Exploring the Rich Behaviour of Developmental Graph Cellular Automata

Riversdale Waldegrave[1], Susan Stepney[1] and Martin A. Trefzer[2]

[1]Department of Computer Science, University of York, UK
[2]School of Physics, Engineering and Technology, University of York, UK
{rraw500,susan.stepney,martin.trefzer}@york.ac.uk

Abstract

We explore a wide variety of behaviours possible with Developmental Graph Cellular Automata. We use novelty search to find more extreme types of behaviour in terms of transient length and attractor cycle length. This also serves as a proof-of-concept that the system is evolvable. We then examine in more detail some individual examples of interesting behaviour, particularly focusing on cases where the graph divides into two or more separate components.

Introduction

In Waldegrave et al. (2023) we introduced a model of *Developmental Graph Cellular Automata* (DGCA) that are inspired by the rich global behaviour that can be generated from local interactions in CAs, as well as the dynamically growing structures that can be modelled using L-Systems. Combining the two ideas points towards dynamically growing graph structures in which each node's behaviour is determined by purely local information.

Development halts when the system enters an attractor, when the graph is isomorphic (conditioned on the node states) with a graph that has been previously seen in the course of development. Because it is a closed, deterministic system, it is impossible to escape such an attractor once it has been reached. Attractors can either be point attractors, in which neither the graph structure nor the node states change from one timestep to the next, or cyclic attractors, in which the graph cycles through a number of topologies and/or configurations of node states. See Waldegrave et al. (2023) for more detail on the workings of the model.

Here we explore further the rich behaviour of DGCAs. Ultimately it is hoped that goal-directed evolution can be used with these systems, in order to generate graphs with particular properties. The aim of the present paper is exploratory, and so here we use novelty search to push the system to exhibit more extreme or unusual behaviours. We then examine a few individual examples of "interesting behaviour" in more detail, focusing particularly on cases where the graph divides into mutiple components, which in some cases are evocative of biological processes such as reproduction.

Related Work

In most traditional graph grammars, subgraphs are replaced according to rewriting rules (Rozenberg, 1997). The step of identifying subgraphs to be replaced is computationally expensive and often nondeterministic. The order of rule application is also a source of nondeterminism. The rewriting process involves using properties of the whole graph rather than information local to each node, which makes it less suitable as a model of biological morphogenesis. In our DGCA model, development can take place in parallel at each node using only local information: this avoids costly subgraph analysis and makes the model entirely deterministic.

The Cellular Encoding model (Gruau, 1994) is a graph developmental model in which each node follows a predetermined microprogram to divide, create or remove connections and so on. Since it does not incorporate input from its neighbourhood, there is no element of feedback to the growth process. This means it is not capable of the rich dynamics we see in CAs, where simple rules and simple seeds can lead to enormous complexity.

The "transition rules" in the DGCA system are encoded in the weights of a small neural network, following the work of Wulff and Hertz (1992); Tavares et al. (2015); Mordvintsev et al. (2020) on Neural Cellular Automata (NCAs).

Grattarola et al. (2021) combine the idea of NCAs with earlier work on CAs structured as graphs rather than regular lattices (Marr and Huett, 2009) to create the model of Graph Neural Cellular Automata (GNCA). Compared with GNCA, our system adds the ability for nodes to be duplicated or removed as part of the transition function, meaning that the graph structure can develop and grow (or shrink). At each timestep, the single-layer perceptron (SLP) that encodes the transition function receives an input vector that contains information about each node's neighbourhood (filtered counts of neighbouring nodes in each state). The output of the SLP gives not only the new state that each node should take, but also a set of flags indicating whether nodes should be duplicated, removed or neither. When a node is duplicated, a new node is created which has the same incoming and outgoing edges as the parent node. When a node is removed, all

of its incoming and outgoing edges are also removed. This can lead to the graph splitting into unconnected components; this particular behaviour is examined in more detail below.

Experiment

Waldegrave et al. (2023) use random search to identify various classes of behaviour that the system can produce. The behaviour was categorised into five classes: runaway growth, in which the size of the graph grows beyond an arbitrary size limit; slow growth, in which an attractor is not reached within an arbitrary timestep limit; death, in which all nodes are removed from the graph; static behaviour, in which the size of the graph does not change at all; and halting growth, in which the size of the graph changes before an attractor is reached, bringing growth to a halt.

The last of these is considered the most "interesting" behaviour because it represents a developmental process that comes to a halt *naturally* without external intervention or the use of something extraneous to the developmental process itself, such as an energy model. This behaviour invites comparisons with the self-regulating growth of natural organisms, in which growth halts once maturity is reached.

Here, we conduct an experiment to see if more novel halting growth behaviours can be discovered using an evolutionary search than using random search. This is both a proof-of-concept that the system is amenable to evolution, and also illustrates the richness of possible behaviours of the system.

Novelty Search

Novelty Search is an open-ended evolutionary search paradigm introduced by Lehman and Stanley (2011). All evolutionary search processes require a fitness function, and in most cases this involves an assessment of how well the individual (ie. candidate solution) achieves some predefined goal. In novelty search the fitness function instead measures how different the individual is from previously assessed individuals. The objective is therefore not fixed at the start, and the "fitness" of an individual changes dynamically as the search proceeds. This can enable more complex behaviour to emerge over time. It is argued that this is often an easier way to achieve such complex behaviour than specifying it in a static fitness function from the start, which can lead to the search process becoming trapped in local fitness optima. In terms of the commonly cited trade-off between exploration and exploitation, Novelty Search focuses only on exploration and does not do any exploitation of found behaviours. Novelty Search has analogies with biological evolution, in which fitness functions are also not fixed but are often contingent on the behaviour of other individuals, giving rise to the concept of ecological niches. This dynamical and open-ended search process seems to give rise to the complexity that we see in the natural world.

Novelty Search is particularly appropriate for the exploratory work here, in which we want to discover what behaviour the system is capable of, but do not have any fixed objectives in terms of graph structures.

Behaviour class	Transient len.	Attractor len.	max size
Death	$\leq max$	1	$\leq max$
Runaway	$\leq max$	—	$= max$
Slow	$= max$	—	$\leq max$
Static	$\leq max$	$1...max - T$	$= init$
Halting	$\leq max$	$1...max - T$	$\leq max$

Table 1: The different classes of behaviour and their corresponding ranges of possible values in behaviour space. The attractor cycle length can be up to the maximum number of timesteps ($max = 256$ here) less the number of steps that have already been used in the transient (T). In the "static" behaviour, the size of the graph does not change so the number of nodes remains the same as its initial value ($init$). The halting growth behaviour is least constrained in the behaviour values it can take, so novelty search should discover more of this type of behaviour.

Behaviour Space

In setting up a Novelty Search, one must define the behaviour space within which to assess novelty. Since we are presently interested in exploring the dynamics of the system, we define the behaviour space over the dimensions of transient length and attractor cycle length. These are commonly used metrics for analysing the behaviour of dynamical systems, see for example Wuensche et al. (1992).

Both of these measures are unbounded: both the transient length and the attractor cycle length (if an attractor is found at all) could be arbitrarily large, since the graph may be arbitrarily large. To make the experiment tractable we set fixed limits of network size and time steps allowed for development (both to 256, as in Waldegrave et al. (2023)). Individuals that do not find an attractor within these time and space limits are not discarded, but left in the results. Since this is a frequently occurring behaviour, novelty search should naturally seek to move away from it and focus on systems that do find an attractor.

We include a third dimension to the behaviour space: the maximum size (number of nodes) that the graph reaches over the course of its development. Since the graphs can grow and shrink over time, we used the maximum size rather than the final size. This means we have usable numbers for individuals that do not find an attractor. Furthermore in the case of graphs where all nodes are removed ("death"), recording the maximum size allows us to distinguish between graphs which grow to a large size before they dwindle to nothing, and those that only shrink.

Some examples of how the three features of our behaviour space can be used to identify different behaviour classes are shown in Table 1. We are most interested in the "halting growth" behaviour class, and this is where novelty search

should push the system since there is most freedom in all three dimensions.

Microbial Genetic Algorithm

In order to implement novelty search, we use the Microbial Genetic Algorithm (MGA) (Harvey, 2011). This is a minimal implementation of a genetic algorithm which uses a steady-state population rather than a generational model. At each timestep, two individuals are randomly chosen from the population and their fitness is assessed (in this case, their novelty compared with previously seen behaviour). The fitter individual remains, whilst the less fit individual is replaced by the "offspring" of the two. The MGA model is inspired by *bacterial conjugation*, in which genetic material is transferred horizontally between bacteria. Interpreted like this, the MGA model does not involve death and reproduction, but continual transfer of genetic material from the stronger to the weaker members of a steady-state population.

For the genetic crossover operation, the half of the weights in the SLP that constitute the "transition rule" of each individual are replaced by weights from the other parent. This wis done on a column-by-column basis: a randomly selected 50% of the columns in the weight matrix of the offspring are taken from each parent. Each column of weights dictates the influence of one element of the input vector to the SLP (ie. information about the node's neighbourhood) in selecting the output (ie. the new state of the node or the action it should take). For more details on how the weights of the SLP define the transition rule of the system, see Waldegrave et al. (2023).

After the crossover operation is performed, a mutation is applied, randomising 5% of the weights in the matrix. This relatively small degree of mutation was chosen through preliminary experiments (not reported here), and found to enable the discovery of new behaviour whilst not completely destroying the behaviour inherited from the parents.

Seed Graph Selection

The purpose of the novelty search experiment is to find transition rules (as encoded in the SLP weights) which produce novel behaviour. To make sure that new behaviour arises from the evolution of the rule rather than from having a different starting point, we use the same starting graph or "seed" graph in each trial.

One interesting property of the system is how the structure of the seed graph influences the structure of the developed graph. For instance, a seed graph without recurrent connections can never develop into a graph *with* recurrent connections. This is because whenever a node is created, its incoming and outgoing edges are copied from its parent node: if there are no recurrent connections in the graph at one timestep, they cannot be created at the next. On the one hand this means that if we want to see a wide variety of different kinds of graphs develop, then we must start with a relatively large seed, containing a variety of different connections that can be developed (uni-directional, bidirectional, self-loops etc). On the other hand, it means that if we wish to restrict the types of graph that are produced, we can do so by using a simple seed without much variety. For instance, Figure 1 shows how a simple seed of three nodes joined in a line by edges pointing in one direction develops into a three layer feedforward network. In this case, both the number of layers of the final network and the fact that it is feedforward were already properties of the seed graph. The number of nodes in each layer and the fact that there is one "skip connection" are a result of the particular developmental process

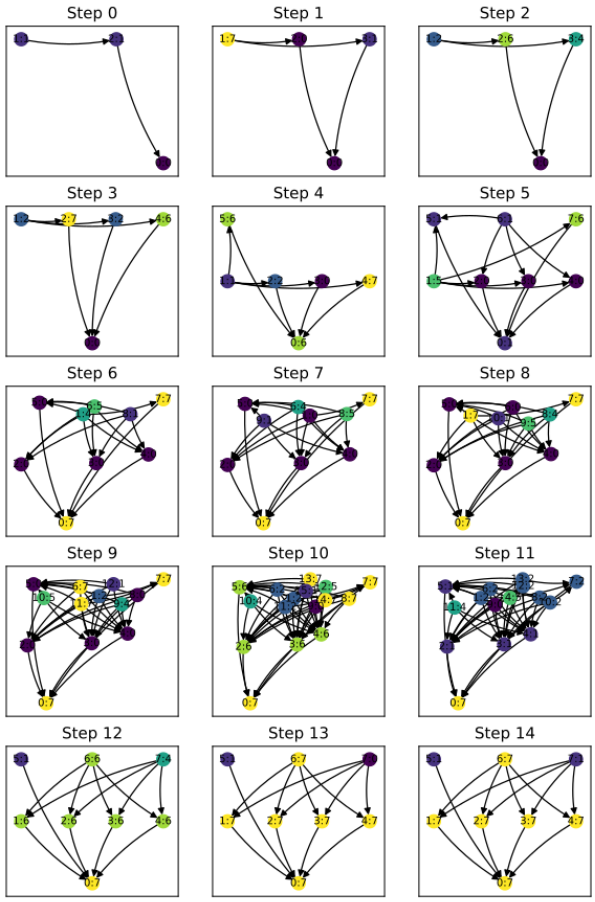

Figure 1: An 8-state system with a simple seed of three nodes in a line develops into a 3-4-1 feedforward network with one skip connection. After an initial burst of growth, eight nodes are removed in step 12. Development halts two steps later as the system reaches a point attractor. Due to the absence of any recurrent connections in the seed, it is impossible for recurrent connections to emerge during the development process. The particular developmental steps in this process are the result of the weights of the SLP which encode the DGCA transition rule (analogous to the genome).

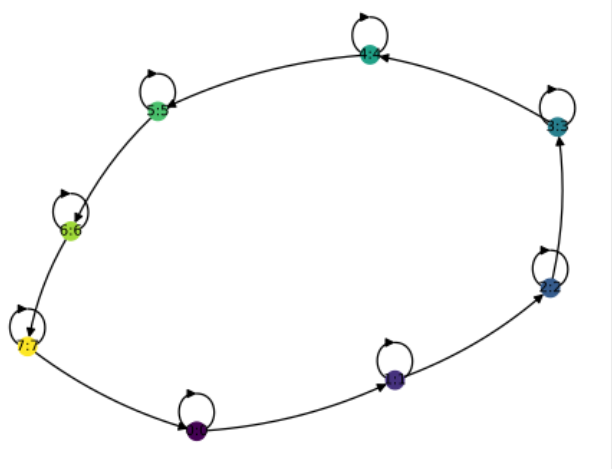

Figure 2: The seed graph used in the random search vs. novelty search experiment. The topology is a unidirectional ring of eight nodes, each of which has a self-loop edge. Node colours and numbers represent the state of the node. Node states in the seed graph are assigned sequentially using the number of states in the system. The graph shown is for an eight state system, so every node has a different state. In a four state system, the states would be repeated around the ring as: 1-2-3-4-1-2-3-4.

this network underwent.

In the novelty search experiment, we want to enable a wide variety of different graph topologies and developmental behaviours (in terms of transient and attractor cycle lengths). For this reason it is desirable to use a relatively complex seed graph. Since we are experimenting using systems with up to eight possible node states, it is desirable to use a seed graph with at least this number of nodes, so that every state can be represented in the seed graph. After some preliminary experiments, we selected a seed graph that is a ring of eight nodes, with edges pointing in one direction, and with every node having a self connection. The node states in the seed graph are set to a repeating sequence of the number of states in the system being used. The seed graph for the eight state system is shown in Figure 2.

Experiment parameters

In our experiment, we implement novelty search using the MGA with a population size of 30, and 1,000 evaluations. Fitness is calculated as the average Euclidean distance from the five nearest neighbour points in the behaviour space. For the random search, we also use 1,000 evaluations. We run the random search and the novelty search once each for systems with 2 states to 8 states.

Results

The charts in Figure 3 compare random search with novelty search for systems with 6, 7, and 8 possible node states.

System States	Random Search	Novelty Search
2 State	18	20
3 State	29	33
4 State	40	42
5 State	34	39
6 State	44	45
7 State	30	54
8 State	50	57

Table 2: The number of 'voxels' that are populated when the behaviour space is quantised into 8 regions along each of the three dimensions (transient length, attractor cycle length, and maximum size reached). In all cases, novelty search achieves a greater spread of behaviour (populates more voxels).

They plot individuals in the behaviour space of transient length and attractor cycle length, with colour used to represent the maximum number of nodes a graph reaches over the course of its development.

In all cases, novelty search seems to be able to explore more of the behaviour space. To quantify the greater spread of behaviour found with novelty search compared with random search, we quantize the behaviour space into 8 regions along each of the three dimensions and count the number of three-dimensional 'voxels' (out of a total of $8^3 = 512$) that are populated (this approach follows the CHARC method used in Dale et al. (2019)). The results for systems with 2 to 8 node states are shown in Table 2. In all cases, more of the behaviour space voxels are populated when novelty search is used, although the degree of improvement varies.

Exploring more of the behaviour space is the whole purpose of novelty search; here it also shows that the system can be evolved towards desired behaviour by using the mutation and crossover operators on the weights matrices of the SLPs which constitute the transition rules. This opens up the possibility for future work to use goal directed evolution to create systems which develop in particular ways. Such goals might include not only the attractor behaviour explored here, but also properties of the developed graphs themselves, such as degree distribution and shortest path lengths. Fitness could also be assessed using the functional properties of the graphs: for example, their usefulness as neural networks in a Echo State Network (Jaeger, 2001) context.

Some of the behaviour found by novelty search is surprisingly complex, with long transients and attractor cycles, and rapid growth to a large size which nevertheless comes to a halt. Some individual examples of interesting behaviour are selected for further discussion below. It should be noted that there is a certain portion of the space that its unreachable, where the transient length plus the attractor cycle length is greater than the cutoff we have set for the number of iterations of the system (256 steps). This is highlighted on the charts in Figure 3 with a red line at the upper right indicating

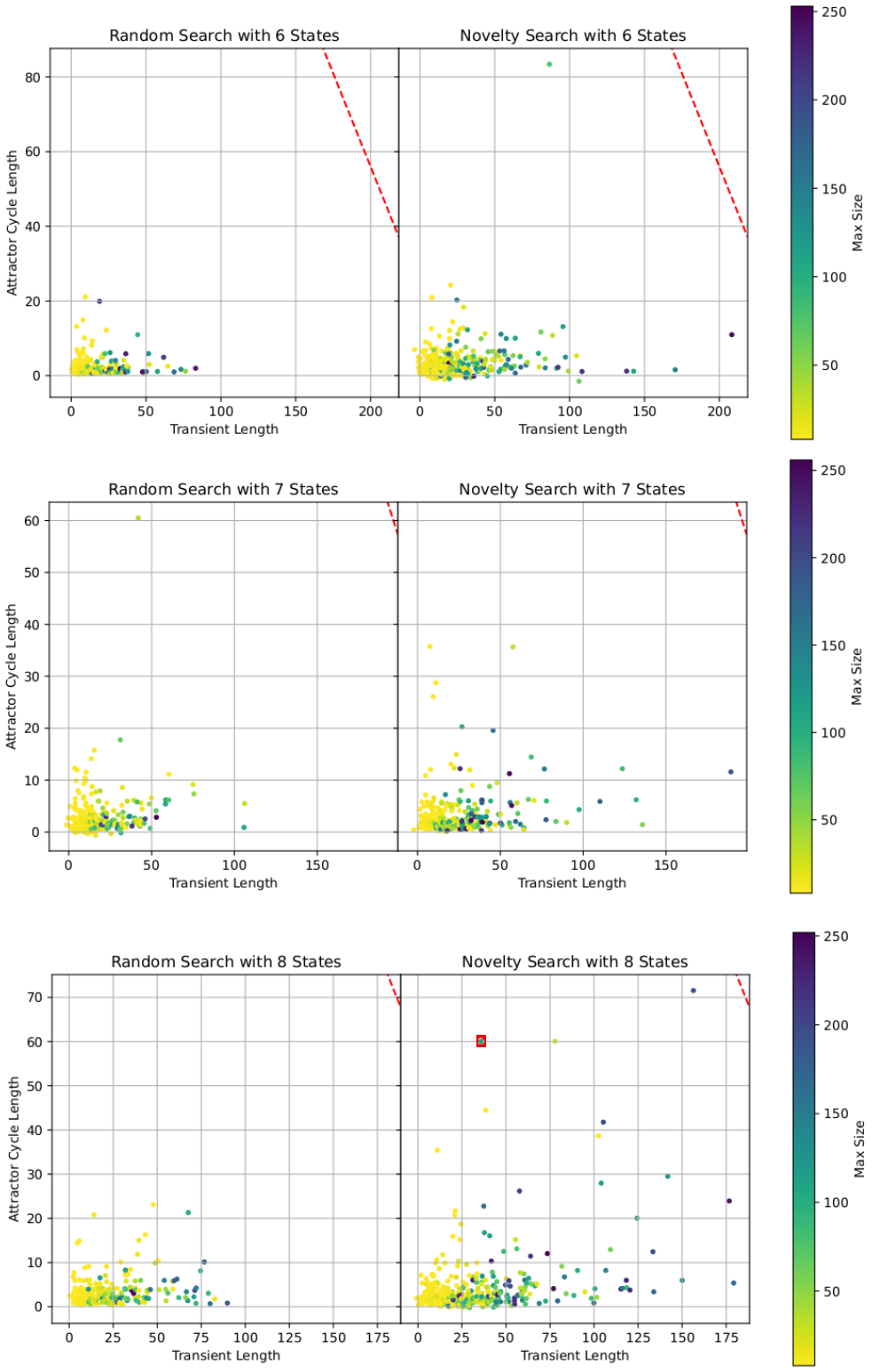

Figure 3: Charts showing random vs novelty search for 6, 7 and 8-state systems. Each point on the charts represents an individual's developmental behaviour in terms of transient length, attractor cycle length and maximum size reached. Only graphs which reached an attractor are plotted. Graphs in which all nodes were removed are also not plotted. The red dashed line in the upper right corner shows the limit of the explorable space, since development was arbitrarily cut off after 256 timesteps. The individual highlighted with a red box in the 8 state novelty search chart is shown in detail in Figure 4

this unreachable portion of the space.

Interesting Behaviours

In the remainder of this paper, we select some individual examples of "interesting" behaviour to examine in more detail. This is similar to the "zoos" of interesting behaviour that have been compiled for Game of Life and other CA rules. For example, Wolfram (2002) catalogues the behaviour of many different Elementary CA rules, whilst Wuensche et al. (1992) provides an "atlas" of basins of attraction in one dimensional CAs.

In the present system, both the seed graph and the "rule" (as encoded in the SLP) can be explored. This gives a huge parameter space to explore. This section does not aim to explore this space systematically or comprehensively, but rather to pick out some examples which indicate the range of behaviour the system can produce. In particular, we highlight a class of behaviour that involves the graph splitting separate components.

Graphs Dividing

Since nodes can be removed as well as added during the development process, it is possible for the graph to split into two or more components. In the present system it is not possible for disconnected components to join together again. Unlike some models of neural development (for example, Miller (2021)) there is no underlying spatial model by which nodes can "move towards" each other and connect. Nodes only "know about" their connected neighbours; once a graph has split, there is no possibility of the nodes in one component forming a connection with the nodes in the other. This mirrors some biological processes such as cell division or indeed reproduction of larger organisms.

When a graph splits, some of the components may enter an attractor, whilst others carry on developing. This behaviour is illustrated in Figure 4. The individual illustrated was found during the novelty search, and is highlighted in Figure 3. The overall graph has a transient of 35 steps and an attractor cycle of 60 steps. However, it results in three separate components. Looking at the space-time diagram one can see that each of the separate components has a much shorter attractor cycle: the component on the left ends in a point attractor with two nodes; the middle component has a four step cyclic attractor with 109 nodes; the right component has a 15 step atrractor cycle which fluctuates in size between 4 and 15 nodes. Since four and 15 share no factors, the two cycles of the middle and right components create an overall attractor cycle at the system level of 60 steps.

The novelty search used above defines attractors only at the whole-system level, but this example makes it clear that individual graph components can have their own attractor behaviour. The example in Figure 5 illustrates a case in which the graph splits in two, with one component quickly entering a point attractor, whilst the other continues to grow

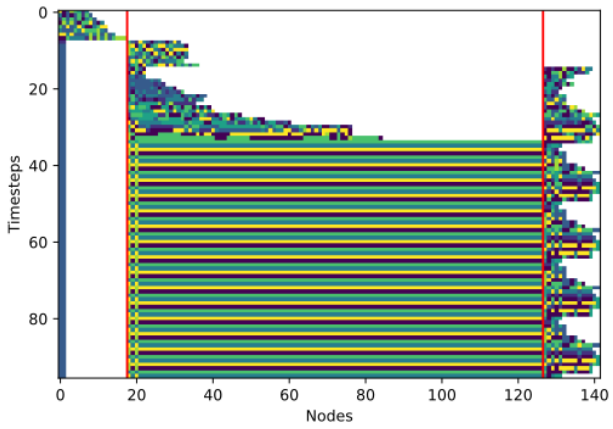

Figure 4: A space-time diagram of the individual highlighted in the 8-state novelty search in Figure 3. The space-time diagram has the same format as those introduced in Waldegrave et al. (2023), ie. the node states are represented by different coloured squares concatenated into a line (the ordering is arbitrary: no information about the graph structure is shown). When nodes are added they are appended to the right hand end of the line, and when they are removed the gap is closed and the line shrinks. The timesteps proceed down the y-axis. When the graph divides into separate components, a red line is drawn and the second component is drawn to the right of it. In this case the graph divides twice: the initial seed graph of 8 nodes grows for 7 timesteps before splitting into two approximately equal sized components, one of which rapidly shrinks, one of which continues to grow. A third component is split off at about timestep 15.

in a fluctuating manner. This individual is not plotted in the novelty search charts in Figure 3 because a system-level attractor is not found within the arbitrary size limit: this individual is classed as having runway growth. It is possible that a system-level attractor would be found if it was run for longer. A complementary visualisation of this behaviour is shown in Figure 6, which shows the size of each graph component over time. This example illustrates that even when an overall attractor is not found, the behaviour can still display a high degree of complexity.

Future work will explore the behaviour of the separate components of graphs that have split, by continuing their development independently. It would be particularly exciting to identify behaviour where a seed graph grows to a certain point and then part of it splits off to create a new seed graph identical to the original. Such self-replicating behaviour would have clear biological parallels. Something approaching this behaviour is shown in Figure 7. Here the seed splits into two identical potions, each of which grow for several more steps before reaching point attractors. This is not true self-replication as seeds do not continue to be produced, but rather a one-off "twin development" as the initial

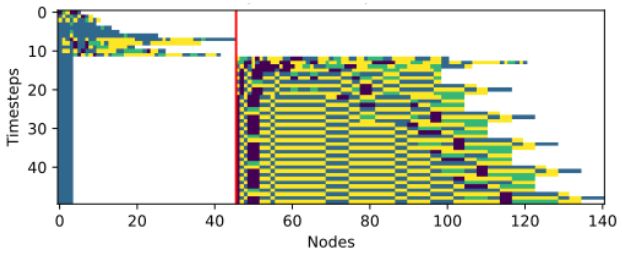

Figure 5: A space-time diagram for a four-state system in which the graph splits into two components at the 12th timestep of development. The component shown on the left settles into a four node point attractor, whilst the component on the right continues to grow with a fluctuating rhythm. The diagram is arbitrarily cut off at 50 timesteps; it is possible that the right hand component would settle into an attractor if the system was run for longer.

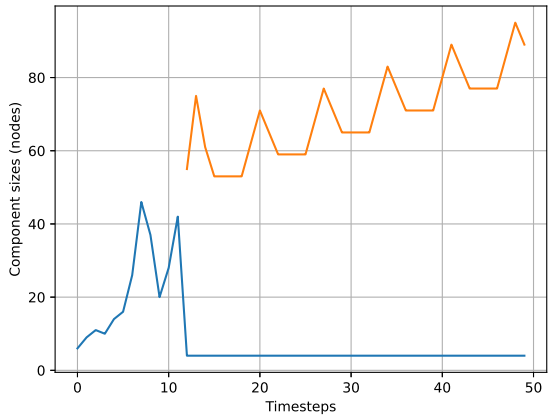

Figure 6: A different visualisation of the system shown in Figure 5, showing the sizes of the two graph components. The diagram is arbitrarily cut off at 50 timesteps.

seed has split into identical portions. This system is further illustrated in Figure 8, which shows the seed graph splitting into two components in the first step of development, and the final graph that each component ends up as, in a point attractor.

Periodic Detachment

Whilst it has not yet been possible to find the self-reproducing behaviour described above, several examples have been found in which elements are detached from the main body of the graph in a periodic cycle. An example is shown in Figure 9. The system as a whole has not found an attractor, since the overall number of nodes continues to increase. However, if one disregards the singleton nodes that are cast off, the main graph component has a six-step attractor cycle. This behaviour is reminiscent of a "glider gun" in

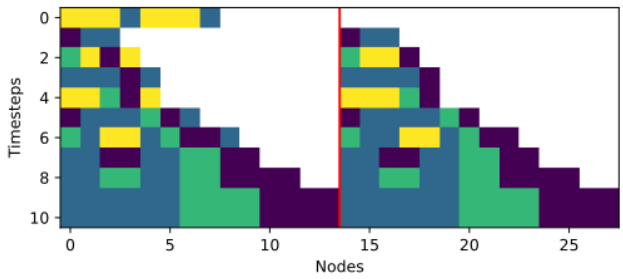

Figure 7: A space-time diagram for a four-state system in which the eight node seed graph immediately splits into two identical three node graphs due to the removal of two nodes. Each component traces the same course of development for eight more steps before reaching a point attractor. Note that the ordering of nodes in each half is arbitrary and differs somewhat (so the colours are not in the same order), but the two graphs are indeed isomorphic.

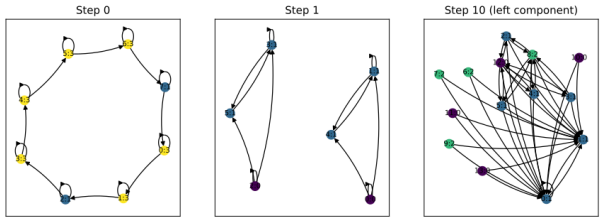

Figure 8: The actual graph developed in the space-time diagram in Figure 7, shown at step 0 (the seed graph), step 1 (when it splits into two identical components), and step 10 (when each component has reached a point attractor; only one component is shown).

Game of Life (Gardner, 2001), although unlike a glider gun, the emitted elements remain inert. Figure 10 shows the main graph component when it is at the smallest size in its cycle (six nodes).

Conclusion

We have shown that the DGCA systems introduced in Waldegrave et al. (2023) are evolvable by using crossover and mutation operations on the weight matrices of the SLPs that define the transition rules of the system. We used novelty search rather than goal-based evolution to demonstrate this, pushing the system into more extreme areas of a behaviour space defined by the transient length and attractor cycle length. Although we have not formally defined what we mean by "rich" behaviour, this experiment demonstrates that the system is capable of a wide range of complex behaviour. Future work will seek to use more formal measures to assess the complexity of the behaviour found.

The selected examples discussed highlight that some of the observed complexity comes from the different behaviour of disconnected graph components. Even if the whole sys-

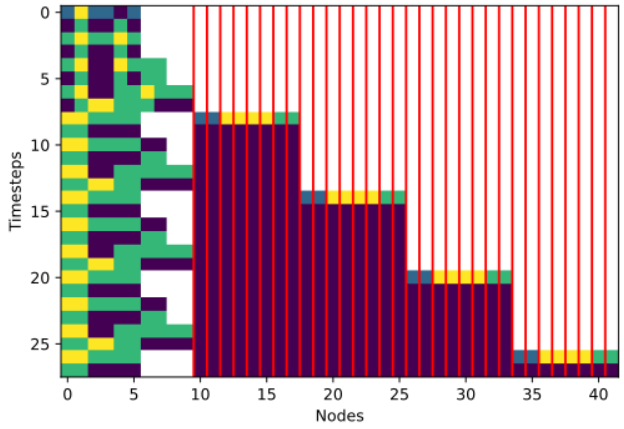

Figure 9: A space-time diagram for a four-state system with interesting periodic behaviour. The largest component of the graph (shown on the left) quickly settles into a six step cycle. However, at the beginning of each cycle, it throws off eight singleton nodes, each of which changes state once before remaining static. As before, a red line is drawn between disconnected graph components. The seed graph in this example is a six-node Kautz graph rather than the eight node ring used in the novelty search experiment. This behaviour was found during an earlier random search.

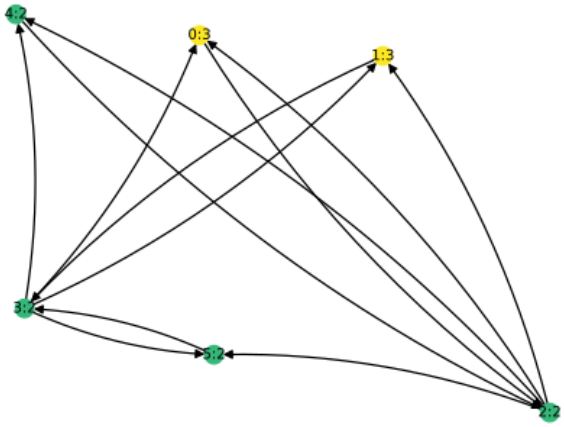

Figure 10: The main graph component from the system shown in Figure 9, shown at the point in its cycle where it has six nodes. Like Gosper's glider gun in Game of Life, this is a relatively simple structure which emits elements at a certain point in its cycle.

tem appears to have a very long attractor cycle, or indeed if it does not find an attractor at all, individual components can have their own shorter attractor cycles.

Future work will seek to use goal-directed evolution to discover particular behaviours such as self-reproduction, as well as to guide development towards graphs with particular properties. These could be either structural properties (eg. node degree distribution) or functional properties (usefulness in computational tasks, for example when the developed graphs are used as substrates for Reservoir Computing).

Source Code

Source code for the DGCA system and the experiments conducted here can be found at https://github.com/rvrsdl/alife2023_dgca. This repository also contains CSV files with the SLP weights for the systems shown in Figures 1, 4, 5, 7, 9.

References

Dale, M., Miller, J., Stepney, S., and Trefzer, M. (2019). A substrate-independent framework to characterize reservoir computers. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475:20180723.

Gardner, M. (2001). *The Colossal Book of Mathematics: Classic Puzzles, Paradoxes, and Problems: Number Theory, Algebra, Geometry, Probability, Topology, Game Theory, Infinity, and Other Topics of Recreational Mathematics*. Norton, New York, 1st ed edition.

Grattarola, D., Livi, L., and Alippi, C. (2021). Learning Graph Cellular Automata. *arXiv:2110.14237 [cs]*.

Gruau, F. (1994). *Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon.

Harvey, I. (2011). The Microbial Genetic Algorithm. In Kampis, G., Karsai, I., and Szathmáry, E., editors, *Advances in Artificial Life. Darwin Meets von Neumann*, volume 5778, pages 126–133. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jaeger, H. (2001). The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148.

Lehman, J. and Stanley, K. O. (2011). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223.

Marr, C. and Huett, M.-T. (2009). Outer-totalistic cellular automata on graphs. *Physics Letters A*, 373(5):546–549.

Miller, J. F. (2021). IMPROBED: Multiple Problem-Solving Brain via Evolved Developmental Programs. *Artificial Life*, pages 1–36.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing Neural Cellular Automata. *Distill*, 5(2):e23.

Rozenberg, G., editor (1997). *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific, Singapore ; New Jersey.

Tavares, J., Kreutzer, C., and Fedor, A. (2015). Neuro-Cellular Automata: Connecting Cellular Automata, Neural Networks and Evolution. In *Complex Systems Summer School*.

Waldegrave, R., Stepney, S., and Trefzer, M. (2023). Developmental Graph Cellular Automata. *ALife Conference 2023*.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, Champaign, IL, 1st edition edition.

Wuensche, A., Lesser, M., and Lesser, M. J. (1992). *Global Dynamics Of Cellular Automata: An Atlas Of Basin Of Attraction Fields Of One-dimensional Cellular Automata*. Andrew Wuensche.

Wulff, N. and Hertz, J. A. (1992). Learning Cellular Automaton Dynamics with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann.