NTNU – Trondheim
Norwegian University of
Science and Technology

Using Genomic Parameters to Predict Structural Complexity in Artificial Organisms

John H. Anthony

Master of Science in Informatics
Submission date: December 2012
Supervisor: Gunnar Tufte, IDI
Co-supervisor: Stefano Nichele, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis investigates the relationship between genomic parameters and properties of the phenotype. More specifically, we claim that there is a connection between the genomic λ parameter and the structural complexity of artificial organisms, and that λ can be used to predict this structural complexity. We use a cellular automata as a model for developing artificial organisms. A measure for structural complexity is defined and then used in experiments to try to find evidence to support our claim.

Sammendrag

I denne oppgaven undersøker vi forholdet mellom genomiske parametre og egenskaper ved fenotyper. Mer presist hevder vi at det finnes en sammenheng mellom den genomiske parameteren λ og den strukturell kompleksiteten til kunstige organismer, og at λ kan brukes for å forutsi denne strukturelle kompleksiteten. Vi bruker cellulære automata som en model for utvikling av kunstige organismer. Et mål for strukturell kompleksitet blir definert og brukt i eksperimenter for å prøve å finne bevis for å underbygge vår påstand.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Hypothesis

The topic of this thesis is bio-inspired evolutionary computation. We want to investigate the relationship between genomic parameters and the phenotypic properties of artificial organisms. Finding a relationship can help us better understand how organisms develop in developmental systems, both natural and artificial. Having a better understanding of the connection between genomic parameters and phenotypic properties can be a valuable tool when analysing such systems.

Further, gaining a better understanding of how organisms develop enables us to enrich the design step of artificial organisms used in evolutionary and developmental systems. If we can predict properties of artificial organisms using genomic parameters we may guide the evolutionary search by narrowing down the search space. Having an indicator of the phenotypic properties of organisms enables us to do selection without having to do the expensive fitness evaluation involving development.

To be more specific we want to study the λ parameter to see if it can be used to predict structural complexity, a phenotypic property, in artificial organisms. We do this by experimentation using cellular automata as a developmental model.

The formal research hypothesis we want to prove is the following:

> Genomic parameters can be used to predict structural complexity in artificial organisms.

To prove this hypothesis we need to define what we mean by structural complexity. Thus a part of this thesis is devoted to defining and testing various complexity measures for measuring structural complexity. After this is done we define and conduct an experiment to see if we can find any evidence to support our hypothesis.

1.2 Structure

In chapter 2 we cover all the background necessary to understand the work done in this thesis. This will present a foundation upon which we can conduct our research. Chapter 3 will discuss how we can measure structural complexity and defines a complexity measure for use in the rest of this thesis. Chapter 4 sets up the experiment required to test our research hypothesis and presents the results. In chapter 5 we summarise the results and conclude our work.

Chapter 2

Background

2.1 Cellular Computing

In conventional computer science, all theory is based on a single architecture, the von Neumann architecture and its variants, where sequential instructions are executed on processing units. There are however other architectures that are possible, most of them inspired by how nature performs computation.

One such alternative architecture is cellular computing[18, 17]. Cellular computing is a model of how multi-cellular organisms in nature work. It is also an example of a complex system[2] where the components, the cells, are simple and well-defined, but the behaviour of the whole, the system containing all cells, is complex.

In cellular computing each cell is a simple computing unit that only communicates with cells in its neighbourhood. A cellular computing system is distributed; there is no single controlling entity that has a global view of the state of the entire system. This system containing a potentially vast amount of simple computation units operating in parallel with only local communication enables emergent complexity in the system as a whole.

Cellular computing might not be a revolutionary new architecture that can replace today's von Neumann based computers, but research has shown that in some areas, there is potentially advantages by using cellular computing, if we can find an efficient way of designing them, and appropriate building blocks (artificial or biological)[18]. Especially the parallel computational power is intriguing.

One of the challenges of cellular computing is the problem of how we can design or make *programs* or *systems* that perform a specific task we are interested in. If we again look to nature we might discover how many of the complex functions we find have emerged through evolution and development. Trying to simulate evolution and development is one of the currently used methods for designing cellular

computers[22, 19, 14, 20].

2.2 Cellular Automata

The cellular automaton is a simple discrete model of cellular systems. It consists of a set of cells arranged in a grid. Each cell has a state and a rule table. Initially each cell is set to a starting state. To generate the next iteration of the automaton, all cell states are updated simultaneously. Each cell is updated to a new state using its rule table and the current state of all cells in a fixed neighbourhood. In other words, we have a vast amount of simple computation elements operating in parallel with only local interconnections, or a cellular computing system.

Important characteristics of a cellular automaton is the number of cells, the number of cell states and the neighbourhood definition. The neighbourhood is typically defined to contain the cell under inspection itself and a set of its closest neighbouring cells. The neighbourhood and the dimensionality of the automaton are closely tied. Typical cellular automata are either one-dimensional or two-dimensional. This means that the cells are organised in a one-dimensional or two-dimensional grid. For a one-dimensional automaton we often label the neighbourhoods as three-, five- or seven-neighbourhood meaning the two, four or six closest surrounding neighbouring cells. In two dimensions we usually have five- or nine-neighbourhoods. For most purposes the grid will wrap around at the edges.

A rule table contains a transition rule for each possible neighbourhood to a new cell state. Thus to find the next state for a cell we first collect the current state of all cells in the neighbourhood (which might or might not contain the cell itself), and then we can use the rule table to look up the new state.

Cellular automata are either uniform or non-uniform. An uniform cellular automaton uses the same rule table for all cells, while a non-uniform cellular automaton can have different rule tables for each cell. We will only be using uniform automatas in this thesis.

Cellular automata are interesting because they are able to perform computations. They are infact Turing complete[17]. Research has been done to try to understand cellular automata better and to classify them based on their ability to compute[27, 13, 14, 10, 12].

2.2.1 Random Boolean Networks

Cellular automata can be generalised in random boolean networks. This allows us to apply the theory of trajectories and basins of attractors[6]. These are useful tools that can help us analyse the development and behaviour of cellular automata[28, 16, 15, 24].

2.3 Artificial Evolution and Development

As we mentioned in Section 2.1 on page 3, evolution and development are commonly used to design cellular computing systems. When in this context we usually talk about artificial evolution and artificial development.

Artificial evolution involved having a population of individual organisms, represented by their genotypes. Through various mechanisms, such as random mutation and crossover, new organisms are added to the population. Selection is then performed to remove unwanted individuals from the population based on a fitness evaluation, before the process is repeated until a suitable set of individuals are found.

In artificial development, or developmental mappings, we start with an zygote, or single cell organisms, containing its genotype. The genotype is then used to develop the zygote into an adult multi-cellular organism. This basically maps the genotype into a phenotype, containing structure and behaviour. Also, we can simulate adaptation and plasticity by letting the environmental influence the growth of the organism and its adult life[19, 20, 25, 21]

When combined into an evolutionary developmental system (EvoDevo) we typically use the evolutionary step to produce genotypes which are then developed in the fitness function to evaluate the resulting phenotype. This combination allows the genotype to remain relatively small, being only the building instructions for the organism, and thus we can keep the evolutionary search space smaller. Also, the genotype is less complex, while the developmental mapping allows complexity to emerge[23].

When we use uniform cellular automata as a model for cellular computing, the rule table plays the role of the genotype. The rule tables are thus evolved through artificial evolution. In the cellular automata the rule table is then used to iterate the automata until an attractor is reached. Here the iteration of the automata represents development. The initial state of the cellular automata will have just one cell set to a non-quiescent state to represent the zygote. After development the grid will contain many cells in various states representing the developed organism.

Trajectories and attractors are useful tools for analysing the development of artificial organism in our cellular automata[24]. We define the growth of the organism, the developmental path from zygote to adult, as the trajectory, and the adult life as the attractor. Since the system is discrete and deterministic we are guaranteed to eventually find a grid state we have already seen in the automata; this is the attractor. It can either be a point attractor, a single state, or a cycle. The number of states in the attractor is defined to be the attractor length. The trajectory length is defined to be the number of states in the entire trajectory, including the attractor.

2.3.1 Genomic Parameters

We can derive genomic parameters from genotypes. [7, 4, 3, 26] These parameters have various uses. For us, the most interesting usage of genomic parameters is for classification. Given a genotype we can classify it based on one or more genomic parameters.

In the context of uniform cellular automata the genotype is the rule table of the automata. Some examples of typical parameters used with cellular automata are: majority, sensitivity and λ. It is worth noting that these parameters are not limited to cellular automata, but can be used with many other genotypes.

The λ parameter is especially interesting to us, and is the parameter we will be using in the thesis. The λ parameter is particularly simple and well studied in the field. It is defined as the ratio of transitions to any state except the quiescent state and the total number of transitions:

$$\lambda = \frac{T - c}{T}$$

where T is the total number of transitions and c is the number of transitions to the quiescent state, which is typically chosen to be state 0.

This parameter is surprisingly simple, but has proven to be reliable, to a certain degree, for predicting cellular automata rules that result in long trajectories and attractors, often associated with the ability to perform computation. The λ has been shown to be able to classify cellular automata into the four Wolfram classes[27]. Thus λ can be used to find cellular automata with the necessary conditions for complex behaviour[7, 13, 14].

Instead of generating rule tables and calculating the λ parameter of them, we will be reversing this process and instead generate rule tables with a target λ value. This is done using a method called the *random table method* which basically randomly assigns transitions to the quiescent state, or uniformly any other state, based on the target λ value. Using this method we can generate a set of cellular automata with varying λ parameters and then analyse the development and behaviour of these automata.

Langton showed how the λ parameter can be used to classify one-dimensional cellular automata into the four Wolfram classes [7][27]. Tufte and Nichele showed similar results for two-dimensional cellular automata and also used λ to predict the phenotypic properties of trajectory and attractor length[26].

2.4 Complexity

We are interested in the structural complexity of organisms as it relates to various genomic parameters. To investigate this we first have to understand what complex-

ity is, but defining complexity is difficult. Many attempts have been made[11, 1], so we won't cover that here.

But we need to define structural complexity. By structural complexity we mean the complexity of a physical, or phenotypic, property of an (artificial) organism. From complexity theory we find that Kolmogorov complexity[8, 9, 1] can be used to measure physical complexity, such as structural complexity.

However, due to the Incomputability Theorem[8, p. 31] we can not use Kolmogorov complexity directly. Instead, an approximation must be used. Commonly, compression algorithms are used, and a popular choice is the Lempel-Ziv compression algorithm (LZ77)[29]. We will use the `DEFLATE`[5] algorithm, which is a variation of LZ77.

Chapter 3

Structural Complexity

3.1 Introduction

In this chapter we will discuss the structural complexity of an organism and how we can measure it. Several measures will be defined and one will be selected for use in the remainder of this thesis after an experiment designed to evaluate and compare the measures to each other.

We have chosen to use the cellular automata as a developmental model. An organism in this model is then represented by the state of the cellular automaton.

First we will discuss structure complexity as a property of a state. Next, we will describe in detail how we can calculate the structural complexity of a given state. We will end up with four different measures. We will then test all four measures to see if they do their job, and if there is any difference between the measures, before we settle on a measure to be used in the remainder of this thesis.

When we later on will use the defined measure to analyse structural complexity of organisms we will actually be looking at the average complexity over a time period, usually during the entire growth of the organism, the trajectory, or during it adult life cycle, the attractor.

3.1.1 Definition

Structural complexity is a measure of the complexity of structures making up an organism. In our model an organism is represented by a cellular automata state. Compared to genomic parameters, the structural complexity is a property of the phenotype, as opposed to genomic parameters which are properties of the genotype.

The structural complexity is interesting because it gives us some indication of the possible capabilities of an organism. The structural complexity can be used as one

of several parameters when evaluating and analysing artificial organisms, or as a target parameter for prediction in artificial evolution.

3.2 Measuring Structural Complexity

As we have already discussed in chapter 2 on page 3 there are many valid choices of complexity measures to chose from. We have chosen to use the `DEFLATE` compression algorithm[5], approximating Kolmogorov complexity[8, 11, 27], as our measure of structural complexity. This choice is based on the fact that compression algorithms tend to compress repeated patterns and structures well, thus being able to detect structural features in our states. It is also a relatively inexpensive operation and easy to use. Also, so long as we define the process precisely, it is independent of the dimensionality of the state.

Due to the nature of state grids and the way we apply the `DEFLATE` algorithm, the position and orientation of structures within the state grid makes a difference. We will discuss how we can take this into account when calculating a complexity measure for a state.

3.2.1 Calculating the Structural Complexity of a State

We will now illustrate how to calculate the complexity of a state s. We will use a two-dimensional grid of size 3×3 as an example, but the process is similar for all dimensions.

The first task is to convert the grid representation s of the state into a string representation r. For a two-dimensional grid we do this by stringing the rows of the grid together to make the state string:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 1 & 0 & 0 \end{pmatrix} \rightarrow \text{“010112100”}$$

This state string can now be compressed using the `DEFLATE` algorithm to produce the compressed state string t:

$$t = \texttt{DEFLATE}(r)$$

The only thing we really need is the length, q, of the compressed string, not the compressed string itself:

$$q = \texttt{LEN}(t)$$

q, the length of the compressed string, can be used to compare the approximate complexity of states. The bigger q is, the less compressable the state is and thus the more complex it is. However, we can't use this value for comparing complexities from different dimensionality and grid sizes. Thus we seek to normalise the value to the range $[0, 1]$.

For a given string length, the `DEFLATE` algorithm guarantees that the compressed string will never be longer than the original string. If we can find a lower and upper bound for the compressed string length, we can use those values to scale q. We can find these bounds by constructing a most and least complex state string for given grid sizes:

$$r_{min} = \text{“000000000”}$$
$$r_{max} = \text{“012345678”}$$

r_{min} is the simplest state possible and will, when compressed, yield the lowest compressed size q_{min} possible for states of the given dimension and grid size. Likewise, r_{max}, which has no two identical symbols[1], yields the highest possible compressed size q_{max} for states of the given dimension and grid size:

$$q_{min} = \texttt{LEN}(\texttt{DEFLATE}(r_{min}))$$
$$q_{max} = \texttt{LEN}(\texttt{DEFLATE}(r_{max}))$$

Now we can scale q to get the final normalised complexity measure c of the state s:

$$c = \frac{q - q_{min}}{q_{max} - q_{min}}$$

c is a normalised approximation to the structural complexity of a state. We could now use the described procedure to measure the structural complexity of states, but there are some additional issues we must discuss first. And we would also like to test the measure before continuing on with the main goals of this thesis.

3.2.2 Position and Orientation of Structures in a State

When measuring the structural complexity of a state we have to take into consideration the position and orientation of the structure within the grid.

Using the process described in the previous section, we run into some problems when structures are transformed. States that contain essentially the same structure, only in a different orientation or position, can get different complexity values (Figure 3.1). We want states that contain the same structure to have equal complexity, for all possible positions and orientations. How can we achieve this?

[1] we can use any symbol in this step, even symbols that do not represent valid cell states

$$\underset{s_1}{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 1 & 2 \end{pmatrix}} \quad \underset{T}{\rightarrow} \quad \underset{s_2}{\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix}}$$

Figure 3.1: *Transforming state s_1 into s_2. T rotates the state $90°$. The complexity of s_1 and s_2 is $c_1 = 0.5$ and $c_2 = 1.0$. Even though the structures in both states are equal, the measured complexity is not.*

Table 3.1: *Complexity measures*

1	simple	S
2	rotation	R
3	translation	T
4	rotation+translation	RT

A naive solution is to simply transform a structure in a given state in all possible positions and orientations; for each such state calculate the complexity and find the average complexity over all possible transformations of the structure. This makes the process of calculating the complexity a more expensive operation, but guarantees that equal structures get the same complexity for any orientation or position within a grid. We will take an experimental approach to finding a suitable complexity measure that can deal with the problem of structural transformations.

We define four measures. The first measure is simply the procedure described previously, with no modification. The second measure rotates the grid in all $90°$ rotations. The sum of the complexities is then divided by the number of states to get the complexity of the state. Similarly, the third measure finds all translations of the state and finds the average complexity. The fourth and last measure is a combination of the second and third, where all rotations and all translations are used.

3.3 Experimental Setup

We now have four complexity measures. To see how they perform in relation to each other, we will conduct an experiment. Using various grid sizes and dimensions, we will plot the structural complexity, using each of the four measures, against the genomic parameter λ averaged over several iterations.

For all the experiments we use 3 states per cell, a step size of 0.01 and 100 iterations per λ. Table 3.2 summarises the experiments.

Table 3.2: *Experiments for testing the complexity measures.*

dim	size	cells	neighbourhood
1D	9	9	3
1D	9	9	5
1D	16	16	5
1D	8	8	7
2D	3×3	9	5
2D	4×4	16	5
3D	$2 \times 2 \times 2$	8	7

3.4 Results and Analysis

The results are presented in Figures A.1 to A.7.

The results indicate that the four complexity measures are more or less the equivalent in a metric setting where we only care about relative complexity. We see that for any measure, the complexity is not constant, and has a similar shape for all experiments.

3.5 Conclusion

We conclude that using the simple measure is sufficient for our needs and it will thus be the sole measure used within the remainder of this thesis.

Chapter 4

λ vs Complexity

4.1 Introduction

We are now ready to conduct the main experiment. As stated before we want to see if there is any connection between the genomic λ parameter and structural complexity. We will do this by generating rule tables in a set of different configurations and then developing them to produce organisms whose structural complexity we measure. The trajectory and attractor lengths are measured and compared and compared against the final averaged measure of structural complexity for both the developmental trajectory and attractor. Thus we can see if we can make any predictions on structural complexity given the λ parameter.

First we will describe the experimental setup, before moving on to presenting the results and an analysis. At the end we conclude.

4.2 Experimental Setup

In order to see if there are any predictable connection between λ and structural complexity we are going to conduct an experiment with five different configurations. We will use a uniform cellular automata with three states per cell. The five configurations are listed in table 4.1. We will generate 1000 different rule tables for each λ in the range $[0.0, 1.0]$ with a step size of 0.01. Then for each rule table, we will develop an initial grid containing states all equal to the quiescent state except for the middle cell which is set to state 1. The automata are simulated until the attractor is reached. At this point we record the number of steps in both the trajectory and the attractor, and measure the average structural complexity over the trajectory and attractor using the measure defined in section 3.2 on page 10. These values are then plotted against each other.

Table 4.1: *λ vs complexity experiments.*

dim	size	cells	neighbourhood
1D	25	25	3
1D	27	27	3
1D	25	25	5
1D	27	27	7
2D	5×5	25	5
3D	$3 \times 3 \times 3$	27	7

By having five different configurations we make it possible to compare grids with the same number of cells, but at different dimensionality and neighbourhoods.

4.3 Results and Analysis

The results from the experiment are available in Figures B.1 to B.6.

There are many interesting points to note in this data. First of all, if we disregard the 3-neighbourhood 1D configuration for a moment, all the other experiments show a clear correlation between attractor and trajectory length, and the structural complexity. Also, for all cases there is a clear trend for the structural complexity to be higher when λ is roughly around 0.7. We can quickly draw the conclusion that there is some connection between λ and structural complexity.

We have two cases where the number of cells are equal, but the dimensionality and neighbourhood is different. If we look at the 27 cell experiments, one is one-dimensional and the other is three-dimensional. If we view both as a single string of cells, the only different is how we sample the neighbourhood. The data shows that, although the shapes are similar, the trajectory and attractor lengths are on average much longer. The same is evident for the 25 cell case, to a lesser degree. This shows that how we sample the neighbourhood in a grid (regardless of dimensionality) plays a role, and that a lower dimensional cellular automaton will have longer trajectories and attractors than a cellular automaton of a higher dimension with the same number of cells.

We see that the trajectory and attractor length match well with the results in [26], as expected, except for something weird happening for the 1D 3-neighbourhood configuration. The trajectory and attractor length do not follow the expected curve for similar experiments. We see for the similar 1D configuration with 5-neighbourhood instead of 3, the trajectory and attractor lengths are more as expected. What could be wrong? Some clue can be found in [7] where Langton finds that λ differentiates poorly for configurations with small number of cell states and a small neighbourhood. According to Langton, we should employ at least four cell states and a neighbourhood of at least five cells for λ to work as expected.

Even though trajectory and attractor lengths seem off, it is interesting to see that structural complexity follows the same shape as in other experiments. Thus, even with a small neighbourhood, when λ can no longer be used to predict trajectory and attractor length, λ can still be used to predict structural complexity.

4.4 Conclusion

In terms of our initial research question, the evidence gathered in this section seem to strengthen the hypothesis that genomic parameters can be used to predict structural complexity.

Chapter 5

Summary and Conclusion

To sum up, we have seen that a simple compression of the state is a sufficient measure of structural complexity when we want to compare the complexity of different states. Thus we don't have to use expensive operations that take into account the position and orientation of structures.

We have also seen that λ can be used to predict structural complexity. This means we can use λ to evaluate organisms in terms of potential structural complexity prior to developing them in an artificial evolution setting.

As a side note we also found that λ does not predict trajectory and attractor lengths for 1D 3-neighbourhoods. However, even when λ cannot be used for trajectory and attractors, it can still be used to predict structural complexity.

Appendices

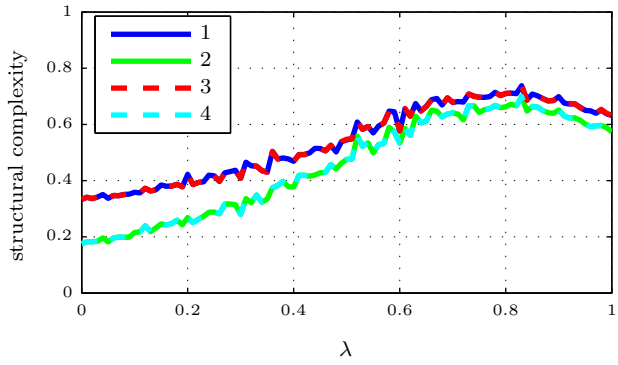
Appendix A

Plots: Complexity Measures

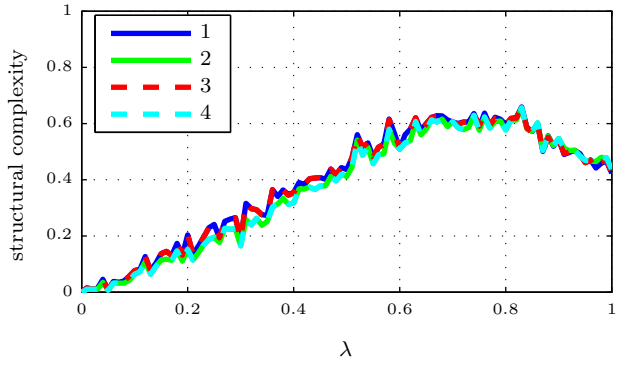
For trajectory and attractor, the complexity is calculated for each state and then summed and divided by the number of states to find the average structural complexity along the trajectory and attractor.

The four complexity measures used here are: simple; translation; rotation+mirror; rotation+mirror and translation.

3 states, step size 0.01 and 100 iterations per λ.

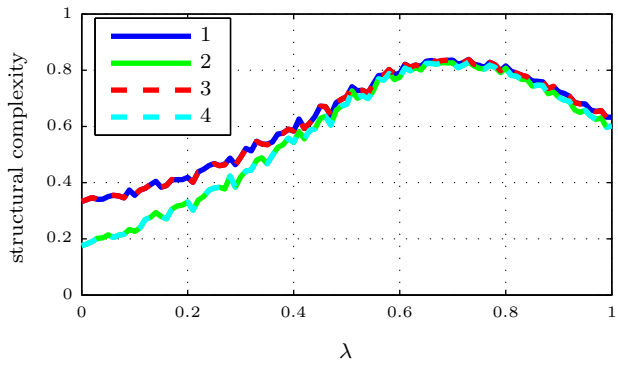


(a) *Trajectory.*

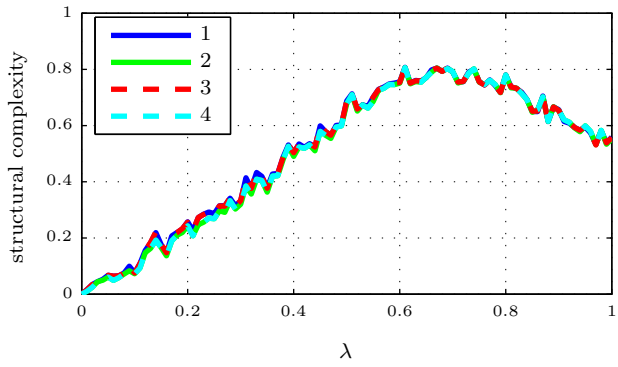


(b) *Attractor.*

Figure A.1: *1D 9 cells, 3-neighbourhood*

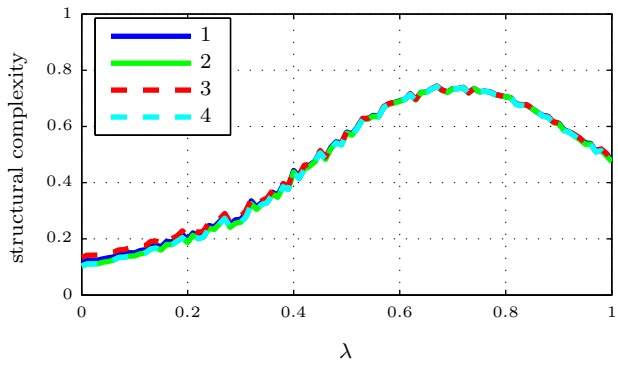


(a) *Trajectory.*

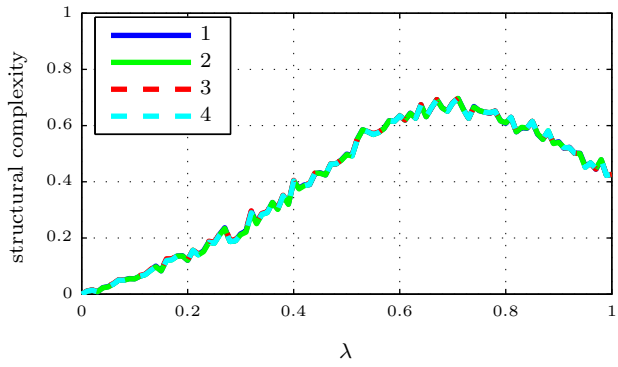


(b) *Attractor.*

Figure A.2: *1D 9 cells, 5-neighbourhood*

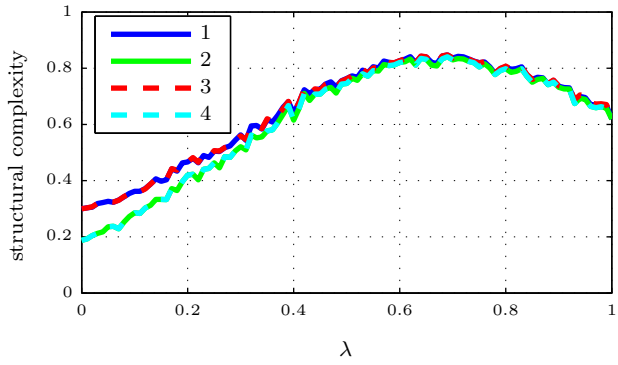


(a) *Trajectory.*

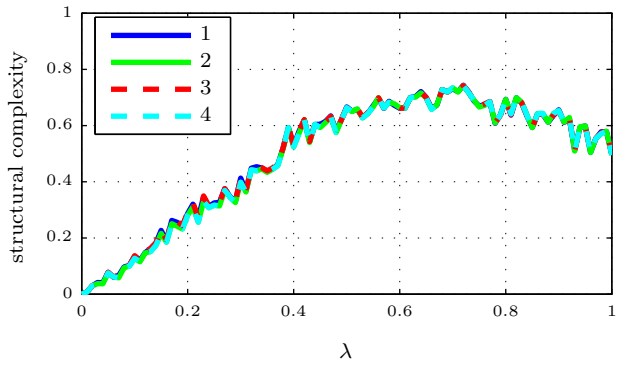


(b) *Attractor.*

Figure A.3: *1D 16 cells, 5-neighbourhood*

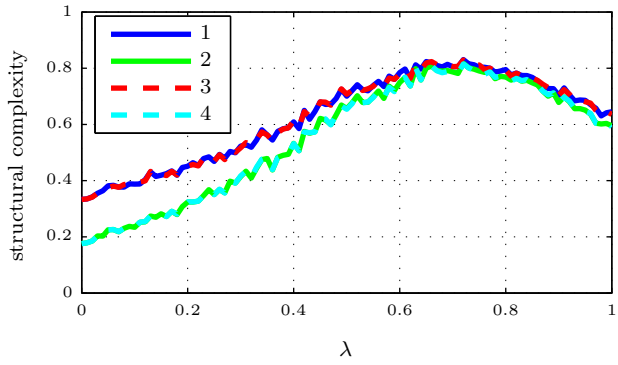


(a) *Trajectory.*

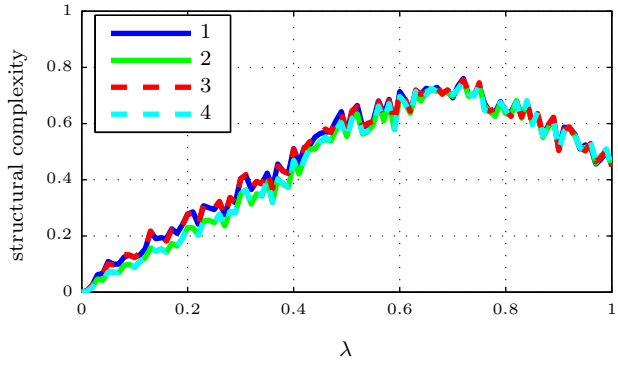


(b) *Attractor.*

Figure A.4: *1D 8 cells, 7-neighbourhood*

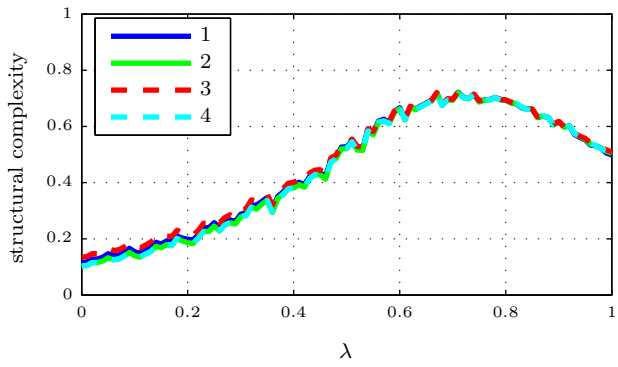


(a) *Trajectory.*

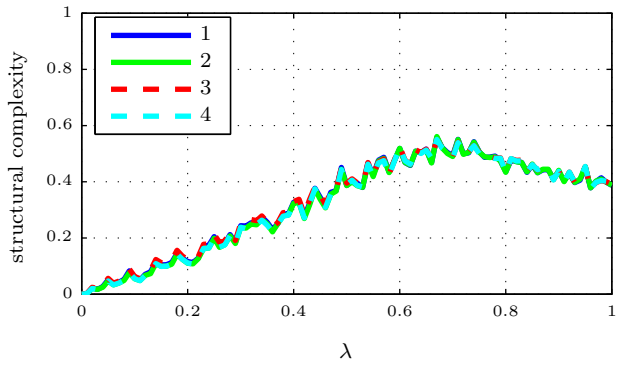


(b) *Attractor.*

Figure A.5: *2D 9 cells (3×3), 5-neighbourhood*

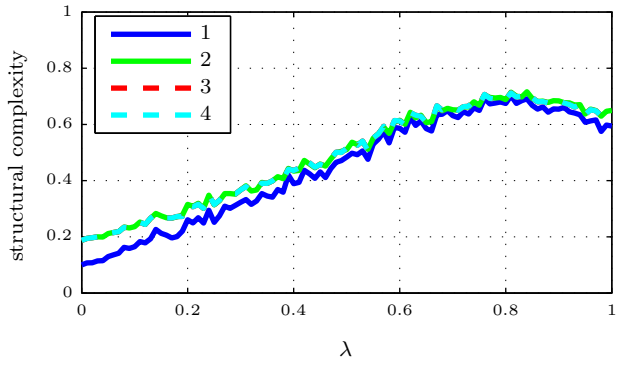


(a) *Trajectory.*

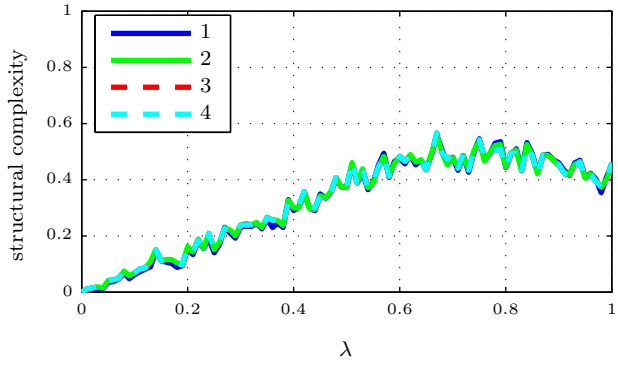


(b) *Attractor.*

Figure A.6: *2D 16 cells* (4×4)*, 5-neighbourhood*



(a) *Trajectory.*



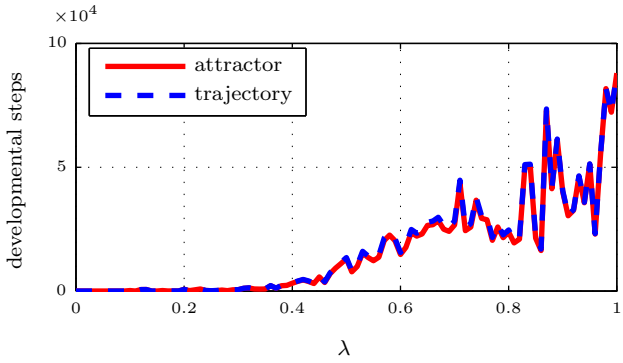
(b) *Attractor.*

Figure A.7: *3D 8 cells ($2 \times 2 \times 2$), 7-neighbourhood*

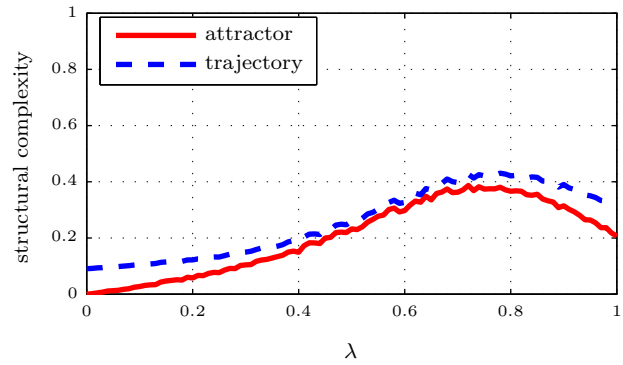
Appendix B

Plots: λ vs Complexity

3 states, step size 0.01 and 1000 iterations per λ.

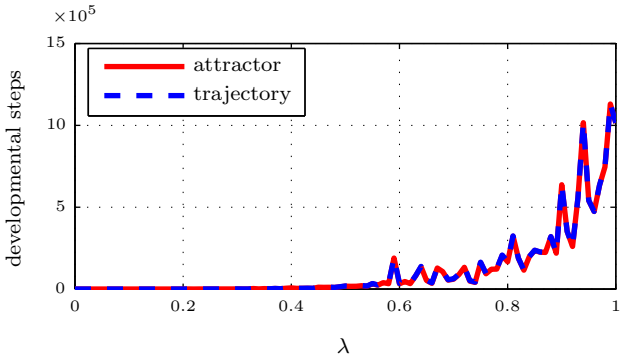


(a) *Average trajectory and attractor length.*

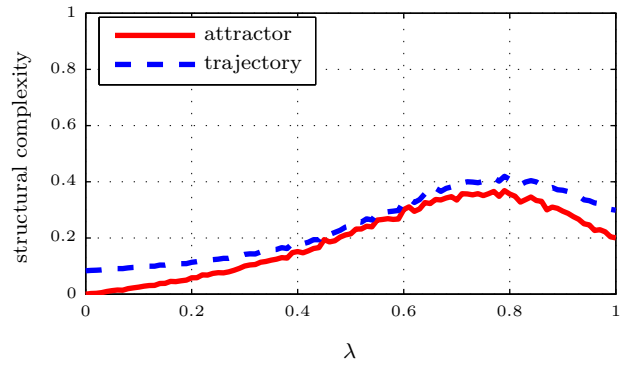


(b) *Structural complexity of trajectory and attractor.*

Figure B.1: *1D 25 cells, 3-neighbourhood*

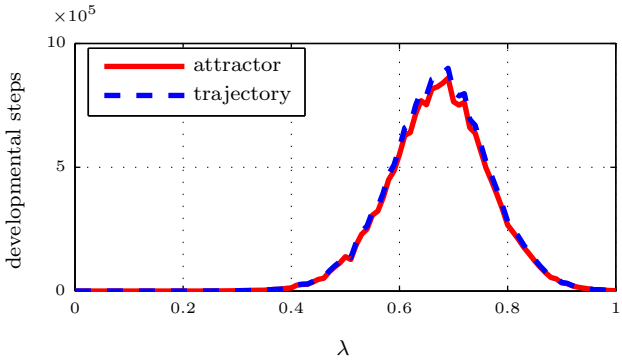


(a) *Average trajectory and attractor length.*

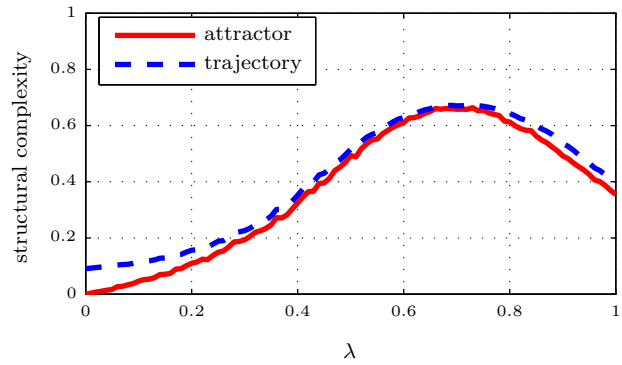


(b) *Structural complexity of trajectory and attractor.*

Figure B.2: *1D 27 cells, 3-neighbourhood*

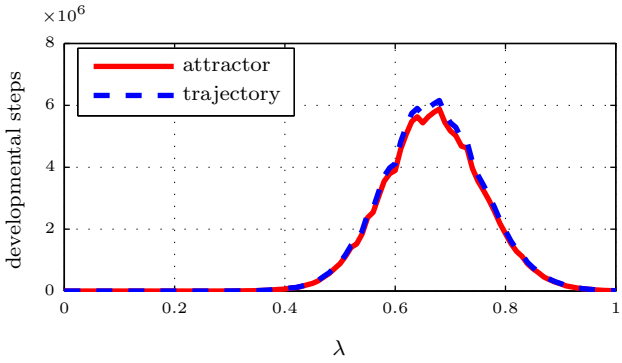


(a) *Average trajectory and attractor length.*

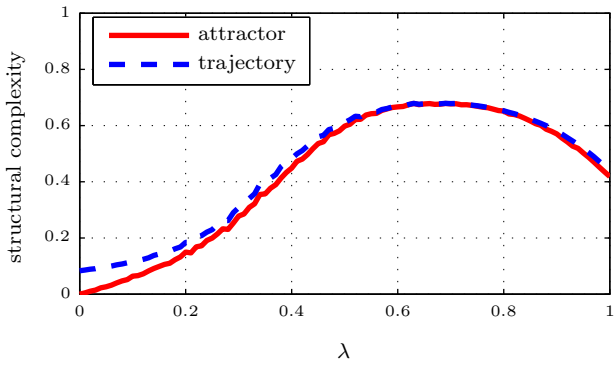


(b) *Structural complexity of trajectory and attractor.*

Figure B.3: *1D 25 cells, 5-neighbourhood*

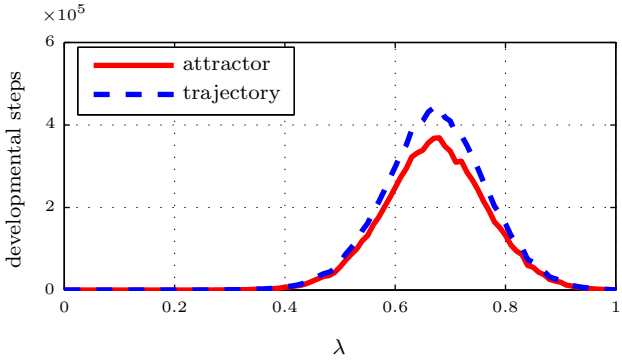


(a) *Average trajectory and attractor length.*

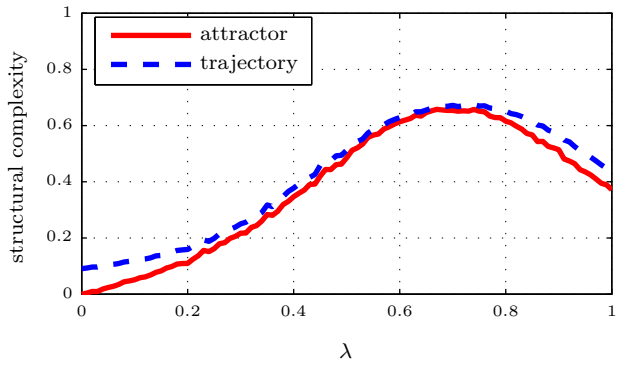


(b) *Structural complexity of trajectory and attractor.*

Figure B.4: *1D 27 cells, 7-neighbourhood*

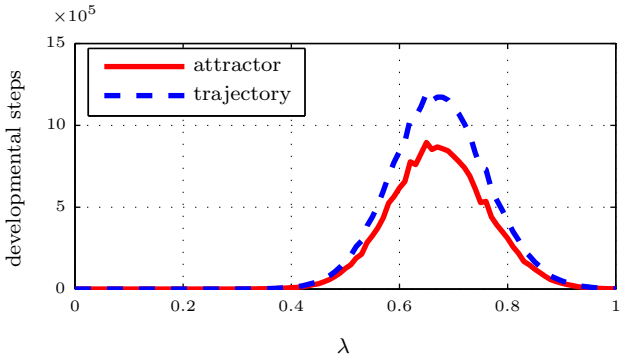


(a) *Average trajectory and attractor length.*

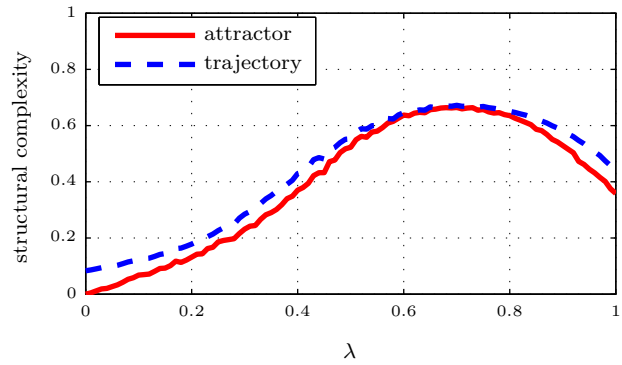


(b) *Structural complexity of trajectory and attractor.*

Figure B.5: *2D 25 cells* (5×5)*, 5-neighbourhood*



(a) *Average trajectory and attractor length.*



(b) *Structural complexity of trajectory and attractor.*

Figure B.6: *3D 27 cells* ($3 \times 3 \times 3$)*, 7-neighbourhood*

Bibliography

[1] Christoph Adami. What is Complexity? *BioEssays*, 24(12):1085–1094, December 2002.

[2] Yaneer Bar-Yam. Overview: The Dynamics of Complex Systems — Examples, Questions, Methods and Concepts. In *Dynamics of Complex Systems*, chapter 0, pages 1–15. Addison-Wesley, 1 edition, 1997.

[3] Gina M. B. de Oliveira, Pedro P. B. de Oliveira, and Nizam Omar. Guidelines for Dynamics-Based parameterization of One-Dimensional Cellular Automata Rule Spaces. *Complexity*, 6(2):63–72, 2000.

[4] Gina M. B. de Oliveira, Pedro P. B. de Oliveira, and Nizam Omar. Definition and Application of a Five-Parameter Characterization of One-Dimensional Cellular Automata Rule Space. *Artificial Life*, 7(3):277–301, January 2001.

[5] L. Peter Deutsch. DEFLATE Compressed Data Format Specification version 1.3, 1996.

[6] Carlos Gershenson. Introduction to Random Boolean Networks. In *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, pages 160–173, 2004.

[7] Christopher Langton. Computation at the Edge of Chaos: Phase Transitions and Emergent Computation. *Physica D: Nonlinear Phenomena*, 42(1-3):12–37, June 1990.

[8] Per Kristian Lehre. *Complexity and Geometry in Artificial Development*. Doctoral thesis, Norwegian University of Science and Technology (NTNU), 2006.

[9] Ming Li and P Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*, volume 34 of *Texts in Computer Science*. Springer, 1997.

[10] Melanie Mitchell. Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA '96)*, Moscow, Russia, 1996. Russian Academy of Sciences.

[11] Melanie Mitchell. *Complexity: A Guided Tour*. Oxford University Press, 2009.

[12] Melanie Mitchell, James P. Crutchfield, and Rajarshi Das. Evolving Cellular Automata to Perform Computations. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter G1.15. Oxford University Press, Oxford, 1997.

[13] Melanie Mitchell, James P. Crutchfield, and Peter T. Hraber. Dynamics, Computation, and the "Edge of Chaos": A Re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models, and Reality*, volume 19, pages 497–513. Addison-Wesley, 1994.

[14] Melanie Mitchell, James P. Crutchfield, and Peter T. Hraber. Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments. *Physica D: Nonlinear Phenomena*, 75(1-3):361–391, 1994.

[15] Stefano Nichele. *Trajectories and Attractor Basins as a Behavioral Description and Evaluation Criteria for Artificial EvoDevo Systems.* Master thesis, Norwegian University of Science and Technology, 2009.

[16] Stefano Nichele and Gunnar Tufte. Trajectories and Attractors as Specification for the Evolution of Behaviour in Cellular Automata. In *IEEE Congress on Evolutionary Computation*, pages 4441–4448. IEEE conference proceedings, 2010.

[17] Moshe Sipper. *Evolution of Parallel Cellular Machines.* Springer-Verlag, Heidelberg, 1997.

[18] Moshe Sipper. The Emergence of Cellular Computing. *IEEE Computer*, 32(7):18–26, 1999.

[19] Gunnar Tufte. Cellular Development: A Search for Functionality. In *IEEE Congress on Evolutionary Computation*, pages 2669–2676. IEEE conference proceedings, 2006.

[20] Gunnar Tufte. Evolution, Development and Environment Toward Adaptation through Phenotypic Plasticity and Exploitation of External Information. In *Artificial life XI The Eleventh International Conference on the Simulation and Synthesis of Living Systems*, volume 11, pages 624–631. MIT Press, 2008.

[21] Gunnar Tufte. Phenotypic, Developmental and Computational Resources: Scaling in Artificial Development. In *Genetic and Evolutionary Computation Conference (GECCO-2008)*, pages 859–866. Association for Computing Machinery (ACM), 2008.

[22] Gunnar Tufte. Evolvable Machines: An EvoDevo Approach. In *Proceedings of the First Norwegian Artificial Intelligence Symposium*, pages 17–28. Tapir Akademisk Forlag, 2009.

[23] Gunnar Tufte. From Evo to EvoDevo: Mapping and Adaptation in Artificial Development. In Wellington Pincheiro dos Santos, editor, *Evolutionary Computation*, chapter 12, pages 219–238. InTech, 2009.

[24] Gunnar Tufte. The Discrete Dynamics of Developmental Systems. In *IEEE Congress on Evolutionary Computation*, pages 2209–2216. IEEE conference proceedings, 2009.

[25] Gunnar Tufte and Pauline Haddow. Achieving Environmental Tolerance through the Initiation and Exploitation of External Information. In *IEEE Congress on Evolutionary Computation*, pages 2485–2492. IEEE conference proceedings, 2007.

[26] Gunnar Tufte and Stefano Nichele. On the Correlations Between Developmental Diversity and Genomic Composition. In *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 1507–1514. Association for Computing Machinery (ACM), 2011.

[27] Stephen Wolfram. Universality and Complexity in Cellular Automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[28] Andrew Wuensche and Mike Lesser. *The Global Dynamics of Cellular Automata*. Addison-Wesley Publishing Company, 1992.

[29] Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.