Automatic Generation of Diverse Cavern Maps with Morphing Cellular Automata

Matthew Kreitzer, Daniel Ashlock, and Rajesh Pereira

***Abstract*—Cellular automata can be used to rapidly generate complex images, but controlling the character of those images can be difficult. This study continues experimentation with *fashion-based* cellular automata that generate cavern-like level maps and provides the beginning of a mathematical theory. Fashion-based automata are defined by a competition matrix with different cell states competing to capture territory. This study co-evolves pairs of competition matrices to permit the evolution of automata rules that can be spatially morphed to provide substantially more diverse types of maps than earlier systems using fashion-based cellular automata. As in earlier studies, the cellular automata rules function in local neighborhoods, meaning that the level generation system scales smoothly to any desired level map size. This reusability also permits variation of the type of morph used: a variety of spatial morphing styles are tested with the evolved rules. The theoretical treatment includes the derivation of a normal form for the cellular automata rules that informs the design of the fitness function and has application to understanding the fitness landscape of fashion based automata.**

I. Introduction

This study continues earlier work using a cellular automaton to design level maps resembling a network of caverns [4]. The properties that make this representation for automatic content generation of level maps desirable include a very high level of scalability, transparent reusability, and, when morphing is enabled, diversity of type of territory within maps. *Morphing*, the continuous change of the cellular automaton rule across space, is a technique for substantially enhancing the diversity of features within a map. The rules for the cellular automata used in this study take the form of square matrices with real-valued entries. These matrices are *competition matrices* that give a score for each possible cell/state pair in the cellular automata.

If M and N are such competition matrices, then we consider the line of matrices

$$\lambda \cdot M + (1 - \lambda) \cdot N \;\; 0 \leq \lambda \leq 1$$

This line, in matrix space, represents a continuously varying collection of automata rules. If we compute λ from a spatial feature – like the horizontal coordinate on the map – then we can create a continuously varying rule that permits a different appearance in the map as different locations as λ changes. This is what is meant by a *morph* between two matrices. An example of a morph is shown in Figure 2. If we choose two matrices that yield good cavern maps, that were evolved independently, then the morph between those matrices usually contains solid barriers or completely empty regions.

In this study, pairs of cellular automaton rules are co-evolved based on the quality of a map generated by their horizontal morph. This co-evolution is needed to allow the rules to work together effectively.

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.

The idea of using a cellular automata to create cavern maps originated in [19]. The automata used here employed a majority based rule.

The use of fashion based cellular automata, the sort which appear in this study, originated in [3]. This study was extended to explore additional control parameters in [4] where the idea of performing morphs between cellular automata rules was first tested, highlighting the need for later studies to co-evolve rules. This study, working on pairs of independently evolved rules, found that morphing between distinct, evolved rules was problematic. In [5], a variety of fitness functions for co-evolving pairs of rules that have the property that morphing between rules behaves well were tested.

This study takes the most successful of the fitness functions located in this study, performs a parameter study to improve the performance of the evolutionary algorithm, and demonstrates several different morphing styles for creating maps.

Cellular automata (CA) are a type of discrete dynamical system that exhibits 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 similar components experiencing local interactions[32], [23]. This paper applies them to create cavern-like level maps for games. The automata in this study are called *fashion-based* cellular automata because the

Matthew Kreitzer, Daniel Ashlock, and Rajesh Pereira are with the Department of Mathematics and Statistics at the University of Guelph, in Guelph, Ontario, Canada, N1G 2W1, {mkreitze|dashlock|pereirar}@uoguelph.ca

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

978-1-7281-1884-0/19/$31.00

updating rule may be thought of as following the current fashion within each neighborhood.

CA have been applied to the study of a diverse range of topics, such as structure formation[14], heat conduction[15], language recognition[22], traffic dynamics[20], modelling of biological phenomena [17], 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 publications on the use of CA for sound synthesis [24], [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[25].

CA have also been applied in the arts. They have been used to produce artistic images[13], [21], and their use has been extended to the fields of architecture and urban design[26], [16]. 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 [27]. Ashlock and Tsang[13] produced evolved art using 1-dimensional CA rules. These systems produced aesthetically pleasing images. In [11] a good deal of information about the fitness landscape of a particular type of cellular automata was derived.

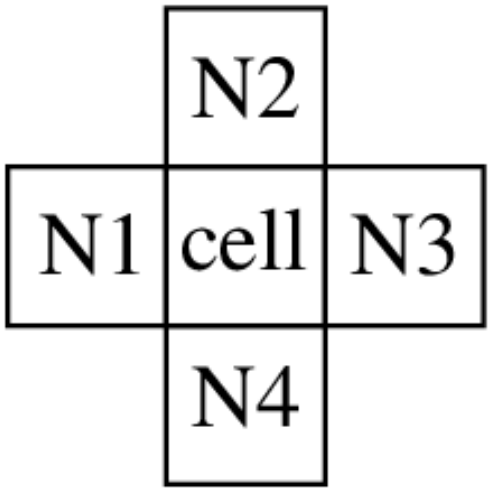

Fig. 1. A von Neumann neighborhood of a cell showing the four neighbors used in updating.

A. Previous Work in Level Map ACG

In addition to the FBCA studies [19], [3], [4], [5] 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 [30] 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 [8].

Automated level generation in video games can arguably be traced back to a number of related games from the 1980s (Rogue, Hack, and NetHack) and the task of automated level generation has recently received substantial interest from the research community. In [28], levels for 2D sidescroller and top-down 2D adventure games are automatically generated using a two population feasible/infeasible evolutionary algorithm. In [29] 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 [6] which introduced checkpoint based fitness functions for evolving maze-like levels. This work was extended in [7] by having multiple types of walls that defined multiple mazes that co-existed in a single level map. Related work includes generating distinct co-existing maps with designed tactical properties [7]. The study in [8] prototyped tile assembly to generate large maps and [9] defines 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. Section III gives theoretical properties of fashion based cellular automata presenting a normal form and proving mathematical facts useful for designing the fitness function. The design of experiments is given in Section IV while section V gives and discusses results. Section VI 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 101×101 grids of cells. The left and right and the top and bottom sides of the grid are considered to be adjacent to one another, making the grid toroidal. The neighborhoods of cells in the grid are von Neumann neighborhoods of the sort shown in Figure 1. The number of cell states used is $n = 6$, a number inherited from a companion study [5]. A single cellular automata 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. In section IV, details of how to store two co-evolving matrices, including the variation operators used by evolution, are given.

When used to generate a map, the automata is updated synchronously. As noted earlier, the matrix is used as a competition matrix, giving the score a cell in one state obtained from having a cell of another state in its neighborhood. 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

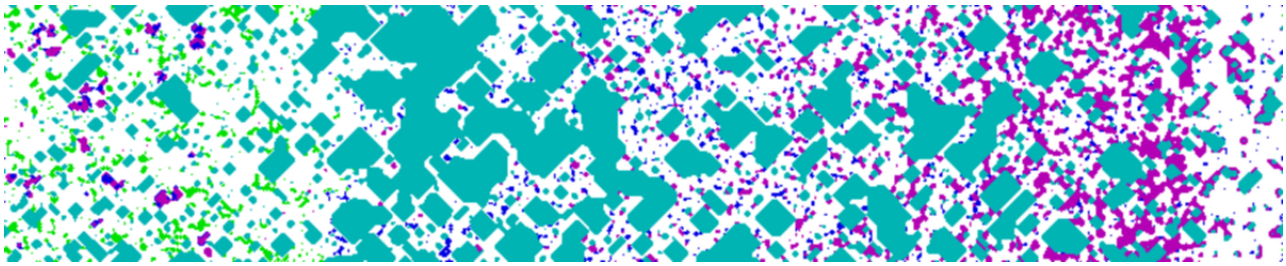

Fig. 2. An example of a map generated by morphing between two CA rules with the λ parameter changing laterally. Notice how the character of the map changes from left to right, the direction in which the morph acts.

of as "following the fashion" of the neighborhood. Fashion-based automata leave homogeneous regions homogeneous, a property that enriches the space of rules with automata that generate cavern-like levels.

The use of six states reflects the fact that six states are enough to encode quite complex automata, balanced against the fact that increasing the number of states increases the size of the search space enormously. Elementary combinatorics shows that the five cell-neighborhood, in the presence of n possible states, may be thought of as placing four balls (representing the neighbor cells) in n-boxes (representing possible states those neighbors may take on), together with n possible states for the central cell of the neighborhood. The number of possible neighborhood states and hence rules implicitly encoded by a competition matrix is given in Equation 1. For the balls-in-boxes counting formula, see [31].

$$\#rules = n \times \binom{n+3}{4} \tag{1}$$

For the six states this implies a competition matrix specifies 756 rules for the cellular automaton.

As there are many different matrix norms, it should be noted that the norm used in this paper is the Euclidean norm (also sometimes called the Frobenius norm or the Hilbert Schmidt norm).

Definition 1: Let A be an n by n real matrix, then the Euclidean norm of A (denoted as $\|A\|$) is defined as follows: $\|A\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{i,j})^2}$.

We note that similar to the Euclidean norm on vectors, the Euclidean norm on matrices satisfies the following Pythagorean property: if A and C are n by n real matrices with $\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}C_{i,j} = 0$, then $\|A + C\|^2 = \|A\|^2 + \|C\|^2$.

III. The Fitness Landscape and a Normal Form

An important observation is that updating of the automata is always based on comparisons of sums of four numbers, the scores a cell with a given state obtains against the states of its neighbors. These numbers consist of four entries from the same row of the matrix, the row that gives the scores of a state against all other states and itself. Since the only thing that matters is which total is largest in these comparisons, there is a good deal of freedom in specifying the competition matrix that yields a CA updating rule.

Definition 2: If two matrices M and N yield exactly the same updating behavior when used to drive an FBCA rule then they are said to be **behaviorally equivalent**. We write $M \sim_b N$.

It is obvious that behavioral equivalence is an equivalence relation in the formal sense. With this notion of equivalence available we can derive lemmas that lead to a normal form for matrices driving a given FBCA rule and help us design the fitness function used in this study. For the remainder of the section, assume all matrices are $n \times n$ real valued matrices.

Lemma 1: Recall that J denotes a matrix with all entries equal to 1. Then

$$M \sim_b (M + cJ)$$

for any constant c.

Proof:

Adding c to every entry of the matrix adds $4c$ to both sides of every comparison and so does not change their outcome. This demonstrates behavioral equivalence. □

Lemma 2: If $k > 0$ is a constant

$$M \sim_b (k \cdot M)$$

Proof:

The transformation of the matrix scales both sides of every comparison by the same positive value – yielding no change in comparisons and hence behavior, yielding a proof of behavioral equivalence. □

An implication of these lemmas is that every competition matrix used to drive an FBCA rule is behaviorally equivalent to a matrix with all its entries positive and with entries that sum to 1. The means that a matrix $M \sim_b M^*$, where M^* is both in the positive orthant of matrix space and in the hyperplane of matrices whose entries sum to 1.

We now consider a special matrix of this type:

$$C = \frac{1}{n^2}J$$

which has all its entries equal. If this matrix is used to drive a CA rule then all comparisons are of equal numbers and so no state every changes (recall a neighbor must have a higher score to take over a cell). We call call C the *frozen* matrix and its associated updating rule the frozen rule. We now have the tools needed to create a normal form for competition matrices.

Theorem 1: There is a hypersphere of finite hypervolume containing all possible FBCA behaviors.
Proof:
The first two lemmas, and the fact C has constant entries, means that

$$M \sim_b C + t \cdot (M^* - C)$$

for any scalar $t > 0$. This gives us a ray, in matrix space, of behaviorally equivalent matrices, starting at but not including C. The matrices on this line may be normalized by dividing by the sum of their entries, which are all positive, to place them in the plane of constant sum one. These normalized matrices form a ray of behaviorally equivalent matrices within the constant sum plane. Let us assume that M^* has been normalized so that its entries sum to one. Then

$$\begin{aligned} \|C + t \cdot (M^* - C)\|^2 &= \|C\|^2 + t^2 \|M^* - C\|^2 \\ &= \frac{1}{n^2} + t^2 \|M^* - C\|^2 \end{aligned}$$

by the Pythagorean property. For any fixed $r > \frac{1}{n}$, we can select a t such that the Euclidean norm of the matrix, $C + t \cdot (M^* - C)$ is r. Obtaining a matrix consisting entirely of positive entries only requires that r not be too large. The matrix C is the center of this hypersphere. Since r is finite, this hypersphere of n by n matrices whose entries sum to one and having Euclidean norm r has finite hypervolume.

Definition 3: For any matrix M the equivalent matrix, within the hypersphere defined in the proof of Theorem 1, is the *normal form* of M, denoted M_{norm}.

Let us motivate the derivation of the normal form. It is possible to find two behaviorally equivalent matrices at arbitrarily large Euclidean distance from one another, treating the entries of the matrix as Euclidean coordinates and using the Euclidean distance. A hypersphere has a finite diameter and so the normal forms are all at a finite distance from one another. The normal form removes both scaling and displacement, sources of behaviorally irrelevant distance, from the matrices. The distance between normal forms, thus, gives a more meaningful measure of behavioral similarity. The problem of behavior similarity and behavioral equivalence, however, has another wrinkle.

Definition 4: For a matrix M used to drive an FBCA rule, define $\Delta(M)$ as the smallest difference between any two of the numbers that can be compared during updating of the cellular automata.

Lemma 3: If M and N differ in only one position by an amount less than $\Delta(M)/4$ then $M \sim_b N$.
Proof:
The largest change in a value being compared during updating of the automata is four times the largest change to a matrix entry. In this case, that is less than Δ and so, by the definition of Δ, not enough to change the outcome of a comparison, demonstrating behavioral equivalence. □

The implication of this lemma tells us a good deal about the fitness landscape. Any matrix M for which $\Delta(M) > 0$ lies in a connected set of behaviorally equivalent matrices within the hypersphere of normal forms. We call this set the *grain* associated with the behavior. This means that many behaviors occupy regions of the fitness landscape with support of positive size. Note that the frozen behavior occurs at a single point (all coordinates equal) within the hypersphere.

The existence of grains means that, when a map is generated by morphing between two rules, there will be stretches of constant behavior within the map, corresponding to the intersection of the line of morphing with a grain. Open questions include the size of grains and their relationship to one another within the fitness landscape.

IV. Design of Experiments

We begin this section with an explanation of the fitness function. This fitness function was first defined in [5], but the explanation given here is more extensive and ties into the theory developed in Section III.

A. The Fitness Function

The fitness function used in this study is the most successful of eight different functions used in [5]. The basis of the fitness function uses a map with white and non-white cells, treating white cells as empty, that attempts to maximize the total path length from the middle of the left face of the grid of cells to the middle of the other three faces. A fixed initial random state for the grid of cells, used in all fitness evaluation, is updated 20 times using the cellular automata to create the map that undergoes quality evaluation. A simple dynamic programming algorithm [10] is used to compute the shortest open path between the centers of the face of the grid and the sum of the three distances from the center of the left face to the center of the other faces is the base fitness value. If one of the faces cannot be reached, the map is awarded a base fitness of zero. The fitness function is subsequently multiplied by three secondary rewards, which leaves a zero penalty fitness at zero. The first modifier is the *density modifier*. The user specifies a desired fraction α of non-empty cells in the final map. The actual fraction β is computed and then the basic fitness is multiplied by

$$R_d = \frac{1}{(\alpha - \beta)^2 + 0.4} \tag{2}$$

The number 0.4 was chosen to make the maximum multiplicative reward 2.5, when $\alpha = \beta$, in order to give it a similar scale to the other rewards. Based on the work in [5], this study uses $\alpha = 0.7$.

The second modifier, the *diversity modifier* multiplies the basic fitness by the entropy of the relative density of cell states that appear in the map. The number of cells of each type are computed and normalized to one to give a rate of empirical occurrence p_i of cell type i. The entropy is then

$$R_e = -\sum_{i=0}^{n-1} p_i \cdot log_2(p_i) \tag{3}$$

The maximum reward is $log_2(n)$ where n is then number of states the automata possesses. The reward R_e is greatest

when the distribution among state types is as even as possible. For the six cell states available in the system used for testing, this reward has a maximum value of $Log_2(6) \cong 2.58$. The third modifier used is the *angular modifier*. This modifier is used to combat the fact demonstrated in Lemma 2 that means matrices that are scalar multipliers of one another are behaviorally equivalent. If we regard a matrix as a vector in $\mathbb{R}^{n^2}$, then we can reward being far from lying on the same behavioral line resulting from scaling if we reward having a larger angle between these vectors. Recall that if $\vec{v_1}$ and $\vec{v_2}$ are non-zero vectors and θ is the angle between then then

$$\cos(\theta) = \frac{v_1 \cdot v_2}{||v_1|| \cdot ||v_2||} \tag{4}$$

The cosine is largest when the angles are most similar so the angular reward is given as the reciprocal of the cosine. In terms of the matrix entries $(M_1)_{i,j}$ and $(M_2)_{i,j}$ this is:

$$R_\theta = \frac{\|M_1\|\|M_2\|}{\sum_{i=1}^{n}\sum_{j=1}^{n}(M_1)_{i,j} \cdot (M_2)_{i,j}} \tag{5}$$

Since this reward is the reciprocal of the cosine of an angle, it has a potential maximum value of infinity, as the angle approaches a right angle, but this did not occur in practice. Among the runs in the earlier study using this fitness function the maximum cosine-based reward observed was 2.43.

To summarize, if TPL is the total path length from the middle of the left face to the middle of the other faces, then the overall fitness is

$$fitness = R_d \times R_e \times R_\theta \times TPL$$

B. The evolutionary algorithm

The goal of the evolutionary algorithm designed in this study is to achieve a pair of $n \times n$ matrices that can generate a spatially varying morphed cellular automata with good map properties. We must specify how the matrices are stored for evolution. We are using six states in the automata; state 0 encodes empty space and states 1-5 encode different types of filled space. The two matrices evolved for the morph are 6×6. We store the data structure as a vector of 72 real numbers in the range $0 \leq M_{i,j} \leq 1$. The first thirty-six members are the entries of the first matrix, read row-wise. The second thirty-six entries designate the second matrix, read in the same fashion.

When two pairs of matrices are selected for reproduction, they are copied into child structures and new rules are generated with two point crossover. After crossover, mutation is performed. From one to a maximum number of mutations (MNM) positions are chosen in each child and those entries replaced with new numbers from the interval [0,1]. The number of positions used it, itself, chosen uniformly at random. Selection of parents and population members to be replaced is performed with size seven single tournament selection. This model of evolution functions by picking seven population members. Within the group of seven, the two most fit are chosen as parents and the two least fit are replaced with the children.

The evolutionary algorithm is run for 10,000 instances of tournament selection called *mating events*. Two parameter setting experiments are performed. The first uses a population size of 360 and MNM values 1, 3, 5, 7, and 9. The second uses population sizes 40, 120, 360, 1080, and 3240 are used with MNM set to 7. Note that one set of runs is used in both parameter setting studies. The values 360 and 7 were used exclusively in the earlier study; this study is the first to perform parameter setting one the morphing cellular automata. We will see that 360 was a good choice while $MNM = 7$ was not.

We say that the two matrices specified by a single member of the evolving population of map-specifiers are *co-evolved* because they undergo joint fitness evaluation and so must adapt to one another in a fashion that permits all the intermediate rules to achieve a high score according to the quality measure. This is a critical feature of the system for producing usefully morphable rules, as can be seen for examples made with matrices that are not co-evolved, as in [4].

The number of mating events chosen, 10,000, was selected to permit experiments to terminate in a reasonable amount of time. It seemed likely, looking at upward trends in fitness, that additional evolution time would pay additional benefits, and so one additional experiment was performed using 100,000 mating events and the parameter values of population size 360 and $MNM = 1$ that yielded the best results. We refer to this collection of thirty runs and the *long experiment*. In order to verify that the long experiment was a good idea we computed the *time of last innovation* for the nine other experiments. This is the fraction of the 10,000 updatings that passed before the last time the best fitness in the population increased by at least 1.0.

V. Results and Discussion

Figure 3 shows the thirty best-of-run results from the runs that used the best parameter settings and extended evolutionary time. The automata are deterministic and all use the same starting conditions, meaning that no two behaviorally equivalent matrices appear as best-of-run results. The algorithm can clearly produce behaviorally diverse results.

The results of the two parameter studies appear in Figure 4 (mutation rate) and Figure 5 (population size). Pleasantly, both studies give unambiguous results. The lowest mutation rate is the best and the middle population size is superior.

The results for time of last innovation, shown in Figure 6, demonstrate that many or most of the individual runs were still improving at updating 10,000. The repetition of the experiment with $MNM = 1$ and population size 360 that was run for 100,000 updatings achieved considerably higher fitness. While the maximum fitness in any of the nine parameter setting runs was 3562.5, the maximum fitness in the long experiment was 5410.42 and thirteen of the thirty best fitness values in the long experiment exceeded 3562.5.

One of the strongest results in [4] is that FBCA may be reused to generate additional maps with similar appearance,

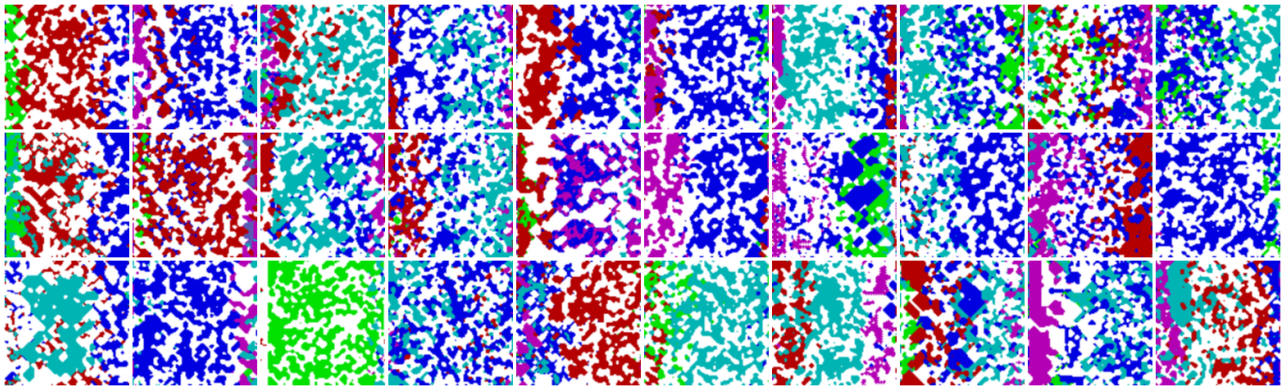

Fig. 3. Renderings of the cellular automata that achieved best fitness in the 30 replicates using $MNM = 1$ and population size 360 from the long experiment.

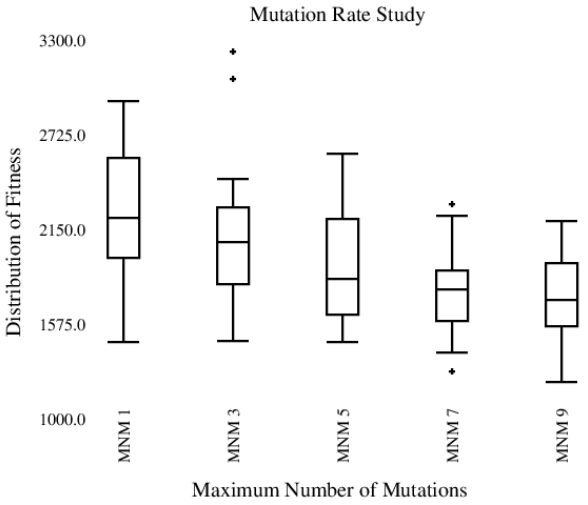

Fig. 4. Fitness distributions of best results from 30 replicates of the evolutionary algorithm in the mutation rate study.

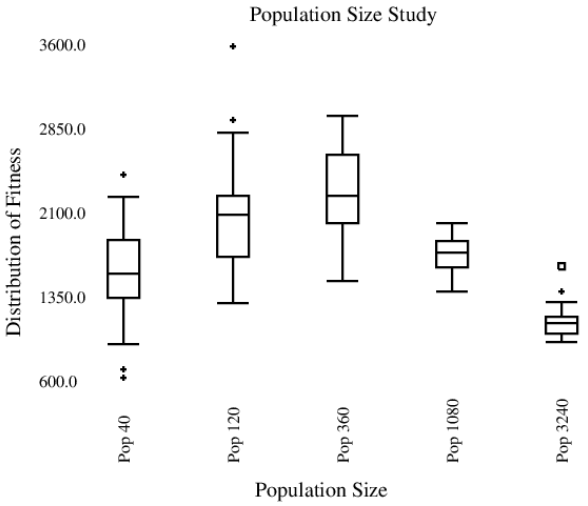

Fig. 5. Fitness distributions of best results from 30 replicates of the evolutionary algorithm in the population size study.

but different details. Figure 7 demonstrates that the morphing cellular automata have this quality as well.

A. *Alternate rendering by morphing differently*

The morphing rules used to generate CA in this study calculated λ in terms of the left-to-right horizontal position. Suppose, instead, we create craters, ridges, or hills with minimum elevation zero and maximum elevation one and then condition λ on the height in a landscape. The technique for generating such numerical height maps is given in [9]. Examples of rendering morphs with these alternate landscapes are shown in Figure 8.

The alternate morphs shown in Figure 8 do not maintain the connectivity required during the fitness evaluation. This can be fixed by using a desired morph as the morph employed during fitness evaluation or my relatively simple dynamic programming based repair. These images to show the potential of the technique to use a variety of morphs to achieve diverse terrain types.

VI. Conclusions and Next Steps

This study has demonstrated that it is practical to co-evolve pairs of matrices so that a morph, in the form of a line in matrix space, can be made to generate a level map that exhibits a substantially greater diversity of local appearances created using a single fashion-based automata. The reusability of CA rules for generating maps also survived from the single matrix version of the experiments.

This study improves our understanding of the behavior of the FBCA rules enough that it is worth making a broader exploration of the potentials of the system including additional fitness functions and increasing the number of states in the cellular automata.

An effort was made in this study to keep the relative importance of the multiplicative fitness modifiers roughly similar. Studying different schemes, such as raising the modifiers to distinct powers, would permit them to be reweighed, a potentially profitable area for future study.

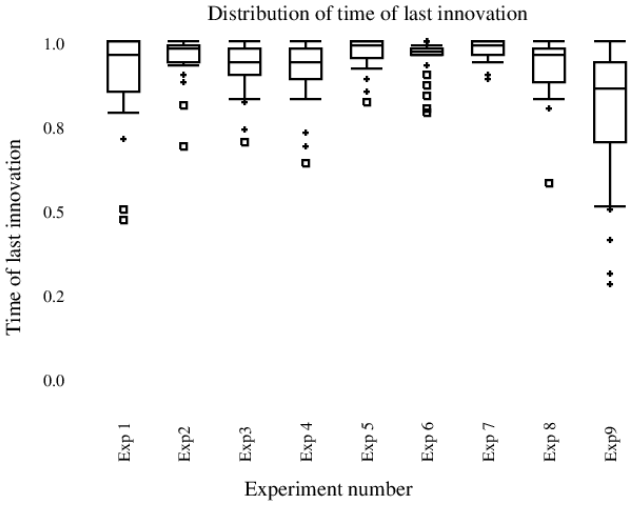

Fig. 6. Shown is the distribution of the time of last innovation in each of the nine experiments performed in the parameter setting study.

A. Changing the representation

The representation used in this study for pairs of real matrices is *direct*, encoding the matrices as lists of numbers. A novel generative representation for evolving matrices was introduced in [12] and it was used to generate matrices that served as rules for fashion based cellular automata in [18]. The automata produced were qualitatively quite different from those produced by direct evolution and so it is likely the optima in matrix space located with this novel representation are in a different part of the space.

It is clearly possible to modify the generative matrix representation to evolve pairs of matrices and perform morphing experiments, paralleling those in this study. It is also practical to extract the final numerical matrices from the generative representation for examination and possible use as seeds in an evolutionary algorithm where evolved matrices are used in initial populations. An interesting question is if the distinct qualitative character of the generatively encoded matrices would be preserved if they were subject to additional evolution.

B. Evolving partner matrices

This study co-evolves pairs of matrices. Another approach that could be investigated is to fix a matrix that produces and interesting type of cavern map as half of a morphing pair and evolve the other half to find good partners. If multiple partners were evolved, then a morph could proceed in a fashion employing distinct morphs in distinct parts of the space.

It would also be interesting to see if some matrices admitted a greater diversity of partners than others. If so, such high-partnering matrices would be useful design elements for additional work.

C. Exploiting theory

The theory developed in this section created a normalization from an infinite gene space to one that was bounded but still contains all possible CA behaviors. The normal form also permits phenotypically meaningful comparisons at the genotype level. This creates the potential for clustering and phylogenetic analysis of the CA-rules produced by evolution, which is potentially valuable.

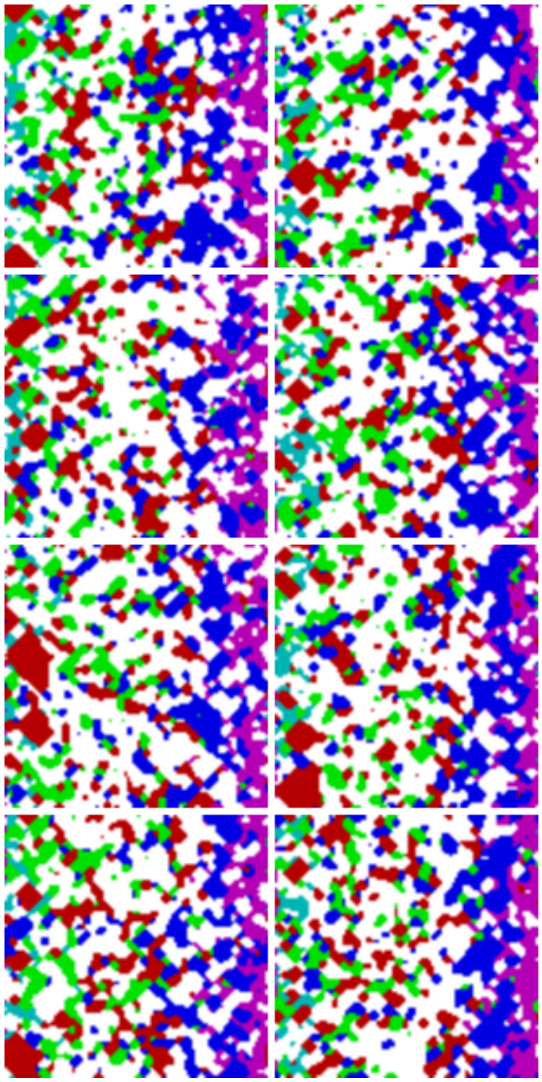

Fig. 7. Eight renderings of a single co-evolved morphing cellular automata using different initial conditions. The matrix used to make these pictures was taken from the long experiment.

The major application of the theory in this study was to explain the utility of the angular modifier to fitness. The theory has untapped potential for the design of additional fitness functions and strongly suggests that multi-criteria optimization might be valuable as well.

The fact there are finitely many neighborhood configurations possible means that the number of behavior equivalence classes of matrices, and so of CA behaviors is finite. The organization of these behaviors into grains implies the presence of an interesting structure to the fitness landscape that could be exploited in the future to improve search. The notion of behavioral equivalence and the normal form for the matrices

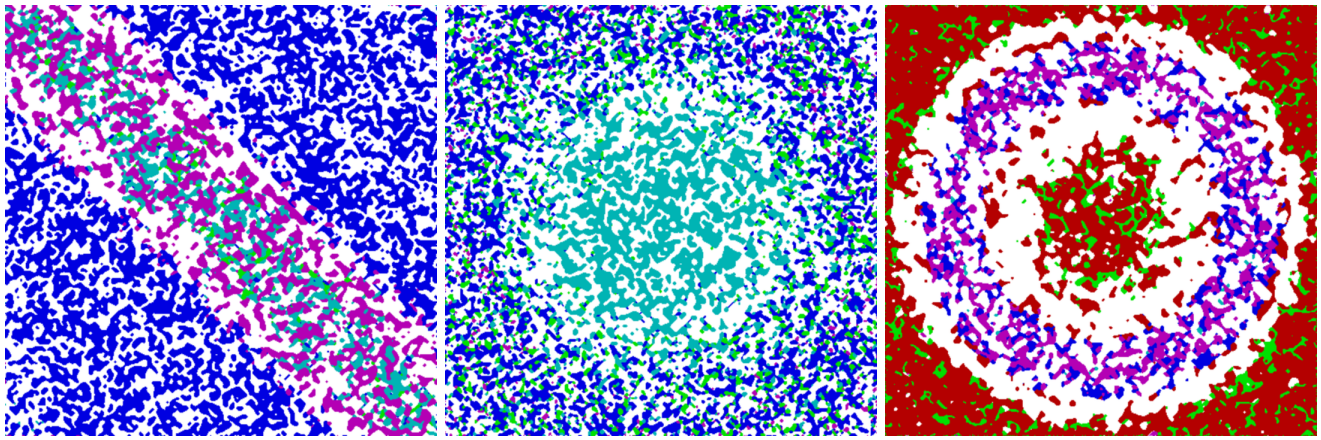

Fig. 8. Examples of morphs using λ-landscapes. In these morphs, λ is derived from, respectively, the height of a ridge, a hill, and a crater.

make novelty search a natural target for future work.

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. Evolvable fashion-based cellular automata for generating cavern systems. In *Proceedings of the 2015 IEEE Conference on Computatational Intelligence in Games*, pages 306–313, 2015.

[4] D. Ashlock and L. Bickley. Rescalable, replayable maps generated with evolved cellular automata. In *Acta Physica Polonica (B), Proceedings Supplement*, volume 9(1), pages 13–22, 2016.

[5] D. Ashlock and M. Kreitzer. Evolving diverse cellular automata based level maps. to appear in the proceedings of the SEDA wargames conference, 2019.

[6] 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.

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

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

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

[10] D. Ashlock and C. McGuinness. Graph-based search for game design. *Game and Puzzle Design*, 2(2):68–75, 2016.

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

[12] D. Ashlock and G. A. Ruz. A novel representation for boolean networks designed to enhance heritability and scalability. In *Proceedings of the 2017 IEEE Conference on Computational Intelligence in Bioiformatics and Computational Biology*, pages 1–8, Piscataway NJ, 2017. IEEE Press.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] C. Gregor, D. Ashlock, G. A. Ruz, D. McKinnon, and D. Kribs. A novel linear representation for evolving matrices. Sumitted to the IEEE Transactions on Evolutionary Computation, 2019.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

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

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

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

[28] 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.

[29] 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.

[30] 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.

[31] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics, second edition*. Cambridge University Press, New York, NY, 2001.

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