, 6\},$$

where the values $\{1, 2, \ldots, 6\}$ signify the directions of the moving particle as given below:




The value 0 in $\mathcal{D}$ means that a cell is empty (not occupied by a moving particle). If we base a movement only on directions, we obtain an unlimited energy movement model (useful in gas modelling tasks, e.g. diffusion). To generate a limited energy movement model, we should supply a state with an energy level value. If we decide that each particle in the lattice can have one of $n_E$ energy levels, i.e.

$$\mathcal{E} = 0, 1, 2, \ldots, n_E,$$

then our state set is of the form

$$\mathcal{S} = \{(d, e) \mid d \in \mathcal{D}, e \in \mathcal{E}\}. \tag{3.12}$$

#### 3.2.2.1. Algorithm

Each cell on a limited-energy walker lattice may be in one of three general states:

- When a *cell c is empty*, its neighbourhood should be checked for a cell occupied by a moving particle and directed to $c$. If more than one particle are directed towards $c$, a collision is detected and the cell $c$ remains in the same state (direction $d_{c,t+1} = 0$ and energy $d_{c,t+1} = 0$). In this case a *collision flag* $\mathcal{F}_c$ is set to be true (this flag is used later in the hidden step). Otherwise, $c$ takes the value of the moving particle directed towards it, and the energy level is decreased.

- When *cell c is occupied by a moving particle* ($d_{c,t} > 0$ and $e_{c,t} > 0$), the neighbouring cell pointed by $d_{c,t}$ is to be checked. If the pointed cell is occupied by a stopover particle ($d_{\gamma,t} > 0$ and $e_{\gamma,t} = 0$), only the energy value is decreased (e.g. $e_{\gamma,t+1} = e_{c,t} - 1$). If the pointed cell is empty and its *collision flag* is clear (set to false), $c$ takes the value of the empty cell

**Tab. 3.1.** External forces for a cell $c$ depending on the direction $d_c$ and the energy level $e_c$ (when assuming that the force is directly proportional to $e_c$).

| $d_c$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |
|---|---|---|---|---|---|---|
| 1 | $e_c$ | $0.5e_c$ | $-0.5e_c$ | $-e_c$ | $-0.5e_c$ | $0.5e_c$ |
| 2 | $0.5e_c$ | $e_c$ | $0.5e_c$ | $-0.5e_c$ | $-e_c$ | $-0.5e_c$ |
| 3 | $-0.5e_c$ | $0.5e_c$ | $e_c$ | $0.5e_c$ | $-0.5e_c$ | $-e_c$ |
| 4 | $-e_c$ | $-0.5e_c$ | $0.5e_c$ | $e_c$ | $0.5e_c$ | $-0.5e_c$ |
| 5 | $-0.5e_c$ | $-e_c$ | $-0.5e_c$ | $0.5e_c$ | $e_c$ | $0.5e_c$ |
| 6 | $0.5e_c$ | $-0.5e_c$ | $-e_c$ | $-0.5e_c$ | $0.5e_c$ | $e_c$ |

($d_{c,t+1} = 0$ and $e_{c,t+1} = 0$). Otherwise, we have a collision. A new direction is selected (we randomly turn the particles left or right). Some exemplary collisions are shown in Fig. 3.14.

- When *cell $c$ is occupied by a stopover particle* ($d_{c,t} > 0$ and $e_{c,t} = 0$), its neighbourhood is checked for a cell occupied by a moving particle and directed to $c$. If such a cell or cells are found, the energy level $e_{c,t+1}$ is increased in proportion to the number of those cells (e.g. if three cells from $N(c)$ point to $c$, then $e_{c,t+1} = e_{c,t} + 3$). The new direction depends on the directions of cells which make the particle in cell $c$ moving, and their energy levels. For the movement on a hexagonal lattice, we can distinguish three main directions, e.g. 1, 2 and 3 (the choice of the opposite directions can be expressed as $-1$, $-2$ and $-3$). It is simple to calculate the force in each of these directions:

$$F_1 = F_{1,4} - F_{4,1} + F_2 \cos\alpha - F_3 \cos\alpha \tag{3.13}$$

where $F_1$ represents the force in direction 1, $F_{1,4}$ is the external force directed to 1 from cell 4 (c.f. Fig. 3.12), and $\alpha$ represents an angle between the directions. The values of external forces are shown in Tab. 3.1. All the forces are presented in Fig. 3.13. Similarly, we calculate the forces in directions 2 and 3:

$$F_2 = F_{2,5} - F_{5,2} + F_1 \cos\alpha + F_3 \cos\alpha \tag{3.14}$$

$$F_3 = F_{3,6} - F_{6,3} - F_1 \cos\alpha + F_2 \cos\alpha \tag{3.15}$$

An exemplary situation of the movement on a stopover cell is shown in Fig. 3.15.

One iteration of this model consists of two steps. First, we update the *collision flag* $\mathcal{F}_c$ for every empty cell $c$. Then, we perform the main step: depending on the state of the cell described above, we update the state value:

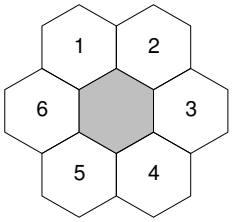



**Fig. 3.12.** Moore neighbourhood for a hexagonal lattice.

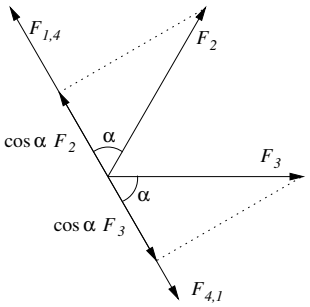



**Fig. 3.13.** Forces which affect $F1$.

- the empty cell with cleared *collision flag* gets the incoming particle or stays empty;
- the empty cell with enabled *collision flag* remains empty;
- the occupied cell with a free way in the neighbourhood for its moving particle passes this particle to it;
- the occupied cell with a blocked way changes the movement direction or the return part of energy for the stopover particle.

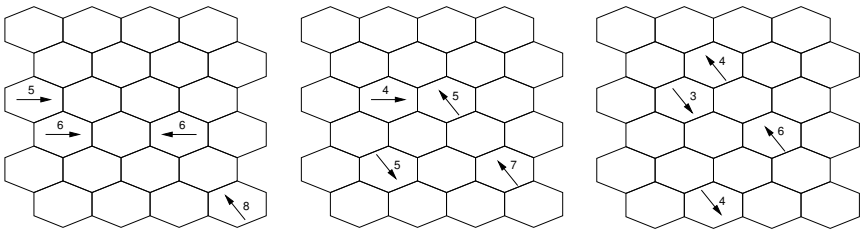



**Fig. 3.14.** An exemplary sequence with collisions.

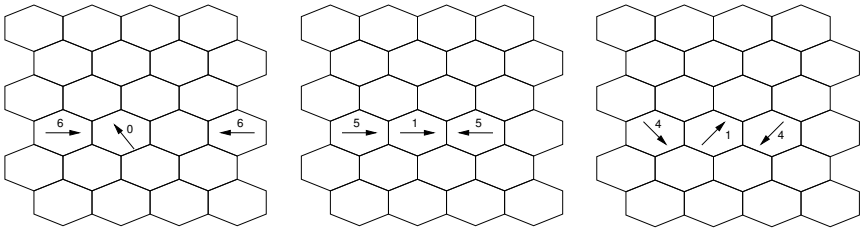



**Fig. 3.15.** An exemplary sequence with energy transfer.

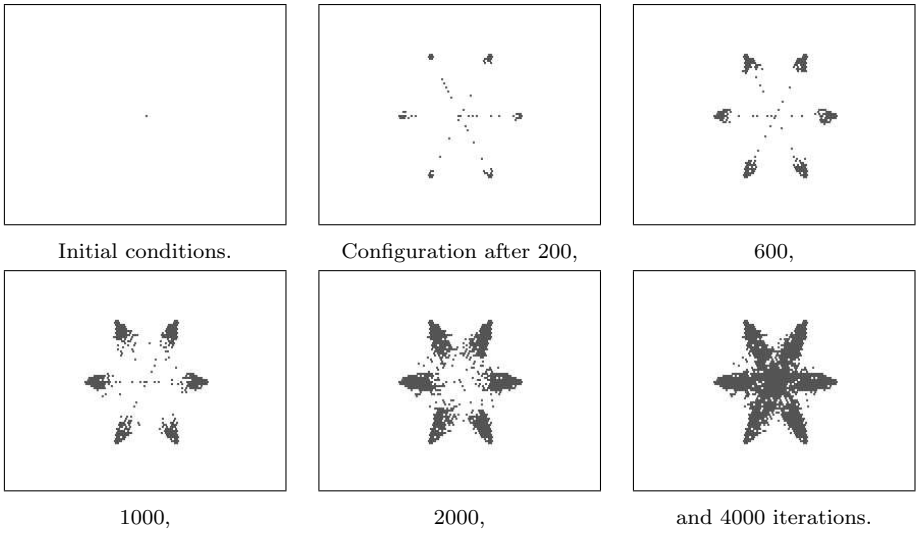



**Fig. 3.16.** Evolution of the CA implementing the limited energy walker model.

**Example 3.4**
Exemplary simulation results of a CA and a hexagonal lattice of 130×100 cells and von Neumann neighbourhood are shown in Fig. 3.16. The source of moving particles are placed in the centre of the lattice. This place represents a control, which by modification of its value generates an evolution of the neighbours. In each iteration, the control generates a state representing a moving particle with random direction and moving energy $e = 30$.

In contrast to another walking particle model [77, 28, 27], in this model the motion of the particles is limited. For this reason, particles form a cluster and they do not disperse. Due to its simplicity (we do not consider the pressure, viscosity and surface film) the model better expresses the character of the moving particles of a solid, but not a fluid where other phenomena are of greater importance.

## 3.3. Conclusions

Spreading is a process in which an object extends itself over an increasingly larger area by incorporating regions adjacent to itself. A wide variety of natural processes can be described by the spreading or kinetic growth model. Examples include tumour growth, epidemic spread, gelation, rumour-mongering, and fluid flow through porous media. In this chapter, four models were proposed to model spreading phenomena via CA's. Two of them (Eden and single percolation cluster models) can be used to describe the epidemic spread of a disease. The next two (invasion percolation and limited-energy walker models) involve the notion of control, which seems to constitute a new idea regarding CA's (CA's are normally considered as fully autonomous systems). The control making a system spreadable is called spray control.

It turns out that simulations of spreadable systems through CA's are relatively simple, and beyond doubt much simpler than those implied by using partial differential equations [44, 45, 78]. The results are especially important e.g. in instances of invasions of a biological entity into an environment previously free from that entity (for example, yellow stripe rust in a field of wheat) or in the case of spreading forest fires. The continued development of the theory is thus needed, as good theory will be invaluable in predicting coming events and hopefully in successfully intervening to check the spread of invaders.

# Chapter 4

# Parameter estimation of cellular automata models

## 4.1. Introduction

Model and modelling are catchwords with many different interpretations. In systems analysis a model is a mathematical description of a real process, built with a definite aim in mind. The model $M$ of the system is a rule to compute, from quantities known *a priori* or measured from the system, other quantities that we are interested in and which we hope will resemble their actual values in the system. Whatever the structure chosen for the model, it will in general involve unknown quantities, usually assumed to be constant, to be estimated from available prior knowledge and data. These quantities are the parameters $\vartheta$. One then speaks of a parametric model, the main type of model to be considered in what follows. The choice of $M$ is a critical step in modelling. Once the model structure has been selected, its parameters must be chosen according to a specified criterion, usually the optimization of some cost function being a measure of the output error, i.e. the difference between the system and model outputs.

As we know from Section 1.5, the transition function $f$ determines the dynamics of the corresponding CA. In some cases the creation of this function may be very simple, e.g. when we know the modelled phenomenon in detail and we can describe it by a transition function directly. Unfortunately, some of the phenomena may be very hard to describe in this manner, because we know only a general form of the rules which govern its behaviour, up to a set of unknown parameters to be determined based on observations. In other situations, even the form of the rules may be unknown, so that we can postulate a model (this is the so-called *black-box* approach) and then the problem reduces to finding the appropriate parameters of the assumed model.

An example of such a parametric transition function (PTF) for a two-dimensional CA is the following formula:

$$\begin{aligned} &f(s_t(c_{i,j}), s_t(c_{i-1,j}), s_t(c_{i+1,j}), s_t(c_{i,j-1}), s_t(c_{i,j+1})) \\ &\quad = \vartheta_0 s_t(c_{i,j}) + \vartheta_1 s_t(c_{i-1,j}) + \vartheta_2 s_t(c_{i+1,j}) + \vartheta_3 s_t(c_{i,j-1}) + \vartheta_4 s_t(c_{i,j+1}) \end{aligned} \quad (4.1)$$

which make it possible to model simple phenomena based on the von Neumann neighbourhood. For example, if we set $\vartheta_1$ as one, and the remaining parameters as zero:

$$\vartheta_i = 0, \quad i = 0, 2, 3, 4,$$

we obtain the up-down scrolling effect. Another model of the form

$$f(s_t(c_{i,j}), s_t(c_{i-1,j}), s_t(c_{i+1,j}), s_t(c_{i,j-1}), s_t(c_{i,j+1})) = \frac{1}{5}(s_t(c_{i,j}) + s_t(c_{i-1,j}) + s_t(c_{i+1,j}) + s_t(c_{i,j-1}) + s_t(c_{i,j+1})) \quad (4.2)$$

is used in graphical applications when introducing a blend effect. In this case parameters $\vartheta_i = 1/5$, $i = 1, \ldots, 5$ determine the intensity of the blend effect in each of the directions. A proper selection of parameters makes it possible to generate a wide range of effects, from delicate blending of sharp edges by smoothing lines to a complete blur of the picture.

A disadvantage of parametric models is sometimes an arbitrary choice of the model structure. For example, we may try to develop a parametric model with a fixed von Neumann neighbourhood type whereas the real dynamics of the studied phenomenon is based on the Moore neighbourhood. In this case even the best selection of parameters produces only an approximate model. On the other hand, if we choose a model with a vast neighbourhood, it will introduce a lot of parameters which must be tuned.

Up to now, the problem of parameter estimation of CA models was only taken into consideration by Uciński [79] and Yang [93]. Here those results are generalized. First, we propose a parameter estimation scheme using some known optimization algorithms. This approach guarantees some robustness against disturbances corrupting observations. Then the parameter estimation problem for probabilistic CA (PCA's) models will be considered.

## 4.2. Parameter estimation problem

In order to define the parameter estimation problem, we need to introduce a parametric transition function. In general, we can describe it as follows [79]:

$$s_{t+1}(c) = f(s_t(N(c)), \vartheta), \quad (4.3)$$

where

$s_{t+1}(c)$ – the state of the cell $c$ at time $t+1$,

$N(c)$ – the set of cells which form the neighbourhood of cell $c$, and

$\vartheta$ – denotes the vector of unknown constant parameters.

An estimate $\hat{\vartheta}$ of the parameter vector $\vartheta$ is to be found based on the results of observations $z_t(c)$ (possibly noise-corrupted) of the actual state $s_t(c)$ for $(c,t) \in Q \subset \mathcal{L} \times T$. As is commonly accepted in dynamic system identification [30,

86], we wish to cast this task as an optimization problem. That problem is not simple to solve in a classical manner because the CA is a discrete system. The commonly used fit-to-data measure like the sum of errors (4.4) emphasizes this discrete character:

$$J(p) = \sum_{(c,t)\in Q} \rho(z_t(c), \hat{s}_t(c;\vartheta)) \longrightarrow \min \tag{4.4}$$

where $\hat{s}_t(c;\vartheta)$ signifies the state of the cell $c$ at time $t$ in the solution to eqn. (4.3) calculated for a given vector $\vartheta$, and $\rho(z,s)$ constitutes a measure of discrepancy between $z$ and $s$, e.g.

$$\rho(z,s) = \begin{cases} 0 & \text{if } z = s, \\ 1 & \text{otherwise} \end{cases} \tag{4.5}$$

or $\rho(z,s) = |z-s|$. Note that in this approach the initial lattice configuration should be completely known (the same is with boundary conditions, but these are usually assumed periodical).

To find a minimum of the cost function $J$, we need to use an optimization method which operates on non-continuous functions with gradient equal zero where it exists. In this chapter we will show the usefulness of the global adaptive-random-search (ARS) algorithm proposed by Walter and Pronzato [86], local search [63] and simulated annealing [65, 66].

### 4.2.1. Applying the local search algorithm

The local search algorithm is one of the simplest optimization methods useful for the discrete optimization process. It bases on a search for a better configuration of parameters in the close neighbourhood of the current configuration. It is stopped after exceeding a maximum number of iterations or when a satisfactory result is found.

A general scheme of the local search algorithm is as follows [63]:

**Initiation:** An initial parameter vector $\vartheta_0$ is generated. Set $n = 0$.

**Looking for a successor:** A direct surrounding of the current $\vartheta_n$ is searched for a better value of the criterion, i.e. we wish to obtain a $\vartheta_{n+1}$ such that $J(\vartheta_n) > J(\vartheta_{n+1})$ (the best of the surrounding combinations is chosen).

**Check stop conditions:** If there are no better values of the parameters vector $\vartheta$, a satisfactory result is achieved or the maximum of iteration is exceeded, then STOP, otherwise increment $n$ and go back to *looking for another successor*.

In order to increase the speed and accuracy, an adaptive step-size can be used. This feature consists in gradually decreasing the radius of the vicinity of the current solution which is used to look for a successor. This decrease is performed from iteration to iteration. Owing to this feature, we ensure a better (more detailed) search for the final result.

Application of this algorithm for parameter estimation of CA models is simple and does not require any special modifications.

### 4.2.2. Application of simulated annealing

The method of simulated annealing is a technique that has attracted significant attention as suitable for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema. The implementation of the algorithm is relatively simple. At the heart of the method is an analogy with thermodynamics, specifically the way that liquids freeze and crystallize, or metals cool and anneal.

The simulated annealing algorithm performs the minimization of a function $E(\vartheta)$ called the energy of the system associated with parameter $\vartheta$. The minimization process is controlled by a parameter $\vartheta$ which is called temperature in the following manner: at the beginning $\vartheta$ is set to a high value which is decreased with the progress of calculations. For large values of $\vartheta$ the energy of the system configuration can be freely changed, whereas for small values only small modifications are acceptable.

The algorithm of this method can be outlined as follows [65, 66]:

**Initiation:** An initial parameter vector $\vartheta_0$ is set and the corresponding energy $E(\vartheta_0)$ is calculated (it is our fit-to-data criterion). Set $n = 0$.

**Try a new parameter:** A new parameter $\vartheta^+$ is randomly created.

**Evaluate the energy:** The energy of the new parameter $E(\vartheta^+)$ is evaluated.

**Calculate the probability of transition:** The probability 'prob' of changing the current parameter $\vartheta_n$ into the new one $\vartheta^+$ is calculated based on their energy levels:

$$\text{prob} = \exp\left(-\frac{E(\vartheta^+) - E(\vartheta_n)}{k\theta}\right)$$

where $k$ is the Boltzmann constant. In the case when $E(\vartheta^+) < E(\vartheta_n)$, the value of 'prob' is greater than 1, so it is truncated to the maximal value $\text{prob} = 1$.

**Toss a coin:** Generate a pseudo-random real number between 0 and 1. If its value is less than 'prob', the new parameter becomes the current parameter, i.e. $\vartheta_{n+1} = \vartheta^+$ and $n$ is incremented, otherwise *try a new parameter*.

**Chill process:** The process temperature $\theta$ is decreased. If it attains its minimal value, then the process is stopped, otherwise *try a new parameter*.

**Example 4.1**
Let us consider the parameter estimation problem of the general CA model as follows:

$$s_{t+1}(c_{i,j}) = \text{nearest}(\vartheta_0 s_t(c_{i,j}) + \vartheta_1 s_t(c_{i-1,j}) + \vartheta_2 s_t(c_{i+1,j}) + \vartheta_3 s_t(c_{i,j-1}) + \vartheta_4 s_t(c_{i,j+1}))$$

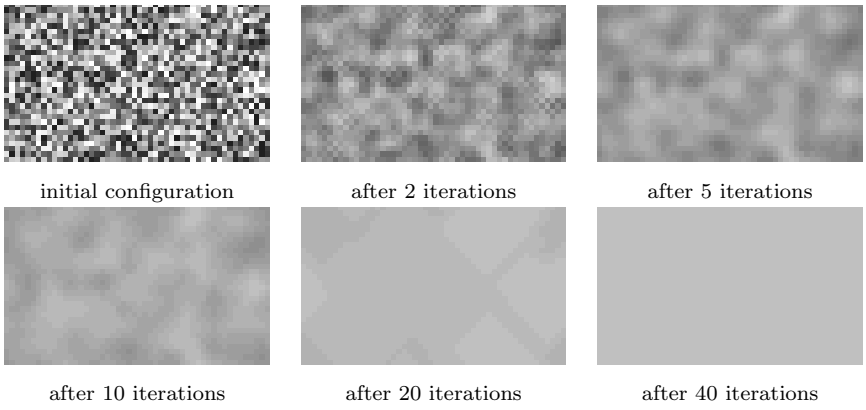



**Fig. 4.1.** An exemplary evolution of the CA modelling heat transfer.

where nearest($a$) means the number from the state set $\mathcal{S}$ which is the nearest to $a$.

Because for the experiment the same value of true parameters $\vartheta_0 = \cdots = \vartheta_4 = 1/5$ is assumed, the model represents the classical blending effect used in graphical applications or phenomena of heat transfer in homogeneous media.

Fig. 4.1 shows an exemplary evolution of this model for the 32-elements state set ($\mathcal{S} = \{1, 2, \ldots, 32\}$) and the rectangular lattice of 50×30 cells.

For the process of parameter estimation by using local search and simulating annealing algorithms, the set of 3000 samples (each sample consisted of the initial and first configurations for the von Neumann neighbourhood) generated from random initial conditions were used. 10% of these data samples were randomly changed to another value randomly drawn from the state set with uniform distribution (consequently, sometimes the difference between real and disturbed states was really large, e.g. $s_t(c) = 1$ could change to $s_t(c) = 32$). The local search was implemented by veryfying all the points constructed by adding or subtracting a fixed step from the components of the vector $\vartheta$ (all possible combinations were thus checked). The step was gradually changed from 1 to $10^{-6}$. As for the simulated annealing, the value of 2.76 was assumed for the product $k\theta$. After 4010 cost evaluations for criterion (4.4) and function (4.5), the following parameters vector was obtained:

- for the local search algorithm:

$$\hat{\vartheta} = [0.201583, 0.193562, 0.196325, 0.210187, 0.189647],$$

the corresponding relative error was

$$\delta_\vartheta = [0.79\%, 3.22\%, 1.84\%, 5.09\%, 5.18\%];$$

- for the simulated annealing:

$$\hat{\vartheta} = [0.201422, 0.200138, 0.190213, 0.205234, 0.192253],$$

the corresponding relative error was

$$\delta_\vartheta = [0.71\%, 0.07\%, 4.89\%, 2.62\%, 3.87\%];$$

where the true values were

$$\vartheta^* = [0.200000, 0.200000, 0.200000, 0.200000, 0.200000].$$

### 4.2.3. Applying an adaptive random search algorithm

The adaptive random search (ARS) algorithm [86] consists in changing the current estimate $\hat{\vartheta}$ of the parameter vector by adding a random vector $r$ which possibly improves the current solution (reduces the performance index $J$). It is assumed that the parameter belongs to a set of admissible values $\Theta$.

The ARS algorithm is especially suited for that purpose if the set of admissible parameters $\Theta$ is a hypercube, i.e. the admissible range for $\vartheta_i$, $i = 1, \ldots, n$ is in the form

$$\vartheta_{i\,\min} \leq \vartheta_i \leq \vartheta_{i\,\max}. \tag{4.6}$$

The routine chooses the initial point $\vartheta^0$ at the centre of $\Theta$. After $q$ iterations, given the current best point $\vartheta^q$, a random displacement vector $\Delta\vartheta$ is generated and the trial point

$$\vartheta^+ = \Pi_\Theta(\vartheta^q + \Delta\vartheta) \tag{4.7}$$

is checked, where $\Delta\vartheta$ follows a multinormal distribution with zero mean and covariance

$$\mathrm{cov}\{\Delta\vartheta\} = \mathrm{diag}[\sigma_1, \ldots, \sigma_n] \tag{4.8}$$

$\Pi_\Theta$ being the projection onto $\Theta$.

If $J(\vartheta^+) < J(\vartheta^q)$, then $\vartheta^+$ is rejected and consequently we set $\vartheta^{q+1} = \vartheta^q$, otherwise $\vartheta^+$ is taken as $\vartheta^{q+1}$. The adaptive strategy consists in repeatedly alternating two phases. During the first one (variance selection) $\mathrm{cov}\{\Delta\vartheta\}$ is selected from among the sequence $\sigma_1, \sigma_2, \ldots, \sigma_5$, where

$$\sigma_1 = \vartheta_{\max} - \vartheta_{\min} \tag{4.9}$$

and

$$\sigma_i = \sigma_{(i-1)}/10, \quad i = 2, \ldots, 5 \tag{4.10}$$

With such a choice, $\sigma_1$ is large enough to allow for an easy exploration of $\Theta$, whereas $\sigma_5$ is small enough for a precise localization of an optimal point. In order to allow a comparison to be drawn, all the possible $\sigma_i$'s are used $100/i$ times,

starting from the same initial value of $\vartheta$. The largest $\sigma_i$'s, designed to escape local maxima, are therefore used more often than the smaller ones.

During the second (exploration) phase, the most successful $\sigma_i$ in terms of the criterion value reached during the variance selection phase is used for 100 random trials started from the best $\vartheta$ obtained so far. The variance-selection phase then resumes, unless the decision to stop is taken.

To use ARS algorithm, the user must only provide:

- the prior feasible domain for the parameters, assumed to be the box

$$\Theta = \{\vartheta : \vartheta_{i\ \min} \leq \vartheta_i \leq \vartheta_{i\ \max},\ i = 1, \ldots, n\}, \tag{4.11}$$

- the definition of the cost function $J(\vartheta)$ for any $\vartheta \in \Theta$,
- the maximum number of evaluations of the cost allowed,
- the minimum cost value which satisfies the user as a solution.

**Example 4.2**
Let us consider the CA model

$$s_{t+1}(c_i) = \lfloor \vartheta_1 s_t(c_{i-1}) + s_t(c) + \vartheta_2 s_t(c_{i+1}) \rfloor \pmod 5 \tag{4.12}$$

$\lfloor x \rfloor$ being the greatest integer less than or equal to $x$. The CA is observed over the whole space-time domain ($Q = \mathcal{L} \times T$), where the simulated data were generated for a 50-cell lattice and 50 time steps, $\vartheta_1 = -1$, $\vartheta_2 = 2$ and the initial state defined as

$$s_0(c_i) = \begin{cases} 1 & \text{if } i = 23, \ldots, 28 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

In the considered example, the admissible set $\Theta_{\text{ad}} = [-100, 100]^2$ was assumed for the estimates. First, the noise-free case was examined. After 131 evaluations of $J$ using the function (4.5), its zero value was attained for the estimate

$$\hat{\vartheta} = (34.003504, 12.054030), \qquad \delta_\vartheta = (0.01\%, 0.45\%)$$

being the relative error. These values are only apparently strange, because one has

$$34 = -1 \pmod 5, \quad 12 = 2 \pmod 5$$

But this phenomenon suggests that the estimates may be non-unique. This issue has been studied more deeply by calculation of the cloud of points representing the boundary of the region around $\hat{\vartheta}$ for which the fit-to-data criterion $J$ also takes the zero value (the appropriate method is delineated in [86, pp.238–239]). This region is shown in Fig. 4.2(a). All the points within this region are equally well, i.e. for them the value of $J$ is minimum.

Furthermore, a noisy case has also been studied. For all 2500 data points, the gathered data were simulated to be smeared with additive noise, each with

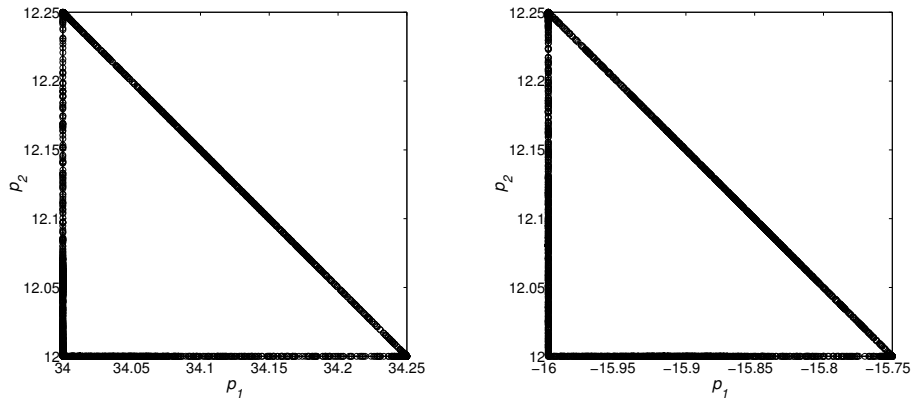



**Fig. 4.2.** Characterization of the boundary of the posterior admissible parameter set by a cloud of 1000 points: noise-free (a) and 20%-noise-level (b) cases.

probability 0.2. Adding noise amounts to adding $+1$ or $-1$ (each value, in turn, with probability 0.5). As a result, 498 data points actually changed their values. After 3080 cost evaluations, the vector

$$\hat{\vartheta} = (-15.902937, 12.030210), \qquad \delta_\vartheta = (6.02\%, 0.25\%)$$

being the relative error, has been obtained, along with the cloud of points evidenced in Fig. 4.2(b). Surprisingly, the results are still correct, which confirms that the adopted estimation method is also robust against outliers [86].

**Example 4.3**
To check the more complex example let us to consider parameter estimation of the crystalization process CA model.

The crystallisation process itself produces some effects that tend to inhibit crystal growth: the diffusion of latent heat at the solid-liquid interface and surface tension. The model employs a square lattice of 405×405 cells. Each site in the lattice has an ordered pair as a value. The first component is a Boolean value, where 1 represents a site that is crystalline and 0 represents a site that is amorphous. The second component of the ordered pair is a nonnegative real number which can change continuously, representing the temperature of the site. At $t = 0$, the lattice consists of $(0, 0)$ pairs except for the central 5×5 crystal seed, consisting of random $(0, 0)$ and $(1, u_c)$, where $u_c$ is a given positive temperature. This initial state is assumed to be known to the experimenter.

The crystallisation process proceeds over $n_t = 40$ time steps, in each of which two consecutively executed events occur (a detailed model is given in [27]):

- An amorphous site becomes crystalline if it is adjacent to at least one crys-

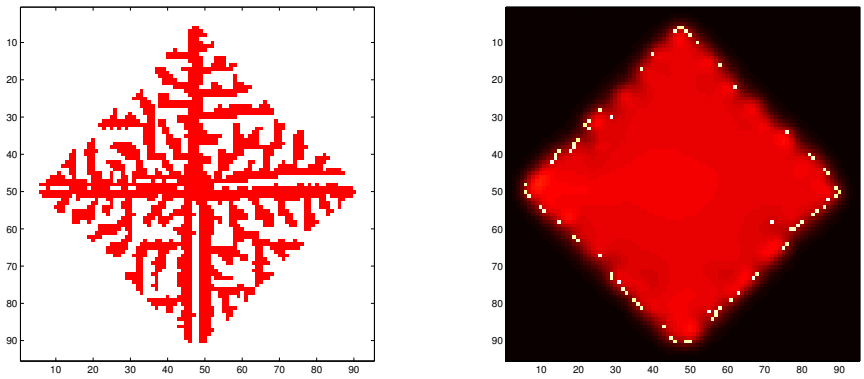



**Fig. 4.3.** An exemplary state ($t = 40$) of the CA modelling growing crystal.

talline site and if some condition related to the local interface curvature and temperature is met. Latent heat of crystallisation is produced at a crystallizing site, which raises its temperature. An amorphous site that does not meet the prerequisite condition remains liquid and a crystalline site remains unchanged (i.e. there is no melting in the model).

The phase and temperature values of each site in the lattice are updated based on its temperature and phase and on the phases of the sites lying in its Moore neighbourhood.

- Heat diffuses to adjacent cells, i.e. the temperature of each site in the lattice and the temperature of the nearest Moore neighbour sites.

As regards observations, we assume that only the temperatures at $t = 10$, 20, 30 and 40 are known (so we do not know e.g. whether a site is crystalline or amorphous). To illustrate evolution of this automaton, Fig. 4.3 shows the solid-liquid configuration (the left panel) and the temperature field (the right panel) at $t = 40$ (it is only a fragment of the whole lattice, but the other sites have zero values).

Based on those measurements, two coefficients: the diffusion constant $D$ and the latent heat $h$ are to be identified. Since significant persistent experimental noise is assumed in the model (10%), the minimized fit-to-data criterion is the sum of absolute values of the errors at the observed sites, as such a criterion is more robust to the presence of outliers in the data.

After 2500 criterion evaluations by the ARS global minimization algorithm, the values $\hat{D} = 3.720626$ and $\hat{h} = 0.917713$ have been obtained, which is quite good when compared with the true values $D^* = 4.0$ and $h^* = 1.0$ (the corresponding relative errors values are $\delta_D = 6.98\%$ and $\delta_h = 8.23\%$). For the same conditions, but with the 5% level of noise, after 3000 criterion evaluations, the values $\hat{D} = 3.983624$ and $\hat{h} = 0.979168$ ($\delta_D = 0.41\%$ and $\delta_h = 2.08\%$) have been achieved.

**Tab. 4.1.** Transition table for an exemplary probabilistic CA.

| $s_t(N(c_i))$ | | | $s_{t+1}(c_i)$ | |
|---|---|---|---|---|
| $s_t(c_{i-1})$ | $s_t(c_i)$ | $s_t(c_{i+1})$ | 0 | 1 |
| 0 | 0 | 0 | $p_0$ | $1-p_0$ |
| 0 | 0 | 1 | $p_1$ | $1-p_1$ |
| 0 | 1 | 0 | $p_2$ | $1-p_2$ |
| 0 | 1 | 1 | $p_3$ | $1-p_3$ |
| 1 | 0 | 0 | $p_4$ | $1-p_4$ |
| 1 | 0 | 1 | $p_5$ | $1-p_5$ |
| 1 | 1 | 0 | $p_6$ | $1-p_6$ |
| 1 | 1 | 1 | $p_7$ | $1-p_7$ |

## 4.3. Parameter estimation of stochastic CA models

In contrast to parameter estimation of deterministic models where deterministic rules were used, for probabilistic cellular automata (PCA) models the transition rules have probabilistic character. Parameter estimation of those models seems to be rather complex and only simple situations are considered in the literature [1]. In this section the maximum-likelihood [86] approach is recommended for this purpose.

In much the same way as for deterministic CA models, we assume that the model structure is given and information about its parameters should be gained from observation of the modelled phenomena. In the general case, probabilistic phenomena will be described by a set of probabilities which define possibilities of state changes for a given configuration. In this case, we can simply put together all these probabilities into a probabilistic transition table. An example of such a table for a one-dimensional automaton, the neighbourhood definition $N(c) = (c_{i-1}, c_i, c_{i+1})$ and a binary state set is shown in Table 4.1.

Each sum of probabilities located in the same row is always 1. In the general case, when the size of the neighbourhood is equal to $n$ and the state set contains $m$ states, we must build a table of $m^n$ rows and $m$ columns related to particular transition probabilities ($p_{i,j}$), where for each $i$ meaning the same configuration of the neighbourhood (the same row), we have

$$\sum_{j=1}^{m} p_{i,j} = 1$$

For the framework so defined, a probabilistic event $E_k$ of changing the state for a given state of the neighbourhood $s_t(N(c))$ to $s_{t+1}(c) = s^k \in \mathcal{S}$ eliminates the possibility of appearance of another event $E_\ell$ (which would change the state from the same state of the neighbourhood $s_t(N(c_i))$ to another value $s^\ell$ ($s^\ell \neq s^k$ and $s^\ell \in \mathcal{S}$). Thus the probability of the event $E_k$ is approximately equal to the relative

frequency of the occurrence of $E_k$ for the same neighbourhood configuration. For example, if we observe 100 changes of the state for the neighbourhood configuration

$$s_t(N(c_i)) = (0,1,0), \quad N(c_i) = (c_{i-1}, c_i, c_{i+1})$$

for the PCA with the state set $\mathcal{S} = \{0,1,2\}$, then observing event $E_0$ 45 times, event $E_1$ 35 times and event $E_2$ 20 times, the probability of changing the state for configuration $s_t(N(c_i)) = (0,1,0)$ to $s_{t+1}(c_i) = 1$ is approximately $p_1 \approx 0.35$. To better understand this mechanism, let us present the observation results in the following table:

| $s_t(N(c_i))$ | | | Number of events | | | Probability of events | | |
|---|---|---|---|---|---|---|---|---|
| $s_t(c_{i-1})$ | $s_t(c_i)$ | $s_t(c_{i+1})$ | $n_{E_0}$ | $n_{E_1}$ | $n_{E_2}$ | $p_{E_0} \approx \frac{n_{E_0}}{n_E}$ | $p_{E_1} \approx \frac{n_{E_1}}{n_E}$ | $E_2 \approx \frac{n_{E_2}}{n_E}$ |
| 0 | 1 | 0 | 45 | 35 | 20 | 0.45 | 0.35 | 0.20 |

In this way we can outline the general algorithm to approximate all the parameters from the probabilistic transition table. Assume that the respective row of the transition table (note that each row corresponds to a unique configuration of the neighbourhood) is indexed by $k$ and the state values are indexed by $\ell$ (so that $\mathcal{S} = \{s^1, \ldots, s^m\}$). The algorithm reads as follows:

**Initiation:** Set all table counters ($n_{k,\ell}$) to zeros.

**Get Data:** Check whether a new observation exists. If *not*, go to Step *Calculate Probabilities*, otherwise go to *Update Counters*.

**Update Counters:** Observe what happens for the neighbourhood configuration $s_t(N(c))$ corresponding to the current observation, i.e. which value $s_{t+1}(c)$ produced. If $s_{t+1}(c) = s^{\ell_0}$, increase the counter of the corresponding event ($n_{k,\ell_0}$). Then go to *Get Data*.

**Calculate Probabilities:** For each row, first of all calculate the global number of events $n_k = \sum_{\ell=1}^{m} n_{k,\ell}$, then evaluate the approximated probabilities for all events ($p_{k,\ell} \approx n_{k,\ell}/n_k$).

It is simple to conclude that the quality of estimation strongly depends on the amount of data and their quality (diversity). The block scheme of the proposed algorithm is presented in Fig. 4.4.

The above method relies on the frequency interpretation of probability. In more complex situations, the maximum-likelihood can turn out to be very useful. This is shown in what follows, based on some examples.

**Example 4.4**
Consider calculation of the immunity probability $p_2$ and lighting probability $p_3$ in the stochastic forest fire model discussed in Sec. 2.2.2, p. 40. These probabilities are jointly present in two rules related to transitions from state 1 to state 2. For brevity, introduce the following notation:

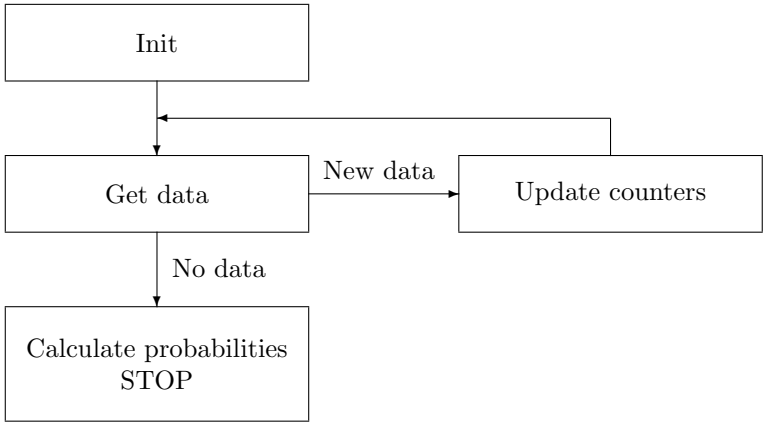



**Fig. 4.4.** Block diagram for the algorithm of parameter estimation of PCA models using the frequency interpretation of probability.

$n_1$–the number of observed transitions from 1 to 2 if at least one neighbour is burning,

$n_2$–the number of observed transitions from 1 to 1 if at least one neighbour is burning,

$n_3$–the number of observed transitions from 1 to 2 if no nearest neighbour is burning,

$n_4$–the number of observed transitions from 1 to 2 if no nearest neighbour is burning.

The likelihood function [86] related to this situation is then given by the formula

$$L = (1-p_2)^{n_1} p_2^{n_2} [p_3(1-p_2)]^{n_3} [1-p_3(1-p_2)]^{n_4}. \tag{4.14}$$

The maximum-likelihood estimates of $p_2$ and $p_3$ [86] are those which maximize $L$, and we can obtain them by calculating the derivatives $\partial(\log L)/\partial p_1$ and $\partial(\log L)/\partial p_2$ and setting the results to zero. Solving the obtained system of equations, we get

$$\hat{p}_2 = \frac{n_2}{n_1+n_2}, \quad \hat{p}_3 = \frac{n_3}{\hat{p}_2(n_3+n_4)} \tag{4.15}$$

For the estimation process it is very important to use a sufficiently diversified set of data. It is very interesting how the initial conditions affect on the quality of results. Because the initial conditions depend on the tree density and fire presence in the initial configuration, we could try to determine how to choose these values so as to improve the quality of results. Simulations confirm that for estimation it is better that the initial configuration be strongly filled with trees and fire. But

**Tab. 4.2.** Probabilistic transition table for the model of the spatial host-parasite dynamics.

| $s_t(c)$ | $P(s_{t+1}(c)=0)$ | $P(s_{t+1}(c)=1)$ | $P(s_{t+1}(c)=2)$ |
|---|---|---|---|
| 0 | $(1-p_G)^{n_1(c)}$ | $1-(1-p_G)^{n_1(c)}$ | 0 |
| 1 | 0 | $(1-p_T)^{n_2(c)}$ | $1-(1-p_T)^{n_2(c)}$ |
| 2 | $p_V$ | 0 | $1-p_V$ |

this is clear, because it is hard to determine the probability of fire if we have only a few events of fire.

Simulations carried out for a lattice of $32 \times 32$ cells and 100 time steps, the following estimates were obtained: $\hat{p}_2 = 0.05$ ($p_2^* = 0.05$) and $\hat{p}_3 = 0.148$ ($p_3^* = 0.15$). This confirms a good agreement with true values and the effectiveness of the method.

**Example 4.5**
Let us try to discover a model describing spatial host-parasite dynamics. The character of this problem is also stochastic, but the process itself is more complicated, as we observe the competition for living space where parasite species need hosts for life and, as for hosts, a meeting with parasites may be tragic.

The basic model of the host-parasite dynamics can be described as a two-dimensional probabilistic CA with periodic boundary conditions [48]. The state set consists of three values representing basic states: *empty* ($s_t(c) = 0$), occupied by a *healthy host* ($s_t(c) = 1$) and occupied by a *parasitised host* ($s_t(c) = 2$). The behaviour of the system is stochastic; the probability of transition to another state is determined by the four nearest neighbours (north, south, east, and west, i.e. the von Neumann neighbourhood).

A healthy host ($s_t(c) = 1$) grows into an empty site ($s_t(c) = 0$) in its neighbourhood with probability of transition $p_G$ (we assume that infected hosts cannot grow) and is infected by a parasitised host in its local neighbourhood with probability of transmission $p_T$. The parasite ($s_t(c) = 2$) is assumed to be fatal, that is, sites occupied by parasitised hosts become empty at the next iteration with probability $p_V$ (it defines the rate of virulence). Table 4.2 summarises the transition probabilities, with $n_1(c)$ and $n_2(c)$ being the number of healthy host

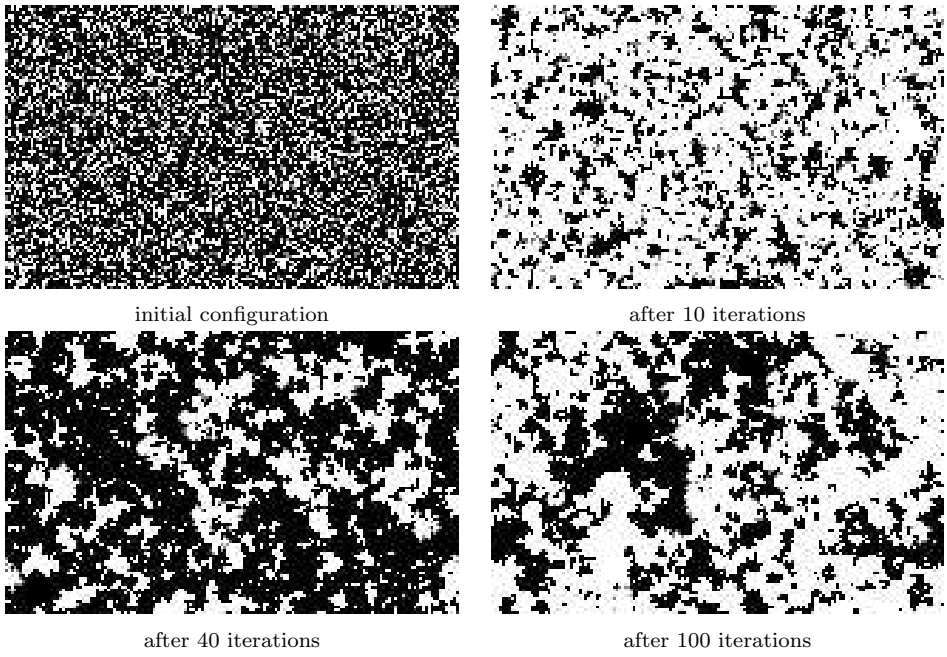



**Fig. 4.5.** An exemplary evolution of the host-parasite model.

and parasite sites in the neighbourhood $N(c)$, respectively:

$$n_1(c) = \sum_{\gamma \in N(c)} \sigma\big(s_t(\gamma), 1\big), \tag{4.16}$$

$$n_2(c) = \sum_{\gamma \in N(c)} \sigma\big(s_t(\gamma), 2\big), \tag{4.17}$$

$$\sigma(s_1, s_2) = \begin{cases} 1 & \text{if } s_1 = s_2, \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

An exemplary dynamics of this model for probability of growth $p_G = 0.05$, probability of transition $p_T = 0.6$ and virulence $p_V = 0.9$ on the two-dimensional lattice of size $160 \times 100$ is shown in Fig. 4.5. The white colour represents vacant sites, the black ones signifies the healthy hosts and the grey ones represents the parasitised hosts. Globally, we can observe the oscillation process of disappearing healthy hosts and, afterwards, an increase in their number.

The idea of estimating the probabilities $p_G$ and $p_T$ will be explained for the former, as the latter is estimated in the same manner. The probability $p_V$ can be estimated based on its frequency interpretation. First note that the coefficient $n_1$ can take values from the set $\{0, \ldots, 4\}$. Denote by $n^1_{1,k}$ the number of observed

transitions from state 0 to state 1 provided that $n_1 = k$, $k = 0, \ldots, 4$. Similarly, write $n^1_{1,k}$ for the number of observations in which cells remain in state 0 while $n_1 = k$, $k = 0, \ldots, 4$.

The likelihood function for these observations takes the following form:

$$L = \prod_{k=0}^{4} (1 - p_G)^{k n^0_{1,k}} \prod_{k=0}^{4} [1 - (1 - p_G)^k]^{n^1_{1,k}} \tag{4.19}$$

Unfortunately, this time it is impossible to obtain a closed-form expression for the estimate of $p_G$ and it has to be calculated numerically. The ARS algorithm turns out to be very convenient in this case.

The data generated by this PCA were used for estimation of the probabilistic transition table. For better quality of these data, the initial configuration was defined randomly.

The results of applying the ARS algorithm to the maximization of the log-likelihood function are as follows (the true values are marked with asterisks and the relative error as $\delta$):

- probability of growth $\hat{p}_G = 0.047834$ ($p^*_G = 0.05$, $\delta_G = 4.33\%$),
- probability of transition $\hat{p}_T = 0.601677$ ($p^*_T = 0.6$, $\delta_T = 0.28\%$),
- virulence $\hat{p}_V = 0.898131$ ($p^*_V = 0.9$, $\delta_V = 0.21\%$).

**Example 4.6**
Let us try to discover two constant parameters of the probabilistic transition function of a generic forest fire model. This model is based on a two-dimensional square lattice where each cell corresponds to 4 ha of forest [21, pp. 109]. The ecological state of each cell is described by its age since the last fire. This age is related to a successional stage of the vegetation, which is not considered in more detail in the model. The model operates on 50 state values $(1, \ldots, 50)$ which correspond to vegetation and one (0) to fire. The following transitions rules were used:

- If a cell represents a vegetation, we check the probability of catching fire from a burning neighbour, according to the formula:

$$p_F(c_{i,j}) = \sum_{\gamma \in N(c_{i,j})} \vartheta_1 + \vartheta_2 s_t(\gamma)^2,$$

  where $\vartheta_1$ and $\vartheta_2$ are constant parameters. If this probability is too small to burn the cell, we increase the vegetation age if it is less than a prescribed maximal value:

$$s_{t+1}(c) = \begin{cases} s_t(c) + 1 & \text{if } s_t(c) < 50, \\ 50 & \text{otherwise.} \end{cases}$$

- If the cell burns, go to the next iteration and start the new vegetation, i.e.

$$s_t(c) = 0 \quad \rightarrow \quad s_{t+1}(c) = 1$$

An exemplary evolution of such a model on the lattice of size 200×200, 100 time steps and the parameters $\vartheta_1 = 0.17$ and $\vartheta_2 = 10^{-5}$ is presented in Fig. 4.6. The same parameters were used to generate the data used for estimation. The following values of estimates were obtained by maximizing the corresponding log-likelihood function using the ARS algorithm (the corresponding true values are marked with asterisks and the relative error as $\delta$): $\hat{\vartheta}_1 = 0.171034$ ($\vartheta_1^* = 0.17$, $\delta_{\vartheta_1} = 0.61\%$) and $\hat{\vartheta}_2 = 9.120501 \times 10^{-5}$ ($\vartheta_2^* = 10^{-5}$, $\delta_{\vartheta_2} = 8.79\%$).

**Example 4.7**
One of the most interesting phenomena for ecological modellers is the spread process of a disease. A good example of such a plague is the spread of rabies observed in the past few decades through Europe from East to West.

In Central Europe, the main agent for the spread of rabies is the red fox (*Vulpes vulpes*). This animal lives in territorial clans of a few adults and come cubs. Rabies is a viral disease transferred by bites. If a member of a clan is infected, it enters into an infectious state and dies after some weeks, but before, it infects the rest of its clan because of the close contact between the animals. In October or November on average four sub-adult foxes leave their home territory and may migrate distances of up to 30 km. In this way, they may recolonize territories left empty because of fox death from rabies, but if they migrate from an infectious territory, they may transfer rabies over long distances.

These few facts are sufficient to construct a preliminary model based on the rectangular lattice of clan territories (by 5 km$^2$, which represents a typical fox clan area) [21, pp. 97]. Each site in the lattice will be empty or occupied by a healthy or infected clan. The dynamics of this model defines the following rules:

- If a site is occupied by an *infected* clan (I), we assume that all foxes are sick and they die (the site becomes empty (E)) with probability $p$:

$$s_{t+1}(c_{i,j}) = \begin{cases} E & \text{with probability } p, \\ I & \text{elsewhere.} \end{cases}$$

- If a site is occupied by a *healthy* clan (H), the probability of infection from neighbouring infected clans $p_T$ is equal to the sum of infection probabilities from each neighbour separately $T$. The probability of infecting clan $c_{i,j}$ by an infected neighbour $c_{k,\ell}$, $T(c_{i,j}, c_{k,\ell})$, depends on the distance between them, in accordance with

$$T(c_{i,j}, c_{k,\ell}) = \begin{cases} 0 & \text{if } c_{k,\ell} \in \{E, H\}, \\ 1 - \exp(-a_T e^{-b_T u(c_{i,j}, c_{k,l})^2}) & \text{for } c_{k,\ell} = I, \end{cases}$$

  where $u(c_{i,j}, c_{k,\ell})$ stands for the distance between $c_{i,j}$ and $c_{k,\ell}$:

$$u(c_{i,j}, c_{k,\ell}) = |i - k| + |j - \ell|,$$

  $a_T$ and $b_T$ being constant globally defined coefficients which define the maxi-

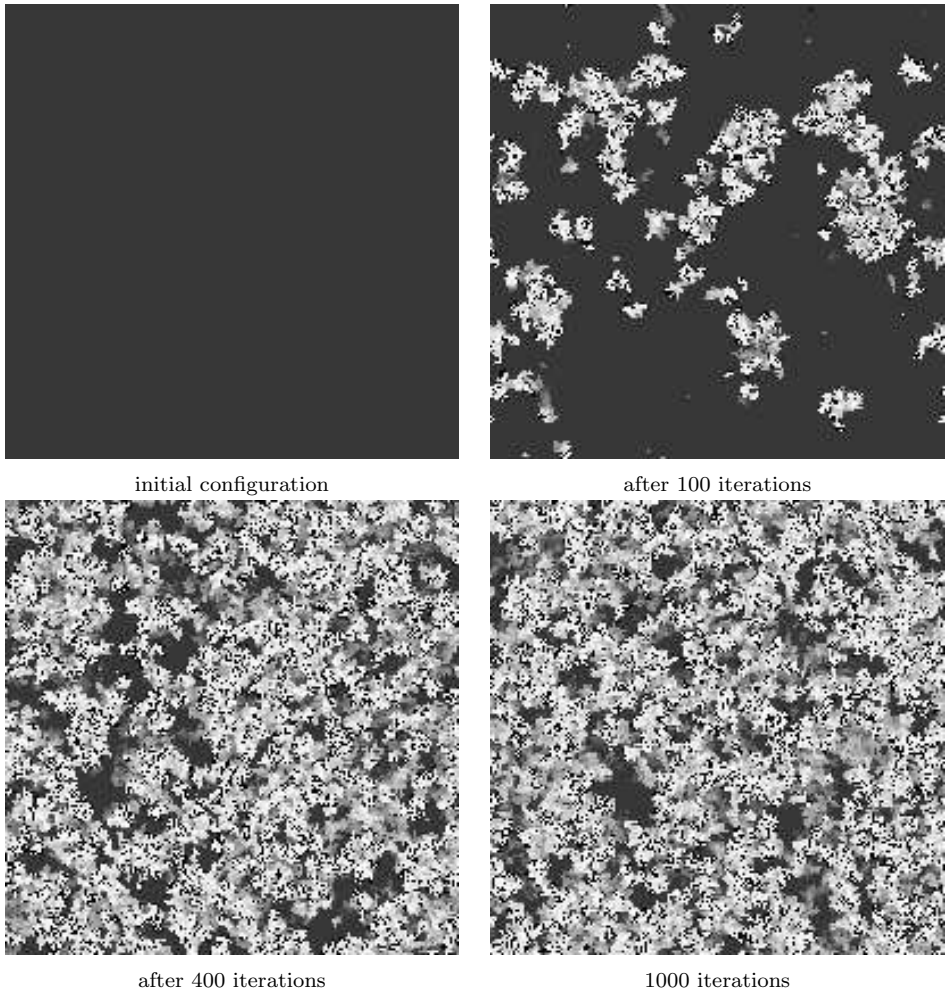



**Fig. 4.6.** Exemplary behaviour of the generic forest fire model.

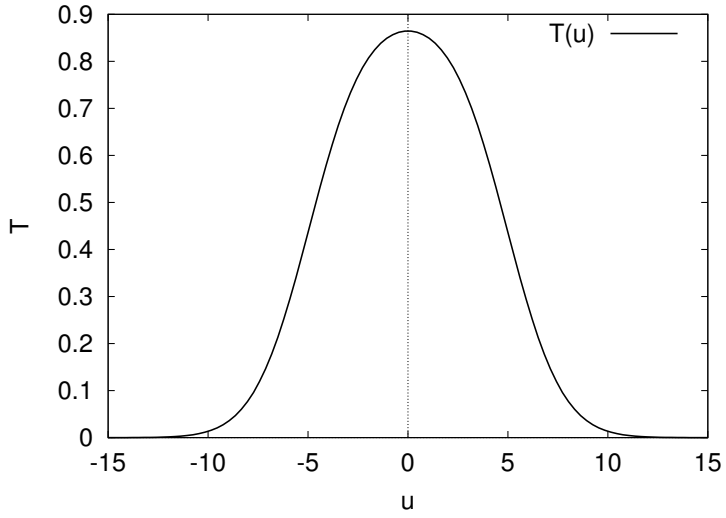



**Fig. 4.7.** Influence of the distance $u$ on the transition probability $T$.

mal range of infection in one jump and the maximal value of the probability, respectively (cf. Fig. 4.7).

Then the probability of infections is to be described as follows:

$$p_T(c) = \sum_{c_i \in N(c)} T(c, c_i).$$

- When a site is *empty* (E), it can be colonized by a healthy (H) clan with the probability proportional to the distance from this clan:

$$U(c, c_i) = \begin{cases} 0 & \text{if site } c_i = \{E, I\}, \\ 1 - \exp(-a_U e^{-b_U u(c,c_i)^2}) & \text{for site } c_i = H, \end{cases}$$

where $u$, $a$ and $b$ are to be interpreted in much the same way as those for the $T(c, c_i)$ function above.

The probability of colonization by healthy neighbours is described as follows:

$$p_U(c) = \sum_{c_i \in N(c)} U(c, c_i)$$

The rules described above can be expressed by the following probabilistic transition

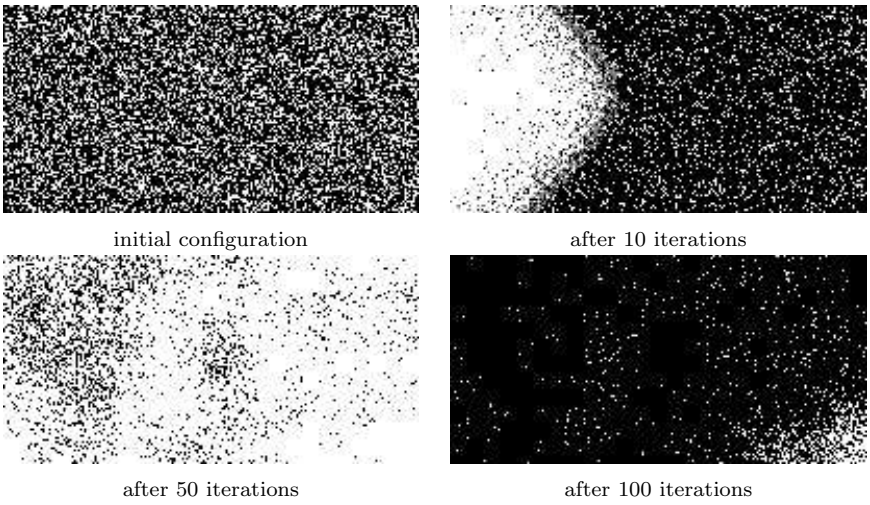



**Fig. 4.8.** Exemplary evolution of PCA modeling for the spatial model of rabies spread. White, black and grey squares represent empty sites, sites occupied by healthy foxes, and those with infected clans, respectively.

table:

| | $s_{t+1}(c)$ | | |
|---|---|---|---|
| $s_t(c)$ | $E$ | $H$ | $I$ |
| $E$ | $1-p_U(c)$ | $p_U(c)$ | $0$ |
| $H$ | $0$ | $1-p_T(c)$ | $p_T(c)$ |
| $I$ | $p$ | $0$ | $1-p$ |

An example evolution of the PCA model using these rules for the lattice of size 200×100, 100 time steps and the following values of parameters $p = 0.4$, $a_T = 2$, $b_T = 0.05$, $a_U = 4 \cdot 10^{-5}$ and $b_U = 10^{-6}$ is shown in Fig. 4.8. The same model was used to generate simulation data.

After the estimation process maximizing the likelihood function using the ARS algorithm for 5% of disturbed data (by changing the states of selected cells by randomly drawn state values), the following values were obtained: $\hat{p} = 0.398432$, $\hat{a}_T = 2.101034$, $\hat{b}_T = 0.051221$, $\hat{a}_U = 0.000041$ and $\hat{b}_U = 1.045321 \cdot 10^{-6}$ (the relative errors are $\delta_p = 0.39\%$, $\delta_{a_T} = 5.05\%$, $\delta_{b_T} = 2.44\%$, $\delta_{a_U} = 2.5\%$ and $\delta_{b_U} = 4.53\%$). A good agreement of the estimated and actual parameter values is thus observed in spite of the disturbances.

## 4.4. Conclusions

In this chapter, the problem of parameter estimation for both deterministic and stochastic CA's has been considered and some computational techniques of

moderate complexity have been proposed and tested for its solution.

In the case of parameter estimation of deterministic CA models several fit-to-data criteria have been tested. In all situations, noise-corrupted data were considered, which constitutes a considerable extension compared with the classical monograph by Adamatzky [1]. This fact qualifies the proposed methods as convenient approaches to parameter estimation of real spatio-temporal systems, where many of measurements are disturbed by noise. It turns out that the estimates may be non-unique, which necessitates characterizing the posterior feasible set for the estimates e.g. by a cloud of points, as is a routine in bounded-error set estimation. Of course, the attained performances are not extraordinary as far as the number of error function evaluations is concerned, but this is a common problem in global discrete optimization and it goes without saying that much can still be done in this connection.

In turn, satisfactory results obtained from application of the maximum-likelihood approach to estimation of parameters of probabilistic transition tables offer new possibilities for developing probabilistic models. It is specially useful for ecological models where, instead of a detailed description of the underlying mechanisms for very complex systems, probabilistic rules are used [68, 69, 67]. The usefulness of the proposed approach is additionally increased by the robustness against disturbances in the observation dataset. In the presented examples, disturbances in 5% of samples did not affect the quality of estimation, and the results were still quite satisfactory.

# Chapter 5

# Structural identification of cellular automata

## 5.1. Introduction

From Chapter 4 we know how to determine parameters of a given CA transition function $f$. But in many cases only the behaviour of the studied phenomenon is known, the neighbourhood and structure of the mapping $f$ describing the automaton being unknown. The difficulty in designing CA's which have to exhibit a specific behaviour or to perform a particular task has severely limited their applications. Automating the design (programming) process would greatly enhance the viability of CA's [73].

One of the promising approaches to alleviate the above-mentioned predicament consists in exploiting some ideas from genetic algorithms. Those concepts have already been employed successfully for both uniform and non-uniform CA's, see e.g. [37] and the excellent monograph [73] with many references given therein. A similar approach proposed in [93] proved the usefulness of genetic algorithms in designing CA transition functions even when the data used for estimation are noise-corrupted. On the other hand, the idea of making use of genetic programming seems to be more appropriate for constructing CA programs and it was already suggested by Koza [51], but the presented examples considered only some elementary problems. All those papers show how to discover a transition function for a given neighbourhood. Of course, we can freely choose its definition, but in many cases the user may make a bad choice many times before satisfactory results are obtained. A good example of this situation represents a model of vegetation dynamics for fens (cf. Sec. 5.3.3), where one part of the neighbourhood is of the von Neumann type and the second is not necesssarily so. In this case, the known approaches may inidicate a relationship between the chosen neighbourhood and system behaviour which do not exist in reality and may result in a too long time of calculations and excessively high errors.

If we introduce an appropriate fitness (target) function for the GP algorithm, we can force it to found both the neighbourhood and the transition function operating on it. Some results have already been published [42, 43, 91, 52] and they consider only the general behaviour of binary CA's. In this chapter, some results are given concerning the application of genetic programming to the

design of CA's modelling the behaviour given as measured data or coming from an unknown system (model). Both of the presented examples try to discover a model emulating the behaviour of a given probabilistic cellular automaton (PCA) model. The stochastic character of the model introduces an additional difficulty to the intuitive detection of a proper neighbourhood and a transition function, but even this proposed approach has made it possible to find quite satisfactory results.

## 5.2. Genetic algorithms and genetic programming

In 1975 John Holland [36] described how the evolutionary process in nature can be applied to artificial systems using the genetic algorithm operating on fixed-length character strings. Holland demonstrated that a population of fixed-length character strings (each representing a proposed solution to a problem) can be genetically bred using the Darwinian operation of fitness proportionate reproduction and the genetic operation of recombination. The *recombination* operation combines parts of two chromosome-like fixed length character strings, each selected on the basis of their fitness, to produce new offspring strings. Holland established, among other things, that the genetic algorithm is a near-optimal approach to adaptation in that it maximises the expected overall average pay-off when the adaptive process is viewed as a multi-armed slot machine problem requiring an optimal allocation of future trials given currently available information. Genetic algorithms have proven successful at searching nonlinear multidimensional spaces in order to solve, or approximately solve, a wide variety of problems [5].

*Genetic programming* [51] (GP), like the conventional genetic algorithm, is a domain-independent method. It extends the genetic model of learning to the space of programs. It is a major variation of genetic algorithms in which the evolving individuals are themselves computer programs instead of fixed-length strings from a limited alphabet of symbols. GP is a form of program induction that allows to automatically discover programs that solve or approximately solve a given task.

Individual programs in GP might be expressed in principle in any current programming language. However, the syntax of most languages is such that GP operators would create a large percentage of syntactically incorrect programs. For this reason, Koza chose a syntax in prefix form analogous to LISP and a restricted language with an appropriate number of variables, constants and operators defined to fit the problem to be solved. In this way, syntax constraints are respected and the program search space is limited. The restricted language is formed by a user-defined primitive function set and a terminal set. The functions chosen are those that are *a priori* believed to be useful for the problem at hand, and the terminals are usually either variables or constants. In addition, each function in the function set must be able to accept as arguments any other function return value and any data type in the terminal set, a property that is called *syntactic closure*. Thus the space of possible programs is constituted by the set of all possible compositions of functions that can be recursively formed from the elements of the function and terminal sets.

It is important to note that programs are thus represented as trees with

ordered branches in which the internal nodes are functions and the leaves are terminals of the problem. Evolution in GP is similar to GA's, except that different individual representations and genetic operators are used. Once suitable functions and terminals have been determined for the problem at hand, an initial random population of trees (programs) is constructed. From there on the population evolves as with a GA where fitness is assigned after actual execution of the program (individual) and with genetic operators adapted to the tree representation.

GP proceeds by genetically breeding populations of compositions of primitive functions and terminals (i.e. computer programs) to solve problems by executing the following three steps:

1. Generate an initial population of random computer programs composed of the primitive functions and terminals of the problem.

2. Iteratively perform the following sub-steps until the termination criterion has been satisfied:

   (a) Execute each program in the population and assign it a fitness value according to how well it solves the problem.

   (b) Create a new population of programs by applying the three primary operations below. The operations are applied to programs in the the population selected with a probability based on the fitness (i.e. the fitter the program, the more likely it is to be selected).

      i. *Reproduction*: Copy existing programs to the new population.

      ii. *Crossover*: Create two new offspring programs for the new population by genetically recombining randomly chosen parts of two existing programs. The genetic crossover (sexual recombination) operation operates on two parental computer programs and produces two offspring programs using parts of each parent.

      iii. *Mutation*: Modify small parts of programs so as to make changes in the fitness.

3. A single best computer program in the population produced during the run is designated as the result of the run of genetic programming. This result may be a solution (or an approximate solution) to the problem.

GP is particularly useful for program discovery, i.e. the induction of programs that correctly solve a given problem with the assumption that the form of the program is unknown and that only the desired behaviour is given, e.g. by specifying input-output relations. GP has been successfully applied to a wide variety of problems from many fields [5]. It has been empirically shown to be quite a powerful automatic or semi-automatic program-induction and machine learning methodology.

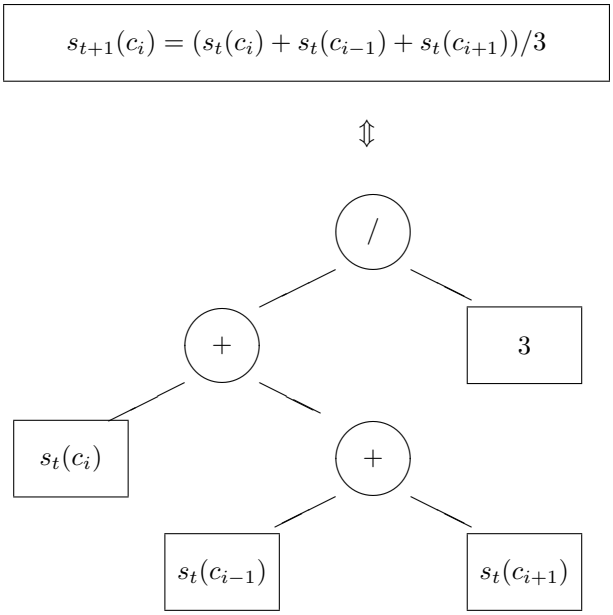



**Fig. 5.1.** An exemplary transition function for a one-dimensional CA and its tree representation.

## 5.3. Discovering state transition rules via genetic programming

Since GP operates on programs, it seems to be ideal for discovering CA transition functions which are a special kind of programs. If we add appropriate information about possible neighbourhood cells which can be used in the searching process, GP is in a position to sort them out and finally return a best function consisting of neighbours chosen as essential with respect to the fitness criterion used. The neighbourhood selected in this manner is recognised as the neighbourhood definition of the selected model. This approach introduces additional values to set up the components used to build up the desired function or program. It is natural for combinatorial problems that a small extension of the parameter set introduces a significant difficulty in discovering the best result. This is especially important in the GP case, where in contrast to the classical approaches to combinatorial problems operating on fixed-size solutions, solutions with varying sizes are generated.

In this section two examples of using genetic programming are used to illustrate the discovering of state transition functions. Both the examples base on the data obtained from probabilistic two-dimensional CA models.

### 5.3.1. Adapting the GP algorithm for identifying CA structures

Consider the transition function defined as follows:

$$s_{t+1}(c) = f(s_t(N(c))) \tag{5.1}$$

and assume that it can be interpreted as a tree whose each node represents one of operators and each leaf can be seen as the state of a selected neighbouring cell in the CA lattice (cf. Fig. 5.1). This method is frequently used for evaluating complex functions on computers: functions and their parameters are stored on a stack as a sequence and later they are taken and evaluated. The function shown in Fig. 5.1 has then the following representation on the stack:

| | | | | | | |
|---|---|---|---|---|---|---|
| / | + | $s_t(c_i)$ | + | $s_t(c_{i-1})$ | $s_t(c_{i+1})$ | 3 |

When this function is to be evaluated, the first component is taken (in this case it is the *division* operator /) because it has two parameters, the first one being taken from the stack (this is the *sum* operator +). Because this operator also has two parameters, the first one is taken (the value of $s_t(c_i)$) and then the second (the sum operator +). In this manner, all the parameters of functions are collected and then the results of the individual operators are evaluated. Finally, the result of the entire function is obtained.

Such a representation offers a possibility of employing GP as a tool for the design of local state transition functions. For a proper implementation of the delineated technique, a fitness criterion should be defined to measure the discrepancy between the observed CA evolution and the desired behaviour. A classical manner to form it is based on the calculation of the modelling error. For a two-dimensional rectangular lattice $\mathcal{L}$ of size $Q \times R$, the most commonly used form looks as follows [51, 42, 52]:

$$J(f,N) = \frac{1}{TQR} \sum_{t=0}^{T-1} \sum_{i=1}^{Q} \sum_{j=1}^{R} \delta\Big(s_{t+1}(c_{i,j}), f\big(s_t(N(c_{i,j}))\big)\Big) \tag{5.2}$$

where $\delta$ represent the error value and can be defined as follows:

$$\delta(s_1, s_2) = \begin{cases} 1 & \text{for } s_1 = s_2, \\ 0 & \text{otherwise.} \end{cases}$$

For some cases it is necessary to emphasise high values of the error so as to distinguish better large deviations of the model from the data. In such a case the criterion consisting of the sum of squared errors could be used (i.e. the classical least-squares criterion with (5.2) for $\delta(s_1, s_2) = (s_1 - s_2)^2$).

If the CA is stochastic in nature, the expectation of the criterion (5.2), i.e. $E\{J(f,N)\}$, should be calculated in lieu of $J(f,N)$. But then the Monte Carlo method can be employed [25, 59] for approximately solving the resulting stochastic optimization problem. It reduces to generating independent and identically distributed initial CA configurations, and then running the system to be

estimated many times, starting from those configurations. This makes it possible to replace the expectation by the sample mean obtained from these runs.

The above-mentioned performance index guarantees finding a function which reflects the observed behaviour satisfactorily in most cases. Unfortunately, it does not limit the size of its tree representation. The depth of the trees may in principle increase without limits under the influence of crossover, a phenomenon that goes under the name of 'bloating'. The increase in the size is often accompanied by a stagnant population fitness. Consequently, a good GP implementation must have a parameter which prevents tress from becoming too deep, thus filling all the available memory and requiring longer evaluation times. To further avoid bloating, a common approach is to introduce a size-penalty term into the fitness expression, possibly in a self-adapting way. As a result, in order to remedy the above inconvenience, we modify the fitness function as follows:

$$J'(f, N) = J(f, N) - W(f) \tag{5.3}$$

where $W(f)$ is a function proportional to the size of the tree representation of $f$ (e.g. the number of all nodes and leafs). Accordingly, we obtain a function which prevents the algorithm from the excessive increase in the size of the sought function. The additional component $W(f)$ causes that for the same fitness values simple functions are preferred.

At this moment, if the fitness criterion is defined, the evolution process can be described. The structure of the algorithm used to obtain the CA transition function can be written as follows:

1. Initiation
   (a) Choose:
      - population size $n_p$,
      - initial lengths $l(f_{i,0})$ of the functions $f_{i,0}$ in the initial generation $G_0$, $i = 1, \ldots, n_p$,
      - mutation probability $P_{\mathrm{mut}}$,
      - maximum number of iterations $n_{\max}$,
      - desired fitness value $J_{\max}$,
      - number of best functions copied to the next generation $n_b$,
      - number of iterations before calculation of the fitness $n_{\mathrm{fit}}$ (if we evaluate it at each iteration, then $n_{\mathrm{fit}} = 1$),
      - size of tested CA lattices $S_{\mathcal{L}}$.
   (b) Clear the iteration counter ($n = 0$).
   (c) Define the function set $\mathbb{F}$ (for a binary CA it can be composed of elementary logical operators, e.g. $\mathbb{F} = \{\mathrm{AND}, \mathrm{OR}, \mathrm{XOR}, \mathrm{NOT}\}$).
   (d) Set the terminal set (leafs in the tree) $\mathbb{T}$ (for a binary, two dimensional CA it can look like $\mathbb{T} = \{s(c_{i,j}), s(c_{i-1,j-1}), s(c_{i,j-1}), \ldots, s(c_{i+1,j+1})\}$).

(e) Draw at random an initial population of CA transition functions $f_{\ell,0}$ (i.e. trees) of size $l(f_{\ell,0})$:

$$G_0 = \{f_{1,0}, f_{2,0}, \ldots, f_{n_p,0}\},$$

where $f_{\ell,n}$ means the $\ell$-th function of generation $n$.

(f) Draw at random $N$ initial CA configurations

$$\mathbb{L} = \{C_{1,0}, \ldots, C_{N,0}\}$$

2. Calculate the fitness:

   (a) Perform $n_{\text{fit}}$ iterations for each of the initial configurations in $\mathbb{L}$.

   (b) Calculate the mean fitness $\bar{J}$ for each transition function from the current generation $G_n$ for all initial conditions belonging to $\mathbb{L}$.

   If $\bar{J} = J_{\max}$, then STOP.

3. Perform selection. In this way, $n_b$ functions are copied from the current generation $G_n$ to the new generation $G_{n+1}$, with the probability proportional to their fitness $J(f, N)$.

4. Perform crossover. The remaining $n_p - n_b$ functions in the new generation $G_{n+1}$ are constructed as new functions which are the result of crossover of randomly selected two functions from the mother generation $G_n$.

   (a) Select two functions $f_{k,n}$ and $f_{\ell,n}$ from population $G_n$ with probability proportional to their fitness ($p_k \propto J(f_{k,n})$ and $p_\ell \propto J(f_{\ell,n})$).

   (b) Select one of the nodes (i.e. a sub-function $f^a_{k,n}$) from function $f_{k,n}$ and one ($f^b_{\ell,n}$) from function $f_{\ell,n}$.

   (c) Create a new function as a copy of $f_{k,n}$ where part $f^a_{k,n}$ is replaced by $f^b_{\ell,n}$ (cf. Fig. 5.2).

5. Perform mutation. For each tree in the current population, remove with probability $P_{\text{mut}}$ a sub-tree at a selected node and replace it with a randomly generated tree.

6. If $n = n_{\max} - 1$ then STOP, otherwise set $n = n + 1$ and go to *Step 2*.

The above algorithm starts with a population of randomly generated state transition functions (*Step 2*). Once an initial population has been created, the algorithm opens the main loop. *Step 2* is to calculate the fitness of each transition function. In *Step 3* a *selection process* is applied to create a new intermediate population of $n_p$ "parent functions", i.e. $n_p$ independent extractions of individuals from the old population are performed, where the probability for each individual of being extracted is proportional to its fitness. Once the intermediate population has been extracted, the new generation of functions will be produced through

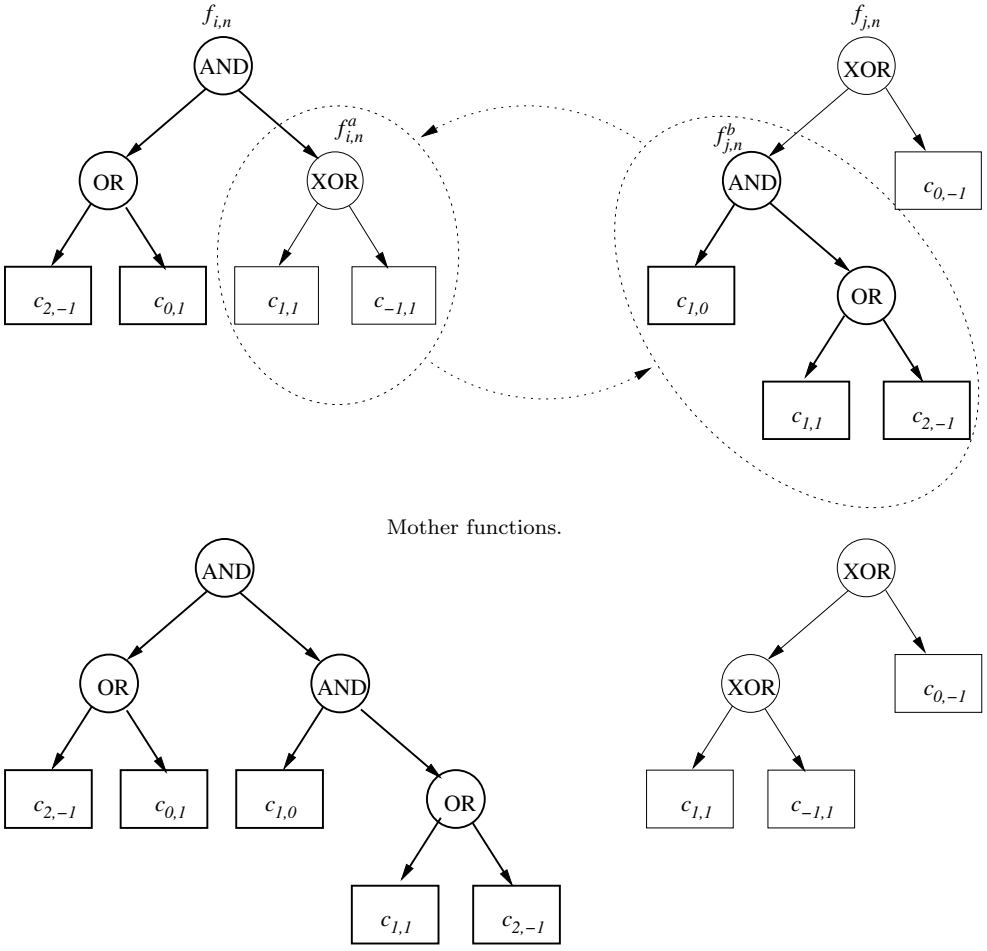



**Fig. 5.2.** Process of creating new functions by crossover of two mother functions.

the application of crossover and mutation. To apply crossover (*Step 4*), couples are formed with the same entries from all parent functions. Then, with some probability $P_{\text{cross}}$, each couple undergoes crossover: for the same entries (trees) in two functions, random crossover points are selected and then the corresponding sub-trees are exchanged. Mutation (*Step 5*) is implemented by a random procedure that, with a certain probability $P_{\text{mut}}$, removes a sub-tree at a selected point and replaces it with a randomly generated tree. *Steps 2–5* are repeated until either the maximum number of iterations is exceeded or some of the state transition function has achieved a perfect fitness $J_{\max}$.

### 5.3.2. Synthesis of CA's for modelling desired behaviours

For the problems described below the value of the mutation probability $P_{\text{mut}} = 0.006$ was used. It should be underlined that the GP algorithm described in Section 5.2 is supplemented by inversion. This is a method which changes randomly the order of the arguments of a selected sub-function. To apply the inversion on a function $f_{k,n}$, two sub-functions should be selected (e.g. $f^a_{k,n}$ and $f^b_{k,n}$), then their positions are exchanged in the main function $f_{k,n}$. For the problems described below, the inversion probability is $P_{\text{inv}} = 0.003$.

### 5.3.3. Discovering the model of a plant population in fens

The approach described above can be used to find a CA model on the basis of observed data. These data can be measured or generated by a model. In what follows, the latter approach was chosen. In order to generate data, the model proposed by Weimar [88] was used.

In a fen the most significant environmental influences which can be managed are floodings (and cutting if we consider a human intervention). Based on this fact, a simple probabilistic model of a plant population in fens was proposed in [88]. This model concentrates on two plant species which are used as representatives of groups of species. They are *Water Sweet-grass* (Glyceria Maxima – it proliferates in sufficiently wet conditions and dies off in excessively dry conditions) and *dandelion* ("Lions tooth" – it proliferates by airborne seeds when conditions are dry enough and dies off if the environment is too wet).

In the case of the extreme discretisation into two values for the above-mentioned vegetation, we can easily define probabilistic rules operating on a state vector. This vector includes two values representing the vegetation levels of both the modelled species (we suppose that both the species can occupy the same terrain) denoted by $s^G_t(c)$ for Water-Sweet grass and $s^L_t(c)$ for dandelion. We construct a CA in two dimensions on a square lattice. There are two sets of rules, one for wet conditions, and one for dry conditions. Since the natural growth cycle is one year, we model one year in one CA step. The type of conditions for consecutive years is given by the sequence $\{e_t\}$, where $e_t = 0$ and $e_t = 1$ stand for dry and wet conditions, respectively.

The first set of rules is for the *Water Sweet-grass* and it says that for dry environmental conditions ($e_t = 0$) the vegetation (which corresponds to $s^G_t(c) = 1$)

**Tab. 5.1.** Transition table for the model of a plant population in fens.

| | | Water Sweet-grass | | Dandelion | |
|---|---|---|---|---|---|
| $(s_t^G(c), s_t^L(c))$ | $e_t$ | $P(s_{t+1}^G(c)=0)$ | $P(s_{t+1}^G(c)=1)$ | $P(s_{t+1}^L(c)=0)$ | $P(s_{t+1}^L(c)=1)$ |
| (0,0) | 0 | 1 | 0 | $p_3$ | $1-p_3$ |
| (0,0) | 1 | $\varrho$ | $1-\varrho$ | 1 | 0 |
| (0,1) | 0 | 1 | 0 | 0 | 1 |
| (0,1) | 1 | $\varrho$ | $1-\varrho$ | $p_2$ | $1-p_2$ |
| (1,0) | 0 | $p_1$ | $1-p_1$ | $p_3$ | $1-p_3$ |
| (1,0) | 1 | 0 | 1 | 1 | 0 |
| (1,1) | 0 | $p_1$ | $1-p_1$ | 0 | 1 |
| (1,1) | 1 | 0 | 1 | $p_2$ | $1-p_2$ |

dies (i.e. $s_{t+1}^G(c)=0$) with probability $p_1$ or it remains the same elsewhere. For a wet vegetation case, i.e. $e_t=1$, it appears, i.e. $s_{t+1}(c)=1$, if at least one neighbour possesses this vegetation, and remains the same for all other situations. For the *Dandelion* case and wet conditions ($e_t=1$), the vegetation ($s_t(c)=1$) dies off (i.e. $s_{t+1}=0$) with probability $p_2$ or remains the same elsewhere. For the dry vegetation case ($e_t=0$), an empty cell ($s_t(c)=0$) starts having the vegetation ($s_{t+1}^L(c)=1$) with probability $p_3$, and the states remain the same in all other situations. This model is described by the probabilistic transition table shown in Tab. 5.1, where

$$\varrho(c)=\begin{cases}0 & \text{if } \sum\limits_{\gamma\in N(c)} s_t^G(\gamma)>0,\\ 1 & \text{otherwise.}\end{cases}$$

An exemplary dynamics of the model for an average wetness of 0.5 and $p_1=0.3$, $p_2=0.4$ and $p_3=0.05$ on the two-dimensional lattice of size 30×30 is shown in Fig. 5.3. The white colour represents vacant sites and the black one signifies the cells occupied by vegetation. The pictures are grouped in four sequences by three pictures to show vegetation of both the species in all conditions.

Experiments with GP were performed for the square lattice $\mathcal{L}$ of size 30×30 with periodical boundary conditions. Because this is a binary CA, logical operators can be used to define the function set

$$\mathbb{F}=\{\text{NOT , AND , OR , XOR , RND}\}$$

(other operators can be obtained as a combination of this basic set), the component RND representing a function which generates 0 or 1 with equal probabilities 0.5. The set of terminals $\mathbb{T}$ was composed of two possible state values (0 and 1) and possible neighbour positions in the neighbourhood of the maximum radius of 3 from the central cell in the von Neumann sense. The size of the drawn functions in the initial generation $G_0$ of size 300 is adjusted as 20. It is assumed that the fitness

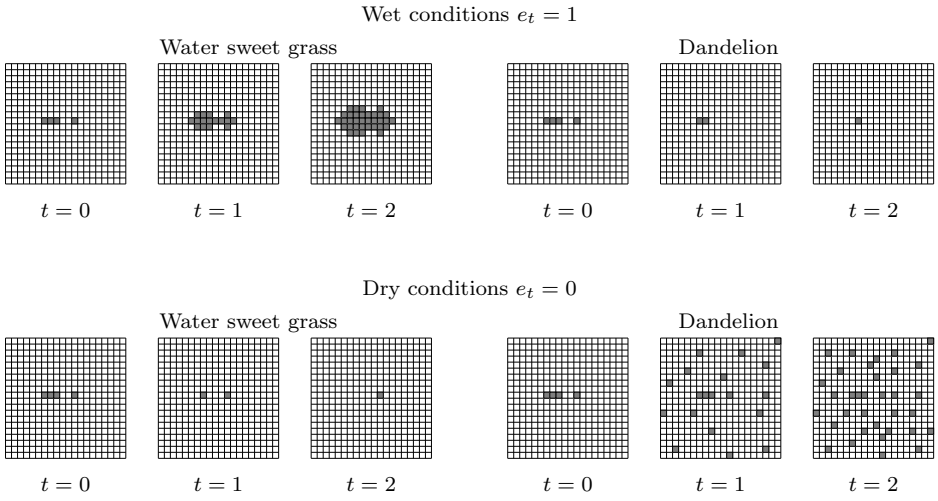



**Fig. 5.3.** An exemplary evolution of plant populations in fens.

function $J$ is the mean value of the performance criterion for 20 initial conditions generated at random every time for each function. As for the probabilities, they were set as $p_1 = 0.5$, $p_2 = 0.5$ and $p_3 = 0.25$.

Three criteria were used so as to evaluate the mean fitness function in GP:

- *Efficiency:*

$$J_E(f, N) = \frac{1}{TQR} \sum_{t=0}^{T-1} \sum_{i=1}^{Q} \sum_{j=1}^{R} \delta\Big(s_{t+1}(c_{i,j}), f\big(s_t(N(c_{i,j}))\big)\Big) - W(f) \tag{5.4}$$

  where

$$\delta(s_1, s_2) = \begin{cases} 1 & \text{for } s_1^G = s_2^G \text{ and } s_1^L = s_2^L, \\ 0 & \text{otherwise,} \end{cases}$$

  $T$ is the length of the discrete time horizon, $Q$ and $R$ represent respectively the width and height of the lattice, and $W(f)$ is the sum of all nodes and leafs in the function $f$ divided by 4. The values of this criterion are between zero, which means the total lack of fitness, and 1, which means the perfect adaptation.

- *Squared efficiency:*

$$J_S(f, N) = J_E(f, N)^2. \tag{5.5}$$

  Similarly to the previous criterion, it evaluates between 0 and 1, but the fitness differences are emphasised.

**Tab. 5.2.** The results obtained after running the GP algorithm for efficiency ($J_E$), squared efficiency ($J_S$) and Median of efficiencies ($J_M$).

| Criterion | Best fitness | Average time [h] |
|---|---|---|
| $J_E$ | 0.922598 | 9.5 |
| $J_S$ | 0.924683 | 8.5 |
| $J_M$ | 0.919802 | 9 |

- *Median of efficiencies.* This notion is well-known in statistics, and constitutes an alternative to the mean as the "centre" of the data. The efficiences are sorted and the median is the value such that half of the efficiences lie above and half of the efficiences below it.

The results obtained after running the GP algorithm for each of these criteria are shown in the Tab. 5.2.

In all cases the desired fitness value was $J_{\max} = 0.95$, but unfortunately, the algorithm did not reach this value and stopped after the assumed maximum number of iterations $n_{\max} = 10\,000$. In this example the criterion of squared efficiency led to the fastest convergence.

For better understanding how GP is capable of discovering the model, let us describe it as follows:

$$
\begin{aligned}
s^G_{t+1}(c_{i,j}) \;=\;& s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ OR} \\
& \text{NOT } s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ AND RND AND} \\
& \quad (s^G_t(c_{i,j+1}) \text{ OR } s^G_t(c_{i,j-1}) \text{ OR } s^G_t(c_{i-1,j}) \text{ OR } s^G_t(c_{i+1,j})) \text{ OR} \\
& s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \\
s^L_{t+1}(c_{i,j}) \;=\;& \text{NOT } s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \text{ AND (RND AND RND) OR} \\
& s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \text{ OR} \\
& s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ AND RND.}
\end{aligned}
$$

The estimated function with the best fitness of the type $J_S$ is as follows:

$$\begin{aligned} s^G_{t+1}(c_{i,j}) &= s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ AND (RND OR RND) OR} \\ &\quad \text{NOT } s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ AND RND AND} \\ &\quad \big(s^G_t(c_{i-1,j}) \text{ OR } s^G_t(c_{i+1,j}) \text{ OR } s^G_t(c_{i,j-1})\big) \text{ AND } s^G_t(c_{i,j+1}) \text{ OR} \\ &\quad s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \\ s^L_{t+1}(c_{i,j}) &= \text{NOT } s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \text{ AND} \\ &\quad \text{(RND OR RND AND RND) OR} \\ &\quad s^G_t(c_{i,j}) \text{ AND NOT } s^L_t(c_{i,j}) \text{ AND RND AND RND OR} \\ &\quad s^G_t(c_{i,j}) \text{ AND } s^L_t(c_{i,j}) \text{ AND RND.} \end{aligned}$$

### 5.3.4. Estimation of the parasite transmission model

Let consider the structural identification problem for the model described in Example 4.5 on p. 96. Experiments with GP were performed for the lattice $\mathcal{L}$ of size 30×30 with periodical boundary conditions. For this case the GP algorithm can be used in order to find the form of probability functions $f_{i,j}$ of transitions from state $i$ to $j$. Because the problem is not binary, logical operators in the functions set $\mathbb{F}$ should be replaced by elementary arithmetic operators completed by constants and information about the neighbourhood configuration. Finally, we get the following functions set:

$$\mathbb{F} = \{+, -, *, /, \text{pow}, \text{abs}\},$$

where 'pow' stands for the power function and 'abs' returns the absolute value. The terminal set was

$$\mathbb{T} = \{1, \ldots, 10, c_{i-2,j-2}, \ldots, c_{i+2,j+2}\}.$$

So as to ensure the compatibility with probability values, it is defined that the results of the evaluation of the resultant functions are truncated to values from the interval $[0, 1]$. The size of the drawn functions in the initial generation $G_0$ of size 300 was selected as 20. It is assumed that the fitness function $J$ is the mean value of evaluating the criterion for each 20 initial conditions drawn at random every time for each function.

The results obtained after running GP for criteria from Section 5.3.3 and $p_G = 0.05$, $p_T = 0.6$ and $p_V = 0.9$ are shown in the Tab. 5.3.

In experiments the following structure of $\delta$ was used:

$$\delta(s_1, s_2) = 2 - |s_1 - s_2|.$$

In all the cases the desired fitness value was $J_{\max} = 0.95$. Unfortunately, the algorithm did not reach this value, and it stopped after do maximum number of iterations $n_{\max} = 10\,000$. In this case too, the criterion of squared efficiency

**Tab. 5.3.** The results obtained after running the GP algorithm for efficiency ($J_E$), squared efficiency ($J_S$), Median of efficiencies ($J_M$) and $p_G = 0.05$, $p_T = 0.6$ and $p_V = 0.9$.

| Criterion type | Best fitness | Average time [h] |
|---|---|---|
| $J_E$ | 0.905473 | 13 |
| $J_S$ | 0.921002 | 12 |
| $J_M$ | 0.903523 | 12.5 |

caused the fastest convergence. The resulting table of probabilities, obtained for the criterion $J_S$ is as follows:

| $s_t(c)$ | $\hat{P}(s_{t+1}(c)=0)$ | $\hat{P}(s_{t+1}(c)=1)$ | $\hat{P}(s_{t+1}(c)=2)$ |
|---|---|---|---|
| 0 | 9/10 | $(1/10)^{\varepsilon_1}$ | 0 |
| 1 | 0 | $(4/10)^{\varepsilon_2}$ | $(8/10)^{\varepsilon_3}$ |
| 2 | 9/10 | 0 | 0 |

where

$$\varepsilon_1 = |s_t(c_{i-1,j}) - 1| + |s_t(c_{i,j-1}) - 1| + |s_t(c_{i+1,j+1}) - 1| + |s_t(c_{i,j+1}) - 1|$$

$$\varepsilon_2 = |s_t(c_{i-1,j}) - 2| + |s_t(c_{i,j-1}) - 2| + |s_t(c_{i+1,j}) - 2| + |s_t(c_{i,j+1}) - 2|$$

$$\varepsilon_3 = |s_t(c_{i-1,j}) - 2| + |s_t(c_{i,j-1}) - 2| + |s_t(c_{i,j+1}) - 2| + 1$$

Please note that the value of the power $\varepsilon_2$ is just the same as in the original function.

## 5.4. Conclusion

In this chapter the usefulness of genetic programming in finding state transition rules of CA's was tested and attention was mainly focused on stochastic CA's, where some concepts of the Monte Carlo method were employed for justification of the choice of the performance index being a mean of the results for many model runs. As can be seen from the presented simulation results, genetic programming constitutes a very useful tool to discover the state transition function in situations for which the size of the transition rules is limited and we can define an appropriate fitness function. Moreover, the choice of an optimal neighbourhood for the lattices was also considered, which has not been considered in the literature so far. In most cases, the cells adjacent to the central cell have been selected as the 'optimal' ones, which confirms in a sense the reasonableness of introducing local rules in general CA's defined on adjacent cells.

The results obtained allow us to classify this approach as useful for build-

ing ecological models on the basis of observations from satellites or airplanes, taken in conjunction with some advanced image processing (based on mathematical morphology), as in those situations the presence of considerable noise should be included into the formulation.

Additionally, the influence of the applied fitness function on the search process was tested for general cases. The efficiency criterion is found to be appropriate. For both the examples discussed here, an adequate function scale was crucial for founding solutions. The criterion of squared efficiency seems to be the best choice, as it produced better results.

# Chapter 6

## Conclusions and future research directions

As pointed out in [21], ecology needs new ideas and methods to deal with dynamics of processes in a spatial setting. These models help in developing intuition about how ecological systems behave and show repeatedly how new, unexpected phenomena emerge when a spatial structure is introduced.

Since the pioneering work of John von Neumann in the 1950's, CA's have been largely employed as a modelling class to approximate nonlinear discrete and continuous dynamical systems in a wide range of applications. However, the inverse problem of determining a CA that satisfies general sets of prespecified constraints has received relatively little attention. One of the most essential problems in this case is the identification of CA's, i.e. how to learn the underlying rule that governs the local behaviour of cells from temporal slices of the global evolution of the spatio-temporal pattern. The existing solutions are concerned with Boolean automata and do not address the general problem of determining the CA rules from an observed, possibly noisy, complex multidimensional pattern. A potential solution would offer an important step forward in the modelling of spatially extended systems that arise in diverse fields such as pattern formation, fluid mixing, brain imaging and in data compression problems.

Bearing this in mind, the original goal of the research reported in this thesis was simply to develop computationally efficient methods to solve practical design problems for a wide class of CA's. In the process of executing this task, some known methods have been employed and several new algorithms have been constructed. The following is a concise summary of the contributions provided by this work to the state-of-the-art in identification of CA models:

- Systematizes characteristic features of the problem and analyses the existing approaches.

- Develops an effective hybrid method for modelling forest succession models. This scheme is based on coupling a completely discrete CA model with a gap model described by systems of ordinary differential equations, which makes it possible to get benefit from the advantages offered by both the approaches when applied separately. This technique was validated on real data.

- Provides original CA models of spreading phenomena for the Eden and single percolation cluster models. Spreadability constitutes the original concept which has been widely developed since seminal works of El Jai and his co-workers [44, 45]. The notion of spray control is then introduced for spreadable systems and examples of its use are given.

- Formulates and solves the problem of parameter estimation of CA models. The framework introduced here is different from the settings considered in the literature so far in that noisy-corrupted data are assumed and stochastic CA's are also addressed. The proposed effective methods of parameter estimation adapt some schemes from general optimization and parameter estimation methods, which paves the way for finding links with some existing and efficient solutions known in systems analysis.

- Employs and adapts genetic programming for finding CA state transition rules. As a result, a very flexible method is obtained which can be used in situations for which the size of transition rules is limited and we can define an appropriate fitness function. In contrast to previous works on this topic, the choice of the optimal neighbourhood for the lattices has also been considered and stochastic CA's have been taken into account.

The approach suggested here has the advantage that it is independent of a particular form of the CA describing the disturbed system under consideration. Moreover, it can easily be generalized to three spatial dimensions and the only limitation is the amount of required computations and memory.

The author believes that his approach has significant advantages which will make it, with sufficient development, a leading approach to solving CA identification problems facing scientists and engineers involved in applications. However, there still remain open problems regarding some important areas. What follows is a discussion of the areas for further investigation, besides applications.

**Further development of the hybrid approach.** Although the proposed hybrid approach to modelling forest dynamics turns out to work correctly for real data, some efforts are still expected towards a better representation of the shapes of trees and validation on larger ecosystems for longer periods (more than 10 years).

**Furher development of parameter estimation methods.** Parameter estimation techniques presented in this thesis have their origins in statistics. In the thesis estimates of model parameters were indicated from the data, but no measures of precision are available. The next step would be (approximate) inference on the model parameters, e.g. the bias and variance of an estimator are desirable. Enough distribution theory at hand would allow for performing this step. Some recent advances in spatial statistics [60] are expected to be helpful to attain this goal.

**Optimum experimental design.** The thesis has been primarily concerned with the problem of extracting information from a given set of data. However, in

most situations there are a number of variables which can be adjusted, subject to certain constraints, so that the information provided by the experiment be maximized. Optimum experimental design is devoted to a study of the design of the experimental conditions so that the experiment is maximally informative. For dynamic systems, experiment design includes choice of input and measurement ports, test signals, initial conditions, sampling instants, etc. In the context of parameter estimation of CA models, this necessitates introduction of some measures of information provided by an experiment and then solving the corresponding optimization problem, which seems to be a quite complex task which is, however, extremely important in applications.

**Possible links with reinforcement learning.** The work on parameter and structure identification suggested that there may be some links of this problem with a reinforcement learning problem. In this language, the state transition function of a CA can be seen as a sought policy, i.e. the decision-making function (control strategy) of the agent, which represents a mapping from situations to actions. Most reinforcement learning algorithms can be viewed as stochastic approximations of exact dynamic programming algorithms, where instead of complete sweeps over the state space, only selected states are backed up (which are sampled according to the underlying probabilistic model). Potential results in this field would be very interesting, since the reinforcement learing has devoloped rapidly over the last years.

# STRESZCZENIE

Układy dynamiczne z czasoprzestrzenną dynamiką, lub inaczej tzw. układy dynamiczne o parametrach rozłożonych, stanowią ważną klasę szeroko rozumianych układów dynamicznych. Wiele różnorodnych procesów fizycznych daje się opisać jedynie za pomocą równań różniczkowych cząstkowych określonych w odpowiednio zdefiniowanych obszarach i przy zadanych różnego typu warunkach brzegowych. W literaturze specjalistycznej dotyczącej teorii systemów i teorii sterowania istnieje bardzo wiele prac związanych bezpośrednio z takimi układami. Związane jest to z jednej strony z szerokimi możliwościami aplikacyjnymi, a z drugiej strony z istnieniem wielu trudnych problemów teoretycznych nie w pełni do tej pory rozwiązanych.

W skutek częstego występowania silnych nieliniowości w układach tego typu, do ich opisu stosuje się nieliniowe równania różniczkowe cząstkowe, których rozwiązywanie nawet na poziomie prostych zagadnień okazuje się bardzo złożone pomimo coraz szerszej dostępności wydajnych narzędzi numerycznych i informatycznych. Przykładowo, rozwiązania mogą być silnie wrażliwe na warunki początkowe, co prowadzi do zachowań chaotycznych. Ponadto, ich analiza wymaga bardzo wyrafinowanego aparatu matematycznego (zaawansowana analiza funkcjonalna). Wyznaczanie rozwiązań prowadzi też zazwyczaj do numerycznego rozwiązywania układów równań o bardzo dużej liczbie niewiadomych ( dokładne modelowanie implikuje stosowanie gęstych siatek przestrzennych), co stanowi problem sam w sobie i wiąże się zarówno z zagadnieniami stabilności i odporności na błędy numeryczne, jak również koniecznością stosowania wysoko wydajnych narzędzi obliczeniowych.

Modele oparte na automatach komórkowych stanowią interesującą alternatywę w badaniach nad układami z czasoprzestrzenną dynamiką. Ze względu na swoją prostotę i dyskretny charakter używanych wielkości, są one potencjalnie prostsze w analizie niż układy ciągłe, a zagadnienia numeryczne są wolne od błędów zaokrągleń. Modele tego typu stają się coraz bardziej popularne, zwłaszcza wśród badaczy zajmujących się zastosowaniami, np. w ekologii, co spowodowane jest ich prostotą i nadzwyczajną łatwością w implementacji komputerowej.

Automat komórkowy jest dyskretnym układem dynamicznym zdefiniowanym w dyskretnym czasie i przestrzeni. Jego podstawowymi elementami składowymi są jednakowe komórki uporządkowane w regularną siatkę przestrzenną (np. prostokątną lub sześciokątną w przypadku dwuwymiarowym). Komórka jest rodzajem pamięci przechowującej stany. Stan każdej komórki jest zmienną, która przyjmuje wartość z zadanego skończonego zbioru wartości dopuszczalnych. Automat ewoluuje w dyskretnych chwilach czasu, w których stany jego komórek są uaktualniane synchronicznie z zastosowaniem zestawu lokalnych reguł przejść, które określają zależność stanu komórki w danej chwili czasowej od stanów jej i komórek sąsiednich w chwili poprzedniej. Wszystkie komórki stosują ten sam zestaw reguł.

Jednym z podstawowych zastosowań automatów komórkowych jest modelowanie złożonych procesów fizycznych, gdzie okazują się często pełnowartościowymi alternatywami modeli w postaci równań różniczkowych cząstkowych. Przykładowo, w symulacjach hydrodynamicznych do modelowania przepływu, jak również w kinetycznej teorii gazu, płynów nie mieszających się, konwekcji i magnetohydrodynamice, bardzo popularne są modele tzw. gazów siatkowych HPP i FHP (można np. pokazać, że gazy siatkowe imitują przepływ Naviera-Stokesa). Komórkowe modele zjawisk czasoprzestrzennych stały się szczególnie rozpowszechnione wśród biologów i badaczy związanych z ekologią, co wiąże się przede wszystkim z niewielkim stopniem skomplikowania tych modeli, intuicyjną interpretacją reguł przejść i prostotą wprowadzania np. składowych losowych. W tym kontekście, równania różniczkowe cząstkowe i automaty komórkowe stanowią dwa odmienne podejścia do modelowania tych samych procesów: pierwsze z nich modelują populacje gatunków opisane gęstościami osobników zmieniającymi się wczasie i przestrzeni, a drugie pojedyncze osobniki poruszające się po siatce przestrzennej i wchodzące w interakcje z osobnikami sąsiednimi. Obydwa podejścia stanowią uproszczenia rzeczywistości. Równania różniczkowe cząstkowe interpretowane jako opisy w wielkiej skali abstrahują od lokalnych korelacji. Z kolei automaty komórkowe operują na poziomie molekularnym i skupiają się na lokalnych korelacjach, jednak nie opisują korelacji dalekiego zasięgu tak dobrze jak równania różniczkowe cząstkowe.

Wraz z intensywnym rozwojem techniki komputerowej obserwuje się coraz większe zainteresowanie symulacjami z zastosowaniem automatów komórkowych i należy spodziewać się, że wraz z upływem czasu modele komórkowe, rozwijane ze zmiennym zainteresowaniem od kilkudziesięciu lat, staną się równoważną alternatywą ich ciągłych odpowiedników klasycznych intensywnie rozwijanych przez długie dziesięciolecia.

Jedną z najistotniejszych przeszkód w rozpowszechnieniu automatów komórkowych jako modeli procesów rzeczywistych jest brak efektywnych metod rozwiązywania zagadnień odwrotnych, a w szczególności konstruowania lokalnych reguł przejść automatu w oparciu o dostępne dane pomiarowe. Zdecydowana większość znanych automatów opiera się bowiem na dokładnej znajomości fizyki rozważanego zjawiska, co pozwala na określenie postaci funkcyjnej reguł, oraz na arbitralne przyjętych wartościach występujących w niej parametrów. Oczywiście, jest to wystarczające na etapie badania przydatności modeli komórkowych do odzwierciedlania pewnych cech jakościowych zjawisk rzeczywistych, jednak nie wystarcza w zastosowaniach mających na celu dokonywanie predykcji zachowania obserwowanego procesu lub określania optymalnego oddziaływania (sterowania) na rozważane zjawisko w celu otrzymania jego określonego zachowania. W dostępnej literaturze znanych jest zaledwie kilka prac z tego zakresu, które opisują na dodatek rozwiązania obciążone wieloma niedogodnościami podającymi w wątpliwość ich praktyczną użyteczność. Klasyczną pozycją jest tu monografia Adamatzky'ego (1994), w której przedstawiono zestaw algorytmów pozwalających określić tablicę przejść automatu (co jest równoważne określeniu lokalnej funkcji przejścia) w oparciu o dane pochodzące z obserwacji zachowania automatu. Autor abstrahuje jednak od modeli parametrycznych i wyklucza możliwość

występowania zakłóceń pomiarowych (jest to podstawowe założenie stosowalności proponowanych algorytmów).

W ostatnich latach można zaobserwować wysiłki rozmaitych badaczy mające na celu wykorzystanie algorytmów genetycznych w celu określenia nieznanych parametrów założonego modelu automatu (Yang i Billing, 2000) lub identyfikacji tablicy przejść automatu (Mitchell, 1975; Crutchfield i Hanson, 1999; Koza i Hall, 1993; Sipper, 1997). Wprawdzie próby te uwzględniają możliwość występowania zakłóceń wdanych pomiarowych, jednak ograniczają się do automatów deterministycznych i binarnych, a przedstawione przykłady obliczeniowe ograniczają się do abstrakcyjnych problemów nie związanych z modelowaniem zjawisk fizycznych. Ograniczenia te stanowiły motywację badań podsumowanych niniejszą rozprawą.

Tematem niniejszej pracy jest zbadanie przydatności wybranych metod i podejść znanych z identyfikacji układów dynamicznych i programowania nieliniowego w konstruowaniu modeli deterministycznych i stochastycznych automatów komórkowych w oparciu o obserwowane dane pomiarowe.

Celem pracy było opracowanie możliwie uniwersalnych i efektywnych metod oraz algorytmów określania reguł przejść automatu zarówno w sytuacji znanej, jak i nieznanej ich postaci funkcyjnej, z zamiarem ich stosowania w sytuacjach gdy przed badaczem staje do rozwiązania konkretne zagadnienie praktyczne związane z modelowaniem realnego procesu.

Na treść książki składa się wstęp, cztery zasadnicze rozdziały, uwagi końcowe oraz spis literatury. We wprowadzeniu przedstawiono definicję automatu komórkowego, omówiono szczegółowo jego podstawowe składowe (typy siatek, rodzaje sąsiedztwa, warunki brzegowe, opisy funkcji przejścia) ilustrując je przykładami (Gra w Życie), a następnie dokonano przeglądu zastosowań automatów, a w szczególności ich wykorzystania w badaniach nad uniwersalnością obliczeniową oraz (znacznie obszerniej) w modelowaniu procesów fizycznych, z przykładami dotyczącymi modelowania procesu dyfuzji i przepływu płynów. Wprowadzono również pojęcie automatu z opóźnieniami. Rozdział kończy przegląd literaturowy istniejących metod rozwiązywania zagadnień odwrotnych, sformułowanie zakresu pracy.

Rozdział drugi poświęcono w całości problemowi modelowania dynamiki wzrostu drzew w obszarach leśnych. Dokonano obszernego przeglądu modeli matematycznych, skupiając się przede wszystkim na najczęściej stosowanym modelu typu szczelina (ang. gap) mającym postać równania różniczkowego zwyczajnego opisującego czasową dynamikę średnicy pojedynczego drzewa. Układ złożony z wielu drzew różnych gatunków modeluje się więc odpowiednim układem równań różniczkowych zwyczajnych, który nie uwzględnia jednak oddziaływań przestrzennych między pojedynczymi drzewami. Oddziaływania te mają charakter lokalny i odnoszą się do transportu nasion i wzajemnego wpływu drzew na warunki nasłonecznienia oraz dostęp do wody i składników odżywczych. Szczegółowy model uwzględniający te czynniki przyjmuje postać układu nieliniowych równań różniczkowych cząstkowych, które są jednak zbyt skomplikowane do zastosowania w praktyce. W rozprawie proponuje się więc wykorzystanie w tym celu automatu komórkowego opisującego przemieszczanie się nasion i generującego impulsy oznaczające obumarcie danego drzewa według odpowiednio zdefiniowanych reguł.

Otrzymuje się w ten sposób model hybrydowy, w którym stany poszczególnych komórek są parami składającymi się ze średnicy drzewa i jego wieku (ich ewolucję opisuje model typu szczelina), a interakcje między poszczególnymi drzewami odbywają się według reguł przejścia automatu. Połączenie tego typu stanowi oryginalny pomysł autora, nie spotykany do tej pory w literaturze i dokładniej oddający rzeczywistość niż oryginalny model szczelinowy , co potwierdziły badania symulacyjne oparte na danych rzeczywistych dotyczących lasu mieszanego w zlewni rzeki Ratanicy koło Krakowa.

Rozdział trzeci dotyczy zagadnienia rozszerzalności w układach opisywanych automatami komórkowymi. Rozszerzalność jest oryginalnym pojęciem wprowadzonym przez prof. El Jai (El Jai i Kassara, 1996; Uciński i El Jai, 1997) w celu scharakteryzowania sytuacji często występującej w praktyce, gdy obszar będący nośnikiem pewnej cechy rozszerza się na coraz większą powierzchnię wchłaniając przyległe do niego punkty. Przykładami tego zjawiska są rozrost komórek nowotworowych, rozprzestrzenianie się epidemii, proces krystalizacji, przepływ płynu w środowisku porowatym, rozprzestrzenianie się obszarów pustynnych lub powiększanie się obszarów pokrytych roślinnością. Dotychczasowe badania wskazują, że zjawisko rozszerzania jest trudne do opisania z zastosowaniem klasycznego aparatu równań różniczkowych cząstkowych (z wyjątkiem równań typu hiperbolicznego ). W pracy pokazano, że może ono być stosunkowo łatwo odzwierciedlone z zastosowaniem automatów komórkowych, na przykładzie dwóch modeli rozprzestrzeniania się epidemii (model Eden i jego pochodna w postaci modelu pojedynczego skupienia perkolacji), które autor proponuje jako komórkowe implementacje modeli stosowanych dotychczas w biologii. W drugiej części rozdziału wprowadzono pojęcie sterowania, co umotywowane jest sytuacją często spotykaną w praktyce, gdy czynniki zewnętrzne wpływają istotnie na zachowanie procesu (automaty komórkowe definiuje się zazwyczaj jako układy autonomiczne). Jako przykłady podaje się model inwazyjnej perkolacji (rozmieszczenie początkowe komórek odpornych na chorobę ma istotny wpływ na sposób rozprzestrzeniania się zarazy) oraz model dyfuzji cząstek o ograniczonej energii (na przebieg procesu wpływa się przez rozmieszczenie źródeł cząstek). Modele te są bardzo proste i zjawisko rozszerzania występuje w nich w sposób naturalny, w przeciwieństwie do modeli ciągłych.

Rozdział czwarty, poświęcony w całości estymacji parametrów modeli komórkowych, zawiera najwięcej oryginalnych rezultatów z uwagi na bardzo niewielką liczbę publikacji literaturowych poświęconych temu zagadnieniu. W proponowanym sformułowaniu postać funkcyjna reguł przejścia jest dana, nie są znane natomiast niektóre z występujących w niej parametrów, które można odtworzyć na podstawie obserwacji zachowania procesu opisywanego modelem komórkowym. W przypadku automatów deterministycznych dopuszczono występowanie obserwacji zaszumi onych i sprowadzono zadanie estymacji do problemu minimalizacji kryterium niedopasowania modelu do danych pomiarowych (badano kryteria sumy odległości Hamminga, sumy wartości bezwzględnych błędów i sumy kwadratów błędów). Ponieważ najczęściej parametry mogą przyjmować wartości dyskretne, podobnie jak minimalizowane kryterium, które na dodatek posiada

wiele minimów lokalnych, rozwiązanie problemu wymaga zastosowania algorytmów optymalizacji globalnej i/lub dyskretnej. W rozprawie zaproponowano w tym celu metodę poszukiwania lokalnego, metodę symulowanego wyżarzania i metodę adaptacyjnych poszukiwań losowych. Przedstawione rezultaty badań symulacyjnych dotyczących zarówno modeli abstrakcyjnych, jak i modeli opisujących złożone zjawiska rzeczywiste (krystalizacja przechłodzonej cieczy), potwierdzają efektywność zastosowanego podejścia. Wskazano ponadto na możliwość występowania niejednoznaczności estymat parametrów. W celu scharakteryzowania niepewności estymat proponuje się zastosowanie podejścia opartego na określeniu granicy obszaru niejednoznaczności przez tzw. chmurę punktów (podejście to zaproponowali Walter i Pronzato (1997) w kontekście ogólnego zadania estymacji parametrów). Druga część rozdziału dotyczy estymacji parametrów stochastycznych modeli komórkowych i stanowi istotne uogólnienie metody zaproponowanej w klasycznej monografii Adamatzky'ego (1994), gdzie rozważano jedynie estymację prawdopodobieństw przejść między stanowych w oparciu o częstościową definicję prawdopodobieństwa (tzn. nieznane prawdopodobieństwo przejścia z jednego stanu do innego przybliżano częstością występowania takiego przejścia w długiej serii obserwacji). W rozprawie autor uogólnia te rezultaty i rozważa o wiele bardziej złożone sytuacje (prawdopodobieństwa przejść międzystanowych mogą mieć złożone postacie funkcyjne ), a do estymacji proponuje zastosowanie metody największej wiarygodności. Podejście ilustrują przykłady, w których rozwiązania otrzymuje się zarówno analitycznie (stochastyczny pożar lasu), jak i z zastosowaniem algorytmu adaptacyjnych poszukiwań losowych (modele przestrzennej dynamiki typu pasożyt-żywiciel, pożar lasu, rozprzestrzenianie się wścieklizny).

W rozdziale piątym zaproponowano podejście do identyfikacji struktury funkcji przejścia automatu komórkowego oparte na zastosowaniu programowania genetycznego. Programowanie genetyczne stanowi jedną z mniej znanych technik ewolucyjnych obejmujących m.in. algorytmy genetyczne, i polega na zastosowaniu klasycznych operatorów genetycznych (reprodukcja, krzyżowanie, mutacja) do populacji funkcji, zamiast populacji ciągów binarnych. Funkcje reprezentowane są w postaci drzew, których węzłami są operatory (np. suma, różnica, potęgowanie), a liściami elementy pewnego skończonego zbioru terminali. W rozprawie pokazano, że taka reprezentacja doskonale nadaje się do odwzorowania struktury funkcji przejścia automatu o ile do zbioru terminali włączy się komórki z założonego sąsiedztwa automatu (np. sąsiedztwa Moore'a). Zadanie określenia struktury automatu sprowadzono w ten sposób do poszukiwania drzewa minimalizującego kryterium niedopasowania modelu do obserwowanych danych pomiarowych. Jako istotne uogólnienia, rozważano dobór optymalnego sąsiedztwa automatu oraz dobór struktury automatów stochastycznych (poprzez zastosowanie podejścia opartego na metodzie Monte Carlo ). Problemów tych nie rozważano dotychczas w literaturze, chociaż pomysł wykorzystania programowania genetycznego w określaniu struktury funkcji przejścia znany jest od dawna (zob. np. publikacje Kozy z początku lat 90-tych, odnoszące się jednak do bardzo prostych automatów binarnych).

Książkę kończy rozdział zawierający podsumowanie i kierunki przyszłych badań,

dotyczących m.in. dalszego rozwoju podejścia hybrydowego i metod estymacji parametrów, zagadnienia optymalnego planowania eksperymentu i potencjalnych związków zadania identyfikacji funkcji przejścia z problemem uczenia ze wzmocnieniem.

Za oryginalne osiągnięcia pracy autor uznaje: opracowanie nowego hybrydowego modelu dynamiki wzrostu drzew w obszarach leśnych, pozwalającego w prosty sposób uwzględniać interakcje między różnymi drzewami wielu gatunków, ijego weryfikacja z zastosowaniem danych rzeczywistych; opracowanie metod estymacji parametrów funkcji przejścia automatów deterministycznycyh w oparciu o wybrane metody optymalizacji globalnej i automatów stochastycznych w oparciu o metodę największej wiarygodności; opracowanie metody identyfikacji struktury funkcji przejścia i otoczenia automatu z zastosowaniem programowania genetycznego.

Książkę realizowano w ramach projektu promotorskiego finansowanego przez KBN, programu polsko-francuskich działań zintegrowanych POLONIUM oraz projektu europejskiego ENV4CT96-0320 (w ramach tego ostatniego, we współpracy z Laboratoire de Théorie des Systèmes Uniwersytetu w Perpignan we Francji, zrealizowano badania opisane w rozdziałach 2 i 3).

# BIBLIOGRAPHY


[1] A. Adamatzky. *Identification of cellular automata*. Taylor & Francis Ed., 1994.

[2] J. Albert and K. II Čulik. A simple universal cellular automaton and its one-way totalistic version. *Complex Systems*, 1:1–16, 1987.

[3] M. Alpert. No just fun and games. *Scientific American*, 499, 1999.

[4] M. Ya. Antonovski, M. T. Ter-Mikaelian, and V. V. Furyaev. A spatial model of long-term forest fire dynamics and its applications to forest in western siberia. In H. H. Shugart, R. Leemans, and G. B. Bonan, editors, *A Systems Analysis of the Global Boreal Forest*. Cambridge University Press, Cambridge, 1992.

[5] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing Ltd, Bristol and Oxford University Press, New York, 1997.

[6] E. R. Banks. Universality in cellular automata. *I.E.E.E. Ann. Symp. Switching and Automata Theory*, 11:194–215, 1970.

[7] D. B. Botkin, J. F. Janak, and J. R. Wallis. Rationale, limitations and assumptions of a northeastern forest growth simulator. *IBM J. Res. Develop.*, 16:101–116, 1972.

[8] D. B. Botkin, J. F. Janak, and J. R. Wallis. Some ecological consequences of a computer model of forest growth. *Journal Ecological*, 60:849–873, 1972.

[9] H. Brufau. *Analyse et Modélisation de Dynamiques de Végétation*. PhD thesis, Universitè de Perpignan, 1995.

[10] E. Burks. *Theory of Self-reproduction*. University of Illinois Press, Chicago, 1966.

[11] P. Callahan. Wonders of math – the game of life. Availabe at: http://www.math.com/students/wonders/life/life.html.

[12] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[13] F. E. Clements. *Plant succession an indicators*. Wilson, New York, 1928.

[14] E. F. Codd. *Cellular Automata*. Academic Press, New York, 1968.

[15] S. Cole. Real-time computation by n-dimensional iterative arrays of finite-state machine. *IEEE Trans. Comput.*, C-18:349–365, 1969.

[16] J. H. Connell and R. O. Slatyer. Mechanisms of succession in natural communities and their role in community stability and organization. *Am. Nat.*, 111:1119–1144, 1977.

[17] M. Cronhjort. *Models and Computer Simulations of Origins of Life and Evolution*. PhD thesis, Kungliga Tekniska Högskolan, Stockholm, Sweden, 1995.

[18] M. Cronhjort and C. Blomberg. A mechanism for resistance against parasites in self-replicating systems. In J. Weinstein, editor, *Artificial Life V*, Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, pages 413–417, Cambridge, 1997. USA: MIT Press.

[19] J. P. Crutchfield and J. E. Hanson. *Computational Mechanics of Cellular Processes*. Princeton Univ Press, Princeton, 1999.

[20] M. Delorme and J. Mazoyer. *Cellular Automata. A Parallel Model*. Kluwer Academic Publishers, Dordrecht, 1999.

[21] U. Dieckmann, R. Law, and J. A. J. Metz, editors. *The Geometry of Ecological Interactions: Simplifying Spatial Complexity*. Cambridge University Press, Cambridge, United Kingdom, 2000.

[22] B. Dossel and F. Schwabl. Formation of space-time structure in a forest-fire model. *Physica A*, 204:212–229, 1994.

[23] W. H. Drury and I. C. T. Nisbet. *Succession*, volume 54, pages 331–368. J. Arnold Arbor, 1973.

[24] A. R. Ek and R. A. Monserud. Forest: A computer model for the growth and reproduction of mixed species forest stands. Technical Report A2635, College of Agricultural and Life Sciences, University of Wisconsin, Madison, 1974.

[25] G. S. Fishman. *Monte Carlo - Concepts, Algorithms, and Applications*. Springer-Verlag, New York, 1996.

[26] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Phys. Rev. Lett.*, 56:1505–1508, 1986.

[27] R. J. Gaylord and K. Nishidate. *Cellular Automata simulations with Mathematica*. Springer–Verlag, New York, 1996.

[28] R. J. Gaylord and P. R. Wellin. *Computer Simulations with Mathematica*. Springer–Verlag, New York, 1995.

[29] H. A. Gleason. The individualistic concept of the plant association. *Am. Midl. Nat.*, 21:92–110, 1939.

[30] G. C. Goodwin and R. L. Payne. *Dynamic System Identification – Experiment Design and Data Analysis*. Mathematics in Science and Engineering. Academic Press, New York, 1977.

[31] D. G. Green. *Cellular automat (Environmental and Information sciences)*. Charles Sturt University, 1989.

[32] K. Grodzińska and R. Laskowski, editors. *Environmental assessment and biogeochemistry of a moderately polluted Ratanica catchment*. Państwowa Inspekcja Ochrony Środowiska, Biblioteka Monitoringu Środowiska, 1996.

[33] L. J. Gross. Behavioral ecology and individual-based modeling. Available at: http://ecology.tiem.utk.edu/ gross/behav.txt, 1998.

[34] J. Hardy, O. de Pazzis, and Y. Pomeau. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. *Phys. Rev.*, A13:1949–1960, 1976.

[35] F. Hegyi. A simulation model for managing jack-pine stands. In J. Fries, editor, *Growth Models for Tree and Stand Simulation*, pages 74–87. Departament of Forest Yeld Research, Stockholm, 1974.

[36] J. H. Holland. *Adaptation in natural and aritificial systems*. The University of Michigan Press, 1975.

[37] W. Hordijk. The structure of the synchronizing-ca landscape. *Available at: ftp://ftp.santafe.edu/pub/wim/struct.CA.ps.Z*, 1996.

[38] H. S. Horn. Forest succession. *Sci. Am.*, 232:90–98, 1975.

[39] H. S. Horn. Succession. *R. M. May (ed.)*, Theoretical Ecology:187–204, 1976.

[40] A. Smith III. Simple computation-universal spaces. *Journal of ACM*, 18:339–353, 1971.

[41] P. Jacewicz and J. Korbicz. A cellular automata approach to modeling forest dynamics. In *Proc. 10th Int. Conf. System Modelling Control*, volume 1, pages 307–314, Zakopane, Poland, 2001.

[42] P. Jacewicz and S. El Yacoubi. A genenetic programming approach to structural identification of cellular automata. In *3rd International Conference on Parallel Processing and Applied Mathematics*, pages 148–157, 1999.

[43] P. Jacewicz and S. El Yacoubi. Structural estimation of cellular automata. In *Proc. 4th IFIP WG 1.5 Meeting AUTOMATA'99 – Workshop on Cellular Automata*, page 31, Lyon, France, 1999.

[44] A. El Jai and K. Kassara. Spreadability of transport systems. *Int. Journal Sys. Sci.*, 27(7):681–688, 1996.

[45] A. El Jai, S. El Yacoubi, and J. Karrakchou. Spreadability and spray actuators. *Applied Mathematics and Computer Science*, 8(2), 1998.

[46] L. Kadanoff. On two levels. *Phisics Today*, 39:9:7–9, 1986.

[47] R. E. Keane, S. F. Arno, and J. K. Brown. Firesum – an ecological process model for fire succession in western conifer forests. Technical Report INT-266, United States Department of Agriculture, Forest Service, 1989.

[48] M. J. Keeling. Evolutionary dynamics in spatial host-parasite systems. In U. Dieckmann, R. Law, and J. A. J. Metz, editors, *The Geometry of Ecological Interactions: Simplifying Spatial Complexity*. Cambridge University Press, Cambridge, United Kingdom, 2000.

[49] A. Kleczkowski, D. J. Bailey, and C. A. Gilligan. *Scaling-up of the variability in the plant-pathogen*. University of Cambridge, 1995.

[50] A. Kowalewski. *Optimal control of infinite dimensional distributed parameter systems with delays*. Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Academy of Mining and Metallurgy, Cracow, Poland, 2001.

[51] J. R. Koza and M. Jacks Hall. Discovery of rewrite rules in lindenmayer systems and state transition rules in cellular automata via genetic programming. In *Symposium on Pattern Formation (SPF-93) at Claremont*, 1993.

[52] B. Kuczewski and D. Uciński. Structure identification of cellular automata using genetic programming. In *10th International Conference on System Modelling Control*, volume 1, pages 413–418, Technical University of Łódź, 2001. Institute of Computer Science.

[53] R. L. Lindeman. The trophic-dynamic aspect of ecology. *Ecology*, 23:399–418, 1942.

[54] R. Margalef. On certain unifying principles in ecology. *Am. Nat.*, 97:357–374, 1963.

[55] N. Margolus. Physics-like models of computation. *Physica*, 10D:128–134, 1984.

[56] J. McCormick. Succession. *Via*, 1:1–16, 1968.

[57] R. P. McIntosh. *Forest Succession: Concepts and Application*, pages 10–23. Springer-Verlag, New York, 1981.

[58] K. J. Mitchell. Dynamics and simulated yeld of douglas-fir. *For. Sci. Monogr.*, 39, 1975.

[59] D. P. Morton and E. Popova. Monte-carlo simulations for stochastic optimization. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 439–447. Kluwer Academic Publishers, 2001.

[60] W. G. Müller. *Collecting Spatial Data (Optimum Design of Experiments for Random Fields)*. Physica-Verlag. Springer-Verlag, Heidelberg, Germany, 2001.

[61] J. D. Murray. *Mathematical Biology*. Springer–Verlag, 1993.

[62] E. P. Odum. The strategy of ecosystem development. *Science*, 164:262–270, 1969.

[63] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: algorithms and Complexity*. Prentice-Hall, New Jersey, 1982.

[64] S. T. A. Pickett. Succession: An evolutionary interpretation. *Am. Nat.*, 110:108–119, 1976.

[65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[66] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2002.

[67] D. A. Rand and H. B. Wilson. Using spatio-temporal chaos and intermediate-scale determinism to quantify spatially extended ecosystems. *Proceedings of the Royal Society of London*, 259:111–117, 1995.

[68] A. Ratz. Long term spatial pattern created by fire: A model oriented towards boreal forests. *International Journal of Wildland Fire*, 5:25–34, 1995.

[69] A. Ratz. *A Generic Forest Fire Model: Spatial Patterns in Forest Fire Ecosystems*. PhD thesis, University of Magdeburg, Germany, 1996.

[70] H. H. Shugart. *A Theory of Forest Dynamics (The Ecological Implications of Forest Succession Models)*. Springer-Verlag, New York, 1984.

[71] H. H. Shugart. *Terrestrial ecosystems in changing environments*. Cambridge University Press, Cambridge, 1998.

[72] H. H. Shugart and D. C. West. Development of an appalachian deciduous forest succession model and its application to assessment of the impact of the chestnut blight. *Journal Environ. Manag.*, 5:161–179, 1977.

[73] M. Sipper. *Evolution of Parallel Cellular Machines. The Cellular Programming Approach*, volume 1194 of *Lecture Notes in Computer Science*. Springer, Berlin, 1997.

[74] D. S. Solomon. Simulation of the development of natural and silviculturally treated stands of even-aged northern hardwoods. In J. Fries, editor, *Growth Models for Trees and Stand Simulation*, pages 327–352. Department of Forest Yeld Research, Royal College of Forestry, Stockholm, 1974.

[75] A. D. Sullivan and J. L. Clutter. A simultaneous growth and yeld model for loblolly pine. *For. Sci.*, 18:76–86, 1972.

[76] T. Suzuki and T. Umemura. Forest transition as a stochastic process. In J. Fries, editor, *Growth Models for Trees and Stand Simulation*, pages 327–352. Department of Forest Yeld Research, Royal College of Forestry, Stockholm, 1974.

[77] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modelling*. MIT Press, 1987.

[78] D. Uciński and A. El Jai. On weak spreadability of distributed-parameter systems and its achievement via linear-quadratic control techniques. *Journal of Mathematical Control and Information*, 14:153–174, 1997.

[79] D. Uciński and S. El Yacoubi. Parameter estimation of cellular models. In *3rd International Conference on Parallel Processing and Applied Mathematics*, Technical University of Częstochowa, 1991. Institute of Mathematics and Computer Science.

[80] O. van Tongeren and I. C. Prentice. A spatial simulation model for vegetation dynamics. *Vegetatio*, 65:163–173, 1986.

[81] J. von Neumann. *Probabilistic logic and the synthesis of reliable organisms from unreliable components*. Princeton University Press, Princeton, 1956.

[82] J. von Neumann. *The Computer and the Brain*. Yale University Press, New Haven, 1958.

[83] J. von Neumann. The general and logical theory of automata. *Collected Works*, 5:288–328, 1963.

[84] J. von Neumann. *Theory of Self-reproducing automata*. University of Illinois Press, Chicago, 1966.

[85] P. E. Waggoner and G. R. Stephens. Transition probabilities for a forest. *Nature*, 255:1160–1161, 1970.

[86] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Communications and Control Engineering. Springer-Verlag, Berlin, 1997.

[87] A. S. Watt. Pattern and process in the plant community. *Journal Ecological*, 35:1–22, 1947.

[88] J. R. Weimar. *Simulation with Cellular Automata*. Logos-Verlag, Berlin, 1998.

[89] S. El Yacoubi and P. Jacewicz. Cellular automata as a means for modelling complex spatio-temporal systems. In *Proc. 5th International Conference Mathematical Population Dynamics*, 1998.

[90] S. El Yacoubi and P. Jacewicz. Systems theory via cellular automata. In *Proc. 2nd International Workshop on Analysis and Optimization of Complex Environemental Systems*, pages 15–20, 1998.

[91] S. El Yacoubi and P. Jacewicz. Cellular automata and controllability problem. In *Proc. 14th Symposium of Mathematical Theory of Networks and Systems MTNS 2000*, pages CD–ROM, Perpignan, France, 2000.

[92] S. El Yacoubi, A. El Jai, and P. Jacewicz. Lucas: an original tool for landscape modelling. Journal of Environmental Modelling & Software (accepted), 2002.

[93] Y. Yang and S. A. Billings. Extracting boolean rules from ca patterns. *Cybernetics*, 30(4), 2000.