Evolution and Dynamics of Small-World Cellular Automata

Marco Tomassini
Mario Giacobini
Christian Darabos
Information Systems Department,
University of Lausanne,
1015 Lausanne, Switzerland

We study an extension of cellular automata to arbitrary interconnection topologies for the majority and the synchronization problems. By using an evolutionary algorithm, we show that small-world type network topologies consistently evolve from regular and random structures without being designed beforehand. These topologies have better performance than regular lattice structures and are easier to evolve, which could explain in part their ubiquity. Moreover, we show experimentally that general graph topologies are much more robust in the face of random faults than lattice structures for these problems.

1. Introduction

Networks, which can be formally described by the tools of graph theory, are a central model for the description of many phenomena of scientific, social, and technological interest. Typical examples include the Internet, the World Wide Web, social acquaintances, electric power networks, neural networks, and many others [1]. In recent years there has been substantial research activity in the science of networks, motivated by a number of new results, both theoretical and applied. The pioneering studies of Watts and Strogatz [2, 3] have been instrumental in initiating the movement, and they have been followed by many others in the subsequent years. Their key observation was that most real networks, both in the biological world as well as man-made structures, have mathematical properties that set them apart from regular lattices and from random graphs, which were the two main topologies that had been studied until then. In particular, they introduced the concept of *small-world networks*, in which most pairs of vertices are connected by a short path through the network. The existence of short paths between any pair of nodes has been found in networks as diverse as the Internet, airline routes, neural networks, metabolic networks, among others. The presence of short paths is also a characteristic of random graphs, but what sets these real networks apart is a larger *clustering coefficient*

than that of random graphs having a comparable number of nodes and links. The clustering coefficient roughly represents the probability that two nodes that are neighbors of a third one, are also neighbors of each other, which means that there is more local structure in these networks than in plain random graphs. In section 2 we offer a brief introduction to relevant graph concepts that are used in this work. An excellent recent review of the field is to be found in [1].

The topological structure of a network has a marked influence on the processes that may take place on it. Regular and random networks have been thoroughly studied from this point of view in many disciplines. In computer science, for instance, variously connected networks of processors have been used in parallel and distributed computing [4], while lattices and random networks of simple automata have also received a great deal of attention [5, 6]. On the other hand, due to their novelty, there are very few studies of the computational properties of small-world networks. One notable exception is Watts' book [3] in which cellular automata (CAs) computation on small-world networks is examined for some representative network structures. However, there is no hint in these works as to how such automata networks could arise in the first place, without being designed by a prescribed algorithm. Many man-made networks have grown, and are still growing incrementally and explanations have been proposed for their actual shape. The Internet is a case in point, for which a *preferential attachment* growth rule has given good results [7]. This rule simply prescribes that the likelihood for a new node of connecting to an existing one depends on the node's degree: high-degree nodes are more likely to attract other nodes. Indeed, many actual networks seem to have undergone some kind of Darwinian variation and selection process.

Thus, how these automata networks might have come to be selected is an interesting yet unanswered question. In this work, we let a simple artificial evolutionary process find "good" network structures according to a predefined fitness measure, without prescribing the fine details of the wiring. We take as prototypical problems the *majority classification* problem and the *synchronization* tasks, which are the same that Watts discusses in [3] as a useful first step. This will also allow us to compare the products of artificial evolution with Watts' results. We then investigate the effect of some structural constraints on the evolutionary process. Another aspect of interest is how evolved networks compare with known lattice-CA solutions in terms of robustness in the presence of noise. This point will be explored in some detail. A preliminary investigation of the density task without noise only has appeared in [8].

In the next section some background material on graphs is briefly discussed. Section 3 describes the CA problems and previous results. Section 4 presents our evolutionary search for efficient networks. In

section 6 the fault-tolerance properties of the evolved networks are studied. Section 7 gives our conclusions and ideas for future work.

2. Useful definitions for graphs

For ease of reference, here we collect a few definitions and some nomenclature for graphs that are used throughout this work. The treatment is necessarily brief: a more detailed account can be found, for example, in [1, 3].

Let V be a nonempty set called the set of *vertices* or *nodes*, and let E be a symmetric binary relation on V, that is, a set of unordered pairs of vertices. $G = (E, V)$ is called an *undirected graph* and E is the set of *edges* or *links* of G. In *directed graphs* edges have a direction, that is, they go from one vertex to another and the pairs of vertices are ordered pairs. Here we only deal with undirected graphs.

When vertices (u, v) of an undirected graph form an edge they are said to be *adjacent* or *neighbors*. The *degree* k of a vertex is the number of edges impinging on it (or, equivalently, the number of neighbors). The *average degree* $\langle k \rangle$ is the average of all the vertex degrees in G.

A *path* from vertex u to vertex v in a graph G is a sequence of edges that are traversed when going from u to v with no edge traversed more than once. The *length* of a path is the number of edges in it. The *shortest path* between two vertices u and v is the path with the smallest length joining u to v.

A graph is *connected* if there is a path between any two vertices. A *completely connected* undirected graph G with $|V| = N$ vertices has an edge between any two vertices. The total number of edges is $N(N-1)/2$.

A *random graph* is a graph in which pairs of nodes are connected with a given probability p. Consequentely, the total number of edges in a random graph is a random variable whose expectation value is $p[N(N-1)/2]$. Several useful results on random graphs are described in [7].

Four statistics are particularly useful for small-world and random graphs: the average degree described above, the *clustering coefficient*, the *characteristic path length*, and the *degree distribution*. They are briefly described below and in more detail in [7].

Let us take a particular node j in a graph, and assume that it has k edges connecting it to its k neighboring nodes. If all k vertices in the neighborhood were completely connected then the number of edges would be equal to $k(k-1)/2$. The clustering coefficient C is defined as the ratio between the E edges that actually exist between the k neighbors and the number of possible edges between these nodes:

$$C = \frac{2E}{k(k-1)}.$$

The clustering coefficient of a random graph is simply $\langle k \rangle / N = p$, where N is the total number of vertices. For a regular lattice, C is given by:

$$\frac{3(k-2)}{4(k-1)},$$

where k is the (constant) number of nodes that are connected to a given node. C is thus independent of N for a regular lattice, and approaches 3/4 as k increases.

The characteristic path length L is defined in [3] as the median of the means of the shortest path lengths connecting each vertex $v \in G$ to all other vertices.

The degree distribution $P(k)$ of a graph G is a function that gives the probability that a randomly selected vertex has k edges incident on it. For a random graph $P(k)$ is a binomial peaked at $P(\langle k \rangle)$. But most real networks do not show this kind of behavior. In particular, in scale-free graphs which seem to be common in real life [7], $P(k)$ follows a power-law distribution: $P(k) = c\,k^{-\gamma}$, with c and γ positive constants.

According to Watts and Strogatz [2, 3], a *small-world graph* can be constructed starting from a regular ring of nodes in which each node has k neighbors ($k \ll N$) by simply systematically going through successive nodes and "rewiring" a link with a certain probability p. When the edge is deleted, it is replaced with an edge to a randomly chosen node. Two vertices are not allowed to be connected by more than one edge. This procedure will create a number of shortcuts that join distant parts of the lattice. Shortcuts are defined to be edges that join vertices that would be more than two edges apart if they were not connected directly. These shortcuts are the hallmark of small worlds and, while L scales logarithmically in the number of nodes for a random graph, in small-world graphs it scales approximately linearly for low rewiring probability and tends to the random graph limit as the probability increases. This is due to the appearance of shortcut edges between distant parts of the graph, which obviously contract the path lengths between many vertices. However, small worlds typically have a higher clustering coefficient than random graphs. Small-world networks have a degree distribution $P(k)$ that is close to binomial for intermediate and large values of the rewiring probability p, while $P(k)$ tends to a delta function for $p \to 0$ (see [7]).

Following Watts [3], we will show our results as a function of the parameter Φ, which is the fraction of edges in a graph that are shortcuts. The range of Φ is $[0, 1]$, where a value of 0 (no shortcuts) corresponds to a perfect regular lattice, and 1 corresponds to the random graph limit (every link is a shortcut on the average). In between lies the small-world range, with the typical small-world behavior already present for low Φ values (around 0.01 to 0.1). For higher Φ values, the graphs tend to be more random-like.

Small-world graphs as defined by Watts and Strogatz are not really structurally representative of networks found in the real world, which are often, but not always, of the scale-free type, if anything [9]. However, they are easy to construct and measure and, for the purpose of the artificial CA problems with which we are concerned, they are a perfectly legitimate choice for studying the influence of network structure on the dynamics.

3. The cellular automata problems

CAs are dynamical systems in which space and time are discrete. A standard CA consists of an array of cells, each of which can be in one of a finite number of possible states. Here we only consider boolean automata for which the cellular state $s \in \{0, 1\}$. The regular cellular array (lattice) is d-dimensional, where $d = 1, 2, 3$ is used in practice. In one-dimensional lattices, the topology used here, a cell is connected to r local neighbors (cells) on either side, where r is referred to as the *radius* (thus, each cell has $2r + 1$ neighbors, including itself).

CAs are updated synchronously in discrete time steps, according to a local, identical rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells, including the cell itself:

$$s_i^{t+1} = f(s_{i-r}^t \ldots, s_i^t, \ldots s_{i+r}^t), \qquad f : k^{2r+1} \to k$$

where s_i^t denotes the value of site i at time t, $f(.)$ represents the local transition rule, and r is the CA radius. The term *configuration* refers to an assignment of ones and zeros to all the cells at a given time step. It can be described by $\mathbf{s}^t = (s_0^t, s_1^t, \ldots, s_{N-1}^t)$, where N is the lattice size. Often CAs have periodic boundary conditions $s_{N+i}^t = s_i^t$. Configurations evolve in time according to a global update rule Φ which applies in parallel to all the cells $\mathbf{s}^{t+1} = \Phi(\mathbf{s}^t)$.

Here we consider an extension of the CA concept in which the rule is the same on each node, but nodes can be connected in any way. That is, the topological structures are general graphs, provided the graph is connected and self and multiple links are disallowed.

3.1 The majority task

The majority (also called density) task is a prototypical distributed computational task for CAs. For a finite CA of size N it is defined as follows. Let ρ^0 be the fraction of ones in the initial configuration (IC) $\mathbf{s}^0$. The task is to determine whether ρ^0 is greater or less than 1/2. If $\rho^0 > 1/2$ then the CA must relax to a fixed-point configuration of all ones; otherwise it must relax to a fixed-point configuration of all zeros, after a number of time steps of the order of the lattice size N (N is odd to avoid

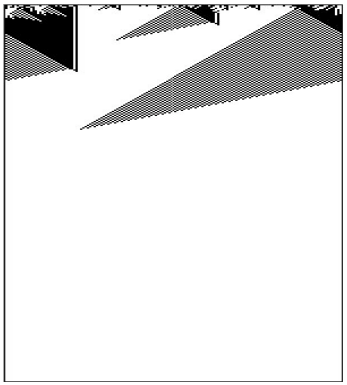


Figure 1. The operation of an evolved one-dimensional, radius three CA for the density task. The CA cell states are represented horizontally (black stands for 1 and white is 0). Time increases down the page. The CA rule has been obtained through artificial evolution by Mitchell *et al.* [13]. The density ρ^0 is 0.416 and the lattice size N is 149.

the case $\rho^0 = 0.5$). This computation is trivial for a computer having a central control. Indeed, just scanning the array and adding up the number of, say, 1-bits will provide the answer in $O(N)$ time. However, it is nontrivial for a small radius-one-dimensional CA since such a CA can only transfer information at finite speed relying on local information exclusively, while density is a global property of the configuration of states [10]. Figure 1 shows the operation of one of the best CAs obtained through artificial evolution.

It has been shown that the density task cannot be solved perfectly by a uniform, two-state CA with finite radius [11], although a slightly modified version of the task can be shown to admit perfect solution by such an automaton [12].

The *performance* P of a given rule on the majority task is defined as the fraction of correct classifications over 10^4 randomly chosen ICs. The ICs are sampled according to a binomial distribution (i.e., each bit is independently drawn with probability 1/2 of being 0). Clearly, this distribution is strongly peaked around $\rho^0 = 1/2$ and thus it makes a difficult case for the CA to solve.

The lack of a perfect solution does not prevent one from searching for imperfect solutions of as good a quality as possible. In general, given a desired global behavior for a CA (e.g., the density task capability), it is extremely difficult to infer the local CA rule that will give rise to the emergence of a desired computation due to possible nonlinearities and large-scale collective effects that cannot in general be predicted from the sole local CA updating rule. Since exhaustive evaluation of all possible

rules is computationally expensive except for elementary ($d = 1, r = 1$) and other fairly simple automata, one possible solution of the problem consists in using evolutionary algorithms, as first proposed by Mitchell *et al.* [10, 13] for uniform CAs, and by Sipper for nonuniform ones [14]. Mitchell *et al.* have shown empirically that homogeneous two-state ring CAs of radius three can be evolved and are capable of reaching a fairly high performance (see below). With $d = 1$ and $r = 3$ there are $2^{128} \sim 10^{36}$ possible rules that the evolutionary algorithm has to search through.

Watts [3] studied a general graph version of the density task. Since a CA rule table depends on the number of neighbors, given that a small-world graph may have vertices with different degrees, he considered the simpler problem of fixing the rule and evaluating the performance of small-world graphs on the task. The chosen rule was a variation of the *majority* rule (not to be confused with the majority problem). The rule simply says that, at each time step, each node will assume the state of the majority of its neighbors in the graph. If the number of neighbors having state 0 is equal to the number of those at 1, then the next state is assigned at random with equal probability. When used in a one-dimensional CA this rule has performance $P \simeq 0$ since it gives rise to stripes of zeros and ones that cannot mix at the borders. Watts, however, has shown that the performance can be good on other network structures, where "long" links somewhat compensate for the lack of information transmission of the regular lattice case, in spite of the fact that the node degrees are still low. Indeed, Watts built several networks with performance values $P > 0.8$, while the best evolved lattices with the same average number of neighbors had P around 0.77 [10, 13] and were difficult to obtain. In fact, according to [15], high-performance strategies were obtained only nine times in 300 runs.

In a remarkable paper [16], Sipper and Ruppin had already examined the influence of different connectivity patterns on the density task. They studied the coevolution of network architectures and CA rules, resulting in nonuniform, high-performance networks, while we are dealing with uniform CAs here. Since those were pre-small world years, it is difficult to state what kind of graphs were obtained. However, it was correctly recognized that reducing the average cellular distance, that is, the characteristic path length, has a positive effect on the performance.

3.2 The synchronization task

The one-dimensional synchronization task was introduced in [17]. In this task the CA, given an arbitrary IC, must reach a final configuration, within $M \simeq 2N$ time steps, that oscillates between all zeros and all ones on successive time steps. Figure 2 depicts the space-time diagram of a CA that solves the task.

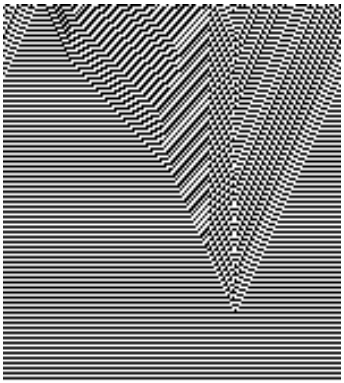


Figure 2. The operation of an evolved one-dimensional CA for synchronization (source: [14]). The ring size N is 149.

As with the density task, synchronization also comprises a nontrivial computation for a small-radius CA, and it is thus extremely difficult to come up with CA rules that, when applied synchronously to the whole lattice produce a stable attractor of oscillating all zeros and all ones configurations. Das *et al.* were able to automatically evolve very good ring CA rules of radius three for the task by using genetic algorithms [17]. Sipper did the same for quasihomogeneous CAs, that is, CAs with a few different rules instead of just one [14], attaining excellent performance for radius-one CAs. The performance of a CA on this task is evaluated by running it on randomly generated ICs, uniformly distributed over densities in the range $[0, 1]$, with the CA being run for $M \simeq 2N$ time steps. Figure 2 is an illustration of the space-time operation of a typical evolved ring CA that solves the synchronization task. Evolved CAs for this task have performance P close to one.

Watts [3] also has a brief section about the synchronization task on small worlds. He finds that a simple variant of the majority rule used above for the density task, works also for the synchronization task. The rule is called the "contrarian" rule, and it operates in the same way as the majority rule, except that it gives the opposite state as output. We adhered to this rule in order to be able to compare our results with Watts'. The synchronization task is probably less interesting than density in a small world because, while the ordinary lattice CA have less than optimal performance on the density task, they are near-perfect for synchronization. Nevertheless, the task is a difficult one as it requires precise coordination among many elementary agents, and it is thus representative of distributed cooperative problem solving and worth studying.

4. Artificial evolution of small worlds

Evolutionary algorithms (EAs) have been successfully used for more than 10 years to evolve network topologies for artificial neural networks and several techniques are available [18]. As far as the network topology is concerned, the present problem is similar, and we use an unsophisticated structured EA with the aim of evolving small-world networks for the density and synchronization tasks. Our EA is spatially structured, as this permits a steady diffusion of good solutions in the population due to a less intense selection pressure [19]. We have chosen this setting which gave good results on other search problems [20], but a simple panmictic EA would probably be equally effective. The population is arranged on a 20×20 square grid for a total of 400 individuals. Each individual represents a network topology and it is coded as an array of integers denoting vertices. Each vertex in the array has a list of the vertices it is connected to. As the graph is undirected, the information is redundant (e.g., if X is connected to Y, then both have the other in their own connections list). The automaton rule is the generalized majority rule in the case of the density task, and the contrarian rule for synchronization as described previously. During the evolution the network nodes are constrained to have a maximum degree of 50. The termination condition is reached after computing exactly 100 generations of the EA.

4.1 Evolution of graphs for the density task

The fitness of a network of automata in the population is calculated by randomly choosing 100 out of the 2^N possible ICs with uniform density (i.e., any initial density has the same probability of being selected) and then iterating the automaton on each IC for $M = 2N$ time steps, where $N = 149$ is the automaton size. The network's fitness is the fraction of ICs for which the rule produced the correct fixed point, given the known IC density. At each generation a different set of ICs is generated for each individual. Selection is done locally using a central individual and its north, east, south, and west first neighbors in the grid. Binary tournament selection is used with this pool. The winner is then mutated (see below) and evaluated. It replaces the central individual if it has a better fitness.

Mutation is designed to operate on the network topology and works as follows. Each node of an individual is mutated with probability 0.5. If chosen, a vertex (called the *target* vertex) will have an edge either added or removed to a randomly chosen vertex (called the *destination* vertex) with probability 0.5. This will only happen if all the requirements are met (minimum and maximum degree are respected). Mutation will not take place if the source vertex has already reached its maximum degree and a vertex should be added. Analogously, mutation

will not happen if an edge has to be removed from a vertex that has minimum degree. If the same case happens with the target, another vertex is randomly chosen. This version of the algorithm does not use recombination operators.

4.1.1 Evolution from regular lattices

In this first series of experiments we started from regular rings, which is the customary way for constructing small-world graphs [3]. The initial population was composed by individuals that are regular rings with node degree $k = 4$, that is, each vertex is connected to its four nearest neighbors in the ring, instead of rings with $k = 6$, which is the case treated by Watts. Moreover, to start with sufficient diversity in the population, we slightly modify each of them by adding an edge with a probability of 0.1 applied to each vertex.

Figure 3 shows the genotypical population entropy, Φ (see section 2), fitness, and performance of the best individual (as defined in sections 3.1 and 4) as a function of the generation number. The curves represent data from a typical run out of 50 independent runs of the EA. Recall that performance is defined off-line as the fraction of correctly classified ICs which are binomially distributed (see section 3.1), while fitness computes the same fraction on a uniformly distributed sample over density, which is an easier task (see section 4.1).

We see that fitness quickly reaches high levels, while performance, which is a harder measure of the generalization capabilities of the evolved networks on the density task, stays lower and then stabilizes at a level greater than 0.8. The population entropy remains high during all runs, meaning that there is little diversity loss during evolution. Note that the entropy refers to the "genotype" and not to fitness. This is unusual and probably due to the spatial structure of the evolutionary algorithm, which only allows slow diffusion of good individuals

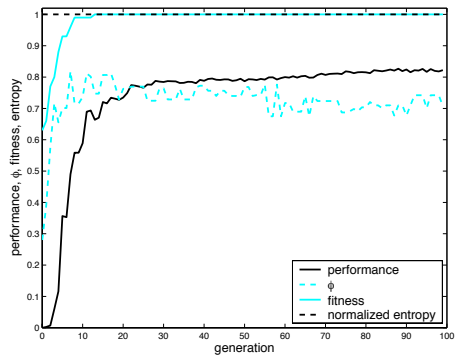


Figure 3. Density task. A typical evolutionary run starting from a perturbed ring population.

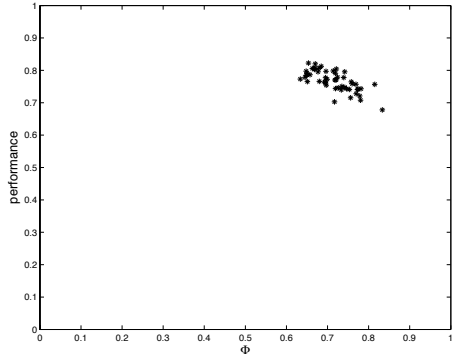
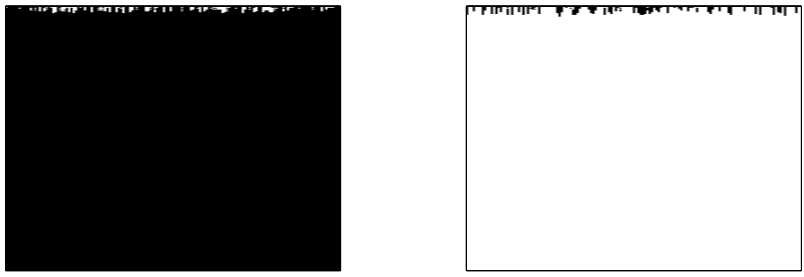


Figure 4. Density task. The Φ *versus* performance values of the 50 best individuals found in the 50 evolutionary runs starting from a population of perturbed rings.

through the grid [19]. The Φ curve is particularly interesting as it permits a direct comparison with Watts' hand-constructed graphs [3]. The results fully confirm his measurements, with networks having best performance clustering around Φ values between 0.6 and 0.8 (see Figure 4). The mean degree $\langle k \rangle$ of the evolved networks is around 7, which compares well with the radius-three lattice CA case and Watts' (see Figure 9 in section 4.1.2). Therefore, we see that even a simple EA is capable of consistently evolving good performance networks in the small-world range. This is not the case for the standard ring CAs for the majority task, where good rules are notoriously difficult to evolve. In fact, while we consistently obtain networks having performance around 0.8 in each evolutionary run, Mitchell *et al.* [13] found that only a small fraction of the runs lead to high-performance CAs. As well, our networks and Watts' reach higher performance: 0.82 against 0.77 for the lattice. Evidently, the original fitness landscape corresponding to the 2^{128} possible ring CAs with radius three is much more difficult to search than the landscape corresponding to all possible graphs with N vertices. To this we may add that the performance of the small-world solutions are better than those of the original lattices as N increases, as was observed by Watts and confirmed by our study (not shown here for lack of space). Work is under way to study the basic statistics of the above landscapes in order to obtain a better understanding of their structures.

The operation of a typical evolved small-world network can be seen in the space-time diagram of Figure 5. Although a direct comparison with the previous Figure 1 is difficult due to the very different network connections, still, one can see that the information transfer is much faster thanks to the distant connections, and the problem is thus solved in fewer steps.



(a) (b)

Figure 5. The operation of an evolved small-world CA for the density task. The density ρ^0 is 0.470 in (a) and 0.523 in (b).

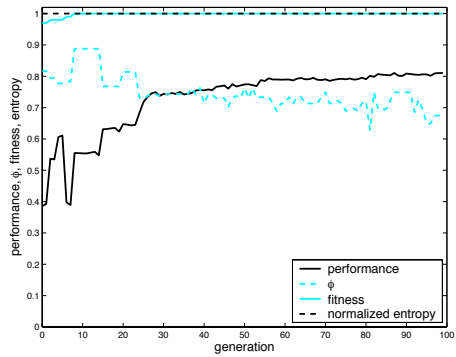


Figure 6. Density task. A typical evolutionary run starting from a random graph population.

4.1.2 Evolution from random graphs

Although the results of artificial evolution from rings are appreciable, giving rise to networks of automata with small-world topology and good performance, the way the initial population is generated might nevertheless contain a bias towards such graphs. In order to really assess the power of this artificial evolution, we designed a second series of experiments in which all the parameters are the same except that the initial population was formed by arbitrary random graphs. A random graph having N vertices can be constructed by taking all possible pairs of vertices and connecting each pair with probability p, or not connecting it with probability $1 - p$. In the experiments $p = 0.03$ and there is no constraint on the minimum node degree, which means that disconnected graphs are also possible. However, we discarded such graphs ensuring that all the networks in the initial population were connected with average degree $\langle k \rangle = Np$ of 4.47.

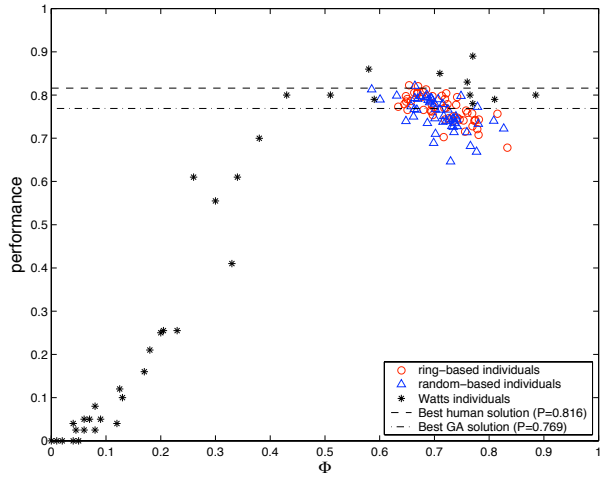
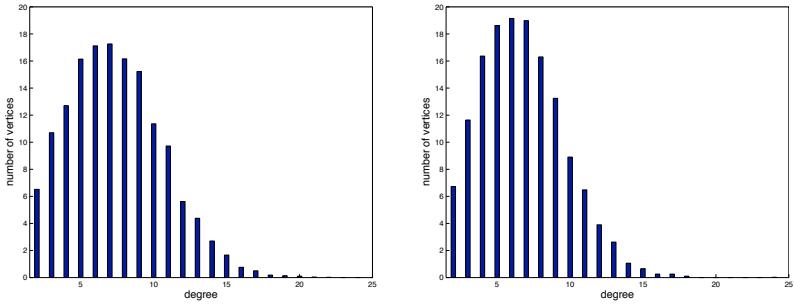


Figure 7. Density task. The Φ *versus* performance values of the best individuals found by evolution starting from rings and random graphs. For comparison, Watts' results are also plotted (redrawn from [3]).

We see again in Figure 6 that genotypic diversity is maintained through evolution as the entropy is always high. Likewise, fitness rises quickly and stays near the maximum. Performance has a different behavior initially. While it starts low and rapidly and steadily increases in the previous case, here it has an approximate value of 0.4 at the beginning. The difference is due to the fact that, in the perturbed ring case, the initial population is still mainly constituted by regular rings, which we know are incapable of performing the density task using the majority rule as the CA rule. In the random graph case, a fraction of the networks in the initial population does a better job on the task. The same conclusion can be reached by looking at the Φ curve. While in the perturbed ring case Φ starts low (Φ is 0 for a lattice) and then slowly increases toward values around 0.7, in the random graph case the contrary happens: Φ is rather high at the beginning because truly random graphs predominate during the first part of the evolution, that is, about 20 generations. After that, graphs are more of the small-world type and converge toward the same Φ region in both cases. This can be clearly seen in Figure 7, where the best 50 individuals of all runs for both initial rings and random graphs are plotted together. The figure also reports the results of Watts for comparison and a couple of lattice CAs that have been hand-designed or evolved with a genetic algorithm [13]. Note that the best evolved ring CA for the task known to date has been obtained by Juillé and Pollack and has performance about 0.86 [21]. It should be noted that in both Figures 3 and 6 performance does



(a) (b)

Figure 8. Degree distribution of best evolved networks for the density task. Initial ring population (a); initial random graph population (b).

Ring-net	$\langle k \rangle$	C	L	Φ	P
A	7.906	0.053	2.649	0.654	0.823
B	7.611	0.053	2.703	0.670	0.820
C	7.409	0.048	2.750	0.685	0.813
D	7.342	0.049	2.736	0.669	0.807
E	7.450	0.057	2.730	0.679	0.807

Figure 9. The five best evolved networks for ring-based initial populations. $\langle k \rangle$ is the mean node degree, C is the clustering coefficient, L is the characteristic path length, Φ is the percentage of shortcuts, and P is the network performance on the density task.

not stop improving even though fitness has reached its maximum value. This is an indication of the good learning and generalization capabilities of the evolved networks.

Figure 8 shows the degree distribution of the best networks found by evolution in the ring case (a), and the random graph case (b). Although the number of vertices is too small for a rigorous statistical treatment, it is easily seen that the distribution is close to binomial in both cases, which is what was expected.

Finally, Figures 9 and 10 summarize the graph-theoretical statistical properties of the five best evolved individuals for the ring case, and for the random graph case. It is interesting that, although no provision was explicitly made for it, the average number of neighbors $\langle k \rangle$ ended up being around seven, very close to six used by construction in Watts [3] (remember that his construction for small-world graphs leaves the initial $\langle k \rangle$ for a ring unchanged). Measured average path lengths L and clustering coefficients C have expected values, given the corresponding Φ values which, without being in the random graph regime, are never-

Rand-net	$\langle k \rangle$	C	L	Φ	P
A	7.798	–	2.695	0.664	0.821
B	7.543	–	2.736	0.585	0.812
C	7.355	–	2.729	0.686	0.800
D	7.422	0.062	2.736	0.631	0.798
E	6.778	–	2.858	0.748	0.797

Figure 10. The five best evolved networks for random-graph-based initial populations. $\langle k \rangle$ is the mean node degree, C is the clustering coefficient, L is the characteristic path length, Φ is the percentage of shortcuts, and P is the network performance on the density task (a "–" means that the clustering coefficient is not computable since those graphs are allowed to have vertices with a degree smaller than two).

theless not far from it for both initial rings and initial random graphs. In other words, the networks with good performance constructed by Watts as well as those artificially evolved have many links rewired. It is worth noticing that, although the EA does not limit the node degree other than establishing a maximum allowed value $k_m = 50$, all the evolved networks have a much smaller $\langle k \rangle$. The operation of graph-CAs evolved from random conditions is qualitatively indistinguishable from those originated from rings (see Figure 5).

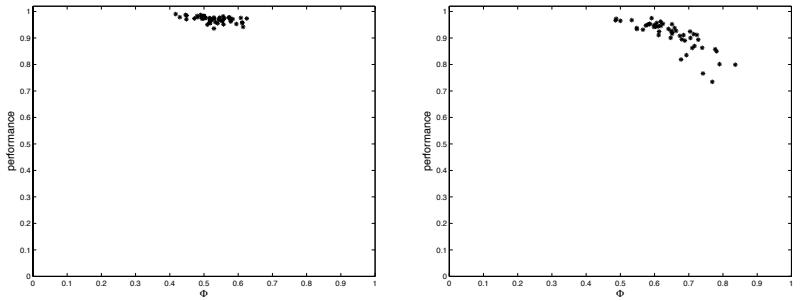
4.2 Evolution of automata graphs for the synchronization task

To artificially evolve automata networks for the synchronization task, we have used exactly the same genetic algorithm setting as for the density task in section 4.1, except for the fitness function, which is the same as the one used by Das *et al.* [17]. We have again two starting points for the initial population: either a population of slightly perturbed radius-two rings, or arbitrary connected random graphs with the same number of vertices.

The results, see Figure 11, are perfectly in line with those obtained by Watts [3]. For reasons of space, we omit the curves representing the evolution of Φ, fitness, and performance through generations, which show behaviors very similar to those seen in the case of the density task (section 4.1).

5. Limiting the number of shortcuts

The network evolutions described in the previous section lead to small-world graphs with a comparatively high proportion of shortcuts, in agreement with the automata built by Watts. Since our systems are just a paradigm for coordinated distributed task solving by simple automata,



(a) (b)

Figure 11. The Φ *versus* performance values of the 50 best individuals found in the 50 evolutionary runs on the synchronization task. (a) starting from rings, (b) starting from random graphs.

we do not take into account real-world constraints such as wire length and other engineering considerations that would be essential for the actual construction of the network. Nevertheless, it is still interesting to study the evolution of the same graphs with the added requirement that Φ is as low as possible. Notice, however, that this does not necessarily imply a shorter wire length in a two-dimensional physical realization.

An easy way to implement this criterion is to include a term in the fitness function which, for a given network fitness, favors networks having a lower Φ value. A similar approach was used by Sipper and Ruppin [16] with the aim of minimizing the connections length of their inhomogeneous CAs. Obviously, the most general way to solve the problem would be to use multiobjective optimization. However, the simpler technique will prove sufficient for our exploration. The new fitness function is thus:

$$f' = f + (1 - \Phi) \times w,$$

where f is the usual CA fitness, w is an empirical weight factor with $w \in [0, 1]$, and f' is the effective fitness. After experimenting with a few different w values, we finally used $w = 0.6$ in the experiments described here, although the precise w value only makes a small difference.

As depicted in Figures 12 and 13 for the density task, we see that the introduction of a selection pressure favoring networks with smaller Φ values is effective in evolving graph-CAs that keep high performance, equal to or better than those previously found using unconstrained evolution (see Figures 3, 4, 6, and 7 for comparison). Here also, starting from a population of perturbed rings or random graphs does not make a big difference although, as expected, starting from slightly perturbed rings, which have low Φ, tends to favor slightly lower Φ values of the evolved networks. The Φ values are around 0.3 (see Figures 12 and 13), while they are about 0.7 in the previous case (Figures 4 and 7).

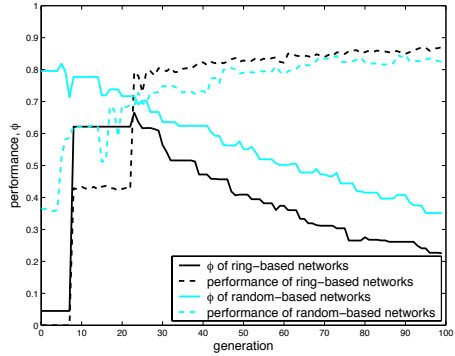


Figure 12. Density task results. Φ and performance curves *versus* generation number for an initial population of perturbed rings and a population of random graphs.

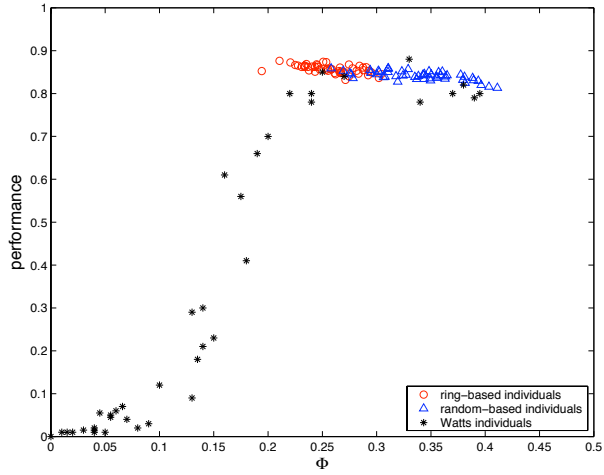
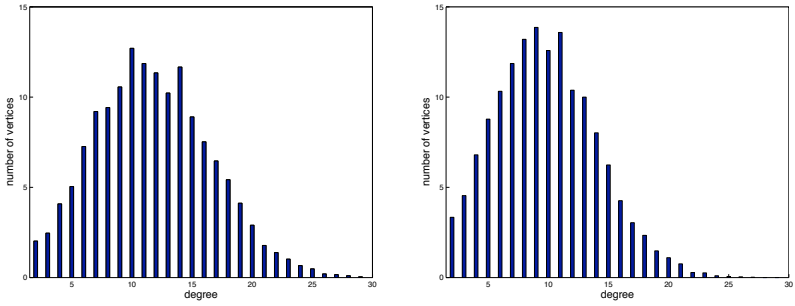


Figure 13. Density task. Φ *versus* performance values of the 50 best individuals found in the 50 evolutionary runs starting from rings and starting from random graphs. Fitness function is f'. For comparison, Watts' results are also reported. They have been redrawn from [3]. Note that the Φ axis range is 0 to 0.5.

The average degrees are somewhat higher however: 11.76 and 9.72 for ring-based and random-graph-based respectively. This compares favorably with Watts' hand-constructed networks (Figure 7.4, p. 192 in [3]), where one can see high-performance networks with Φ around 0.3 but with average degree $\langle k \rangle$ equal to 12. It is clear thus that, to some extent, having more neighbors on the average compensates for the reduced number of shortcut links. The degree distribution for evolved



(a) (b)

Figure 14. Degree distribution of best evolved networks for the density task, using f' as a fitness function. Initial ring population (a); initial random graph population (b). Mean degrees $\langle k \rangle$ are 11.76 in (a) and 9.72 in (b).

networks is shown in Figure 14 and confirms that $P(k)$ is approximately binomial.

Experiments of the same type on the synchronization task (not shown here for reasons of space), give similar results, in the sense that high-performance graph-CAs are obtained easily by artificial evolution. The average values of Φ starting from perturbed rings and random graphs are 0.19 and 0.34 respectively. The degree distribution is again approximately binomial and the mean degrees $\langle k \rangle$ are 13.06 for ring-based individuals, and 10.04 for random-based ones.

5.1 Task flexibility of the evolved networks

As we have seen, it is much easier to evolve small-world networks rather than regular lattices for both tasks. This is also manifest in the fact that networks evolved specifically for one task yield good performance when used for the other one. As noted by Watts [3], the two tasks are nearly identical and thus this finding is not surprising. Furthermore, this remains true for the whole range of Φ values for which automata have been evolved or generated by hand.

For instance, Figures 15 and 16 show how networks evolved for the density task using Φ as a second objective (see previous section) are also well-suited for synchronization (of course, upon changing the rule). The opposite is also true: namely, that networks evolved for the synchronization task can be used for solving the density problem.

6. Robustness in the presence of random faults

Noisy environments are the rule in the real world. Since these automata networks are toy examples of distributed computing systems, it is interesting and legitimate to ask questions about their fault-tolerance aspects.

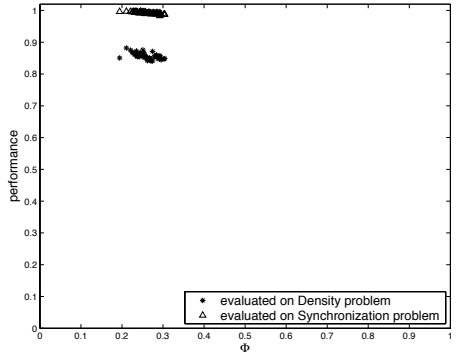


Figure 15. Performance *versus* Φ of networks evolved for the density task on both density and synchronization with ring-based networks.

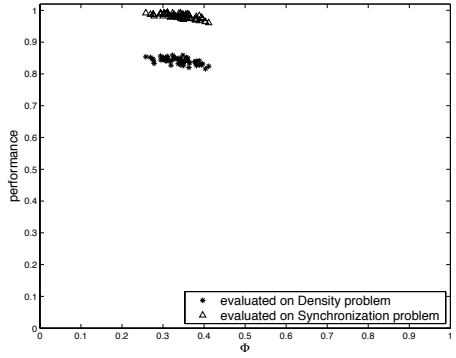


Figure 16. Performance *versus* Φ of networks evolved for the density task on both density and synchronization with random-graph-based networks.

A network of automata may fail in various ways when random noise is allowed. For instance, the cells may fail temporarily or they may die altogether; links may be cut, or both things may happen. In this section, we compare the robustness of standard lattice-CAs and small-world CAs with respect to a specific kind of perturbation, which we call *probabilistic updating*. It is defined as follows: the CA rule may yield the incorrect output bit with probability p_f, and thus the probability of correct functioning will be $(1 - p_f)$. Futhermore, we assume that errors are uncorrelated. This implies that, for a network with N vertices, the probability $P(N, m)$ that m cells (vertices) are faulty at any given time step t is given by

$$P(N, m) = \binom{N}{m} {p_f}^m \, (1 - p_f)^{N-m}$$

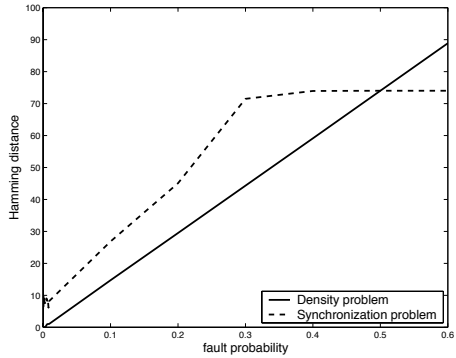


Figure 17. Hamming distance (y-axis) *versus* fault probability (x-axis) for the density problem (full line), and the synchronization problem (dashed line). The curves are averages over 10^3 distinct ICs.

that is, it is binomially distributed. It should be noted that we do not try to correct or compensate for the errors, which is important in an engineered system but very complicated and outside our scope. Instead, we focus on the "natural" fault-tolerance and self-recovering capabilities of the systems under study.

To observe the effects of probabilistic updating on the CA dynamics, two initially identical copies of the system are maintained. One proceeds undisturbed with $p_f = 0$, while the second is submitted to a nonzero probability of fault. We can then measure such things as Hamming distances between unperturbed and faulty configurations, which give information on the spreading of damage (e.g., [22] where the case of synchronous, nonuniform CAs is examined). Figure 17 shows that, for the density task, the amount of disorder is linearly related to the fault probability. This is an excellent result when compared with ring CAs where already at $p_f = 0.001$ the average Hamming distance is about 20 [22], and tends to grow exponentially. At $p_f = 0.1$ it saturates at about 95, while it is still only about 20 for the small-world CA.

This striking difference is perhaps more intuitively clear by looking at Figures 18 and 19. The faulty CA depicted in Figure 19 is the best one obtained by artificial evolution in [10, 13] and it is called EvCA here. It is clear that even small amounts of noise are able to perturb the lattice CA so much that either it classifies the configuration incorrectly (c), or it cannot accomplish the task any longer (d) as p_f increases further. For the same amount of noise the behavior of the small-world CA is much more robust and even for $p_f = 0.01$ the fixed point configuration is only slightly altered. Note also that the EvCA configuration has $\rho^0 = 0.416$ whereas the one used in the small-world CA has $\rho^0 = 0.490$, and it is thus more difficult to classify. For completeness, we note

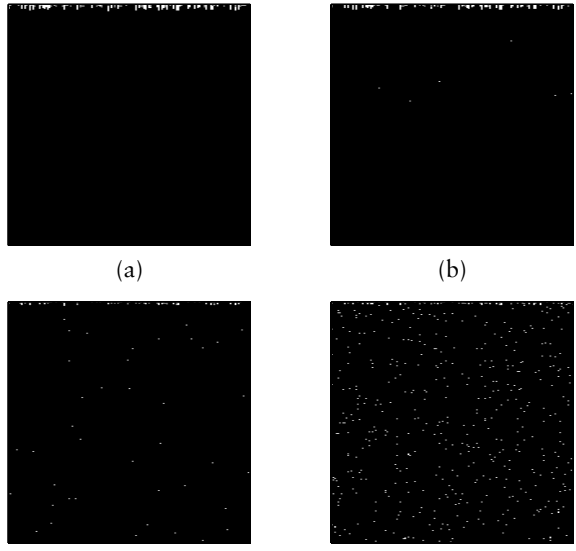


Figure 18. Typical behavior of a small-world CA for the density task under probabilistic updating. The density ρ^0 is 0.490 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

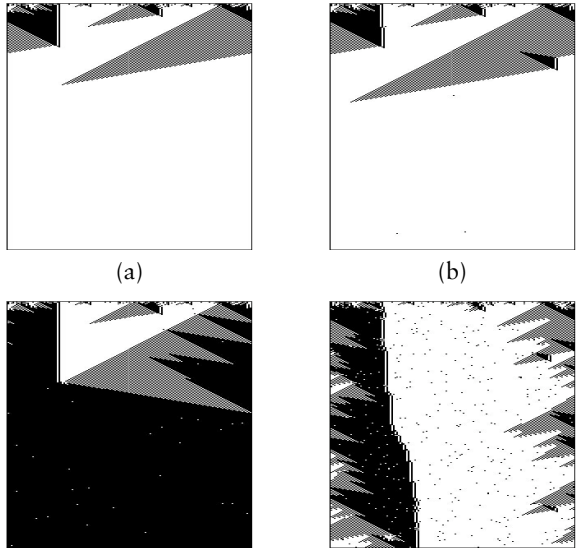


Figure 19. Typical behavior of EvCA [10] under probabilistic updating on the density task. The density ρ^0 is 0.416 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

that in a previous study [23] we investigated the behavior of evolved *asynchronous* lattice CAs for the density task under probabilistic noise. We found that, while asynchronous CAs are much more fault-tolerant than synchronous ones, their robustness is not as good as that of small-world CAs and their performance is significantly lower.

Looking again at Figure 17 we see that the behavior of the synchronization task (dashed line) under noise is poorer. In fact, it is not possible to maintain strict synchronization in the presence of faults. The system manages to limit the damage for low fault probabilities but it goes completely out of phase over $p_f = 0.2$. For higher probabilities the distance stabilizes around 75 (i.e., half of the cells on average are in the wrong state). In spite of this, the behavior is still much better than the one observed for ring CAs, where at $p_f = 0.01$ the Hamming distance is already about 55 [22], while it is only about eight in the small-world CA.

7. Conclusions

Starting from the work of Watts on small-world cellular automata (CAs), we have used an evolutionary algorithm (EA) to evolve networks that have similar computational capabilities. Without including any preconceived design issue, the EA has been consistently able to find high-performance automata networks in the same class of those constructed by Watts. In addition, by giving some evolutionary advantage to low-Φ networks, the evolutionary process has been able to find networks with a low Φ and excellent performance for both tasks.

These results have been easy to find even though the EA is unsophisticated. This means that the space of small-world networks is "solutions rich," which is the contrary of what one observes in the rule space for standard ring CA, where evolving good rules has proved difficult. The power of artificial evolution is seen in the fact that, even starting from a population of completely random graphs, the algorithm finds automata in the same class. This result is an indication that small-world network automata in this range have above average distributed computation capabilities, although we only studied two problems of this type and any generalization would be unwarranted at this stage.

Not only are these networks extremely efficient, they also feature above average robustness against transient probabilistic faults. A comparison with standard lattice CAs shows that small-world CAs are much less affected by random noise. The difference is striking, and could be one of the reasons that explain the ubiquity of irregular "natural" collective computational systems with respect to regular structures.

It is also clear at this point that we have not used the power of artificial evolution at its best. In particular, we adopted the fixed rules of Watts and let the networks evolve. It would probably pay if we would

let the rule evolve together with the network topology. This has been suggested by Watts [3] and has previously been attempted by Sipper and Ruppin with good results [16]. Further work along these lines is needed. We also plan to study the collective computational capabilities and the phase space structure of other small-world graph structures, especially scale-free networks.

References

[1] M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Review*, **45** (2003) 167–256.

[2] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature*, **393** (1998) 440–442.

[3] D. J. Watts, *Small Worlds: The Dynamics of Networks between Order and Randomness* (Princeton University Press, Princeton NJ, 1999).

[4] F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* (Morgan Kaufmann, San Mateo, CA, 1992).

[5] M. Garzon, *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks* (Springer-Verlag, Berlin, 1995).

[6] S. A. Kauffman, *The Origins of Order* (Oxford University Press, New York, 1993).

[7] R. Albert and A.-L. Barabasi, "Statistical Mechanics of Complex Networks," *Reviews of Modern Physics*, **74** (2002) 47–97.

[8] M. Tomassini, M. Giacobini, and C. Darabos, "Evolution of Small-World Networks of Automata for Computation," in *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, edited by X. Yao *et al.* (Springer-Verlag, Berlin, 2004).

[9] L. A. N. Amaral, A. Scala, M. Barthélemy, and H. E. Stanley, "Classes of Small-World Networks," *Proceedings of the National Academy of Sciences USA*, **97**(21) (2000) 11149–11152.

[10] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, **7** (1993) 89–130.

[11] M. Land and R. K. Belew, "No Perfect Two-state Cellular Automata for Density Classification Exists," *Physical Review Letters*, **74**(25) (1995) 5148–5150.

[12] M. S. Capcarrère, M. Sipper, and M. Tomassini, "Two-state, $r = 1$ Cellular Automaton that Classifies Density," *Physical Review Letters*, **77**(24) (1996) 4969–4971.

[13] M. Mitchell, J. P. Crutchfield, and P. T. Hraber, "Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments," *Physica D*, **75** (1994) 361–391.

[14] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach* (Springer-Verlag, Heidelberg, 1997).

[15] J. P. Crutchfield, M. Mitchell, and R. Das, "Evolutionary Design of Collective Computation in Cellular Automata," in *Evolutionary Dynamics: Exploring the Interplay of Selection, Accident, Neutrality, and Function*, edited by J. P. Crutchfield and P. Schuster (Oxford University Press, Oxford, 2003).

[16] M. Sipper and E. Ruppin, "Co-evolving Architectures for Cellular Machines," *Physica D*, **99** (1997) 428–441.

[17] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson, "Evolving Globally Synchronized Cellular Automata," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, edited by L. J. Eshelman (Morgan Kaufmann, San Francisco, CA, 1995).

[18] A. Tettamanzi and M. Tomassini, *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems* (Springer, New York, 2001).

[19] M. Giacobini, E. Alba, A. Tettamanzi, and M. Tomassini, "Modeling Selection Intensity for Toroidal Cellular Evolutionary Algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'04*, volume 3102 of *Lecture Notes in Computer Science*, edited by K. Deb *et al.* (Springer-Verlag, Berlin, 2004).

[20] B. Dorronsoro, E. Alba, M. Giacobini, and M. Tomassini, "The Influence of Grid Shape and Asynchronicity on Cellular Evolutionary Algorithms," in *2004 Congress on Evolutionary Computation (CEC 2004)* (IEEE Press, Piscataway, NJ, 2004).

[21] H. Juillé and J. B. Pollack, "Coevolving the Ideal Trainer: Application to the Discovery of Cellular Automata Rules," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, edited by J. R. Koza *et al.* (Morgan Kaufmann, San Francisco, CA, 1998).

[22] M. Sipper, M. Tomassini, and O. Beuret, "Studying Probabilistic Faults in Evolved Non-uniform Cellular Automata," *International Journal of Modern Physics C*, **7**(6) (1996) 923–939.

[23] M. Tomassini and M. Venzi, "Evolving Robust Asynchronous CA for the Density Task," *Complex Systems*, **13**(3) (2002) 185–204.