

# Bluename: A Novel Developmental Model of Artificial Morphogenesis

T. Kowaliw, P. Grogono, and N. Kharmia

Departments of Computer Science and  
Computer and Electrical Engineering, Concordia University,  
1455 de Maisonneuve Blvd. Ouest, Montréal, QC, Canada, H3G 1M8  
taras.kowaliw@utoronto.ca, grogono@cs.concordia.ca,  
kharma@ece.concordia.ca

**Abstract.** The Bluename Model of Development is introduced. The Bluename model is a developmental model of Artificial Morphogenesis, inspired by biological development, instantiating a subset of two-dimensional Cellular Automata. The Bluename model is cast as a general model, one which generates organizational topologies for finite sets of component types, assuming only local interactions between components. Its key feature is that there exists no relation between genotypic complexity and phenotypic complexity, implying its potential application in high-dimensional evolutionary problems. Additionally, genomes from the Bluename Model are shown to be capable of re-development in differing environments, retaining many relevant phenotypic properties.

## 1 Introduction

Typical applications in Evolutionary Computation often involve a direct and simple relation between genotype and phenotype; Commonly, values from the genome are simply slotted into a fitness function in a bijective mapping. While this approach is sufficient for most practitioners, the dimensionality of the solution space (phenotypes) is directly translated into the dimensionality of the space of genotypes, potentially exceeding the size of space capable of being searched by a Genetic Algorithm. For larger and more complex problems direct relations between genotype and phenotype may be insufficient.

In a field inspired by Biology, it is often useful to re-examine the source: the human genome may be viewed as a tremendous compression of phenotypic complexity; The approximately 3 billion chemical base pairs of the genome map to approximately 100 trillion cells [8]. It is clear that the process of development plays a significant role in the addition of information to the phenotype; Indeed, models of biology often attempt to re-create the hierarchical structure inherently formed by the differentiation process, as in [5].

An emerging trend in Evolutionary Computation is to create a (relatively) small and simple genotype, and to increase the complexity of the phenotype through a developmental process. The field of Artificial Morphogenesis spans a wide array of

approaches, varying on a gradient between total information being contained in the genotype, to a bare minimum, where the genotype is a program designed to spawn the phenotype. Some approaches use simple techniques, such as the use of repeated structure when interpreting the genome. Another more complex approach is the use of Grammars to develop agents, where the grammar and initial conditions form the genotype. Such systems have enjoyed much success, as in [10], where they were used in a theoretical experiment to study the development of modular forms, or in [7], who used the model to develop a neural network controlling a foveating retina. However, these systems are far separated from their original biological metaphors.

Other equally complex examples include developmental models, inspired by Embryogenesis. Examples which attempt to model actual biological embryogenesis have also enjoyed much success, as in the case of the particularly successful modeling of the embryogenesis of a drosophilae, as found in [6]. In between direct models of biology and models which are entirely separated, there exists a class of developmental models which seeks to abstract the developmental process to high-level systems capable of artificial evolution, hopefully retaining some of the high-level features of biological development. Examples include Bentley et al. [1] and Eggenberger [3], who both propose highly-simplified models which are used to demonstrate the development of geometric and aesthetic principles. Perhaps most closely related to the subject of this paper, however, is work undertaken by Dellaert and Beer [2]; 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 performed a series of divisions and differentiations, controlled by a series of Genetic Regulatory Networks. Drawbacks to this model 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, and again later by simplifying their model. As is noted in a review by Stanley and Miikulainen, simpler solutions may outperform biologically plausible ones, and a need exists for abstraction [12].

This viewpoint, that of increasing the complexity of the phenotype through the developmental process, forms an interesting starting point for another emerging line of thought: It has been postulated (most notably by Wolfram [13]) that natural selection serves not to increase the complexity of agents through time, but instead to limit the complexity inherent in a complex and unwieldy developmental process. If this suspicion is correct, then the current typical use of Evolutionary Computation may not utilize the metaphor of natural selection to its fullest potential.

In this paper, we present the *Bluename*<sup>1</sup> model of development. *Bluename* is a highly abstracted model for developing application-neutral agents composed of an arbitrarily large network of components chosen from a finite set of types. *Bluename* uses a subset of the space of two-dimensional CAs, evolved in a genetic algorithm, starting from a single neutral cell. Ideally, *Bluename* is designed to recover both the inherent structure associated with developed organisms, and also the unwieldy complexity found in Cellular Automata.

---

<sup>1</sup> **Genome** as a **blueprint** for development.

## 2 The Bluename Model<sup>2</sup>

The Bluename 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*, the number of types of components which might be included in an agent. 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.

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:

<i>colour</i>	<i>hormone<sub>i</sub></i>	...	<i>hormone<sub>numColours</sub></i>	<i>action</i>
---------------	----------------------------	-----	-------------------------------------	---------------

where  $colour \in [1, numColours]$ ,  $hormone_i \in [1, 12]$ , and  $action \in [1, numColours+3]$ .

Initially, an agent begins as a single neutral cell, centred in an environment (a square matrix of Grid Cells). When activated, a cell (*currCell*) in the environment will collect hormones from its twelve neighbourhood, storing the number of occurrences of each cell colour in a list. The exception is in the case of cells on the periphery; These cells are only included in the count if the cells in between are empty<sup>3</sup> – hence the cell on the far, far left will be included in the count only if the cell on the left is empty. What results is a list of numbers of length *numColours*, each of value between zero and twelve.

Once any particular cell has collected information regarding its neighbours, it searches the genome to find the closest matching rule: First, it collects all rules in the genome such that the current colour of the cell and the first argument in the rule match (If no such rule is found, the cell takes no further action this time step). Next, it searches that collection of rules for the one most closely matching its list of current hormone levels (Euclidean distance). Finally, the action is extracted from that rule. The action is executed by first decrementing the cell's internal telomere counter (hence a cell may execute only *numTel* actions), then executing the action corresponding to *theRule<sub>action</sub>*. Possible actions include: **Die**, where a cell is removed, leaving an empty Grid Cell; **Specialize(*colour*)**, where a cell changes its colour to *colour*; **Divide**, where a copy of the cell is placed in the best free location; and **Move**, where the cell is relocated to the best free location (if there are no free locations, no action is taken). It should be noted that the best free location is defined as the Grid Cell in *currCell*'s four-neighbourhood furthest away from the largest mass of cells in the eight-neighbourhood. In the case of equal distribution, the best free location includes a directional bias – left, then counter-clockwise.

<sup>2</sup> For brevity, technical details involving algorithms and parameters have been omitted. The interested reader is urged to consult [9].

<sup>3</sup> To model the difference between contact-dependent and diffusible communication [4].

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, beyond that inherent in the directional bias.

One view of this process is as a subset of all 2-dimensional 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<sup>4</sup>; (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 set; (3) Bluenome utilizes a measure to compute distance to a rule, unlike CAs, which are precise. This may be viewed as collapsing several similar rules into a single outcome; and (4) The lack of consideration of peripheral cells in the twelve-neighbourhood may be viewed as a further grouping of CA rules.

Fig. 2.1 shows the development of an interesting agent taken from Phase One, shown at various times in the development. An unfortunate point is that the majority of genotypes generate trivial agents – nearly 80% of random samples. However, it will be shown that selection quickly improves matters.

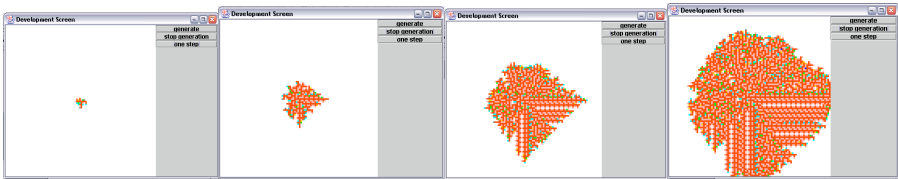


Fig. 2.1. Development of an interesting agent.

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, an agent with  $numTel = 6$  will have a maximum phenotypic size of 98 cells; 882 cells for an agent with  $numTel = 20$ . In contrast, an agent with  $numColours = 5$  and  $numRules = 25$  will have a genotypic size of 175, regardless of phenotypic size. The complexity of the developmental process is  $O(numTel^3 * numRules)$ .

### 3 Phase One: Application-Neutral Experiments

The evolution of cellular automata is a notoriously difficult problem: the highly non-linear nature of the space of CAs, as well as the unpredictability of the forecasting problem [15], [13] makes the prospect of the evolution of complex patterns using GAs seem grim. As we have recognized the Bluenome model as a subset of the space of

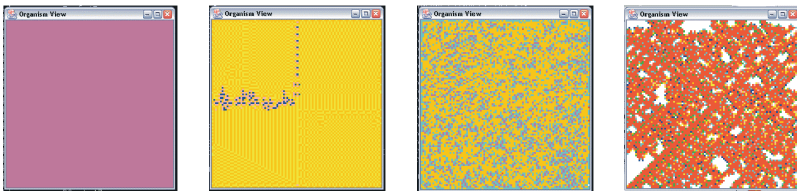
<sup>4</sup> Similar to a Totalistic CA [15].

two-dimensional CAs, it is not obvious that evolution is possible in any reasonable measure of time. Phase One was a set of experiments utilizing an array of fitness functions to determine which high-level axis could potentially be evolved.

Phase one utilized a neutral view of phenotypes – an organism was treated as an image, fitness function being based on techniques from Image Processing. In all cases, grids of size 100x100 were used, and the agents were allowed to grow to fit the grid. A genetic algorithm was used to evolve phenotypes, typically running for 100 generations with a population size of 30.

Successful experiments showed promising results; Typically, highly fit population members were discovered prior to generation 100, with steady increase in mean fitness. These fitness functions included: selection for complexity, selection for similar but disconnected regions, selection for highly distinctive regions, and selection for a transport system.

An example of one such successful run was the attempt to generate images of increasing complexity. Here, fitness was a function of the size of the grown agent and the number of different cell types included in the phenotype. By generation 100 a robust agent utilizing all colours can be seen. Members of the population can be seen in Fig. 3.1. From generation 100 of this same experiment.



**Fig. 3.1.** Images from a Phase One experiment using complexity as a fitness. Members are shown from generations 0, 10, 40 and 100.

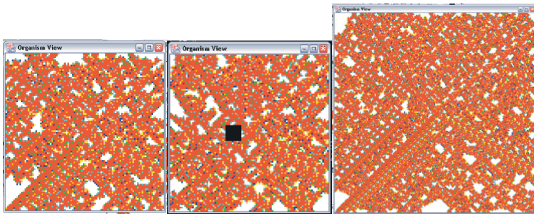
Fig. 3.2 shows a set of exemplar members chosen from the various experiments. In all cases, these members were found in less than 100 generations of evolution, utilizing population sizes of less than 30.



**Fig. 3.2.** Exemplar members from Phase One experiments, identified by generating fitness: (*far left*) maximal thin coverage; (*left*) disconnected regions of similarity; (*right*) disconnected regions of similarity; (*far right*) highly distinctive regions.

Another interesting result from Phase One was the attempt to re-grow a genotype in differing environments. This attempt is illustrated in Fig. 3.3, where again

visual similarities may be seen most clearly. Additionally, under the fitness used in the experiment (the complexity function), fitness was nearly identical for each re-growth.



**Fig. 3.3.** The same genotype re-grown in differing environments. (left) original phenotype; (centre) environment with non-interacting foreign cells (black); (right) environment of double the size.

## 4 Phase Two: Application to an Artificial Problem

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. The artificial agents are presented; An agent is a collection of cells, each with a defined behavior. These cells are laid out (connected) in a matrix of Grid Cells, and provided with an amount of food initially, a cell using one unit of food per discrete time step. Also in this environment are laid out patches of food; To survive longer, an agent 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.

Additionally, a second model of development is presented, one in which the relation between genotype and phenotype is bijective. The purpose of this inclusion is to demonstrate that a developmental model may outperform a bijective model.

**Worlds:** A world is an infinite two-dimensional matrix of Grid Cells. Each world contains one agent at the centre, and a distribution of food. There are no collisions - instead, an agent 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. Food is distributed differently, depending on world type. Distances between the agent's starting point and the food batches varies between low and high phenotypic complexity runs, the former being placed closer.

*Type 0* worlds contain eight batches of food, laid out in a circle surrounding the agent. *Type 1* worlds consist of a line of four patches of food, these patches being placed in successively longer distances in one direction. *Type 2* worlds consist of four patches of food placed in random locations, slightly farther away than the range of vision of the closest possible eye cell. *Type 3* worlds consist of 40 small batches of food distributed randomly in a donut shape surrounding the agent.

**Agents:** An agent is a collection of one or more cells, assumed to be connected. Each cell occupies one grid location. Agents behave as the sum of the behaviours of their cells. So long as one cell is declared "active", an agent is declared "active" - otherwise "inactive".

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 or absorbing food. All cells belong to one of the following classes: Eyes (Green), Nerves (Orange), Feet (Blue), Transports (Red) and Structure Cells (Gray).

*Eyes:* Eye cells can sense their external environment, and return a boolean value on the basis of the existence of food. However, the presence of other cells within its field of vision will block its ability to sense.

*Nerves:* Nerves are cells, which accept information from neighbouring eye or nerve cells, with up to four inputs, four outputs, or any combination thereof, determined by connections to eye cells. Nerves output the sum (*identity* nerves), the negative of the sum (*inverse* nerves), or the sum plus a random value from  $\{-1, 0, 1\}$  (*random* nerves).

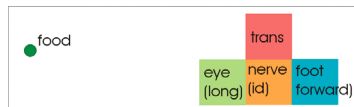
*Feet:* Foot cells accept input from all neighbouring nerve cells. Following all other computation, an agent sums the motion of each foot, and moves accordingly (weighted by total size of agent). *Forward* foot cells move *forward* (backward for negative input), and *rotation* foot cells rotate counter-clockwise (clockwise).

*Transports:* Transport cells manage the collection and distribution of food. At each time step, a transport cell will: collect food from its environment, and pass food to all neighbours in the eight-neighbourhood.

**An Agent in the World:** An agent is initialized in the centre of a world, each cell containing 200 units of food, with time defined as zero<sup>5</sup>. The agent next executes the following process at every time step, considering *only* active cells:

1. Replace any cells with no food with inactive cells (black, in the GUI)
2. Each transport cell collects any food from the environment
3. Each transport cell passes food to its neighbours
4. Compute the depth of each nerve cell, where a nerve has depth 1 if it is connected to an eye cell, 2 if it is connected to a nerve connected to an eye, etc. Random nerve cells which are not connected to an eye cell are also labeled depth one.
5. Eye cells are activated, returning 1 if food is within field of vision.
6. All nerve cells of depth one collect input and compute output, continue for each successive depth
7. Each foot cell collects input, adding output to the total
8. The agent moves accordingly.

Fig. 4.1 is an illustration of perhaps the simplest agents capable of finding and absorbing food. As a curiosity, consider Fig. 4.2, an agent in a similar situation; This agent's actions would cancel each other out, leading to immobility.<sup>6</sup>



**Fig. 4.1.** One of the simplest agents capable of finding and absorbing food

<sup>5</sup> Hence, if an agent does not find food, all cells will die at time 200; Typically, most cells do, as most agents have imperfect methods for food distribution.

<sup>6</sup> Schopenhauer, eat your heart out.

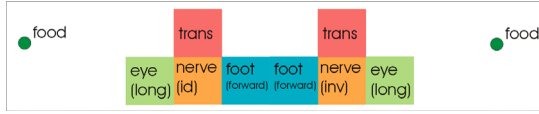


Fig. 4.2. An immobile agent

**Development:** Two methods of development are used: the Bluenome method, as introduced in section 2, and a Bijective method. The bijective method of agent 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 agent is developed by laying out the values of those integers one by one, in a spiral pattern, eventually forming a diamond of area  $2*(numTel+1)^2$  – hence, an agent with  $numTel = 6$  will have at most a genotypic and phenotypic complexity of 98; With  $numTel = 20$ , a complexity of 882. 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.

**Fitness:** The base fitness of an agent in the world is a measure of its size and length of life, relative to a world  $w$ .

$$fitness_{base}(a)^W = \sum_t numCells(a,t) \tag{1}$$

where  $numCells(a,t)$  is the number of living cells in agent  $a$  at time  $t$ . Note: since the amount of food in a world is finite, so is  $fitness_{base}$ .

To help the Bluenome model overcome the development of trivial agents, we introduce a bonus to fitness (for both Bluenome and bijective versions). We define  $numClasses \in [4]$  to be the number of those classes for which at least one cell exists in the fully developed agent.

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

Finally, our fitness function is:

$$fitness(a)^W = fitness_{base}(a)^W + fitness_{bonus}(a) \tag{3}$$

In any particular generation, an agent  $a$  will be subjected to two worlds,  $w_1$  and  $w_2$ ., chosen at random (our fitness is stochastic):

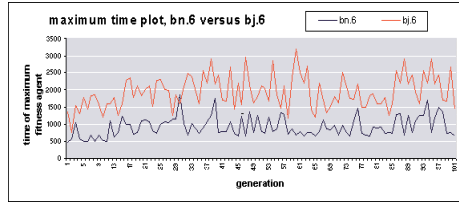
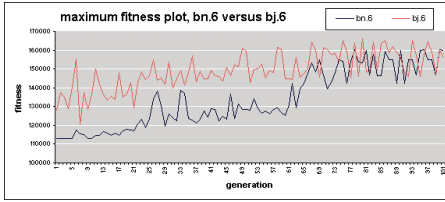
$$fitness(a) = fitness(a)^{w_1} + fitness(a)^{w_2} \tag{4}$$

**Experiments:** Evaluation of the Bluenome system involves a series of experiments, differentiated by the model for growth (Bluenome versus Bijective, or bn versus bj), and also by value of  $numTel$ . The experiments consisted of one run of each the Bluenome and Bijective systems for  $numTel \in \{6, 8, 10, 12\}$  (low phenotypic complexity). Additionally, there were three runs for each of the Bluenome and Bijective systems with  $numTel = 20$ ; (high phenotypic complexity).

## 5 Data and Analysis

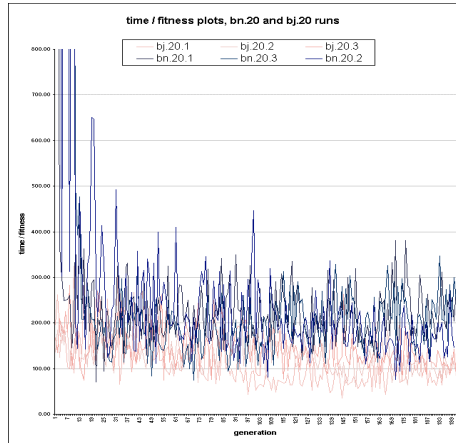
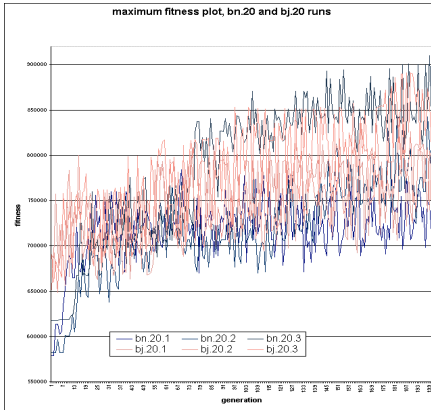
Data for the low phenotypic complexity runs ( $numTel \in \{6, 8, 10, 12\}$ ) showed little variance between values of  $numTel$ ; Hence, only data for the  $numTel = 6$  runs are shown.

In the low phenotypic complexity runs, the bijective runs outperform substantially, as illustrated in Fig. 5.1. Also, Figure 5.2 shows a comparison between maximum time for the  $numTel = 6$  runs – the bijective version typically outperforms the Bluenumo version. Contrary to initial expectations, this is not a boon, but instead a drawback. 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.



**Fig. 5.1.** Maximum fitness of the Bluenumo versus the Bijective run for  $numTel = 6$ . The Bijective run clearly outperforms initially; While the Bluenumo run catches up in later generations, it never reaches the same levels.

**Fig. 5.2.** Maximum time (of the most fit agent) plot for the Bluenumo versus the Bijective run, with  $numTel = 6$ . The Bluenumo version clearly shows a lower maximum time consistently.



**Fig. 5.3.** Maximum Fitness plots for three of each of the Bluenumo and Bijective runs, with  $numTel = 20$ . The Bijective runs (*light lines*) outperform initially, but two of the Bluenumo runs (*dark lines*) catch up by generation 150. One run can be seen overtaking the Bijective runs, beginning with generation 70.

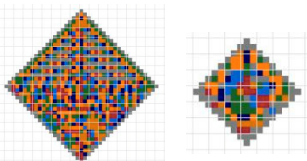
**Fig. 5.4.** Time / Fitness plots (of the most fit agent) of three Bluenumo runs (*dark lines*) versus the Bijective runs (*light lines*). The Bluenumo runs are clearly higher consistently.

Fig. 5.3 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 lowest of the Bluenome runs shows a similar course to the bijective runs, with some initial evolution and a seemingly random cycling of values following. However, the other two Bluenome plots show a continuous evolution proceeding up to generation 200, potentially continuing beyond this point. Additionally, the highest run quickly shows consistent maximum fitness values, which exceed the maximum, found in any of the bijective runs. Figure 5.4 clearly shows the continuation of the fitness / time trend – the Bluenome models clearly distribute food between body components more evenly.

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. In Phase Two, an experiment was undertaken which tested a similar situation – that of the re-use of an agent’s genome in a differing setting. Populations of genomes were selected from a high-phenotypic complexity run ( $numTel = 20$ ) 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). The values obtained are comparable to the  $numTel = 8$  run. In Table 5.1, maximum and mean fitness values are compared between the re-grown agents and a late population from the  $numTel = 8$  run. While the original 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. Fig. 5.5 shows the first agent of the  $numTel = 20$  run 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 (see below).

**Table 5.1.** Maximum and Mean fitness values from re-grown agents

	mean fitness	maximum fitness
original $numTel = 8$ pop., generation 94	75 252.67	126 364
re-grown agents, mean over three evaluations	73 211.06	119 344

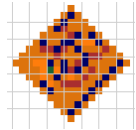


**Fig. 5.5.** An agent from a run with  $numTel = 20$ , generation 180 (left), re-grown having changes the value of  $numTel$  to 8 (below).

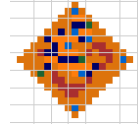
It has been noted that Phase Two presents an artificial problem for which human designers would experience difficulty; Three (of many) identified agent strategies are presented in the following figures: Fig. 5.6 illustrates a member of the *blind-back-and-forth* strategy: This strategy may be viewed as a local optimum which often

dominates early generations. These agents typically do not include eye or rotation foot cells, relying instead solely on random nerves and forward cells, moving back and forth on the x-axis. This strategy works marginally well for worlds of type 0 and 3, but rarely for other worlds; Fig. 5.7 illustrates a member of the *rotate-then-forward* strategy, perhaps the most successful strategy found - The agent rotates randomly, until it sees food, then moves forward; Fig. 5.8 illustrates a member of the *position-then-rotate* strategy, another local optimum: initially the agent moves forward, until it is at the same distance as the first batch of food. Then, it begins to trace a constant circular path – this strategy works poorly on most worlds, except on type zero, where it may be a global optimum.

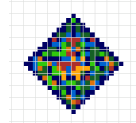
**Fig. 5.6.** Example of the *Blind Back-and-Forth* Strategy. The dark blue cells are Forward Foot cells, the mid-orange cells Random Nerve cells. The agent contains no Eye of Rotating Foot cells whatsoever, save a single Eye cell near the centre (its location guarantees it will never fire). The single Eye cell is included probably for the sole reason of maximizing fitness<sub>bonus</sub>



**Fig. 5.7.** An example of the *Rotate-then-Forward* Strategy. There are two Eye cells on the periphery of the agent – 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.



**Fig. 5.8.** Example of 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.



## 6 Conclusions

In Phase One, several fitness functions were used to evolve images demonstrating suggestive principles. In nearly all cases, successful evolution was achieved quickly, generating images recovering some of the complexity of two-dimensional CAs.

In Phase Two, the Bluename model was applied to a non-trivial artificial problem, one which involved the coordination of many non-linearly interacting components. In cases of low phenotypic complexity, the bijective methodology tended to outperform the Bluename method, with a wide margin in initial generations, barely so in later generations. In cases of high phenotypic complexity, however, one of the Bluename runs clearly outperformed all of the bijective runs, with a second matching with potential for further growth in later generations. The Bluename methodology continued to develop in a high-dimensional space, while the bijective methodology stagnated early on.

In addition to this performance increase in high complexity runs, the Bluename model showed an inherent ability to generate agents with better developed systems for the distribution of food throughout the body – this is no doubt a result of the

inheritance of cell specialization creating a network of transport cells, a readily observed instance of the sorts of structural patterns inherent to the developmental process [5]. Finally, the resistance of the developmental process to changes in the environment was demonstrated. More intriguing still is the continuation of performance by the re-developed agents, both in terms of valuation by the fitness function in question, and in visual appearance.

A matter touched upon in the above discussions is the view of evolution as a mechanism for controlling complex processes; Indeed, this is an intriguing hypothesis, perhaps contributing to the success of the above system; If true, however, it begs an obvious question: by what mechanism? It is the hope of the authors that systems like *Bluename* may serve as a test bed by which this claim may be evaluated and studied further.

## References

1. 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
2. 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, (1996)
3. 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)
4. Fagotto, F., Gumbiner, B., Cell contact-dependent signaling, in *Developmental Biology* 180 (1996)
5. Furusawa, C., Kaneko, K., Emergence of Rules in Cell Society: Differentiation, Hierarchy, and Stability, in the *Bulletin of Mathematical Biology* (1998)
6. Hamahashi, S., Kitano, H. Simulation of *Drosophila* Embryogenesis, in the Proceedings of the Sixth International Conference on Artificial Life (1998)
7. Hotz, P., Gomez, G., Pfeifer, 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* (2003)
8. Karp, G., *Cell and Molecular Biology*, 3<sup>rd</sup> Ed., (John Wiley & Sons, Inc.; 2001)
9. Kowaliw, T., *Bluename: A Novel Developmental Model for the Evolution of Artificial Agents*, Master's Thesis, Concordia University (2003)
10. Kvasnicka, V., Pospicjal, J., Emergence of Modularity in Genotype-Phenotype Mappings, in *Artificial Life*, v. 8, no. 4 (2002)
11. 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, 2001
12. Stanley, K., Miikkulainen, R., A Taxonomy for Artificial Embryogeny, in *Artificial Life*, v.9, no. 2 (2003)
13. Wolfram, S.: *A New Kind of Science*. Wolfram Media New York (2002)