

# A Cell-based Developmental Model to Generate Robot Morphologies

Sylvain Cussat-Blanc

DEMO Lab, Volen National Center for Complex System - Brandeis University MS018  
415 South street, Waltham, MA 02454, USA  
cussat@brandeis.edu

Jordan Pollack

DEMO Lab, Volen National Center for Complex System - Brandeis University MS018  
415 South street, Waltham, MA 02454, USA  
pollack@brandeis.edu

## ABSTRACT

This paper presents a new method to generate the body plans of modular robots. In this work, we use a developmental model where cells are controlled by a gene regulatory network. Instead of using morphogens as in many existing works, we evolve a more flexible “hormonal system” that controls the inputs of the regulatory network. By evolving the regulatory network and the hormonal system in parallel with a blind watchmaker, we have generated various virtual robots with interesting inherent properties such as regularity and symmetry. The prototypes of the robotic blocks that will be used to actually build the real machines are also presented in this paper.

## Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics

## Keywords

Modular robots, Cell-based developmental model, Gene regulatory network, Hormonal system, Body plans

## 1. INTRODUCTION

Modular robots built over the last several years have gotten more and more complex. Usually based on cubic blocks, they embed complex mechatronics to make them more accurate, more efficient and more diversified [25]. In this domain, the “Molecube” may be seen as an example. First based on a cube able to turn around its central axis to demonstrate limited self-replication, Lipson and his team went on to develop new modules to build more complex machines [26]. For example, we can cite the gripper modules able to pick up objects, the wheel cubes, the camera cubes, etc. To control these blocks, a neural network is then evolved. However, the user has to imagine the morphology of the robots by himself. It means that, in case of modification of the machine’s purpose, a human has to redesign the morphology of the robot before evolving its controller. In 2007, Christensen proposed

a robotic approach where the behavior of the machine, also composed of rotating cubic blocks, is strongly linked to its morphology [5]. Indeed, instead of having a fixed morphology, the robot is self-reconfiguring in order to accomplish its function. This method generates a very interesting property: generated robots are capable of self-healing, as multicellular organisms are.

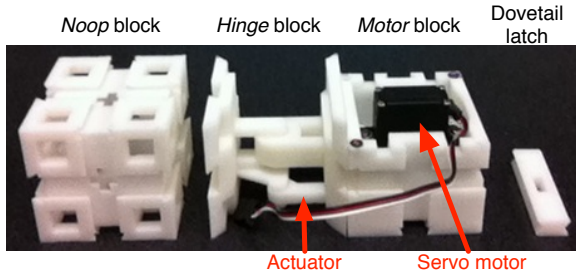
On the other side, many works have been done in simulation to generate virtual modular robots. The first major work evolving realistic virtual robots is Karl Sims in 1994 [24]. It took another six years before evolved robots took a step from virtual to reality [18]. Sims used a graph-based genotype-phenotype map to generate a morphology controlled by a neural network, while Lipson & Pollack directly evolved the morphology. Both systems are coevolved so that the controller is as much as possible adapted to the morphology. L-Systems have also been used for the same purpose. Komosinski has made one of the most advanced work in this domain. Here again, a neural network is coevolved with the morphology generator (a L-System) to produce artificial robots. More recently, artificial Gene Regulatory Networks (GRN’s) appear to be able, at the first hand, to generate complex morphologies when they control a developmental model [11, 3] and, at the second hand, to control different kinds of agents [16, 19]. Schramm was even able to grow a digital aquatic worm that moves by oscillation generated by the same regulatory network [22].

However, few of these works have been designed to actually build a real robot. With this idea in mind, Hornby proposed a model where a L-System is evolved to control a “turtle” factory which could produce a virtual robot controlled by a neural network [14]. Here, the best robots have actually been built by hand and were capable of moving like the virtual robots.

The main motivation of this paper is to use a cell-based developmental system driven by a gene regulatory network to generate real robot morphologies. To present this new method to generate robots’ body plans, the paper is organized as follows. The next section presents the prototypes of real robotic blocks that will be used to build the robots. Section 3 presents the developmental model, the gene regulatory network and the hormonal system. In section 4, the blind watchmaker interactive evolutionary method is described. Section 5 presents a set of results obtained with this model and proposes a discussion about the complexity of the morphologies and the emergent properties obtained such as regularity and symmetry. The paper concludes with possible extensions of this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO’12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.  
Copyright 2012 ACM 978-1-4503-1177-9/12/07 ...\$10.00.



**Figure 1: The robots are composed of three different blocks (from left to right): a *noop* block, a *hinge* block and a *motor* block. Blocks are tied together by the use of dovetail *latches* inserted between the blocks.**

## 2. THE ROBOTIC BLOCKS

In this work, we took inspiration from living organisms to design our machine. An organism is usually comprised of billions of functional units: the cells. Cells can have various functions (neural, muscular, etc.). Thus, our machine uses multiple blocks that have different functions. The blocks must be easy to assemble to be able to conceive an automatic assembly process. When a machine is built, its morphology will not evolve anymore (as many grown living beings do not modify their morphology). With this constraint, the blocks do not need a dynamic linking and unlinking system. Finally, each block must be cheap to produce to keep the robot's overall cost low.

With all these requirements, we have designed each block as a cube, scaled to 2 inches (about 5cm) per side. As presented in figure 1, three different units are used in the work: (1) A *noop* block is used to build the structure of the robots (such as bones are the structure of the body). This block does not have any function, except the capacity to be linked to other blocks. (2) A *hinge* block can rotate around a central axis within the range  $[-\pi/3; \pi/3]$ . This block does not move by itself, but can be actuated by a neighboring motor block through two actuators, as tendons are actuating a real articulation. (3) A *motor* block actuates the *hinge* block. The actuators protubing from the hinge blocks are tied to the motor so that the motor can actuate the hinge.

This choice to separate the hinge and the motor comes from a biological observation. In Nature, most articulations and muscles are separated. They are linked together with tendons. With this method, muscles can have various shapes and sizes, according to local constraints (space, needed strength, etc.). We decided to use this idea for our building blocks for two main reasons. First, motors are usually the heaviest parts of the robots. Modifying the position of the motors in the robot can modify its center of mass, which can be relevant. Secondly, we can also imagine the possibility to increase the strength of a joint by having multiple motors actuating the same hinge (as multiple muscular cells actuate the same articulation).

Because, our robotic system does not reconfigure itself since each block's position is given by the evolved body plan of the robot (its morphology), we can keep the linking system as simple as possible. As presented in figure 1, it consists in a dovetail latch slipped between two blocks.

These blocks are currently at the stage of prototypes but are close to a final version. A 3-D printer is used to produce

the plastic structure of the blocks and the dovetail latches. Servo motors (Futaba S3102) are embedded in the motor blocks and are linked to the hinge block via the actuators. These actuators are still the main issue of realization: they need to be strong, light and inelastic. Three solutions have been tested: using nylon strings, steel strings or having the actuators embedded to the hinge, as presented in figure 1. Because the two first solutions provide very unprecise movement due to the complexity of the strings' length calibration, the last solution should be the one adopted for the final robots. It requires a more elaborated conception of the hinge blocks but strongly simplify the assembly process of the blocks and increase the precision of the movement of the global articulation.

To produce the plans of the robots, we propose a novel approach based on a developmental model in which cells are controlled by a gene regulatory network coupled to a novel hormonal system. In comparison to other methods such as Graphtals or L-Systems, this approach has the advantage to be biologically plausible. Moreover, some properties such as regularity and symmetry emerge from the system, without any prior encoding. The next section presents this model.

## 3. THE DEVELOPMENTAL MODEL

### 3.1 The cells and their environment

The developmental model is based on cells that act in a 3-D discrete environment. Each cell has a unique identifier given at its birth, a division plan and a position in the environment. Two main types of cells are available: (1) *undifferentiated* cells or *stem* cells that can divide as much as they want to and (2) *differentiated* cells coming from a specialization of a stem cell to one of the three following types of cells: *B-cells*, *H-cells* or *M-cells*. When a cell is differentiated, it cannot divide anymore. The use of stem cells to drive the organism's growth has already been successfully used to developed complex shapes by Fontana in [12].

At each time step of the simulation, the stem cells can perform one of the following actions:

- *Symmetric division*: when a stem cell performs this action, a new stem cell is positioned in the position of the mother cell while the mother cell migrates in direction to its division plan. Whereas the mother cell keeps the same identifier, the new cell receives a new unique one. Its initial division plan is oriented as its mother's one.
- *Asymmetric division*: a stem cell can divide to create a new differentiated cell. In this case, the mother cell first produces an exact copy (same internal state) of itself in the direction indicated by its division plan. While the newly created cell stays a stem cell, the mother cell differentiates to a *B-cell*. Thus, this cell cannot divide anymore.
- *Differentiation*: a stem cell can become a *B-cell*. Because a differentiated cell cannot divide, this action stops the proliferation of a stem cell.
- *Rotation*: a cell can reorient its division plan. When it choses this action, it can turn in one of the four following directions: *right*, *left*, *top* or *bottom*. While turning, the cell stays at the exact same position.

- *Special division*: a cell can divide asymmetrically twice during one time step. In this case, it first produces a *H-cell* at the initial position, a *M-cell* one voxel away from the initial position in the direction of its division plan and a copy of itself two voxels away.

Naturally, two cells cannot be in the same voxel of the environment. To satisfy this requirement, all inconsistent actions are disabled at a time step of the growth. For example, if a cell wants to divide, the position in direction of the division plan is first checked. If the position is not free (in other words, already occupied by another cell), the division is disabled. The cell will have to choose another action. At least two actions are guaranteed to be always available: *rotation* and *differentiation*. Indeed, both actions only affect the cell itself: no new voxels are necessary.

After the growth of the organism from a single cell to a connected set of cells, each cell is translated to a robotic block: *B-cells* and *stem cells* are interpreted as *noop* blocks, *M-cells* produce *motor* blocks, and, finally, *H-cells* produces *hinge* blocks.

To evaluate the morphologies generated, a virtual robot is built and dropped down in a 3-D physics environment. This simulator is used to visualize the produced robots before actually building them. We use the well-known Bullet physics engine coupled with OpenGL for the graphic visualization.

To select an action, cells are using a gene regulatory network. The next section details our model based on a simplification of Banzhaf’s GRN [1].

## 3.2 The gene regulatory network

### 3.2.1 Background on GRN’s

In Nature, each cell of a living organism has a gene regulatory network (GRN) that controls its behavior. The possible actions are coded by genes in its DNA and their expressions are controlled by the regulatory network [8]. The GRN uses external proteic signals from the cell’s environment to activate or inhibit the transcription of genes. Cells can collect external signals thanks to protein sensors localized on the cell membrane. The expression of genes determines the cell’s behavior.

Artificial regulatory networks are based on the same principle. Torsten Reil was one of the first to propose a biologically plausible simulation of a GRN [20]. He defined the genome as a bit string. In his model, each gene started with the particular sequence of 4 bits, named the “promoter”. The dynamics of this regulatory network is very close to Banzhaf’s, detailed below.

Banzhaf proposed an artificial regulation network model strongly inspired by real gene regulation [1]. The aim of this model was to study the behaviors generated by a biologically plausible regulatory network. He has generated many random bit strings and observed various behaviors from oscillators to faster or slower transitory patterns.

Regulatory networks have been proven successful to grow cell based artificial creatures with more or less complex morphologies [11, 10, 15]. More recently, regulatory networks have also been used as behavior generators to solve the pole balancing problem [19] or to control a foraging virtual robot [16]. Schramm also used a GRN to control the morphology development and the behavior of swimming worms [22].

However, no real robotic application has been yet proposed for this system. Our work proposes to use a simplifi-

cation of Banzhaf’s regulatory network to control the cells of our developmental model.

### 3.2.2 Presentation of Banzhaf’s regulatory network

Banzhaf’s GRN may be currently one of the best models. It has been designed to be as close as possible to a real gene regulatory network. As DNA is composed of a sequence of nucleotides, Banzhaf’s network is encoded within a sequence of bits. As a real gene starts with the particular sequence of nucleotides, a gene in Banzhaf’s network starts with a particular sequence of 8 bits named the “promoter”. A gene is then encoded next to this sequence by 5 32-bit integers, named the “sites”. This mechanism allows the generation of a variable number of genes in a fixed size chromosome. However, as in Nature, it also generates a certain amount of noncoding DNA, the probability to have a promoter being very low ( $2^{-8}$ ). This noncoding DNA (98% of human DNA) is thought to be used in Nature to protect the genome from mutation by lowering the probability that a mutation will affect a coding nucleotide.

Banzhaf’s model has not been designed to be evolved nor to control any kind of agent. However, Nicolau used an evolution strategy to evolve the GRN to control a pole-balancing cart [19]. Even if the cart behavior has been shown consistent within its environment, the evolution of the GRN has been an issue. In our opinion, the difficulty of the evolution is due to: (1) the *noncoding DNA* and (2) the *dynamics* of the network. The next section details these two points and proposes a possible adaptation of the model to be more efficient as a computational model.

### 3.2.3 Adaptation of Banzhaf’s model to an efficient computational model

#### Encoding of the network

As presented above, the *noncoding DNA*, which may have positive benefits in natural systems, impedes the evolution of artificial GRNs.

To get rid of the noncoding DNA, we propose a new encoding of the network. In our model, each protein is encoded as an indivisible component composed of four attributes:

- The protein *identifier* coded as an integer between 0 and  $p$ . The upper value  $p$  of the domain can be changed in order to control the precision of the GRN. In Banzhaf’s work,  $p$  is equal to 32, which is the size of a site. In our work, we have decided to increase the precision by setting  $p$  to 64.
- The *enhancer identifier* coded as an integer between 0 and  $p$ . The enhancer identifier is used to calculate the enhancing matching factor between two proteins.
- The *inhibitor identifier* coded as an integer between 0 and  $p$ . The inhibitor identifier is used to calculate the inhibiting matching factor between two proteins.
- The *type* determines if the protein is an *input* protein (whose concentration is given by the environment of the regulatory network and which regulates other proteins but is not regulated), an *output* protein (whose concentration is used as output of the network and who is regulated but does not regulate other proteins) or a *regulatory* protein (internal protein that regulates and is regulated by other proteins).

The GRN is defined as a set of these proteins. A GRN can have a variable number of proteins. To be evolved, the GRN is encoded in a genome defined as a variable length chromosome of indivisible proteins. Each protein is encoded within four integers: three between 0 and  $p$  for the three different identifiers and one between 0 and 2 for the type of the protein. If an evolutionary algorithm has to evolve this chromosome, the modification operators have to be re-defined. First, the *crossover* consists in exchanging subparts of two different networks. Because proteins are indivisible, the crossover points have to be chosen between two proteins. It ensures the integrity of each sub-network. The local connectivity is thus kept. Only new links between the different sub-networks are created. The *mutation* can be applied in three equiprobable ways: *mutating an existing protein* by randomly changing one of its four integers, *adding a new protein* randomly generated or *removing one protein* randomly chosen in the network.

### Dynamics of the network

The *dynamics* of the network is, in our opinion, a crucial point that determines the reactivity of the network. In Banzhaf's network, the enhancing dynamics of each protein are given by the following equation:

$$e_i = \frac{1}{N} \sum_j^N c_j e^{\beta u_j^+ - u_{max}^+}$$

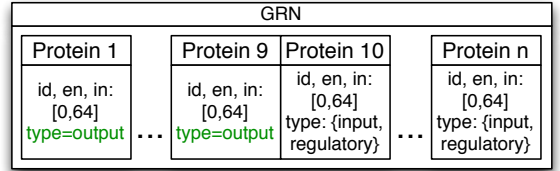
where  $e_i$  is the enhancing value for a protein  $i$ ,  $N$  is the number of proteins in the network,  $c_j$  is the concentration of protein  $j$ ,  $u_j^+$  is the enhancing matching factor between protein  $i$  and protein  $j$  (calculated by counting the number of different bit of the proteins signatures) and  $u_{max}^+$  is the maximum enhancing matching factor observed.  $\beta$  is a control parameter described later. The modification of the concentration of a protein  $i$  is then given by the differential equation  $dc_i/dt = \delta(e_i - h_i)c_i - \Phi$ , where  $e_i$  is the enhancing signal previously presented,  $h_i$  is the inhibiting signal (calculated exactly as the enhancing signal by using inhibiting matching factors instead of the enhancing matching factors),  $c_i$  is the current concentration of the protein  $i$  and  $\Phi$  is a function that keeps of the sum of all protein concentrations equal to 1.

As presented in this last equation, the evolution of the concentration of a protein  $i$  is directly related to its current concentration. In other words, if a protein's concentration appears to be null, this protein will never be produced anymore. This is problematic because, in the case of a computational model, it means that an output completely disabled at a particular time of the simulation will never be activated anymore. With this observation, we propose to change the differential equation by the following one:

$$\frac{dc_i}{dt} = \frac{\delta(e_i - h_i)}{\Phi}$$

With this new equation, the current concentration of the protein  $i$  does not intervene anymore. The scaling function is also divided instead of subtracted in order to have a more proportional scaling factor.

With this new encoding and these new dynamics, the regulatory network should be easier to evolve and more reactive to its environment. Next section presents how this regulatory network is embedded within the cells of our developmental system.



**Figure 2: The chromosome that encodes the GRN is subdivided into two parts: the first nine proteins are output proteins and the remainder of the chromosomes encodes input and regulatory proteins.**

### 3.2.4 Encapsulation into the cells

Cells have 5 different actions: *symmetric division*, *asymmetric division*, *differentiation*, *division plan reorientation* (rotation) and *special division*. However, if a cell chooses to rotate, it also has to decide in which direction it wants to (left, right, top or bottom). To encode this in the GRN, we decided to subdivide the outputs into two groups of output proteins: the first group contains 5 output proteins that match with the 5 possible actions and the second group contains 4 output proteins that encodes the 4 possible rotation directions when the rotation is selected in the first group of proteins. Thus, 9 output proteins are necessary to encode the possible actions of the cells.

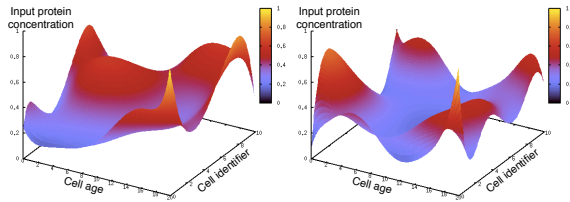
As illustrated by figure 2, these 9 output proteins are encoded at the beginning of the genome. Their three identifier (protein, enhancer and inhibitor) can be mutated but the type cannot. The remaining elements of the genome are composed of 25% of input proteins and 75% of regulatory proteins (empirically chosen<sup>1</sup>). It can be mutated and crossed over as previously presented.

At each step of the simulation, the cells execute the regulatory network for 25 steps (found to be within reasonable range during preliminary tests) and select the action corresponding to the maximum protein concentration of the first 5 proteins: if the first protein has the maximum concentration, the cell divides symmetrically, if the second protein has the maximum concentration, the cell divides asymmetrically and so on. When the *rotation* action is chosen, it gives access to the second group of output proteins. The cell calculates the maximum value of the last 4 output proteins' concentrations to determine in which direction to turn.

The coefficients  $\beta$  and  $\delta$  presented in the dynamics model are also encoded in the genome. They modify the dynamics of the network.  $\beta$  affects the importance of the matching factor and  $\delta$  affects the level of production of the protein in the differential equation. The lower both values, the smoother the regulation. At the opposite, the higher the values, the more sudden the regulation. Both values are encoded as a second independent chromosome that only contains both values. They are coded with double-precision floats in  $[0.5; 2]$  (empirically chosen<sup>1</sup>).

Finally, we must provide inputs to the regulatory network in order to be able to select an action at every time step of the growth. In most developmental models, cells can produce morphogens such as in [17, 15]. They are informative molecules that diffuse in the environment and are the in-

<sup>1</sup>Empirically chosen values have been determined by a series of pre-tests to find a reasonable range.



**Figure 3: Examples of generated inputs with Bézier surfaces. They show the diversity of signal we can obtain by using this generative method.**

puts of the GRN. In few other models, the morphogens are hand-coded in the environment [4, 7]. This method is less realistic than the first one but strongly reduces the complexity of the simulation. Because of its complexity, the first method generally leads to simple shapes such as 2-D colored rectangles, 2-D salamanders or 3-D tubes. The second method make possible to generate more complex 3-D colored structures such as cubes, diamonds or spheres. In this work, we choose an intermediate solution. Cells have a hormonal system that produces the inputs of the regulatory network. The next section details this system.

### 3.3 The hormonal system

The genotype of our developmental model also describes a novel hormonal system. This hormonal system generates the inputs to the gene regulatory network based on the timestep and index number of the cell. In order to be efficient, each input protein of the GRN must have a different hormonal production profile that provides the concentration of the input protein all along the simulation. Moreover, this profile must also be different for each stem cell in the organism. Indeed, if all the cells would have the same profile, it would not allow any variation of their behaviors: they would all perform the same sequence of actions. In summary, the concentration  $c_i$  of an input protein  $i$  is related to two parameters: the current time step  $t$  of the simulation and the identifier of the cell in which the concentration is calculated.

Many models of artificial hormones exist in the literature. From those applied to robotics, we can cite AHHS used to control the behavior of Symbion and Replicator reconfigurable robots [13]. Here, an hormone is diffusing through the robot and is used for local communication purpose.

Thus, the hormonal profile of an input protein can be coded by a 3-D surface. The first two dimensions of the surface correspond to the simulation time in abscissa and the identifier of the cells in ordinate. The third dimension gives the concentration of the corresponding input protein's concentration according to the two first dimensions. The global hormonal system is thus defined as a set of  $p$  3-D surfaces. Each surface is associate with the identifier of the input protein (which varies between 0 and  $p$ ,  $p$  being the precision of the GRN).

To generate these surfaces, we used Bézier's algorithm [2]. Originally used to design car bodyworks, it allows the generation of a surface by the mean of a given set of checkpoints. In this work, we use a Bézier surface of order ( $n = 10, m = 10$ ). This order corresponds to the number of checkpoints. For a (10, 10) order surface,  $(10 + 1) * (10 + 1) = 121$  checkpoint  $k_{i,j}$  are necessary. With these checkpoints, the  $z$  coordinate of every point of coordinate  $(x, y, z)$  of the surface

is given by the following equation:

$$z = \sum_{i=0}^{n=10} \sum_{j=0}^{m=10} B_i^n(x) B_j^m(y) k_{i,j}$$

where  $B_i^n(x)$  is the Bernstein polynomial:

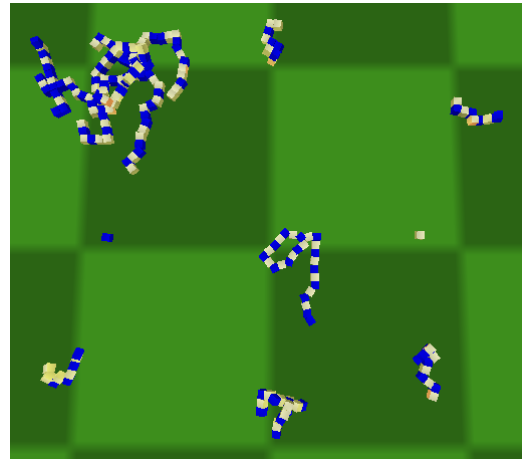
$$B_i^n(x) = \frac{n!}{i!(n-i)!} x^i (1-x)^{n-i}$$

Figure 3 presents some examples of randomly generated surfaces using this method. They show the diversity of possible landscape generated with only 121 checkpoints.

Each possible input protein must have a Bézier surface assigned. To do so, we have decided to allocate a surface to each possible protein identifier. When the identifier of an input protein is changed by mutation to another value, a new surface is thus assigned. With this method,  $p$  surfaces are necessary for a GRN of precision  $p$  (a protein identifier is coded with an integer between 0 and  $p$ . Here  $p = 64$ ). To encode this system into the genome, the checkpoints are encoded in the third chromosome that consists in a common set of  $121 * p$  double-precision floats. Each value can vary in the range  $[-1.0; 1.0]$ . This chromosome can be classically crossed over and mutated. In order not to overflow the regulatory network with the input proteins, the input concentrations given by Bézier surfaces are normalized between 0 and 0.05.

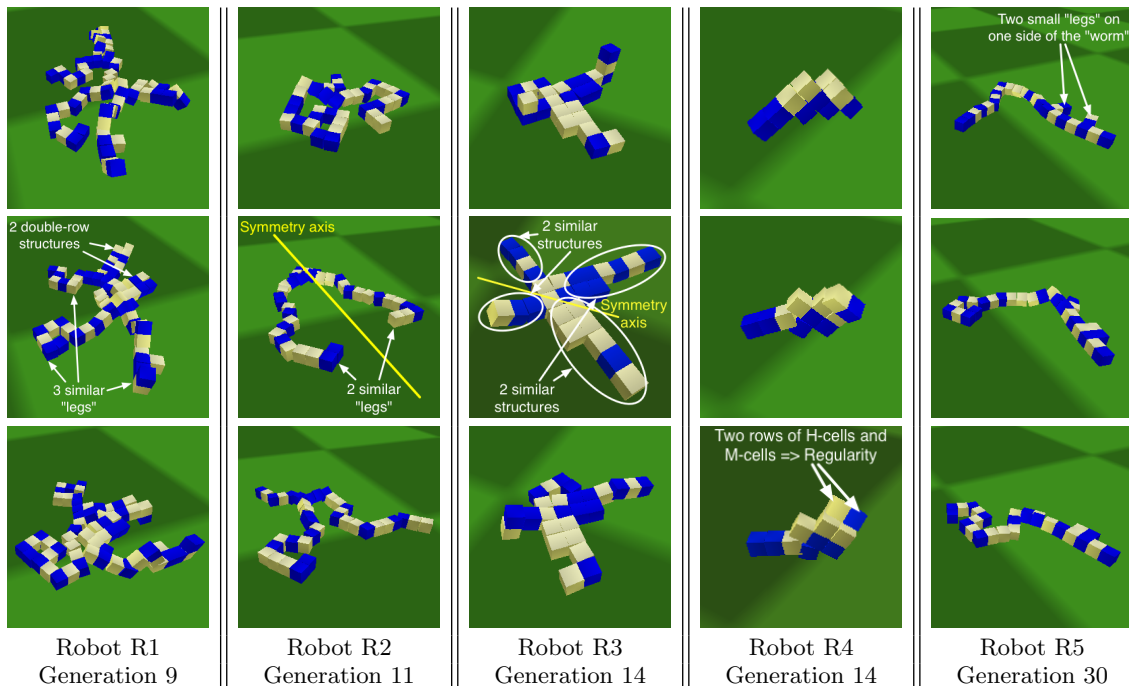
### 3.4 Summary of the model

To summarize, cells are acting in a 3-D discrete environment. Stem cells can divide symmetrically, asymmetrically or twice in the same time step to produce an articulation. They can also reorient their division plan and differentiate to *B-cells*. To control their developmental behavior, cells have a gene regulatory network coupled to a hormonal system. We have proposed an adaptation of Banzhaf's well-known GRN to make it more efficient for computational purposes.



**Figure 4: 9 robots evolve at the same time in the physics simulator. A 6-second phase oscillator generates the robots' movements. The user can select the robot that he wants to evolve. The evolution only consists in mutating 10% of the genome. Spacing between robots has been reduced for this particular picture.**





**Figure 5: Examples of robots obtained in a single run of the blind watchmaker. Generated robots are complex and diversified, especially in few generations with few individuals evaluated at each generation (9 robots only).**

The hormonal system uses Bézier surfaces to generate the inputs to the GRN. Each input protein is produced by a hormone, a 3-D surface indexed by the simulation time and the cell identifier.

In order to evolve the organisms (and thus the robots' body plan), both systems are encoded in a genome composed of three independent chromosomes: the first chromosome encodes a set of indivisible proteins used to build the architecture of the GRN, the second chromosome encodes the dynamics parameters and the third chromosome encodes the checkpoints necessary to generate the hormonal system's Bézier surfaces.

In order to evaluate the complexity of possible morphologies generated by this model, we have decided to use a Blind WatchMaker instead of a classical evolutionary algorithm. The aim is to study the capacity of the GRN to inherently produce properties that other models usually include such as symmetry or regularity. We also want to verify that our system is able to generate a wide variety of morphologies keeping in mind its possible evolution with an evolutionary algorithm.

#### 4. EVOLUTION OF THE ROBOTS: THE BLIND WATCHMAKER

The Blind WatchMaker is an interactive evolutionary method first proposed in 1986 by Richard Dawkins [9]. He originally used this method to sustain the theory of natural evolution using a pedagogical model called *biomorphs*, fractal-like creatures generated with a small set of genes. This method gave birth more recently to the field of interactive evolution. Many applications are nowadays based on this principle to solve various problems. It has for example been used

with genetic programming to generate realistic camouflage [21], or with HyperNeat to generate 2-D pictures [23] or 3-D shapes [6].

In this work, we first generate 9 random genomes. The developmental model is then executed to generate the corresponding virtual robots. When the robots are created, they are dropped down in the physics simulator in 9 different points of the environment so that they cannot interact with one another. Figure 4 illustrates nine random robots evolving in the physics simulator. In this particular experiment, we have limited the number of developmental steps to 20 and the number of stem cells to 10 in order to control the scale of morphologies. Later, these limits will be needed based on the actual availability of robotic cells, but might be based on simulating biologically plausible mechanisms.

During the first 5 seconds, the robots do not move in order to stabilize their structure on the floor. Then, to make the robots move in the environment, an oscillator controls the movement of every hinge of the robots: they rotate clockwise for 3 seconds (the rotation angle of a hinge cannot be above  $-\pi/3$ ) and then counterclockwise for 3 seconds (within a  $\pi/3$  limit). A third of the oscillators are inverted: when it should turn clockwise, an inverted hinge actually turns counterclockwise and vice versa. Even though this method generates simple movements, it is sufficient to visualize the potential of the generated morphologies.

The user can use window controls to navigate from a robot to another to observe its movement and its properties. The user can also select the robots he wants to evolve. The application then generates 9 new robots by mutating 10% of the selected robot's genome. We have decided not to use the crossover operator to enhance the diversity of generated

robots. For the same reason, the mutation rate has been deliberately chosen high.

With this method, we have generated a pool of diversified robots. The next section presents some of them and discusses properties of robots obtained.

## 5. RESULTS AND DISCUSSION

Figure 5 presents 5 robots obtained during the same run of the interactive evolutionary algorithm. These examples show the diversity of morphologies obtained.












The first observation we can make about these robots is the capacity of the regulatory network to generate robots of different sizes. The GRN is able to generate as well small robots (such as robot R4) or bigger ones (such as R1 or R2). This is mainly possible by using the differentiation action that stops the proliferation of the stem cells.

Moreover, interesting properties emerge during the evolution. After only 9 generations, the robot R1 already presents some regularity in its morphology. We call a *regularity* a set of cells with the same organization (relative positioning and differentiation of the cells) that appears at different places in the organism. For example, the robot R1 is composed of 3 similar “legs” and two identical structure build with double-row structure of cells (see the second picture of R1 on figure 5). The robot R2 also has some regularities. It is composed of 2 long “legs” of the same size. The same observation can be done on robots R3 and R5 (2 small legs on one side for the robot R5).

In some cases, these regularities can be very interesting: on the robot R4, it produces a double row of H-cells and M-cells, which generates a double joint in the robot. This structure multiplies the strength of the resulting joint. Figure 6 presents the growth sequence of the robot R4. The double-row cell structure appears at the beginning of the growth, from step 1 to 3. The initial cell first divides. Then both cells perform the same actions for 2 steps, before changing their behaviors, certainly because of a change in the hormone profile of some input proteins at this particular point of the simulation. While the first cell produces the left part of the robot, the second cell quickly stops its development after few more (parasitic) reorientations of its division plan.

As proven by the blind watchmaker approach, some of the robots can be symmetric such as robots R2 and R3 on figure 5. This property is particularly interesting because it is usually expected in robot’s morphologies. A symmetric machine will have more chances to be stable. The symmetry seems to be achieved using a similar method which achieves regularity – but with a rotation before two equivalent sequences of actions are performed.

The use of a developmental model to generate robot body plans seems to be a promising approach. Some desirable properties such as regularity and symmetry is easily obtained with this system. As presented above, we have obtained these properties in few generations with a very small population. This is not necessarily the case in works based on HyperNEAT or L-Systems which requires many more generations of evolution. Moreover, the hormonal system provides for much more creativity than a classical morphogens production regulated by the GRN. For now, the profiles of the hormone are encoded in the genome, but we could easily imagine using hormones and morphogens simultaneously. Morphogens produced by the cells could, for example, control the production profile of the hormones.

Step	Cell 1	Cell 2	Cell 3	Cell 4	Organism
0	Birth				
1	SyD2	Birth			
2	TurnB	TurnB			
3	SpD	SpD			
4	SyD3	TurnB	Birth		
5	TurnB	TurnB	SyD4	Birth	
6	TurnB	Diff	Diff	TurnB	
7	TurnB	-	-	Diff	
8	SpD	-	-	-	
9	AsD	-	-	-	
10	Diff	-	-	-	

**Figure 6: Actions performed by the stem cells only. Each line corresponds to a growth step. Signification of the acronyms: SyD $x$ =Symmetric division to create a new cell  $x$ ; AsD=Asymmetric Division, SpD=Special division, Diff=Differentiation and TurnB=Turn to Bottom**

## 6. CONCLUSION AND PERSPECTIVES

This paper presents an innovative system to generate robot body plans. It is based on a developmental model in which cells are controlled by a gene regulatory network. We have based our GRN model on Banzhaf’s by modifying the encoding of the GRN into a chromosome without any non-coding DNA and by modifying its dynamics. The resulting GRN is easier to evolve and is more adapted to a computational purpose. Instead of using morphogens as in most developmental models, we have decided to provide input to the GRN using a hormonal system which is differentiated by the cell identifier, the input protein, and the simulation time. These hormone profiles are generated by the use of Bézier surfaces controlled by a finite set of 121 checkpoints which are evolved with the GRN.

We have produced various virtual robots by evolving in the same time the GRN and the hormonal system with a blind watchmaker. Resulting robots have various sizes, from small to very large, and two interesting properties emerge from the systems: many creatures have regularities in their morphologies, such as “legs” or double-joints, and some of them are symmetric. This is possible because of the oscillations, which is a well-known property of GRN system.

We anticipate much future work on this model. First, we have to produce more robotic blocks and actually build the virtual robots into reality. The prototypes of the robotic blocks already exist but the building of a whole robot will likely require some modifications and re-engineering of the robotic blocks (actuators, motors, dovetail latches, etc.) as

well as cause constraints to be added to the physics simulator (motor strength, block friction, etc.).

The joints' controllers are also something we have to work on. They only consist of oscillators moving the joints clockwise and then counterclockwise. Two main solutions are currently considered to improve the controller. First, we can imagine that the same GRN also controls the joints when the growth of the robot is finished. This method has already been applied with success by Schramm in [22]. The second solution could be to plug a neural network or an equivalent method to the joints. Here, two solutions are possible: at the first hand, the neural network and the developmental model can be evolved at the same time or, at the second hand, it can be evolved after the evolution of the morphology.

In order to generate efficient robots, their genomes will have to be evolved by an evolutionary algorithm. The fitness function continues to be the main issue, as the field needs to move beyond linear movement in a simulated environment. To have other kinds of morphologies, we are considering including complexity, symmetry and regularity measures to the fitness function as well as utilizing multi-objective techniques for the co-evolution of body and behavior.

## 7. REFERENCES

- [1] W. Banzhaf. Artificial regulatory networks and genetic programming. *Genetic Programming Theory and Practice*, pages 43–62, 2003.
- [2] P. Bézier. *Numerical control; mathematics and applications*. John Wiley & Sons, 1972.
- [3] J. Bongard and R. Pfeifer. Evolving complete agents using artificial ontogeny. *Morpho-functional machines: The new species (designing embodied intelligence)*, pages 237–258, 2003.
- [4] A. Chavoya and Y. Duthen. A cell pattern generation model based on an extended artificial regulatory network. *Biosystems*, 94(1-2):95–101, 2008.
- [5] D. Christensen. Experiments on Fault-Tolerant Self-Reconfiguration and Emergent Self-Repair. In *IEEE Symposium on Artificial Life (ALIFE'07)*, pages 355–361, 2007.
- [6] J. Clune, B. E. Beckmann, P. K. McKinley, and C. Ofria. Investigating whether hyperneat produces modular neural networks. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 635–642. ACM, 2010.
- [7] S. Cussat-Blanc, N. Bredeche, H. Luga, Y. Duthen, and M. Schoenauer. Artificial gene regulatory networks and spatial computation: A case study. In *Proceedings of the European Conference on Artificial Life (ECAL'11)*. MIT Press, Cambridge, MA, 2011.
- [8] E. H. Davidson. *The regulatory genome: gene regulatory networks in development and evolution*. Academic Press, 2006.
- [9] R. Dawkins. *The blind watchmaker*. Longman Scientific & Technical, 1986.
- [10] R. Doursat. Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. *Organic Computing*, pages 167–200, 2008.
- [11] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213. MIT Press, 1997.
- [12] A. Fontana. Devo co-evolution of shape and metabolism for an artificial organ. *Artificial Life XII*, H. Fellermann et al., Eds. Boston: MIT Press, pages 16–23, 2010.
- [13] H. Hamann, J. Stradner, T. Schmickl, and K. Crailsheim. A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.
- [14] G. Hornby, H. Lipson, and J. Pollack. Generative representations for the automated design of modular physical robots. *Robotics and Automation, IEEE Transactions on*, 19(4):703–719, 2003.
- [15] M. Joachimczak and B. Wróbel. Evo-devo in silico: a model of a gene network regulating multicellular development in 3d space with artificial physics. In *Proceedings of the 11th International Conference on Artificial Life*, pages 297–304. MIT Press, 2008.
- [16] M. Joachimczak and B. Wróbel. Evolving Gene Regulatory Networks for Real Time Control of Foraging Behaviours. In *Proceedings of the 12th International Conference on Artificial Life*, 2010.
- [17] J. Knabe, M. Schilstra, and C. Nehaniv. Evolution and morphogenesis of differentiated multicellular organisms: autonomously generated diffusion gradients for positional information. *Artificial Life XI*, 11:321, 2008.
- [18] H. Lipson and J. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.
- [19] M. Nicolau, M. Schoenauer, and W. Banzhaf. Evolving genes to balance a pole. *Genetic Programming*, pages 196–207, 2010.
- [20] T. Reil. Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. *Advances in Artificial Life*, 1999.
- [21] C. Reynolds. Interactive evolution of camouflage. *Artificial Life*, 17(2):123–136, 2011.
- [22] L. Schramm, Y. Jin, and B. Sendhoff. Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. *Advances in Artificial Life: Darwin Meets von Neumann*, pages 27–34, 2011.
- [23] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Conference on Human Factors in Computing Systems*, CHI '08, pages 1759–1768. ACM, 2008.
- [24] K. Sims. Evolving 3d morphology and behavior by competition. *Artificial Life IV*, pages 28–39, 1994.
- [25] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.
- [26] V. Zykov, W. Phelps, N. Lassabe, and H. Lipson. Molecubes Extended: Diversifying Capabilities of Open-Source Modular Robotics. In *International Conference on Intelligent RObots and Systems, Self-Reconfigurable Robots Workshop (IROS 08)*, 2008.