A Good Number of Forms Fairly Beautiful
An Exploration of Biologically-Inspired Automated Design

Taras Kowaliw

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
at Concordia University
Montréal, Québec, Canada

September 2007

Your file Votre référence
ISBN: 978-0-494-31134-9
Our file Notre référence
ISBN: 978-0-494-31134-9

ABSTRACT

"A Good Number of Forms Fairly Beautiful: An Exploration of Biologically-Inspired Automated Design"
Taras Kowaliw, Doctoral Candidate
Concordia University, 2007

Artificial Embryogeny (AE) can be described as the use of a dynamical system as a mid-step in a design process; Through emulating Biological Embryogenesis, we hope to reach levels of complexity and robustness currently impossible. AE is a new field, and suffers from a lack of standards and meaningful means of evaluation. In this document, we review existing work, discussing motivations and merits of existing approaches. Throughout, we argue that a viewpoint which does not regard environment as a primary source of information risks taking a naïve view of evolution. We argue that "complexity" is vaguely and inconsistently defined, and propose several novel measures; Perhaps the simplest model of AE, the Terminating Cellular Automaton, is introduced, and used to compute and contrast our measures. Next, the Deva family of AE algorithms is introduced, a modular Cellular Automaton-like group. A domain of application from Civil Engineering is chosen as an interpretation of the grown organisms. It is initially shown that it is possible to use a Deva algorithm to evolve Plane Trusses successfully, this interpretation providing a discipline-independent measure of success. A series of empirical experiments is undertaken, showing the relative efficacy and effects of several model-level strategies in the context of the evolution of structural design. Finally, we explore the role of environment as a constraint on development of structural form. We demonstrate a strong resistance to environmental change by successfully re-growing the organisms in new environments, showing that some Deva organisms are adding information from the environment to their overall morphology; This provides an artificial analogue to the re-use of genes which characterizes biological development.

RÉSUMÉ

"A Good Number of Forms Fairly Beautiful: Une exploration de la conception automatisée inspirée de la biologie"
Taras Kowaliw, candidat au doctorat
Université Concordia, 2007

L'embryogenèse artificielle (AE) concerne l'emploi d'un système dynamique en tant qu'un pas au mi-chemin dans un processus de conception; en imitant l'embryogenèse biologique, nous espérons atteindre des niveaux de complexité et de solidité actuellement impossible. L'AE est un nouveau domaine où il y a un véritable manque de normes et de moyens d'évaluation significatifs. Dans ce document, nous faisons le bilan de l'oeuvre actuel, avec une discussion des motivations et des avantages des approches actuelles. Nous maintenons qu'un point de vue qui ne considère pas l'environnement comme une source principale d'information risque de prendre une vue naïve de l'évolution. Nous soutenons que la *complexité* est définie de manière vague et incohérente, et nous proposons plusieurs nouvelles mesures. Le *Terminating Cellular Automaton*, peut-être le modèle le plus simple de l'AE, est introduit et utilisé afin de calculer et de mettre en contraste nos mesures. Ensuite, la famille Deva d'algorithmes de l'AE est présentée, un groupe modulaire semblable à l'automate cellulaire. Un domaine d'application du génie civil est choisit pour l'interpretation des organismes adultes. D'abord, nous montrons qu'il est possible d'utiliser un algorithme Deva pour l'évolution réussie des treillis, une interpretation qui fournit une mesure de succès indépendante de la matière. Une série d'expériences empiriques est réalisée, montrant l'efficacité et les effets relatifs de plusieurs stratégies du niveau du modle au contexte de l'évolution de la conception structurale. Enfin, nous examinons le rôle de l'environnement en tant que contrainte du développement de la forme structurale. Avec le succès des cultures subséquentes dans de nouveaux environnements, nous démontrons une forte résistance au changement environnemental; cela montre que certains organismes Deva ajoutent de l'information cueillie de l'environnement à leur morphologie globale. D'où l'analogue artificiel de la réutilisation des gènes qui charactérise le développement biologique.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help of many people: First and foremost, my thanks to my family and to Kara-Anne Fraser, whose constant presence, advice and support have helped make any of my achievements possible; My thanks to the Department of Computer Science at Concordia University, and to the Department of Mathematics at the University of Toronto, for their excellent education and resources; Finally, my gratitude to Jutta Treviranus and the Adaptive Technology Resource Centre, for their aid and understanding throughout my studies.

Further, I am indebted to Ramin Sedaghati, for his help in understanding and confirming correctness of the Civil Engineering portions of this work; to Nicolas Brodu, for valuable discussions and advice; and to Motria Spolska, Myron Kowaliw, Larissa Kowaliw, and Kara-Anne Fraser for their help on the draft of this document.

My thanks to Nawwaf Kharma, for continued support and guidance. Much of my understanding of the practice of research comes from his advice, and much of my confidence comes from his continued support.

Finally, my gratitude to Peter Grogono; This certainly for his support and aid, but primarily for years of interesting discussions of shared interests; My thinking has been deeply influenced by his, and much of the material in this thesis is derived ultimately from his ideas.

Photo credits:
"Slogor", by Stephen Notley (http://angryflower.com), page xii
"The Hearn 2" by Charles Bodi (http://ridemypony.com), page 115

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> From the war of nature, from famine and death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone circling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being evolved.
>
> *Charles Darwin, The Origin of Species*

We have certainly never stopped marvelling at the beauty and diversity of Darwin's "endless forms most beautiful", the multicellular organisms; Indeed, after 150 years of constant scientific progress and understanding, it seems at times that we are not much closer to understanding the origins of life than was Darwin himself. A much discussed and pursued area of biology today, colloquially termed "EvoDevo" for Evolutionary Development, seeks to bridge the gap between the far-better understood fields of Genetics and Ecology; The object of study is Embryogenesis, the process by which a single egg becomes a multicellular organism. This field is considered imperative to the understanding of evolution and the diversity of multicellular life, and, as such, to understanding the ultimate causes of a good portion of all of biology.

Computer Science is a field of study often concerned with the construction of solutions to mathematical questions. The primary thrust of Computer Science is on the design of algorithms or on programming, the means of constructing functions which serve our needs. Indeed, a communication of the ACM once declared that the fundamental underlying question of Computer Science was "What can be (efficiently) automated?"[1] There is, however, a second perspective — this is the viewpoint of mathematicians such as Kolmogorov or Chaitin — which sees computation as compression, and the computable functions as the set of all human-understandable patterns, or human-understandable relations between sets of data. This suggests a different role for Computer Scientists, other than that of programming; This suggests the role of a Computer Scientist as a discoverer of patterns that exist in the larger space of all functions. Or, given some existing and poorly understood source of pattern in nature, a discoverer of a means of understanding for that pattern.

[1] Computing as a discipline, Communications of the ACM, January 1989

The information contained in a developed foetus or organism results from several initial sources, which we will group in the following three categories: the genome of the fertilized egg or seed, the environment in which the organism grows, and the dynamics of cell reproduction in this environment. It will be argued that the relevance of the latter two of these categories, the dynamics and the environment, are often underestimated; That is, both contribute substantially to pattern formation and the generation of information in the organism. Additionally, it will be noted that the dynamics of development is generally poorly understood mathematically. We will review some existing measures of complexity, and propose some novel measures, which may help in making these notions explicit.

There has been much interesting work in Artificial Embryogeny (AE) in modelling the developmental process; This both in terms of understanding biology, and in terms of exploiting the process for the design of useful structure. These systems have shown much success on both counts, both in simulating the function of biological organisms, and in creating real-world structures or software, useful by the standards of engineers. We will next define a domain of application from Structural Engineering; The importance of using a model which comes supplied with an externally-defined measure of success should be clear in Artificial Life, and we will use that of Plane Trusses to supply a notion of meaning to our experiments.

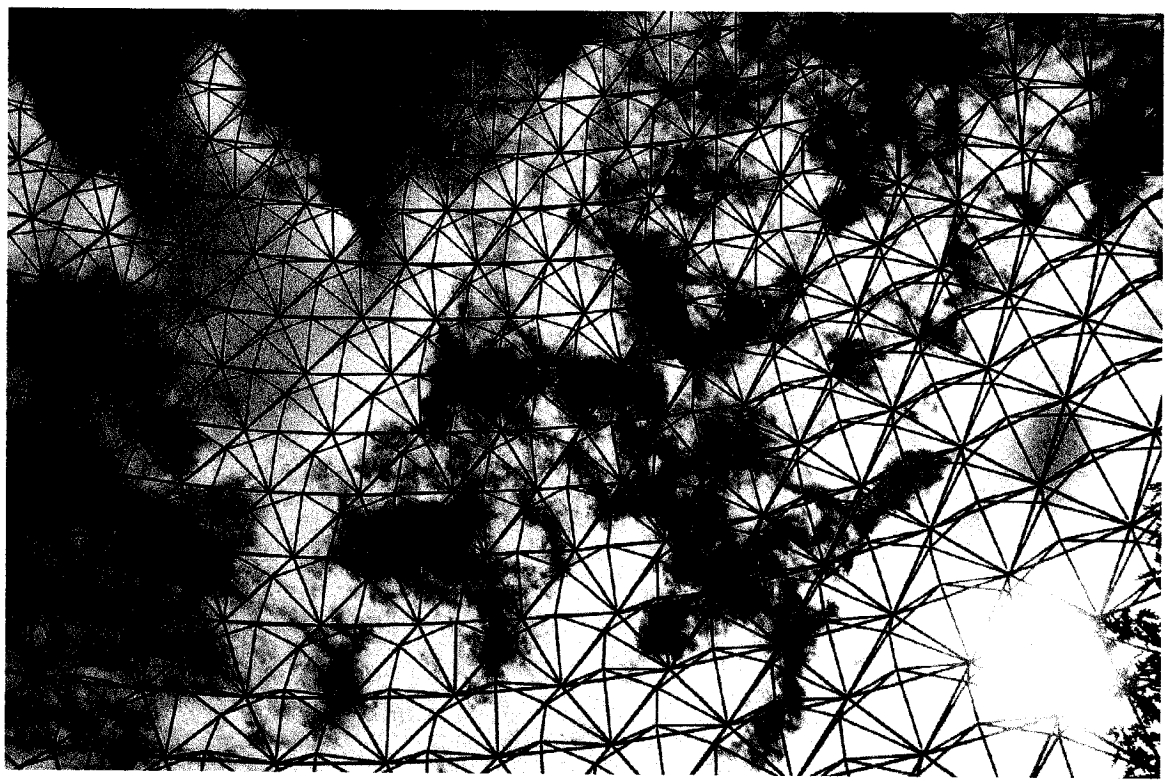
AE typically involves the use of a non-linear dynamical system to map between genotype and phenotype; The unpredictability of dynamical systems is well known, as is the fact that in any non-linear system, small perturbations may lead to large changes in aggregate behaviour over time. At the same time, Machine Learning, and Evolutionary Computation in particular, serve to move a population of agents towards some optimum or attractor. The interplay between Evolutionary Computation and Dynamical Systems, a key feature of current work in AE, utilizes two seemingly opposing processes. As such, it is not at all obvious what effect a model-level perturbation will have on an AE model.

While minimal models (two- or three-colour image generation, for instance) may be sufficient for generating proof-of-concepts or demonstrating general properties, we suspect that they do not suffice for the selection of model-level strategies; For the practical application of AE, or the simulation of biological phenomena, the phenotypic search space is likely to be largely pre-specified by the chosen domain of application, with little variance in the possible components and dimensionality of representation. Yet, many existing AE representations grow exponentially in these two variables, making them intractable choices.

We will utilize our models and domain of application to evaluate model-level decisions in Artificial Embryogeny empirically. In so doing, we will demonstrate that these decisions, including seemingly small and insignificant choices, indeed can sabotage the process of evolution in terms of optimizing our fitness function. Factors which contribute significantly will be shown to include the dimensionality of phenotypic space, suggesting that hypothesis which hold true for AE models in some given domain of application need not hold true at other dimensionalities.

Finally, we will explore the role that environment can play in Artificial Embryogeny. Rather than control evolution simply through a fitness function, we will impose additional constraints via choice of environment. In so doing, we will show that our models are capable of adapting to novel environments, despite having been evolved elsewhere. Indeed, we will discover several simple agents capable of growing stable trusses in nearly all explored environments; This capability, that

of a simple genome to exploit environmental information provided at the time of growth, is an embodiment of the principle of biological channelling which we will highlight in our discussions of complexity measures.



Chapter 2

Review

2.1 Measures of Complexity

There have been many attempts to define measures of complexity; Here we review several, taken from both Mathematics and Biology, and discuss their relative strengths and weaknesses. This is by no means an exhaustive list. Instead, we choose measures that are (a) readily defined in the space in which AE models operate, and (b) directly relevant to embryogenic growth.

Structural Complexity in organisms is an attempt to define complexity in a fashion that corresponds with our naïve conception, using the gross number of cells, cell types, organs, etc. Bell and Moorers, for instance, use the number of types of cell specialization, and show a correlation with the estimated number of cells in an organism overall. They interpret the results to indicate a greater capacity for specialization in a cooperative division of labour (as opposed to competitive division of labour, the number of cell types in a population). Interestingly, they find that major groupings of organisms differ in complexity by size: animals have more cell types per total cell number than plants, for instance [BM97].

A similar notion is that of McShae's *Functional Complexity*, which aims to measure the complexity of an organism by the number of functions it performs. Since an enumeration of functions is impossible directly, there have been attempts to use "parts as proxy"; That is, if one makes the assumption that parts in an organism exist because they play a functional role, then the number of parts forms a rough estimate for the number of functions which an organism carries out. Parts is a vague term, but could include the number of cells, number of organs, morphological categories, etc. [McS00].

Shannon Entropy (sometimes Shannon Information) was developed by Shannon as a means of estimating the information content of a transmitted symbol [Sha48]; It is quite similar to notions of entropy from thermodynamics, hence the title. Shannon entropy has since become a common measure of "complexity", often used in cryptography and communication theory, and also at times as a means of measuring self-organization. Shannon entropy is maximized for random sources and normal numbers[1], and low Shannon entropy is an indication of easily compressible data.

Let X be a random variable taking values on alphabet Σ, and $p(x) = Pr(X = x)$. The

[1] A normal number is one in which in the expansion every digit occurs with equal frequency.

Shannon entropy function is defined as

$$H(X) = E[I(X)] = -\sum_{x \in X} p(x) \log p(x) \tag{2.1}$$

where $I(X)$ is the self-information, or the information contributed by a single symbol. Making the assumption that $0 \log 0 = 0$, H assumes values on the range $[0, \log |\Sigma|]$.

Kolmorogov Complexity, also known as the *Algorithmic Information Content*, was developed independently by Kolmogorov, Chaitin and Solomonoff [GV06]. The measure is intended to capture the notion of incompressibility of data, where we consider every knowable means of compression; This is done by considering a Universal Turing Machine, U, which, by the Church-Turing thesis, implements every possible algorithmic technique. The particular choice of U is unimportant (from the perspective of mathematics, at least) since each can implement every other in constant time. This leads to our definition:

$$K_U(x) = \min_{|p|} \{p \mid p \vdash_U x\}$$

where x is the data in question, p is a program, and $p \vdash_U x$ means the execution of program p by machine U produces output x. Hence, we seek the shortest possible program that computes the data in question.

For both periodic and random strings (that is, strings generated by an i.i.d. variable), it is known that Shannon entropy and Kolmogorov complexity agree up to a constant [LV93]. Moreover, it is known that there is a link between entropy and the expectation of Kolmogorov complexity for a distribution [GV06]. For these reasons, H and K are often linked — note, however, that these linkages do not imply any particular relation in the general case, and especially not if we restrict our attention to finite data. A serious drawback of Kolmogorov complexity is that it is known to be uncomputable [Cha98], and for that reason is typically rejected as a measure without much philosophical consideration.

While Kolmogorov Complexity may be known to be uncomputable, often the attempt is made to use compression algorithms as means of measuring the meaningful content of a pattern, having removed the obvious regularities and redundancies. *Lempel-Ziv compression* is a popular algorithm for the lossless compression of binary data used, for instance, in the GIF image format. The original concept was proposed by Lempel and Ziv [ZL78], and later utilized in the specific LZW algorithm by Welch [Wel84].

The LZW algorithm works by creating dictionaries of necessary symbols from the base alphabet; Given a sequence in order, the system will create dictionary entries for one- and two-character sequences. As the process continues, successively longer sequences of symbols will be created — at each stage, the dictionary code for the best new symbol will be output, until the entire message has been codified. Hence, LZW is a frequency-based algorithm which removes repeated sequences. The particular LZ algorithm LZW is not optimal, in fact, the precursor LZ78 created better codes, but LZW is significantly faster[2]. We can construct a complexity

[2]Note that LZ algorithms are often implemented with a maximum code size, skewing their compression rates for large data.

measure from the LZW compression as follows:

$$LZW(x) = |x'| \tag{2.2}$$

where x' is the lossless compression of string x by the LZW algorithm.

Compression algorithms, especially Lempel-Ziv algorithms, have been used as an approximation of Kolmogorov complexity on several occasions. This usage, in any individual case, is likely a poor choice. While LZW may be a good measure of regularities based on frequency, Kolmogorov complexity is far more robust than this; Although it has been shown to agree with Kolmogorov complexity is cases of high regularity and high randomness, it can easily be shown to disagree with K on simple and fundamental examples[3].

Often, we would like to include a notion of the computational complexity required to compute a pattern; While computational complexity is a useful notion for the discussion of algorithms, it fails for the discussion of patterns, since one can always design a program that outputs some given data in time $O(1)$. One solution to this is the use of Bennet's *Logical Depth* [Ben86]; In this definition, the complexity of a given piece of data is related to the running time required by the shortest algorithm which computes it. That is, given a universal Turing machine U and data x, we define the logical depth, D, to be:

$$D_U(x) = \tau_U(p*), \text{ where } p* \vdash_U x \tag{2.3}$$

where τ is a measure of execution time, and $p*$ is a shortest-length program computing x.

Note that by the criterion of logical depth, both highly ordered and random strings are not complex — since they can be computed with simple '**print** x' statements. Numbers such as π or e, on the other hand, are considered complex, since it takes effort to print their digit expansions.

Another interesting approach comes from Adami; That of *Physical Complexity*, related directly to an organism's environment. Adami sets out to measure the amount of information that a given genome (or population of genomes) has about the environment; Ideally, he desires the shared Kolmogorov complexity between a sequence and a description of the environment. Since it is known that Shannon Entropy is related to the expectation of Kolmogorov complexity, it may be used as an approximation for K for populations. The measure may be approximated by computing the entropies of the values at single-sites of the genome. Adami states his suspicion that biological evolution will tend towards greater complexity in time, and provides a digital experiment as illustration [Ada02]. Unfortunately, Adami's simple model aside, the Kolmogorov complexity of the environment is likely more difficult to describe than that of an organism's genome, if for only the simple reason of relative size.

2.2 Double-Discrete Dynamical Systems

A dynamical system is a system that evolves through a parameter, usually interpreted as time. A double-discrete dynamical system (DDS) occupies a discrete universe; that is, a universe of

[3]This is easily seen by considering the compression of successively longer sequences of the expansion of a computable but normal number (one which is normal in every base is known to exist [BF02]), which will continue to increase in size, in contrast with its near-constant Kolmogorov complexity.

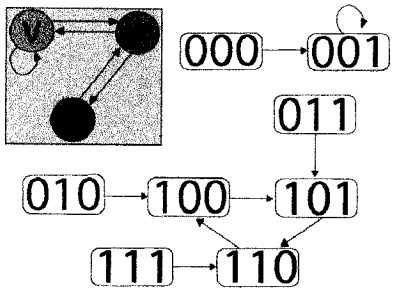


Figure 2.1: A boolean network (*inset*) along with state transition diagram; Arrows indicate movement from a state to its successor under an application of the (implicit) transition function.

isolated points, where a point may usually be denoted by a list of integers[4]. Each point in the universe corresponds to a state. We express states by time, as in S_{t-1}, S_t, S_{t+1}, where a state S is a representation of the state of our universe of discourse. Movement through states is usually represented via a transition function, Φ, such that $\Phi S_t = S_{t+1}$; The set of all possible states is known as the *state space*, and any particular representation thereof as the *phase space*.

A DDS may be displayed as a directed graph, where the transition function defines connections between nodes (states); This *state transition diagram* allows for some useful definitions: An *attractor* is a node or cycle from whence there is no exit; A *basin of attraction* are all nodes which lead into an attractor; A *garden of eden* state is one that has no predecessor. Figure 2.1 shows the state transition diagram, with connectivity, for a Boolean Network.

2.2.1 Cellular Automata

There exist many models of DDSs; Perhaps the most common of these is the Cellular Automaton, introduced by Stanislaw Ulam and John von Neumann in the 1960s [vNB66]. Cellular Automata (CAs) were an attempt to describe a discrete counterpart for continuous dynamical systems. For an introduction to CAs, see Ilachinski's text [Ila01]. They are often used as simple models in physics, as they are simple enough to allow for convenient calculation of various measures. Increasingly, CAs are also used as models of biological phenomena, including as models of pattern formation [DD05] [CDF+01].

A cellular automaton is a lattice of "cells", each of a particular *cell type*. Each cell may accept information from its neighbours, and decide what its state will be in the next time step; Hence, every cell in a CA may be viewed as an agent which acts on the basis of local information.

Formally, a CA may be described as follows:

- An alphabet of cell types (or "colours"), Σ; usually $\Sigma = \{0, ..., k-1\}$
- A (possibly infinite) lattice of cells, written (in the one-dimensional case)

$$S = ..., s^{i-1}, s^i, s^{i+1}, ...$$

- A *neighbourhood* of cells, μ^i consisting of a set of cells from the lattice; In one dimension, a radius r is specified, and $\mu^i = \{s^{i-r}, ..., s^{i+r}\}$.

[4] I use the term double-discrete to distinguish from systems which are discrete in time, but continuous in space, such as Iterated Function Systems.

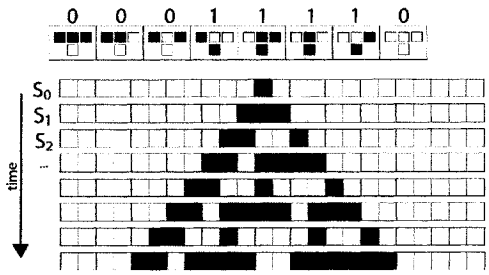


Figure 2.2: A representation of ECA rule 30 at the top, and the space-time diagram of its application to a finite lattice below.

- A *transition function*, $\phi : \Sigma^n \rightarrow \Sigma$ which maps from any possible neighbourhood of cells to a new cell type.

Hence, if we define Φ be the parallel application of ϕ to every site in the lattice, and S to be the entire lattice at time t, we may write the evolution of a CA as $S_{t+1} = \Phi S_t$. We refer to the application of time to the lattice as *evolution* (or *growth* or *development*[5]).

In one dimension, the space of all CAs is specified by the size of the neighbourhood, r, and the size of the alphabet, k; The total number of neighbourhoods is then $k^{(2r+1)}$, and the total number of CAs is then $k^{k^{(2r+1)}}$. The *Elementary Cellular Automata*, or ECA, are all CAs with $k = 2$ and $r = 1$, and are numbered from 0 to 255, by reading off the transition function value in the order defined by the domain; Neighbourhoods, however, become significantly more complex in higher dimensions. The application of a particular ECA, rule 30, is shown in Figure 2.2.

2.2.2 Classification of Cellular Automata

Wolfram [Wol83] provided an attempt at classifying the various types of CA:

- Class One: evolution leads to a homogeneous state.
- Class Two: evolution leads to a set of stable or periodic structures that are separated and simple.
- Class Three: evolution leads to a chaotic pattern.
- Class Four: evolution leads to complex structures, sometimes long-lived.

Perhaps a more interesting way to view class three is to see that any particular site will rely on progressively more distant initial sites; A very similar classification was used to describe another model of DDSs, Random Boolean Networks, including an "edge of chaos", analogous to class four [Kau00]. This classification is similar to the classification of continuous dynamical systems, except that the continuous version does not include an analogue of discrete Class Four [Sha81].

Unfortunately, despite the intuitive appeal of the CA classes, there does not appear to be any way to distinguish between the classes besides (the possibly flawed) visual inspection. To begin with, it is undecidable which class a given CA lives in, *even between class one and class two* [CY88]. Additionally, many attempts to re-formulate the CA classes have also proven uncomputable or otherwise problematic; These include techniques based on the decidability of achieving exemplar states [CY88], ability to grow patterns [BCFV95], fractal properties of

[5]Note that this terminology is likely to cause confusion in this document, given that we also discuss evolution and development proper; We leave it to context to disambiguate.

states [Wil84], and the extraction of attractors from time-limit sets [BKM97]. Shalizi refers to Wolfram's Classes (in the context of measuring complexity) as one of his zombie ideas — "ideas that should be dead but continue to eat brains"; Still, the division is intuitive, and many researchers attempt to classify CAs through empirical means. This includes Wuensche [Wue99] and Olivera *et al* [OOO01].

2.2.3 Computation in DDSs?

It has been repeatedly shown that Cellular Automata are a model for computation — this has been accomplished by the recreation of several known models of computation, including the general Turing Machine, in CAs; This work is reviewed by Mitchell [Mit98]. However, it is readily apparent that a functioning CA does not, in any immediately recognizable way, organize itself to replicate a known model of computation when "solving" problems; When a CA is found capable of solving a particular problem, the result is what we have termed elsewhere a "useful mess". The constructive dynamics of CAs are buried, and difficult to bring forth.

However, there is significant, although inconclusive, evidence to suggest that properties of known computational systems exist in Wolfram's classes, and hence exist in a "natural" division of the space of CAs. For example, Wolfram states his suspicion that all Class Four CAs are universal [Wol02]. Cook, in fact, has shown that this is true for ECA rule 110 [Coo04]. More interestingly, Li has put forward the thesis that the classes are closely linked to the Chomsky Hierarchy of Languages, that:

- Class One and Two CAs form regular languages
- Class Three CAs (appear to) form context-sensitive languages
- Class Four CAs (appear to) form unrestricted languages

He does so by comparing the power spectra of weighted minimum state-transition graphs for the ECAs, then comparing this statistically to the state complexity of the various languages [Li88]. Unfortunately, though tantalizing, both of these suspicions are far from proof: Firstly, it is difficult to claim that ECAs serve as an exemplar for the trends of CAs at large; Secondly, statistical similarity is far from proof; And finally, there does not exist any strictly formal characterization of CAs to compare to, just visual suspicions.

Hence, there exists no formal definition of Wolfram's CA classes, although there are several interesting correlations for ECAs; What we may be looking for instead is definitions thereof. Although similarly uncomputable, exploration of the relations between "natural" classes of CAs and formal languages may serve to help explain the constructive tendencies of DDSs in general.

2.3 Development

It is generally agreed that natural selection is the basis of the variety of multicellular life; This evaluation is based primarily on tremendous fossil evidence, but also increasingly on genetic analysis[6]. This is a most astonishing fact, that unselected properties of eukaryotes, along with natural selection, can give rise to the remarkable diversity of form and behaviour seen today.

[6] A comprehensive catalogue of the history of (primarily) multicellular life, along with discussion of supporting evidence, may be found in Dawkin's recent text [Daw04].

Figure 2.3: Perfunctory inclusion of some examples of the diversity of multicellular organisms. *(Photos taken at the Natural History Museum, London, U.K.)*

However, precisely how the genetic search associated with natural selection gives rise to the phenotypic patterns seen in nature is a poorly understood phenomenon, and the subject of a rapidly expanding field of Biology, Evolutionary Development.

A naïve view of the origins of multicellular life may be summarized by a position of Alfred Russel Wallace, written in 1870[7] [Wal70]:

> Universal variability — small in amount but in every direction, ever fluctuating about a mean condition until made to advance in a given direction by 'selection', natural or artificial — is the simple basis for the indefinite modification of the forms of life.

What Wallace is envisioning is, translated into modern terms, a view of phenotypic space as divided along a large number of axis, where points vary along some distribution. Organisms' traits vary along each axis independently, and through this random motion, evolution occurs. This view is quite close to what is practised currently in Evolutionary Computation, say, through a Genetic Algorithm performing an optimization on a set of real or boolean genetic variables.

Approximately one billion years ago, Eucaryotic cells began to work in tandem; One popular theory states that cells began to form rigid structures reaching upwards; This would allow the cells near the top to be carried by a stream to ever further locations, allowing for the creation of far-away colonies of genetically identical cells. This despite limiting the chances of survival of the cells near the bottom or middle of the structure; That is, the theory states that the first multicellular organisms were composed of identical cells which increased their genetic fitness by collaborating to form spores. The tremendous complexity of multicellular life — that is, morphology, specialization, and all other functions — followed from such a simple collaboration, benefitting genes, not cells [Fen02]. An important note may be made about this view of multicellular beginnings: it is nearly entirely dominated by physics and environment, and operates opportunistically on those; That is, the genetic component is almost certainly an accidental creation of a cell whose properties supported upwards growth and stability... everything else was the dynamics of the environment.

DNA is more of a program than a catalogue; This is largely evident from simply comparing the relative size of DNA to the number of cells in a human body, or approximately 3 billion base pairs versus 75 and 100 trillion cells. Indeed, there is no obvious relation between the size of DNA and the apparent complexity of a generated organism; For example, the onion genome is more than five times the length of a human's.

[7] Turns out that scarecrows are easier to find in century-old literature than in today's.

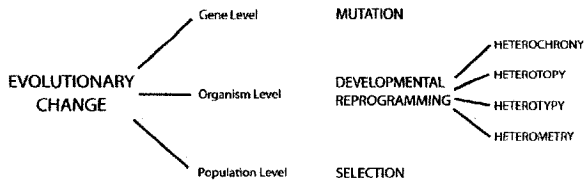


Figure 2.4: Arthur's logical relationships between the levels of evolutionary change.

Indeed, Carroll *et al* point out we have much more evidence than that; He notes that only a tiny fraction of our DNA, approximately 1.5 percent, codes for the approximately 25000 proteins found in our bodies. An additional 3 percent, or approximately 100 million bits, is regulatory — that is, it determines when, where and how much of the various proteins are made. The remainder is "junk", or DNA not believed to influence the development of the organism at all[8]. Hence, most functional DNA is regulatory, DNA which does not specify specific protein construction. Instead, regulatory DNA (somehow) determines when, where and how much of the protein are made; "Regulatory DNA is organized into fantastic little devices that integrate information about position in the embryo and the time of development ... ultimately transformed into pieces of anatomy that made up animal forms. This regulatory DNA contains the instructions for building anatomy, and evolutionary changes within this regulatory DNA lead to the diversity of form" [CGW05].

Müller [MÖ3] notes that any theory of development will need to account for "(1) the generation of initial parts; (2) the fixation of such parts in lineage-specific combinations; (3) the modification of parts; (4) the loss of parts; (5) the reappearance of lost parts; and (6) the addition of new parts". He goes on to note that proximate and ultimate causes will eventually need to be specified as a result as well[9]. Arthur [Art04] provides a break-down of the mechanisms believed necessary and sufficient for the emergence of those phenomena; He terms these the steps in "Developmental Reprogramming"[10]:

- Heterochrony: Different timing in the expression of genes.
- Heterotopy: Occurrence of different gene expression in different areas of space.
- Heterotypy: The amount of product being made by a gene, independent of time and place.
- Heterometry: A change in the nature of the produced gene expression.

Arthur considers this part of a larger picture of evolution, one which provides explanation for all the "natural levels" of evolutionary action; Mutation at the gene level, developmental reprogramming at the organism level, and selection at the population level (See Figure 2.4 for a schematic).

It is our belief that in Arthur's model the difference between "explanation" as understood by Computer Scientists and Biologists will be more clear; It seems likely that there are entire worlds of undiscovered computational theory between the models given above and mathematical explanation of developmental phenomena.

[8]This information is slightly out of date: recent research has suggested that "junk" may play some role in development after all; Regardless, Carroll's point still holds, as the magnitudes of numbers discussed increases only slightly (by far less than a factor of 10).

[9]Note that the informality of terms, such as "parts" and "precursor organs", seems to be common in developmental literature; I believe this to be a limitation of the precision of current language and understanding.

[10]Rephrased for brevity.

The role of environment should not be underemphasized in this discussion; Development is not possible unless embodied in some environment. There is, however, significant disagreement on how much role environment plays in embryogenesis. Many phenotypic features, for example, can be influenced in a lab either genetically or environmentally. For example: Waddington showed that the introduction of ether vapour to drosophila eggs can induce an effect otherwise caused by the biothorax mutation; An increase in temperature will lead to an increase in the size of "eye spots" on African moths, also known to be controlled genetically as well. Arthur terms these equivalences between environmental and genetic factors as phenotypic plasticity [Art04]. Certainly, no biologist denies the role of environment in shaping the development of an organism. However, these authors have significant doubt as to its ability to add structural information to an organism beyond simply canalizing possibilities.

The primary source of naïvité in Wallace's quotation, however, we have yet to detail; The view of development as a program, rather than as a catalogue of locations on a hugely-dimensional space, trivializes a canalization; This canalization is often termed "bias" by biologists, and reflects the interdependence of genetic "boxes", and the effects brought on by the environment. This is perhaps most easily seen by considering the evolution of the phenotypic qualities of animals' legs; It is not correct to say that the length of animal's legs are independent of each other, since the same genetic boxes control the determination of that factor. Similarly, it is not correct to say that animal's legs are all the same length, since genes may effect that property in some locations and not others, as in many animals with different lengths of fore and aft legs. This quality, often noted, is termed "modularity enhancing evolution", where modules are considered quasi-autonomous parts of the developmental system. Perhaps most interestingly is the occurrence of these modules *between species* — as is the case with the hindlegs of grasshoppers and moths — where *homologous* phenotypes (coding for hind legs) are shared, despite the gross enlargement in the grasshopper's case [Art04].

A commonly seen example of homologous phenotypes is seen in the phenomena termed "duplication and divergence"; This is the occurrence of many similar structures occurring repeatedly with slight divergence in form between occurrences, such as the digits of a hand or centipedes' legs (see Figure 2.5 for an example). Genetically, these are beginning to be understood; For example, the patterning of the antiposterior axis in animals appears controlled by Hox genes, believed to come from a single initial Hox gene early in genetic history. The Hox genes are believed responsible for the control over these modules (i.e. a regulatory gene), controlling the particular patterns and frequency of their occurrence over the embryo. As usual, however, we understand the phenomena significantly less well than could be desired - it is unexplained why certain species which appear to have less antoposterior patterning have much more Hox gene than do, say, humans — fish, for example, have double the number [Art04].

It is often said that the apparent complexity of developed organisms comes from genetics and the environment; To this, most evolutionary developmentalists would, in our opinion, object; It seems quite clear that significant amounts of this apparent complexity results from the dynamics of cell growth as specified by both environment and genome; Additionally, that these dynamics are not only influenced by environment, but also show remarkable resistance to environmental changes at times as well; This is evidenced by the re-use of genetic boxes in differing species. Hence, a third factor is required: that which Arthur terms "developmental drives and con-

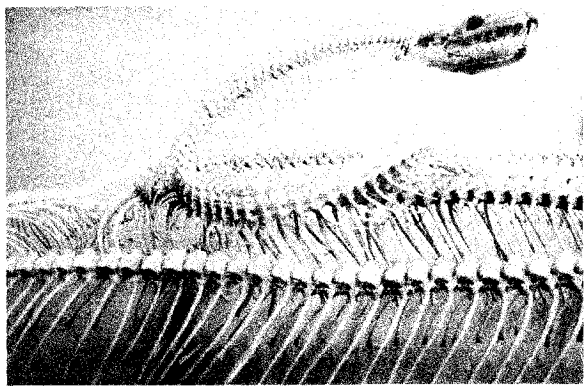


Figure 2.5: Example of "duplication and divergence" in the bone structure of a snake. *(Photo taken at the Natural History Museum, London, U.K.)*

straints". Many of these drives and constraints appear across environments and species, leading to the belief that they may be studied and understood in a general and abstract way [Art04]. Indeed, as much as environment and genetics are factors affecting development, there exists times where development seems impervious to their influence — for example, if one activates the gene named *tangled1* in the maize leaf primordia, one sees a highly irregular arrangement of cells, quite disruptive of the usual symmetric and structured order of the tissue; Regardless, the overall shape of the leaf will be unaffected, that is, the constituted organ will be unaffected by a large change to the constituent tissue [LD03]. We will discuss this matter further, in the context of artificial development, in section 3.1.

Finally, it should be noted that although in animals there is usually a clear distinction between developmental and adult life (egg hatching, birth, etc.), developmental dynamics never completely end. For example, Bissell *et al* stress the continuing importance of environment and dynamics in multicellularity; "It may be that cells never lose an intrinsic ability to morph from one cell type to another, and that they maintain a stable phenotype by integrating cues from the extra- and intercellular milieu. Indeed, there is also ample evidence to support the notion that, for a cell to continue functioning properly in a tissue-specific way, it must receive continuous signals to prevent growth or apoptosis and to maintain an appropriate structure and differentiation state." The authors go on to question the importance of mutations relative to microenvironment in the determination of an organism from a genome [BMRT03]. This property of developing systems is responsible for features such as self-repair and maintenance. Further, in some cases, such as the brain and its continual creation of new neural connections, development for an organism may never be viewed as complete.

Development is sometimes referred to as "natural computation" [CDF+01]. This is a naïve usage of the term, and likely refers to the process of self-organization also seen in visual representations of models of computation. That some models of computation easily produce patterns which resemble those found in nature (both biological and physical) is, however, a striking fact, and likely a motivation behind much current research.

Figure 2.6: Examples of natural patterns easily modelled by computational systems. (*left to right*): *Giant ammonite*; *Mesolite*, a crystal form found in igneous rocks; Zebra stripes; *Hebrew volute*, one of many shells displaying CA-like patterning. *(Photos taken at the Natural History Museum, London, U.K.)*

2.4 Artificial Embryogeny

There is significant interest at present in the use of development in automated design. Artificial Embryogeny (AE)[11] is a sub-field Evolutionary Computation concerned with biasing search through computation-like processes. In short, AE consists of the use of a non-linear dynamical system to map from representation to organism, inspired by, but not necessarily resembling, the mapping from genotype to phenotype in nature. There does not appear to be any accepted crisp definition; Indeed, the "Generative and Developmental Systems" track at the world's largest Evolutionary Computation conference accepts papers that deal with "the use, construction or evolution of genotype-phenotype mappings that involve either re-writing, iteration, time, or environmental interaction"[12]. Examples of AE in the literature include the use of: L-Systems, Cellular Automata, Morphogenic gradients, various models of Genetic Regulatory Networks, Multi-agent systems, and others (examples follow). The process of mapping from genotype to phenotype is often described as "pattern formation"[13], "development", "generation", or (confusingly) in the case of automata, "evolution".

Perhaps an appropriate distinction between developmental and non-developmental algorithms may be derived from Section 2.1; That, under some appropriate definition of complexity, a developmental algorithm has no relationship between the complexity of the genotype relative to the complexity of the phenotype, that the complexity of the phenotype is instead based on the environment, through constraining space or through provision of building materials.

Regardless, for now we shall refer to any system utilizing a non-linear dynamical system to map to a distinct phenotype as an AE model. When convenient, we shall refer to the space of genotypes as $g \in G$, and the space of phenotypes as $p \in P$, with the embryogenic mapping $\phi : G \rightarrow P$. A particular phenotype p will be a configuration of the developmental environment E, where E is typically a lattice, and p a specification of cell types on that lattice.

The first computational models of embryogenesis include chemical diffusion work by Turing [Tur52] and work with simple automata by Lindenmayer [Lin68]. Perhaps the best known models, however, are exceedingly simple proofs of concept using recursive systems: These include

[11] Also known as "Artificial Ontogeny", "Computational Development" and "Generative" or "Developmental Systems". We choose AE for purely aesthetic reasons, and hope that by being stubborn we can help create a standard. A taxonomy has also recently been proposed [SM03], although we will not utilize it here.

[12] From the GECCO-2007 website, description of track with organizers J. Miller, S. Kumar and K. Stanley, http://www.sigevo.org/gecco-2007

[13] Note that the term "pattern formation" is often used to describe strictly self-organizing phenomena [CDF+01]. We opt for the more general definition of simple growth and specialization.

Dawkins' BioMorphs, a simple user-guided evolutionary strategy that controlls the development of stick-based trees [Daw96]; A more impressive but similar example was the three dimensional morphologies developed by Sims as interactive electronic art [Sim99].

Much research revolves around the attempt to reverse-engineer Evo Devo, or to create "plausible" models of embryogeny. Other researchers attempt to exploit useful properties of AE in order to find solutions to engineering problems, or to develop better techniques for doing so. The following sections explore such models, divided by the aforementioned distinction.

2.4.1 Reverse-Engineering EvoDevo

Biologists refer to the (not entirely understood) interplay between genes and protein expression as a Genetic Regulatory Network — understanding these GRNs is considered key to understanding cellular differentiation. Many models exist to attempt to model GRNs in hopes of the emergence of existing phenomena — of current interest is modelling the Hox gene, and other regulatory mechanisms, believed responsible for many familiar developmental patterns.

One of the first proposed models of GRNs was Random Boolean Networks, proposed by Kauffman; These have quickly become a standard model for Discrete Dynamics in general, and have been studied extensively for their statistical properties [Kau00]. An early example of their use is by Delleart, who attempted to create a "biologically defensible" model of development. That is, a model which encompassed all relevant factors from biological development in realistic ways. Delleart's model consisted of a blastula of neutral cells which, after a pre-programmed symmetry-breaking, used boolean networks to develop an agent. After pre-seeding a GA with a hand-designed genome, agents evolved capable of performing simple locomotive tasks [DB96].

Others have used L-Systems, especially to model the growth of plants. This includes the attempt to model plant growth by Prusinkiewicz *et al*, who have recently successfully re-created phenomena including meristems, branching, tropism, amongst many others [PL90], [PRL06].

Kumar and Bentley present a system called the Evolutionary Development System (EDS); This software is designed to encode the primary features of biological development into a system capable of running detailed experiments. The system consists of pre-programmed concepts of proteins, cells, cytoplasm, genes, etc., and is intended to model gene expression with cis-regulatory regions. The functionality of the system is largely controlled by the interaction between existence of proteins and the genome, through the following mechanisms: Rate of Synthesis; Rate of Decay; The Coefficient of Diffusion; Interaction Strength; And the Protein Type. Hence, proteins diffuse throughout the system according to a differential equation, then influence the genomes of nearby cells in the creation of new proteins. The genes code for individual proteins, and expression rates are controlled by models of the cis-regions. These actions are undertaken by cells, each of which may be viewed as an autonomous agent.The EDS has been used for experiments involving assessing the abilities of the system, and has been shown to be capable of mimicking GRNs within cells, and some ability to grow embryos of particular shapes [KB03].

A more detailed and physically motivated model is provided by Eggenberger, whose model is similar to the EDS model described above, but includes significant modelling of the environment in which the cells grow as well, including the force of cells on each other. Eggenberger creates a good model of cell adhesion, and hence describes another physically-bound canalization of

development. Through this, he creates models of the growth of three-dimensional structures, including invagination [Hot97], [Hot03].

2.4.2 The State of the Applied Art

Most relevant to our current interest are cases where Artificial Embryogeny has been applied to the design of solutions to problems from engineering and related fields. AE possess several attractive properties which imply their potential use in situations where direct encoding might be impossible or intractable. AE techniques are believed to be capable of exploiting a canalization of development, allowing for the design of organisms too large for design by bijective encodings [KGK04], [HM06]. AE is believed to be a mechanism by which large complex systems may maintain themselves, executing self-repair following damage [Mil04]. AE allows for significant environmental influence on the development of organisms, allowing for the same representations to be used in the development of several different organisms [KGK04]. Finally, it has been suggested that AE growth might be used to generate not only the final organism, but also a contructive map, detailing a plan for the assembly of the final design [RP04].

Eggenberger Hotz *et al* have used development to grow neural network architectures of impressive size and complexity, capable of controlling a robot arm intelligently [HGP03].

Sekanina and Bidlo used evolution and a developmental algorithm to evolve sorting networks [SB05]; They began with a small, pre-designed sorting network, termed an *embryo*, and a genome (and constructor) — the genome was a linear list of instructions which copied or modified elements of the network; Through this developmental scheme, they were able to create a series of ever-larger working sorting networks reasonably independent of genome length.

Stoy and Nagpal use a Cellular-Automata-like technique to allow an undifferentiated mass of components to self-organize into a pre-determined shape; They have recently described the reconstruction of a CAD 3D model of a plane from an arbitrary configuration of (the correct number of) atomic components [SN04b]. There is some question of whether this could be considered Computational Development by the above criteria, since the action of the algorithm reconstructs a compressed pattern implicitly contained in the Cellular Automaton-like representation.

Often, the stated goal of such research is self-repair (as in [Mil04], [SN04a]), and occasionally it is hoped that applications may lead to biological understanding ([KB03]); However, the primary motivation found in nearly all applications of AE is scale-free design, where genotypic specification is mapped via developmental process and environmental factors to arbitrarily large phenotypes. The applications above, and many others, have shown that this is possible in a wide array of micro-worlds; Indeed, it is easy to see the potential for application in: Software engineering, as is explored by Miller [Mil04]; Circuit design, as is demonstrated by Sekanina and Bidlo [SB05]; Or robotic self-organization, as in the work of Stoy and Nagpal [SN04b].

There exists another field of application as well, less standard than the engineering-problem approaches above; This involves the growth of organisms for the purpose of commercial or fine art; Indeed, Sims' primary goal was purely artistic, an exploration of the space between an audience and an installation. Additional work has been made, including the modelling of plants, using principles of Computational Development. This includes the modelling of plant growth for the purpose of simulated graphics, accomplished through several means: L-Systems

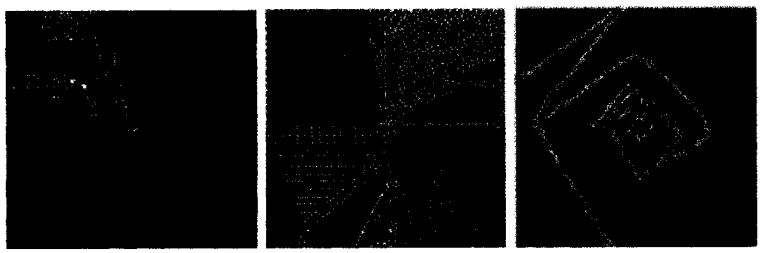


Figure 2.7: Interesting agents from Bluenome Phase One

were developed early on specifically for this purpose [PL90], and more recent work uses a more biologically informed model of genotype-phenotype mapping [Coa97], [GMS03].

There are two existing models of embryogenesis closely related to Cellular Automata. In Basanta's model, EmbryoCA, a single seed cell is placed in the centre of a two-dimensional grid, with successive growth controlled by a modified two-dimensional CA. Basanta *et al* demonstrate the ability of their model to grow simple geometric shapes [BMBH04]. A second example is the work of Kowaliw *et al*, Bluenome, described in detail and critiqued below.

Bluenome

The following is a description of the author's previous work [KGK04]; A more detailed description may be found in the earlier Master's Thesis [Kow03]. Bluenome is a model of AE, similar to Cellular Automata; Indeed, generally speaking, Bluenome may be viewed as a subspace of the space of CAs, although we will utilize metaphoric language from biology more often than the language of automata. Although Bluenome is a better model of biological phenomena than general CAs, it should not be viewed as a model with any level of detail or predictability; At best, Bluenome should be viewed as an abstract model of embryogenesis which takes differentiation as a pre-programmed given. Our interest in this particular model stems from a suspicion that the sort of canalization imposed by biological constraints will lead to a more evolvable and controllable model of dynamics than CAs; Indeed, two phases of experiments were undertaken to attempt to show precisely that.

The Bluenome Phase One experiments were designed to demonstrate the evolvability of the Bluenome model. Simple rectangular environments were designed, system parameters for the number of cell types and length of transition function were set, and agents were allowed to grow without bound (until the environment was filled); Various experiments utilized different fitness functions, working towards the demonstration of reliable increase in fitness (or lack thereof). Figure 2.7 shows some agents found throughout all runs; These images show that naïve complexity, symmetry and lack of symmetry, and visually impressive designs are possible.

Figure 2.8 shows an example of several members of a population of agents generated through the "complexity" measure after approximately 100 evolutionary time steps. A visual continuity may certainly be seen between them, not only by the cell types included, but by the propensity to grow "branches", and the almost fractal boundaries with the empty (white) cells.

There were also several axes shown to be (at least) difficult to evolve; These included entropy, where organisms resembling "pure noise" were selected for; And self-termination, or organisms which would grow to some size then stop without external influence.

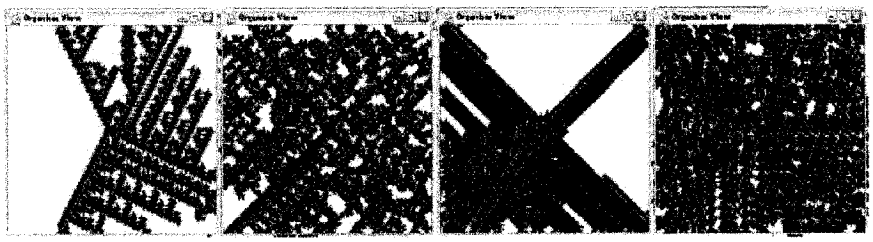


Figure 2.8: Snapshot of several members of a population.

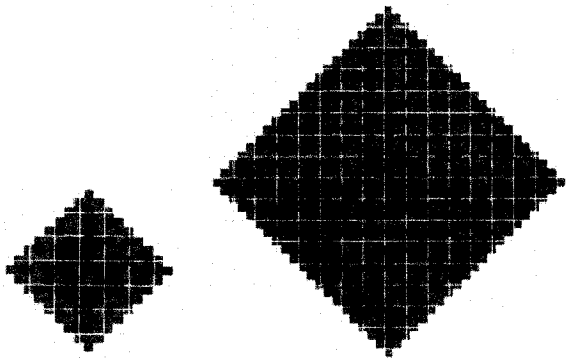


Figure 2.9: Two "Tarasanoids" regrown from the same genotype.

In Phase Two, vastly simplified animal cells were defined, and embodied in a two-dimensional world containing a variety of distributions of "food". The Bluenome model was used to develop agents adept at surviving in this world; That is, conglomerations of the cells were grown through the Bluenome algorithm, provided with some initial amount of "food", then were released into a world. The fitness of an agent was the sum of the survival times of the cells — through this fitness function, agents were found able to sense, approach, absorb and distribute food throughout their bodies.

Several exciting results were found through experimentation with these "Tarasanoids". This included the comparison between the growth of agents using the Bluenome technique and a bijective technique; As expected, there was a point at which the phenotypic size caused the bijective technique to "bog down", that is, fail to improve with time, while the Bluenome technique did show additional improvement. Additionally, several recognizable high-level strategies did emerge from evolution, and were informally catalogued by function; Finally, it was informally demonstrated that genotypes useful at one level of phenotypic complexity were also useful at other phenotypic complexities, relative to fitness evaluation (See Figure 2.9 for a visual example).

The Bluenome experiments have several directions for possible improvement; The shortcoming of the Phase One experiments was in a lack of "meaning" of the axis of evolution — naive definitions of complexity, entropy and tissues may serve as a proof of concept, but do not readily extend beyond this role. Phase Two made steps towards correcting this. In choosing an application in which it seemed likely that Bluenome would be effective, a biologically-inspired application was chosen; Additionally, computational constraints demanded that the world also be highly simplified — the result, unfortunately, was a system too simple for biologists, too biologically-minded for practical use.

2.4.3 A Critique of Existing Work

There are, in our opinion, two areas in which current work in AE needs improvement: The first of these, in analysis, involves the level of explanation which we would like to obtain for our models; The second, balancing of biological plausibility with potential application, is more subjective, and is tied up in a researcher's motivations.

The main feature of simplified modelling for AE is the ability to generate near-limitless amounts of data, and to examine that data as closely as possible; It also involves the use of counterfactual reasoning to see generalizations of real-world trends. Unfortunately, there is no single obvious means of describing the functioning of a developing algorithm at intermediate stages; There is the level of atomic components, too large and "messy" to be understandable, and there is the level of fitness, too coarse to indicate anything save the efficacy of the result towards some goal. There have been some limited attempts at characterizing the patterns of growth, as in Miller's French flag model [Mil04], but these have little explanatory value. The value of the computational approach, and the potential improvement of the models, requires a more constructive understanding than is provided through statistical correlation. Empirical study on model-level perturbations is one route to this understanding; A better route although beyond the scope of this thesis, would be the automated discovery of predictive sub-environment-level structure.

The second issue with current work is the tenuous relation between modelling and algorithm design; There is good reason to accept biological mechanisms as inspiration for design reasons[14], but there are limits, and we question the value of the creation of ever more realistic models. Statistical correlations to models of reality will only serve as an approximate of what can only be proven in a lab, and the more accurately a system models reality will only constrain the observable phenomenon — this leaves little room for improvement on simple lab work. While these models may serve to confirm in the future that all relevant components required for the development of a known organism are sufficient, it is doubtful that such a model will provide additional understanding of the phenomenon in question. Instead, we are more interested in what occurs when a biologically-informed system diverges from reality. Indeed, there is significant history of simple counterfactual experiments in AE (re-)discovering interesting principles;

As an example, consider the topic of design of environments, where Kumar and Bentley note, "the reason for the success ... [was that] their environments were rich with dynamics such as surface tension, gravity, friction, inertia, moments and energy dissipation... A simple environment reduces the capacities of development. A rich and complex environment enhances the potential for development" [KB03]. In brief, this may be summarized as: the environment provides a canalization of development, a tunnel through the large space of possible phenotypes. (Perhaps unlike Kumar and Bentley) we do not consider this a reason to abandon simple environments in favour of richer ones, but instead to explore the limits of what can be done in the simpler environments relative to the richer ones — it is well known that simple worlds can lead to remarkable patter formation (as in Packard's Snowflakes, for example [PW85]), and we are left with the question of what additional requirements there exist for automated design, or canalization of space so as to make it searchable. This, in the author's opinion, is the primary means by which AE may contribute to understanding biology — by discovering general prin-

[14] We have many working examples, after all!

ciples through abstracted experimentation which, ideally, may be tied to real-world data in a predictive way at some later date.

2.4.4 A Critique of a Critique

On the topic of modelling EvoDevo, Carroll writes [Car05]:

> Still, the central importance of genetic switches to pattern formation has not yet fully penetrated the computational modelling world; for example see S. Wolfram .... The continuing mistake is being seduced into believing that simple rules that can generate patterns on a computer screen are the rules that generate patterns in biology.

This is, unfortunately, a simplification of Wolfram's concepts and arguments[15]. Wolfram's primary argument (in this context) involves the universality of computation in nature, and the equivalence of the various models thereof [Wol02]. While Wolfram's argument might be vague and largely unsupported, it contains Genetic Regulatory Networks as a subset; Hence, a central tenet of Carroll's book, that GRNs along with switches are capable of supporting the patterns found in nature, is in fact *evidence* for Wolfram's thesis. The claim that other systems, such as Cellular Automata, *are* capable of supporting the same patterns as GRNs and switches becomes trivial: GRNs and switches are an algorithmic process, and CAs are a model for computation. The confusion here seems to be in comparing a higher-level of theorizing with direct modelling; Wolfram's "suspicions" are not claimed to be a direct, simple or efficient model, and the level of vagueness in his claim affords the possibility that the CAs be used to model GRNs, or some equivalent model. Carroll seems to miss the possibility that a smaller-scale CA may allow GRNs as an emergent phenomenon, or that a coarser-scale CA may model, precisely or approximately, the behaviour of the GRNs — the latter possibility is, in fact, crucial to the success of Deva.

In any event, it should *not* be, in our opinion, the goal of computational development to model real-world embryogenesis, any more that the goal of calculus should be the modelling of thrown baseballs, or dynamical systems the modelling of pendula; Although there is always a hope of reverse-engineering principles applicable to biology, the goals of computer science and mathematics should, in general, be wider and aimed at greater abstraction — this, in my opinion, is the source of mathematical understanding. Of course, there is never a clear notion of what should be dictated by real-world data and where abstraction should begin; The only approach is to make an educated guess and try.

2.4.5 Complexity and AE

AE has largely been spawned from interest in the canalization of development (channelling) found in natural systems, and the hope that general and useful principles may be reproduced by computational models. Developmental systems, both natural and artificial, may be viewed as constraints of the space of all possible phenotypic configurations, and, at least in artificial cases, prevent large portions of the phenotypic space from being searched [HM06].

[15] Otherwise, Carroll's book was a wonderful introduction, from which I have borrowed sources and quotations liberally

There is ample demonstration that this canalization may be used for constructive purposes, allowing, amongst other things, for very large organisms to be designed. Banzhaf and Miller phrase one of the goals of AE eloquently: "the challenge is to radically dispose of the complexity limits for the evolution of computer code, and aim at complexities heretofore only achieved by large teams of human programmers" [BM04].

Unfortunately, in Artificial Embryogeny and related systems, the definition of complexity is usually left implicit. For instance, in his discussion of the generation of complex programs, Banzhaf discusses complexity only in terms of the number of components, stating simply that the generation of a program with a trillion components is "impossible" by "conventional methods" [Ban04]; Of course, this is an approximation of truth. It is quite easy to, for instance, design a set-theoretic program which outputs the number one trillion using a trillion nested successor functions. In another paper, Banzhaf and Miller instead use a more Algorithmic Information-like discussion of complexity, stating an overall goal to be to "evolve a program whose purpose is so complex that it *requires* 100,000 or a million lines of code or 10,000 modules of average size 100 lines of code" ([BM04], emphasis added). Of course, this is also an approximation — a strict reading of this goal would require that the developmental solution also require a million lines of code to achieve the program in question[16], raising the question - why use an embryogenic approach?

Another approach has been taken by Hornby, who is interested in the use of developmental systems for evolutionary design. For Hornby, the term complexity should refer to our "interest and the ability to produce designs"; Hence, his notion of complexity implicitly encapsulates characteristics which directly contribute to the fitness of the generated organism. Hornby shows that the inclusion of explicit mechanisms encouraging modularity, regularity and hierarchy are positively correlated to the generation of highly fit organisms in the development of table designs; "These measures of design complexity more intuitively measure the structural complexity of an object than the Kolmogorov complexity" [Hor05]. Hornby's usage actually implies the opposite of the notion of complexity utilized by most physicists, most easily seen when he demonstrates that a random bit string would have very low complexity by his measure.

The actual need for a developmental stage in complexification has been called into question recently by Stanley; Stanley uses the term complexification to mean that "evolution can elaborate and increase the complexity of its products by adding new refinements and divisions [new genes] during embryogenesis... this process of complexification allows evolution to discover more complex phenotypes than would be possible through optimizing a fixed set of genes." He then proposes a network of functions applied in composition as a model in which complexification may occur; These CPPNs (Compositional Pattern Producing Networks) accept a location in space as input, then output a value by feeding through the network of functions, outputting a single value representing the specialization at the original site. Interestingly, there does not exist any obvious developmental stage. Through these networks, CPPNs may be found to implement many regularities believed significant to the canalization of space for biological design. Stanley uses a subjective evolution to generate images displaying the desired regularities — throughout

[16]Consider: our task may be solved by a program developed by an embryogenic technique. Then, the Algorithmic Information of our task is no greater than the length of the embryogenic system, plus a command which runs the output. So, either our initial goal was not, in fact, that complex initially, or our embryogenic technique requires a million lines of code.

the "complexity" of the image is referred to implicitly as the amount of (visually estimated) intricacy in the output image [Sta06].

Some authors in AE literature are directly interested in compression-based measures; For instance, Lehre and Hartmann [LH04] use the Lempel-Ziv compression algorithm as an approximation of Kolmogorov complexity (See Section 2.1 for a discussion of this approximation). Lehre and Hartmann show that the LZ compression algorithm corresponds to an intuitive and predictive measure of problem complexity, helping to determine the appropriateness of AE techniques for problem solving. A similar measure is utilized by Federici and Downing who order target patterns by "complexity" in an argument involving the inclusion of "evolutionary stages". The problem is comprised of growing images to match given patterns, each of which is given a complexity rank through a general-purpose compression algorithm, ARJ. Approximations of these patterns are then evolved through the developmental algorithm, implementing a neutral complexification, that is, a designed set of complexifying steps [FD06].

The above discussions no doubt take a more literal reading than was intended by the original authors; Indeed, Banzhaf and Miller foreshadow some of our arguments later in their paper when they note that "the complexity of the organism stems mainly from outside and has not to be provided by the genotype". However, this precise interpretation does highlight the difficulty in finding a more precise formulation, and outlines how ambiguity can slip into our discussions as a result.

2.5 Static Structures

Trusses are well studied examples of structural design, being used by architects and engineers in nearly all construction; Often, they are cited as the simplest such model. Still, as an approximation of actual real-world structures, trusses are close enough to be suitable models for most small construction projects, and are typically used at least in the initial design phase of nearly all large construction. Truss-based structures invisibly form the basis of nearly every large building or tower, but are most obviously visible in bridges, hydro towers, house roofing (Several examples of constructions resembling space trusses are shown in Figure 2.10) — although a simple model, truss design can be exceedingly complex. As such, trusses are an appropriate choice for evaluating a model's ability to perform structural design, allowing for an evaluation of those designs from a completely independent context.

We are interested in a model which consists of joints, beams and grounds. All beams are connected via joints, which may be connected to grounds. We will provide a structural diagram composed of these three sorts of components, and ask if the structure (a) is stable, and (b) is capable of supporting weight. Figure 2.11 shows two trusses; The first is stable, but the second is not — any external force would cause the second to deform drastically. Another important issue involves the deformation of the truss members under strain; Given some beam and an external force, a beam will either compress or stretch, which in turn will cause the truss' joints to dislocate. Some dislocation is to be expected, indeed, some dislocation is precisely the advantage to the use of joints rather than rigid connections. Too much, of course, will compromise the design. Such a model is called a truss — in our case, a *plane truss*, as our model is two-dimensional.

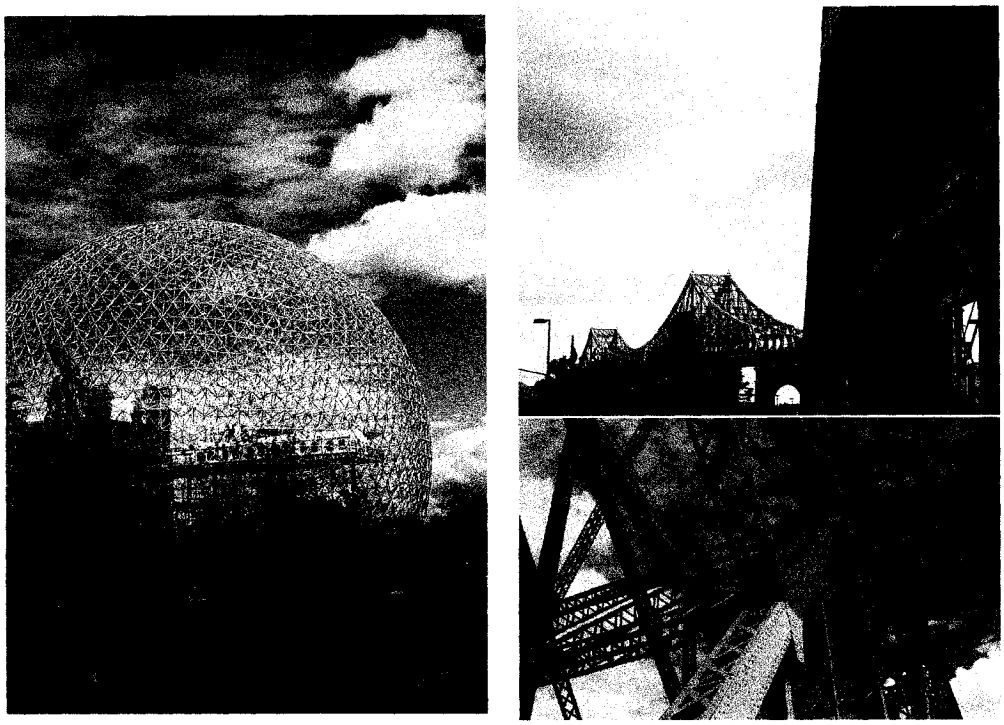


Figure 2.10: Space Trusses in construction: *(left)* la Biosphère d'Environnement Canada, *(right)* Pont Jacques Cartier; Montréal, Canada

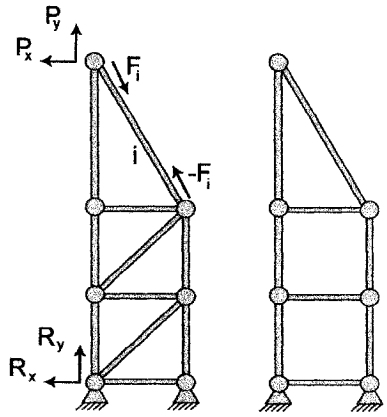


Figure 2.11: Two plane trusses, the left is stable, the right unstable. Labelled on the left: external force P applied to the top joint, reactive force R from a base joint, member force F_i of the ith member.

We will analyze a given truss by assuming that the structure is resting at equilibrium, that is, that every force acting on the truss is balanced by an equal reacting force — this will imply that the force in each beam, either compression or tension, is balanced. If this is not possible, then we are assured that the structure is not stable — if it may be balanced, then we may ask questions about the maximum amount of force going through any particular beam, and reason about maximum weight load.

The assumptions implicit in our model are as follows:

- All member beams are pin connected, at an infinitely fine point

- All external loads are applied only at joints[17]
- All members are two-force members

These assumptions are made to simplify computation, but can be shown to reasonably approximate real-world conditions.

2.5.1 Truss Stress Analysis

We now examine the computation of member-forces in an arbitrary plane truss[18]. This section is a shortened version of Appendix C.1, derivation of the following techniques may be found there.

There exist some simple counting tests that may determine if a given truss is unstable. Failing that, we must attempt to compute the equilibrium state given some external forces — in the process, we obtain values for all member forces. In our example, all truss members are identical in terms of material and area, grown in a developmental space where units (and hence also r_c) are measured in meters; We set $EA = 1.57 \times 10^4$ N, corresponding to a modulus of elasticity for steel [Meg05] and a cylindrical member of diameter 1 cm.

Consider a general truss with n joints and m beams; We are provided with external forces to be applied at joints, and wish to determine the member forces. Let our structure forces be $\{P\} = \{P^1, ..., P^n\}^T$, structure displacements be $\{\Delta\} = \{\Delta^1, ..., \Delta^n\}^T$ and member forces be $\{F\} = \{F^1, ..., F^m\}^T$. We may relate the individual member forces to displacement and structure forces as follows:

$$\{F\}^i = [k]_a^i [\beta]^i \{\Delta\} \tag{2.4}$$

where $[\beta]^i$ is the connectivity matrix for the ith member beam, and $[k]_a^i$ is its stiffness matrix, relating the deformation of the beam under a given force to the displacement at the joint. Hence, to solve for forces, it suffices to compute the displacements. The displacements may be computed through a truss stiffness matrix, a combination of the individual member stiffness matrices:

$$\{\Delta\} = [K]^{-1}\{P\} \tag{2.5}$$

Hence, given a plane truss, we may first compute the stiffness matrix, then compute the displacements, then the individual member forces. The entire process is bounded by the calculation of a matrix inversion (or LU-Decomposition), and hence has running time $O(m^3)$.

2.5.2 Evolution of Structures

There has been significant interest in the evolution of structural designs. This has included several frameworks for their analysis, including plane and space trusses, simplified models of Lego, and others. The Lego and related simple models have led to some interesting research in design, including the early development of buildable structures [FP97], or, more recently, the use of AE for the design of a simple arch, including scaffolding [RP04]. However, since we desire a notion of structural design which may be evaluated through means external to the A-Life community, we will instead concentrate on models taken directly from Engineering.

[17] We will simulate the weight of each beam by adding downward forces to each joint.
[18] This analysis is taken largely from West's treatment [Wes89].

There have been many attempts to use EC for structural design — an extensive recent review was conducted by Kicinger *et al* [KAJ05]. Typically, use of EC in structural design concentrates on optimizing the sizing or shape of existing frameworks — our work, however, involves topological design. Use of a GA to optimize a topological design through a relatively bijective relation between genotype and phenotype has been conducted by Rajan [Raj95] (who also optimized sizing and shape). A more complex approach was undertaken by Yang and Soh, who used a GP approach to optimize topology in the context of tall buildings [YS02]. Kawamura and Ohmori used a constructive (but bijective) technique to design a series of small but architecturally pleasing trusses [KO01].

Chapter 3

A Simple Complexity

Complexity is a topic often discussed in Artificial Embryogeny (AE) and Biology; Indeed, the notion is fundamental to several interesting open topics. Unfortunately, there does not exist any single accepted definition of the term, and many authors use it implicitly; This severely undercuts our ability to evaluate related hypothesis.

For instance, consider the term "complexification", typically used to describe the increase in phenotypic (organismal) complexity which accompanies the growth of genomic complexity during evolution. Whether such a (causal) link exists in natural history is a much discussed topic in biology today ([McS91], [Gou02]), and indeed, has generated some interest in Artificial Embryogeny as well ([Sta06], [Alt94]). However, without any existing standard for measuring genotypic or phenotypic complexity, any relevant claims are simply not falsifiable. This is a pity, since Artificial Embryogeny is uniquely capable of evaluating these claims, as AE is uniquely capable of generating control data, consisting of, say, random genetic drift bounded below. As a second example, consider the notion of the existence of an "edge of chaos", that is, a region between orderly and chaotic growth where "interesting" algorithms lie; Evaluation of any such notion is also highly dependent upon the choice of complexity taken [Bro07].

This chapter begins with a discussion of notions of complexity in Biology and AE, reviewing existing work related to the topic, discussing motivations, weaknesses, and interest specifically to AE. Of particular interst will be the role of the dynamic and the environment in these measures, and the biological motivations for such considerations. We will *not* be interested in related topics such as Complexity Theory or Self-Organization, believing this first stage to be a necessary precursor to those more ambitious topics.

Consider the question, posed quite informally: does the genome of a tree anywhere encode the information that "the bottom of the tree is fat, and the top skinny"? We suspect that it does not; Instead, we believe that the genome of a tree specifies some cellular layout for branches generally, while the relative size achieved is controlled by the availability of resources, constrained by the thickness of the base. Another: do the genes controlling the development of a vascular system specify its overall morphology? Current research suggests that it does not, that vascular growth instead expands to accommodate available structure, overall morphology being specified by other genes (amongst other factors).

It is our suspicion that current work in AE, especially views on complexity, do not sufficiently

consider the role of environment and dynamics in development. We will propose a modification of Kolmogorov complexity, along with some other measures, in an attempt to rectify this. In so doing, we will derive a definition with properties quite different from those of traditional Kolmogorov complexity, notably, computability.

Next, we will define perhaps the simplest model of AE possible, the Terminating Cellular Automaton (TCA). Using a one-dimensional version of TCAs on simple binary strings, we will be able to compute our defined measures (although doing so via brute force would likely be intractable in some other, less minimal model). We then contrast the measures with each other, discussing similarities and differences.

3.1 The Importance of Environment to Embryogenesis

AE has largely been spawned from interest in the canalization of development (channelling) found in natural systems, and the hope that general and useful principles may be reproduced by computational models. Developmental systems, both natural and artificial, may be viewed as constraints of the space of all possible phenotypic configurations, and, at least in artificial cases, prevent large portions of the phenotypic space from being searched [HM06]. Of course, the trade-off involves easier access to other portions of a typically large and un-evolvable phenotypic space; The same principle is described by Gould in natural systems: "Lest we begin to suspect that rigid limitation must represent the major evolutionary implication of such a constraint, I must re-emphasize the positive aspect of constraint as fruitful channelling, along lines of favourable variation that can accelerate or enhance the work of natural selection" [Gou02].

For instance, consider the case of the divergence of the echinoderms from their bilateral ancestors. Bilaterals are characterized by, amongst other traits, a vascular system carrying blood, a bilateral morphology; The echinoderms, on the other hand, have a radial morphology, and a transport system which carries water. Remarkably, the regulatory genes which code for these developmental traits displayed minimal changes — instead, it is their role in the dynamic of development which has altered in evolution. "The highly derived body architecture of echinoderms evolved at least in part through extensive modifications in the roles and expression domains of regulatory genes inherited from their bilateral ancestors. Even the limited number of genes and species we examined demonstrates a remarkable evolutionary flexibility" [LW97]. It is surprising, as Gould notes, "that the evolution of differentiated and specialized *Baüplane* from a presumably homonomous common ancestor proceeds... by reduction and restriction, rather than by addition of genes or expansion of their domains of activity," quite unlike the mechanism specified by theories of complexification [Gou02].

This is quite a divergence from the motivations of Physical Complexity proposed by Adami. Here, we are emphasizing the ability of the dynamic of development to utilize the environment in which it is found and adapt; This is not the same concept as the genome "having information about the environment"; Instead, the genome specifies a program which can adapt to any of a wide class of environments. It is having provided the environment "for free" that we see value in measures for practitioners of AE; We do not seek a genome which *describes* our environment, but instead, a genome which can *exploit* an arbitrary environment. A motivation to do so for the AE practitioner is that the simpler the genome, the easier it is to find through Machine

Figure 3.1: A miscellaneous echinoderm (*left*) and a miscellaneous bilateral (*right*). *(The Basket Star was photographed at the Natural History Museum, London, U.K.)*

Learing.

Several issues typically resolved somewhat arbitrarily in AE systems might become more intuitive when environment is included more explicitly. For instance, termination is an issue that pervades the Theory of Computation, and has been inherited by many AE models — when do we stop re-writing the graph, or running the CA or replacing the sub-components? Increased consideration of the developmental environment should provide natural answers, shifting the focus to the question "does the cell / component have the requisite materials for reproduction / specialization?"

3.2 What Notion of Complexity is Desired?

We aim to discuss which measures of complexity are most appropriate for AE — definable and computable descriptions which can help to formalize our intuitions, and, hopefully, to design and guide future AE models. Our perspective for this complexity tends to be based on a cellular granularity; That is, there appears to be a tendency to visually display cells or components, but not morphogens or resources, when discussing phenotypic complexity. This is likely influenced by biology, where time-lapse photographs of cells are sometimes used to study embryogenic phenomena (as in the study of angiogenesis by Zamir *et al* [ZRL06]).

Further, we shall have no issue with awarding random data the highest complexity value with our measures; Random configurations require the most effort to specify, and in our opinion, thus qualify as the most complex. In the absence of a cohesive Complexity Theory, we should not make assumptions about what Self-Organization is at this particular stage, instead satisfying ourselves with a more comprehensive measure of developmental effort.

3.2.1 "Naïve" Organismal Complexity

It is clear from the above discussion that there exists a naïve notion of phenotypic complexity which pervades our discussions; This is not a formally defined notion, but instead an ad hoc description based on visual inspection. The obvious analogue from Biology is Structural Complexity; Unfortunately, its use in AE directly is of dubious value. While the existence of several forms of differentiated cell in an organism is evidence of some level of effort in the developmental stage, it is not nearly so indicative as that found in biological systems. In natural systems, the inclusion of the possibility of different cell specializations requires many resources and great

sophistication, whereas in most AE models, the cell types are simply pre-programmed, waiting to be activated through a single available variable. We consider it highly unlikely that the predictive value found in biology will be recovered in the AE analogue.

Instead, we desire some measure that will capture the effort required, or the information contained, in the development of an organism. Actual computation of information would likely require a notion of work, which is probably impossible given our perspective; Instead, we propose using a familiar substitute, Shannon Entropy. In cases where the AE model begins with a static number of undifferentiated cells, normalized entropy may be used. If the set of all cell types is C, we have

$$organplexity(A) = -\frac{1}{\log |C|} \sum_{i \in C} p_i \log p_i \tag{3.1}$$

where p_i is the frequency of cells of type i in organism A. In other cases, where the number of cells grows from some small mass, we may multiply by the number of cells,

$$organplexity(A) = -\frac{|n|}{\log |C|} \sum_{i \in C} p_i \log p_i \tag{3.2}$$

where $|n|$ is the number of cells in the organism. The use of entropy as a measure has the desirable property of additivity, corresponding to intuition.

Note the significant divergence from notions of self-organization in statistical physics, where many authors use a *decrease* in entropy to describe self-organization [SSH04]. While this might be appropriate for the description of how a randomized medium transforms to an ordered one, it is not appropriate for our current perspective on AE, which begins with a uniform mass of cells (or lack of cells altogether) that then grows and specializes[1]. Embryogenesis, viewed at the cellular level, is an increase in disorder overall.

Unfortunately, it is impossible to compare this measure to any known relations to determine its correctness; Indeed, some biologists point out that the evidence for complexification in nature in general is inconclusive [McS91], which may undercut our reason for believing that any such measure exists at all; Regardless, we offer *organplexity* as an improvement over implicitly defined notions, and hope that usage proves it useful.

3.2.2 Genomic Complexity

Genomic size is easily defined in AE — for variable-length representations, simple length will do. A better measure is often the number of utilized genes, those sections of the genome which are activated during development, this latter measure being applicable to many fixed-length representations as well. However, we desire to extract the important parts of the genome, removing the redundant and non-functional portions; This will help us deal with issues such as redundancy and (the genomic data formerly known as) "junk". The effect of so doing should

[1] Differential models of morphogenesis may seem exceptions to this trend. For instance, Merks *et al* note that when simulating vasculogenesis using a differential model, it is best to begin with a collection endothelian cells distributed randomly over a lattice; It is likely that application of their model decreases entropy in this case [MG05]. While this may be an accurate description, we note two issues: Firstly, the effort required to generate the random distribution is discounted, which may have been greater than the effort involved in the subsequent vasculogenesis; Secondly, it is possible that although a random distribution models reality well, that some other process, not random at all in the sense of Kolmogorov, may also serve, hence not necessarily requiring a decrease in entropy in the first place.

not be underestimated: Recall, after all, that the onion genome is five times the length of that of a human's!

The most direct means of capturing this is with Kolmogorov complexity. We wish to redefine this measure so as to be more applicable in describing the information in an organism's genome, and not what is given "for free" by the environment.

Environmental Kolmogorov Complexity

Let us assume a space of genomes, $g \in G$, a space of organisms, $p \in P$, and a process, AE, for mapping from one to the other: $g \vdash_{AE} p$. Also, let us assume that we have a measure on G capturing the length of a representation, $|\cdot| : G \to \mathbf{R}$.

Given some target organism (configuration of our developmental-space), x, we may ask the minimum size of program necessary to specify it under our scheme:

$$eK_{AE}(x) = \min_{|g|}\{g \in G \mid g \vdash_{AE} x\} \tag{3.3}$$

An important point about eK, as opposed to the original formulation of Kolmogorov complexity is that eK need not depend on a Turing-machine[2]; Indeed, it need not depend on any particularly complex grammar at all[3]! Since we have a finite world imposed by the developmental (phenotypic) space, the usual trappings of uncomputability and unpredictability need not apply to eK[4]. Of course, the choice of embryogenic model becomes very important, as we no longer have any guarantee of one model implementing another, as we do with Turing machines.

Note further that we may also define environmental logical depth:

$$eD_{AE}(x) = \tau_{AE}(g*), \text{ where } g* \vdash_{AE} x \tag{3.4}$$

where $g*$ is a genome which minimizes eK_{AE}.

3.2.3 Functional Complexity

For the same reasons we reject Structural Complexity for AE, we must also reject the use of "parts as proxy" for Functional Complexity — the importance of the parts has been undercut by the fact that they have been purposefully included in the model in the artificial case. However, in AE unlike Biology, we typically have the advantage that we have a *raison d'être* for our model: either a fitness function we are seeking to optimize, or a set of goals we are hoping to reach through machine learning. Given this, we can attempt a more direct definition of Functional Complexity.

[2]Of course, one might argue that since a traditional definition of K uses a universal Turing-machine U, and since any developmental system can be implemented by U in constant time, that K is an appropriate measure. Here, two factors should be noted: firstly, that U might be capable of finding patterns faster than our developmental system by *not* implementing the developmental stage; and secondly, that since our developmental system need not be Turing-complete, that U might reach patterns that our system cannot.

[3]After all, we have yet to see a demonstration that Turing-completeness is required for a model of the development of living systems.

[4]Unfortunately, although eK is computable for most models of AE, it is not likely tractable for a substantial model. However, it is likely that a version of Universal Codes [GV06], adapted to the AE model in question, would be readily computable, allowing for good approximation in reasonable time. Or, perhaps simple random search might yield reasonable approximations, at least for the purposes of empirical evaluation of hypothesis.

Niche- and Funcplexity

Here we will define a set of boolean objectives, and measure complexity in terms of which boolean objectives an organism is capable of fullfilling. Let's consider some specific boolean objective, ϕ. An objective can be designed in an AE system in a variety of ways: by using the fitness function ("$\phi(x) \iff f(x) > 0.5$"), satisfying criteria in a multi-objective optimization, or meeting some specific task ("reproduce" or "survive for 100 time steps"), etc.

Then, we may consider the set of all organisms which meet objective ϕ:

$$S_\phi = \{\alpha \mid \alpha \vdash_{AE} x \ \wedge \ \phi(x)\} \tag{3.5}$$

Given two objectives, we may compute the set as the intersections: $S_{\{\phi,\psi\}} = S_\phi \cap S_\psi$. Hence, given some objective, we have a group of agents that (potentially) fill a niche. The complexity of the niche may be defined as:

$$\textit{nicheplexity}(\phi) = \min_{|\alpha|}\{\alpha \in S_\phi\} \tag{3.6}$$

We have chosen a measure ultimately linked to genomic length since we seek a measure of ease of discovery in the context of evolution; We may well have defined an alternative *nicheplexity* in terms of organismal complexity, say:

$$\textit{altnicheplexity}(\phi) = \min_{\textit{organplexity}(x)}\{x \mid \alpha \vdash_{AE} x \ \wedge \ \alpha \in S_\phi\}$$

Now, let us enumerate all objectives. Let us say that the set $O = \{o_1, o_2, ..., o_n\}$ is the set of all objectives which we are presently concerned with. Then, given some organism, x, we define $F_x = \{o \in O \mid o(x)\}$. F is a description of x's functionality relative to O.

Now, let's define the functional-complexity of x:

$$\textit{funcplexity}(x, O) = \textit{nicheplexity}(S_{F_x}) \tag{3.7}$$

So, relative to objectives O, *funcplexity* first asks "what does organism x accomplish?", then asks, "what is the simplest agent which accomplishes everything that x does, with respect to objectives O?", and finally returns the genomic complexity of that organism.

3.3 A Domain of Discourse: TCAs

Here we attempt to create a very simple model of Artificial Embryogenesis, for the purpose of concrete discussion. We will consider a simplified and finite version of Cellular Automata, the Terminating Cellular Automata (TCAs).

3.3.1 One-dimensional TCAs

We restrict our domain to finite one-dimensional terminating Cellular Automata, defined as follows: Let our developmental environment, $E \subset \mathbb{Z}$, be a one-dimensional toroidal line of cells of length l, let our alphabet be $\Sigma = \{0, 1\}$, and consider our system governed by a discrete time.

At initialization, or time $t = 0$, our environment begins with all states of colour "0", save a single central cell of colour "1". Our system terminates after a specified number of steps.

Then, any particular TCA, α, is specified by a triple (d, t, ϕ):

- A diameter, $0 < \alpha_d \leq l$.
- A running time, $0 \leq \alpha_t < 2^l$.
- A transition function, α_ϕ.

We will define the genomic size of any TCA to be its diameter:

$$|\alpha| = \alpha_d \tag{3.8}$$

A rule set may be defined by enumerating all possible neighbourhoods of length α_d and specifying an output from Σ for each[5].

We will write that a particular TCA, α produces pattern $p = (x_1, ..., x_l)$, or

$$\alpha \vdash_{TCA} p \tag{3.9}$$

if running our TCA α for α_t steps produces lattice configuration p. Given some arbitrary pattern p, we know there exists *some* TCA which produces it (see Appendix A).

3.3.2 Measures of Complexity on TCAs

In the case of TCAs, we may define the environmental-Kolmogorov complexity as:

$$eK_{TCA}(x) = \min_{\alpha_d}\{\alpha \mid \alpha \vdash_{TCA} x\} \tag{3.10}$$

where x is a binary string of length l. Similarly, we can define e-Logical Depth,

$$eD_{TCA}(x) = \alpha*_t, \text{ where } \alpha* \vdash_{TCA} x \tag{3.11}$$

where $\alpha*$ minimizes eK_{TCA}.

Given some set of objectives, $O = \{o_1, ..., 0_k\}$, we may define *niche* and *funcplexity*:

$$nicheplexity(O) = \min_{|\alpha|}\{\alpha \mid \alpha \vdash_{TCA} x \ \wedge \ o_1(x) \ \wedge \ ... \ \wedge \ o_k(x)\} \tag{3.12}$$

$$funcplexity(x, O) = nicheplexity(\{o \in O \mid o(x)\}) \tag{3.13}$$

3.4 Experiments with One-dimensional TCAs

We have implemented TCAs, and used them to contrast the notions of complexity introduced above. Our domain of discourse is the set of all binary strings of length l. For computational reasons, l is kept reasonably small, assuming the values $l = 5, 6, ..., 15$. For each pattern of

[5]Note: the Elementary Cellular Automata may be considered TCAs with $\Sigma = \{0, 1\}$, $\alpha_d = 3$, $l = \infty$ and $\alpha_t = \infty \ \forall \alpha$.

length l, we have measured complexity using: Shannon Entropy H (Eq. 2.1), Environmental-Logical Depth eD_{TCA} (Eq. 3.11), the LZW algorithm (Eq. 2.2), and Environmental-Kolmogorov Complexity eK_{TCA} (Eq. 3.10). In all cases, there were no patterns of eK greater than five[6].

Additionally, we defined a set of objectives. These were:

- o_{symm}: "x is symmetric", excluding the central cell for odd l.
- o_{ratio2}: "the number of 1's and 0's in x differs by no more than 2"
- o_{trans4}: "the number of transitions from 0 to 1 (reading from left to right) is no greater than 4"
- o_{right0}: "the rightmost cell has value 0"
- $o_{black80}$: "the ratio of 1s in x is more than 0.8"

These were divided into two sets, $O_1 = \{o_{symm}, o_{ratio2}, o_{trans4}\}$ and $O_2 = \{o_{symm}, o_{right0}, o_{black80}\}$. So, O_1 requires a symmetric, equally distributed pattern with few colour changes. O_2 requires a pattern which is mostly 1s, with a right-most value of 0, also symmetric. Using this set of objectives, we were able to compute the *nicheplexity* and *funcplexity* of the given patterns (Eq's 3.12 and 3.13).

Using the patterns and their associated values as data points, we have computed the statistical correlation between the pairs (graphed in Figure 3.2). Further, for many of the pairs, we have computed the lines of best fit, in order to show the directness of the relation between the data.

Note firstly that all measures are generally positively correlated; This is not surprising, since all measures reward very simple data low values, and all measures award random data with high values (save eD, the measure with the lowest correlation values). We will consider higher correlation values to indicate more agreement between measures (although that relation need not necessarily be direct).

The tendency of correlations involving H to oscillate is probably due to the difference between even and odd lengths; H is capable of reaching maximum value on even lengths, but not for odd lengths, since it is impossible to have equal proportion of cell types. This artifact would quickly become negligible for large l.

The measure eD is not well correlated to any of our other measures of complexity at all; This is not a drawback in and of itself, the computational complexity of a growing agent is quite a unique and important concept to AE, capturing the notion of difficulty of development. Unfortunately, eD did not correlate to our functional measures either, meaning that patterns with close functional value will not generally have close running times. Further, values of eD varied substantially for patterns of the same functional class which had the same measures of other complexity measures, such as eK. Hence, we find it unlikely that eD would be of practical use otherwise.

High correlation may be observed between H and *funcplexity*; This is due to the specification in the objective sets of particular proportions of cells of type 0 and 1, leading to particular values for H. Figure 3.3 compares the lines of best fit for data plots of H versus *funcplexity* for the $l = 12$ data; The relation for O_1 is far more direct, as to be expected from the criterion o_{ratio2}: "the number of 1's and 0's in x differs by no more than 2".

[6] At these small lengths, in turns out, there is no such things as a TCA-random string

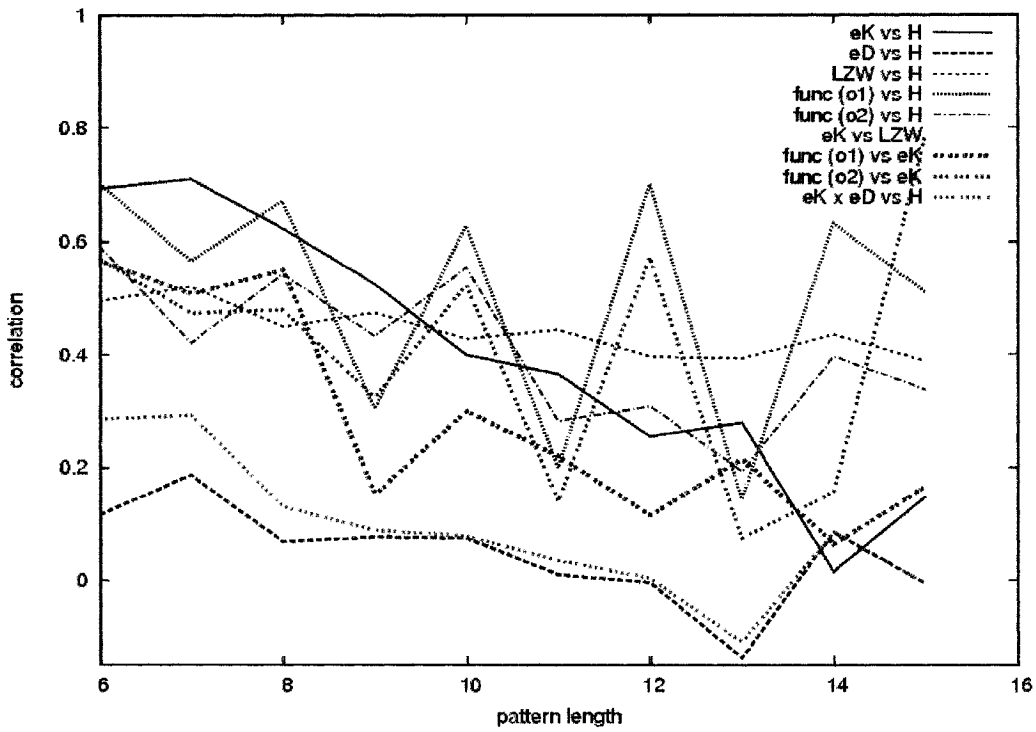


Figure 3.2: Correlations between complexity measures by pattern length.

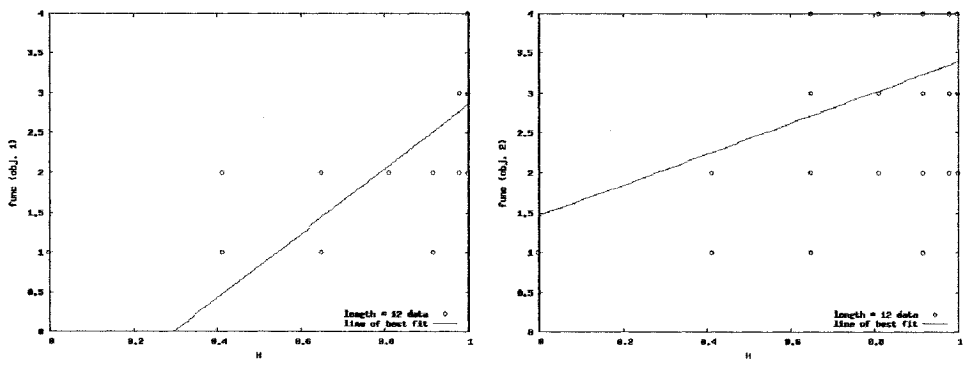


Figure 3.3: Lines of best fit for H and *funcplexity* for objectives O_1 (*left*) and O_2 (*right*), where $l = 12$.

Of course, there is no guarantee that patterns which satisfy some given set of objectives will have high complexity by any non-functional measure; For example, at $l = 10$, the number of patterns satisfying all criteria of set O_1 was 22, while there was only one pattern which satisfied all criteria of set O_2. All agents satisfying O_1 had an entropy of 0.971, due to specification. Values for eK and LZW were slightly higher than mean, but not significantly so. The agent satisfying O_2 in fact had low values for all fields, such as an $eK = 2$ and $H = 0.7219$, this since the pattern meeting the criterion was quite simple. A sample of the discussed patters are shown in Table 3.1.

Note further that the correlation between LZW and H is generally much higher than the correlation between either LZW and eK, or between H and eK; Lines of best fit are shown in Figure 3.4 for $l = 12$. This is not at all surprising, since both H and LZW are frequency-

Table 3.1: Examples of growth of patterns of length 10 which satisfy (*left*) objectives O_2 and (*right*) O_1, ordered by "complexity" (eK).

Pattern	*eK*	*Development*	*Pattern*	*eK*	*Development*
0111111110	2		0001111000	2	
			1110000111	2	
			0010110100	3	
			0011001100	3	
			0101001010	4	
			1010110101	4	

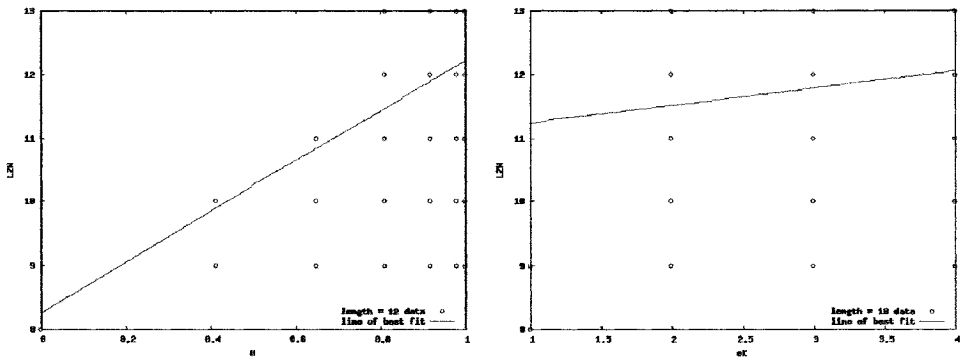


Figure 3.4: Lines of best fit for H and LZW (*left*), and LZW and eK (*right*), where $l = 12$.

based measures, while Kolmogorov complexity is far more general than that. We expect that most any compression-based mechanism will serve as a measure of entropy-plus, rather than approximating Kolmogorov complexity, this through any process less robust than Universal Codes. While compression algorithms may correlate to more difficult problems, and may be perfectly valid as a statistical estimate of problem complexity, there do exist forms of pattern formation which they will not measure, even in the simple case of TCAs. Indeed, the cases where the difference between eK and LZW were maximized involved the growth of simple TCAs ($eK = 2$) which produced patterns which would not compress well due to a lack of regularity; Two such TCAs are shown in Figure 3.5.

3.5 Conclusions

We have argued that it is not sufficient to use terms such as "complexity" or "complexification" naively; It is necessary to spell out what form of complexity one is interested in, even if the definition in the domain of application is vague. This, since the stated interests that exist in the literature are not complementary, potentially leading to unnecessary misunderstanding. We

Figure 3.5: Two TCAs with low eK generating relatively difficult to compress strings, where (*left three*) $l = 11$ and (*right four*) $l = 14$.

have discussed several possible notions of complexity in our review. A simple proposal was made for a naïve notion of organismal complexity.

We have further argued that Kolmogorov complexity in its classic definition is probably not desired; Instead, an environment-based Kolmogorov complexity is needed to discuss the "easiness" of specification of pattern formation in a given embryogenic context. From similar arguments, we further define some additional measures, eD, and functionally-defined measures, *niche-* and $funcplexity$.

Based on our initial experimentation with short binary strings and TCAs, we have contrasted our measures and derived some insight into their behaviour. Firstly, the various measures disagree on wide classes of data, showing clearly that they are not equivalent. Frequency-based measures, such as entropy and LZW compression do not agree well with Kolmogorov-like measures like eK, which do not correlate well with functional measures, such as $funcplexity$. There does not appear to be any immediately obvious "correct" notion of complexity — for now, a practitioner must simply choose a perspective based on intuition. This is likely to make evaluation of claims like the "edge of chaos" or complexification more difficult, as lacking a single accepted notion of complexity, the hypothesis lose falsifiability. To properly evaluate such claims, we must first find which class of measures, if any, has predictive value in the evolutionary process.

Further, we have determined several specific insights: (a) that environmental-Logical Depth does not display any particularly predictive properties, at least not with our simple example; (b) that our functional examples mapped to very specific values for other more general complexity measures, those values being dictated by the objectives which the patterns met. Perhaps this was due to the specific objectives chosen, but the example serves to highlight that high fitness agents need not necessarily correspond to high complexity by other more general measures at all! Finally, (c) that no other measure correlated well with environmental-Kolmogorov complexity.

It is our suspicion that measures based on minimal genomic length, like environmental-Kolmogorov complexity, will prove most useful to practitioners in AE, since they most readily capture the notion of how easily a string can be found via Machine Learning. Alas, although computable, eK is likely intractable for a problem of any significant phenotypic dimension. However, it is our hope that further research can find reasonable approximations, perhaps through statistical approximation or some implementation of an AE-based Universal Codes.

Chapter 4

The Model

4.1 Deva Overview

Deva (**Dev**elopmental **A**lgorithm) is a collection of algorithms[1] designed to implement artificial embryogeny; The goal of Deva is to exploit the pattern formation routinely found in natural embryogeny, balanced against the need for computational efficiency.

A Deva algorithm is a discrete dynamical system; It consists of a single "cell" developing through discrete time via the repeated application of a transition function. A Deva algorithm may be viewed as a metaphor for the embryogenesis of a multicellular organism from a single seed or zygote.

Initially, a single cell of type "neutral", with some given genome, is placed in the centre of a discrete developmental space; All other points in this space begin as "empty". Every non-empty cell in developmental space may be viewed as an independent agent — each makes decisions on the basis of its local neighbourhood and the genome. At every time step, a cell in the developmental space examines its neighbours, consults its genome and some internal variables, then executes an action. An action might be division (copying oneself to an adjacent point), specialization (changing the cell's type), or any of several others. Hence, entirely through cell actions, a developmental space changes in time — we will refer to this change as "growth" or "development". The algorithm terminates when a developmental space is identical to the space that preceded it immediately in time.

Much like a cellular automaton, a Deva algorithm uses only local information to implement a dynamic. If we fix our choice of cell types and developmental space, the dynamic depends only on the choice of genome (which defines the transition function). Typically, the number of transition functions is astronomically large, and the relation between genotype and developed agent wildly non-linear.

A particular example of Deva growth is shown in Figure 4.1, and a second with a different transition function in Figure 4.2.

4.1.1 Integral Components

The integral components of a Deva algorithm are:

[1] We will happily refer to such algorithms as "Deva algorithms" despite the redundancy.

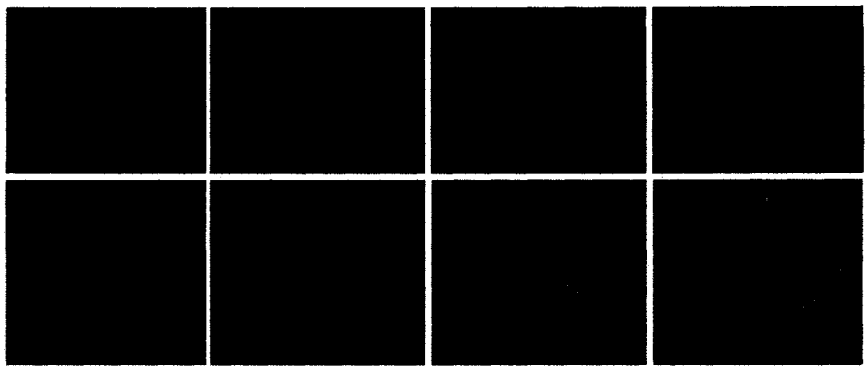


Figure 4.1: An example of growth in developmental space by a Deva 1 algorithm, where cell type is indicated by colour, black being empty. *(left to right, top row)*: time 0; 2; 4; 6; *(bottom row)*: time 10, 20, 30, 40.

Figure 4.2: A second example of deva growth. *(left to right)*: time 0; 10, 20, 30, 44 (termination).

- A *developmental space*, or a lattice of points with topology. Here, we will consider subsets of $\mathbb{Z}^2$, although application to other lattices should be easily defined. This space will change through discrete time.
- A finite collection of *cell types*; These are a finite number of component types which will make up the final organism — the cell types may also be defined through possible combinations of "genes" turned on or off[2].
- A *genome*, a genetic structure which encodes the *transition function*. The transition function is a function which maps from a neighbourhood of points to an action.
- A set of *actions*, describing what a cell may do in the following time step.

Below, we describe the genome and the actions in a Deva algorithm; The developmental space and cell types are a matter of perspective, defined instead by the application or interpretation in question (we define a particular application, Trusses, in Section 4.4).

4.1.2 Actions

Every cell in developmental space may be seen as an independent agent. Each is activated at every time step, possibly executing an action.

Each (non-empty) cell contains a set of internal variables, notably an *age*, and a resource counter r_c. A cell may only become active if its age has passed a minimum threshhold (i.e. a latency period for new cells), and if it has sufficient resources.

The following are a list of actions which *might* be included in a Deva algorithm:

[2]The "genes" we discuss here may be viewed sometimes as metaphors for genes proper, and sometimes as metaphors for expressed protein; Essentially, they may be viewed as (possibly empty) factors in a cell's specialization.

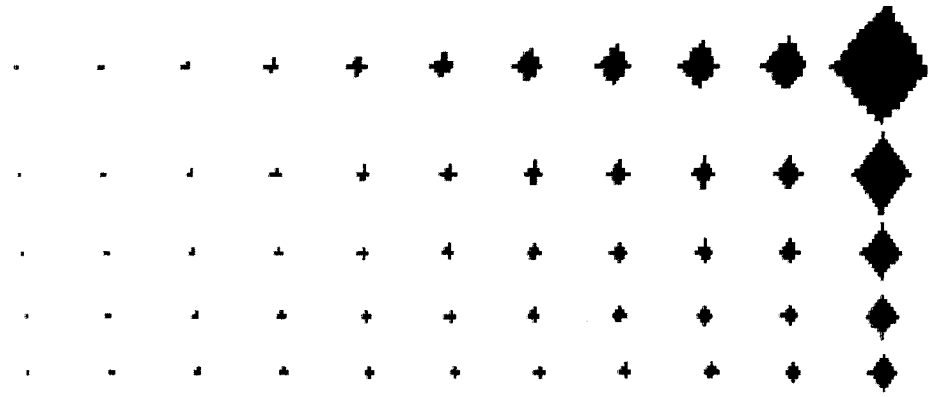


Figure 4.3: A series of growths of an organism whose transition function returns only the "divide" command, with rows corresponding to *age* = 0, 1, 2, 3, and 4, from top to bottom. The far left contains a single, active, aged cell, followed by nine successive time steps; Finally, the same organism an additional ten time steps later is shown.

- "nothing", the empty action.
- "die", the cessation of future functioning of the cell. This removes the cell, leaving an empty point in the developmental space.
- "divide", which makes a copy of the cell in a neighbouring location, usually the direction away from the existing centre of mass (the *best free location*). The cell usually inherits the colour of its parent.
- "specialize(colour)", which causes a cell to change specialization to the specified colour. The neutral colour is omitted.
- "elongate", which elongates the cell in the best free location, or in a specified direction.
- "on(gene)", which activates a particular gene.
- "off(gene)", which deactivates a particular gene.

Any cell action that creates, moves or specializes a cell will cause the resource counter r_c to be appropriately decremented; This is intended to encapsulate the materials required to implement the cell action in the chosen domain of application. The decrement is performed prior to any division, hence affecting both the source and the copied cell.

Figure 4.3 shows the growth of an organism whose transition function returns only the "divide" command, using the *best free location* algorithm (see App. B.1.2), with a sufficiently large value of r_c. The only varying parameter is the *age* parameter, shown on the rows of the figure. As can be seen, the initial values vary somewhat, but the later values, *age* = 3 and 4, create a series of perfect diamonds, *age* = 4 doing so slightly more slowly than *age* = 3. It is expected that larger values will also create a series of perfect diamonds at successively slower rates.

4.1.3 Transition Functions

A Deva transition function may be viewed as an approximation of the more typical transition function used for other sorts of automata. In a cellular automaton, for example, a transition function is usually defined for each and every possible neighbourhood of a cell; This leads to

a representation length of k^r, where k is the number of cell types, and r is the size of the neighbourhood. Of course, for large k or r this quickly becomes intractable.

A deva transition function, ϕ, is a function which maps from a description of a neighbourhood to an action, i.e.,

$$\phi : S \rightarrow A$$

where S is the set of all neighbourhood descriptions, and A is the action list associated with the particular deva model. Any such function could, in principle, be used. The genome of any agent is simply a representation of this function.

The majority, although not all, of our deva models will be rule-based models, meaning that the transition function may be written as a series of pattern-matching rules. The rules are of the form (s, a), where s is a description of a possible state of a neighbourhood, and a is an action:

$$s \rightarrow a$$

i.e., that matching state description s leads to action a. Hence, ϕ may be thought of either as a function, or as a listing of rules. Given some current cell c with neighbourhood μ_c, we interpret the rules as a function according to the following algorithm:

Neighbourhood description $s_c \leftarrow \textit{description}(c, \mu_c)$
Rule $r_{min} = (s_{min}, a_{min}) \leftarrow (\emptyset, \emptyset)$
double $min \leftarrow \infty$
for all $r_i = (s_i, a_i), r_i \in \phi$ **do**
  if $\textit{distance}(s_c, s_i) < min$ **then**
    $r_{min} \leftarrow r_i$
    $min \leftarrow \textit{distance}(s_c, s_i)$
  end if
end for
return a_{min}

This function is made precise through a definition of the neighbourhood state description, and a notion of distance between these descriptions. Assuming that examining and constructing a neighbourhood description may be accomplished in constant time, the time necessary for obtaining a value from the transition function is $O(|\phi|)$.

4.1.4 Deva Growth

Given definitions of neighbourhood state descriptions (and distances between them) and a set of actions, we may define a deva growth algorithm. Growth proceeds according to the following process:

Time $t \leftarrow 0$
Initialize developmental space D_t
while $D_t \neq D_{t-1}$ **do**
  $t \leftarrow t + 1$
  $D_t \leftarrow D_{t-1}$
  for all Cell $c \in D_{t-1}$ **do**

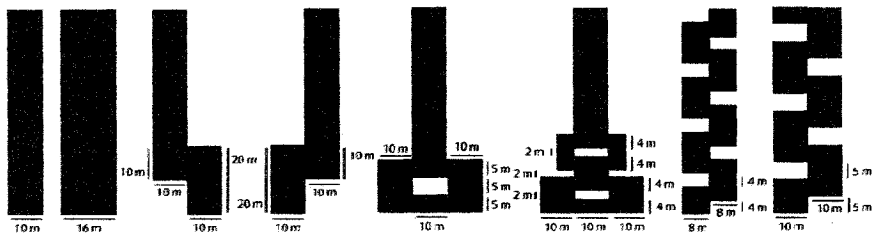


Figure 4.4: Environments *(left to right)*: thin, thick, swerve-l, swerve-r, bulb, bulbs, zigzag-s, zigzag-l.

```
        if c has sufficient age and c_{r_c} then
          Action a ← φ(μ_c)
          Decrement c_{r_c} appropriately for a
          Execute a in D_t
        end if
      end for
    end while
```

A more explicit version may be found in Algorithm 1 in Appendix B.1.3.

Termination of the deva growth algorithm is guaranteed by the limitations of a finite resource counter, r_c; The nature of cell division and the square lattice guarantees that a grown agent will fit in a diamond of size $(2r_c+1) \times (2r_c+1)$. Given this limitation, we have a worst-case running time of $O(r_c^2)$ in terms of cell operations; Given that each cell operation requires a query to the transition function, our total running time for development is bounded by $O(r_c^2 \cdot |\phi|)$, which, since $|\phi|$ is a constant system parameter, is $O(r_c^2)$.

4.1.5 Developmental Environments

In most experiments, we use a simple environment where the initial cell begins in a central location, located at the bottom of the space. However, additional constraints may be placed on development, and hence on evolution, through the choice of developmental space for growth. Here we outline a few possible variants, used in later experiments.

Each environment is a connected subset of $\mathbb{Z}^2$, where the initial cell is placed in the lowest central location of the space. We use several environments for growth (with lengths specified in meters, for reasons soon to be clear):

- *thin* and *thick*, environments of width 10 m and 16 m, respectively, each pointing straight up.
- *swerve-l* and *swerve-r*, environments of width 10 m which, at height 10 m, swerve left or right, continuing straight up indefinitely.
- *zigzag-l* and *zigzag-s*, a zig-zagging line of width 10 m and 8 m, with corners at heights of multiples of 5 m and 4 m, respectively ("l" and "s" for large and small).
- *bulb* and *bulbs*, environments composed of one or two bulbs, respectively, followed by a shaft of width 10 and unlimited height.
- *normal*, an environment large enough to contain growth.

An illustration of the environments may be seen in Figure 4.4.

These environments are created by filling a sufficiently large space with "barrier" cells, that is, cells which are neither empty, nor active. These cells do not contribute hormones to the description of neighbourhoods.

4.2 A Variety of Deva Models

Here we loosely describe our various Deva models. The interested reader is urged to consult Appendix B for specifics. The various *Deva* models we will discuss are summarized in Table 4.1.

4.2.1 Deva 1 Models

The deva implementation *Deva 1* is our original Deva implementation, and is the model most often used as a control for future experiments; As such, we will describe it here in greater detail than the other algorithms.

Deva 1 makes liberal use of the best free location algorithm (See Appendix B.1.2). The set of actions includes is the set $A = \{die, divide, elongate, specialize(2), ..., specialize(n_c)\}$. Hence, $|A| = 2 + n_c$.

A *Deva 1* transition function is a listing of descriptions of possible neighbourhoods of a specified length, $|\phi|$. These rules are tuples of the form:

$$(c, h_1, ..., h_{n_c}, a)$$

where c is a colour, a an action, and h_i is a count of the number of neighbours of cell type i, or a *hormone-level.* The size of such a transition function is hence $|\phi| \cdot (n_c + 2)$ integers, and the total number of such transition functions is $|\phi| \cdot n_c \cdot 12^{n_c+1} \cdot |A|$.

Any particular cell may consult its transition function to determine an action appropriate to its neighbourhood. By consulting its neighbourhood, it may construct n_c *environmental-hormone-readings,* or a count of the number of cells of type i in its twelve-neighbourhood (relative to the source cell); These will be written $\{e_1, ..., e_{n_c}\}$; Next we find the rule in our transition function which closely matches the set of e_i to the set of h_i using Euclidean distance (See Algorithm 2 in Appendix B).

We would like a means of initializing a set of rules such that hormone values in the rules are likely to match possible neighbourhoods in developmental space. As the integer values may range between 0 and 12, uniform random selection will likely result in an impossible[3] pattern (i.e. one in which $\sum h_i > 12$, in two-dimensions). Hence, we desire a power-law distribution which favours 0:

$$Pr[X = i \mid 0 \leq i \leq 12] = \frac{1}{\sum_{j=0}^{12} \beta^j} \beta^{12-i}$$

We would like the expected value of the generation of any particular rule to be somewhere in the range of $\{0, .., 12\}$, where selecting a larger value is preferable, since the majority of computation will occur in a full neighbourhood; Hence, we take $E[n_c \cdot X] \approx 12$. Solving (numerically) the

[3]Impossible for exact matching, that is, and less likely for more reasonable distributions for closest distance-matching.

equation

$$\frac{12}{n_c} = \frac{1}{\sum_{j=0}^{12} \beta^j} \sum_{i=0}^{12} i \cdot \beta^{(12-j)}$$

we obtain $\beta \approx 3.6$ for $n_c = 32$ and $\beta \approx 6.3$ for $n_c = 64$

We also require a probability distribution for possible actions:

$$Pr[Y = a] = \begin{cases} 1/4, & \text{if } a = \text{“divide”} \\ 1/4, & \text{if } a = \text{“die”} \\ 1/4, & \text{if } a = \text{“elongate”} \\ 1/4, & \text{if } a = \text{“specialize”} \end{cases}$$

where all colours of specialization are equally likely, save “0” or “1”, which are excluded. If the “move” action is included in the run, it simply replaces then “elongate” action.

Hence, we may generate a random rule by (uniformly) randomly generating an initial rule colour, then generating n_c hormones and one action according to the above distributions. The same distributions may be used for the genetic operator mutation.

4.2.2 Deva 1 Variants

As *Deva 1* was the first implemented algorithm, it was chosen most often for comparison; For this reason, there are many variants of this particular model used for our comparative study.

The ***Deva 1.N*** Model is the same as the *Deva 1* model with the addition of an explicit “nothing” command. This was added after initial experiments with *Deva 1* showed the model using repeated elongation as a means of preventing growth in a region of developmental space.

There were several models designed to test the efficacy of various alternatives (to the *best free location*) abstractions of cell division: The ***Deva 1.Dir*** model replaces the divide comand with “divide(d)”, where the direction d is explicitly specified genetically; The ***Deva 1.Allway*** model causes a cell to divide in all directions simultaneously, thus influencing a quasi-symmetric morphology; The ***Deva 1.Clockwise*** model specifies the location of division to simply be the next free location clockwise from the left.

The transition function of the *Deva 1* Model specified that in order to be considered, the rule in question had to match the colour of the local cell; This caused a partitioning of the genome, so that different cell types would query different subsets of ϕ. The ***Deva 1.Stateless*** Model changed the form of the transition function, removing the specification of the state of the querying cell, hence removing the state-based division of the genome. This option is included for study as many existing models of AE in the literature both include or exclude such a mechanism in the decision making process, seemingly arbitrarily. For instance, many morphogenic models reason simply on the basis of morphogens, leaving the decision making process blind to a cell's current state; This is true of Turing's original differential model [Tur52], and is true of most similar systems (e.g. the Cellular Potts model [MG05]); Grammatical approaches, on the other hand, include a cell's current state as an integral part of the pattern matching mechanism, as in most L-Systems models of plant development [PL90].

There also exists a version of *Deva 1*, named ***Deva 1.NoInher***, in which the “division”

action causes the creation of a cloned cell in the *best free location* of type "neutral"; This lack of cell inheritance is explored as it has been postulated as a positive mechanism of canalization of the search space: Furusawa lists it as a key factor in the development of hierarchy and stability in differentiation [FK98]; Indeed, this author posited inheritance of cell specialization as a key factor in the improved performance of a developmental model for the growth of artificial agents over a direct encoding [KGK04].

4.2.3 Deva 2 Models

The ***Deva 2*** model was an attempt at decreasing the information present in the description of a neighbourhood; Whereas *Deva 1* assumed a cell type-based view of a neighbourhood, *Deva 2* regards cells as collections of genes, which may or may not be activated.

A *Deva 2* transition function consist of a series of rules of the form $(c, h_0, ..., h_{n_g}, n_j, a)$, where h_i is the number of cells with gene i activated, and n_j is the number of cells with a joint. The size of such a transition function is hence $|\phi| \cdot (n_g + 3)$ integers, and the total number of such transition functions is $|\phi| \cdot n_c \cdot 12^{n_g+1} \cdot |A|$.

Hence, given a neighbourhood, rather than count the occurrences of a given specialization, we count the number of cells with a given gene activated; This decreases the number of integers required to specify a neighbourhood from n_c to n_g (recall that $n_g \cong \log_2 n_c$). Since the transition function consists of pairs (s, a) of descriptions of a state s (n_g integers), and actions a (one integer), we shorten the length of the genome substantially. However, in the process, we throw away all information regarding which genes are activated in conjunction.

We have less difficulty with initialization for *Deva 2* than we did for *Deva 1*, since all possible hormone-descriptions are potential cells. However, in practice, completely maximized neighbourhoods are rare, and not useful to our particular domain of application. Hence, we use a power-law distribution for initialization of *Deva 2* as well, but one that favours half-full neighbourhoods (see Appendix B.4 for details).

The ***Deva 2.Stateless*** model is precisely the same as the *Deva 2* model, but with states removed, as in *Deva 1.Stateless*. The ***Deva 2.NoInher*** model is also like the *Deva 2* model, but without inheritance of cell specialization.

4.2.4 The Deva 3 Model

The ***Deva 3*** model was chosen so as to model Cellular Automata more closely. In this scenario, we do not approximate the description of a neighbourhood, but instead provide a complete description to the transition function. Hence, the *Deva 3* model provides a greater amount of information regarding the extended Moore neighbourhood which surrounds a cell than did *Deva 1* or *2*.

The transition function consists of a series of lossless neighbourhood descriptions, meaning that each site in the neighbourhood is specified precisely. Since any complete such description of neighbourhoods would require a staggering amount of information (32^{12} integers), any particular transition function consists of exemplar patterns which approximately describe any particular cell's current neighbourhood. A Manhattan metric was defined on these descriptions, used to choose appropriate rules. There was no need to explore the use of states, since the precise

neighbourhood description is capable of implementing the states naturally.

The initialization of a *Deva 3* transition function is far simpler than the other models, since a uniform selection of values is guaranteed to describe a possible neighbourhood; Hence, simple uniformly random initialization is used.

4.2.5 Deva 4 Models

The *Deva 4* models were designed to minimize the specificity of the action set associated with our developmental models. Instead of abrupt changes in cell specialization, we attempt a more continuous version here, where cell specializations may be viewed as the cumulative action of several genes.

Rather than include a "specialize" command for transforming cell types, we include commands for activating and deactivating genes, "on" and "off". The cost of activating a gene is defined by the domain of application, while the deactivation is free (assuming that the genes specify some constructive property). Hence, our action list is $A = \{die, divide, on(g_0), ..., on(g_{n_g}), off(g_0), ..., off(g_{n_g})\}$.

There are two versions of *Deva 4*: ***4.GeneType*** and ***4.CellType***. The first uses the transition function of *Deva 2*, hence describing neighbourhoods as collections of gene types; These descriptions match the language of the actions. The second uses the transition function of *Deva 1*, describing neighbourhoods as collections of colours.

Both are initialized utilizing the distributions associated with the transitions functions' home models, save for the distribution specifying the actions. Just as *Deva 2* was a simplification of the description of neighbourhoods, the *Deva 4* algorithms may be viewed as a simplification of the specification of cell actions.

Table 4.1: A summary of the various *Deva* models by choice of location, mode of neighbourhood description, action list, and means of decision making.

Model	Location	Nbhd. Desc.	Actions	Decision
Deva 1	Best free loc.	Count of cell types.	"divide", "specialize", "elongate", "die"	Rule-based with states.
Deva 1.N	Best free loc.	Count of cell types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 1.Stateless	Best free loc.	Count of cell types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based.
Deva 1.Dir	Specified explicitly.	Count of cell types.	"divide(direction)", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 1.Clockwise	Clockwise.	Count of cell types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 1.Allway	All directions simultaneously.	Count of cell types.	"divide", "specialize", "elongate(UP)", "die", "nothing"	Rule-based with states.
Deva 1.NoInher	Best free loc.	Count of cell types.	"divide without inheritance", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 2	Best free loc.	Count of genes types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 2.Stateless	Best free loc.	Count of gene types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based.
Deva 2	Best free loc.	Count of genes types.	"divide without inheritance", "specialize", "elongate", "die", "nothing"	Rule-based with states.
Deva 3	Best free loc.	List of cell types.	"divide", "specialize", "elongate", "die", "nothing"	Rule-based.
Deva 4.CellType	Best free loc.	Count of cell types.	"divide", "on", "off", "elongate", "die", "nothing"	Rule-based.
Deva 4.GeneType	Best free loc.	Count of gene types.	"divide", "on", "off", "elongate", "die", "nothing"	Rule-based.

4.3 Evolution

The Deva framework is driven through Evolutionary Computation. Evolution acts on a set of genomes (transition functions), and evaluates grown organisms. Excluding the inclusion of a mid-step between genotype and phenotype, we use a typical Genetic Algorithm, as described by Eiben and Smith [ES03].

A population of agents is created, each represented by a transition function. The genomes are simply a list of integers consisting of the rules of the transition function spelled out. Initial values for the first population are specified by a distribution, defined separately for each Deva algorithm. There exists a boolean "useSeed" option by which the first rule in an initialized genome is a "divide" action designed to match the initial conditions of growth.

Genetic operators are the same for all Deva algorithms[4]. Crossover is always pairwise, static length and single point, swapping tail ends of genomes for two agents. Mutation comes in two possible forms[5] (both point-mutation), controlled by a system parameter: *power-mutation*, which selects a new integer using the same distribution used for initialization; and *copy-mutation*, which selects an integer uniformly from elsewhere in the genome, substituting that value[6].

We use tournament selection by default, using a tournament size of 5 agents regardless of population size, and a tournament probability of $p = 0.7$. This form of selection was chosen in order to encourage diversity. As will soon become clear, convergence is difficult to recognize, so trials were run for a fixed number of generations. Additionally, the initial population size was larger than the population size for successive generations, by a factor of $initMult$.

4.4 Application: Plane Trusses

As previously described, Plane Trusses may be viewed as a simple and abstract way of capturing the notion of structure — indeed, trusses form the basis for most modern construction. Our overall goal here is two-fold:

- To demonstrate that a Deva algorithm is capable of designing non-trivial structures
- To provide an independent and external notion of "meaning" to organisms grown through Deva algorithms

4.4.1 Representation and Cell Types

We begin by defining a representation of general plane trusses constructively on a lattice. We aim to be able to map any lattice of integers to some plane truss, allowing for trivial, unconnected and otherwise useless truss designs.

Firstly, we define a set of cell types — each non-empty cell will contain a joint, and between zero and five beams. Beams will be assumed to be cylindrical and made of steel, although this

[4]This for analytic convenience only, it is almost certainly possible to tailor EC strategies to aid the particular algorithms, but this is not our current interest.

[5]We also experimented with a *uniform-mutation* operator which simply chose a value uniformly from the possible range — this mutation was dropped after poor experimental results, probably for the reasons described in the initialization section.

[6]The copy-mutation operator is investigated due to recent suggestion that it aids in moving the population towards an appropriate distribution of genomic values (as well as simple biological precedence) [MH05].

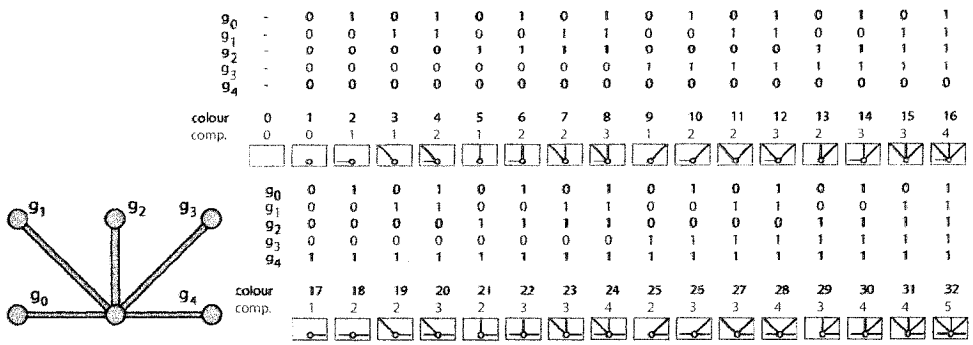


Figure 4.5: (*left*) Listing of the cell colours and their mapping to genes; (*right*) An illustration of the four genes.

choice is rather arbitrary, and could be replaced with minimal effort. The beams will extend in directions π, $3\pi/4$, $\pi/2$, $\pi/4$ and 0, labelled g_0 through g_4 respectively; Hence, a non-empty cell may be labelled by the presence or absence of these genes. This leads to 33 cell types, including the empty cell, shown in figure 4.5. We may refer to any cell in the lattice either through a series of boolean gene values, or as an integer in $\{0, ..., 32\}$. Conversion from boolean gene values to an integer is accomplished through the following equation:

$$colour = 2^4 g_4 + 2^3 g_3 + 2^2 g_2 + 2^1 g_1 + 2^0 g_0 + 1$$

The zero cell type is reserved for the empty cell, while the one value is for a joint with no beams[7]. The number of cell types, or colours, n_c is

$$n_c = 2^{n_g} + 1$$

where n_g is the number of genes, $n_g \geq 5$.

Finally, we may allow cells to be elongated in either the x or y directions by an arbitrary number of cell lengths. So, a cell of type 1 has an angle of $3\pi/4$ with the x-axis, and a length of $\sqrt{2}$; A single elongation in the y-direction would lead to a length of $\sqrt{5}$, and an angle of $7\pi/8$ with the x-axis. In the case of elongation, we refer to the originating cell as the "source" cell — this is the elonga*ted* cell, as opposed to the elonga*tions*[8]. A unit cell length will be mapped to one meter, although this choice is also quite arbitrary at this stage.

Hence, excluding elongation, any two-dimensional lattice of integers may be mapped to (some) truss[9]. One such mapping, including elongations, is shown in Figure 4.6.

An example of deva growth, interpreted as a plane truss with elongation, is shown in Figure 4.7.

[7]It is also possible to add additional *hidden genes* — these are genes which are not expressed in the phenotype, perhaps allowing the developmental process additional "memory". Of course, this increases the number of cells in general.

[8]As elongation extends in a specified direction, the source cell may always be recognized as the first cell in the list, for instance, the lowest cell in a line of cells elongated up.

[9]Once elongation is included, we must make some constraints on the relative location of the elongation integers — this is, however, unimportant for our purposes, as deva growth would not allow for illegal combinations.

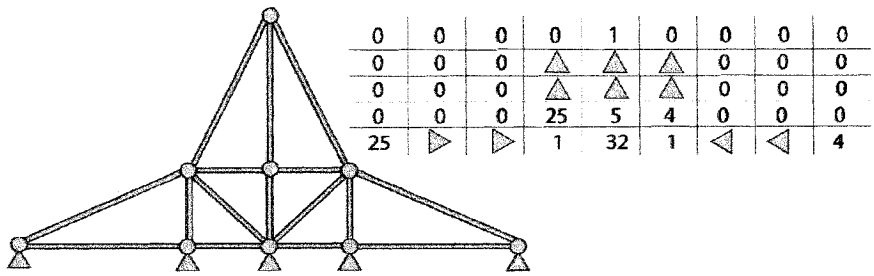


Figure 4.6: Example of a translation between a lattice of integers and a plane truss. The triangles are interpreted as elongations of the adjacent cells.

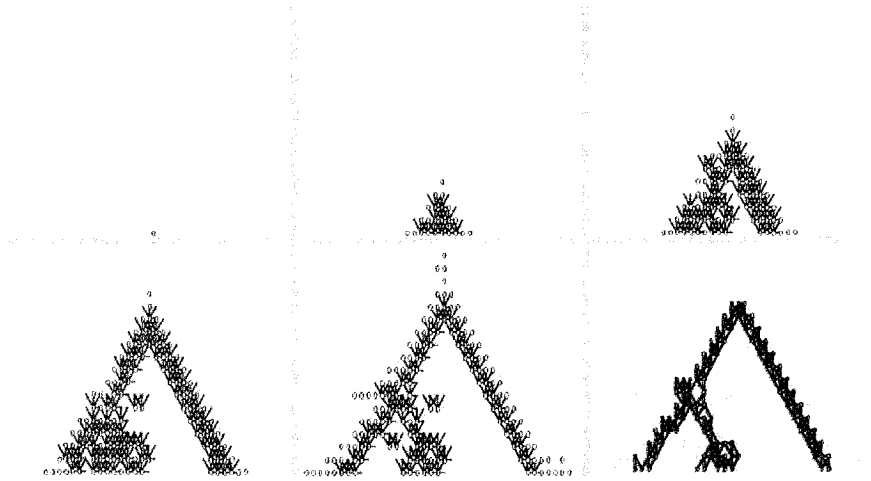


Figure 4.7: An example of growth in developmental space by a Deva 1 algorithm, under the plane truss interpretation *(from left to right, top to bottom)*: time 0; time 10; time 20; time 30, time 40, the final trimmed agent at time 45.

4.4.2 Trimming Grown Plane Trusses

A plane truss grown through a Deva algorithm comes with no guarantee of being well designed, in general. In Figure 4.7, the growth of an agent is shown (in grey), whereas the final organism (in black) is smaller — this is the result of a trimming process, applied to every organism following development. The trimming process serves to: (a) remove obviously unstable sections, such as beams which do not connect to joints at both ends; (b) to remove sections which are not connected to the base of the structure; and (c) to remove redundant joints, replacing them with longer beams.

The trimming process operates recursively, beginning with the base of the structure (hence, unconnected portions of the truss are automatically excluded). First, obviously unstable structures are trimmed: At every joint, it removes any beams with no second connecting joint; Next, if there is only one single beam connected to it, it removes itself. Crawling back from the extremities to the base, any "arms" (collections of beams and joints that do not form cyclic structures) are removed. Once the obviously unstable "arms" are removed, the second part of trimming is activated, removing redundant joints. The trimming process crawls through the remaining structure, and examines every joint; If the joint has precisely two beams, and those beams have the same angle, the beams and joint are removed, replaced with a new beam of length equal to the sum of the two removed (So, for example, a straight line of beams connected by joints would

be replaced with a single long beam). As each of the two processes requires a single recursive walk through the truss graph, running time for both is bounded by $O(n)$, where n is the number of beams in the grown truss. Note that there is no guarantee of stability at this stage, simply an improved probability thereof.

4.4.3 Plane Truss Evaluation

The evolution of Plane Trusses may be viewed as a multi-objective evaluation; As such, there are many factors involved in the computation of fitness. The factors include selection for non-triviality (t), height (h), minimal use of materials (m), stability (s), and pressure distribution (p). Each is described below for a general plane truss T.

A truss T is trivial if it contains less than three beams[10]:

$$t(T) = \begin{cases} \frac{1}{2}, & T \text{ is trivial} \\ 1, & \text{ow} \end{cases}$$

The height factor, $h \in [0,1]$, varies linearly between 0 for no height and 1 for the maximum height; This is defined by $h(T) = \frac{h}{r_c+1}$, where h is the raw height of T.

Material use, $m \in [0,1]$, varies linearly between 0 for maximal use of materials, $(r_c+1)^2$ in beam length, and 1 for no use of materials.

The base factor, $b \in [1/2,1]$, is designed to be minimized when there are few base joints in the centre of the space. We define the base count, b' to be the sum of the distance of the existing base joints from the centre,

$$b' = \sum_{i=-r_c}^{r_c} \begin{cases} |i| & \text{there exists a joint at location } i \\ 0 & ow. \end{cases}$$

Then the base factor may be defined as

$$b = \frac{1}{2} + \frac{1}{2}\left(\frac{\sum_{i=-r_c}^{r_c}|i| - b'}{\sum_{i=-r_c}^{r_c}|i|}\right)$$

A truss is stable if it passes the stability test. In this test, a stress analysis is performed on the truss. Initially, some trusses are declared unstable via a simple counting test — failing this, we compute the stiffness matrix, and we declare a truss stable if the stiffness matrix is invertible (see Appendix C.1), and if the maximum displacement is not absurdly high[11]. The stability criterion is then defined as

$$s(T) = \begin{cases} 1, & T \text{ is stable} \\ \frac{1}{4}, & \text{ow} \end{cases}$$

Given a stable truss, we may apply an external force to determine the stress on the member beams and the displacement of the joints. We aim to describe a pressure distribution factor,

[10]Note: a different definition of "trivial" was used in the initial Deva 1 trials, where a truss was trivial if it contained less that five cells or three non-empty cell types.

[11]Where absurdity kicks in at ten meters or more; This is necessary as the equilibrium process may sometimes find stable points through profoundly unrealistic stretching of materials.

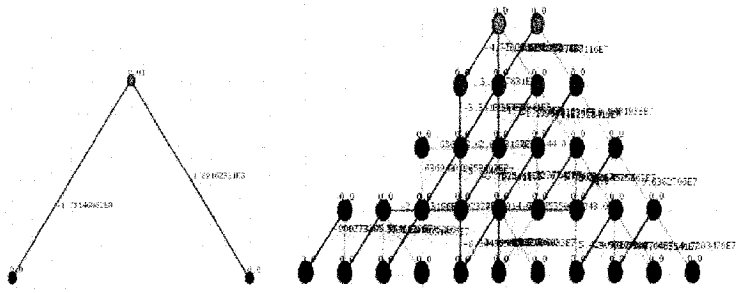


Figure 4.8: A screen shot of the stress analysis of two trusses under f_1 — compressed beams are green, stretched beams blue, beams beyond the yield of steel are red. The simpler truss on the left is both stable, tall, and uses little material. However, it is incapable of supporting the load. The truss on the right uses additional materials, but successfully distributes the load between two joints, scoring a higher overall fitness.

$p \in [1/2, 1]$. External forces are applied to the truss, and the stress in each member beam and dislocation at every joint are computed. The external forces are as follows: At every joint, 50 N down and 50 N left are applied, exceeding expected force of gravity and mild wind[12]. Depending on the fitness function, we apply additional external forces to particular joints, which far exceed the effects of the above forces. If the maximum member-force leads to more than $\pm$ 165 MPa of pressure[13], or if any joint is displaced farther than one meter, we return 1/2. Otherwise, we let the maximum pressure (in absolute value) be m', and set

$$p(T) = \frac{1}{2} + \frac{1}{2}\frac{(165 \text{ MPa} - |m'|)}{165 \text{ MPa}}$$

We may now define several fitness functions. Our first, f_{mat}, is designed to maximize height while minimizing overall material. We evaluate pressure by applying 20 kN down and 5 kN right[14] at the highest joint; In the case of several joints, the force is divided evenly between the joints. Hence, we seek a tall, minimal structure, capable of supporting a large mass at the top, much like a tower supporting some additional structure at the peak. The fitness of a truss T is defined as

$$f_{mat}(T) = t(T) \cdot h(T) \cdot m(T) \cdot s(T) \cdot p(T) \tag{4.1}$$

Our second fitness function, f_{stoch}, is similar to f_{mat} in all ways except that rather than apply external forces at the highest joint, we apply them randomly. Hence, three non-base joints are selected at random, and at each we apply 5 kN down and 500 N either right or left (with equal probability at each joint). Hence, $f_{stoch} = f_{mat}$ in definition, but f_{stoch} is a stochastic function.

Our third fitness function, f_{base} is again similar to f_{mat} in that again, we apply 20 kN down and 5 kN right divided between the highest joints. However, we remove the length minimization factor and instead include the base minimization factor — hence, f_{base} selects for tall trusses

[12]The density of steel is approximately 77 000 $\frac{\text{N}}{\text{m}^3}$ [Meg05], giving a truss of area 7.85×10^{-5} m^2 and length 1 m a weight of approximately 6 N; Hence, our trials far exceed the effects of gravity. Proper testing for wind exceeds the scope of this project as it requires dynamic analysis, although 50 N is considered generous for a static perspective.

[13]165 MPa is approximately 80% of the yield limit for structural steel [Meg05])

[14]20 kN is approximately the force exerted by 2 000 kg of mass — on a single beam it exerts approximately 255 MPa of pressure, more than the yield of steel.

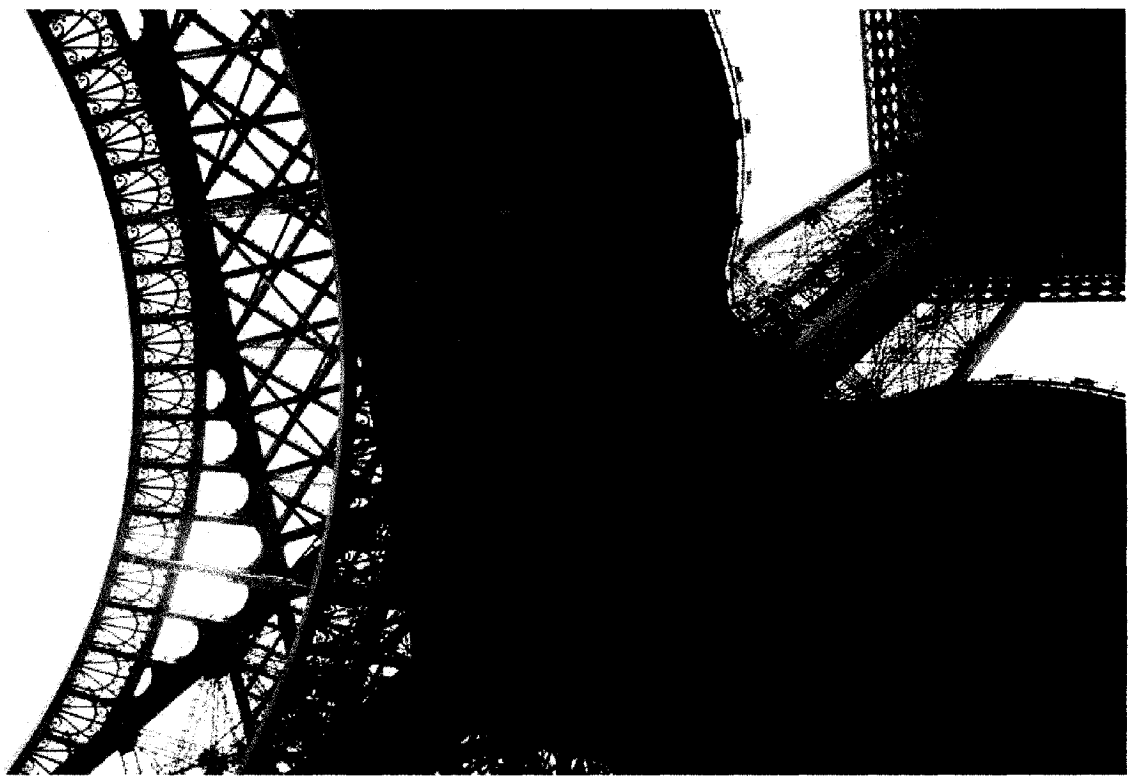
capable of supporting a load at the peak occupying as little ground space as possible:

$$f_{base}(T) = t(T) \cdot h(T) \cdot b(T) \cdot s(T) \cdot p(T) \tag{4.2}$$

Our next fitness function, f_{light} is also similar to f_{mat}, save that less external load is applied; This fitness function is used for scenarios where additional constraints are placed on the designs. At every joint, we apply 10 N down and 10 N left, simulating gravity and a mild horizontal force. Additionally, we apply 2500 N down and 500 N right at the highest joint; In the case of several joints, the force is divided evenly between them. The fitness of a truss T is thus defined as

$$f_{light}(T) = h(T) \cdot m(T) \cdot s(T) \cdot p(T) \tag{4.3}$$

Since each fitness calculation is dominated by a matrix inversion (see Appendix C), running time is $O(n^3)$, where n is the number of truss members. This is improved slightly by using an LU-Decomposition.



Chapter 5

Deva 1 Experiments

In this chapter, we discuss our initial attempts to evolve Plane Trusses via our first Deva algorithm, *Deva 1*. We explore the use of several different fitness functions and discuss the discovered designs for each.

5.1 Experimental Set-up

Many trials of many different sizes (values of r_c) were run. We chose to investigate the use of different fitness functions (the Fitness Trials). Data was collected for runs of size $r_c = 16, 24$; In general, we would have preferred larger runs as there is suggestion that AE approaches are particularly suited for problems with a large number of components [HM06], [KGK04], but the computational expense of the repeated runs necessary for establishing a trend are great. Some runs with different values of r_c were conducted, including $r_c = 40$, the last of these having required more than one week of time on a dual-processor 2.4GHz machine.

Parameter settings for the Fitness Trials runs are:

population size	200	init. pop. size	2000
prob. crossover	0.8	rate. elitism	0.01
prob. copy-mut.	0.05	prob. power-mut.	0.05
useSeed	true	$\|\phi\|$	100

There were ten runs of the Fitness Trials for each fitness and r_c setting, $r_c = 16, 24$. These runs are referred to as $fit.x.y.z$, where x is a fitness function, y is an r_c value, and $z \in \{0, .., 9\}$ is an index. Hence, $fit.stoch.24.3$ is the third run of the $r_c = 24$ trial using the fitness function f_{stoch}.

5.2 Data and Analysis

In all trials, successful trusses were evolved — all runs found stable trusses, and 56 of the 60 found trusses capable of supporting the applied external forces[1]. In general, heights of approximately 9m were found in the $r_c = 16$ trials, and heights of 18m when $r_c = 24$.

[1] At least, in the case of the f_{stoch} trials, capable of supporting the forces as applied in the 100th generation.

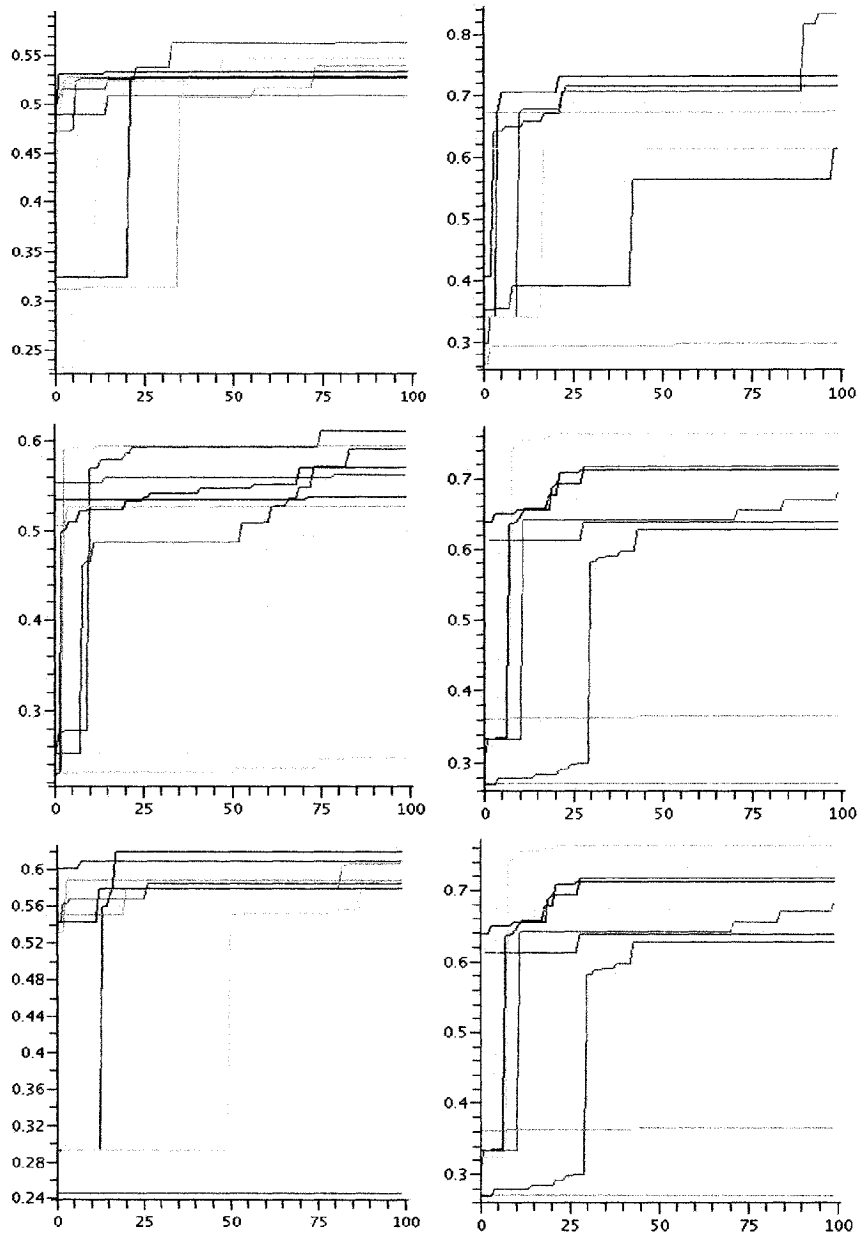


Figure 5.1: Plots of fitness (y) versus generations (x). Maximum fitness of each of the (*top to bottom*) *fit.mat*.16 and *fit.mat*.24; *fit.stoch*.16 and *fit.stoch*.24; *fit.base*.16 and *fit.base*.24 runs.

The maximum fitness of agents in the $r_c = 16$ Fitness Trials are graphed in Figure 5.1. Note that frequent plateaus are found in each run, also present in the $r_c = 24$ runs; This suggests that the genetic operators are more frequently impotent or destructive, relative to more typical experiments in GAs. It is also reminiscent of theories of evolution via punctuated equilibria, where (real-world) evolution is believed to work via infrequent jumps.

There are some general remarks about Deva 1 evolution in general; As illustrated in Figure 5.2, a visual continuity between the phenotypes of members could typically be seen — in the example, agents show many similar qualities, including the presence of single-unit length crossbeams, hollowed-out centres, and elongated base supports.

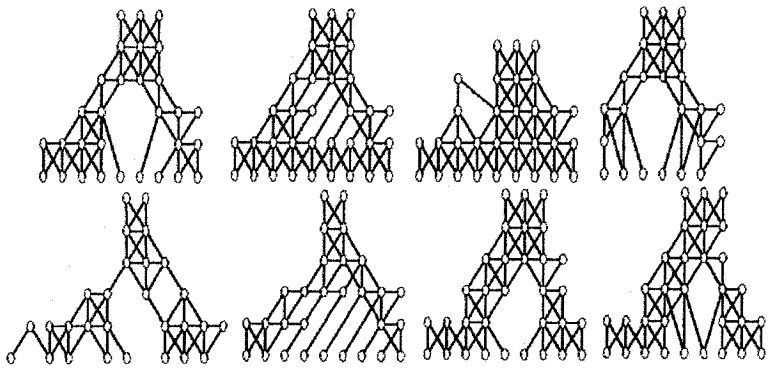


Figure 5.2: The top ten (by fitness, excluding repeats) individuals of the 100th generation of an $r_c = 12$, f_{mat} run. All shown agents were stable and capable of supporting the external load save the fifth and sixth trusses.

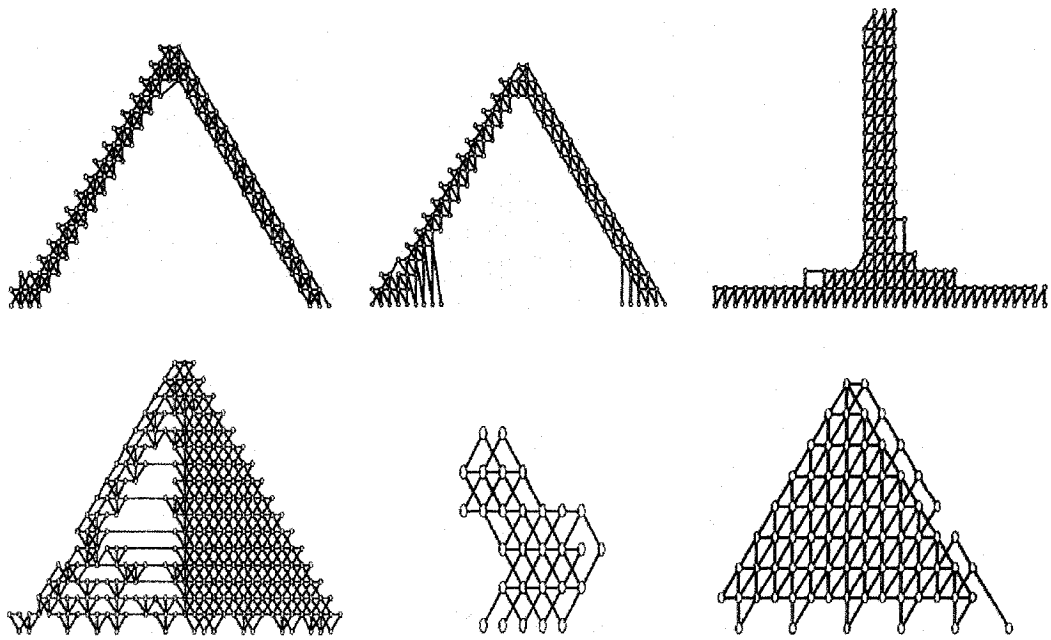


Figure 5.3: Exemplar organisms from the Fitness Trials: (*top*) two organisms from the f_{mat} function, *fit.mat*.24.4, *fit.mat*.24.1; (*middle*) two organisms from the f_{stoch} fitness function, *fit.stoch*.24.4, *fit.stoch*.24.3; (*bottom*) two organisms from the f_{base} runs, *fit.base*.16.6 and *fit.base*.16.1

There were several general trends in evolved solutions for each fitness function. For the f_{mat} function, all high fitness population members somewhat resembled the exemplar, a simple triangle shaped truss. Organisms varied greatly, however, in the f_{stoch} and f_{base} runs. For the f_{stoch} function, sparse pyramids were common; Also, there were many agents with thin, tall upper sections and large bases. For the f_{base} function, some tall trusses with small, central bases were found; Additionally, large pyramid trusses with sparse bases were also common. Figure 5.3 show exemplar population members illustrating these phenomena.

Also note the presence of "cheaters" in the trends pictured in Figure 5.3. For instance, the f_{stoch} truss shown on the left has a large and unnecessary base; Since the joints at which the external loads are applied are chosen randomly for this fitness function, selecting joints near the base helps maximize fitness in the general case by decreasing the probability that the tower will

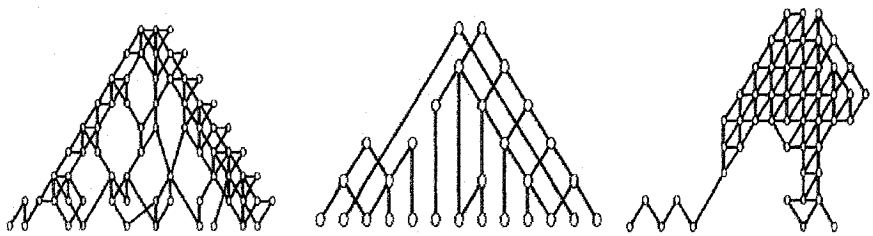


Figure 5.4: Unusual organisms: (*left*) an organism notable for a lack of repetitive pattern, and use of odd polygons, in what could have been a simple support; (*middle*) a sparse pyramid in which the second highest support is completely unconnected to the first; (*right*) a truss which minimizes the number of base joints without minimizing mass at the top, distributing weight asymmetrically.

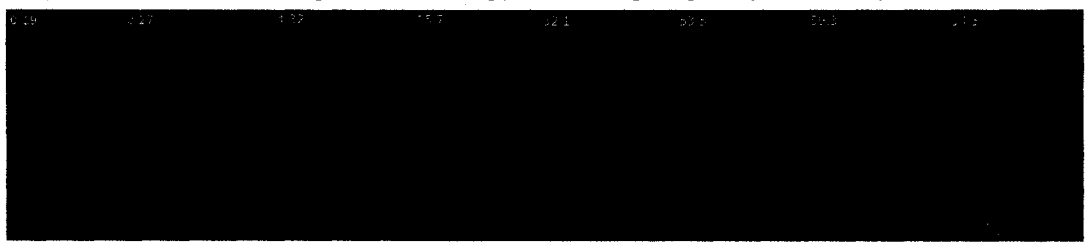


Figure 5.5: Organplexity of a Deva 1 organism. Organism is shown at development times 0, 5, 10, 15, 20, 25, 30, and 34 (termination). Inset is the *organplexity*, as defined in Eq. 3.2. Top row shows the cellular view, bottom the interpretation as a truss.

experience load. Or, for instance, consider the f_{base} truss on the right; This truss has grown a full pyramid, a generally poor design, but then removed the joints at the base, thus achieving a reasonably high fitness. As usual in Evolutionary Computation, fitness functions often have unintended loopholes.

A final note about these trusses in general is their occasional "oddness" — it is both an advantage and a drawback of Evolutionary Computation that evolution often finds solutions to problems which do not readily resemble human-designed solutions. In our trusses, this is often a result of a lack of right-left symmetry[2] — in other cases, it takes the form of unusual (and typically unnecessary) complexity in solving simple problems. Figure 5.4 shows several unusual agents, that is, agents which are reasonably functional, but bearing designs unlikely to have been created by an engineer.

Figure 5.5 shows the computation of *organplexity* for one organism in various stages of development; One may see a slow increase in the complexity measure initially, due to undifferentiated division, then faster increases as the organism grows larger. Interestingly, the peak of *organplexity* does not occur at the end, but ten time steps before, reaching a maximum of 64.3, compared to the 54.5 value at termination. This may be a measurement of some of the self-organization which is traditionally associated with a decrease in entropy.

In summary, we have presented a new model of AE, Deva 1. Deva 1 has been shown to be capable of evolving plane trusses — that is, evolving designs of structure which are stable, capable of effectively distributing external forces, and also meeting other constraints imposed by a fitness function. We have further shown that the results found by evolution can be effectively controlled through the choice of fitness function, although this is subject to the usual trappings

[2]It is expected that an addition to the fitness function could help induce symmetrical designs.

of fitness function design in evolutionary computation.

Chapter 6

Comparative Study

In this section we will perform a comparative analysis on several model-level strategies in the design of Deva algorithms. We will do so by comparing the performance of the algorithms on the previously defined tasks, that is, designing Plane Trusses to optimize one of our fitness functions.

Retrospectively, we can claim with good confidence to know the general form of the optimum for the Plane Truss application using the f_{mat} and f_{light} fitness functions; This consists of two diagonal beams supporting two or three high nodes; Either the form can consist of simply six (or four or two for f_{light}) beams forming the triangle shape, or consist of a collection of smaller beams creating two supporting diagonal frames. There does not appear to be any significant difference in fitness between the two, as what the more minimal model gains in decreased material usage it loses in less effective distribution of pressure.

We will provide evidence that the optimum of a space is defined by the fitness function and the phenotypic representation, and that there exists several AE models which will find this optimum with varying degrees of efficacy and reliability; However, many model-level choices will have the capacity to "sabotage" the discovery of this optimum.

Further, we will reinforce the idea that although selection acts on the space of phenotypes, that genetic search operates on the level of genotype, and that, as such, design decisions at the genotypic level matter greatly to the success of evolutionary search.

Finally, through our exploration of model-level mechanisms, we will evaluate the inclusion of two regularities believed important for the canalization of development in AE systems: the inheritance of cell specialization, which tends to encourage continuous and connected regions of like cell types; and the use of simultaneous all-direction division, which forces a left-right quasi-symmetrical form.

In general, our experiments will take the following form. Two sets of runs will be executed for each model-level decision: A smaller set (with $r_c = 15$, approximately) will have many runs (approximately 30) so that a general trend can be reliably established; A larger set (with $r_c = 25$, approximately) will be run as well. Since the larger runs require much more computational resources, fewer will be undertaken — the larger runs may be viewed as a means of ensuring the trend discovered at the smaller size continues at larger sizes. Evolutionary runs are undertaken until convergence can be reliably detected through visual inspection — usually 100 generations will do. The genetic parameters have been chosen from our initial experiments with *Deva*

1, and used again for all trials in the section. Experimentation with these parameters was not undertaken as our informal experimentation with the *Deva 1* trials showed little variance resulting from altering the values; Hence, we opted for the rather arbitrary re-use of original values. These values, unless otherwise noted, are:

population size	200	init. pop. size	2000
prob. crossover	0.8	rate. elitism	0.01
prob. copy-mut.	0.05	prob. power-mut.	0.05

Other parameters, such as the choice of fitness and the cardinality of the transition function, $|\phi|$, vary from experiment to experiment.

6.1 Seed Experiments

An important distinction in models of AE is the use of cellular-growth models versus the use of an initial static-sized mass of undifferentiated cells; The Deva models are obviously members of the former category, while natural systems appear to exist in between — in natural embryogenesis, an initial cleavage creates a blastula, which then begins to specialize and grow simultaneously. Cellular growth models, however, can hope to have the initial cleavage as an emergent behaviour, or seek to include it more explicitly. In this section, we explore the addition of an initial cleavage to our system via a *seed* parameter. This seed consists of a single rule placed in the first (i.e. highest priority) location of the genome, matching the empty neighbourhood, containing the action "divide", thus causing the first set of actions to be unspecialized division.

6.1.1 Experimental Set-up

Many trials of different sizes were run for both values of the *useSeed* parameter (The Seed Trials). Data was collected for runs of size $r_c = 16, 24$. The Seed Trials use the f_{base} fitness function, and a value of 100 for $|\phi|$. Twenty runs were executed with $useSeed = true$, then again with $useSeed = false$ — these runs are labelled $us.x.y$, where x is 1 if $useSeed = true$, and $y \in \{0, .., 19\}$.

6.1.2 Data and Analysis

The simplest comparison of the *us* trials is a direct comparison of the fitness of the maximum agents — since all were evolved using the f_{mat} fitness function, this is a direct measure of the effectiveness of the models. Such a comparison may be seen in Figure 6.1, where the mean fitness of the twenty $us.0$ runs is contrasted against the mean fitness of the twenty $us.1$ runs. It is evident that there exists a large initial advantage to the $us.1$ runs — in later generations this advantage lessens, where final average fitness is within a standard deviation. However, there is a clear improvement shown overall for the $us.1$ runs, with a smaller variance.

One possible cause for the difference in fitness between the two *us* trials was the proportion of trivial trusses[1] in each initial population. The occurrence of trivial organisms is common

[1] In this section, we use an older definition of trivial — less than five cells, or less than three cell types. This accounts for the inflated numbers, relative to later trials.

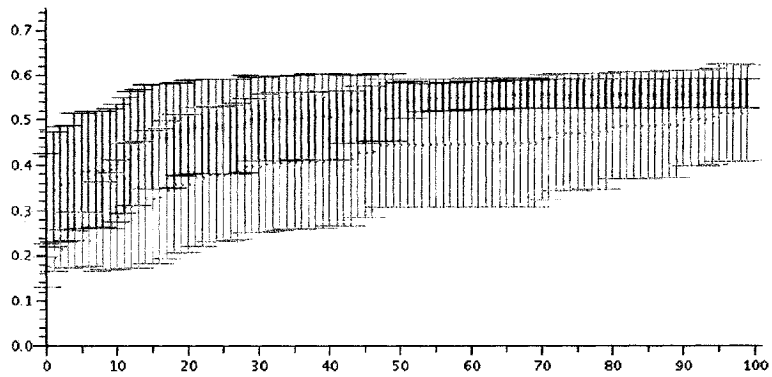


Figure 6.1: Plot of fitness (y) versus generation (x). Mean fitness of the *useSeed* runs, *us*.0 runs: (*lighter*) and *us*.1 runs (*darker*) — the bars indicate a single standard deviation.

in the *Deva 1* model, typically comprising the majority of the population[2]. In the *us*.0 trial, the initial population was 99% trivial (mean of 20 runs), compared with 95% for the *us*.1 trial. However, this initial advantage soon disappeared, as the proportion of trivial agents soon settled at approximately 60% for both trials. The initial advantage in the *us*.1 trial, allowed for a more diverse initial population as fodder for the remaining genetic search, likely explaining the initial fitness advantage.

In summary, we have empirically explored a model-level addition, the *useSeed* parameter, a metaphor for the initial cleavage in real-world embryogeny. Relative to the task of designing a stable, effective, tall truss with minimal material usage, the *useSeed* strategy has proven effective — that is, inclusion of the seeded rules leads to an improvement in expected fitness generally, and a decrease in the variance between runs, suggesting a greater probability of generating a good truss design on a typical run.

6.2 Neighbourhood Experiments

It is not obvious which choice of neighbourhood description is ideal for the decision-making required of a transition function. Here, we restrict our attention to rule-based (or pattern-matching) transition functions, and explore several levels of description of cell neighbourhoods. We are not interested in the size of the neighbourhood, which has already been explored in the context of understanding Cellular Automata; Instead, we explore means of compressing the description of these neighbourhoods, so as to limit the necessary representation size.

Our trade-off here is the amount of specificity in any particular neighbourhood description, which we would like to minimize, versus the ability of a rule-based genome to intelligently make decisions for growth. Given a neighbourhood of size 12 (see Appendix B for description), we might include **all** relevant information by simply listing all cell types, as we do in *Deva 3*; This leads to n_c^{12} possible neighbourhoods, making, as we shall see, difficult demands on the initialization necessary for a representative population.

Here, we explore two means of describing neighbourhoods in less detail, *Deva 1* and *Deva 2* — we may view *Deva 3* as retaining the largest amount of information regarding the neighbourhood,

[2]We are relatively unconcerned, since these trusses can usually be dismissed without significant processing — hence, we simply use large population sizes.

followed by *Deva 1*, then finally *Deva 2*. Additionally, we explore the use of a "state" in partitioning the DNA; That is, in some models, we allow the DNA to be partitioned on the basis of the cell type of the querying cell. Hence, a cell of type "1" queries different sections of the genome as does a cell of type "2". In total, we consider five models, *Deva 1.N*, *1.Stateless*, *2*, *2.Stateless*, and *3*. More precise descriptions of the algorithms may be found in Sections B.2, B.4, and B.5.

In this section, we use the f_{light} fitness function, similar to f_{mat} but with a substantially lighter external load. For this function, it is possible to support the external load on a single joint, with some additional buttressing. Global optimum appears to be either two triangles of equal height, or a single point with some minimal buttressing — neither of the two substantially outperforms the other.

6.2.1 Choice of $|\phi|$

In this set of experiments, we explore a range of values for the parameter $|\phi|$ in various Deva models. This is done both to illustrate the relation between the length of ϕ and obtainable fitness, and also so as to choose the parameter intelligently for future experiments.

100 runs of 50 GA generations were undertaken for Deva models *1.N*, *1.Stateless*, *2*, *2.Stateless*, and *3*. These trials used an r_c value of 14, and had an initial population size of 1000. The parameter $|\phi|$ was chosen independently, uniformly and randomly from the range $\{0, 400\}$ for each run.

Results of the runs are shown in Figure 6.2; These plots show the maximum fitness achieved in the initial (random) population and after 50 generations, according to value of $|\phi|$. For each, a line of best fit is plotted, showing the general trend as determined by linear sum-of-square analysis.

6.2.2 Comparative Neighbourhood Experiments

In this set of experiments, we explore the effects on evolution of our choice of the five models. We do so by considering two sets of trials for different values of r_c.

Experimental Set-up

30 runs of 150 GA generations were undertaken for Deva models *1.N*, *1.Stateless*, *2*, *2.Stateless*, and *3* for $r_c = 14$ and 20 runs of 100 GA generations with $r_c = 22$. These runs are labelled *nbhd.x.y.z*, where x is an r_c value, y is a Deva model, and z is an index. These trials used the f_{light} fitness function.

The parameter $|\phi|$ was set according to model type, based on data from the previous section: for *Deva 1.N*, $|\phi| = 300$; for *Deva 1.Stateless*, $|\phi| = 150$; for *Deva 2*, $|\phi| = 250$; for *Deva 2.Stateless*, $|\phi| = 100$; and for *Deva 3*, $|\phi| = 250$.

Comparison of Neighbourhood Data

We plot the results of evolution in Figures 6.4, 6.5, and 6.6, and summarize data regarding the maximum fitness in Table 6.1. We note that organisms approaching global optimum were found

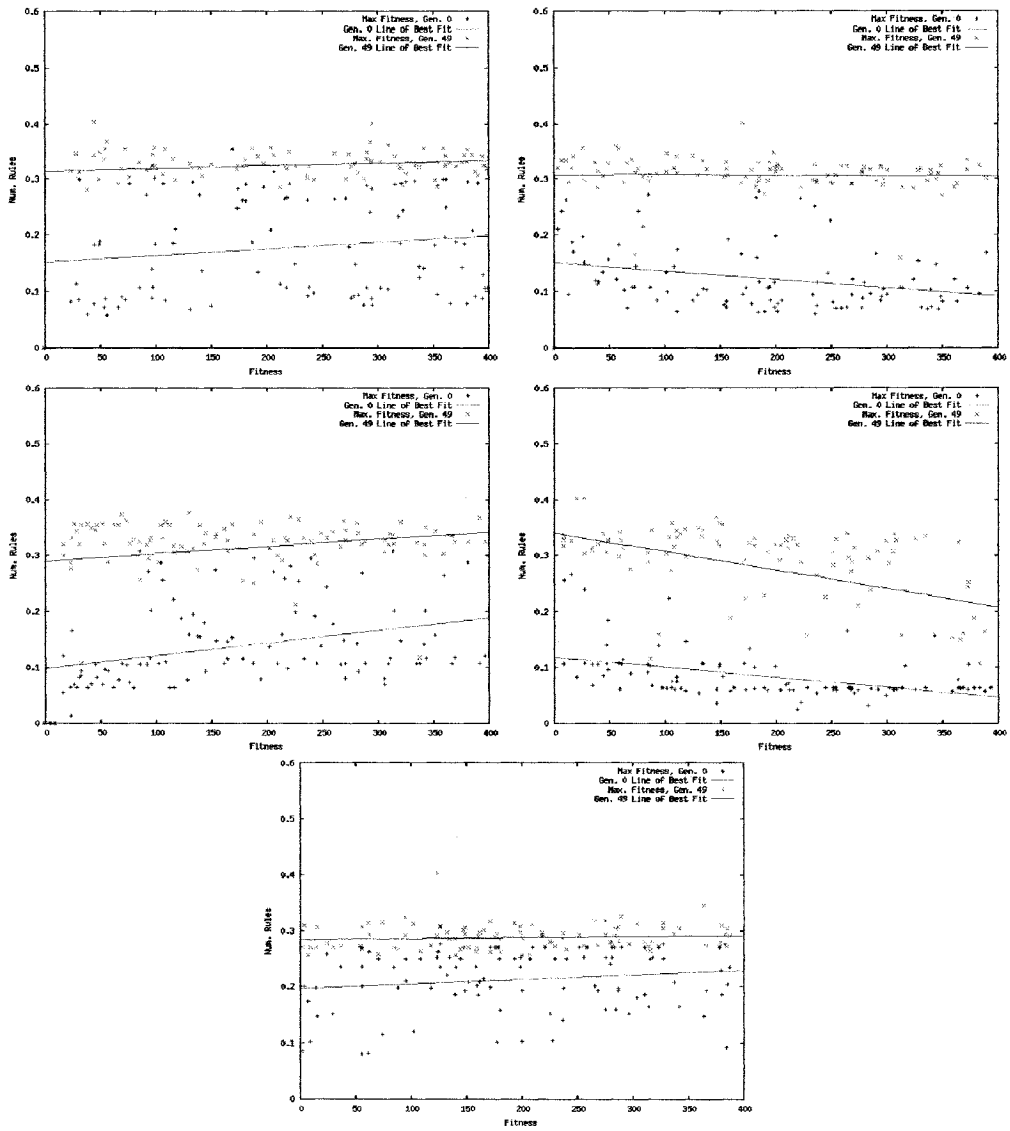


Figure 6.2: Plot of the maximum fitness for various values of $|\phi|$ with fitness (y-axis) versus generation (x-axis); (*top left*) Model *Deva 1.N*; (*top right*) Model *Deva 1.Stateless*; (*left*) Model *Deva 2*; (*right*) Model *Deva 2.Stateless*; (*bottom*) Model *Deva 3*;

in the *nbhd.22.1.N* and *nbhd.22.2* trials, although not in any others. In Figure 6.3, we show the maximum and closest-to-mean organisms of the *nbhd.14* runs.

Our initial question involves the use of "states"; This through the *Deva 1.N* and *Deva 2* experiments, illustrated in Figures 6.4 and 6.5, respectively. In the case of *Deva 1.N*, there is a clear increase in expected maximum fitness, although not greater than one standard deviation, and a significant decrease in variance. In the case of *Deva 2*, there is no obvious advantage of either the original or the stateless model, save that *Deva 2* seems to have a smaller variance in the *nbhd.22* runs.

Examining Figure 6.3, we note that the mean agents of the *Deva 1.N* and *2* models seem to be closer to our expected global optimum, while the "stateless" models seem to be constructing

Table 6.1: Maximum and mean of maximum fitness agents from the final GA generation of the *nbhd.14* and *nbhd.22* runs.

Deva Type	$r_c = 14$ Max. Fit.	Mean Fit.(s.d.)	$r_c = 22$ Max. Fit.	Mean Fit. (s.d.)
1.N	0.420	0.334 (0.023)	0.566	0.427 (0.034)
1.Stateless	0.339	0.318 (0.013)	0.437	0.359 (0.129)
2	0.400	0.338 (0.028)	0.566	0.389 (0.113)
2.Stateless	0.389	0.333 (0.034)	0.491	0.399 (0.059)
3	0.352	0.305 (0.019)	0.415	0.376 (0.028)

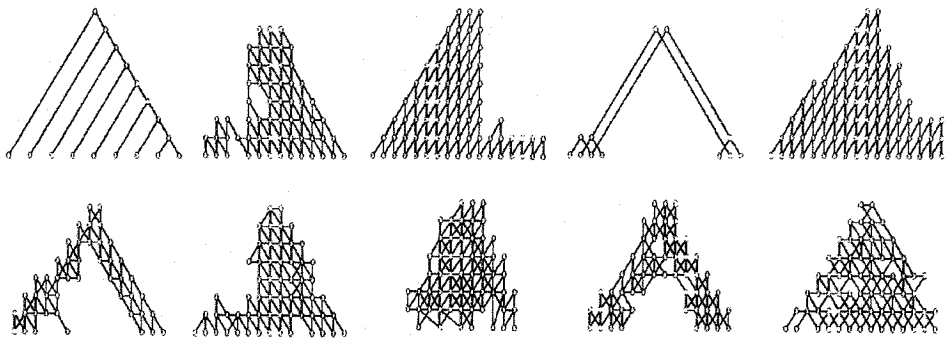


Figure 6.3: Maximum fitness agents from the maximum (*top*) and closest to mean (*bottom*) runs for the (*left to right*): *Deva 1.N, 1.Stateless, 2.Stateless, 2* and *3 nbhd.14* trials.

tall, thin structures, known to be a less efficient strategy (for the similar f_{mat} fitness function). The best runs of the *Deva 1.N* and *2* models outperformed the best runs of the "stateless" models, as did speed of evolution, suggesting that the "stateless" models were pursuing less effective strategies in general.

Finally, we compare the *Deva 1.N*, *Deva 2*, and *Deva 3* models, illustrated in Figure 6.6. There does not appear to be any significant difference between *Deva 1.N* and *Deva 2*, save that *Deva 2* evolved more slowly in the *nbhd.22* trials — this suggests that the difference in description in these two models is largely insignificant. However, the *Deva 3* model performed far worse than the two previous models. Most likely, this is due to the tremendous size of the

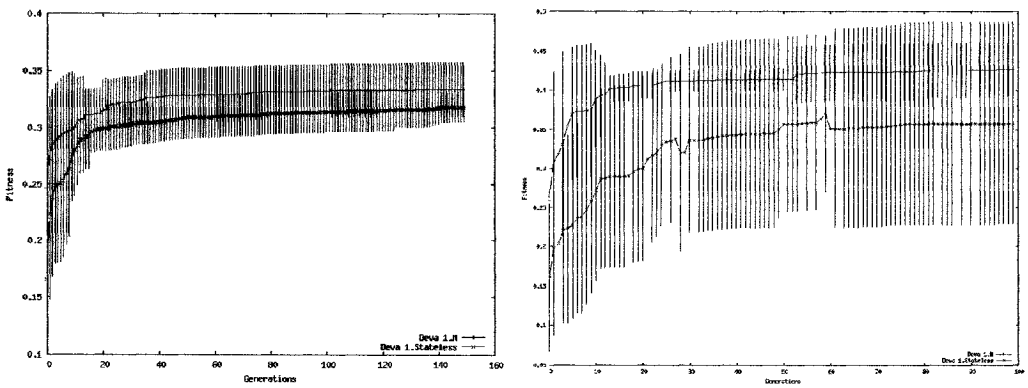


Figure 6.4: Plot of mean of maximum fitness for the *Deva 1.N* and *1.Stateless* runs.

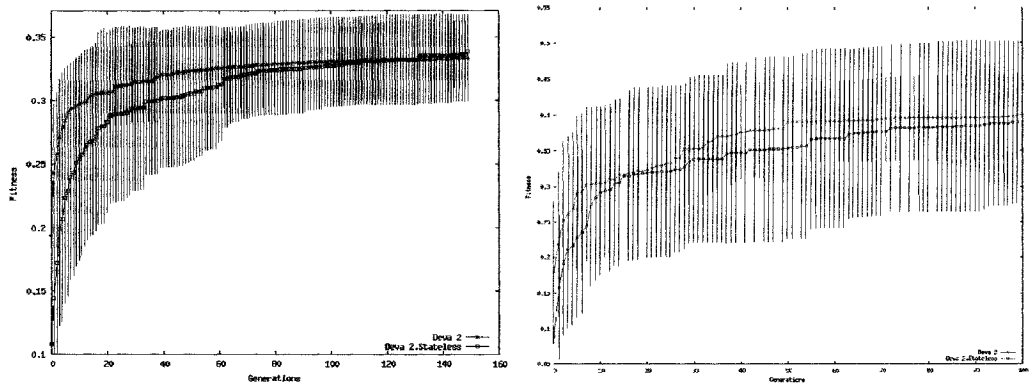


Figure 6.5: Plot of mean of maximum fitness for the *Deva 2* and *2.Stateless* runs.

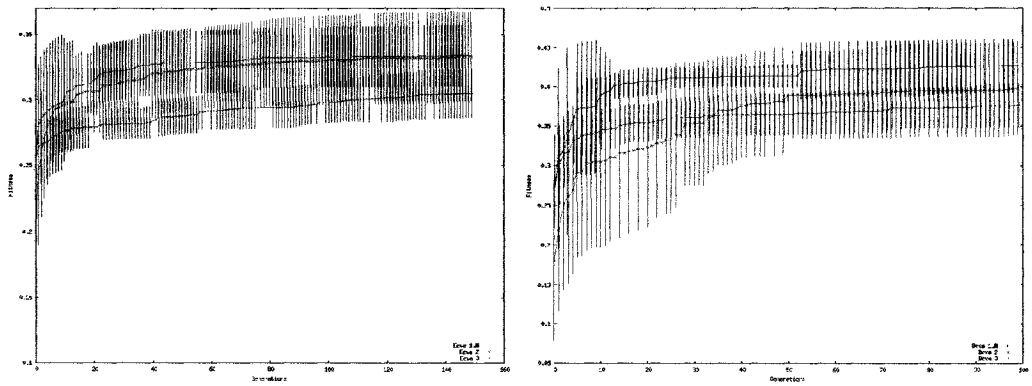


Figure 6.6: Plot of mean of maximum fitness for the *Deva 1.N*, *Deva 2*, and *Deva 3* runs.

genetic search space, coupled with the precision of the description. If so, it is likely that a smaller set of colours would lead to different results in this latter experiment.

For any given organism, we may enquire as to the number of used rules in the genome — these are rules in the genome activated at least once during development, and hence serve as a loose measure of the complexity of the decision-making mechanism. The number of used rules tends to be a small fraction of the total number of rules, and tends towards a certain level for each deva algorithm. Figure 6.7 shows the general trend for each of the runs, averaging the values for each of the maximum fit agents. In this figure it can be seen that there exists a general trend towards more rules for less descriptive Deva algorithms; For instance, there is a clear increase in number of used rules for *Deva 1.Stateless* over *Deva 1.N*, as there is for *Deva 2.Stateless* relative to *Deva 2*. Also, *Deva 2* models tend to use more rules than *Deva 1*, which in turn use significantly more rules than the highly precise *Deva 3* model. Since the pattern-description occupied the majority of the space of any particular rule, the size of genetic space is controlled largely through the specificity of these neighbourhood descriptions, no doubt influencing the effectiveness of the GA.

It is likely that the additional rules used serve to provide a specificity which used to be provided by the model; For instance, the *Deva 2.Stateless* algorithm often uses additional cell specializations and rules to accomplish what is done by a simple specification of state in *Deva 2*, effectively breaking the genome into modular sub-genomes.

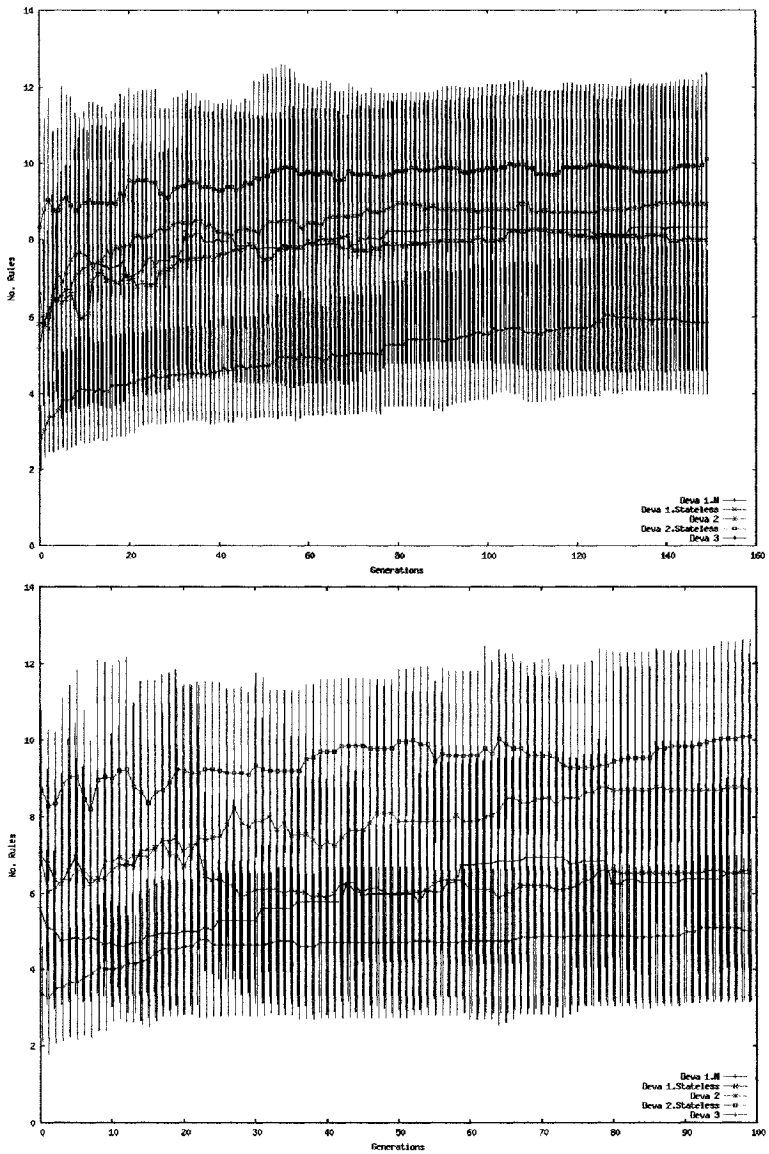


Figure 6.7: Mean number of rules used by the maximum fit agents for the *nbhd.14* (*top*) and *nbhd.22* (*bottom*) runs.

From this, we may conclude that the choice of neighbourhood description is not trivial. The *Deva 3* model, which, as the closest to general Cellular Automaton description, resembles several models used in the literature — this model does not lead to the same quality of evolved organism as the other models. The distinction between the descriptions used in *Deva 1* and *Deva 2* have proven far less significant, however, showing that the performance differential resulting from this choice need not be overly sensitive, that evolution is able to some degree to compensate, in this case by selecting the number of used rules. All of these results are likely highly dependant on the number of cell types chosen.

Additionally, in nearly all examined cases, there was a clear preference for the specification of "states" in expected best run if not mean, probably reproducible by any model which allows the genome to be partitioned into exclusive sub-genomes selected by cell state.

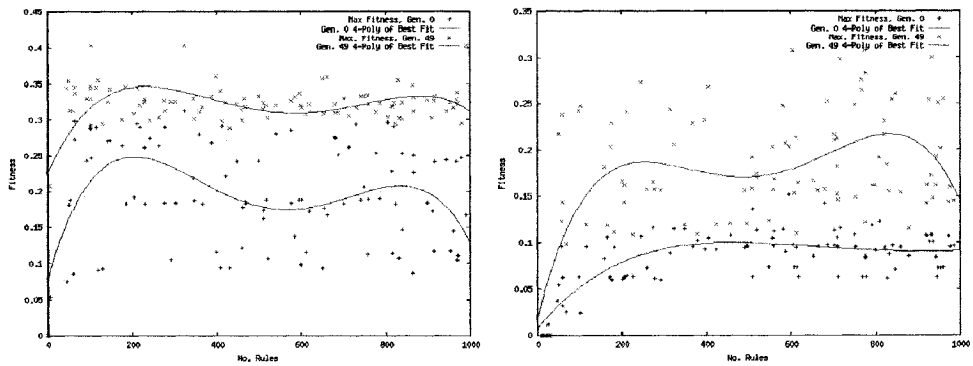


Figure 6.8: Plot of the maximum fitness for various values of $|\phi|$ with fitness (y-axis) versus generation (x-axis); (*left*) Model *Deva 1.N*; (*right*) Model *Deva 1.Dir*.

6.3 Division Experiments

In this section, we will continue to explore the Deva 1 model in particular; This choice as it was most capable of maximizing fitness in previous experiments. However, in this case, we consider several variants, varying on the basis of the action list chosen. As was already implied, but not formally explored, it is known that minor modifications of the action list may improve overall evolutionary performance (hence, our concentration on *Deva 1.N* rather than *Deva 1*).

We will consider several models, initially described in Section 4.2.2; These models include:

- *Deva 1.N*, our original model using the ***best free location*** algorithm for choice of direction.
- *Deva 1.Dir*, where directions are specified directly by the genome.
- *Deva 1.Clockwise*, where the direction chosen is a simple, heavily biased function.
- *Deva 1.AllWay*, where division occurs in all four directions simultaneously.

6.3.1 Choice of $|\phi|$

We do not expect significant difference between the choice of $|\phi|$ for *Deva 1.Clockwise* and *Deva 1.AllWay* relative to *Deva 1.N* for the reason that the "divide" actions in each play the same roles; However, the *Deva 1.Dir* model is a sufficient divergence that we re-run the experiment for choice of $|\phi|$, since the number of "divide" commands will likely need to be higher. These runs both explore the *Deva 1.Dir* model, and solidify our understanding for the *Deva 1.N* model.

500 runs of 100 GA generations were undertaken for Deva models *1.N*, and *1.Dir*. We used an initial population size of 1000, and let $r_c = 12$. The parameter $|\phi|$ was chosen independently, uniformly and randomly from the range $\{0, 1000\}$ for each run.

Results of the runs are shown in Figure 6.8; These plots show the maximum fitness achieved in the initial (random) population and after 50 generations, according to value of $|\phi|$. For each, a 4-term polynomial of best fit is plotted, showing the general trend as determined by sum-of-square analysis. A 4-term polynomial was chosen as the *Deva 1.Dir* trials appeared to be bi-modal.

As in initial experiments, we select $|\phi| = 200$ for *Deva 1.N*, and hence, our other action models; Further, we select $|\phi| = 800$ for *Deva 1.Dir*. We also note that variance in the latter runs is significantly higher than in the former.

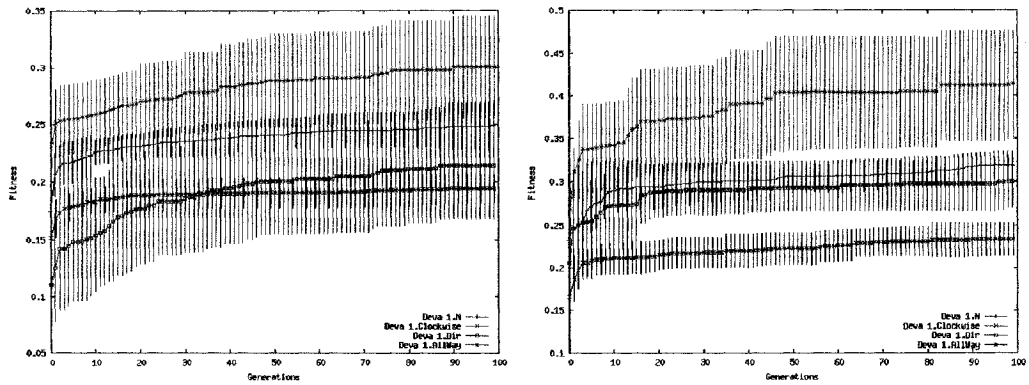


Figure 6.9: Plot of mean of maximum fitness for the *Deva 1.N, 1.Clockwise, 1.Dir* and *1.Allway* runs.

Table 6.2: Maximum and mean of maximum fitness agents from the final GA generation of the *div.14* and *div.22* runs.

Deva Type	$r_c = 14$ Max. Fit.	Mean Fit.(s.d.)	$r_c = 22$ Max. Fit.	Mean Fit. (s.d.)
1.N	0.317	0.249 (0.022)	0.357	0.319 (0.016)
1.Dir	0.287	0.214 (0.046)	0.356	0.301 (0.030)
1.Clockwise	0.224	0.194 (0.012)	0.286	0.234 (0.019)
1.AllWay	0.361	0.301 (0.045)	0.475	0.413 (0.064)

6.3.2 Division Experiments

The division trials are labelled *div.x.y.z*, where: x is an r_c value, either 14 or 22; y is a Deva Model, one of *Deva 1.N, 1.Clockwise, 1.Dir* or *1.Allway*; and z is an index, between 0 and 29 for $r_c = 14$, or between 0 and 19 for $r_c = 22$. In these experiments we used the fitness function f_{mat}. $|\phi|$ was set according to the values specified above.

6.3.3 Comparison of Division Data

We plot the result of evolution in Figure 6.9; The maximum and mean of the fitness of the best agent in GA generation 100 is listed in Table 6.2 for each division type. In terms of overall performance, as measured by the maximum fitness agent at generation 100, there was a clear separation between the *div* runs, moreso than in other comparative experiments. The best performance was observed in the *div.Allway* runs, which outperformed the next best division type by a full standard deviation in both r_c settings. This was followed by the *div.1.N* runs, then the *div.Dir* runs, with the lowest performing runs being the *div.Clockwise*.

Figure 6.10 show exemplar agents from the *div.14* runs, both the best organism from the maximum fitness run, and the best organism from the run that was closest in performance to the mean.

The performance of *Deva 1.Clockwise* was unsurprisingly poor; Unsurprisingly, as the method for location selection was quite arbitrary, with a strong bias towards the left. Organisms evolved through *Deva 1.Clockwise* had a tendency to resemble large groups of identical beams and joints leaning left, of lesser height than other trusses from these experiments; The decrease in height was due to the left-leaning divisions, which used up resources prior to growth encouraging height.

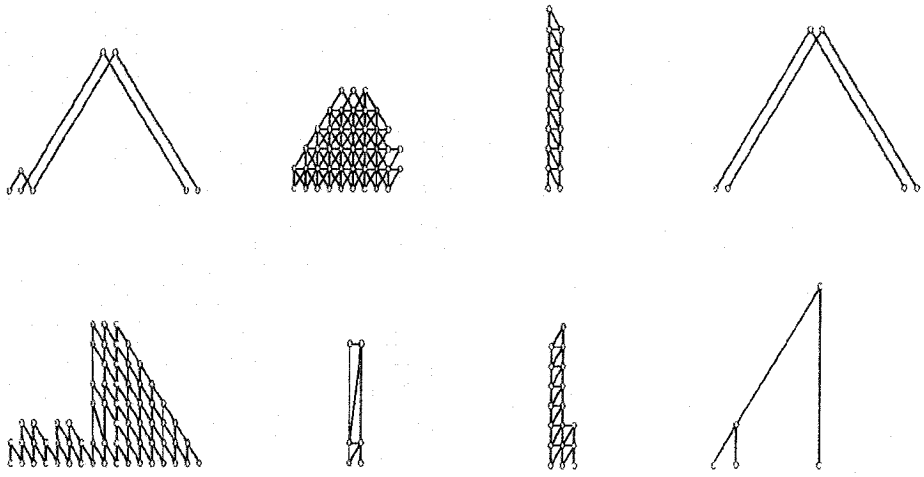


Figure 6.10: Maximum fitness agents from the maximum (*top*) and closest to mean (*bottom*) runs for the (*left to right*): *Deva 1.N, 1.Clockwise, 1.Dir* and *1.Allway* for the *div.14* trials. The maximum and mean organisms for the *1.Dir*, along with the mean agents for the *1.Clockwise* and the *1.AllWay* runs were stable, but not capable of supporting the applied external load.

The *Deva 1.Dir* model has a strong tendency to become stuck in a local optimum. Nearly all discovered trusses consisted of very tall and thin structures, which while stable, cannot support the applied external load. These trusses seem to maximize the height, material usage and stability factors of the fitness function very quickly, but cannot, from this stage, evolve into load-bearing trusses. Hence, the allowance of full genetic control in the division operator seems to be a hindrance, while the tyranny of a pre-specified procedure for cell growth, such as *best free location* or *all-way* forces a more robust approach.

The *Deva 1.AllWay* model exploits a propensity for bilateral symmetry (quasi-symmetry, that is, see Appendix B.3.3). That is, in this simple Plane Truss application, there does not appear to be any advantage to having different structures over the $x = 0$ axis, meaning that the typical *Deva* evolution spends unnecessary time designing both the right and the left halves. The growth of *Deva 1.AllWay* organisms are always quasi-symmetric, which require less specification at the genomic level, allowing for a faster and more reliable evolution. Hence, the *Deva 1.AllWay* model is capable of successfully exploiting a useful regularity on the space of Plane Trusses.

6.4 Gene-based Specialization

In this section we explore the use of an alternate form of specialization, one more closely related to that found in morphogen-based experiments. In the *Deva 4* models, we have removed much of the specificity of the action set, replacing the highly specified colour-specialization commands with more continuous gene-based differentiations.

6.4.1 Experimental Set-up

30 runs of 100 GA generations were undertaken for each of Deva models *1.N*, *2*, *4.CellType*, and *4.GeneType* with $r_c = 15$, and 10 with $r_c = 25$. In these experiments, we let $|\phi| = 250$, and

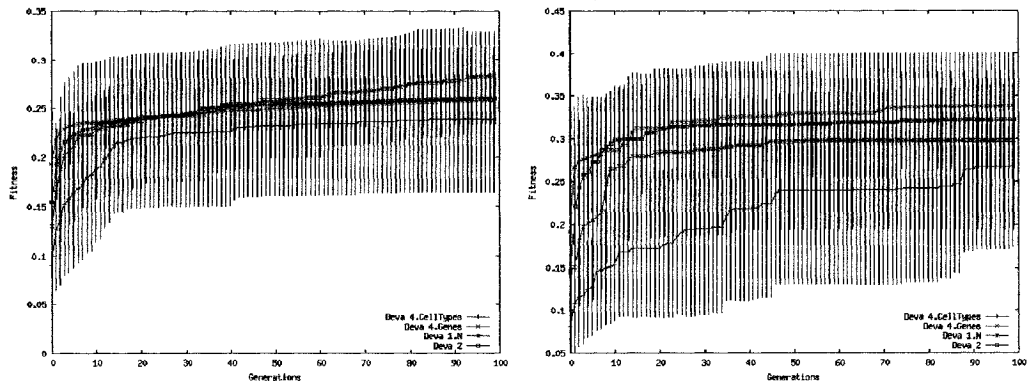


Figure 6.11: Plot of mean of maximum fitness for the *gb.15* runs (*left*) and the *gb.25* runs (*right*)

used the fitness function f_{mat}. The runs were labelled $gb.r_c.x.y$, where x is a Deva model and y an index.

6.4.2 Data and Analysis

Mean of maximum fitness is plotted in Figure 6.11. Maximums and means are listed in Table 6.3. The mean of number of used rules for maximum fit agents is plotted in Figure 6.12. The performance of the *Deva 4.GeneType* was slightly better than that of *Deva 1* and *2*, although within a single standard deviation of both results. The performance of *Deva 4.CellType* was worse than all others, falling below that of *Deva 4.GeneType* in both absolute fitness, and speed towards convergence; Again, the results were reasonably close, less than a single standard deviation of spread. Further, these trends were not continued in the larger runs, where it appears that *Deva 4.GeneType* does not outperform the other models at all.

Still, there does appear to be some preference for *Deva 4.GeneType* over *Deva 4.CellType*; One possibility is that the use of gene-based description in both the action specification and the neighbourhood description makes it easier to borrow genetic material from one section of the genome and copy it successfully to another, as through crossover or copy-mutation.

Comparing the number of used rules between the models, we see that it is likely that evolution is adapting to the model at hand. In the case of *Deva 1.N* and *Deva 4.CellType*, whose genomic structure is nearly identical save for the action specifications, we see a clear increase in the number of rule specifications for the *Deva 4* model; The same phenomenon may be seen for the comparison between the *Deva 2* and the *Deva 4.GeneType* models. Hence, it seems that the *Deva 4* models compensate for less capacity for specification of cell specialization in the genome by finding additional rules to make up the difference.

All in all, these results are quite weak, and there does not appear to be any substantial reason to prefer or eschew *Deva 4* relative to other models; It seems that evolution can typically adapt to accommodate this change.

The most likely explanation for the failure of the gene-based specialization experiments to produce improvements is as follows: The pattern-based Deva style of representation requires significantly more information to describe an action than a pattern. In the case of colour-based descriptions (as in *Deva 1.N* and *Deva 4.CellType*), the number of "slots" for description was

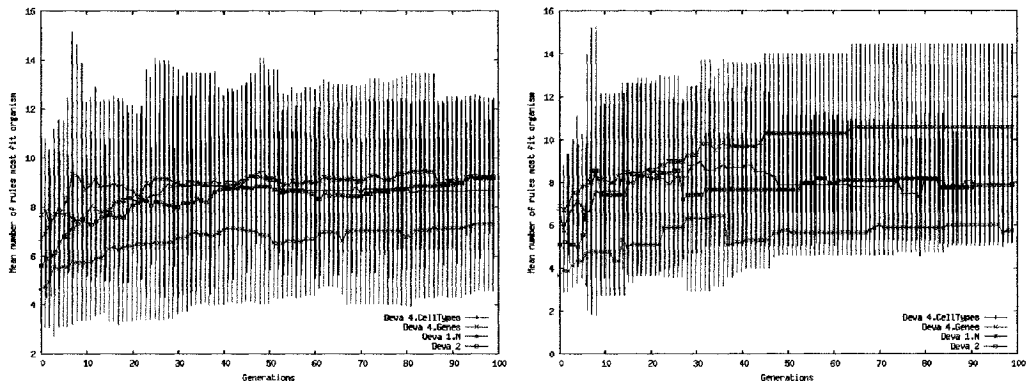


Figure 6.12: Plot of mean of number of rules used for the *gb.15* runs (*left*) and the *gb.25* runs (*right*)

Table 6.3: Maximum and mean of maximum fitness agents from the final GA generation of the *gb.15* and *gb.25* runs.

Deva Type	$r_c = 15$ Max. Fit.	Mean Fit.(s.d.)	$r_c = 25$ Max. Fit.	Mean Fit. (s.d.)
4.GeneType	0.338	0.282 (0.048)	0.418	0.298 (0.081)
4.CellType	0.338	0.238 (0.073)	0.395	0.269 (0.091)
1.N	0.337	0.258 (0.021)	0.368	0.339 (0.091)
2	0.340	0.260 (0.026)	0.362	0.322 (0.067)

$|C| = 33$, while the number of slots for gene-based description (*Deva 2* and *Deva 4.GeneType*) was six. This compared with a single slot required to store the action — hence, the size of genotypic space was far more influenced through the descriptions of patterns than through resulting actions. Unsurprisingly, the genomic complexity (as measured by the number of active rules) was controlled by the neighbourhood descriptions rather than the actions; By sacrificing the specificity of the actions, we gained little in terms of decreasing the size of the genomic space, but sacrificed much in terms of control of phenotypic growth.

6.5 Inheritance of Cell Specialization

In this section we explore the inheritance of cell specialization in division. As previously mentioned, this inheritance is an important mechanism in development, believed responsible for a regularity in developed form which serves as a canalization of development.

6.5.1 Experimental Set-up

30 runs of 100 GA generations were undertaken for each of Deva models *1.N*, *2*, *1.NoInher*, and *2.NoInher* with $r_c = 15$, and 10 with $r_c = 25$ (data for Deva *1.N* and *2* re-used from previous experiments). In these trials, we let $|\phi| = 250$, and the f_{mat} fitness function was used. We labelled these runs *inher*.r_c.x.y, where x is a Deva model and y is an index.

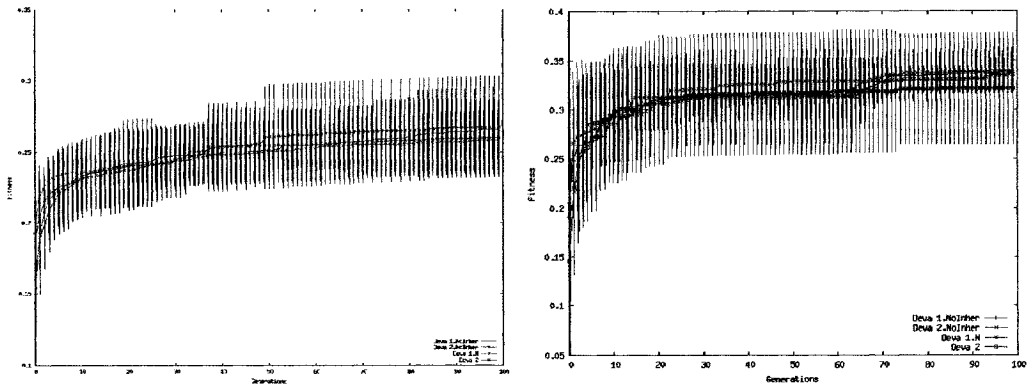


Figure 6.13: Plot of mean of maximum fitness for the *inher.15* runs (*left*) and the *inher.25* runs (*right*)

Table 6.4: Maximum and mean of maximum fitness agents from the final GA generation of the *inher.15* and *inher.25* runs.

Deva Type	$r_c = 15$ Max. Fit.	Mean Fit.(s.d.)	$r_c = 25$ Max. Fit.	Mean Fit. (s.d.)
1.N	0.337	0.258 (0.021)	0.368	0.339 (0.091)
1.NoInher	0.340	0.266 (0.029)	0.392	0.340 (0.085)
2	0.340	0.260 (0.026)	0.362	0.322 (0.067)
2.NoInher	0.340	0.268 (0.035)	0.369	0.337 (0.082)

6.5.2 Data and Analysis

Mean of maximum fitness is plotted in Figure 6.13. Maximums and means are listed in Table 6.4. All runs converge to approximately the same value, all within a single standard deviation of each other; Further, this trend continues through all of evolution, so that no algorithm appears significantly more efficient than any other.

Surprisingly, we found no advantage to the automatic inheritance of cell type by child cells. Note that this is quite a break from several other experiments in which inheritance of cell specialization is posited as a significant factor contributing to the beneficial regularities of AE systems. It seems that in the example of Deva growth and Plane Trusses, totempotic cell division is equally efficient to the typical AE choice of automatic inheritance.

This does not necessarily discount the regularity imposed by automated specialization, though. Given that organisms still appear to be continuous regions of collections of like cells, it is likely that evolution can, in this case, simply quickly learn to perform specialization to the same type as neighbouring cells, thus achieving a similar regularity as would be gained from inheritance.

6.6 Summary of Results

In general, we have not seen improvement on the overall optimum found by the system relative to the best of our original *Deva 1* experiments; Indeed, it is likely the optimum is defined by the phenotypic representation and fitness function, and that there are several good models capable

of finding it. We have significant evidence, however, that choice of Deva model can affect the speed and reliability of discovery of this optimum, and that poor model-level decisions can cause a system to get stuck at local optima.

Further, we have explored several model-level decisions, with a variety of results, detailed below:

- We have explored the inclusion of an initial cleavage stage, potentially recovering a middle ground between models of cellular growth and models involving the differentiation of a mass of unspecialized cells. This initial cleavage stage was shown to improve the reliability of the algorithm overall, reducing the number of trivial agents.
- We explored the description of neighbourhood used, and showed that although some adaptation is present in successful descriptions, a poor choice will severely limit the capacity for evolution. In this example, we found a wide array of behaviours based on the relatively simple decision of how precisely to describe a cell's neighbourhood. The precise description sabotaged evolution entirely, while evolution seemed capable of adapting to other changes in genomic cardinality; This suggests that the evolutionary engine was bogged down by the size of the search space, further implying that different choices of domain of application (and hence, different genomic dimensionalities) will have substantively different results.
- We explored several options for cell division, finding ways to abstract the division of cells onto a simple lattice. Here, we discovered a strong preference for a quasi-symmetric division operator, and good results from our *best free location* algorithm, while other choices, including full genetic control, were stuck in local optima.
- We explored the use of gene-based specialization, as opposed to the more specific cell-based specialization. In this case, results were too weak to specify any particular preference; Again, it appears that evolution has found a means to adapt to the model-level change through increasing the number of used rules in the genome.
- Finally, we found no particular difference in the inclusion of a mechanism for the inheritance of cell specialization; This is somewhat of a surprise, since this is a mechanism identified previously as a key factor in the development of an important biological regularity. It is likely that in this case, evolution simply found a simple way to achieve the same result though additional cell specializations.

We see reinforcement of two general trends in the results above: Firstly, that the canalization of development found in AE systems is likely due to the exploitation of regularities formed by model-level mechanisms, although that the appropriateness of those regularities likely depends on the domain of application; And secondly, the need to view evolution as selecting on the space of phenotypes, but acting on the space of genotypes.

The former point, the exploitation of regularities, was reinforced by the experiments in division mechanisms; There, one model, through exploiting a phenotypic symmetry, managed to find solutions far more efficiently and reliably than the all other explored models. The applicability of left-right symmetry is, of course, a feature of this particular domain of application, and might not exist in other problems. The second explored mechanism, inheritance of cell specialization, proved impotent, but likely because evolution found a way to achieve a similar regularity through alternate means.

The latter point, the importance of viewing evolution as selecting phenotypes but acting on genotypes, is evident firstly in the definition of the optimum by the choice of phenotypic representation and fitness function, contrasted to the richness of behaviour found in genetic changes. The model-level mechanisms which affected the actions of an organism had a tendency to be compensated for by evolution, as in the inheritance and the gene-based trials; No doubt this is due to the relative small size of genomic representation of the actions. In contrast, the neighbourhood descriptions proved potentially fatal to evolution, likely due to their dominance of the genomic space.

This final point is especially important when considering the evaluation of model-level decisions in AE experiments; One cannot by default assume that a result which holds in a low-dimensional problem will hold also in a higher-dimensional space. It is often the case in models of AE that the size of the genomic representation is exponential in the number of phenotypic primitives, as in the case of morphogenic models and Cellular Automata, which, of course, means that many models defined on two- or three-colour problems will simply be intractable for other domains of application.

Chapter 7

Environment Experiments

In this chapter, we explore the use of environment as a guide for embryogenesis. This is achieved through the expression of our phenotypes in a variety of developmental spaces, controlling their growth through their geometric shape.

Inspired by earlier discussion regarding the role of environment in canalizing (channelling) development, here we attempt to first use the environment as a spatial constraint (to an extent, replacing the use of a fitness function as a selector of overall morphological design), and second, to use this constraint not as a final evaluation, but instead as a guiding force during the developmental process.

Following this is a set of experiments devoted to studying the re-growth of genomes evolved in some environment, transplanted into another. As a metaphor for re-growth, imagine the following situation: A scientist finds a recently impregnated Bonobo monkey, whose womb contains a single fertilized cell, ready to begin growing into a new Bonobo child. The scientist removes this zygote from the Bonobo monkey, and transplants it into the womb of, say, a Chimpanzee. The scientist then studies the development of this child in the Chimp womb. Similarly, we select genomes of trusses evolved in some environment, then allow them to re-develop in a different environment, and observe the results in terms of maximization of a fitness function.

The re-growth experiments take two forms: Firstly, we consider the re-growth of genomes under different phenotypic sizes. That is, we take a genome evolved under the assumption that growth would continue until a height of 30 m, and re-grow the truss to a height of 40 m; Secondly, we re-grow genomes in different environments altogether.

The purpose of the re-growth experiments is to measure the utility of genomes when translated into phenotypes under perturbed environments; This is done for two reasons: Firstly, to explore the possibility that good developmental patterns exist in several contexts, somewhat like the genetic toolkit currently being explored in real-world Embryology; Secondly, to reinforce the claim that Artificial Embryogeny is a robust and perturbation-resistant technique for automated design, and that good genomes may be re-used in new environments without repeating the evolutionary optimization.

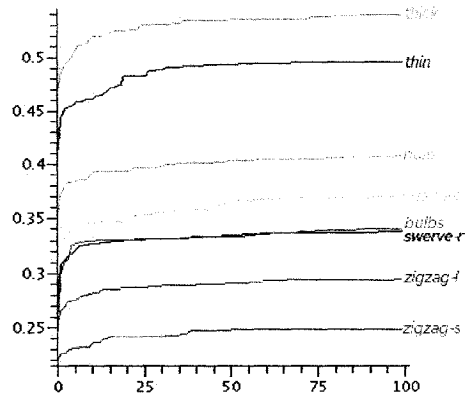


Figure 7.1: Plot of the mean of maximum fitness for the sets of runs in each environment: fitness (y-axis) versus generation (x-axis).

7.1 Environment as a Spatial Constraint

7.1.1 Experimental Set-up

80 runs of 100 GA generations were undertaken, grouped together by environment. These runs will be referred to as $r.env.x$, where env is the name of the environment, and $x \in \{0, .., 9\}$ is an index. Parameter settings for the trials are:

population size	100	init. pop. size	1000
prob. crossover	0.8	rate. elitism	0.01
prob. copy-mut.	0.05	prob. power-mut.	0.05
r_c	40	$\|\phi\|$	200

Unless otherwise noted, these parameters are used for all experiments, including re-growth.

7.1.2 Initial Experiments

In all environments, stable agents capable of supporting the external load were found in nearly all trials, save the *zigzag-s* environment, where only one trial yielded a non-trivial truss capable of supporting the external load. The failure of evolution of a tall, load-bearing truss in the *zigzag-s* environment is a mild disappointment, since it is possible (for a human engineer) to design a truss capable of reaching non-trivial height for this environment. The relative success of the GA, as measured by maximum fitness, varied substantially between runs; A plot of the mean of the maximum fitness for each set of runs may be seen in Figure 7.1. Some examples of trusses from these runs are shown in Figure 7.2.

Speaking informally, there seems to exist an inverse relation between the maximum fitness achieved and both of: (a) the complexity of the environment (as measured by, say, number of corners), and (b) the amount of space in the path for growth. For instance, both *thick* and *thin* do well due to both a wide path and low complexity — the failure of *zigzag-s* is likely due to both high complexity, and a too-narrow path vertically (only three meters of space for vertical connections).

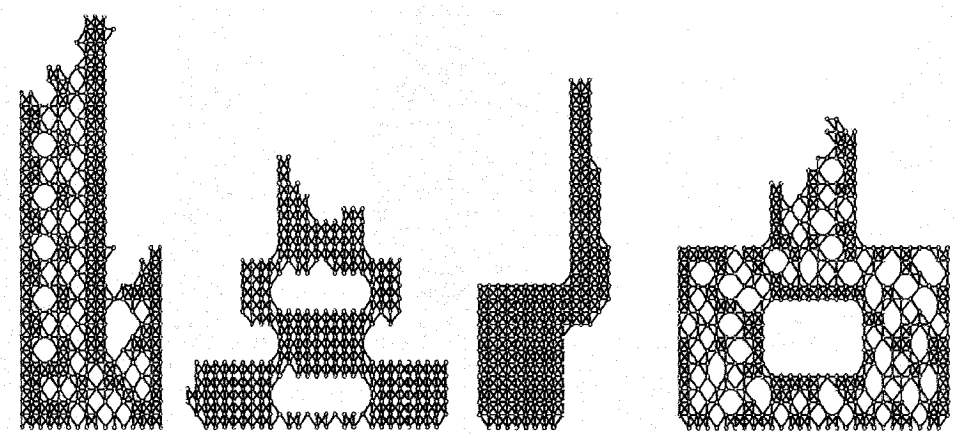


Figure 7.2: Examples of trusses evolved in the environments: (*left to right*) *thin*, *bulbs*, *swerve-r*, and *bulb*.

7.2 Re-use of Genetic Material

In the following experiments, we explore the re-use of genetic material in an environment other than the one for which it was evolved.

7.2.1 Re-growth at Different Sizes

An additional set of runs of the *zigzag-l* environment were undertaken, ten with $r_c = 40$, and an additional ten with $r_c = 30$; We shall refer to these sets as *r.zigzag-l-30* and *r.zigzag-s-40*. These sets afford us opportunity to view the re-growth of populations of agents at a different size of environment then the one in which they were evolved. Each of the populations from the above runs were re-grown using the alternate value of r_c; By re-grown, we mean that the population at GA generation 100 was re-developed and re-evaluated using the new value of r_c. The results are summarized in Figure 7.4; Additionally, data for the expected performance of randomly generated genomes (over 2000 genomes) are also plotted. An example of the re-growth of the maximum fit genome from generation 100 of run *r.zigzag-l-30.3* is shown in Figure 7.3.

Is it evident from Figure 7.4 that genomes evolved in a different size perform better, in terms of fitness and proportion of stable agents, than random genomes — the comparison for the $r_c = 30$ *zigzag-l* trials was an expected maximum fitness of 0.225 versus 0.303, or 135% performance for the latter (these values encompassing the difference between finding a load-bearing truss and not); Similar results are true for mean and proportion in both the $r_c = 30$ and $r_c = 40$ experiments. A more meaningful comparison is between the evaluation of genomes evolved at the present size, and those transplanted from a different size. In the $r_c = 30$ trials, the original agents have an expected maximum fitness of approximately 0.341 versus 0.303 for the transplanted agents, meaning 113% the performance for the original. In the $r_c = 40$ trials, results are much closer: an expected maximum fitness of 0.293 for the original, versus 0.277 for the transplanted genomes, or 105% the performance for the original. Indeed, in the $r_c = 40$ case, the mean fitness of proportion of stable agents was higher for the transplanted genomes than the originals.

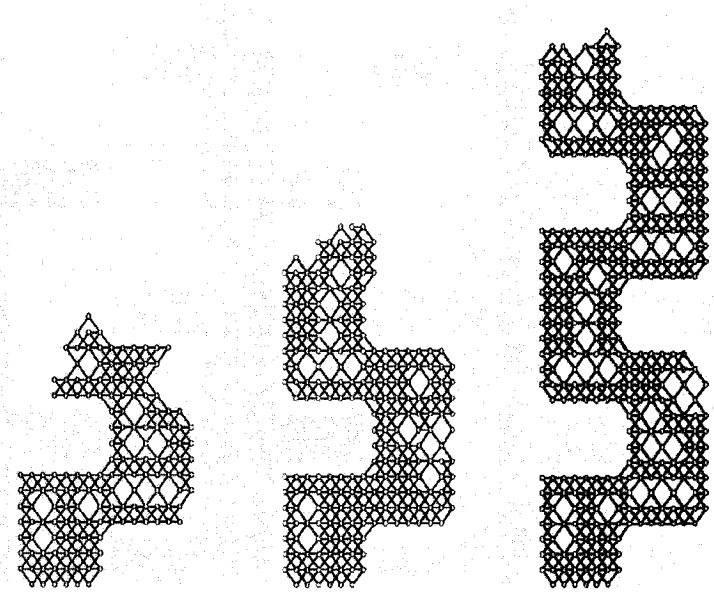


Figure 7.3: Re-growth of the maximum fit genome from generation 100 of the run *run.zigzag-l-30.3* with (*left to right*) $r_c = 30$ (original), $r_c = 40$ and $r_c = 60$.

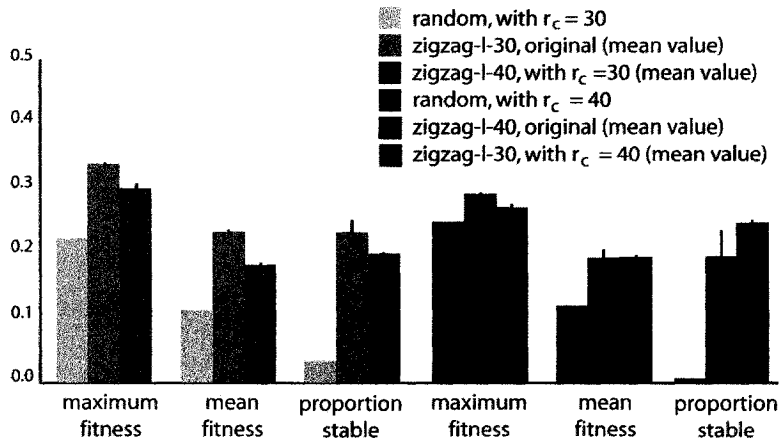


Figure 7.4: Selected statistics from random genomes and the *r.zigzag-l-30* and *r.zigzag-l-40* runs, both originals and re-growth at $r_c = 30$, 40.

7.2.2 Re-growth in Different Environments

For the *r.thick* and *r.bulb* runs from Section 7.1.2, the maximum fitness population (at GA generation 100) was chosen for further experimentation in different environments; These two particular environments were chosen as examples of a simple and a complex environment, by the informal standard of number of corners. The re-growth experiments were accomplished by re-developing the population members using the same parameters, but a different environment. Mean data from these runs is summarized in Figure 7.6, compared to both the maximum fitness of a set of 2000 random genomes, and against the mean maximum fitness obtained from the initial runs in Section 7.1.2. A more detailed view of the data may be seen in Tables 7.1 and 7.2.

Performance of the first set, the *r.thick* trials, showed some improvement over the maximum

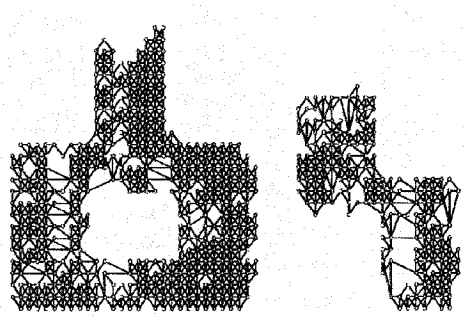


Figure 7.5: Growth of the maximum fit genome from generation 100 of the run *run.bulb.1* in (*left*) its original environment and (*right*) re-grown in the *swerve-l* environment.

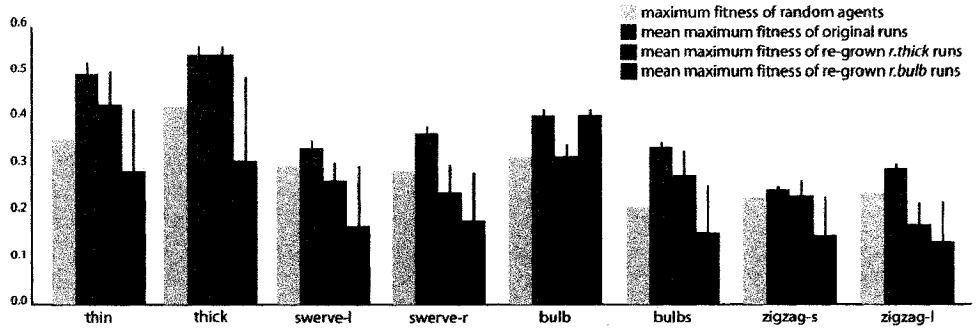


Figure 7.6: Selected statistics from random genomes and the *r.thick* and *r.bulb* runs, both originals and re-growth in a variety of environments

fitness of the random set of agents in most environments. In five out of eight environments, the fitness of re-grown genomes exceeded that of the best of the set of 2000 random genomes. Performance of the second set, the *r.bulb* trials, was more disappointing overall, with most mean values falling below the maximum fitness found during the search of random genomes. This is due largely to several genomes, particularly those from runs with indices 1, 2 and 4, who performed very poorly in any foreign environment — likely, the relative complexity of the *bulb* environment invites over-specialization during evolution. One such unsuccessful re-growth is shown in Figure 7.5. However, there exist several genomes which have performed admirably in all environments, in some cases exceeding the mean maximum fitness of the original sets; One such example is the maximum fitness agent from run *r.bulb.2*, hereafter "Zelig"[1], illustrated in Figure 7.7.

[1]After the Woody Allen character who would adapt to any social environment.

Table 7.1: Maximum fitness from the re-growth of the *run.thick* trials (maximum fit agent at generation 100) in different environments.

orig. pop.	thin	thick	swerve-l	swerve-r	bulb	bulbs	zigzag-s	zigzag-l	nothing
r.thick.0	0.463	0.545	0.358	0.279	0.344	0.327	0.176	0.255	0.073
r.thick.1	0.399	0.557	0.311	0.281	0.281	0.323	0.224	0.136	0.509
r.thick.2	0.460	0.522	0.232	0.279	0.345	0.342	0.240	0.206	0.525
r.thick.3	0.524	0.536	0.234	0.234	0.290	0.251	0.263	0.196	0.073
r.thick.4	0.468	0.567	0.231	0.231	0.341	0.303	0.241	0.114	0.073
r.thick.5	0.300	0.507	0.273	0.241	0.335	0.313	0.248	0.195	0.514
r.thick.6	0.510	0.548	0.235	0.319	0.292	0.179	0.263	0.115	0.544
r.thick.7	0.495	0.567	0.238	0.239	0.297	0.302	0.193	0.123	0.073
r.thick.8	0.384	0.520	0.313	0.074	0.338	0.275	0.282	0.189	0.502
r.thick.9	0.299	0.507	0.256	0.240	0.335	0.178	0.235	0.208	0.528

Table 7.2: Maximum fitness from the re-growth of the *run.bulb* trials (maximum fit agent at generation 100) in different environments.

orig. pop.	thin	thick	swerve-l	swerve-r	bulb	bulbs	zigzag-s	zigzag-l	nothing
r.bulb.0	0.345	0.489	0.049	0.072	0.403	0.052	0.134	0.052	0.544
r.bulb.1	0.090	0.076	0.053	0.072	0.404	0.043	0.061	0.052	0.073
r.bulb.2	0.442	0.493	0.324	0.293	0.424	0.342	0.048	0.253	0.552
r.bulb.3	0.350	0.457	0.055	0.293	0.412	0.049	0.064	0.055	0.555
r.bulb.4	0.349	0.071	0.049	0.072	0.395	0.199	0.249	0.053	0.545
r.bulb.5	0.357	0.470	0.226	0.074	0.422	0.051	0.255	0.048	0.562
r.bulb.6	0.083	0.074	0.057	0.297	0.395	0.297	0.060	0.187	0.071
r.bulb.7	0.086	0.485	0.225	0.283	0.408	0.044	0.245	0.218	0.544
r.bulb.8	0.347	0.071	0.336	0.072	0.389	0.259	0.248	0.214	0.545
r.bulb.9	0.433	0.420	0.326	0.290	0.424	0.225	0.134	0.227	0.559

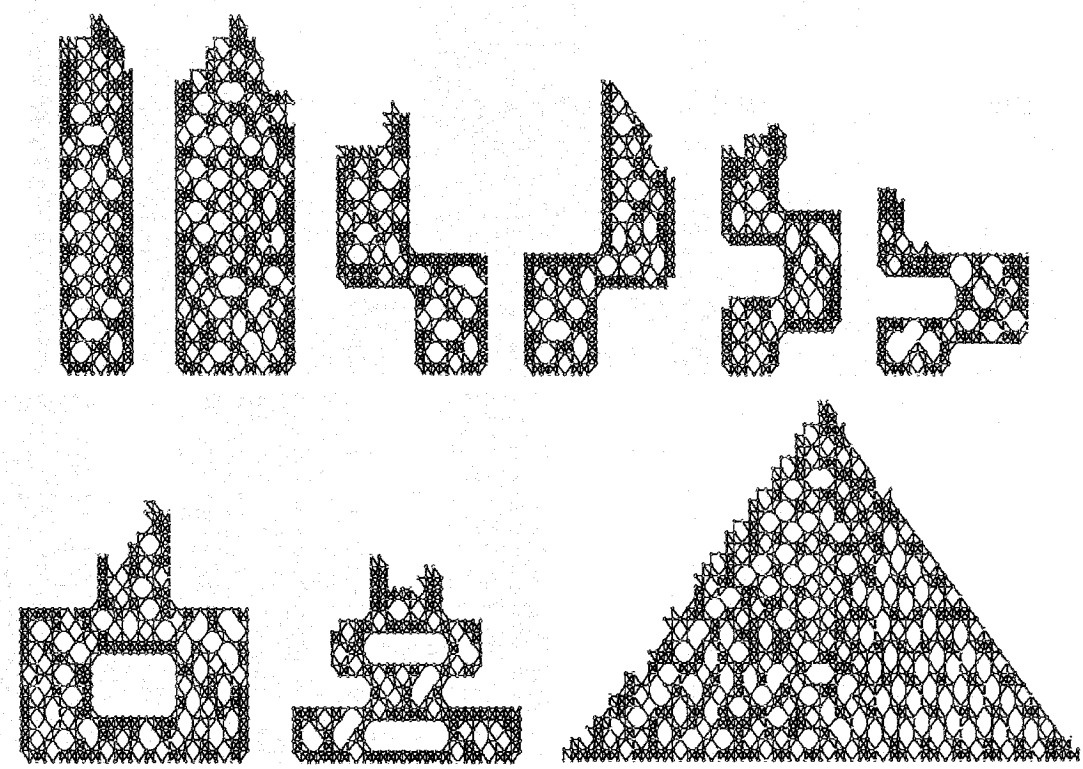


Figure 7.7: Re-growth of the maximum fit genome from generation 100 of the run *run.bulb.2* (Zelig) in all environments.

7.2.3 Zelig

The maximum fit agent from the run *r.bulb.2*, named Zelig, is here explored further. Zelig has been chosen not only for its ability to grow in many environments, but also for having a relatively short and simple genome (in terms of the number of rules activated during growth as a portion of $|\phi|$). We aim to informally describe Zelig's developmental dynamics. Note that the name Zelig points to the genome, not the phenotype, so Zelig may be instantiated several times under the same name (though we would probably not recommend this nomenclature for human twins).

A snapshot of Zelig's growth in the *nothing* environment may be seen in Figure 7.8; From these (and other) images, it is possible to loosely describe Zelig's growth strategy:

- Grow a maximal boundary that expands to fit any available space.
- Any cells on the boundary remain where they are.

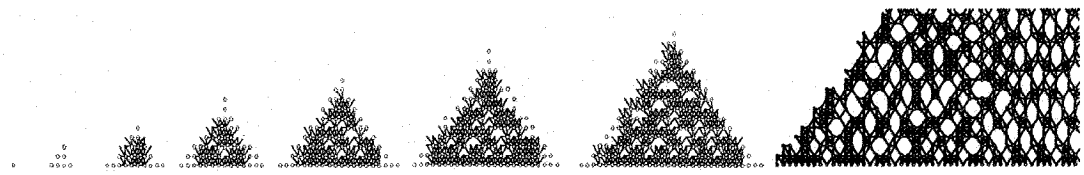


Figure 7.8: Growth of Zelig in the *nothing* environment at times (*left to right*): 0, 5, 10, 15, 20, 25, 30, and a portion of the final trimmed truss at time 84.

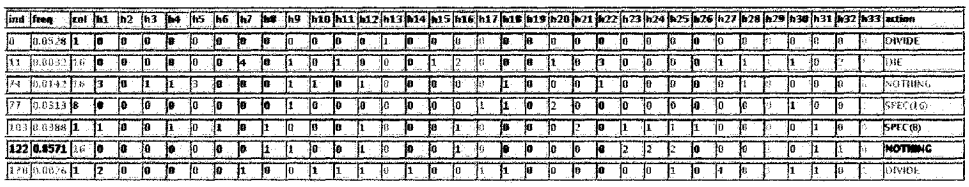


Figure 7.9: A visualization of the activated portions of the Zelig genome, activation from growth in the *bulb* environment.

- Cells on the interior get thinned out.
- The interplay between division and specialization leaves rough boundaries at times.

A more detailed description of Zelig's dynamics is, of course, available from the genome. In considering the genome, we need only consider activated rules, that is, rules which returned actions during the developmental process; In Zelig's case, this was a relatively small eight. A visualization of those eight rules is shown in Figure 7.9, along with their activation frequency. Below, we provide a natural language description of those rules, described not as pattern-matchers, but instead as approximate rules. In brackets is the frequency of activation, as a portion of all activations in the developmental process.

- Rule 0 (0.05): If I am of type 1, and my neighbourhood is approximately empty, then "divide"
- Rule 170 (0.00): If I am of type 1, and my neighbourhood contains cells of type 1 and 16, then "divide"
- Rule 103 (0.04): If I am of type 1, and my neighbourhood contains many cells of type 1, then "specialize" to type 8.
- Rule 77 (0.03): If I am of type 8, then specialize to type 16
- Rule 74 (0.02): If I am of type 16, and my neighbourhood contains many cells of type 1, but few cells of type 8 or 16, then do "nothing".
- Rule 122 (0.86): If I am of type 16, and my neighbourhood contains no cells of type 1, and a small number of cells of type 8 and 16, then do "nothing"
- Rule 11 (0.00): If I am of type 16, and my neighbourhood has a large number of cells of type 16, but no cells of type 1 or 8, then "die".

A detailed description of the developmental dynamics in the *thin* environment, parsed in terms of the above rules, may be seen in Figure 7.10.

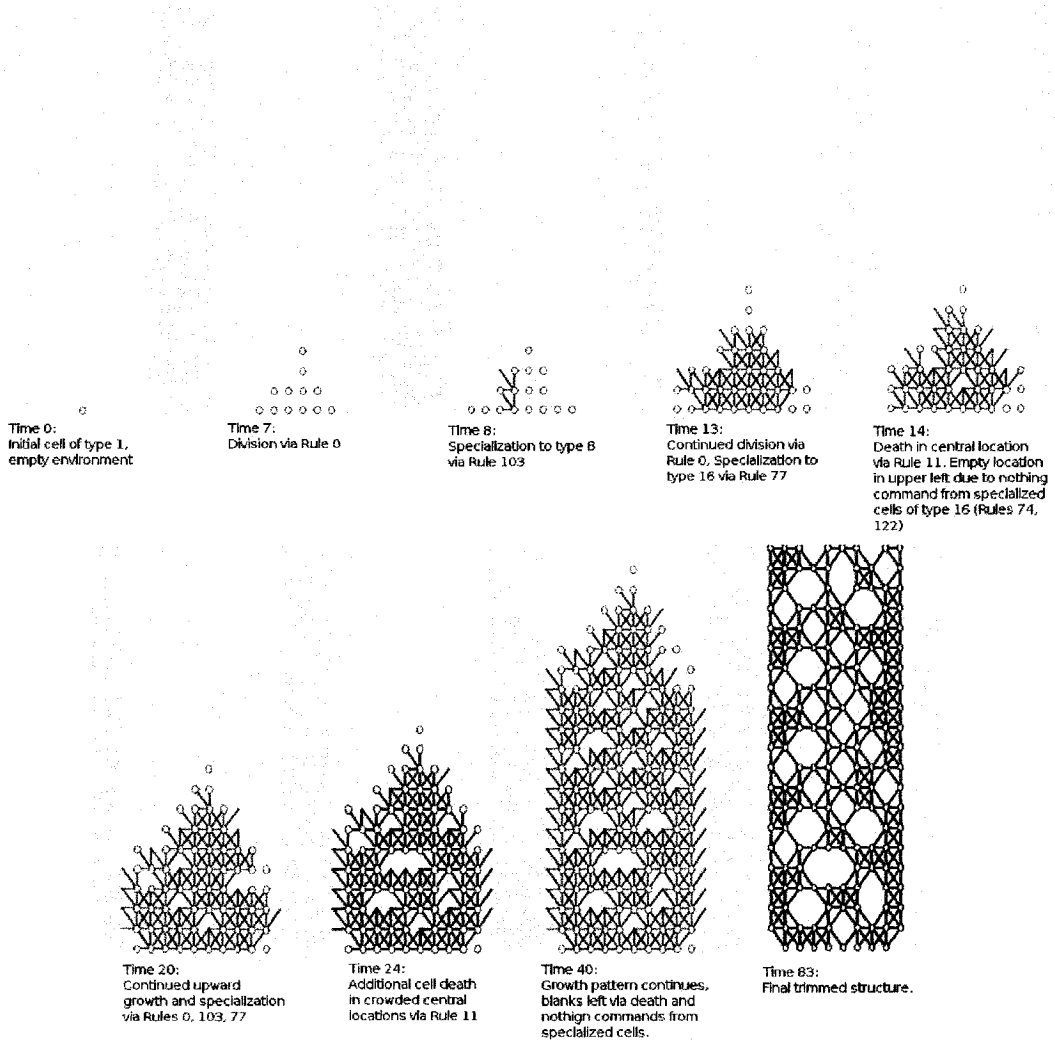


Figure 7.10: A detailed description of the growth of Zelig in the *thin* environment.

7.3 Summary

The primary purpose of these experiments has been to show that environment may be used as a spatial constraint in the design of structural form, using AE. Indeed, for several diverse and non-trivial environments, this has been shown to be the case.

Further, we explored, through additional experiments, the possibility of re-use of genomes from a particular environment in different settings. This is both a means of evaluating the general claim that AE growth is robust and resistant to environmental perturbations, and also a means of demonstrating that AE genomes may be re-used in different contexts without re-running the evolutionary process.

The experiments involving re-growth at different values of r_c showed that, in the *zigzag-l* environment, the fitness of re-grown agents far exceeds that of random genomes, and approaches the fitness of genomes evolved at the new size in question. Although we must hesitate before extending these results to all sizes and environments, it seems likely that genomes evolved at some particular size have utility when re-grown at different sizes.

The re-growth experiments involving different environments were less successful as those involving phenotypic size; This was largely due to several genomes which failed to perform well in any environment other than the one in which they evolved, suggesting a measure of over-specialization. In these final experiments, however, a set of genomes capable of developing into high fitness trusses in nearly any environment were found; These latter agents suggest the existence of a general toolkit which might be useful for a wide class of AE design problems. Analysis of these genomes may, in the future, help to design better algorithms for AE.

The experiments in re-growth generally raise the possibility that genomes evolved via AE may be re-used in slightly different contexts, notably contexts of different phenotypic size or different environment, without significant re-design; This strengthens the claim that growth through AE in general is a robust and perturbation-resistant means of automated design.

Chapter 8

Conclusions

Artificial Embryogeny is a new field, indeed, one with a great deal of potential, but also one which requires a great deal of work. Fundamental concepts are referred to only vaguely, and often contradictory notions masquerade under the same terminology. Indeed, a crisp definition of the field has thus far eluded practitioners, and there is no clear distinction between AE and the nascent fields of Evolvable Hardware or Genetic Programming, nor any explanation of why evolution should be used at all. Add to this the existence of a plethora of models, and no standard test problems save ones which could be solved entirely trivially through non-AE techniques.

It has been our aim to help formalize some of the fundamental theoretical concepts that govern AE, as opposed to other sub-genres of Machine Learning. One of the concepts we have helped to elucidate is that of "complexity", not a single term, but in fact, a family. Relying solely on implicit definition of this term has allowed AE practitioners to naïvely assume that evolution functions via a process of complexification, a highly controversial notion in the world of Biology. A second, and related, fundamental concept is the description of AE through concentration on the dynamical systems aspect. This concentration immediately communicates to the practitioner that the resulting pattern is not simply the result of a genome, but also contains information provided by both the dynamic, and the environment in which the dynamic is instantiated. To paraphrase Peter Grogono, a dinosaur genome is not sufficient to create a new dinosaur — one also needs a mother.

Initially our concentration was on measures of complexity for AE; We have proposed several new measures, including *organplexity*, environmental Kolmogorov complexity, and *funcplexity*, these measures encapsulating organismal, genomic, and functional notions respectively. These measures were applied to a highly simplified model of AE, the Terminating Cellular Automaton, where the results were contrasted for some small examples. It was readily shown that those measures do not agree with each other, nor particularly with other existing measures, reinforcing the notion that there does not exist any single notion of complexity. To evaluate a theory such as complexification, it would be necessary to first choose the appropriate measure, then perform the requisite experiments. Further, these new measures of complexity accept the environment and the dynamic "for free", that is, they use a genomic measure which discounts any information added by dynamics and environment. To summarize this reasoning, we note that a practitioner

should not be interested in the information that a genome contains *about* its environment, but instead, on the capacity of such a genome to *exploit* the information provided by the environment during development. It is through this means that we suspect the paltry thirty thousand genes of the human genome can encode the positional information of a hundred trillion cells.

Next, our concentration turned to the imposition of a notion of meaning to our experiments; Rather than utilize simple abstract and easily solved problems for experimentation, we chose to define a non-trivial domain of application by which the developmental stage could be meaningfully evaluated. A model from Structural Design was chosen, providing the discipline-independent notions of stability and distribution of pressure; In implementing this domain of application, a high-dimensional phenotypic space was required (one with many cell types and an intricate connectivity), as we suspect that it would be for any sufficiently interesting problem. This forced us to consider the problem of scaling up, and whether our results would apply for a problem of complexity approaching the ambitions of our discipline.

The Deva family was introduced as a modular family of evolvable AE algorithms. It was shown that some Deva algorithms have the capacity to successfully evolve high fitness trusses, that is, trusses capable of reaching tall heights while supporting external load. It was shown that different objective functions push Machine Learning to different but interesting attractors; But, as is typical in Evolutionary Computation, it was seen that there existed a tendency for evolution to find "loopholes", sometimes maximizing fitness in unintended ways.

Given our external methodology for evaluation, we have a means of evaluating organisms. This gives us the means to evaluate model-level strategies in the context of the evolution of structural form, through empirical comparative study. We did so, evaluating many different model-level decisions, using our fitness functions as overall measures of success. Generally, we found that the optimum was defined by the phenotypic representation and the fitness function, and that the model-level decisions could only improve efficiency or sabotage evolution relative to our original models. However, we discovered that some model-level decisions would greatly affect the efficiency and reliability of evolution. We reinforced the notion that low-level mechanisms can create exploitable phenotypic regularities, as was the case with quasi-symmetry; We also showed that those regularities can be achieved through other means as well, as was the case with the inheritance of cell specialization. And finally, we reinforced the notion that evolutionary search selects at the phenotypic level, but operates on the genomic level: simplifications which had little impact on the genotypic space were of little value (action-based changes), while decisions which had great impact on genotypic space were of paramount importance (neighbourhood-based changes).

Finally, we further explored the use of environment as a means of constraining the growth of an organism; Not only do we aim for a constraint here, but also at the "fruitful channelling" that characterizes biological growth, which allows the dynamic to exploit environmental information while expressing a phenotype. Firstly, we showed that organisms may be evolved in a variety of geometric environments, allowing for precise constraints which would be difficult to achieve through a fitness function; Secondly, and more interestingly, we showed that organisms evolved in particular environments could be re-grown at different phenotypic sizes, and in different environments, without having been selected to do so. This suggests that *some* genomes codified programs which used a great deal of feedback from their environment in order to grow successful

designs (although others were overspecialized to the environment in which they evolved). We analyzed one such adaptable organism, Zelig, and discovered that its program was shorter than average, and its strategy relatively understandable.

Theories such as complexification tell us that we wish to constantly increase genomic complexity in order to increase phenotypic complexity; However, there exist a plethora of biological examples which demonstrate that this need not necessarily be the case, that genetic material often radically changes its role during evolution without significant genetic change. It is our belief that genomic measures like eK will play a larger role in the future, allowing researchers to (a) quantify the difficulty of finding a particular pattern, since eK will describe the minimal string needed to be discovered by a Machine Learning technique, and (b) by helping to formalize Occam's Razor, the notion that it is not the larger genome which is desired, but instead, typically, the smaller which achieves the same result.

Bibliography

[Ada02] C. Adami. What is complexity? *BioEssays*, 24:1085–1094, 2002.

[Alt94] L. Altenberg. Evolving better representations through selective genome growth. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1994.

[Art04] W. Arthur. *Biased Embryos and Evolution.* Cambridge University Press, 2004.

[Ban04] W. Banzhaf. On evolutionary design, embodiment, and artificial regulatory networks. In F. Iida *et al*, editor, *Embodied Artificial Intelligence, LNAI.* Springer-Verlag, 2004.

[BCFV95] G. Braga, G. Cattaneo, P. Flocchini, and C. Vogliotti. Pattern growth in elementary cellular automata. *Theoretical Computer Science*, 145(1-2):1–26, 1995.

[Ben86] C. H. Bennet. On the nature and origin of complexity in discrete, homogeneous, locally interacting systems. *Foundations of Physics*, 16(585), 1986.

[BF02] V. Becher and S. Figueira. An example of a computable absolutely normal number. *Theoretical Computer Science Archive*, 270(1-2):947–958, 2002.

[BKM97] F. Blanchard, P. Kurka, and A. Maass. Topological and measure-theoretic properties of one-dimensional cellular automata. *Physica D*, 103(1-4):86–99, 1997.

[BM97] G. Bell and A. Mooers. Size and complexity among multicellular organisms. *Biological Journal of the Linnean Society*, 60:345–363, 1997.

[BM04] W. Banzhaf and J. Miller. The challenge of complexity. In A. Menon, editor, *Frontiers in Evolutionary Computation.* Kluwer Academic, 2004.

[BMBH04] D. Basanta, M. Miodownik, P. Bentley, and E. Holm. Evolving automata to grow patterns. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, 2004.

[BMRT03] M. Bissel, S. Mian, D. Radisky, and E. Turley. Tissue specificity: Structural cues allow diverse phenotypes from a constant genome. In G. Müller and S. Newman, editors, *Origination of Organismal Form: Beyond the Gene in Developmental and Evolutionary Biology.* MIT Press, 2003.

[Bro07] Nicolas Brodu. *Practical Investigations of Complex Systems.* Ph.D. in computer science, Concordia University, 2007.

[Car05] S. Carroll. *Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom.* W. W. Norton and Company, 2005.

[CDF+01] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems.* Princeton University Press, 2001.

[CGW05] S. Carroll, J. Grenier, and S. Weatherbee. *From DNA to Diversity: Molecular Genetics and the Evolution of Animal Design, 2nd Ed.* Blackwell Publishing, 2005.

[Cha98] G. Chaitin. *The Limits of Mathematics.* Springer-Verlag, 1998.

[Coa97] P. Coates. Using genetic programming and l-systems to explore 3d design worlds. In Junge, editor, *CAADFutures'97*, 1997.

[Coo04] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.

[CY88] K. Culik and S. Yu. Undecidability of ca classification schemes. *Complex Systems*, 2(177), 1988.

[Daw96] R. Dawkins. *Climbing Mount Improbable.* W.W. Norton & Company, 1996.

[Daw04] R. Dawkins. *The Ancestor's Tale: A Pilgrimage to the Dawn of Evolution.* Houghton Mifflin, 2004.

[DB96] F. Dellaert and R. Beer. Developmental model for the evolution of complete autonomous agents. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour*, 1996.

[DD05] A. Deutsch and S. Dormann. *Cellular Automaton Modelling of Biological Pattern Formation: Characterization, Applications and Analysis.* Birkauser, 2005.

[ES03] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003.

[FD06] D. Federici and K. Downing. Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. *Artificial Life*, 12(3):381–409, 2006.

[Fen02] T. Fenchel. *Origin and Early Evolution of Life.* Oxford Biology, 2002.

[FK98] C. Furusawa and K. Kaneko. Emergence of rules in cell society: Differentiation, hierarchy, and stability. *Bulletin of Mathematical Biology*, 1998.

[FP97] P. Funes and J. Pollack. Computer evolution of buildable objects. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 358–367, 1997.

[GMS03] A. Gal, G. Mahal, and M. Sipper. Evolutionary plantographics. *Artificial Life*, 9(2):191–205, 2003.

[Gou02] S. J. Gould. *The Structure of Evolutionary Theory.* The Belknap Press of Harvard University Press, 2002.

[GV06] P. Grünwald and P. Vitányi. Shannon information and Kolmogorov complexity. *arXiv:cs.IT/0410002*, 2006.

[HGP03] P. Eggenberger Hotz, G. Gomez, and R. Pfeifer. Evolving the morphology of a neural network for controlling a foveating retina and its test on a real robot. In *Artificial Life VIII: 8th Int. Conf. on the Simulation and Synthesis of Living Systems*, 2003.

[HM06] S. Harding and J. Miller. The dead state: A comparison between direct and developmental encodings. In *GECCO*, 2006.

[Hor05] G. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *GECCO*, 2005.

[Hot97] P. Eggenberger Hotz. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213. Elsevier Academic Press, 1997.

[Hot03] P. Eggenberger Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In S. Kumar and P. Bentley, editors, *On Growth, Form and Computers*, pages 302–318. Elsevier Academic Press, 2003.

[Ila01] A. Ilachinski. *Cellular Automata: A Discrete Universe.* World Scientific, 2001.

[KAJ05] R. Kicinger, T. Arciszewski, and K. A. De Jong. Evolutionary computation and structural design: a survey of the state of the art. *Computers & Structures*, 83(23-24):1943–1978, 2005.

[Kau00] S. Kauffman. *Investigations.* Oxford University Press, 2000.

[KB03] S. Kumar and P. Bentley. *On Growth, Form and Computers.* Elsevier Academic Press, 2003.

[KGK04] T. Kowaliw, P. Grogono, and N. Kharma. Bluenome: A novel developmental model of artificial morphogenesis. In *Genetic and Evolutionary Computation GECCO-2004*, 2004.

[KO01] H. Kawamura and H. Ohmori. Computational morphogenesis of discrete structures via genetic algorithms. *Memoirs of the School of Engineering, Nagoya University*, 53(1/2):28–56, 2001.

[Kow03] T. Kowaliw. Bluenome: A novel developmental model for the evolution of artificial agents. Master's thesis, Concordia University, 2003.

[LD03] O. Leyser and S. Day. *Mechanisms in Plant Development.* Blackwell Publishing, 2003.

[LH04] P. Lehre and M. Hartmann. Development and complexity-based fitness function modifiers. In *Workshop on Regeneration and Learning in Developmental Systems*, 2004.

[Li88] W. Li. Power spectra of regular languages and cellular automata. *Complex Systems*, 1(1), 1988.

[Lin68] A. Lindenmayer. Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, 18:280–315, 1968.

[LV93] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 2nd Ed.* Springer, 1993.

[LW97] C. J. Lowe and G. A. Wray. Radical alterations in the roles of homeobox genes during echinoderm evolution. *Nature*, 389:718–721, 1997.

[MÖ3] G. Müller. Homology: The evolution of morphological organization. In G. Müller and S. Newman, editors, *Origination of Organismal Form: Beyond the Gene in Developmental and Evolutionary Biology.* MIT Press, 2003.

[McS91] D. W. McShae. Complexity and evolution: What everybody knows. *Biology and Philosophy*, 6:303–324, 1991.

[McS00] D. W. McShae. Functional complexity in organisms: Parts as proxies. *Biology and Philosophy*, 15:641–668, 2000.

[Meg05] T. H. G. Megson. *Structural and Stress Analysis, 2nd Ed.* Elsevier Butterworth Heinmann, 2005.

[MG05] R. M. H. Merks and J. A. Glazier. A cell-centred approach to developmental biology. *Physica A*, 352:113–130, 2005.

[MH05] S. Macgregor and I. Harvey. Embracing plagiarism: Theoretical, biological and empirical justification for copy operators in genetic optimization. *Genetic Programming and Evolvable Machines*, 6-4:407–420, 2005.

[Mil04] J. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *GECCO*, 2004.

[Mit98] M. Mitchell. Computation in cellular automata: A selected review. In T. Gramss, M. Bornholdt, M. Mitchell, and T. Pellizzari, editors, *Nonstandard Computation*, pages 95–140. VCH Verlagsgesellschaft, 1998.

[OOO01] G. Olivera, P. Olivera, and N. Omar. Definition and application of a five parameter characterization of a one-dimensional cellular automata rule space. *Artificial Life*, 7(3):277–302, 2001.

[PL90] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[PRL06] P. Prusinkiewicz and A. Rolland-Lagan. Modeling plant morphogenesis. *Current Opinion in Plant Biology*, 9:83–88, 2006.

[PW85] N. Packard and S. Wolfram. Two-dimensional cellular automata. *Journal of Statistical Physics*, 38:901–946, 1985.

[Raj95] S. D. Rajan. Sizing, shape and topology design optimization of trusses using a genetic algorithm. *Journal of Structural Engineering*, 121:1480–1487, 1995.

[RP04] J. Rieffel and J. Pollack. The emergence of ontogenic scaffolding in a stochastic development environment. In *GECCO*, 2004.

[SB05] L. Sekanina and M. Bidlo. Evolutionary design of arbitrarily large sorting networks using development. *Genetic Programming and Evolvable Machines*, 6(3):319–347, 2005.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379?423 & 623?656, 1948.

[Sha81] R. Shaw. Strange attractors, chaotic behaviour, and information flow. *Zeitschrift fur Naturforschung*, 36A(80), 1981.

[Sim99] K. Sims. Evolving three-dimensional morphology and behaviour. In P. Bentley, editor, *Evolutionary Design by Computers*, pages 1–10. Morgan Kauffman, 1999.

[SM03] K. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[SN04a] K. Stoy and R. Nagpal. Self-reconfiguration using directed growth. In *Proc. 7th Int. Symp. on Distributed Autonomous Robotic Systems*, pages 1–10, 2004.

[SN04b] K. Stoy and R. Nagpal. Self-repair through scale independent self-reconfiguration. In *Proceedings of IEEE/RSJ International Conference on Robots and Systems (IROS)*, pages 2062–2067, 2004.

[SSH04] C. Shalizi, K. Shalizi, and R. Haslinger. Quantifying self-organization with optimal predictors. *Phys. Rev. Lett.*, 93(118701), 2004.

[Sta06] K. Stanley. Exploiting regularity without development. In *Proceedings of the 2006 AAAI Fall Symposium on Developmental Systems*. AAAI Press, 2006.

[Tur52] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

[vNB66] J. von Neumann and A. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[Wal70] A. Wallace. *Contributions to the Theory of Natural Selection: A Series of Essays*. Macmillan, 1870.

[Wel84] T. A. Welch. A technique for high-performance data compression. *Computer*, 17:8–19, 1984.

[Wes89] H. H. West. *Analysis of Structures: An Integration of Classical and Modern Methods*. John Wiley and Sons, 1989.

[Wil84] S. Wilson. Growth rates and fractional dimensions in cellular automata. *Physica D*, 10:69–74, 1984.

[Wol83] S. Wolfram. Cellular automata. *Los Alamos Science*, 9:2–21, 1983.

[Wol02] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[Wue99] A. Wuensche. Classifying cellular automata automatically. *Complexity*, 4(3):47–66, 1999.

[YS02] Y. Yang and C. K. Soh. Automated optimum design of structures using genetic programming. *Computers & Structures*, 80(18-19):1537–1546, 2002.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24(5), 1978.

[ZRL06] E. Zamir, P. Rupp, and C. Little. Studying in vivo dynamics of vasculogenesis using time-lapse computational imaging. In R. Forroughs, editor, *New Frontiers in Angiogenesis*, pages 31–42. Springer-Verlag, 2006.

Appendix A

Complexity Addenudum

Additional materials for our discussion of measure of complexity.

Table A.1 summarizes the correlations between our various measures, listing the associated p-value.

Below, we include proofs of some of the claims regarding TCAs.

Theorem A.0.1 *For every pattern $p = (p_1, ..., p_l)$, there exists some TCA α such that $\alpha \vdash_{TCA} p$.*

Proof We define a TCA α which fits the bill. We assume $l > 2$, proof is trivial otherwise. Let $\alpha_t = 1$ and $\alpha_d = l$. Consider each point on the lattice at time $t = 0$: for each location x_i, we

Table A.1: Correlation values between complexity measures. Notation: $\times$ — versus; f_1 — $funcplexity(O_1)$; f_2 — $funcplexity(O_2)$; Z — LZW compression; $eK \cdot D$ — the product of normalized eK and normalized D.

l	dof	$eK \times H$	$p \leq$	$D \times H$	$p \leq$	$Z \times H$	$p \leq$	$f_1 \times H$	$p \leq$	$f_2 \times H$	$p \leq$
5	30	.733	.0001	.441	.0074	.502	.0024	.733	.0001	.530	.0013
6	62	.693	.0001	.117	.269	.495	.0027	.702	.0001	.593	.0003
7	126	.710	.0001	.187	.018	.521	.0001	.566	.0001	.421	.0001
8	254	.624	.0001	.069	.1366	.450	.0001	.671	.0001	.543	.0001
9	510	.526	.0001	.077	.0412	.475	.0001	.304	.0001	.433	.0001
10	1022	.400	.0001	.075	.0082	.428	.0001	.628	.0001	.554	.0001
11	2046	.366	.0001	.010	.3256	.444	.0001	.200	.0001	.282	.0001
12	4094	.255	.0001	-.004	.3587	.396	.0001	.701	.0001	.309	.0001
13	8190	.280	.0001	-.137	.0001	.393	.0001	.144	.0001	.193	.0001
14	16382	.016	.0200	.085	.0001	.435	.0001	.632	.0001	.396	.0001
15	32766	.150	.0001	-.006	.1387	.388	.0001	.510	.0001	.338	.0001

l	$eK \times Z$	$p \leq$	$eK \times f_1$	$p \leq$	$eKxf_2$	$p \leq$	$eK \cdot D \times H$	$p \leq$
5	.338	.0339	1.00	.0001	.787	.0001	.464	.0049
6	.241	.0998	.567	.0005	.571	.0005	.286	.0627
7	.290	.0005	.508	.0001	.474	.0001	.293	.0004
8	.397	.0001	.550	.0001	.480	.0001	.132	.0178
9	.302	.0001	.153	.0003	.324	.0001	.089	.0223
10	.229	.0001	.300	.0001	.524	.0001	.079	.0058
11	.270	.0001	.219	.0001	.142	.0001	.035	.0567
12	.156	.0001	.116	.0001	.572	.0001	.004	.3990
13	.184	.0001	.215	.0001	.075	.0001	-.109	.0001
14	.006	.221	.064	.0001	.157	.0001	.085	.0001
15	.069	.0001	.167	.0001	.784	.0001	-.005	.1827

may describe its neighbourhood, $\mu(x_i)$:

$$\mu(x_i) = (x_{i-\frac{l}{2}}, ..., x_{i-1}, x_i, x_{i+1}, ..., x_{i+\frac{l}{2}})$$

Since $x_j = 1$ for exactly one j (by initialization), we have $\mu(x_i) \neq \mu(x_j)\ \forall i \neq j$.

Now, let's define a rule set for α:

$$\mu(x_1) \rightarrow p_1$$

$$...$$

$$\mu(x_l) \rightarrow p_l$$

with all other neighbourhoods mapping to arbitrarily chosen values from Σ. Clearly, it may be shown point-wise that following one time step, the lattice will contain pattern p. ∎

Theorem A.0.2 *Given some environmental configuration p on environmental space E, and some two-dimensional TCA α such that $\alpha \vdash_{TCA} p$, then $\forall k$ st $\alpha_d < k < |E|$, there exists a TCA α' such that $\alpha' \vdash_{TCA} p$*

Proof Let $k' \in \{\alpha_d + 1, ..., |E|\}$. We construct an α' such that $\alpha'_d = k'$ and computes p. Let $\alpha'_s = \alpha_s$

For each rule in α_ϕ, we define a set of $2^{k'-k}$ rules for our new transition function: Let any particular rule be written

$$\mu(x_1, ..., x_k) \rightarrow y$$

Then, we define a new rule

$$\mu'(x_1, ..., x_k, x_{k+1}, ..., x_{k'}) \rightarrow y$$

for each combination of the boolean values $\{x_{k+1}, ..., x_{k'}\}$.

Since the rules of α_ϕ span all combinations of the initial variables, $\{x_1, ..., x_k\}$, we have a complete rule listing, and hence, a complete transitions function, α'_ϕ.

Now consider some arbitrary environmental configuration, p'. Transition function α_ϕ maps each index in p' to a new value based on a neighbourhood about that index through some rule, say:

$$\mu(p'_i, p'_{j_1}, ..., p'_{j_{k-1}}) \rightarrow p''_i$$

for some indices $j_1, ..., j_{k-1}$. But, regardless of the values in the larger surrounding neighbourhood, by construction, there exists a rule in α'_ϕ such that

$$\mu(p'_i, p'_{j_1}, ..., p'_{j_{k-1}}, p'_{j_k}, ..., p'_{j_{k'-1}}) \rightarrow p''_i$$

Hence, we have that

$$\alpha_\phi(p') = p'' \Rightarrow \alpha'_\phi(p') = p''$$

Since by the definition of TCAs both α and α' begin with the same initial configuration of E, the evolution of *alpha* and α' is identical, showing that $\alpha' \vdash_{TCA} p$. ∎

Appendix B

Explicit Deva Algorithms

In this appendix we outline some explicit algorithms for Deva growth, and the various Deva Models. A summary of the Deva models is contained in Table 4.1 on page 47.

B.1 General Deva Growth Algorithms

The following are several concepts used in the Deva algorithms.

B.1.1 Neighbourhoods

We will frequently refer to neighbourhoods, which for our purposes may be classified into three types: *von Neumann* neighbourhoods, *Moore* neighbourhoods, and *Deva* neighbourhoods; For simplicity in two dimensions, we will refer to them as four-, eight- and twelve-neighbourhoods. See Figure B.1 for illustration in two and three dimensions.

B.1.2 Best Free Location

The *best free location* is a means of choosing a location from a cell's neighbourhood for division (or movement); Loosely, it is intended to mimic the natural growth of a cell. It does so by determining which cell direction has the largest mass surrounding it, then selecting the opposite,

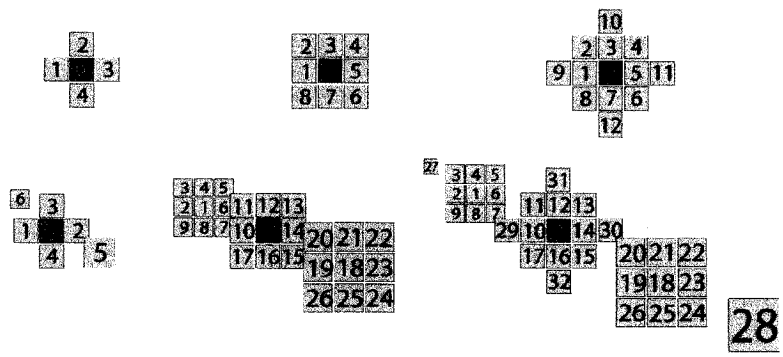


Figure B.1: Neighbourhoods of type (*left*) von Neumann, (*center*) Moore and (*right*) Extended von Neumann, in (*top*) two and (*bottom*) three dimensions.

when the space is free. If there are no free locations, the algorithm returns null; Otherwise, in case of a tie, the algorithm contains a bias. This bias is unavoidable, some axis of symmetry must be broken in discrete cell division on a lattice[1].

The following is an algorithm for the determination of the *best free location* in two and three dimensions: Let the cell of interest be known as the origin, o; We define the direction cells to be the von Neumann neighbours of the origin, D; Hence, D is a set of four cells in two dimensions, and six in three dimensions. For any cell location c, let $full(c)$ be equal to 0 if the cell location is empty, 1 otherwise. Let us call the Moore neighbourhood of the origin N, with N/c to be the set N modulo the cell c for any cell c, and similarly for N/S for any set of cells S.

Next, we define the *tangent* at a direction relative to the origin cell: The tangent set, $T(o,d)$, is defined to be all cells c in the Moore-neighbourhood of the origin, o, such that the line connecting c to d is orthogonal to the line connecting d to o — one can imagine this to be the intersection of the tangent line or plane with the neighbourhood about o. For example, in two dimensions, the tangent to direction "5" is $T(0,5) = \{4,5,6\}$. Note that cells may be elongated — in this case, all directions are relative to the "source" cell.

Firstly, we remove any non-empty directions from the set D; If D is empty following this, we exit, returning null. Otherwise, for any direction $d \in D$, we may define the $OpMass$ to be

$$OpMass(d) = \sum_{c \in N/T(o,d)} full(c)$$

Hence, the $OpMass$ of a direction is a count of all cells not located in the direction location and tangent, i.e. the mass of the opposite direction. For example, in two dimensions, the mass of direction "1" would be the sum $full(3) + full(4) + full(5) + full(6) + full(7)$.

Now, we may define the best free location as the proper maximum, $\max_{OpMass(d)} D$; If there are several maximal values, then we choose amongst them via our *developmental bias*; In two dimensions, this is the prioritized list of directions $\{1,3,5,7\}$; In three dimensions, the prioritized list is $\{10,14,12,18,16,1\}$.

B.1.3 Deva Growth

Given definitions of neighbourhood state descriptions (and distances between them) and a set of actions, we may define a deva growth algorithm; This algorithm is summarized in Algorithm 1 (we detail the algorithm for each cell action below). Note that information is gathered from the preceding developmental space, D_{time-1}, but executed in the current developmental space, D_{time}.

Individual cell actions are detailed individually. For each case, if applicable, we assume that we are considering cell c in developmental space D_{time}, which has sufficient resources, has an available location for action execution, and such an action is legal. Note that in any particular deva implementation, not all listed actions will be present.

- "nothing": does not cause any change.
- "die":

[1]It is quite possible, and probably preferable, to define the "best" direction randomly in cases of a tie; This has not been done here to keep results deterministic.

Algorithm 1 Deva Growth Algorithm

int $time \leftarrow 0$
boolean $hasChanged \leftarrow TRUE$
Initialize developmental space D_0
while hasChanged **do**
  $hasChanged \leftarrow FALSE$
  $time = time + 1$
  developmental space $D_{time} = D_{time-1}$
  for all Active cells c **do**
    Neighbourhood $\mu_c \leftarrow getNeighbourhood(D_{time-1}, c)$
    Action $a_c \leftarrow \phi(\mu_c)$
    if $\exists$ space to execute $a_c \wedge c_{r_c} > cost(a_c, c) \wedge$ execution of a_c is legal **then**
      $c_{r_c} \leftarrow c_{r_c} - cost(a_c, c)$
      Execute(c, a_c, D_{time})
      if $a_c \neq$ "nothing" **then**
        $hasChanged \leftarrow TRUE$
      end if
    end if
  end for
end while

Delete(c, D_{time})

- "divide":

  $c' \leftarrow copy(c)$
  $bestFreeLocation(c) \leftarrow c'$

- "divide(noinher)":

  $c' \leftarrow copy(c)$
  $c'_{colour} \leftarrow 1$
  $bestFreeLocation(c) \leftarrow c'$

- "divide(dir)":

  $c' \leftarrow copy(c)$
  $neighbour(c, dir) \leftarrow c'$

- "divide(allway)":

  for all Direction dir **do**
    $c' \leftarrow copy(c)$
    $neighbour(c, dir) \leftarrow c'$
  end for

- "divide(clockwise)":

  for all $dir = [left, up, right, down]$ **do**
    if $isEmpty(neighbour(c, dir))$ **then**
      $c' \leftarrow copy(c)$
      $neighbour(c, dir) \leftarrow c'$
      Break.
    end if
  end for

- "specialize(x)", where $x \in \{2, ..., n_c\}$:

  $c_{colour} \leftarrow x$

- "on(x)", where $0 \leq x \leq n_g$:

  $c_{colour} \leftarrow c_{colour} + 2^x$

- "off(x)", where $0 \leq x \leq n_g$:

  $c_{colour} \leftarrow c_{colour} - 2^x$

- "move":

  $c' \leftarrow copy(c)$
  $bestFreeLocation(c) \leftarrow c'$
  Delete(c, D_{time})

- "elongate":

  if $\exists$ $elongationDirection(c) \wedge free(location(elongationDirection(c)))$ **then**
    $elongate(c, elongationDirection(c))$
  else if $\exists$ $bestFreeLocation(location(c))$ **then**
    $elongate(c, elongate(bestFreeLocation(location(c))))$
  end if

- "elongate(dir)":

  $elongate(c, dir)$

B.2 Deva 1

The deva implementation *Deva 1* is designed to be as simple, in terms of transition function specification, as possible; This is accomplished through liberal use of the best free location algorithm (See Appendix B.1.2). The set of actions includes at least the set $A = \{die, divide, specialize(2), ..., specialize(n_c)\}$. Based on a system parameter, this set may be augmented with one of $\{move, elongate\}$. Hence, without movement or elongation, $|A| = 1 + n_c$, and with either, $|A| = 2 + n_c$. If a "nothing" action is explicitly included, $|A| = 2 + n_c$ or $|A| = 3 + n_c$.

B.2.1 Deva 1 Transition Function

A *Deva 1* transition function is a listing of descriptions of possible neighbourhoods of a specified length, $|\phi|$. These rules are tuples of the form:

$$(c, h_1, ..., h_{n_c}, a)$$

where c is a colour, a an action, and h_i is a count of the number of neighbours of cell type i, or a *hormone-level*. Hence, the size of the representation of such a transition function is $O(|\phi| \cdot n_c)$, and the number of possible transition functions[2] is $n_c \cdot |\mu|^{n_c} \cdot |A|$, where A is the set of all actions and $|\mu|$ is the size of a deva-neighbourhood (12 in two dimensions).

[2]Note that many transition functions will be functionally equivalent.

Any particular cell may consult its transition function to determine an action appropriate to its neighbourhood. By consulting its neighbourhood, it may construct n_c *environmental-hormone-readings*, or a count of the number of cells of type i in its deva neighbourhood (relative to the source cell); These will be written $\{e_1, ..., e_{n_c}\}$; Next we find the rule in our transition function which closely matches the set of e_i to the set of h_i using Euclidean distance — This process is made precise in Algorithm 2.

Algorithm 2 Deva 1 Transition Function Application Algorithm

```
Current cell c_0
Deva-neighbourhood μ_{c_0}
Environmental-Hormones e_i ← 0 for all e_i ∈ {e_0, ..., e_{n_c}}
for all Cells c ∈ μ_{c_0} do
  int type ← cellType(c)
  e_type ← e_type + 1
end for
Rule r_min = (s_min, a_min) ← (∅, ∅)
float min ← ∞
for all Rules r ∈ φ do
  if r_colour = colour(c_0) then
    float distance ← (e_0 − r_{h_0})^2 + ... + (e_{n_c} − r_{h_{n_c}})^2
    if distance < min then
      min ← distance
      r_min ← r
    end if
  end if
end for
return a_min
```

Genetic Initialization

We would like a means of initializing a set of rules such that hormone values in the rules are likely to match possible neighbourhoods in developmental space. As the integer values may range between 0 and 12, random selection will likely result in an impossible[3] pattern (i.e. one in which $\sum h_i > 12$, in two-dimensions). Hence, we desire a power-law distribution which favours 0:

$$Pr[X = i \mid 0 \leq i \leq 12] = \frac{1}{\sum_{j=0}^{12} \beta^j} \beta^{12-i}$$

We would like the expected value of the generation of any particular rule to be somewhere in the range of $\{0, .., 12\}$, where selecting a larger value is preferable, since the majority of computation will occur in a full neighbourhood; Hence, we take $E[n_c \cdot X] \approx 12$. Solving (numerically) the equation

$$\frac{12}{n_c} = \frac{1}{\sum_{j=0}^{12} \beta^j} \sum_{i=0}^{12} i \cdot \beta^{(12-j)}$$

we obtain $\beta \approx 3.6$ for $n_c = 32$ and $\beta \approx 6.3$ for $n_c = 64$

[3]Impossible for exact matching, that is, and less likely for more reasonable distributions for closest distance-matching.

We also require a probability distribution for possible actions:

$$Pr[Y = a] = \begin{cases} 1/4, & \text{if } a = \text{“divide”} \\ 1/4, & \text{if } a = \text{“die”} \\ 1/4, & \text{if } a = \text{“elongate”} \\ 1/4, & \text{if } a = \text{“specialize”} \end{cases}$$

where all colours of specialization are equally likely, save “0” or “1”, which are excluded. If the “move” action is included in the run, it simply replaces then “elongate” action.

Hence, we may generate a random rule by (uniformly) randomly generating an initial rule colour, then generating n_c hormones and one action according to the above distributions. The same distributions may be used for the genetic operator mutation.

Finally, we note a system parameter, *useSeed*, which forces the first rule in any initialized agent to be: $(1, 0, ..., 0,$ “divide”$)$. This rule may be viewed as similar to the undifferentiated cleavage found in the initial stage of animal embryogenesis. Note that *useSeed* has no effect after initialization, hence allowing the “seeded” rules to possibly be removed from the population, depending on the whims of evolution.

B.3 Deva 1 Variants

Here we introduce several variants of the *Deva 1* model; These are designed to allow for the comparative study of various model-level choices made in the design of *Deva 1*.

B.3.1 Deva 1.N

Deva 1.N is equivalent to *Deva 1* in all ways, save for the inclusion of an explicit “nothing” command in the action list.

Initialization varies slightly for the *Deva 1.N* genome in the generation of a random action, which follows the following distribution:

$$Pr[Y = a] = \begin{cases} 1/5, & \text{if } a = \text{“divide”} \\ 1/5, & \text{if } a = \text{“die”} \\ 1/5, & \text{if } a = \text{“nothing”} \\ 1/5, & \text{if } a = \text{“elongate”} \\ 1/5, & \text{if } a = \text{“specialize”} \end{cases}$$

where each sort of specialization is equally likely.

B.3.2 Deva 1.Dir

The *Deva 1.Dir* model is an attempt to give control over the direction of actions to the genome, rather than use the *best free location* algorithm. Hence, here there is no directional bias, instead any asymmetry results from the genome.

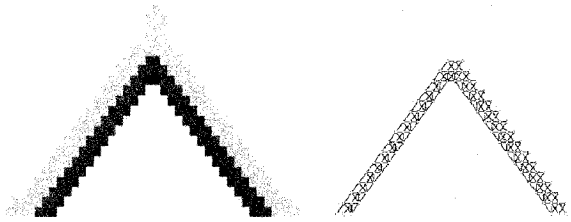


Figure B.2: An organism grown via the *Deva 1.Allway* technique showing morphological but not perfect symmetry. On the left, the perfectly symmetric cell view, contrasted by the non-symmetric truss view on the right.

Deva 1.Dir uses a larger action list, where directions are explicitly included: A = {"nothing", "die", "divide(left)", ..., "divide(down)", "elongate(left)", ..., "elongate(down)", "specialize(x)"}, where $x \in \{2, ..., 32\}$, making $|A| = 41$. Initialization follows the following distribution:

$$Pr[Y = a] = \begin{cases} 1/2, & \text{if } a = \text{"divide(d)"} \\ 1/10, & \text{if } a = \text{"die"} \\ 1/10, & \text{if } a = \text{"nothing"} \\ 1/10, & \text{if } a = \text{"elongate(d)"} \\ 1/5, & \text{if } a = \text{"specialize(x)"} \end{cases}$$

where each colour of specialization and direction of elongation or division is equally likely. The *useSeed* option for *Deva 1.Dir* causes the initial rule to be of the form $(1, 0, ..., 0,$ "divide(d)"), where d is some random direction.

B.3.3 Deva 1.AllWay

The *Deva 1.AllWay* model is very similar to the *Deva 1.N* model, with a different mechanism for division. The new mechanism divides in all ways simultaneously, greatly increasing the speed of organism development. This mechanism has been included for two reasons: Firstly, to explore the addition of a nearly symmetric growth model; Secondly, as similar division mechanisms have been used in other CA-based AE experiments.

Noted above, the symmetry of the organisms will not, in general, be perfect; That is, there will exist two-fold symmetry of cell location and type, mirrored over the $x = 0$ line; But, since cell types need not be symmetric themselves, there may be asymmetry in the beams, even more so following trimming. We shall refer to this as "morphological symmetry" in the future. An illustration of an organism which has morphological symmetry but not perfect symmetry is shown in Figure B.2.

The action list is identical to that of *Deva 1.N*, save the replacement of the "divide" (into best free location) action with the "divide(allway)". Additionally, the elongation options have been greatly limited so as to preserve morphological symmetry — we allow only the "elongate(up)" action. Hence, our total action list is A = {"nothing", "die", "divide(allway)", "elongate(up)", "specialize(x)"}, where $x \in \{2, ..., 32\}$, making $|A| = 34$.

Initialization varies slightly for the *Deva 1.N* genome in the generation of a random action,

which follows the following distribution:

$$Pr[Y = a] = \begin{cases} 3/10, & \text{if } a = \text{“divide(allway)”} \\ 1/5, & \text{if } a = \text{“die”} \\ 1/5, & \text{if } a = \text{“nothing”} \\ 1/10, & \text{if } a = \text{“elongate(up)”} \\ 1/5, & \text{if } a = \text{“specialize”} \end{cases}$$

where each sort of specialization is equally likely.

B.3.4 Deva 1.Clockwise

The *Deva 1.Clockwise* model is identical to the *Deva 1.N* model, save for the replacement of division into the *best free location* by division into a location chosen more simply. The next location for cell division is the first free location found in the four-neighbourhood, starting with the left and proceeding clockwise. This model is included in order to provide a middle step between *Deva 1.N* and *Deva 1.Dir* in terms of intelligence provided by designer to the developmental process. Action list and initialization are identical to that of *Deva 1.N*, save the replacement of the “divide” and “elongate” (into best free location) actions with “divide(clockwise)” and “elongate(clockwise)”.

B.3.5 Deva 1.Stateless

The *Deva 1.Stateless* model is a model in which the colour of the active cell does not partition the DNA of the transition function. Whereas in the typical *Deva 1* model, a rule in the transition function is of the form

$$(c, h_1, ..., h_{n_c}, a)$$

in *Deva 1.Stateless*, rules instead follow the template

$$(h_1, ..., h_{n_c}, a)$$

This means that the transition function is incapable of making decisions on the basis of the colour of the active cell, and is hence a decrease in information, and a divergence from biology. The *Deva 1.Stateless* transition function follows the algorithm described in Algorithm 3.

B.4 Deva 2

A *Deva 2* transition function is a listing of $|\phi|$ rules of the form:

$$(c, h_1, ..., h_{n_g}, h_j, a)$$

where c is a colour, a an action, and h_i is a count of the number of activations of gene g_i in the twelve-neighbourhood. The h_j hormone counts the number of joints, relative to empty cells. The transition function then compares the current description of a cell to the various patterns in

Algorithm 3 Deva 1.Stateless Transition Function Application Algorithm

Current cell c_0
Deva-neighbourhood μ_{c_0}
Environmental-Hormones $e_i \leftarrow 0$ for all $e_i \in \{e_0, ..., e_{n_c}\}$
for all Cells $c \in \mu_{c_0}$ **do**
  int $type \leftarrow cellType(c)$
  $e_{type} \leftarrow e_{type} + 1$
end for
Rule $r_{min} = (s_{min}, a_{min}) \leftarrow (\emptyset, \emptyset)$
float $min \leftarrow \infty$
for all Rules $r \in \phi$ **do**
  float $distance \leftarrow (e_0 - r_{h_0})^2 + ... + (e_{n_c} - r_{h_{n_c}})^2$
  if $distance < min$ **then**
    $min \leftarrow distance$
    $r_{min} \leftarrow r$
  end if
end for
return a_{min}

ϕ, outputting the closest matching rule, if one exists. This process is made precise in Algorithm 4.

Algorithm 4 Deva 2 Transition Function Application Algorithm

Current cell c_0
Deva-neighbourhood μ_{c_0}
Environmental-Hormones $e_i \leftarrow 0$ for all $e_i \in \{e_0, ..., e_{n_g}\}$
for all Cells $c \in \mu_{c_0}$ **do**
  for all Gene types g_i **do**
    if g_i is active in c **then**
      $e_i \leftarrow e_i + 1$
    end if
  end for
end for
Rule $r_{min} = (s_{min}, a_{min}) \leftarrow (\emptyset, \emptyset)$
float $min \leftarrow \infty$
for all Rules $r \in \phi$ **do**
  if $r_{colour} = colour(c_0)$ **then**
    float $distance \leftarrow (e_0 - r_{h_0})^2 + ... + (e_{n_g} - r_{h_{n_g}})^2 + (e_j - r_{h_j})^2$
    if $distance < min$ **then**
      $min \leftarrow distance$
      $r_{min} \leftarrow r$
    end if
  end if
end for
return a_{min}

B.4.1 Deva 2.Stateless

In the *Deva 2.Stateless* model, much like in the *Deva 1.Stateless* model, the initial identifying colour in the genome rules is removed. Hence, a rule in ϕ may be written as

$$(h_1, ..., h_{n_g}, h_j, a)$$

This again removes information from the decision-making process, and moves *Deva 2.Stateless* further from biological plausibility. It is included for contrast, for the same reasons as the inclusion of *Deva 1.Stateless.*

B.5 Deva 3

A *Deva 3* transition function is a listing of $|\phi|$ rules of the form:

$$(h_0, h_1, ..., h_{11}, a)$$

where h_i represents a colour from the twelve-neighbourhood (See Figure B.1 for indices). Let us denote the hormones from the rule $\mathbf{h}$. The transition function creates a description of a given cell's current neighbourhood, $\mathbf{c} = (c_0, c_1, ..., c_{11})$, consisting of a listing of the colours surrounding the current cell. Then, the Manhattan distance is computed:

$$mandistance(\mathbf{h}, \mathbf{c}) = \sum_{0 \leq i < 12} \begin{cases} 1; & c_i = h_i \\ 0; & \text{ow.} \end{cases}$$

The first rule in the transition function with minimal *mandistance* will specify the action taken by the cell.

Initialization of a *Deva 3* genome begins with a list of twelve colours, each chosen uniformly and randomly from the set C. The action is chosen from the same distribution as in the *Deva 1* Model.

B.6 Deva 4

The *Deva 4* model uses the action set $A = \{die, divide, on(g_0), ..., on(g_{n_g}), off(g_0), ..., off(g_{n_g})\}$. The *Deva 4.CellType* uses transition function and initialization from *Deva 1*, while the *Deva 4.GeneType* uses transition function and initialization from *Deva 2.* Initialization of the actions for both *Deva 4* models uses the distribution:

$$Pr[Y = a] = \begin{cases} 1/6, & \text{if } a = \text{“divide”} \\ 1/6, & \text{if } a = \text{“die”} \\ 1/6, & \text{if } a = \text{“nothing”} \\ 1/6, & \text{if } a = \text{“elongate”} \\ 1/6, & \text{if } a = \text{“on”} \\ 1/6, & \text{if } a = \text{“on”} \end{cases}$$

with each gene being equally likely for “on” and “off”.

Appendix C

Truss Analysis

C.1 Truss Analysis

In this section, we discuss the stress analysis of a general plane truss. The assumption is made that the truss is topologically connected, pin-connected, friction-free, and that force is applied only at joints; However, no other assumptions are made, notably about the stability or static determinacy of the truss.

In our example, all truss members are identical in terms of material and area; We use a $EA = 2.00 \times 10^9 \frac{\text{N}}{\text{m}^2} \cdot 7.85 \times 10^{-5}\text{m}^2 = 1.57 \times 10^4$ N, corresponding to a modulus of elasticity for steel [Meg05] and a cylindrical member of diameter 1 cm ($A = \frac{\pi(0.01)^2}{4}$).

An important question regarding structures is whether or not they are stable — this question will determine if the structure is capable of supporting itself without deformation. The following is a discussion of some conditions which may be used to test for the static determinacy of a general plane truss.

If a reaction component were added to the truss without including a condition that would allow for its determination, we would call the truss statically indeterminate externally. If a member were added without a joint, the structure would be called statically indeterminate internally.

We may test for external determinacy and stability according to the following process: Let r be the least number of reaction components required for external stability, and r_a the actual number of reaction components, then:

- if $r_a < r$ then the structure is statically unstable externally
- if $r_a = r$ then the structure is statically determinate externally
- if $r_a > r$ then the structure is statically indeterminate externally

Hence, the condition $r_a \geq r$ is a necessary but insufficient condition for statical classification.

We must also guarantee internal determinacy; Let m be the total number of beams, and j the number of joints. Then internal determinacy is satisfied when:

$$2j = m + r$$

This is true since the left-hand side is the number of equilibrium equations that are available,

while the right-hand side represents the total number of unknown bar and reactive forces. Hence,

$$m = 2j - r$$

Or, m members are required to form an internally static structure. If m_a is the actual number of bar forces, we get:

- if $m_a < m$ then the structure is statically unstable internally
- if $m_a = m$ then the structure is statically determinate internally
- if $m_a > m$ then the structure is statically indeterminate internally

So, is $m_a < m$ then the truss is certainly unstable. However, if $m_a \geq m$, it does not necessarily follow that the truss is stable — it is possible that the beams are not properly configured to ensure stability.

Hence, thus far, we have necessary, but not sufficient, criteria for the determination of stability in a general plane truss.

Ideally, the majority of unstable trusses can be thrown away by the simple test above; However, while useful for easing computation, we cannot make this assumption in general. Instead, we assume that all trusses are statically indeterminate, and will expose the unstable ones by failing to solve the relations. Below, we describe a matrix-based equilibrium method for the analysis of plane trusses using force and deformation. Both technique and derivation are from West's treatment [Wes89].

C.1.1 Equilibrium Method of Truss Analysis

We present a summary of the equilibrium method of truss analysis; a more explicit description and derivations may be found in Section C.2.

Consider a general truss with n joints and m beams; We are provided with external forces to be applied at joints, and wish to determine the member forces. Let our structure forces be $\{P\} = \{P^1, ..., P^n\}^T$, structure displacements be $\{\Delta\} = \{\Delta^1, ..., \Delta^n\}^T$ and member forces be $\{F\} = \{F^1, ..., F^m\}^T$.

We may relate the individual member forces to displacement and structure forces as follows:

$$\{F\}^i = [k]_a^i [\beta]^i \{\Delta\} \tag{2.4'}$$

where $[\beta]^i$ is the connectivity matrix for the ith member beam, and $[k]_a^i$ is its stiffness matrix. Hence, it suffices to compute the displacements. The displacements may be computed through a truss stiffness matrix, a combination of the individual member stiffness matrices:

$$\{\Delta\} = [K]^{-1}\{P\} \tag{2.5'}$$

Hence, given a plane truss, we may first compute the stiffness matrix, then compute the displacements, then the individual member forces.

The entire process is bounded by the calculation of a matrix inversion[1], and hence has running time $O(m^3)$.

[1]Actually, a slightly faster LU-Decomposition, as implemented by the Java numerics library Jama.

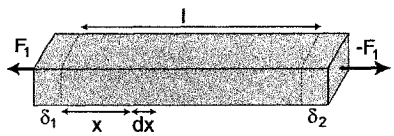


Figure C.1: Illustration of terminology in member-deformation.

C.2 Derivation of Equilibrium Method of Plane Truss Analysis

C.2.1 Derivation of Axial-Force Deformation Relationships

Here we define several terms necessary for the analysis of pin-connected trusses through deformation[2]. Given a truss member-beam (see Figure C.1) of length l, area A, modulus of elasticity E, and subjected to member force $F_1 = F_2$, we aim to compute its deformation. If our deformation is $\Delta = \delta_1 + \delta_2$, we may do so by computing the member axial stiffness matrix, which relates deformation and member force:

$$\{F\} = [k]_a\{\delta\} \tag{C.1}$$

We consider a truss member beam under stress; See Figure C.1 for an illustration — our member is assumed to have length l, area A and be subjected to axial force F. An element of length is written dx. We may represent the stress as

$$\sigma = \frac{F}{A}$$

and the strain as

$$\epsilon = \frac{dl}{dx}$$

where dl is the change in length (deformation) that accompanies force F.

For a linearly elastic material, we may relate stress and strain through Young's modulus (modulus of elasticity)

$$\sigma = E\epsilon$$

leading to the relation

$$dl = \frac{F}{EA}dx$$

Hence, our total elongation, Δ, is obtained by integrating over the length of the member:

$$\Delta = \int_{[0,1]} dl = \frac{F}{EA}\int_{[0,1]} dx = \frac{Fl}{EA}$$

otherwise written

$$\Delta = \left(\frac{l}{EA}\right)F$$

[2]In general, we use superscripts for the index number of a beam or joint, and subscripts for directions.

Inverting, we obtain

$$F = \left(\frac{EA}{l}\right)\Delta \tag{C.2}$$

the middle term, $k_a = \frac{EA}{l}$ being the axial stiffness.

We note that $F = F_1 = F_2$, and $\Delta = \delta_1 + \delta_2$; Substituting into equation C.2, we obtain

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} = \begin{bmatrix} \frac{EA}{l} & \frac{EA}{l} \\ \frac{EA}{l} & \frac{EA}{l} \end{bmatrix} \begin{Bmatrix} \delta_1 \\ \delta_2 \end{Bmatrix} \tag{C.3}$$

which we abbreviate as expressed in equation C.1.

C.2.2 Derivation of Equilibrium Method

Here we derive the equilibrium method of truss analysis described in Section C.1.1. As noted before, we have a general truss with n joints and m beams; We are provided with external forces to be applied at joints, and wish to determine the member forces. Let our structure forces be $\{P\} = \{P^1, ..., P^n\}^T$, structure displacements be $\{\Delta\} = \{\Delta^1, ..., \Delta^n\}^T$ and member forces be $\{F\} = \{F^1, ..., F^m\}^T$.

For each truss member (beam), i, from equation C.1, we have the following relation:

$$\{F\}^i = [k]_a^i \{\delta\}^i \tag{C.1'}$$

To ensure that members fit into the structure as a whole, it is necessary that member displacement be compatible with structural displacement:

$$\{\delta\}^i = [\beta]^i \{\Delta\} \tag{C.4}$$

where $[\beta]^i$ is the connectivity matrix for the ith member.

The matrix $[\beta]^i$ is a $2 \times n$ matrix which relates the member deformations of beams to the total structural deformation at joints. Assume that member i has angle θ and ϕ with the x- and y-axis, respectively. If beam i meets joint j, we may relate the displacement, Δ^j, to the deformation of beam, δ^i, with the following relation

$$\delta^i = -(\Delta_x^j \cos\theta + \Delta_y^j \sin\phi) \tag{C.5}$$

We may do similarly for the joint at the other end of beam i as well. Hence, we define $[\beta]_{g,h}^i$ to be either the x or y component of equation C.5 if beam i meets joint m, and 0 otherwise.

We may now relate the member forces with the displacements

$$F^i = [k]_a^i [\beta]^i \{\Delta\} \tag{2.4"}$$

Considering the work done as measured in the structural coordinate system must equal the work done in the member coordinate system; The work for a member induced by a force with magnitude P, given that it is proportional to displacement, is

$$W = \frac{1}{2} P\Delta$$

Or, for a constant load over m components

$$W = \sum_{i=1}^{m} \frac{1}{2} P^i \Delta^i = \frac{1}{2} \{P\}^T \{\Delta\}$$

Hence, for the m members of our truss, we have

$$\frac{1}{2} \{P\}^T \{\Delta\} = \sum_{i=1}^{m} \frac{1}{2} \{F\}^{iT} \{\delta^i\}$$

Substituting equation C.4, we obtain

$$\frac{1}{2} \{P\}^T \{\Delta\} = \sum_{i=1}^{m} \frac{1}{2} \{F\}^{iT} [\beta]^i \{\Delta\}$$

or

$$\{\Delta\}^T \left(\{P\} - \sum_{i=1}^{m} [\beta]^{iT} \{F\}^i \right) = 0$$

Since we may design any arbitrary initial configuration, we may assume that $\{\Delta\} \neq 0$, leading to

$$\{P\} = \sum_{i=1}^{m} [\beta]^{iT} \{F\}^i$$

or, substituting equation 2.4,

$$\{P\} = \sum_{i=1}^{m} [\beta]^{iT} [k]_a^i [\beta]^i \{\Delta\}$$

However, we may consider the general stiffness matrix

$$[K] = \sum_{i=1}^{m} [\beta]^{iT} [k]_a^i [\beta]^i$$

simplifying our equation to $\{P\} = [K]\{\Delta\}$, or

$$\{\Delta\} = [K]^{-1} \{P\} \tag{2.5"}$$

Once structural displacements have been obtained, we may compute member-forces by substituting into equation 2.4.

Appendix D

Code Documentation

D.1 Terminating Cellular Automata Documentation

The Terminating Cellular Automata are implemented as a Java package, TCA. This package is freely available at http://kowaliw.net/tca.html, along with all computed data. Note that images are saved using the package *javax.imageio*, meaning that some window manager must be running in order for the program to execute properly. Permission to re-use, re-distribute and modify granted, but please make note of any modification, and reference this thesis.

To compute complexity measure for one-dim TCAs:

```
% java TCALauncher <length> <maxDiameter>
```

e.g.

```
% java TCALauncher 10 4
```

output in a directory named "l<length>"; Caution: program deletes and overwrites any existing data.

See directory tcadata for pre-computed data, along with gnuplot scripts for plotting and PERL script for computing correlations.

To compute optima for small environments, uncomment the appropriate portion of EnvOptimum2D, compile and run.

```
% java EnvOptimum2D
```

Output goes to a file in the local dir, "./env<whichEnv>/patterns/env<whichEnv>Optima.patterns".

To find TCAs which compute any of a list of patterns (say, from EnvOptimum2D), use:

```
% java PatternFinder2D <whichEnv> <whichDiameter>
```

<whichDiameter> specifies a specific diameter, not a maximum. For each pattern, a directory will be create in the aforementioned pattern dir, and a list of TCAs of specified diameter will be saved (if there are no such TCAs, an empty file will be saved).

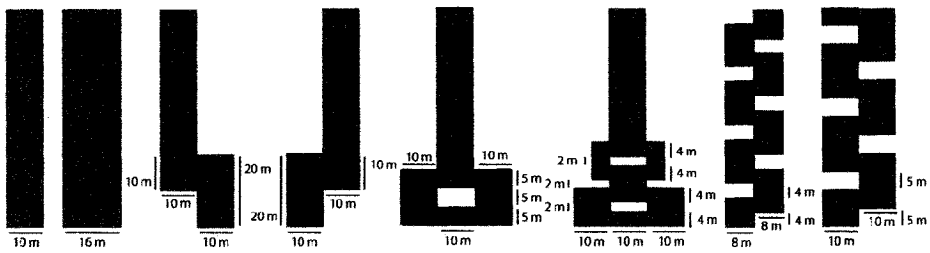


Figure D.1: Possible environments for growth, with specification code: (*left to right*) 0: *unlimited growth* (not shown), 1: *very-thin*, 2: *thin*, 3: *right-hook*, 4: *left-hook*, 5: *bulbs1*, 6: *bulbs2*, 7: *zig-zag1*, 8: *zig-zag2*, 9: *zig-zag3*, 10: *squat*.

D.2 Deva Documentation

Deva Version 1.0 is a two-dimensional implementation of three variations of the Deva algorithms, *deva1*, *deva2* and *deva3*. Deva implements a Genetic Algorithm, producing successive populations of agents. Produced agents are interpreted as Plane Trusses and evaluated in terms of a fitness function incorporating static determinacy. Deva Version 1.0 is packaged in *deva.jar*, containing the library *net.kowaliw.deva*.

D.2.1 Usage

Deva is packaged as a java archive, and used through the command line. Visualization requires Java Swing and Java 2D — hence, we require a graphical OS for viewing results. However, all other components do not utilize any graphical routines, so it is possible to run the majority of an experiment in a console-based OS. This is particularly useful for the computation of the Genetic Algorithm, as development and fitness tend to be intense computationally.

Deva is completely contained within the *deva.jar* file, and will use the current directory as a location to store information. Output will be stored in created directories; All output is of the form of either plain text documents or JPEG images.

Java 1.5 or better is required, available from http://java.sun.com. Deva also uses Jama[1], a Java package for matrix-computation — Jama is included in the file deva.jar. Deva is activated via the command-line interface.

D.2.2 Creating a New Experiment

Deva is initialized by creating a new *run*, created via the command:

```
% java -jar deva.jar -n <dirName>
```

This will generate a new directory named *dirName*, containing a Java properties file, *deva.data*; This file may be edited with a standard text editor, format should be clear from context[2]. Customizable parameters are listed in Table D.1, all other parameters should be left unchanged.

[1] Jama was developed by MathWorks and the American National Institute of Standards and Technology (NIST), and released into the public domain; More information at http://math.nist.gov/javanumerics.

[2] The program will conveniently inform users of input errors by crashing.

Table D.1: Parameters and explanations in file *deva.dat*.

Parameter	Type	Description
dirName	String	Directory (and run) name
resourceCounter	float	Controls maximum size of the agent (r_c)
devType	int	Type of growth, see Table D.3
numRules	int	Number of rules in genome (or $\|\phi\|$)
moveOrElongation	int	(*deva1*) 0: neither, 1: move, 2: elongation
useSeed	int	(*deva1, 2*) Seed an initial divide command?
minAge	int	(*deva1, 2*) Minimum age for cell action
numGenes	int	Number of genes (or n_g), at least 5
popSize	int	Population size
initialMultiplier	int	Multiplier for initial generation
numGenSave	int	Frequency of auto-save, in generations
saveFitnessJump	int	Save every time maximum fitness increases?
whichFitness	int	See Table D.2
whichEnv	int	0 - 10, see Figure D.1
whichInit	int	1: power-distribution, 2: uniform-distribution
whichMut	int	1: power-distribution, 2: uniform, 3: copy
percentageElite	float	Rate of elitism, in [0,1]
probMutation	float	Rate of mutation, in [0,1]
probCross	float	Rate of crossover, in [0,1]
useTournament	int	0: fitness-proportional, 1: tournament selection
numTournament	int	Number of agents to use in tournament
tournamentProbability	float	p, used for tournament selection, in [0,1]
memberArea	float	Area (m^2) of a beam
modElasticity	float	Young's modulus of elasticity
unitDistanceInMetersX	float	Length (m) of a unit in the x-axis
unitDistanceInMetersY	float	Length (m) of a unit in the y-axis
maxDisplacement	float	Displacement (m) before truss labelled unstable

Table D.2: Fitness functions and associated specification codes.

Fitness function	Code	Fitness function	Code
f_{mat}	0	f_{stoch}	1
f_{base}	2	f_{fat}	4
f_{se}	5	f_{me}	6
f_{fe}	7	f_{light}	9

Table D.3: Types of deva growth and associated specification codes.

Deva Algorithm	Code	Deva Algorithm	Code
Deva 1	1	*Deva 1.N*	11
Deva 1.Dir	12	*Deva 1.Stateless*	13
Deva 1.Clockwise	14	*Deva 1.AllWay*	15
Deva 1.NoSpec	16		
Deva 2	23	*Deva 2.Stateless*	22
Deva 2.NoSpec	26		
Deva 3	3		
Deva 4.GeneBased	4	*Deva 4.CellBased*	41
Deva 5	5		
Deva 99	99	*Deva 999*	999

Computing the Genetic Algorithm

Given some created and configured run, we may compute a series of steps of the genetic algorithm:

```
% java -jar deva.jar -d <dirName> <numGen>
```

The program will retrieve the last saved generation in directory *dirName* (or, initialize the first generation), and execute *numGen* generations. To standard out will be printed a progress indicator, along with summary statistics of the completed generations. The GA will automatically save each *numGenSave* steps, and at the end of the prescribed number of generations.

Visualization

Given a created and configured run, we may visualize the current population:

```
% java -jar deva.jar -v <dirName>
```

This command will load the population viewer for the last computed generation, allowing a user to browse visual representations of the agents, trusses, and their development. To view a previous saved generation, use

```
% java -jar deva.jar -v <dirName> <genNum>
```

where *genNum* is a saved population. To view a particular agent (useful for minimizing memory use when r_c is high), use the command

```
% java -jar deva.jar -o <dirName> <genNum> <agentIndex>
```

to launch agent *agentIndex* from saved generation *genNum*, or

```
% java -jar deva.jar -o <dirName> <agentIndex>
```

where the *genNum* will be understood to be the most recently saved. Note that a population file is saved unsorted; Hence, there is no reason to believe that the ith agent in the saved population will resemble the ith agent in a PopulationViewer. To rectify this, use the command

```
% java -jar deva.jar -s <dirName> <genNum>
```

to sort the population file for saved generation *genNum*, overwriting the original file.

The primary visualization interface is the Population Viewer; This is a control frame containing a set of descriptors and controls for each agent in the population. The bottom panel of the Population Viewer contains a set of descriptors for the agent in question, and the top contains a series of controls for the various means of visualization. These controls will allow a user to view the agent in several ways:

- The *Agent Viewer*, which depicts the agent as a simple set of coloured cells.
- The *Agent Development Viewer*, which shows a trace of the development of the agent, including several internal variables from the DevSpace and DevSpaceGridCell classes.
- The *Truss Viewer*, which depicts the agent as a plane truss, trimmed.

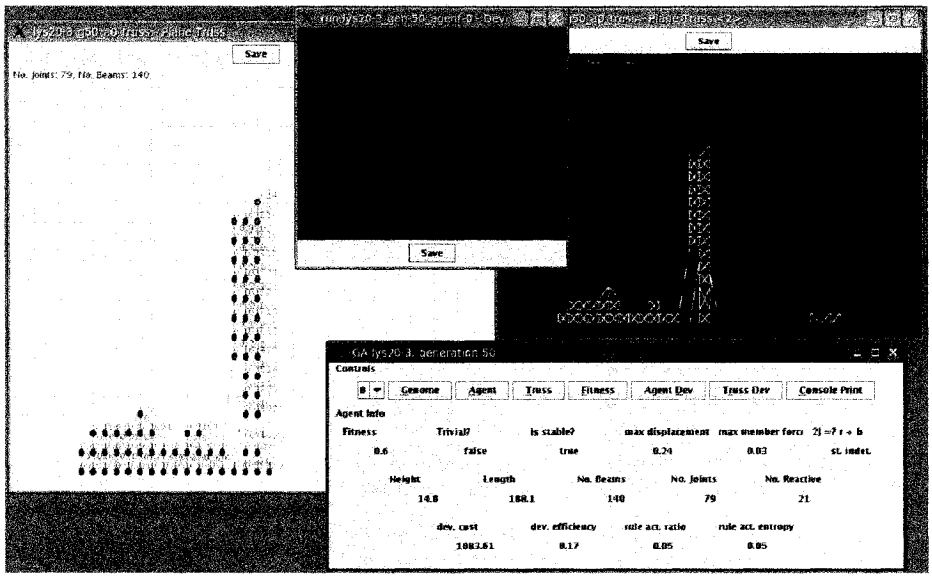


Figure D.2: Screen shot of some of Deva's visualization tools: (*bottom right*) the Population Viewer, (*left*) Fitness Viewer, (*top center*) Agent Viewer, (*right*) Truss Viewer.

- The *Truss Development Viewer*, which shows the agent's development as a series of images of plane trusses.
- The *Fitness Viewer*, which (if the truss has not been previously declared unstable) the stress analysis of the truss components

See Figure D.2 for a screen shot of the Population Viewer and associated visualization tools.

Debugging Routines

There is also a debugging routine included, allowing one to compute the fitness of, and visualize, a truss provided in matrix form. A truss may be specified in a plain text file, in the following format:

```
<width>
<height>
<cellType> ... <cellType>
...
<cellType> ... <cellType>
```

where the cell types are specified as integers between 0 and 32[3]. Elongations are controlled by the addition of a multiple of 100 to the integer specification; For example, the integer 329 specifies a cell of colour 29 which is an elongation upward, and integer 429 specifies a cell of colour 29 which has been elongated upward — see Table D.4 for the codes. It is easy to provide improper input at this stage, results in this case will be unpredictable. Note that excess white space will break the truss debugger — leave precisely one space between all line entries, and use three digits for each number to have the file line up properly (See Figure D.3). The truss debugger may be activated with the command

[3]There is no need to include hidden genes, as these do not effect phenotype directly.

Table D.4: Mapping of integer additions to cell properties.

Integer	Interpretation	Integer	Interpretation
0	Regular cell.		
100	Elongation left.	200	Elongated left.
300	Elongation up.	400	Elongated up.
500	Elongation right.	600	Elongated right.
700	Elongation down.	800	Elongated down.

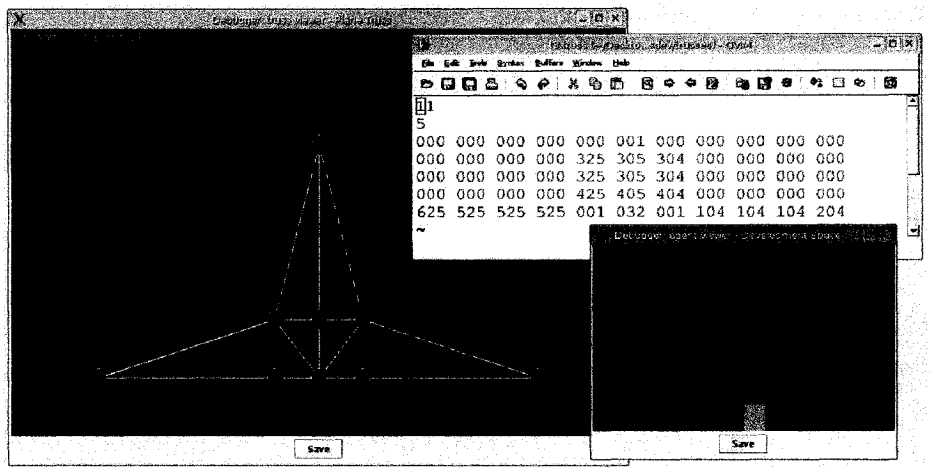


Figure D.3: Visualization of a provided truss: on the right is a view of the plain text file specifying the truss, below the agent visualization, and on the left the truss visualization.

```
% java -jar deva.jar -t <trussFile>
```

A screenshot of an example truss is shown in Figure D.3.

Another debugging routine included deals with specified growth; This routine allows a user to grow a single agent utilizing a specified deva properties file and an *action list* file (.al). Use of this option will grow and provide visualization (agent, truss, agent-development and truss-development) of the agent. For programmatic ease, all cells in developmental space execute the same action in a given developmental time-step. The growth debugger may be activated as follows:

```
% java -jar deva.jar -a <dirName> <actionList>
```

where *dirName* contains the appropriate properties file, and *actionList* is a file specifying the sequence of actions undertaken by the developmental process (See the example action list files in directory *debug* for example format). The code to map from integers to actions is shown in Table D.5.

Data Logging

Deva automatically creates datalogs — these are located in the run directory, under the names *log.data* and *log.html*. Both are identical in content, but one is formatted for a web browser, while the other is raw whitespace-delimited plain text, suitable for feeding to a mathematical software package.

Table D.5: Mapping of integers to actions in action list files.

Integer	Action	Integer	Action
0	Nothing	1	Die
10	Divide	20	Move or Elongate.
21	Elongate left	22	Elongate up
23	Elongate right	24	Elongate down
25	Move left	26	Move up
27	Move right	28	Move down
$100 + x$	Gene x on	$200 + x$	Gene x off
$1000 + x$	Specialize(x)		

It is also possible to save images from the various agent viewers, by using the *Save* command in the agent or truss viewer, or the *Save-Play* command from the agent or truss trace viewer. In the former case, a JPEG image is saved to the current directory, while in the latter a new directory is created in the current directory containing the images.