

Co-evolving Non-Uniform Cellular Automata to Perform Computations

Moshe Sipper
Logic Systems Laboratory
Swiss Federal Institute of Technology
IN-Ecublens
CH-1015 Lausanne, Switzerland
e-mail: Moshe.Sipper@di.epfl.ch

Abstract

A major impediment of cellular automata (CA) stems from the difficulty of utilizing their complex behavior to perform useful computations. Recent studies by [Packard, 1988, Mitchell *et al.*, 1994b] have shown that CAs can be evolved to perform a computational task. In this paper *non-uniform* CAs are studied, where each cell may contain a *different* rule, in contrast to the original, uniform model. We describe experiments in which non-uniform CAs are *evolved* to perform the computational task using a *local, co-evolutionary* algorithm. For radius $r = 3$ we attain peak performance values of 0.92 comparable to those obtained for uniform CAs (0.93 – 0.95). This is notable considering the huge search spaces involved, much larger than the uniform case. Smaller radius CAs (previously unstudied in this context) attain performance values of 0.93 – 0.94. For $r = 1$ this is considerably higher than the maximal *possible* uniform CA performance of 0.83, suggesting that non-uniformity reduces connectivity requirements. We thus demonstrate that: (1) non-uniform CAs can attain high computational performance, and (2) such systems can be evolved rather than designed.

1 Introduction

Cellular automata (CA) are dynamical systems in which space and time are discrete. The states of cells in a regular grid are updated synchronously according to a local interaction rule. Each cell obeys the same rule and can be in one of k discrete states [Toffoli and Margolus, 1987].

The model was originally conceived by Ulam and von Neumann in the 1950's to provide a framework for studying the behavior of complex, extended systems [von Neumann, 1966]. It has been applied to the study of general phenomenological aspects of the world, including: communication, computation, construction, growth, reproduction, competition

and evolution [Toffoli and Margolus, 1987, Langton, 1984, Burks, 1970, Smith, 1969]. CAs also provide a means for modeling physical phenomena by reducing them to their basic, elemental laws (rules) [Toffoli, 1980, Fredkin and Toffoli, 1982, Margolus, 1984, Vichniac, 1984, Bennett and Grinstein, 1985].

CAs exhibit three notable features: massive parallelism, locality of cellular interactions, and simplicity of basic components (cells). They perform computations in a distributed fashion on a spatially extended grid. As such they differ from the standard approach to parallel computation in which a problem is split into independent sub-problems, each solved by a different processor, later to be combined in order to yield the final solution. CAs suggest a new approach in which complex behavior arises in a bottom-up manner from non-linear, spatially extended, local interactions [Mitchell *et al.*, 1994b].

A major impediment of CAs stems from the difficulty of utilizing their complex behavior to perform useful computations. As noted by [Mitchell *et al.*, 1994b], the difficulty of designing CAs to have a specific behavior or perform a particular task has severely limited their applications; automating the design (programming) process would greatly enhance the viability of CAs.

In this paper we describe experiments in which CAs are evolved to perform a particular computational task. Our work derives from that of [Packard, 1988, Mitchell *et al.*, 1993, Mitchell *et al.*, 1994b, Mitchell *et al.*, 1994a, Das *et al.*, 1994, Das *et al.*, 1995] in which a genetic algorithm (GA) is used to evolve CAs. The goals of their research are [Mitchell *et al.*, 1994b]: (1) to enhance our understanding of the ways CAs perform computations, (2) to learn how CAs may be evolved, rather than designed, to perform computational tasks, (3) to understand how evolution creates complex, global behavior in a locally interconnected system of simple parts.

We wish to study these issues in *non-uniform CAs*, in which each cell may contain a *different* rule. The relaxation of the uniformity restriction is the only difference of this model from the original one. The operation of non-uniform CAs is identical to that of uniform CAs and the features noted above concerning the original model apply here as well¹.

Non-uniform CAs have been investigated by [Vichniac *et al.*, 1986] who discuss a one-dimensional CA in which a cell probabilistically selects one of two rules, at each time step. They showed that complex patterns appear, characteristic of class IV behavior² (see also [Hartman and Vichniac, 1986]). The issue of evolving non-uniform CAs was discussed by us in [Sipper, 1994, Sipper, 1995b], where complex phenomena were observed, though we did not evolve CAs to perform a particular computational task. In [Sipper, 1995a] we examined the issue of universal computation in non-uniform CAs (see Section 5).

A prime motivation for studying non-uniform CAs stems from the observation that the uniform model is essentially “programmed” at an extremely low-level [Rasmussen *et al.*, 1992]. A *single* rule is sought that must be universally applied to all cells in the grid,

¹Note that from a hardware point of view the resources required by non-uniform CAs are identical to those of uniform ones since a cell in both cases contains a rule (albeit not necessarily the same one in our case).

²CA class numbers are those introduced by [Wolfram, 1984].

a task which may be arduous even for evolutionary approaches. For non-uniform CAs search space sizes are vastly larger than for uniform CAs, a fact which initially seems as an impediment. However, as we shall see, increased search space size in fact engenders new evolutionary paths, leading to high performance systems. Our evolutionary algorithm is different than the standard GA, where a population of *independent* problem solutions (in our case candidate rules) *globally* evolves. Rather, our approach is one in which a grid of rules *locally co-evolves* to perform a given task (see Section 3). Our results demonstrate that non-uniform CAs can be co-evolved to exhibit high computational performance.

The non-uniform CAs discussed in this paper are one-dimensional with $k = 2$, i.e., two possible states per cell $(0, 1)$. In such CAs the neighborhood of a cell includes itself and r cells on both sides, where r is referred to as the *radius*. Spatially periodic boundary conditions are used, resulting in a circular grid. A common method of examining the behavior of one-dimensional CAs is to display a two-dimensional space-time diagram, where the horizontal axis depicts the configuration at a certain time t and the vertical axis depicts successive time steps³. An example is given in Figure 1.

The paper is organized as follows: Section 2 presents the particular computational task we investigate and details of previous experiments. Section 3 delineates our co-evolutionary algorithm. Results using non-uniform CAs with radius $r = 3$ are discussed in Section 4 and smaller radius CAs ($r = 1, 2$) are analyzed in Section 5. Finally, our conclusions are presented in Section 6.

2 The computational task and previous experiments

The experiments carried out in [Packard, 1988, Mitchell *et al.*, 1993, Mitchell *et al.*, 1994b, Mitchell *et al.*, 1994a] involve the *evolution* of *uniform*, one-dimensional CAs with $k = 2$ and $r = 3$. The computational task is to decide whether or not the initial configuration contains more than 50% 1s. Following Mitchell *et al.*, let ρ denote the density of 1s in a grid configuration, $\rho(t)$ the density at time t , and ρ_c the threshold density for classification (in our case 0.5). The desired behavior (i.e., the result of the computation) is for the CA to relax to a fixed-point pattern of all 1s if $\rho(0) > \rho_c$ and all 0s if $\rho(0) < \rho_c$. If $\rho(0) = \rho_c$ the desired behavior is undefined.

As noted by Mitchell *et al.*, the $\rho_c = 0.5$ density task is a nontrivial computation for a small radius CA ($r \ll N$, where N is the grid size). The density is a global property of a configuration whereas a small-radius CA relies solely on local interactions. The minimum amount of memory required for the task is $O(\log N)$ using a serial scan algorithm. Thus, the computation involved corresponds to recognition of a non-regular language. Since the 1s can be distributed throughout the grid, propagation of information must occur over large distances (i.e., $O(N)$).

A $k = 2$, $r = 3$ rule which purportedly performs this task was discussed by [Packard, 1988]. This is the Gacs-Kurdyumov-Levin (GKL) rule defined as follows [Gacs *et al.*, 1978,

³Throughout this paper the term ‘state’ refers to the value of a single cell; ‘configuration’ refers to an assignment of 1 states to several cells, and 0s otherwise.

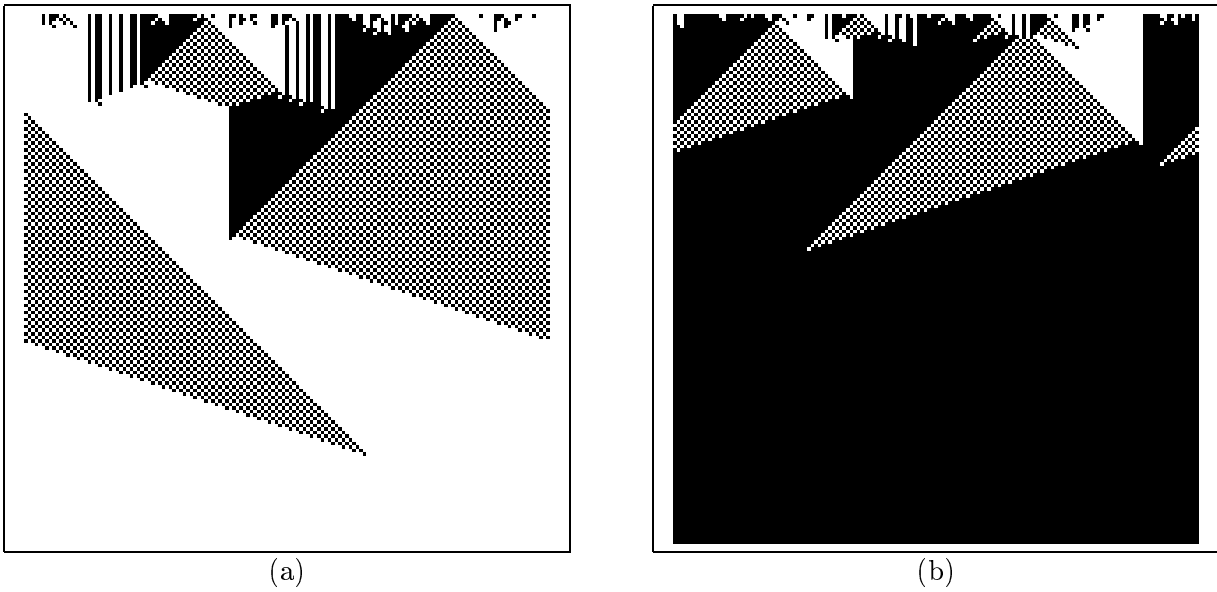


Figure 1: Space-time diagrams of the binary state GKL rule. CA size is $N = 149$. White squares represent cells in state 0, black squares represent cells in state 1. The pattern of configurations is shown through time (which increases down the page). (a) Initial density of 1s is $\rho(0) \approx 0.47$ and final density at time $t = 150$ is $\rho(150) = 0$. (b) Initial density of 1s is $\rho(0) \approx 0.53$ and final density at time $t = 150$ is $\rho(150) = 1$. The CA relaxes in both cases to a fixed pattern of all 0s or all 1s, correctly classifying the initial configuration.

Gonzaga de Sá and Maes, 1992]:

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

where $s_i(t)$ is the state of cell i at time t .

Figure 1 depicts the behavior of the GKL rule on two initial configurations, $\rho(0) < \rho_c$ and $\rho(0) > \rho_c$. We observe that propagation of information about local neighborhoods takes place to produce the final fixed-point configuration. Essentially, the rule’s strategy is to successively classify local densities with the locality range increasing over time. In regions of ambiguity a “signal” is propagated, seen either as a checkerboard pattern in space-time or as a vertical white-to-black boundary [Mitchell *et al.*, 1993].

In the experiments carried out by [Packard, 1988] and Mitchell *et al.*, a genetic algorithm was used to evolve uniform CAs to perform the $\rho_c = 0.5$ density task. We describe below the work reported in [Mitchell *et al.*, 1993]⁴. The GA uses a randomly generated initial population of CA rules with $k = 2$, $r = 3$. Each rule is represented by a bit string containing the output bits of the rule table, i.e., the bit at position 0 is the state to which neighborhood configuration 0000000 is mapped to and so on until bit 127 corresponding to neighborhood configuration 1111111. The bit string, also referred to as the genome or chromosome, is of size $2^{2r+1} = 128$, resulting in a huge search space of size 2^{128} . The grid size used was $N = 149$ with periodic boundary conditions. The population size was $P = 100$ and the GA was iterated for 100 generations.

Each rule was run for a maximum number of M iterations (time steps), where M is selected anew for each rule from a Poisson distribution with mean 320. A rule’s fitness is its average score, defined as the fraction of cell states correct at the last iteration, over 300 initial configurations. At each GA generation a new set of 300 configurations is generated at random, uniformly distributed over $\rho(0) \in [0.0, 1.0]$, half with $\rho < \rho_c$, half with $\rho > \rho_c$. All rules are tested on this set and the population of the next generation is created by copying the top half of the current population (ranked according to fitness) unmodified; the remaining half of the next generation population is created by applying the genetic operators of crossover and mutation on the rules of the current population [Goldberg, 1989].

Using the algorithm described above highly successful uniform CA rules were found [Mitchell *et al.*, 1993, Mitchell *et al.*, 1994b]. The best fitness values were in the range 0.93 – 0.95. Under the above fitness function the (designed) GKL rule has fitness ≈ 0.98 (a detailed analysis of this rule is provided in [Mitchell *et al.*, 1994b]). The GA never found a rule with fitness above 0.95. To date, we are not aware of any rule (designed or evolved) with a fitness of 1, and it is possible that no CA exists which performs the task perfectly for all N . However, the GKL rule’s error appears to decrease as $N \rightarrow \infty$ [Li, 1992].

Another result of Mitchell *et al.* concerns the λ parameter introduced by [Langton, 1990, Langton, 1992] in order to study the structure of the space of CA rules. The λ of a given CA rule is the fraction of non-quiescent output states in the rule table, where the quiescent

⁴Though the GA used by [Packard, 1988] is somewhat different this is not crucial for our current work.

state is arbitrarily chosen as one of the possible k states. For binary-state CAs the quiescent state is usually 0 and therefore λ equals the fraction of output 1 bits in the rule table.

In recent years it has been speculated that computational capability can be associated with phase transitions in CA rule space [Langton, 1990, Li *et al.*, 1990, Langton, 1992]. This phenomenon, generally referred to as the “edge of chaos”, asserts that dynamical systems are partitioned into ordered regimes of operation and chaotic ones with complex regimes arising on the edge between them. These complex regimes are hypothesized to give rise to computational capabilities. For CAs this means that there exist critical λ values at which phase transitions occur. It has been hypothesized that this phenomenon exists in other dynamical systems as well [Kauffman, 1993].

One of the main results of [Mitchell *et al.*, 1993] is that most of the rules evolved to perform the $\rho_c = 0.5$ task are clustered around $\lambda \approx 0.43$ or $\lambda \approx 0.57$. This is in contrast to [Packard, 1988] where most rules are clustered around $\lambda \approx 0.24$ or $\lambda \approx 0.83$, which correspond to λ_c values, i.e., critical values near the transition to chaos.

The results obtained by [Mitchell *et al.*, 1993] coupled with a theoretical argument given in their paper lead to the conclusion that the λ value of successful rules performing the $\rho_c = 0.5$ task is more likely to be close to 0.5, i.e., depends upon the ρ_c value. 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.

A study of non-uniform CA rule space has not been carried out to our knowledge. Furthermore, a new parameter must be defined since a non-uniform CA contains cells with different λ values. Nonetheless, we have obtained results which lend support to the conclusions of [Mitchell *et al.*, 1993] (see Section 4).

3 The co-evolutionary algorithm

In our experiments non-uniform CAs are used, i.e., CAs in which each cell may contain a different rule. As in the previous work (Section 2), a cell’s rule table is encoded as a bit string containing the output bits for all possible neighborhood configurations. Evolution, however, proceeds in a different manner.

Whereas the work described in Section 2 consists of a *population* of evolving, uniform CA rules our case involves a *single*, non-uniform CA (of size $N = 149$). Cell rules are initially generated at random, but constrained to be uniformly distributed among different λ values. Initial configurations are then generated at random, uniformly distributed over $\rho(0) \in [0.0, 1.0]$, half with $\rho > \rho_c$ and half with $\rho < \rho_c$. For each such configuration the CA is run for $M = 150$ iterations (thus, computation time is linear with grid size). Each cell’s fitness is accumulated over several runs of initial configurations, where a single run’s score is 1 if the cell is in the correct state after M iterations and 0 otherwise. Note that fitness is assigned per cell and not per CA since each cell maintains its own genome.

Fitness scores of each cell are accumulated over $C = 300$ configurations at which time evolution of rules occurs by applying crossover and mutation. This is done in a completely *local* manner. Let $nf_i(c)$ denote the number of fitter neighbors of cell i after c configurations.

Then the following procedure is carried out every $C = 300$ configurations:

- if $nf_i(c) = 0$ then rule i is left unchanged.
- if $nf_i(c) = 1$ then rule i is replaced by the fitter neighboring rule, followed by mutation.
- if $nf_i(c) = 2$ then rule i is replaced by the crossover of the two fitter neighboring rules, followed by mutation.
- if $nf_i(c) > 2$ then rule i is replaced by the crossover of two randomly chosen fitter neighboring rules, followed by mutation.

Crossover between two rules is performed by selecting at random (with uniform probability) a single crossover point and creating a new rule by combining the first rule's bit string before the crossover point with the second rule's bit string from this point onward. Mutation is applied to the bit string of a rule with probability 0.001 per bit. The total number of initial configurations per evolutionary run was between $2 \cdot 10^5 - 2 \cdot 10^6$. This number was $3 \cdot 10^6$ in [Mitchell *et al.*, 1993]⁵.

There are some notable differences between our algorithm and that used by [Packard, 1988, Mitchell *et al.*, 1993]. In their work a standard GA is used, employing a population of evolving, uniform CAs. All CA rules are ranked according to fitness, with crossover occurring between *any* two rules. Thus, while the CA runs in accordance with a local rule, evolution proceeds in a *global* manner. In contrast, evolution in our case proceeds *locally*; each cell has access only to its local neighborhood, not only during the run but also during the evolutionary phase, and no global fitness ranking is performed.

The standard GA, as used by [Packard, 1988, Mitchell *et al.*, 1993], involves a population of *independent* problem solutions. Each CA is run independently, after which genetic operators are applied to produce a new population. In contrast, our non-uniform CA *co-evolves* since each cell's fitness depends upon its evolving neighbors. Thus, we utilize a local, co-evolutionary algorithm in search of solutions for a computational task.

4 Results using radius $r = 3$

Our initial experiments were carried out with $r = 3$, i.e., a neighborhood of seven cells, as used by [Packard, 1988] and Mitchell *et al.* Performance values reported hereafter represent the average fitness of all grid cells after C configurations, normalized to the range $[0, 1]$. Before proceeding, we point out that this is somewhat different than the work of Mitchell *et al.*, who defined three measures: (1) performance- the number of correct classifications on a sample of initial configurations chosen from an unbiased distribution, binomially distributed over $\rho(0) \in [0.0, 1.0]$, (2) performance fitness- the number of correct classifications on a sample of C initial configurations chosen from a uniform distribution over $\rho(0) \in [0.0, 1.0]$ (no partial credit is given for partially correct final configurations), and (3) proportional fitness- the fraction of cell states correct at the last iteration, averaged over C initial configurations,

⁵100 generations \times 100 CA rules \times 300 configurations per rule.

uniformly distributed over $\rho(0) \in [0.0, 1.0]$ (partial credit is given)⁶. Our performance measure is analogous to the latter measure, however, there is an important difference: as our evolutionary algorithm is local, fitness values are computed for each individual cell; global fitness of the CA can then be *observed* by averaging these values over the entire grid.

Figure 2 displays the results of two typical runs. Each graph depicts the average fitness of all grid cells after C configurations (performance) and the best cell’s fitness, both as a function of the number of configurations run so far. Figure 2a displays a successful run in which the average fitness reaches a peak value of 0.92. Some runs were unsuccessful, i.e., average fitness did not rise above 0.5, the expected random value (Figure 2b). Observing Figure 2a, we note how a successful run comes about: at an early stage of evolution a high fitness rule is discovered as evident by the top curve depicting best fitness. Such a rule is sufficient in order to “drive” evolution in a direction of improved performance. The threshold fitness that must be attained by this rule is approximately 0.65; a lower value is insufficient to drive evolution “upwards”.

Peak fitness values represent an average accumulated over the preceding 300 configurations. This serves to define the performance of a particular makeup of rules, i.e., a particular non-uniform CA found by evolution. The CA with the highest performance in a run is saved and then tested on 10,000 initial configurations, generated as detailed in Section 3. A slight drop in fitness has been observed though the value is still high at approximately 0.91.

The success of our co-evolutionary algorithm in finding good solutions (i.e., high fitness, non-uniform CAs) is notable if one considers the search space involved: since each cell contains one of 2^{128} possible rules, and there are $N = 149$ such cells, our space is of size $(2^{128})^{149} = 2^{19072}$. This number is vastly larger than that involved in the uniform case. Nonetheless, using a local, co-evolutionary algorithm non-uniform CAs are discovered which exhibit high performance.

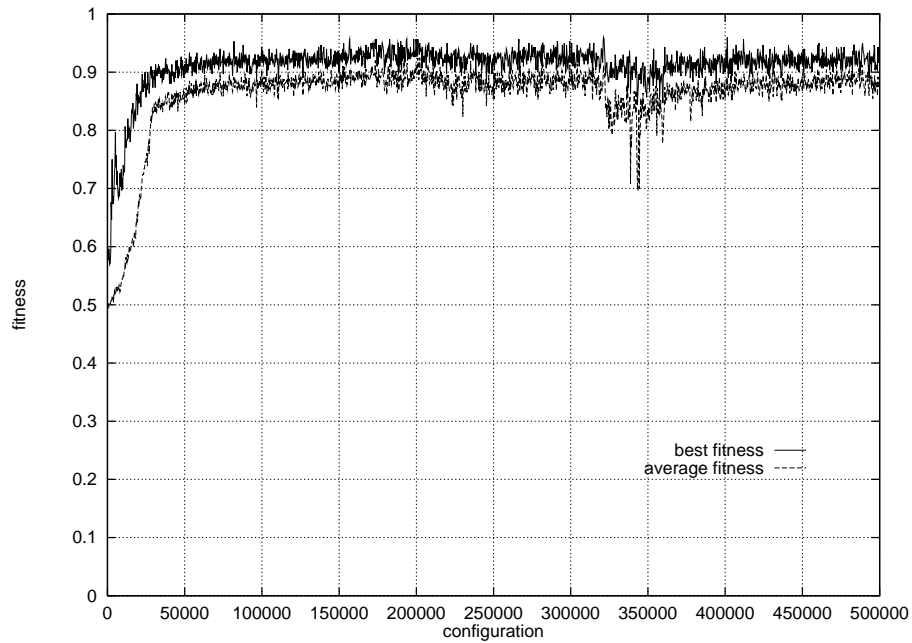
A histogram of the total number of rules as a function of λ is presented in Figure 3. It is evident that rules are clustered in the vicinity of $\lambda = 0.5$, mostly in the region 0.45 – 0.55. As noted in Section 2, a study of non-uniform CA rule space has not been carried out to our knowledge. Nonetheless, we feel that this result lends support to the conclusion of [Mitchell *et al.*, 1993] stating that λ values for the $\rho_c = 0.5$ task are likely to be close to 0.5, i.e., depending on the ρ_c value.

5 Using smaller radiuses

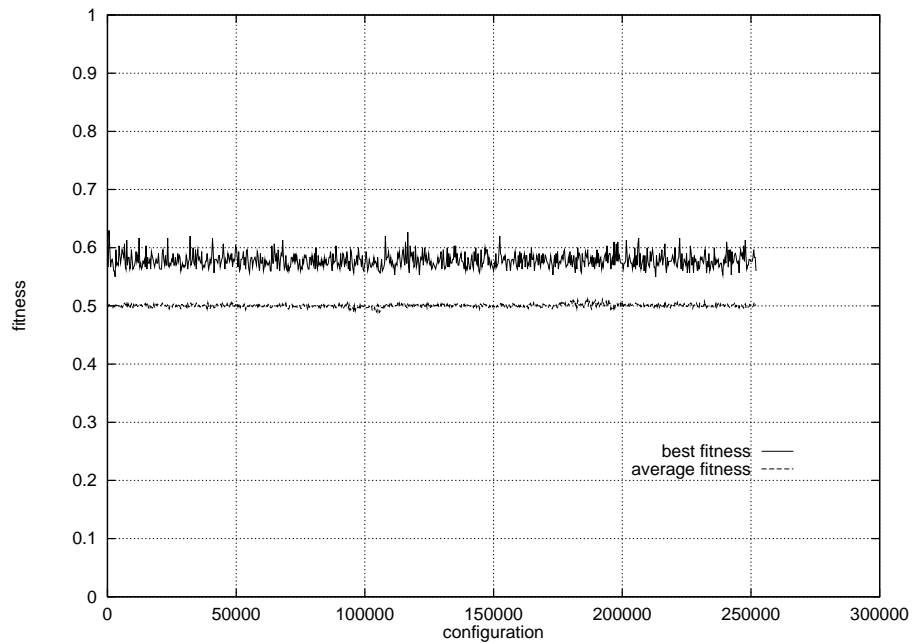
The work of [Packard, 1988] and Mitchell *et al.* concentrated solely on CAs with $r = 3$, i.e., seven neighbors per cell. In this section we study smaller radius, non-uniform CAs asking whether they can perform well on the $\rho_c = 0.5$ task, and whether such CAs can be co-evolved as in Section 4.

Toward this end we performed the above runs (as detailed in Section 3) using $r = 2$. Results were slightly better than the $r = 3$ case with peak performance values reaching as high as 0.94. We note in passing that the search space involved is of size $(2^{32})^{149} = 2^{4768}$.

⁶For a discussion of these measures the reader is referred to [Mitchell *et al.*, 1994b, Das *et al.*, 1994].



(a)



(b)

Figure 2: Results of two typical evolutionary runs with $r = 3$. (a) A successful run. (b) An unsuccessful run. In both graphs, the bottom curve depicts the average fitness of all grid cells (rules), and the top curve depicts the fitness of the best cell (rule) in the grid.

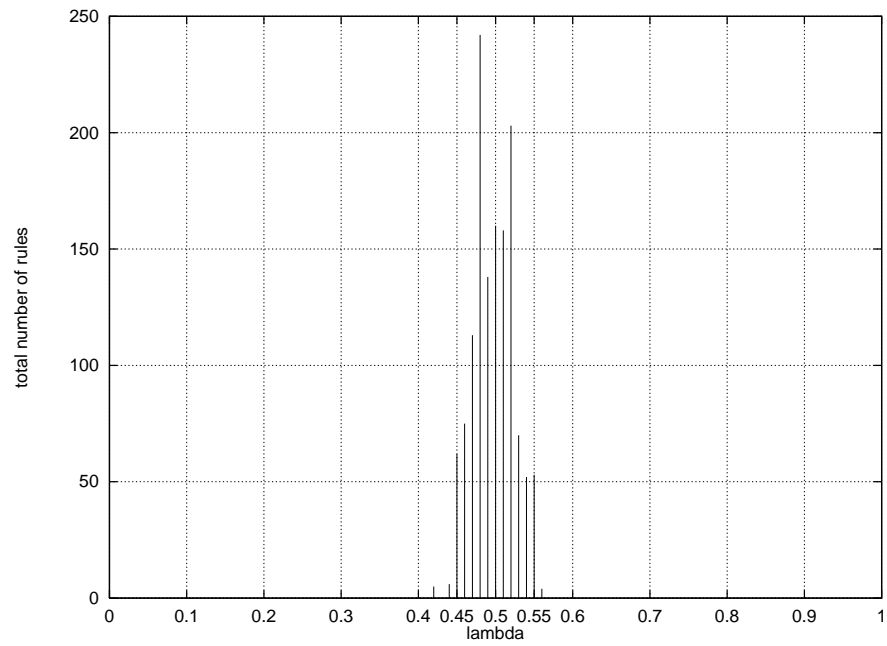


Figure 3: Histogram depicting the total number of rules (summed over several runs) as a function of λ .

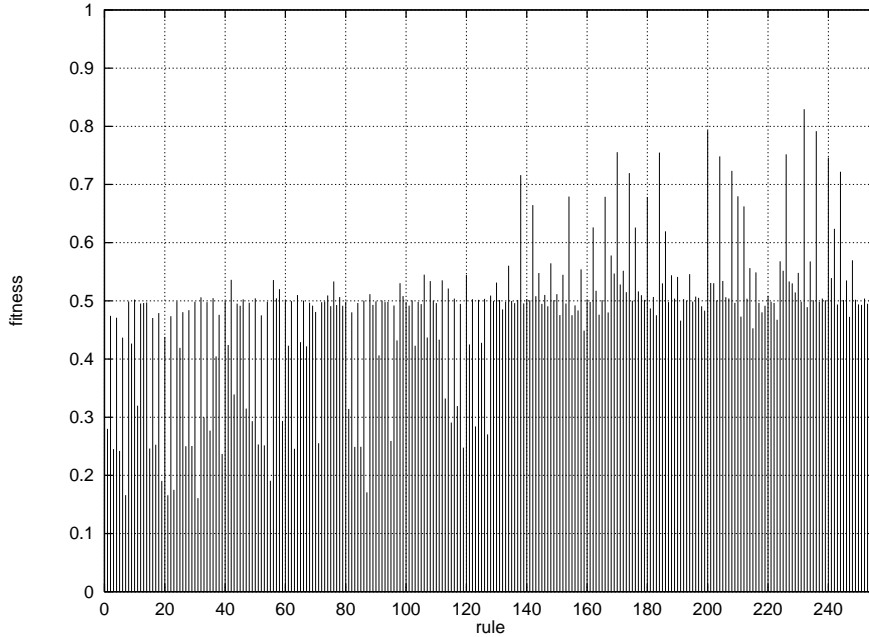


Figure 4: Fitness results of all possible uniform, $r = 1$ CA rules.

Can CAs of minimal radius ($r = 1$), in which each cell has access only to its own state and that of its two adjoining neighbors, attain high performance on the $\rho_c = 0.5$ task? Again it is noted that the search space in this case is still extremely large: $(2^8)^{149} = 2^{1192}$.

The size of *uniform*, $r = 1$ CA rule space is small, consisting of only $2^8 = 256$ rules. This enables us to test each and every one of these rules on the $\rho_c = 0.5$ task, a feat not possible for larger values of r . We performed several runs in which each rule was tested on 1000 configurations. Results are depicted in Figure 4 as average fitness verses rule number⁷.

The highest fitness value is 0.83 and was attained by rule 232 which performs a majority vote among the three neighborhood states. Thus, the maximal performance of uniform, $r = 1$ CAs is known and we now ask whether non-uniform CAs can attain higher performance, and whether such CAs can be discovered using our co-evolutionary algorithm.

Observing Figure 6, depicting the result of a typical run, we note that peak performance reaches a value of 0.93. Thus, we have demonstrated that a non-uniform CA with minimal radius can attain high performance, exceeding that of the best uniform CA; furthermore, the system is evolved rather than designed. The high performance of non-uniform, $r = 1$ CAs, similar to $r = 2, 3$ ones and notably higher than uniform, $r = 1$ CAs, suggests that

⁷Results of one run are displayed- no significant differences were detected among runs. Rule numbers are given in accordance with Wolfram's convention [Wolfram, 1983], representing the decimal equivalent of the binary number encoding the rule table (see Figure 5)

	Rule 224								Rule 226							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
neighborhood	111	110	101	100	011	010	001	000	111	110	101	100	011	010	001	000
output bit	1	1	1	0	0	0	0	0	1	1	1	0	0	0	1	0

	Rule 232								Rule 234							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
neighborhood	111	110	101	100	011	010	001	000	111	110	101	100	011	010	001	000
output bit	1	1	1	0	1	0	0	0	1	1	1	0	1	0	1	0

Figure 5: Rules involved in the density task.

non-uniformity reduces connectivity requirements, i.e., the use of smaller radiuses is made possible.

How do our evolved, non-uniform CAs manage to out-perform the best uniform CA? Figure 7 displays four uniform CA runs, two of rule 232 (majority) which is the highest uniform CA rule with fitness 0.83, and two of rule 226 which has fitness 0.75 (rules are delineated in Figure 5). We note that rule 232 exhibits a small amount of information propagation during the first few time steps, however it quickly settles into an (incorrect) fixed-point. Rule 226 shows patterns of information propagation similar to those observed with the GKL rule (Figure 1), however no “decision” is reached.

The evolved, non-uniform CAs consisted of a grid in which one rule dominated, i.e., occupied most grid cells. In the lower performance CAs the dominant rule was rule 232 (majority) whereas in the high performance CAs rule 226 had gained dominance. We noted that rule 226 attains a fitness of only 0.75 on its own, though it is better than rule 232 at information propagation. Evolution led our non-uniform CAs toward a grid in which *most but not all* of the cells contain rule 226. Thus, instead of a low performance uniform CA, evolution has found a high performance *quasi*-uniform CA, with one dominant rule occupying most grid cells, while the other cells contain other rules.

The operation of one such CA is demonstrated in Figure 8. The grid consists of 146 cells containing rule 226, 2 cells containing rule 224 and one cell containing rule 234. Rules 224 and 234 differ by one bit from rule 226 (Figure 5). In Figure 8a, $\rho(0) \approx 0.40$ and we note that behavior is different at the two cells located about one third from the right, which contain rule 224. This rule maps neighborhood 001 to state 0 instead of state 1 as does rule 226, thus enhancing a neighborhood with a majority of 0s. The cells act as “buffers” which prevent erroneous information from flowing across them. In Figure 8b, $\rho(0) \approx 0.60$ and the cell located near the right side of the grid, containing rule 234 acts as a buffer. This rule maps neighborhood 011 to 1 instead of 0 as does rule 226, thus enhancing a neighborhood with a majority of 1s.

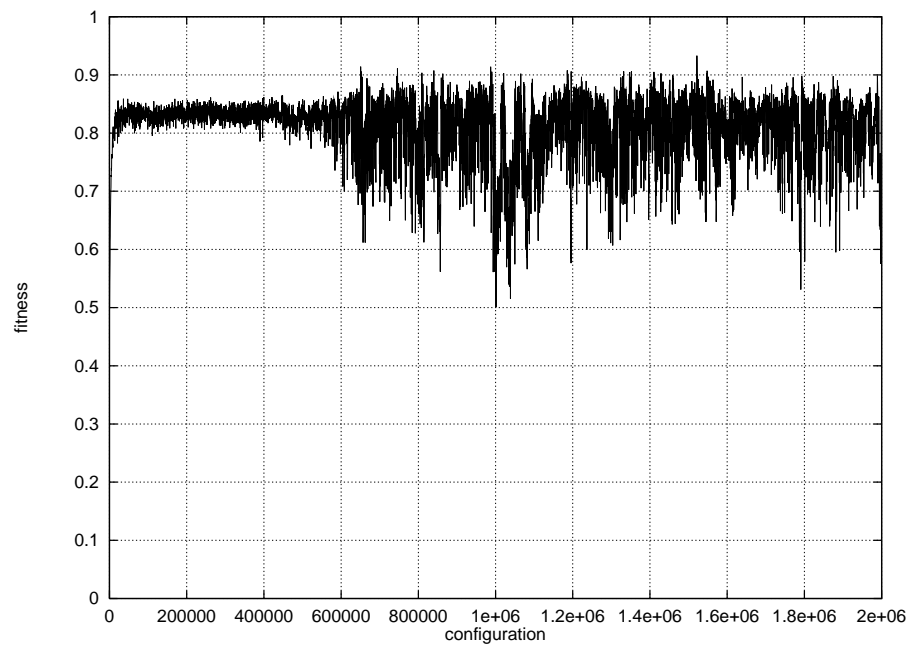
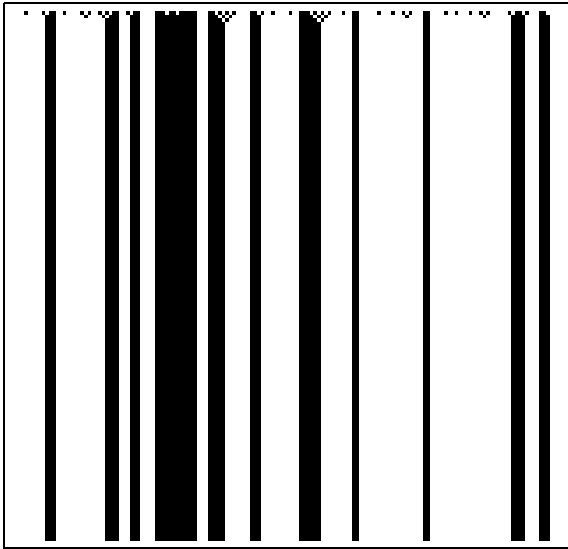
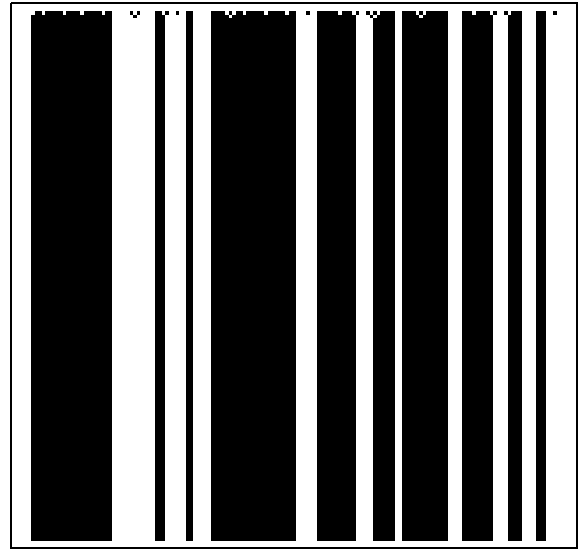


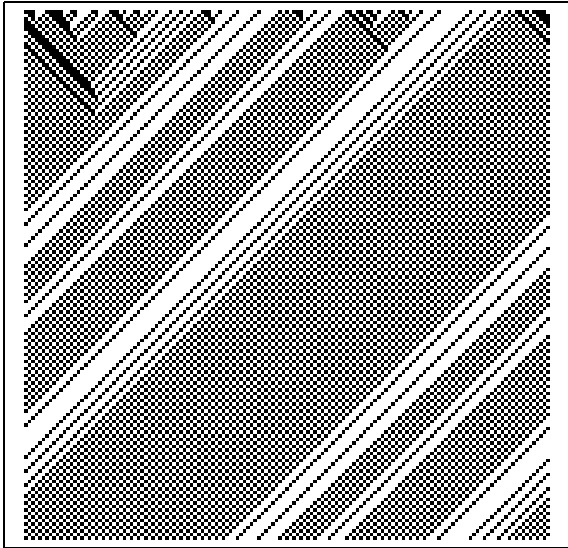
Figure 6: Result of a typical evolutionary run with $r = 1$. Graph depicts the average fitness of all grid cells (rules).



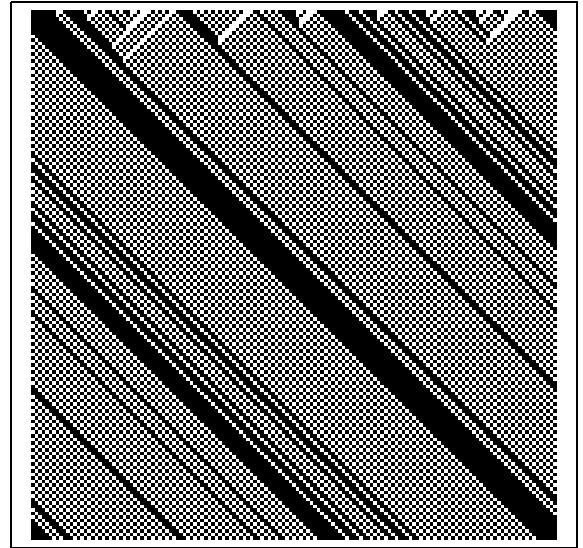
(a)



(b)



(c)



(d)

Figure 7: Space-time diagrams of uniform, $r = 1$ rules. (a) Rule 232 (majority). $\rho(0) \approx 0.40$, $\rho(150) \approx 0.32$. (b) Rule 232. $\rho(0) \approx 0.60$, $\rho(150) \approx 0.66$. (c) Rule 226. $\rho(0) \approx 0.40$, $\rho(150) \approx 0.40$. (d) Rule 226. $\rho(0) \approx 0.60$, $\rho(150) \approx 0.60$.

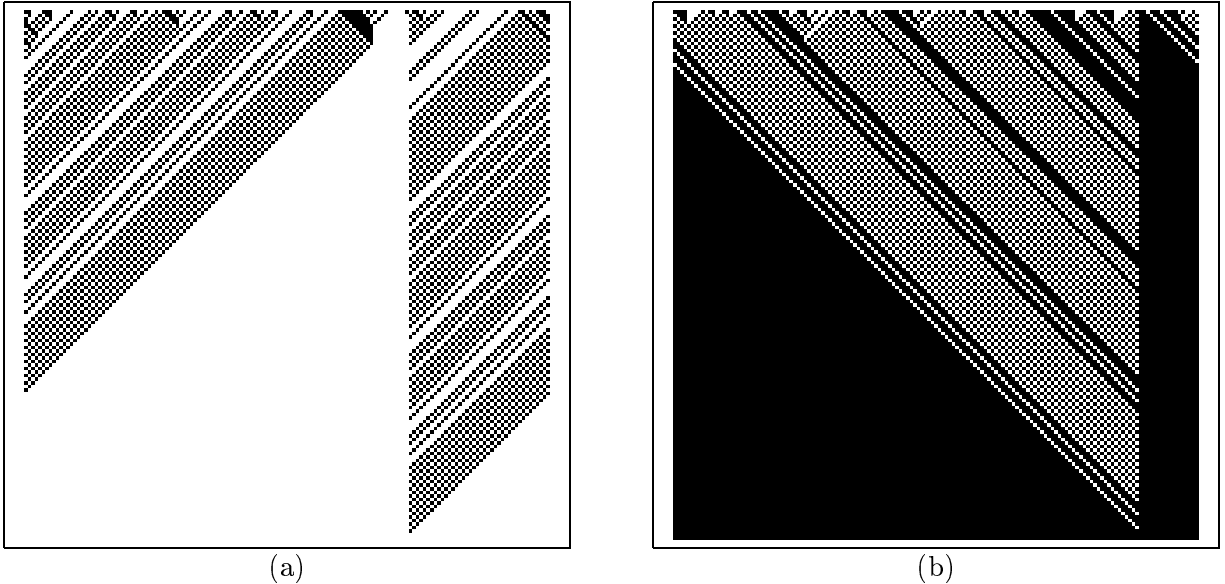


Figure 8: Space-time diagrams of a co-evolved, non-uniform, $r = 1$ CA. (a) $\rho(0) \approx 0.40$, $\rho(150) = 0$. (b) $\rho(0) \approx 0.60$, $\rho(150) = 1$.

The uniform, rule 226 CA is capable of global information propagation, however erroneous decisions are reached. The non-uniform CA uses the “capability” of rule 226 by inserting buffers in order to prevent information from flowing *too* freely. The buffers make local corrections to the signals which are then enhanced through time, ultimately resulting in a correct decision. Thus an evolved, quasi-uniform CA out-performs a uniform CA.

We have recently shown that two-dimensional, 2-state, 5-neighbor, quasi-uniform CAs can attain universal computation [Sipper, 1995a] which is not possible for the uniform case as proven by [Codd, 1968]⁸. For infinite grids quasi-uniformity is defined as follows: let $R_j(N)$ denote the number of cells with rule j in a grid of size N . Then, there exists m such that:

$$\lim_{N \rightarrow \infty} \frac{\sum_{n \neq m} R_n(N)}{R_m(N)} = 0$$

Essentially, this means that most of the grid contains the dominant rule (m), except for an infinitely small region which contains the others.

Uniform, two-dimensional, 2-state, 5-neighbor CAs cannot attain universal computation [Codd, 1968] as also uniform, one-dimensional, 2-state, 3-neighbor ($r = 1$) [Lindgren and

⁸Universal computation proofs assume an infinite grid. Codd proved that there does not exist a computation-universal, uniform, 2-state, 5-neighbor CA with *finite* initial configuration. Such a configuration consists of an assignment of non-zero states to a finite number of cells in the grid. We have shown that computation universality is possible using quasi-uniform CAs with finite initial configurations.

Nordahl, 1990]. Quasi-uniform CAs are capable of universal computation [Sipper, 1995a] and we have demonstrated above that evolution on a nontrivial task has discovered a quasi-uniform grid⁹. These results offer a possible path toward complex computation in non-uniform CAs.

Further insight may be gained by studying the *genescape* of our problem. The genescape, standing for evolutionary genetic landscape, depicts the incorporation of new genetic material into an evolving population. It was introduced by us in [Sipper, 1995b] and is derived from the work of [Bedau and Packard, 1992]. They discuss the issue of how to discern whether or not evolution is taking place in an observed system, defining evolutionary activity as the rate at which useful genetic innovations are absorbed into the population. They point out that the rate at which new genes are introduced does not reflect genuine evolutionary activity, for the new genes may be useless. Rather *persistent usage* of new genes is the defining characteristic of genuine evolutionary activity.

The model studied by [Bedau and Packard, 1992] is that of strategic bugs in which a bug’s genome consists of a look-up table, with an entry for every possible combination of states. They attach to each gene (i.e., each table entry) a “usage counter”, which is initialized to zero. Every time a particular table entry is used the corresponding usage counter is incremented. Mutation sets the counter to zero, while during crossover genes are exchanged along with their counters. By keeping track of how many times each gene is invoked, waves of evolutionary activity are observed through a global histogram of gene usage plotted as a function of time. As long as activity waves continue to occur, the population is continually incorporating new genetic material, i.e., evolution is occurring [Bedau and Packard, 1992]. While this measure is extremely difficult to obtain in biological settings, it is easy to do so in artificial ones, providing insight into the evolutionary process.

We have applied the idea of usage counters in [Sipper, 1995b]. Each gene in our genome (rule table) corresponds to a certain neighborhood configuration (input), specifying the appropriate action to be performed (output). In this respect it is similar to the strategic bugs model of [Bedau and Packard, 1992] and usage counters are attached to each gene and updated as described above. In [Bedau and Packard, 1992] the usage distribution function is defined, which is then used to derive the $A(t)$ measure of evolutionary activity. Since our genome is small (8 “genes”, i.e., rule table entries, for $r = 1$), we have opted for a more direct approach in which we study the total usage of each gene throughout the entire grid as a function of the number of configurations run so far. This measure is computed by summing the usage counters of all cells at a given configuration. Our measurements can then be presented in a three dimensional plot denoted the genescape [Sipper, 1995b].

The genescape of the non-uniform CA of Figure 8 is presented in Figure 9. We observe that genes 0 and 7 are the ones used most extensively, corresponding to neighborhoods 000 (which is mapped to 0 by all grid rules) and 111 (which is mapped to 1). This demonstrates the preservation of correct local information. Genes 4 (neighborhood 100) and 6 (neighborhood 110), of intermediate usage, also act to preserve the current state.

Observing Figure 9, we note that after several thousand configurations in which the

⁹For finite grids quasi-uniformity simply implies that the above ratio is small.

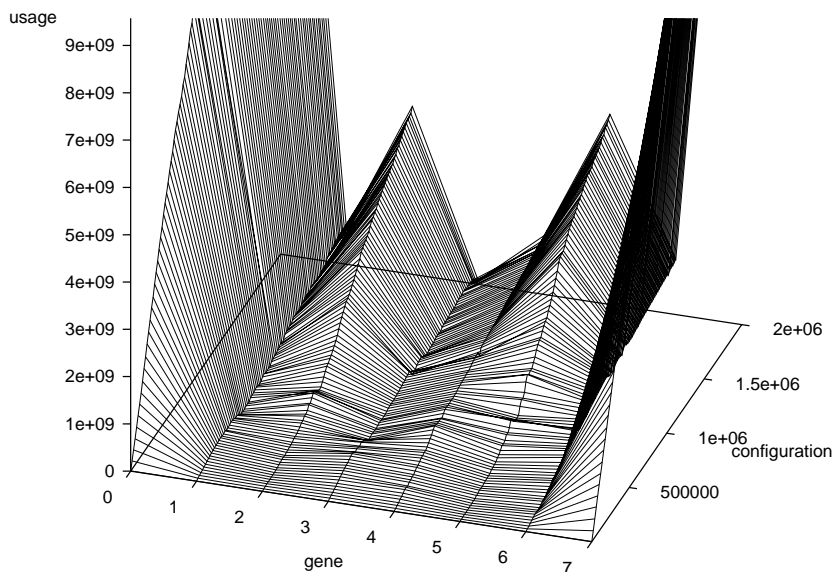


Figure 9: Genescape of an evolutionary run with $r = 1$.

landscape between genes 0 and 7 is mostly flat a growing ridge appears. This ridge reflects the increasing use of genes 2 and 5, corresponding to neighborhoods 010 (which is mapped to 0) and 101 (which is mapped to 1). These genes change the state of the central cell, reflecting an incorrect state with respect to the local neighborhood. Essentially, use of these two genes is related to information propagation which increases as evolution proceeds. The least used genes are 1 and 3 which are exactly those genes by which the dominant rule (226) differs from the other two (224, 234). Though used sparingly, they are crucial to the success of the system as noted above.

Returning to Figure 6, we observe that the evolutionary pattern consists of a period of relative stability followed by a period of instability. The high performance CA is found during the unstable period. This pattern resembles the punctuated equilibria phenomenon observed in nature [Eldredge and Gould, 1972] and also in artificial life models [Lindgren, 1992, Lindgren and Nordahl, 1994, Sipper, 1995b]. This phenomenon was not observed for $r = 2, 3$ (e.g., Figure 2a). It is noted that for $r = 1$ the rule table contains only 8 bits and therefore 1 bit changes may have more drastic effects than with $r = 2, 3$. However, this does not account for the initial period of stability. Moreover, note that the graph depicts average fitness and therefore instability is due to a system-wide phenomenon taking place. Fitness variations of a small number of cells would not be sufficient to create the observed evolutionary pattern. By comparing the genescape with Figure 6 some insight may be gained.

The configuration at which the ridge begins its ascent (Figure 9) coincides with the

beginning of the unstable period (Figure 6). As argued above the ridge represents the growing use of information propagation. Thus, we speculatively suggest that the shift from stability to instability represents a phase transition for our evolving CAs. As with the “edge of chaos” (Section 2) such a shift is associated with computational capability, evident in our case by periods of increased computational performance. While a full explanation of the punctuated equilibria phenomenon does not yet exist, evidence has been mounting as to its ubiquity among various evolutionary systems. Our above result suggests a possible connection between this phenomenon and the “edge of chaos” (see also [Kauffman, 1993], page 269, on the relation between co-evolution and punctuated equilibria).

Some qualitative differences have been detected between different values of r . As noted above, the $r = 2, 3$ cases did not exhibit punctuated equilibria as with $r = 1$. For $r = 1, 2$ all runs were successful, i.e., average fitness increased, as opposed to $r = 3$ where some runs were unsuccessful (Figure 2b). Quasi-uniform grids were not evolved for $r = 2, 3$ as with $r = 1$. It cannot be ascertained at this point whether these are genuine differences or simply a result of our use of insufficient computational resources (grid size, number of configurations), however there is some evidence pointing to the latter. For example, though quasi-uniformity was not observed for $r = 2, 3$ we did note that the average Hamming distance between rules decreases as performance increases¹⁰. With $r = 2$ average Hamming distance reached approximately 3 bits for high performance CAs. It would be interesting to run our experiments using increased computational resources to discover whether even higher performance can be attained.

6 Conclusions

We have described experiments in which non-uniform CAs are evolved to perform a computational task. The evolutionary algorithm presented involves local, co-evolution in contrast to the standard GA in which independent problem solutions are globally evolved. Our results demonstrate that: (1) non-uniform CAs can attain high computational performance, and (2) such systems can be evolved rather than designed. This is notable when one considers the huge search spaces involved, much larger than for uniform CAs.

Evolved, non-uniform CAs with radius $r = 3$ attained high peak performance. We noted that rules are clustered in the vicinity of $\lambda = 0.5$, mostly in the region $0.45 - 0.55$. It is argued that these results lend support to the conclusion of [Mitchell *et al.*, 1993] stating that λ values for the $\rho_c = 0.5$ task are likely to be close to 0.5 rather than to critical λ_c values as put forward by [Packard, 1988].

We have shown that smaller radius, non-uniform CAs can also be evolved to perform the task with high performance similar to (or even exceeding) $r = 3$ CAs. The case of $r = 1$ is perhaps most notable. The small size of uniform, $r = 1$ CA rule space enabled us to test all possible rules, thereby finding the uniform rule of maximal performance. We then demonstrated that non-uniform, $r = 1$ CAs can be evolved to perform the task with

¹⁰The Hamming distance between two rules is the number of bits by which their bit strings differ.

high performance similar to $r = 2, 3$ CAs and notably higher than uniform, $r = 1$ CAs. This suggests that non-uniformity reduces connectivity requirements, i.e., the use of smaller radiuses is made possible.

For $r = 1$ it was discovered that evolution tends to progress toward quasi-uniform grids which are dominated by one rule. The crucial factor pertains to the fact that dominance is not total (in which case a uniform CA would result), rather a small number of other rules exists. High performance is achieved by having the non-dominant rules prevent information from flowing too freely and make local corrections to passing signals. A study of the genescape provided us with further insight into the evolutionary process and a possible connection between the punctuated equilibria phenomenon and the “edge of chaos”.

A major impediment of CAs stems from the difficulty of utilizing their complex behavior to perform useful computations (Section 1). For example, we noted in Section 2 that the GKL rule’s fitness on the $\rho_c = 0.5$ task is 0.98, a value which has not been attained by us nor by [Packard, 1988, Mitchell *et al.*, 1993]. However, this value is simply a serendipitous effect since the GKL rule was not invented for the purpose of performing any particular computational task [Mitchell *et al.*, 1993]. Indeed, the difficulty of designing CAs to exhibit a specific behavior or perform a particular task has severely limited their applications; automating the design (programming) process would greatly enhance the viability of CAs. Our results offer encouraging prospects in this respect for non-uniform CAs.

An important issue when considering systems such as ours is that of scaling, where two separate matters are of concern: the genetic algorithm used to evolve CAs, and the CAs themselves. Genetic algorithms on uniform CAs scale at least on the order of $O(PN)$ where P is the population size and N is the grid size. Our algorithm co-evolves locally and therefore scales as $O(N)$ ¹¹. Thus, increasing computational resources for the evolutionary process is facilitated by our approach which is highly suited for fine grained, massively parallel architectures. As for scaling the CAs themselves (i.e., the successful solutions discovered by evolution) this is easier using uniform CAs, *assuming* the task can be solved by a small grid (otherwise the scaling problem involved pertains to the evolutionary algorithm). For non-uniform CAs quasi-uniformity may facilitate scaling since only a small number of rules must be considered, with their precise positions proving irrelevant in many cases. To date, we have performed some preliminary experiments where grids were scaled by simple duplication, resulting in successful systems.

As noted in the introduction our goals are similar to those put forward by [Mitchell *et al.*, 1994b], but in the context of non-uniform CAs rather than uniform ones: (1) to enhance our understanding of the ways CAs perform computations, (2) to learn how CAs may be evolved, rather than designed, to perform computational tasks, (3) to understand how evolution creates complex, global behavior in a locally interconnected system of simple parts. We believe some light has been shed on these issues and that our approach offers interesting paths regarding the evolution of spatially distributed systems. Non-uniform CAs hold potential for studying phenomena of interest in areas such as complex systems,

¹¹Note that P may be a function of N , $P(N)$, since larger grids may require larger populations. This is important when one considers the complexity gap between $O(N)$ and $O(PN)$.

artificial life, and parallel computation.

Acknowledgments

I am grateful to Melanie Mitchell for her careful reading of this manuscript and her many helpful suggestions. I am also grateful to Rajarshi Das, Eytan Ruppin, Eduardo Sanchez, Marco Tomassini, and an anonymous reviewer, for their helpful comments.

References

- [Bedau and Packard, 1992] M. A. Bedau and N. H. Packard. Measurement of evolutionary activity, teleology, and life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 431–461, Redwood City, CA, 1992. Addison-Wesley.
- [Bennett and Grinstein, 1985] C. Bennett and G. Grinstein. Role of irreversibility in stabilizing complex and nonenergetic behavior in locally interacting discrete systems. *Physical Review Letters*, 55:657–660, 1985.
- [Burks, 1970] A. Burks, editor. *Essays on cellular automata*. University of Illinois Press, Urbana, Illinois, 1970.
- [Codd, 1968] E. F. Codd. *Cellular Automata*. Academic Press, New York, 1968.
- [Das *et al.*, 1994] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H. -P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature- PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 344–353, Berlin, 1994. Springer-Verlag.
- [Das *et al.*, 1995] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.
- [Eldredge and Gould, 1972] N. Eldredge and S. J. Gould. Punctuated equilibria: and alternative to phyletic gradualism. In T. J. M. Schopf, editor, *Models in Paleobiology*, pages 82–115. Freeman Cooper, San Francisco, 1972.
- [Fredkin and Toffoli, 1982] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21:219–253, 1982.
- [Gacs *et al.*, 1978] P. Gacs, G. L. Kurdyumov, and L. A. Levin. One-dimensional uniform arrays that wash out finite islands. *Problemy Peredachi Informatsii*, 14:92–98, 1978.

- [Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- [Gonzaga de Sá and Maes, 1992] P. Gonzaga de Sá and C. Maes. The Gacs-Kurdyumov-Levin automaton revisited. *Journal of Statistical Physics*, 67(3/4):507–522, 1992.
- [Hartman and Vichniac, 1986] H. Hartman and G. Y. Vichniac. Inhomogeneous cellular automata. In E. Bienenstock, F. Fogelman, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, pages 53–57. Springer-Verlag, Berlin, 1986.
- [Kauffman, 1993] S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.
- [Langton, 1984] C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.
- [Langton, 1990] C. G. Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.
- [Langton, 1992] C. G. Langton. Life at the edge of chaos. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 41–91, Redwood City, CA, 1992. Addison-Wesley.
- [Li *et al.*, 1990] W. Li, N. H. Packard, and C. G. Langton. Transition phenomena in cellular automata rule space. *Physica D*, 45:77–94, 1990.
- [Li, 1992] W. Li. Non-local cellular automata. In L. Nadel and D. Stein, editors, *1991 Lectures in Complex Systems*, pages 317–327. Addison-Wesley, Redwood City, CA, 1992.
- [Lindgren and Nordahl, 1990] K. Lindgren and M. G. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.
- [Lindgren and Nordahl, 1994] K. Lindgren and M. G. Nordahl. Cooperation and community structure in artificial ecosystems. *Artificial Life Journal*, 1(1/2):15–37, 1994. The MIT Press, Cambridge, MA.
- [Lindgren, 1992] K. Lindgren. Evolutionary phenomena in simple dynamics. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 295–312, Redwood City, CA, 1992. Addison-Wesley.
- [Margolus, 1984] N. Margolus. Physics-like models of computation. *Physica D*, 10:81–95, 1984.
- [Mitchell *et al.*, 1993] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

- [Mitchell *et al.*, 1994a] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Dynamics, computation, and the “edge of chaos”: A re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models and Reality*, pages 491–513. Addison-Wesley, Reading, MA, 1994.
- [Mitchell *et al.*, 1994b] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994.
- [Packard, 1988] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301. World Scientific, Singapore, 1988.
- [Rasmussen *et al.*, 1992] S. Rasmussen, C. Knudsen, and R. Feldberg. Dynamics of programmable matter. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 211–254, Redwood City, CA, 1992. Addison-Wesley.
- [Sipper, 1994] M. Sipper. Non-uniform cellular automata: Evolution in rule space and formation of complex structures. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 394–399, Cambridge, Massachusetts, 1994. The MIT Press.
- [Sipper, 1995a] M. Sipper. Quasi-uniform computation-universal cellular automata. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *ECAL'95: Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pages 544–554, Berlin, 1995. Springer-Verlag.
- [Sipper, 1995b] M. Sipper. Studying artificial life using a simple, general cellular model. *Artificial Life Journal*, 2(1):1–35, 1995. The MIT Press, Cambridge, MA.
- [Smith, 1969] A. Smith. Cellular automata theory. Technical Report 2, Stanford Electronic Lab., Stanford University, 1969.
- [Toffoli and Margolus, 1987] T. Toffoli and N. Margolus. *Cellular Automata Machines*. The MIT Press, Cambridge, Massachusetts, 1987.
- [Toffoli, 1980] T. Toffoli. Reversible computing. In J. W. De Bakker and J. Van Leeuwen, editors, *Automata, Languages and Programming*, pages 632–644. Springer-Verlag, 1980.
- [Vichniac *et al.*, 1986] G. Y. Vichniac, P. Tamayo, and H. Hartman. Annealed and quenched inhomogeneous cellular automata. *Journal of Statistical Physics*, 45:875–883, 1986.
- [Vichniac, 1984] G. Vichniac. Simulating physics with cellular automata. *Physica D*, 10:96–115, 1984.

- [von Neumann, 1966] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A.W. Burks.
- [Wolfram, 1983] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983.
- [Wolfram, 1984] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.