Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments

Melanie Mitchell[1], James P. Crutchfield[2], and Peter T. Hraber[1]

Physica D, 75, 361–391, 1994.

Abstract

We present results from experiments in which a genetic algorithm (GA) was used to evolve cellular automata (CAs) to perform a particular computational task—one-dimensional density classification. We look in detail at the evolutionary mechanisms producing the GA's behavior on this task and the impediments faced by the GA. In particular, we identify four "epochs of innovation" in which new CA strategies for solving the problem are discovered by the GA, describe how these strategies are implemented in CA rule tables, and identify the GA mechanisms underlying their discovery. The epochs are characterized by a breaking of the task's symmetries on the part of the GA. The symmetry breaking results in a short-term fitness gain but ultimately prevents the discovery of the most highly fit strategies. We discuss the extent to which symmetry breaking and other impediments are general phenomena in any GA search.

1. Introduction

Cellular automata (CAs) are spatially-extended discrete dynamical systems whose architecture has many desirable features for a large class of parallel computations. In scientific modeling applications, CAs have been used to simulate, for example, magnetic spin systems [15, 90], fluid dynamics [21, 27], chemical oscillations [53, 64], crystal growth [52, 65], galaxy formation [30], stellar accretion disks [77], dynamics in cytoskeletal lattices [82], and the formation of biological patterns (e.g., the intricate fractal patterns seen on mollusk shells [11], or vertebrate pigment patterns [92]). Common to all these modeling applications is the belief that CAs can capture essential features of physical systems in which large-scale behavior arises from the collective effect of large numbers of locally interacting simple components. In engineering applications, CAs have been used to perform, among other things, parallel formal-language recognition [70, 81] and a range of image-processing tasks [55, 72, 73, 76, 83, 93]. There are many other potential engineering applications of CAs, such as forecasting, spatio-temporal noise-reduction, the discovery of coherent structures in data, texture detection, and so on.

The massive parallelism and local connection architecture of CAs, as well as their capacity for resistance to error and noise, means that hardware implementations have the potential for extremely fast and reliable computation that is robust to noisy input data and

[1]Santa Fe Institute, 1660 Old Pecos Trail, Suite A, Santa Fe, New Mexico, U.S.A. 87501.
Email: mm@santafe.edu, pth@santafe.edu
[2]Physics Department, University of California, Berkeley, CA, U.S.A. 94720.
Email: chaos@gojira.berkeley.edu

component failure [28]. The standard approach to parallel computation is to split up a problem into independent modules that are then parceled out to different processors, solved simultaneously, and the piecewise solutions recombined. In contrast, a CA performs computation in a distributed fashion on a spatially-extended lattice. CAs suggest new ways of parallelizing problems that are hard to split up and parcel out. Recent work on CAs has yielded much new insight into the mechanisms by which complex behavior can arise in nonlinear spatially-extended systems with local interactions (e.g., see [23, 34, 88, 91]). However, little is known about how to harness this complex behavior to perform useful computation, since in general it is hard to predict, much less design, the behavior of such systems. The study of CA-based information processing is a case of the general problem of harnessing the computational power of spatially-extended dynamical systems. The difficulty of designing CAs to have desired behavior or to perform a particular task has up to now severely limited their applications in science and engineering, and for general computation. Finding a way to automate the design of CAs would thus have great significance for a number of fields.

In this paper we describe research on using genetic algorithms (GAs) to evolve CAs to perform computations. GAs are search and optimization methods based on ideas from natural genetics and evolution [19, 31, 42, 56]. A GA works on populations of "chromosomes" that represent candidate solutions to a given problem, applying "genetic" operators such as fitness-based reproduction, crossover, and mutation to members of the population over a number of "generations". GAs have become increasingly popular in recent years in machine learning and other disciplines because of their utility in a range of applications. Examples of application areas include engineering design (e.g., aircraft design [10], circuit design [80], and engine-turbine design [71]), operations research (e.g., [4, 36]), automatic programming (e.g., [40, 46]), neural-network design (e.g., [8, 12, 38, 58, 62, 33]), robot control (e.g., [22, 39, 18]), and molecular biology (e.g., DNA sequence assembly [69] and protein-structure prediction [17, 79, 89]). GAs have also been used as scientific models of evolutionary processes in natural systems. Examples include models of economic systems (e.g., [3, 44]), models of the immune system (e.g., [26]), models of ecological phenomena such as biological arms races, host-parasite co-evolution, symbiosis, and resource flow in ecologies (e.g., [5, 6, 13, 14, 40, 42, 43, 45, 50, 51, 67, 74, 75, 85]), models of phenomena in population genetics such as the evolution of recombination (e.g., [9, 24, 54, 78]), and models of the interaction between evolution and learning (e.g., [1, 2, 7, 25, 41, 63, 57, 68, 86, 87]).

The goals of our research are: (1) to better understand the ways in which CAs can perform computations; (2) to learn how best to use GAs to evolve computationally useful CAs; and (3) to understand the mechanisms by which evolution—as modeled by a GA—can create complex, coordinated global behavior in a system consisting of many locally interacting simple parts. CAs are perhaps the simplest examples of such systems. In nature, evolution has resulted in high levels of computational capability within much more complicated systems—a preeminent example being the human nervous system.

In this paper we analyze the GA's behavior in evolving one-dimensional CAs to perform a particular computational task. We investigate both the mechanisms underlying the GA's performance and the impediments it faces in finding CAs that achieve high performance. We argue that the results of our analysis are relevant not only to the particular task we have chosen, but to GA behavior in general.

2. CA Review and Terminology

A CA is spatial lattice of N cells, each of which is in one of k states at time t. Each cell follows the same simple rule for updating its state; the cell's state s at time $t+1$ depends on its own state and the states of some number of neighboring cells at time t. For one-dimensional CAs, the neighborhood of a cell consists of the cell itself and r neighbors on either side. The number of states k and the radius r are parameters of the CA.

The CA starts out with some initial configuration (IC) of cell states, and at each time step the states of all cells in the lattice are synchronously updated. We use the term "state" to refer to the local state s — the value of a single cell. Here we will restrict our attention to binary ($k = 2$) CAs with $s \in \{0, 1\}$. The state at site i is denoted by s^i. The term "configuration" will refer to the pattern of local states over the entire lattice. This is the CA's global state, denoted $\mathbf{s} = s^0 s^1 \ldots s^{N-1}$. The density of 1s in a configuration $\mathbf{s}$ will be denoted $\rho(\mathbf{s})$.

The equations of motion ϕ for a CA (the CA "rule") can be expressed as a look-up table that lists, for each local neighborhood, the update state for the neighborhood's central cell. A sample rule (the "majority" rule) for a one-dimensional "elementary" ($k = 2, r = 1$) CA is the following. Each possible neighborhood η is given along with the "output bit" $s = \phi(\eta)$ to which the central cell is updated.

η	000	001	010	011	100	101	110	111
s	0	0	0	1	0	1	1	1

In words, this rule says that for each neighborhood of three adjacent cells, the new state is decided by a majority vote among the three cells. At time step t, this look-up table is applied to each neighborhood in the current lattice configuration, respecting the choice of boundary conditions, to produce the configuration at $t+1$. The configuration at time t will be denoted $\mathbf{s}_t = s_t^0 s_t^1 \ldots s_t^{N-1}$, where s_t^i is the local state of site i at time t. The CA equations of motion then specify a spatially-local update of a site's value as a function of its neighborhood: $s_{t+1}^i = \phi(\eta_t^i)$, where η_t^i is the neighborhood pattern about site i at time t. This local update induces a global mapping Φ that takes a lattice configuration at t to a new configuration at $t+1$: $\mathbf{s}_{t+1} = \Phi(\mathbf{s}_t)$. This can also be denoted in terms of the t^{th} iterate of Φ as: $\mathbf{s}_{t+1} = \Phi_{t+1}(\mathbf{s}_0)$.

The λ value of a binary CA is defined as the fraction of 1s in the output bits of its rule ϕ. For example, the λ value of the majority rule is 1/2. The λ parameter was originally used in studies of CA behavior [47], but, as we will show, it turns out to be useful in understanding the GA's behavior. (There is a simple interpretation of how λ is related to a CA's behavior. λ gives the density of 1s in the first iterate of a random initial configuration $\mathbf{s}_0$: $\rho(\mathbf{s}_1) = \lambda$.)

The behavior of a one-dimensional CA is often presented as a "space-time diagram", a plot of $\mathbf{s}_t$ over a range of time steps. Two examples are given in Figure 1. These show the actions of the Gacs-Kurdyumov-Levin (GKL) CA [29] on two random ICs; one with $\rho_0 > 1/2$ and the other with $\rho_0 < 1/2$. (Here and later on we use the shorthand ρ_0 for the density $\rho(\mathbf{s}_0)$ of an IC.) In both cases, the CA relaxes to a fixed pattern—in one case all 0s ($\rho(\mathbf{s}_\infty) = 0$) and in the other case all 1s ($\rho(\mathbf{s}_\infty) = 1$). The GKL CA will be discussed further below.

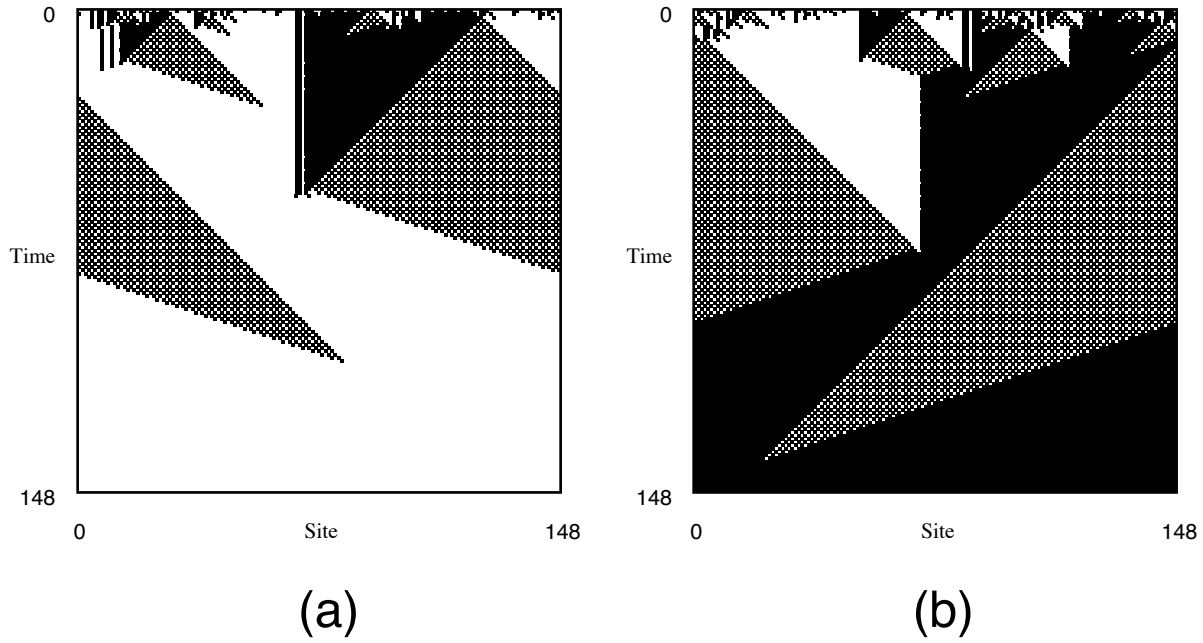

Figure 1: Two space-time diagrams for the binary-state Gacs-Kurdyumov-Levin CA. $N = 149$ sites are shown evolving, with time increasing down the page, from two different ICs over 149 time steps. Here cells with state 0 are white, and cells with state 1 are black. In (a), $\rho_0 \approx 0.48$, and in (b), $\rho_0 \approx 0.52$. Notice that by the last time step the CA has converged to a fixed pattern of (a) all 0s and (b) all 1s. In this way the CA has classified the ICs according to whether $\rho_0 > 1/2$ or $\rho_0 < 1/2$.

In this paper we restrict the discussion to one-dimensional CAs with $k = 2$ and $r = 3$, and with spatially periodic boundary conditions: $s_t^i = s_t^{i+N}$. We most often set N to 149, but also look at the behavior of CA on larger N (up to 999).

3. Previous Work

In [61] we reported results of evolving one-dimensional CAs to perform a particular density classification task: the "$\rho_c = 1/2$" task. This work was a re-examination of an experiment performed by Packard [66], meant to test the hypothesis that a GA evolving CA rules to perform a difficult computational task will tend to select rules close to conjectured phase transitions in rule space between ordered and chaotic behavior ("the edge of chaos"). In [66], the locations of these phase transitions were claimed to correspond to "critical" λ values, λ_c. In [66] the GA tended to select rules close to these critical values; these results were interpreted by Packard as supporting the "edge of chaos" hypothesis. As reported in [61], however, our similar experiments did not support this hypothesis. We also gave a theoretical argument that the $\rho_c = 1/2$ task requires rules with $\lambda = 1/2$ rather than the λ_c values given in [66]. We argued that the results reported in [66] were an artifact of the particular GA used there rather than due to any intrinsic computational advantage of rules with $\lambda = \lambda_c$,

and concluded that to date there is no direct experimental evidence linking computational capability with λ in cellular automata. For a review of these issues and of relations among computation, dynamics, and cellular automata, see [59].

Although the results in [66] were not replicated in our experiments, we did observe several interesting phenomena in the GA's behavior. In [61] we qualitatively described the "epochs of innovation" in the GA's search for successful rules, as well as a breaking of the task's symmetries on the part of the GA. We interpreted the symmetry-breaking as impeding the GA's ability to find the best-performing rules. We also described the competing pressures of selection and "combinatorial drift". In this paper we analyze these phenomena in detail and explain the mechanisms underlying the GA's behavior and the impediments the GA encounters.

4. The Computational Task

The $\rho_c = 1/2$ task is defined as follows. If $\rho_0 < \rho_c$, then the CA is to relax, after a certain number M of time steps, to a fixed pattern of all 0s; otherwise, it is to relax to a fixed pattern of all 1s. The desired behavior is undefined at $\rho_0 = \rho_c$; this case will be precluded by using odd N. On a N site lattice then we have

$$\mathbf{T}_{\rho_c}(N, M) = \left\{ \begin{array}{ll} \Phi_M(\mathbf{s}_0) = 0^N & \text{if } \rho(\mathbf{s}_0) < \rho_c \\ \Phi_M(\mathbf{s}_0) = 1^N & \text{if } \rho(\mathbf{s}_0) > \rho_c \\ \text{undefined} & \text{if } \rho(\mathbf{s}_0) = \rho_c \end{array} \right\} \forall \mathbf{s}_0 \in \{0,1\}^N$$

In this notation the $\rho_c = 1/2$ task is denoted $\mathbf{T}_{1/2}$. This task is an example of "useful computation" in our characterization of the different types of computation in CAs [61]. That is, the global mapping Φ_M is interpreted as a program for performing a useful computation, the IC $\mathbf{s}_0$ is interpreted as the input to that program, and the CA runs for some specified number M of time steps or until it reaches one of a set of "goal" patterns, 0^N or 1^N. The final pattern is interpreted as the output.

The task $\mathbf{T}_{1/2}$ is interesting for a number of reasons. Density classification is closely related to a number of image-processing tasks, and studying simple versions of such tasks in one dimension will help in understanding how to use the GA to scale up to more complex two-dimensional tasks. In addition, the task is nontrivial for a small-radius ($r \ll N$) CA, since density is a global property of a configuration, whereas a small-radius CA relies only on local interactions. In other words, the task difficulty derives from the fact that a CA is specified by ϕ but the useful computation is effected by the global map Φ_M. In fact, the minimum amount of memory for $\mathbf{T}_{1/2}$ is proportional to $log(N)$, since the equivalent of a counter register is required to track the excess of 1s in a serial scan of the IC. In other words, the task requires computation which corresponds to the recognition of a non-regular language. Since the 1s can be distributed throughout the CA lattice, the CA must transfer information over large space-time distances ($\approx N$).

$\mathbf{T}_{1/2}$ possesses two symmetries. Denoting the task's global mapping of strings $\mathbf{s} \in \{0,1\}^N$ to classifications {LO, HI} by T, these are given as follows.

1. If an IC $\mathbf{s}_0$ is spatially reversed on the lattice, T gives the same classification. That is, $T(\mathbf{s}_0) = T(\mathcal{R}\mathbf{s}_0)$, where the symmetry operator $\mathcal{R}$ reverses the order of the bits in

$\mathbf{s}_0$.

2. If all the bits in $\mathbf{s}_0$ are flipped (i.e., 1s are exchanged with 0s and 0s with 1s), then T gives the opposite classification. That is, $T(\mathbf{s}_0) = \mathcal{F}T(\mathcal{F}\mathbf{s}_0)$, where the symmetry operator $\mathcal{F}$ flips the bits in $\mathbf{s}_0$. $\mathcal{F}$ is its own inverse.

Thus, there are at least two symmetries in $\mathbf{T}_{1/2}$ that we expect any high-performance candidate rule to respect—either locally (i.e., with respect to individual neighborhoods in the rule table ϕ) or globally (i.e., with respect to Φ, the global mapping), or both.

The second symmetry of $\mathbf{T}_{1/2}$ has an important consequence when interpreted with respect to ρ_0. Recall that on exactly half the possible ICs — the low density $\mathbf{s}_0$ — the desired behavior is to relax to a fixed point of all 0s, and on the other half — the high density $\mathbf{s}_0$ — the desired behavior is to relax to a fixed point of all 1s. This ρ_0 symmetry requires that any rule that performs this task has $\lambda = 1/2$. Suppose, for example, a rule that carries out the $\mathbf{T}_{1/2}$ task has $\lambda < 1/2$. This implies that for the majority of neighborhoods η, $\phi(\eta) = 0$. This, in turn, means that there will be some $\mathbf{s}_0$ with $\rho(\mathbf{s}_0) > \rho_c$ on which the action of the rule will decrease $\rho(\mathbf{s})$. This is the opposite of the desired action. If the rule acts to decrease $\rho(\mathbf{s})$, it risks producing an intermediate configuration $\mathbf{s}_{t'}$ with $\rho(\mathbf{s}_{t'}) < \rho_c$. This then would lead, under the original assumption that the rule carries out the task correctly, to a fixed point of all 0s, misclassifying $\mathbf{s}_0$. A similar argument holds in the other direction for $\lambda > 1/2$. This informal argument shows that a rule with $\lambda \neq 1/2$ will misclassify certain ICs. Generally, the further away the rule is from $\lambda = 1/2$, the larger the fraction of misclassified ICs.

5. The Strategy of a Hand-Designed CA

Does there exist a CA that can perform the $\rho_c = 1/2$ task? It is possible that no CA exists which performs the task perfectly for all N. However, a $k = 2, r = 3$ rule designed by Gacs, Kurdyumov, and Levin (the GKL rule) [29] appears to perform the task with error decreasing as $N \to \infty$ [48]. (We are not aware of any proof of this, however.) The observed classification performance of the GKL rule as a function of ρ_0 is given in Figure 2 for $N =$ 149, 599, and 999. To make this plot, we ran the GKL rule on 500 randomly generated ICs close to each of 19 densities $\rho \in [0.0, 1.0]$. The fraction of correct classifications was then plotted at each ρ_0. The rule was run either until a fixed point was reached or for a fixed maximum number of time steps $M = 10 \times N$.

Figure 2 indicates that all the misclassifications occur for $\rho_0 \approx \rho_c$, with the width of the error region decreasing as N increases. At $\rho_0 = \rho_c$, in fact, it appears no better than an unbiased random classification. We found that most errors were a result of relaxing to the wrong fixed point (e.g., all 0s for $\rho_0 > \rho_c$). For future reference note that on an $N = 149$ lattice the GKL rule's performance on $\rho_c = 1/2$ classification is ≈ 0.972 when averaged over 10^4 ICs uniformly distributed in 19 equally-spaced ρ bins.

The GKL rule is instructive for the density-classification task in that it happens to give a concrete, though approximate, solution to the optimization facing the GA. The manner in which it implements the required computation, its "strategy", is of equal importance. The rule's "strategy" here refers to the behavioral elements employed during its temporal

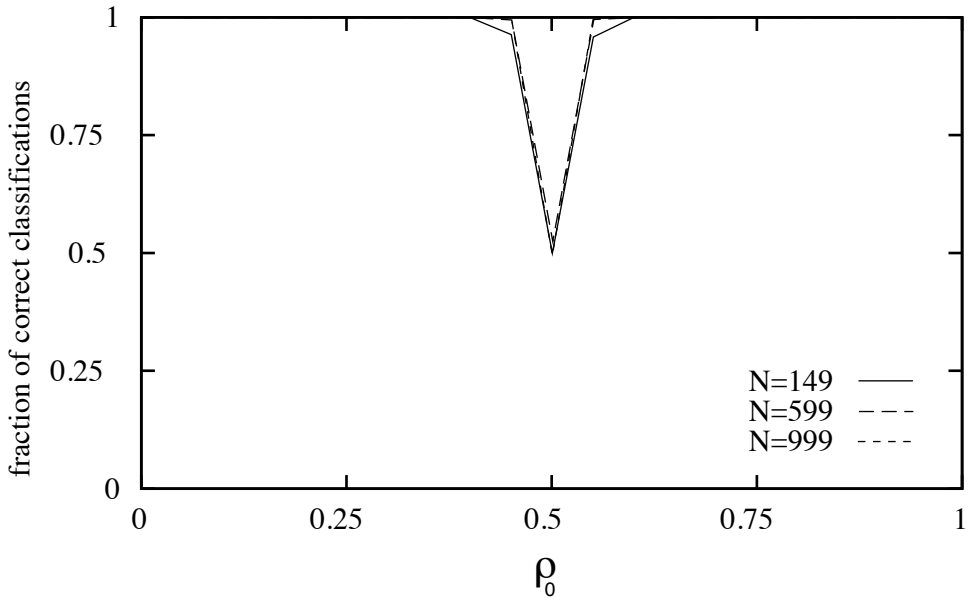

Figure 2: Experimental performance of the GKL rule as a function of ρ_0 for the $\rho_c = 1/2$ task. Performance plots are given for three lattice sizes: $N = 149$ (the size of the lattice used in the GA runs), 599, and 999. Note that the $N = 599$ and $N = 999$ curves are almost indistinguishable. (This figure differs slightly from Figure 4 in [61], since 21 density bins were used there.)

evolution that effect the classification.

It should be emphasized, however, that the GKL rule was invented not for the purpose of performing any particular computational task, but rather as part of studies of reliable computation and phase transitions in one spatial dimension. The goal was to find a rule whose behavior is robust to small errors in the rule's update of the configuration. Reliable computation in this context meant the robust storage of a single bit of information in the presence of arbitrarily small noise. A zero or one was encoded as the CA configuration being close to an all 0s or all 1s pattern, respectively. In the absence of noise, it has been proved that the GKL rule has only two attracting patterns, either all 0s or all 1s [20]. Attracting patterns here are those invariant patterns which, when perturbed a small amount, return to the same pattern under the noise-free rule. Figure 2 shows that the basins of attraction for the all-1 and all-0 patterns are not precisely the ICs with $\rho_0 > 1/2$ or $\rho_0 < 1/2$, respectively. If they did coincide then the GKL rule would exactly implement $\rho_c = 1/2$-density classification.

The GKL rule is given by an equation of motion ϕ that updates the current configuration $\mathbf{s}_t = s_t^0, s_t^1, \ldots, s_t^{N-1}$ as follows

$$s_{t+1}^i = \phi(\eta_t^i) = \begin{cases} \text{majority}[s_t^i, s_t^{i-1}, s_t^{i-3}] & \text{if } s_t^i = 0 \\ \text{majority}[s_t^i, s_t^{i+1}, s_t^{i+3}] & \text{if } s_t^i = 1 \end{cases}$$

In words, this rule says that for each neighborhood η^i of seven adjacent cells, if the state of the central cell is 0, then its new state is decided by a majority vote among itself, its left neighbor, and the cell three sites to the left. Likewise, if the state of the central cell is 1,

then its new state is decided by a majority vote among itself, its right neighbor, and the cell three sites to the right.

By listing the neighborhoods in lexicographic order, increasing from 0000000 to 1111111, the output bits of the GKL rule table can be given by a string $\mathbf{C}$ as follows.

$$\begin{aligned} \mathbf{C} \;=\; & 00000000010111110000000001011111 \\ & 00000000010111110000000001011111 \\ & 00000000010111111111111101011111 \\ & 00000000010111111111111101011111 \end{aligned} \tag{5.1}$$

This lexicographic ordering is how CA rule tables will be represented as chromosomes to the GA. As expected, the GKL rule's λ value is exactly 1/2.

Locally in space-time the GKL dynamic ϕ does not satisfy the task symmetries individually. Over one time step it does so in a composite way: $\phi(\eta) = \mathcal{F} \circ \phi(\mathcal{F} \circ \mathcal{R}\eta)$. That is, if the neighborhood pattern is spatially-reversed and the bits are flipped, the opposite output bit results. This can be seen in Figure 1. Roughly speaking, inverting white to black and black to white and spatially reversing the patterns takes the downward pointing cross-hatched region in a black sea (Figure 1(b)) to the same in a white sea (Figure 1(a)). This composite symmetry is more stringent than that required by $\mathbf{T}_{1/2}$. Moreover, under the lexicographic ordering of neighborhoods, the composite symmetry imposes constraints on pairs of output bits that are spread throughout $\mathbf{C}$. The functionality of contiguous bits is a feature to which the GA's genetic operators can be sensitive.

Typical space-time behaviors of the GKL rule for ICs with $\rho_0 < \rho_c$ and $\rho_0 > \rho_c$ were shown in Figure 1. It can be seen that, although the patterns eventually converge to fixed points, there is a transient phase during which a spatial and temporal transfer of information about local regions takes place. This local information interacts with other local information to produce the desired final state. Very crudely, the GKL rule successively classifies "local" densities with a locality range that increases with time. In regions where there is some ambiguity, a "signal" is propagated. This is seen either as a checkerboard pattern propagated in both spatial directions or as a vertical white-to-black boundary. These signals indicate that the classification is to be made at a larger scale. Note that regions centered about each signal locally have $\rho = \rho_c$. The consequence is that the signal patterns can propagate, since the density of patterns with $\rho = \rho_c$ is neither increased nor decreased under the rule.

In this way, local information processing at later times classifies larger patches of the IC. In a simple sense, this summarizes the rule's "strategy" for performing the computational task. But how is the strategy related to the CA's dynamical behavior? The overall strategy can be decomposed into the CA's intrinsic computational elements: domains, particles, and particle interactions [37]. There are three time-invariant spatially-homogeneous domains: (i) all white, $W = 0^*$, (ii) all black, $B = 1^*$, and (iii) checkerboard, $\# = (10)^* \bigcup (01)^*$. The results in [20] establish that W and B are regular attractors, as defined in [37]. When the domains are filtered out using the methods of [16], one finds that the domain boundaries form six particles, the first five of which are time-invariant. These are listed in Table 1. The types of interactions between particles are also evident when the space-time diagrams are filtered as in [16]. There are two annihilative interactions: $c + b \to \emptyset$ and $d + e \to \emptyset$. Three of the interactions are reactive: $a + d \to c$, $b + a \to e$, and $c + e \to a$. There is one

Particle	Wall Type	Velocity
a	WB	0
b	$\#W$	1
c	$W\#$	3
d	$B\#$	-1
e	$\#B$	-3
f	BW	0

Table 1: The six particles generated by the GKL CA.

spontaneous decay: $f \rightarrow d + b$. At moderate to long times, it is these particle interactions which perform the local, but "emergent" logic that classifies successively larger portions of the IC. (A more complete analysis along these lines will be presented elsewhere.) As will be seen shortly, dynamical structures like these, and a few others, will be what the GA takes advantage of in evolving CA to implement the computational task.

6. Details of the GA and CAs in Our Experiments

Following Packard [66], we used a form of the GA to evolve one dimensional $k = 2$, $r = 3$ CAs to perform the $\rho_c = 1/2$ task. The k and r values were chosen to match those of the GKL rule. The GA begins with a population of P randomly generated rules: the "chromosomes", which are strings containing the rule table output bits. Like the bit-string listing of the GKL rule given above (equation 5.5.2), the output bits are given in lexicographic order of neighborhood patterns. For $k = 2$, $r = 3$ rules, the chromosomes representing rules are of length $2^{2r+1} = 128$. The size of the rule space the GA searches is thus 2^{128}—far too large for any kind of exhaustive search.

The fitness of a rule in the population is calculated by: (i) randomly choosing I ICs that are uniformly distributed over $\rho_0 \in [0.0, 1.0]$, with exactly half with $\rho_0 < \rho_c$ and half with $\rho_0 > \rho_c$; (ii) running the rule on each IC either until it arrives at a fixed point or for a maximum of M time steps; (iii) determining whether or not the final pattern is correct—i.e., $\mathbf{s}_M = 0^N$ with $\rho_0 < \rho_c$ and $\mathbf{s}_M = 1^N$ with $\rho_0 > \rho_c$. ρ_0 is never exactly $1/2$, since N is chosen to be odd. The rule's fitness is the fraction of the I ICs on which the rule produces the correct final pattern.

This fitness function was termed "performance fitness" in [61]. It differs from "proportional fitness" in which the rule is given partial credit equal to the fraction of correct bits in the final pattern. The runs using performance fitness produced qualitatively similar results to those using proportional fitness [61], and in this paper we restrict our attention to the former. We denote the performance-fitness function using I ICs by F_I.

It should be pointed out as an aside that sampling ICs with uniform distribution over $\rho \in [0.0, 1.0]$ is highly biased with respect to an unbiased distribution of ICs, which is binomially distributed over $\rho \in [0.0, 1.0]$, and very strongly peaked at $\rho = 1/2$. However, preliminary experiments indicated a need for such a biased distribution in order for the GA to make progress in early generations. As we will discuss below, this biased distribution turns out to impede the GA in later generations because, as increasingly fitter rules are

evolved, the IC sample becomes less and less challenging for the GA.

Our GA works as follows. At each generation:

1. A new set of I ICs is generated.
2. $F_I(\phi)$ is calculated for each rule ϕ in the population.
3. The population is ranked in order of fitness. (The ranking of rules with equal fitness is decided at random.)
4. A number E of the highest fitness ("elite") rules is copied without modification to the next generation.
5. The remaining $P - E$ rules for the next generation are formed by crossovers between randomly chosen pairs of elite rules. The parent rules are chosen from the elite with replacement. The offspring from each crossover are each mutated m times.

This defines one generation of the GA; it is repeated G times for one run of the GA. An experiment consists of a set of runs with identical parameters but different random number seeds.

Our experiments used single-point crossover, which takes two strings, selects a position at random, and forms two offspring by exchanging parts of the strings before and after that position. Mutation consists of flipping a randomly chosen bit in a string.

The fitness function F_I is an estimate of the true fitness F_{2^N}. It is a random variable, in fact, since the precise value it returns for a given rule depends on the particular set of I ICs used to test the rule. Thus a rule's fitness can vary stochastically from generation to generation. For this reason, at each generation the entire population, including the elite rules, is re-evaluated on a new set of ICs.

The parameter values in our main experiment were the following. (Subsequent sections will describe other experiments in which some parameter values were modified.) For each CA in the population: $N = 149$; $I = 100$, with ICs uniformly distributed over $\rho_0 \in [0.0, 1.0]$, half with $\rho_0 < \rho_c$ and half with $\rho_0 > \rho_c$; and $M \approx 320$. Each time a CA was simulated, M was chosen from a Poisson distribution with mean 320. This mean is the measured maximum amount of time for the GKL CA to reach an invariant pattern over a large number of ICs on lattice size 149. Varying M prevents overfitting of rules to a particular M; see [61]. M was held to approximately 320 in order to force the GA to evolve rules that would perform the computation in at most the maximum amount of time taken by the GKL rule. To test the effect of this limit, we performed some experiments setting M to $10 \times N$. These runs produced rules with similar strategies to those produced when $M \approx 320$. The only difference was that, not too surprisingly, the similar strategies took longer to reach a fixed point.

In [61], I was set to 300, but we later found that setting I to 100 did not significantly change the results of our experiments and greatly reduced the required computation time. For the GA runs the chromosomes in the initial population were uniformly distributed over $\lambda \in [0.0, 1.0]$ and we set $P = 100$; $E = 20$; $m = 2$; and $G = 100$.

In GA parlance, our GA has a "generation gap"—the fraction of new strings in the next generation—of $1 - E/P = 0.8$. That is, once the population is ordered according to fitness,

the top 20% of the population—the set of elite strings—is copied without modification into the next generation. Since testing a rule on 100 ICs provides only an approximate gauge of the true fitness, this relatively small generation gap was a good way of making a "first cut" and allowing rules that survive to be tested over more ICs. Since a new set of ICs was produced every generation, rules that were copied without modification were always retested on this new set. If a rule performed well and thus survived over a large number of generations, then it was likely to be a genuinely better rule than those that were not selected, since it was tested with a large set of ICs. An alternative method would be to test every rule in each generation on a much larger set of ICs, but this would waste computation time. Too much effort, for example, would go into testing very weak rules, which can safely be weeded out early using our method. As in most GA applications, in our GA the fitness-function evaluation dominates the required computational resources.

7. The GA's Epochs of Innovation

In [61] we qualitatively described a series of "epochs of innovation" that we observed in the GA runs using the proportional fitness function. We defined the "onset" of an epoch to be the generation at which a rule with a significant innovation in strategy was discovered. The onset generation of each epoch corresponded to a marked jump in the best fitness measured in the population. In this section we describe in more detail similar phenomena that we observed in a set of experiments using F_{100} that were performed subsequent to those reported in [61]. The account of the epochs given here differs slightly from—and is more rigorous than—that given in [61]. We will distinguish the onset generation from the "takeover" generation in which all or almost all of the elite rules implement the epoch's strategy.

We performed a total of 50 runs of the GA with the parameters given above. We also performed 50 runs of the GA with no crossover, 50 runs of the GA with no crossover and an initial population clustered close to $\lambda = 1/2$, and 50 runs of a Monte Carlo search method. Some statistics from these various runs are given in Table 2. Those given in columns 2–4 will be discussed in Section 11.

Figure 3 displays the best fitness at each generation for two typical GA runs (with crossover). The best fitnesses found under F_{100} ranged from 0.9–1.0. The standard deviation of F_{100}, when run 100 times on the same rule, is approximately 0.02. Naturally, it would be preferable to use a larger number of ICs to evaluate fitness during the evolution process, but this is computationally expensive. To obtain a truer value for best fitness after each run, we evaluated each of the best rules at the last generation of each run with 10^4 randomly chosen ICs, uniformly distributed over $\rho \in [0.0, 1.0]$. This fitness function is denoted F_{10^4}. Under F_{10^4}, with the exception of one rule in the most successful of the 50 runs, all of the best fitnesses were between 0.883 and 0.936. The standard deviation of F_{10^4}, when run 100 times on the same rule, is approximately 0.002. On the most successful of the 50 runs, a rule with $F_{10^4} = 0.945$ was discovered. On one an addition run, performed subsequent to the set of 50 runs described in this paper, the GA discovered a rule with an even higher fitness—$F_{10^4} = 0.954$. These two rules—to be discussed later—were substantially different from those found in typical runs; one of them had behavior very similar to that of the GKL rule. Recall that under F_{10^4}, the fitness of the GKL rule is 0.972. Under F_{10^4}, this is a significantly higher level of fitness than the level achieved in any of the GA runs. Therefore, the GA did not

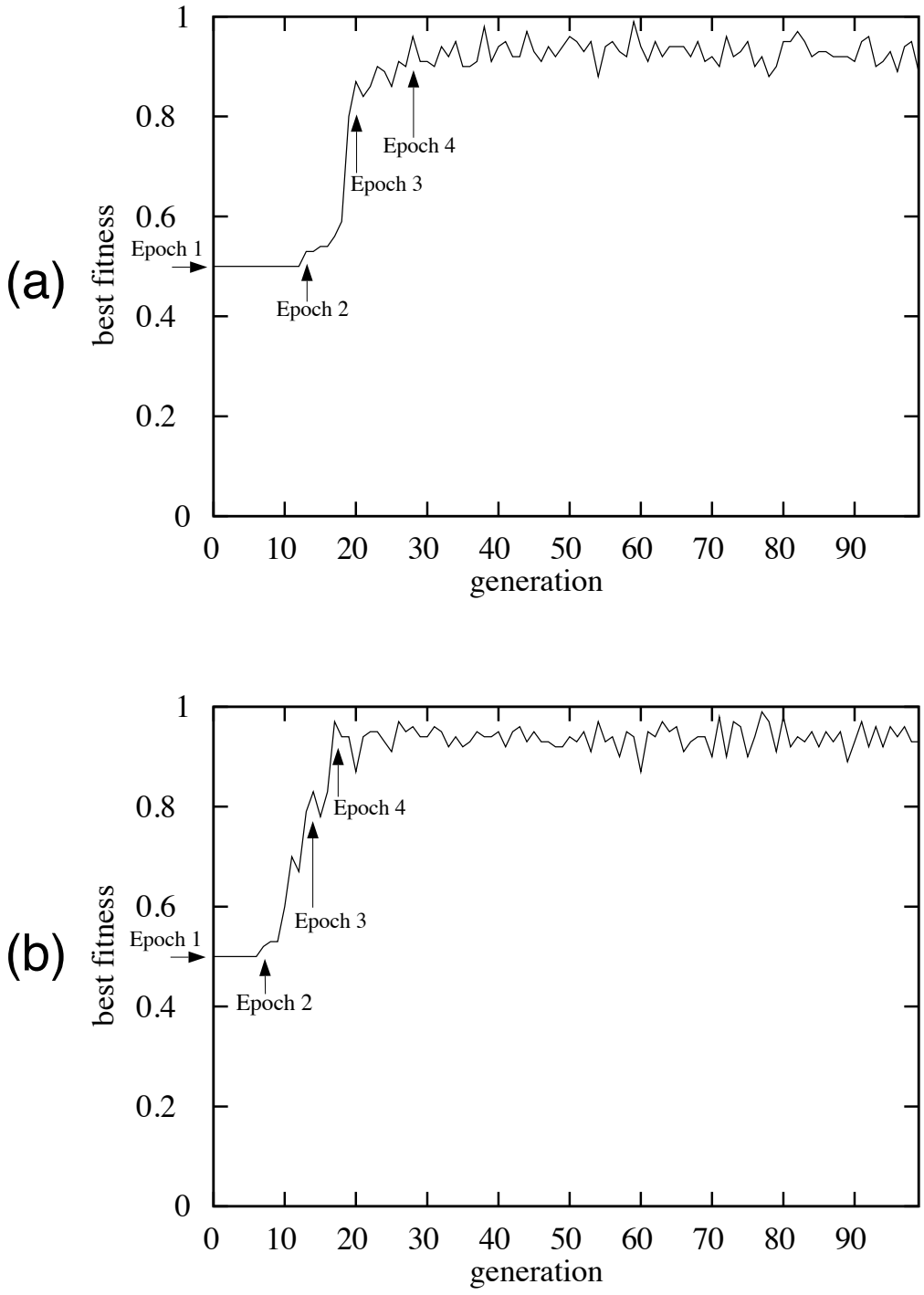

Figure 3: Best fitness versus generation for two typical runs. The onsets of four epochs of innovation—each corresponding to the generation discovery of a new, fitter strategy—are marked.

	GA, xover	GA, no xover	GA, no xover, initpop 1/2	Monte Carlo
Runs reaching Epoch 3	46/50 (92%)	13/50 (26%)	22/50 (44%)	21/50 (42%)
Runs used in averages	44/50	13/50	22/50	21/50
T_2	3.6 (3.3)	65.9 (12.9)	39.3 (20.6)	44.6 (25.6)
$T_3 - T_2$	3.8 (2.8)	9.9 (5.3)	7.6 (3.5)	2.5 (2.6)

Table 2: Fraction of runs reaching Epoch 3, fraction of runs used to compute averages (for "GA, xover" case, two outlyer runs were ommitted), mean generations to onset of Epoch 2 (T_2), and mean length of Epoch 2 in generations ($T_3 - T_2$) for those runs reaching Epoch 3 by generation 99 in four different experiments. Standard deviations are given in parentheses. The experiments are: the original experiment ("GA, xover"), an experiment in which crossover was turned off ("GA, no xover"), an experiment in which crossover was turned off and the strings in the initial GA population all had $\lambda \approx 1/2$, and an experiment in which a Monte Carlo method instead of a GA was used to search the space of CA rules. The last three will be discussed in Section 11.

succeed in evolving the GKL rule or a rule at an equal level of performance, though on one run it came close. On the other runs, the GA evolved a different set of strategies than ones that might be expected from the GKL rule. Here we will primarily discuss the GA's typical behavior on the ρ_c task rather than its much rarer but more successful behavior.

The two plots in Figure 3 have four similar large-scale features, which turn out to correspond to four epochs of innovation. As was defined in [61], the onset of each epoch corresponds to the discovery of a new, higher-fitness strategy. The onset generation of each new epoch was determined by examining the actual strategies carried out by the elite rules in each run—i.e., the actual space-time dynamics of each rule. The onset generations are indicated in Figures 3. At generation 0, the onset of Epoch 1, the best fitness is 0.5 and remains at that level for several generations. Then there is a gradual or sharp rise in best fitness to $F_{100} \approx 0.53 - 0.70$ at the onset of Epoch 2. This is followed by a sharp rise to 0.80 or higher at the onset of Epoch 3. The sharp rise is followed by a sharp or gradual rise to 0.9 or higher, corresponding to the onset of Epoch 4. The fitness then stays relatively constant, with moderate fluctuations arising from the stochastic nature of F_{100}. These same large-scale features are seen in the best-fitness time histories for almost every run. We examined rules in each run at different stages and, in almost all runs, observed roughly the same progressions through epochs and similar strategies at each epoch. Out of the 50 GA runs performed, 46 displayed a best-fitness plot similar to those in Figure 3. The only major differences were the generation of onset of Epoch 2 (T_2) and the length of Epoch 2 ($T_3 - T_2$, where T_3 is the generation of onset of Epoch 3). On a small number of runs, the GA jumped from Epoch 1 directly to Epoch 3 without passing through Epoch 2.

T_2 and T_3 are defined by the emergence of new strategies, but they can be determined from best fitness values. We found that, with very few execptions, T_2 corresponded to the best fitness rising to 0.52 or higher, and T_3 corresponded to the best fitness rising to 0.78 or

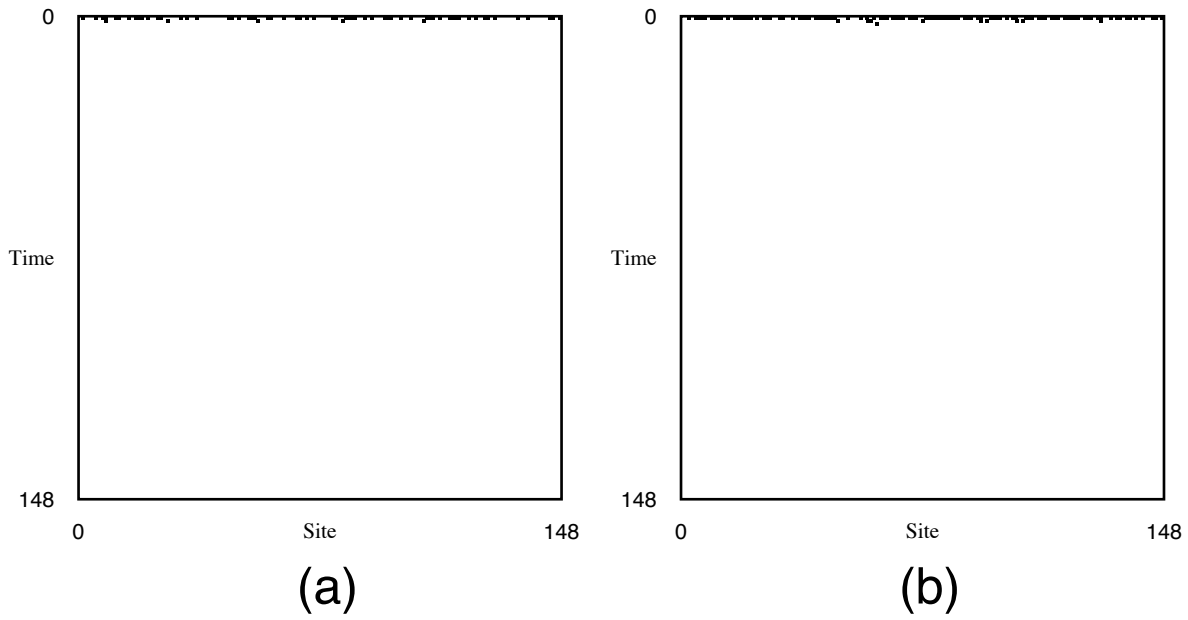

Figure 4: Two spacetime diagrams for an Epoch 1 rule with $\lambda \approx 0.05$. In (a), $\rho_0 \approx 0.40$ and in (b), $\rho_0 \approx 0.67$.

higher. We used these indicators to calculate the mean values of T_2 and $T_3 - T_2$ over 44 of the 50 runs. These means are given in the first column of Table 2. Two "outlier" runs were omitted from these averages. In these, $T_2 = 48, T_3 = 61$ and $T_2 = 49, T_3 = 58$ respectively. The best fitnesses in the remaining four runs never went beyond 0.5; we assume that if those runs had been allowed to continue, Epochs 2–4 eventually would have been reached.

The common strategies implemented by the best rules in different epochs in almost all runs are illustrated by the space-time diagrams in Figures 4–7. These are discussed in order below.

Epoch 1: Best rules specialize on low ρ_0 or high ρ_0

In Epoch 1 there are two best-performing strategies: rules that always relax to a fixed point of all 0s and rules that always relax to a fixed point of all 1s. Figure 4 illustrates the former strategy on two ICs with low and high ρ_0, respectively. Since exactly half the ICs at each generation have $\rho_0 < \rho_c$ and exactly half have $\rho_0 > \rho_c$, each of these strategies is correct on exactly half the ICs, so each has fitness 0.5. This default behavior is hardly worthy of the name "strategy", but these rules perform significantly better than randomly chosen rules, which classify almost no ICs correctly. Since the rules in the initial populations are uniformly distributed over $\lambda \in [0.0, 1.0]$, the initial population always contains both very low and very high λ rules, which tend to have this behavior. This is why the best fitness in the initial population is almost always 0.5.

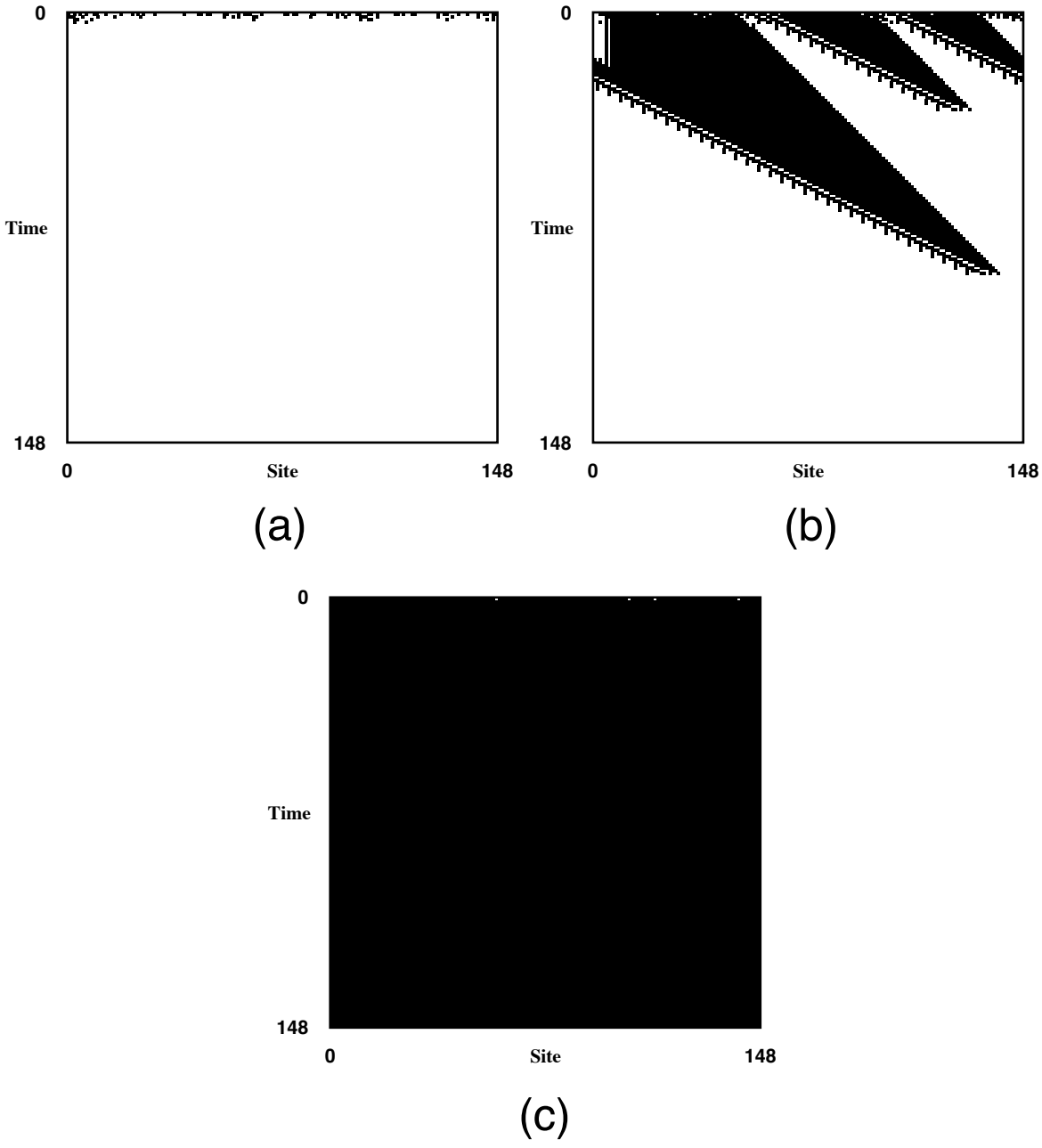

Figure 5: Three spacetime diagrams for an Epoch 2 rule with $\lambda \approx 0.33$. In (a), $\rho_0 \approx 0.37$, in (b), $\rho_0 \approx 0.86$, and in (c), $\rho_0 \approx 0.97$.

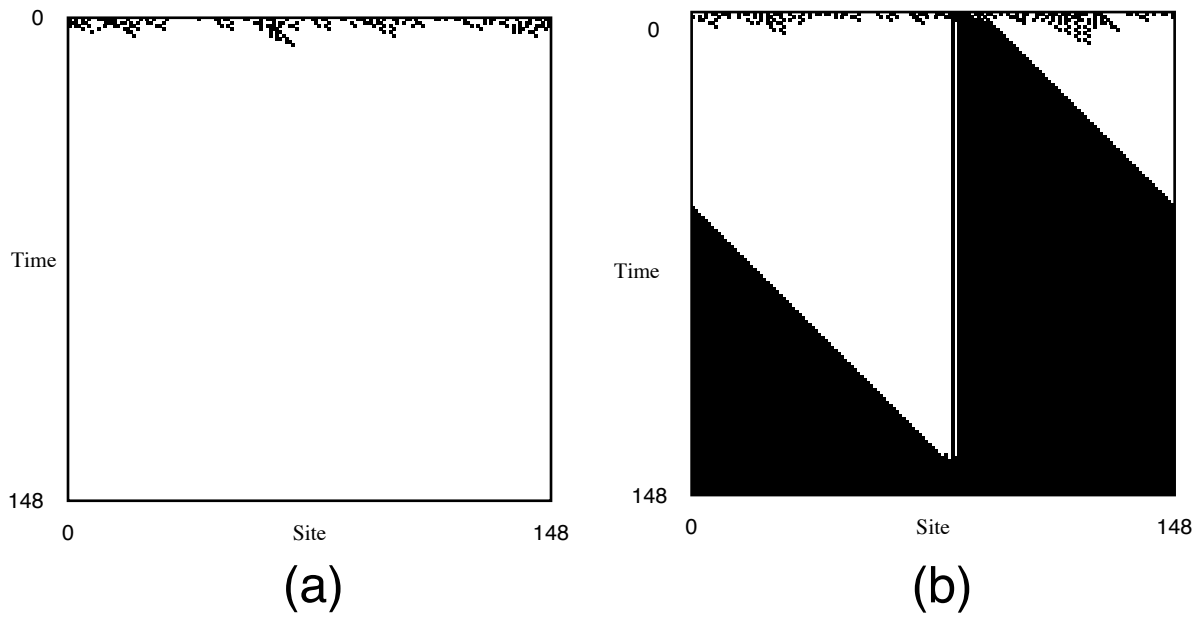

Figure 6: Two spacetime diagrams for an Epoch 4 rule with $\lambda \approx 0.38$ that implements Strategy 1. In (a), $\rho_0 \approx 0.41$ and in (b), $\rho_0 \approx 0.52$.

Epoch 2: Best rules correctly classify additional "extreme" ρ_0

At Epoch 2, the GA discovers rules that, while similar in behavior to Epoch 1 rules, correctly classify some additional ICs with extreme ρ_0. The behavior of one such rule is illustrated in Figure 5. Like the rule illustrated in Figure 4, this rule is a "low-ρ_0 specialist". But unlike the previous rule, it correctly classifies some very high ρ_0 ICs as well. In Figure 5(a), $\rho_0 < \rho_c$ and the CA quickly relaxes to all 0s. In Figure 5(b), $\rho_0 > \rho_c$ and the CA again relaxes to all 0s (a misclassification), but information from high-density blocks in the IC persist for some time. In Figure 5(c), $\rho_0 \gg \rho_c$ and the pattern is correctly classified. An Epoch 2 rule's additional correct classifications of very high (or very low) ρ ICs yields a slightly higher fitness, as seen in Figure 3. On approximately half the runs the GA discovers Epoch 2 low-ρ_0 specialists and on the other half it discovers Epoch 2 high-ρ_0 specialists. The strategies are almost never found together in the same run. Rules in Epoch 2 have fitnesses ranging from 0.51 to about 0.75, depending on how many additional high-density ICs are classified correctly and on the particular set of ICs being used at a given generation.

Epochs 3 and 4: Expanding blocks of 0s or 1s

Epoch 3 is characterized by a major innovation discovered by the GA. As in Epochs 1 and 2, there are two opposite strategies in Epoch 3:

- *Strategy 1:* Relax to a fixed point of all 0s unless there is a sufficiently large block of adjacent (or almost adjacent) 1s in the IC. If so, expand that block.

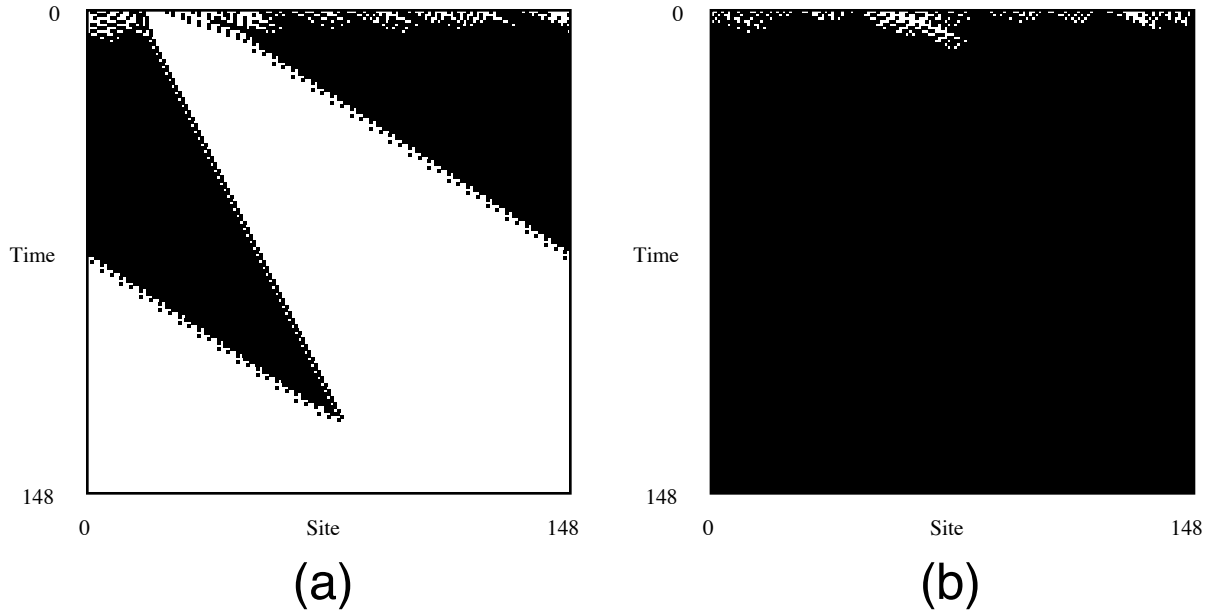

Figure 7: Two spacetime diagrams for an Epoch 4 rule with $\lambda \approx 0.59$ that implements Strategy 2. In (a), $\rho_0 \approx 0.39$ and in (b), $\rho_0 \approx 0.54$.

- *Strategy 2:* Relax to a fixed point of all 1s unless there is a sufficiently large block of adjacent (or almost adjacent) 0s in the IC. If so, expand that block.

The meaning of "sufficiently large" varies from rule to rule, and generally ranges from around 6 to 11 cells. (Note that "sufficiently large" can be larger than the neighborhood size $2r+1 = 7$. This can occur via the interaction between adjacent neighborhoods on the lattice.) As will be seen, in the higher-fitness rules, the size of blocks that are expanded is tuned to be a good predictor of high or low density for $N = 149$.

Epoch 3 begins with the discovery of such a rule, which typically has fitness $F_{100} \approx 0.8$. During Epoch 3, the GA discovers variants on the original rule and small improvements to these rules, having the effect of raising their fitnesses to $F_{100} \approx 0.9$. Epoch 4 begins when no additional improvements are made. From that time on, the best fitnesses remain approximately constant, though there is moderately high variation in F_{100} as seen in Figure 3. Two examples of such rules from Epoch 4 are given in Figures 6 and 7. The rule in Figure 6 implements Strategy 1. In 6(a), $\rho_0 < \rho_c$, and the CA quickly relaxes to all 0s. In 6(b), $\rho_0 > \rho_c$, and there is a sufficiently large block of 1s, which the CA expands toward the right until the configuration reaches a fixed point of all 1s. Both ICs are correctly classified. Figure 7 displays a rule with Strategy 2. In 7(a), $\rho_0 < \rho_c$, and there is a sufficiently large block of 0s which is expanded. In 7(b), $\rho_0 > \rho_c$, and the configuration quickly relaxes to all 1s. Again, both ICs are correctly classified.

The best rules in Epochs 3 and 4 are specialists for low or high ρ_0, but rather than ignoring the opposite half of the ICs as in Epoch 1 or only dealing with special extreme cases as in Epoch 2, these rules deal with the other half of the ICs by expanding sufficiently large blocks of the non-default state. In this way they obtain a marked increase in fitness.

In effect, the strategy of Epochs 3 and 4 use the presence or absence of such blocks as local predictors of the global ρ_0.

Typical errors made in Epoch 3 are illustrated in Figure 8. In 8(a), an Epoch 3 rule expands blocks that are too small, resulting in an incorrect classification for $\rho_0 < \rho_c$. In 8(b), another Epoch 3 rule expands blocks too slowly from an IC with $\rho_0 > \rho_c$, eventually reaching a fixed point of all 1s but not by the maximum allotted number M of iterations. (In the figure, $M = 149$; in our experiments, $M \approx 320$.) This also results in a failure to correctly classify within the given time. In Figure 8(c), a third Epoch 3 rule *creates* a block not present in $\mathbf{s}_0$ and expands it, resulting in a misclassification for $\rho_0 \ll \rho_c$. Such errors are largely corrected by Epoch 4, though even the best Epoch 4 rules still misclassify a number of ICs. The experimental performance of an Epoch 4 rule as a function of ρ_0 is given in Figure 9. This plot was made in the same way as that in Figure 2. Like the GKL rule, most of the classification errors occur close to $\lambda = 1/2$, though the width of the error region is much larger here than that seen for the GKL rule in Figure 2.

8. λ, Selection, and Combinatorial Drift

Up to this point we have described the GA's behavior in terms of (i) the large-scale time history of the best fitness, (ii) the strategy epochs in this time history, and (iii) the details of the actual strategies discovered at each epoch. This description examined properties of the best individual rules rather than properties of the entire elite population. We now present an intermediate-level description of the GA's behavior in terms of the distribution of λ in the elite population over time. This will reveal how population-level structures emerge in the different epochs and will aid in understanding the mechanisms by which the GA progresses through the epochs.

The strategies described in the previous section each have two opposite instantiations—one that specializes for low ρ_0 and the other that specializes for high ρ_0. On a given run, the GA discovers one or the other class of strategies, but not both. Figure 10 displays histograms of rule frequency versus λ for the elite rules at generation 99 in two typical runs. In 10(a) all the elite rules have $\lambda < 1/2$; this run resulted in a population of low-ρ_0 specialists which implement Strategy 1. In 10(b), all the elite rules have $\lambda > 1/2$; this run resulted in a population of high-ρ_0 specialists which implement Strategy 2.

Figure 11 is a mosaic of histograms plotting the frequency of elite rules versus λ, where the elite rules from 44 different runs are merged together. Each histogram therefore contains counts from $20 \times 44 = 880$ elite rules. These 44 runs were the same ones for which statistics are given in Table 2. The figure shows how the structure of the elite populations changes with time. In generation 0, the elite rules are clustered close to $\lambda = 0$ and $\lambda = 1$. Why is this? Recall that in each run, the rules in the initial population are uniformly distributed over $\lambda \in [0.0, 1.0]$. Most of these rules have very low fitness; the best strategies ϕ found are those of Epoch 1 ("always relax to all 0s" or "always relax to all 1s"). These have $F_{100}(\phi) = 0.5$. At generation 0 the rules implementing these strategies have either very low or very high λ—for example, a rule with $\lambda = 0$ maps all neighborhoods to 0 and thus implements the all-0s strategy. This results in the peaks at extreme λ in the initial generation.

Very quickly, however, the elite populations move away from these extremes and towards

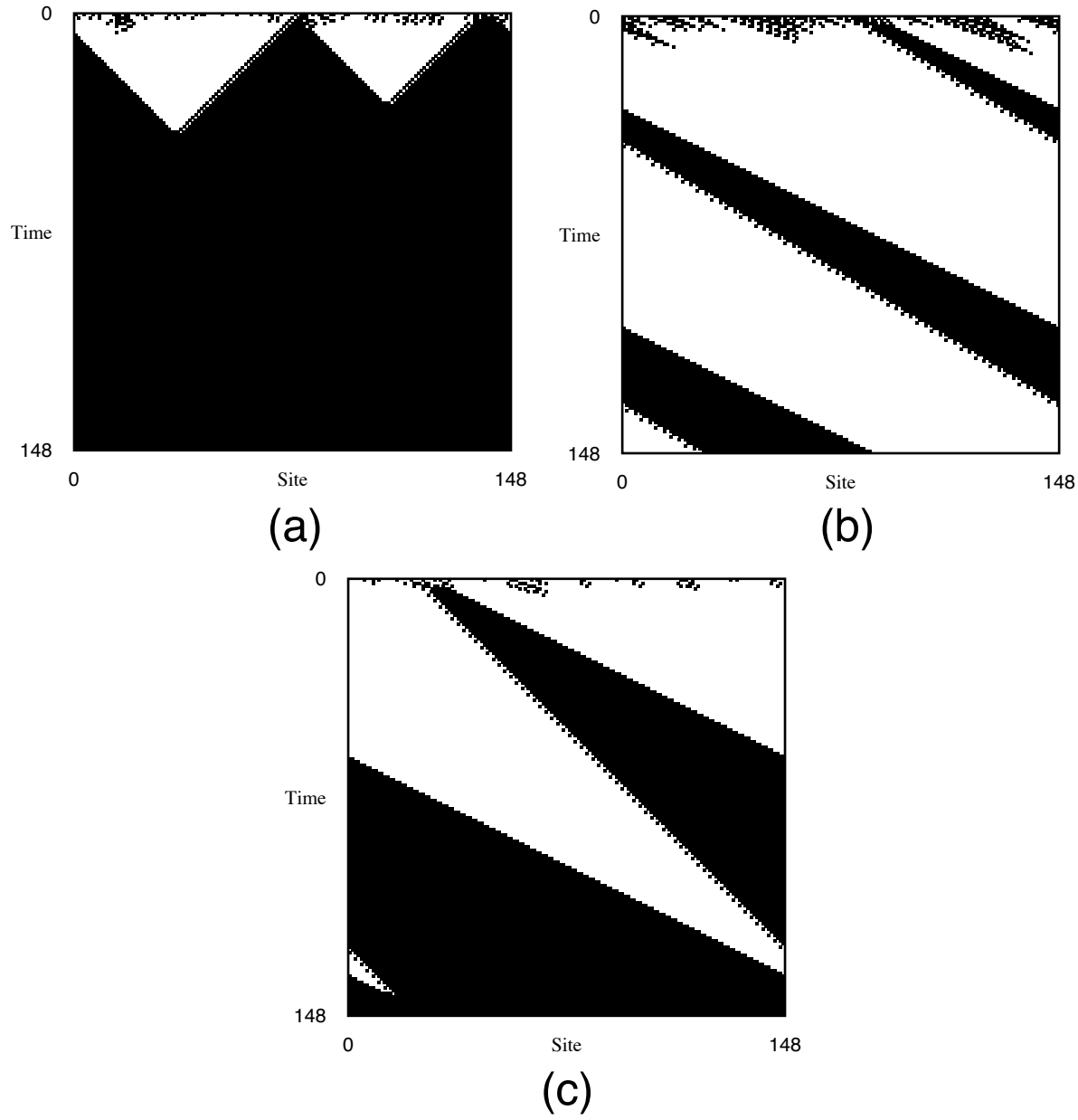

Figure 8: Three typical errors made by Epoch 3 rules. In (a), a rule with $\lambda \approx 0.49$ incorrectly expands blocks in an IC with $\rho_0 \approx 0.38$. In (b), a rule with $\lambda \approx 0.42$ expands blocks too slowly on an IC with $\rho_0 \approx 0.56$. In (c), a rule with $\lambda \approx 0.52$ *creates* a block that was not present in $\mathbf{s}_0$ with $\rho_0 \approx 0.19$, and expands it. All these examples led to incorrect classifications.

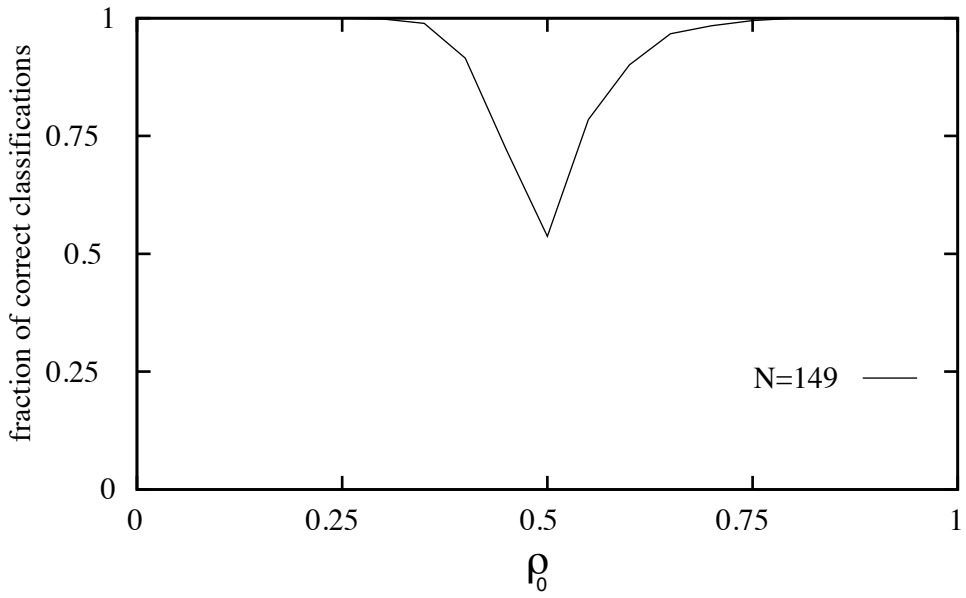

Figure 9: Performance of an Epoch 4 rule, plotted as a function of ρ_0, with $N = 149$. This rule has $\lambda \approx 0.59$.

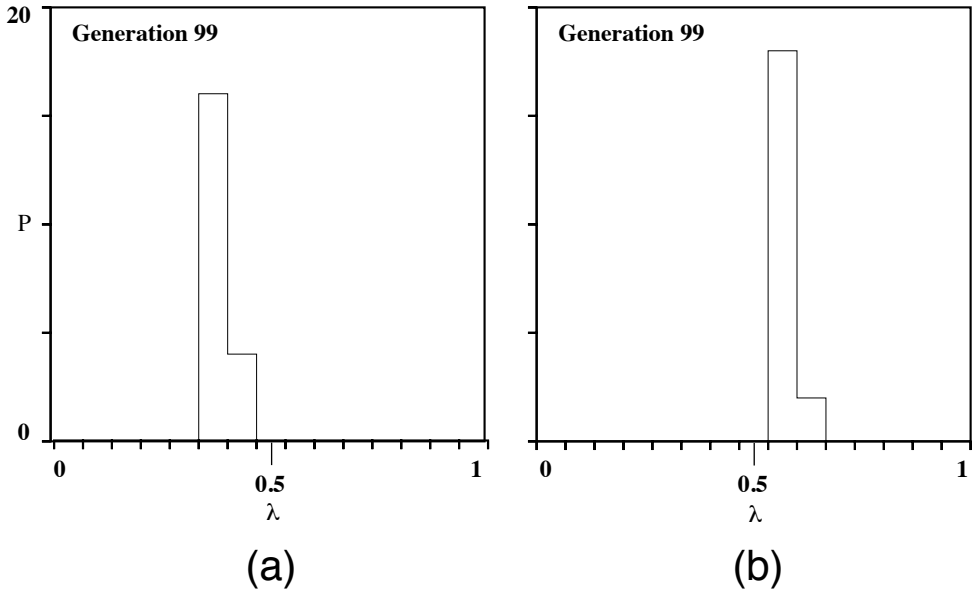

Figure 10: Histograms of the frequency of elite rules as a function of λ for two typical runs. The x-axis is divided into 15 bins of length 0.0667 each. In (a), all the elite rules have $\lambda < 1/2$; in (b), all the elite rules have $\lambda > 1/2$.

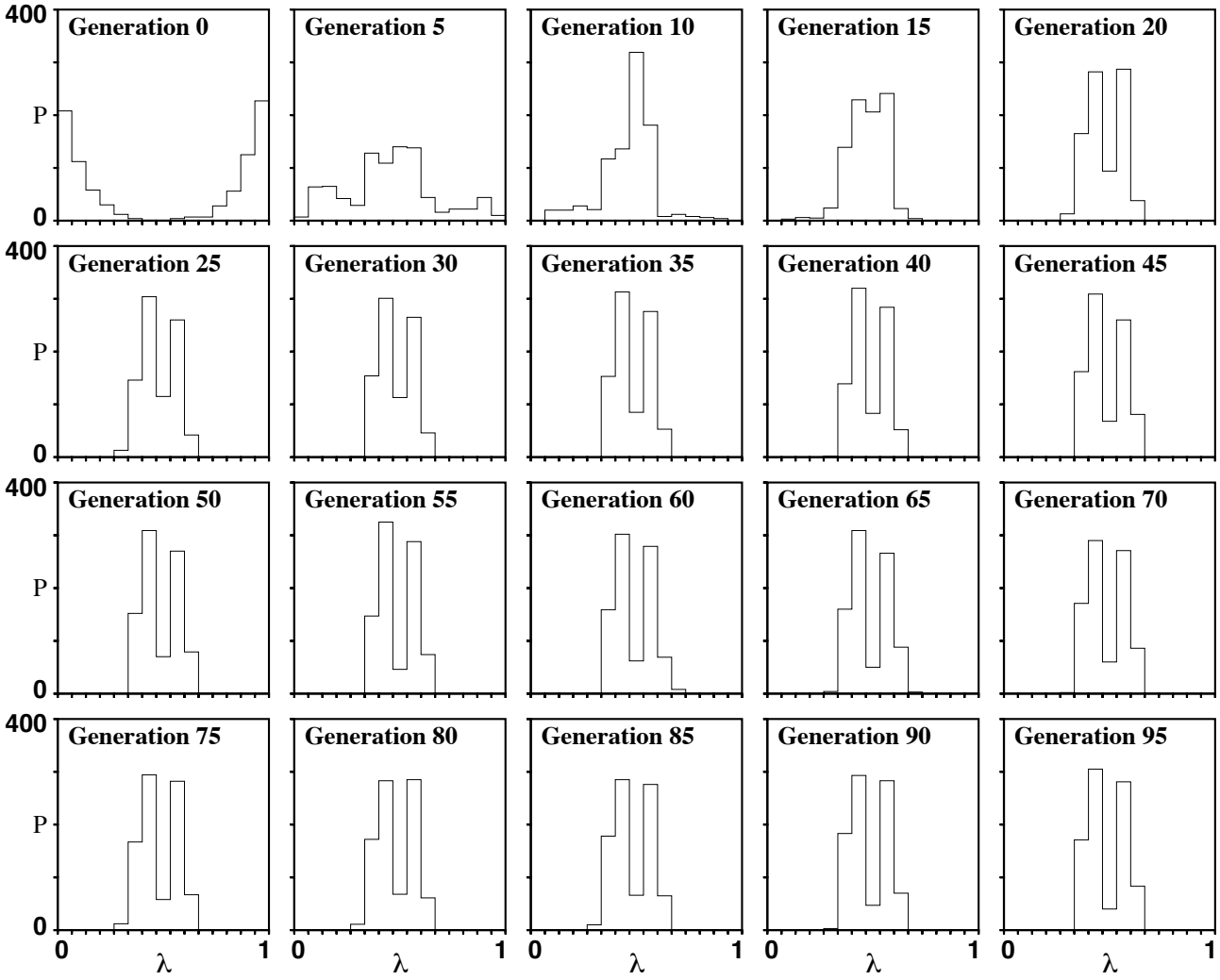

Figure 11: Frequency of elite rules versus λ given every five generations, merged from 44 GA runs with fitness function F_{100}.

$\lambda = 1/2$. The populations peak there between generations 5 and 10. The values for T_2 and $T_3 - T_2$ given in the first column of Table 2 indicate that the appearance of the peak roughly corresponds with Epochs 2 and 3. By generation 15 the distribution has changed again—it now has two peaks on either side of $\lambda = 1/2$. By generation 20, these two peaks have grown and the dip at $\lambda = 1/2$ has deepened. After generation 20 or so, the distribution does not change appreciably.

The two peaks on either side of $\lambda = 1/2$ result from merging together the elite rules from runs with low-ρ_0 specialists and runs with high-ρ_0 specialists. Each peak represents rules from runs of one or the other type, as seen in Figure 10. What is seen clearly in Figure 11 at generation 15 is a symmetry breaking on the part of the GA: as we discussed above, the $\rho_c = 1/2$ task requires certain symmetries, in particular, the 0–1 exchange symmetry $\mathcal{F}$ that requires $\lambda = 1/2$ for high performance. The GA breaks this symmetry, producing rules on either side of $\lambda = 1/2$.

The spatial-reverse symmetry $\mathcal{R}$ is also broken as seen in Figures 5(b), 6(b), 7(a), 8(b), and 8(c). Since this need not lead to a bias in λs, it is not directly reflected in the histograms we will use here; another coordinate would be more appropriate for monitoring this symmetry breaking.

To understand the degree to which selection for performance fitness rather than intrinsic effects of crossover and mutation cause the effects seen in Figure 11, we performed 50 runs of the GA with *random* selection. Everything about the GA was the same as in the original experiment, except that F_{100} was not calculated and instead at each generation fitnesses were assigned at random. Figure 12 is a mosaic of histograms from these runs. Each histogram plots the frequency of elite rules at the given generation as a function of λ. Since the fitness function is not calculated, all effects seen in Figure 12 are due to the combined intrinsic effects of random selection, crossover, and mutation, which we term "combinatorial drift". As can be seen, by generation 10 the population has largely drifted to the region of $\lambda = 1/2$ and this clustering becomes increasingly pronounced as the run continues.

This drift to $\lambda = 1/2$ is related to the combinatorics of the space of bit strings. For binary CA with neighborhood size n $(= 2r + 1)$, the space consists of all 2^{2^n} binary strings of length 2^n. Denoting the subspace of CAs with a fixed λ and n as $\mathrm{CA}(\lambda, n)$, we see that the size of the subspace is binomially distributed with respect to λ:

$$|\mathrm{CA}(\lambda, n)| = \left(\begin{array}{c} 2^n \\ \lambda 2^n \end{array} \right)$$

where $|S|$ denotes the size of set S. The distribution is symmetric in λ and tightly peaked about $\lambda = 1/2$ with variance $2^{-n}/4$. Thus, the vast majority of rules is found at $\lambda = 1/2$. Using Sanov's theorem, for example, with $r = 3$ there are about 10^{-16} fewer rules at $\lambda_c \approx 0.146$ [49] than at $\lambda = 1/2$. The steepness of the binomial distribution near its maximum gives an indication of the magnitude of the drift "force". Note that the last histogram in Figure 12 gives the GA's rough approximation of this distribution.

Drift is thus a powerful force moving the population to cluster around $\lambda = 1/2$. It is partially responsible for the initial clustering around $\lambda = 1/2$ seen in Figure 11. However, the distribution in early generations (e.g., generation 10) in Figure 11 is more sharply peaked at $\lambda = 1/2$ than that for the same generation in Figure 12, indicating that there is an additional

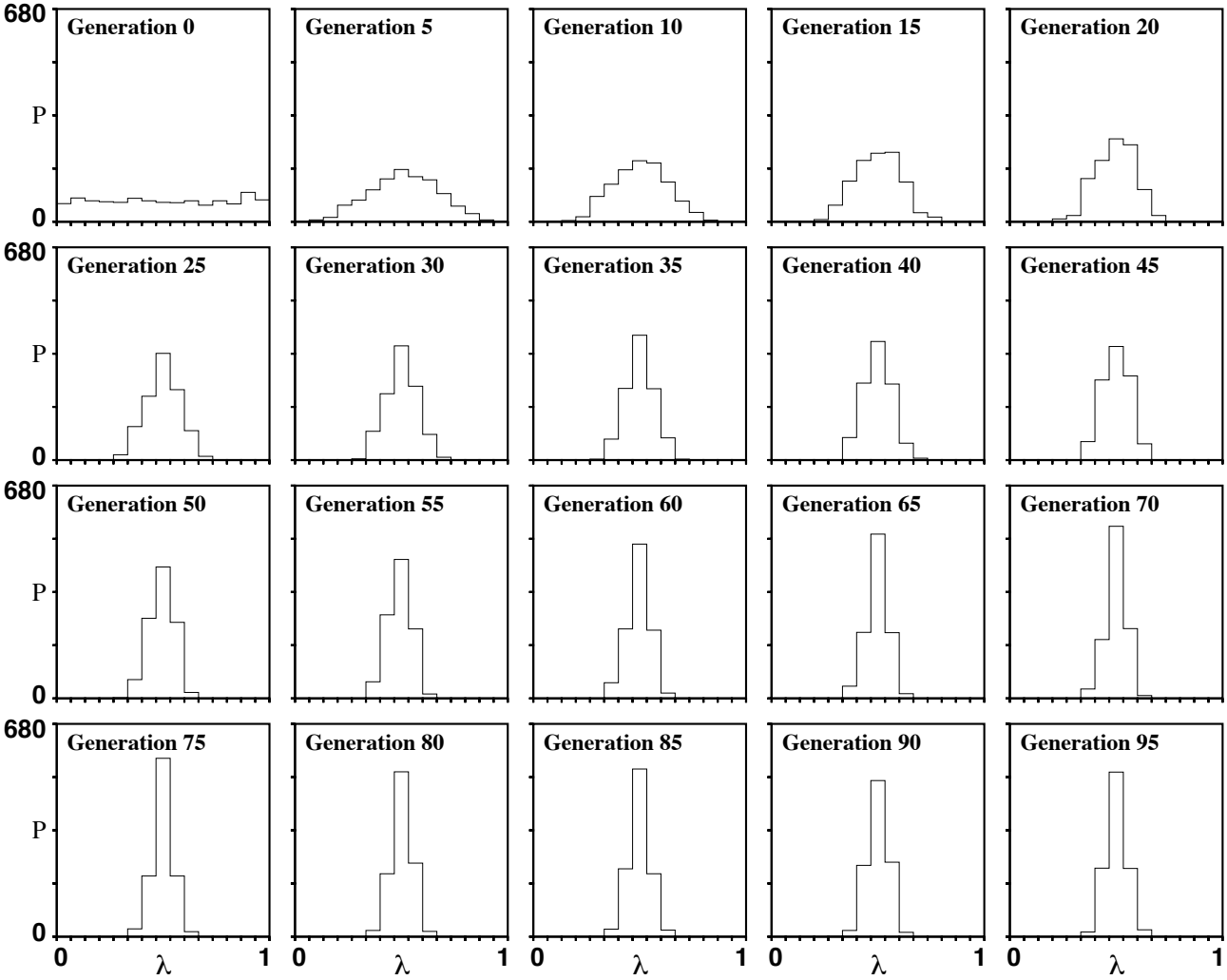

Figure 12: Frequency of elite rules versus λ given every five generations, merged from 50 GA runs with random fitnesses assigned at each generation.

clustering force due to selection for performance fitness. The striking difference in the two distributions in later generations shows that the symmetry breaking seen at generation 15 is due to selection forces rather than drift forces.

This completes our overview of the phenomena that were observed in the GA runs. In the remainder of this paper, we answer the following questions:

Major questions:

- How are the strategies in each epoch implemented in the rule tables?
- In what way are the macroscopic properties of the λ distributions presented in Figure 11 related to the four epochs? In particular, what causes the symmetry breaking seen at generation 15 in Figure 11?
- By what mechanisms does the GA produce the behavior that we have observed? In particular, what are the mechanisms underlying the epochs of innovation?
- What impedes the GA from discovering better strategies? In particular, what prevents the GA from discovering the GKL rule or similar rules, except on rare occasions?

9. Implementation of Strategies

To investigate how the strategies in each epoch are implemented in the rule tables, we will define a new statistic over rule tables, denoted $A_s(d)$, that measures the degree of agreement of output bits with neighborhood densities. (This statistic is similar, though not identical, to Gutowitz's mean-field theory for CAs [35].) Let the density of symbol s in a neighborhood pattern η be denoted by $\rho^s(\eta)$. For example, $\rho^1(0000001) = 1/7$; $\rho^0(0000001) = 6/7$. Let the set of neighborhoods η for which $\rho^s(\eta) \geq d$ be denoted by $\mathcal{N}_s(d)$, where $d \in [0.0, 1.0]$ is some constant, then

$$\mathcal{N}_s(d) = \{\eta : \rho^s(\eta) \geq d\}$$

Note that $|\mathcal{N}_s(d)|$ is monotonically decreasing from 128 to 1 as d varies from 0 to 1. For $k = 2, r = 3$ rules, $\mathcal{N}_1(1/2)$ is the set of 64 neighborhoods with a majority of 1s in the neighborhood pattern, and $\mathcal{N}_1(6/7)$ is the set of eight neighborhoods with at least 6 1s in the neighborhood pattern. (Note that $|\mathcal{N}_1(d)| = |\mathcal{N}_0(d)|$ for all d, by the 0-1 exchange symmetry.)

Then for a given rule table ϕ, consider the set $\mathcal{M}_s(d)$ of neighborhoods $\eta \in \mathcal{N}_s(d)$, that map to output symbol s:

$$\mathcal{M}_s(d) = \{\eta : \eta \in \mathcal{N}_s(d) \text{ and } \phi(\eta) = s\}$$

The "s-agreement" $A_s(d)$ is the fraction of these neighborhoods; that is

$$A_s(d) = \frac{|\mathcal{M}_s(d)|}{|\mathcal{N}_s(d)|}$$

Note that $A_1(0) = \lambda$. $A_1(1/2)$ is the fraction of rule-table neighborhoods with a majority of 1s whose output bits are 1. For the "majority" rule given in Section 2, $A_1(1/2) = 1$, since this is precisely how the rule was defined. For the GKL rule, $A_1(1/2) = 0.75$.

The temporal development of $A_s(1/2)$ and $A_s(6/7)$ for $s = 0$ and $s = 1$ averaged over the elite population helps identify how the different epochs' strategies are implemented. For rules with 7-bit neighborhoods, $|\eta| = 7$, $A_s(1/2)$ measures the degree to which $\phi(\eta)$ agrees with $\rho^s(\eta)$ in neighborhoods that have at least 4 s's ("majority agreement"). Similarly, $A_s(6/7)$ measures the degree to which $\phi(\eta)$ agrees with with $\rho^s(\eta)$ in neighborhoods that have at least 6 s's ("super-majority agreement").

> *Figure 13 caption:* $A_0(d)$ statistics for a run that resulted in low-ρ_0 specialists. (a) Mean and standard deviation of elite 0-agreement $A_0(6/7)$ versus generation. The mean elite fitness $\overline{F}_{\text{elite}}$ is plotted for reference. (b) Mean and standard deviation of elite 0-agreement $A_0(1/2)$ versus generation. Mean elite fitness $\overline{F}_{\text{elite}}$ is plotted for reference. (c) Scatter plot of elite λ values. The takeover generations of Epochs 1, 2, and 3 are marked by vertical dashed lines. In this run the onset generations of Epochs 1, 2, and 3—the generation in which the first instance of a new strategy was discovered (not shown here since the fitness shown is an average over the elite)—were 0, 13, and 19, respectively.

Figure 13 displays plots for $\{s = 0\}$-agreement and Figure 14 displays plots for $\{s = 1\}$-agreement, for one run that resulted in low-ρ_0 specialists. This run is atypical in that the onset of Epoch 2 is later than average, as is the time from Epoch 2 onset to Epoch 3 onset (cf. Table stats-table, column 1). This run was chosen for illustration because the relatively stretched-out time scales make it easier to see how changes in $A_s(d)$ correlate with epochs. The correlations seen in this run were also seen in almost all of the other runs.

Figures 13(a) and 14(a) plot the mean and standard deviation σ of $A_s(6/7)$ over the elite rules at each generation. For reference, the mean fitness $\overline{F}_{\text{elite}}$ of the elite rules is also plotted. The takeover generations of Epochs 1, 2, and 3 in the elite are marked by vertical dashed lines. Recall that the takeover generation of a given epoch is defined as the first generation at which all or almost all elite rules implement the strategy associated with that epoch. For example, here the Epoch 1 takeover generation is the generation at which all elite rules implement the "always relax to all 0s" strategy.

The $A_0(d)$ statistics reveal how the Epoch 1 strategies ("always relax to all 0s") are implemented. In Figure 13(a), $A_0(6/7)$ rises quickly and saturates at 1.0 (with $\sigma = 0.0$) at generation 9. The initial sharp rise does not indicate the onset of a new epoch, since no new strategy is discovered. Rather, the rise is due to the depletion of high-λ, high-ρ_0 specialist rules, as can be seen in Figure 13(c), a scatter plot of the elite λ values at each generation. At generation 0 the λ values are clustered at the two extremes, but they quickly consolidate at low values, since the chromosomes with $\lambda = 0$ and $\lambda = 1$, to take the extremes, are destroyed by recombination and mutation without compromising fitness. The population consists entirely of low values by generation 9, at which time $A_0(6/7)$ is essentially saturated at 1.0. This saturation indicates that in all rules in the elite, the eight neighborhood patterns consisting of at least six out of seven 0s map to $s = 0$. This is a necessary condition for implementing the Epoch 1 strategy of "always relax to all 0s".

The saturation of $A_0(6/7)$ marks the takeover of the Epoch 1 strategy. From the onset of Epoch 1 at generation 1 to the takeover at generation 9, the shape of the 0-agreement

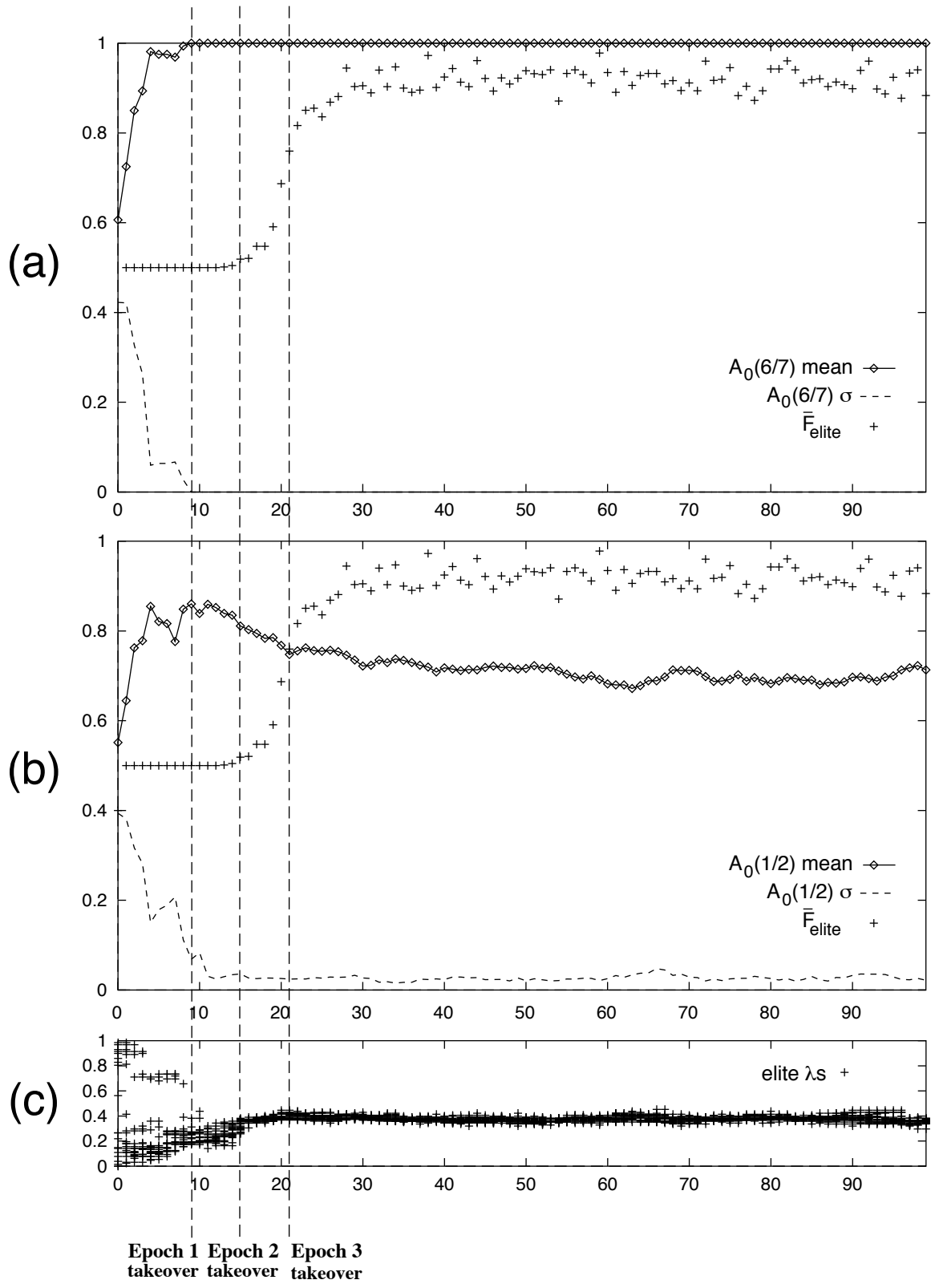

Figure 13: Caption is given in text.

statistics indicates the rate at which the strategy spreads in the population. As Figures 13(a) and 13(b) show, this occurs without an increase in mean fitness $\overline{F}_{\text{elite}}$. The extreme λ rules die out, but their strategy lives on in later generations.

Figure 13(b) plots the mean and σ of $A_0(1/2)$ over the elite rules at each generation. The mean $A_0(1/2)$ also rises quickly during Epoch 1 and σ sharply decreases, particularly at generation 9, indicating that most neighborhoods with even four and five 0s have 0 as the output bit. A high $A_0(1/2)$ is necessary for the "always relax to all 0s" strategy, since both low and high ρ_0 configurations must relax to all 0s. The rise in mean $A_0(1/2)$ corresponds to more and more rules implementing this strategy. The sharp drop in σ corresponds to this consolidation. $A_0(1/2)$ begins to fall close to the Epoch 2 takeover: when the dominant strategy is no longer "always relax to all 0s", the high level of agreement is no longer necessary, and some of the previously agreeing bits can be mutated from 0 to 1 without harm to an Epoch 2 strategy.

The $A_1(d)$ statistics reveal how the Epoch 2 strategies are implemented. Figure 14(a) plots the mean and σ of the 1-agreement $A_1(6/7)$ averaged over the elite rules at each generation of the same run. During Epoch 1, the mean $A_1(6/7)$ is noisy, with σ remaining above 0.15, though σ drops sharply at the Epoch 1 takeover, corresponding to the loss of high-λ, high-ρ_0 specialist rules. Four generations after the Epoch 1 takeover, the mean $A_1(6/7)$ begins to rise, coinciding with a slight rise in $\overline{F}_{\text{elite}}$. This is the onset of Epoch 2, at generation 13, which in this run is about three times longer than the average T_2 quoted in Table 2. At the Epoch 2 takeover—at generation 16—the mean $A_1(6/7)$ makes a sharp jump to become saturated at 1.0 with $\sigma = 0.0$. A similar sharp jump at the Epoch 2 takeover is observed in every run in which Epoch 2 was reached. (In high-ρ_0 specialist runs, the sharp jump is seen in mean $A_0(6/7)$.) What does this sharp jump tell us? Recall that the initial Epoch 2 low-ρ_0 specialists always relax to 0 except on ICs with extremely high density (cf. Figure 5). The GA implements this strategy by finding rules with $A_1(6/7) = 1$. Most neighborhoods in an IC with very high density will thus map to 1, quickly filling the lattice with 1s as the CA is iterated.

Figure 14(b) plots the mean and σ of $A_1(1/2)$ over the elite rules at each generation. Again the mean $A_1(1/2)$ is noisy over most of Epoch 1, though again there is a sharp drop in σ at the Epoch 1 takeover, corresponding to the sudden disappearance of high-ρ_0 specialists. But close to the Epoch 2 onset, $A_1(1/2)$ begins to rise—at the same time as, though less sharply than $A_1(6/7)$—and rises significantly at the takeover as σ falls to near 0. (Again, a similar rise was seen in every run.) This rise is partially due to the saturation of $A_1(6/7)$, which reflects a uniform mapping of $\rho^1(\eta) \geq 6/7$ neighborhoods to 1s. But it is also due to the additional mapping of some $1/2 \leq \rho^1(\eta) < 6/7$ neighborhoods to 1s. This rise coincides with a rise in $\overline{F}_{\text{elite}}$. By mapping many of the $1/2 \leq \rho^1(\eta) < 6/7$ neighborhoods to 1s, the GA is discovering rules that are doing an increasingly good job of implementing the Epoch 2 strategies. That is, it is finding rules that correctly classify an increasing number of high-density ICs, and thus obtain increasingly higher fitness.

At the Epoch 3 takeover, the mean $A_1(1/2)$ begins to fall, and continues to fall appreciably for several generations. (A similar fall was seen in every run.) Note that this is different from the behavior of the mean $A_1(6/7)$, which remains saturated at 1.0. The reason for this decrease is the following. The elite rules at Epoch 3 implement a wholly new strategy—relaxing to all 0s by default but expanding sufficiently large blocks of 1s if they are present in

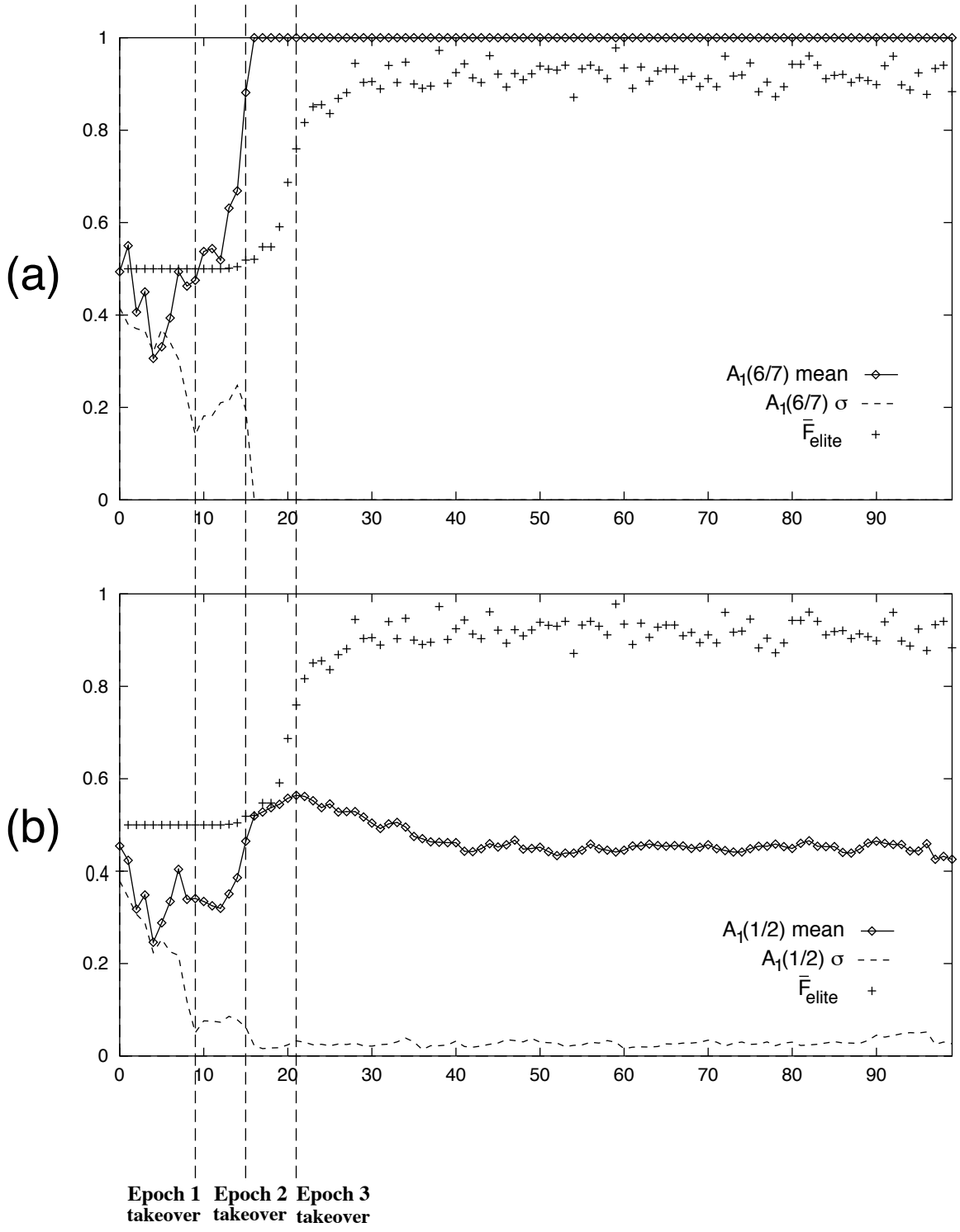

Figure 14: $A_1(d)$ statistics for the same run that was represented in Figure 13. (a) Mean and standard deviation of elite 1-agreement $A_1(6/7)$ versus generation. Mean elite fitness $\overline{F}_{\text{elite}}$ is plotted for reference. (b) Mean and standard deviation of elite 1-agreement $A_1(1/2)$ versus generation. Mean elite fitness $\overline{F}_{\text{elite}}$ is plotted for reference. The takeover generations of of Epochs 1, 2, and 3 are marked by vertical dashed lines.

the IC. Once $A_1(6/7) = 1.0$, implementing the block-expanding strategy only requires setting a few additional neighborhoods to 1s—this does not affect $A_1(1/2)$ appreciably. This block-expanding strategy accomplishes the same thing as the Epoch 2 trick of increasing $A_1(1/2)$, but in a different way. In the later part of Epoch 2, the elite rules correctly classify many high-density ICs because $\rho^1(\eta) \geq 1/2$ neighborhoods map to 1. However, a high-density IC is very likely to contain at least one, if not many, sufficiently large blocks of 1s, and so the Epoch 3 rules do not need this trick of immediately mapping most of the $\rho^1(\eta) \geq 1/2$ neighborhoods to 1—they can rely on expanding blocks to do the job instead. Thus many of the $\rho^1(\eta) \geq 1/2$ neighborhoods are not required have output 1s for high fitness, so under mutation, some of these 1s drift to 0s. The latter also enhances fitness since it reduces the creation of spurious 1-blocks as shown in Figure 8(c).

Let us summarize this section briefly. For runs resulting in low-ρ_0 specialists, Epoch 1 strategies ("relax to all 0s") are implemented by mapping almost all $\rho^0(\eta) \geq 1/2$ neighborhoods to 0. Epoch 2 strategies ("relax to all 0s, unless the IC has very high density, in which case relax to all 1s") are implemented by mapping many of $\rho^1(\eta) \geq 1/2$ neighborhoods to 1s. The more such mappings, the more high-density ICs will be correctly classified. Epoch 3 strategies ("relax to all 0s, unless the IC contains a sufficiently large block of 1s, in which case expand it") are implemented, once all the $\rho^1(\eta) \geq 6/7$ neighborhoods map to 1s, by mapping a small number of specific neighborhoods to 1s.

Which bits in the rule table need to be set in order to expand 1-blocks? This can be determined by direct enumeration. To expand a 1-block in both directions at equal velocities in a sea of 0s, for example, a $\ldots 111000 \ldots$ wall must be propagated to the right and a $\ldots 000111 \ldots$ wall must be propagated to the left. (Note that walls can be more complicated than this, as seen in Figures 5(b), 6(b), and 7(a), for example.) The neighborhoods which participate in this are those patterns of length $2r + 1$ that contain one or both types of wall. The required output bit for each such neighborhood is simply read off the space-time diagram from the cell below the pattern's center at the next time step. From this it can be seen that a bi-directional expansion of 1-blocks of length greater than the neighborhood size requires 14 bits in the chromosome to be properly set. Presumably, these bits or similar constellations that support the observed strategies are set during Epoch 2 and become fixed in Epochs 3 and 4. In light of this, a better statistic for Epoch 3 would be based not on $A_1(6/7)$ but on the appearance of the constellations of output bits supporting walls that expand blocks. Such a statistic would be correlated with more detailed behavior than is the case for $A_s(d)$ (or for the mean field theory statistics proposed by Gutowitz [35]).

In any case, the discovery of a strategy to expand 1-blocks relaxes the constraints on many of the $\rho^1(\eta) \geq 1/2$ neighborhoods that were set to 1 in Epoch 2; many of these drift back to 0, possibly reducing the tendency to create spurious blocks.

This account of how strategies are implemented applies to runs that evolve low-ρ_0 specialists. A similar account applies to runs that evolve high-ρ_0 specialists with the roles of 0 and 1 reversed.

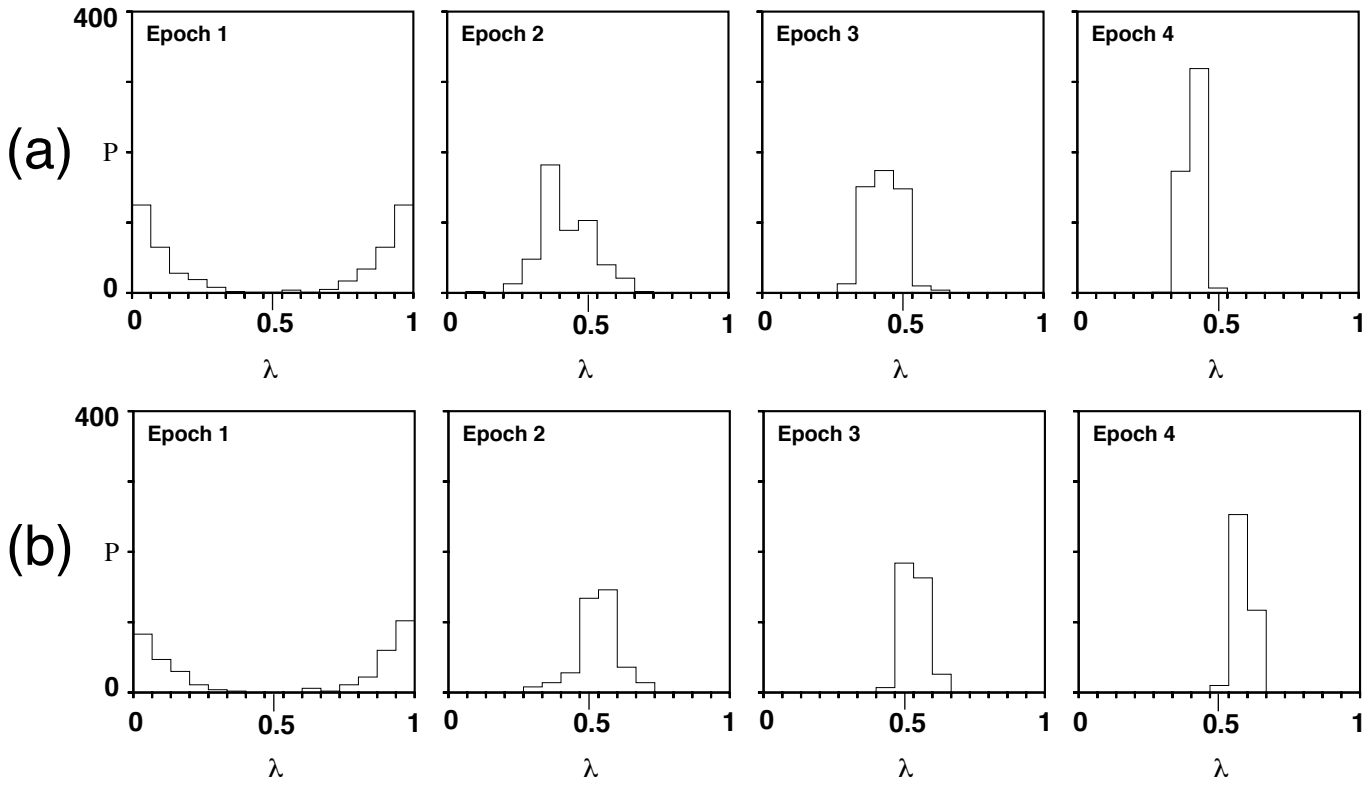

Figure 15: (a) Histograms of elite rule frequency versus λ at each epoch, with rules merged from 25 runs that evolved low-ρ_0 specialists. (b) Histograms of elite rule frequency versus λ at each epoch, with rules merged from 19 runs that evolved high-ρ_0 specialists.

10. Epochs and λ Distributions

We have now described in some detail how strategies at different epochs are implemented in the rule table. This answers the first of our major questions. This understanding helps us to answer the second question: In what way are the macroscopic properties of the λ distributions presented in Figure 11 related to the four epochs? In particular, what causes the symmetry breaking seen at generation 15 in Figure 11?

Figure 15 gives two sets of histograms similar to those in Figure 11. Each histogram in Figure 15(a) represents elite rules merged from 25 runs that evolved low-ρ_0 specialists. The "Epoch 1" histogram plots the elite rules from each run at generation 0. The "Epoch 2" histogram plots the elite rules from each run at the generation of Epoch 2 takeover—defined, as above, as the first generation at which all or almost all the elite rules are implementing Epoch 2 strategies. This generation is different for each run, so the rules represented in this histogram are from different generations on different runs. In short, the runs are lined up with respect to epoch's takeover generation. The "Epoch 3" histogram plots rules at the takeover generation of Epoch 3 in each run, and the "Epoch 4" histogram plots the elite rules at generation 99 in each run. Figure 15(b) gives the same histograms for 19 runs that evolved high-ρ_0 specialists. (These are the same $19 + 25 = 44$ runs for which statistics are

given in the first column of Table 2.)

Both Epoch 1 histograms show most elite rules to be clustered at very low and very high λ values. As was noted above, these are the rules that are selected in the first generation because they are the ones that initially implement the Epoch 1 strategies.

The generation 0 clustering of rules at high and low λs could be considered to be an early "symmetry breaking" that is an artifact of the initial population's λ distribution. However, both Epoch 2 histograms show the elite population clustered much closer to $\lambda = 1/2$—on the low side in 15(a) and on the high side in 15(b). This movement of the elite population towards $\lambda = 1/2$ has two sources. The first is combinatorial drift, which moves Epoch 1 rules closer to $\lambda = 1/2$. The second is the selection of rules implementing Epoch 2 strategies. In runs that evolve low-ρ_0 specialists, most Epoch 1 rules have low λ (e.g., see Figure 13(c)). The innovation at Epoch 2 is to increase the number of $\rho^1(\eta) \geq 1/2$ neighborhoods with output bit 1. These two trends result in an increase in λ. The opposite is true for runs that evolve high-ρ_0 specialists. In both cases, the result is to move closer to $\lambda = 1/2$.

Both Epoch 3 histograms show even narrower distributions, now close to being peaked at $\lambda = 1/2$. As was said above, the Epoch 3 block-expanding strategies require only relatively few bits to be set in the rule table, so the discovery of these strategies does not appreciably change the λ distribution. (Generally, as Epoch 3 is reached the utility of λ declines as an information projective coordinate for monitoring changes in the population structure.) Drift continues to move rules closer to $\lambda = 1/2$, and most rules implementing Epoch 3 strategies have $\lambda \approx 1/2$. But by Epoch 4, the populations have moved back to either side of $\lambda = 1/2$. This Epoch 4 move is what was unexpected, given the argument that rules with good performance should have $\lambda \approx 1/2$. This is what we refer to as "broken symmetry".

What causes it? As was seen in Figure 8, Epoch 3 rules make a number of errors, such as expanding blocks that are too small, or creating and expanding blocks that were not in the IC. There are two ways the GA can correct such errors without destroying the new strategy: (1) by setting bits so as to increase the minimum block size required for expansion, and (2) by ensuring that if there are no sufficiently large blocks present in the IC, that the CA very quickly relaxes to the default fixed-point configuration. For low-ρ_0 specialists, both these corrections require mapping more neighborhoods to 0s. This is a way to ensure that the all-0s fixed point is reached quickly on ICs without sufficiently large blocks. For high-ρ_0 specialists, they require mapping more neighborhoods to 1s. For low-ρ_0 specialists, the corrections decrease λ; for high-ρ_0 specialists, they increase it. This is what seems to cause the broken symmetry seen in the Epoch 4 histograms. In short, The symmetry breaking in Epoch 3 results from improvements in the block-expanding strategies. The result is clearly seen in Epoch 4 where the $\lambda = 1/2$ rules are largely suppressed.

11. GA Mechanisms of Innovation

We have now answered the first two of our major questions. In this section we address the third: What GA mechanisms underlie the epochs of innovation? In particular, we investigate the roles of crossover and mutation in producing the behavior that we have observed.

To better understand the role of crossover, we performed a set of 50 GA runs with the

same parameter values as were described in Section 6, but with crossover turned off. In these runs, the new 80 rules at each generation were created from the 20 elite rules by mutation only—pairs of parents were chosen at random from the elite as before, but no crossover was performed and each offspring was a copy of its parent with exactly two mutations.

Figure 16 displays the best fitness at each generation for two of the runs without crossover. In the run displayed in Figure 16(a), the GA never found a rule with fitness greater than 0.5. This occurred in 37 out of the 50 runs, compared with 4 out of 50 runs when crossover was turned on. These statistics are given in Table 2.

The other 13 runs were similar to Figure 16(b). More detailed examination of these runs showed that the GA made the same progression through strategy epochs as in the runs with crossover, but the onset of Epoch 2 was, on average, much later. However, once Epoch 2 rules were discovered, the GA moved on to Epoch 3 rules very quickly. The first two columns of Table 2 compare these times for the 44 runs with crossover and for the 13 runs with no crossover that reached Epoch 3.

Crossover clearly plays a role in speeding up the onset of Epoch 2. However, its role in the move from Epoch 2 to Epoch 3 is much less pronounced. The analysis we gave above of how Epoch 2 strategies are implemented in the rule tables helps to explain why. Consider, for example, a run that evolves low-ρ_0 specialists. To get to Epoch 2, the GA must discover a low-λ rule with $A_1(6/7) = 1$. The lexicographic ordering of neighborhoods in the rule-table chromosome happens to allow single-point crossover to create such a rule in one time step. This is because, under our encoding, most of the eight $\rho^1(\eta) \geq 6/7$ neighborhoods are at the extreme "right-hand" side of the chromosome. A crossover between an Epoch 1 low-λ rule and an Epoch 1 high-λ rule thus has a fair chance of yielding an Epoch 2 rule. And, since low and high λ rules are in the initial population to begin with, it does not take much time to discover an Epoch 2 rule when crossover is in effect. However, when crossover is turned off, the GA must rely on mutation alone to set the $\rho^1(\eta) \geq 6/7$ neighborhood bits correctly. The waiting time for this is reflected in the "GA, no xover" statistics given in Table 2. In 37 out of 50 runs, the no-crossover waiting time was greater than 99 generations.

Once an Epoch 2 rule is discovered, a small number of mutations can turn it into an Epoch 3 rule. This is seen in the $T_3 - T_2$ statistics given in Table 2. The mean length of Epoch 2 is small for both the crossover and no-crossover runs. Thus, mutation alone suffices to quickly move to Epochs 3 and 4 and to discover the associated strategies. Crossover does not play a large role, though it does appear to shorten the times.

We performed an additional experiment without crossover in which, for each run, the initial population was not uniformly distributed over $\lambda \in [0.0, 1.0]$, but rather each initial rule had $\lambda \approx 1/2$ ("GA, no xover, initpop 1/2"). Our hypothesis was that there would be more rules in the initial population with, say, low λ but high $A_1(6/7)$, and thus the rules in the population would be closer than in the original no-crossover experiment to the conditions necessary for the discovery of Epoch 2 strategies. The results, given in column 3 of Table 2 supported this hypothesis: the number of runs reaching Epoch 3 and the mean values for T_2 and $T_3 - T_2$ were intermediate between those measured in the crossover and no-crossover experiments. In the no-crossover case, the original uniform-λ initial population is responsible for a long transient; it substantially slows down the GA.

We performed a final experiment in which we used a simple Monte Carlo method instead

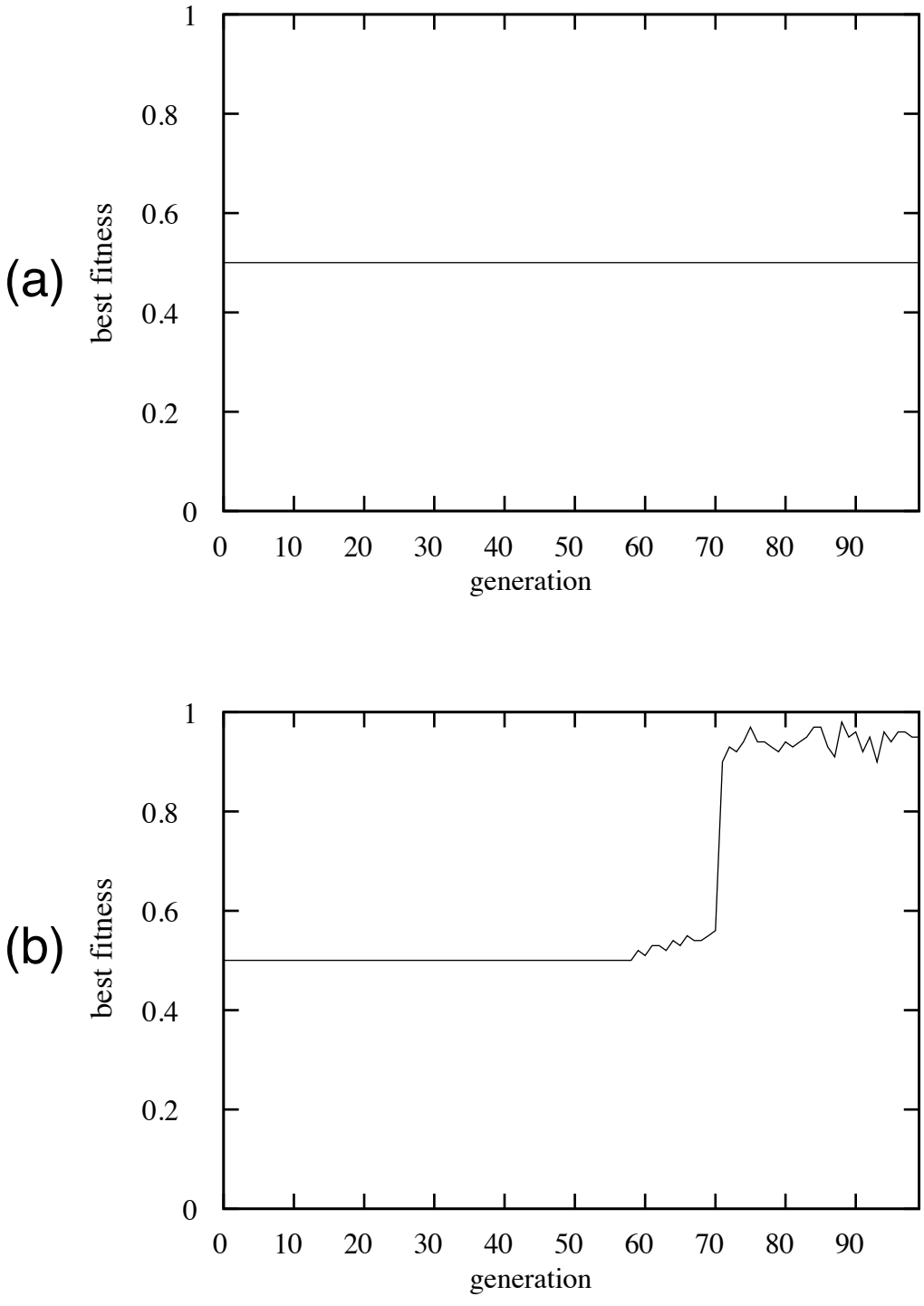

Figure 16: Best fitness versus generation for two of the 50 runs performed with no crossover.

of a GA to search the space of rules. A run of this method is the following. An initial bit string is chosen by some procedure and its fitness is evaluated. A random bit is flipped, and if the new fitness is equal to or higher than the original fitness, the mutation is retained; if not, the original string is retained. This process continues for $10,000$ total evaluations—the same total number of evaluations as performed in one run of the GA (i.e., the population size times the number of generations). We performed 50 such runs, each with a different random-number seed, all starting with an initial string of all 0s (equivalent to one of the fittest strings in the initial GA population). The purpose of this experiment was, again, to test the hypothesis that crossover confers an advantage for reaching Epoch 2. The results of 50 runs of the Monte Carlo method, given in column 4 of Table 2, further support this hypothesis. Under Monte Carlo search Epoch 3 was reached in only 21 out of 50 runs and T_2 (given here in generations, where each generation equals 100 fitness evaluations) is close to that of the "GA, no xover, initpop 1/2" experiment. It is interesting that the average duration of Epoch 2 is smaller with Monte Carlo search than in any of the other experiments.

12. GA Impediments

With this understanding of the GA's behavior on the $\rho_c = 1/2$ task, we now can address the last of our major questions: What impedes the GA from discovering better strategies? In particular, what prevents the GA from discovering the GKL rule or similar rules, except on rare occasions? Here we list a number of impediments that are, or might be, faced by our GA on this problem. We will discuss their relevance to GAs in general and propose ways in which they could be overcome. Perhaps surprisingly, most of the impediments we identify are also forces that help the GA in the initial stages of its search. What is needed is a theory of how the costs and benefits of these various forces trade off. Such a theory would allow for active monitoring of the change from beneficial to harmful effects.

Symmetry breaking

A primary impediment is the GA's tendency to break the task's symmetries by producing low-ρ_0 or high-ρ_0 specialists. A pressure towards symmetry breaking is effectively built into our fitness function, since specializing on one half of the ICs is an easy way to obtain a higher fitness than that of a random rule. This kind of symmetry breaking occurs in generation 0 with the selection of the two types of Epoch 1 rules and, in subsequent generations, the entire elite population naturally drifts into one or the other specialist "camps". The Epoch 2 and Epoch 3 strategies are simply elaborations of these original symmetry-broken Epoch 1 strategies. (Similar types of symmetry breaking occur even when the initial population is peaked around $\lambda = 1/2$ rather than uniformly distributed over λ.) Symmetry breaking thus produces a short-term gain for the GA, but later prevents it from making improvements beyond Epoch 4 strategies, as seen in the long periods of stasis appearing in Figures 3, 11, 13, and 14 over generations 20 to 99. We hypothesize that this propensity to break symmetries for short-term gain is a general feature of GAs and even of natural evolution. This suggests that when one wants to apply a GA to a particular problem, one should first determine all the relevant symmetries in the optimization, and then restrict the GA's search space to candidate solutions with those symmetries. This can be done either by having the fitness

function penalize asymmetric candidate solutions or by building the desired symmetries into the representation. This is akin to the general problem of using domain knowledge to assist the GA's search (e.g., see [19]). We plan to investigate the effect of the latter approach on the GA's performance on the $\rho_c = 1/2$ task. As pointed out in our description of $\mathbf{T}_{1/2}$, the task has a number of symmetries in addition to the ρ_0 symmetry that requires $\lambda = 1/2$ for high performance. One possible problem in imposing symmetries on the GA, though, is that this could make innovation substantially more difficult to achieve. In other words, it could be the case that broken-symmetry solutions can lead to symmetry-respecting ones more quickly than in a symmetry-restricted chromosome space. This is exactly what has not happened in our experiments, however.

Drift

A second possible impediment is the force due to combinatorial drift. As was seen in Figure 12, the intrinsic effects of crossover and mutation, apart from selection, produce a strong drift force moving the population close to $\lambda = 1/2$. This stochastic drift is the force that produces conditions necessary for Epoch 2 strategies to be discovered. For example, it creates low-ρ_0 specialists with higher λs, in some cases creating low-ρ_0 specialists with $A_1(6/7) = 1$. However, later in the run drift also restricts the GA's search to one part of the chromosome space. In the absence of strong selection, it is difficult for the GA to maintain candidate solutions far away from $\lambda = 1/2$. This may present a problem for some GA applications, though not necessarily for the $\rho_c = 1/2$ task. The force due to drift is something GA practitioners should take into account when designing a GA for a particular application, and it may be necessary to design operators to counteract this force.

$\rho_c = 1/2$ fitness landscape

A third possible impediment is the effective fitness landscape of the $\rho_c = 1/2$ task. We have seen that there is a ready path for the GA to take from the easily discovered Epoch 1 rules to Epoch 4 rules—almost every run of the GA follows this path by generation 20 or so. Following this path leads to one or the other of two relatively high-fitness "potential wells"—to make a physical analogy—by a breaking of symmetries. But if the GA could avoid this symmetry-broken potential, is there another readily accessible path that the GA could follow to discover GKL-rule-like behavior?

We performed some preliminary experiments that indicate that such a path could exist. We ran the GA on populations of mutants of the GKL rule and found that many different rules have GKL-like behavior, using signals such as those described in Section 5 to classify IC density. Such rules were found at Hamming distances of up to 30 bits or more from the GKL rule. They had $F_{10^4} \approx 0.96$ and thus indicate an intermediate fitness plateau between that of the Epoch 4 rules with maximum $F_{10^4} \approx 0.945$ and that of the GKL rule with $F_{10^4} \approx 0.972$. Further investigations of the landscape around the GKL rule will be reported in future work.

Finally, a more detailed analysis using F_{10^4} of the generation-99 populations revealed that fairly sophisticated rules had been evolved in two runs (one of them performed subsequent to the set of 50 runs described here). One run evolved a rule with $F_{10^4} \approx 0.945$ (most

Epoch 4 rules had $F_{10^4} < 0.930$). This rule did not expand blocks in a straightforward way; rather, it exhibited signaling mechanisms with some similarity to those of the GKL rule. The most sophisticated rule found by the GA was evolved in a second run. This rule had $F_{10^4} \approx 0.954$, and its spacetime behavior was very similar to that of the GKL rule ($F_{10^4} \approx 0.972$) and the mutants mentioned above (though its Hamming distance from both the GKL rule and a spatially reversed version of the GKL rule was 49 bits and 35 bits respectively). We will not discuss the behavior of these rules or the steps leading to their discovery in detail here, since the statistics that would be required to indicate the emergence of the computational strategies are complicated. These details will be reported elsewhere. But these rules do suggest the existence of two additional fitness plateaus between Epoch 4 and the GKL rule. The emergence of such rules, even if only rarely, demonstrates that the GA, acting on rule tables alone, is in principle capable of discovering rules with sophisticated computational behavior. It is notable that the GA discovered, albeit rarely, rules similar to the GKL rule, since the fitness function used in these simulations in no way specified what particular spacetime behavior is desirable beyond specifying the desired final outcome. The discoveries are also notable in light of the apparent mean-field nature of the density-classification task. In particular, the computational strategies use signalling mechanisms that are not describable in terms of the mean-field theory for CAs [35].

Stochastic nature of F_{100}

A fourth impediment is the stochastic nature of F_{100}. The small sample of ICs used to compute fitness limits the resolution available to the GA for distinguishing among competing rules. This limited resolution obscures differences in fitness that might be significant. This was observed in our experiments with mutations of the GKL rule mentioned above. Even when the initial population consisted of rules that were each one bit different from the GKL rule, the GA did not rediscover and retain the GKL rule because F_{100} could not reliably distinguish the mutated rules with $F_{10^4} \approx 0.96$ from the GKL rule with $F_{10^4} \approx 0.972$. This problem of low resolution could be solved—at considerable computational cost—by using a much enlarged sample of ICs. An intermediate solution would be for the GA to retain and use accumulated fitness information for individuals over many generations; for example, it could keep a running fitness average for rules that survive. This differs from the present method which discards fitness information from previous generations, determining the elite rules only from the fitnesses calculated on the given generation. More experiments need to be performed to determine what level of fitness resolution is needed to obtain improved performance.

Structure of IC sample

A fifth impediment is presented by the structure of the IC sample chosen at each generation. Our current method is to choose a sample uniformly distributed over $\rho_0 \in [0.0, 1.0]$, with exactly half the sample having $\rho_0 < \rho_c$ and exactly half having $\rho_0 > \rho_c$. This distribution was meant to present some "easy-to-classify" extreme-ρ_0 ICs to the evolving rules in order to allow evolution to get off the ground. However, aside from the above-mentioned pressure towards very early symmetry breaking arising from this distribution, there is another

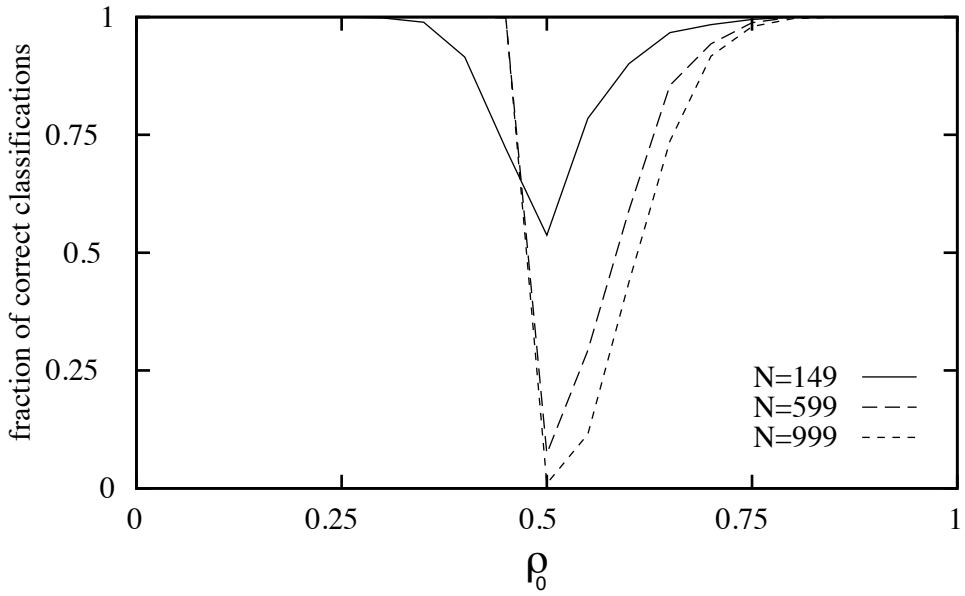

Figure 17: Performance of an Epoch 4 rule (the same one whose performance was plotted in Figure 9) plotted as a function of ρ_0. Performance plots are given for three lattice sizes: 149, 599, and 999. This rule has $\lambda \approx 0.59$.

impediment. After a small number of generations, this IC distribution does not present a sufficient challenge to the evolving rules. For example, all Epoch 2 rules correctly classify half the distribution in addition to the extreme-ρ_0 cases of the other half. This means that the fitness differences among Epoch 2 rules are being judged on the basis of the remaining ICs—less than half of the original 100—which exacerbates the fitness-resolution problems discussed above. That is, the reduction in the fraction of informative test ICs reduces the number of useful fitness evaluations and so increases the variance in the mean fitness. Epoch 4 rules, for example, routinely achieve 100% correct classification on some set of ICs during a run of the GA, whereas under F_{10^4}, they never reach fitnesses above ≈ 0.95. One possible solution to this problem is to co-evolve a population of IC samples with the population of CA rules, with the fitness of an IC sample being inversely related to the classification performance of the current CA population on this sample. In principle, such co-evolution—analogous to biological "arms races" seen in nature—should produce sets of ICs that are tuned expressly to present challenges to rules in the current population. In this situation, the absolute meaning of the fitness function F_{100} would change over the generations. This approach should help alleviate the problem of fitness accuracy without requiring computationally intractable sample sizes. Such a co-evolutionary approach has been studied in the context of using GAs to discover efficient sorting networks [40]. Another alternative would be to use modern statistical evaluation methods that make more efficient use of the available fitness evaluations.

Fixed lattice size

A sixth impediment, due to our particular method, is the restriction of fitness evaluation to a fixed lattice size—here, $N = 149$. As was shown in Figure 2, the GKL rule's classification performance improves slightly as lattice size increases. The opposite is true of the fittest evolved rules in our experiments. The performance of one Epoch 4 rule as a function of ρ_0 is plotted in Figure 17 for lattice sizes of $N = 149$, 599, and 999. (This is the same rule whose performance was plotted in Figure 9.) This rule has $\lambda \approx 0.59$; it increases sufficiently large blocks of adjacent or nearly adjacent 0s. We used the same procedure to make these plots as was described for Figure 2. As can be seen, the performance according to this measure is not only significantly worse than that of the GKL rule on $N = 149$ lattices, but also decreases dramatically for larger N. The worst performances for $N = 599$ and $N = 999$ are centered slightly above $\rho_0 = \rho_c$. (Since we used only odd N, the actual ρ_0s plotted at 0.5 are slightly above 0.5.) With $\rho_0 > \rho_c$, the CA should relax to a fixed point of all 1s. Detailed inspection, however, revealed that on almost every IC with ρ_0 slightly above ρ_c, the CA is relaxing to a fixed point of all 0s. This is a result of this rule's strategy of expanding "sufficiently large" blocks of 0s. The appropriate block size b to expand was evolved to be a good predictor of ρ_0 for $N = 149$. With larger lattices the probability of b-length 0-blocks in ICs with $\rho_0 > \rho_c$ increases. And so the closer high ρ_0s are to ρ_c, the more likely such blocks are to occur. In the CA we tested with $N = 599$ and $N = 999$, such blocks occurred in most ICs with ρ_0 slightly above ρ_c, always leading to incorrect classifications. This shows that keeping the lattice size fixed during GA evolution can lead to overfitting for the particular lattice size. We plan to experiment with varying the lattice size during evolution in an attempt to prevent such overfitting. We predict that the block-expanding strategy will not arise on runs performed with significantly larger lattices; for small-radius CA the strategy of expanding blocks can work well only on relatively small lattices since the maximum-length block that can be expanded by a CA is a function of (though not strictly equal to) its neighborhood size.

Representation

A seventh impediment is the lexicographically ordered bit-string representation used for the CA rules. This ordering has the initially beneficial effect of grouping together most of the $\rho^s(\eta) \geq 6/7$ neighborhoods, with $s = 0$ neighborhoods at the "left" extreme and $s = 1$ neighborhoods at the "right" extreme. As pointed out above, this ordering enables crossover to quickly produce Epoch 2 rules—low-ρ_0 specialists with $A_1(6/7) = 1$ or high-ρ_0 specialists with $A_0(6/7) = 0$. However, the lexicographic ordering of output bits may hinder the GA's progress in later generations, since to produce the kinds of coordinated signals used in the GKL rule, a number of neighborhood output bits must work in concert. These co-active neighborhoods are unlikely to be adjacent in a lexicographic ordering, and thus cannot be moved together from a parent to an offspring via simple crossover. More disruptive crossover operators such as uniform crossover [84] run the risk of destroying the necessary structures. The problem of designing a representation that will work well with genetic operators is a general one for GAs. One solution that has been explored is adapting the representation to suit the operators (e.g., see [32]). Phenomena in natural genetics such as inversion and jumping genes may be a form of representation adaptation. These have inspired some work

in GAs along these lines (e.g., see [31, 42]).

Loss of diversity

Finally, an eighth impediment is the loss of diversity over time in the population. When a new strategy is discovered, it sweeps through the population and quickly all the elite rules are representatives of that strategy. This convergence aids the rapid moves from Epoch 2 to Epoch 3 to Epoch 4. However, convergence, like drift, limits the region of chromosome space that the GA is searching. Controlling convergence in GAs has been the subject of much research. (See [31] for a review of work in this area.) In our experiments with $E = 20$, diversity (measured as the mean pairwise Hamming distance in the elite) falls quickly—more rapidly, in fact, than it did in previous experiments with $E = 50$. However, the smaller E also sped up the onsets of the different epochs, since the newly discovered strategies were able to invade the population more quickly. We also performed experiments in which a minimum level of diversity was explicitly maintained. This scheme did not yield improved performance [61]. But in spite of these results, it may be the case that the rapid decrease in diversity is an impediment for moving beyond Epoch 4 strategies. The need to balance the level of population diversity with the need to quickly propagate newly discovered innovations to the rest of the population is discussed in several places in GA literature (e.g., see [60]).

As was noted above, most of these impediments are also forces that help the GA in the initial stages of its search. None are specific to the $\rho_c = 1/2$ task or even to the problem of evolving CAs. Rather, they are general issues in any GA application, and some of them are relevant to any machine-learning method. In this work our analysis tools have enabled us to observe some of these forces (e.g., symmetry breaking) quite clearly, and to study them carefully. Going beyond this to develop a predictive theory of the tradeoffs these forces produce in GA efficiency is one of our long-term objectives.

13. Conclusion

As was said in the introduction, the goals of our research are (i) to better understand the ways in which CAs can perform computations; (ii) to learn how to best use GAs to evolve computationally useful CAs and (iii) to understand the mechanisms by which GAs can produce complex and innovative behavior in systems with simple components and local interactions.

This paper has reported progress on these goals obtained by analyzing in detail a GA's behavior on evolving CAs to perform a particular computation: $\rho_c = 1/2$ density classification. We analyzed the strategy of the GKL rule for performing this task, and used it as a benchmark with which to compare the rules evolved by the GA. We have described the epochs of innovation in most runs of the GA and the strategies corresponding to these epochs, and have understood in detail how these strategies are implemented and how these epochs manifest themselves in large-scale population structures. We then explained the respective roles of crossover and mutation in the discovery of new strategies and identified several impediments for the GA in achieving higher computational capability in CAs. Primary among the impediments is the GA's breaking of task symmetries in the pursuit of short-term gains

in fitness. We believe that this type of detailed analysis is essential in order to understand and improve the GA's behavior and to develop predictive theories of the tradeoffs among different evolutionary forces. The results are relevant to the application of GAs in general, and they point the way to a more general analysis of the evolutionary forces we have identified. This work is also a first step in developing methods for automatic programming of CAs and other spatially-distributed parallel computers. Success in this area should have significance for the field of parallel computation and for nonlinear spatial modeling.

Acknowledgments

We thank Jonathan Amsterdam, Rajarshi Das, Jim Hanson, and Dan Upper for helpful comments on an earlier draft of this paper. We also thank Rajarshi Das for assistance on running the GA and for pointing out the discovery of the most sophisticated rule found by the GA. This research was supported by the Santa Fe Institute, under the Adaptive Computation, Core Research, and External Faculty Programs, and by the University of California, Berkeley, under contract AFOSR 91-0293.

References

[1] D. H. Ackley and M. L. Littman. Interactions between learning and evolution. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 487–507, Reading, MA, 1992. Addison-Wesley.

[2] D. H. Ackley and M. L. Littman. A case for Lamarckian evolution. In C. G. Langton, editor, *Artificial Life III*, Reading, MA, 1993. Addison-Wesley.

[3] J. Andreoni and J. H. Miller. Auctions with adaptive artificial agents. Technical Report 91-01-004, Santa Fe Institute, Santa Fe, New Mexico 87501, 1991.

[4] J. C. Bean. Genetics and random keys for sequencing and optimization. Technical Report 92-43, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, 1992.

[5] M. A. Bedau and N. H. Packard. Measurement of evolutionary activity, teleology, and life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 431–461, Reading, MA, 1992. Addison-Wesley.

[6] M. A. Bedau, F. Ronneburg, and M. Zwick. Dynamics of diversity in an evolving population. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 95–104, Amsterdam, 1992. North Holland.

[7] R. K. Belew. Evolution, learning, and culture: Computational metaphors for adaptive algorithms. *Complex Systems*, 4:11–49, 1990.

[8] R. K. Belew, J. McInerney, and N. N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, pages 511–547, Reading, MA, 1992. Addison-Wesley.

[9] A. Bergman and M. W. Feldman. Recombination dynamics and the fitness landscape. *Physica D*, 56:57–67, 1992.

[10] M. F. Bramlette and E. E. Bouchard. Genetic algorithms in parametric design of aircraft. In L. D. Davis, editor, *Handbook of Genetic Algorithms*, pages 109–123. Van Nostrand Reinhold, 1991.

[11] J. Campbell, B. Ermentrout, and G. Oster. A model for mollusk shell patterns based on neural activity. *The Veliger*, 28:369, 1986.

[12] D. J. Chalmers. The evolution of learning: An experiment in genetic connectionism. In D. S. Touretzky et al., editor, *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, CA, 1990. Morgan Kaufmann.

[13] R. J. Collins and D. R. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256, San Mateo, CA, 1991. Morgan Kaufmann.

[14] R. J. Collins and D. R. Jefferson. The evolution of sexual selection and female choice. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 327–336, Cambridge, MA, 1992. MIT Press/Bradford Books.

[15] M. Creutz. Deterministic Ising dynamics. *Ann. Phys.*, 167:62, 1986.

[16] J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, 69:279–301, 1993.

[17] T. Dandekar and P. Argos. Potential of genetic algorithms in protein folding and protein engineering simulations. *Protein Engineering*, 5(7):637–645, 1992.

[18] Y. Davidor. *Genetic algorithms and robotics*. Robotics and Automated Systems. World Scientific, Singapore, 1991.

[19] L. D. Davis, editor. *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[20] P. Gonzaga de Sá and C. Maes. The Gacs-Kurdyumov-Levin automaton revisited. *Journal of Statistical Physics*, 67(3/4):507–522, 1992.

[21] D. d'Humières, P. Lallemand, J. P. Boon, D. Dab, and A. Noullez. Fluid dynamics with lattice gases. In R. Livi, S. Ruffo, S. Ciliberto, and M. Buiatti, editors, *Workshop on Chaos and Complexity*, pages 278–301, Singapore, 1988. World Scientific.

[22] M. Dorigo and E. Sirtori. Alecsys: A parallel laboratory for learning classifier systems. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 296–302, San Mateo, CA, 1991. Morgan Kaufmann.

[23] D. Farmer, T. Toffoli, and S. Wolfram, editors. *Cellular Automata: Proceedings of an Interdisciplinary Workshop*. North Holland, Amsterdam, 1984.

[24] D. B. Fogel and J. W. Atmar. Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear search. *Biological Cybernetics*, 63:111–114, 1990.

[25] J. F. Fontanari and R. Meir. The effect of learning on the evolution of asexual populations. *Complex Systems*, 4:401–414, 1990.

[26] S. Forrest, B. Javornik, R. Smith, and A. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, In press.

[27] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Physical Review Letters*, 56:1505, 1986.

[28] P. Gacs. Nonergodic one-dimensional media and reliable computation. *Contemporary Mathematics*, 41:125, 1985.

[29] P. Gacs, G. L. Kurdyumov, and L. A. Levin. One-dimensional uniform arrays that wash out finite islands. *Probl. Peredachi. Inform.*, 14:92–98, 1978.

[30] H. Gerola and P. Seiden. Stochastic star formation and spiral structure of galaxies. *Astrophysical Journal*, 223:129, 1978.

[31] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[32] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1990.

[33] F. Gruau. Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In L. D. Whitley and J. D. Schaffer, editors, *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 55–72, Los Alamitos, CA, 1992. IEEE Computer Society Press.

[34] H. A. Gutowitz, editor. *Cellular Automata*. MIT Press, Cambridge, MA, 1990.

[35] H. A. Gutowitz. A hierarchical classification of cellular automata. *Physica D*, 45:136–156, 1990.

[36] A. B. Hadj-Alouane and J. C. Bean. A genetic algorithm for the multiple-choice integer program. Technical Report 92-50, Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, 1992.

[37] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of Statistical Physics*, 66(5/6):1415–1462, 1992.

[38] S. A. Harp and T. Samad. Genetic synthesis of neural network architecture. In L. D. Davis, editor, *Handbook of Genetic Algorithms*, pages 202–221. Van Nostrand Reinhold, 1991.

[39] I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From Animals to Animats 2: Proceedings of the second international conference on simulation of adaptive behavior*, pages 364–373, Cambridge, MA, 1993. MIT Press.

[40] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[41] G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[42] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992. Second edition (First edition, 1975).

[43] J. H. Holland. Echoing emergence: Objectives, rough definitions, and speculations for Echo-class models. In *Complexity: Metaphors, Models, and Reality*, Reading, MA, 1994. Addison-Wesley.

[44] J. H. Holland and J. H. Miller. Artificial adaptive agents in economic theory. Technical Report 91-05-025, Santa Fe Institute, Santa Fe, New Mexico, 1991.

[45] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a theme in artificial life: The Genesys/Tracker system. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 549–577, Reading, MA, 1992. Addison-Wesley.

[46] J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1993.

[47] C. G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.

[48] W. Li. Non-local cellular automata. In L. Nadel and D. Stein, editors, *1991 Lectures in Complex Systems*, pages 317–327. Addison-Wesley, Redwood City, CA, 1992.

[49] W. Li, N. H. Packard, and C. G. Langton. Transition phenomena in cellular automata rule space. *Physica D*, 45:77–94, 1990.

[50] K. Lindgren. Evolutionary phenomena in simple dynamics. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 295–312, Reading, MA, 1992. Addison-Wesley.

[51] K. Lindgren and M. G. Nordhal. Artificial food webs. In C. G. Langton, editor, *Artificial Life III*, Reading, MA, 1993. Addison-Wesley.

[52] A. Mackay. Crystal symmetry. *Phys. Bull.*, 27:495, 1976.

[53] B. Madore and W. Freedman. Computer simulations of the Belousov-Zhabotinsky reaction. *Science*, 222:615, 1983.

[54] F. Menczer and D. Parisi. A model for the emergence of sex in evolving networks: Adaptive advantage or random drift? In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Cambridge, MA, 1992. MIT Press/Bradford Books.

[55] M. Meriaux. A cellular architecture for image synthesis. *Microprocess. and Microprogram*, 13:179, 1984.

[56] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence Series. Springer-Verlag, 1992.

[57] G. F. Miller and P. M. Todd. Exploring adaptive agency I: Theory and methods for simulating the evolution of learning. In D. S. Touretzky et al., editor, *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, CA, 1990. Morgan Kaufmann.

[58] G. F. Miller, P. M. Todd, and S. U. Hegde. Designing neural networks using genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384. Morgan Kaufmann, San Mateo, CA, 1989.

[59] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Dynamics, computation, and the "edge of chaos": A re-examination. In *Complexity: Metaphors, Models, and Reality*, pages 497–513, Reading, MA, 1994. Addison-Wesley.

[60] M. Mitchell, J. H. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing? To appear in J. D. Cowan, G. Tesauro, and J. Alspector (editors), *Advances in Neural Information Processing Systems 6*. San Mateo, CA: Morgan Kaufmann.

[61] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

[62] D. J. Montana and L. D. Davis. Training feedforward networks using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, San Mateo, CA, 1989. Morgan Kaufmann.

[63] S. Nolfi, J. L. Elman, and D. Parisi. Learning and evolution in neural networks. Technical Report CRL 9019, Center for Research in Language, University of California, San Diego, 1990.

[64] Y. Oono and M. Kohmoto. A discrete model of chemical turbulence. *Phys. Rev. Lett.*, 55:2927, 1985.

[65] N. H. Packard. Lattice models for solidification and aggregation. In Y. Katoh, R. Takaki, J. Toriwaki, and S. Ishizaka, editors, *Proceedings of the First International Symposium for Science on Form*. KTK Scientific Publishers, 1986.

[66] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301, Singapore, 1988. World Scientific.

[67] N. H. Packard. Intrinsic adaptation in a simple model for evolution. In C. G. Langton, editor, *Artificial Life*, pages 141–155, Reading, MA, 1989. Addison-Wesley.

[68] D. Parisi, S. Nolfi, and F. Cecconi. Learning, behavior, and evolution. In *Proceedings of the First European Conference on Artificial Life*, Cambridge, MA, 1992. MIT Press/Bradford Books.

[69] R. Parsons, S. Forrest, and C. Burks. Genetic algorithms for DNA sequence assembly. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proceedings of the first international conference on intelligent systems for molecular biology*, pages 310–318, Menlo Park, CA, 1993. AAAI Press.

[70] J. Pecht. On the real-time recognition of formal languages in cellular automata. *Acta Cybernetica*, 6:33, 1983.

[71] D. J. Powell, M. M. Skolnick, and S. S. Tong. Interdigitation: A hybrid technique for engineering design optimization employing genetic algorithms, expert systems, and numerical optimization. In L. D. Davis, editor, *Handbook of Genetic Algorithms*, pages 312–331. Van Nostrand Reinhold, 1991.

[72] K. Preston. Basics of cellular logic with some applications in medical image processing. *Proc. IEEE*, 67:826, 1979.

[73] K. Preston and M. Duff. *Modern Cellular Automata*. Plenum, New York, 1984.

[74] T. S. Ray. Is it alive, or is it GA? In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 527–534, San Mateo, CA, 1991. Morgan Kaufmann.

[75] T. S. Ray. An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 371–408, Reading, MA, 1992. Addison-Wesley.

[76] A. Rosenfeld. Parallel image processing using cellular arrays. *Computer*, 16:14, 1983.

[77] J. D. Scargle, D. L. Donoho, J. P. Crutchfield, T. Steiman-Cameron, J. Imamura, and K. Young. The quasi-periodic oscillations and very low frequency noise of Scorpius X-1 as transient chaos: A dripping handrail? *Astrophy. J. Lett.*, 411:L91 – L94, 1993.

[78] J. D. Schaffer and L. J. Eshelman. On crossover as an evolutionarily viable strategy. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68, San Mateo, CA, 1991. Morgan Kaufmann.

[79] S. Schulze-Kremer. Genetic algorithms for protein tertiary structure prediction. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 391–400, Amsterdam, 1992. North Holland.

[80] K. Shahookar and P. Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Transactions on Computer-Aided Design*, 9(5):500–511, 1990.

[81] A. R. Smith. Real-time language recognition by one-dimensional cellular automata. *J. Comput. System Sci.*, 6:233, 1972.

[82] S. A. Smith, R. C. Watt, and S. R. Hameroff. Cellular automata in cytoskeletal lattices. *Physica D*, 10:168–174, 1984.

[83] S. Sternberg. Language and architecture for parallel image processing. In *Proceedings of the Conference in Pattern Recognition in Practice*, Amsterdam, 1980. North-Holland.

[84] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1989. Morgan Kaufmann.

[85] C. E. Taylor, D. R. Jefferson, S. R. Turner, and S. R. Goldman. RAM: Artificial life for the exploration of complex biological systems. In C. G. Langton, editor, *Artificial Life*, pages 275–295, Reading, MA, 1989. Addison-Wesley.

[86] P. M. Todd and G. F. Miller. Exploring adaptive agency III: Simulating the evolution of habituation and sensitization. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, Berlin, 1990. Springer-Verlag (Lecture Notes in Computer Science).

[87] P. M. Todd and G. F. Miller. Exploring adaptive agency II: Simulating the evolution of associative learning. In J.-A. Meyer and S. W. Wilson, editors, *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, pages 306–315, Cambridge, MA, 1991. MIT Press.

[88] T. Toffoli and N. Margolus. *Cellular Automata Machines: A new environment for modeling*. MIT Press, Cambridge, MA, 1987.

[89] R. Unger and J. Moult. A genetic algorithm for 3d protein folding simulations. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 581–588, San Mateo, CA, 1993. Morgan Kaufmann.

[90] G. Y. Vichniac. Simulating physics with cellular automata. *Physica D*, 10:96–116, 1984.

[91] S. Wolfram, editor. *Theory and applications of cellular automata*. World Scientific, Singapore, 1986.

[92] D. A. Young. A local activator-inhibitor model of vertebrate skin patterns. *Math. Biosciences*, 72:51, 1984.

[93] P. Zamperoni. Some reversible image operators from the point of view of cellular automata. *Biological Cybernetics*, 54:253–261, 1986.