Evolved Art via Control of Cellular Automata

Daniel Ashlock and Jeffrey Tsang

Abstract—This is the second study exploring the creation of evolved art through evolutionary control of a dynamical system. Here 1-dimensional cellular automata rules are evolved to exhibit slow but persistent growth or to undergo planned senescence. These simple constraints encourage the automata to develop complex and visually pleasing behavior. Isotropic automata with a forced quiescent state are used, with rules evolved using a simple string representation; the fitness landscapes for both fitness functions are found to be quite rugged with many local optima. This is a desirable feature in an evolved art system as it yields a rich variety of outputs for the artist to use as image elements. A parameter study is performed and it is found that optimization of the slow-growth fitness function favors the use of large populations.

I. INTRODUCTION

In [15] initial conditions for a set of position/momentum symmetric planets orbiting a central star were evolved to exhibit stability under numerical integration of Newtonian dynamics. Given the symmetry of the planets, classical Newtonian dynamics would not permit the star to move at all. Motion of the central star, beyond a very small threshold, thus permits the easy detection of numerical instability. The paths of the orbiting planets form intriguing patterns and were used as a type of evolved art. A surprising result of this research was that it was possible to evolve rare configurations that were \(10^5\) times more stable than a random sample of initial configurations. In other words the number of numerical integration steps in the most fit evolved configurations required 10,000 times as many numerical integration steps before significant motion of the central star occurred. This study checks for a similar phenomenon in a very different domain. Rather than attempting to find numerical stability in a model of a continuous dynamical system, this study deals with a type of discrete time dynamical system: cellular automata.

Cellular automata have been used to solve a broad variety of problems with varying degrees of success [17], [16]. There has been a good deal of work on evolving cellular automata to perform specific computations done by the Evolving Cellular Automata group at the Sante Fe Institute [12], [13], [7], [8]; a survey of much of their work appears in [11]. Cellular automata often have a striking visual appearance, see Figures 1 and 2 for some examples. Efforts to use cellular automata in media and art include [17], [6]. A recent collection on evolved art and music [14], does not include even one chapter on cellular automata however. Using cellular automata for art and design is not uncommon, and the

New Kind of Science conferences sponsored by Wolfram Research have many presentations in this area. For details see: http://www.wolframscience.com/conference/2007/

In this study we have two goals. The first is to see if the sort of control of the stability of planetary dynamics achieved in [15] can be repeated for cellular automata. The second is to demonstrate that evolutionary algorithms can yield a simple rule selection method to constrain the behavior of cellular automata. To meet the first goal we evolve cellular automata to grow (in width) as slowly as possible without achieving a fixed width or dying out. In this case we equate width of the automata with numerical stability in the planetary orbit experiments. Cellular automata that grow slowly while continuing to grow will need to exhibit fairly complex behavior and so may yield esthetically pleasing appearances. Evolved examples appear in Figure 1.

A second set of experiments attempts to control cellular automata in the opposite direction. We evolve automata to cover as much area as possible but to die (have all cell states quiescent) by the 600th time step. Evolved examples appear in Figure 2. Both the slow-growth and bounded-growth experiments serve the second goal of producing visually pleasing images.

The remainder of this study is organized as follows. Section II gives the design of the experiments performed, including the selection of the cellular automaton model and the design of the evolutionary algorithm. Section III gives and discusses the results. Section IV draws conclusions and Section V outlines possible next steps.

II. DESIGN OF EXPERIMENTS

The design of the experiments in this study have two components. First is the cellular automaton model - there are a huge number of possible spaces of cellular automata rules depending on the choice of cell space, the type of updating rule, and the method of updating (synchronous, asynchronous). The second component is the design of the evolutionary algorithm. As we will see an entirely vanilla design suffices for the cellular automaton rule selection problems posed here.

A. The Cellular Automaton Model

For the first and second set of experiments, linear one-dimensional totalistic cellular automata with 101 or 600 cells respectively are used, cells having a set of six possible states: \(\{0, 1, 2, 3, 4, 5\}\). The automata are updated synchronously. Neighborhoods used in updating for a cell consist of the cell itself and its two adjacent neighbors. Updating rules are functions from the set \(\{0, \ldots, 15\}\) to the cell state, encoded as a string of 16 integers from the set of cell

Daniel Ashlock and Jeffrey Tsang are with the Department of Mathematics and Statistics at the University of Guelph, in Guelph, Ontario, Canada, N1G 2W1. Email: dashlock@uoguelph.ca, jtsang@uoguelph.ca

The authors thank the National Science and Engineering Research Council of Canada (NSERC) for supporting this work.
Fig. 1. Examples of evolved cellular automata from the slow-growth experiment.
Fig. 2. Examples of evolved cellular automata from the minimal-growth experiment. Several of these automata time-histories were inverted to fit more into the available space.
states. Rules are applied by summing the values in a cell’s neighborhood to obtain a number in the range 0-15 and then looking up the value in the string of values encoding the cellular automaton rule. The first element of the automaton rule is always zero so that the neighborhood state \[0][0][0]\] always maps to zero. Zero is called the quiescent or dead state. All other states are considered active or alive. Notice that this class of rules is symmetric with neighborhood states \[a][b][c]\] and \[c][b][a]\] being updated in an identical fashion.

Fitness evaluation takes place in the context of repeated updating of the cell states to obtain the time-evolution of the automata. Figures 1 and 2 contain examples of such time evolutions. We call the time history of the automata, viewed as rows of a two-dimensional array being filled by time-evolution of an automata, as an evaluation strip for computing fitness. Two fitness functions are defined. The slow growth fitness function works as follows:

The automaton is 101 cells wide and positions 49, 50, and 51, using 0-base counting are initialized to cell states 1, 2, and 1, respectively. All other states are initialized to 0. The automaton is updated until one of the following termination conditions is met.

- If the cells at the ends of the one-dimensional array become active, the number of updating steps this takes is returned as the automaton’s fitness.
- If at some point every cell is inactive, a fitness of zero is returned.
- If the automaton reaches 10,100 updatings without the cells at the ends of the one dimensional array becoming active, a fitness of zero is returned.

The first termination condition selects for automata of interest and encourages those that grow slowly. The second termination condition expresses disinterest in automata that die before 10,100 updatings occur. The third termination condition detects automata that reach a point where they neither grow nor die before they make the end cells active. This third condition does not ensure that the automata will continue to grow after hitting the end cells, but it does require some growth from 3 to 101 cells to take place. Automata evolved with the slow growth function are shown in Figure 1.

The minimal-growth fitness function is similar but it takes place with 600 cells. This function returns the total number of cells that were ever active in the evaluation strip as the fitness so long as the automata dies before it reaches the 601st evaluation. Otherwise it returns a fitness of zero. This evaluation function is much faster to compute and it enforces no more than 600 updatings. Examples of automata evolved with the minimal growth function are shown in Figure 2.

The decision to use six cell states was made to provide a rich space of rules. With six states and a neighborhood of 3 cells the maximum neighborhood sum is 15 yielding (with the requirement that \[0][0][0]\] map to \[0]\] a total of \(6^{15} \approx 4.70 \times 10^{15}\) rules. In order to render the automata as images we use the mapping given in Table I.

<table>
<thead>
<tr>
<th>Cell State</th>
<th>Color</th>
<th>(R,G,B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>White</td>
<td>(255,255,255)</td>
</tr>
<tr>
<td>1</td>
<td>Blue</td>
<td>(0,0,255)</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>(0,255,0)</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>(255,0,0)</td>
</tr>
<tr>
<td>4</td>
<td>Yellow</td>
<td>(255,255,0)</td>
</tr>
<tr>
<td>5</td>
<td>Magenta</td>
<td>(235,0,235)</td>
</tr>
</tbody>
</table>

B. Evolutionary Algorithm Design

Other than using the fitness functions defined in Section II-A, the evolutionary algorithm in this study is a standard one. The cellular automata rules are stored as strings of 16 integers with values in the range 0-5, corresponding to the cell states. Two variation operators are used: two point crossover of the string of values and single point mutation that replaces the value at a single position selected uniformly at random. Selection and replacement are accomplished with generational size-four tournament selection. The population is shuffled into groups of four CA-rules. The two more fit are copied over the two less fit. The copies are subjected to crossover and mutation.

Three sets of experiments, each consisting of 30 evolutionary replicates, are run for each fitness function. These experiments are a parameter study to estimate the correct population size. Population sizes of 12, 120, and 1200 are used. The number of generations are adjusted to keep the total number of fitness evaluations roughly constant. When the population size is 1200, 100 generations are used. For 120 member populations, 10000 generations are used. For 12 member populations, 10000 generations are used. It is only “roughly” constant because the larger populations gain a small advantage - only the number of tournaments is kept constant; fitness evaluations for the initial population are not counted.

Using a small population size can result in an enhancement in both the amount of hill climbing and the amount of genetic drift throughout evolution. In [4] it was found that a number of genetic programming tasks [9], [10], [5], including the parity problem, PORS, and Tartarus[1] exhibited small population effects: performance was improved by using a small population. In the case of parity it was found that reducing the population size to ten (from hundreds) improved time-to-solution by an order of magnitude. For PORS the probability of premature convergence was found to decrease by a statistically significant degree when small populations were used. It was also found that small populations, used with the correct representation, made progress on the Tar-tarus task at an enhanced rate. We call such results small population effects. Our group routinely tests small population sizes on new problems, hence the use of populations of size 12.
The most startling result in [15] was the 10,000-fold increase in numerical stability of simulated Newtonian dynamics obtained in the best evolutionary outputs. For the slow-growth fitness function the best fitness located was 6951 in the 1200-member population experiment. This gives a maximum enhancement over the mean fitness of a starting population of 140-fold. For the minimal-growth experiments the best fitness obtained is 30,645 yielding an enhancement over mean initial fitness of 8756. This latter result is not as large as, but close to, the enhancement achieved in the earlier study stabilizing numerical integration of Newtonian dynamics. This demonstrates a second problem in which evolutionary control of a dynamical system yields many-

III. RESULTS AND DISCUSSION

A 95% confidence interval on the mean fitness of a random cellular automata rules for the slow-growth experiment, derived from a random sample of 100,000 genes is (49.47,49.86). The modal fitness of random genes is 49 because most cellular automata grow at a rate of one pixel width per time step. The next most common outcome is a fitness of 0 of genes that fail to reach the edges of the evaluation strip. A similar confidence interval for the minimal-growth experiment yielded a 95% confidence interval on mean fitness of random genes of (3.06,4.04). Almost all genes fail to die by the end of the evaluation strip and so have fitness zero.

Fig. 3. Maximum and 95% confidence intervals of the mean fitness, averaged over all 30 replicates for the slow-growth experiment. The top panel shows results for population size 1200, the middle 120, the bottom 12. Note that generations are scaled to population size for comparison.

Fig. 4. Maximum and 95% confidence intervals of the mean fitness, averaged over all 30 replicates for the minimal-growth experiment. The top panel shows results for population size 1200, the middle 120, the bottom 12. Note that generations are scaled to population size for comparison.
We now look at the impact of population size. Figure 3 shows the average behavior of fitness over the course of evolution for the slow-growth experiments, while Figure 4 does the same for the minimal-growth experiments. For both fitness functions there is a clear benefit to using a larger population. The small populations are of uniformly lower fitness than the largest. Another feature obvious in both set of experiments is that the average (over replicates) maximum fitness is remarkably higher for the largest population size used.

Turning to the trend of fitness over evolution, the slow-growth experiments show a modest upward trend in both mean and maximum fitness at the end of evolution. This is most clearly visible for the largest population size. In the minimal-growth experiments, the small and middle population size experiments appear to have reached a plateau while the largest population size experiments are continuing to trend upward. This suggests that there are fairly broad local optima in the adaptive landscape that a larger population avoids more efficiently. This leads to the question of the roughness of the fitness landscape.

In Figure 5 the mean trajectory of best fitness for the slow-growth experiments with population size 1200 is unpacked to show the individual tracks, left unlabeled. The shape of the curves is revealing: maximum fitness does not increase steadily, nor does it increase in small jumps. Rather there are huge leaps in the value of the fitness function. The abruptness of many of the transitions is somewhat masked by the 5-fold increase, to 6951, in one of the tracks. Hand examination of some of the high fitness rules shows that many of their mutations have fitness 0 (they die or fail to grow) or 49 (they grow at the maximum possible rate of 1:1). This is evidence of an extremely rough fitness landscape.

IV. CONCLUSIONS AND NEXT STEPS

This study demonstrates two fitness functions for evolving simple automata that locate automata, examples of which are shown in Figure 1 and 2 with interesting appearances. While it is clear that more care could be taken in choosing the colors used to render the automata, many of them exhibit complex and intriguing dynamics. The two fitness functions encourage almost opposite behaviors: slow growth for an enormous number of updatings of the automaton and automatic senescence in which the automaton blooms and then dies in a manner determined entirely by its internal dynamics. Both fitness functions are highly optimized by the same evolutionary algorithm. Since the algorithm is quite simple, this suggests that the rich space of cellular automata selected, the representation used, is responsible for the success.

The second goal of this study was to check if a very different type of dynamical system from the one stabilized in [15] could also be controlled to exhibit long term stability. In this study the definition of stability used was (i) avoiding the edges of the evaluation strip while nevertheless continuing to grow or (ii) exhibiting substantial but bounded growth followed by cessation of dynamics based only on internal model behavior. For the slow-growth fitness function an improvement of 140-fold in stability was found. For the minimal-growth function an improvement of 8756-fold was found. This suggests that evolution was able to find very stable rules for the discrete dynamical systems (cellular automata) in a very similar manner to which it found stable initial conditions for continuous dynamical systems in the earlier study. It is worth noting that, to the degree they were thousand fold increases in fitness.
assessed, the fitness landscapes for initial conditions for the continuous dynamical system and for both fitness functions used on discrete dynamical systems in this study were found to be quite rough.

The difference in magnitude of improvement between the slow-growth and minimal-growth functions may be the result of how they were evaluated. The minimal growth has a hard-coded maximum value of 10,100, the point after which the automaton was estimated to be not growing at all and awarded a fitness. Given that a fitness of 6951 was located with a roughly five-fold increase in maximum fitness in a single generation, it is quite likely that slow growing genes that took far longer than 10,100 generations were evolved and discarded after they were mistaken for non-growing rules. The limit of 10,100 was chosen arbitrarily and should be revisited in future studies.

V. Next Steps

There are several natural directions in which to generalize this research. The simplest is to check how enhancing or degrading the richness of the cellular automata rule set impacts the behavior of the system. This is easily done by changing the number of cell-states or the size of the neighborhoods used to update the automata. Increasing neighborhood size would permit rules to stay complex while decreasing the number of cell states. In a similar direction, a more complex class of rules than simply totalistic ones could be implemented. Such asymmetric rules would yield a far richer space of rules.

In [2] the authors note that local optima in the search space of an aesthetic fitness function may be more visually appealing than global optima. At present the evolutionary algorithm maximized the two different fitness functions defined in this study. By simply creating a new fitness function:

$$f'(CA - \text{rule}) = |f(CA - \text{rule}) - C|$$

(1)

to be minimized, for a desired level of fitness $C$ would cause the algorithm to focus on parts of the original fitness landscape near height $C$.

The two fitness functions used in this study both sort cellular automata by fitness in a manner based on very simple metrics of their performance: how fast do they grow and for how long do they stay active. Following ideas from [3] another approach might be to give the automata templates to match. A pattern, imposed on the evaluation strip, with the complex images arising as a time history. While requiring a substantial increase in the amount of computation required it would be simple to generalize the system to three dimensions in which the image is a solid rendering of the time history of a two dimensional automata. The number of possible shape-based constraints for such a system is much larger than in a one-dimensional automata rule space.

Finally, the very close correspondence in fitness landscapes between a discretized continuous system and an inherently discrete system bears closer investigation. A broad survey of different dynamical systems to determine whether those also exhibit this sharp structure could be done.

References


