A Novel Generative Encoding for Evolving Modular, Regular and Scalable Networks

Marcin Suchorzewski
Artificial Intelligence Laboratory
West Pomeranian University of Technology
msuchorzewski@wi.zut.edu.pl

Jeff Clune
Creative Machines Laboratory
Cornell University
jeffclune@cornell.edu

ABSTRACT

In this paper we introduce the Developmental Symbolic Encoding (DSE), a new generative encoding for evolving networks (e.g. neural or boolean). DSE combines elements of two powerful generative encodings, Cellular Encoding and HyperNEAT, in order to evolve networks that are modular, regular, scale-free, and scalable. Generating networks with these properties is important because they can enhance performance and evolvability. We test DSE's ability to generate scale-free and modular networks by explicitly rewarding these properties and seeing whether evolution can produce networks that possess them. We compare the networks DSE evolves to those of HyperNEAT. The results show that both encodings can produce scale-free networks, although DSE performs slightly, but significantly, better on this objective. DSE networks are far more modular than HyperNEAT networks. Both encodings produce regular networks. We further demonstrate that individual DSE genomes during development can scale up a network pattern to accommodate different numbers of inputs. We also compare DSE to HyperNEAT on a pattern recognition problem. DSE significantly outperforms HyperNEAT, suggesting that its potential lay not just in the properties of the networks it produces, but also because it can compete with leading encodings at solving challenging problems. These preliminary results imply that DSE is an interesting new encoding worthy of additional study. The results also raise questions about which network properties are more likely to be produced by different types of generative encodings.

Categories and Subject Descriptors:
I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms

Keywords: Generative and Developmental Representations, Neuroevolution, Indirect Encoding, Networks, Modularity, Regularity, Scale-free, Scalability

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Networks designed by humans and evolution have certain properties that are thought to enhance both their performance and adaptability, such as modularity, regularity, hierarchy, scalability, and being scale-free [12, 11, 1]. In this paper we introduce a new generative representation called the Developmental Symbolic Encoding that evolves networks that possess these properties. We first review the meaning of these terms before discussing them with respect to previous evolutionary algorithms.

Modularity is the encapsulation of function within mostly independently operating units, where the internal workings of the module tend to only affect the rest of the structure through the module's inputs and outputs [12, 11]. For example, a car wheel is a module because it is a separate functional unit and changing its design is unlikely to affect other car systems (e.g. the muffler). This functional independence enables the wheel design to be optimized independently from most other car modules. In networks, modules are clusters of densely connected nodes, where connectivity is high within the cluster and low to nodes outside the cluster [11, 12].

Regularity describes the compressibility of the network description, and often involves symmetries and the repetitions of modules [12]. Regularity and modularity are distinct concepts. A unicycle has one wheel module (modularity without regularity), whereas a bicycle's repetition of the wheel module is a form of regularity. Hierarchy is the recursive composition of lower-level modules [12].

Scalability in evolved networks refers to the ability to produce effective solutions at different networks sizes. The concept applies both to the idea of being able to evolve large, but fixed-size networks [17], or to a single genome that can produce networks of different sizes as needed [6, 16, 5]. The latter capability is important for transferring knowledge to related problems of different complexity, and is a main motivation for DSE. Scalability can be enhanced by other properties, such as modularity, regularity and hierarchy [12].

Scale-free networks have a few nodes with many connections, and many nodes with few connections, making them efficient at transmitting data and robust to random node failure [1]. Many different types of networks are scale-free, from computer science (e.g. the Internet) and sociology (e.g. social networks) to biology (e.g. protein and ecological networks) [1]. Animal brains, including the human brain, are scale-free, and it is thought that this property is one reason they are so capable [18]. While the scale-free property may not help performance on all problems, it is evidently bene-

ficial on many problems, and it is therefore beneficial if an encoding can produce it when it is useful.

Previous evolutionary algorithms have evolved networks that have some of the aforementioned network properties. Most such algorithms are *generative encodings* [17], wherein information in a genotype can influence multiple parts of a phenotype, instead of *direct encodings*, wherein genotypic information specifies separate aspects of phenotypes. Generative encodings tend to produce more regular phenotypes that outperform direct encodings [3, 8, 10] and can produce modular and hierarchical structures [10]. Individual generative genomes can also produce networks of various sizes, as appropriate [16, 9, 7, 5]. We are not aware of research that has investigated whether evolved networks are scale-free.

Two generative encodings that DSE is inspired by have become well-known in part because of their ability to produce important network properties. Cellular Encoding (CE) iteratively applies instructions to manipulate network nodes and links. The encoding allows cells to call functions, which involve an entire sequence of instructions (encoded in a grammar tree), facilitating the production of regular, modular, and hierarchical networks [5]. Intriguingly, because CE allows a repeated symbol to unpack into the same subnetwork, the exhibited regularity, while high, tends to be a formulaic repetition of the same subnetwork, without complex variation.

HyperNEAT generates patterns in neural networks via an abstraction of biological development [16]. Just as in natural developing organisms, phenotypic attributes (in this case, edge weights in a network) are a function of evolved geometric coordinate frames. HyperNEAT departs from nature, however, by not having an iterative growth process. A single HyperNEAT genome can encode networks of different sizes; one was scaled up from thousands of connections to millions and still performed well [16]. The neural wiring patterns HyperNEAT produces are highly functional, complex, and regular, including symmetries and repetition, with and without variation [3]. However, HyperNEAT struggles to produce modular networks [2] unless it is manually injected with a bias towards modular wiring patterns [20].

Intriguingly, CE excels at producing modularity, hierarchy, and formulaic regularity, but fails to generate complex regularities with variation. HyperNEAT, on the other hand, excels at producing complex regularities, but tends to create non-modular networks. This raises the question of whether an encoding that combines the iterative growth of CE with pattern-generating capabilities similar to HyperNEAT can create modular, hierarchical networks with complex regularities. DSE was designed with this motivation.

We will describe DSE (see [19] for additional details) and then present preliminary investigations into the properties of the networks it evolves. The properties under focus are the modular, scalable, and scale-free nature of DSE networks. We explicitly reward for networks with these properties to test if DSE can produce them, and compare the results to HyperNEAT. Unfortunately, we were unable to obtain a functioning distribution of CE, making a comparison with that encoding impossible. By explicitly selecting for a desired property, we can study an encoding's ability to produce it. While we could have evolved on a problem where it is supposed that scale-free, modular, scalable networks would be beneficial, it is often difficult to ensure that these properties are beneficial on a specific problem. To perform a more traditional test of the capability of an encoding, we also compare DSE to HyperNEAT on a visual pattern recognition task. On all three challenges DSE outperforms HyperNEAT, suggesting that it is an encoding that deserves additional investigation.

2. DESCRIPTION OF DSE

2.1 Phenotype

The phenotype of the individual is the network:

$$G = (\mathbf{V}, E) = ([v_1 \ \ldots \ v_N], \{e_{ij}\}), \tag{1}$$

where $\mathbf{V}$ is a vector of nodes and E is a set of connections. Nodes and connections can have attribute values, which play a role either in development or in computation. Weighted connections have a weight attribute and plastic connections can have a number of attributes (parameters). The representation of G is a bit different from conventional notation for directed graphs, where nodes and connections are contained in sets and the attributes are eventually imposed using functions. This departure follows from the assumed sequential model of computation, in which nodes are evaluated in an ordered manner. It is therefore natural to have them explicitly ordered in a vector.

The vector $\mathbf{V}$ is composed of 3 contiguous subvectors: input $\mathbf{V}_u$, hidden $\mathbf{V}_h$ and output $\mathbf{V}_y$. Input and output subvectors constitute the network interface and their length depends on the problem. The hidden part of the network is free to vary in size.

DSE network phenotypes are executed similarly to artificial neural networks. The network is initialized before an evaluation by zeroing the values of all nodes and setting the weights of any plastic connections to their initial values. Then, for any input vector $\mathbf{u}$, the network is evaluated as follows:

1: Assign inputs $\mathbf{V}_u := \mathbf{u}$
2: **For** each $v_j \in [\mathbf{V}_h \ \mathbf{V}_y]$ **do**
3: Evaluate node: $x_j := f_j(\{e_{ij}(x_i)\}_{i=1,\ldots,n_i})$
4: **For** each connection e_{ij} **do**
5: Update e_{ij} (if applicable)
6: Return $\mathbf{V}_y$,

where x_j denotes the node's value, $e_{ij}(x_i)$ denotes the operation performed by the connection (usually multiplying the source node x_i value by the weight of the connection), and f_j denotes a transfer function assigned to node j. The transfer function operates on the set of incoming signals, which are usually aggregated by summing. While DSE can handle plastic connections, network plasticity is not explored in this paper and is therefore not described further.

2.2 Genotype

The genotype is a program to grow the network. The program is tree-structured, with nodes being routines. The root of the tree constitutes the main routine and is sufficient to solve many test problems. Each routine tree has the same structure: $\gamma = (R, \text{Body}, \text{Tail}, \gamma_\gamma)$, where R is an identifier, Body is the main list of instructions, Tail is a list of terminating instructions, and γ_γ is a set of subroutines.

The whole genomic program Γ, i.e. the tree of γ routines, grows the network from its initial state into its final state, i.e. $G_\tau = \Gamma(G_0)$. The main component of the routine is Body, which contains instructions that act on the network to develop it. Each instruction in Body can be seen as a function

λ taking the network G_t and returning a new network, i.e. $G_{t+1} = \lambda(G_t)$. The instructions and their properties are described in the subsequent list. Note that the arguments X and Y following an instruction can select many nodes at once, allowing operations to be carried out on groups of nodes. A central concept of DSE is that nodes are identified by an index and a *label*. Nodes having the same label make up groups (e.g. inputs), and instructions can act on all nodes from a group at once. Within groups, index values provide a geometry for patterns of nodes to be selected and acted upon.

1. Con X Y C: connect nodes X and Y, with the connection having the same attribute values as the reference connection C.
2. Cut X Y: cut connection(s) between X and Y.
3. ConE X Y C E: connect nodes X and Y that satisfy the expression E, using C as reference connection.
4. CutE X Y E: cut connection(s) between X and Y that satisfy the expression E.
5. DivP X Y: divide node(s) X in parallel, which duplicates the node(s) along with all its connections, places the new node(s) in $\mathbf{V}$ right after the original and assigns label Y to them.
6. DivS X Y: same as DivP, except the division is sequential, which means that instead of duplicating connections, Y takes over the outgoing connections of X and a connection from X to Y is created.
7. Subst X Y: substitute node symbols X with Y.
8. Call R X: call the subroutine R for each node X.
9. Term X Y: like Subst, except Y is necessarily terminal, i.e. it denotes a transfer function.

These instructions provide DSE with interesting capabilities motivated from Cellular Encoding [5] and HyperNEAT [16]. As in Cellular Encoding, nodes can divide in a parallel or sequential manner, and subroutines can be called. These features may encourage modularity and hierarchy. Similar to HyperNEAT, complex geometric patterns can be evolved with an expression tree (E), which is a genetic program that takes arguments derived from index values of nodes, which correspond to geometric coordinates in HyperNEAT. Expression tree outputs determine connection weights via the ConE and CutE instructions. The Con and Cut operations can also create regular patterns by selecting groups of nodes, or can adjust individual connections. The ability to create regular and irregular changes, including changes to single weights, can improve the performance of an encoding [3]. The substitution operation can also create regular patterns by selecting groups of nodes and assigning transfer functions to them.

An initial population of genotypes consists of either empty or randomly-generated genotypes, the latter of which have instructions with arguments that are *undetermined* (explained in Section 2.4).

2.3 Development

Development starts from an initial network G_0 with an optional hidden node called A_0 that is fully connected (weights = 1) to inputs and outputs. The final network G_τ is developed by executing the Body and Tail parts of the genomic program, i.e. $G_\tau = \Gamma(G_0)$.

The network G_τ is guaranteed to be functionally valid due to the Tail and the covering operator, which automatically inserts Term instructions for any non-terminal symbol in $\mathbf{V}$.

Figure 1 provides a simple example of network development. Multiplication nodes (marked with $*$) without inputs produce an output of 1.0. If the output is thresholded at 0.5, the network solves the logical equals function.

An important feature of DSE is the reuse of code via routines. For example, when the instruction Call R X is executed, if a subroutine with an identifier matching R exists in the list of subroutines, development replaces X with a new subnetwork. Specifically, the new subnetwork takes on the inputs and outputs of X, and X becomes the hidden node in the newly developed subnetwork.

Figure 2 provides an example of this feature, and also illustrates how DSE can encode networks with regularity, modularity, and scalability. This network evolved to solve the n-bit parity problem, which identifies whether a given input vector contains an even number of ones. The network is composed of n-input versions of two logical functions: NAND (n), which returns 1 for 0 inputs and otherwise returns 1 iff all inputs are 0; and OR (|), which returns 0 for 0 inputs and otherwise returns 1 if at least one input is 1.

The solution is partially scalable in that a single genotype encodes networks that solve several different problem sizes: the genotype produces valid solutions for $2, 3, \ldots, 9$ inputs, but it fails for higher numbers. A subroutine (labeled r214 in Figure 2), which is a genotypic module, produces a five-node phenotypic module that computes a 2-input XOR function. The routine is repeatedly called to generate a regular pattern of three of these subnetwork building blocks. This genome has thus discovered the key regularity of the problem, which would likely enhance its evolvability were it further evolved to solve problems with more than 9 inputs, since doing so would only require appropriately calling the r214 subroutine.

2.4 Genetic operators

The four operators that modify the genotype are mutation, crossover, cleaning and covering. We describe them in general terms, because they are not essential to the encoding and different implementations could be chosen.

Mutations are performed with a fixed probability (here 0.8) during reproduction. They start with the main routine and recurse deeper into the tree, and can insert a new instruction, delete an instruction, mutate an instruction's argument, or duplicate an instruction while mutating an argument. A mutated argument is changed to *undetermined* and is determined again during the next development of the network. Undetermined arguments are *determined* during network development to select a randomly-chosen set of nodes already in the network (otherwise, arguments would not match nodes in the network and instructions would be mostly ineffective). This pertains to all node-selecting arguments (denoted as X and Y); expression trees are mutated as is typically done in Genetic Programming [15].

The crossover operator transfers two randomly selected instructions from a donor's routine to a recipient's routine, much as in Linear Genetic Programming [15]. The cleaning operator periodically (here, 5% of reproductions) removes genotypic instructions that have no effect. This operator improves genotype readability and can impact search by removing neutral variation, which improved performance on some problems in preliminary experiments, but further experiments are necessary to determine its impact.

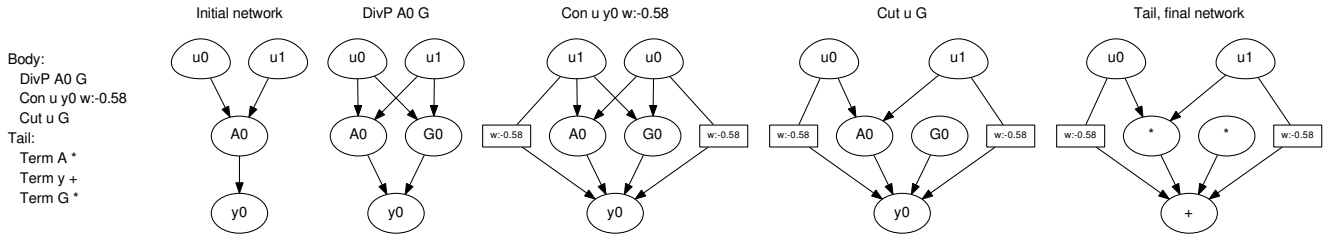

Figure 1: Example of network development. The three Body **instructions in the genome (left) are executed before the** Tail**, which assigns valid transfer functions. The final network performs the logical equivalence (equals) operation. Input nodes have irregular ovals and the output is the lowest node.**

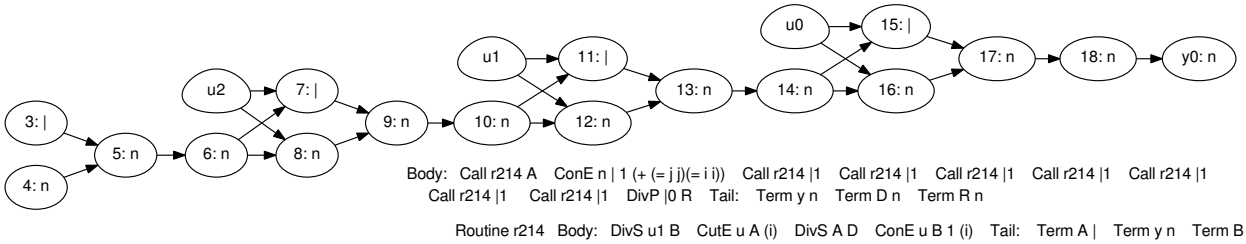

Figure 2: A modular, partially-scalable solution to the n-bit parity problem. The visualized network is developed for 3 inputs (irregular ovals), although the genome produces solutions for 2, 3, . . . , 9 inputs. The output is the rightmost node.

Unlike the previously-described operators, covering is not applied during reproduction, but at the end of development. For each non-terminal symbol X remaining in $\mathbf{V}$ at the end of Tail, the operator appends a new Term X Y instruction. Covering may be applied for each routine or for the main routine only. Either way it guarantees that all the nodes in the final network are assigned valid transfer functions.

3. EXPERIMENTS

In all experiments the population size and number of generations is 200, and networks are feed-forward. Experiments are implemented identically in DSE and HyperNEAT, with default parameter settings. Full details for HyperNEAT are in Stanley et al. [16]. Briefly, HyperNEAT evolves genomic networks that encode the weights of neural network phenotypes as a function of the geometric location of the weights.

3.1 Evolving scale-free networks

Networks are scale-free when the distribution of the number of connections per node (its *degree*) follows a power law, such that the proportion of nodes having degree k is:

$$P(k) \propto k^{-\gamma}, \tag{2}$$

where γ is an exponent parameter, typically in the range [2, 3] (set to 2 in our experiments). To test whether DSE and HyperNEAT can produce scalable, scale-free networks, we rewarded genotypes that produce scale-free networks at several different target sizes.

The scale-free property is measured with the Bhattacharyya coefficient as the overlap between the distributions P (the power law distribution) and q (the distribution of the evolved network):

$$B(P,q) = \sum_k \sqrt{P(k)q(k)}. \tag{3}$$

A single HyperNEAT genotype generates networks for 7 different substrate sizes: $(i+3)^2$, where $i = 0, \ldots, 6$, and for each of these resolutions the scale-free property is measured. The fitness measure for each network size is

$$f_i = 100 + B(P,q) - (n_c - 1), \tag{4}$$

where n_c is the number of separate network components. The second term penalizes networks with disconnected pieces. The 100 ensures all fitnesses are positive.

DSE is similarly challenged with generating genotypes that encode for networks that maximize the scale-free property and have approximate sizes of $n_i = (i+3)^2$, $i = n_u = 0, \ldots, 6$. The genome produces different sized networks based on the number of inputs n_u. To encourage correct network sizes, an additional penalty term is added to the DSE fitness measure for each network size

$$f_i = 1 - B(P,q) + (n_c - 1) + |1 - n/n_i|, \tag{5}$$

where n is the number of nodes in the developed network. Contrary to HyperNEAT, the DSE fitness function is minimized. In both cases the overall fitness function is the following weighted average from all fitness cases:

$$F = \frac{\sum_i f_i \sqrt{n_i}}{\sum_i \sqrt{n_i}}. \tag{6}$$

This equation weights the results for bigger networks as more important than smaller networks in square root proportions. Because the two fitness functions differ, we will compare only the B coefficients of evolved networks.

The results from 30 runs of DSE and HyperNEAT indicate that both encodings produce networks at different network sizes that are scale-free (Figure 3). The DSE results are slightly, but significantly, better ($p < 0.001$, permutation test), although the networks from both encodings had B

Table 1: Target and evolved sizes of networks evolved with DSE in both scale-free and modular networks experiments.

Target size	9	16	25	36	49	64	81
Scale-free							
Mean size	8.7	16.5	25.4	36.3	49.0	64.1	76.3
Std. dev.	0.4	0.7	0.7	0.8	1.4	1.6	3.6
Modular							
Mean size	9.2	18.2	28.0	39.3	52.5	65.3	76.5
Std. dev.	0.4	1.8	2.0	2.9	2.6	2.8	2.8

coefficients near the maximum value of 1.0. The plotted B coefficients are weighted in the same way that fitness is weighted in eq. 6. DSE was also nearly perfect at producing networks of the target size (Table 1). HyperNEAT networks had fixed, predefined sizes, and thus cannot be compared on this dimension.

Visualizing the networks reveals interesting differences in the types of networks each encoding produces (Figures 4 and 5; produced in *Graphviz* with the radial layout). While both encodings produce regular networks, the DSE networks appear more modular, even though modularity was not rewarded in these experiments. By quantifying modularity with the Q metric [14] (see the next section), we validate that DSE networks from these experiments are significantly more modular (DSE $Q = 0.48 \pm 0.08$, HyperNEAT $Q = .04 \pm .05, p < 0.001$, permutation test). The DSE networks also appear hierarchical in that nodes typically have only one path through multiple other nodes to hub nodes, although quantifying hierarchy is beyond the scope of this paper.

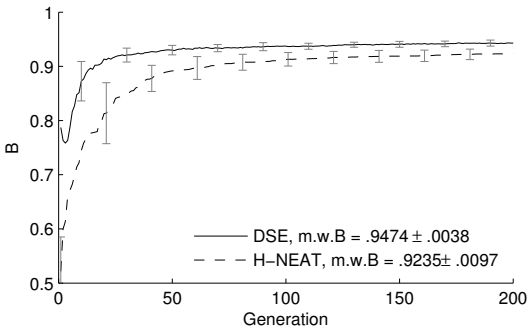

Figure 3: Both DSE and HyperNEAT can evolve genomes that produce networks of different size that are scale-free. Plotted are weighted averages ± SD of the best B coefficients from 30 runs.

3.2 Evolving modular networks

Testing the ability to evolve modular networks is formulated identically to the scale-free test, except the quantity maximized is the modularity measure Q from Newman and Girvan [14, 13]. Let e_{ij} denote the proportion of connections between the i-th and j-th group of nodes (a module), and let $a_i = \sum_j e_{ij}$, then

$$Q = \sum_i (e_{ii} - a_i)^2 \quad (7)$$

is the proportion of all connections within modules minus an expected value of the same quantity if the connections were distributed uniformly randomly. In other words, Q indicates how much the modularity of the given network differs from a network with all random connections. For a fully random network, Q is about 0, while values above 0.3 indicate some modularity in the network, and the the highest Q observed at the time of publication was 0.807 [13]. Computation of the Q coefficient requires the network to be partitioned into modules in advance, otherwise it is not possible to calculate the value e_{ij}. Ideally Q would be calculated for all possible partitions of the network and the partition maximizing Q would be selected. In practice this exhaustive approach is computationally prohibitive, so a partition is selected with a greedy hill-climbing approach for which Q is the maximum [13]. The algorithm starts by considering each node a separate module and iteratively joins modules for which the increase in Q is the highest. The complexity of the algorithm is $O((m+n)n)$, where n and m respectively denote the number of nodes and connections.

DSE significantly outperforms HyperNEAT at evolving modular networks of different sizes (Figure 6, $p < 0.001$, permutation test). The Q values plotted are weighted identically to the B values in the previous section, averaged over 20 runs. Interestingly, DSE networks are initially quite modular, and reach near-maximum modularity within a few generations, whereas HyperNEAT required many generations to approach its maximum, revealing the default tendencies of these encodings. DSE also evolved to produce networks closely approximating the required size (Table 1). Interestingly, additional experiments where HyperNEAT genomes were tested on only one network size reveal that HyperNEAT can produce highly-modular networks (data not shown), suggesting that the need to simultaneously produce modularity on networks of different sizes is a unique challenge for HyperNEAT, but not DSE.

Figures 7 and 8 (*Graphviz* with spring-model layout) show the best evolved networks from both methods for increasing numbers of inputs (DSE) or increasing network size (HyperNEAT). Evolved networks display some topological patterns that scale up in interesting ways. The DSE networks resemble starfish that have one more arm than the number of inputs. The HyperNEAT networks consist of clusters that increase in number along with the network size, but are too densely interconnected to yield a high average Q score.

To further probe differences between the encodings on properties not directly selected for, we measured the scale-free property for the networks from this experiment. Although they were not selected for it, the DSE networks were significantly more scale-free (DSE $B = 0.67 \pm 0.07$, HyperNEAT $B = 0.50 \pm 0.18, p < 0.001$, permutation test).

3.3 A visual pattern recognition problem

It is also important to test if DSE is competitive with cutting-edge generative encodings at solving problems. We chose a visual pattern recognition problem that HyperNEAT was recently tested on [4], which itself was motivated by a problem from the paper that introduced HyperNEAT [16].

In this problem, 3 shapes are visible and the network has to identify the location of a target shape while ignoring the other two shapes. Each shape fits within a 3×3 grid of pixels (Figure 9) and all 3 shapes are distributed on a 9×9 pixel canvas. The S shape is most unique and thus easiest to distinguish. The networks are fixed-size, with a 9×9 input sheet of nodes and a corresponding 9×9 output sheet. For

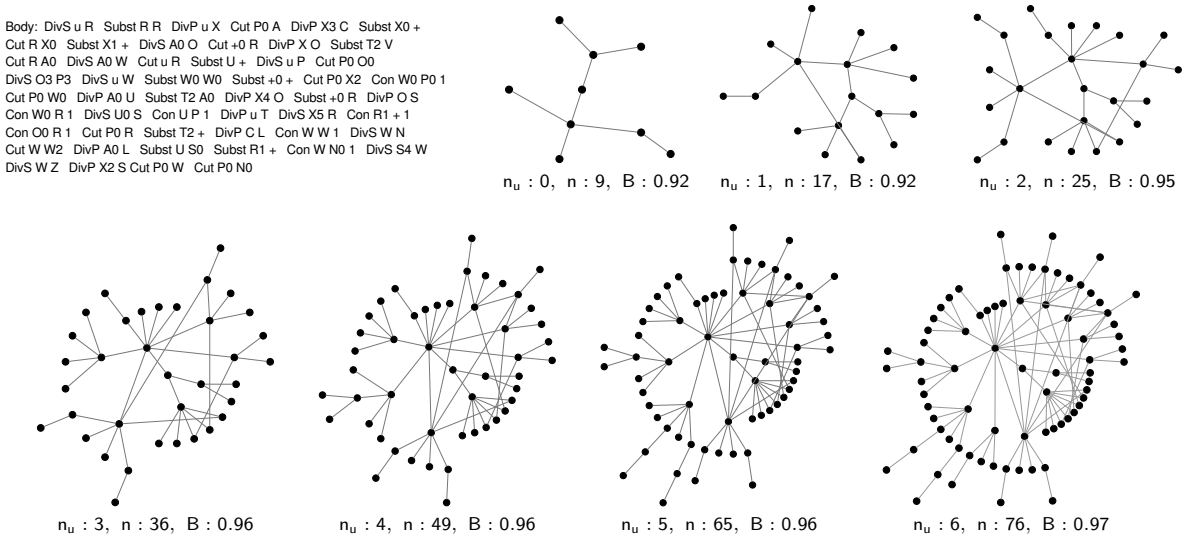

Figure 4: The networks produced by the DSE genome that evolved the highest average B coefficient. n_u = number of inputs, n = network size, and B measures the scale-free property.

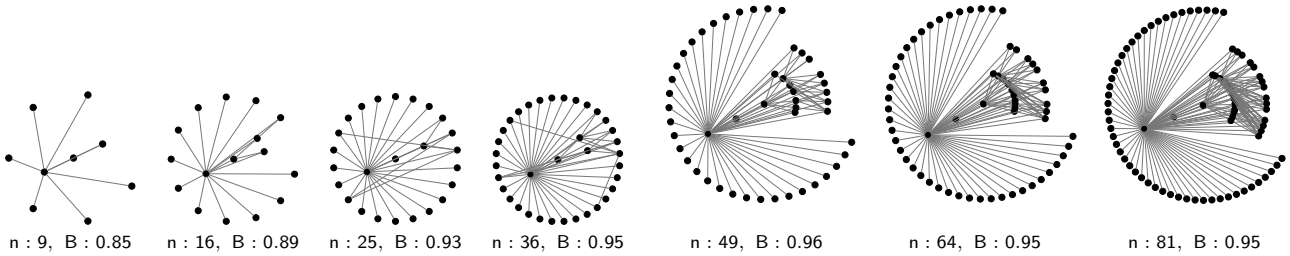

Figure 5: Network phenotypes for the HyperNEAT genotype that evolved the highest average B coefficient.

a single evaluation, shapes are placed at random locations without overlap. A network evaluation is considered a success if the output node with the highest activation value is in the same location as the center of the target shape, otherwise an error is scored. Averaged over 20 evaluations, the fitness for DSE genotypes is the error rate (to be minimized) and for HyperNEAT is the success rate (to be maximized). There are 3 variants of the problem, with each shape respectively being the target and the other two being distractions.

Because this task is two-dimensional, DSE uses 2D variants of the ConE and CutE instructions, which can create connectivity patterns via an expression tree. Since these are the only two instructions in this experiment, networks remain a fixed size. Nodes have two indexes as in a matrix.

A key element of both encodings is that evolved functions create geometric patterns that specify network connectivity. In DSE, such patterns are created by expression trees (Section 2.2). In both encodings, the functions that produce geometric patterns have 8 arguments: the x and y values of the input and output nodes (normalized to $[0, 1)$), Δ_x and Δ_y (the difference between input and output nodes in the x and y dimensions), and the straight-line distance and polar angle between input and output nodes. For DSE, the node coordinates come from normalizing their indexes.

One of the main differences between HyperNEAT and expression trees are the primitive (building block) operations allowed in both systems. In these experiments, HyperNEAT

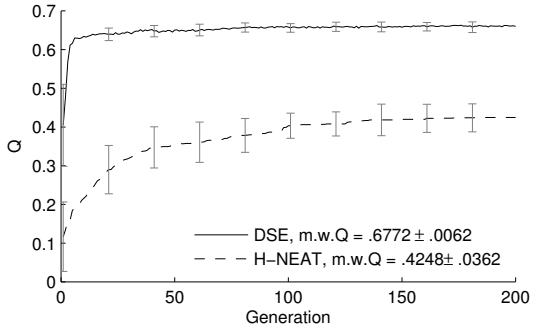

Figure 6: Resultant modularity values (weighted mean $Q \pm$ SD) when modularity is selected for.

has a default set of operations following previous studies [16, 3], with sigmoid, sine, Gaussian, and bounded linear functions. DSE has $+$, $-$, $*$, abs(), $<$, Gauss(), and constants randomly generated from a normal distribution (mean 0 ± 1 SD). For both HyperNEAT and DSE, the output of an evolved pattern-generating function specifies a network weight (set to 0 if between -0.2 and 0.2). Another important difference is that in DSE weights produced by subsequent ConE2D instructions accumulate. CutE2D instructions cut any connections for which the expression value is positive.

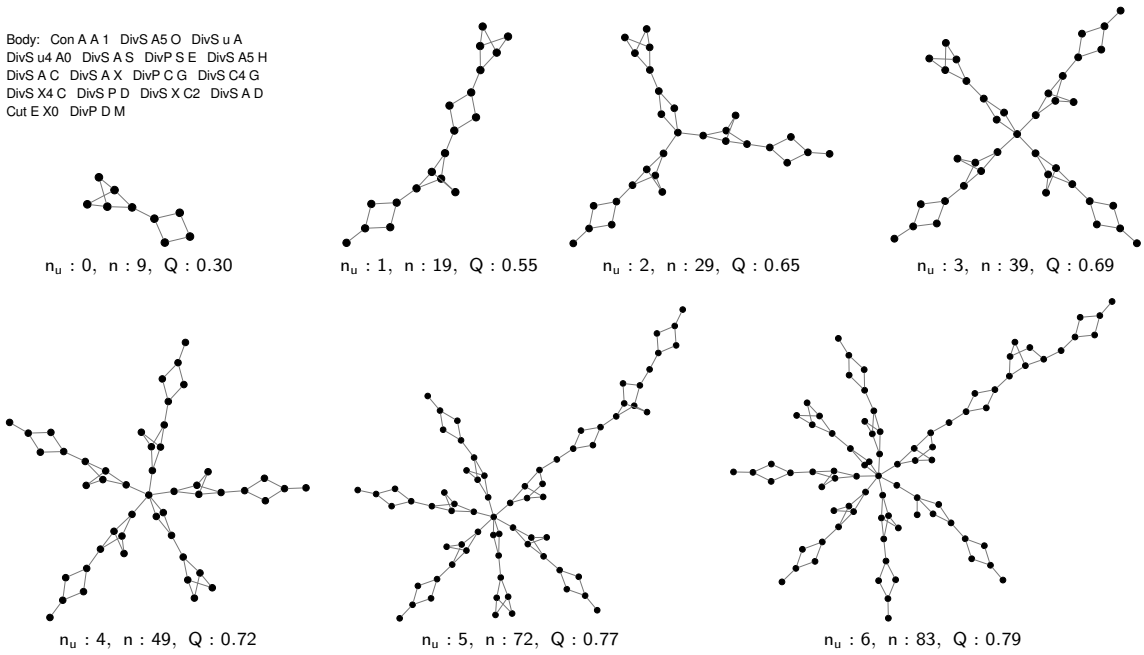

Figure 7: The DSE genotype that produced the highest Q coefficient and the 7 networks it develops for different numbers of inputs $n_u = 0, \ldots, 6$.

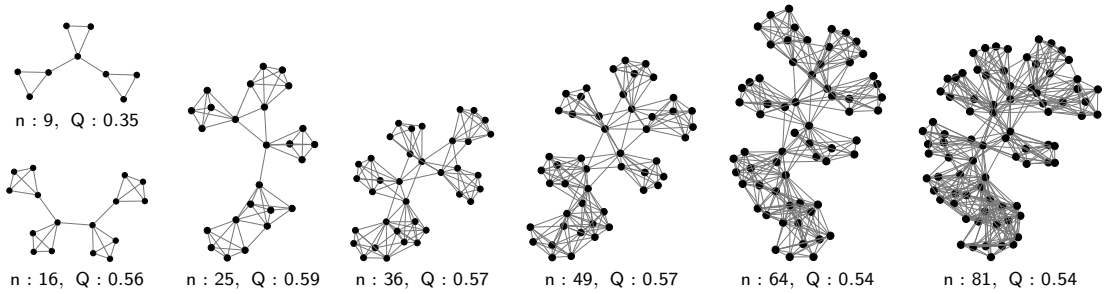

Figure 8: Network phenotypes for the HyperNEAT genome with the highest Q value.

We perform 30 evolutionary runs for each of the three task variants. DSE substantially and significantly outperforms HyperNEAT on all three problem variants (Table 2, $p < 0.01$). These scores underrepresent DSE's advantage, because plots over time (not shown) suggest that only DSE would improve with additional generations on the X and O problems, and because DSE achieved near-perfect fitness on the S problem much earlier than HyperNEAT. While only from a single problem, these results suggest that DSE can compete with cutting-edge generative encodings, such as HyperNEAT, especially given that this problem type was chosen to highlight HyperNEAT's merits.

A benefit of encodings like DSE that have separate instructions or rules is that individual instructions can serve as genetic modules that create wiring motifs that may be beneficial on related problems. To test whether instructions from one task aid in solving another task, we implemented an island model, where organisms on three islands are rewarded for their performance on the three different task variants. Migrants travel between islands, where they can be crossed with native individuals. 5% of individuals migrate between each pair of islands each generation, crossover occurs 40% of the time, and 60% of genomes are mutated. Exchanging genetic information between organisms rewarded for different tasks should only be beneficial if instructions contain partial solutions common to multiple tasks.

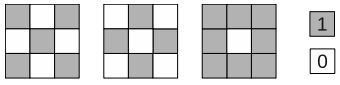

Figure 9: Shape X (cross), O (circle) and S (square).

DSE performance significantly improves with migration and crossover on the X and O problems (DSE 11, Table 2, $p < 0.01$, permutation test). There was no meaningful room for improvement on the S variant. To ensure that performance gains are due to exchanging beneficial instructions, we test DSE with migration, but not crossover (DSE 10) and with crossover, but not migration (DSE 01). Neither of these controls significantly improve performance over regular DSE (Table 2, $p > 0.05$, permutation test), demonstrating that DSE can exchange functional building blocks between individuals working on similar problems to enhance perfor-

Table 2: Error rates (percent wrong) on the pattern recognition problem. Averages of the final best fitness values are shown, tested on 600 shape placements to increase reporting accuracy. The letters X, O, and S indicates which pattern was the target to be recognized.

	X	O	S
H-NEAT	67.1 ± 5.0	63.6 ± 4.2	1.3 ± 5.5
DSE	29.0 ± 22.8	20.4 ± 29.1	0.1 ± 0.3
DSE 10	31.8 ± 20.0	12.1 ± 21.9	0.0 ± 0.1
DSE 01	32.6 ± 21.7	22.9 ± 29.8	0.1 ± 0.3
DSE 11	14.2 ± 15.8	06.9 ± 15.3	0.0 ± 0.1

mance. This capability should be increasingly beneficial as the number of objectives simultaneously being solved rises. It is an open question whether other leading generative encodings, such as HyperNEAT, can similarly exchange partial solutions to different problems between individuals.

To illustrate how expression trees can solve problems by creating connectivity patterns, we describe one of the shortest DSE solutions for the S problem variant, which was the easiest. This solution contained only one instruction that had an effect: ConE2D u y (abs (< u2 0.22)). It creates connections between all input-output node pairs with a straight-line distance between them of less than 0.22, which connects each output node to its 3×3-neighborhood counterpart in the input layer. This connectivity pattern ensures that the highest activation in the output layer is the node corresponding to the center of the square shape.

4. DISCUSSION AND CONCLUSION

This paper demonstrates that a novel encoding for evolving networks, DSE, can produce networks that are regular, modular, scalable, and scale-free. The networks also appear hierarchical, although additional work is needed to quantify this property. As such, DSE produces networks with many properties thought to enhance evolvability and performance [12, 11, 1]. DSE is interesting because it combines concepts from two different camps of generative encodings. Like Cellular Encoding and L-Systems, DSE has a developmental process that iteratively rewrites symbols according to instructions. Like HyperNEAT, it can also specify connectivity as a function of evolved geometric patterns. Our results suggest that this hybridization can combine the best attributes of both camps of generative encodings. For example, DSE can create scalable and modular networks (features typically associated with iterative-rewrite encodings [10, 8, 5]) with regular network patterns (HyperNEAT's forte [16, 3]). It is interesting, for instance, that DSE produced more modular and scale-free networks than HyperNEAT when these properties were not explicitly rewarded. Future work is required to better understand the relative strengths and weaknesses of both these camps of encodings and their hybridizations. DSE thus raises interesting questions about what types of networks different encodings produce. It also appears to be a promising encoding in its own right worthy of additional investigation.

Acknowledgments: NSF Postdoctoral Research Fellowship in Biology to Jeff Clune (award number DBI-1003220).

5. REFERENCES

[1] A. Barabási. Scale-Free Networks: A Decade and Beyond. *Science*, 1173299(412):325, 2009.

[2] J. Clune, B. E. Beckmann, P. K. McKinley, and C. Ofria. Investigating whether HyperNEAT produces modular neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 635–642. ACM, 2010.

[3] J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, To appear, 2011.

[4] O. J. Coleman. Evolving neural networks for visual processing. B.S. Thesis. U. New South Wales, 2010.

[5] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Supérieure de Lyon, France, 1994.

[6] S. Harding and W. Banzhaf. Artificial development. *Organic Computing*, pages 201–219, 2008.

[7] S. Harding, J. Miller, and W. Banzhaf. Self modifying cartesian genetic programming: Parity. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 285–292. IEEE, 2009.

[8] G. Hornby, H. Lipson, and J. Pollack. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719, 2003.

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

[10] G. S. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2005.

[11] N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proc. Natl. Acad. Sci. U.S.A.*, 102(39):13773, 2005.

[12] H. Lipson. Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, 7(4):125, 2007.

[13] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[14] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):26113, 2004.

[15] R. Poli, W. Langdon, and N. McPhee. *A Field Guide to Genetic Programming*. Lulu Press, 2008.

[16] K. Stanley, D. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[17] K. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2), 2003.

[18] G. Striedter. *Principles of brain evolution*. Sinauer Associates Sunderland, MA, 2005.

[19] M. Suchorzewski. Evolving scalable and modular adaptive networks with Developmental Symbolic Encoding. *Evolutionary Intelligence*, To appear, 2011.

[20] P. Verbancsics and K. Stanley. Constraining Connectivity to Encourage Modularity in HyperNEAT. *U. Central Florida Tech Report*, 2010.