

## Chapter 2

# Cellular Automaton-Based Shading System (CASS)

**Abstract** This chapter collects the findings of the research on the cellular automaton-based shading systems (CASS) for building envelopes. CASS is based on congruent modular units, thus it has the potential of being inexpensive and robust. Two approaches for the realization of CASS are presented: based on the liquid crystal technology, and based on the rotation of polarized film elements. Several optimization methods of CASS are presented. The optimization criteria include: the “grayness” monotonicity, and cellular automaton (CA) pattern distribution error which represent: the level of control over the CA pattern, and its uniformity over entire array of cells, respectively. The robustness of CASS for various types of failure is discussed.

**Keywords** Evolutionary algorithm · Emergence · Grayness · LCD · Organic · Pattern · Regular grid · Shading

## 2.1 Cellular Automata

Cellular automata (CA) are dynamical systems which are discrete in space and time, operate on a topologically regular lattice and are characterized by local interactions [48]. CA are discrete complex systems which can be regarded as a fully distributed computational system with local processing only of very simple components [49]. CA models are used in computability theory, mathematics, physics, complexity science, theoretical biology, and microstructure modeling.

The history of CA can be traced down to 1930s, where the ideas of automata was introduced by Post [33] and Turing [44]. Commenced in the late 1940s, von Neumann’s studies on the complexity required for a device or system to be self-reproductive. He pursued the idea of a kinematic automaton which could, using a description of itself, proceed to mechanically assemble a duplicate from available

parts [47]. However, due to difficulties in providing the rigorous and explicit rules and instructions for such physical automaton, he redirected to modeling of self-reproduction using an array of computing elements, thereby conceiving the concept of CA. The first cellular logic machine [35], which was not a CA emulator in the strictest sense, since it used “look-back” circuitry, was created a decade later. This and other special purpose machines emulated CA by using a single high-speed processing element to operate sequentially on an array of binary data. Historical examples, especially ones designed for image processing are described in [34]. Within the following decade, a general purpose CA simulator was developed [8] and in mid 1980s more direct and efficient hardware realization – a CA machine [41] was completed. More of historical examples are described in [42]. Presently CAs are the most successfully implemented in computer simulations and relatively few physical devices have been built. This chapter concerns itself with particular class of engineering problems, that is adaptation to dynamically changing environment. The most relevant examples of CA applications in this field are: CA-based modular illumination system [3], shading façade prototype [58], and the electric  $3 \times 8$  cells, universal, that is emulating 256 elementary cellular automata (ECA) educational toy [58]. Another interesting examples, although not exactly implementing CAs, are modular robots as multi-agent system applied for self-adaptive tasks [50].

Presently, the practical implementations of CAs are limited mostly to mathematical modeling, simulation and optimization, and as mentioned above, only a few CA-based physical devices have been created and documented. Probably the most interesting property of CAs is the fact that despite their underlying simplicity, they are capable of *strong emergence*. In other words, CAs can produce properties which are not reducible to their individual constituents [11, 14]. This phenomenon is exhibited through visual patterns which often possess intriguing aesthetics. They can be creatively applied in various fields of visually oriented design.

CA cells are locally connected to the neighboring cells and respond to the states of: their own cell, and its neighbors. For each CA the responses for all possible neighborhood-states are collected in so-called local transition rules (TR). E.g.: in a one-dimensional CA, the transition rule:

$$(0, \mathbf{1}, 1) \rightarrow 0$$

means that, if the central cell (boldfaced) has value 1 and the values in the neighboring left and right cells are: 0 and 1, respectively – in the subsequent time-step of CA evolution the value of central cell becomes 0. Since in this neighborhood-state there are three binary cells, there are a total of  $2^3 = 8$  TRs. The decimal notation of the neighborhood-state outputs is the CA rule number. For example, the following set of TRs:

$$\begin{aligned} (1, \mathbf{1}, 1) &\rightarrow 1 \\ (1, \mathbf{1}, 0) &\rightarrow 0 \\ (1, \mathbf{0}, 1) &\rightarrow 0 \\ (1, \mathbf{0}, 0) &\rightarrow 1 \\ (0, \mathbf{1}, 1) &\rightarrow 0 \end{aligned} \tag{2.1}$$

$$(0, \mathbf{1}, 0) \rightarrow 0$$

$$(0, \mathbf{0}, 1) \rightarrow 1$$

$$(0, \mathbf{0}, 0) \rightarrow 0$$

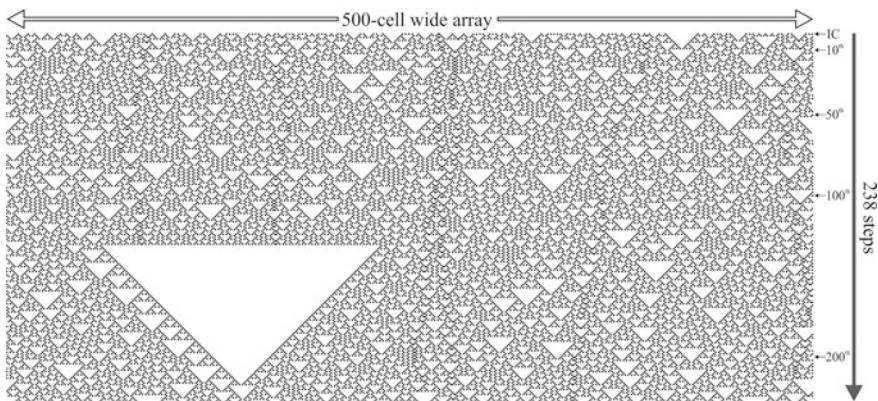
is named “rule 146”  $\leftarrow 146_{10} = 10010010_2 \leftarrow (1, 0, 0, 1, 0, 0, 1, 0)$ .

In a case of three possible values of the CA cell states: (2, 1, 0), the rule number is a decimal notation of the sequence of ternary TR outputs etc. In CA, the time-steps are discrete and the number of states of every cell is finite. CAs are modular, as all cells are identical. It is a very practical property in fabrication. Although the materialization of a single CA cell is relatively challenging, it has the potential of being mass-produced inexpensively [58].

The behavior of CAs is determined by:

- The dimension of the domain;
- The type and size of the neighborhood (defined by so-called radius/range);
- The number of possible states (colors) of each cell;
- The type of border conditions (BC);
- The initial conditions (IC).

Even some of the simplest nontrivial CAs, i.e.: one-dimensional (1D), two-color (2C), radius/range one (r1), so-called elementary cellular automata (ECA) are capable of emergence. The common convention of presenting 1D CA is by showing the history of generation changes, where one time-step is recorded as one row of cells in their present state. Such row becomes the initial condition (IC) for the subsequent row, and so on, as illustrated in Fig. 2.1.



**Fig. 2.1** Starting from certain IC, after 130 time-steps of ECA rule 146, emerges an unexpected pattern formation, so-called *monolith*

There are three basic types of automata [15], based on the “treatment” of the central and neighboring cells in the algorithm for computing the value for the central cell in subsequent time-step

- *General CA*: the direct values of the central and neighboring cells;
- *Semi-totalistic (ST)*: the direct value of the central cell and the summation of values in the neighboring cells;
- *Totalistic (T)*: the summation of all values, i.e., both in the central and neighboring cells.

Relatively common variations of automata

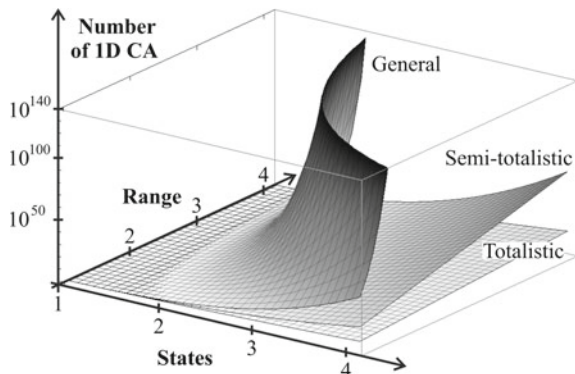
- *Half-distance CA* rules are created by shifting the successive rows, so the number of input cells becomes even;
- *Higher order CA* depends on both: the present and past states of the cells.

If an underlying cell is placed between the two cells above, it is called radius  $\frac{1}{2}$  ( $r - \frac{1}{2}$ ) CA. For radii:  $\frac{3}{2}$  ( $r - \frac{3}{2}$ ) and  $\frac{5}{2}$  ( $r - \frac{5}{2}$ ), the underlying cell is placed between the corresponding: 4 and 6 cells, respectively. For an illustrative interactive demonstration see [29].

A second-order CA (SOCA) is an example of a higher order CA. In this case, the state of the central cell at time-step  $t$  depends on the states of the central cell and the states of its neighboring cells at time  $t - 1$ , and state of the central cell at time  $t - 2$ . SOCA is also an example of *reversible CA*, RCA for short, i.e., a type of automaton where every configuration has only one previous configuration, and hence its evolution process can be traced backward uniquely [27].

Figure 2.2 illustrates the growth of the number of all possible rules of the three basic types of CA with increasing: size of neighborhood (range) and number of states (colors).

**Fig. 2.2** A visualization of the combinatorial explosion of the domain of 1D CA as a function of the number of states and the neighborhood size. Beyond two-state and range-two, it is substantially more convenient to investigate the totalistic or semi-totalistic CAs. The surfaces interpolate the discrete values



**Table 2.1** Selected types of CAs with examples and corresponding TRs

Type	Abbrev.	Cardinality	Example	
			Local transition rules (TRs)	Rule
General	(ECA)	256	$(1, 1, 1) \rightarrow 1, (1, 1, 0) \rightarrow 0,$ $(1, 0, 1) \rightarrow 0, (1, 0, 0) \rightarrow 1,$ $(0, 1, 1) \rightarrow 0, (0, 1, 0) \rightarrow 0,$ $(0, 0, 1) \rightarrow 1, (0, 0, 0) \rightarrow 0$	146
Radius-2	r2	$4.29 \times 10^9$	$(1, 1, 1, 1, 1) \rightarrow 1, (1, 1, 1, 1, 0) \rightarrow 1,$ $(1, 1, 1, 0, 1) \rightarrow 1, (1, 1, 1, 0, 0) \rightarrow 0,$ $(1, 1, 0, 1, 1) \rightarrow 0, (1, 1, 0, 1, 0) \rightarrow 0,$ $(1, 1, 0, 0, 1) \rightarrow 1, (1, 1, 0, 0, 0) \rightarrow 1,$ $(1, 0, 1, 1, 1) \rightarrow 1, (1, 0, 1, 1, 0) \rightarrow 0,$ $(1, 0, 1, 0, 1) \rightarrow 0, (1, 0, 1, 0, 0) \rightarrow 1,$ $(1, 0, 0, 1, 1) \rightarrow 1, (1, 0, 0, 1, 0) \rightarrow 1,$ $(1, 0, 0, 0, 1) \rightarrow 1, (1, 0, 0, 0, 0) \rightarrow 0,$ $(0, 1, 1, 1, 1) \rightarrow 1, (0, 1, 1, 1, 0) \rightarrow 0,$ $(0, 1, 1, 0, 1) \rightarrow 0, (0, 1, 1, 0, 0) \rightarrow 0,$ $(0, 1, 0, 1, 1) \rightarrow 0, (0, 1, 0, 1, 0) \rightarrow 1,$ $(0, 1, 0, 0, 1) \rightarrow 1, (0, 1, 0, 0, 0) \rightarrow 0,$ $(0, 0, 1, 1, 1) \rightarrow 0, (0, 0, 1, 1, 0) \rightarrow 0,$ $(0, 0, 1, 0, 1) \rightarrow 1, (0, 0, 1, 0, 0) \rightarrow 1,$ $(0, 0, 0, 1, 1) \rightarrow 1, (0, 0, 0, 1, 0) \rightarrow 0,$ $(0, 0, 0, 0, 1) \rightarrow 0, (0, 0, 0, 0, 0) \rightarrow 0$	3818817080
3-color	3C	$7.62 \times 10^{12}$	$(2, 2, 2) \rightarrow 0, (2, 2, 1) \rightarrow 0,$ $(2, 2, 0) \rightarrow 0, (2, 1, 2) \rightarrow 0,$ $(2, 1, 1) \rightarrow 1, (2, 1, 0) \rightarrow 1,$ $(2, 0, 2) \rightarrow 1, (2, 0, 1) \rightarrow 0,$ $(2, 0, 0) \rightarrow 0, (1, 2, 2) \rightarrow 0,$ $(1, 2, 1) \rightarrow 0, (1, 2, 0) \rightarrow 1,$ $(1, 1, 2) \rightarrow 1, (1, 1, 1) \rightarrow 1,$ $(1, 1, 0) \rightarrow 1, (1, 0, 2) \rightarrow 0,$ $(1, 0, 1) \rightarrow 1, (1, 0, 0) \rightarrow 1,$ $(0, 2, 2) \rightarrow 1, (0, 2, 1) \rightarrow 0,$ $(0, 2, 0) \rightarrow 0, (0, 1, 2) \rightarrow 1,$ $(0, 1, 1) \rightarrow 0, (0, 1, 0) \rightarrow 0, (0, 0, 2) \rightarrow 1,$ $(0, 0, 1) \rightarrow 1, (0, 0, 0) \rightarrow 1$	2378345543463
Totalistic	T	16	$3 \rightarrow 1, 2 \rightarrow 0, 1 \rightarrow 1, 0 \rightarrow 0$	10
Semi-totalistic	ST	64	$(1, 3) \rightarrow 0, (1, 2) \rightarrow 0, (1, 1) \rightarrow 0,$ $(1, 0) \rightarrow 0, (0, 3) \rightarrow 1, (0, 2) \rightarrow 0,$ $(0, 1) \rightarrow 0, (0, 0) \rightarrow 1,$	9
Half-distance	r-1/2	16	$(1, 1) \rightarrow 0, (1, 0) \rightarrow 0, (0, 1) \rightarrow 1,$ $(0, 0) \rightarrow 1$	3
3-color half-distance	3Cr-1/2	19683	$(2, 2) \rightarrow 2, (2, 1) \rightarrow 0, (2, 0) \rightarrow 1,$ $(1, 2) \rightarrow 1, (1, 1) \rightarrow 1, (1, 0) \rightarrow 2,$ $(0, 2) \rightarrow 0, (0, 1) \rightarrow 2, (0, 0) \rightarrow 2$	14237
3-color half-distance totalistic	3Cr-1/2T	243	$4 \rightarrow 2, 3 \rightarrow 2, 2 \rightarrow 1,$ $1 \rightarrow 0, 0 \rightarrow 1$	226
Totalistic second-order	TSO	128	$6 \rightarrow 0, 5 \rightarrow 1, 4 \rightarrow 1,$ $3 \rightarrow 0, 2 \rightarrow 1, 1 \rightarrow 0,$ $0 \rightarrow 0$	52

As Fig. 2.2 indicates, the number of possible *general* 1D CAs grows astronomically with the range and number of states. Thus in search for the rules of desired behavior, it is advisable to explore various simpler types of CA before expanding the neighborhood or increasing the number of states. Table 2.1 shows examples of TRs of the aforementioned automata.

## 2.2 Why Cellular Automata to Drive Shading of a Building Envelope?

The ability of a building façade to adjust dynamically to the changes in the environment is considered as an “organic” property. Moreover, the cellular (biomimetic) composition of the BE surface allows for implementation of cellular automaton for alternative approach to the dynamic control of BE. Table 2.2 collects the key properties of cellular automata (CA) which are suitable for potential application in shading of building façades.

*Modularity* is the most relevant to the fabrication process described in Chap. 4: “Prototypes.” CA modules are topologically uniform, however they may have various geometry suitable for free-form BE surfaces described in Sect. 4.5. *Determinism* and *emergence* derive from the sequence of CA patterns and their combination is the most challenging.

For shading control of BE, CA patterns and their transitions must meet the following criteria:

- CA patterns must be visually attractive;
- Ability to produce CA patterns of any average density between full: transparency to opacity;
- The transitions between CA patterns of different densities to be gradual and appear as “organic”;
- CA patterns must be evenly distributed over the BE surface.

**Table 2.2** The relationship between the key properties of CA and BE

Cellular automaton	Building envelope
Modularity: generally all CA cells of a given rule are identical; only border-cells might be different	Modularity is highly desirable in building industry, as it allows for mass prefabrication and easy assembly
Determinism	The control of the state of BE, at least statistical, is essential
Capability of strong emergence	Potentially novel, intriguing, organic-like aesthetics

The visual attractiveness of CA patterns can be attributed to the manifested *emergence*. According to Ref. [24]

**Emergence** is a property of a complex system that is not exhibited by its individual component parts determined from a model of the system.

According to Ref. [18]:

Emergence occurs in CAs by the simultaneous formation of boundaries between the domains, in a shape of borderlines or “particles”, that is small regions of cells separating two domains and persisting for relatively many time-steps.

So-called “solitons,” i.e., moving persistent structures which pass through one another while preserving their identities [39] is a particular type of particle.

Two complementary methods for automatic filtering of the changing configurations of spatial dynamical systems and extracting coherent structures in CAs have been proposed in [38]. Namely, *local sensitivity*, which calculates the degree to which local perturbations alter the system, and picks out autonomous features; and *local statistical complexity*, which calculates the amount of historical information required for optimal prediction and identifies the most highly organized features. In this chapter, however, a simple inspection of the spatio-temporal patterns and/or statistical regularities in the values of CA cells is used for emergence identification.

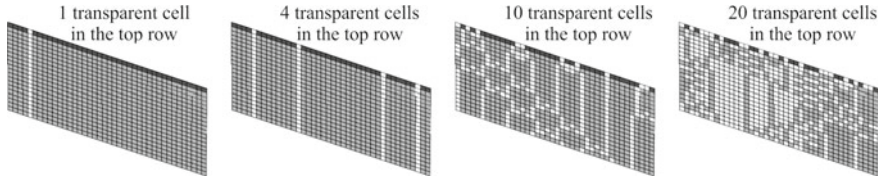
The judgment of visual attractiveness of BE is even more difficult than judgment of the aesthetics of a pure CA pattern. For example, different tessellations (see Chap. 3 “Polarized Film Shading System in regular grids”) produce different visual impacts. Nevertheless, despite these ambiguities, CASS clearly introduces new aesthetic possibilities, as illustrated with a number of figures in text.

In the following examples, two-state/color (2C) that is *binary* CAs are considered. A common convention is to assign 0 and 1 to: white and black, respectively. CA rules are encoded as decimals of the binary TR outputs. In an odd-number CA rule, the last neighborhood-state transition is from all 0s to 1 (compare to 2.1). This results in a very quick transition of the entire CA array to 1s and thus precludes the gradual transitions between the CA array states. In other words, even-number (EN) CA rules should be considered only. For simplicity, it is assumed that the shading action starts from full transparency, i.e., 0 is assigned to all CA cells.

In the concept of cellular automaton shading system (CASS) introduced in [52], the control over the CA array is executed by directly setting the values in the top row to give the initial conditions (ICs). The rest of the cells synchronously evolve down the array as illustrated by Fig. 2.3.

Some of the “shading” criteria listed above, like in many of real-life problems, are conflicting. For example, a perfectly distributed pattern, that is a “random noise,” is not acceptable from the aesthetic perspective.

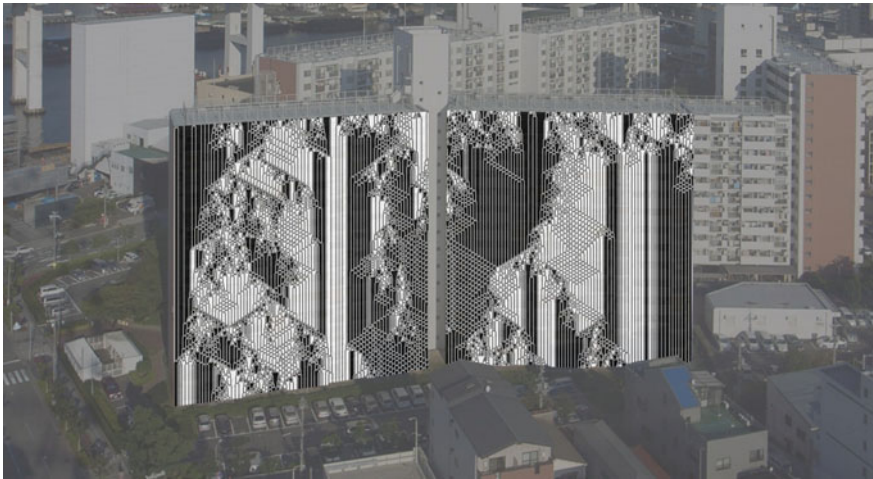




**Fig. 2.3** The state of the entire CA array is controlled by initial conditions (IC) set in the *top row*. From the *left* four subfigures correspond to arrays with patterns evolved from ICs with 1, 4, 10, and 20 transparent cells in the *top row*, respectively

Since none of 256 ECAs meets all four “shading criteria”, the domain has been extended to range-two (r2) CAs as described in [52]. Unfortunately, there are still no robust methods for designing CAs of a predefined behavior [49]. The search among over four million of such automata has been based on rule symmetry, i.e., for inverted ICs, the generated CA pattern must be exactly inverted:  $(0 \rightarrow 1, 1 \rightarrow 0)$ .

The first, and the most difficult to formalize criterion, is the visual attractiveness of a CA pattern. Although the final choice described in [52] has been arbitrary, “organic qualities,” were the most important aesthetic criteria. The organic approach to architecture is described in [17, 32], and specific discussion on CAs in the context of design, in [54]. A general two-color (2C) one-dimension (1D) range-two (r2) CA {3818817080, 2, 2},  $CA_{SH}$  for short, was proposed. The standard *Mathematica* notation is used, which for this kind of CA is in the following form:  $n, k, r$ , where  $n$  is the decimal enumeration of the list of outputs of the local transition rules (TR),  $k$  is the number of possible colors (states) of every cell, and  $r$  is the neighborhood size. A visualization of a façade with a simulated  $CA_{SH}$  pattern is shown in Fig. 2.4.



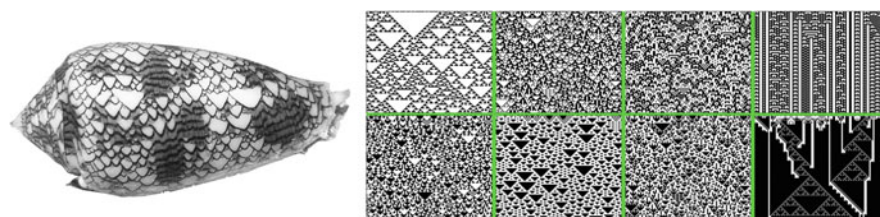
**Fig. 2.4** A visualization of  $CA_{SH}$



2.3 Are Cellular Automata Organic?

The most important quality of CA is the emergence: a collection of very simple cells may produce a complex function, as shown in Fig. 2.5.

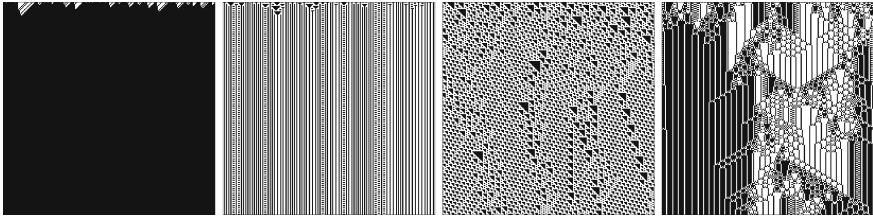
In case of regular CAs, the rules determining the behavior of individual cells are the same for each cell and often very simple. Their interactions, however, can produce patterns of unexpected complexity. In nature, the mechanisms are more subtle, but the general concept seems to be similar. In the design practice, the term “organic” has a broader meaning than *of, relating to, or derived from living organisms* [32]. It is often used to describe any naturally occurring phenomenon, for example a snowflake, which in a scientific sense is inorganic, as shown in Fig. 2.6.



**Fig. 2.5** On the *left* a mollusk shell *Conus textile*; photograph © 2005 Richard Ling. On the *right* seven examples of patterns generated by elementary cellular automata (ECA) and one by a three-color range-2 1D CA (*bottom right*). A mollusk shell, like a 1D CA grows one line at a time, with new shell material being produced by a lip of soft tissue at the edge of the animal inside the shell

Man-made (non-organic)		Natural (organic)	
2-color hexagonal CA rule 3629 at 43 <sup>rd</sup> step	2-color hexagonal CA rule 1126 at 43 <sup>rd</sup> step	2-color hexagonal CA rule 12837 at 44 <sup>th</sup> step	A real snowflake (Photo © 2010 Northwoods Press)
Simple pattern	Nested pattern	Organic pattern	Natural object
Geometry: Simple	Nested	Complex	Complex+stochastic
Perception: Trivial	Complicated, but easily understandable	Complicated, not easily understandable but intuitively rational	Complicated, rational and imperfect
Structure: Global shape has no interactions among the cells on the lower level	Nested shape, the same structure is repeated at every level, no interactions among cells	The shape emerges from multiple simple interactions at the cellular level	The shape emerges from not-so-simple multiple interactions at the cellular level

**Fig. 2.6** Manmade versus organic appearance



**Fig. 2.7** Four Wolfram classes of CA behavior: (1) Constant, (2) Periodic, (3) (Pseudo) Random, (4) Complex. Although some are more visually interesting than others, all have certain organic qualities. While it is questionable in class 1, 2, and 3, it is clear that class 4 is organic

Since CA is a biomimetic concept, its emergent behavior represented graphically by patterns has also certain organic quality. There are four main (Wolfram) classes of CA behavior, as distinguished and shown in Fig. 2.7. For each one of these classes there are cases which can to a certain degree be considered organic. Class 4 can be clearly identified as the one producing organic patterns.

CA is a decentralized system, where all cells respond only to their neighbors and their state cannot be altered “manually” (besides their initial conditions which are set directly). Usually the state of the cells can be predicted only to a certain degree, e.g., the average density. Although stochasticity can be implemented, regular CAs are deterministic, but in most cases (classes: 3 and 4) due to their computational irreducibility, the state of a certain cell can only be known by experiment.

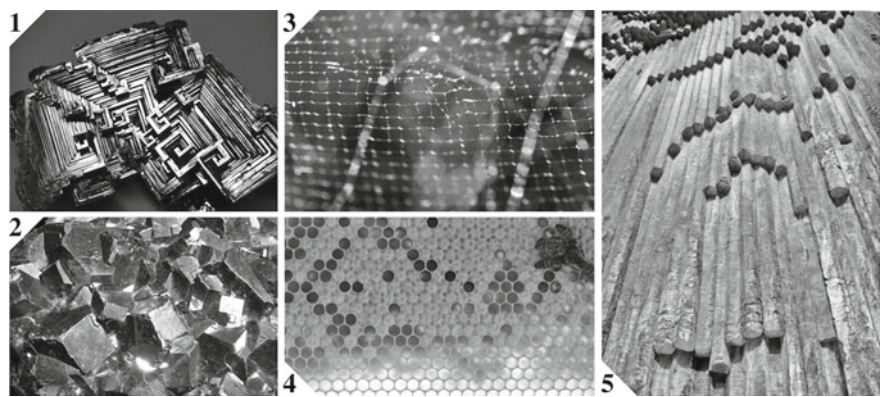
## 2.4 CASS in Regular Tessellations

There are three regular, also called “Platonic” tessellations: square, triangular and hexagonal. The symmetry group of regular tilings is transitive on the tiles. That is, they are homogeneous with respect to vertexes, tiles and edges and are strongly edge-homogeneous [9]. This is equivalent to an edge-to-edge tiling by congruent regular polygons. Architectural practice has been using this property since antiquity. In early 17<sup>th</sup> century Kepler has conducted its mathematical systematization in [21].

CAs have been applied in all three regular tessellations, in vast majority – in the square one. The examples of CAs implemented in other types of grids: computational universality of an 8-state triangular reversible partitioned CA has been presented in [19]; life-like rules in Moore’s neighborhood in triangular grid have been investigated in [4]; the effect of simple memory on a particular reversible, structurally dynamic CA in triangular tessellation has been investigated in [1]. For an overview of CAs, especially so-called Game of Life (GOL) in hexagonal, pentagonal and triangular grids see [6]; for the corresponding interactive demonstration see [5]. Modest-size GOL examples in triangular, square, and hexagonal planar topologies and hierarchical hexagonal grid on the sphere have been presented in [22]. The motivation there was to model bio-geographical, ecological and epidemiological processes on

**Table 2.3** Platonic tessellations in the context of design. \*For given CA, all types of modules are based on identical TRs and “CA-logic.” However, in some cases, they require shapes of two types

	Occurrence in nature	Occurrence in design	Visual attractiveness	CA shading
			Impact on the pattern	
□	Square grid occurs extremely rarely in macro-scale in nature (e.g.: bismuth and galena crystals, cobwebs of <i>Cyrtophora citricola</i> [31])	Rectangular or square meshes are prevalent in built environment	Not particularly attractive	The easiest to apply: all modules are identical
			Neutral	
▽	Regular triangular meshes do not occur naturally in macro scale	Triangles are the only polygons with the property of planar rigidity [25]. Although it is particularly useful in architecture and engineering, triangular grid is relatively rare in the built environment	Relatively interesting	Some CASS require two types of modules*
			Strong	
⬡	Hexagonal grid occurs in macro-scale in nature more often than other types of tessellations (e.g.: honeycomb, basalt columns). Therefore it carries certain visual organic appeal	Since it is the only regular tessellation without single points of contact, the patterns appear as the most coherent. Nevertheless, it is relatively rare in human design	Very attractive	Some CASS require two types of modules*
			Minimal	

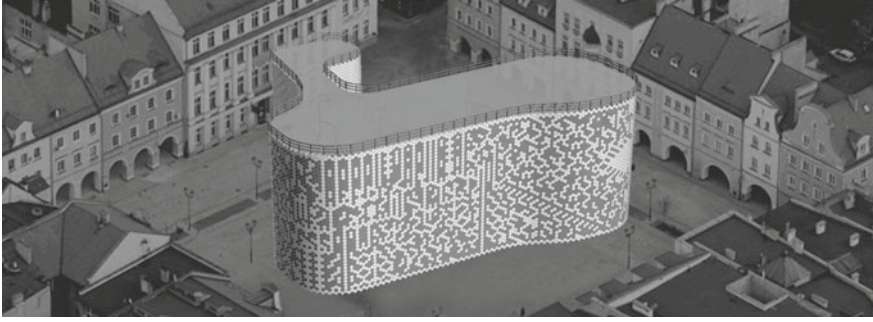


**Fig. 2.8** 1 Bismuth crystal, photograph © amazingrust.com; 2 Galena crystals, © irishminerals.com; 3 *Cyrtophora citricola* cobweb, © vi.wikipedia.org; 4 Honeycomb, © oldvan.com; 5 Basalt columns, © Colin Carlaw



**Fig. 2.9** A visualization of CASS in square tessellation

the globe. Since hexagonal lattice is free from spurious symmetries of the square grid it has been implemented for predicting the spreading of wildfire in [43]. GOL on the surface of geodesic sphere, with all triangular facets, has been studied in [46]. For a corresponding animation see [45]; and for a corresponding interactive demonstration of ST CA on icosahedral geodesic sphere see [56]. Table 2.3 collects the properties of regular tessellations in the context of design.



**Fig. 2.10** A visualization of CASS in hexagonal tessellation



**Fig. 2.11** A visualization of CASS in triangular tessellation

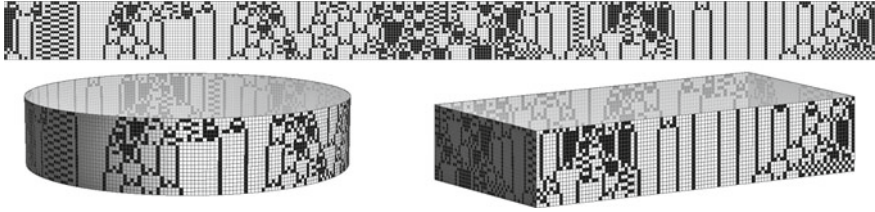
Figure 2.8 shows examples of naturally occurring regular tessellations in macro-scale.

Examples of CASS in the: square, hexagonal, and triangular tessellations are visualized in Figs. 2.9, 2.10, and 2.11, respectively.

## 2.5 Definitions of CASS Parameters

This section presents a number of notions pertaining to CA and terms which are useful in the CASS context. For simplicity, they are formulated and illustrated with two-color (2C) CA in square grid.





**Fig. 2.12** PBC: A  $300 \times 20$  array of  $CA_{SH}$  wrapped around a cylinder and a cuboid

### 2.5.1 Boundary Conditions

One of the factors influencing CAs is the type of boundary conditions (BC). The degree of this influence is related to the neighborhood size, TRs, and proportions of the array of cells. The most common type are the periodic boundary conditions (PBC), as shown in Fig. 2.12. For an interactive demonstration see [55]. In the visualizations the general two-color (2C) one-dimension (1D) range-two (r2) CA  $\{3818817080, 2, 2\}$ ,  $CA_{SH}$  for short, originally proposed in [52] for CASS is used.

If a single CASS is to be applied on all façades, as shown in Fig. 2.12, PBC is the most natural choice. In reality, however, the required level of opacity depends on the position of a façade surface in relation to the sun. In such a case certain parts of array require different pattern density. Moreover, relatively few buildings are cylindrical, and in most of prismatic buildings the façades are designed separately – so would be the shading systems. In the case of a flat single-face façade, PBC is not acceptable since the pattern at the very ends of CA array are highly influenced by the opposite ends. This causes disturbing patterns from the observer's perspective, as illustrated in Fig. 2.13.

Although reflective boundary condition (RBC) seem a rational choice, in practical applications it would require special types of cells, therefore was not considered. The remaining commonly used type is fixed boundary condition (FBC) which is realized by assigning constant values to the boundary cells. For 2C CA, as in this case – 0 or 1 s. Since the range of  $CA_{SH}$  is 2, there are two boundary cells on each side. Four symmetric cases with 0 and 1 s in two boundary cells are shown in Fig. 2.13. In certain cases  $FBC_{01}$  and  $FBC_{10}$  produce slightly different patterns, however, both are acceptable. Since they are symmetrical,  $FBC_{01}$  was arbitrarily selected and applied in all further examples.

### 2.5.2 The Sequence of Initial Conditions (SIC)

The pattern on CASS remains still until the next alteration in IC. Such changes to IC form a sequence (SIC).





SIC\* using order-based representation (OBR) [16]. A sequence of  $n$  ICs is represented by a list of integers of length  $n - 1$ . Each integer is the “distance to the most recently added 1 among the available positions.” Thus SIC\* is a “sequence of the leap lengths where consecutive 1s are located.” The positioning is done from left to right. Since in the very last step there is only one available position, it is dropped. The detailed explanation of this encoding method can be found in [57]. For CASS displaying patterns resulting from this particular SIC see Fig. 2.38.

### 2.5.3 Quantitative Evaluation of a CA Pattern

The most fundamental quantitative measurement of a CA pattern is the “pattern grayness,” in other words, the average pattern density.

**Pattern grayness** is the ratio between the number of black (opaque) cells (with values equal to 1) to the total number of cells in the array.

A façade with all white cells (0s) is considered as completely transparent (grayness = 0); all black cells (1s) form a completely opaque array (grayness = 1).

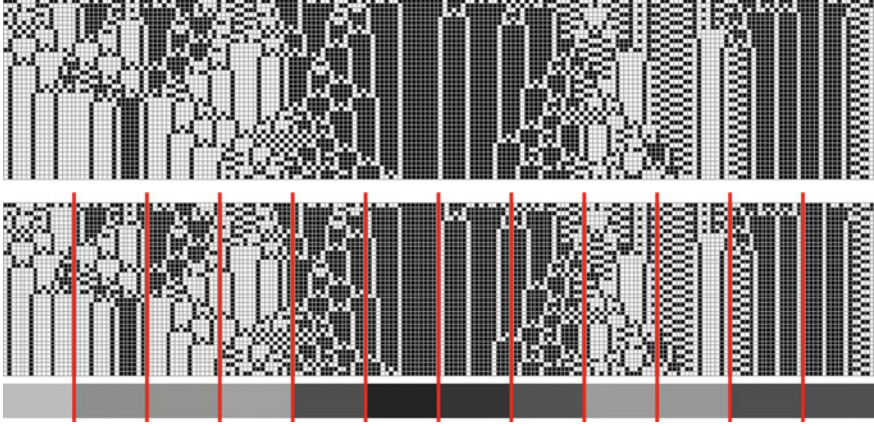
Pattern grayness applies to any sub-matrix, row of cells or group of cells.

For an interactive demonstration see [51]. Grayness function (GF) links the grayness of IC with the grayness of the entire CA array. In other words:

- **Grayness function** (GF) is a series of graynesses for consecutive ICs.
- For CASS, GF should be monotonic, ideally proportional, i.e., the grayness of the entire array should be proportional to the grayness of IC.
- GF should render values from the entire range from 0 to 1.

Such GF ensures that the average opacity of the array (i.e., grayness) can be fully controlled. For an interactive demonstration see [30].

**Grayness function error** (GFE) is the difference between the ideal (proportional) GF and the GF of a particular CA at given SIC.



**Fig. 2.15** From the *top* an example of a single, unevenly distributed,  $CA_{SH}$  pattern on  $192 \times 40$  array. In the *middle* the same pattern divided into 12 vertical stripes of width  $r = 16$ . On the *bottom* the corresponding graynesses of the sub-arrays, which ideally, should have the same level of gray. This would be equivalent to the uniform pattern distribution over entire  $192 \times 40$  array

For a single CA array  $A$  at the given IC it is expressed as:

$$GFE[A] = \frac{\sum_{w=1}^W \sum_{h=1}^H a_{h,w}}{WH} - \frac{\sum_{w=1}^W a_{1,w}}{W} \quad (2.2)$$

where  $H$  and  $W$  are the height and width of an array (façade) respectively.

GFE for SIC is the result of the summation for all ICs:

$$GFE = \frac{1}{F} \sum_{i=1}^F |GFE[A_i]| \quad (2.3)$$

where  $F$  is the number of façades' states which equal to the length of SIC,  $A_i$  is the  $i^{th}$  CA array (for the  $i^{th}$  IC of SIC).

**Grayness monotonicity and pattern distribution error (GDE)** measures combined grayness function monotonicity and the uniformity of the pattern distribution.

In order to evaluate the pattern distribution, it is subdivided into vertical stripes of width  $r$ , as shown in Fig. 2.15.

$$GDE[A, r] = \sum_{k=1}^{\lfloor \frac{W}{r} \rfloor - 1} \frac{\left| \frac{1}{Hr} \sum_{x=kr}^{(k+1)r} \sum_{h=1}^H a_{h,w} - \frac{1}{W} \sum_{w=1}^W a_{1,w} \right|}{\lfloor \frac{W}{r} \rfloor} \quad (2.4)$$

where  $A$  is a CA array (façade),  $H$  and  $W$  are the height and width of the array, respectively;  $r$  and  $k$  are the width and the number of vertical stripes in which the entire array is subdivided, respectively.

The summation over all CA patterns produced at each IC from the given SIC

$$\text{GDE} = \frac{1}{F} \sum_{f=1}^F \text{GDE}[A_f] \quad (2.5)$$

where  $F$  is the number of façades' states which equals to the length of SIC,  $A_f$  is an  $f^{\text{th}}$  CA array corresponding to the  $f^{\text{th}}$  IC in SIC.

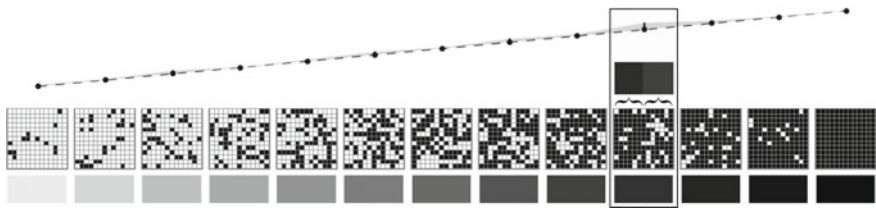
If all  $r$ -wide sub-arrays of the given CA pattern have the same graynesses, the values of GDE and GFE are equal. For further details see [59].

Randomly scattered black cells over entire array are an example of a good pattern distribution. Figure 2.16 shows 13 square arrays with patterns that mimic  $CA_{SH}$ : the very top row ( $13 \times 1$  array) has randomly distributed 1, 2, 3 . . . 13 black cells – which is analog to IC. The rest of array contains the same ratio of black cells as the top row. For example in the first pattern – there is a single black cell out of 13, which is approximately 8%. The rest of array ( $13 \times 12$  cells) has randomly distributed 12 (= approx.  $13 \times 12 \times 0.08$ ) black cells.

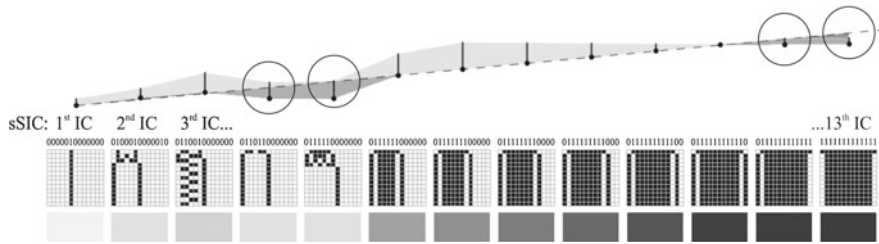
Random patterns (RPs) cannot be produced by any r1 or r2 1D CA, and are introduced as a reference for the actual SICs.

On the other hand, for  $CA_{SH}$  a nearly perfect transition of average density from 0 to almost 1 can be constructed simply by adding consecutive 1s in ICs from the sixth IC onwards. The sequence of the first five ICs that give the minimal GFE were found by trial-and-error. However, in this **straightforward sequence of the initial conditions** (sSIC), for the fourth, fifth and the last two ICs, GFE is non-zero regardless of the width of an array, as shown in Fig. 2.17. sSICs have minimal GFE values achievable by  $CA_{SH}$  for arrays wider than seven cells.

Figure 2.18 compares GFEs of sSICs with SICs generated by random search (RS). RS produces solutions of rather poor quality, but it is a simple and quick comparative evaluation method. Although sSICs have very low GFE values and can be generated



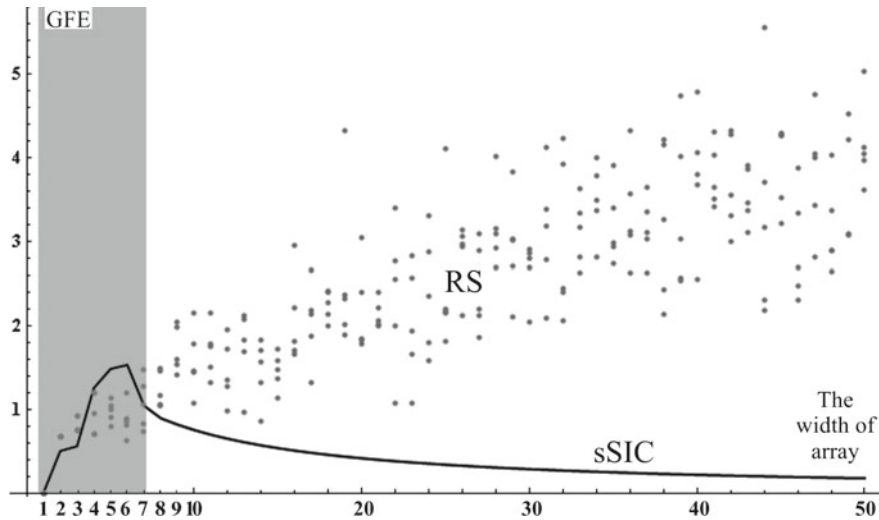
**Fig. 2.16** A sequence of RPs as a referential, highly uniform pattern distribution. The width of the vertical stripe  $r = 6$ .  $\text{GDE} = 0.293$  ( $\text{GFE} = 0$ ). At the 10<sup>th</sup> step the *right part* of the array has more white cells than the *left part*. This corresponds to the small peak in the GDE plot



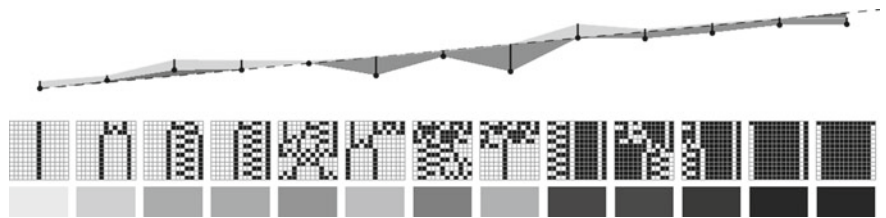
**Fig. 2.17** sSIC for  $13 \times 13$  array of cells. From the *top* (1) GF: the *dashed line* indicates the referential proportionality, *dark gray* filling indicates the differences from that line – too high density is over and too low density is under it. *Black dots* indicate the value of the average density of each  $CA_{SH}$  pattern. *Light gray* filling indicates GDE. (2) The sequence of  $CA_{SH}$  patterns. (3) The *gray levels* equivalent to the grayness of the  $CA_{SH}$  patterns shown above. *Circles* indicate the fourth, fifth, the second last and the last ICs where the grayness of the array is too low. For this sSIC,  $GFE = 0.6$ ,  $GDE = 2.17$

for any size of array at very little computational cost, their  $CA_{SH}$  patterns are not visually attractive and are poorly distributed.

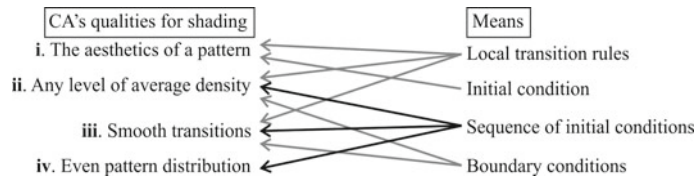
In practical daylight control system, uneven shading distribution may cause unwanted differences in the illumination levels, that is glare in one part of the interior and insufficient levels of light elsewhere. A simple fully controlled mechanism producing vertical or diagonal stripes with good pattern distribution is easily imaginable. However, the aesthetic value of such solution is very low. Figure 2.19 shows



**Fig. 2.18** The *black line* shows GFEs for sSICs. For arrays wider than seven cells sSICs have the smallest possible values of GFE achievable by  $CA_{SH}$ . The *gray dots* indicate GFE values of SICs by RS – five for each array of widths ranging from 1 to 50



**Fig. 2.19**  $CA_{SH}$  at randomly generated SIC is more suitable for shading than the corresponding  $sSIC$  from Fig. 2.17 for the following reasons: (a) the patterns are visually more interesting, (b) GF is almost monotonic (although worse than for  $sSIC$ ), (c) the patterns are rather evenly distributed.  $GFE = 1.10059$ , which is higher than for the corresponding  $sSIC$ . However,  $GDE = 1.56$  is lower



**Fig. 2.20** Although CAs are simple in principle, due to their dynamic nature, their behavior is a result of a rather complex network of relations

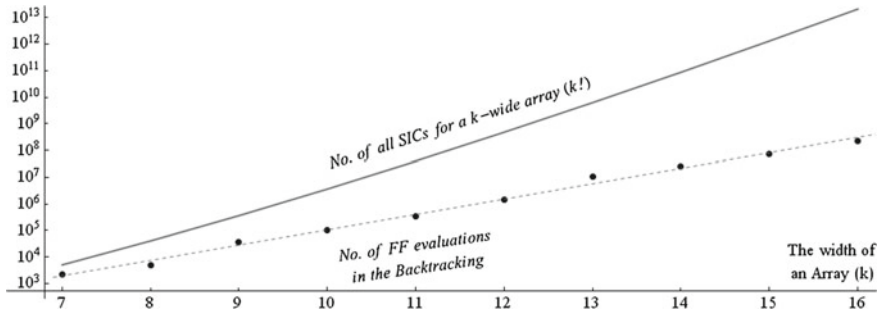
randomly generated SIC which has higher (worse) GFE for  $CA_{SH}$  than the corresponding  $sSIC$ , but producing a sequence of much more interesting and more evenly distributed CA patterns.

In conclusion, CA for shading must be

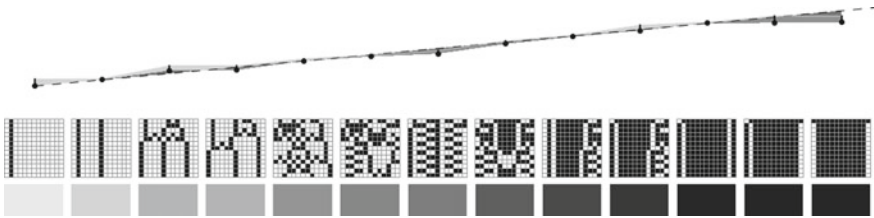
- Visually interesting, i.e., manifesting emergent behavior (Wolfram class 4)
- Controllable, i.e., having sufficiently low **GDE**.

2.6 Optimization of  $CA_{SH}$

The parameters and conditions that influence CA patterns form simple principles, but effectively quite complex network of relations, which most importantly, usually can not be clearly disjointed. The scheme in Fig. 2.20 shows the most evident relations among the desired qualities and the available means of optimizing a CA pattern for shading. The sequence of initial conditions (SIC) is particularly important for CASS. Further in text, a proper CA for shading is equivalent of its proper local transition rules (TRs). The following subsections describe the specific aspects of CASS optimization.



**Fig. 2.21** The number of computations in BA grows substantially slower comparing to the number of all possible solutions. For larger arrays, however, the number of computations makes the method unrealistic, it takes almost four days for uncompiled *Mathematica* code to complete the search for a 16-cell wide array on an Intel Core2 Duo CPU 6550 2 × 2.33 GHz PC

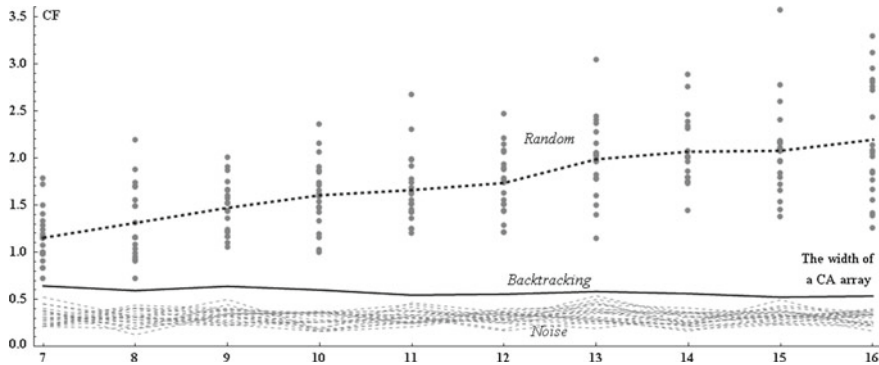


**Fig. 2.22** The ideal solution for the 13 × 13 array found by backtracking. GDE = 0.58 (indicated by light gray in the top plot) and GFE = 0.42 (dark gray in the top plot)

### 2.6.1 Ideal SIC by Backtracking

The number of all possible SICs grows as factorial of the number of cells in the array width. Therefore the evaluation of each candidate solution, i.e., the intensive search, is unrealistic for any problem of practical size. However, so called backtracking algorithm (BA) [23], significantly reduces the number of necessary computations and allows to find the ideal solution for modest size cases, as illustrated in Fig. 2.21.

BA is a refinement of the exhaustive strategy, and systematically searches for a solution among all available options. The solutions are represented by vectors  $(v_1, \dots, v_m)$  and the entire domain of solutions is traversed by a depth-first manner. The algorithm starts with an empty vector. At each stage it extends the partial vector with a new value. Upon reaching a partial vector  $(v_1, \dots, v_i)$  which is worse than the best-so-far partial solution, the algorithm backtracks by removing the trailing value from the vector, and then proceeds by extending it with alternative values. Although this meta-heuristic guarantees to find the ideal solution in a bounded amount of time, depending on a case, the computation time varies substantially. Figure 2.22 shows the ideal solution for the 13 × 13 CA array.



**Fig. 2.23** The comparison of CF values for RS, BA and RP. From the top: (1) RS: 20 randomly generated SICs indicated by *gray dots* with mean values shown with *black dashed line*, (2) BA: the ideal SICs indicated by the *black solid line*, and (3) RP: 20 randomly generated sequences of “noisy” patterns, analog to what is shown in Fig. 2.16

Figure 2.23 visualizes the relationship between SICs by random search (RS) and backtracking algorithm (BA) and random (“noisy”) patterns (RPs).

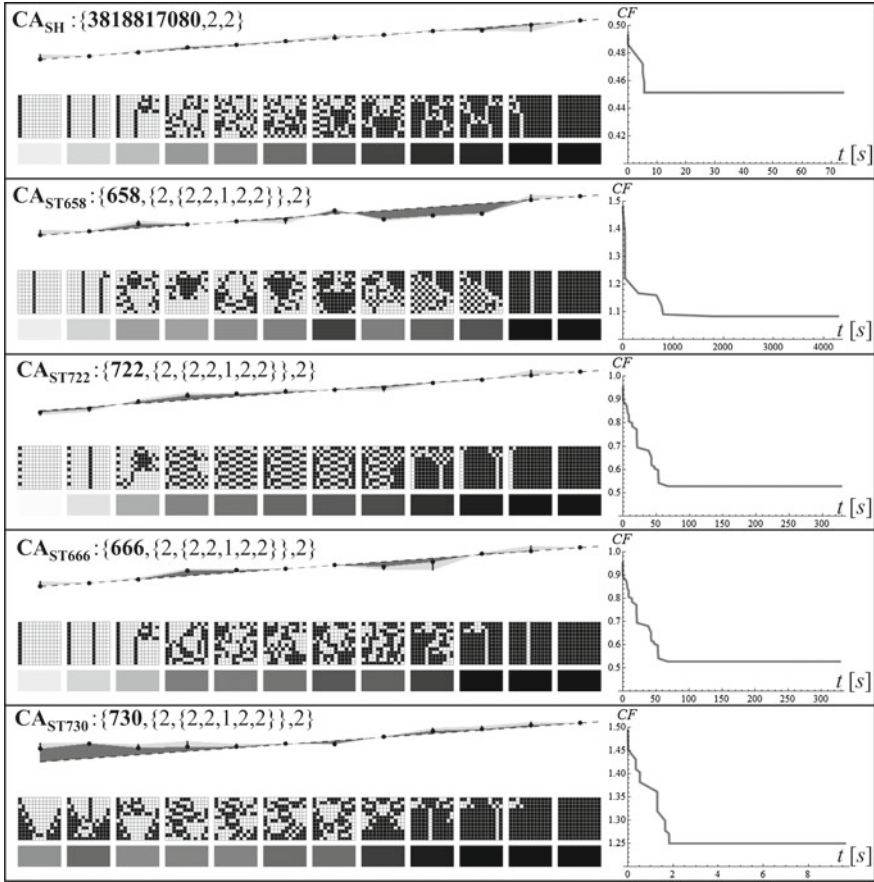
In order to evaluate the effectiveness of backtracking, let us consider a family of CAs on a  $12 \times 12$  array. There are  $12! = 479,001,600$  possible SICs for this array. In the worst, although extremely unlikely case the algorithm would evaluate all of the candidate solutions, which would take 3.4 days on an Intel Core i7–2640M CPU @ 2.8 GHz. In the experiment which results are presented below, the longest measured time was over 71 min with 6,863,516 GDE evaluations and the shortest – 9.6 sec with 15,080 GDE evaluations. The results are collected in Fig. 2.24. The types of automata shown there are explained further in text.

### 2.6.2 Implementation of Evolutionary Algorithms (EA)

In any realistic-size problem, the exhaustive search for ideal solution of the entire domain is impractical. Therefore, a population-based search for at least very good solutions is natural. Evolutionary algorithms (EA) apply the concepts based on the phenomena observed in Nature and use the nomenclature characteristic to the evolution of species. A set of potential solutions is called a *population*, the encoded parameters of a solution is called a *genotype*, a random change of these parameters is a *mutation*, the creation of a new solution by recombination of parts of two selected solutions (parents) is a *crossover*, and the process of choosing the solutions for the crossover is called a *selection*.

Evolutionary algorithms (EAs) are well established methodologies which have been successfully applied for a number of CA-related problems, e.g.: automated design of CA-based complex systems [40], evolving a nonuniform CA where each



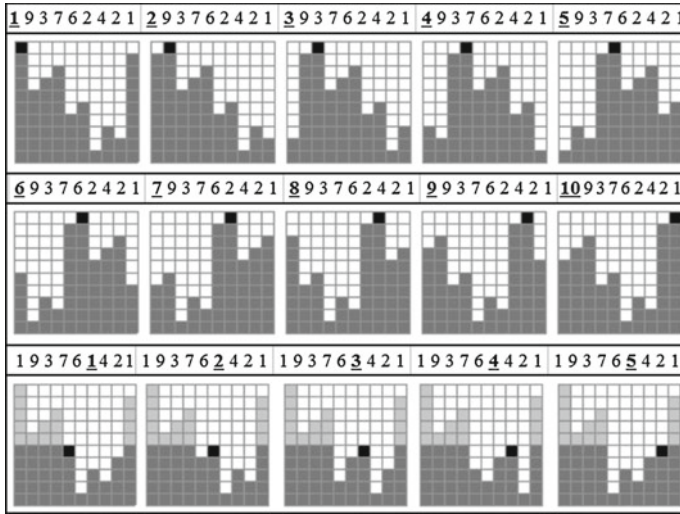


**Fig. 2.24** BA results: the best possible transitions from transparency to opacity of a  $12 \times 12$  array with  $CA_{SH}$  and its four ST siblings. The *dotted line in the left part* indicates the perfect transition of the average density. *Dark gray filling over and under that line* indicate excessive and insufficient average densities of pattern, respectively. The width of vertical stripes:  $r = 6$ . *Light gray fillings* indicate unevenness of pattern distributions. Respective BA convergences are shown on the *right*

cell in the lattice does not use the same rule set [12]; finding Wolfram class 4, that is complex rules [7]; density classification task [28] and the parity problem [26, 49]; discovering and designing cell-state transition functions, where CA are designed to satisfy certain global conditions [13], etc.

### 2.6.2.1 Encoding of SIC

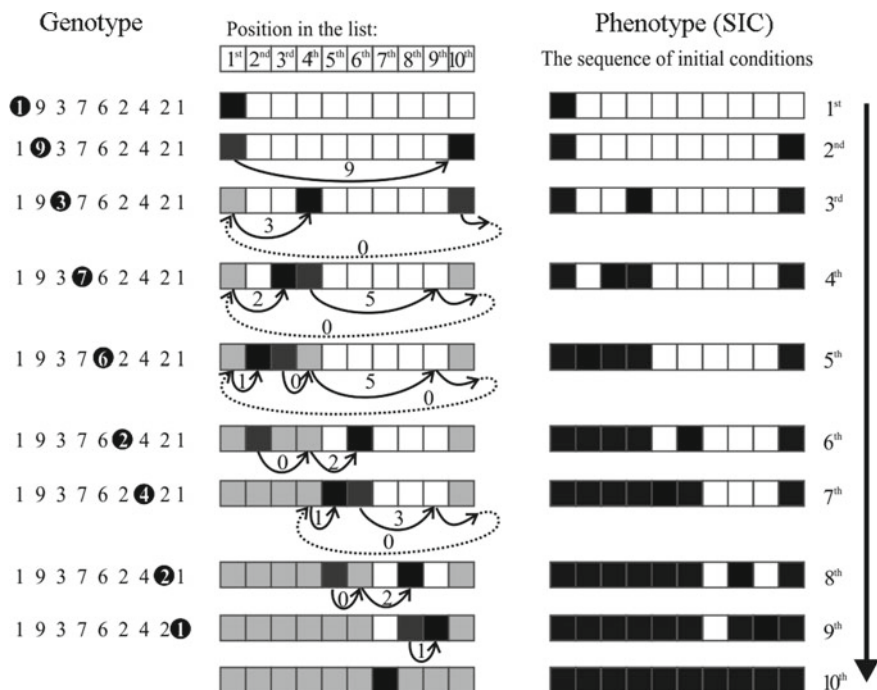
In order to apply EA, it is necessary to encode each candidate solution (phenotype), into a genotype, i.e., a list of symbols (usually integers). SICs can be encoded in



**Fig. 2.25** All possible values (alleles) of the 1<sup>st</sup> and 6<sup>th</sup> genes in a 10-cell wide array. The  $i^{th}$  gene can have  $k - i + 2$  possible values, where  $k$  is the length of the genotype ( $k = n - 1$ ), and  $n$  is the length of IC. *Top row* the gene at the 1<sup>st</sup> position (locus) can have 10( $= 9 - 1 + 2$ ) values. *Bottom row* the gene at 6<sup>th</sup> locus can have 5( $= 9 - 6 + 2$ ). Altered genes are underlined and indicated in *black* in the corresponding SIC charts

a number of ways. Here, the order-based representation (OBR) [16, 37] is used. A sequence of  $n$  ICs is represented by a list of integers of length  $n - 1$ . Each integer is the “distance to the most recently added 1 among the available positions.” The lists have periodic boundaries. In the actual IC for a CA, which is a binary list, 1 can only replace a 0. Thus, the replacement is allowed at an “available position.” In a case when a given position is already occupied by a 1, the next “empty” (0) slot is selected. Such a genotype of SIC becomes a “sequence of the leap lengths where consecutive 1s are located.” The positioning is arranged from left to right. In the last step there is only one position, thus it is dropped to shorten the encoding. In this type of genotype, possible genes values, so-called *allele* are constrained. Their values depend on *loci*, i.e., the positions in genotype strings. Initially, there are  $n$  available positions in the list, thus a gene can have  $n$  possible values. In next step, since one *locus* is already occupied, the allele value is limited to  $n - 1$ , and so forth, up to the last gene in the genotype, which can only have two values:  $n - ((n - 1) - 1) = 2$ . Two examples of all possible alleles are shown in Fig. 2.25.

Decoding of a sample genotype is explained in Fig. 2.26.



**Fig. 2.26** The decoding of a genotype {1, 9, 3, 7, 6, 2, 4, 2, 1} into the phenotype - SIC. Center column: the available positions are shown in white; *dark* and *light gray* indicate the position taken in the previous step - the one from which the next leap is counted, and the occupied cells in the earlier steps, respectively; *black* indicates the final position at the current step. *Dashed arrow* is not an actual leap, but illustrates the periodicity of the boundaries

### 2.6.2.2 Genetic Operations on SIC\*

Here, the genetic operations are performed directly on SIC\*s (genotypes), i.e., OBR-encoded SICs. The main advantage of OBR encoding of genes is the simplicity of recombination which always produces feasible SIC\*s that represent allowable SICs.

- One-point crossover (OPX): a locus on the genotypes of both parents is randomly selected, and the gene segments are exchanged between them;
- Uniform crossover (UX): the genes for the offspring are drawn randomly from both parents.

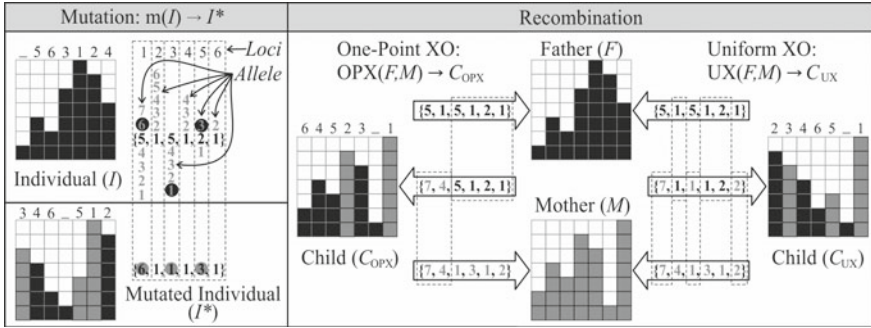


Fig. 2.27 Examples of basic genetic operations on SIC\*s

The operation of mutation is controlled by two parameters:

- Mutation radius  $m_r$ , i.e., the maximum difference between the original value and the new value of the mutated gene;
- Mutation intensity  $m_i$ , i.e., the maximum number of genes to be mutated. Here  $m_i$  is normalized to the genotype length.

Figure 2.27 illustrates mutation and two kinds of recombination OPX and UX.

### 2.6.2.3 Evolution Strategy (ES): No Crossover, Intensive Mutation

At first, the simplest EA setup was implemented, where the genetic operations are limited to an intensive mutation. This type of algorithm is known as evolution strategy (ES) [36].

The general procedure of ES is based on iterations of the following loop:

1. Check the stop criterion. If it does not hold, proceed;
2. Evaluate all individuals (solutions) in the population;
3. Randomly select individuals for the next generation favoring “better” ones;
4. Mutate selected individuals and goto 1.

In the subsequent experiments so-called “tournament” selection mechanism has been used: at each draw for the next generation, unique pairs of candidates are randomly pre-selected, and the better is chosen. In order to maintain the constant size of the population, this process is done twice. The first experiment for seven arrays: from  $7 \times 7$  to  $13 \times 13$  has been performed with the following parameters values:

- The width of the vertical stripe, as in Formula (2.4),  $r = \text{Round}[\frac{W}{2}]$ , where  $W$  is the width of a CA array;
- Mutation parameters: intensity  $m_i = 1$ , radius  $m_r = 1$  (no crossover);
- Population  $p = 100$ ; 10 trials with 20 generations and 10 with 40 generations.

The results are shown in Fig. 2.28 and compared to BA (ideal solutions).

Next experiment has been performed for 10 arrays: from  $7 \times 7$  to  $16 \times 16$  with larger population sizes, and the results are compared with the RS method. The parameter values were set as follows:

- The width of the vertical stripe is constant  $r = 6$ ;
- Mutation parameters: intensity  $m_i = 1$ , radius  $m_r = 1$  (no crossover);
- Population  $p = 100$  and 200 for 20 and 40 generations; 10 trials for each setup.

RS has been performed 80,000 times which is equivalent to 10 trials of ES with population  $p = 200$  at 40 generations ( $80,000 = 10 \times 200 \times 40$ ). The results are shown in Fig. 2.29 and compared with the ideal solutions found by BA.

As Fig. 2.29 indicates, the evolutionary approach based on mutation gives better results than RS. This positive tendency increases with the width of the array. The influence of recombination operation on the performance of analogous optimization process is examined next.

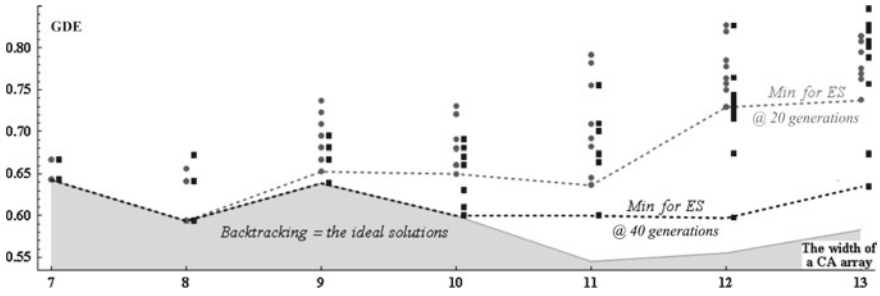
#### 2.6.2.4 EA with Crossover, and No Mutation

The next experiment has been performed on  $100 \times 100$  cell array at two types of recombination: one-point (OPX) and uniform (UX) crossovers, without mutation. The algorithm is similar to ES, except that in the 4<sup>th</sup> step the mutation is replaced by crossover between selected pairs of solutions. The parameter values have been set as follows:

- $r = 6$ ;
- Recombination types: OPX and UX (no mutation);
- Population  $p = 50$ ; 10 trials for each combination of parameters; 100 generations.

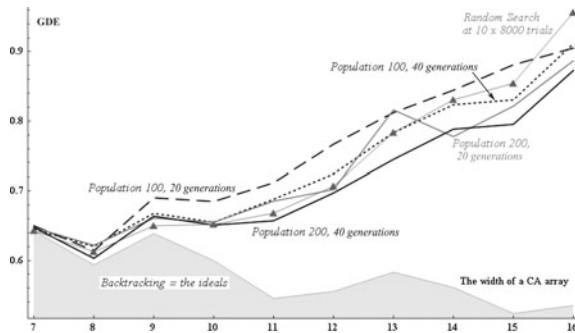
Additionally, ES with  $m_r = 1$ ,  $r = 6$ ,  $p = 50$  at 100 generations in 10 trials, and RS at 50,000 trials which is equivalent to 10 trials of EA with population  $p = 50$  at 100 generations have been performed ( $50,000 = 10 \times 50 \times 100$ ), as shown in Fig. 2.30.

As expected, RS produces the worst solutions. Although the best solution was found by ES (GDE=8.48), the best improvement of the mean value of GDE is observed in  $EA_{OPX}$  setup. However, around the 60<sup>th</sup> generation the standard deviation in  $EA_{OPX}$ , becomes very low, which indicates the degeneration of populations, i.e., the diversity of solutions decreases, which causes no further improvement. The means of 10 trials in each method are compared in Fig. 2.31.



**Fig. 2.28** SICs generated by 20 trials of ES. The gray dots and black squares indicate the results generated at: 20 and 40 generations, respectively. The best results are indicated with dashed lines; gray filling indicates the ideal solutions found by BA. Although the results of ES are rather scattered, they tend to be relatively better at higher number of generations

**Fig. 2.29** Four series of ES with different combinations of parameters: the population size and the number of generations. The mean results from 10 trials are compared with RS (indicated with triangles). Beyond a certain size of array (15 cells) each setup of ES gives better results on average than RS



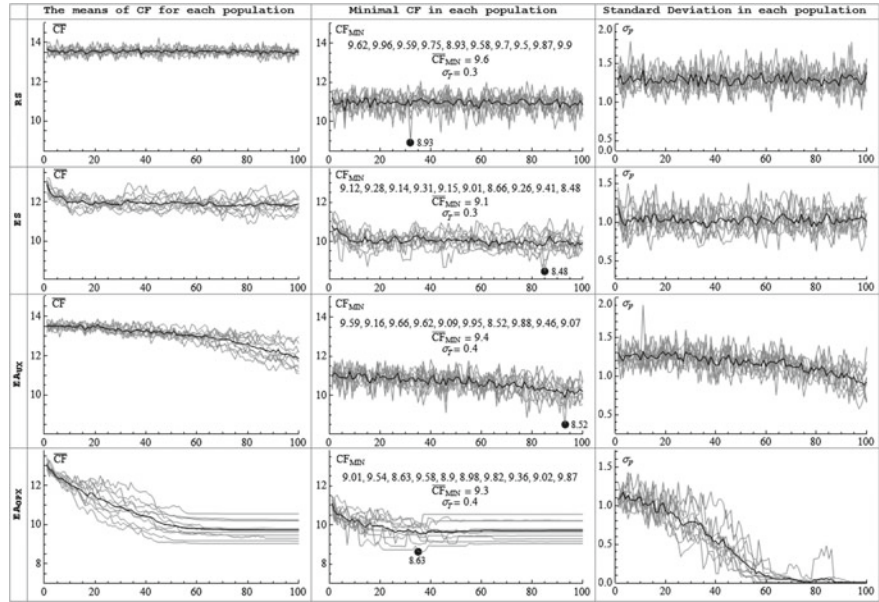
### 2.6.2.5 Regular EA: With Crossover, and Mutation

The final experiment for the same  $100 \times 100$  array has been based on regular EA, i.e., employing both: crossover and mutation. The parameters have been set as follows:

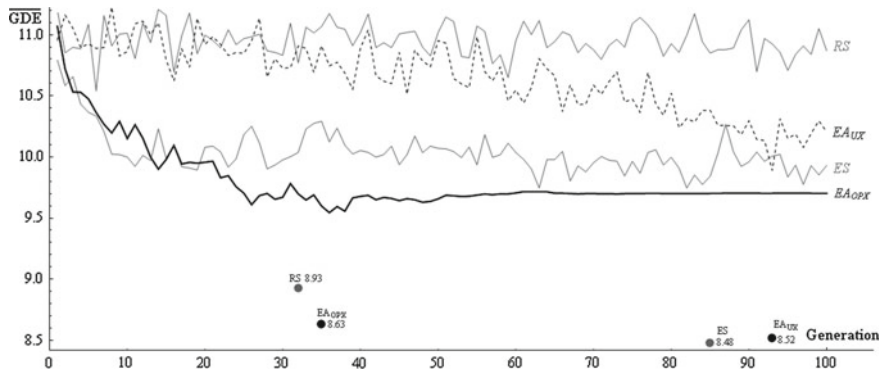
- $r = 6$ ;
- Mutation: intensity  $m_i = 1$ , radii:  $m_r = 0.1$  and  $0.4$ ; recombination: OPX and UX;
- Population  $p = 50$ ; 10 trials for each setup; 200 generations.

Additionally, ES with  $m_r = 1$ ,  $r = 6$ ,  $p = 50$  at 200 generations in 10 trials has been performed. The best result has been found by EA with one-point crossover and mutation rate  $0.4$  ( $EA_{OPX-0.4}$ , for short). Figure 2.32 shows the plot of its GFE and GDE evaluations.

As Fig. 2.32 indicates, at the 33<sup>rd</sup> step, the CA pattern grayness is proper (GFE is nearly 0). However, the pattern distribution is poor (as the peak in light gray



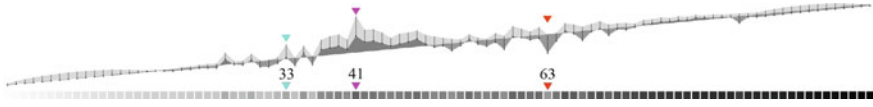
**Fig. 2.30** EA with crossover, and no mutation. The *gray* and *black lines* indicate: 10 trials and their means, respectively. In the center column, for each method, the list of minimal (best) values in each trial, their mean and standard deviation are shown. The best solution for each method is indicated by a *black dot*



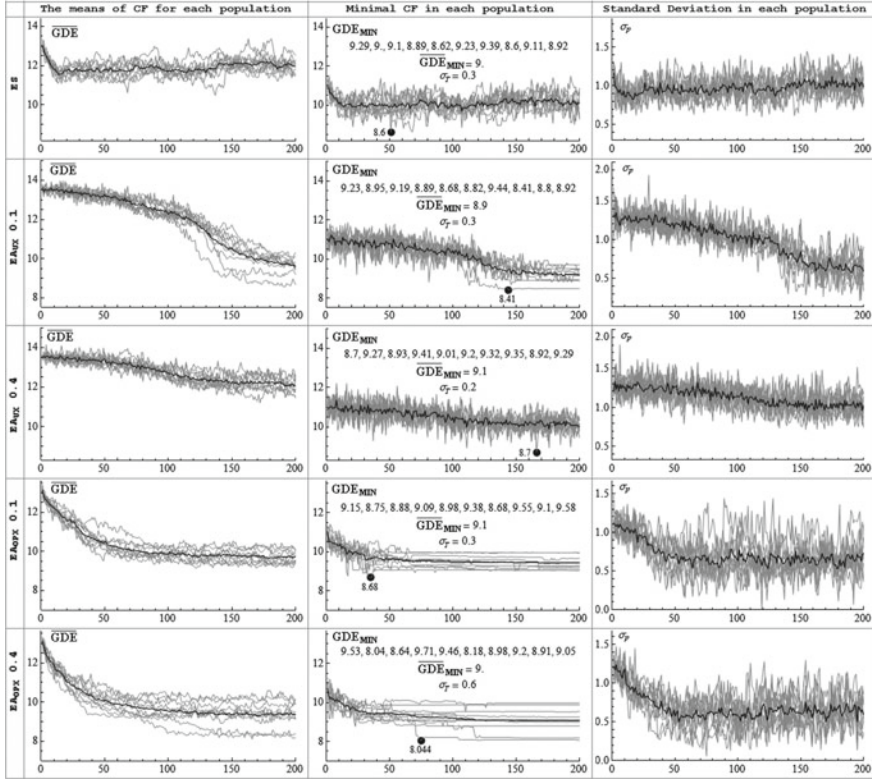
**Fig. 2.31** The comparison of the means of CFs in a population for each generation in 10 trials for four methods. The best solution for each method is indicated by a dot in the corresponding level of *gray*

indicates); at the 63<sup>rd</sup> step the density (GFE) is too low and also the distribution is poor; at the 41<sup>st</sup> step, the density (GFE) is too high and the distribution is poor (the peak in light gray over the peak in dark gray). The results of all experiments are collected in Fig. 2.33.





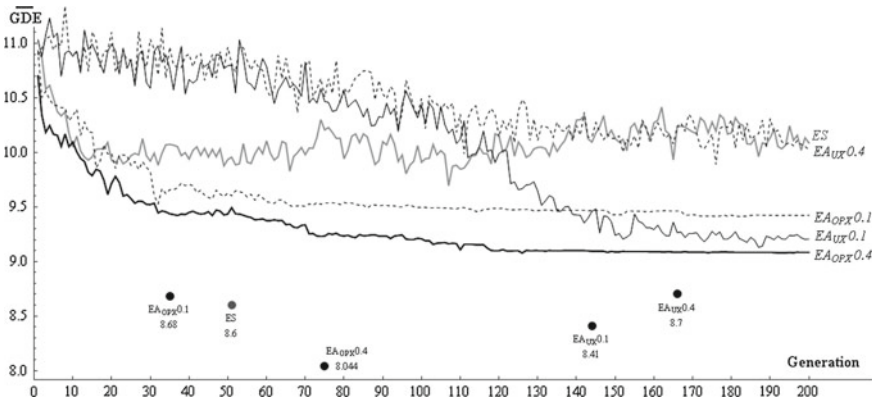
**Fig. 2.32** The best solution produced by  $EA_{OPX} - 0.4$  for the  $100 \times 100$  array. 100 steps from a single to all *black* cells in IC. GFE and GDE are shown in *dark* and *light* gray respectively



**Fig. 2.33** EA with crossover and mutation. The *gray* and *black* lines indicate 10 trials and their means respectively. In the center column, for each method, the list of minimal (best) values in each trial, their mean and standard deviation are shown. The best solution for each method is indicated by a *black dot*

The means of 10 trials in each setup are compared in Fig. 2.34.

The series of CA patterns produced by  $CA_{SH}$  at the best SIC are shown in Figs. 2.35 and 2.36.



**Fig. 2.34** The comparison of the means of CFs in a population for each generation in 10 trials for five methods. The best solution for each method is indicated by a dot in a corresponding level of gray.  $EA_{OPX}$  with mutation intensity 0.4 produced consistently the best solutions

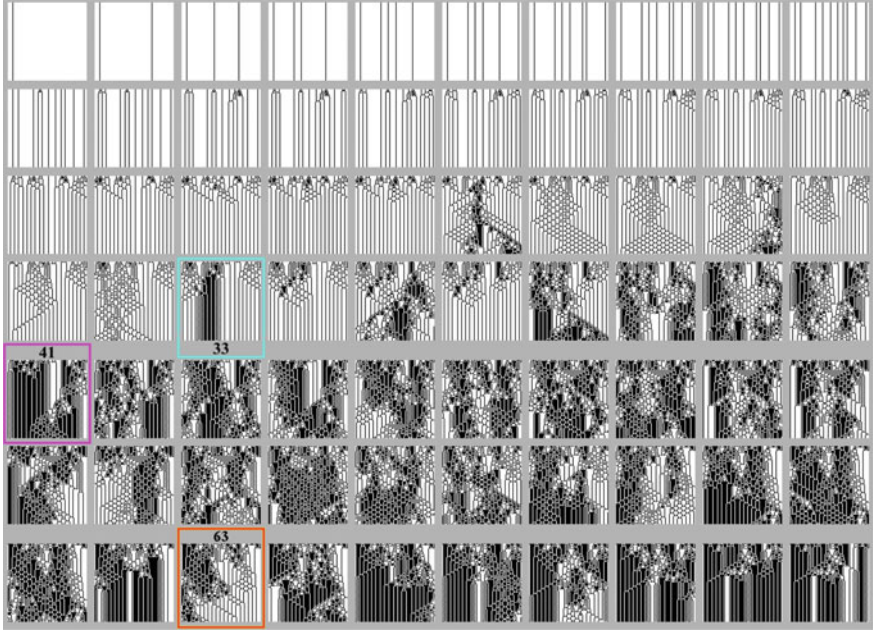
### 2.6.3 Discussion

Although CAs have been studied for well over half of a century, the real-life implementations are still very scarce. CAs draw attention of not only scientists, but due to their intriguing appearance, also designers. The presented methodology produced a good solution for an architecture-related problem that can be considered relatively realistic. As nearly any real-life problem, shading of a building is a multi-objective optimization. Moreover, some of the criteria, just like in many practical problems, are conflicting - in this case a perfectly distributed pattern, that is “noise” or simple dithering is not acceptable from the aesthetic perspective. GDE which serves here as an objective function can be extended to include other criteria and constraints such as opacity thresholds or other specific requirements for the CA<sub>SH</sub> pattern.

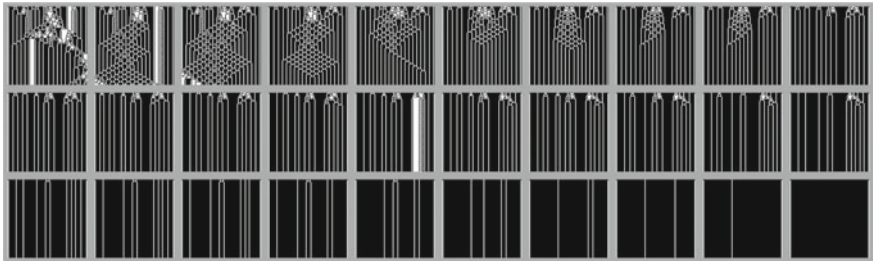
In the course of a number of experiments,  $EA_{OPX}$  with mutation intensity  $m_1 = 1$  and radius  $m_r = 0.4$  produced the best overall result. Table 2.4 lists the main results of all experimental setups. For detailed analysis of the results see [57].

The computation time was rather reasonable, that is 47 min for one trial on an Intel Core2 Duo CPU 6550  $2 \times 2.33$  GHz PC. The experiments have been performed with uncompiled Mathematica code. Thus a substantial speed-up can be easily achieved. Parallelization of this algorithm is also straightforward. Moreover, the problem is well suited for two-level hierarchy of evolutionary computation [49].

Because in the final experiment, the width of the vertical stripe is set to a constant value  $r = 6$ , the number of vertical stripes grow with the width of array, as expressed



**Fig. 2.35** The sequence of the first 70 out of 100  $CA_{SH}$  patterns for the best SIC for the  $100 \times 100$  array. Quite smooth and well distributed transition from 1 % (*top left*) to approximately 70 % opacity (*bottom right*). Three worst patterns corresponding to Fig. 2.32 at steps: 33, 41 and 63 are framed IN: *cyan, magenta and orange*, respectively

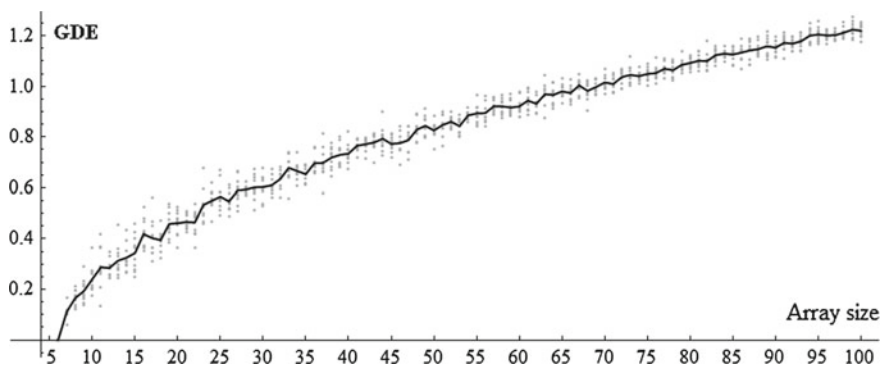


**Fig. 2.36** The sequence of remaining 30 out of 100  $CA_{SH}$  patterns for the best SIC for the  $100 \times 100$  array from approximately 71 % (*top left*) to 98 % opacity (*bottom right*)

in formula (2.4). Greater number of sub-arrays accumulate more differences in pattern distribution among them. As a result, GDE value naturally increases with the width of array as shown in Fig. 2.37.

**Table 2.4** The main results of all experimental setups for CA<sub>SH</sub> in the 100 × 100-cell array. The best results are boldfaced

	ES	EA <sub>UX</sub>		EA <sub>OPX</sub>	
$m_i$	1	0.1	0.4	0.1	0.4
$GDE_{min}$	8.6	8.41	8.7	8.68	<b>8.04</b>
$GDE_{max}$	<b>9.39</b>	9.44	9.41	9.58	9.71
$\overline{GDE}_{min}$	9	<b>8.9</b>	9.1	9.1	9
$\sigma_T$	0.3	0.3	<b>0.2</b>	0.3	0.6



**Fig. 2.37** The increase of GDE for the referential “noisy” patterns (RPs, for reference see Fig. 2.16) from 6 × 6 to 100 × 100 arrays. The width of the vertical stripe is constant ( $r = 6$ ). GDE values for 10 randomly generated “noisy” samples are shown as *gray dots* and their mean values are plotted as the *black line*

## 2.7 One-Dimensional Cellular Automata Applied on Surfaces

In principle, building envelopes (BEs) are in form of surface. Such surfaces are not necessarily planar, but can be considered as two-dimensional (2D). One-dimensional (1D) CAs seem the most practical for such application. It matches the common convention of presentation of 1D CAs, as their evolution history. Each “one-dimensional” CA row corresponds to a time-step in this history, and becomes an initial condition (IC) for the next row, etc. This cascade-like process propagates over entire array of cells.

Use of two-dimensional (2D) CAs also seems intuitive for at least two reasons: the inter-connections among CA cells seem straightforward, and there are many more simple 2D CAs which increases the chances of finding the most appropriate automaton for CASS. However, the major difficulty of such application is the control of the state of 2D CAs. They continuously update all the cells until an equilibrium state is reached. Practically always it leads to a uniform, banal pattern. It is imaginable to “freeze” the array at a certain time-step and not allow it to evolve further. However,

presently, it seems to be a difficult technical problem. Most of the cells turn white (0s) or black (1s), often with artifacts of relatively small areas of the opposite state. So-called “strobing,” i.e., endless state-switching at every time-step is also common, as discussed in [52]. Another major difficulty is the setting of ICs. How to set the initial input to the cells of a 2D array? A possible solution where only cells on the edges are used for the ICs has been proposed in [54]. Nevertheless, this method needs further investigation. Finally, in principle the exact final state of the 2D CA array is difficult or impossible to predict due to the *computational irreducibility*.

However, with the adopted common convention of displaying 1D CA, many of these problems do not occur. Every row in 1D CA displays the state at a certain time-step, and once set it is maintained, which substantially helps to solve the issue of control.

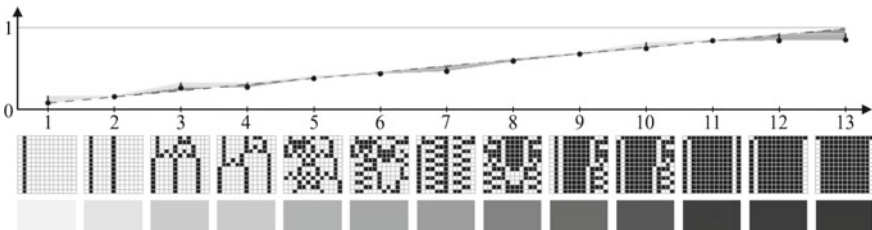
### 2.7.1 Elementary Cellular Automata for CASS

There are 256 2C1Dr1 i.e.: two-color (state) one-dimensional range-one, so-called Elementary Cellular Automata (ECA). However, due to symmetries, the number of fundamentally inequivalent ECAs is 88 [20]. For an illustrative demonstration of CASS based on ECAs see [53].

### 2.7.2 The Original CA for Shading (CA<sub>SH</sub>)

As investigated in [52], none of ECAs meets both “shading criteria,” i.e.: visual attractiveness, and controllability measured by low GDE. Also there, 2C1Dr2 CA code {3818817080,2,2}, CA<sub>SH</sub> for short, has been proposed for CASS.

The first, second, and third values in the CA<sub>SH</sub> code correspond to: the decimal enumeration of the TR outputs, the number of colors (states) and the neighborhood range, respectively. Figure 2.38 shows the ideal SIC for CA<sub>SH</sub> in  $13 \times 13$ -cell array at fixed boundary conditions (FBC, see Fig. 2.13).



**Fig. 2.38**  $13 \times 13$  array CA<sub>SH</sub> at SIC\*: {252175415411} gives the transition from transparency to opacity with the lowest  $GDE = 0.58$  and  $GFE = 0.42$ . The convention as in Fig. 2.17

2.7.3 Four Semi-totalistic Siblings of CA<sub>SH</sub>

As shown in Fig. 2.39, where TRs are rearranged to reveal the symmetries - the neighbor-state transition rules of CA<sub>SH</sub> are almost perfectly symmetrical.

All but one group can be reduced to a much simpler, semi-totalistic (ST) rule set. The encoding convention of ST CAs is:  $\{n, \{k, \{k, k, 1, k, k\}, r\};$  where  $n$  is the decimal enumeration of the TR outputs,  $k$  is the number of colors and  $r$  is size of the neighborhood; the expression in the innermost curly brackets reflects the structure of r2 CA with 1 as the central cell with two neighboring cells on each side. However, the non-totalistic group of rules, indicated by dashed rectangle in Fig. 2.39, can be converted to a single ST rule in four ways, as shown in columns 3–6. As Fig. 2.39 indicates, a much simpler, since it is defined by only 10 TRs - CA{666,{2,{2,2,1,2,2}},2}, CA<sub>ST666</sub> for short, shares certain aesthetic properties with CA<sub>SH</sub> which is defined by 32 TRs. The applicability of all ST siblings of CA<sub>SH</sub> is examined by finding ideal SICs for  $12 \times 12$  arrays by backtracking as described in the next sub-section.

2.7.4 Totalistic Siblings of CA<sub>SH</sub>

After discovering that there are at least a few relatively simple r2 ST CAs that can be appropriated for rather complex task of shading, a natural question arose: are even simpler CAs capable of it? As mentioned above, general 1D2Cr2 CAs are defined by 32 TRs. There are therefore  $2^{32} = 4,294,967,296$  such automata. ST1D2Cr2 CAs are defined by 10 TRs (see Fig. 2.39), thus there are only  $2^{10} = 1024$  such automata. Finally, totalistic (T) CAs, that is T1D2Cr2 CAs are defined by only 6 TRs. An

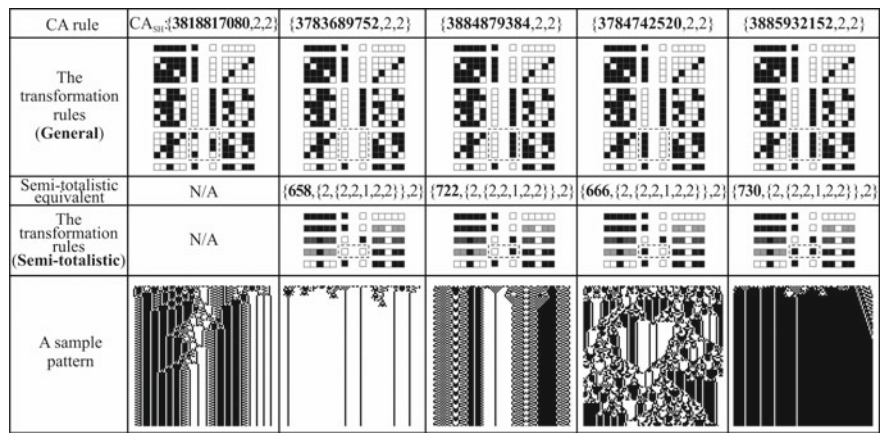
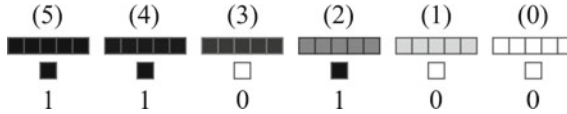
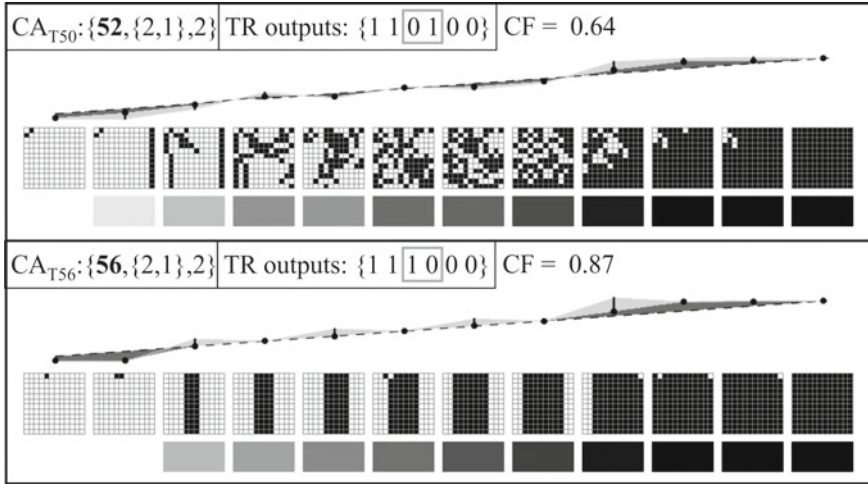


Fig. 2.39 CA<sub>SH</sub> and its four ST siblings





**Fig. 2.40** TR outputs:  $110100_2 = 52_{10}$ ; CA  $\{52, \{2, 1\}, 2\}$  as an example of T1D2Cr2 CA. At each step of CA evolution, the sum of values in all five cells is taken as input. The level of gray is inversely proportional to the averaged value in each cell. For example, values 1 and 0.8 correspond to: 0 and 0.2 levels of *gray*, respectively



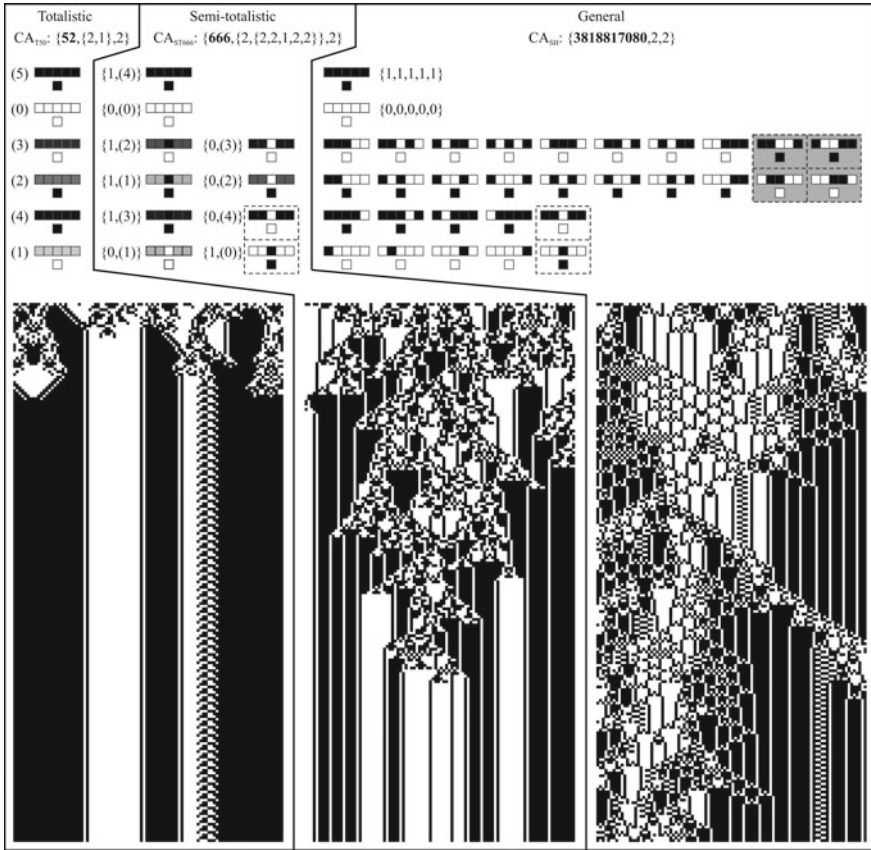
**Fig. 2.41** Two T1D2Cr2 CAs with good basic shading properties:  $CA_{T52}$  and  $CA_{T56}$

example is shown in Fig. 2.40. In order to distinguish the sums from the direct values in cells, the former are bracketed. Are there appropriate automata among merely  $2^6 = 64$  such CAs?

The encoding convention of T1D2Cr2 CAs is:  $\{n, \{k, 1\}, r\}$ , where  $n$  is the decimal enumeration of TR outputs,  $k$  is the number of colors and  $r$  is the range of neighborhood. The best of such automata for shading are shown in Fig. 2.41.

The best two T CAs have similar TRs, with only one pair of outputs exchanged, as indicated by gray rectangles in Fig. 2.41. Notably, the one that produces more complex pattern, that is  $CA_{T52}$  has also better basic shading properties reflected in lower GDE value. Nevertheless,  $CA_{T56}$  does not meet the aesthetic requirements for CASS. TRs and patterns produced by  $CA_{SH}$ ,  $CA_{ST666}$  and  $CA_{T52}$  from the same 100-cell IC at 200 steps are compared in Fig. 2.42. The same Figure also shows that  $CA_{ST666}$  introduces subtle modification to  $CA_{T52}$ , namely exchanges the outputs in two out of ten TRs.  $CA_{SH}$  further introduces only four more changes among 32 TRs.





**Fig. 2.42** The family of three “shading automata.” TRs are rearranged to emphasize their similarities among the automata. The exceptional TRs from totalistic rules are indicated by *dashed rectangles*, the exceptional rules from both totalistic and semi-totalistic TRs are indicated by *gray rectangles*

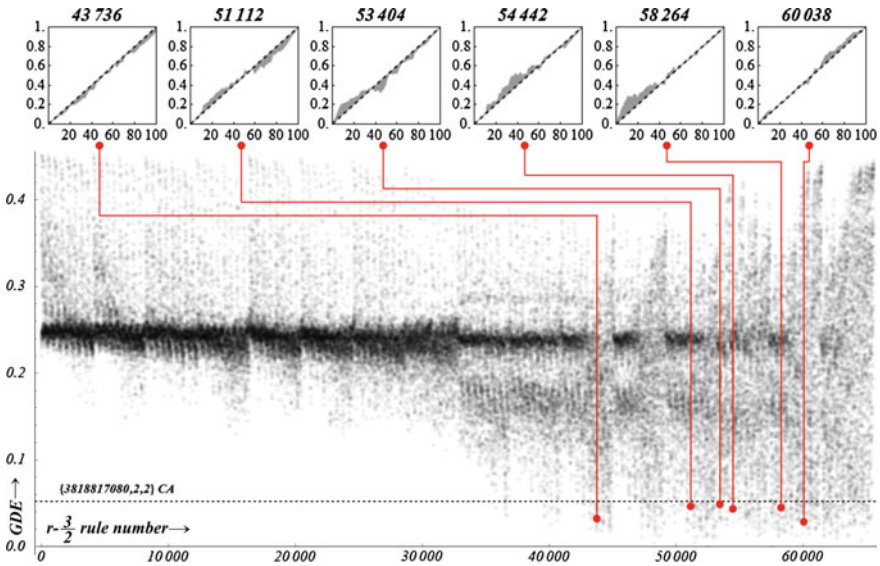
These alterations, although maintain the general characteristics of these cellular automata, introduce certain nuances to their patterns, which makes them “more complex.” Exchanging the output bits of  $CA_{SH}$  in two neighbor-state transition rules indicated by dashed rectangles in Fig. 2.42 so that group of rules “becomes totalistic,” defines  $CA \{3953034792, 2, 2\}$ . This cellular automaton, however, is Wolfram class 2, which is not acceptable for CASS, as it does not produce visually attractive patterns.

### 2.7.5 Half-Distance Cellular Automata

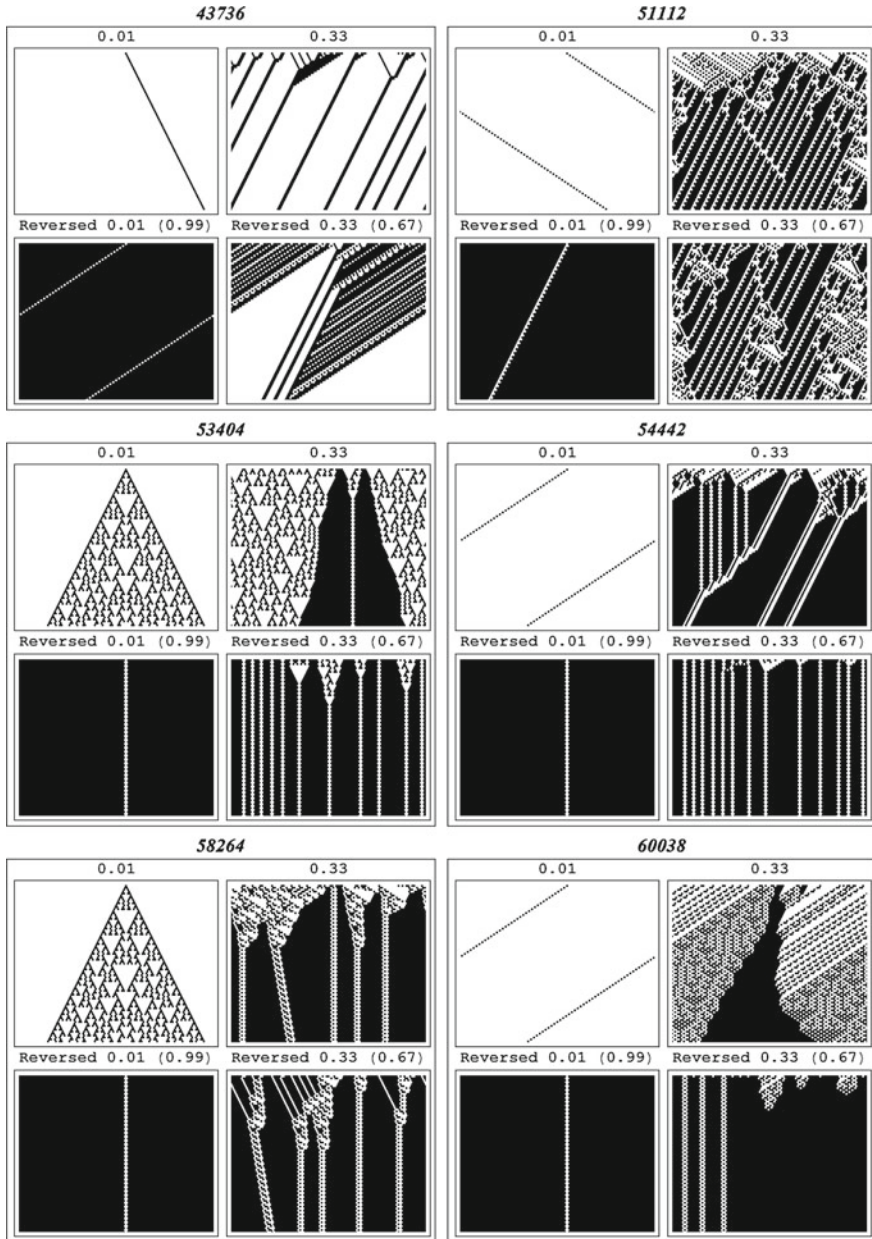
As mentioned in Sect. 2.1: “Cellular Automata,” so-called half-distance rules are created by shifting the successive rows, so the number of input cells becomes even. This type of CAs is especially interesting for CASS, due to its relatively straightforward applicability also in hexagonal and triangular tessellations, as described in Chap. 3: “Polarized Film Shading System in regular grids” in Sect. 3.3.2: “Hexagonal Grid,” and Sect. 3.3.3: “Triangular Grid ( $PFSS_T$ ),” respectively.

There are  $16 \cdot r - \frac{1}{2}$  1D2C such CAs only. None of them is sufficiently aesthetically interesting. However, the number of  $r - \frac{3}{2}$  automata is 65,536. Half of them, i.e., 32,768 are odd-numbered rules, thus unsuitable for CASS. Figure 2.43 shows some examples with low GFE values. In the following examples (Figs. 2.43, 2.44, 2.45 and 2.46) GFE has been calculated once for each CA rule at the same sequence of randomly generated initial conditions. Therefore, the results are rather illustrative than conclusive.

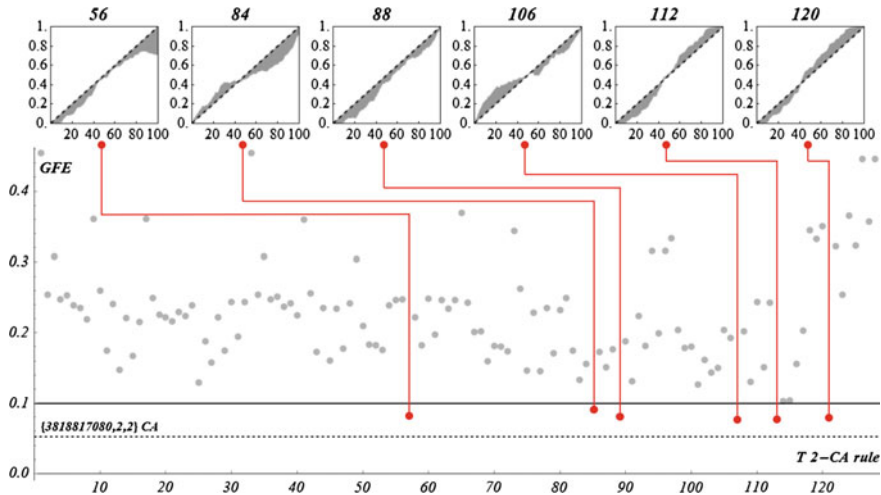
As Fig. 2.43 indicates, many  $r - \frac{3}{2}$  CAs produce pattern of very low GFE values for the given SIC. 720 such automata returned GFE below 0.05, which is even better than  $CA_{SH}$ . Although the pattern distribution has not been considered, some of these automata are visually attractive, as shown in Fig. 2.44.



**Fig. 2.43** On the *bottom* the GFE plot for all  $r - \frac{3}{2}$  CAs. Some of them produce very low GFE values, even below  $CA_{SH}$  ( $\{3818817080, 2, 2\}$  CA), as indicated by the *dotted line*. On the *top* the GF plots for selected  $r - \frac{3}{2}$  CAs. *Dotted lines* and *gray fillings* indicate the referential proportionality, and visualize the differences to it, respectively



**Fig. 2.44** Four CA patterns produced by each of six selected  $r - \frac{3}{2}$  rules: 43736, 51112, 53404, 54442, 58264 and 60038. For each CA rule, the *left* column shows on the *top* and *bottom* coupled (inverted) patterns starting from: a single 1, and single 0, respectively. The *right* columns show on the *top* and *bottom* coupled (inverted) patterns starting from: 33 % rate of 1s, and 67 % of 1s, respectively. The corresponding ICs are the same for each automaton

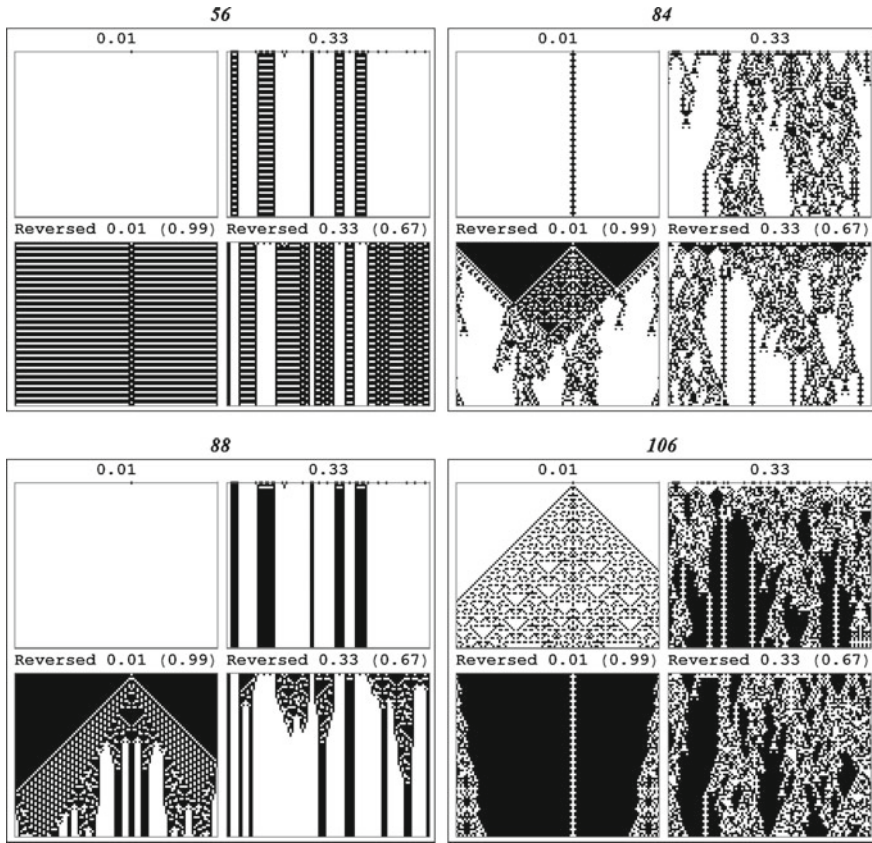


**Fig. 2.45** On the *bottom* GFE values for 128 T 2-CAs at the same randomly generated SIC. Six of these automata show relatively low GFE (*below* 0.1). The referential value for  $CA_{SH}$  is shown as the *dotted* line, and indicates that T 2-CAs have worse shading properties. On the *top* the corresponding GF plots. The graphic convention as in Fig. 2.43

### 2.7.6 Higher Order Cellular Automata

As mentioned in Sect. 2.1: “Cellular Automata,” higher order automata depend on both: the present and past states of their cells. A  $k$ -order automata are a type of *reversible* CAs where the state of cells at time-step  $t$  depend not only on their neighborhood at time-step  $t-1$ , but also on their states at time-steps  $\{t-1, \dots, t-k\}$ . These CAs are especially attractive due to their intriguing properties [2] and relatively straightforward implementation for shading. The number of second-order 1D2Cr1 CAs, or 2-CAs for short, is not manageable for a “manual” search. There are  $2^{2^6} = 1.84 \times 10^{19}$  such automata, half of them being even-number rules. However, among only 64 much simpler even-number *totalistic* 1D2Cr1 CAs, T 2-CAs for short, there are some automata with potentially good shading properties. Figure 2.45 shows GFE values for all 128 and GF plots for selected T 2-CAs.

Sample CA patterns of the selected T 2-CAs are shown in Fig. 2.46. Rules T 2-CA: 112 and 120 are omitted due to low visual attractiveness.



**Fig. 2.46** The patterns of selected T 2-CAs: 56, 84, 88 and 106. The convention as in Fig. 2.44

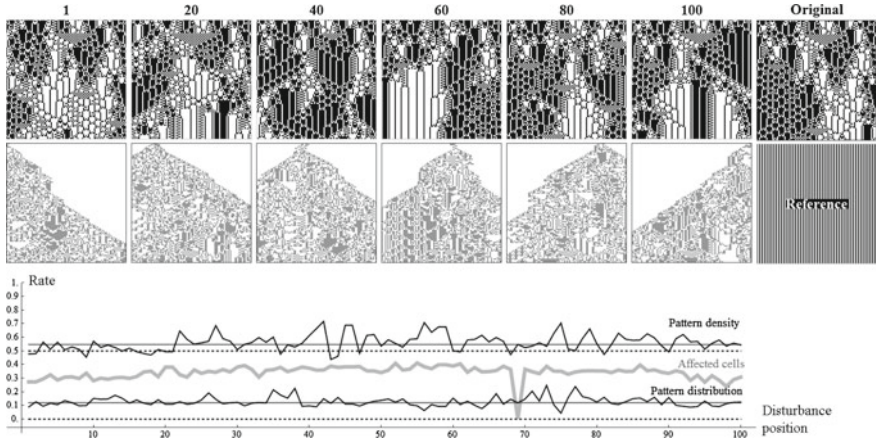
## 2.8 Robustness of CASS

In this brief robustness analysis, two kinds of failure are investigated: a permanent cell malfunction, i.e., when a single cell in IC yields wrong, in this case the opposite output, and a certain kind of edge failure [10], called here an electric failure. In this case an entire column of cells fails and causes separation of the CA array into two disjointed parts. These two kinds of failure have been analyzed for three shading automata on  $100 \times 100$  array at randomly generated 50 % black cells IC.

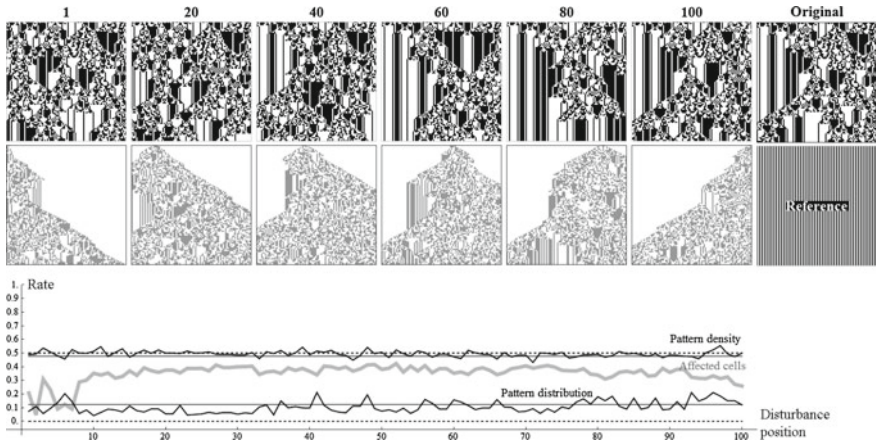
### 2.8.1 Permanent Malfunction of a Single Cell in IC

Hundred consecutive CA patterns with reversed value in a single cell of IC were generated and compared to the original (undisturbed) patterns. The positions of the





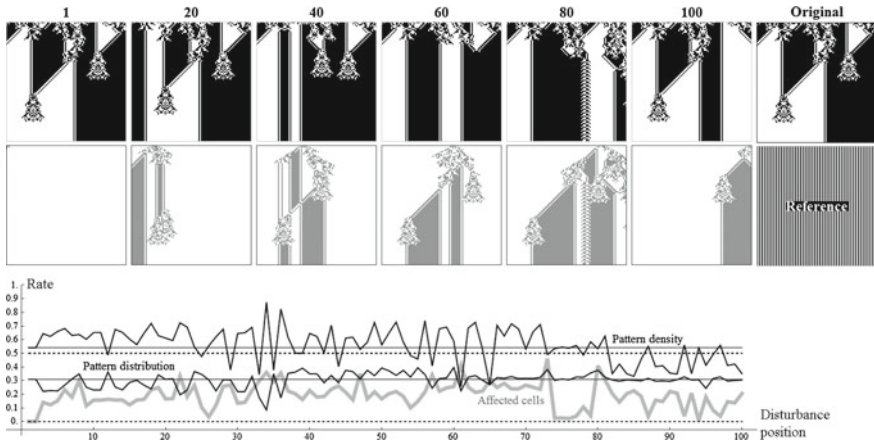
**Fig. 2.47** A single cell failure in IC of general automaton  $CA_{SH}$ . The *top* row of patterns: six sample positions of the cell failure (1, 20, ..., 100), followed by the original, undisturbed pattern for comparison. The *middle* row of patterns: the difference between the original and disturbed patterns followed by referential pattern which has ideal density (0.5) and distribution ( $GDE = 0$ ). On the *bottom* the pattern densities and distributions for disturbance positions from 1 to 100 are plotted in *black*, the respective values for the original pattern are shown in *gray*, and for the referential pattern - as *dashed* lines. The *thick gray line* indicates the rate of affected cells. For clarity, the actual patterns are shown in *black and white*, and the differences - in *gray*



**Fig. 2.48** A single cell failure in IC of ST CA -  $CA_{ST666}$ . The convention as in Fig. 2.47

malfunctioning cell range from 1 to 100 starting from the left. The results for  $CA_{SH}$  are shown in Fig. 2.47. Figures 2.48 and 2.49 show: the influence of the failure of a single cell in IC of ST CA ( $CA_{ST666}$ ), and T CA ( $CA_{T52}$ ), respectively.



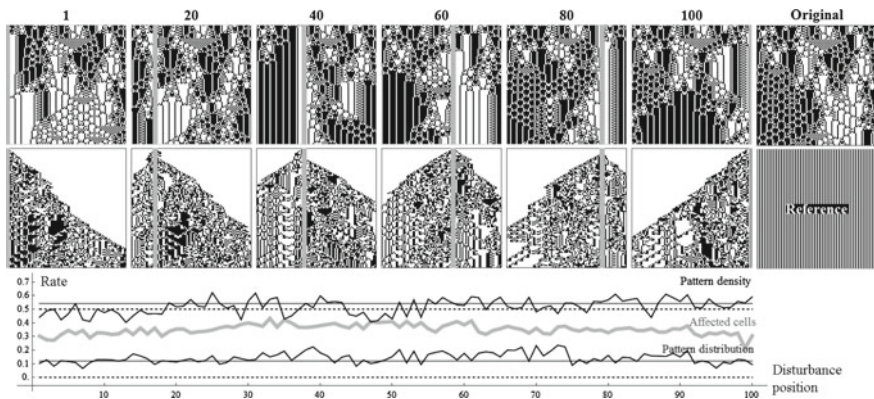


**Fig. 2.49** A single cell failure in IC of T CA - CA<sub>T52</sub>. The convention as in Fig. 2.47

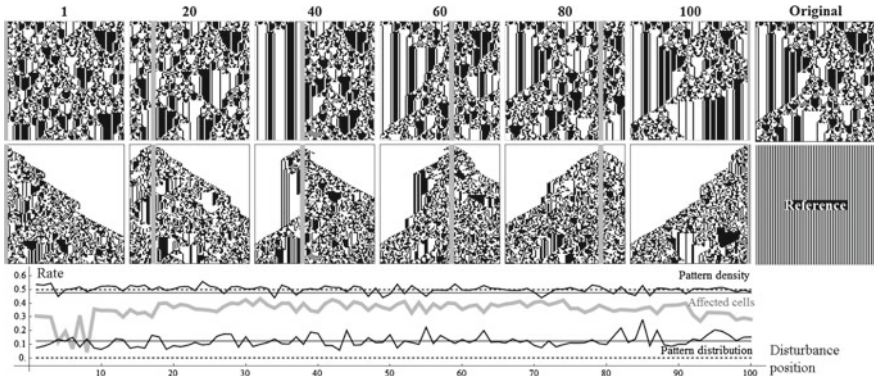
### 2.8.2 Electric Failure

Figure 2.50 shows the results of the electric failure simulation for CA<sub>SH</sub>.

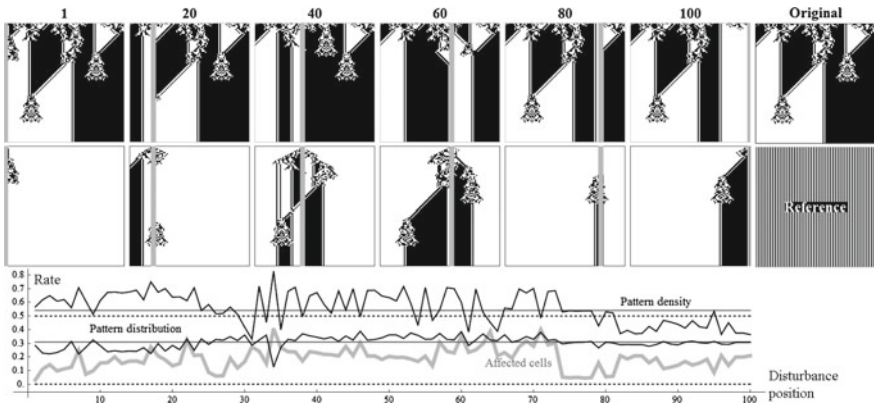
The imitation of an entire column of cells being deactivated, as in a power failure, is done by introducing an additional state of cell of value of 2. It does not mean,



**Fig. 2.50** Electric failure of a column of cells of the *general* automaton CA<sub>SH</sub>. The *top* row of patterns: six sample positions of the malfunctioning column (1, 20, ..., 100), followed by the undisturbed pattern for comparison. The *middle* row of patterns: the differences with the undisturbed patterns, followed by referential pattern which has ideal density (0.5) and distribution ( $GDE = 0$ ). On the *bottom*, the pattern densities and distributions for disturbance positions from 1 to 100 are plotted in *black*, the respective constant values for the original pattern are shown in *gray*, and for referential pattern - as *dashed* lines. *Thick gray line* indicates the rate of affected cells. For clarity, the actual patterns and differences are shown in *black and white*, and the malfunctioning column - in *gray*



**Fig. 2.51** Electric failure of a column of cells of *semi-totalistic* automaton  $CA_{ST666}$ . The convention as in Fig. 2.50



**Fig. 2.52** Electric failure of a column of cells of *totalistic* automaton  $CA_{T52}$ . The convention as in Fig. 2.50

however, that it becomes a three color (state) CA. 2 is an intrusion which does not interact with other values of cell states. Such a modified CA follows two additional simple rules: 2 acts like a 0 and 2 never changes. The same method was used for implementing voids in a surface for a 2D CA as described in [54]. 100 consecutive arrays with a single inactive column of cells are generated and compared to original patterns. The positions of malfunctioning column range from 1 to 100 starting from the left. Figures 2.51 and 2.52 show the influence of electric failure of a column of cells of: ST CA ( $CA_{ST666}$ ), and T CA ( $CA_{T52}$ ), respectively.

CA patterns of the shading automata are quite sensitive to the failures described above. However, their basic shading properties remain relatively unaffected as shown in Table 2.5.

**Table 2.5** Standard deviations of pattern graynesses ( $\sigma_\rho$ ), pattern distributions ( $\sigma_d$ ) and mean rates of affected cells ( $\mu$ ) of 100 disturbed patterns to the undisturbed patterns. Levels of gray in the backgrounds reflect the sensitivities, i.e., darker grays correspond to more severe impact of failures

	Cell failure			Electric failure		
	Density $\sigma_\rho$	Distribution $\sigma_d$	Affected cells ( $\mu$ )	Density $\sigma_\rho$	Distribution $\sigma_d$	Affected cells ( $\mu$ )
<b>General</b> CA <sub>SH</sub>	0.062	0.034	0.342	0.055	0.038	0.352
<b>Semi-totalistic</b> CA <sub>ST666</sub>	0.022	0.042	0.352	0.025	0.041	0.354
<b>Totalistic</b> CA <sub>T52</sub>	0.126	0.054	0.191	0.117	0.043	0.189

As Table 2.5 indicates, the number of affected cells in the case of *totalistic* automaton (CA<sub>T52</sub>) is the lowest. However, its basic shading properties, i.e., pattern grayness and distribution are affected the most. The *general* automaton (CA<sub>SH</sub>) is the most robust in respect to the pattern distribution. The *semi-totalistic* automaton (CA<sub>ST666</sub>) is the most robust in respect to the pattern grayness. In case of both types of failure, however, it is possible, to reconfigure ICs in order to optimize the array despite the malfunction. That is, it is imaginable to adjust CA patterns without actually repairing the malfunctioning cell or column of cells.

## References

1. Alonso-Sanz R, Martín M (2006) A structurally dynamic cellular automaton with memory in the hexagonal tessellation. In: Cellular automata. Springer, Heidelberg, pp 30–40
2. Baas NA, Torbjorn H (2005) Higher order cellular automata. Adv Complex Syst 8(2–3): 169–192
3. Bandini S, Bonomi A, Vizzari G, Acconci V (2010) A cellular automata-based modular lighting system. In: Cellular automata. Springer, Heidelberg, pp 334–344
4. Bays C (1994) Cellular automata in the triangular tessellation. Complex Syst 8:127–150
5. Bays C (2001) Cellular automata and the game of life in the hexagonal grid. <http://www.cse.sc.edu/~bays/h6h6h6/>
6. Bays C (2012) Cellular automata in triangular, pentagonal and hexagonal tessellations. In: Complexity C (ed) Meyers RA. Springer, New York, pp 434–442
7. Bilotta E, Lafusa A, Pantano P (2003) Searching for complex CA rules with GAs. Complexity 8(3):56–67
8. Brender RF (1970) A programming system for the simulation of cellular spaces. Technical report, DTIC document
9. Chavey D (1989) Tilings by regular polygons II: a catalog of tilings. Comput Math Appl 17(1–3):147–165
10. Darabos C, Giacobini M, Tomassini M (2007) Performance and robustness of cellular automata computation on irregular networks. Adv Complex Syst 10(supp01):85–110
11. Das R (1998) The evolution of emergent computation in cellular automata. Colorado State University

12. Das R, Mitchell M, Crutchfield J (1994) A genetic algorithm discovers particle based computation in cellular automata. In: Davidor Y (ed) *Parallel problem solving from nature PPSN III*, Lecture Notes in Computer Science, vol 866. Springer, Heidelberg, pp 244–353
13. El Yacoubi S, Jacewicz P (2007) A genetic programming approach to structural identification of cellular automata. *J Cell Automata* 2:67–76
14. Faraco G, Pantano P, Servidio R (2006) The use of cellular automata in the learning of emergence. *Comput Educ* 47(3):280–297
15. Garzon M (1995) *Models of massive parallelism: analysis of cellular automata and neural networks.*, European association for theoretical computer science Springer, Heidelberg
16. Grefenstette J, Gopal R, Rosimaita B, Gucht D (1985) Genetic algorithms for the traveling salesman problem. *Proceedings of the 1st international conference on genetic algorithms and their applications*. Psychology Press, Pittsburgh, pp 160–168
17. Gruber P (2010) *Biomimetics in architecture*. Springer, Wien
18. Hanson JE (2009) Cellular automata, emergent phenomena. In: Meyers RA (ed) *Encyclopedia of complexity and systems science*. Springer, Heidelberg, pp 768–778
19. Imai K, Morita K (2000) A computation-universal two-dimensional 8-state triangular reversible cellular automaton. *Theoret Comput Sci* 231(2):181–191
20. Kari J (2005) Theory of cellular automata: a survey. *Theoret Comput Sci* 304(1–3):3–33
21. Kepler J (1938) *Harmonice mundi* (linz, 1619). English edition: *Harmonies of the world*, Book 5
22. Kiester RA, Sahr K (2008) Planar and spherical hierarchical, multi-resolution cellular automata. *Comput Environ Urban Syst* 32:204–213
23. Knuth D (1968) *The art of computer programming 1: fundamental algorithms 2: seminumerical algorithms 3: sorting and searching*
24. Konopka AK (2006) *Systems biology: principles, methods and concepts* taylor and francis. Taylor and Francis, Boca Raton
25. Laman G (1970) On graphs and rigidity of plane skeletal structures. *J Eng Math* 4:331–340
26. Lee K, Xu H, Chau H (2001) Parity problem with a cellular automaton solution. *Phys Rev E* 026702(64):1–4
27. Morita K (2012) Reversible cellular automata. *Handbook of natural computing*. Springer, Heidelberg, pp 231–257
28. de Oliveira P, Bortot J, Oliveira G (2006) The best currently known class of dynamically equivalent cellular automata rules for density classification. *Neurocomputing* 70(1–3):35–43
29. Pegg E (2000) Half-distance rules with low resolution, an interactive demonstration. <http://demonstrations.wolfram.com/HalfDistanceRulesWithLowResolution/>
30. Pegg E, Zawidzki M (2008) Cellular shading, an interactive demonstration. <http://demonstrations.wolfram.com/CellularShading/>
31. Peters HM (1993) Functional organization of the spinning apparatus of *Cyrtophora citricola* with regard to the evolution of the web (Araneae, Araneidae). *Zoomorphology* 113(3):153–163
32. Phillips C, Gans D, Kuz Z (2003) *The organic approach to architecture*. Academy Press, New York
33. Post EL (1936) Finite combinatory processes-formulation. *J Symbolic Logic* 1(03):103–105
34. Preston KJ, Duff MJ (1984) *Modern Cellular Automata: theory and applications*. Plenum Press, New York
35. Preston Jr K (1961) The cellscan system-tm a leucocyte pattern analyzer. In: Western joint IRE-AIEE-ACM computer conference, ACM, pp 173–183, papers presented at the 9-11 May 1961
36. Rechenberg I (1973) *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution* (in German). Ph.D. thesis, stuttgart
37. Rocha M, Vilela C, Neves J (2000) A study of order based genetic and evolutionary algorithms in combinatorial optimization problems. *Intelligent problem solving., Methodologies and approaches* Springer, Heidelberg, pp 601–611
38. Shalizi CR, Haslinger R, Rouquier JB, Klinkner KL, Moore C (2006) Automatic filters for the detection of coherent structure in spatiotemporal systems. *Phys Rev E* 73(3):036–104

39. Steiglitz K, Kamal I, Watson A (1988) Embedding computation in one-dimensional automata by phase coding solitons. *IEEE Trans Comput* 37(2):138–145
40. Terrazas G, Siepmann P, Krasnogor N (2008) An evolutionary methodology for the automated design of cellular automaton-based complex systems. *J Cell Automata* 2(1):77–102
41. Toffoli T (1984) Cam: a high-performance cellular-automaton machine. *Phys D* 10(1):195–204
42. Toffoli T, Margolus N (1987) *Cellular automata machines: a new environment for modeling*. MIT press, California
43. Trunfio GA (2004) Predicting wildfire spreading through a hexagonal cellular automata model. In: *Cellular automata*. Springer, Heidelberg, pp 385–394
44. Turing AM (1936) On computable numbers, with an application to the entscheidungsproblem. *J Math* 58(345–363):5
45. Ventrella J (2009) Earth day 2009 – a spherical cellular automaton. <http://www.ventrella.com/EarthDay/EarthDay.html>
46. Ventrella J (2011) Glider dynamics on the sphere: exploring cellular automata on geodesic grids. *J Cell Automata* 6(2–3):245–256
47. Von Neumann J (1951) The general and logical theory of automata. *Cerebral mechanisms in behavior*, pp 1–41
48. Wojtowicz M (2005) Exploring cellular automata with MCell. In: *Artificial life models in software*. Springer, Heidelberg, pp 233–261
49. Wolz D, De Oliveira PP (2008) Very effective evolutionary techniques for searching cellular automata rule spaces. *J Cell Automata* 3(4):289–312
50. Yu CH, Nagpal R (2010) A self-adaptive framework for modular robots in dynamic environment: theory and applications. *Int J Rob Res* p 0278364910384753
51. Zawidzki M (2008) Window opacity controlled by cellular automata, an interactive demonstration. <http://demonstrations.wolfram.com/WindowOpacityControlledByCellularAutomata/>
52. Zawidzki M (2009) Implementing cellular automata for dynamically shading a building facade. *Complex Syst* 18(3):287
53. Zawidzki M (2010) Delayed CA, an interactive demonstration. <http://demonstrations.wolfram.com/DelayedCA/>
54. Zawidzki M (2011) Application of semitotalistic 2d cellular automata on a triangulated 3d surface. *Int J Des Nat Ecodyn* 6(1):34–51
55. Zawidzki M (2012) A Cellular Automaton Mapped on the Surface of a Cylinder. <http://demonstrations.wolfram.com/ACellularAutomatonMappedOnTheSurfaceOfACylinder/>, an interactive demonstration
56. Zawidzki M (2014) Semitotalistic Triangular Cellular Automata on a Geodesic Sphere. <http://demonstrations.wolfram.com/SemitotalisticTriangularCellularAutomataOnAGeodesicSphere>, an interactive demonstration
57. Zawidzki M, Bator M (2012) Application of evolutionary algorithm for optimization of the sequence of initial conditions for the cellular automaton-based shading. *Journal of Cellular Automata* 7(5–6):363–384
58. Zawidzki M, Fujieda I (2010) The prototyping of a shading device controlled by a cellular automaton. *Complex-Systems* 19(2):157–175
59. Zawidzki M, Nishinari K (2013) Shading for building facade with two-color one-dimensional range-two cellular automata on a square grid. *Journal of Cellular Automata* 8(3–4):147–163

Discrete Optimization in Architecture

Building Envelope

Zawidzki, M.

2017, XIV, 121 p. 118 illus., 35 illus. in color., Softcover

ISBN: 978-981-10-1390-4