# How Generative Encodings Fare on Less Regular Problems

Jeff Clune
Digital Evolution Lab
Michigan State University
jclune@msu.edu

Charles Ofria
Digital Evolution Lab
Michigan State University
ofria@msu.edu

Robert T. Pennock*
Digital Evolution Lab
Michigan State University
pennock5@msu.edu

## ABSTRACT

Generative representations allow the reuse of code and thus facilitate the evolution of repeated phenotypic themes or modules. It has been shown that generative representations perform well on highly regular problems. To date, however, generative representations have not been tested on irregular problems. It is unknown how fast their performance degrades as the regularity of the problem decreases. In this report, we test a generative representation on a problem where we can scale a type of regularity in the problem. The generative representation outperforms a direct encoding control when the regularity of the problem is high but degrades to, and then underperforms, the direct control as the regularity of the problem decreases. Importantly, this decrease is not linear. The boost provided by the generative encoding is only significant for very high levels of regularity.

## Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Connectionism and Neural Nets

## General Terms

Experimentation, Algorithms

## Keywords

Evolution, regularity, modularity, ANN, NEAT, HyperNEAT

## 1. INTRODUCTION AND MOTIVATION

Generative encodings, in which one element in a genome can produce many elements in a phenotype, have generated interest because their reuse of code facilitates the evolution of modular phenotypes [1]. In contrast, direct encodings, which involve a one to one mapping between a genotype and a phenotype, do not facilitate the evolution of repeated phenotypic modules. While many researchers have shown that generative encodings can outperform direct encodings [1], in every case we are aware of the problem was nearly perfectly regular or the regularity of the problem was unknown. By 'regular' we mean problem domains for which solutions consist of repeating the same motif multiple times. To date it is unknown how generative encodings do as the regularity of the problem decreases. For example, how good are generative encodings at producing an exception to the rule? What is needed is a test of generative versus direct encodings on a problem that allows us to explicitly vary *only* the regularity of the problem. Here we report on such a study.

## 2. THE EXPERIMENTAL SYSTEM

We conduct our investigation using Hypercube-based NEAT ('HyperNEAT') [2, 3], a recently introduced generative encoding that can evolve neural nets using the principles of the widely used NEAT algorithm [4]. HyperNEAT evolves Compositional Pattern Producing Networks (CPPNs), each of which is a function that takes as inputs a constant bias value and the locations on a Cartesian grid of both an input node (e.g. <x=1, y=1>) and an output node (e.g <x=2, y=2>). The output value of the function determines the weight of the link between the input and output node. All pairwise combinations of input and output nodes are iteratively passed to the CPPN to generatively create a neural network phenotype. Each CPPN is itself a directed graph network where each node is a math function (e.g. sine or Gaussian). The nature of the functions used can create a wide variety of desirable properties, such as symmetry and repetition. Perceptron NEAT ('P-NEAT') has been previously used as a direct encoding control for HyperNEAT [2, 3]. It is similar to HyperNEAT but directly evolves the neural net phenotypes instead of using generative CPPNs to generate them. A full explanation of HyperNEAT, P-NEAT and NEAT can be found elsewhere [2, 3. 4].

Following [3], we use a configuration that separates the inputs and outputs onto two separate planes. This configuration features a two dimensional, n-by-n Cartesian grid of inputs and a corresponding n-by-n grid of outputs. There are no hidden nodes and recurrence is disabled. The parameter configurations for HyperNEAT are the same as in [3]. We created a problem we call 'bit mirroring,' where for each input we assign a target output. Negative or positive ones are randomly provided as inputs and the fitness is incremented if the input is reflected in the target output. This step is repeated 10 times for generality. The correct wiring is to create a positive valued link between each input node and its target output and, importantly, to zero out all links between each input node and non-target output nodes. That there is a correct wiring motif that needs to be repeated for each input cell creates an inherent regularity to the problem. However, this inherent regularity is held constant for a given grid size. The regularity of the problem can be scaled by changing the number of rows of nodes that are 'regular', where the target node is directly across (i.e. the x and y coordinates of the input and output node are the same). For randomized rows the target is randomly chosen within the row, meaning the x value of the input and output node may be different.

*This research was supported by a grant from the Cambridge Templeton Consortium, "Emerging Intelligence: Contingency, Convergence and Constraints in the Evolution of Intelligent Behavior."

*GECCO'08*, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.

# 3. RESULTS

Our first experiment tests the generative encoding HyperNEAT on the 3-by-3 bit-mirroring problem with varying degrees of regularity. Each of 10 trials was run for 250 generations for every experimental treatment (running the experiments longer did not qualitatively change the results). Figure 1 (left side) shows that HyperNEAT does better as the regularity of the problem increases. However, the only treatment that is significantly different in performance from the zero-rows regular treatment is the three-rows regular treatment (p < .01; this and all future p values report a two-sample t-test). Figure 2 (right side) shows that the regularity of the problem does not impact the performance of the direct encoding P-NEAT (there is no statistical significance between the zero-rows regular treatment and the others, p > .05 in all cases). Comparing HyperNEAT to P-NEAT illustrates the costs and benefits of using a generative encoding. HyperNEAT beats P-Neat when the regularity of the problem is high (3-rows regular, p < .01), ties it on problems of intermediate regularity (p > .05 on 1- and 2-rows regular treatments), and performs worse than P-NEAT on the problem with low regularity (p < .05 on the 0-rows regular treatment).

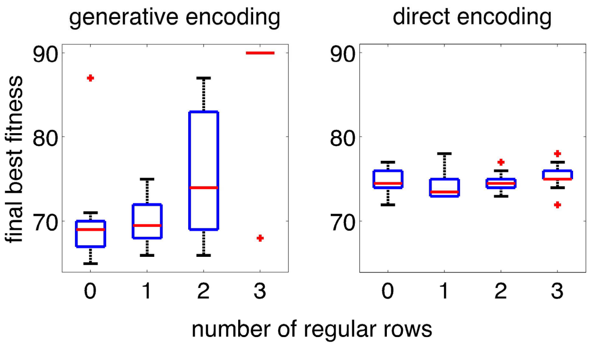

**Figure 2. Generative vs. direct encodings on the 3x3 bit-mirroring problem with varying amounts of regularity.**

While, the first experiment suggests that the HyperNEAT generative encoding only provides a boost in performance when there is complete regularity in the problem, the coarse granularity of the 3-by-3 problem makes it difficult to determine if perfect regularity is required. To get a better understanding of how HyperNeat takes advantage of increasing regularities in the problem, an experiment was conducted with the bit-mirroring problem on a 9x9 grid (Fig. 4). HyperNEAT's performance does improve as a function of the regularity of the problem, but the difference is not significantly better than the zero-rows regular treatment until the number of regular rows is six or greater (p >.05 for less than 6 regular rows, otherwise p < .001). These results suggest that, while HyperNEAT can take advantage of regularity in problems as that regularity increases, this pickup is slight until the regularity is high. While it is nice that generative encodings can provide a performance boost at all, the boost from this generative encoding on this problem is rather small in the majority of cases tested (treatments with five or fewer regular rows). HyperNEAT beat the direct encoding control on the 9-by-9, zero-rows regular bit-mirroring problem (p < .01) because the 9x9 grid has enough inherent regularity to provide HyperNEAT an advantage even when the type of regularity we were explicitly scaling was eliminated. That HyperNEAT does even better as a different type of regularity is added (Fig. 4) is a testament to HyperNEAT's ability to exploit multiple regularities simultaneously.

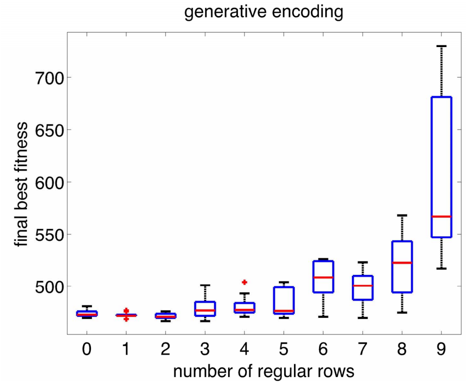

**Figure 4. A more fine-grained look at the correlation between increasing regularity and performance for the HyperNEAT generative encoding, here on the 9x9 bit-mirroring problem.**

# 4. CONCLUSION

Prior to this study, it was largely unknown how generative encodings perform as the regularity of a problem decreases. We found that if there is sufficient regularity in the problem, a generative encoding can provide a performance boost over a direct encoding. However, that benefit can fall off fast as the regularity of a problem decreases. Furthermore, a generative encoding can perform worse when the regularity of the problem is low. Finally, the generative encoding tested here showed the ability to simultaneously exploit at least two types of regularity. Given that most interesting problem domains probably contain many different types of regularity, we suspect that on most challenging problems generative encodings will outcompete direct encodings because they can exploit such regularities. More work is needed to see if the results found here with a particular generative encoding and its direct control on one problem domain apply more generally to generative encodings on most problems.

# 5. REFERENCES

[1] Stanley, K. O. Miikkulainen. A taxonomy for artificial embryogeny. Artificial Life, 9(2): 93-130, 2003.

[2] D'Ambrosio, D. B. and Stanley, K. O. A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '07)* (London, U.K., July 7-11, 2007). ACM Press, New York, NY, 2007, 974-981.

[3] Gauci, J. J. and Stanley, K. O. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '07)* (London, U.K., July 7-11, 2007). ACM Press, New York, NY, 2007, 997-1004.

[4] Stanley, K. O. and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2): 99-127, 2002