Evolvable Fashion-based Cellular Automata for Generating Cavern Systems

Daniel Ashlock

***Abstract*—Cellular automata can be used to rapidly generate complex images. This study introduces *fashion-based* cellular automata that generate cavern-like level maps. Fashion-based automata are defined by a competition matrix that defines the benefit to a given cell state of having a neighbor of each posible cell state. A simple fitness function permits this type of automata to be evolved to produce a variety of level maps. A parameter study is performed and a variety of level maps are evolved with a toroidal grid, ensuring that the level maps tile. The parameter study demonstrates a robustness of the fashion based representation to the variation of parameters. The appearance of a given cavern-like level is encoded in the evolved automaton rule permitting the creation of many levels with a similar character simply by varying initial conditions. The cellular automata rules function in local neighborhoods meaning that the level generation system scales smoothly to any desired level map size.**

I. Introduction

This study builds on earlier work using a cellular automaton to design level maps resembling a network of caverns [14]. A system for automatic content generation (ACG) of level maps with cellular automata is developed. Cellular automata instantiate discrete models of computation. A cellular automaton has three components:

1) A collection of cells divided into neighborhoods of each cell,
2) A set of states that cells can take on,
3) A rule that maps the set of possible cell states of a neighborhood to a new state for the cell with which the neighborhood is associated.

In practice, CA are a type of discrete dynamical systems that exhibit self-organizing behavior. When a cell population is updated according to local transition rules, it can form complex patterns. The updating may be synchronous, as it is in this study, or asynchronous. CA are potentially valuable models for complex natural systems that contain large numbers of identical components experiencing local interactions[27], [18]. This paper applies them to create cavern-like level maps for games. Examples of the technique are shown in Figure 1. The automata in this study are called *fashion-based* cellular automaton because the updating rule may be thought of as following the current fashion within each neighborhood.

Daniel Ashlock is with the Department of Mathematics and Statistics at the University of Guelph, in Guelph, Ontario, Canada, N1G 2W1, dashlock@uoguelph.ca

The authors thank the University of Guelph and Canadian Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this work.

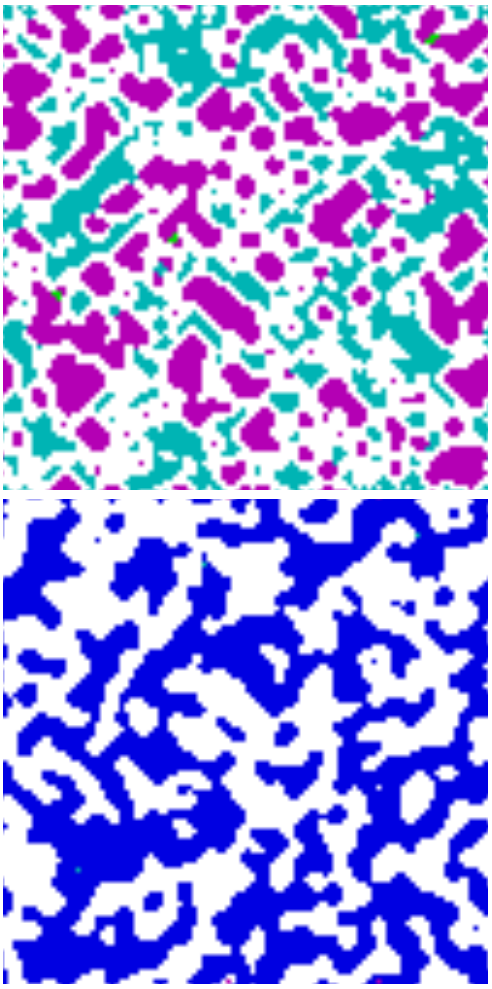

Fig. 1. Examples of cavern-like level maps drawn by evolved cellular automata.

CA have been applied to the study of a diverse range of topics, such as structure formation[10], heat conduction[11], language recognition[17], traffic dynamics[15], modeling of biological phenomena [13], and cryptography[2], to name a few. CA have also been used for image and sound generation. Serquera and Miranda of the Interdisciplinary Center for Computer Music Research, UK, have many pulications on the use of CA for sound synthesis [19], [1]. Much of their work consists of mapping the histogram sequence of a CA evolution onto a sound spectrogram, which produces spectral structures that unfolds in a patterned fashion over time. The authors claim that the mapping produces a “natural”

behavior, and can replicate acoustic instruments[20].

CA have also been applied in the arts. They have been used to produce artistic images[9], [16], and their use has been extended to the fields of architecture and urban design[21], [12]. 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[22]. Ashlock and Tsang[9] produced evolved art using 1-dimensional CA rules. These systems produced aesthetically pleasing images. In [8] a good deal of information about the fitness landscape of a particular type of cellular automata was derived.

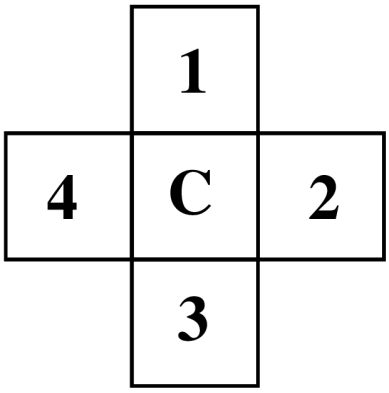

Fig. 2. A von Neumann neighborhood of the cell **C**. The members of the neighborhood are the cells are those numbered 1, 2, 3, 4.

This study introduces a novel type of cellular automata, called a *fashion-based* cellular automata, that is relatively easy to represent for evolution. The CA uses a toroidal two-dimensional grid as its cell space. The automata uses *von Neuman* neighborhoods, of the sort shown in Figure 2. The automaton is synchronous with an updating rule is governed by a score matrix that gives the benefit a cell has from having neighbors of each possible cell type.

A. Previous Work in Level Map ACG

In addition to the study [14] mentioned above there have been a number of efforts to use evolutionary computation to automatically design level maps. Procedural content generation (PCG) consists of finding algorithmic methods of generating content for games. Search based PCG [26] uses search methods rather than composing algorithms that generate acceptable content in a single pass. Both sorts of content generation can suffer materially from scaling problems which can be addressed by problem decomposition with an off-line and an on-line phase to the software.

Automated level generation in video games can arguably be traced back to a number of related games from the 1980s (Rogue, Hack, and NetHack), the task has recently received some interest from the research community. In [23] levels for 2D sidescroller and top-down 2D adventure games are automatically generated using a two population feasible/infeasible evolutionary algorithm. In [25] multiobjective optimization is applied to the task of search-based procedural content generation for real time strategy maps. This study gives an alternate representation for tasks similar to those done in [4] which introduced checkpoint based fitness functions for evolving maze-like levels. This work was extended in [5] by having multiple types of walls that defined multiple mazes that co-existed in a single level map. Related work includes [6] which prototyped tile assembly to generate large maps and [7] which gave a state conditioned representation that could generate level maps of landscape height maps.

The remainder of this study is structured as follows. Section II supplies the background on the representation used to evolve fashion-based cellular automata and discusses a novel evaluation tool. The design of experiments is given in Section III while section IV gives and discusses results. Section V draws conclusions and outlines potential next steps.

II. Background

We begin by precisely defining fashion-based cellular automata. The automata in this study have a cell space consisting of 100×100 grids. The left and right and the top and bottom sides of the grid are considered to be adjacent making the grid toroidal. The use of a toroidal grid means that the resulting pictures will tile correctly without edge artifacts. The neighborhoods are von Neumann neighborhoods of the sort shown in Figure 2. The number of cell states used is $n = 6$, a number chosen by preliminary experimentation. The rule is specified as a $n \times n$ real matrix M, indexed by cell states, with entry $M_{i,j}$ giving the score a cell with state i gets if it has a neighbor in state j.

The automata is updated synchronously. The updating rule computes the score of each cell, based on its state and the states of its neighbors. Each cell then either stays in the same state, if its score is at least as high as those of its neighbors, or adopts the state of its highest scoring neighbor if that score is higher than its own. This is though of as "following the fashion" of the neighborhood. Fashion-based automata leave homogeneous regions homogeneous, a property that makes the space of rules rich with automata that generate cavern-like levels.

A. Representation

With $n = 6$ cell states, the rule is given by a 6×6 real matrix that is specified by 36 real parameters. This is encoded by a vector of 36 real numbers in the interval [0,2.0]. Since scoring is relative, the range of values is not critical. The evolutionary algorithm uses two point crossover. Point mutation is performed by modifying one of the numbers by adding a uniformly distributed random variable in the interval $[-\epsilon, \epsilon]$ with $\epsilon = 0.1$. If a value leaves the interval [0,2] as the result of mutation a new value is generated uniformly at random in the range [0,2]. The entries of the vector are the rows of the matrix, meaning that contiguous groups of 6 entries specify the scores one cell state obtains when matched against others.

B. Evaluation Tools

When using evolutionary computation as an optimizer it is difficult to determine if the algorithm has been run for a sufficient time. In this study the evolutionary algorithm was permitted 20,000 fitness evaluations (two per mating event). Eleven different sets of parameters were run. It would be interesting to have some idea which were closer to having converged. The *time of last innovation* (TLI) assessment can provide some perspective on this issue.

For all of the experiments we examine the fitness tracks of the experiments and compute the last time the current best fitness changes. This time of last innovation is then divided by the time the algorithm ran to obtain a *fractional time of last innovation*. Figures 3, 4, and 5 include results of the TLI evaluation. The TLI statistic serves as a much more compact way of displaying information about algorithm innovation behavior as opposed to the more traditional maximum-fitness-over-time plots, commonly used to assess if an algorithm was run long enough.

III. Design of Experiments

The evolutionary algorithm used is steady state [24]. The model of evolution used is size seven single tournament selection. In this model of evolution, seven members of the population are selected uniformly at random. The two most fit members of this group are copied over the two least fit and the copies are subjected to crossover and mutation. Mutation consists of a number of point mutations selected uniformly at random in the range 1 to M, the *maximum number of mutations*. The default value for the maximum number of mutations is $M = 3$.

The algorithm operates on a population of cellular automata rules with a default size of $P = 60$. A single set of initial conditions on a 100×100 grid is filled in uniformly at random and then saved for used in all fitness evaluations. Automata are evaluated for a default of $T = 20$ updatings of the cell states and then fitness is assessed. A parameter study, varying the default values, was performed. The values used, varying one value at a time, are taken from $M = 1$, 3, 5, and 7; $T = 10$, 20, 30, 40, and 50; P=10, 60, 360, and 2160.

The automata is run for 10,000 instances of tournament selection, called *mating events*. Every 100 mating events summary fitness statistics, including mean, variance, and maximum are saved, as is the fraction of the population that has fitness zero. The definition of the fitness function will explain how these zero-fitness individuals arise.

A. The Fitness Function

After T updatings of the fashion-based cellular automata rule from the fixed initial state the cell states of the automata are used to evaluate fitness. Before fitness is evaluated and single application of a majority rule, like that used in [14], is applied to eliminate isolated pixels. This has little effect on the fitness but yields a better looking level map. The state 0 is taken to represent empty space, all others are taken to represent obstructed cells. Using a recursive fill from a cell in the center of the state space, the number N of empty cells that can be reached from that central cell are computed. The fraction of unobstructed cell states U, accessible or not, are also computed. The fitness of an automaton rule is:

$$fit = \frac{N}{1 + |2 * U - 1|} \tag{1}$$

This function rewards cells accessible from the central cell of the grid and penalizes the grid if it does not have half its cells in state 0 (unobstructed). This encourages a half-unobstructed cell space much of which is connected. If the central cell used for the recursive fill is obstructed then a rule receives a fitness of zero. Evolution rapidly eliminates such individuals.

TABLE I
SHOWN ARE THE VALUES FOR THE EXPERIMENTAL PARAMETERS USED IN EXPERIMENTS. THE PARAMETERS VARIED ARE: MAXIMUM NUMBER OF MUTATIONS M; AUTOMATA UPDATINGS T; POPULATION SIZE P.

Experiment	M	T	P
1	1	20	60
2	3	20	60
3	5	20	60
4	7	20	60
5	3	10	60
6	3	30	60
7	3	40	60
8	3	50	60
9	3	20	10
10	3	20	360
11	3	20	2160

B. Experiments Performed

A collection of eleven experiments, each consisting of thirty independent runs of the evolutionary algorithm, were performed. The parameter values used in these experiments are shown in Table I. The experiments are based on the default parameter values $M = 3$, $T = 20$, and $P = 60$, chosen with preliminary experimentation during testing and debugging of the code. The experiments form three parameter studies. Experiments 1, 2, 3, and 4 compare different maximum mutation rates; experiments 2, 5, 6, 7, and 8 compare different numbers of updatings of the fashion-based cellular automata during fitness evaluation; experiments 2, 9, 10, and 11 compare different population sizes. The default parameters are those used in Experiment 2 which is why it appears in all three parameter studies.

IV. Results and Discussion

Examining Figure 3 it is not difficult to see that the maximum number of mutations is a relatively soft parameter. Juxtaposing the fitness information with the TLI data it is clear that the algorithm converges sooner when the

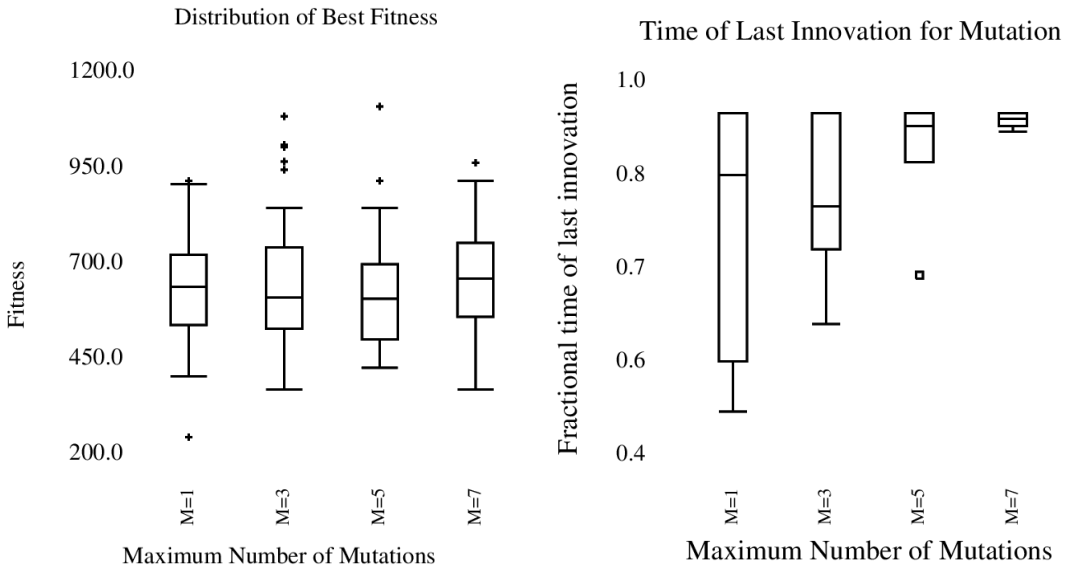

Fig. 3. Shown are box plots of the distribution of best fitness values and the fractional times of last innovation for the mutation rate parameter study.

maximum number of mutations is lower. This suggests that the algorithm is undergoing greater exploration when the number of mutations is higher (not surprising), but given the lack of impact on fitness that the exploration is not particularly useful. It probably consists of minor variations that make a few more cells accessible. The best balance of final results and convergence weakly supports the default choice of $M = 3$.

When examining the number of updatings of the cell states of the cellular automata during fitness evaluation, there is a strong motive to favor smaller values of T: time to evaluate fitness varies directly as T. Given that there is a lack of fitness variation between the different values of T, there is a temptation to set $T = 10$. Examining the TLI data shows that $T = 30$ has anomalously early convergence - probably an actual anomaly and something that should be checked in the future. The results for $T = 10$ show the worst convergence behavior - suggesting that the randomness in the initial conditions is still strongly influencing the final cave-like level that evolves. This leave $T = 20$ as the best choice, balancing second lowest fitness evaluation cost with good convergence behavior. It is also worth noting the high-fitness outliers for $T = 20$.

The only parameter for which the fitness data returned significant differences is the population size study which provided clear evidence that a very large population is not a good idea. The best performance was turned in by the default value of $P = 60$ and, of the three populations sizes with very similar fitness results, had the best convergence behavior as measured by the TLI results. Overall the three parameter studies support the chosen default values $M = 3$, $T = 20$, and $P = 60$ in the weak sense that no better values are available from the perspective of fitness and the chosen values exhibit superior convergence behavior.

In the experiments performed in this study, a single set of initial conditions was used for all fitness evaluations. This was done to permit repeatability, from a random number seed, of the experiments in case they needed to be rerun. The experiments used a 100×100 cell space meaning that one set of initial conditions is statistically similar to another, differing only in details. This, in turn, has an additional advantage. Once an automata rule has been located that creates a level with a desirable character then any number of levels with a similar character may be generated by generating new initial conditions. Examples of eight similar level maps for two different evolved automata rules are shown in Figure 6. In addition to being able to generate multiple maps from any given rule, the fact that the cellular automata act locally by neighborhoods means that we can change the size of the grids and get larger of smaller maps. Figure 8 shows an example of this.

The algorithm located a broad diversity of results. A sampling of the cave-like levels evolved are shown in Figure 7. Monochrome or almost monochrome levels are the most common and are more common for larger values of T. The last image in the second row is unusual in having horizontal and vertical corridor-like structure. All of the structures in the second row have a more block-like structure which is relatively rare. The last two rows of Figure 7 were chosen for their polychromatic character.

V. Conclusions and Next Steps

This study demonstrated that fashion-based cellular automata can evolve a broad variety of cavern-like level maps. The parameter study yielded little practical advice, other than avoiding very large population sizes, and demonstrated that the system - at least when run with 10,000 mating events, is relatively robust to the choice of parameters. Taken as a whole, the TLI data suggest that, good results

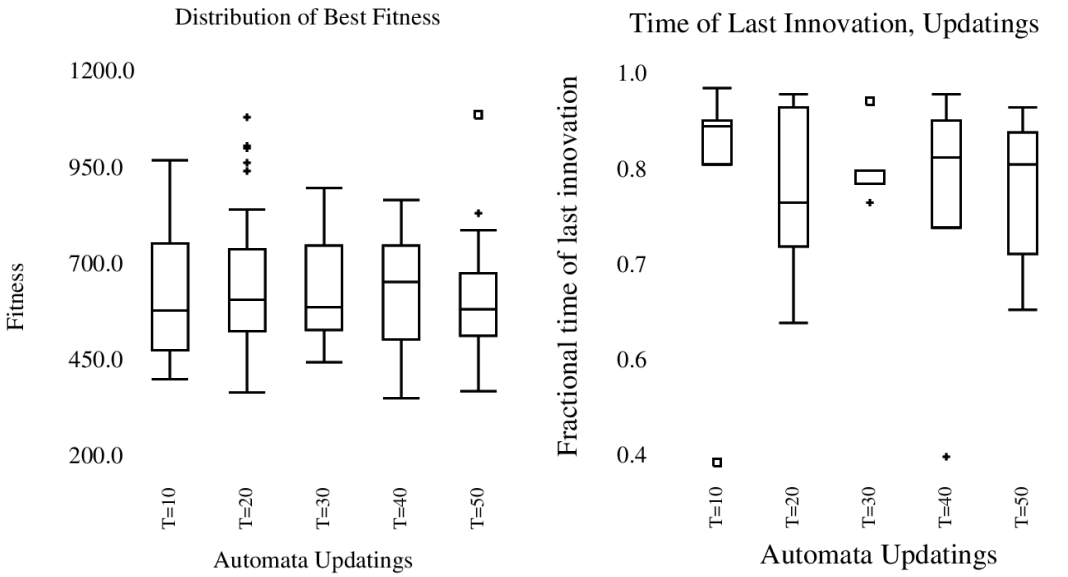

Fig. 4. Shown are box plots of the distribution of best fitness values and the fractional times of last innovation for the number of automata updatings parameter study.

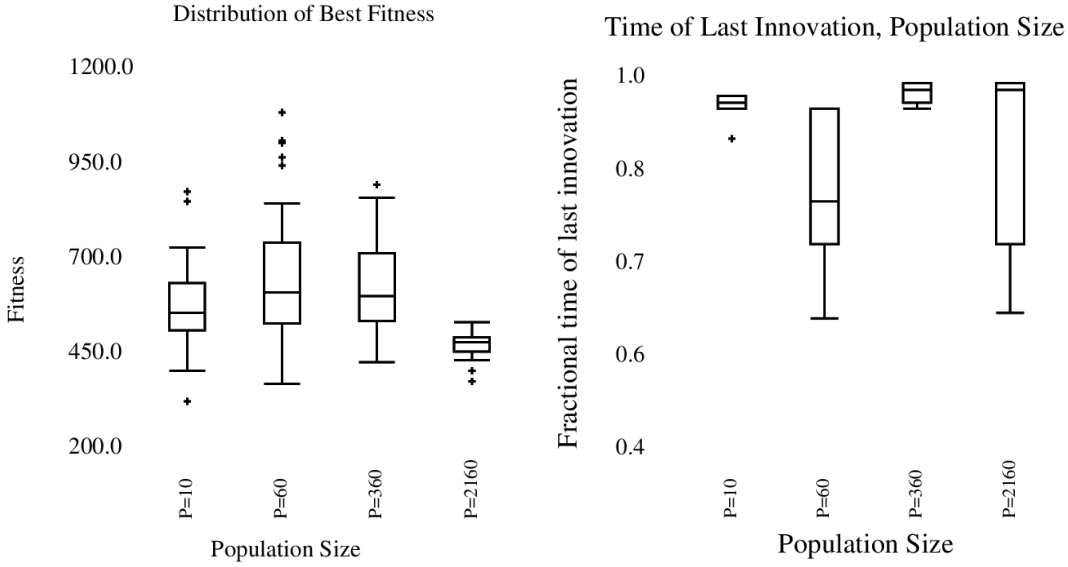

Fig. 5. Shown are box plots of the distribution of best fitness values and the fractional times of last innovation for the population size parameter study.

notwithstanding, running the system for a longer time might yield better fitness values.

This raises another issue. The fitness function was chosen to reward having a large connected component and have the space in a level map 50% obstructed. This fitness function in no way encodes the idea of having a "cool looking" map. This means that, as with many evolutionary design systems that perform ACG, the local optima of the fitness landscape may be more desirable from the perspective of a game designer than global optima.

The use of six states, only one representing empty space, was a choice made to give the algorithm greater expressive power. If a level map requiring only one type of rock is needed then the solid states can simply be merged into a single color. In many cases the obstructions of different color could represent different types of obstructions: rock, water features, pits, or even pools of lava. Fully exploiting this potentiality would require writing more complex fitness functions.

An early priority for additional research is to investigate the impact of changing the number of cell states. This study did not have room to encompass this variation of the system and so six were chosen as being a relatively high number that enables expressibility. Two or more cell states would permit the system to function; larger numbers of cell states enable enormously larger spaces of rules.

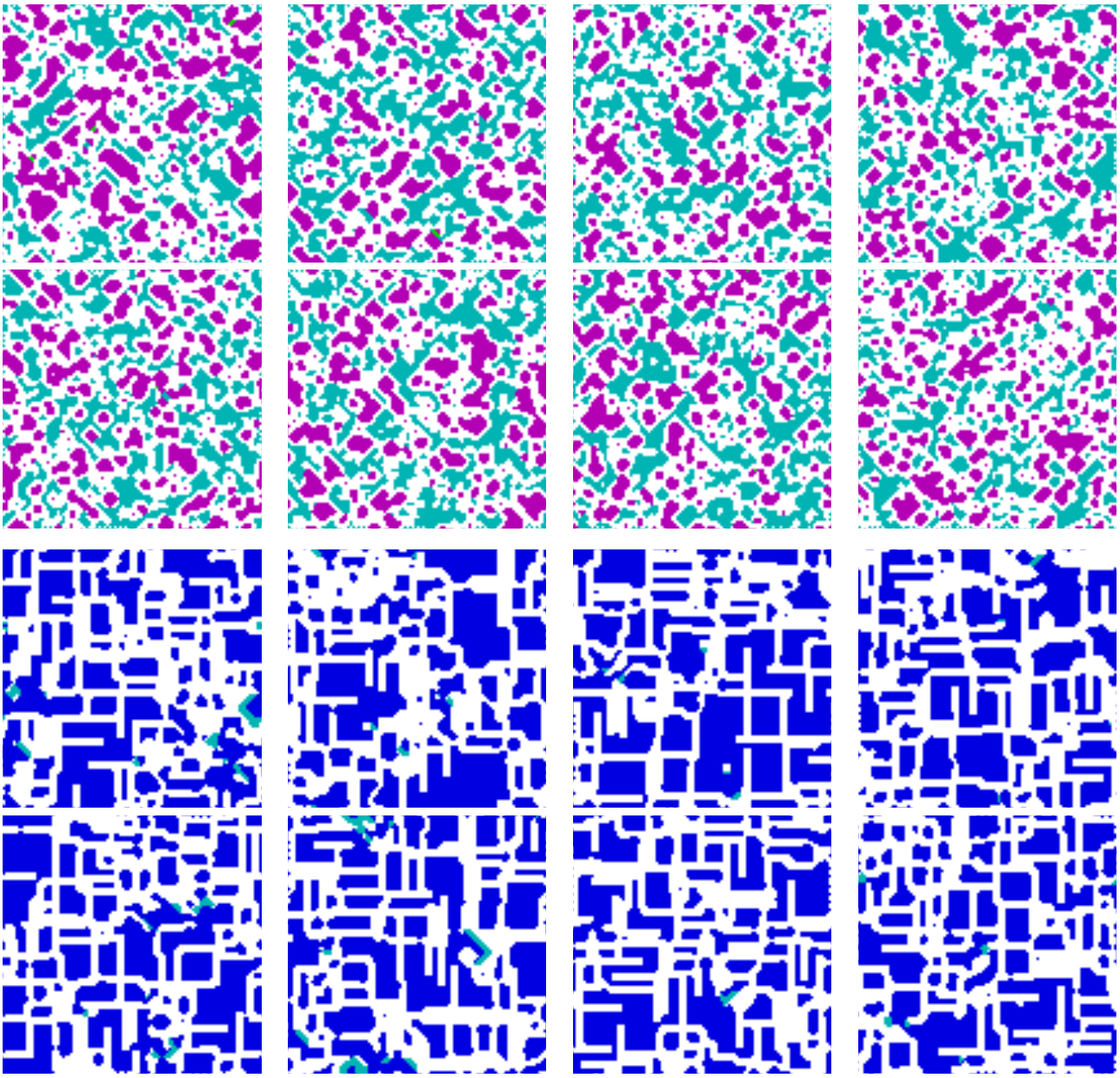

Fig. 6. Examples of final cell states for eight distinct initial conditions using the highest fitness fashion-based cellular automaton from Experiment 2 and an interesting rule that yields horizontal and vertical corridors.

A. Other Fitness Functions

The fitness function used in this study was designed in an ad-hoc manner and produced a very broad variety of cavern like level maps. It rewards a level with a large connected component and which is as close to 50% obstructed. If α is the desired fraction of open cells, and recalling that N is the size of the connected component and U is the fraction of unobstructed cells the equation:

$$fit_\alpha = \frac{N}{1 + |U/\alpha - 1|} \tag{2}$$

is a modification of the fitness function that rewards having an α fraction of the cells unobstructed.

In [3] a fitness function for mazes was used that maximized the distance from an entrance to an exit. It seems likely that this would be a challenging fitness function for the fashion-based cellular automata representation. In [6] a variation of this function with more than two entrances/exits was used to create tiles that were combined to form levels. This, also, would create interesting maze-like levels. Following the dual mazes created in [5] there is substantial potential for creating fitness functions that use the fact there are different types of obstructions.

B. Leveraging Evolved Automata Rules

The automata used in this study update a grid of cell values. By changing the initial conditions and grid size these

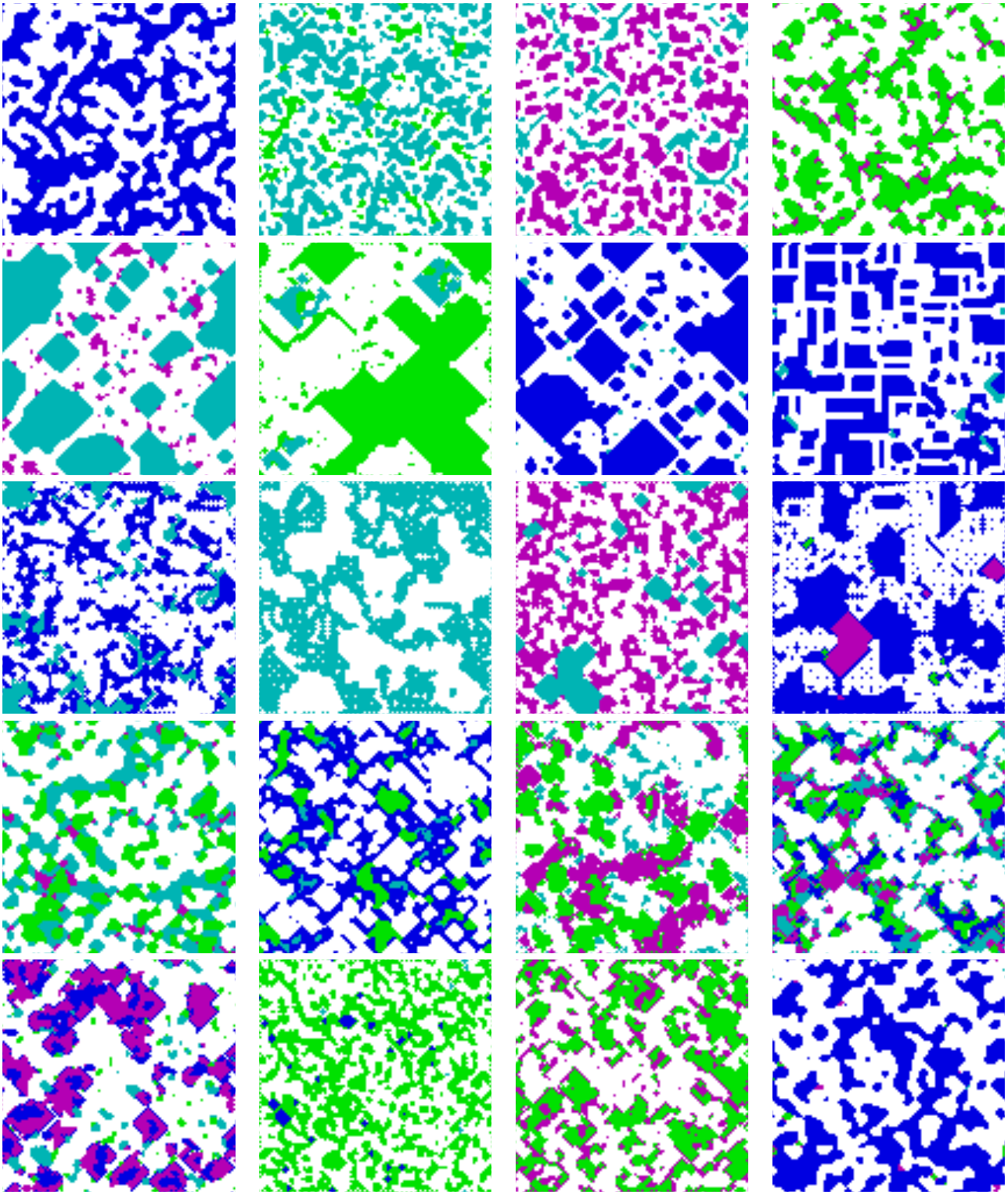

Fig. 7. Examples of final cell states rendered as level maps.

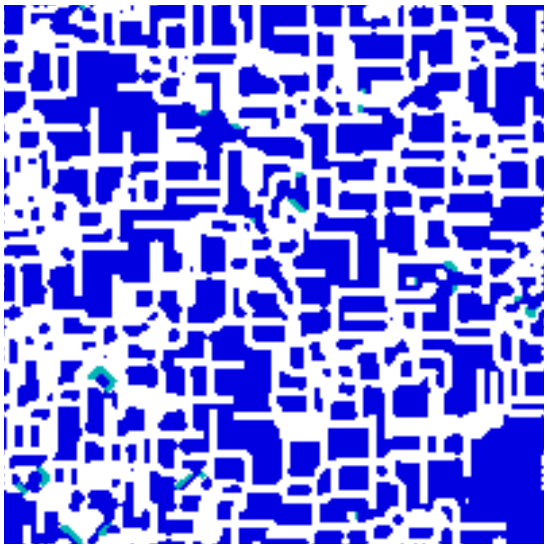

Fig. 8. A 200×200 version of the second type of automata rule shown in Figure 6.

rules can be used to generate a large variety of different cavern systems. The rules require only 36 real numbers to specify. There are two ways, other than crossover, to use evolved rules to generate more rules.

If M and Q are both matrices that specify automata rules then so is

$$(1 - \lambda)M + \lambda Q$$

Values of λ in the interval [0,1] create an interval of automata that can be checked to see what sorts of level maps they create. Since updating is based on comparison, the interval will contain sub-intervals of nearly identical automata rules.

Another way of combining rules is to use a linear representation to specify an order in which to apply rules. A gene 13233211231331322211 would specify to used rule 1, then rule 3, then rule 2, and so on. If the rules used as "characters" of the linear gene make cavern-like levels then the combined rule may well have an enhanced probability of generating high-quality caverns.

Both these methods of re-using evolved rules have nugatory additional computational cost to generate a level map. If, as expected, they have enhanced probabilities of generating cavern-like maps then these techniques will permit far more efficient methods of generating level maps. It is likely that maps made by combining evolved rules will share some of the appearance of their progenitors.

REFERENCES

[1] A. Adamatzky, J. Serquera, and E.R. Miranda. *Automata-2008: Theory and Applications of Cellular Automata: "Cellular automata sound synthesis: From histograms to spectrograms"*. Luniver Press, 2008.

[2] P. Anghelescu. Encryption algorithm using programmable cellular automata. *IEEE 2011 World Congress on Internet Security (WorldCIS)*, pages 233 – 239, 2011.

[3] D. Ashlock. Automatic generation of game elements via evolution. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games*, pages 289–296, 2010.

[4] D. Ashlock, C. Lee, and C. McGuinness. Search based procedural generation of maze like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011.

[5] D. Ashlock, C. Lee, and C. McGuinness. Simultaneous dual level creation for games. *Computational Intelligence Magazine*, 2(6):26–37, 2011.

[6] D. Ashlock and C. McGuinness. Decomposing the level generation problem with tiles. In *Proceedings of CEC 2011*, pages 849–856, 2011.

[7] D. Ashlock and C. McGuinness. Landscape automata for search based procedural content generation. In *Proceedings of IEEE CIG 2013*, pages 9–16, 2013.

[8] D. Ashlock and S. McNicholas. Fitness landscapes of evolved cellular automata. *IEEE Transaction on Evolutionary Computation*, 17(2):198–212, 2013.

[9] D. Ashlock and J. Tsang. Evolved art via control of cellular automata. In *IEEE Congress on Evolutionary Computation, 2009*, pages 3338 – 3344, May 2009.

[10] A.A. Burbelko, E. Fras, W. Kapturkiewicz, and D. Gurgul. Modelling of dendritic growth during unidirectional solidification by the method of cellular automata. *Materials Science Forum*, 649:217–222, 2010.

[11] A.A. Burbelko and D. Gurgul. Simulation of austenite and graphite growth in ductile iron by means of cellular automata. *Archives of Metallurgy and Materials*, 55(1):53–60, 2010.

[12] M. Devetakovic, L. Petrusevski, M. Dabic, and B. Mitrovic. Les folies cellulaires an exploration in architectural design using cellular automata. *12th Generative Art Conference*, pages 181–192, 2009.

[13] G. B. Ermentrout and L. Edelstein-Keshet. Cellular automata approaches to biological modeling. *Journal of Theoretical Biology*, 160(1):97 – 133, 1993.

[14] Lawrence Johnson, Georgios N. Yannakakis, and Julian Togelius. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, pages 10:1–10:4, New York, NY, USA, 2010. ACM.

[15] M. E. Lrraga and L. Alvarez-Icaza. Cellular automaton model for traffic flow based on safe driving policies and human reactions. *Physica A*, 389(23):5425–5438, 2010.

[16] G. Monro. Emergence and generative art. *Leonardo - MIT Press*, 42(5):476–477, 2009.

[17] K. Nakamura and K. Imada. Incremental learning of cellular automata for parallel recognition of formal languages. In *Proceedings of the 13th international conference on Discovery science*, DS'10, pages 117–131, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] E. Sapin, O. Bailleux, and J. Chabrier. Research of complexity in cellular automata through evolutionary algorithms. *Complex Systems*, 11, 1997.

[19] J. Serquera and E. R. Miranda. Cellular automata sound synthesis with an extended version of the multitype voter model. In *Audio Engineering Society Convention 128*, 5 2010.

[20] J. Serquera and E.R. Miranda. *Applications of Evolutionary Computation: "Evolutionary Sound Synthesis: Rendering Spectrograms from Cellular Automata Histograms"*. Springer Berlin / Heidelberg, 2010.

[21] V. Singh and N. Gu. Towards an integrated generative design framework. *Design Studies*, in press, 2011.

[22] H. Situngkir. Exploring ancient architectural designs with cellular automata. *BFI Working Paper No. WP-9-2010*, 2010.

[23] N. Sorenson and P. Pasquier. Towards a generic framework for automated video game level creation. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, volume 6024, pages 130–139. Springer LNCS, 2010.

[24] Gilbert Syswerda. A study of reproduction in generational and steady state genetic algorithms. In *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[25] Julian Togelius, Mike Preuss, and Georgios N. Yannakakis. Towards multiobjective procedural map generation. In *PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–8, New York, NY, USA, 2010. ACM.

[26] Julian Togelius, Georgios Yannakakis, Kenneth Stanley, and Cameron Browne. Search-based procedural content generation. In *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 141–150. Springer Berlin / Heidelberg, 2010.

[27] S. Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.