Exploiting Regularity Without Development

Kenneth O. Stanley
School of Electrical Engineering and Computer Science
The University of Central Florida, Orlando, FL 32816 USA
kstanley@cs.ucf.edu

Abstract

A major challenge in evolutionary computation is to find the right level of abstraction of biological development to capture its essential properties without introducing unnecessary inefficiencies. In this paper, a novel abstraction of natural development, called Compositional Pattern Producing Networks (CPPNs), is proposed. Unlike most computational abstractions of natural development, CPPNs do not include a developmental phase, differentiating them from developmental encodings. Instead of development, CPPNs employ compositions of functions derived from gradient patterns present in developing natural organisms. In this paper, a variant of the NeuroEvolution of Augmenting Topologies (NEAT) method, called CPPN-NEAT, evolves increasingly complex CPPNs, producing patterns with strikingly natural characteristics.

Introduction

The discovery of systems as complex as humans is only possible through extraordinarily efficient encoding. In general, the more dimensions there are in the search space, the more difficult the search. Searching through thousands of dimensions is prohibitive; trillions, on the other hand, is likely intractable. Yet there are trillions of connections in the human brain (Zigmond *et al.*, 1999). The only way to discover such high complexity may be through a mapping between genotype and phenotype that translates few dimensions into many, i.e. through an *indirect encoding*.

A most promising form of indirect encoding is *developmental encoding*, which is motivated directly from biology (Bentley & Kumar, 1999; Hornby & Pollack, 2002; Stanley & Miikkulainen, 2003). In biological development, DNA maps to the mature phenotype through a process of growth that builds the phenotype over time. Development facilitates the reuse of genes because the same gene can be activated at any location and any time during the development process. Thus a small set of genes can encode a much larger set of structural components.

This observation has inspired an active field of research in artificial developmental encodings (Bentley & Kumar, 1999; Bongard, 2002; Dellaert & Beer, 1996; Federici, 2004; Gruau, Whitley, & Pyeatt, 1996; Hornby & Pollack, 2002; Miller, 2004). The aim is to find the right abstraction of natural development for a computer running an evolutionary algorithm, so that it can begin to discover complexity on a natural scale. Abstractions range from low-level cell chemistry simulations to high-level grammatical rewrite systems (see Stanley & Miikkulainen, 2003 for a review). Yet none so far have come close to discovering the level of complexity seen in nature.

This paper proposes a novel abstraction of natural development that breaks a strong tradition in developmental encoding research: The proposed abstraction captures the essential properties of natural developmental encoding without implementing a process of development. The fundamental insight behind this new encoding, called Compositional Pattern Producing Networks (CPPNs), is that it is possible to directly *describe* the structural relationships that result from a process of development without simulating the process itself. Instead, the description is encoded through a composition of functions, each of which is based on observed gradient patterns in natural embryos.

Because CPPNs are structurally similar to artificial neural networks, they can right away take advantage of existing effective methods for evolving neural networks. In particular, the Neuroevolution of Augmenting Topologies (NEAT) method evolves increasingly complex neural networks over generations (Stanley & Miikkulainen, 2002, 2004). In this paper, with only slight adjustment, NEAT is modified to create CPPN-NEAT, which evolves increasingly complex CPPNs.

Experimental results obtained through an interactive evolutionary process show striking examples of many of the fundamental structural motifs and elaborations of nature being created by CPPN-NEAT. Symmetry, reuse, reuse with variation, preservation of regularities, and elaboration of existing regularities all are demonstrated and capitalized on by CPPN-NEAT, establishing it as a promising new abstraction of natural developmental encoding, and one that is able to evolve increasingly complex patterns.

The next section provides background on the role of development in both natural evolution and artificial evolutionary algorithms. The CPPN approach is then explained, followed by how CPPN-NEAT evolves increasingly complex patterns. The final sections present and discuss experimental results.

Background

This section introduces important concepts in developmental encoding and the special connection between evolution and development.

Artificial Developmental Encodings

The apparent connection between development and complexity in biology has inspired considerable research into artificial developmental encoding in recent years (Astor & Adami, 2000; Bentley & Kumar, 1999; Bongard, 2002; Dellaert & Beer, 1996; Federici, 2004; Gruau, Whitley, & Pyeatt, 1996; Hornby & Pollack, 2002; Komosinski & Rotaru-Varga, 2001; Lindenmayer, 1968; Miller, 2004; Sims, 1994; Turing, 1952). Just as a biological embryo starts from a single cell and through a series of genetic instructions achieves structures of astronomical complexity, artificial developmental encodings strive to map an artificial genome to a comparatively more complex phenotype through a series of growth steps.

Development makes such representational efficiency possible by allowing genes to be reused in the process of assembling the phenotype. Artificial developmental encodings attempt to exploit this capability in a similar way to nature. However, because computers differ from the natural world and because it is not necessary to simulate every facet of a biological process in order to capture its essential properties, finding the right level of abstraction remains an active research area. Low-level abstractions often evolve genes that produce simulated proteins that diffuse and react as the phenotype develops. The networks formed by genes that send and receive signals through their protein products are called *genetic regulatory networks* (GRNs; Bongard, 2002; Dellaert & Beer, 1996). In contrast, higher-level abstractions iteratively apply replacement rules to grow a final structure from a single starting symbol (Hornby & Pollack, 2002).

Developmental approaches attempt to capture the essential properties of development at the right level of abstraction. However, no approach has achieved complexity close to that seen in nature. Thus, the search for powerful developmental encodings continues (Stanley & Miikkulainen, 2003).

Complexification

Development and evolution are inextricably linked. The process of development provides a framework within which evolution can elaborate and increase the complexity of its products by adding new refinements and divisions during embryogenesis. The added instructions that execute these elaborations are the products of *new genes* added over the course of evolution (Martin, 1999; Watson *et al.*, 1987). This process of *complexification* allows evolution to discover more complex phenotypes than would be possible through optimizing a fixed set of genes.

In any search process, the more dimensions there are in a solution, the harder it is to discover. In other words, complex solutions are more difficult to evolve than simple ones. Yet natural evolution can nevertheless discover organisms with tens of thousands of genes because it does not start searching in a space of the same complexity as the final solution. By occasionally adding genes, evolution can perform a complexifying function over and above optimization. This process of gradually adding new genes has been confirmed in natural evolution (Martin, 1999; Watson *et al.*, 1987) and shown to improve adaptation (Altenberg, 1994).

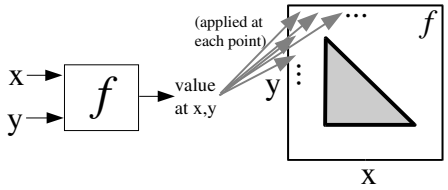


Figure 1: **A Function Produces a Phenotype.** The function f takes arguments x and y, which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of f at that coordinate, the result is a pattern, which can be viewed as a phenotype whose genotype is f. In this example, f produces a triangular phenotype.

Through complexification, major animal body plans can be established early in evolution and then elaborated and refined as new genes are added (Martin, 1999). General conventions that are encapsulated in only a few genes, such as bilateral symmetry, can be captured in early organisms and later elaborated on with increasingly intricate limbs and segments. Thus, it is important that any encoding abstracted from biology be able to complexify in addition to efficiently encode regularities.

The next section presents an abstraction of development with a novel twist.

Patterns Without Development

A strong assumption behind much recent work in developmental encoding is that indeed *development* in a simulation is an abstraction that captures the essential properties of development in nature. Although this assumption is powerfully intuitive, in fact development need not be part of such an abstraction.

This section introduces a method for representing complex systems that does not employ a developmental process yet still captures its essential properties. The central insight is that complex natural patterns can be generated by a process more fundamental than development, capturing its essential properties, yet skipping its process of growth.

Functions and Regularities

A phenotype can be described as a function of n dimensions, where n is the number dimensions in the physical world.[1] For each coordinate, the presence or absence of a point, or a continuous value representing its level of expression, is an output of the function that describes the phenotype. Figure 1 shows how a two-dimensional phenotype can be generated by a function of two parameters. This view of the phenotype is significant because it implies that its regularities and symmetries can be captured through a purely functional, non-developmental description. A mathematical abstraction of each stage of development then can be represented explicitly inside the function.

For example, a significant task at early stages of development in natural embryogeny is to define a *coordinate*

[1]The phenotype need not be a physical morphology; physical space in this section is a conceptual device for describing an abstract configuration of any type.

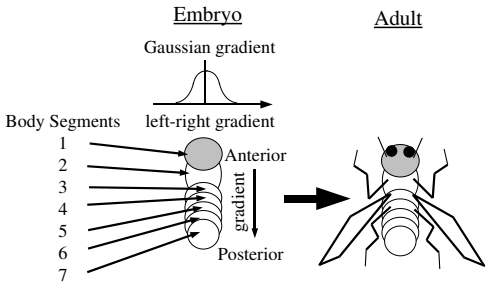


Figure 2: **Gradients Define the Body Plan.** A simplified insect model with discrete body segments and bilateral symmetry is depicted. Gradients along the major axes define the the overall body plan. The anterior-posterior gradient allows segments to situate in the embryo along that axis, each one destined to grow its own associated body parts such as wings or legs. The Gaussian gradient, a function of the original left-right axis gradient, is distributed symmetrically about that axis, allowing bilateral symmetry to develop from it.

frame, i.e. a set of virtual coordinate axes, upon which future stages of development will be based (Meinhardt, 1982; Raff, 1996). The simplest and most basic of these coordinate frames are the main axes of the body, which are defined at the very beginning of development, inside the egg itself (Raff, 1996, p.188). These axes include the *anterior-posterior* axis (i.e. head to feet), and the *dorsal-ventral* axis (i.e. back and front). Just like the x and y axes in a simple Cartesian coordinate system, these two main axes are orthogonal and roughly linear. The initial axes are determined through a cascade of gene interactions defined by a GRN, that is, a signaling pattern that develops over time culminates in a chemical gradient along each initial axis.

Once the initial axes are defined, new coordinate frames for subregions of the body can be appropriately spatially situated (Meinhardt, 1982). In this way, gradients that define earlier coordinate frames are *inputs* to future cascades that produce more localized coordinate frames. For example, the body plan of the *Drosophila* fly includes a series of segments, each with its own internal coordinate axes. Each segment is the origin of a specific body part or region, such as legs, wings, and the head. Importantly, the internal coordinate frame of each segment must be determined by where that segment lies along the anterior-posterior axis of the complete body. That is, higher-level coordinate frames are determined by lower-level frames. In fact, the genes that determine the segmentation of the body by defining their coordinate frames (called *HOX genes*) are activated from the initial axial gradients (Curtis, Apfeld, & Lehmann, 1995; Lall & Patel, 2001). Figure 2 shows the anterior-posterior axis of a simplified insect model, and segments along its axis.

Interestingly, although the coordinate frames in natural development are defined through GRNs that unfold over time, in principle the same axes can be described functionally instead, simply by composing the functions that describe them. Through function composition, biologically plausible new coordinate frames can be derived from preexisting ones without a developmental process. Consider the *left-right* axis, along which bilateral symmetry is present in many organisms. Coordinates along this axis can be defined simply as $f(x) = x$. However, in order to become bilaterally symmetric along this axis, a developing embryo must organize a gradient that peaks at the center and diminishes towards both the left and right sides. The organizing process to achieve such a gradient may be complex, yet functionally it can be described simply as a Gaussian: $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ (figure 2).

Just as a symmetric function can yield bilateral symmetry, segmentation can be represented through periodic functions. The function $j(y) = \sin(y)$ effectively assigns a repeating coordinate frame (i.e. a set of segments) laid along the anterior-posterior axis. In the case of sine, each coordinate frame is itself symmetric about its peak value. Repeating asymmetric frames can be created analogously using the modulus function, as in $k(y) = y \bmod 1$.

Coordinate frames created through a developmental process interact with each other and organize in parallel to produce complex patterns with regularities. In the same way, functionally-represented frames can be composed to create complex regularities. For example, bilateral symmetry and segmentation along the left-right axis can produce two sets of segments with opposite polarities. This phenomenon is achieved simply by feeding the output of a symmetric function into a periodic function. This simple composition creates a new set of coordinate frames that can now be used for further refinement, all without a process of development.

Thus, the fundamental phenomena of development are in fact analogous to a series of function compositions that each in turn create a new, more refined coordinate frame. Eventually, these frames can combine to form patterns much like those observed in nature. This analogy raises the intriguing possibility that function composition is the right level of abstraction for development, even though function composition is not intrinsically a developmental process. The order in which functions are composed is an abstraction for the chronology of events over the course of development through time.

The next section explains how compositions of functions can be evolved in a way that maintains the analogy with development: Major body-plan features can be established first and then refined in future generations, just as happens over the course of natural evolution.

Evolving Compositional Pattern Producing Networks

A natural way to represent a composition of many functions is as a connected graph (figure 3). The initial coordinate axes (i.e. the most basic coordinate frame) can be provided as inputs to the graph. The next level of nodes are descriptions of the first stages of development, such as establishing bilateral symmetry. Higher level nodes then establish increasingly refined coordinate frames. The final outputs are thus informed by each transformation that takes place before them. In this way, the entire graph is like a diagram of the sequence of steps that happen over a developmental chronology.

Interestingly, a function composition graph of this type is very similar to an artificial neural network with arbitrary

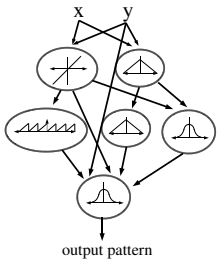


Figure 3: **Composition of Functions as a Graph.** The graph determines which functions connect to which. The connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. If multiple connections feed into the same function then the downstream function takes the sum of their weighted outputs. Note that the topology is unconstrained and can represent any possible relationships (including recurrent).

topology, contrasting only in the set of activation functions. The analogy between a function composition graph and an artificial neural network (ANN) is so strong, in fact, that it is tempting to equate the two. However, while they are clearly related, using the term *artificial neural network* would be misleading in the context of this research because artificial neural networks were so named in order to establish a metaphor with a *different* biological phenomenon, i.e. the brain. The terminology should avoid making the implication that biological, thinking brains are in effect the same as developing embryos. Therefore, this paper uses the term *Compositional Pattern Producing Network* (CPPN) to refer to graph constructs that describe compositions of functions intended to produce regular patterns.

It is fortuitous that CPPNs and ANNs are virtually the same from a structural perspective because an effective method already exists to evolve the topology and weights of ANNs that is also designed to produce the process of complexification discussed earlier: The NeuroEvolution of Augmenting Topologies method (NEAT; Stanley & Miikkulainen, 2002, 2004) evolves increasingly complex ANNs over generations, and addresses the challenges that come with evolving a population of diverse complexifying topologies.

Although NEAT was originally designed to evolve ANNs, it requires only trivial modification to evolve CPPNs, since they are so similar. The next section reviews the NEAT method, followed by a description of how NEAT is modified to produce CPPN-NEAT.

NeuroEvolution of Augmenting Topologies (NEAT)

This section briefly reviews the NEAT method. See also Stanley & Miikkulainen (2002, 2004) for detailed descriptions of original NEAT. The NEAT method was originally developed to solve difficult control and sequential decision tasks. The neural networks control agents that select actions based on their sensory inputs. While previous methods that evolved neural networks, i.e. *neuroevolution* methods, evolved either fixed topology networks (Gomez & Miikkulainen, 1999; Saravanan & Fogel, 1995), or arbitrary random-topology networks (Angeline, Saunders, & Pollack, 1993; Gruau, Whitley, & Pyeatt, 1996; Yao, 1999), NEAT is the first to begin evolution with a population of small, simple networks and *complexify* the network topology over generations, leading to increasingly sophisticated behavior.

Before describing the CPPN extension, let us review the three key ideas on which the basic NEAT method is based. First, in order to allow network structures to increase in complexity over generations, a method is needed to keep track of which gene is which. Otherwise, it is not clear in later generations which individual is compatible with which, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique *historical marking* to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis. That way, genomes of different organizations and sizes stay compatible throughout evolution, solving the previously open problem of matching different topologies (Radcliffe, 1993) in an evolving population.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, other systems that evolve network topologies and weights begin evolution with a population of random topologies (Gruau, Whitley, & Pyeatt, 1996; Yao, 1999). In contrast, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial random weights. Speciation protects new innovations, allowing diverse topologies to gradually accumulate over evolution, and thus diverse and complex phenotype patterns to be represented. Thus, NEAT can start minimally, and grow the necessary structure over generations.

CPPN-NEAT

CPPN-NEAT is an extension of NEAT that allows it to evolve CPPNs. While networks in original NEAT only include hidden nodes with sigmoid functions, node genes in CPPN-NEAT include a field for specifying the activation function. When a new node is created, it is assigned a random activation function from a canonical set (e.g. including Gaussian, sigmoid, and periodic functions). Furthermore, the compatibility distance function that is utilized to determine whether two networks belong in the same species includes an additional argument that counts how many activation functions differ between the two individuals. This allows speciation to take activation function differences into account. Because CPPN-NEAT is an enhancement of an effective preexisting method, it provides a reliable platform for studying the evolution of increasingly complex forms. While other graph-structure evolution methods such as genetic programming (Koza, 1992; Miller & Thomson, 2000) do not explicitly implement complexification, NEAT can evolve increasingly complex CPPNs, mapping to a succession of increasingly elaborate phenotypic body plans. The

next section explains why CPPNs can potentially save significant computational resources compared to developmental encodings.

Computational Advantages of CPPNs

The CPPN abstraction affords several computational advantages over traditional developmental encodings:

- CPPNs map to phenotypes without simulating a process of development.
- CPPNs store patterns at infinite resolution since they are mathematical structures.
- Users can bias the kinds of patterns produced by CPPNs by inputting user-defined coordinate frames in addition to x and y.
- Unlike other developmental encodings (Andersen *et al.*, 2005; Miller, 2004; Roggen & Federici, 2004), CPPNs can always regenerate damaged structure *perfectly* because each coordinate is independently computed.

The next section demonstrates through several experiments that CPPN-NEAT produces phenotypes that exhibit strikingly natural characteristics.

Experiments

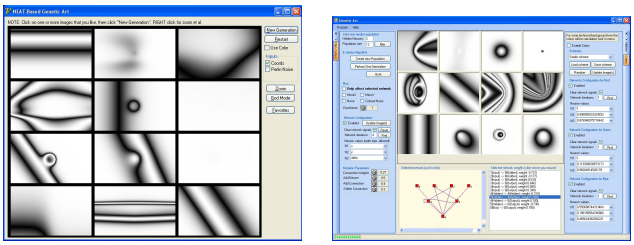
Developmental encodings are are traditionally evaluated for their performance on benchmark tasks (Bentley & Kumar, 1999; Bongard, 2002; Federici, 2004; Gruau, Whitley, & Pyeatt, 1996; Hornby & Pollack, 2002) such as evolving to a target or comparing with direct encoding. For three reasons, this paper takes a different experimental approach. First, the primary aim is to demonstrate that CPPNs abstract the essential properties of natural development even though they do not themselves employ a process of development. Developmental encodings, on the other hand, explicitly implement a process of development and therefore researchers generally need not attempt to establish such a link.

Second, natural organisms do not appear to be able to take on any arbitrary form. For example, no macroscopic organism has any appendage analogous to a wheel, even though the wheel is a fundamental engineering structure. Therefore, rather than focus on whether a CPPN can represent a certain *class* of object, the following experiments focus on its ability to establish and exploit regularities in general.

Third, natural evolution does not evolve to specified targets. Rather, it exploits established regularities in creative ways. Thus, evaluating against a specific target or task does not necessarily reveal CPPNs' ability to similarly establish and then exploit regularities. The major contribution of this paper is thus to establish that a new kind of abstraction of development is admissible.

CPPN-NEAT-based Interactive Pattern Evolution

In order to explore both the patterns that CPPNs produce and the way evolution varies those patterns, an interactive evolutionary approach was taken. In interactive evolution (Takagi, 2001), the user performs the selection step that determines which individuals will be parents of the next generation. By introducing this interactive role for the experimenter, it is possible to intentionally explore the space of generated patterns in ways that would otherwise be impossible.



(a) DNGA (b) SNGA

Figure 4: **Interactive Evolution Experimental Interfaces.** The two platforms shown above allow an experimenter to evolve CPPN-generated patterns interactively. The set of patterns shown are the current population, and the experimenter can select the parents of the next generation (i.e. fitnesses are not assigned). Both platforms were used in the experiments in this section. Random initial generations are shown for each. (a) DelphiNEAT-based Genetic Art (DNGA) is by Mattias Fagerlund, and uses his DelphiNEAT platform to evolve CPPNs. (b) SharpNEAT-based Genetic Art (SNGA), which uses Colin Green's SharpNEAT platform, is by Holger Ferstl. Both platforms are freely available, including source code (see Acknowledgements). The interactive interface makes it possible to explore the way evolution establishes and varies regularities, as opposed to attempting to solve a particular benchmark.

The interactive approach works by visually presenting the experimenter with all the patterns in the current generation (figure 4 introduces the two programs used in the experiments). The experimenter then picks the parents of the next generation by clicking on them. Thus, the selection step is entirely determined by the human experimenter. The aim is to intentionally search for certain types of patterns in order to ascertain whether they can be elaborated and maintained in the same ways as in natural evolution.

This approach resembles what are commonly called *genetic art* programs, which are traditionally used to generate aesthetically pleasing abstract pictures (Sims, 1991; Takagi, 2001). Genetic art programs closely follow the original *Blind Watchmaker* idea from Dawkins (1986), in which simple genetically-encoded patterns are evolved through an interactive interface. While the interactive genetic art concept is often considered an entertainment application or artistic tool, it is also a rich an underutilized experimental medium for exploring the characteristics of various encodings.

In this paper, this procedure is used to analyze how CPPN-NEAT encodes regularities. Note that the aim is not to suggest that interactive evolution is a necessary component of CPPN-NEAT, but rather to use interactivity to explore its encoding properties experimentally. Both programs begin with a population of randomly initialized networks with one or two hidden nodes. The user picks one or two parents of the next generation, which is then displayed. Gaussian and sigmoid functions are the main primitives composed by both programs to produce phenotypes in a two-dimensional Cartesian space. Furthermore, special coordinate frames are provided at the inputs in addition to x and y. DNGA (figure 4a) provides the CPPN with a *distance from the center* coordinate (d), while SNGA (figure 4b) allows the experimenter to provide any arbitrary coordinates frames, includ-

ing sine waves of varying period. Thus, these programs allow the user to both explore and observe explicit examples of how CPPNs compose functions and derive patterns from provided coordinate frames.

Each new generation is created by mutating and mating the chosen parents. CPPNs complexify, with mutations occasionally adding more functions and connections in the CPPN. In SNGA, the probability of adding a neuron to a child genome is 0.5 and the probability of adding a connection is 0.4. In DNGA, the probabilities are 0.06 for both. Extensive experimentation verified that both systems consistently evolve complex regular phenotypes, indicating that interactive CPPN-NEAT is robust to a wide range of these parameters. In both systems, function outputs range between $[-1, 1]$. However, ink level is darker the closer the output is to zero. Therefore, an output of either -1 or 1 produces white. This approach was found to produce the most interesting patterns in early generations.

The following experiments evaluate evolved patterns against the characteristics of natural biological patterns.

Elaboration of Regularity

During a period of evolution known as the *Cambrian Explosion*, when numerous new animal body plans were arising, the *bilateria* appeared. The simplest of the bilateria were a kind of flatworm (Raff, 1996, pp. 38-41). This original body plan has been elaborated for hundreds of millions of years, all the while preserving its fundamental design. Can a CPPN analogously capture a bilateral body plan and maintain and elaborate on bilateral symmetry for generations?

To address this question, a spaceship morphology was interactively evolved with DNGA. The first step was to determine whether the CPPN can discover and establish bilateral symmetry. As figure 4 shows, symmetry is by no means ubiquitous in random initial populations. However, in 20 separate runs, bilateral symmetry arises within the first three generations 100% of the time. Evolution quickly finds a CPPN structure that defines symmetry across a single axis, thereby creating a framework for bilateral symmetry.

The first individual with a recognizable spaceship-like pattern is shown in figure 5a. It has seventeen connections and four hidden functions that together produce a tail, wings, and a nose. These features persist throughout evolution. A succession of descendants that elaborate on this initial theme is shown in figure 5. What is striking about these elaborations is how they genuinely preserve the underlying regularities but also become more complex as their respective CPPNs complexify.

Figure 5h shows a remarkable and surprising variation on the tail in which evolution invents tail fins. The tail fins respect the underlying symmetry and at the same time introduce entirely new and elegant features. This kind of elaboration suggests that the complexifying CPPNs in effect implement the fundamental process of continual elaboration of form. This level of meaningful elaboration is rarely seen in artificial developmental encodings.

Developmental encoding is often cited for its potential to discover and elaborate on useful modules (Bentley & Kumar, 1999; De Jong & Thierens, 2004; Hornby & Pollack, 2002). The spaceships in figure 5 exhibit distinct modules that respect a regular structure. For example, the wing concept still conforms to the same general organization and structure even as it undergoes significant changes: Between figures 5h and 5i, the wing is varied significantly while preserving the overall spaceship structure. A similarly significant elaboration, in this case of the tail fin module, happens between figures 5i and 5j. Thus, a composition of functions can implicitly encode discrete modular structures, each with its own ability to vary in meaningful ways.

Not only can the CPPN encoding maintain and elaborate on existing regularities, but it can create new ones. Evolution creates a novel "cockpit" in figure 5j. Looking closely, it can be seen that the cockpit is *co-opted* from line decorations on the tail in figure 5i. Natural evolution also frequently innovatives through *exaptation*, i.e. by co-opting an existing trait for a new purpose (True & Carroll, 2002).

Whereas the initial spaceship CPPN contains four hidden functions and 16 connections (figure 5a), the final, significantly more complex spaceship CPPN includes nine hidden functions and 38 connections (figure 5k). Thus, the size of the genotype slightly more than doubled due to CPPN-NEAT's complexification. The added structure allows CPPN-NEAT to lay new features on top of established coordinate frames. It is also notable that only 38 connections code for the complex pattern shown in figure 5k, indicating that CPPNs have the potential to reuse information and represent patterns in a way that enhances representational efficiency, just as developmental encodings do.

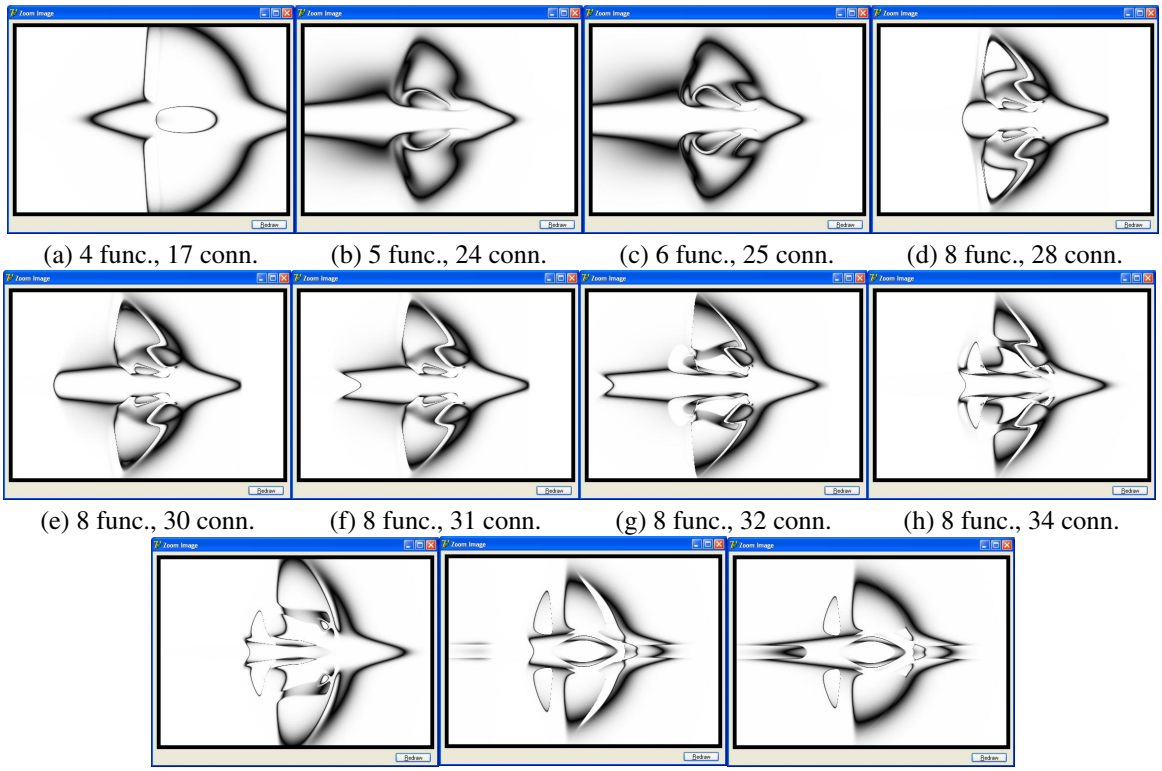
The succession of spaceship phenotypes exhibits several of the fundamental characteristics of developmental encoding in nature (Stanley, 2006). Symmetry is established early and then elaborated. Regularity is preserved and expanded. Mutations respect the underlying body plan as it elaborates through complexification. The next section addresses whether CPPNs can also create regularities that vary.

Repetition with Variation

Reuse through repetition with variation is crucial in natural organisms. Fingers and toes share fundamental regularities yet also differ in size and proportion. Cortical columns in the brain also share structure yet are not identical (Zigmond *et al.*, 1999). If every slight variation on a theme required an entirely different set of genes, the representational space of many organisms would be prohibitively complex. Can a CPPN capture this important kind of regularity?

To answer this question, instead of inputting the x,y, and d coordinates into the CPPN in SNGA as normal, $\sin(10x)$, $\sin(10y)$, and d were input instead. By inputting sine waves, the same coordinate frame repeats multiple times. However, the d coordinate, i.e. *distance from center*, does not repeat. Therefore, functions higher in the CPPN can utilize both the repeating frame of reference and the absolute reference *at the same time*.

Figure 6a shows a repeating pattern generated by the CPPN with sine inputs. The period of the sine wave is apparent in the image, but more remarkable is that the design at the center of each repeated instance is slightly different (figure 6b). This example demonstrates how a CPPN can

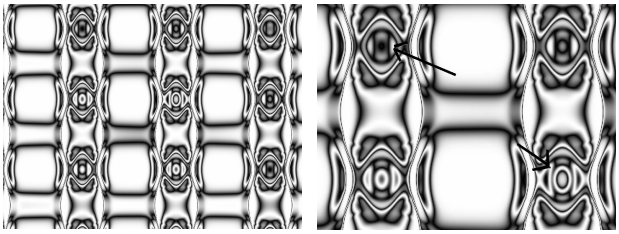


(a) 4 func., 17 conn. (b) 5 func., 24 conn. (c) 6 func., 25 conn. (d) 8 func., 28 conn.

(e) 8 func., 30 conn. (f) 8 func., 31 conn. (g) 8 func., 32 conn. (h) 8 func., 34 conn.

(i) 8 func., 36 conn. (j) 9 func., 36 conn. (k) 9 func., 38 conn.

Figure 5: **Sequence of Descendant Spaceship Patterns.** The chronological sequence (a)–(k) displays successive progeny evolved with interactive CPPN-NEAT. The number of hidden functions and connections in the generating CPPN is shown below each pattern. The sequence exhibits a remarkably natural continual elaboration of regular form.



(a) Complete Pattern (b) Zoomed In

Figure 6: **Repetition with Variation.** The pattern in (b) magnifies the upper-left portion of (a). Arrows in (b) point to significant differences between motifs that are otherwise similar. By mixing inputs from separate coordinate frames, one repeating and one not, a CPPN can generate a large variety of patterns with this property. This phenomenon is similar to the interaction of two chemical gradients in a developing embryo, one periodic and one not.

generate repetition with variation, a phenomenon commonly attributed to natural developmental encodings.

Discussion and Future Work

Even without development, CPPNs exhibit many of the the essential properties of natural development, including symmetry, repetition, repetition with variation, elaboration of existing regularities through complexification, and preservation of regularities (Stanley, 2006). CPPNs complexify in a way analogous to natural evolution. For these reasons, CPPNs are admissible into the class of valid abstractions of biological developmental encoding. Establishing this link is the main contribution of this paper. In doing so, an entirely new and promising research direction is revealed.

If CPPNs were only used to draw pictures, their utility would be limited. However, the patterns in this paper have a greater significance. An encoding that produces a pattern in space can produce such patterns for any objective in any substrate. If the substrate is neural, the CPPN can describe the neural pathways, including all the symmetries, repeating patterns, and elegant group-wise connectivity. If the substrate is physical, the CPPN can encode the building blocks that compose the body, which can then be run in simulation, and later manufactured in the real world. Patterns underly almost all natural and artificial structures; thus a general pattern-generating mechanism can represent any of them and more. Future research will determine the best way to translate a spatial pattern into a particular substrate.

Conclusion

Research in developmental encoding to this date can be characterized as a search for the right abstraction. Computational abstractions of natural development generally have included, not surprisingly, a process of development over time. In this paper, a novel abstraction was proposed that breaks this tradition by counterintuitively not including development. Nevertheless, this abstraction, called Compositional Pattern

Producing Networks (CPPNs), still captures the essential properties of natural developmental encoding. A series of interactive evolutionary experiments with CPPNs demonstrated that they indeed produce patterns and sequences of patterns with properties analogous to those seen in natural evolution and development. Furthermore, an algorithm for evolving increasingly complex CPPNs, called CPPN-NEAT, was introduced. Future research will focus on converting the patterns output by CPPNs into important substrates such as large-scale neural networks, physical morphologies, and building architectures.

Acknowledgments

Special thanks to Mattias Fagerlund for creating the first NEAT-based genetic art program based on his DelphiNEAT implementation, to Holger Ferstl for creating the second NEAT-based genetic art program based on SharpNEAT, and to Colin Green for creating SharpNEAT. All the software and source code used in this paper, including DNGA, SNGA, DelphiNEAT, and SharpNEAT, is available through `http://www.cs.ucf.edu/~kstanley`.

References

Altenberg, L. 1994. Evolving better representations through selective genome growth. In *Proc. of the IEEE World Congress on Comp. Intelligence*, 182–187. Piscataway, NJ: IEEE Press.

Andersen, T.; Otter, C.; Petschulat, C.; Eoff, U.; Menten, T.; Davis, R.; and Crowley, B. 2005. A biologically-derived approach to tissue modeling. In et al, J. W., ed., *Technology and Informatics*. Amsterdam: IOS Press. 15–21.

Angeline, P. J.; Saunders, G. M.; and Pollack, J. B. 1993. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* 5:54–65.

Astor, J., and Adami, C. 2000. A developmental model for the evolution of artificial neural networks. *Artificial Life* 6(3):189–218.

Bentley, P. J., and Kumar, S. 1999. The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-1999)*, 35–43. San Francisco: Kaufmann.

Bongard, J. C. 2002. Evolving modular genetic regulatory networks. In *Proc. of the 2002 Congress on Evolutionary Comp.*.

Curtis, D.; Apfeld, J.; and Lehmann, R. 1995. Nanos is an evolutionarily conserved organizer of anterior-posterior polarity. *Development* 121:1899–1910.

Dawkins, R. 1986. *The Blind Watchmaker.* Essex, U.K.: Longman.

De Jong, E. D., and Thierens, D. 2004. Exploiting modularity, hierarchy, and repetition in variable-length problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 1030–1040. Berlin: Springer Verlag.

Dellaert, F., and Beer, R. 1996. A developmental model for the evolution of complete autonomous agents. In Maes, P.; Mataric, M. J.; Meyer, J.-A.; Pollack, J.; and Wilson, S. W., eds., *From Animals to Animats 4: Proc. of the Fourth International Conf. on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

Federici, D. 2004. Evolving a neurocontroller through a process of embryogeny. In Schaal, S.; Ijspeert, A. J.; Billard, A.; Vijayakumar, S.; Hallam, J.; and Jean-Arcady., eds., *Proc. of the Eighth International Conf. on Simulation and Adaptive Behavior (SAB-2004)*, 373–384. Cambridge, MA: MIT Press.

Gomez, F., and Miikkulainen, R. 1999. Solving non-Markovian control tasks with neuroevolution. In *Proc. of the 16th Int. Joint Conf. on Artif. Int.*, 1356–1361. San Francisco: Kaufmann.

Gruau, F.; Whitley, D.; and Pyeatt, L. 1996. A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R.; Goldberg, D. E.; Fogel, D. B.; and Riolo, R. L., eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

Hornby, G. S., and Pollack, J. B. 2002. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life* 8(3).

Komosinski, M., and Rotaru-Varga, A. 2001. Comparison of different genotype encodings for simulated 3D agents. *Artificial Life* 7(4):395–418.

Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Lall, S., and Patel, N. 2001. Conservation and divergence in molecular mechanisms of axis formation. *Annual Review of Genetics* 35:407–447.

Lindenmayer, A. 1968. Mathematical models for cellular interaction in development parts I and II. *Journal of Theoretical Biology* 18:280–299 and 300–315.

Martin, A. P. 1999. Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist* 154(2):111–128.

Meinhardt, H. 1982. *Models of Biological Pattern Formation*. London: Academic Press.

Miller, J. F., and Thomson, P. 2000. Cartesian genetic programming. In *Proc. of the 3rd European Conf. on Genetic Programming publ. as Lect. Notes in Comp. Sci., Vol. 1802*, 121–132.

Miller, J. F. 2004. Evolving a self-repairing, self-regulating, French flag organism. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2004)*. Berlin: Springer Verlag.

Radcliffe, N. J. 1993. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications* 1(1):67–90.

Raff, R. A. 1996. *The Shape of Life: Genes, Development, and the Evolution of Animal Form*. Chicago: The U. of Chicago Press.

Roggen, D., and Federici, D. 2004. Multi-cellular development: Is there scalability and robustness to gain. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, 391–400.

Saravanan, N., and Fogel, D. B. 1995. Evolving neural control systems. *IEEE Expert* 23–27.

Sims, K. 1991. Artificial evolution for computer graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques (SIGGRAPH '91)*, 319–328. New York, NY: ACM Press.

Sims, K. 1994. Evolving 3D morphology and behavior by competition. In Brooks, R. A., and Maes, P., eds., *Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*. Cambridge, MA: MIT Press. 28–39.

Stanley, K. O., and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10:99–127.

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

Stanley, K. O., and Miikkulainen, R. 2004. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research* 21:63–100.

Stanley, K. O. 2006. Comparing artificial phenotypes with natural biological patterns. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program*. New York, NY: ACM Press.

Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE* 89(9):1275–1296.

True, J., and Carroll, S. 2002. Gene co-option in physiological and morphological evolution. *Annual Review of Cell and Developmtnal Biology* 18:53–80.

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

Watson, J. D.; Hopkins, N. H.; Roberts, J. W.; Steitz, J. A.; and Weiner, A. M. 1987. *Molecular Biology of the Gene*. Menlo Park, CA: The Benjamin Cummings Publishing Company, Inc., fourth edition.

Yao, X. 1999. Evolving artificial neural networks. *Proceedings of the IEEE* 87(9):1423–1447.

Zigmond, M. J.; Bloom, F. E.; Landis, S. C.; Roberts, J. L.; and Squire, L. R., eds. 1999. *Fundamental Neuroscience*. London: Academic Press.