Complex Systems **7** (1993) 89–130

Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations

Melanie Mitchell*
Peter T. Hraber*
Santa Fe Institute,
1660 Old Pecos Trail, Suite A,
Santa Fe, NM 87501, USA

James P. Crutchfield†
Physics Department, University of California,
Berkeley, CA 94720, USA

Abstract. We present results from an experiment similar to one performed by Packard [24], in which a genetic algorithm is used to evolve cellular automata (CAs) to perform a particular computational task. Packard examined the frequency of evolved CA rules as a function of Langton's λ parameter [17]; he interpreted the results of his experiment as giving evidence for two hypotheses: (1) CA rules that are able to perform complex computations are most likely to be found near "critical" λ values (which have been claimed to correlate with a phase transition between ordered and chaotic behavioral regimes for CAs); (2) When CA rules are evolved to perform a complex computation, evolution will tend to select rules with λ values close to the critical values. Our experiment produced quite different results, and we suggest that the interpretation of the original results is not correct. We also review and discuss issues related to λ, dynamical-behavior classes, and computation in CAs. The primary constructive results of our study are the identification of the emergence and competition of computational strategies, and the analysis of the central role of symmetries in an evolutionary system. In particular, we demonstrate how symmetry breaking can impede evolution toward higher computational capability.

*Email: mm@santafe.edu, pth@santafe.edu
†Email: chaos@gojira.berkeley.edu

1. Introduction

The notion of "computation at the edge of chaos" has attracted considerable attention in the study of complex systems and artificial life (see, for example, [4, 5, 15, 17, 24, 31]). This notion is related to the broad question of the relationship between a computational system's ability for complex information processing and other measures of the system's behavior. In particular, does the ability to perform nontrivial computation require that a system's dynamical behavior be "near a transition to chaos"? Likewise, much attention has been given to the notion of "the edge of chaos" in the context of evolution. In particular, it has been hypothesized that when biological systems must perform complex computation to survive, the process of evolution under natural selection tends to select such systems near a phase transition from ordered to chaotic behavior [14, 15, 24].

In this paper, we reexamine one study that addressed these questions in the context of cellular automata [24]. The results of the original study were interpreted as evidence that an evolutionary process in which cellular-automata rules had been selected to perform a nontrivial computation preferentially selected rules near transitions to chaos. We show that this conclusion is neither supported by our experimental results nor consistent with basic mathematical properties of the computation being evolved. We also review and clarify, in the context of cellular automata, notions relating to such terms as "computation," "dynamical behavior," and "edge of chaos."

2. Cellular automata and dynamics

Cellular automata (CAs) are discrete, spatially extended dynamical systems that have been studied extensively as models of physical processes and as computational devices [7, 11, 26, 30, 32]. In their simplest form, CAs consist of spatial lattices of *cells*, each of which, at time t, can be in one of k states. We denote the lattice size or number of cells as N. A CA has a single fixed rule, which is used to update each cell; the rule maps from the states in a neighborhood of a cell—for example, the states of a cell and its nearest neighbors—to a single state, which is the update value for that cell. The lattice begins with an initial configuration of local 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 value of a single cell—for example, 0 or 1—and "configuration" to mean the pattern of states over the entire lattice.

The CAs that we discuss in this paper are all one-dimensional, with two possible states per cell (0 and 1). In a one-dimensional CA, the neighborhood of a cell includes the cell itself and some *radius* r of neighbors on either side of the cell. All of the simulations will be of CAs with spatially periodic boundary conditions (in other words, the one-dimensional lattice is viewed as a circle, with the right neighbor of the rightmost cell being the leftmost cell, and vice versa).

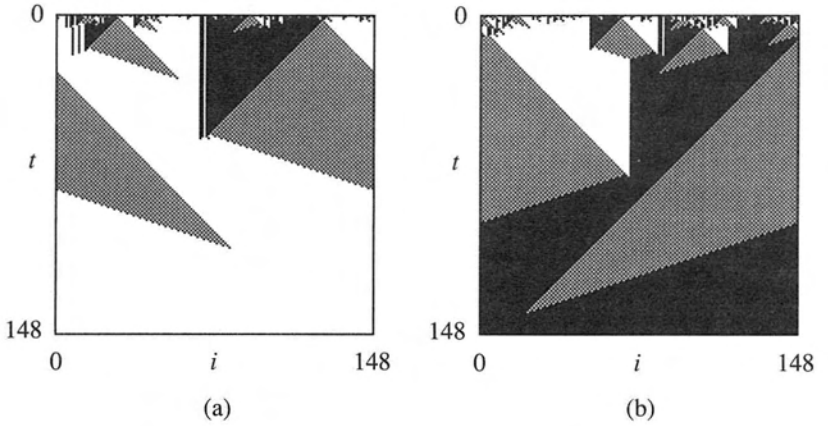

Figure 1: Two space-time diagrams for the binary-state Gacs-Kurdyumov-Levin CA. $N = 149$ sites are shown evolving over 149 time steps, with time increasing down the page, from each of two different initial configurations. In (a), the initial configuration has a density of 1s of approximately 0.48; in (b), a density of approximately 0.52. 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 initial configurations according to their density.

The equations of motion for a CA are often expressed in the form of a *rule table*: a lookup table listing each of the neighborhood patterns, and the state to which the central cell in that neighborhood is mapped. For example, the following displays one possible rule table for a one-dimensional, two-state CA with radius $r = 1$. 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. To run the CA, this lookup table is applied to each neighborhood in the current lattice configuration, respecting the choice of boundary conditions, to produce the configuration at the next time step.

A method commonly used to examine the behavior of a two-state, one-dimensional CA is the display of its space-time diagram (a two-dimensional picture that vertically strings together the one-dimensional CA lattice configurations at the successive time steps, with white squares corresponding to cells in state 0, and black squares corresponding to cells in state 1). Two such space-time diagrams, reproduced in Figure 1, show the actions of the Gacs-Kurdyumov-Levin (GKL) binary-state CA on two random initial configurations of different densities of 1s [6, 8]. In both cases, the CA relaxes to

a fixed pattern over time—in one case, all 0s, and in the other case, all 1s. These patterns are, in fact, fixed points of the GKL CA. That is, once they are reached, further applications of the CA do not change the pattern. We will discuss the GKL CA in further detail below.

CAs are of interest as models of physical processes because, like many physical systems, they consist of a large number of simple components (cells) that are modified only by local interactions, but which, acting together, can produce global complex behavior. Like the class of dissipative dynamical systems, the class of one-dimensional CAs exhibit the full spectrum of dynamical behavior: from fixed points, as seen in Figure 1, to limit cycles (periodic behavior), to unpredictable ("chaotic") behavior. Wolfram considered a coarse classification of CA behavior in terms of these categories; he proposed the following four classes with the intention of capturing all possible CA behavior [31].

> *Class 1:* Almost all initial configurations relax after a transient period to the same fixed configuration (for example, all 1s).
>
> *Class 2:* Almost all initial configurations relax after a transient period, either to some fixed point or to some temporally periodic cycle of configurations, depending on the initial configuration.
>
> *Class 3:* Almost all initial configurations relax after a transient period to chaotic behavior. (The term "chaotic" refers, in this paper, to apparently unpredictable space-time behavior.)
>
> *Class 4:* Some initial configurations result in complex localized structures, sometimes long-lived.

Wolfram does not state the requirements for membership in Class 4 any more precisely than this. Thus, unlike the categories derived from dynamical systems theory, Class 4 is not rigorously defined.

It should be pointed out that, on finite lattices, there is only a finite number (2^N) of possible configurations, so all rules ultimately lead to periodic behavior. Class 2 refers not to this type of periodic behavior, but to cycles with periods much shorter than 2^N.

3. Cellular automata and computation

CAs are of interest also as computational devices, both as theoretical tools and as practical, highly efficient parallel machines [26, 27, 30, 32].

"Computation" has several possible meanings in the context of CAs. The most common meaning is that a CA does some "useful" computational task. In that case, the rule is interpreted as the "program," the initial configuration is interpreted as the "input," and the CA runs for some specified number of time steps or until it reaches some "goal" pattern—possibly a fixed point pattern. The final pattern is interpreted as the "output." An example of this meaning is the use of CAs to perform image-processing tasks [27].

A second meaning of computation by CAs is that a CA, given particular initial configurations, is capable of universal computation. That is, given the right initial configuration, the CA can simulate a programmable computer—complete with logical gates, timing devices, and so on. Conway's Game of Life [1] is such a CA; one construction for universal computation in the Game of Life is given in [1]. Similar constructions have been made for one-dimensional CAs [21]. Wolfram speculated that all Class 4 rules have the capacity for universal computation [31]; however, given the informality of the definition of Class 4 (not to mention the difficulty of proving that a given rule is, or is not, capable of universal computation), this hypothesis is impossible to verify.

A third meaning of computation by CAs involves the behavior of a given CA on an ensemble of initial configurations, interpreted as a kind of "intrinsic" computation. Such computation is not interpreted as the performance of "useful" transformations of input to produce output; rather, it is measured in terms of generic, structural computational elements such as memory, information production, information transfer, logical operations, and so on. It is important to emphasize that the measurement of such intrinsic computational elements does not rely on a semantics of utility (as do the preceding computation types). That is, these elements can be detected and quantified without reference to any specific "useful" computation performed by the CA—such as enhancing edges in an image or computing the digits of π. This notion of intrinsic computation is central to the work of Crutchfield, Hanson, and Young [4, 12].

In general, CAs have the capacity for all kinds of both dynamical and computational behaviors. For this reason—in addition to the computational ease of simulating them—CAs have been considered a good class of models for use in the study of how dynamical behavior and computational ability are related. Similar questions have been addressed in the context of other dynamical systems, including continuous-state dynamical systems (such as iterated maps and differential equations) [4, 5], Boolean networks [14], and recurrent neural networks [25]. We confine our discussion to CAs.

With this background in mind, the broad questions presented in Section 1 can now be rephrased in the context of CAs, as follows.

- What properties must a CA possess to perform nontrivial computation?
- In particular, does a capacity for nontrivial computation (in any of the three senses previously described) require that a CA be in a region of rule space near a transition from ordered to chaotic behavior?
- When CA rules are evolved to perform nontrivial computation, will evolution tend to select rules near such a transition to chaos?

4. Structure of CA rule space

A number of studies conducted during the last decade have addressed our first question. We focus on Langton's empirical investigations of the second

question in terms of the structure of the space of CA rules [17]. The relationship of the first two questions to the third—evolving CAs—is described subsequently.

One of the primary difficulties in understanding the structure of the space of CA rules (and its relation to computational capability) is its discrete nature. In contrast to the well-developed theory of bifurcations for continuous-state dynamical systems [10], there appears to be little or no geometry in CA space, and no notion of smoothly changing a CA to get another that is "nearby in behavior." In an attempt to emulate such a change, however, Langton defined a parameter, λ, that varies incrementally as single output bits are turned on or off in a given rule table. For a given CA rule table, λ is computed as follows. For a k-state CA, one state, q, is arbitrarily chosen to be "quiescent." (In [17], all states obeyed a "strong quiescence" requirement: for any state $s \in \{0, \ldots, k-1\}$, the neighborhood consisting entirely of state s must map to s.) The λ of a given CA rule is the fraction of nonquiescent output states in the rule table. For a binary-state CA, if 0 is chosen to be the quiescent state, then λ is simply the fraction of output 1 bits in the rule table. Typically, there are many CA rules with a given λ value. For a binary CA, the number is strongly peaked at $\lambda = 1/2$, due to the combinatorial dependence on the radius r and the number of states k. It is also symmetric about $\lambda = 1/2$, due to the symmetry of exchanging 0s and 1s. Generally, as λ is increased from 0 to $1-(1/k)$, the associated CAs shift from those having the most homogeneous rule tables to those having the most heterogeneous.

Langton performed a range of Monte Carlo samples of two-dimensional CAs, in an attempt to characterize their average behavior as a function of λ [17]. The notion of "average behavior" was intended to capture the most likely behavior observed with a randomly chosen initial configuration for CAs randomly selected in a fixed-λ subspace. His observation was that the average behavior of rules passed through the following regimes, as λ was incremented from 0 to $1-(1/k)$:

$$\text{fixed point} \Rightarrow \text{periodic} \Rightarrow \text{"complex"} \Rightarrow \text{chaotic.}$$

That is, the average behavior at low λ was for a rule to relax to a fixed point after a relatively short transient phase (see Figure 16 in [17], for example). As λ was increased, rules tended to relax to periodic patterns, again after a relatively short transient phase. As λ reached a "critical value" λ_c, rules tended to have longer and longer transient phases. Additionally, the behavior in this regime exhibited long-lived, "complex" patterns—nonperiodic, but nonrandom. As λ was increased further, the average transient length decreased, and rules tended to relax to apparently random space-time patterns. The actual value of λ_c depended on r, k, and the actual path of the CA found as λ was incremented.

These four behavioral regimes roughly correspond to Wolfram's four classes. Langton's claim was that, as λ was increased from 0 to $1-(1/k)$, the classes were passed through in the order 1, 2, 4, 3. He noted that, as λ increases, "... one observes a *phase transition* between highly *ordered* and

highly *disordered* dynamics, analogous to the phase transition between the *solid* and *fluid* states of matter." ([17], p. 13.) According to Langton, as λ is increased from $1 - (1/k)$ to 1, the four regimes will occur in the reverse order, subject to some constraints for $k > 2$ [17]. For two-state CAs, there are two values of λ_c at which the complex regime will occur, since behavior is necessarily symmetric about $\lambda = 1/2$.

How is λ_c determined? Following standard practice, Langton used various statistics (such as single-site entropy, two-site mutual information, and transient length) to classify CA behavior. His additional step was to correlate behavior with λ via these statistics. Langton's Monte Carlo samples showed that there was some correlation between the statistics and λ. But the averaged statistics did not reveal a sharp transition in average behavior, a basic property of a phase transition in which macroscopic highly averaged quantities do make marked changes. We note that Wootters and Langton [33] gave evidence that the transition region narrows in the limit of an increasing number of states. Their main result indicates that there is a sharp transition in single-site entropy at $\lambda_c \approx 0.27$, in one class of two-dimensional infinite-state stochastic CAs.

The existence of a critical λ, and the dependence of the critical region's width on r and k, are less clear for finite-state CAs. Nonetheless, Packard empirically determined rough values of λ_c for $r = 3, k = 2$ CAs by looking at the *difference-pattern spreading rate*, γ, as a function of λ [24]. The spreading rate is a measure of unpredictability in spatiotemporal patterns, and is thus one possible measure of chaotic behavior [22, 31]. It is analogous (but not identical) to the Lyapunov exponent for continuous-state dynamical systems. In the case of CAs, it indicates the average propagation speed of information through space-time, though not the rate of production of local information.

At each λ, a large number of rules was sampled, and γ was estimated for each CA. The average for γ over the selected CA was taken as the average spreading rate at the given λ (the results are reproduced in Figure 2). At low and high λ, γ vanishes; at intermediate λ, it is maximal; and in the "critical" λ regions—centered about $\lambda \approx 0.23$ and $\lambda \approx 0.83$—it rises or falls gradually. (Li et al. (see [20], Appendix B) define λ_c as the onset of nonzero γ, and use mean-field theory to estimate λ_c in terms of r for two-state CAs. The value from their formula, setting $r = 3$, is $\lambda_c = 0.146$, which roughly matches the value for the onset of nonzero γ seen in Figure 2.)

Though not shown in Figure 2, the variance of γ is high for most values of λ. The same is true for single-site entropy and two-site mutual information as a function of λ [17]. In other words, the behavior of any *particular* rule at a given value of λ might be very different from the *average* behavior at that value. Thus, interpretation of these averages is somewhat problematic. The preceding account of the behavioral structure of CA rule space (as parameterized by λ) is based on statistics taken from Langton's and Packard's Monte Carlo simulations. (Various problems in correlating λ with behavior will be discussed in Section 8; a detailed analysis of some of these

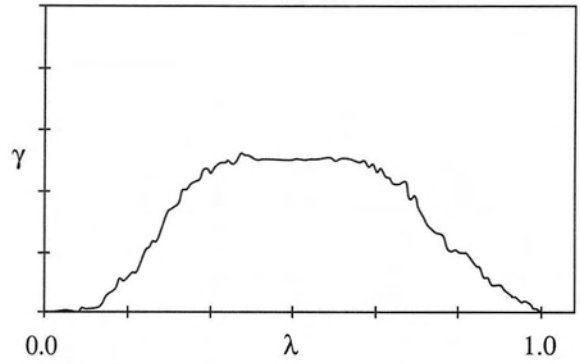

Figure 2: A graph of the average difference-pattern spreading rate, γ, of a large number of randomly chosen $r = 3, k = 2$ CAs, as a function of λ. (Adapted from [24], with the author's permission. No vertical scale was provided in the original.)

problems can be found in [3].) Other investigations of the structure of CA rule space are reported in [19, 20].

It is claimed in [17] that λ accurately predicts dynamical behavior only when the space of rules is large enough. Apparently, λ is not intended to be a good behavioral predictor for the space of elementary ($r = 1, k = 2$) CA rules and possibly not for $r = 3, k = 2$ rules either).

5. CA rule space and computation

Langton (in [17]) hypothesized that a CA's computational capability is related to its average dynamical behavior, which λ is claimed to predict. In particular, he hypothesized that CAs capable of performing nontrivial computation—including universal computation—are most likely to be found in the vicinity of "phase transitions" between order and chaos; that is, near λ_c values. This hypothesis relies on a basic observation of computation theory, that any form of computation requires memory (information storage) and communication (information transmission), and interaction between stored and transmitted information. In addition, however, universal computation requires memory and communication over arbitrary distances in time and space. Thus, complex computation requires significantly long transients and space-time correlation lengths; in the case of universal computation, arbitrarily long transients and correlations are required. Langton claimed that these phenomena are most likely to be seen near λ_c values—near "phase transitions" between order and chaos. This intuition was behind Langton's notion of "computation at the edge of chaos" for CA. (This should be contrasted with the analysis in [4, 5] of computation at the onset of chaos and, in particular, with the discussion, also in [4, 5], of the structure of CA space.)

6. Evolving CA

The empirical studies that we have just described addressed only the relationship between λ and the dynamical behavior of CA, as revealed by several statistics. They did not correlate λ, or behavior, with an independent measure of computation. Packard [24] addressed this issue by using a genetic algorithm (GA) [9, 13] to evolve CA rules to perform a particular computation. His experiment was meant to test two hypotheses: (1) CA rules able to perform complex computations are most likely to be found near λ_c values; and (2) when CA rules are evolved to perform a complex computation, evolution will tend to select rules near λ_c values.

6.1 The computational task, and an example CA

Packard's experiment consisted of evolving two-state ($s \in \{0, 1\}$) one-dimensional CAs with $r = 3$. The computational task for the CA was to decide whether or not the initial configuration consisted of more than half 1s. If so, the desired behavior for the CA was to relax, after some number of time steps, to a fixed-point pattern of all 1s. If the initial configuration consisted of less than half 1s, the desired behavior for the CA was to relax, after some number of time steps, to a fixed-point pattern of all 0s. If the initial configuration contained exactly half 1s, then the desired behavior was undefined. (This situation can be avoided in practice by requiring that the CA lattice be of odd length.) Thus, the desired CA had only two invariant patterns, all 1s or all 0s. In the following, we denote the density of 1s in a lattice configuration by ρ, the density of 1s in the configuration at time t by $\rho(t)$, and the threshold density for classification by ρ_c.

Does the $\rho_c = 1/2$ classification task qualify as a nontrivial computation for a small-radius ($r \ll N$) CA? Though the term "nontrivial" was not rigorously defined in [17] or [24], one possible definition might be any computation for which the memory requirement increases with N (that is, any computation which corresponds to the recognition of a nonregular language), and in which information must be transmitted over significant space-time distances (on the order of N). Under this definition, the $\rho_c = 1/2$ classification task can be thought of as a nontrivial computation for a small-radius CA. The effective minimum amount of memory required is proportional to $\log(N)$, because the equivalent of a counter register is required to track the excess of 1s in a serial scan of the initial pattern. And because the 1s can be distributed throughout the lattice, information transfer over long space-time distances must occur. This is supported in a CA by the nonlocal interactions among many different neighborhoods after some period of time.

Packard cited a $k = 2, r = 3$ rule constructed by Gacs, Kurdyumov, and Levin [6, 8], which purportedly performs this task. The Gacs-Kurdyumov-Levin (GKL) CA is defined by the following rule.

$$\text{If } s_i(t) = 0, \text{ then } s_i(t+1) = \text{ majority } [s_i(t), s_{i-1}(t), s_{i-3}(t)]$$
$$\text{If } s_i(t) = 1, \text{ then } s_i(t+1) = \text{ majority } [s_i(t), s_{i+1}(t), s_{i+3}(t)]$$

where $s_i(t)$ is the state of site i at time t. In words, this rule says that for each neighborhood 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.

Figure 1 gives space-time diagrams for the action of the GKL rule on two initial configurations: with $\rho < \rho_c$, and with $\rho > \rho_c$. Although the CA eventually converges to a fixed point, it can be seen that there is a transient phase during which a spatial and temporal transfer of information about local neighborhoods takes place; this local information interacts with other local information to produce the desired final state. Stated crudely, the GKL CA successively classifies "local" densities, with the locality range increasing with time. In regions where there is ambiguity, a "signal" is propagated. This is seen as either a checkerboard pattern propagated in both spatial directions or a vertical white-to-black boundary. These signals indicate that the classification is to be made at a larger scale. Note that both signals locally have $\rho = \rho_c$; as a result, the signal patterns can propagate, since the density of patterns with $\rho = \rho_c$ is not increased or decreased under the rule. In a simple sense, this is the CA's "strategy" for performing the computational task.

It has been claimed that the GKL CA performs the $\rho_c = 1/2$ task [18]; in fact, this is true only to an approximation. The GKL rule was not developed for the purpose of performing any particular computational task, but as part of studies of reliable computation and phase transitions in one spatial dimension. (The goal in the computation studies, for example, was to find a CA whose behavior is robust to small errors in the rule's update of the configuration.) It has been proved that the GKL rule has only two attracting patterns, all 1s or all 0s [6]. Attracting patterns in this context are those invariant patterns which return to the same pattern when slightly perturbed. It turns out that the basins of attraction for the all-1 and all-0 patterns are not precisely the initial configurations with $\rho > 0.5$ or $\rho < 0.5$, respectively. On finite lattices the GKL rule does classify most initial configurations according to this criterion, but on a significant number the "incorrect" attractor is reached. (The terms "attractor" and "basin of attraction" are used here in the sense of [6] and [12]. This differs substantially from the notion used, for example, in [34], where "attractor" refers to any invariant or time-periodic pattern, and "basin of attraction" refers to that set of finite lattice configurations relaxing to an attractor.)

One set of experimental measures of the GKL CA's classification performance is displayed in Figure 3. To make this plot, we ran the GKL CA on 500 randomly generated initial configurations close to each of 21 densities $\rho \in [0.0, 1.0]$. The fraction of correct classifications was then plotted at each ρ. The rule was run either until a fixed point was reached or for a maximum number of time steps equal to $10 \times N$. This was done for CA with three different lattice sizes: $N \in \{149, 599, 999\}$. Note that approximately 30% of

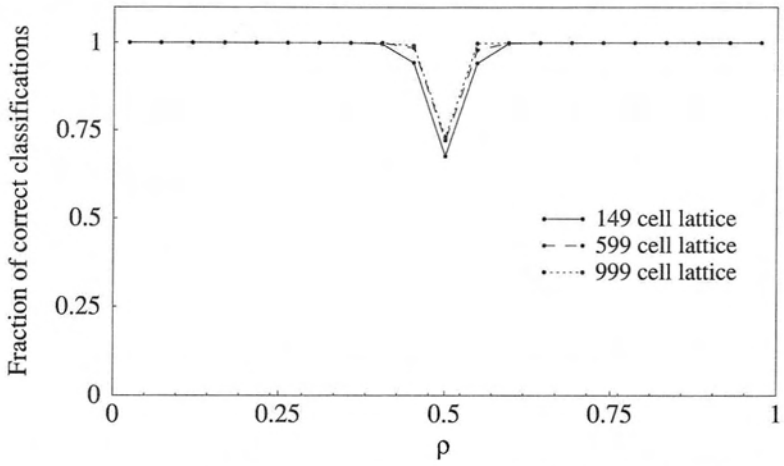

Figure 3: Experimental performance of the GKL rule as a function of $\rho(0)$ for the $\rho_c = 0.5$ task. Performance plots are given for three lattice sizes: $N = 149$ (the size of the lattice used in the GA runs), 599, and 999.

the initial configurations with $\rho \approx \rho_c$ were misclassified. All the incorrect classifications are made for initial configurations with $\rho \approx \rho_c$. In fact, the worst performances occur at $\rho = \rho_c$. The error region narrows with increasing lattice size.

The GKL rule table has $\lambda = 1/2$, not $\lambda = \lambda_c$. Given that it appears to perform a computational task of some complexity, at a minimum it is a deviation from the "edge of chaos" hypothesis for CA computation. The GKL rule's $\lambda = 1/2$ puts it right at the center of the "chaotic" region in Figure 2. This may seem puzzling, because the GKL rule clearly does not produce chaotic behavior during either its transient or asymptotic epochs—far from it, in fact. However, the λ parameter was intended to correlate with "average" behavior of CA rules at a given λ value. Recall that γ in Figure 2 represents an *average* over a large number of randomly chosen CA rules and, though not shown in that plot, the variance in γ for most λ values is high. Thus, as previously mentioned, the behavior of any *particular* rule at its λ value might be very different from the *average* behavior at that value.

More to the point, we *expect* a λ value close to $1/2$ for a rule that performs well on the $\rho_c = 1/2$ task. This is the case primarily because the task is symmetric with respect to the exchange of 1s and 0s. Suppose, for example, that a rule that carries out the $\rho_c = 1/2$ task has $\lambda < 1/2$. This implies that there are more neighborhoods in the rule table that map to output bit 0 than to output bit 1. This, in turn, means that there will be *some* initial configurations with $\rho > \rho_c$ on which the action of the rule will *decrease* the number of 1s, which is the opposite of the desired action. However, if the rule acts to *decrease* the number of 1s on an initial configuration with $\rho > \rho_c$,

it risks producing an intermediate configuration with $\rho < \rho_c$, which then would lead (under the original assumption that the rule carries out the task correctly) to a fixed point of all 0s, misclassifying the initial configuration. A similar argument holds in the other direction if the rule's λ value is greater than 1/2. This informal argument shows that a rule with $\lambda \neq 1/2$ will misclassify certain initial configurations. Generally, the further away from $\lambda = 1/2$ that the rule is, the greater the number of such initial configurations. Such rules may perform fairly well, classifying most initial configurations correctly. However, we expect any rule that performs reasonably well on the this task—in the sense of being close to the GKL CA's average performance shown in Figure 3—to have a λ value close to 1/2.

This analysis points to a problem with using the $\rho_c = 1/2$ task as an evolutionary goal for the study of the relationship between computation and λ. As shown in Figure 2, for an $r = 3, k = 2$ CA the λ_c values occur at roughly 0.23 and 0.83; one hypothesis that was to be tested by Packard's original experiment is that the GA will tend to select rules close to these λ_c values. But for the ρ-classification tasks, the range of λ values required for good performance is simply a function of the task and, specifically, of ρ_c. For example, the underlying 0-1 exchange symmetry of the $\rho_c = 1/2$ task implies that if a CA exists that can perform the task at an acceptable level, then it has $\lambda \approx 1/2$. Though this does not directly invalidate the adaptation hypothesis or claims about λ's correlation with *average* behavior, it presents problems for the use of ρ-classification tasks to gain evidence about a generic relation between λ and computational capability.

6.2 The original experiment

Packard used a GA to evolve CA rules to perform the $\rho_c = 1/2$ task [24]. His GA began with a randomly generated initial population of CA rules. Each rule was represented as a bit string containing the output bits of the rule table. That is, the bit at position 0 (i.e., the leftmost position) in the string is the state to which the neighborhood 0000000 is mapped, the bit at position 1 in the string is the state to which the neighborhood 0000001 is mapped, and so on. The initial population was randomly generated, but it was constrained to be uniformly distributed across λ values between 0.0 and 1.0.

A given CA rule in the population was evaluated for ability to perform the classification task by choosing an initial configuration at random, running the CA on that initial configuration for some specified number of time steps, and measuring the fraction of cells in the lattice that had the correct state at the final time step. (For initial configurations with $\rho > \rho_c$, the correct final state for each cell is 1, and for initial configurations with $\rho < \rho_c$, the correct final state for each cell is 0.) For example, if the CA were run on an initial configuration with $\rho > \rho_c$ and at the final time step the lattice contained 90% 1s, the CA's score on that initial configuration would be 0.9. The fitness of a rule was simply the rule's average score over a set of initial

configurations. For each rule in the population, Packard generated a set of initial configurations that were uniformly distributed across ρ values from 0 to 1.

Actually, a slight variation on this method was used in [24]. Instead of measuring the fraction of correct states in the final lattice, the GA measured the fraction of correct states over configurations from a small number n of final time steps [23]. This prevented the GA from evolving rules that were temporally periodic; for example, those with patterns that alternated between all 0s and all 1s. Such rules obtained higher than average fitness at early generations by often landing at the "correct" phase of the oscillation for a given initial configuration. On the next time step the classification would have been incorrect. In our experiments we used a slightly different method to address this problem, which is explained in subsection 7.1.

Packard's GA worked as follows. At each generation

1. The fitness of each rule in the population was calculated,
2. The population was ranked by fitness,
3. Some fraction of the lowest fitness rules were removed,
4. The removed rules were replaced by new rules formed by crossover and mutation from the remaining rules.

Crossover between two strings involves randomly selecting a position in the strings and exchanging parts of the strings before and after that position. Mutation involves flipping one or more bits in a string, with some low probability. A diversity-enforcement scheme was also used to prevent the population from converging too early and losing diversity [23]. If a rule was formed that was too close in Hamming distance (i.e., the number of matching bits) to existing rules in the population, its fitness was decreased.

The results from Packard's experiment are displayed in Figure 4. The two histograms display the observed frequency of rules in the GA population as a function of λ, with rules merged from a number of different runs. The top graph gives this data for the initial generation; the rules are uniformly distributed over λ values. The middle graph gives the same data for the final generation—in this case, after the GA has run for 100 generations. The rules now cluster around the two λ_c regions, as can be seen by comparison with the difference-pattern spreading rate plot, reprinted at the bottom of the figure. Note that each individual run produced rules at one or the other peak in the middle graph, so when the runs were merged together, both peaks appear [23]. Packard interpreted these results as evidence for the hypothesis that, when an ability for complex computation is required, evolution tends to select rules near the transition to chaos. Like Langton, he argues that this result intuitively makes sense, because "rules near the transition to chaos have the capability to selectively communicate information with complex structures in space-time, thus enabling computation" [24].

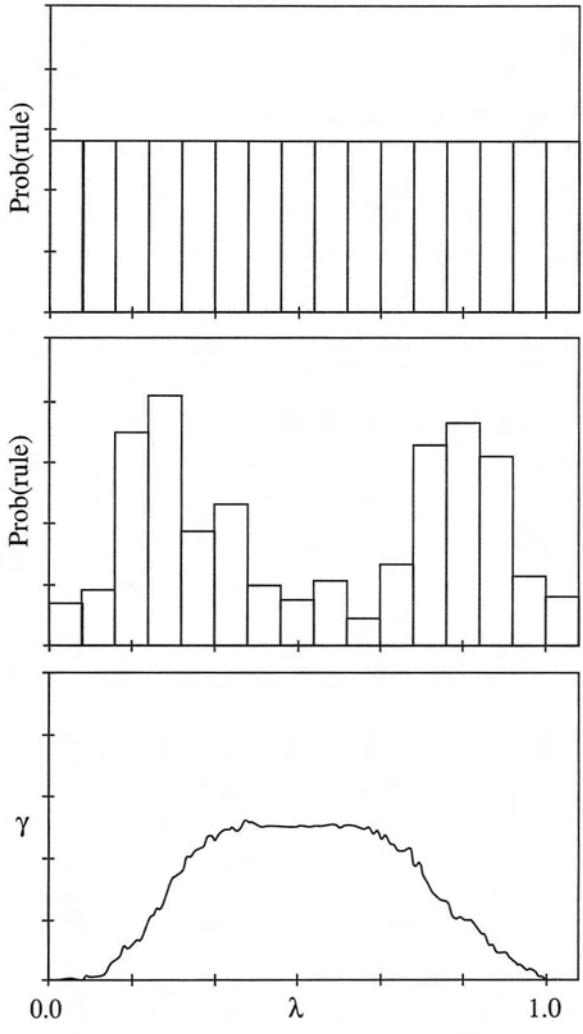

Figure 4: Results from Packard's original experiment on GA evolution of CA for the $\rho_c = 1/2$ classification task. The top two figures are populations of CA at generations 0 and 100, respectively, versus λ. The bottom figure is Figure 2, reproduced here for reference. (Adapted from [24], with the author's permission.)

7. New experiments

As the first step in a study of the reliability of these general conclusions, we carried out a set of experiments similar to those that we have just described. We were unable to obtain precise details of some of the original experiment's parameters, such as the exact population size for the GA, the mutation rate, and so on. As a result, we used what we felt were reasonable values for these parameters. We carried out a number of parameter sensitivity tests which indicated that varying the parameters within small bounds did not change our qualitative results.

7.1 Details of our experiments

In our experiments, as in the original, the CA rules in the population all had $r = 3$ and $k = 2$. Thus, the bit strings that represented the rules were of length $2^{2r+1} = 128$. The size of this search space is huge—the number of possible CA rules is 2^{128}. The tests for each CA rule were carried out on lattices of length $N = 149$ with periodic boundary conditions. The population size was 100, which was roughly the population size used in the original experiment [23]. The initial population was generated at random, but constrained to be uniformly distributed among different λ values. A rule's fitness was estimated by running the rule on 300 randomly generated initial configurations that were uniformly distributed over $\rho \in [0.0, 1.0]$. Exactly half the initial configurations had $\rho < \rho_c$, and the other half had $\rho > \rho_c$.

Exact symmetry in the initial configurations at each generation was necessary to avoid early biases in the λ of selected rules. If 49%, say, of the initial configurations had $\rho < \rho_c$, and 51% had $\rho > \rho_c$, rules with λ close to 1 would obtain slightly higher fitness than rules with λ close to 0, because rules with λ close to 1 would map most initial configurations to all 1s. A rule with, say, $\lambda \approx 1$ would in this case classify 51% of the initial configurations correctly whereas a rule with $\lambda \approx 0$ would classify only 49% correctly. But such slight differences in fitness have a large effect in the initial generation, when all rules have fitness close to 0.5, because the GA selects the 50 *best* rules, even if they are only very slightly better than the 50 *worst* rules. This biases the representative rules in the early population. And this bias can persist well into the later generations.

We allowed each rule to run for a maximum number M of iterations, where a new M was selected for each rule from a Poisson distribution with mean 320. This is the measured maximum amount of time for the GKL CA to reach an invariant pattern over a large number of initial configurations on lattice size 149.[1] A rule's fitness was its average score—the fraction of cell

[1]It may not be necessary to allow the maximum number of iterations to vary. However, in some early tests with smaller sets of fixed initial configurations, we found the same problem that Packard reported [23]: if M was fixed, then period-2 rules evolved that alternated between all 0s and all 1s. These rules adapted to the small set of initial configurations and the fixed M by landing at the "correct" pattern for a given initial configuration at time step M, only to move to the opposite pattern and wrong classification

states correct at the last iteration—over the 300 initial configurations. We termed this fitness function *proportional fitness*, to contrast with a second fitness function—*performance fitness*—which we describe subsequently. A new set of 300 initial configurations was generated every generation; at each generation, all the rules in the population were tested on this set. Notice that this fitness function is stochastic—the fitness of a given rule may vary a small amount from generation to generation depending on the particular set of 300 initial configurations used in testing it.

Our GA was similar to Packard's: the fraction of new strings each new generation—the "generation gap"—was 0.5. In other words, once the population was ordered according to fitness, the top half of the population—the set of "elite" strings—was copied without modification into the next generation. To GA practitioners more familiar with nonoverlapping generations, this may sound like a small generation gap. However, testing a rule on 300 "training cases" does not necessarily provide a very reliable gauge of what the fitness would be over a larger set of training cases; our selected gap was a good way of making a "first cut," and allowing rules that survived to be tested over more initial configurations. As a new set of initial configurations 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 had been tested with a large set of initial configurations. An alternative method would have been to test every rule in every generation on a much larger set of initial configurations but, given the amount of computer time involved, that method seemed unnecessarily wasteful. Too much effort, for example, would have gone into testing very weak rules, which could safely be weeded out early using our method.

The remaining half of the population for each new generation was created by crossover and mutation from the previous generation's population. (This method of producing non-elite strings differs from that in [24], where the non-elite strings were formed from crossover and mutation among the elite strings only, rather than from the entire population. We observed no statistically significant differences in our tests using the latter mechanism, other than a modest difference in time scale.) Twenty-five pairs of parent rules were chosen at random with replacement from the entire previous population. For each pair, a single crossover point was selected at random, and two offspring were created by exchanging the subparts of each parent before and after the crossover point. The two offspring then underwent mutation, which consisted of flipping a randomly chosen bit in the string. The number of mutations for a given string was chosen from a Poisson distribution with a mean of 3.8 (this is equivalent to a per-bit mutation rate of 0.03). Again, to GA practitioners this may seem to be a high mutation rate, but one must take into account that half the population was copied without modification at every generation.

---

at time step $M + 1$. These rules performed very poorly when tested on a different set of initial configurations—evidence for "overfitting."

7.2 Results of the proportional-fitness experiment

We performed 30 runs of the GA with the parameters described above, each with a different random-number seed. On each run the GA was iterated for 100 generations. We found that running the GA longer (up to 300 generations) did not result in improved fitness. The results of this set of runs are displayed in Figure 5. Figure 5(a) is a histogram of the frequency of rules in the initial populations as a function of λ, merging the rules from all 30 initial populations; thus, the total number of rules represented in this histogram is 3000. The λ bins in this histogram are those that were used by Packard, each of width 0.0667. Packard's highest bin contained only rules with $\lambda = 1$ (that is, rules that consist of all 1s). We have merged this bin with the immediately lower bin.

The initial populations consisted of randomly generated rules uniformly spread over the λ values between 0.0 and 1.0. The mean and best fitness values for each bin are also plotted. These are all near 0.5, which is to be expected for a set of randomly generated rules under this fitness function. The best fitnesses are slightly higher in the very low and very high λ bins, because rules with output bits that are almost all 0s (or 1s) correctly classify all low density (or all high density) initial configurations. In addition, such CAs obtain small partial credit on some high density (low density) initial configurations and, thus, have fitnesses sightly higher than 0.5.

Figure 5(b) shows the histogram for the final generation (100), merging rules from the final generations of all 30 runs. Again, the mean and best fitness values for each bin are plotted. In the final generation the mean fitnesses are all near 0.8. The exceptions are the central bin (with a mean fitness of 0.72) and the leftmost bin (with a mean fitness of 0.75). The leftmost bin contains only five rules—each at $\lambda \approx 0.33$, right next to the bin's upper λ limit. The standard deviations of mean fitness for each bin (not shown in the figure) are all approximately 0.15—except for the leftmost bin, which has a standard deviation of 0.20. The best fitnesses for each bin are all between 0.93 and 0.95—except for the leftmost bin, which has a best fitness of 0.90. Under this fitness function the GKL rule has fitness ≈ 0.98; the GA never found a rule with fitness above 0.95.

The fitness function is stochastic: a given rule might be assigned a different fitness each time the fitness function is evaluated. The standard deviation for a given rule under the present fitness scheme is approximately 0.015. This indicates that the differences among the best fitnesses plotted in the histogram are not significant, except for that in the leftmost bin.

The lower mean fitness in the central bin is due to the fact that the rules in that bin predominantly come from non-elite rules generated by crossover and mutation in the final generation. This is a combinatorial effect: the density of CA rules as a function of λ is very highly peaked about $\lambda = 1/2$. (We will return to this "combinatorial drift" effect.) Many of the rules in the middle bin have not yet undergone selection and, thus, tend to have lower fitnesses than rules that have been selected in the elite. This effect disappears

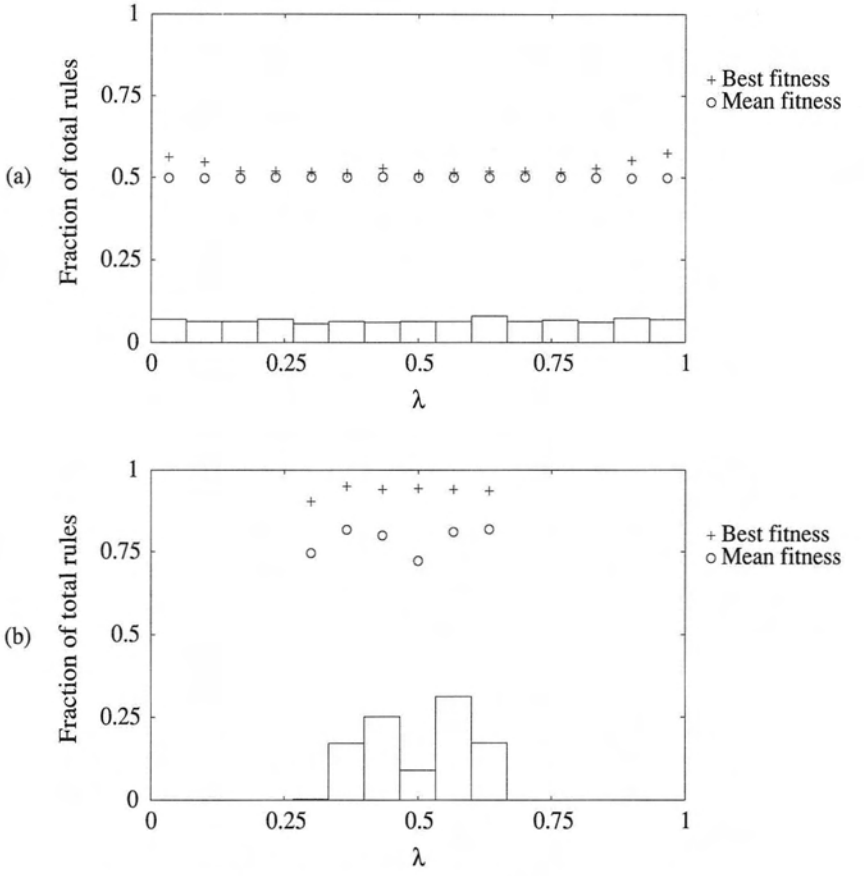

Figure 5: Results from our experiment with proportional fitness. Histogram (a) plots the frequencies of rules merged from the initial generations of 30 runs as a function of λ. Following [24], the x-axis is divided into 15 bins of length 0.0667 each. The rules with $\lambda = 1.0$ are included in the rightmost bin. Histogram (b) plots the frequencies of rules merged from the final generations (generation 100) of these 30 runs. In each histogram the best and mean fitnesses are plotted for each bin. (The y-axis interval for fitnesses is also [0,1]).

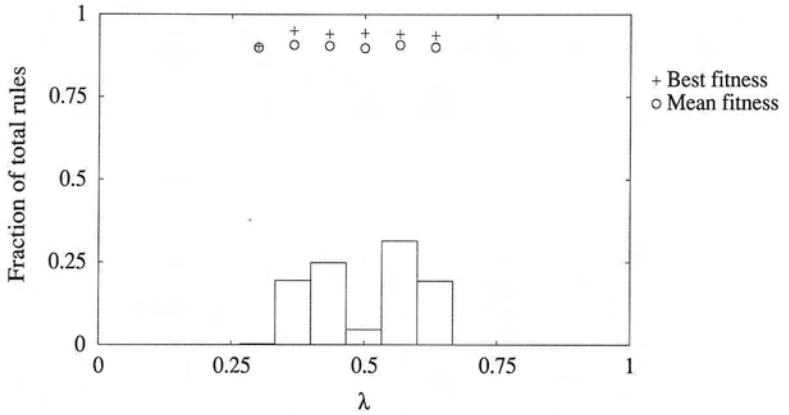

Figure 6: Histogram including only the elite rules from the final generations of the 30 runs with the proportional-fitness function.

in Figure 6, which includes only the elite rules at generation 100 for the 30 runs: the difference in mean fitness disappears and the height of the central bin is decreased by half.

The results presented in Figure 5(b) are strikingly different from the results of Packard's experiment. In the final generation histogram in Figure 4, most of the rules clustered around $\lambda \approx 0.23$ or $\lambda \approx 0.83$. In Figure 5(b), however, there are no rules in these λ_c regions. Rather, the rules cluster much more closely around $\lambda = 1/2$—with a ratio of variances of 4 between the two distributions. Recall that this clustering is what we would expect from the basic 0–1 exchange symmetry of the $\rho_c = 1/2$ task.

One rough similarity between the results of the two experiments is the presence of two peaks centered around a dip at $\lambda \approx 0.5$—a phenomenon which we will explain shortly, and which is a key to understanding the GA's behavior on this problem. Nonetheless, there are significant differences, even within this similarity. As was already noted in Packard's experiment, the peaks are in bins centered about $\lambda \approx 0.23$ and $\lambda \approx 0.83$, but in Figure 5(b), the peaks are very close to $\lambda = 1/2$, being centered in the neighboring bins—those with $\lambda \approx 0.43$ and $\lambda \approx 0.57$. Thus, the ratio of original to current peak spread is roughly a factor of 4. Additionally, in the final-generation histogram of Figure 4, the two highest bin populations are roughly five times as high as the central bin, whereas in Figure 5(b) the two highest bins are roughly three times as high as the central bin. Finally, the final-generation histogram in Figure 4 shows the presence of rules in every bin; in Figure 5(b), there are rules in six of the central bins only.

As in Packard's experiment, we found that on any given run the population was clustered about one or the other peak but not both. (Thus, in the histograms that merge all runs, two peaks appear.) This is illustrated in

Figure 7, which displays histograms from the final generations of two individual runs. In one of these runs, the population clustered to the left of the central bin and in the other run it clustered to the right of the center. The fact that different runs resulted in different clustering locations was our reason for performing many runs and merging the results, rather than performing a single run with a much larger population—the latter method might have yielded only one peak. In other words, independent of the population size, a given run will be driven by (and the population organized around) the fit individuals that appear earliest. Thus, examining an ensemble of individual runs reveals additional details of the evolutionary dynamics.

The asymmetry in the heights of the two peaks in Figure 5(b) results from a small statistical asymmetry in the results of the 30 runs. In 14 runs, the rules clustered at the lower λ bin, and in 16 runs, the rules clustered at the higher λ bin. This difference is not significant, but it explains the small asymmetry in the peak heights.

We extended 16 of the 30 runs to 300 generations, and found that the basic shape of the histogram does not change significantly (just as the fitnesses do not increase).

7.3 Effects of drift

The results of our experiments suggest that an evolutionary process modeled by a genetic algorithm tends to select rules with $\lambda \approx 1/2$, for the $\rho_c = 1/2$ task. This is what we had expected, given our prior theoretical discussion concerning this task and its symmetries. We postpone until the next section a discussion of the curious feature near $\lambda = 1/2$ (the dip surrounded by two peaks). In this section, we focus on the larger-scale clustering in that λ region.

To understand that clustering we need to understand the degree to which the selection of rules close to $\lambda = 1/2$ is due to an intrinsic selection pressure, and the degree to which it is due to "drift." By drift we refer to the force that derives from the combinatorial aspects of CA space as explored by random selection ("genetic drift"), combined with the effects of crossover and mutation. The intrinsic effect of random selection with crossover and mutation is to move the population, irrespective of any selection pressure, to $\lambda = 1/2$. This is illustrated in Figure 8 by a histogram mosaic. These histograms show the frequencies of the rules in the population as a function of λ, for every 5 generations. Rules were merged from 30 runs in which selection according to fitness was turned off—that is, the fitnesses of the rules in the population were never calculated, and at each generation the selection of the elite group of strings was performed at random. Otherwise, the runs remained the same as previously. Because there is no fitness-based selection, drift is the only force at work. Under the effects of random selection, crossover, and mutation, by the tenth generation the population has largely drifted to the region of $\lambda = 1/2$, and this clustering becomes increasingly pronounced as the run continues.

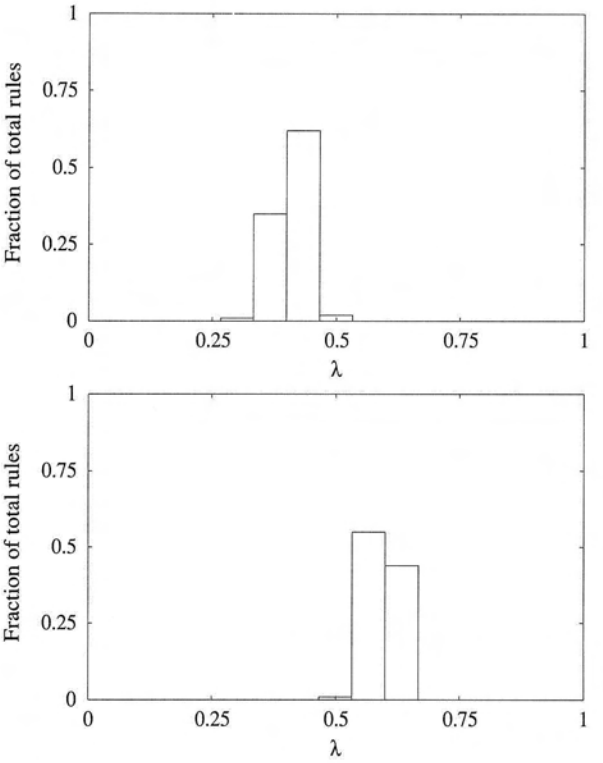

Figure 7: Histograms from the final generations of two individual runs of the GA employing proportional fitness. Each run had a population of 100 rules. The final distribution of rules in each of the 30 runs resembled one of these two histograms.

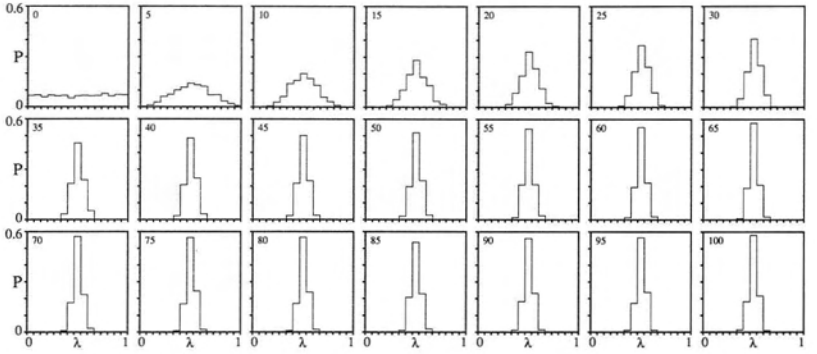

Figure 8: *No-selection mosaic.* Rule-frequency-versus-λ histograms, given every five generations, for populations evolved under the genetic algorithm with no selection (that is, the fitness function was not calculated and the selection of elite rules was performed at random). Each histogram was merged from 30 runs; each run had a population of 100 rules. The generation number is given in the upper left corner of each histogram.

This drift to $\lambda = 1/2$ is related to the combinatorics of the space of bit strings. For binary CA rules with neighborhood size n $(= 2r + 1)$, this 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 point out that the size of the subspace is binomially distributed with respect to λ, as follows.

$$|\mathrm{CA}(\lambda, n)| = \binom{2^n}{\lambda 2^n}$$

The distribution is symmetric in λ and tightly peaked about $\lambda = 1/2$, with variance $\propto 2^{-n}$. Thus, the vast majority of rules are found 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 8 gives the GA's rough approximation of this distribution.

Drift is thus a powerful force moving the population to cluster around $\lambda = 1/2$. For comparison, Figure 9 gives the rule-frequency-versus-λ histograms for the merged populations from 30 runs of our proportional-fitness experiment, for every five generations. (A similar mosaic plotting only the elite strings at each generation looks qualitatively similar.) The last histogram in this figure is the same as the one that was displayed in Figure 5(b).

The histograms in Figure 9 look roughly similar to those in Figure 8, up to generation 35. The primary difference in generations 0–30 is that Figure 9 indicates a more rapid peaking about $\lambda = 1/2$. The increased speed of movement to the center is presumably due to the additional evolutionary pressure of proportional fitness. At generation 35, a new feature appears. The peak in the center has begun to shrink significantly and the two surrounding bins are beginning to rival it in magnitude. By generation 40 the right-of-center

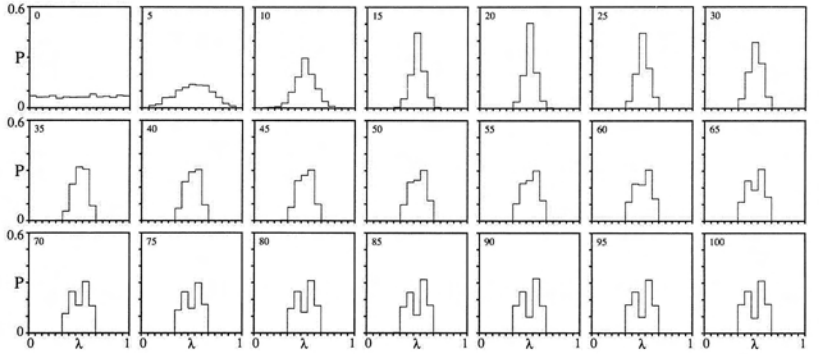

Figure 9: *Proportional-fitness mosaic.* Rule-frequency-versus-λ histograms, given every five generations, merged from the 30 GA runs with proportional fitness. Each run had a population of 100 rules.

bin has exceeded the central bin, and by generation 65 the histogram has developed two peaks surrounding a dip in the center. The dip becomes increasingly pronounced as the run continues, but stabilizes by generation 85 or so.

The differences between Figures 8 and 9, over all 100 generations, show that the population's structure in each generation in Figure 9 is not solely due to drift. Indeed, after generation 35, the distinctive features of the population indicate new, qualitatively different, and unique properties due to the selection mechanism. The two peaks represent a symmetry breaking in the evolutionary process—the rules in each individual run initially are clustered around $\lambda = 1/2$, but move to one side or the other of the central bin by about generation 35. We discuss the causes of this symmetry breaking in the next subsection.

7.4 Evolutionary mechanisms: symmetry breaking and the dip at $\lambda = 1/2$

At this point we move away from questions related to Packard's experiment, and concentrate on the mechanisms involved in producing our results. Two major questions must be answered: Why are there significantly fewer rules in the central bin than in the two surrounding bins, in the final generation? What causes the symmetry breaking that begins near generation 35 (as seen in Figure 9)?

The answers (in the briefest terms) obtained by detailed analysis of the 30 GA runs, are as follows. The course of CA evolution under our GA falls roughly into four "strategy" epochs. Each epoch is associated with an innovation discovered by the GA for solving the problem. Though the absolute time at which these innovations appear in each run varies somewhat, each run essentially passes through the four epochs in succession. The epochs

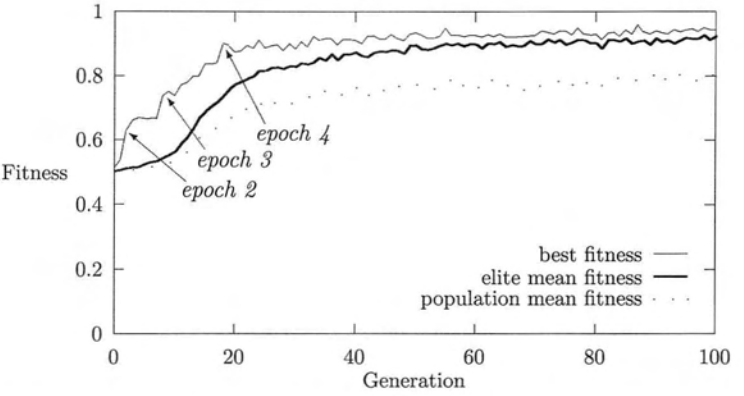

Figure 10: Best fitness, elite mean fitness, and population mean fitness, versus generation for one typical run. The beginnings of epochs 2–4 are pointed out on the best-fitness plot. Epoch 1 begins at generation 0.

are shown in Figure 10 (which plots the best fitness, the mean fitness of the elite strings, and the mean fitness of the population, versus generation, for one typical run of the GA). The beginnings of epochs 2 through 4 are indicated on the best-fitness plot. Epoch 1 begins at generation 0.

Epoch 1: Randomly generated rules

The first epoch starts at generation 0, when the best fitness in the initial generation is approximately 0.5 and the λ values are uniformly distributed between 0.0 and 1.0. No rule is much fitter than any other rule, though rules with very low and very high λ tend to have slightly higher fitness, as shown in Figure 5(a). The strategy in this epoch—if it can be called a strategy at all—derives from only the most elementary aspect of the task. Rules either specialize for $\rho > \rho_c$ configurations by mapping high-density neighborhoods in the CA rule table to 1, or specialize for $\rho < \rho_c$ configurations by mapping low-density neighborhoods to 0.

Epoch 2: Discovery of the two halves of the rule table

The second epoch begins when a rule is discovered for which most neighborhood patterns in the rule table that have $\rho < \rho_c$ map to 0, and most neighborhood patterns in the rule table that have $\rho > \rho_c$ map to 1. Under the coding scheme we have used, this is roughly correlated with the left and right halves of the rule table: neighborhoods 0000000 to 0111111, and 1000000 to 1111111, respectively. Such a strategy is (presumably) easy for the GA to discover, due to the tendency of single-point crossover to preserve contiguous sections of the rule table. It differs from the accidental strategy

of epoch 1 in that there is an organization to the rule table: output bits are roughly associated with densities of neighborhood patterns. It is the first significant attempt at distinguishing initial configurations with more 1s than 0s, and vice versa. Under our fitness function, the fitness of such rules is, approximately, between 0.6 and 0.7, which is significantly higher than the fitness of the initial random rules. This innovation typically occurs between generations 1 and 10; in the run displayed in Figure 10 it occurred in generation 2, and can be seen as the steep rise in the best-fitness plot at that generation. All such rules tend to have λ close to 1/2. There are many possible variations on these rules, with similar fitness, so such rules—all close to $\lambda = 1/2$—begin to dominate in the population. This fact, along with the natural tendency for the population to drift toward $\lambda = 1/2$, is the cause of the clustering around $\lambda = 1/2$ seen by generation 10 in Figure 9. For the next several generations the population tends to explore small variations on this broad strategy. This can be seen in Figure 10 as the leveling off in the best-fitness plot between generations 2 and 10.

Epoch 3: Growing blocks of 1s or 0s

The next epoch begins when the GA discovers either of two new strategies. The first strategy is to increase the size of a sufficiently large block of adjacent or nearly adjacent 1s; the second strategy is to increase the size of a sufficiently large block of adjacent or nearly adjacent 0s.

Examples of these two strategies are illustrated in Figures 11 and 12. These figures give space-time diagrams from two rules that marked the beginning of this epoch in two different runs of the GA. Figure 11 illustrates the action of a rule discovered at generation 9 of one run. This rule has $\lambda \approx 0.41$, which means that the rule maps most neighborhoods to 0. Its strategy is to map initial configurations to mostly 0s—the configurations it produces have $\rho < \rho_c$, unless the initial configuration contains a sufficiently large block of 1s, in which case it increases the size of that block. Figure 11(a) shows how the rule evolves an initial configuration with $\rho < \rho_c$ to a final lattice with mostly 0s. This produces a fairly good score. Figure 11(b) shows how the rule evolves an initial configuration with $\rho > \rho_c$. The initial configuration contains a few sufficiently large blocks of adjacent or nearly adjacent 1s, and the size of these blocks is quickly increased to yield a final lattice with all 1s for a perfect score. The fitness of this rule at generation 9 was ≈ 0.80.

Figure 12 illustrates the action of a second rule, discovered at generation 20 in another run. This rule has $\lambda \approx 0.58$, which means that the rule maps most neighborhoods to 1. Its strategy is the inverse of the previous rule. It maps initial configurations to mostly 1s unless the initial configuration contains a sufficiently large block of 0s, in which case it increases the size of that block. Figure 12(a) illustrates this for an initial configuration with $\rho < \rho_c$; here a sufficiently large block of 0s appears in the initial configuration and is increased in size, yielding a perfect score. Figure 12(b) shows the action of the same rule on an initial configuration with $\rho > \rho_c$. Most neighborhoods

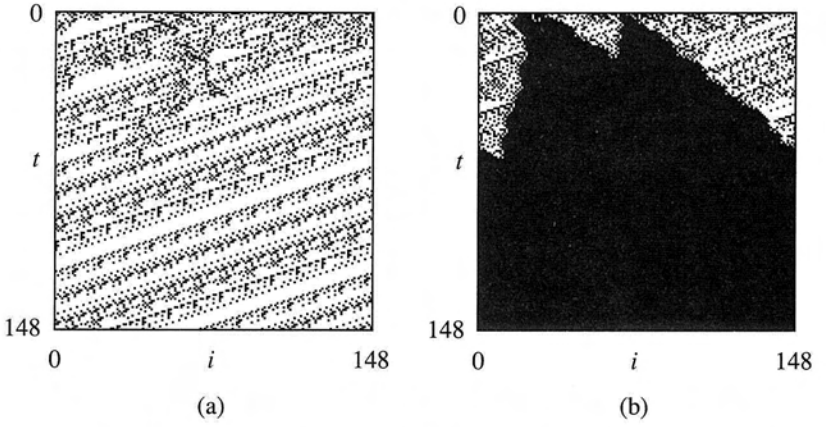

Figure 11: Space-time diagrams of one epoch-3 rule with $\lambda \approx 0.41$ that increases sufficiently large blocks of adjacent or nearly adjacent 1s. Both diagrams have $N = 149$ and are iterated for 149 time steps (the time displayed here is shorter than the actual time allotted under the GA). In (a), $\rho(0) \approx 0.40$ and $\rho(148) \approx 0.17$. In (b), $\rho(0) \approx 0.54$ and $\rho(148) = 1.0$. Thus, in (a) the classification is incorrect, but partial credit is given; in (b) it is correct.

are mapped to 1 so the final configuration contains mostly 1s, yielding a fairly high score. The fitness of this rule at generation 20 was ≈ 0.87.

The general idea behind these two strategies is to rely on statistical fluctuations in the initial configurations. An initial configuration with $\rho > \rho_c$ is likely to contain a sufficiently large block of adjacent or nearly adjacent 1s. A rule like the one illustrated in Figure 11 then increases this region's size to yield the correct classification. This holds similarly for rules like the one illustrated in Figure 12, with respect to blocks of 0s in initial configurations with $\rho < \rho_c$. In short, these strategies are assuming that the presence of a sufficiently large block of 1s or 0s is a good predictor of $\rho(0)$.

Similar strategies were discovered in every run; they typically emerged by generation 20. Any single strategy increased blocks of 0s or blocks of 1s, but not both. These strategies result in a significant jump in fitness: typical fitnesses for the first instances of such strategies range from 0.75 to 0.85. This jump in fitness can be seen in the run of Figure 10 at approximately generation 10, and is marked as the beginning of epoch 3. This is the first epoch in which a substantial increase in fitness is associated with a symmetry breaking in the population. The symmetry breaking involves deciding whether to increase blocks of 1s or blocks of 0s. The GKL rule is perfectly symmetric with respect to the increase of blocks of 1s and 0s. The GA on the other hand tends to discover one or the other strategy, and the one that is discovered first tends to take over the population, moving the population

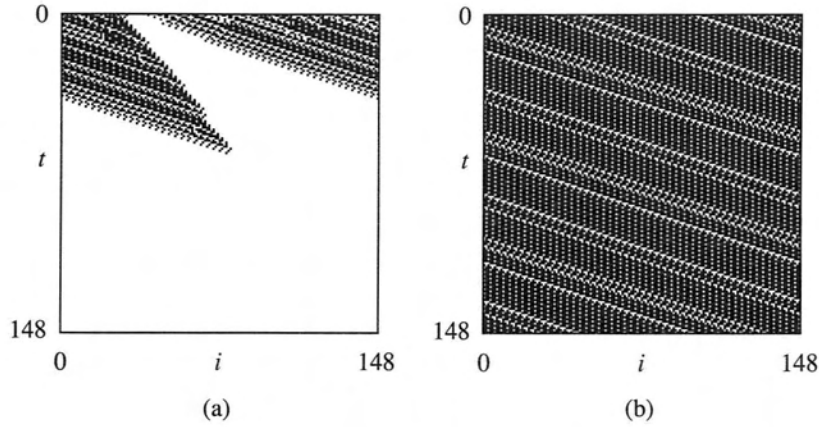

Figure 12: Space-time diagrams of one epoch-3 rule with $\lambda \approx 0.58$ that increases sufficiently large blocks of adjacent or nearly adjacent 0s. In (a), the initial configuration with $\rho \approx 0.42$ maps to a correct classification pattern of all 0s. In (b), the initial configuration with $\rho \approx 0.56$ is not correctly classified ($\rho(148) \approx 0.75$), but partial credit is given.

λ's to one or the other side of 1/2. The causes of the symmetry breaking are explained following the description of epoch 4.

Typically, the first instances of epoch-3 strategies have a number of problems. As shown in Figures 11 and 12, the rules often rely on partial credit to achieve fairly high fitness on structurally incorrect classification. They do not get perfect scores on many initial configurations, and they often make mistakes in classification. Three common types of classification errors are illustrated in Figure 13. Figure 13(a) illustrates a rule increasing a too-small block of 1s and thus misclassifying an initial configuration with $\rho < \rho_c$. Figure 13(b) illustrates a rule that does not increase blocks of 1s fast enough on an initial configuration with $\rho > \rho_c$, leaving many incorrect bits in the final lattice. Figure 13(c) illustrates the *creation* of a block of 1s that did not appear in an initial configuration with $\rho < \rho_c$, ultimately leading to a misclassification. The rules that produced these diagrams come from epoch 3 in various GA runs.

The increase in fitness seen in Figure 10 between generations 10 and 20 or so is due to further refinements of the basic strategies, which correct these problems to some extent.

Epoch 4: Reaching and staying at a maximal fitness

In most runs, the best fitness is at its maximum value of 0.90 to 0.95 by generation 40 or so. In Figure 10 this occurs at approximately generation 20, and is marked as the beginning of epoch 4. The best fitness does not increase

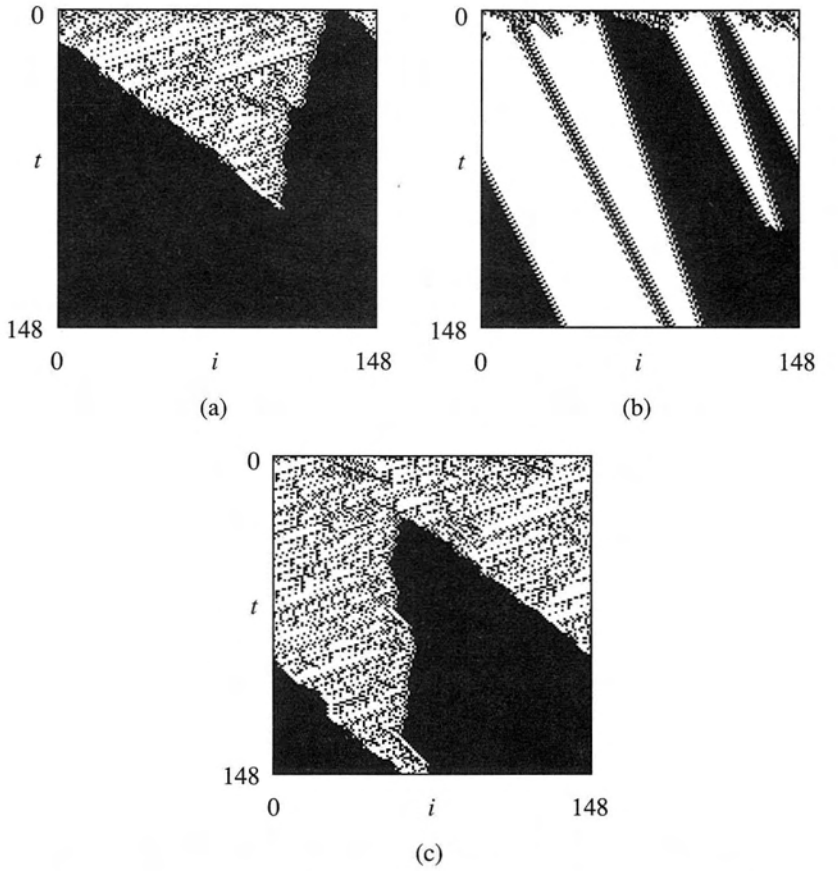

Figure 13: Space-time diagrams illustrating three types of classification errors committed by epoch-3 rules: (a) growing a block of 1s in a sea of $\rho < \rho_c$, (b) growing blocks of 1s too slowly for an initial configuration with $\rho > \rho_c$ (the correct fixed point of all 1s does not occur until iteration 480), and (c) generating a block of 1s from a sea of $\rho < \rho_c$ and growing it so that $\rho > \rho_c$ (the incorrect fixed point of all 1s occurs at iteration 180). The initial configuration densities are (a) $\rho(0) \approx 0.39$, (b) $\rho(0) \approx 0.59$, and (c) $\rho(0) \approx 0.45$.

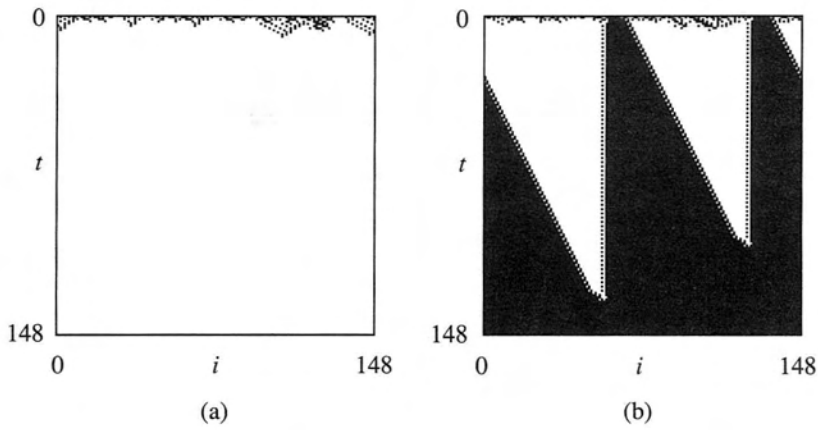

Figure 14: Space-time diagrams of one epoch-4 rule with $\lambda \approx 0.38$ that increases sufficiently large blocks of adjacent or nearly adjacent 1s. In (a), $\rho(0) \approx 0.44$; in (b), $\rho(0) \approx 0.52$. Both initial configurations are correctly classified.

significantly after this; the GA simply finds a number of variations of the best strategies, which all have roughly the same fitness. When we extended 16 of the 30 runs to 300 generations, we did not see any appreciable increase in the best fitness.

The actions of the best rules from generation 100 of two separate runs are shown in Figures 14 and 15. The space-time diagrams on the left in each figure are for initial configurations with $\rho < \rho_c$, and the diagrams on the right are for initial configurations with $\rho > \rho_c$. The rule illustrated in Figure 14 has $\lambda \approx 0.38$; its strategy is to map initial configurations to 0s unless there is a sufficiently large block of adjacent or nearly adjacent 1s, which if present is increased. The rule shown in Figure 15 has $\lambda = 0.59$ and has the opposite strategy. Each of these rules has fitness ≈ 0.93. They are better tuned versions of the rules in Figures 11 and 12.

Symmetry breaking in epoch 3

Notice that the λ values of the rules that have been described are in the bins centered around 0.43 and 0.57 rather than 1/2. In fact, it seems to be much easier for the GA to discover versions of the successful strategies close to $\lambda = 0.43$ and $\lambda = 0.57$ than to discover them close to $\lambda = 1/2$, though some instances of the latter rules were found. Why is this the case? One reason is that rules with high or low λ work well by *specializing*. The rules with low λ map most neighborhoods to 0s and then increase sufficiently large blocks of 1s when they appear. Rules with high λ specialize in the opposite direction. A rule at $\lambda = 1/2$ cannot easily specialize in this way. Another reason is that a successful rule that grows sufficiently large blocks of (say) 1s must avoid *creating* a sufficiently large block of 1s from an initial configuration with less

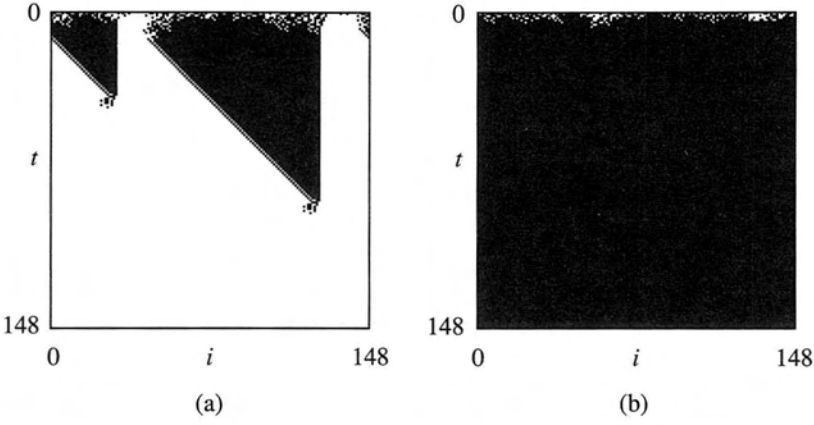

Figure 15: Space-time diagrams of one epoch-4 rule with $\lambda \approx 0.59$ that increases sufficiently large blocks of adjacent or nearly adjacent 0s. In (a), $\rho(0) \approx 0.40$; in (b), $\rho(0) \approx 0.56$. Both initial configurations are correctly classified.

than half 1s. Doing so will lead it to increase the block of 1s and produce an incorrect answer, as was seen in Figure 13(c). An easy way for a rule to avoid creating a sufficiently large block of 1s is to have a low λ. This ensures that low-density initial configurations will quickly map to all 0s, as was seen in Figure 14(a). Likewise, if a rule increases sufficiently large blocks of 0s, it is safer for the rule to have a high λ value so it will avoid creating sufficiently large blocks of 0s where none existed. A rule close to $\lambda = 1/2$ will not have this safety margin, and may be more likely to inadvertently create a block of 0s or 1s that will lead it to a wrong answer. A final feature that contributes to the difficulty of finding good rules with $\lambda = 1/2$ is the combinatorially large number of rules there. In effect, the search space is much larger, which makes the global search more difficult. Locally, about a given adequate rule at $\lambda = 1/2$, there are many rules close in Hamming distance, and thus reachable via mutation, that are not markedly better than the given rule.

Once the more successful versions of the epoch-3 strategies are discovered in epoch 4, their variants spread in the population, and the most successful rules have λ on the low or high side of $\lambda = 1/2$. This explains the shift from the clustering around $\lambda = 1/2$, as seen in generations 10–30 in Figure 9, to a two-peaked distribution that becomes clear around generation 65. The rules in each run cluster around one or the other peak, specializing in one or the other way. We believe this type of symmetry breaking may be a key mechanism that determines much of the population dynamics and the GA's success—or lack thereof—in optimization.

How does the preceding analysis of the symmetry breaking jibe with the argument, given earlier, that the best rules for the $\rho_c = 1/2$ task must be

close to $\lambda = 1/2$? None of the rules found by the GA had a fitness as high as 0.98—the fitness of the GKL rule, whose λ is exactly $1/2$. That is, the evolved rules make significantly more classification errors than the GKL rule and, as will be seen below, the measured fitness of the best evolved rules is much worse on larger lattice sizes, whereas the GKL rule's fitness increases with increasing lattice sizes. To obtain the fitness of the GKL rule a number of careful balances in the rule table must be achieved. This is evidently very hard for the GA to do, especially in light of the symmetries in the task and their suboptimal breaking by the GA.

7.5 Performance of the evolved rules

Recall that the proportional fitness of a rule is the fraction of correct cell states at the final time step, averaged over 300 initial configurations. This calculation of fitness gives a rule partial credit for getting some final cell states correct. However, the actual task is to relax to either all 1s or all 0s, depending on the initial configuration. In order to measure how well the evolved rules actually perform the task, we define the *performance* of a rule to be the fraction of times the rule correctly classifies initial configurations, averaged over a large number of initial configurations. In this case, credit is given only if the initial configuration relaxes to exactly the correct fixed point after some number of time steps. We measured the performance of each of the elite rules in the final generations of the 30 runs, by testing each rule on 300 randomly generated initial configurations that were uniformly distributed in the interval $0 \leq \rho \leq 1$, and letting the rule iterate on each initial condition for 1000 time steps. Figure 16 displays the mean performance (diamonds) and best performance (squares) in each λ bin. This figure shows that while the mean performances in each bin are much lower than the mean fitnesses for the elite rules shown in Figure 6, the best performance in each bin is roughly the same as the best fitness in that bin. (In some cases the best performance in a bin is slightly higher than the best fitness shown in Figure 6. This is because different sets of 300 initial conditions were used to calculate fitness and performance. This difference can produce small variations in the fitness or performance values.) The best performance we measured was ≈ 0.95. Under this measure the performance of the GKL rule is ≈ 0.98. Thus the GA never discovered a rule that performed as well as the GKL rule, even up to 300 generations. In addition, when we measure the performance of the fittest evolved rules on larger lattice sizes, their performances decrease significantly, while that of the GKL rule remains roughly the same.

7.6 Using performance as the fitness criterion

Can the GA evolve better-performing rules on this task? To find out, we conducted an additional experiment in which performance (as defined in the previous section) was the fitness criterion. As in the previous experiments, at each generation each rule was tested on 300 initial configurations that were uniformly distributed over density values. However, for this experiment, a

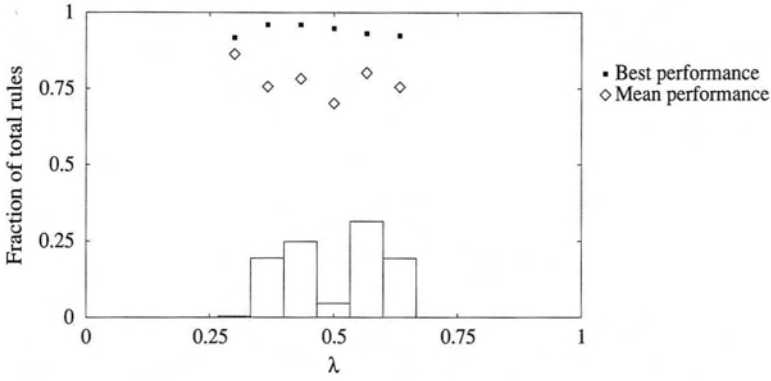

Figure 16: Performances of the final generation elite rules (merged from the 30 runs using the proportional fitness function). The mean and best performances in each bin are plotted on the same histogram as that in Figure 6.

rule's fitness was defined as the fraction of initial configurations that were correctly classified. An initial configuration was considered to be incorrectly classified if any bits in the final lattice were incorrect. Aside from this modified fitness function, the GA remained the same as in the proportional-fitness experiments. We performed 30 runs of the GA for 100 generations each. The results are given in Figure 17, which gives a histogram plotting the frequencies of the elite rules from generation 100 of all 30 runs, as a function of λ. The shape of the histogram again has two peaks centered around a dip at $\lambda = 1/2$. This shape results from the same symmetry-breaking effect that occurred in the proportional-fitness case; these runs evolved essentially the same strategies as the epoch-3 strategies described previously. The best performances found were ≈ 0.95; these are comparable to the best performances in the proportional-fitness case.

The performance of one of the best rules evolved with performance fitness is plotted as a function of $\rho(0)$ in Figure 18, for lattice sizes of 149 (the lattice size used for testing the rules in the GA runs), 599, and 999. This rule has $\lambda \approx 0.54$, and its strategy is similar to that shown in Figure 15: it increases sufficiently large blocks of adjacent or nearly adjacent 0s. We used the same procedure to make these plots as described for Figure 3. The performance, according to this measure, is significantly worse than that of the GKL rule (see Figure 3), especially on larger lattice sizes. The worst performances for the larger lattice sizes are centered slightly above $\rho = 1/2$. On such initial configurations the CA should relax to a fixed point of all 1s, but more detailed inspection of these results revealed that on almost every initial configuration with ρ slightly above $1/2$, the CA relaxed to a fixed point of all 0s. This is a result of the rule's strategy of increasing "sufficiently large" blocks of 0s: the appropriate size to increase was evolved for a lattice with $N = 149$. With larger lattices, the probability of such blocks in initial configurations

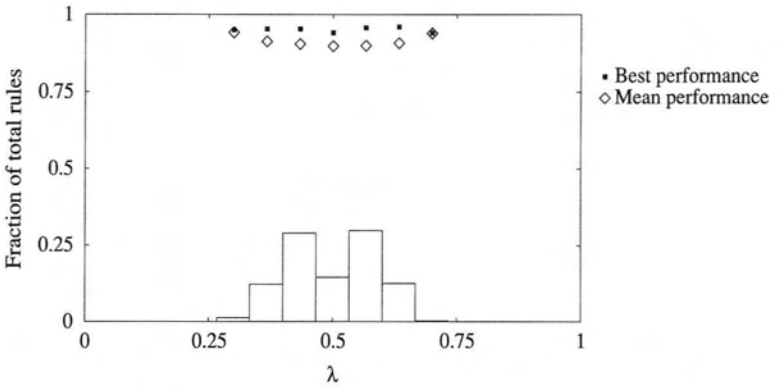

Figure 17: Results from our experiment with performance as the fitness criterion. The histogram plots the frequencies of elite rules merged from the final generations (generation 100) of 30 runs in which the performance-fitness function was used.

with $\rho > 1/2$ increases, and the closer the ρ of such initial conditions to $1/2$, the more likely such blocks are to occur. In the CA we tested with $N = 599$ and $N = 999$, such blocks occured in most initial configurations with ρ slightly above $1/2$, and these initial conditions were always classified incorrectly. 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 lattice-size variation during evolution in an attempt to prevent such overfitting.

7.7 Adding a diversity-enforcement mechanism

Our description of the four epochs in the GA's search explains the results of our experiment, but it does not explain the difference between our results and those of Packard's experiment reported in [24]. One difference between our GA and the original was the inclusion in the original of a diversity-enforcement scheme that penalized newly formed rules that were too similar in Hamming distance to existing rules in the population. To test the effect of such a scheme on our results, we included a similar scheme in one set of experiments. In our scheme, every time a new string is created through crossover and mutation, the average Hamming distance between the new string and the elite strings—the 50 strings that are copied unchanged—is measured. If this average distance is less than 30% of the string length (here 38 bits), then the new string is not allowed in the new population. New strings continue to be created through crossover and mutation until 50 new strings have met this diversity criterion. We note that many other diversity-enforcement schemes have been developed in the GA literature; one example is "crowding" [9].

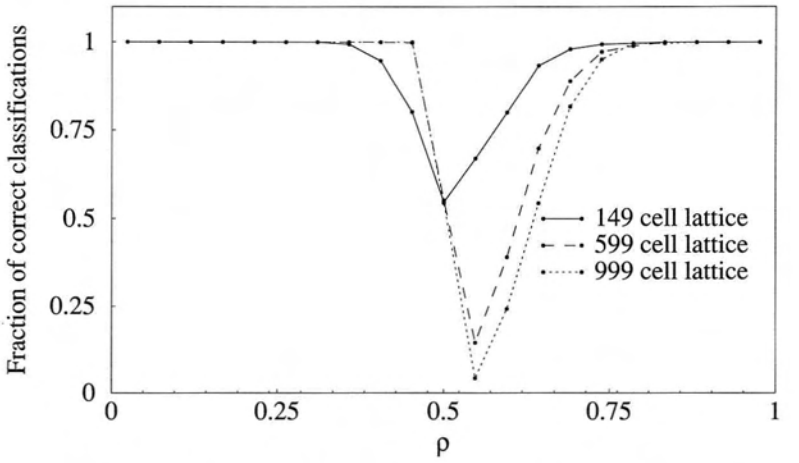

Figure 18: Performance of one of the best rules evolved using performance fitness, plotted as a function of $\rho(0)$. Performance plots are given for three lattice sizes: 149 (the size of the lattice used in the GA runs), 599, and 999. This rule has $\lambda \approx 0.54$.

The results of this experiment are given in Figure 19. The histogram in that figure represents the merged rules from the entire population at generation 100 of 20 runs of the GA, using the proportional-fitness function and our diversity-enforcement scheme. The histogram in this figure is very similar to that in Figure 5(b). The only major difference is the significantly lower mean fitness in the middle and leftmost bins, which results from the increased requirement for diversity in the final non-elite population. We conclude that the use of a similar diversity-enforcement scheme was not the factor responsible for the difference between the results in [24] and our results.

7.8 Differences between our results and the original experiment

As shown in Figure 5(b), our results are strikingly different from those reported in [24]. These experimental results, along with the theoretical argument that the most successful rules for this task should have λ close to $1/2$, lead us to conclude that Packard's interpretation of his results (as giving evidence for the two hypotheses concerning evolution, computation, and λ) is not correct. However, we do not know what accounts for the differences between our results and those obtained in the original experiment. We speculate that the differences are due to additional mechanisms in the GA used in Packard's experiment, which were not reported in [24]. For example, the original experiment included a number of additional sources of randomness, such as the regular injection of new random rules at various λ values and a much higher mutation rate than that in our experiment [23]. These sources of randomness may have slowed the GA's search for high-fitness rules, and

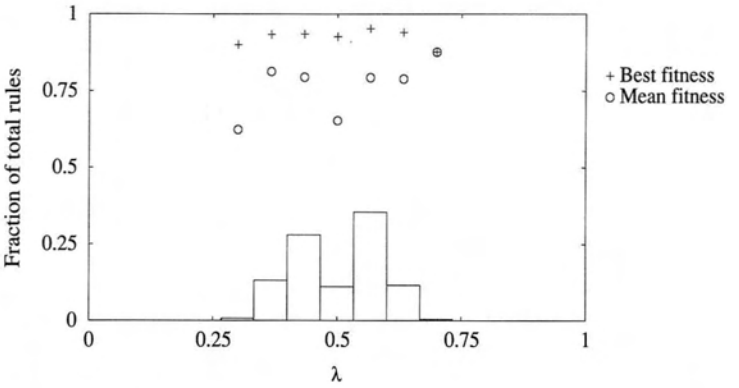

Figure 19: Results from our experiment in which a diversity-enforcement mechanism was added to the GA. The histogram plots the frequencies of rules merged from the entire population at generation 100 of 20 runs with the diversity-enforcement scheme.

prevented it from converging on rules close to $\lambda = 1/2$. Our experimental results and theoretical analysis give strong reason to believe that the clustering close to λ_c seen in Figure 4 is an artifact of mechanisms in the particular GA that was used, rather than a result of any computational advantage conferred by the λ_c regions.

Although the results were very different, there is one qualitative similarity: the rule-frequency-versus-λ histograms in both cases contained two peaks separated by a dip in the center. As we have noted, the two peaks in our histogram were closer to $\lambda = 1/2$ by a factor of 4, but it is possible that Packard's original results were due to a mechanism similar to either the epoch-1 sensitivity to initial configuration and population asymmetry about $\lambda = 1/2$, or the symmetry breaking we observed in epoch 3. Perhaps these were combined with additional forces, such as additional sources of randomness, that kept rules far away from $\lambda = 1/2$. Unfortunately, the best and mean fitnesses for the λ bins were not reported in [24]. As a consequence we do not know whether the peaks in the original histogram contained high-fitness rules, or even if they contained rules that were more fit than rules in other bins. Our results, and the basic symmetry in the problem, suggest that they did not.

8. General discussion

8.1 What we have shown

The results reported in this paper have demonstrated that Packard's results are not reproduced by our experiments. We conclude that the original experiment does not give firm evidence for the hypotheses it was meant to test: first, that rules capable of performing complex computation are most likely

to be found close to λ_c values; second, that when CA rules are evolved by a GA to perform a nontrivial computation, evolution will tend to select rules close to λ_c values.

As we argued theoretically, and as our experimental results suggest, the most successful rules for performing a given ρ-classification task will be close to a particular value of λ that depends on the particular ρ_c of the task. Thus, for this class of computational tasks, the λ_c values associated with an "edge of chaos" are not correlated with the ability of rules to perform the task.

The results that we have presented do not disprove the hypothesis that computational capability can be correlated with phase transitions in CA rule space. Individual CAs have been known for some time to exhibit phase transitions with the requisite divergence of correlation length required for infinite memory capacity [2]. Indeed, a correlation between computational capability and phase transitions has been noted for other dynamical systems. In the context of continuous-state dynamical systems, it has been shown that there is a direct relationship between the intrinsic computational capability of a process and the degree of randomness of that process at the phase transition from order to chaos. Computational capability was quantified in terms of the statistical complexity, a measure of the amount of memory of a process, and via the detection of an embedded computational mechanism equivalent to a stack automaton [4, 5]. More generally, the computational capacity of evolving systems may very well require dynamical properties characteristic of phase transitions, if they are to increase their complexity. We have shown only that the published experimental support cited for hypotheses relating λ_c and computational capability in CA was not reproduced.

In the remainder of this section, we step back from particular experiments and discuss in more general terms the ideas that motivated these studies.

8.2 λ, dynamical behavior, and computation

As noted in section 4, Langton presented evidence that, given certain caveats regarding the radius r and number of states k, there is some correlation between λ and the behavior of an "average" CA on an "average" initial configuration [17]. Behavior was characterized in terms of such quantities as single-site entropy, two-site mutual information, difference-pattern spreading rate, and average transient length. The correlation is quite strong for very low and very high λ values, which predict fixed-point or short-period behavior. However, for intermediate λ values, there is a large degree of variation in behavior. Moreover, there is no precise correlation between these λ values and the location of a behavioral "phase transition," other than that described by Wootters and Langton in the limit of infinite k [33].

These remarks, and the experimental results in [17], are concerned with the relationship between λ and the dynamical behavior of CAs—they do not directly address the relationship between λ and computational capability of CAs. The basic hypothesis was that λ correlates with computational capability, in the sense that rules capable of complex (and in particular,

universal) computation must be, or at least are most likely to be, found near λ_c values. As far as CAs are concerned, the hypothesis was based on the intuition that complex computation cannot be supported in the short-period or chaotic regimes because phenomena such as long transients and long space-time correlation, necessary to support complex computation, apparently occur in "complex" (nonperiodic, nonchaotic) regimes only. Thus far, there has been no experimental evidence correlating λ with an independent measure of computation. Packard's experiment was intended to address this issue, as it involved an independent measure of computation—performance on a particular complex computational task—but, as we have shown, it did not provide evidence for the hypothesis linking λ_c values with computational ability.

One problem is that these hypotheses have not been rigorously formulated. If the hypotheses put forth in [17] and [24] are interpreted to mean that *any* rule performing complex computation (as exemplified by the $\rho = 1/2$ task) must be close to λ_c, then we have shown them to be false with our argument that correct performance on the $\rho = 1/2$ task requires $\lambda = 1/2$. If the hypotheses are concerned instead with generic, statistical properties of CA rule space—the "average" behavior of an "average" CA at a given λ—then the notion of "average behavior" must be better defined. Additionally, more appropriate measures of dynamical behavior and computational capability must be formulated, and the notion of the "edge of chaos" must also be well defined.

The argument that complex computation cannot occur in chaotic regimes may seem intuitively correct, but there is a theoretical framework and strong experimental evidence to the contrary. Hanson and Crutchfield [3, 12] have developed a method for filtering out chaotic "domains" in the space-time diagram of a CA, sometimes revealing "particles" that have the nonperiodic, nonchaotic properties of structures in Wolfram's Class 4 CA. In other words, with the application of the appropriate filter, complex structures can be uncovered in a space-time diagram that, to the human eye (and to the statistics used in [17] and [24]) appears to be completely random. As an extreme example, it is conceivable that such filters could be applied to a seemingly chaotic CA and reveal that the CA is actually implementing a universal computer (with glider guns implementing AND, OR, and NOT gates, and so on). Hanson and Crutchfield's results strikingly illustrate the fact that apparent complexity of behavior—and apparent computational capability—can depend on the implicit "filter" imposed by one's chosen statistics.

8.3 What kind of computation in CA do we care about?

In the previous section, the phrases "complex computation" and "computational capability" were used somewhat loosely. As was discussed in section 3, there are at least three different interpretations of the notion of computation in CAs. The notion of a CA being able to perform a "complex computation" such as the $\rho_c = 1/2$ task, where the CA performs the same computation

on all initial configurations, is very different from the notion of a CA being capable of simulating, under some special set of initial configurations, a universal computer. Langton's speculations regarding the relationship between dynamical behavior and computational capability seem to be more concerned with the latter than the former, though they imply that the capability to sustain long transients, long correlation lengths, and so on, is necessary for both notions of computation.

If "computationally capable" is taken to mean "capable, under some initial configurations, of universal computation," then one might ask why this is a particularly important property of CAs on which to focus attention. In [17], CAs were used as a vehicle to study the relationship between phase transitions and computation, with an emphasis on universal computation. But for those wishing to use CAs as scientific models or practical computational tools, a focus on the capacity for universal computation may be misguided. If a CA is being used as a model of a natural process (e.g., turbulence), then it is of limited interest to know whether or not the CA is, in principle, capable of universal computation (especially if universal computation will arise only under some specially engineered initial configurations that the natural process is extremely unlikely ever to encounter). To understand emergent computation in natural phenomena as modeled by CAs, one should try to understand what computation the CA does "intrinsically" [3, 12], rather than what it is capable of doing "in principle" (and only under some very special initial configurations). Thus, understanding the conditions under which a capacity for universal computation is possible will not be of much value in understanding the natural systems modeled by CAs.

This general point is neither new nor deep. Analogous arguments have been put forward in the context of neural networks, for example. While many constructions of universal computation in neural networks have been made (e.g., [29]), some psychologists (e.g., [28]) have argued that this has little to do with understanding how brains or minds work in the natural world.

Similarly, if one wishes to use a CA as a parallel computer for solving a real problem—such as face recognition—it would be very inefficient, if not practically impossible, to solve the problem by (say) programming Conway's Game of Life CA to be a universal computer that simulates the action of the desired face recognizer. Thus, understanding the conditions under which universal computation is possible in CAs is not of much practical value either.

In addition, it is not clear that anything like a drive toward universal computational capabilities is an important force in the evolution of biological organisms. It seems likely that substantially less computationally-capable properties play a more frequent and robust role. Thus, asking under what conditions evolution will create entities (including CAs) that are capable of universal computation may not be of great importance in understanding natural evolutionary mechanisms.

In short, it is mathematically important to know that some CAs are, in principle, capable of universal computation. But we argue that this is by no

means the most scientifically interesting property of CAs. More to the point, this property does not help scientists much in understanding the emergence of complexity in nature, or in harnessing the computational capabilities of CAs to solve real problems.

9. Conclusion

The main purpose of this study was to examine and clarify the evidence for various hypotheses related to evolution, dynamics, and the computational capability of cellular automata. As a result of our study we have identified a number of evolutionary mechanisms, such as the role of combinatorial drift, and the role of symmetry. We have also found that the breaking of the goal task's symmetries in the early generations can be an impediment to further optimization of individuals in the population. Symmetry breaking results in a kind of suboptimal speciation in a population that is stable (or at least metastable) over long times. The symmetry-breaking effects we have described may be similar to symmetry-breaking phenomena that emerge in biological evolution, such as brain hemispheric dominance and handedness, or the breaking of the spherical symmetry of a blastula which results in bilateral symmetry. It is our goal to develop a more rigorous framework for understanding these mechanisms in the context of evolving CAs. We believe that a deep understanding of these mechanisms in this relatively simple context can yield insights for understanding evolutionary processes in general, and for successfully applying evolutionary computation methods to complex problems.

Though our experiments did not reproduce the results reported in [24], we believe that Packard's original strategy of using GAs to evolve computation in CAs is an important idea. In addition to its potential for the study of various theoretical issues, it has a practical potential that could be significant. As previously mentioned, CAs are increasingly being studied as a class of efficient parallel computers; the main bottleneck in applying CAs more widely to parallel computation is *programming*—in general, it is very difficult to program CAs to perform complex tasks. Our results suggest that the GA has promise as a method for accomplishing such programming automatically. In order to test further the GA's effectiveness when compared with other search methods, we performed an additional experiment, comparing the performance of our GA on the $\rho_c = 1/2$ task with the performance of a simple steepest-ascent hill-climbing method. We found that the GA significantly outperformed hill climbing, reaching much higher fitnesses for an equivalent number of fitness evaluations. This gives some evidence for the relative effectiveness of GAs when compared with simple gradient ascent methods for programming CAs. Koza [16] has also evolved CA rules using a very different type of representation scheme; the relationship between representation and GA success on such tasks is a topic of substantial practical interest.

Acknowlegments

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. Thanks to Doyne Farmer, Jim Hanson, Erica Jen, Chris Langton, Wentian Li, Cris Moore, and Norman Packard for many helpful discussions and suggestions concerning this project. Thanks also to Emily Dickinson and Terry Jones for technical advice.

References

[1] E. Berlekamp, J. H. Conway, and R. Guy, *Winning Ways for Your Mathematical Plays* (New York, Academic Press, 1982).

[2] M. Creutz, "Deterministic Ising Dynamics," *Annals of Physics*, **167** (1986) 62.

[3] J. P. Crutchfield and J. E. Hanson, "Turbulent Pattern Bases for Cellular Automata," Technical Report 93-03-010 (1993), Santa Fe Institute; *Physica D*, in press.

[4] J. P. Crutchfield and K. Young, "Inferring Statistical Complexity," *Physical Review Letters*, **63** (1989) 105.

[5] J. P. Crutchfield and K. Young, "Computation at the Onset of Chaos," in *Complexity, Entropy, and the Physics of Information*, edited by W. H. Zurek (Redwood City, CA, Addison-Wesley, 1990).

[6] P. Gonzaga de Sá and C. Maes, "The Gacs-Kurdyumov-Levin Automaton Revisited," *Journal of Statistical Physics*, **67(3/4)** (1992) 507–522.

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

[8] P. Gacs, G. L. Kurdyumov, and L. A. Levin, "One-dimensional Uniform Arrays That Wash Out Finite Islands," *Problemy Peredachi Informatsii*, **14** (1978) 92–98.

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

[10] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (New York, Springer-Verlag, 1983).

[11] H. A. Gutowitz (editor), *Cellular Automata* (Cambridge, MIT Press, 1990).

[12] J. E. Hanson and J. P. Crutchfield, "The Attractor-Basin Portrait of a Cellular Automaton," *Journal of Statistical Physics*, **66(5/6)** (1992) 1415–1462.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems* (Ann Arbor, University of Michigan Press, 1975).

[14] S. A. Kauffman, "Requirements for Evolvability in Complex Systems: Orderly Dynamics and Frozen Components," *Physica D*, **42** (1990) 135–152.

[15] S. A. Kauffman and S. Johnson, "Co-evolution to the Edge of Chaos: Coupled Fitness Landscapes, Poised States, and Co-evolutionary Avalanches," in *Artificial Life II*, edited by C. G. Langton et al. (Redwood City, CA, Addison-Wesley, 1992).

[16] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Cambridge, MIT Press, 1993).

[17] C. G. Langton, "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation," *Physica D*, **42** (1990) 12–37.

[18] W. Li, "Non-local Cellular Automata," in *1991 Lectures in Complex Systems*, edited by L. Nadel and D. Stein (Redwood City, Addison-Wesley, 1992).

[19] W. Li and N. H. Packard, "The Structure of the Elementary Cellular Automata Rule Space," *Complex Systems*, **4** (1990) 281–297.

[20] W. Li, N. H. Packard, and C. G. Langton, "Transition Phenomena in Cellular Automata Rule Space," *Physica D*, **45** (1990) 77–94.

[21] K. Lindgren and M. G. Nordahl, "Universal Computation in a Simple One-dimensional Cellular Automaton," *Complex Systems*, **4** (1990) 299–318.

[22] N. Packard, "Complexity of Growing Patterns in Cellular Automata," in *Dynamical Behavior of Automata: Theory and Applications*, edited by J. Demongeot et al. (New York, Academic Press, 1984).

[23] N. H. Packard, personal communication.

[24] N. H. Packard, "Adaptation Toward the Edge of Chaos," in *Dynamic Patterns in Complex Systems*, edited by J. A. S. Kelso et al. (Singapore, World Scientific 1988).

[25] J. B. Pollack, "The Induction of Dynamical Recognizers," *Machine Learning*, **7** (1991) 227–252.

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

[27] A. Rosenfeld, "Parallel Image Processing Using Cellular Arrays," *Computer*, **16** (1983) 14.

[28] D. E. Rumelhart and J. L. McClelland, "PDP Models and General Issues in Cognitive Science," in *Parallel Distributed Processing*, Vol. 1, edited by D. E. Rumelhart and J. L. McClelland (Cambridge, MIT Press, 1986).

[29] H. Siegelman and E. D. Sontag, "Neural Networks are Universal Computing Devices," Technical Report SYCON-91-08 (1991) (Rutgers Center for Systems and Control, Rutgers University, New Brunswick, NJ, 08903).

[30] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling* (Cambridge, MIT Press, 1987).

[31] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D*, **10** (1984) 1–35.

[32] S. Wolfram (editor), *Theory and Applications of Cellular Automata* (Singapore, World Scientific, 1986).

[33] W. K. Wootters and C. G. Langton, "Is There a Sharp Phase Transition for Deterministic Cellular Automata?" *Physica D*, **45** (1990) 95–104.

[34] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity (Reading, Addison-Wesley, 1992).