Evolution of 2D Apoptotic Cellular Automata

Jennifer Garner and Daniel Ashlock

Abstract—An apoptotic cellular automata consists of an initial state and an updating rule. These specify an automata that grows for a time and then enters a quiescent state. This study generalizes earlier work on evolving 1D apoptotic automata to evolving 2D automata, producing a type of evolved art. Parameter studies are performed and it is found that the most important factors are algorithm runtime and the symmetry of the initial conditions of the automata. Other parameters such as mutation rate and tournament size are found to be relatively soft, as long as they do not take on extreme values. A collection of examples of renderings of evolved cellular automata are provided and steps for additional work to improve the system are outlined. Examination of automata with asymmetric starting conditions shows that the highest fitness individuals are those that follow a growth pattern that restores symmetry. This strongly suggests that optimizing the size of an apoptotic automata that has a symmetric pattern of states is a substantially easier problem.

Keywords: Cellular automata, evolutionary computation, symmetry, automata rule induction.

I. INTRODUCTION

Cellular automata are a discrete model of computation with three distinct parts:

1) A collection of cells. For each individual cell, the “surrounding” cells that influence the state of that cell are called its neighborhood,

2) a set of possible cell states, and

3) an updating rule or transition rule that maps the collection of possible cell states of each neighborhood to a new state for the cell with which the neighborhood is associated.

Cellular automata (CA) can be described as discrete dynamical systems, some of which exhibit self-organizing behavior. A cell population will evolve according to the transition rule for neighborhoods, which may be synchronous or asynchronous. Many real-world systems are dynamic in nature, and can be modeled as CA [35]. Cellular automata can be used as models for complex natural systems that contain large numbers of identical components experiencing local interactions [34], [27].

This study generalizes earlier work [4] on apoptotic cellular automata from one to two dimensions. Because automata are rendered as time histories, the dimension of the resulting evolved art is increased from two to three. The problem under study requires that one of the designated CA cell states is defined as being quiescent or “dead”. For “apoptotic” growth, the optimal outcome is to produce an automata containing as many live cells as possible, subject to the constraint that the automata enters a quiescent state in which all cells are dead before a predefined number of iterations (time steps). This predefined number of iterations is called the time limit for the system. Fitness is defined as the number of live cells produced, but a fitness of zero is awarded to automata that exceed the time limit and/or hit the boundaries of their cell array. Forbidding the automata from growing outside of a set horizontal region prevents problems that arise from periodic boundary conditions. When periodic boundary conditions are permitted, the automata is sometimes apoptotic only for particular sizes of the cell array, an undesirable property. Since the automata undergo a self-organized form of planned senescence, we call them apoptotic cellular automata, in an analogy to apoptotic programmed cell death that occurs in some biological cells.

A commonly studied, interesting problem for binary automata is the synchronization problem [11], [17], in which all cells of the automata must be synchronized. Once synchronized, the cells must continue to exhibit dynamic behavior, alternating states through time. This requirement of dynamic behavior avoids the solution where the automata synchronizes trivially by only allowing one state in any update. This task is an easier version of the firing-squad problem [36], which requires that the synchronization occur the first time the synchronized state is entered. A common feature of this research is that it creates hard problems by limiting the number of available cell states, whilst using small neighborhoods. This approach produces models of computation that are as simple as possible and makes analysis easier.

Much early research in evolving cellular automata rules focused on one-dimensional automata with two states and neighborhoods of three adjacent cells [23], [11], [19], [18]. The most common problem studied was the unsolvable density problem [19]. Given any set of initial cell states, the automata is supposed to take all cells to whichever state formed the majority, initially. The justification for this line of research was that reducing the number of unsolved cases for an impossible task was a good test problem. A more recent paper on this problem is [33]. An advance in this period was the discovery of particle-based computation [10], in which transient blocks of states move through the cell space and interact to perform computations. In [19], the co-evolution of training cases for the density problem was explored. A more recent publication in this line of research [18] showed that performance on the density problem could be enhanced by using both resource sharing and co-evolution of the test.
cases used in fitness evaluation.

Due to the ability of CA to model natural systems, CA models are finding applications in the field of medicine [12]. One of the first applications of CA to the study of cancer was a study of radio–therapy [14]. Since then, CA have been used to investigate the glycolytic phenotype [26], tumour morphology [15] and tumour invasion [15], [16]. For example, in Bankhead and Heckendorn [7] cancer is described by an evolutionary process, where mutated cells undergo selection for abnormal growth and survival that lead to a tumor. CA were used to model this process, as they can incorporate mutation, heritability, and selection. This CA model is being used to investigate why hereditary forms of breast cancer evolve more aggressive growth. A hybrid CA–partial differential equation model has been used to describe the interactions between a tumor and its nutrient source, and the immune system of the host organism [21]. This model considers both the spatio–temporal and partially stochastic interactions that exist between individual tumor cells and multiple populations of individually recognized immune cells.

As well as medical applications, CA have been applied to the study of a diverse range of topics, such as structure formation [8], heat conduction [9], language recognition [25], traffic dynamics [20] and cryptography [2], to name a few. CA have also been used for more aesthetic purposes, such as image and sound generation. Serquera and Miranda of the Interdisciplinary Center for Computer Music Research, UK, have published many works on the use of CA for sound synthesis [28], [1]. Much of their work consists of mapping the histogram sequence of a CA evolution onto a sound spectrogram, which produces spectral structures evolving in time. It is claimed that the mapping produces a “natural” behavior, and can replicate acoustic instruments [29].

CA have also been applied in the visual arts to produce artistic images [5], [24]. They have also been extended to the fields of architecture and urban design [30], [13]. An interesting application has been the use of CA in simulating the emergence of the complex architectural features found in ancient Indonesian structures, such as the Borobudur Temple [31]. Ashlock and Tsang [5] produced evolved art using one-dimensional CA rules. CA rules were evolved using a string representation. The CA either underwent slow persistent growth, or planned senescence. The resulting fitness landscapes were conjectured to be rugged with many local optima. These systems produced aesthetically pleasing images.

II. A REPRESENTATION FOR 2D CA RULES

The cellular automata used in this study are synchronous and have a set of cells comprising a two-dimensional, square grid 101 cells on a side. Examples of renderings of evolved automata rules appear in Figure 1. The neighborhood of a cell is the 3x3 square with the cell at its center. The automata have $K$ cell states realized as integers $n$ with $0 \leq n \leq K - 1$. The value 0 is designated as the dead or quiescent state. The updating rule is stored as a vector of cell states. In order to compute the updating rule, the neighborhood of a cell is summed. This sum is an integer in the range $0 \leq s \leq 9(K - 1)$. The sum $s$ is used as an index into the vector of cell states that specify the updating rule, meaning that the vector has a length of $9(K - 1) + 1$.

Fig. 1. Examples of renderings of 2D cellular automata.

As well as medical applications, CA have been applied to the study of a diverse range of topics, such as structure formation [8], heat conduction [9], language recognition [25], traffic dynamics [20] and cryptography [2], to name a few. CA have also been used for more aesthetic purposes, such as image and sound generation. Serquera and Miranda of the Interdisciplinary Center for Computer Music Research, UK, have published many works on the use of CA for sound synthesis [28], [1]. Much of their work consists of mapping the histogram sequence of a CA evolution onto a sound spectrogram, which produces spectral structures evolving in time. It is claimed that the mapping produces a “natural” behavior, and can replicate acoustic instruments [29].

Earlier studies on apoptotic cellular automata have shown that the representation is highly epistatic and so one point crossover was used. Mutations consist of replacing from 1 to $M$ of the entries in the updating rule with new ones. The value of $M$ is specified in individual experiments. We use a time limit of 300 updatings of the cell states. The number of loci replaced was selected uniformly at random. The fitness of an automata rule was computed as follows:

- The cell array was loaded with the initial state shown in Figure 2
- The automata was updated, synchronously, until every state was dead (had the value zero) or until the time limit expired.

Fig. 2. The initial state of the cellular automata is 0 everywhere except for five cells centered in the cell array with the values shown in bold above.

- The cell array was loaded with the initial state shown in Figure 2
- The automata was updated, synchronously, until every state was dead (had the value zero) or until the time limit expired.
• If the automata still had living (nonzero) cells at the end of the time limit or if one of the cells at the edge of the cell array was alive, the automata was awarded a fitness of zero.
• Otherwise the fitness of the automata rule was the number of living cells, summing over all updatings of the cell states.

Notice that the fitness evaluation of an automata is based on its time history, so the actual object produced is three-dimensional. The two dimensions of the cell array and the time dimension are combined in fitness evaluation, counting the number of living cells at any point during the automata’s growth.

An automata rule was rendered by superimposing each time step upon the previous ones, with the exception of dead pixels. The individual cell states were assigned colors. A cell in the rendering has the color associated with the most recent living state for that cell. A cell is white in the rendering only if it was never alive. Figure 1 shows four automata rendered in this fashion.

III. DESIGN OF EXPERIMENTS

The evolutionary algorithm used is steady state [32] with single tournament selection. This model of evolution picks a tournament of some number of individuals. The two most fit members of the tournament reproduce with crossover and mutation and replace the two least fit. One act of tournament selection is called a mating event. A default set of parameters, given in Table I, was chosen by preliminary experimentation. The experiments vary the parameters individually. In each experiment, 30 independent runs of the evolutionary algorithm are performed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automata states</td>
<td>8</td>
</tr>
<tr>
<td>Initial state</td>
<td>Figure 2</td>
</tr>
<tr>
<td>Mating events</td>
<td>10,000</td>
</tr>
<tr>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td>Tournament size</td>
<td>7</td>
</tr>
<tr>
<td>Max. mutations $M$</td>
<td>3</td>
</tr>
<tr>
<td>Crossover</td>
<td>One point</td>
</tr>
<tr>
<td>Time limit</td>
<td>300</td>
</tr>
<tr>
<td>Cell array size</td>
<td>101</td>
</tr>
</tbody>
</table>

Initial populations of automata rules are filled in uniformly at random from the set of available cell states, except that the first entry, corresponding to the updating of a neighborhood comprised entirely of dead cells, must be set to zero. Six experiments were run, each varying parameters one at a time, as follows:

1) The number of mating events was varied with the values 5,000, 10,000, 20,000, and 200,000.
2) maximum number of mutations was chosen from $M \in \{1, 2, 3, 4, 5\}$, 
3) population size was one of 100, 500, 750, 1000, 1250, or 10,000,
4) automata states were one of 4, 6, 8, or 10,
5) tournament sizes was one of 4, 7, 10, or 13, and
6) initial state of the automata was chosen from those shown in Figure 3 when the initial state shown in Figure 2 is not used.

![Figure 1](image1.png)

![Figure 2](image2.png)

![Figure 3](image3.png)

IV. RESULTS AND DISCUSSION

The results for the six experiments are shown in Figure 4 using box plots. The first panel of the figure shows that increasing mating events increases fitness, but it necessarily also causes a large increase in the time needed to run the code. This experiment suggests that running the code for an even longer time may pay additional benefits. The experiment with 20,000 mating events has a high fitness outlier, something consistent in the earlier work on 1D apoptotic automata that found that the fitness landscape was quite rugose. A check, shown in Figure 6, demonstrates that the 2D automata retain this rugosity.

The mutation experiment suggests that mutation is a soft parameter as long as the maximum number of mutations is not too small. The only significant difference is between one mutation and all the other values tested. In contrast, the population size experiment showed strong results from having too many or too few population members. The usual explanation suffices for too many - given that the number of mating events was held constant at 10,000 in this experiment, the largest population size permitted an average of one mating event per population member. The small population, however, with many mating events per population member, might have turned in a better performance. The poor performance at small population sizes is explicable by the fact that most...
Fig. 4. Experimental Results
members of the initial population have zero fitness, because they grow past the time limit or hit the sizes of the array.

This effect was quantified by sampling 100,000 genes for the default parameter settings. The fraction of nonzero genes was computed. Treating it as a binomial parameter, a confidence interval for the probability of a nonzero fitness gene was constructed using the normal approximation to the binomial distribution. The resulting 95% confidence interval for the parameter was:

$$0.115 \pm 0.002$$

Therefore, using the default algorithm parameters, a 100-member population would contain about 11-12 genes with non-zero fitness, which is not enough diversity to give evolution a good start.

Changing the number of states in the automata had relatively little impact on fitness, except that four states are clearly not enough with the default algorithm parameters. Recall that the gene length is $9(K - 1) + 1$ for $K$ states. Thus four states gives a gene length of 28, six 46, eight 64, and ten 82. The size of the search space is $K^{9(K-1)}$ (the terminal +1 is sacrificed to the need for the first state of the gene to be zero). From this we see that the size of the search space grows astronomically with the number of states available. Simple sampling theory suggests that the ten state space has higher fitness individuals in it than the eight state space, but with only 10,000 mating events it seems likely that the algorithm is simply not locating them. None of the experiments approached the theoretical maximum of roughly $\frac{1}{3} \times 99^2 \times 299 \approx 976,000$. This maximum is computed by assuming the automata fill a square-based cone with side length $S = 99$ and height $H = 299$ which is the maximum possible volume if the automata misses the sides of the array and the time limit. The volume of such a cone is given by:

$$V = \frac{1}{3} S^2 H$$

The experiments with tournament size, other than showing a spectacular outlier for size 4, demonstrated that tournament size is another soft parameter. The best size is ten, but the result is not statistically significant.

The results for changing the initial conditions are striking and the key concept is symmetry. The three best results are for initial conditions $A$, $D$, and $G$ which are the three symmetric initial conditions. Next best is $F$ which is symmetric in one of two possible directions. A collection of final best automata for symmetric initial conditions are shown in Figure 7, while maximally asymmetric results evolved with asymmetric initial conditions are shown in Figure 5. The asymmetric results do a poor job of covering the cell array.

One of the conditions of fitness evaluation is that the living cells must not reach the border of the array. An asymmetric automata is likely to approach the border in one direction, rather than four simultaneously. The volume filled by the automata is, itself, asymmetric and so must be smaller than a symmetric volume in order to miss the border.

Changes in the fitness of a cellular automaton rule come in quantum leaps. Examples of this effect are shown in Figure 6, where changes in fitness are often large jumps after a period of apparent stasis. What is not apparent is that the sort of changes to a rule that create such jumps have the potential to hit the boundaries of the cell array or to exceed the time limit, thus yielding zero fitness. The small sizes of the asymmetric results in Figure 5 are probably the result of
having many potential mutants with more living cells that receive a fitness of zero for hitting a boundary.

Within the runs using asymmetric initial conditions, the higher fitness individuals were those that eliminated their asymmetry early in their time history. The first automata in Figure 5 is an example of this - the symmetric portion arose in later time steps. Given that the goal of this study is a novel form of evolved art, this result is a convenient one. Symmetry is typically a more aesthetically pleasing state.

V. CONCLUSIONS AND NEXT STEPS

This study performed parameter-setting for a system that evolves 2D cellular automata and demonstrated that the system can be used to produce evolved art. The most important parameters were found to be algorithm runtime and the choice of symmetric initial conditions, which helped to increase the fitness and granted an artistically desirable characteristic to later renderings. The mutation rate and number of cell states parameters proved moderately soft so long as they were not set too low. Population size, similarly, must not be too low because of the high probability, roughly 88%, that a randomly generated rule will have zero fitness. It is worth noting that larger populations may prove superior if algorithm runtime is increased, an early priority for additional work.

A. Single Parent Techniques

Earlier work with one-dimensional automata [22] demonstrated that single parent techniques [6] can be used to produce automata that preserve the appearance of a progenitor rule while filling a larger space. Single parent techniques rely on a population of examples, called ancestors. They add a new variation operator called single parent crossover, which applies crossover to a population member and a copy of the ancestor. This copy is subsequently discarded." This technique makes the information in the ancestor(s) constantly available to the algorithm. Earlier research with a single ancestor found the intriguing side effect that evolved automata often preserved the look-and-feel of their ancestor(s). Given that fitness evaluation of the 2D automata presented in this study is quite slow, implementation of a single parent evolution system would be a good direction for additional research.

B. Fitness Landscapes

In [4] and [3], the fitness landscapes for apoptotic cellular automata were investigated; it was found that the apoptotic rules were strongly clustered in a small part of the space of rules. Therefore, chains of substitutions of loci in one high-fitness rule for elements of another high-fitness rule produced rules that, while less fit than the two original rules, had an enormously enhanced probability of not having zero fitness. It seems likely that these same properties hold for the fitness landscape of 2D cellular automata, but the experiments remain to be done. Given this issue, with a high probability of a random rule having zero fitness, it also seems likely that uniform crossover of high fitness rules might be a good initialization strategy for populations used in the evolution of apoptotic cellular automata in one or two dimensions.

C. Better Rendering

The objects produced by the evolutionary algorithm presented in this study are, in fact, three dimensional. The authors managed to observe some of this three dimensional character by examining animations of the layers of the automata as they formed (examples of these animations are
Fig. 7. Shown are examples of best-of-run cellular automata for the eight-state version of the system with initial condition A.
available from the second author on request). As one might expect from examining time histories of one-dimensional automata, the geometric shapes of the automata are complex and beautiful, with hanging turrets and multiple tube-like supports. The format of a paper does not yet support animation, and so better rendering of the time histories is another early priority for future research.

D. Better Algorithms

The study used a simple evolutionary algorithm to evolve apoptotic cellular automata rules. The rugosity of the CA landscape strongly suggests that algorithms with enhanced exploratory power might be quite useful. Geographically structured evolutionary algorithms like graph based evolutionary algorithms [?] or ring optimizers [?] are substantially more exploratory than standard evolutionary algorithms with well-mixed populations. Trying this sort of algorithm on the apoptotic cellular automata rule induction problem is an early priority for additional research.

E. Going Parallel

A synchronous cellular automata is transparently parallelizable. The updating of neighborhoods, other than needing access to the same data items, are completely independent. The new value of one cell is independent of the others in the current time. The relatively small default number of fitness evaluations used, 10,000, reflects the high cost of fitness evaluation, and so using parallel methods on a cluster would permit much greater exploration of the space of 2D apoptotic cellular automata. In particular, larger cell arrays could be used to produce fine-grained, poster-sized automata.

REFERENCES


