Bluenome: A Novel Developmental Model for the Evolution of Artificial Agents.

Taras Kowaliw

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montréal, Québec, Canada

August 2003

Your file Votre référence
ISBN: 0-612-83907-9
Our file Notre référence
ISBN: 0-612-83907-9

ABSTRACT

Bluenome: A Novel Developmental Model for the Evolution of Artificial Agents.

Taras Kowaliw

The Bluenome Model of Development is introduced. The Bluenome model is a developmental model of Artificial Morphogenesis, inspired by biological development, instantiating a subset of two-dimensional Cellular Automata. The Bluenome model is cast as a general model, one which generates organizational topologies for finite sets of component types, assuming local interactions between components. Its key feature is that there exists no relation between genotopic complexity and phenotopic complexity, implying its potential application in high-dimensional evolutionary problems. The Bluenome model is first applied to a series of application-neutral experiments, in which it is shown experimentally that it is capable of producing robust agents in a reasonable amount of computation. Next, it is applied to an application involving the design of embedded agents. This second series of experiments contrasts the Bluenome model against a model in which there exists a bijective relation between genotype and phenotype, showing that the Bluenome model is capable of performing as well or better in cases of high phenotopic complexity. Additionally, genomes from the Bluenome Model are shown to be capable of re-development in differing environments, retaining many relevant phenotopic properties.

Acknowledgements

I would like to thank the University of Toronto, specifically the department of Mathematics and the Adaptive Technology Resource Centre, for their training and support. Without either of these resources, I would not be where I am today. Additionally, I am indebted to Concordia University, specifically the departments of Computer Science and Computer & Electrical Engineering for allowing me access to their excellent program and resources.

I would also like to thank Nawwaf Kharma. My work on his projects have provided me with much experience in dealing with evolutionary computation, and with conducting research in general. His advice and support are much appreciated.

Finally, I would like to thank Peter Grogono. This project was born out of my conversations with him regarding his ambitious “Laws and Life” project; Indeed, it would be difficult to distinguish where his ideas end and mine begin, and equally difficult to imagine this project’s completion without his advice and support. It is my sincere hope that this work will provide motivation and insight into his larger project.

This thesis is dedicated to Kara-Anne Fraser.

Table of Contents

List Of Tables

List of Figures

1. - Introduction

The issue of representation in evolutionary computation has long been recognized as a key component, and the best choice of a representation for any particular problem has long since been identified as a key component in the design of Genetic Algorithms. The typical application of a Genetic Algorithm focuses on simple solutions, with as little dimensionality as possible, typically bijective between genotype and phenotype. In these models, virtually no information results from the translation between genotype and phenotype – essentially, values from the genotype are simply slotted into some evaluating function.

However, in the past twenty years, many other key approaches for mapping between genotype and phenotype have emerged – methods in which the genome is interpreted as a program or scheme for constructing a phenotype. In these models, some degree of the information in the phenotype is a result of not only the genome, but also of the mechanism from which it is spawned.

An emerging trend in evolutionary computation lately has been for the other extreme – situations in which the genome is a (relatively) small and simple entity, while the developmental process increases drastically in complexity.

In a field inspired by biological processes, it is often beneficial to examine the natural analogue. Consider that the number of cells within the human body is estimated to be between 75 and 100 trillion, while human DNA consists only of an estimated 3 billion chemical base pairs. Consider also that there exists no intuitive relation between the complexity of an animal genome and its phenotopic size; For example, the onion genome is ten times the length of that of a human. It is quite obvious in this context that there is a great deal occurring in the developmental process which compresses the developed complexity, and exploits the complexity inherent in the developmental process to generate the robust patterns found in nature.

Regarding biology, arguments involving the nature of the developmental process often involve its necessity for complex design. Most notably, Wolfram has recently taken this viewpoint, heralding Cellular Automata (CA) as a means for modeling these processes [Wolfram; 2002]. He argues that the typical view of natural selection as a mechanism of increasing the complexity of agents is flawed; Instead, complexity is an inherent component of the developmental process, while natural selection serves to limit its unwieldy nature. The inherent assumption here is that many of the adaptations which are commonly present amongst unrelated species are the result not of an adapted genome, but instead the result of their ease of generation under the associated developmental process. In defense of his claim that complexity is omnipresent in the development of an organism, Wolfram presents a wealth of visually appealing evidence (photographs of elaborate skin pigmentations, shell growths, plant structures, etc.), some of which are drawn from fossils remaining from early periods in evolutionary history, which resemble very closely easily generated CAs.

If the above viewpoint is correct, the implication for Evolutionary Computation is that the typical approach does not utilize the mechanisms of selection to their fullest potential. That is, if the sorts of robust complexity we see in natural organisms is largely the result of a refinement of a computationally complete process, mimicry of natural selection ought to follow this lead. If the sorts of robust organization present in natural organisms are encoded for largely by the developmental process, there is little reason to believe that another process is easily capable of developing them.

Indeed, the detractor might point out that the developmental process in natural organisms follows the course that it does out of necessities inflicted by the natural world – that the exclusive use of local cell-based information is the norm for the sole reason that this is the only tool presented to nature. This being said, it is possible that a system involving a global overlord might prove a better choice than a local information-based model. Such a system, however, would still need to deal with the problems of large-scale complexity in the genome – the size of many design tasks is daunting, while the limits of the potential for a search mechanism to operate in a high-dimensional space are well known. It is reasonable to assume that any system capable of developing large organisms must be computationally complete. If this is the case, our best course of action is to select one based on the natural developmental process, as, to the best of my knowledge, it is the only known working example.

With this in mind, my aim in this thesis is to present a model for mapping between genotype and phenotype inspired by biological cell differentiation. The Bluenome[1] Model of Development is both a simplified model of two-dimensional CAs, and a simplified model of natural development. In the Bluenome model, an agent's genome consists of a series of rules fired by matching patterns modeling a component's local neighbourhood. An agent undergoes a development phase prior to evaluation; In this development phase, it begins as a single neutral cell, with a copy of the agent's genome. Through successive time steps in the developmental process, the cell will collect information from its local neighbourhood, and use its genome to determine its next action. If the genome so dictates, the cell will divide, producing an exact copy of itself, including an identical genome. The process continues producing new cells, until a specially included internal counter, the number of telomeres, is exhausted. Hence, phenotopic complexity is controlled by a system parameter, rather than by the complexity of the genome (Or, the growth of the agent may be limited instead by the size of the environment, again a system parameter).

It is not obvious, given a developmental model, that it is useful for the generation of agents – questions involving the robustness of the phenotopic space of possible agents and the feasibility of search through the genotopic space abound. In Phase One of this project, this technique is used to generate images, where components are simply blocks of colours – this approach was chosen in the attempt to demonstrate principles regarding the Bluenome technique in an application-neutral context. In Phase One, the attempt is made

[1] The ge**nome** as a **blue**print for development.

to demonstrate that the Bluenome technique is both robust enough to generate interesting agents, but also capable of evolution under phenotopic pressures.

If the theories presented above are true, one would expect that there exists a level of phenotopic complexity at which a developmental process exceeds the abilities of a more typical approach; Phase Two of this paper aims to demonstrate precisely that. In Phase Two an application inspired by biology (but limited for computational concerns) is presented. An artificial organism, the Tarasanoid, is presented, along with an environment. The goal is to develop a Tarasanoid capable of surviving for the longest possible period of time, utilizing a set of pre-defined components. Doing so implies the development of mechanisms for interpreting sensory input and transforming that input into action, as well as the development of a transportation network for distributing found food throughout the agent's body; This is by no means a trivial problem. In Phase Two, two methodologies for translating between genotype and phenotype are explored: the Bluenome Model, and a bijective model, in which there exists a one-to-one correspondence between genes and components.

The intention of this project was not to provide proof of the above theories (although success might be interpreted as positive evidence). Also, it is not to create a model of biological development – several models exist at present, all of which explore the phenomena to a far greater level of detail than the Bluenome model. Instead, Bluenome is presented as a tool for the development of large and complex agents, utilizing natural selection to its fullest potential.

2. - Review

2.1 - Embryology[2]

Morphogenesis describes the process by which a zygote grows into an (at least partially) mature organism. Morphogenesis, (a.k.a. Embryogenesis), is sometimes taken to also describe development outside an organism's developmental environment; However, here we consider it to mean solely the initial development which occurs within an organisms uterine environment. Described below is the developmental process for vertebrates, although we note that the process is similar for invertebrates and plants.

The process begins with a fertilized egg (a zygote), typically spherical and about 0.15 mm in diameter. Following this is a string of stages of growth; The underlying mechanism behind all is the synthesis of new protoplasm through previously existing protoplasm, realized through successive cell divisions.

Initially, the zygote grows through a series of uniform and unorganized cell divisions, in a process known as cleavage. Cleavage divisions continue for a short period, resulting in a solid ball of uniform cells (morula). Following this, the ball becomes filled with intra-uterine fluid, forming a sphere of uniform cells (blastocyst), enclosing a central cavity (although in some species several distinct blastomere may be distinguished nearly

[2] Information for the review of Embryology is drawn largely from [Karp; 2001] and [Patten; 1964].

immediately). There is often a point of asymmetry in the blastocyst, which will be the focal point for the further development of the embryonic bodies.

In the blastula stage, cells begin to arrange themselves along three layers, known as the primary germ layers (the ectoderm, the entoderm and the mesoderm). Once separated, these cells become determined, meaning that following these divisions, only subsets of all possible specializations may be expected within the layers. This process is mediated through gene control, operating in the immediate environment furnished by the growing body of the embryo, although it has been demonstrated that these conditions are further mediated by the intra-uterine environment.

An initial and important mechanism in this development is that of cell migration - cells become determined in the germ layers following a series of cell migrations which distinguish them. Both cell migration and specialization are mediated via inter-cell communications.

Inter-cell communications occur via two primary mechanisms (reviewed in [Fagotto, Gumbiner, 1996]): The first involves directly connected cells, known as contact-dependant signaling. Contact-dependant signaling typically involve transmembrane protein, connecting via cell surface receptors; The best known examples are communications through cell adhesion molecules, although also of importance are juxtacrine signals, in which both ligands and receptors are membrane-anchored proteins, and communication is accomplished through direct control of the adjacent cells ligands, or through directly regulating cytoplasmic signaling proteins; The second method of communication involves signaling over a diffusible intermediate (paracrine communications), through diffusible microhormones, excreted by cells in the extracellular medium. Microhormone communication is known to have a reach of several cell perimeters.

While morphogenesis refers to the entire process, different stages are labeled otherwise: Successive differentiations within the determined regions are known as organogenesis, and later (as in the case of specific tissues), as histogenesis. During these stages, cells continue divisions, also begining phases of recognizable specialization.

An important phenomenon during growth phases is that of the molding of structures via cell death - for example, the disappearance of cartilage cells during endochrondral bone formation is regarded as key in the creation of this unique and important structure.

Following various stages of organogenesis and histogenesis is the integration phase, in which various specialized regions interconnect, largely via cell migration, but in some cases through continued growth.

Both heredity and environment play significant roles in this process. While the primary means of specifying differentiation within a cell is communication from neighbouring cells, also of importance is the prenatal environment furnished during intra-uterine life.

2.2 - Evolutionary Computation & Genetic Algorithms[3]

Evolutionary Computation is a blanket term for several probabilistic approaches involving finding minima. The roots of Evolutionary Computation may be traced back to two early algorithms: Evolutionary Strategies (ES) and Genetic Algorithms (GA); ESs originates from [Rechenberg; 1973], initially consisting of a form of Random Heuristic Search, where a single agent is optimized through a successive series of mutations. ESs were later adapted to a more modern form by [Schwefel; 1981], who made the addition of multi-agent population sizes, and a re-combination operator (crossover). GAs originate from [Holland; 1975], who posited a multi-agent population which goes through successive generations of re-combination; The modern versions of both ESs and GAs resemble each other quite closely, with most differences consisting of varying representations and models for recombination.

The following is a simple view of the Genetic Algorithm; The essential problem to be solved is the problem of finding an optimum for a given function $f: G \rightarrow R$. Initially, a representation for the domain G is chosen, traditionally a binary string, where $G \subset N$. Any $g \in G$ is termed a "genome". A collection of genomes are chosen randomly, forming the initial population P_0. Following, a loop is undertaken:

1. evaluate(P_i);
2. selectParents(P_i);

[3] Information and notation for the Evolutionary Computation & Genetic Algorithms review was largely

```
3. crossover($P_i$);
4. mutation($P_i$);
5. i++;
6. goto 1
```

Evaluate ranks all population members according to the function *f*. *SelectParents* chooses from the previous population a selection of the best genomes as parents for the next generation. *Crossover* is an operator which accepts two (or more) genomes and performs re-combination to produce two (or more) child genomes. *Mutation* changes bits at random within the genomes. Hence, through this algorithm, populations evolve over time, tending toward populations of higher fitness (returning higher values of *f*).

There are many alternative implementations of the function described above, as well as alternatives to the algorithm itself; Listing them is beyond the scope of this thesis. Instead, I describe some typical choices (Fitness-Proportional Selection, Single-Point Crossover, Elitism) in the Glossary. These particular strategies are perhaps the most common, being recommended as defaults in some introductory texts ([Haupt, Haupt; 1997] and [Mitchell; 1998]).

The phrasing of the GA makes it a very general framework; Once the extension is made to include variable-length genomes (the domain $G = N$), it is difficult to imagine an optimization problem that does not fit into this paradigm; However, for reasons of clarity, several interpetive extensions may be added.

drawn from [Hart, Kammeyer, Belew; 1994] and [Spears, De Jong, Baeck, Fogel, Garis; 1993].

One such extension is Genetic Programming; The primary algorithm behind Genetic Programming (GP) is equivalent to the GA, save the requirement that a genome specify a program. Rather than simply map $f(g)$ in R, g is interpreted as a program, executed, with fitness assigned on the basis of its performance. Early examples of these sorts of systems include: Lisp expressions used to solve variants of the Prisoners Dilemma.; Mathematical expressions as trees, used to match polynomial expressions; and Neural Networks used to model complex behavior [Spears, De Jong, Baeck, Fogel, Garis; 1993].

A generalization of the GP extension might be a more specific and recent extension. Many GA researchers are now exploring the addition of the concept of Development, consisting of a "maturation" and a "learning" phase.

Maturation refers to a intermittent period between the genome and the evaluation in which a phenotype emerges. Hence, we posit a function

$\delta : G \text{->} Ph$; where Ph is the space of phenotypes

and modify the fitness function such that

$f : Ph \text{->} R$

The function δ might map from the space of bit string to the space of programs, trees, etc. (See section 2.3 for examples). Maturation is also commonly used in many very simple forms; For example, the widespread use of Gray codes, in which binary numbers are mapped to an alternative notation where any two successive numbers differ by at most one bit.

Learning is an extension which operates at a later date; Essentially, learning is a process by which a phenotype may be mapped to another,

$\lambda : Ph \rightarrow Ph,$

This typically involves the collection of information from the agent's environment (or, occasionally from other agents, or from the parent agents) which cause a change in the agent's phenotype. Examples of this include Lamarkian Evolution, where an agent is given all learned knowledge from its parents, or in cases where environmental changes promote changes in an agents phenotype.

Although the Bluenome model presented in this paper might be said to implement a Learning strategy (specifically in its resistance to environmental perturbations, c.f Section 4.10) I will not concentrate on the matter further. Instead, I will focus on the maturation process, with respect to a sub-field of Evolutionary Computation, Artificial Morphogenesis.

2.3 - Artificial Morphogenesis

There is a great deal of interest regarding the relationship between genotype and phenotype in artificial agents, and a variety of approaches towards Artificial Morphogenesis (a.k.a. Artificial Embryogenesis, a.k.a. Artificial Ontogenesis). These examples involve a trade-off between the amount of information explicitly encoded in the genome, versus the amount of information implicitly encoded in the maturation (developmental) process, and examples exist ranging this gradient.

There are several models in which the genome is interpreted as a program, typically utilizing modular designs of repeated structures. This approach may be said to begin with GP, as described above, but becomes more interesting in systems which interpret the genome recursively. Many such systems involve the use of Grammars, serving as programs which develop a phenotype. These systems have enjoyed much success, for example: in [Hornby, Pollack; 2002] where L-systems were used as a genotype for generating phenotype, applied to a locomotive task; or in [Kvasnicka, Pospicijal; 2002] where a binary genotype was mapped to a set of phenotype characters, for the purpose of studying modular forms.

Grammatical approaches may be said to be inspired by biology, but exist in a form so high-level and abstract that there exists little of the original metaphor. Occassionally, there are examples which challenge this statement, such as the use of a differentiation model as in [Hotz, Gomez, Pfiefer; 2003] who used the model to develop a neural network to control a foveating retina. However, the use of L-systems and Grammatical development in general is a highly abstract domain, where particular instances are often more like short-cuts and heuristics to more expansive relations between genotype and phenotype, rather than as full developmental models.

On another extreme, there is substantial work being undertaken in the modeling of actual biological development. This includes work undertaken in the simulation of internal cell dynamics, modeling the process of specialization within individual cells. A crucial factor in these experiments is the development of several high-level hierarchical principles, believed to be key factors in the development of an embryo. These principles involve the inheritance of specialization, and spontaneous symmetry-breaking [Furusawa, Kaneko; 1998]; Both these principles are fundamental aspects of the Bluenome Model. Also, there is work being undertaken in which the above principles, somewhat simplified, are applied to the task of the simulation of conglomerates of cells, modeling the development of embryos. A very successful example of this can be found in [Hamahashi,, Kitano; 1998], in a simulation of the development of a drosophilae.

The above (simulations of actual biological development) are fascinating endevours, but are typically very complex computationally. The approach undertaken in this paper is significantly simpler, and aimed not at modeling biology, but instead at the development of a tool for the generation of artificial agents.

In contrast to the pure simulations of biology above, there have been several attempts to utilize various abstractions of the developmental process in the evolution of agents designed to solve particular problems, more directly related to biological systems than the grammatical approaches listed above.

Perhaps the most similar experiment to the one presented in this paper was undertaken by [Dellaert, Beer; 1996]. This experiment aimed at the creation of a computationally-tractable but biologically-defensible model of development, aimed towards the evolution of agents capable of a locomotive task. Dellaert and Beer's model consists of a conglomerate of cellular material, which undergoes a series of differentiations. Differentiations begin as a single symmetry-breaking division, followed by divisions controlled by a series of Genetic Regulatory Networks. The Genetic Regulatory Networks are networks of interactions which accept information from an agent's genome and environment and lead to a series of state-changes within a cell. The Genetic Regulatory Networks were implemented via a series of Boolean function, labelled operons, modelling the conditional production of protein - hence, this approach may be viewed as a discrete approximation to the modelling of diffusion through differential equations.

Drawbacks to the model proposed by Dellaert and Beer resulted chiefly from the size of the search space associated with their system. For example, they were unable to evolve fit agents from scratch and hence began their experiments with a hand-coded agent, from there obtaining results. Following this, Dellaert and Beer simplified their model, using Random Boolean Networks (introduced by [Kauffman; 1993]), which are a simpler version of the original Genetic Regulatory Netwroks. As is noted by [Stanley, Miikulainen; 2003], simpler solutions may outperform biologically plausible ones, and a need exists for abstraction. A second draw-back might be the use of an initial symmetry-breaking condition in the development of an agent - while the biological motivation for such a mechanism is clear, the particular implementation is not, potentially leading to a programmer's bias in agent development.

Another similar system is proposed by [Eggenberger; 1997], applied to a simpler task. Rather tha focus on Neural Networks, Eggenberger evolved bilaterally symmetric shapes, to show that symmetry was an emergent property. Eggenberger's experiments focused on the evolution of simple three-dimensional morphologies, and demonstrated that the development of symmetry is a natural consequence. Although Eggenberger's approach was simple and aimed at a very high-level of abstraction, the demonstration that bilateral symmetry is a viable strategy demonstrates properties about the developmental approach that applications do not; That is, given a successful experiment utilizing a developmental technique to solve a problem, there is little reason to accept that other non-developmental techniques would not achieve the same results [Stanley, Miikulainen; 2003]. Another such experiment was undertaken by [Bentley, Kumar; 1999], who employed a simple grid-based model of coloured cells to evolve tessellating tiles, showing that indirect encodings can do so more reliably than several encodings which did not employ recursive interpretations of the genome.

The experiments described above contain techniques and goals similar to those found in this thesis. The primary differences between the Bluenome Model and those described above may be summarized as:

1. A higher level of abstraction in terms of differentiation - the Bluenome Model utilizes a Cellular Automata, whereas many approaches utilize more complex

models of Cell Differentiation. As can be seen in the early attempts of [Dellaert, Beer; 1996], such models introduce much complexity into the system, often to the point of making such systems computationally infeasible.

2. A more physically-minded topology, where physical mechanisms and logical control mechanisms are situated in the same space, unlike many experiments in which logical control is a second layer of morphology.
3. A rich collection of materials as initial conditions of a developing agent - in the Bluenome Model, an agent begins as a single cell, while in other models, the initial development begins as a blastocyst (a larger collection of unspecialized components). While this starting point is probably an efficient choice (given the prevalence of this process in nature), in my opinion, it is behaviour which ought exist as an emergent property of the system, rather than as a pre-condition.
4. Although in Phase Two Bluenome is a specific application, the casting of the Bluenome Model is general enough that nearly no assumptions are made regarding the application to which the system is applied. The same cannot be said for systems which specifically promote the creation of distinct constructs, such as high-level neural networks which ride atop a body.

2.4 – The Evolution of Cellular Automata

Cellular Automata (CAs) first appeared in the 1940's, created by von Neumann and Stanislaw Ulam (von Neumann; 1966) as an alternative to continuous dynamical systems. Although similar systems existed prior to this, the distinguishing feature of Ulam and von

Neumann's systems is that they considered discrete dynamical systems as a field of study in its own right, rather than as an approximation to the continuous counterpart. Discrete dynamical systems were popularized in the late 1960's by John Conway, specifically in the creation of a simple one-person game, the Game of Life (Gardner; 1970).

CAs are a model of computation which operate using only local-based information and a series of universal rules. They have been studied extensively in a wide variety of roles: firstly, as a model for computation and mathematical exploration, and secondly as a means of simulating complex systems, including swarm intelligence, colony and population dynamics, and models of economic and urban behavior, to name a few (reviewed in [Wolfram; 2002]). As a model of computation, the array of uses for CA are as wide as that of the Turing Machine, although it is still poorly understood how CAs can calculate the simplest of procedural programs.

A CA consists of a series of rules, and an infinite lattice of cells; Rules are a collection of patterns which match the state of any particular cell based on the state of its neighbours, determining a new state for the cell in the next time step. For example, in a one-dimensional CA using two colours and a neighbour radius of one ($r=1,\ c=2$), there exists a series of eight ($c^{cr+1} = 8$) rules: one for each possible combination of states of a cell and its immediate neighbours. At every time step, the successive state of a cell is determined by the matching rule.

For example, the following illustrates a one-dimensional cellular automata with two colours and a radius of one; The series of rules are represented as a series of bits, characterizing all possible combinations of a cell's state and its neighbours, along with a new state:

pattern	111	110	101	100	011	010	001	000
new state	1	0	1	1	0	1	1	0

Additionally, initial conditions are specified for the infinite lattice of cells:

…	0	0	1	1	0	1	…

A Cellular Automata proceeds through time steps, in which every cell in the lattice is updated, using the rule set above. The new state for location j is determined by matching the states of locations *j-1, j* and *j+1* to a pattern in the rule set, updating the state of cell *j* in time *t+1*:

location / time	…	-2	-1	0	1	2	3	…
0	…	0	0	1	1	0	1	…
1	…	NA	1	0	0	1	NA	…

In this manner, a CA proceeds indefinitely. There are many variations on this basic algorithm, in which additional dimensions, more colours, larger radii, alternate topologies are added. CAs are quickly shown to exhibit emergent and chaotic behavior, leading to their wide-spread study in the field of complexity.

Some work has been undertaken involving the evolution of CAs, for example, as reviewed in [Mitchell; 1996]; This work involved the evolution of one-dimensional CAs for the solution of two problems, density classification and synchronization, using stochastic fitness functions. Success was achieved at both tasks, but results were at a high computational cost, solving problems which, while daunting using only local-based information, are simple to solve in a typical computational approach (More interesting in these approaches was the method of analysis suggested, based on particle interactions). Evolved CAs are obviously in an early stage of development.

These experiments aside, search through a space of possible CAs is a daunting task. Indeed, Wolfram writes, regarding finding a method to automatically find CAs with particular properties, "I have tried hard to do this. But even using a whole range of tricks suggested by biology – as well as quite a number that are not – I have never been successful" [Wolfram; 2002]. Indeed, it is for this reason, as well as reasons of borrowing process from biology, that the Bluenome Model, a much simpler model than the general CA, has been adopted.

3. - The Bluenome Developmental Model

The Bluenome Developmental Model is a highly simplified version of biological embryogenesis. It involves the inclusion of a single component (Cell) into an array of spaces (Grid Cells), and a methodology for that cell to grow into an agent, utilizing only local information. The Cell contains a single piece of DNA, which it interprets to decide its next action - the complexity of a piece of DNA is governed by a system parameter, *numRules*, which limits its precision. The number of possible cells is governed by a system parameter, *numColours*, which limits the number of types of components which might be included in an agent – they are referred to as colours as the development model is application-independent, meaning that any components might be used, while it is convenient to represent classes of components as coloured squares in a GUI. This process is limited by a system parameter *numTel* (number of telomeres) which acts as a counter in each cell, decrementing with each action that a cell undertakes.

In brief, given a genome, development of a phenotype proceeds as follows:

1. A single neutral cell (gray) is placed in the centre of an environment (a matrix of Grid Cells)
2. For each cell in the environment, hormones are collected. That is; each cell creates a count of cell types in the twelve-neighbourhood, storing the information.
3. Each cell finds the closest matching rule in the genome, relative to the amounts of hormones collected.

4. Assuming a cell's internal *numTel* counter is non-zero, the action associated with that rule is fired; Actions are one of {divide, die, move, specialize(*colour*)}
5. Every cell which executes an action decrements its *numTel* counter
6. If the process is not complete, goto 2.

Hence, Bluenome is a model for developing agents from a structured genome of complexity proportional to ***numRules***, resulting in an agent consisting of a topology of cells selected from ***numColours*** types, with a phenotopic size proportional to ***numTel***. The algorithm for doing so is made precise below.

An agent's genome is comprised of a series of ***numRules*** rules, ***numRules*** $\in \mathbb{N}^+$. Each rule is (*numColours+2*) integers long, leading to a total genome length of (*numColours+2*)****numRules***. Each rule is structured as:

colour	*hormone$_1$*	...	*hormone$_{numColours}$*	*action*

where *colour* $\in$ {1, ..., ***numColours***}, ***hormone$_i$*** $\in$[1, 12], and *action* $\in$ [1, 20].

Initially, an agent begins as a single cell of type 1 (1 is considered a Structural cell, coloured gray), centred in an environment (a square matrix of Grid Cells). Development proceeds through the following algorithm:

When activated, a cell (*currCell*) in the environment will collect hormones from all neighbouring cells.

```
integer $t_1 = 0$
...
integer $t_{numColours} = 0$
Foreach Grid Cell $g$ in the eight-neighbourhood {
    if ($g$ contains a cell of type $K$) $t_K$ ++
}
Foreach Grid Cell $g$ in the twelve-neighbourhood which was not
                                        previously considered {
    if (there exists no cell in the Grid cell between $g$ and $currCell$
        && $g$ contains a cell of type $K$) $t_K$ ++
}
```

Hence, it will collect a count of all neighbouring cell types, except those on the periphery of the twelve-neighbourhood which are blocked by an intermediate cell. Hormones for each cell in the environment are collected before proceeding to the next step[4].

Once any particular cell has collected information regarding its neighbours, it searches the genome to find the closest matching rule, using Euclidean distance:

```
$min\_distance$ = infinity
$theRule$ = null
Foreach rule $r = <r_{colour}, r_{hormone1}, \ldots, r_{hormonenumColours}, r_{action}>$ {
        if ($r_{colour} == currCell$.colour) {
                $r_{distance} = (t_1 - r_{hormone1})^2 + \ldots + (t_{numColours} - r_{hormonenumColours})^2$
                if ($r_{distance} < min\_distance$) {
```

[4] Note that the collected hormones constitute a count of neighbouring cell types, but does not include any notion of the direction from whence the hormone came.

```
                    min_distance = $r_{distance}$
                    theRule = r
               }
          }
     }
     if (theRule != null) execute action $theRule_{action}$
     else take no action
```

With this in hand, a cell may execute its next action. This involves first decrementing the cell's internal telomere counter (hence a cell may execute only *numTel* actions), then executing the action corresponding to $theRule_{action}$. Possible actions include:

1. Die: the cell is removed from the environment, leaving an empty Grid Cell
2. Specialize(*colour*): The cell's type is changed to type *colour*, where *colour* is any cell type save the Structural type.
3. Divide: An exact copy of the cell is made and placed in the best free location in the four-neighbourhood. The best free location is defined as a free location located furthest from the centre of mass in the original cell's eight-neighbourhood (cf. Appendix B). If no free location is available, no action is taken.
4. Move: The cell is moved from its current location to the best free location. If no free location is available, no action is taken.

In this manner, a "good" genome will allow a single initial cell to grow to a robust agent. The process terminates when all telomeres are expended, or when there is no change from one developmental time step to the next. No other mechanisms are included – there are no special parameters for symmetry breaking.

One view of this process is as a subset of all 2-dimentional Cellular Automata with radius 3. The key differences between CAs and Bluenome are:

1. Bluenome begins with a single cell in the centre of a finite grid. Empty (white) cells cannot change their colour without a non-white neighbour[5].
2. As the hormone collection process does not note direction, the rules instantiated by the Bluenome genome map to symmetrical patterns in a CA rule – hence, only rules of a symmetrical form in CA space are used by Bluenome.
3. Bluenome utilizes a measure to compute distance to a rule, unlike CAs, which are precise. This may be viewed as a partition on the space of CA rules, collapsing several similar rules into a single outcome.
4. The lack of consideration of peripheral cells in the twelve-neighbourhood may be viewed as a further grouping of CA rules.

Hence, the space of all Bluenome genotypes is significantly smaller than that of two-dimensional CAs. Phase One aims to demonstrate that these simplifications do not limit the robustness of the possible phenotypes, but, due to the decreased dimensionality of the genotopic space, does allow the space of all Bluenome genotypes to be more readily pressured by selection.

[5] Similar to a Totalistic CA [Wolfram; 2003].

Figures 3.1 and 3.2 show the development of agents taken from Phase One, shown at various times in the development. The parameter *numTel* has been set to infinity, hence the agents grow to fit the allotted size in the environment. Figure 3.1 shows an uninteresting agent, the likes of which unfortunately abound from random genomes. Figure 3.2 shows the development of a more interesting agent, one which demonstrates some of the robustness characterized by CAs.

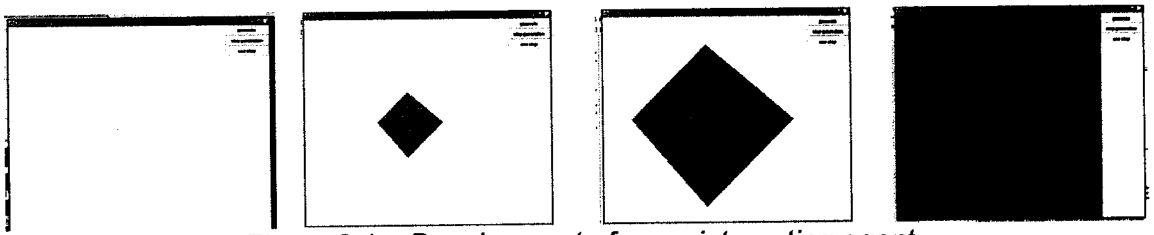


Figure 3.1 – Development of an uninteresting agent.

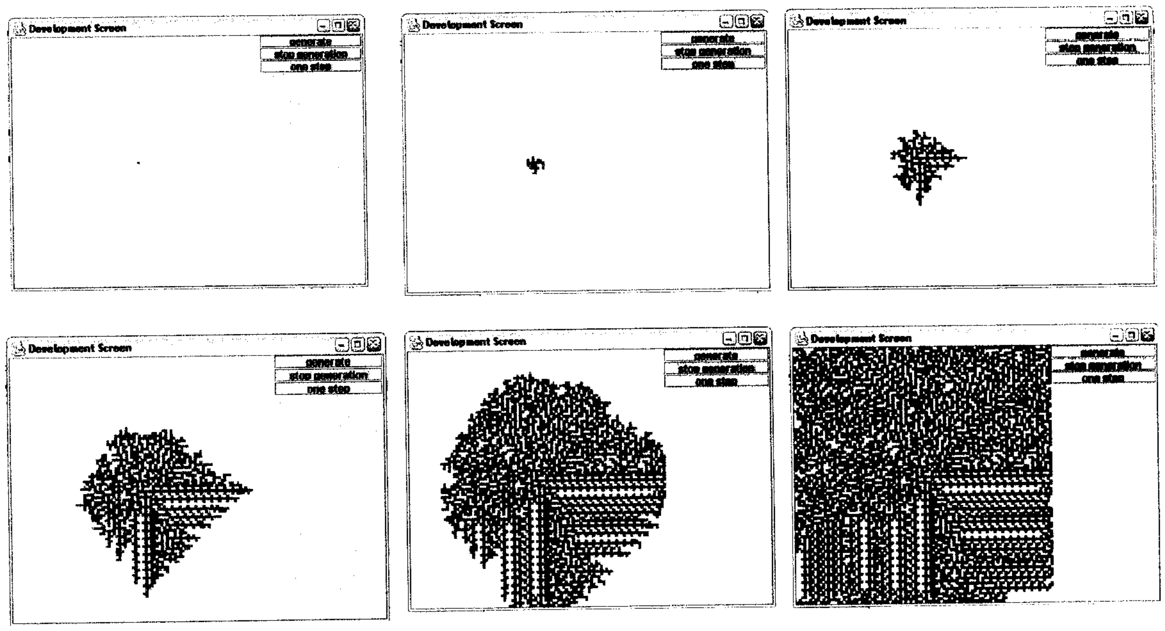


Figure 3.2 – Development of a more interesting agent.

An important question to ask regarding this method of growth is how likely a random genome is to produce a non-trivial agent. The answer is unfortunately quite dismal: A random genome with 50 rules and an unlimited number of telomeres is approximately

85% likely to generate a "trivial" pattern[6]. However, it is shown in section 4.4, and explored in more detail in section 5.8, that evolutionary pressures will quickly lower this proportion significantly.

We can now make some estimates involving size:

Firstly, we note that the maximum size of an agent with ***numTel*** telomeres will be $2*(numTel+1)^2$ – this is the size of a diamond with sides of length *(numTel+1)*. Hence, the following relation between numTel and maximum phenotopic size may be made, independent of other system parameters, summarized in Table 3.3.

Table 3.3 – Relationship between numTel and Phenotopic Size

numTel	1	6	8	10	12	20	100
maximum phenotopic size	6	98	162	242	288	882	20402

Also, as is the case in Phase One and Phase Two, the following relations hold between <***numColours***, ***numRules***> and genotopic complexity, shown in Table 3.4.

Table 3.4 – Relationship between <numColours, numRules> and genotopic complexity

numColours	6	9	10
numRules	30	50	30
genome size	240	550	360

The complexity of the developmental process is $O(numTel^3*numRules)$.

[6] By trivial, I mean a phenotype consisting of less than 3 cells, or one which contains only one cell type, as shown, for example, in Figure 3.1

4. - Phase One: Application-Neutral Experiments

4.1 - Introduction

The evolution of cellular automata is a notoriously difficult problem: the highly non-linear nature of the space of CAs, as well as their unpredictability [Wolfram; 1994] makes the prospect of the evolution of complex patterns using Genetic Algorithms seem grim. As we have recognized the Bluenome model as a subset of the space of two-dimensional CAs, it is not obvious that evolution is possible in any reasonable measure of time. Yet, intrinsic to the success of the Bluenome project is the assumption that a wide range of properties may be evolved. Phase One of this project involves a demonstration of precisely that; Given an application-neutral interpretation of developed agents, a variety of fitness functions are defined in which several properties are shown to evolve in a reasonable number of generations. In Phase One, agents are treated as images of a fixed number of colours (RGB codes) – fitness functions are designed drawing inspiration from Image Processing.

4.2 - The Model

In Phase One, agents are simply images – there is no additional interpretation. An environment is created, consisting of an array of 100 x 100 Grid Cells (unless otherwise noted). A single structural cell is placed in the centre, and development takes place as described in section 3. In Phase One, *numTel* is set to infinity; That is, agents are allowed to grow until their environment is filled. The parameter *numRules* is set to 30, and *numColours* to 6 (unless otherwise noted).

The grown agents are evaluated through a fitness function, then subjected to a Genetic Algorithm. The GA is a Standard GA [Mitchell; 1998], utilizing fitness-proportional (a.k.a. roulette-wheel) selection, with the following parameters:

Population Size:	50
Initial Population Size:	100
Proportion of Elites:	0.0
Probability of Crossover:	0.8
Probability of Mutation:	0.01

Runs are distinguished by the fitness function chosen, which are described in parts 4.3 to 4.9 of this section. These fitness functions are all defined on images, hence do not make any assumptions regarding the nature of the coloured cells, nor the development model (with the exception of sections 4.3 and 4.4, where the number of rules used in the generation of the image were included as a component of the fitness function).

The runs show exemplar members – members whose fitness most closely matches that of the mean. Appendix A contains images of highly fit members.

4.3 - Selection for Complexity

In this first example run, it is demonstrated that evolution can take place within the Bluenome system. This run utilizes 10 colours, and the simple fitness function:

$$f = 30*numColours + 5*numRulesUsed + 20*(1 - \frac{numEmptyCells}{environmentArea})$$

where *numColours* is the number of cell types present in the agent, *numRulesUsed* was the number of rules fired in the generation of the agent, *numEmptyCells* is the number of empty Grid Cells in the environment, and *environmentArea* was the number of Grid Cells in the environment.

Selective pressures proved quite effective in this example, as the population average increased quickly, while population members approaching the optimal fitness were found within the first 40 generations – Figure 4.3.1 illustrates the growth in fitness over 100 generations, while Figure 4.3.2 shows several exemplar members of the population.

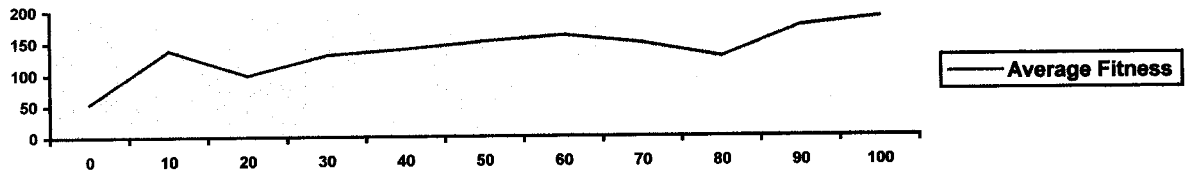


Figure 4.3.1 – Fitness plot over 100 generations

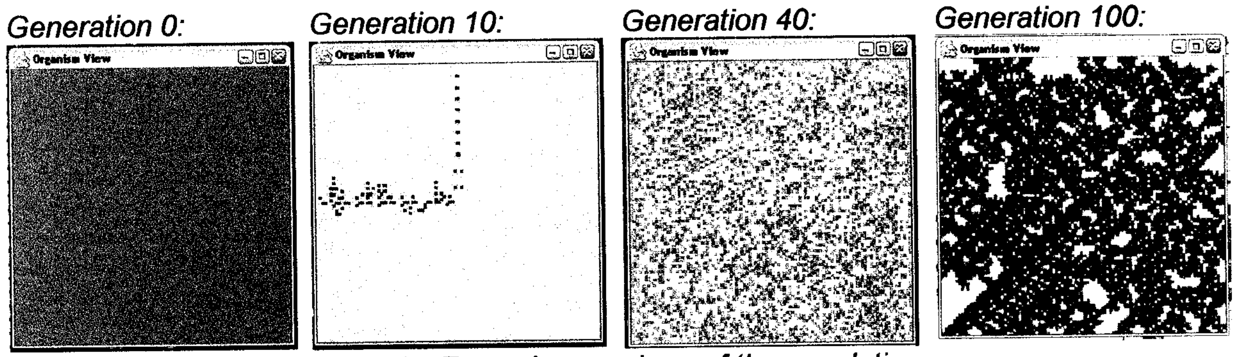


Figure 4.3.2 - Exemplar members of the population

Additionally, as generations increased (i.e. population converged toward some optima), an increased visual similarity between population members was noted – this implies significant regions in which small variations in genotype produced small variation in phenotype. Figure 4.3.3 shows several population members from generation 100, where several visual similarities may be seen.

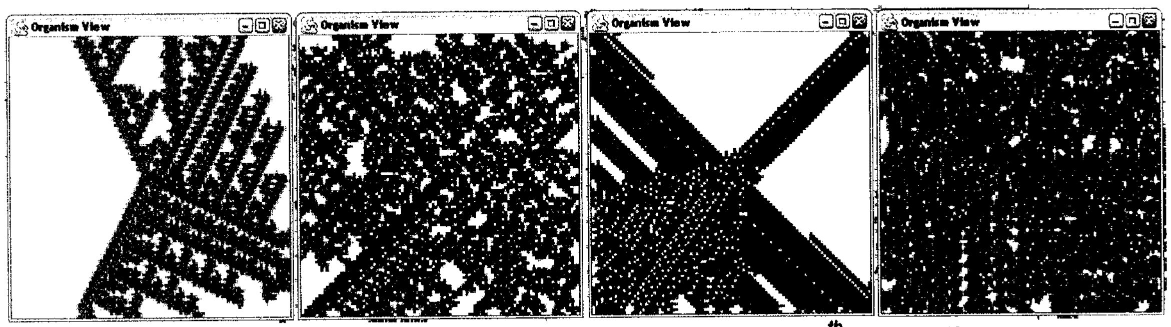


Figure 4.3.3 - 4 population members from the 100th generation

4.4 - Selection for Size

In this run we attempt to evolve organisms in a manner similar to the last, save that we select for organisms which grow to only half the size of the environment. The goal is to create organisms which themselves decide when to stop growing, rather than being limited by their environment. Several fitness functions were attempted, but to no effect. Below, is one such fitness function:

$$f = 30*numColours + 5*numRulesUsed + 20*\left(1-2*\left|\frac{numEmptyCells - 0.5*environmentArea}{environmentArea}\right|\right)$$

where numColours is the number of cell types present in the agent, numRulesUsed was the number of rules fired in the generation of the agent, numEmptyCells is the number of empty Grid Cells in the environment, and environmentArea was the number of Grid Cells in the environment

Results for this run were unsuccessful – the average fitness over a 300-generation run did not yield an increase in average fitness, save for immediately following the initial generations. Furthermore, rather than find organisms which would stop growing after a time period, organisms were found in which a large portion of the internal cells chose to die. Figure 4.4.1 shows the fitness plot for the fitness function, and figure 4.4.2 show some members which performed well under that function.

Negative results for this run are perhaps not surprising – in natural biological development, it is largely unknown how a foetus is declared "complete". The reason is suspected to be a combination of pressures from the mother[7], the foetus itself, and a specialized area of the DNA (telomeres) which control the number of cell divisions in a manner similar to that of a decreasing counter. It is for this reason that the *numTel* parameter was included in the Bluenome Model.

Figure 4.4.1 – Fitness Plot over 300 generations

[7] One theorized method of pressure is the amount of available resources – leading to the alternative tactic for growth limitations, simply providing a finite amount of food to the agent initially. This case provides an interesting alternative, unexplored in this paper.

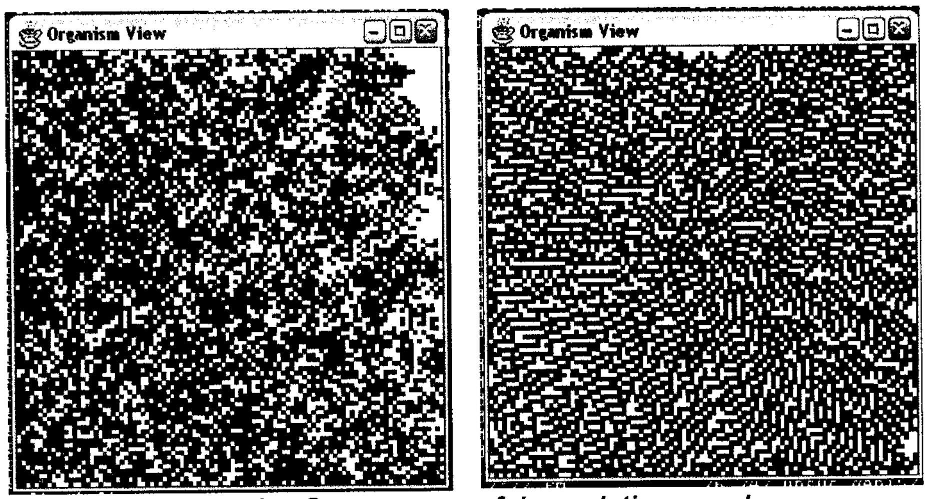


Figure 4.4.2 – Some successful population members

4.5 - Selection for Entropy

An important question for the Bluenome system is one of breadth: given that the space of phenotypes is so much smaller than the space of all possible environment-sized images, is the Bluenome system capable of creating phenotypes which are not trivial? To answer this question, a run of the system was arranged in which the selective pressure was information theoretic entropy. The goal was to attempt to generate "pure noise" – phenotypes with as little relation between cells as possible.

The assumption made in this run was that a specialized cell is more likely to occur as a neighbour of a specialized cell of the same type. From this assumption, the following uncertainties, based on an implicit pseudo-probability[8], were defined:

$$P(X_0 = i \mid X_1 = j, X_2 = k, X_r = l) = \begin{array}{l} -0.9 * \log 0.9\ ;\ i = j = k = l \\ -0.8 * \log 0.8\ ;\ i \text{ equal to two of } \{j, k, l\} \\ -0.6 * \log 0.6\ ;\ i \in \{j, k, l\} \end{array}$$

[8] Note that this could be made into a proper probability through padding values and additional definitions to cover all permutations. Since other values are plentiful and irrelevant, and since the information estimates are monotone increasing, doing so provides no advantage.

$$-0.5 * \log 0.5 \text{ ; otherwise}$$

Next, an information function was defined:

$$I(x_0, x_1, x_2, x_r) = -P(X_0=x_0 \mid X_1=x_1, X_2=x_2, X_r=x_r) * \log P(X_0=x_0 \mid X_1=x_1, X_2=x_2, X_r=x_r)$$

Finally, we obtain our fitness function:

$$f = \Sigma_{x0}\, I(x_0, x_1, x_2, x_r) - E_{min}$$

where $E_{min} = \Sigma_{x0}\, I(x_0, x_0, x_0, x_0) = -\Sigma_{x0}\, 0.9 * \log 0.9 = 929$

Note: Maximum occurs at $E_{max} = \Sigma_{x0}\ -0.5 * \log 0.5 - 929 = 2663.2$

X_0 is the cell in question, X_1 the cell immediately below it, X_2 the cell immediately to the right. The fourth term, X_r, is a cell from a random location in the environment. This parameter was included to penalize tessellating patterns.

The actual maximal value for this fitness function, E_{max}, is an overestimate of the valuation of pure noise. Since there are only 6 colours in our palette, we should only expect the least probable value (-0.5 log 0.5) to be instantiated for 5/6 of the cells in the dish. Hence, we may expect the valuation which corresponds to pure noise to be:

$$E_{exp\_max} = \frac{5}{6} * \Sigma_{x0}\ -0.5 * \log 0.5 + \frac{1}{6} * \Sigma_{x0}\ -0.6 * \log 0.6 = 2493.5$$

Results for this run were not positive with respect to evolution – after an initial jump in fitness following the first generation, the system oscillated randomly within a large range. Particular individuals were found which achieved fitness values which approached the maximal expected entropy, but it is unlikely that these were found for reasons other than random chance. Figure 4.5.1 shows a fitness plot for the above fitness function, and Figure 4.5.2 shows some successful agents.

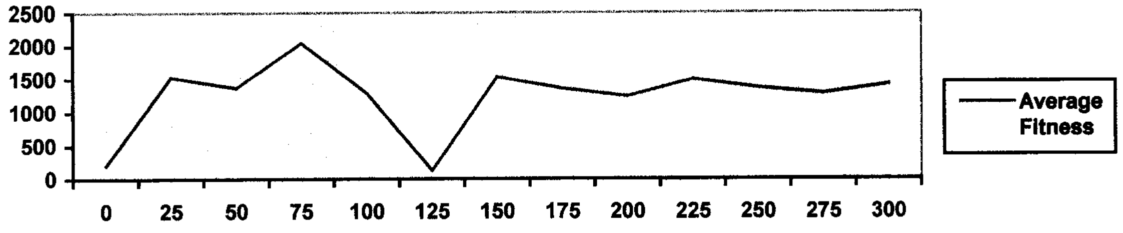


Figure 4.5.1 – Fitness plot over 300 generations

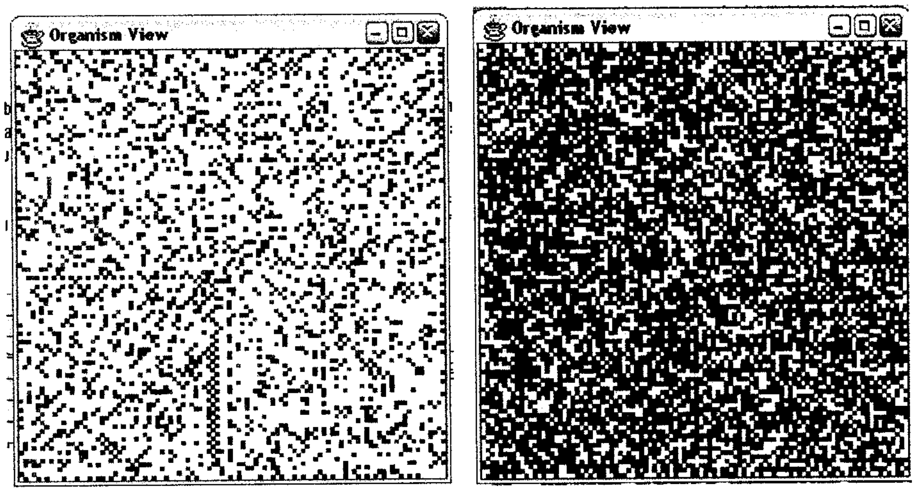


Figure 4.5.2 - Some successful population members.

Negative results for this run are not at all surprising – the generative process in Bluenome was chosen precisely to facilitate the creation of hierarchical structure, and is a deterministic process (with respect to a particular genotype). That individual members which approached pure noise were found is sufficient to achieve the goals of this run, whether or not evolution could occur for this run is secondary.

Indeed, the inability of the system to evolve potentially indicates a lack of continuous regions in the genotopic space which are inherently random – this may be viewed as an indication that the hierarchical organization resulting from inherited specialization (typically stressed by researchers modeling cell differentiation) permeates the entire space, rather than only subsets, as is the case with two-dimensional CAs.

4.6 - Processing Techniques

Pattern Flattening: Introduced here is a new technique for image processing; One issue with the Bluenome generated images is that an "organ" might be more than simply a region of specialized cells. We wish to allow for a repeating pattern of specialized cells in addition. Pattern Flattening is a technique in which dominant patterns are detected, and replaced in the phenotype with a new colour. In essence, pattern flattening allows small neighbourhoods of cells to be processed as though they were regions of identical specialized cells. Illustrated in Figure 4.6.1.

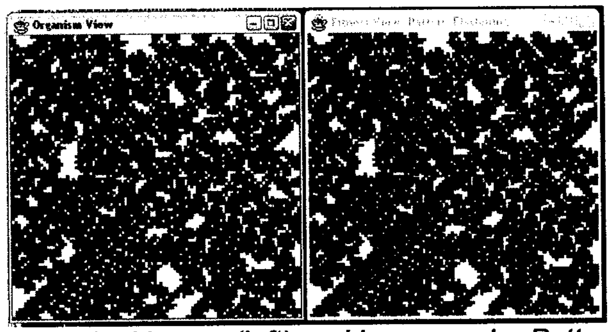


Figure 4.6.1 – Original image (left) and image under Pattern Flattening.

Boundary Detection: A process used after thresholding which extracts only the boundary. This is done by cycling through all black cells, deleting those which have four adjacent black neighbours.

Closed Thresholding: A process intended to threshold, keeping only points which lie within a closed boundary. All points within a regular threshold are considered – for each point, a walk is taken in four directions. If, in any of the directions, no white cell is found before the edge of the environment, the point is deleted.

This technique is not guaranteed to succeed – only to provide an approximation to success. When this technique fails, it provides an over-estimate. For computational reasons, this approximation is necessary, as computation of boundaries based on connectivity is far too complex for repeated application. Illustrated in Figure 4.6.2.

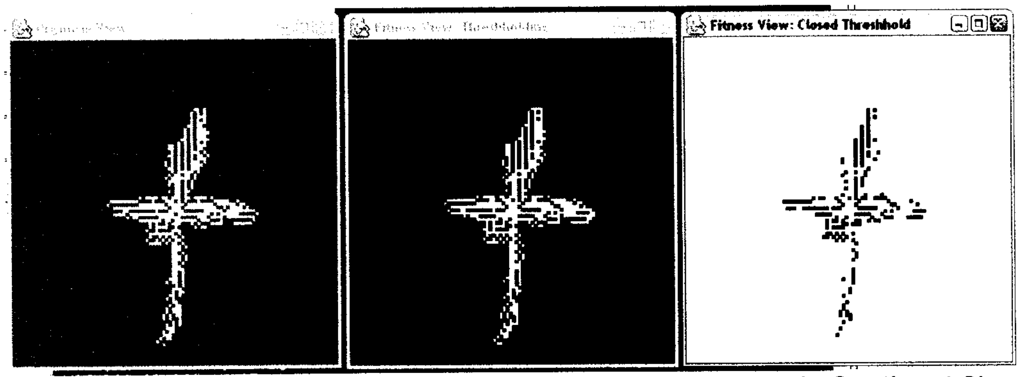


Figure 4.6.2 – Left: original image (drawn from the run in Section 4.9), Centre: Threshholded image, Right: Closed Threshholded Image

Closed Boundary: The boundary of a closed threshold, again computed by deleting all black squares with four adjacent black neighbours. This process also falls victim to the approximation of the Closed Threshholding.

4.7 - Selection for Maximal Connected Thin Coverage

In this run, we attempt to select for a thin coverage; That is, we choose the dominant colour or pattern and define a valuation of the organism on the basis of how well it reaches all cells in the organism, with regularization on the basis of how connected the dominant colour or pattern is. The goal of this endevour is to show that Bluenome is capable of generating a "transport system" – that is, a structure which might be used to transport materials to as many cells in the environment as possible, while taking the smallest amount of space.

The fitness function:

$$f = \Sigma_{\{x0\ \varepsilon\ I\}}\ \varphi(x_0)$$
$$\varphi(x_0) = \begin{cases} 10; & x_0 = \textit{black}, \textit{numBlackNeighbours} = 2 \\ 3; & x_0 = \textit{black}, \textit{numBlackNeighbours} = 1, 3 \\ -2; & x_0 = \textit{black}, \textit{numBlackNeighbours} = 8 \\ -28; & x_0 = \textit{black}, \textit{numBlackNeighbours} = 0 \\ -1; & x_0 = \textit{black}, \textit{otherwise} \\ 3; & x_0 = \textit{white}, \textit{numBlackNeighbours} = 1, 2, 3 \\ -2; & x_0 = \textit{white}, \textit{numBlackNeighbours} = 0 \\ -1; & \textit{otherwise} \end{cases}$$

where *numBlackNeighbours* is the number of neighbours of Grid Cell (x_0) coloured black. The above valuation works on a point basis – points are allotted when a particular cell is in contact with a few "covering cells", and removed if there are too many covering cells or none at all. Valuation at a covering cell is designed to reward connectivity, but penalize large continuous regions of covering cells. The results are quite good – after only a hundred generations an organism was found which covered over 95% of the surface of the environment in only a few disconnected "trails", with virtually no contiguous regions of covering cells.

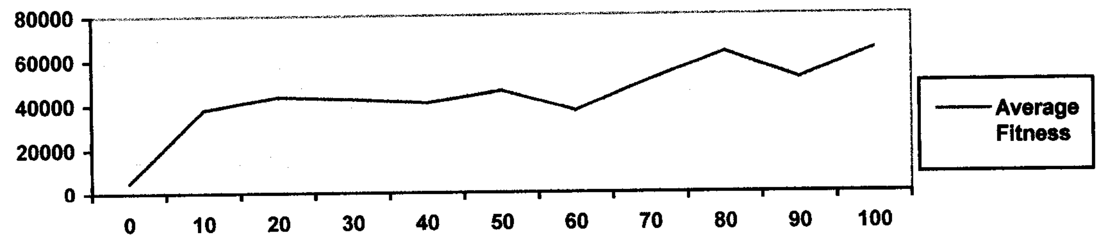


Figure 4.7.1 – Fitness Plot over 100 generations

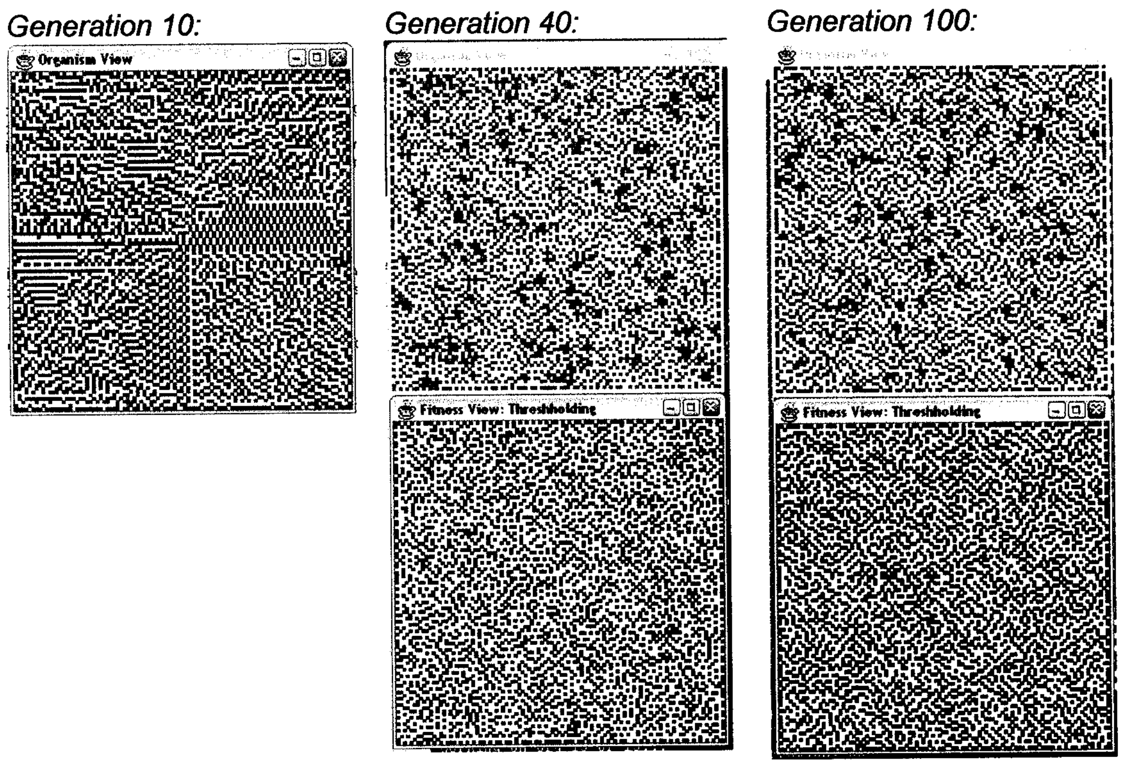


Figure 4.7.2 - Exemplar Population Members

4.8 - Selection for Disconnected Similar Regions

In this run we attempt to select for disconnected regions of similarity. This is a key feature in complex designs, as a large organization will typically re-use specialized components in several different components.

There is a difficulty in measuring disconnectivity similar to the difficulty to measuring connectivity – it is computationally cumbersome. Instead of dividing the environment into connected regions, the system provides a measure of disconnectivity by attempting to draw lines between threshholded regions. All possible horizontal, vertical and increasing/decreasing lines are attempted, where fitness is rewarded for every line which consists entirely of white cells, with black cells existing on either side.

Our process is:

a. pattern flattening
b. closed threshholding
c. attempt at fitting straight lines

The valuation function is:

$$f = 100 * numLines + \frac{numCells_{closedThreshhold}}{10}$$

where *numLines* is the number of straight lines successfully fit between regions of the dominant pattern, and $numCells_{closedThreshhold}$ is the number of cells within the closed threshold. Results in terms of evolvability were quite promising – the system showed marked improvement in terms of fitness very quickly. The members of the final population proved quite sparse, however – given further valuation and less weight to the *numLines* component in the fitness function, this would probably correct itself.

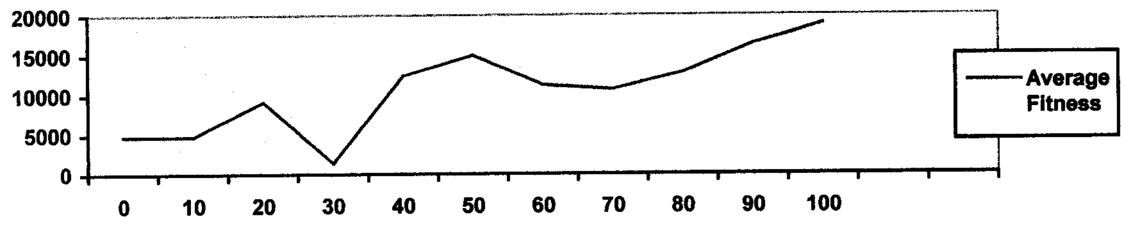


Figure 4.8.1 – Fitness Plot over 100 generations

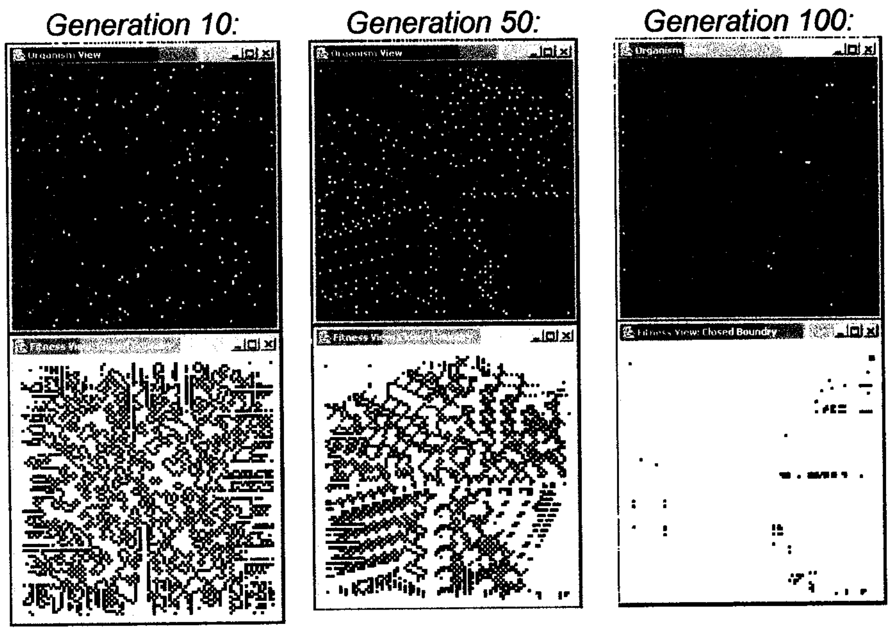


Figure 4.8.2 – Examplar members

4.9 - Selection for Disconnected Distinct Regions

In this run we attempt to select for disconnected regions of distinction. This feature is important is demonstrating the ability of the Bluenome system in generating fields of specialization. The metaphor for this would be the development of distinct organs in a biological organism.

Our process is:

1. double pattern flattening (same as pattern flattening, save that two patterns are chosen)
2. double thresholding (shown as black and cyan)
3. attempt at fitting straight lines between areas of black and cyan

The valuation function is:

$$f = 100*numLines + \frac{numCells_{closedThreshhold}}{10}$$

where *numLines* is the number of straight lines successfully fit between the dominant patterns, and $numCells_{closedThreshhold}$ is the number of cells within the closed threshold for both dominant patterns.

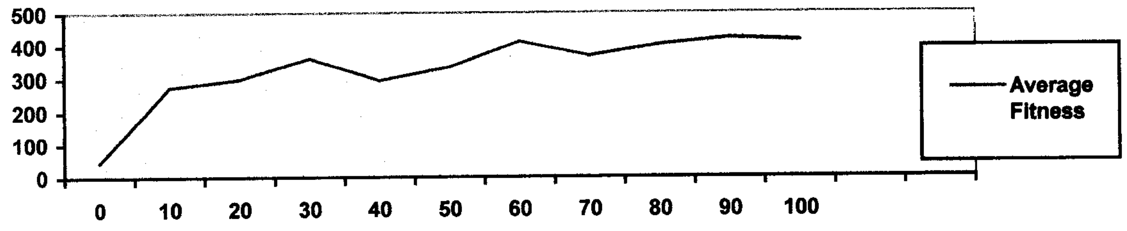


Figure 4.9.1 – Fitness Plot over 100 generations

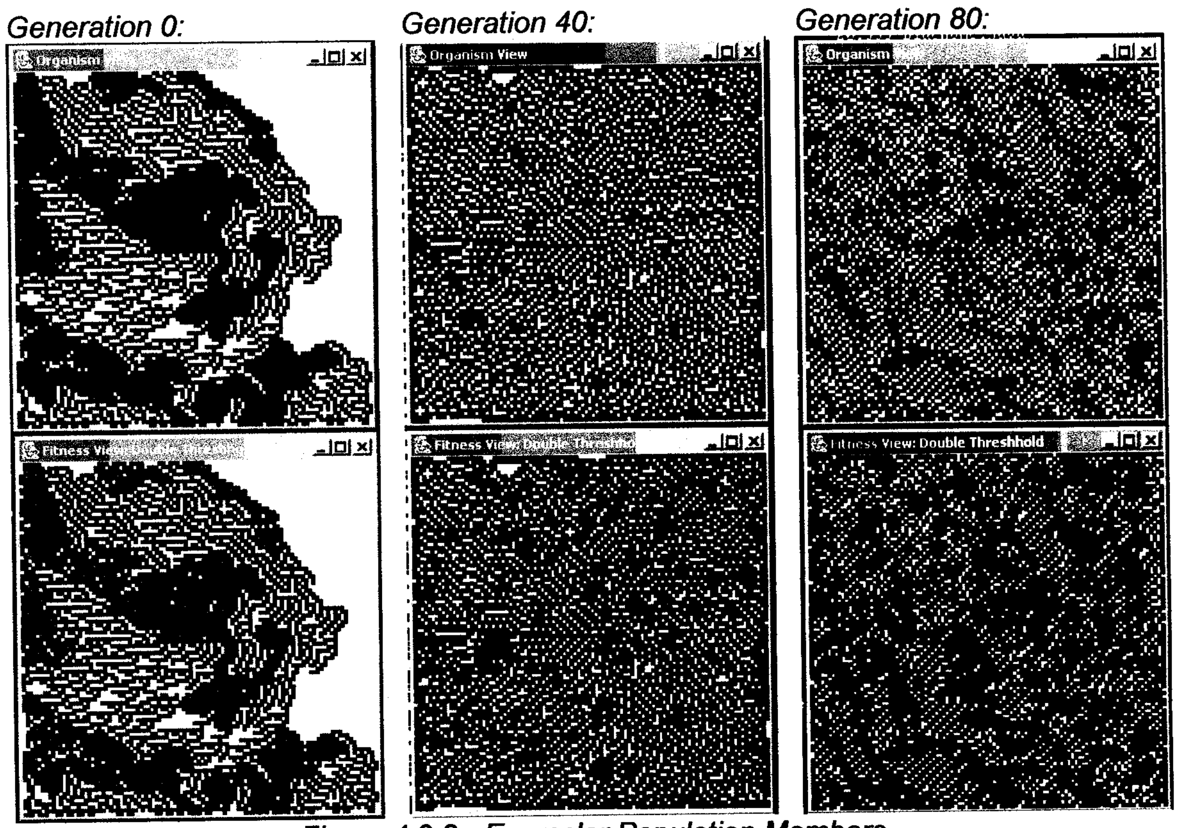

Generation 0:
Generation 40:
Generation 80:
Organism
Organism View
Organism
Fitness View: Double Threshhold
Fitness View: Double Threshhold
Fitness View: Double Threshhold

Figure 4.9.2 - Exemplar Population Members

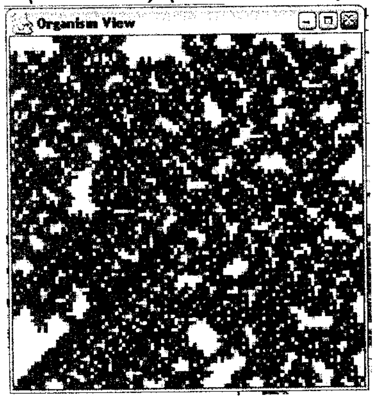
4.10 - Resistance to Pertubations in Environment

An important feature in the generation of a phenotype from a genotype is a resistance to perturbations. A pertubation assumes one of two forms: a subtle variation in the genotype, or a variation in the environment in which the phenotype will express itself.

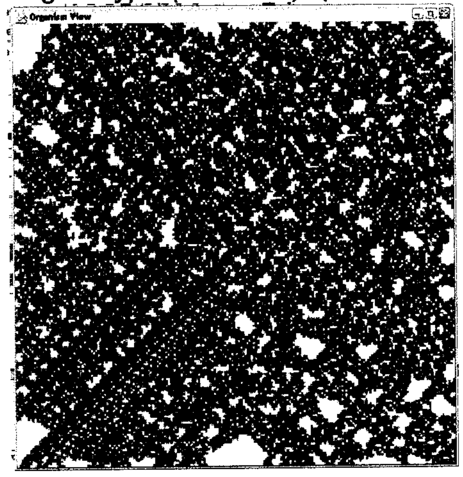
Subtle variations in genotype are demonstrated at length in the preceding sections. That evolution occurs at all in the Bluenome runs demonstrates that there are many axis along which there exists a continuity between genotype and phenotype. The existence of trivial and empty phenotypes in virtually all generations of all runs, however, demonstrates that this continuity is routinely broken. The obvious suspect points in the genotype are the first and last bits in a "rule" (c.f. Section 3) – those which specify the colour which dominates the rule, and those which specify which action the rule will execute. It is possible to modify the crossover and mutation processes to add "protection" to these sites, in the form of a decreased probability of destruction. In the interest of simplicity of developmental process, however, this has not been done.

Variations in the environment are also an interesting matter – indeed, this is an area in which a generative process between genotype and phenotype bears its most distinct advantage relative to a more bijective relation. Figure 4.10.1 shows the result of application of perturbations to the environment. The results are very promising, both in terms of preservation of visual features, and in terms of constancy of fitness evaluation (using the function defined in section 4.3).

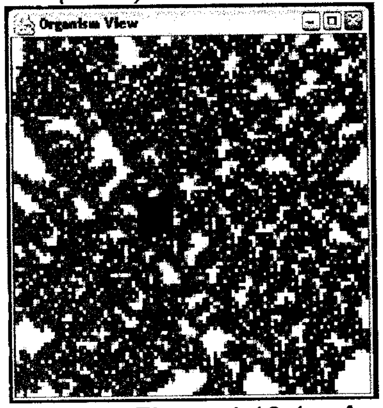
Original Phenotype (100x100):(Fitness: 230)



The same genotype (200x200): (Fitness: 230)



Phenotype with "foreign" cells (black): Fitness: 260



The same genotype (100x200): Fitness: 240

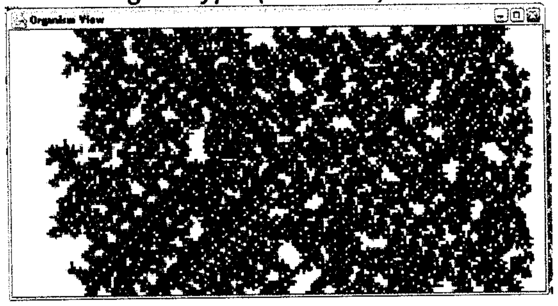


Figure 4.10.1 – A genotype extracted from 4.3 generation 100, subjected to various perturbations.

5. - Phase Two: Application

5.1 - Introduction

Phase Two of this thesis presents the application of the Bluenome Model of Development to a non-trivial evolutionary task. The task chosen is new, consisting of a situation inspired by biology. The goal of the experiments in Phase Two is the evolution of multi-cellular agents, capable of surviving as long as possible in an artificial world.

An artificial agent, the Tarasanoid, is presented; A Tarasanoid is a collection of cells, of type chosen from the set { Eye, Nerve, Foot, Transport, Structural}. These cells are laid out (connected) in a matrix of Grid Cells, with a defined set of interactions between them. Each cell is provided with an amount of food initially, a cell using one unit per discrete time step. Also in this environment are laid out patches of food, which transport cells may absorb and distribute. To survive longer, a Tarasanoid must detect this food, move over top of it, absorb it, and distribute it to the remainder of cells in its body. All cells are capable of local interactions only – a cell communicates or passes food only in its local neighbourhood. The measure of a fit Tarasanoid is a measure of the amount of time it survives, and the number of cells surviving at any given time step. Figure 5.1.1 shows a snap shot of the GUI for Phase Two.

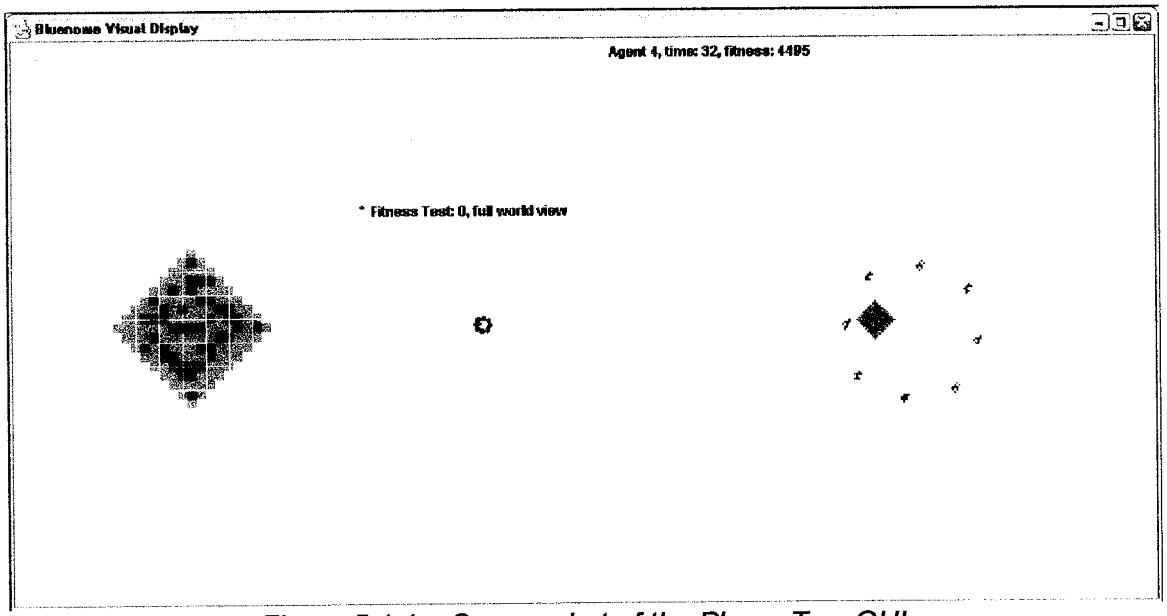

Bluenome Visual Display
Agent 4, time: 32, fitness: 4495
Fitness Test: 0, full world view

Figure 5.1.1 – Screen shot of the Phase Two GUI.
The panel on the far left shows initially the Tarasanoid's development, and later the fully developed Tarasanoid independent of the current fitness evaluation (The thin gray lines are for reference only). The two panels on the right show the Tarasanoid in the world, at different levels of zoom. Food patches are coloured dark green, agent cells vary in colour, indicating cell type.

The task presented was chosen for several reasons:

It is a highly complicated task, one for which a human designer would experience difficulty.

1. It is a task which easily lends itself to any phenotopic complexity.
2. Its solution may potentially involve mechanisms found in nature.
3. It combines the need for an internal agent logic embedded as a physical component in the agent.

The combination of these criteria form a robust platform in which the Bluenome system may be tested.

Additionally, Bluenome is not the only mode of development which is applied to this task – a second method is presented, one in which the relation between genotype and phenotype is bijective. The purpose of this inclusion is to evaluate the claims made regarding the developmental process in agent design (cf. Section 1) – to attempt to demonstrate that a developmental model may outperform a bijective model. This is not a claim that the Bluenome method is the best available, however, rather that it is simply a viable option for high-dimensional design problems.

5.2 – The Worlds

A world is an infinite two-dimensional matrix of Grid Cells. Each world contains one Tarasanoid at the centre, and a distribution of food. The centre of the world is defined to be the centre of the Tarasanoid, with a Tarasanoid's movement rotating or translating the world as required. There are no collisions - instead, a Tarasanoid will pass directly over top of food in the world, possibly absorbing it.

Food is parceled in food pieces, each occupying one Grid Cell, having a value of $2*(numTel+1)^2$ food units. Each world contains precisely 80 such parcels, the distribution depending on world type. Hence, each world contains precisely $160*(numTel+1)^2$ units of food.

Type 0 Worlds: Type 0 worlds contain eight batches of food, each consisting of a square of 10 food parcels. The parcels are centred at the Grid Cell locations: <0,50>, <30,30>,

<50,0>, <30,-30>, <0,-50>, <-50,0>, <-30,-30>, <-30,30>. This is an "easy" world, in that random motion will typically find some food. In the numTel = 20 runs, these values were increased to: <0,70>, <40,40>, <70,0>, <40,-40>, <0,-70>, <-70,0>, <-40,-40>, <-40,40>.

Type 1 Worlds: Type 1 worlds consist of a line of four batches of food, each of increasing distance. A direction is chosen at random and food is placed in a line, in batches of 20 parcels, at distances of 40, 80, 120 and 180 units (60, 120, 180 and 240 units in the *numTel = 20* runs). These worlds test the ability of an agent to see food and move towards it in an arbitrary direction.

Type 2 Worlds: Type 2 worlds consist of four batches of 50 parcels of food, each placed at a random location at a distance of 80 units (120 units for the *numTel = 20* runs) from the agent's start position. These worlds test the agent's ability to search for food when none is within immediate visual range.

Type 3 Worlds: Type 3 worlds consist of 40 batches of 2 parcels of food, distributed randomly at a distance of between 40 and 140 (between 60 and 180 units for the *numTel = 20* runs) units from the Tarasanoid's start position. These worlds test the agent's ability to quickly move about gathering food.

5.3 – Tarasanoids

A Tarasanoid is a collection of one or more cells, assumed to be connected, located in a two-dimensional grid. Each cell occupies one grid location. Tarasanoids behave as the sum of the behaviours of their cells. So long as one cell is declared "active", a Tarasanoid is declared "active" - otherwise "inactive".

The size of a Tarasanoid is determined largely by the system parameter, *numTel*. Tarasanoid size (in either mode of development) is bounded above by $2*(numTel+1)^2$ (the size of a diamond with sides $numTel + 1$) and typically approaches that number.

Cells may be viewed as independent agents of their own right - each maintains a food supply, and executes a particular program based on input and internal variables. Cells may communicate and pass food between adjacent cells (four or eight-neighbourhoods). A cell is "active" (coloured) if its food supply is greater than zero, otherwise "inactive" (black). An inactive cell will continue to occupy physical space, but will no longer be capable of processing input or output.

All cells belong to one of the following classes: Eyes (Green), Nerves (Orange), Feet (Blue), Transports (Red) and Structure Cells (Gray), with some subclasses existing between those categories.

Eyes: Eye cells can sense their external environment, and return a boolean value on the basis of the existence of food. An eye cell sees in a radius about it, accepting a value of 1 if food exists in that radius, 0 otherwise. Amount of food is irrelevant. However, the presence of other cells within its field of vision will block its ability to sense – hence, an eye cell must be located on the periphery of a Tarasanoid in order to be capable of functioning.

The question of whether or not a piece of food can be seen is calculated as follows (assuming the distance between the food and the eye cell is less than the radius of the eye cell type):

1. A linear function between the cell and the food is computed
2. That function is discretized as a series <x,y> locations in the Grid at intervals of less than one unit

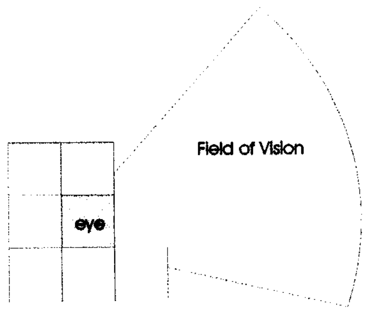


Figure 5.3.1 – Illustration of an eye cell with associated field of vision.

If there exists a cell in any of those locations, the food cannot be seen. Note that this leads to a slightly non-intuitive cone shape, due to the rounding involved in discretization. This is illustrated in figure 5.3.1.

Eye cells come in two types: “long” (radius of 100 units) and “short” (radius of 50 units).

Nerves: Nerves are cells which accept information from neighbouring eye or nerve cells, and output information to neighbouring nerve or foot cells. Each nerve can have up to four inputs, four outputs, or any combination thereof, corresponding to the four sides of the cell. Whether a side is an input or an output is determined by its layout in a particular Tarasanoid.

Nerves collect all input from activated nerves and eyes, and store the sum of those inputs. When called upon for output, a nerve will output the value:

$\Sigma_{input\ in}\ in$; for "identity" nerves

$-\Sigma_{input\ in}\ in$; for "inversion" nerves

$(\Sigma_{input\ in}\ in) + r$; for "random" nerves, where r is a random value in $\{-1,0,1\}$

Feet: Foot cells accept input from all neighbouring nerve cells, and store it as a sum. At the end of a time step, all foot cells contribute that sum to a total, which defines the movement of a Tarasanoid relative to its centre of mass. Foot cells come in two sub-classes: "forward" and "turn" – forward cells contribute to forward movement (left, in the GUI), and turn cells contribute to a counter-clockwise turn. Note that a foot cell can receive a negative value, causing backward or clockwise motion.

Movement of a Tarasanoid is weighted by the number of cells in that Tarasanoid. For example, if the total of forward feet contributions is 100, a Tarasanoid will move forward by *100*(1–0.002*Tarasanoid_size)* units, where *Tarasanoid_size* is the number of cells in the Tarasanoid. If the total of turn feet contributions is 100, a Tarasanoid will turn by

100(1 – 0.002* Tarasanoid_size)* degrees.

Transports: Transport cells manage the collection and distribution of food. At each time step, a transport cell will:

1. Collect any food from its environment (four-neighbourhood plus the Grid Cell which the transport occupies)
2. Distribute food to its neighbouring non-transport cells (eight-neighbourhood)
3. Normalize the amount of food which it contains relative to its neighbour transport cells. i.e. if it contains more food than an adjacent transport, the difference will be computed, and half that difference passed to the neighbour.

In this manner, a transport cell may absorb food from the environment and gradually pass that food to all cells in an area connected by other transport cells.

Structure Cells: Structure cells serve no purpose in the functioning of a Tarasanoid. They exist as a starting point for the Bluenome development model, but may remain as part of a Tarasanoid following its development. As such, they have been included as a possibility in the bijective model as well.

5.4 - A Tarasanoid in the World

A Tarasanoid is initialized in the centre of a world, each cell containing 200 units of food, with time defined as zero[9]. The Tarasanoid next executes the following process at every time step:

```
foreach cell c
    c eats one unit of food from its supply
    if c has no food left, it is declared "inactive"
    if there are no active cells, the Tarasanoid is declared "inactive",
                                        program termination

// any cells referenced later than this point are assumed to be
                                                            active.

foreach transport t
    foreach GridCell g which t occupies, or which exists in the four-
    neighbourhood
        t eats food at g
        foreach non-transport cell c in the adjacent eight-
        neighbourhood
            if (t.foodSupply > c.foodSupply + 8)
                pass 2 units of food to c
        foreach transport cell in the adjacent eight-neighbourhood t2
            diff = the difference between t.foodSupply and
                    t2.foodSupply / 2
            pass diff between t and t2
```

[9] Hence, if a Tarasanoid does not find food, all cells will die at time 200; Typically, most cells do, as most Tarasanoids have imperfect methods for food distribution.

```
foreach eye
   if there exists a piece of food within the field of vision, set output
                                                                   value as 1
   otherwise, set output value as 0

compute the depth of each cell. Eyes have a depth of zero, for all other
cells depth is equal to the Manhattan metric from the closest eye.

for k from 1 to MAX_DEPTH
   foreach nerve of depth k
      collect input from adjacent nerves of depth k-1 (or eyes, if
                                                                  k=1)
      sum the input
      prepare output on the basis of nerve type

foreach foot f
   collect input from adjacent nerve cells
   if (f of type "forward")
      forward_sum += f.output * (1 – 0.002* Tarasanoid_size)
   if (f of type "turn")
      rotation_sum += f.output* (1 0.002*Tarasanoid_size)

move the Tarasanoid, first rotating by rotation_sum, then translating
forward by forward_sum
```

Note 1: in a system in which a random nerve cell is not connected to an eye, it will still output a random value in {-1,0,1}.

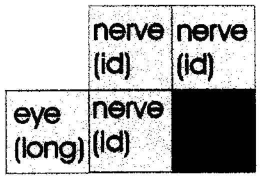


Figure 5.4.1 – Illustration of Nerve connections between Eye Cell and Foot Cell.

Note 2: in the above system, all nerve cells activate prior to feet accepting input. Hence, in the situation illustrated in Figure 5.4.1, the input to the foot cell would be 2, rather than 1 (assuming the eye has output a value of 1).

Examples:

Figure 5.4.2 is an illustration of one of the simplest Tarasanoids capable of finding and absorbing food. The process at its current location in the world is:

1. The Transport cell does not absorb any food, since there is none in its four-neighbourhood. If there is an abundance of food in its store (relative to neighbouring cells), it will distribute food to its eight-neighbourhood.
2. The Eye cell accepts a value of 1, since the food is within its field of vision
3. The Nerve cell accepts as input of 1, prepares an output value of 1
4. The Foot cell accepts a value of 1
5. The sum of all Foot cells values is one, hence the Tarasanoid moves forward one unit. (Normally this would be penalized on the basis of the size of the agent)

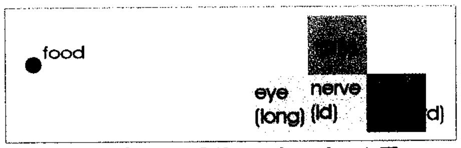


Figure 5.4.2 – One of the simplest Tarasanoids capable of finding and absorbing food.

As a curiosity, consider Figure 5.4.3, an agent in a similar situation; This agent's actions would cancel each other out, leading to an immobile agent.[10]

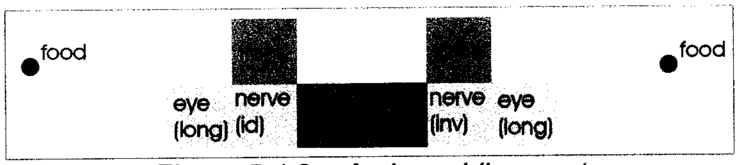


Figure 5.4.3 – An immobile agent

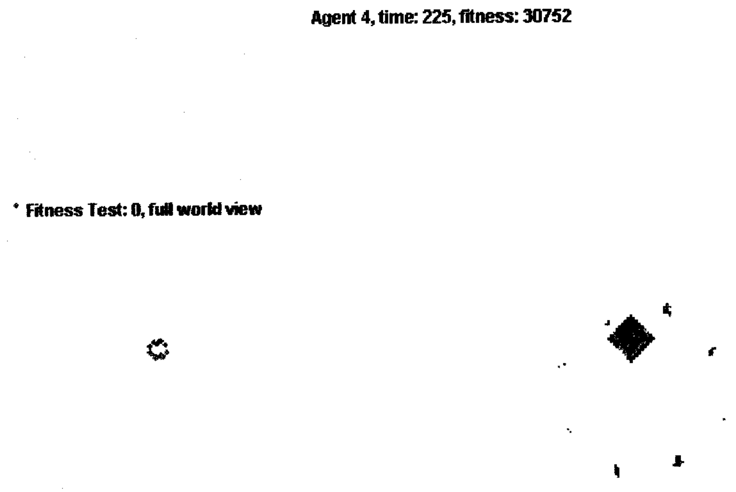

Agent 4, time: 225, fitness: 30752
• Fitness Test: 0, full world view

Figure 5.4.4 – Screen shot of a Tarasanoid in a type 0 World. This shot is taken at time 225 from run bn8, generation 30. The majority of cells in the Tarasanoid have been declared "inactive" (coloured black), but some have survived.

[10] Schopenhauer, eat your heart out.

5.5 - Development

The reader will note that the above definition of a Tarasanoid makes no assumption regarding its representation or development. In Phase Two, two methods are used: the Bluenome method, and a Bijective method.

The Bluenome Method: The method of growth for a Tarasanoid in a Bluenome run corresponds to the description provided in section 3. Unlike the implementation in Phase One, the growth of a Tarasanoid is controlled through the ***numTel*** parameter, implying a maximum size of $2*(numTel+1)^2$.

The Bijective Method: The bijective method of Tarasanoid development is a simple model, in which there exists a one-to-one correspondence between elements in the genome, and cells present in the agent. The genome for an agent consists of an array of integer values, all between 0 and 9, inclusively. A Bijective Tarasanoid is developed by laying out the values of those integers, one by one, in a spiral pattern, eventually forming a diamond of area $2*(numTel+1)^2$. The genome values are mapped to cell types, where the 0 value is mapped to the empty cell. The spiral layout begins with the central point, and proceeds biased downwards and clock-wise (see Appendix B for details). Figure 5.5.1 shows some genotopic complexities for values of ***numTel***. It is evident that these values quickly become too difficult for a standard GA.

Table 5.5.1 – Genotopic complexities per values of numTel in the Bijective model of development

numTel	1	6	8	12	20
Genotopic Complexity	8	98	182	288	882

5.6 - Fitness

The fitness of a Tarasanoid in the world is a measure of its size and length of life[11]. Additionally, to help guide the evolution of the Bluenome version, a bonus has been added. This bonus is meant to artificially inflate the fitness of partially developed agents in the initial generations – this was included after initial tests showed early stagnation. That is, agents which showed some non-trivial development were awarded the same rank as trivial agents by the bonus-free fitness function. The bonus rewards agents which grows to maximal size and utilize several classes of cells, since these agents are the most likely to breed successful agents in future generations.

Base fitness is the sum of the number of living cells in the lifespan of an agent, in a world W.

$$fitness_{base}(a)^{W} = \Sigma_t\, numCells(a,t)$$

where *numCells(a,t)* is the number of living cells in agent *a* at time *t*. Note that since the amount of food in a world is finite, so is $fitness_{base}$. Also note that this fitness function implicitly rewards agents of larger size.

From the discussion of a Tarasanoid above, it is obvious that any successful Tarasanoid will need to contain cells from a variety of classes. Hence, a Tarasanoid containing cells of many types should be considered more fit than a Tarasanoid containing only one or

[11] An attractive initial hypothesis for a fitness function might be simply the length of time that a Tarasanoid survives in the world. However, this does not consider the possibility that a Tarasanoid with a partially developed transport system might collect some initial food, hoarding it in a single cell. In this case, the single cell will survive for a large amount of

two. We define *numClasses* as the number of classes of cell types found in a Tarasanoid; Hence, *numClasses* $\in$ [4] is the number of those classes for which at least one cell exists in the fully developed Tarasanoid.

$$numClasses(a) = \Sigma_{class \in \{eye, nerve, foot, transport\}} exists(class, a)$$

where *exists(class,a)* returns 1 if agent *a* contains at least one cell of type *class*, 0 otherwise

$$fitness_{bonus}(a) = numClasses\ (a)^2 * 20 * (numTel+1)^2$$

where *numTel* is the number of telomeres.

Indeed, the fitness bonus is nearly always maximized for bijective agents. However, it was retained for both Bluenome and Bijective runs of the system, for reasons of avoiding bias and easy comparison between the two. It implies, however, a large increase in fitness in the early generations of the Bijective runs, relative to the Bluenome runs.

Finally, our fitness function is:

$$fitness(a)^{W} = fitness_{base}(a)^{W} + fitness_{bonus}\ (a)$$

In any particular generation, an agent *a* will be subjected to two worlds, w_1 and w_2. Hence, at any generation, the fitness returned to the GA is:

$$fitness(a) = fitness(a)^{W1} + fitness(a)^{W2}$$

time, while the majority of the agent will die. The base fitness function defined above removes any bonus for such behavior.

Note that fitness is relative to the worlds provided, as well as to the behavior of random elements in an agent. Although elitism is present in Phase Two, this does not imply that fitness between generations will be monotone increasing.

Figure 5.6.1 shows the minimum-bonus fitness (the fitness of an agent which has maximized its size and $fitness_{bonus}$, but which does not collect any food) and the maximum fitness of agents relative to the parameter *numTel*.

Table 5.6.1 – Minimum-bonus and Maximum fitness values for numTel $\in \{6, 8, 20\}$

numTel	6	8	20
Minimum-bonus Fitness	70 560	116 600	645 040
Maximum Fitness	101 920	168 440	927 280

5.7 – Experiments

A series of eleven experiments were executed: six utilizing the Bluenome model of development (the Bluenome runs), and five utilizing the Bijective model of development (the Bijective runs). The Genetic Algorithm used corresponds to the Standard Genetic Algorithm, utilizing elitism and fitness-proportional (a.k.a. roulette-wheel) selection [Mitchell; 1998]. All experiments in Phase Two used the following GA parameters:

Proportion of Elites:	0.1
Probability of Crossover:	0.9
Probability of Mutation:	0.01
Number of Rules (Bluenome runs):	50

The runs are distinguished by the *numTel* parameter. Bluenome runs are labeled "bn", followed by the value of *numTel*, shown in Table 5.7.1.

Table 5.7.1 – Parameters for the Bluenome Runs.

run	bn.6	bn.8	bn.12	bn.20.1	bn.20.2	bn.20.3
numTel	6	8	12	20	20	20
population	200	200	200	50	50	50
initial population	400	400	400	100	100	100
maximum generation	100	100	100	200	200	200

Bijective runs are labeled "bj", followed by the value of *numTel*, described in Table 5.7.2. Note that in all runs in which *numTel* = *20*, the distance to food locations from the centre of the world was increased (c.f. section 5.2).

Table 5.7.2 – Parameters for the Bijective Runs.

run	bj.6	bj.8	bj.12	bj.20.1	bj.20.2	bj.20.3
numTel	6	8	12	20	20	20
population	200	200	200	50	50	50
initial population	400	400	400	100	100	100
maximum generation	100	100	100	200	200	200

5.8 - Initial Development and General Principles in Bluenome Runs

It has been noted earlier that a significant proportion of agents developed through the Bluenome Model of Development can be considered trivial; That is, they consist of a single cell, or they instantiate exceedingly simple configurations of cells. The proportions of agents which are trivial is an important question, as it is a measure of the wasted computational time associated with the use of the Bluenome system. Figure 5.8.1 shows the proportions of such trivial agents in the initial populations (and hence, as expected in a set of randomly generated agents)

Table 5.8.1: Proportion of "trivial" Tarasanoids in the initial generations of the various Bluenome runs.

Run	bn.6	bn.8	bn.12	bn20.1	bn.20.2	bn.20.3	mean
Proportion with less than 3 cells	0.85	0.85	0.84	0.85	0.84	0.85	0.85
Proportion with *numClasses* < 2	0.98	0.99	0.98	0.98	0.97	0.98	0.98

These rather dismal figures are quickly weeded out in the initial generations, due largely to the *fitness*$_{bonus}$ measure (cf. section 5.6). Figure 5.8.2 and 5.8.3 shows the proportions in the first 20 generations for the same runs:

Table 5.8.2 – Proportion of agents with less than 3 cells after development

Run Generation	bn.6	bn.8	bn.12	bn.20.1	bn.20.2	bn.20.3	mean
0	0.85	0.85	0.84	0.85	0.84	0.85	0.85
5	0.51	0.31	0.55	0.66	0.54	0.76	0.56
10	0.42	0.54	0.53	0.57	0.41	0.33	0.47
15	0.47	0.45	0.52	0.51	0.46	0.38	0.47
20	0.30	0.40	0.45	0.63	0.32	0.29	0.40

Table 5.8.3 – Proportion of agents with numClasses =1 after development

Run Generation	bn.6	bn.8	bn.12	bn.20.1	bn.20.2	bn.20.3	mean
0	0.44	0.45	0.48	0.47	0.44	0.44	0.45
5	0.56	0.61	0.61	0.59	0.52	0.48	0.56
10	0.40	0.53	0.45	0.33	0.32	0.56	0.43
15	0.39	0.50	0.47	0.38	0.32	0.38	0.41
20	0.25	0.48	0.50	0.47	0.26	0.31	0.38

Above, one might note a change in proportional values relative to the size of the parameter *numTel*. Indeed, it appears that larger values of *numTel* will skew the population towards more diversity within an agent. This is not surprising, as a larger value of *numTel* will allow for more computation within the Bluenome Cellular Automata. Figure 5.8.4 and Figure 5.8.5 shows the proportions of agents partitioned by the value of *numClasses*.

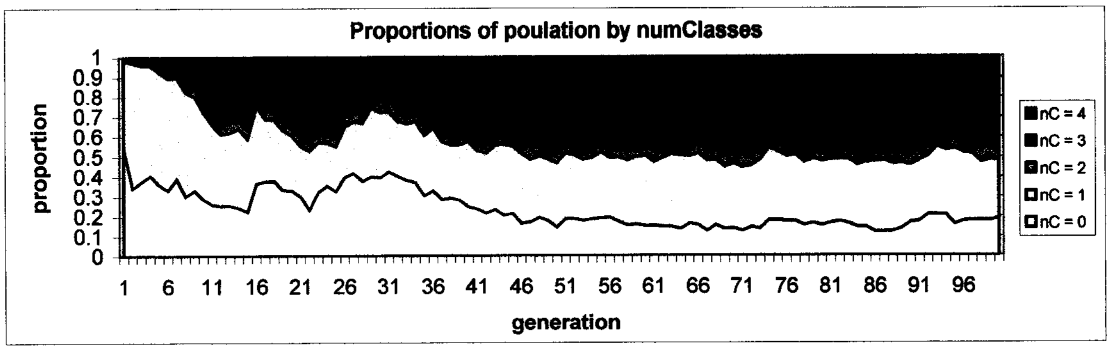


Figure 5.8.4 – Proportions of population by value of numClasses in run bn.6

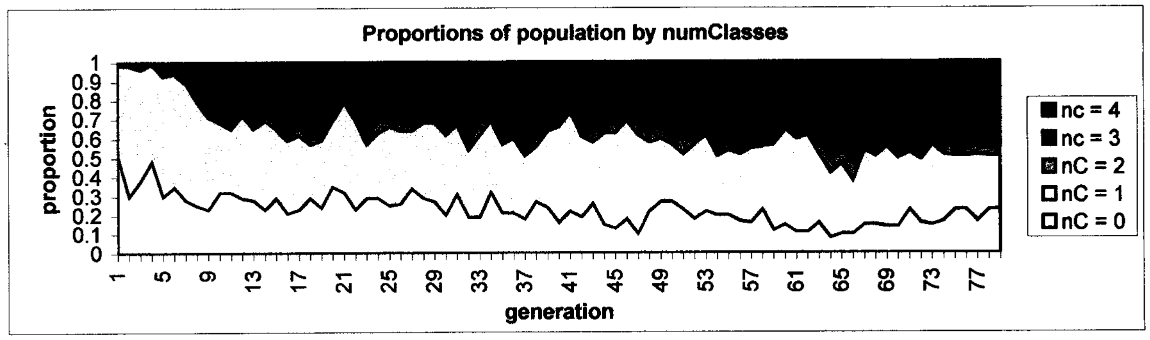


Figure 5.8.5 – Proportions of population by value of numClasses in run bn.20.1

Between runs, the proportion of agents with *numClasses* < 3 appears to settle at 0.4.

In the Bluenome model, it takes several generations for a population to evolve to the stage where any of the agents are capable of finding food – this point may be observed by contrasting with the maximum-bonus fitness, calculated in section 5.7 (Figure 5.8.6 illustrates the initial fitness values for run bn.6.1). The initial generations of the Bijective runs immediately surpass the maximum-bonus fitness level, as shown in Figure 5.8.7. The advantage seen in the initial generations of the Bijective run are not surprising, as the Worlds were specifically designed to show some meager results for random agents – but these Figures do demonstrate the existence of an initial start-up cost for the use of the Bluenome model.

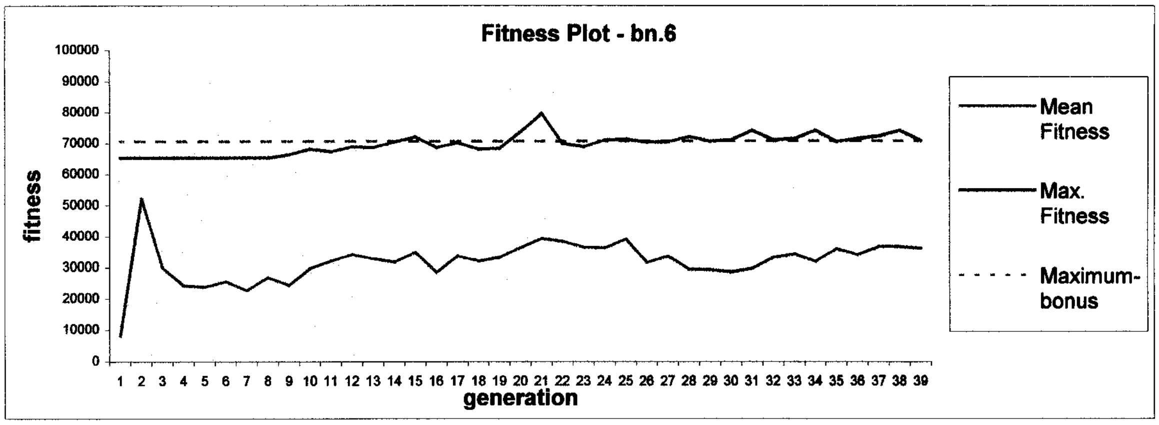


Figure 5.8.6 – Fitness plot of initial generations of run bn.6

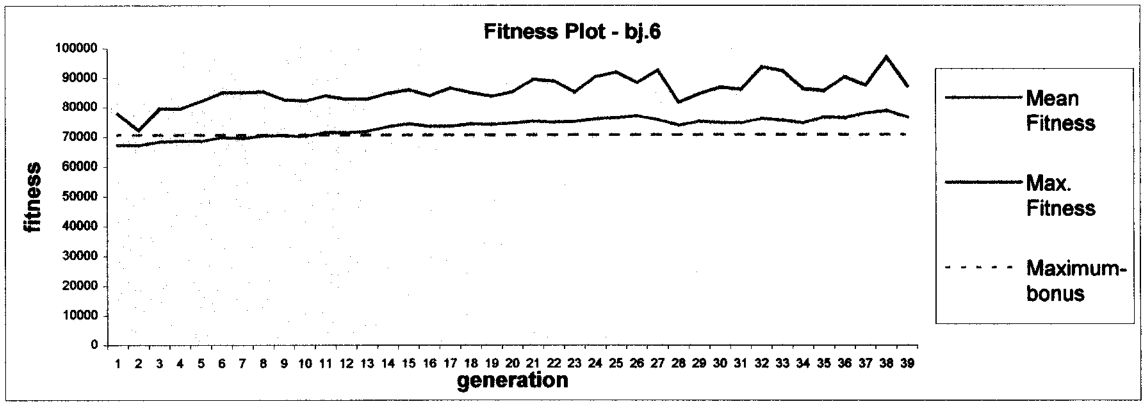


Figure 5.8.7 – Fitness plot of initial generations of run bj.6

5.9 – Data and Analysis

Comparison of the Bluenome and Bijective Runs: In the low phenotopic complexity runs, the bijective runs outperform substantially, as illustrated in Figures 5.9.1, 5.9.3 and 5.9.4. Also, Figure 5.9.2 shows a comparison between maximum time for the *numTel* = 6 runs – the Bijective version typically outperforms the Bluenome version. Contrary to initial expectations, this is not a boon, but instead a draw-back. The primary failing of the Bijective method is its inability to generate an adequate transport system for distributing food throughout its body. The successes of the Bijective model typically involve small groups of cells hoarding food, while no new food is found following time step 200.

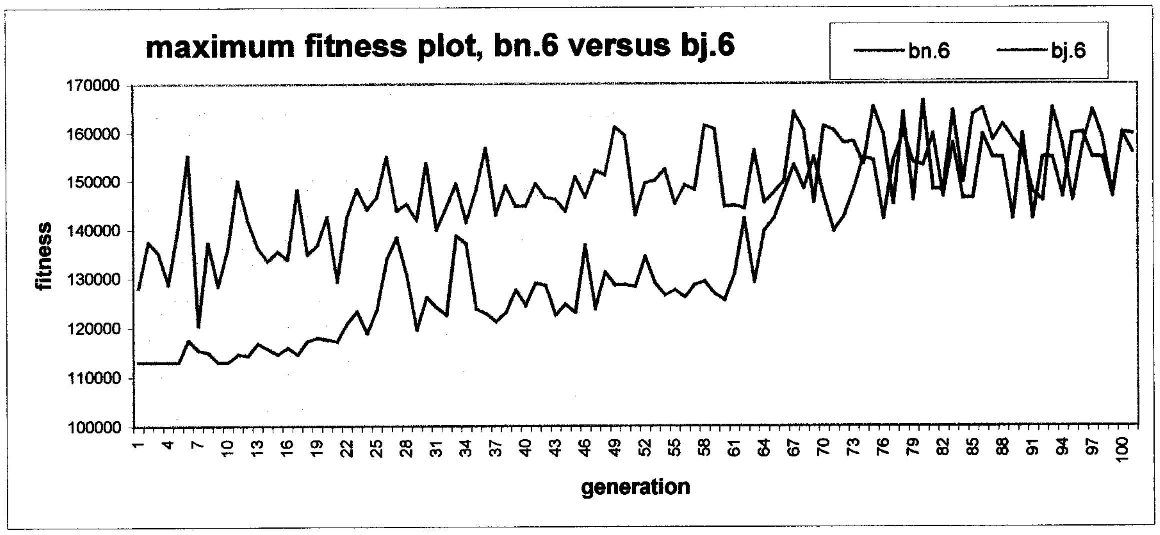


Figure 5.9.1: Comparison of the bn.6 and bj.6 runs, fitness plot

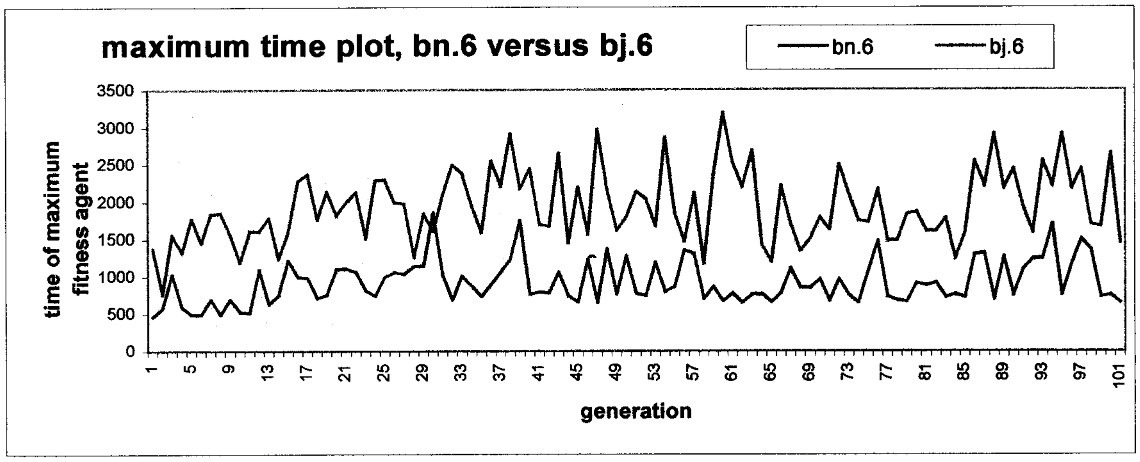


Figure 5.9.2: Comparison of the bn.6 and bj.6 runs, maximum time plot

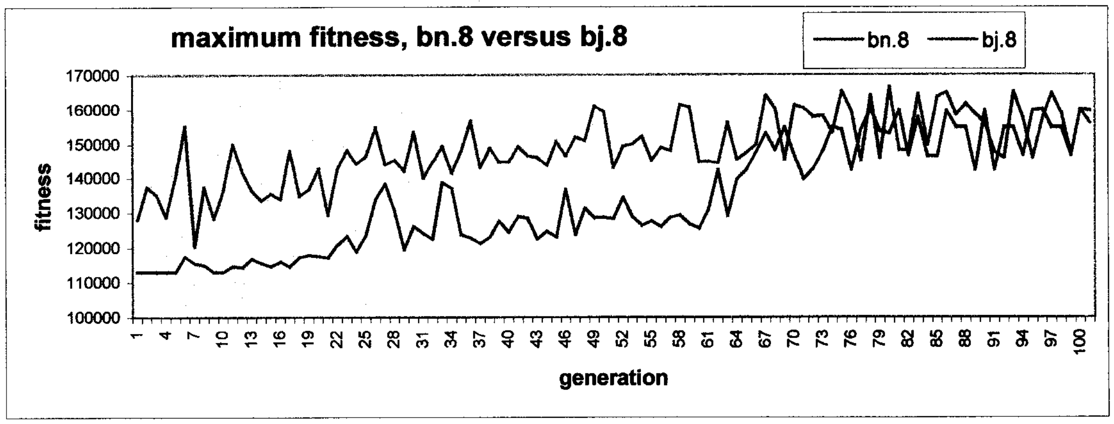


Figure 5.9.3: Comparison of the bn.8 and bj.8 runs, fitness plot

To reinforce the claims regarding the differences between the amount of food found by an agent and the time which the agent survived, a better measure than fitness might be that of fitness divided by time. Below (Figure 5.9.4) is plotted the fitness divided by time of the maximum-fitness agent between generations of the bn8 and bj8 runs. The Bluenome version clearly shows an advantage.

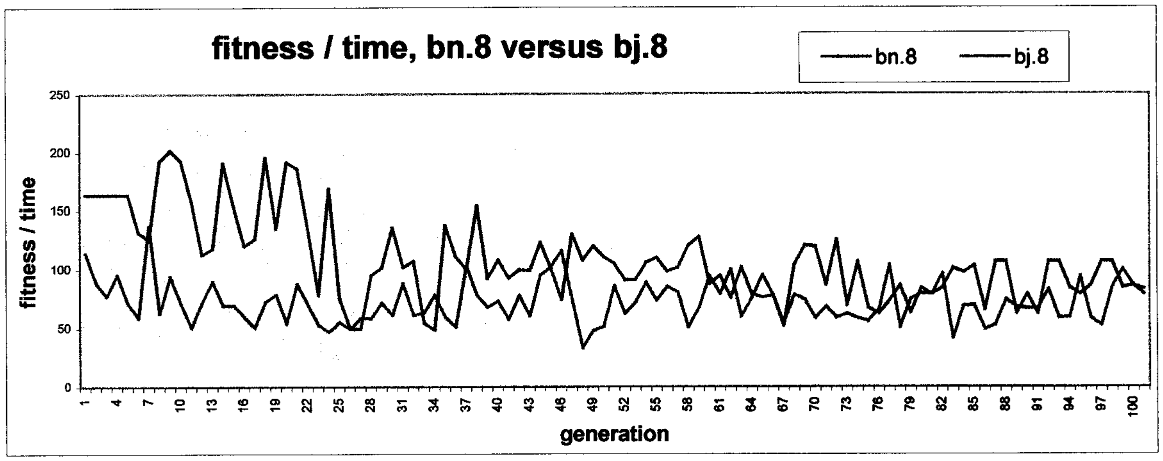


Figure 5.9.3: Comparison of the bn.8 and bj.8 runs, fitness / time plot

Data from the runs of the *numTel = 12* trials followed a similar course; That is, the bijective (bj.12) version began with a high fitness value, showing evolution and performance which exceeded that of the Bluenome version (bn.12). Informally, additional experiments were conducted including values of *numTel* between 12 and 19, with similar results.

Figure 5.9.5 show the fitness plots of the *numTel = 20* runs. In these runs, a different trend is seen; Here, the bijective runs all follow a similar course. They begin with some visible evolution, until they reach maximum fitness values in a range of about 720 000 to 850 000 (in all cases prior to generation 100), where they appear to oscillate between values randomly; It appears that the complexity of the space involved exceeds the GAs ability to improve. The bn.20.2 shows a similar course to the Bijective runs, with some initial evolution and a seemingly random cycling of values following. However, the bn.20.1 and bn.20.3 plots show a continuous evolution proceeding up to generation 200, potentially continuing beyond this point. Additionally, the bn.20.3 run quickly shows consistent maximum fitness values which exceed the maximum found in any of the bj.20 runs. Figure 5.9.6 clearly shows the continuation of the fitness / time trend – the Bluenome models clearly distribute food between body components more evenly.

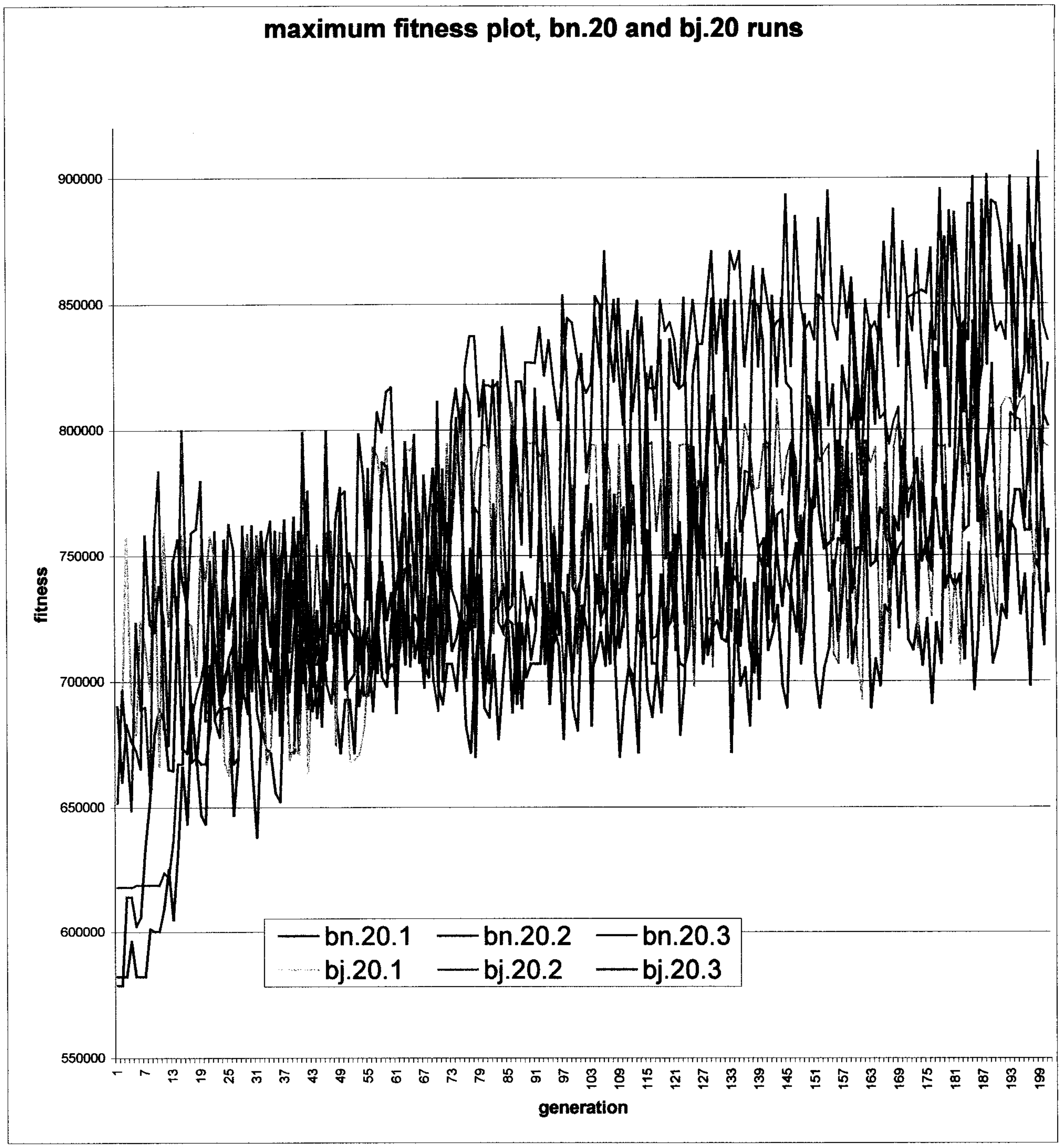


Figure 5.9.5: Comparison of the bn.20 and bj.20 runs, maximum fitness plot

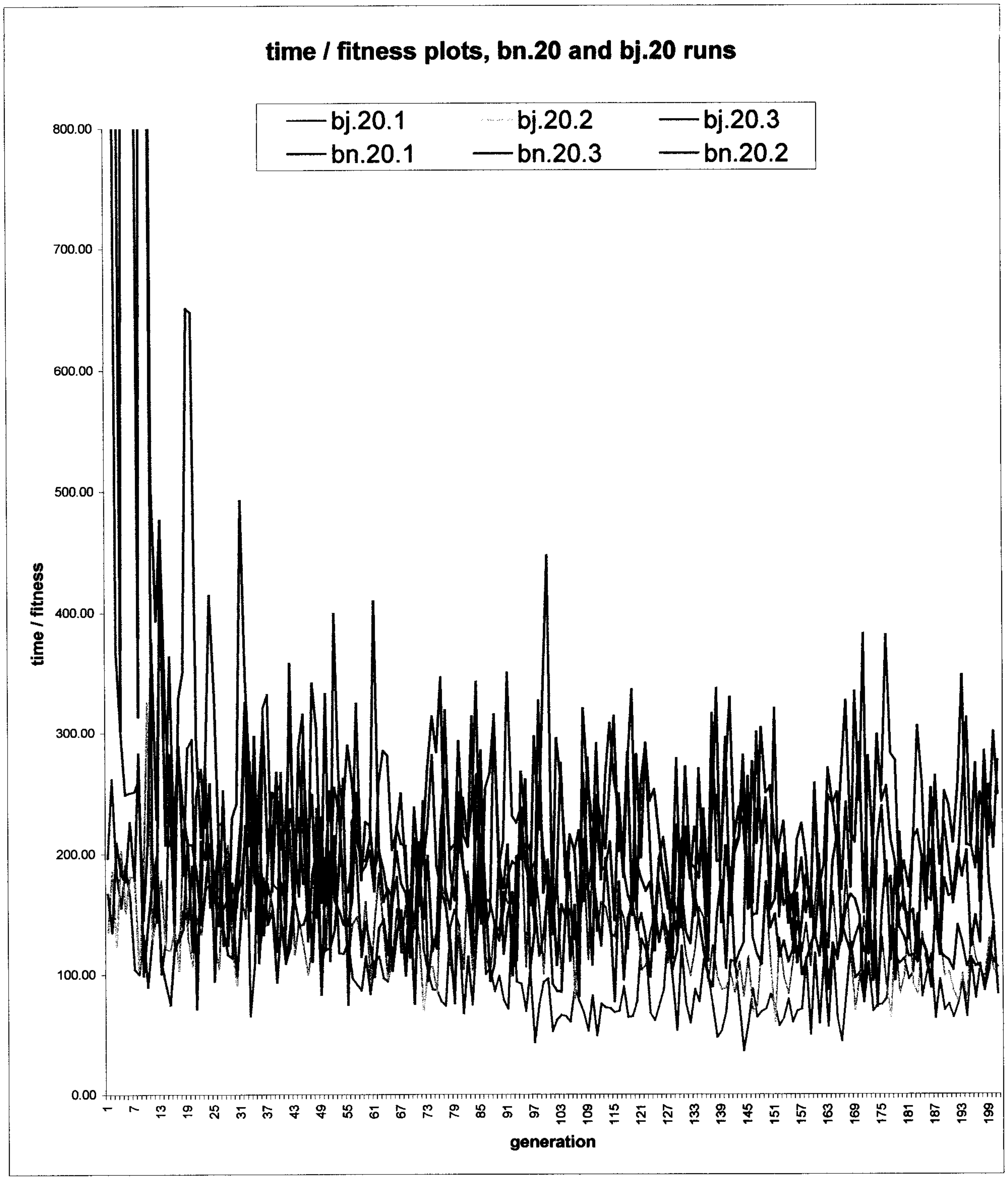


Figure 5.9.6: Plots of (fitness / time) of maximum-fitness agent, bn20 and bj20 runs

5.10 – Re-Growth of Agents under Different Values of ***numTel***

An interesting and important property determined from Phase One was that of the Bluenome Model's resistance to changes in environment with respect to agent growth (c.f. Section 4.10). In this section, an experiment was undertaken which tested a similar situation – that of the re-use of an agent's genome in a differing setting. A population of genomes were selected from a high-phenotopic complexity run (bn.20.2) at a period late in evolution (generation 180). These genomes were re-developed, this time using a value of *numTel* = *8*, rather than 20. The developed agents were evaluated as normal in the *numTel* = *8* context (that is, using the lower distances for food locations in the worlds (c.f. Section 5.2). The values obtained are comparable to the bn.8 run. In Figure 5.10.1, maximum and mean fitness values are compared between the re-grown agents and a population from bn.8. Generation 94 of bn.8 was chosen as it was this generation which showed the highest mean fitness through the entire run. While the bn8 population outperforms the re-grown agents slightly, the mean and maximum fitnesses of the re-grown agents are comparable to those found in the later stages of bn.8. Figure 5.10.2 shows the first agent of bn.20.2 grown with *numTel* = *8, 20*; Visual similarities between the two are immediately visible, and both agents are members of the "Position-then-Rotate Strategy" family of agents (c.f. Section 5.11).

Table 5.10.1 – Maximum and Mean fitness values from re-grown agents

	mean fitness	*max. fitness*
bn.8, generation 94	*75 252.67*	*126 364*
re-grown agents, bn.20.2, generation 180, numTel=8		
evaluation 1	*73 558.64*	*118 912*
evaluation 2	*73 111.52*	*120 856*
evaluation 3	*72 963.56*	*118 264*
mean	*73 211.06*	*119 344*

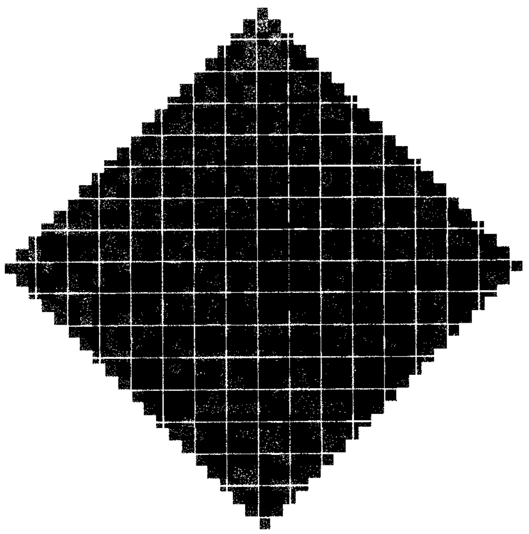
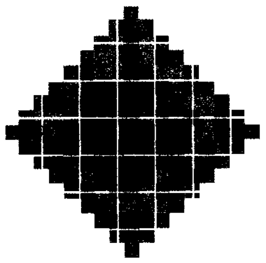



Figure 5.10.2 – An agent from run bn.20.2, generation 180 (left), re-grown having changed the value of numTel to 8 (below).

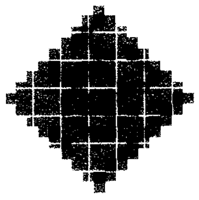
5.11 – Tarasanoid Strategies

In the evolution of Tarasanoids, many interesting strategies have been uncovered. This following section describes some which have been found, although, undoubtedly, many others exist which have escaped notice.

Blind Back-and-Forth Strategy: This strategy may be interpreted as a local optima which often dominates early generations – instances have been found in nearly every run, Bijective and Bluenome. Tarasanoids do not rely on input from Eye cells (often failing to include any in their make-up), instead relying solely on Forward Foot cells and Random Nerve Cells. The result is a Tarasanoid which slides back and forth randomly, never rotating. This strategy will find a measure of food in worlds of Type 0 and 3 with near certainty, occasionally finding some measure of food in worlds of Type 1 and 2, depending on the random distribution of food. The youngest[12] such agent was discovered in generation 30 of run bn.8, and is illustrated below, in Figure 5.11.1.

Figure 5.11.1 – Tarasanoid which implements the Blind Back-and-Forth Strategy.

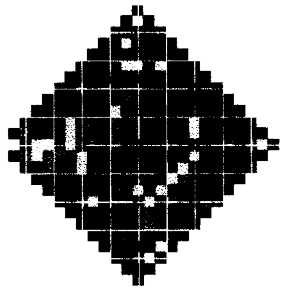
The dark blue cells are Forward Foot cells, the mid-orange cells Random Nerve cells. The agent contains no Eye of Rotating Foot cells whatsoever, save a single Eye cell near the centre (its location guarantees it will never fire). The single Eye cell is included probably for the sole reason of maximizing fitness$_{bonus}$



Larger-Circles Strategy: Another local optima occurring with high frequency in both Bluenome and Bijective runs, the Larger-Circle Strategy utilizes and agent which executes a spiral motion, moving in successively larger circles. Due to the construction of the worlds, this process will often find food, and will find it with near certainty in worlds 0 and 3, although it will typically fail in world 1. Figure 5.11.2 illustrates one such agent, discovered in generation 80 of bn.12.

Figure 5.11.2 – Tarasanoid which implements the Larger-Circles Strategy.

A wide array of Rotation Foot cells, connected both to Eye cells and Random Nerve Cells guarantees a contstant rotation. The presence of Forward foot cells but no Inverse Nerve cells guarantees a constant forward motion. The net result is a gradually increasing spiral motion.



[12] youngest :- the first discovered Tarasanoid in any run, with respect to the number of generations executed in the GA.

Rotate-then-Forward Strategy: This strategy is perhaps the most continuous and the most successful found. It consists of an agent with relatively few focused Eye cells, looking forward and / or backward. These Eyes are connected through a series of Identity or Inverse Nerve cells connected to Forward Foot cells, which drive the agent forward or backward when food is within visual range. It is augmented with a set of Random Nerve cells, connected to Rotation Foot cells, which rotate the Tarasanoid randomly. The base strategy is that the agent rotates until food is in view along the x-axis, then moves forward to retrieve it. Perhaps the youngest agent of this sort is pictured in Figure 5.11.3, discovered in generation 30 of run bn.8.

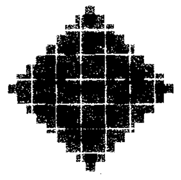


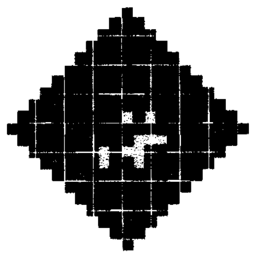
Figure 5.11.3 – Tarasanoid which implements the Rotate-then-Forward Strategy.

There are two Eye cells on the periphery of the Tarasanoid – centre left and upper right. These cells are somewhat buried, guaranteeing a narrow focus, and are connected through a large series of Identity Nerve cells to Forward Foot cells. In the centre of the agent are many Random Nerve cells, connected to Rotation Foot cells, providing the random rotation.

Position-then-Rotate Strategy: The Position-then-Rotate strategy is a local optima which occurs with far less regularity, but occurs all the same. The agent has eyes biased in a particular direction, initially moving in that direction on feedback from Eye cells. Once food is out of visual range (since the food is directly adjacent), the agent moves in a circular path of constant radius. This strategy is particularily useful in World 0, but nearly useless in all others. Figure 5.11.4 illustrates perhaps the youngest such agent, found in generation 40 of run bn.12. Of the exposited strategies, this one may be the most interesting, as it represents what is very likely an optimal strategy for world 0 – no other agents have been found which can consistently eat all food in this type of world.

Figure 5.11.4 – Agent which implements the Position-then-Rotate strategy.

The agent has Eye cells connected to Forward Foot cells on both the left and right hand side, with more Forward foot cells on the left. Towards the centre of the agent, a series of Random Nerves connect to Rotation Foot cells.



Development of a Transport System: As was evident from the fitness / time plots of section 5.9, one key advantage of the Bluenome-generated Tarasanoids was the quick development of agents with robust transport systems.

Figure 5.11.5 - The maximum fitness agent from the bj.6 run, generation 60, around time step 210. This agent successfully implements the Rotate-then-Forward strategy, having collected most of the food in the environment. Unfortunately, most of the food if hoarded in a small number of cells, hence the agent is incapable of finding more after time step 200. This is the common failing for an agent developed through the Bijective method.

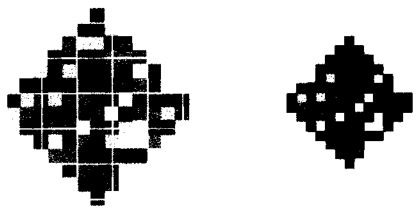
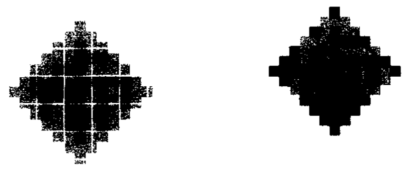


Figure 5.11.6
An agent from run bn6 (obviously a member of the Blind-Back-and-Forth family) with a well-developed transport system. Shown on the right is the agent after 200 time steps – the majority of cells are still active, and the agent is capable of finding additional food.



6. - Conclusions

The purpose of Phase One was the demonstration of the evolvability of the genomes in a Bluenome system; the shown experiments demonstrate this along several interesting axis. These experiments demonstrate that the Bluenome Model of Development is potentially a viable system for representation and maturation of an artificial agent within an evolutionary experiment. While Section 4 details the evolution along the axis of the fitness functions, the reader might draw more inspiration from images included in Appendix A; These images hint at many interesting and complex biological forms which are not easily quantified.

The purpose of Phase Two was the demonstration of a practical application of the Bluenome Model of development to a non-trivial application. The development of Tarasanoids was a highly difficult task, involving the simultaneous evolution of several complex systems connected in a non-linear fashion. The difficulty of this task can perhaps be seen best in viewing the relatively poor performance of the Bijective models on high phenotopic complexities (the *numTel = 20* runs).

There is good reason to believe that the extent of a GAs abilities with a bijective representation has been reached – This is because the three bijective runs of the system show relatively little variance. All three oscillate between the same range of values, all three show little signs of improvement beyond generation 100. It appears that in these high phenotopic complexity cases, the GA has achieved it's limits in this high-dimensional space. Although further improvements would no doubt occur in time, there is little reason to believe that these further improvements would occur with any more efficiency than random search.

However, one of the three Bluenome runs (bn.20.3) outperforms all three of the bijective runs, with a second (bn.20.2) showing promise with more computation. These runs demonstrate a performance that continues to improve between generations after the bijective runs have bogged down in a space too large. The limited size of the genotype appears to be a significant boon in the evolution of the larger types of Tarasanoids.

A second and perhaps more interesting point is the ability of the Bluenome-developed Tarasanoids to develop better systems for distributing food between body components. This characteristic is most likely due to the hierarchical organisation promoted by developmental models; These results may be viewed as providing supporting evidence to claims involving the importance of inherited-specialization development (as exposited in Section 2.3) outside of the field of simulated biology. It is also a parallel claim to the phenotopic-complexity-free explanation of why the Bluenome models continue to evolve after the bijective model has stagnated.

For both these reasons, it is expected that utilizing larger population sizes and more running time, consistent results could be obtained in which the Bluenome Model outperforms the Bijective model.

This, of course, is a meagre first statement – this does not imply that Bluenome is superior to other models of Artificial Morphogenesis, and although the Tarasanoids task appears quite general, does not necessarily imply that the Bluenome model will work for any task. These questions require a great deal more research to answer.

However, the implication made by the above experiments is that the Bluenome model is a powerful and robust model, which may operate in spaces whose size causes other methods to be infeasible. Also, for the reasons outlined in the review of Artificial Morphogenesis (Section 2.3), existing models do not readily lend themselves to application in as general a setting, or prove computationally intractable as well. Hence, Bluenome serves as an important and successful first step into finding an appropriate high-level approximation to biological development, which might utilize the strengths of the developmental process in human design.

The additional benefit to the Bluenome Model discovered in the above experiments was the resistance to environmental and global changes, as evident in sections 4.9 and 5.10. These characteristics are no doubt properties of the hierarchical computational process introduced in the review of Artificial Morphogenesis (Section 2.3). It is difficult to quantify how successful these resistances are outside of any particular domain of application, but informally, the existence of visible similarities in both Phase One and Two conveys that many distinct properties of growth are preserved.

A continuing problem, however, is the dismal proportion of agents which fail to develop. Contrasting the mean fitness between the Bluenome and the Bijective runs might demonstrate a preference for the Bijective, but for the maximum fitness measure. Indeed, the prevalence for agents in all generations which do not develop (a proportion of approximately 0.4 in smaller runs, 0.3 in larger ones) seems to indicate a constant waste of computation in the Bluenome method. It is easy to envision methods by which this might be limited; For example, one might ensure that each genome in the algorithm contains a rule which would cause the initial structural cell to divide[13]; Or, one might declare the initial integer in a genome to be a protected site (i.e. less likely to be affected by crossover or mutation), as these are most likely the points of highest discontinuity in the space. However successful these techniques might be, however, the potential for a complete solution to this problem is most likely intractable, given Bluenome's similarity to the CA, and the uncomputability of the forecasting problem in that context [Olivera, Olivera, Omar; 2001]. It is an interesting but unanswered question: Is the Bluenome CA also a model for computation? Images from Phase One suggest that the answer is yes, but the failure of the evolution of the Entropy run (Section 4.5) might suggest otherwise.

[13] This particular strategy is interesting as it resembles biological development, in which initially a zygote develops into a blastion, a collection of unspecialized cells prior to differentiation [Karp; 2001].

7. - Future Directions

There are many means by which the Bluenome method might be further evaluated or extended. Limitations of the Bluenome method at present are largely expressions of simplifications which have been adopted in an effort to decrease computational complexity, especially with regards to the evaluation of fitness. Additionally, the method of evaluation is somewhat limited, in that from the current context of the experiments, it is difficult to make predictions on how the system will scale up.

The choice of the Tarasanoid-World model for evaluation was an attempt to elicit the advantages of the Bluenome model's intrinsic hierarchy. A more interesting question might be to ask; In which situations will a Bluenome model of development provide an advantage to alternative representations? To approach this question, a far more general environment in which to evaluate agents is required. It seems a logical step to construct an environment in which agents are interpreted as programs or mathematical routines, where a particular problem may be posed to the system.

A second necessary condition of the environment is that of more efficient evaluation of an agent, which was the computational hurdle primarily holding back experimentation. In a field in which one attempts to find an appropriate high-level functional behavior, the need for experimentation is paramount. In creating the Bluenome system, many simplifying assumptions were made regarding the nature of an agent. These include:

1. That connectivity between cells and neighbours is simple and grid-based: In biological situations cells may deform, and the movement of a cell does not necessarily disrupt its connectivity. This allows for the easy creation of membranes, which may serve to easily divide an agent into regions.[14]
2. That the universe in two-dimensional; Three-dimensional structures contain many possibilities which simply cannot be explored in two-dimensional space, which undoubtedly limit the functionality of the Tarasanoid experiment.
3. The functionality of the cell types involve many simplifying assumptions relative to the actual cells they were modeled from. Issues such as: what to do with the material which comprised an inactive cell, or, the near-magic by which cells in the developmental phase divide and specialize might be re-visited, attempting several alternative behaviours to determine the difference to overall performance.

These are questions unlikely to be answered through any means save experimentation – a mechanism for agent evaluation must be fast and general enough to support them.

A final possibility presents itself at this point. Wolfram's earlier point is summarized in this passage below:

> I have come to believe that many of the most obvious examples of complexity in biological systems actually have very little to do with adaptation or natural selection... my strong suspicion is that in fact the main effect of natural selection is almost exactly the opposite: it tends to

[14] An example of paramount importance in the development of vertebrates is the determination of the blastocyst into the three germ layers, with analogous developments for plants and invertebrates. The universality of this occurrence suggests its value.

make biological systems avoid complexity, and be more like systems in engineering.

Indeed, this is an intriguing hypothesis, but begs an obvious question: If so, how? If the function of natural selection is to limit the wildness of the developmental process, how does it do so? Answering this question might normally be a task for biologists; However, this might not necessarily be the case, if the Bluenome system models the correct functional aspect of development. It is my hope that systems of the sort of processing which Bluenome undertakes might some day yield an answer to this question.

References

Bentley, P., Kumar, S., *The Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem*, in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-1999

Dellaert, F., Beer, R., *A 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 Behavior, pp. 393-401, (1996)

Eggenberger, P., *Evolving Morphologies of Simulated 3D Organisms Based on Differential Gene Expression.*, in Husbands, P., Harvey, I. (editors) Proceedings of the Fourth European Conference on Artificial Life (1997)

Fagotto, F., Gumbiner, B., *Cell contact-dependent signaling*, in Developmental Biology 180, pp. 445 – 454, (1996)

Furusawa, C., Kaneko, K., *Emergence of Rules in Cell Society: Differentiation, Hierarchy, and Stability,* in the Bulletin of Mathematical Biology, pp. 79 – 93, (1998)

Hamahashi, S., Kitano, H. *Simulation of Drosophilae Embryogenesis*, in the Proceedings of the Sixth International Conference on Artificial Life (1998)

Hart, W., Kammeyer, T., Belew, R., *The Role of Development in Genetic Algorithms*, in D. Whitley, M. Vose, editors, Foundations of Genetic Algorithms III. (Morgan Kauffman; 1994)

Haupt, R., Haupt, S., *Practical Genetic Algorithms*, (John Wiley and Sons; 1997)

Holland, H., *Adaptation in Natural and Artificial Systems*, (University of Michigan Press; 1975)

Hornby, G., Pollack, J., *Creating High-Level Components with Generative Representation for Body-Brain Evolution*, in Artificial Life v. 8, no. 3, pp. 223 - 246 (2002)

Hotz, P., Gomez, G., Pfiefer, R., *Evolving the morphology of a neural network for controlling a foveating retina - and its test on a real robot*, in Artificial Life VIII: The 8th International Conference on the Simulation and Synthesis of Living Systems , p. 243, (2003)

Karp, G., *Cell and Molecular Biology, 3rd Ed.*, (John Wiley & Sons, Inc.; 2001)

Kauffman, S., *The Origins of Order*, (Oxford University Press; 1993)

Kvasnicka, V., Pospicjal, J., *Emergence of Modularity in Genotype-Phenotype Mappings*, in Artificial Life, v. 8, no. 4, pp. 295 - 310 (2002)

Mitchell, M., *An Introduction to Genetic Algorithms*, (MIT Press; 1998)

Mitchell, M., Crutchfield, J., Das, R., *Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work*, in Proceedings of the First International Conference on Evolutionary Computation and Its Applications (1996)

Olivera, G., Olivera, P. Omar, N. *Definition and Application of a Five Parameter Characterization of One-Dimensional Cellular Automata Rule Space*, in Artificial Life, Volume 7 Number 3, pp. 277 – 302, 2001

Patten, B., *Foundations of Embryology, 2nd Ed.*, (McGraw-Hill; 1964)

Rechenberg, I., *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, (Frommann-Holzboog; 1973)

Schwefel, H., *Numerical Optimization of Computer Models*, (John Wiley and Sons; 1981)

Spears, W., De Jong, K., Baeck, T., Fogel, D., de Garis, H., *An Overview of Evolutionary Computation*, in the Proceedings of the European Conference on Machine Learning, pp. 442 – 459, (1993) Stanley, K., Miikkulainen, R., *A Taxonomy for Artificial Embryogeny*, in Artificial Life, v. 9, no. 2, pp. 93 – 130, (2003)

von Neumann, J., *Theory of Self-Reproducing Automata*, (University of Illoinoise Press; 1966)

Wolfram, S., *Universality and Complexity in Cellular Automata*, reprinted in Cellular Automata and Complexity: Collected Papers, pp. 115 – 158, (Westview Press; 1994)

Wolfram, S., *A New Kind of Science*, (Wolfram Media Inc.; 2002)

Appendix A: Additional Images from Phase One

The following are a set of images found during Phase One. They are included here as they demonstrate principles which I have not been able to quantify, principles demonstrating the robustness of the Bluenome Model of Development in relation to two-dimensional CAs, or simply because they are aesthetically pleasing. Figures A.6 is relevant to demonstrating the breadth of the Bluenome system with respect to that of two-dimensional CAs – communication of information [Mitchell; 1998] is an important component is demonstrating the universality of CA computation. Figure A.5 is interesting as it demonstrates Bluenome's ability to produce fractals (given infinite resources, naturally).

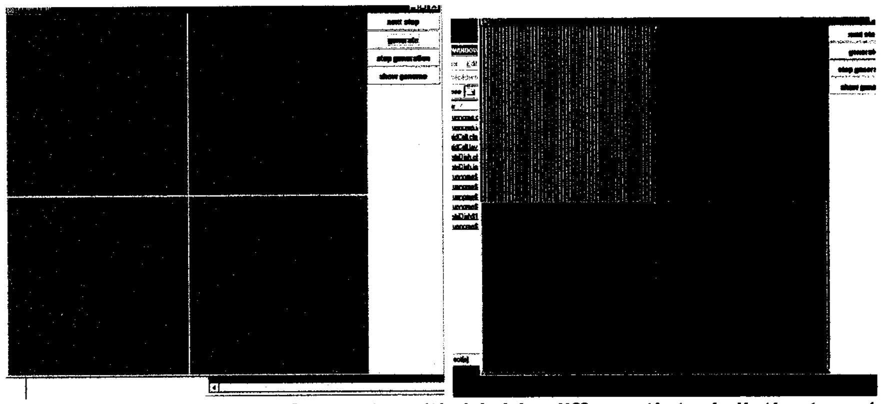


Figure A.1 – Examples of agents with highly differentiated distinct regions

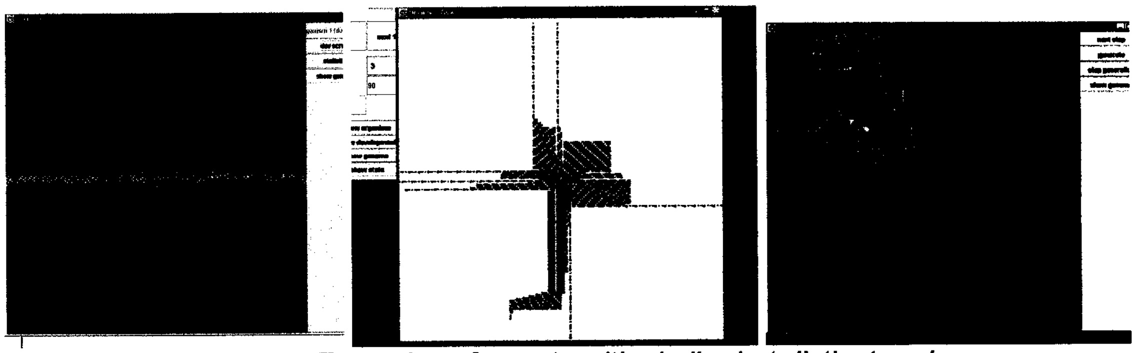


Figure A.2 – Examples of agents with similar but distinct regions

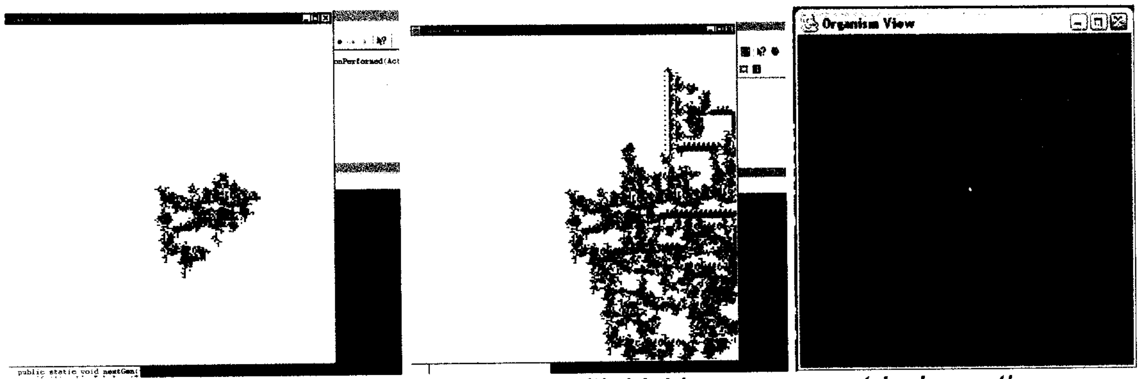


Figure A.3 – Examples of agents with highly non-symmetrical growth
The left and centre images are from the same genotype,
in different stages of growth.

Figure A.4 – Agent with growth resembling a biological transport system, half-way through the developmental process.

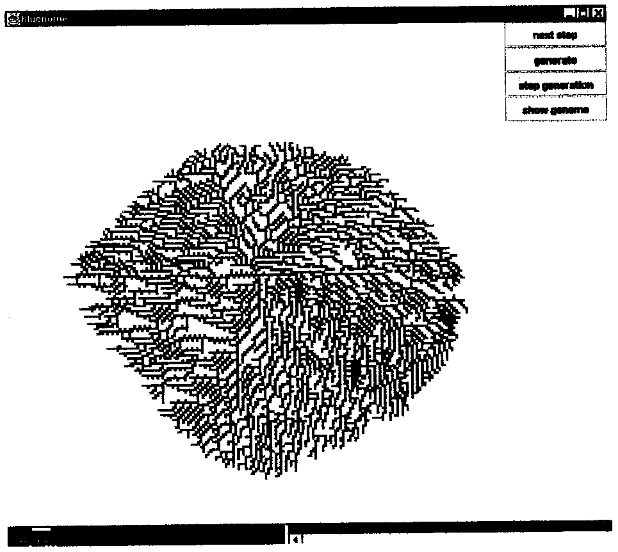
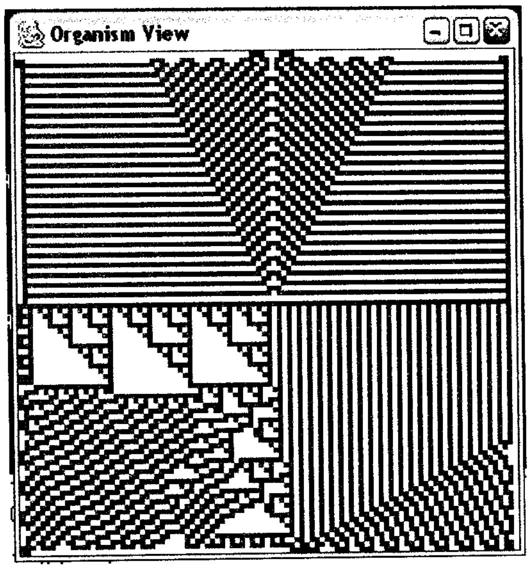


Figure A.5 – Agent which partially recovers the Sierpinski Pyramid



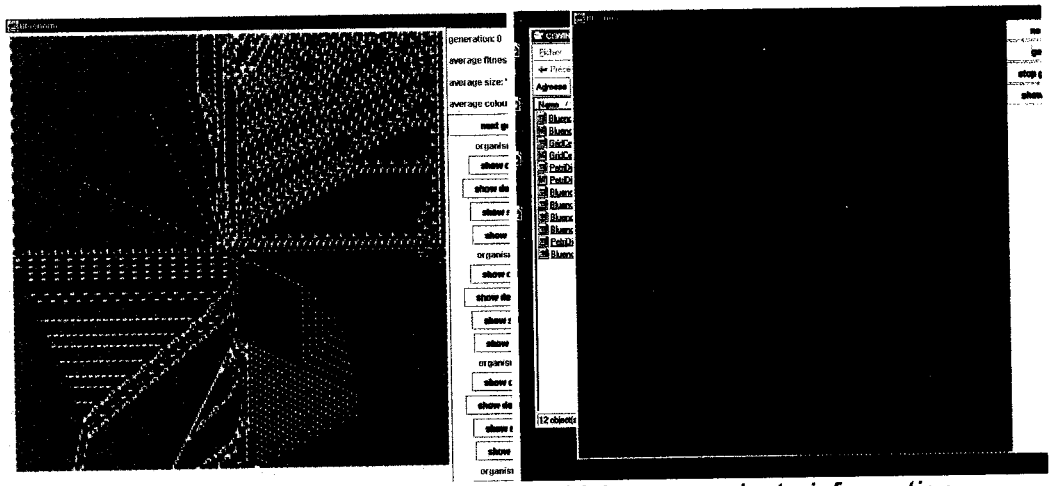


Figure A.6 – Examples of agents which communicate information over distance; The interactions between the "branches" in these agents recover the particle-interactions discussed in [Mitchell; 1996].

Appendix B: Precise Algorithms for Development

Bijective Spiral Development: The bijective model of agent development accepts an array of cell types, and constructs an agent by laying out those cells in a spiral, beginning with the centre, biased down and clockwise.

The spiral layout is generated through the following algorithm:

> Initially, the first element of the genome is mapped to a cell and placed in the centre of the development grid. That cell is labeled currCell.
>
> For any cell, currCell, enumerate its neighbours from 0 to 7, where 0 is the cell directly to the left,1 the next cell clockwise. A neighbourhood can then be characterized by an 8-tuple of values from {0, 1, *}, where 0 means that the neighbouring cell is empty, 1 means that it is full, * matching either case. Hence, the pattern <0,*,*,*,*,*,*,1> is a template for any neighbourhood in which the cell directly left is empty, and the cell in the bottom left is full.

```
for k = 1 ... genome_size
        consider the neighbourhood N surrounding currCell
        if N is characterized by <0,*,*,*,0,0,0,0>
                place cell k in location 0, label it currCell
        else if N is characterized by <*,0,1,*,*,*,0,*>
                place cell k in location 1, label it currCell
```

```
else if N is characterized by <0,*,*,0,1,*,*,*>
        place cell k in location 3, label it currCell
else if N is characterized by <*,*,0,*,*,*,1,0>
        place cell k in location 5, label it currCell
else if N is characterized by <1,*,*,*,0,*,*,0>
        place cell k in location 7, label it currCell
```

Best Free Location: – During the developmental process, a cell will select the "best free location" from its four-neighbourhood in deciding the location of a cell division or movement. Loosely, the "best free location" is the empty cell farthest away from the centre of mass of cells surrounding the current.

```
Let the the neighbours of the current cell be labeled 0 – 7, where 0 is the
cell directly to the left, 1 the next cell clockwise.

Let d(x) = 1 if there exists a cell in location x, 0 otherwise
Let the mass of a side,
mass(x) =     infinity; if d(i) = 1
              d(x – 1) + d(x) + d(x+1 mod 7); otherwise
Let minSide = min { mass(0), mass(2), mass(4), mass(6) }
If minSide is finite, return it. Otherwise, return null.
```

Appendix C: Glossary of Terms Used

Bluenome: A novel model of translating genotype to phenotype, inspired by biological Cell Differentiation. *Bluenome Model of Development* refers to the use of Bluenome for development, and *Bluenome system* refers to a Genetic Algorithm which utilizes the Bluenome model of development to generate agents.

Bluenome Cellular Automata: An alternative view of Bluenome Development, where growth is seen as computation in the instantiated two-dimensional CA.

Bijective Development: A bijective translation between genotype and phenotype. In Phase Two, this is implemented as mapping a list of cell types to a diamond of cells, via a spiral layout (See Appendix B)

Cell: A single component which comprises part of an agent. In Phase One, a cell is simply a block of colour in an image. In Phase Two, it is one of 9 types of cells, each of which plays an autonomous role in an agent.

Crossover: A Genetic Operator, as used in Evolutionary Computation. Crossover accepts two genomes (as used here, two lists of integers). A single cross-point is chosen, and the latter parts of the strings are swapped, creating two new strings of integers. Note that the selection of the cross-point 0 returns the two original strings. Crossing two agents over means that their respective genomes where crossed, producing two new agents from the resulting genomes.

Elitism: A Genetic Operator which selects the most fit agents in a population, and places them directly into the next population. Therefore, specifying a rate of elitism of 0.1 implies that the top 10% of any given population will appear in the following generation without modifications. Typically, elitism is used to ensure that fitness never decreases between generations. The probabilistic nature of fitness evaluations in the Bluenome experiments, however, make this impossible. In Bluenome Phase Two, elitism is used simply to limit the destructive qualities of the Crossover and Mutation operators.

Environment: A rectangular matrix of Grid Cells, or atomic units of space. An agent develops and lives in this matrix.

Eye: A cell type, utilized in Phase Two. An eye cell detects food within its field of vision, returning 1 or 0. Note that an eye's field of vision will be disrupted by another cell. Eyes may be short-range or long-range.

Fitness-Proportional Selection (a.k.a. Roulette-Wheel Selection): The following algorithm selects an agent from the population. Let our population be $p \varepsilon P$, and let f be our fitness function. Let each agent in P be assigned a proportion of the total sum of fitness:

$$fitnessSum = \Sigma_{p \varepsilon P} f(p)$$

$$proportion(p) = f(p) / fitnessSum$$

Partition the space $[0, fitnessSum]$ into $|P|$ parts:

$$wheel_0 = [0, f(p_0)],\ wheel_1 = [f(p_0), f(p_1) + f(p_0)], \text{ etc.}$$

Let r be a random value, $r \varepsilon [0, fitnessSum]$; then r falls in interval $wheel_j$ for some j. Return agent p_j.

Foot: A cell type, utilized in Phase Two. A Foot cell accepts input from neighbouring Nerve cells, causing the overall agent (Tarasanoid) to either move forward, or rotate. Foot cells may be of type forward or rotate.

Grid Cell: A unit of space in an agent's environment. A Grid Cell in Phase Two may contain a single cell and / or a single patch of food. An agent's environment is a rectangular matrix of Grid Cells.

Mutation: A Genetic Operator used in Evolutionary Computation in the preparation of a new generation. The mutation operator, given a genome (g) and a probability of mutation (p), changes each bit in g (from 1 to 0, or from 0 to 1) with probability p. In the case of a genome as used in the experiments described in this paper (either Bijective or Bluenome), an integer in the genome g is replaced with a new random integer, with probability p.

Neighbourhood: Given a Grid Cell in a matrix, a neighbourhood is the collection of cells of specified size which surround it. Figure C.1 shows neighbourhoods of size four, eight and twelve, along with numbering as is used in the pseudo-code in this paper.

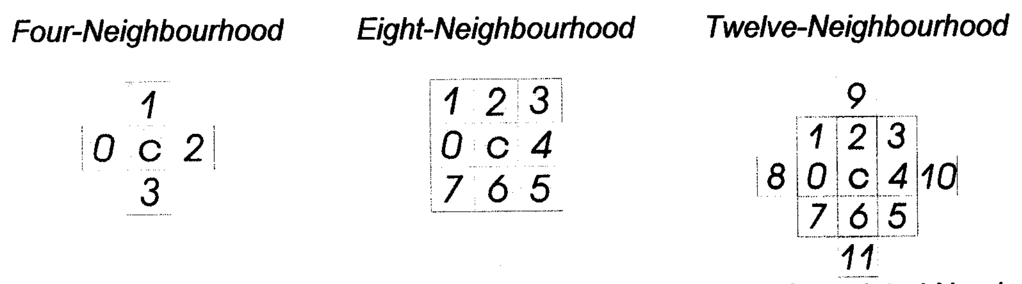


Figure C.1: Neighbourhoods about a Grid Cell c, and Their Associated Numbering.

Nerve: A cell type, utilized in Phase Two. A Nerve cell accepts input from an Eye cell or another Nerve, outputting a transformed value to another Nerve cell, or a Foot cell. Nerves may be of type Identity, Inversion or Random.

Pattern Flattening: A technique by which additional colours are added to an image, by finding the most common pattern within an image and redeclaring them a different colour. A pattern in an image is a matrix of 2x2 or 3x3 cells.

Telomere: A telomere (in Biology) is a specialized section of a cell's DNA – between replications, this section is cloned, leading to an increasing number of appearances in cells spawned later in development. It is suspected that telomeres play a role in limiting initial development, as well as in mediating later mature-organism growth. In the Bluenome system, each cell in the developmental phase of an agent contains a telomere variable – each successive action causes the telomere to decrement. Once a telomere reaches zero, a cell may make no more actions. Hence, an agent developed via the Bluenome model may reach a size of at most $2*(numTel + 1)^2$, where *numTel* is the number of telomeres.

Transport: A cell type, utilized in Phase Two. Transport cells play two roles: The first is to absorb food from the environment, if food occupies the same Grid Cell, or any in its four-neighbourhood; The second role is the distribution of food between itself and its neighbours. There is only one kind of Transport cell.

Structural: A cell type, utilized in Phase Two. Structurals are "neutral" cell types, in that they undertake no action in the behavior of a Tarasanoid, save to absorb food. Every Tarasanoid begins its development as a single Structural cell.