CA-NEAT: An Evolved Cellular Automata Morphogenetic System Based On Compositional Pattern-Producing Network Developmental Mappings

Stefano Nichele[1], Mathias Berild Ose[2], Sebastian Risi[3] and Gunnar Tufte[2]

[1]Oslo and Akershus University College of Applied Sciences, Oslo, Norway
[2]Norwegian University of Science and Technology, Trondheim, Norway
[3]IT University of Copenhagen, Copenhagen, Denmark
stefano.nichele@hioa.no

Abstract

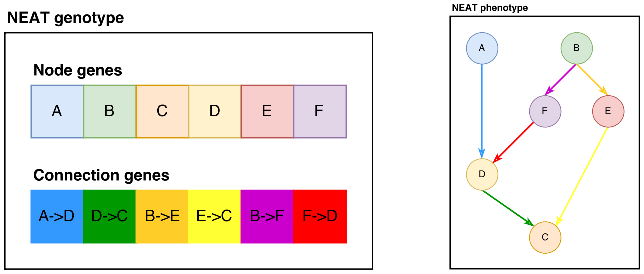
Complex self-architecturing systems are difficult to program, i.e. by top-down engineering. Kowaliw and Banzhaf (Kowaliw and Banzhaf, 2012) argue that the bottom-up methodology of artificial development is an appropriate means of approaching complex systems engineering. However, achieving some sort of self-architecturing properties, e.g. morphogenesis or self-replication, is not trivial. One way of "programming" such developmental systems is through artificial evolution, i.e. a combined evolutionary and developmental approach (EvoDevo). Searching for a solution for an artificial EvoDevo system that targets levels of complexity found in nature can be intractable. Therefore, an appropriate mapping that scales well and at the same time allows solutions to evolve incrementally, starting with a solution encoded into a small genome gradually complexified by adding new degrees of freedom, is desired.

In this work a cellular system is used as testbed for morphogenetic engineering. A traditional CA table-based encoding is replaced by a *Compositional Pattern Producing Network* (CPPN) mapping, a developmental encoding often used in systems without local interactions (Stanley, 2007). In our work a CPPN is used as developmental encoding based on local interactions, i.e. a true morphogenetic cellular system. The cellular automata CPPNs are evolved through a *NeuroEvolution of Augmenting Topologies* (NEAT) algorithm, a method that evolves increasingly complex networks (Stanley and Miikkulainen, 2002).

A NEAT genome consists of genes that encode nodes and connections between them. Figure 1 shows an example genotype-to-phenotype mapping. NEAT starts with an initial population of very simple networks, typically with just the input and output nodes and connections between them. Over generations, more nodes and vertices are added or disabled, activation functions are changed, and weights are adjusted. The process of gradually expanding the genome is called *complexification*, and intends to reflect how life on earth is believed to have started with simple organisms and gradually evolved into more complex creatures (Darnell and Doolittle, 1986; Pross, 2005).

The approach described in this work is termed CA-NEAT. All cells in the systems are uniform, i.e. they share the same genome network. Two benchmark problems are investigated: 2D morphogenesis and replication of structures of increasing complexity.

Figure 2 shows the results for the evolution of the "Tricolor" flag pattern morphogenesis in 100 independent runs. In 100 generations, 93 of the independent runs achieved a perfect solution. The initial populations contained 200 genomes which consisted of an input layer with one node



(a) Genotype

(b) Phenotype

Figure 1: NEAT genotype and phenotype examples. The phenotype only shows the topology that the genotype encodes (weights and activation functions are omitted).

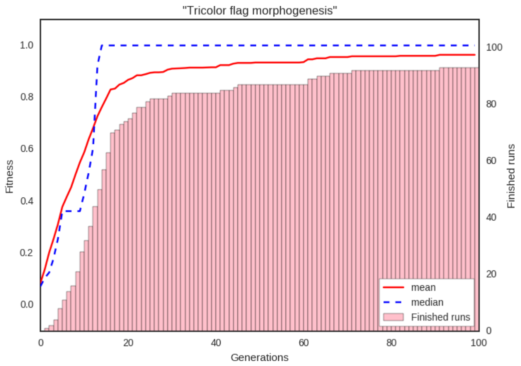


Figure 2: Tricolor flag pattern morphogenesis, first 100 generations.

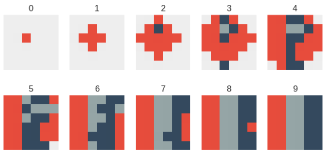


Figure 3: Example of morphogenesis.

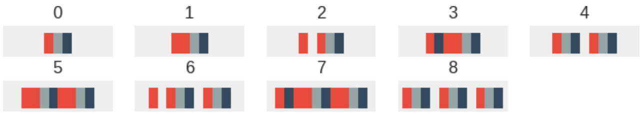


Figure 4: Example of replication.

per CA neighbor (von Neumann 5 neighbors) and one output layer with one node per possible cell state.

An example of evolved network for the “Tricolor” morphogenesis problem is shown in Figure 5. The two hidden nodes are not connected to output nodes and are thus “vestigial”. Dashed lines represent disabled connections. An example of morphogenesis process is depicted in Figure 3 and an example of replication is represented in Figure 4. Morphologies and structures of increasing complexity have also been investigated (Nichele et al., 2017), but are not included in this abstract due to space constrains.

Results show that CA-NEAT is an appropriate means of approaching cellular systems engineering. We argue that CA-NEAT could provide a valuable mapping for morphogenetic systems, beyond cellular automata systems, where development through local interactions is desired. In natural processes of development such as embryogenesis, local interactions and developmental time are key requirements. Biological morphogenetic systems are the result of a continuous computation, i.e. development, where intermediate phenotypes emerge along the developmental path, and these intermediate phenotypes influence the decoding and regulation of the genotype for the next phenotypic stage.

References

Darnell, J. and Doolittle, W. (1986). Speculations on the early course of evolution. *Proceedings of the National Academy of Sciences*, 83(5):1271–1275.

Kowaliw, T. and Banzhaf, W. (2012). Mechanisms for complex systems engineering through artificial development. In *Morphogenetic Engineering*, pages 331–351. Springer.

Nichele, S., Ose, M., Risi, S., and Tufte, G. (IN PRESS, 2017). Ca-neat: Evolved computational pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems*.

Pross, A. (2005). On the emergence of biological complexity: life as a kinetic state of matter. *Origins of Life and Evolution of Biospheres*, 35(2):151–166.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

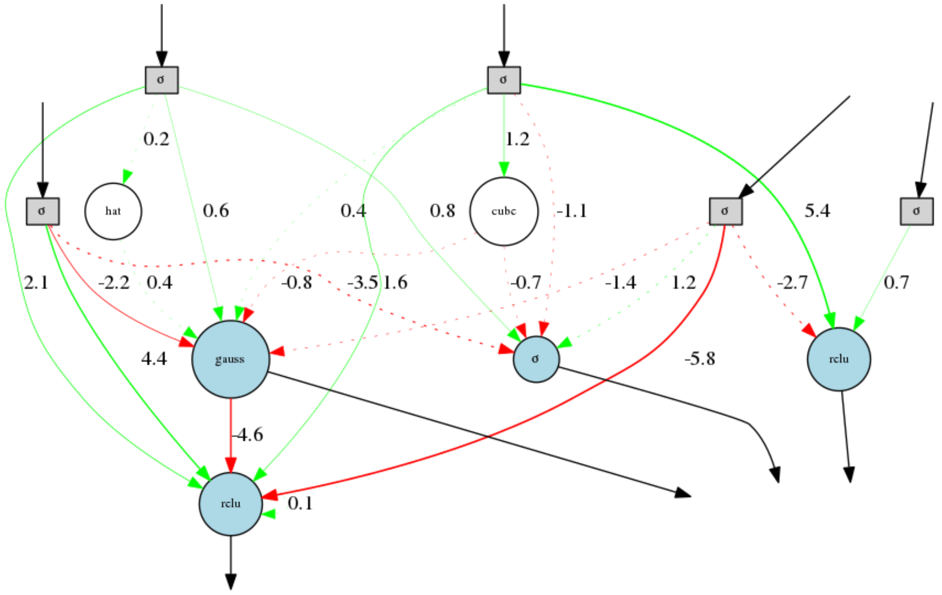


Figure 5: Network for “Tricolor” morphogenesis that reaches a point attractor equal to the target pattern. Dashed lines represent disabled connections. Green and red represent positive and negative values. The thickness represents the value intensity. Nodes can have different activation functions (sigmoid, gaussian, cube, hat, rectified linear unit, etc.) (Nichele et al., 2017).