2001

Complexity as Fitness for Evolved Cellular Automata Update Rules

Em Ward

Doug Blank
Bryn Mawr College, dblank@brynmawr.edu

Douglas Rolniak

Dale R. Thompson

Custom Citation

Ward, Em, Blank, Douglas S., Rolniak, Douglas, and Thompson, Dale R. (2001). Complexity as Fitness for Evolved Cellular Automata Update Rules. In Late Breaking Papers of the 2001 Genetic and Evolutionary Computation Conference.

Complexity as Fitness for Evolved Cellular Automata Update Rules

Em Ward
EE Department
University of Arkansas
Fayetteville, AR 72701
eward@uark.edu

Douglas S. Blank
CSCE Department
University of Arkansas
Fayetteville, AR 72701
dblank@uark.edu

Douglas Rolniak
CSCE Department
University of Arkansas
Fayetteville, AR 72701
drolnia@uark.edu

Dale R. Thompson
CSCE Department
University of Arkansas
Fayetteville, AR 72701
drt@uark.edu

Abstract

We investigate the state change behavior of one-dimensional cellular automata during the solution of the binary density-classification task. Update rules of high, low and unknown fitness are applied to cellular automata, thereby providing examples of high and low rates of successful classification. A spread factor, ω, is introduced and investigated as a numerical marker of state change behavior. The nature of ω describes complex or particle-like behavior on the part of the cellular automata over the middle region of initial configuration density-distribution, but breaks down at the ends. Because of the limitation on ω, a related jump-out term, *jot*, is selected for incorporation into the finess function for genetic algorithm evolution of update rules. The inclusion of *jot* in the fitness function significantly reduces the number of generations required to reach high rates of successful classification ($\geq$90%).

1 INTRODUCTION

This paper investigates state changes of one-dimensional, binary, cellular automata (CA) through time and space during attempts to solve the density-classification task. CAs are viewed as models of dynamical systems and computation [10], [11], [4]. The state of each cell may then change over time. The frequency of state changes and the spatial distribution of the state changes may reflect complex behavior by the CA [12], [6]. Our study looks at these state changes in CAs updated by rules evolved by a genetic algorithm (GA) for good performance in the density-classification task and compares them to those of CAs updated by evolved rules with poor performance. We describe our attempt to find a simple measure that can represent complex, or particle-like, behavior by CA, its inclusion in the fitness function for evolving update rules and its effect on evolution time.

1.1 BACKGROUND

Cellular automata are arrays, or lattices, of cells. Each cell may be in a finite number of states. At discrete time steps, the states of the cells are changed according to update rules [10], [11]. The solution of the density-classification task may be viewed as nontrivial computation that requires a CA to determine the majority state in its initial configuration (IC) and update itself within a given number of time steps to achieve the majority state in every cell [9]. Figure 1 shows a time-space diagram of this computation. The IC contains a majority of 0 states (off or white). Correct classification would require all cells in the lattice to achieve a 0 state. The CA has a solution when two successive time steps have identical states for all cells. The maximum number of time steps allowed for a solution is probabilistically determined as a function of the number of cells in the CA. Previous authors have examined this scheme and concluded that CAs performing the density-classification task exhibit a range of behaviors [12], [6], [9], [3]. When the behavior is one of self-organized cell state changes through time and space, it is called "complex". or particle-like [12], [9], [5].

Genetic algorithms have been used to evolve these rules [9], [8]. Mitchell, Hraber and Crutchfield described a GA approach by Packard [9]. Previous application of GA-evolved update rules have had lower performance than the Gacs-Kurdyamov-Levin (GKL) rule, with approximately 95% best performance of correct classification, with performance decreasing as lattice size increases [9], [8], [5]. The GKL, though not designed for the density-classification task, has properties (all 1s or all 0s attraction states) that lend it

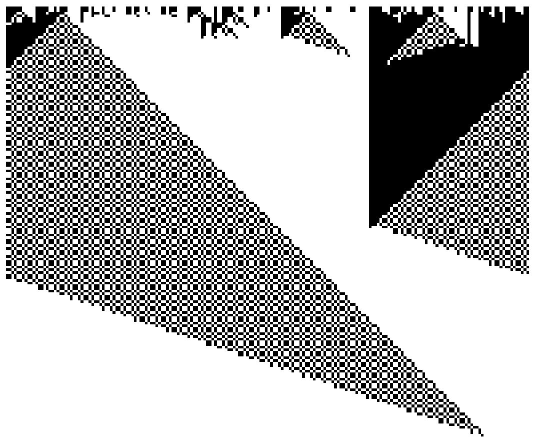

Figure 1: Diagram of a cellular automaton (CA) solving density-classification.

to successful application to the problem [2], [9]. The GKL rule consistently maintains approximately 98% average performance on increasingly large lattice sizes and number of trials [9], [8]. Investigators at the Santa Fe Institute found an evolved rule, ϕ_{par}, which induced self-organized behavior and performed similarly to the GKL rule [8].

1.2 RATIONALE

The rationale for examining cellular state changes through time is based on previous assertions that complex behavior, described by Wolfram as propagating, localized structures, sometimes long-lived, is required for computation [12], [6]. Different parameters have been offered in the literature as markers of complex behavior, including entropy, mutual-site information and difference-pattern spreading factor [7], [6], [1].

We investigate a spread factor, ω, and a jump out term, jot, that represent simpler, more intuitive measures of complex behavior. Incorporation of some measure of complexity into the fitness function of the GA rewards behavior. Presumably, this would speed evolution of high-performance update rules.

2 METHODS

2.1 UPDATE RULE GENERATION

We used the following parameters for the evolution of update rules: intital configuration (IC) length $N = 149$, radius $r = 3$, population size $p = 100$, number of generations $g = 100$, and the number of IC trials $t = 100$.

Each set of randomly generated ICs had uniform distribution of 1s and 0s. The radius is the number of cells to either side of the index cell, which constituted the neighborhood of the index cell. The lattice should be considered circular, as the neighborhoods wrap around. The neighborhood of 1s and 0s, read from left to right, served as rule table input. The inputs were ordered in the rule table from 0 to 2^{2r+1}. A genetic algorithm evolved the output column of the rule table. Fixed-point crossover, followed by two-point mutation was the means of reproduction. Twenty percent of the rule table outputs with highest fitness were kept in the population for each generation. The remaining 80% of the population were replaced with the produced children. Fitness was defined as the number of ICs correctly classified by M time steps, where M was a Poisson distribution with mean twice the initial configuration length, $2N$.

2.2 STATE CHANGE INVESTIGATION

From the evolved population of rule table outputs (rules) five with the highest fitness and five with the lowest fitness were fitness-tested on a very large set of trials, $t = 10,000$. This was done to insure we selected truly good performers and truly poor performers to use for the state change investigation. The good rules had fitness $\geq$89% (mean = 90.52%), and the five bad rules had fitness $\leq$51% (mean = 31.05%) on the 10,000 trials. In addition to these ten, five random rules were generated. The random rules were created as a population with no evolution. Five random rules and the GKL (fitness = 97.69%) rule were used as benchmark comparisons to the findings from the good and bad rules.

Each of the 16 total rules was then applied to ten trial ICs. The state change characteristics of these ICs were examined. We defined the state change frequency, f, as the number of state changes of a cell divided by the number of time steps before the automaton reached a fixed point or the maximum number of time steps allowed (M).

We analyzed f with the following measures: mean and lattice range (the maximum f in the automaton lattice minus the minimum f in the automaton lattice). Our measure of interest, ω, was intended to be a measure, a spread factor, of f across the lattice space and was defined as

$$\omega = range/mean. \quad (1)$$

Here is an example calculation - For cell #14, there were six state changes before a fixed point was reached at the 12th time step: $f_{14} = 6/12 = 0.5$. Across the lattice (149 cells), the minimum f (of 149) was 0 (no state change in at least one cell), and the maximum f (of 149) was 0.833 (ten state changes in at least one cell). Thus, the range is 0.833. Let the mean f across the lattice be $f_{ave} = 0.425$:

$$\omega = 0.833/0.425 = 1.960. \tag{2}$$

Since each set of ICs contained a uniform distribution of 1s, ω could give a divide-by-zero error if the ICs were all 0s. Also, at the ends of the spectrum, ω could be very large, but not representative of complex behavior. Therefore, we also examined ω over the section of the distribution, where 1s density ranges from 10% to 90% ($\hat{\omega}$). This includes the region of greatest difficulty for classification, where 1s density approaches 50%. Statistics were examined for each rule and each rule group (good, bad, random, GKL).

2.3 COMPLEXITY AS FITNESS FUNCTION

For incorporation into the fitness function, ω was examined and rejected because of the problem with high ω values on the ends of the IC density spectrum. This posed a problem with online or realtime evolution, as a correct classification also could be rewarded for a high ω value for solving the almost-all-1s or almost-all-0s tasks. Therefore, we introduced a jump-out term, jot, defined as

$$jot = m/M, \tag{3}$$

where m is the number of time steps to a solution and M is the maximum allowed time steps. jot captures the idea that complex behavior by CA requires more computational time (long-transients, or long-lived propagating structures). The normalization was required for jot to be less than one, since the term was added to the original performance fitness function, to reward complex behavior, only for a correct classification. Thus, the new fitness function was a real number with integer and fractional components. In this scheme, fitness could be greater than 100 (<101), but the performance, equal to the integer component of the fitness divided by total number of trials, could not exceed 100%.

For evaluation of a complexity measure incorporated into the fitness function, 30 runs with the same parameters describe above, were performed with each of three different fitness functions: standard, standard with jot and standard with a random number added to the right of the decimal point. The test group of 100 ICs was maintained for all functions.

Table 1: Spread Factors, ω and $\hat{\omega}$.

Parameter Rule Group	ω	$\hat{\omega}$	n	$\hat{n}$
Good	6.620	6.493	50	40
Bad	2.702	2.959	50	40
Random	3.266	3.932	50	40
GKL	7.514	7.294	10	8

n and $\hat{n}$ are number of trials for calculating ω and $\hat{\omega}$, respectively.

Table 2: Generations to High Performance (≥90%).

Fitness Function	mean	s. d.	variance
Standard	34.733	50.053	2505.306
jot	12.133	5.894	34.740
Random	26.067	41.460	1718.892

s. d. = standard deviation

3 RESULTS

Table 1 contains ω and $\hat{\omega}$ for each rule group. ω was calculated replacing 0/0 (range = 0, mean = 0) terms with ones. One can clearly appreciate the difference in ω for the good and bad rule groups (6.620 and 2.702, respectively). The GKL rule spread factor ($\omega = 7.514$) was close in value to the good rule group. The random rule group has ω similar to the bad rule group–3.266. The values for $\hat{\omega}$ are close to those for ω in each rule group.

Figure 2 is a representative plot of state change frequency, f across the cell lattice for a computation from a good rule ($\omega = 5.787$, averaged over ten ICs. Figure 3 is f from a bad rule ($\omega = 0.152$), averaged over ten ICs. These two plots show the frequency range across the lattice and will be discussed in the next section. There were instances in all groups where the minimum and maximum frequencies and the mean frequency were zero. In our samples, a bad rule rarely produced plots with state changes every single time step.

Table 2 shows the comparisons between the standard, standard with jot, and standard with random fitness functions with regard to number of generations to high performance (≥90%).

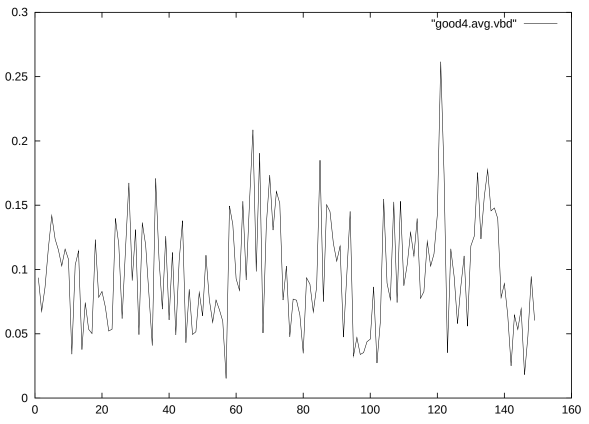

Figure 2: State change frequency, f, across space, good rule ($\omega = 5.787$). The vertical axis is f; the horizontal axis is cell number of the lattice. f is averaged over ten ICs.

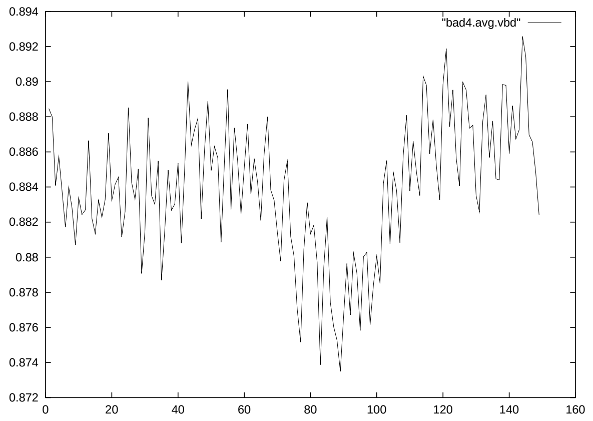

Figure 3: State change frequency, f, across space, bad rule ($\omega = 0.152$). The vertical axis is f; the horizontal axis is cell number of the lattice. f is averaged over ten ICs.

Table 3: Data Analysis of ω.

Comparison	Result	Confidence
ω_g vs. ω_b	$\omega_g > \omega_b$	>99.9%
ω_g vs. ω_r	$\omega_g > \omega_r$	>99.9%
ω_g vs. ω_{GKL}	$\omega_g = \omega_{GKL}$	NR

Subscripts indicate the following: g - good rule group, b - bad rule group, r - random rule group, and GKL - GKL rule group. NR = not rejected

4 DISCUSSION

4.1 DATA ANALYSIS

Difference of means tests for ω were applied to three group comparisons: good vs. bad, good vs. random and good vs GKL. The hypotheses tested were that the average ω for good rules are significantly higher than for bad or random rules and the same as the GKL rule. Table 3 contains the findings. With a high degree of confidence ($> 99.9\%$), the spread factor, ω, is significantly larger for a high rate of classification success than it is with a low rate of classification success. The analysis of $\hat{\omega}$ gave the same results, except the confidence of $\hat{\omega_g} > \hat{\omega_r}$ was 98.8%.

4.2 COMPLEX BEHAVIOR

In reports of computational models, the dynamical behavior of the computing entity has been studied [12], [6], [9]. Wolfram described four classes of behavior:

I. Homogeneous state
II. Separated simple stable or periodic structures
III. Chaotic pattern
IV. Complex localized structures [12].

In our investigation, ω and $\hat{\omega}$ represent patterns of state changes of a CA through time. Our results indicate that successful computation, or correct classification, occurs more reliably when lattice segments undergoing a high state change frequency are interspersed with segments of low state change frequency (see Figure 2). Classification is less successful when f is relatively uniform across the lattice (see Figure 3). This is consistent with the patterns of long-transients previously described [12], [6], [9], [8].

Table 4: Data Analysis of Fitness Functions.

Comparison	Result	Confidence
standard vs. *jot*	standard > *jot*	=99.3%
standard vs. rnd	standard = rnd	NR

rnd = random, NR = not rejected

4.3 FITNESS FUNCTION

Evolved GA update rules have had best performance ranging from 90% to 95%, occurring, generally, by the generation 40 [9]. When an evolved rule induces particle-like, or complex, behavior in the CA, the number of generations to high performance decreases [8]. We believe that our spread factor, ω, represents such behavior on the part of the CA. However, a limitation to ω is that it will be falsely high at the ends of the 1s density spectrum of the IC. This is a byproduct of the forced bias of uniform distribution of states across the initial configuration–a move to reduce the likelihood of having to classify difficult densities (near 0.5). This problem can be reduced with a form of $\hat{\omega}$ as described above, though $\hat{\omega}$ is not easily incorporated into the fitness function during online or realtime evolution. We believe ω would speed evolution on ICs with random distribution, since the probability of almost-all-1s and almost-all-0s would be very low. This is left as future work. Another comparison to investigate is the range of f (the numerator of ω) as a fitness function, removing the problem with the denominator being zero or very small, giving very high ω.

A simpler parameter, jot, represents computation time by the CA, an indicator of complex behavior. When jot is incorporated into the fitness function, complexity, in addition to performance, is awarded. When this happens, the number of generations to high performance is significantly reduced. Table 4 contains the analysis results for the number of generations to high performance for these comparisons: standard vs. jot and standard vs. random. Our samples did not provide support for rejecting the hypothesis that speed of evolution is the same for the standard fitness and standard with random number added.

5 CONCLUSIONS

This paper describes an investigation into the state change frequency of cellular automata during performance of the density classification task. We introduced a spread factor, ω, whose value was significantly higher when CAs had a high rates of successful classification compared to low rates. ω is the range of state change frequencies of the cells in the CA lattice divided by the average state change frequency. It is a numerical representation of the activity of the CA and consistent with previous descriptions of complex behavior required for computation.

A jump out term, jot, captures the idea that complex behavior by CA requires more computational time (long-transients, or long-lived propagating structures). When jot is incorporated into the fitness function for evolving CA update rules with a GA, the number of generations to high performance rules is significantly reduced.

Acknowledgments

We thank Ross Williamson for his significant programming contributions and Russell Deaton for his helpful commentary.

References

[1] R. Badii and A. Politi. Thermodynamics and complexity of cellular automata. *Physical Review Letters*, 78(3):444–447, 1997.

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

[3] M. Garzon. *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*, chapter 8, Classification, pages 141–166. Springer, New York, 1995.

[4] J.R. Koza. *Genetic Programming*, chapter 14, Entropy-Driven Evolution, pages 395–417. MIT Press, 1992.

[5] M. Land and R.K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):5148–5150, 1995.

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

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

[8] M. Mitchell, J.P. Crutchfield, and R. Das. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications*, Moscow, 1996. Russian Academy of Sciences.

[9] 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.

[10] J. von Neumann. *John von Neumann Collected Works*, chapter 9, The General and Logical Theory of Automata, pages 288–328. Pergamon Press, New York, 1961.

[11] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55:601–644, 1983.

[12] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.