A Model of Cell Division and Differentiation with 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

Developmental Graph Cellular Automata (DGCA) are a type of Graph CA which can modify the structure of the graph through their update rule. This can lead to the phenomenon of graphs dividing into disconnected components. We use this as an endogenously driven analogy of cell division. Each connected graph component (or "cell") can exhibit distinct dynamics, including entering an attractor cycle. This can be used as an analogy of cell differentiation, in which each graph component can end up with a completely different structure. A single DGCA rule ("genome") can give rise to a system of dividing graph components ("cells") each of which may end up in a different attractor with different graph structure to become a fully differentiated cell. All this can arise from a relatively compact growth rule being applied repeatedly to a simple seed graph, showing the emergence of complexity from simple processes that is typical of living systems. We present the results of some experiments to evolve growth rules to produce a desired level of cell differentiation.

Code available at: https://github.com/rvrsdl/alife2025

Introduction

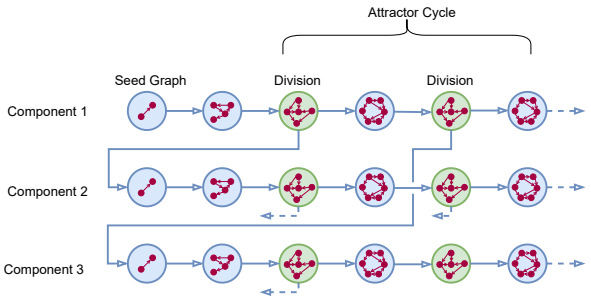
Cellular Automata (CA) are famous for being able to create complex patterns using a simple rule which is applied in parallel at an array of linked nodes.[1] Each node can take on one of a finite number of states and the update rule determines the next state of each node based on the states of its neighbours. Each node has access only to local information from its neighbourhood, and yet over time complex global patterns can emerge. Traditionally, CAs operate on a lattice of nodes with a fixed (or infinite) size: this represents the "space" or the "universe" within which patterns of node states can emerge. Here, we use a different paradigm, in which the CA operates on an arbitrary graph rather than a lattice, as in Graph CAs (Marr and Huett, 2009). In addition, the DGCA model allows the node update rule to modify the structure of the graph itself. Both the patterns of node states and the structure of the graph are driven by linked dynamics: in this sense it is like a Dynamical System with Dynamical Structure, $(DS)^2$ (Giavitto et al., 2011).

Whereas in lattice-based CAs, much of the literature is concerned with the formation of static or moving patterns of node states, such as "gliders", here we are concerned with the creation of arbitrary graph structures. The graphs in the DGCA model are not spatially-embedded so there is no sense of them moving through space as with gliders. However, it is possible for the graph to divide into separate components, because the structural update process allows for the removal as well as the addition of nodes. Since the graphs are not spatially-embedded, once a graph has divided into separate components, there is no way for these components to reconnect or even have any communication with each other.

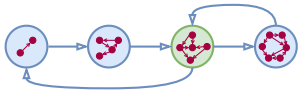
One phenomenon which is frequently observed when running the system is that a small "seed" graph can develop into a larger graph and at a certain point may split off a small component which is the same as the original seed graph. Since the system is deterministic, this new seed graph will follow exactly the same process. Sometimes the larger "parent" component enters a point or cyclic attractor in which it splits off a seed each time it goes round the cycle. A diagram of this prototypical case is shown in Figure 1. There are various different biological analogies which could be drawn with this process, including most obviously that of a whole organism reproducing itself. This would necessarily be asexual reproduction since, as mentioned, there is no way for the graph components or "organisms" in this analogy to encounter each other again after they have divided. In the prototypical example described here, bacterial division could be a good analogy.

However, in many cases, when a graph divides it is *not* the case that a perfect copy of the original seed

[1]In most of the literature the nodes of a CA are referred to as "cells". In this paper, for the avoidance of confusion we use the term "nodes", since we are using an analogy with biological cells at a higher level of organisation.



(a) Full schematic of graph division.



(b) More abstract representation of the same process.

Figure 1: A prototypical graph division is shown in (a). Each large blue/green circle represents the state of a graph component at a point in time (the actual graph is shown inside the circle). A seed graph develops for two steps before dividing into two components (indicated by the fact that two arrows are coming out of the green circle). One of the components is a copy of the seed graph which undergoes the same growth process. The other enters a two step attractor cycle, dividing every other step. All division points are shown in green, with two arrows coming out of them. To save space, not all products of division are shown. The same system is shown in (b) in an abbreviated form: when a previously seen graph component is created, the arrow points back to that circle. This depiction highlights that there are two interlocking cycles.

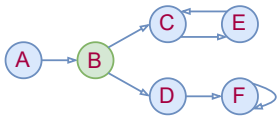


Figure 2: A different attractor structure. This diagram uses the same representation as Fig. 1b except that graph states are abbreviated to symbols. Seed graph A develops into B, which splits into components C and D. Component C enters an attractor cycle (C–E), and component D goes through one more developmental step before reaching a point attractor, F.

graph is produced. This can lead to the kind of situation shown in Figure 2, where a seed graph divides into two or more components at some point during its developmental trajectory. The components produced by this division may continue developing and eventually reach separate attractors. This invites comparison with biological cell differentiation via asymmetric division, where a pluripotent stem cell divides and after some intermediate steps a variety of specialised cells are produced. In this analogy the specialised nature of the cells is represented by the fact that the graph components may have very different structures.

Whilst this analogy is clearly not a realistic model of biological cell division, it allows discussion of division and differentiation in the same framework. This paper examines both aspects in tandem, analysing the complex graph division process and the variety of differentiated graph structures it can create.

A more pragmatic motivation for this work is that the growth rule of a DGCA can be considered a highly compressed encoding of a graph structure. With only the seed graph (which can be a single node) and the growth rule (which in the present implementation consists of 180 floating point numbers), it is possible to develop a graph of arbitrary size through the deterministic running of the DGCA. Since the graph can divide, we could say that a single growth rule can encode a collection of graphs. It would be noteworthy if the graphs produced had appreciably different structure from each other. This would mean that a single compressed encoding could be unfolded into a number of distinct graphs which could have different functional purposes.

Background

DGCA

The DGCA system was introduced in Waldegrave et al. (2023a). It uses a directed graph in which each node has a state from a finite alphabet Σ. At each timestep the node states are updated (as in a conventional graph CA), as well as the graph structure. Nodes may divide or be removed from the graph. In the case of node division, the child node may be connected to some or all of the local neighbourhood of the parent node. Waldegrave et al. (2024) improved the model to allow greater specification of child node connectivity, allowing complex structure to be created more easily. It is this model which is used here. Whilst edges between existing nodes may not be modified in the model, an equivalent effect can be achieved by adding a child node (with specified connections) and removing the parent node.

The graph at a particular point in time is represented as an adjacency matrix $\mathbf{A}$ of size $(n \times n)$, where n is the number of nodes in the graph, and a node state matrix $\mathbf{S}$ with size $n \times s$, where $s = |\Sigma|$. Each row of $\mathbf{S}$ represents the state of that node as a one-hot encoded vector. The update function uses two small neural networks: one for node state update and one for structural updates. In the model used in this paper we use single-layer perceptrons (SLP) with one-hot activation functions. The input to these networks are the counts of incoming and outgoing nodes in each state, as well as the central node's own state, allowing the implementation of outer-totalistic update rules. This input is constructed as the horizontally concatenated matrix $[\mathbf{S} \mid \mathbf{A} \cdot \mathbf{S} \mid \mathbf{A}^T \cdot \mathbf{S}]$. For the node state update, the one-hot activation allows the network to output the new node states directly. The output of the structural update SLP specifies the node

action (deletion, division, stasis) and in the event of division, specifies how the child node should connect to the parent node's neighbourhood (connections to remote nodes are not allowed as the nodes only have access to local information). The experiments in this paper use a three-state DGCA ($s = 3$) which results in 180 weights in the SLPs. See Waldegrave et al. (2024) for a fuller description of the model.

The graph structure and node states together form the *graph-state*. If the system reaches a graph-state which has been seen before then it has entered an attractor cycle. In other words if the graph is isomorphic (conditioned on node states) with a previous graph, we can say that the system has returned to a previous graph-state. Since the system is deterministic, it will repeat this cycle forever.

The growth rule is effectively encoded by the weights of the SLPs. These can be evolved to produce different dynamics (transient and attractor cycle lengths) as shown in Waldegrave et al. (2023a,b); they can also be evolved to develop graphs with different structural properties, as shown in Waldegrave et al. (2024).

Related Work

Grattarola et al. (2021) introduce a model of Graph CA with a neural network (NN) based update function. They show that the NN can be trained to reproduce the behaviour of different CA rules. A second experiment interprets the node state vectors as spatial coordinates and shows that GNCA can be used to implement the flocking behaviour seen in the 'boids' model of Reynolds (1987). The number of nodes remains fixed throughout.

Najarro et al. (2023) combines GNCA with a node-addition model (also NN based) to grow neural networks. The development rules can be adapted to produce performant networks either through evolution or through gradient descent (only in the case of feedforward networks).

In contrast to these models, DGCA is focused on creating specific network structure through a dynamical development process which can come to a halt by itself if it reaches an attractor. Importantly for the present work, nodes may be added *and* removed, allowing graph division.

Cell Division and Differentiation

C. H. Waddington's influential metaphor of the *epigenetic landscape* imagines developing cells as marbles rolling down a slope containing branching valleys (Waddington, 1957). They eventually come to rest at the lowest points of the landscape, representing fully differentiated cell types. This view has enabled cell biologists to draw on the dynamical systems literature when modelling cell development. This usually takes the form of continuous dynamical systems that are open to environmental influence (e.g. Brackston et al. (2018)).

One example of a *discrete* dynamical system being used to model epigenesis is the Random Boolean Network (RBN) model of Kauffman (1969). The overall state of the RBN, which encompasses all of the individual node states, is updated at each timestep. The system may move from its starting state, along a transient and end up in an attractor. The RBN is used as a model of a gene regulatory network (GRN). Different attractor cycles in an RBN state space are analogised different operational modes of the GRN, indicating different cell types. However, since RBNs are deterministic, any one starting state leads to only one attractor cycle. In order to model movement between cell types, Kauffman introduces small perturbations, effectively making the RBN into an open dynamical system. Some of these external perturbations are able to push the system into different basins of attraction. This is similar to Waddington's original conception of the epigenetic landscape, in which the "marbles" are pushed into different valleys in the slope by exogenous forces including environmental factors and external signals. Cell division is not modelled in RBNs.

Here, we use unequal graph division in DGCA as an analogy of asymmetric cell division (ACD), giving an endogenous mechanism for moving between cycles. ACD has been documented in *Drosophila* neurogenesis, where segregation of transcription factors between daughter cells leads to differing cell fates (Matsuzaki, 2000). Even the differing cytoskeletons of daughter cells may differentially affect regulatory pathways (Dye et al., 1998). Recent work on single-cell transcriptomics reveals further mechanisms by which differing transcription factor regulatory networks (TFRNs) are expressed in daughter cells (Zhang et al., 2023). In the DGCA analogy, the actual graphs "inside" cells could be thought of as representing these different TFRNs, or as a whole complex of cell characteristics (e.g. including the cytoskeleton).

In the DGCA analogy of cell development we start with a seed graph-state, which can be thought of as a germ cell or pluripotent stem cell. A graph-state represents the state of the cell at a particular point in time. Some of these states are transitory and can be thought of as states that a cell passes through on its way to becoming specialised (for example, graph-state D in Figure 2). Other graph-states form part an attractor: these can be thought of as the states experienced by a fully specialised "final" cell type. It is not the case that a specialised cell is always in a fixed state. As in the RBN

model, different cell types are represented by different attractor cycles.

In Figure 2, the cycle $C \leftrightarrow E$ represents one cell type which can exist in either state C or state E. It is also possible to have a differentiated cell which exists in only one state, as in graph-state F, which is in a point attractor.

In using the DGCA as an analogy for cell division and differentiation, no external perturbation or signalling is needed in order for multiple attractors in graph-state space to be explored (unlike the RBN model which uses external noise to move between attractors). When a graph divides asymmetrically, the products of division can follow their own trajectories, potentially into different attractors in graph-state space. The DGCA model is a *closed* and deterministic discrete dynamical system.

The biological analogy breaks down when we consider that individual graph-states can be shared between cycles, as is the case with the third graph-state in Figure 1b. In biology, it is hard to imagine any meaningful sense in which it could be said that a liver cell is in the "same state" as a neuron, for example. It is worth noting that in the DGCA system, a graph-state can form part of more than one cycle only if it leads to a division. A "terminal cycle", which does not lead to any division (such as $C \leftrightarrow E$ in Figure 2), can contain only graph-states which uniquely belong to that cycle.

Graph motif analysis

In order to evaluate the level of "cell differentiation" that the DGCA system can produce, we need a measure of the difference between graph-states. Various options for this exist including the graph edit distance, the adjacency spectral distance, and the difference between various graph features, such as degree distribution (Wills and Meyer, 2020). Some of these are not appropriate for comparing graphs of different sizes, as we wish to here, whilst others focus on global properties rather than local structure. We use the difference between motif significance profile (MSP). The MSP is frequently used to characterise graphs in the systems biology literature (Alon, 2007; Milo et al., 2002)[2]. It summarises the graph microstructure by counting the occurrence of each of the 13 possible three node subgraphs (triads) and comparing the observed number with the counts in an ensemble of randomised graphs of the same size and degree distribution. The purpose is to discover which triads occur more frequently than we would expect, possibly indicating that they have been selected for by evolution (Shellman et al., 2014). The triads can also be linked with computational primitives (e.g. inhibitory feedback loops), enabling the network microstructure to be linked with functional properties. The result is expressed as a normalised vector of 13 Z-scores indicating the enrichment or diminution of each triad compared to the random graphs. Graph MSPs have been used to characterise various kinds of biological networks, GRNs, NNs, protein-protein interaction networks, signal transduction networks. This can enable the categorisation of networks into families based on their structure (Milo et al., 2004).

Since the MSP of a graph is a vector of 13 numbers we use the Mean Absolute Difference (MAD) between two MSP vectors as a measure for the difference between two graphs:

$$MAD = \frac{\sum_{i=1}^{13} |x_i - y_i|}{13} \tag{1}$$

Waldegrave et al. (2024) used DGCA to create networks with MSPs similar to various real biological networks, by using a genetic algorithm to search over the DGCA growth rules. That work did not analyse graph division so calculated MSP values for whole graphs (potentially containing disconnected components). Here we build on that work, analysing graphs produced by a DGCA growth rule at the level of connected components. We aim to use a *single* growth rule to create multiple graphs that have significantly different MSPs, as an analogue of cell differentiation. Using the MSP to describe the graphs works well with this analogy, since it highlights that different "cell types" may have different functional properties.

Methods

A DGCA system can in general be represented using a dynamical system equation:

$$g_{t+1} = f(g_t; \theta) \tag{2}$$

where f is the update function, parametrised by θ, representing the weights of the neural network used to calculate state and structural changes at each node, and g is a vertex labelled directed graph (actually represented by the $\mathbf{A}$ and $\mathbf{S}$ matrices as explained above. Under previously introduced DGCA models (Waldegrave et al., 2023a, 2024), the graph produced at each update step may or may not be connected, and the update function takes one graph as input and produces one graph as output:

$$f : G \to G \tag{3}$$

where G is the set of all vertex-labelled directed graphs.

However, since this paper aims to examine graph division behaviour, we here constrain all graphs to be

[2] In much of the literature this is called the Triad Significance Profile (MSP). We here use the acronym MSP in order to avoid confusion with the Travelling Salesman Problem!

Algorithm 1 Recursive DGCA update

```
f ← parametrised update function
g ← single node seed graph
archive ← ∅
max_nodes ← 300
procedure RunDGCA(g)
    if (g ∈ archive ‖ #(g) = 0
        ‖ #(g) > max_nodes) then
        return ▷ graph seen before/empty/too big
    archive ← archive ∪ g
    new_graphs ← f(g) ▷ Run a step
    for idx = 1 to |new_graphs| do
        RunDGCA(new_graphs[idx])
```

weakly connected (i.e. connected disregarding the direction of the edges). We indicate the set of all weakly connected vertex-labelled graphs by $G_{conn} \subset G$. The update function can then be said to take in a single graph but produce zero or more graph components:

$$f : G_{conn} \to \mathbb{P}\, G_{conn} \tag{4}$$

where $\mathbb{P}\, G_{conn}$ indicates the power-set of connected graphs.

Running the system involves recursively updating each component currently in the system, as shown in Algorithm 1. Starting with the seed graph (for which we use a single node graph for all experiments in this paper), we iteratively apply the update function. After each application, we examine how many connected graphs have been produced. We then apply the update function to each of them in turn, and repeat the process. We record each connected graph in an archive together with the graph at the previous timestep. This allows us to construct a graph-state transition diagram, as shown in stylised form in Figures 1b & 2, with a real example shown in Figure 5. If at any point in the run a graph-state is produced which is already in the archive, we do not need to continue running that element: since the system is deterministic, its onward trajectory is already known.

Experiments

Maximising Differentiation

We first conduct an experiment to investigate whether a single DGCA growth rule can produce what we have described as "cell differentiation": whether it can produce separate graph-states that have significantly different structure from each other. As discussed above, we use the MSP to characterise graph structure. MSPs are calculated by comparing the counts of the 13 possible triads in the subject graph to the counts in 500 randomised graphs to generate a vector of Z-scores. This is computationally expensive as it involves enumerating the triads in the subject graph and the randomised graphs. The RAND-ESU algorithm introduced in Wernicke (2006) and implemented in the Python `graph-tool` library (Peixoto, 2014) is used for this. The MAD of the MSP vectors of two graphs is used as a measure of the distance between them (Equation 1).

Because we use cycles in the graph-state transition diagram as analogy of cell types, we wish to examine the difference between graph-states on different cycles in the graph-state transition diagram. We choose the largest graph-state (by number of nodes) as the representative graph-state for that cell type. Because the calculation of MSPs is computationally expensive, we calculate them only for the two largest graph-states on different cycles in the whole transition diagram. The chosen graph states must also meet some conditions of size and connection density, because the MSP metric does not work well on graphs that are too small or too densely connected. The graphs must be between 100 and 300 nodes in size and have a connection density between 0.002 and 0.04.

We use a genetic algorithm (GA) to maximise the MAD between the MSPs of these two chosen graph-states. The "genomes" in the GA are the DGCA SLP weights. We use a population size of 100 and use "natural elitism", conserving all individuals from the parent population that have a higher fitness than the offspring population into the next generation.

The results are shown in Figure 3.[3] After only a small number of generations, the GA was able to find growth rules that could generate graph-states on separate cycles ("cells") with significantly different structure from each other. The MAD between two biological MSPs from very different families, the *E. Coli* GRN and the *C. Elegans* NN is only 0.25, which was exceeded in this experiment after 18 generations.

Maximising Cycles and Differentiation

We hypothesise that an individual growth rule will have more likelihood of creating highly differentiated "cells" if it produces more and longer cycles. Exploring more attractor cycles in the graph-state space gives more opportunity to encounter graph-states with very different structure. Furthermore, the longer the length of the cycles, the more steps the cell has available to develop differentiated structure.

To test this hypothesis, the second experiment uses a multi-objective GA to maximise MSP MAD between the two largest graph-states on different cycles (as before) and also to maximise the number of cycles and the average cycle length.

[3] In all experiments, five runs were conducted, with the best results shown.

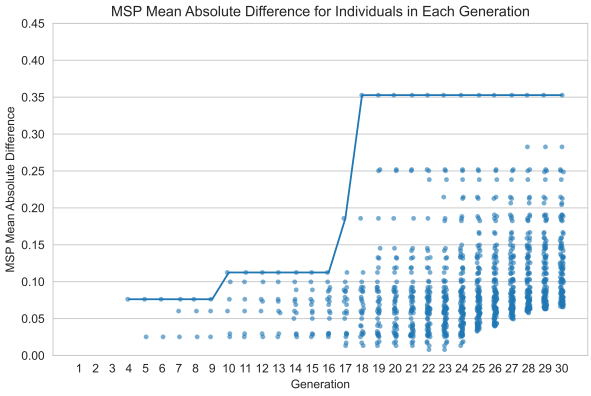


Figure 3: Maximising Differentiation experiment. The MAD between the MSPs of the two largest graph-states on different cycles are plotted for each individual in columns for each generation. Not all individuals have two valid graphs on different cycles to compare. The line shows the best (highest) MAD in each generation.

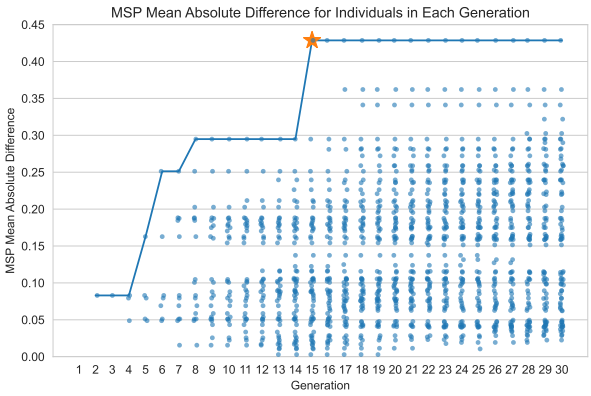


Figure 4: Maximising Cycles and Differentiation experiment. Just the MSP MAD is plotted for ease of comparison with Fig. 3, although three features are used in the multi-objective GA. The individual growth rule which generates maximum differentiation (MSP MAD) was found in generation 15 and is highlighted with a star.

We use the NSGA-II algorithm introduced by Deb et al. (2002), which uses "natural elitism" by performing a non-dominated sort on the parent population and offspring population together and keeps members of the successive non-dominated (Pareto) fronts until the next generation is full. It also maintains population diversity by using a crowding distance metric for selection.

The results are shown in Figure 4. As hoped, in the multi-objective context the GA is able to maximise MSP MAD faster, breaching the biologically relevant 0.25 threshold after only 6 generations, and reaching a high point of 0.43 in the 15th generation. This suggests an interesting interplay between the structural growth of the graphs and the cycle dynamics, which are an emergent property of the system.

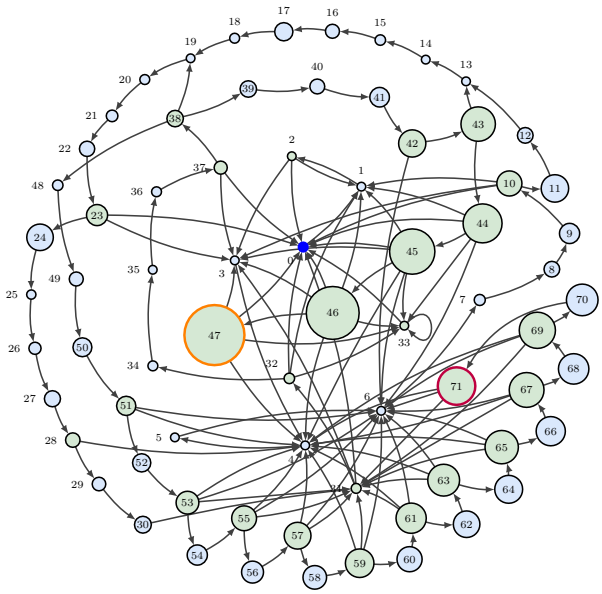
The graph-state transition diagram for the best individual found in this experiment is shown in Figure 5a. This is the same kind of diagram as in the simplified example in Figure 1b. In total there are 70 cycles (graph-states may be members of more than one cycle), with an average cycle length of 35.8 steps. The two largest graph-states on separate cycles are those numbered 47 and 71, and circled in orange and purple respectively, containing 292 and 114 nodes respectively. (Whilst graph-states 43–46 are bigger than graph-state 71, they are on the same cycle as 47 so are not considered a different "cell type".) The actual graph-states 47 and 71 are shown in Figures 5c & 5d, along with their very different MSPs in Figure 5b.

Natural MSP Targets

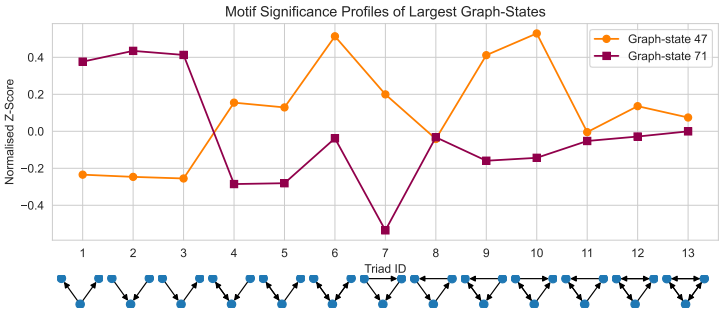
Having shown that it is possible to find single growth rules that produce separate graphs with dissimilar structures, we next evaluate whether a single growth rule can produce graphs that are close to two or more target graphs. Waldegrave et al. (2024) use a GA to search for growth rules that generate graphs with MSPs close to those of two natural networks: the *E. Coli* GRN and the *C. Elegans* NN. However, that work conducted two separate searches: each trying to optimise for proximity to only one target. Since here we are concerned with differentiation, the present experiment uses multi-objective optimisation (again using NSGA-II) to find a single growth rule that produces two separate graphs each of which has a small MSP distance to its respective target. We follow Waldegrave et al. (2024) in using the *E. Coli* GRN and the *C. Elegans* NN as target networks. This is a challenging goal as these networks have been identified by Milo et al. (2004) as belonging to different "families" because of their very different MSPs. If the DGCA system can use a single growth rule to develop two networks matching each of these profiles, this would be a good indication of the power of the the division and differentiation process described here.

As in the second experiment (Fig. 4), it was found that maximising the number of cycles and average cycle length sped up convergence. The GA used in this experiment therefore has *four* objectives: minimising the MSP error of the two largest graph-states (on different cycles) with respect to each of the two target MSPs, and maximising the number and length of cycles.

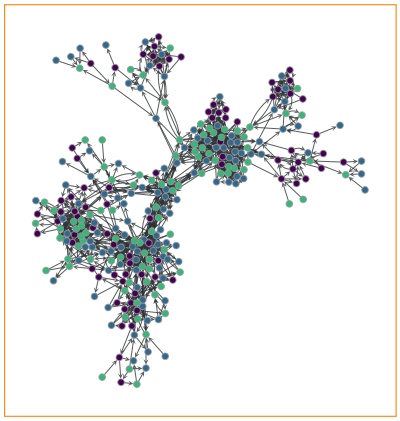
In order to prevent the evolution of a single graph-state with an MSP somewhere in between the two target MSPs, we allow each graph-state to be compared to only one or the other targets, not to both. We again use NSGA-II with a population size of 100. We use the same size and connection density conditions as before for choosing "valid" graph-states for which to calculate



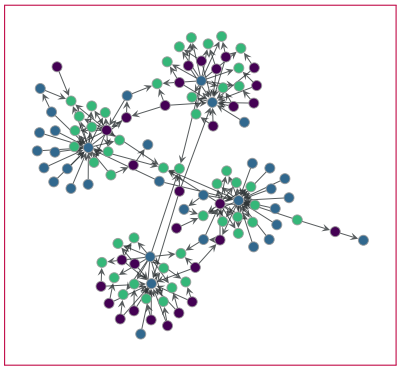
(a) Graph-state transition diagram



(b) Motif Significance Profiles of graph-states 47 and 71.



(c) Graph-state 47: orange circle in (a)



(d) Graph-state 71: purple circle in (a)

Figure 5: (a) shows the graph-state transition diagram for the best individual growth rule found in Experiment 2 (highlighted with a star in Fig. 4. Each circle represents one graph-state and has an area proportional to the number of nodes in the graph. Arrows represent transitions between graph-states on each DGCA timestep. The starting state (single node seed graph) is highlighted in bright blue and labelled 0. Graph-states which divide are coloured green (as in Figs. 1 & 2). The two largest graph-states on different cycles are 47 and 71 and are circled in orange and purple respectively. Their actual graph states are shown in (c) and (d). Their two Motif Significance Profiles are shown in (b), indicating the enrichment or diminution of the 13 three-node motifs compared to random networks.

the MSP.

The results of the experiment are shown in Figure 6. Note that in this experiment the aim is to *minimise* the MAD of the found MSPs versus the target MSPs, whereas in the first two experiments it was to *maximise* the MAD of the MSPs of two graph-states produced by an individual growth rule.

Figure 7 shows the actual MSPs of graph-states produced by the best growth rule, which was found in generation 16 of the GA. The red lines indicate the generated MSPs and the blue lines indicate the target MSPs of the *E. Coli* GRN and *C. Elegans* NN. The MSP MADs versus each target are 0.06 and 0.10 respectively. Whilst this is a greater error than was found by Waldegrave et al. (2024) in evolving separate growth rules for each target, it is nevertheless noteworthy that a single growth rule can generate two graphs with MSPs close to these two very different targets.

Conclusion

This work has illustrated behaviour of the DGCA system which is analogous to cell division and differentiation. Normally in a deterministic dynamical system such as DGCA, we would expect that given a starting

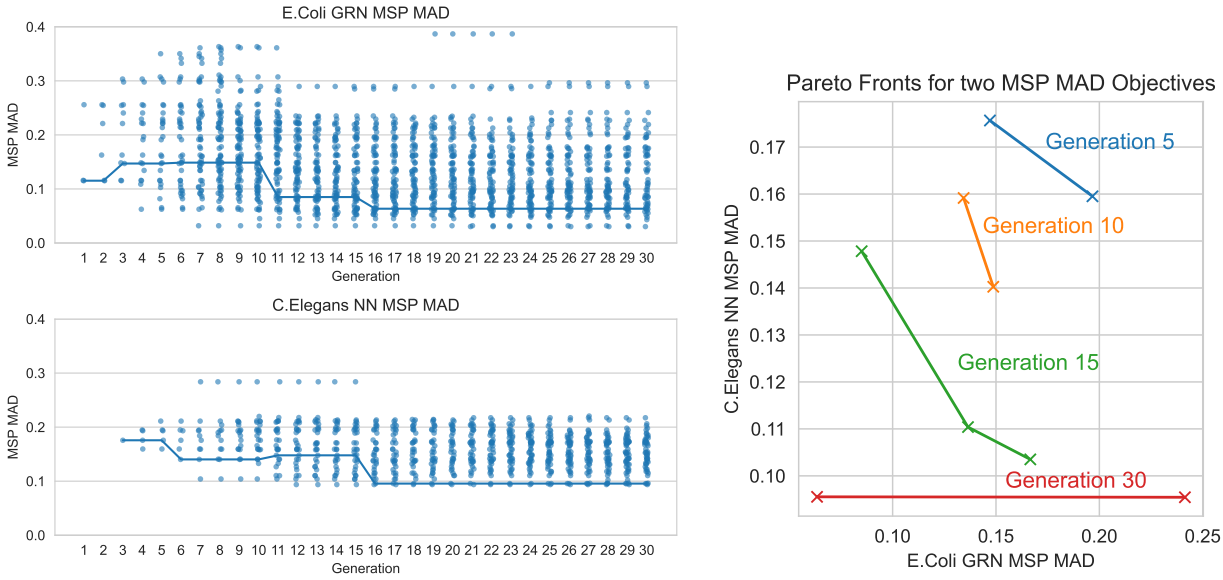


Figure 6: The swarm plots show the MSP MAD for individuals in each generation vs. the *E. Coli* GRN MSP (upper chart) and the *C. Elegans* NN MSP (lower chart). The lines on these charts indicate the best individual in each generation, averaging across both errors. The right hand chart shows the Pareto fronts across the two MSP objectives for selected generations.

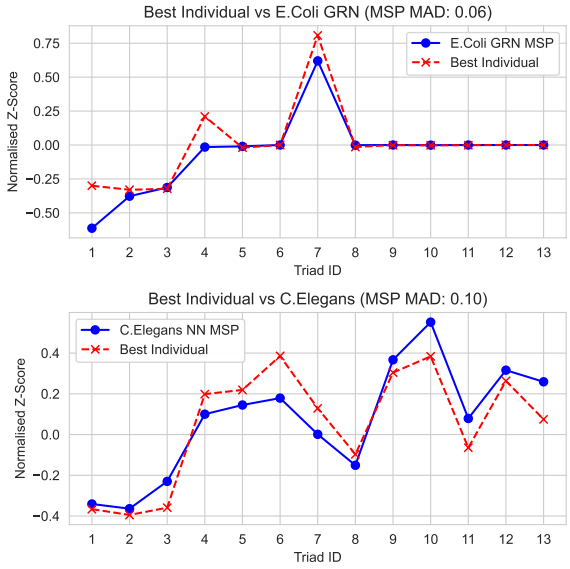


Figure 7: The actual MSPs of the two largest graph-states on separate cycles (generated by a single genome / growth rule). They are plotted against their most closely matching target MSPs the *E. Coli* GRN and *C. Elegans* NN respectively. See Fig. 5b for the meaning of the triad IDs along the x-axis.

state and an update rule, at most one attractor could be reached. However, each time a graph-state undergoes division, it effectively gives us several new "starting states" each of which may continue into a separate attractor. In a sense this is an extension of Waddington's epigenetic landscape metaphor: each marble can now divide, and the products of this division may end up in different grooves in the landscape (i.e. different basins of attraction).

The number and length of cycles are emergent properties of the system, but surprisingly we are able to evolve them. We also find that greater differentiation of structure is possible when there are more and longer cycles. The final experiment shows that a single growth rule can be evolved to produce two graph-states which are close to the MSPs of two very different target networks. This highlights that the system can "encode" *multiple* network topologies using a *single* DGCA growth rule: this could have applications in compressing network representations.

Future Work

This work has only analysed the expressiveness of the model through the lens of triad motifs. Many other metrics are possible for analysing and comparing graph structure (e.g. the graph edit distance). Owing to the computational cost of calculating MSPs, this work has only examined two of the graph-states produced by each growth rule. Future work will examine more of the graph-states using a variety of metrics, in order to evaluate whether a single DGCA rule can generate whole ensembles of structurally distinct graphs.

References

Alon, U. (2007). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Number 10 in Chap-

man & Hall/CRC Mathematical and Computational Biology Series. Chapman & Hall/CRC, Boca Raton, FL.

Brackston, R. D., Lakatos, E., and Stumpf, M. P. H. (2018). Transition state characteristics during cell differentiation. *PLoS Computational Biology*, 14(9):e1006405.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Dye, C. A., Lee, J. K., Atkinson, R. C., Brewster, R., Han, P. L., and Bellen, H. J. (1998). The Drosophila sanpodo gene controls sibling cell fate and encodes a tropomodulin homolog, an actin/tropomyosin-associated protein. *Development (Cambridge, England)*, 125(10):1845–1856.

Giavitto, J.-L., Michel, O., and Spicher, A. (2011). Interaction Based Simulation of Dynamical System with a Dynamical Structure (DS)2 in MGS. *Proceedings of the 2011 Summer Computer Simulation Conference*, pages 97–106.

Grattarola, D., Livi, L., and Alippi, C. (2021). Learning Graph Cellular Automata. In *Advances in Neural Information Processing Systems*, volume 34, pages 20983–20994. Curran Associates, Inc.

Kauffman, S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467. Publisher: Elsevier.

Marr, C. and Huett, M.-T. (2009). Outer-totalistic cellular automata on graphs. *Physics Letters A*, 373(5):546–549. arXiv:0812.2408 [nlin].

Matsuzaki, F. (2000). Asymmetric division of *Drosophila* neural stem cells: a basis for neural diversity. *Current Opinion in Neurobiology*, 10(1):38–44.

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. (2004). Superfamilies of Evolved and Designed Networks. *Science*, 303(5663):1538–1542.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827. Publisher: American Association for the Advancement of Science.

Najarro, E., Sudhakaran, S., and Risi, S. (2023). Towards Self-Assembling Artificial Neural Networks through Neural Developmental Programs. In *Proceedings of the 2023 Artificial Life Conference*. MIT Press.

Peixoto, T. P. (2014). The graph-tool python library. *figshare*.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA. Association for Computing Machinery.

Shellman, E. R., Chen, Y., Lin, X., Burant, C. F., and Schnell, S. (2014). Metabolic network motifs can provide novel insights into evolution: The evolutionary origin of Eukaryotic organelles as a case study. *Computational biology and chemistry*, 53PB:242–250.

Waddington, C. H. (1957). *The strategy of the genes. A discussion of some aspects of theoretical biology. With an appendix by H. Kacser.* London: George Allen & Unwin, Ltd.

Waldegrave, R., Stepney, S., and Trefzer, M. A. (2023a). Developmental Graph Cellular Automata. In *Proceedings of the 2023 Artificial Life Conference*. MIT Press.

Waldegrave, R., Stepney, S., and Trefzer, M. A. (2023b). Exploring the Rich Behaviour of Developmental Graph Cellular Automata. In *Proceedings of the 2023 Artificial Life Conference*. MIT Press.

Waldegrave, R., Stepney, S., and Trefzer, M. A. (2024). Creating Network Motifs with Developmental Graph Cellular Automata. In *Proceedings of the 2024 Artificial Life Conference*. MIT Press.

Wernicke, S. (2006). Efficient Detection of Network Motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):347–359. Conference Name: IEEE/ACM Transactions on Computational Biology and Bioinformatics.

Wills, P. and Meyer, F. G. (2020). Metrics for graph comparison: A practitioner's guide. *PLOS ONE*, 15(2):e0228728.

Zhang, L., He, C., Lai, Y., Wang, Y., Kang, L., Liu, A., Lan, C., Su, H., Gao, Y., Li, Z., Yang, F., Li, Q., Mao, H., Chen, D., Chen, W., Kaufmann, K., and Yan, W. (2023). Asymmetric gene expression and cell-type-specific regulatory networks in the root of bread wheat revealed by single-cell multiomics analysis. *Genome Biology*, 24(1):65.