CA-NEAT: Evolved Compositional Pattern Producing Networks for Cellular Automata Morphogenesis and Replication

Stefano Nichele, Mathias Berild Ose, Sebastian Risi, and Gunnar Tufte

***Abstract*—Cellular Automata (CA) are a remarkable example of morphogenetic system, where cells grow and self-organize through local interactions. CA have been used as abstractions of biological development and artificial life. Such systems have been able to show properties that are often desirable but difficult to achieve in engineered systems, e.g., morphogenesis and replication of regular patterns without any form of centralized coordination. However, cellular systems are hard to program (i.e., evolve) and control, especially when the number of cell states and neighborhood increase. In this paper, we propose a new principle of morphogenesis based on compositional pattern producing networks (CPPNs), an abstraction of development that has been able to produce complex structural motifs without local interactions. CPPNs are used as CA genotypes and evolved with a neuroevolution of augmenting topologies (NEATs) algorithm. This allows complexification of genomes throughout evolution with phenotypes emerging from self-organization through development based on local interactions. In this paper, the problems of 2-D pattern morphogenesis and replication are investigated. Results show that CA-NEAT is an appropriate means of approaching cellular systems engineering, especially for future applications where natural levels of complexity are targeted. We argue that CA-NEAT could provide a valuable mapping for morphogenetic systems, beyond CA systems, where development through local interactions is desired.**

***Index Terms*—Artificial life, cellular automata (CA), compositional pattern producing network (CPPN), evolutionary and developmental (EvoDevo) approach, neuroevolution of augmenting topologies (NEAT).**

I. Introduction

COMPLEX self-architecturing systems are difficult to program, i.e., by top-down engineering. Kowaliw and Banzhaf [1] and Banzhaf and Pillay [2] argued 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., an evolutionary and developmental approach (EvoDevo) [3]. Searching for a solution for an artificial EvoDevo system that targets levels of complexity found in nature can be intractable, e.g., a cellular automata (CA) system with hundreds of cellular states and large neighborhoods or a deep neural network with millions of nodes and weights. An appropriate mapping which scales well and at the same time allows solutions to be evolved incrementally, starting with a solution encoded into a small genome that is gradually complexified by adding new degrees of freedom, is desired. However, exploring a high-dimensional space in search for a solution can take prohibitively long time, regardless of the encoding [4].

In this paper, a cellular system is used as test bed of morphogenetic engineering. A traditional CA table-based encoding is replaced by a compositional pattern producing networks (CPPNs) mapping, a developmental encoding often used in systems without local interactions. In this paper, CPPN is used as developmental encoding based on local interactions, i.e., a true morphogenetic cellular system. The CA CPPNs are evolved through a neuroevolution of augmenting topologies (NEAT) algorithm, a method that evolves increasingly complex networks. The approach is termed CA-NEAT. All cells in the systems are uniform, i.e., they share the same genome network. Two benchmark problems are investigated: 1) 2-D morphogenesis and 2) replication of structures of increasing complexities.

This paper is organized as follows. Section II gives background information on CA, CPPNs, and NEAT, together with a motivation for using them together in a morphogenetic engineering system. Section III describes the investigated problems and the experimental setup. Section IV presents the results of the experiments, which are further discussed in Section V. Finally, Section VI describes some possible directions for future work and Section VII concludes this paper. Appendix A, in the supplementary material, contains some graphical visualizations of the solutions found in the outlined experiments.

Manuscript received February 10, 2017; revised June 7, 2017; accepted July 17, 2017. Date of publication August 8, 2017; date of current version September 7, 2018. *(Corresponding author: Stefano Nichele.)*

S. Nichele is with the Oslo and Akershus University College of Applied Sciences, 0130 Oslo, Norway, and also with the Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: stefano.nichele@hioa.no).

M. B. Ose and G. Tufte are with the Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: mathiabo@stud.ntnu.no; gunnar.tufte@idi.ntnu.no).

S. Risi is with the IT University of Copenhagen, 2300 Copenhagen, Denmark (e-mail: sebr@itu.dk).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2017.2737082

II. Background and Motivation

A. Cellular Automata

CA were first studied in the 1940s by von Neumann [5]. CA were inspired by biological organisms, and introduced as models that could emulate some of these organisms' interesting and remarkable properties, such as multicellular development (e.g., embryogenesis), reproduction (clonal or sexual), and robustness (e.g., self-repair).

CA have been extensively studied, in particular within the field of *artificial life* [6]. Flynn [7] has speculated that CA and cellular computing might be the path forward for novel bio-inspired computational machines. Matthew Cook proved that a CA of a certain type, e.g., Rule 110, can be Turing complete [8].

A CA consists of a grid of very simple units called cells. A cell can be in one out of a finite set of states. Sipper [9] described the three core principles of the cellular computing paradigm.

1) *Simplicity:* A cell is simple and can do very little by itself.
2) *Vast Parallelism:* The number of cells is very large, much larger than the number of processors in a conventional parallel computer.
3) *Locality:* All interactions between cells take place on a purely local basis. No cell knows or controls the entire system.

The cells in a CA use the information available from their neighbors and a set of rules to transition from one state to the next, i.e., CA transition function. Depending on the starting state of the whole system and the transition function, it is possible to observe interesting emergent or self-organizing behavior over time and space, e.g., ordered, periodic, chaotic patterns. These interesting CA often enter an *attractor* [10], [11]. If a sequence of states repeats periodically it is referred to as *cyclic attractor*, and if the CA stabilizes into a permanent state it is called a *point attractor*.

1) Transition Functions: Langton [6] formally described finite CA as consisting of a finite set of cell states Σ of size $K = |\Sigma|$, a finite input alphabet α, and a transition function Δ. Each cell has a N-sized neighborhood. The number of possible neighborhoods can be expressed by the following equation:

$$|\alpha| = |\Sigma^N| = K^N. \tag{1}$$

The transition function for a CA must thus encode a mapping of $|\alpha|$ different inputs to one of K states. The number of possible unique transition functions is thus $K^{(K^N)}$.

Traditionally Δ has been encoded as a complete mapping $\Delta : \Sigma^N \rightarrow \Sigma$, which can be implemented as a lookup table. When working with nontrivial CA where both K and N can be relatively large numbers, it becomes a problem to store the mapping Δ in an efficient way, and the space of possible Δ becomes too large to be explored by exhaustive enumeration.

Elementary CA transition functions have been studied extensively, and it has been reported that most rules lead to "uninteresting" behavior, either falling into an "order" which is either static or repeating periodically, or into chaos, where all useful information is quickly lost in noise. It has been speculated that it is in the critical border region between these behaviors where interesting computations can occur [6]. In order to find these "interesting" Δ, smart heuristic searches are often applied [12], e.g., evolutionary computation.

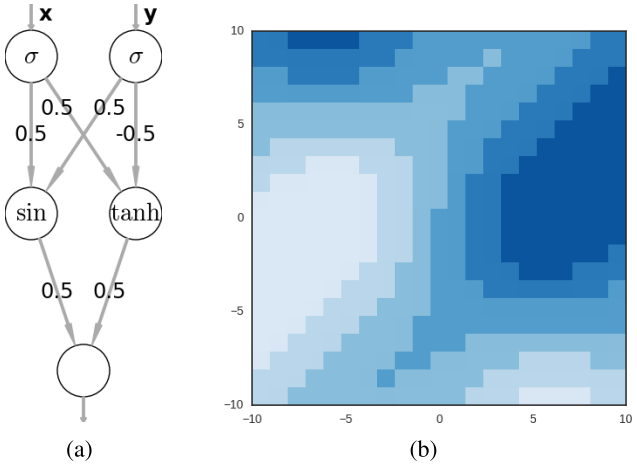


Fig. 1. Example composition of the sigmoid, sinusoid, and hyperbolic tangent functions. The discrete coordinates of (b) are first normalized to $[-1.0, 1.0]$ and then mapped to various output values through the CPPN (a).

B. CPPNs

Artificial neural networks (ANNs) [13, Ch. 1] have been used in many different applications related to artificial life and intelligence, such as robotics or machine learning. An ANN is a directed graph structure, with vertices (referred to as neurons) and edges (referred to as connections). This is inspired by neuroscience, with the brain consisting of neurons and synapse connections. ANNs are useful because they consists of many discrete parts that can be individually or collectively tuned by some adaptive process, and are easily expanded. The *universal approximation theorem* [14] states that relatively simple ANNs can approximate a wide variety of functions, and the field of deep learning [15] shows that a large complex structure with enough tuning can perform very complex tasks, such as image classification [16] or natural language processing [17]. A CPPN, which was introduced by Stanley [18], is a special type of ANN that is employed as an *artificial development encoding*. Like an ANN, a CPPN consists of a set of nodes with activation functions, weights and biases, as well as weighted connections between nodes. Likewise, external values are input to the first layer, then undergo transformation by weights and activation functions before being output by the final layer.

In contrast to ANNs, which are usually structured with neurons of the same activation functions arranged in layers, the CPPN has few such restrictions on topology and layer-wise heterogeneity and often employs a variety of different activation functions. Different activation functions are included to capture specific patterns seen in natural development, such as a Gaussian function to create symmetric patterns, or sine functions to create repeating patterns.

Additionally, CPPNs are normally applied across a broader range of possible inputs than ANNs. For example, Fig. 1 shows

a CPPN and its output when mapped over a 2-D Cartesian grid. The particular composition of functions in the CPPN produces a particular pattern, hence the name. A CPPN is able to produce a pattern without multiple steps of development, in contrast to a CA, where local interactions and time is required. CPPNs have been used both to produce patterns for the sake of the patterns, e.g., as evolutionary art [19], [20], 3-D forms [21], musical accompaniments [22], or artificial flowers [23], but also to create the neural connectivity patterns of larger ANNs for agent [24]–[27] and robot control tasks [28]–[30].

C. Artificial Evolution and Development

The bio-inspired design methods of *artificial evolution* and *artificial development* take principles from the natural processes of evolution [31], exploration and adaption of populations to environmental conditions, and development [32], the processes enabling a multicellular organism to emerge by growth and differentiation from a single cell. Artificial development and artificial evolution take inspiration from biology's EvoDevo process in order to explore and handle large and complex solution spaces.

Algorithms for artificial evolution, e.g., genetic algorithms (GAs) [33] or genetic programming [34], are based on populations of candidate solutions (*genotypes*), starting with a random generated initial population, each candidate is evaluated and assigned a *fitness*. A selection process picks individuals from the population that get to reproduce. This selection process is often stochastic, with a bias toward picking the individuals with the highest fitness, but some chance of picking a less fit individuals. Individuals that are selected for reproduction are paired up. Genotypes of the pair are combined in some fashion to create a new genotype. Random mutations are applied in order to produce new features not present in either parent. Repeating this generational algorithm the search space, i.e., the space of all solutions the genotypes can represent, is explored and exploited toward novel genotypes that encode good solution to the problem at hand.

Artificial development is an *indirect mapping* process. In contrast, in a *direct mapping* the genotype encodes the entire information of each candidate solution, i.e., *phenotype* [29]. Such indirect mappings can be inspired by biological development where an initial unit, a cell, holds the complete building plan (DNA) for an organism. It is important to note that this plan is generative, it describes how to build the system, not what the system will look like. The indirect mapping process maps genetic information in the genotype to a phenotype that expresses structure behavior and function [3]. In a system including a developmental mapping, the role of the genome is radically changed, from a system where the genome is considered a description to a system where the genome may be viewed as information on how to build the system. The "build" process can be based on self-organization governed by rules given in the genotype, e.g., gene regulation [35]. As such the phenotype is an emerging structure. Therefore, the genome size may not reflect the size or complexity of the phenotype and opens for systems that can generate very large-scale repetitive structures [36], or even structure of arbitrary size [37]. Further, a developmental mapping is not a process that is "turned off" when the finalized adult stage is reached. The process is working on the organism/artefact throughout its lifetime. Banzhaf and Miller [38] discussed the "challenge of complexity" in evolving systems and argue that nature solved such challenge by "inventing" developmental processes.

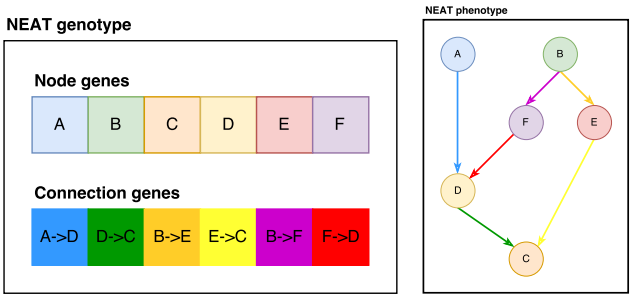


Fig. 2. Example NEAT genotype and corresponding phenotype. This example only shows the topology that the genotype encodes, leaving out the weights and activation functions.

1) NEAT: NEAT is a GA variant introduced by Stanley and Miikkulainen [39], designed specifically to evolve ANNs. Because CPPNs are special variants of ANNs, they can also be evolved with the NEAT algorithm [18]. NEAT has been applied successful to a variety of complex control tasks [40], [41].

An NEAT genome consists of genes that encode nodes and connections between them. Fig. 2 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 reflects how life on earth is believed to have started with simple organisms and gradually evolved into more complex creatures [42], [43].

The genes that make up an NEAT genome are marked with an *innovation number* so that they may be recognized as the same gene in different individuals. As new features are added to the genomes, the individuals making up the population become gradually less similar. The degree of similarity is measured through a measure called the *compatibility distance*. When the distance between individuals pass a certain threshold, they are segregated into separate species. This process is called *speciation*. Pair selection for reproduction happens within species. Typically the species that have the most fit individuals will produce more children, while the less fit species will produce fewer (but not 0) children.

When a new species appears with a new feature, the feature will not be tuned and likely affect the fitness of the individuals negatively. NEAT protects new species for a certain amount of time, allowing them time to adjust before being evaluated and, if performing poorly, being eliminated to make more room for the more fit species.

One notable variation of NEAT is called *HyperNEAT* [44]. In this process, NEAT is used to evolve CPPNs whose output determine the topology of ANNs. The indirect HyperNEAT encoding allows larger networks to be evolved with complex connectivity patterns. Additionally, because HyperNEAT can learn from the geometry of the task, it is possible to increase the number of ANN inputs and outputs without further training [45], [46]. If the evolved CPPN creates a useful network connectivity pattern at a small scale, it often also produces a useful output at a larger scale.

D. Motivation

NEATs has been shown to evolve CPPNs that produce patterns with repetitions, repetition with variation, symmetries, and different kinds of regularities, without using temporal development and local interactions. However, 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. Development is a combination of interactions between genotype and phenotypes, and with the environment. As such, natural development can be considered a dynamical systems where the phenotype changes continuously due to growth of new cells and adaptation to external perturbations, i.e., environment. Morphogenetic processes may be considered as dynamical systems with dynamical structures $(DS)^2$ [47]. In such $(DS)^2$ systems, state transition functions and the set of state variables can change over time (caused by morphogenetic processes). In this paper, we argue that CPPN is an appropriate means for developmental systems based on local interactions, which provides a mapping for the next phenotypic stage for each component of the cellular system. Such mapping uses only local information, i.e., the state of each cell and its neighbors. NEAT provides a practical evolutionary strategy for CPPN complexification.

E. Other Related Work

Wolper and Abraham [48] used evolved CPPNs to find seed patterns for Conway's Game of Life [49]. Both CPPN-NEAT (objective search) and novelty search [50] were investigated. However, CPPNs were not used as developmental encodings. Many different kinds of CA encodings have been previously investigated. These include *conditionally matching rules* [51]–[53] where conditions have to be satisfied to determine the next state of a cell, *instruction-based development* [54], [55] where transition functions are replaced by a program, *self-modifying cartesian genetic programming* [56] where a variation of genetic programming is used, and *variable length gene regulatory networks* [57]. Nichele and Tufte [58] investigated an instruction-based encoding where genomes could complexify during evolution. In [59], traditional CA transition functions are evolved through complexification. Nichele *et al.* [60] also proposed an instruction-based development with instructions that could self-modify the genome program during evolution. Cheney and Lipson [61] investigated the evolution of 3-D CA soft-robot morphologies through CPPNs. However, in this last work CPPNs make use of topological information instead of a developmental approach, as often done with CA.

III. Methodology and Experimental Setup

In order to conduct the experiments described in the following sections, a custom Python framework was developed to implement CA-NEAT. The experiments presented target problems and patterns of different complexities, allowing for direct comparison of results with [58].

Briefly described, the system consists of an evolutionary loop based on the NEAT algorithm. Each individual NEAT genotype is developed into a CPPN genotype, which is used as the transition function for a CA system. The performance of the developed phenotype for the CA problem at hand is used as fitness measure.

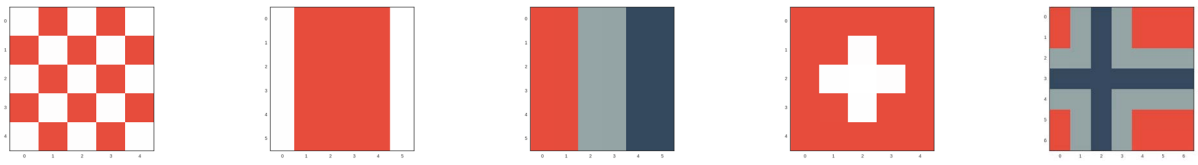
A. Problems Under Investigation

Morphogenesis and replication are two distinct yet fundamental processes in biological systems, and their complexity is not fully understood yet. For example, Hutchison *et al.* [62] have chemically synthesized a minimal bacterial genome that includes only the genes essential for sustaining life, e.g., metabolism and growth. However, out of the total 473 genes, 149 have unknown function. Self-replication in machines was first studied by Burks and Von Neumann [63] using a 29 states 2-D cellular automaton and has been a central problem in artificial life since then. Von Neumann was interested in the general question "What kind of logical organization is sufficient for an automaton to be able to reproduce itself?" In both biological and artificial cellular systems, the cell is an autonomous unit that serve as construct and constructor of the emerging organism. By being able to transfer biological properties of replication and growth in artificial systems, artificial morphogenetic systems could move toward frontiers that are not reachable by current methodologies.

For the investigation in this paper, the problems of 2-D replication and morphogenesis are chosen, as to be able to compare results with those in [58]. The five "flag" patterns shown in Fig. 3 were investigated. These patterns represent a wide variation of properties such as number of states and symmetries.

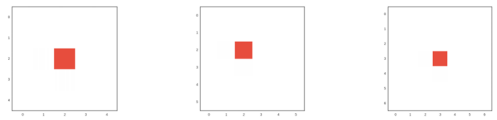
Each experiment consists of 100 independent runs. All experiments have a population size of 200 individuals and elitism degree of 1. Each generation-population is segregated into species by NEAT, with selection and reproduction happening within these groups. *Sigma scaled selection* [64] is used to select pairs for reproduction.

During development of the system, a variation of different configurations was tested for different problems. However, for the results included herein, a choice was made to use the same CPPN-NEAT configuration, and as close to the same CA configuration as possible for all problems. This makes it easier to



(a) (b) (c) (d) (e)

Fig. 3. Patterns being investigated. Each color represents a different cell state, white represents the quiescent state. (a) 5×5 *Mosaic* 2 states. (b) 6×6 *Border* 2 states. (c) 6×6 *Tricolor* 4 states. (d) 5×5 *Swiss* 2 states. (e) 7×7 *Nordic* 4 states.



(a) (b) (c)

Fig. 4. Seed patterns for morphogenesis. For the 6×6 patterns there is no central cell, so the seed is not symmetric. (a) 5×5. (b) 6×6. (c) 7×7.

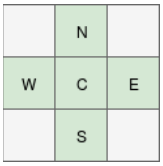


Fig. 5. Von Neumann neighborhood includes the four cardinal directions as well as the center.

make comparison between experiments, but also means that the settings chosen may favor some experiments over others. The optimization of CPPN-NEAT parameters is outside the scope of this paper.

1) Morphogenesis: The morphogenesis problem is defined as the development of a complex pattern from a simple "seed" pattern. The biological analogy and inspiration is *embryonic development*, with the seed pattern also referred to as *zygote*. Fig. 4 shows the seed patterns used in these experiments.

The fitness evaluation for a morphogenesis phenotype consists of the following steps.

1) Develop seed pattern for 30 iterations.
2) For each stage:
   a) compare cell by cell with target pattern;
   b) calculate ratio of correct out of total cells.
3) Pick max of values from step 2.
4) Use function (2) with value from step 3 as x

$$f(x) = x * \frac{e^{5*x}}{e^5}. \tag{2}$$

Function (2) is used to reduce the contribution to the score from quiescent cells, while ensuring that $f(1.0) = 1.0$. For instance, with the *Mosaic* pattern shown in Fig. 3(a) a completely quiescent pattern would have a ratio of 0.52. With the correction the fitness in this case is reduced to 0.05, which is much more appropriate for a "lifeless" CA.

Because every iteration of the CA is counted equally and separately, the fitness evaluation does not care if the CA becomes stable, enters a cycle, or neither within the 30 allotted iterations.

2) Replication: The replication problem start with one instance of some pattern in a larger grid, and over time develop into a state where multiple copies of the pattern exists in the grid, which may then replicate again. The biological analogy of this is cell division and asexual (clonal) reproduction. For the replication problem the seed pattern is thus one copy of the target pattern in a larger grid.

The fitness evaluation for a replication phenotype is as follows.

1) Develop seed pattern for 30 iterations.
2) For each stage:
   a) for each region of target pattern size:
      i) compare cell by cell with target pattern;
      ii) calculate ratio of correct out of total cells;
   b) pick max three values from step a);
   c) multiply any non-1.0 value by a penalty factor of 0.9;
   d) calculate mean of three values.
3) Pick max value from stage 2).

In this case, the number of replicas sought is three. There is no further contribution to the score if there are more than three perfect replicas. Once again a penalty is applied, this time to penalize the contribution from any imperfect replica pattern.

Compared to the evaluation of morphogenesis of the same pattern, the replication evaluation is much more computationally expensive. Therefore it will always take longer to collect results for a replication problem than the same-pattern morphogenesis problem. However, both morphogenesis and replication are properties that are present in biological systems and are highly decided in artificial morphogenetic systems.

B. Cellular Model

For both aforementioned problems, a 2-D CA model was used. For the morphogenesis problem the grid is of fixed size with toroidal border conditions. For the replication problem the grid is automatically expanding to accommodate growth in any direction. In theory this means an infinite grid, but since the CA may only iterate 30 times, there is a practical limit to how large it may grow. In both problems, the von Neumann neighborhood (Fig. 5) is used.

C. CPPN-NEAT

The nodes of the evolved CPPNs can have any of the activation functions listed in Table I.

TABLE I
POSSIBLE ACTIVATION FUNCTIONS

Type	Equation
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Hyperbolic tangent	$f(x) = tanh(x)$
Sinusoid	$f(x) = sin(x)$
Gaussian	$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$ a, b, and c are constants
Rectified linear unit	$f(x) = max(0, x)$
Identity	$f(x) = x$
Clamped	$f(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x < 1 \\ 1 & x \geq 1 \end{cases}$
Inverse	$f(x) = \frac{1}{x}$
Logarithmic	$f(x) = log(x)$
Exponential	$f(x) = e^x$
Absolute value	$f(x) = \lvert x \rvert$
Hat	$f(x) = \begin{cases} 1 - \lvert x \rvert & \lvert x \rvert < 1 \\ 0 & otherwise \end{cases}$
Square	$f(x) = x^2$
Cube	$f(x) = x^3$

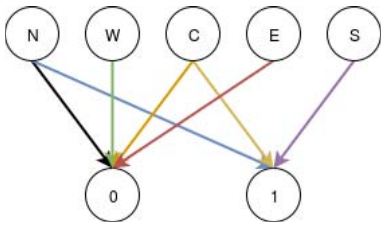


Fig. 6. Example first-generation CPPN with seven out of ten possible connections.

TABLE II
EXAMPLE NORMALIZED STATES

State	Normalised value
A	-1.0
B	-0.5
C	0.0
D	0.5

In each run of each experiment, an initial population of 200 genomes is generated. These have an input layer with one node per member of the CA neighborhood, and an output layer with one node per possible cell state. The input layer neurons have a sigmoid activation function which is never changed. When the initial population is created, the number of connections for each individual is randomly initialized to be between 50% and 100% of being fully connected. Fig. 6 illustrates this. As the evolutionary algorithm iterates on the population, new individuals come into existence that have hidden nodes and different connections and activation functions.

The inputs to the CPPNs are the values of the neighborhood, but normalized to the range $[-1.0, 1.0)$ to accommodate the sigmoid input layer. Table II shows an example of this normalization. The output of the CPPN is a set of values, one per possible CA state. The final step of the transition function finds the highest of these values, and outputs the state that correspond to that output node.

There is also a *quiescent rule* hardcoded into the transition function. One of the CA cell states is considered to be the *quiescent* or *dead* state. If all the values in the input neighborhood are quiescent, the output is always quiescent.

The speciation rule in NEAT puts the individuals of the population into separate species. Sometimes a mutated child will be so dissimilar from its parents that it will be placed in a new species. The parameter called the *stagnation limit* determines how soon a new species may be eliminated if not performing well. This value is set to 15 generations in these experiments.

D. Implementation

The system developed consists of a combination of self-made and library code. The CA subsystem was built from scratch to fit the available CPPN-NEAT implementations. It supports 1-D and 2-D topologies with various border conditions. For each problem there is a problem-specific fitness function which receives a genotype as input from the NEAT subsystem, develops the transition function and iterates the CA before evaluating the performance and returning a fitness value to the NEAT system.

The NEAT portion of the system is mostly based on the library `neat-python`.[1] Data structures for genomes and networks as well as various functions have been used without modifications. The main evolutionary loop was reimplemented with modifications. This was done for multiple reasons, including to take advantage of parallelism using `Celery`[2] and to store the results in a database using `SQLAlchemy`.[3] Other software dependencies include `matplotlib`[4] and `seaborn`[5] for visualization, and `dill`[6] for data and code serialization. The complete framework software for CA-NEAT is available on Github.[7]

IV. RESULTS

In this section, results for morphogenesis and replication of different structures are given. Each experiment consists of 100 independent runs using the same configuration, with different initial populations (randomly initialized). Each run continues the evolutionary loop until an optimal solution has been found or it is stopped when the maximum number of generations has been reached. Because of the difference in number of generations required, some experiments finish quickly, while others are executed for hundreds of generations until stopped.

Some of the optimal solutions found were visually inspected to check the correctness of the evaluation algorithms. Appendix A, in the supplementary material, contains some example solutions for various problems, selected to display a variation of behaviors. Such figures provide valuable insight on the inherent difficulty of the targeted problems.

Table III depicts the obtained results. It is particularly important to highlight the evolution over time (generations),

[1]https://github.com/CodeReclaimers/neat-python/
[2]http://celeryproject.org/
[3]http://sqlalchemy.org/
[4]http://matplotlib.org/
[5]http://seaborn.pydata.org/
[6]https://github.com/uqfoundation/dill
[7]https://github.com/mathiasose/CA-NEAT

TABLE III
SUMMARY OF RESULTS. THE METRICS SHOWN ARE THE SUCCESS RATE AND THE MEAN NUMBER OF GENERATIONS UNTIL A SOLUTION IS FOUND, WITH STANDARD DEVIATION ALSO SHOWN. IN THE CASE OF 100% SUCCESS RATE, THE NUMBER OF GENERATIONS COLUMN SHOWS HOW MANY GENERATIONS IT TOOK UNTIL THE FINAL SOLUTION WAS FOUND. IN THE CASE OF LESS THAN 100% SUCCESS RATE, THE COLUMN SHOWS HOW MANY GENERATIONS WERE RUN UNTIL THE EXPERIMENT WAS STOPPED

Problem	Success rate %	Mean generations	σ generations	Generations until stop
Mosaic morphogenesis	100	1.2	0.4	2
Border morphogenesis	1	270	0	509
Tricolor morphogenesis	100	56.5	228.8	2189
Swiss morphogenesis	76	147.7	158.9	600
Mosaic replication	100	4.2	10.6	99
Swiss replication	100	7.7	5	20
Tricolor replication	55	55.8	52.6	200
Nordic replication	0	-	-	200

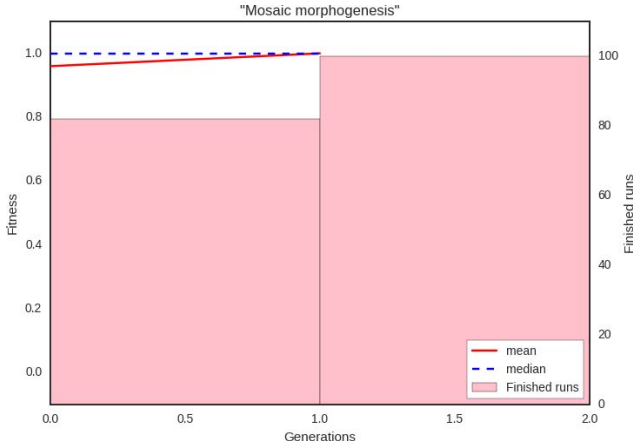


Fig. 7. *Mosaic* pattern morphogenesis, all generations.

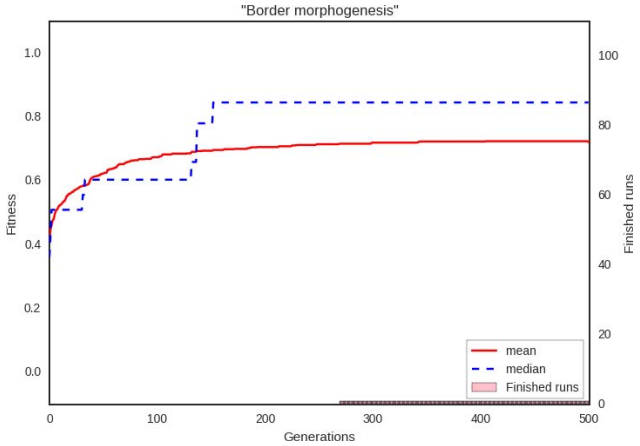


Fig. 8. *Border* pattern morphogenesis, 500 first generations. The value where the median stabilizes represents the fitness for a solution with one wrong cell.

not just the final result achieved at the end of the experiment. There are multiple metrics that can be used to visualize such data. In the figures shown in this section, three metrics have been used.

1) The *mean* of the *highest fitness* for each generation of each run.
2) The *median* of the *highest fitness* for each generation of each run.
3) The *cumulative* number of *runs that have finished* (success rate).

The choice of the *highest* value (out of 200 individuals) as representative of a whole generation, combined with the fact that the populations have elitism, means that the values shown in the graphs will only increase over time, never decrease.

Some of the figures show all generations of the experiment, until every single run has succeeded. Others are cut short, either because the experiment was stopped or to exclude statistically insignificant outlier values that makes the figure less clear. In these cases, this is clearly mentioned in text.

A. Morphogenesis

Fig. 7 shows the results of the *Mosaic* morphogenesis. In this particular case, there existed an optimal solution among the initial population in 80% of the runs. The remaining runs succeeded after one reproduction cycle.

Fig. 8 shows the results from the *Border* morphogenesis. This has a very different evolutionary process compared to the *Mosaic* morphogenesis. In this case, no optimal solution

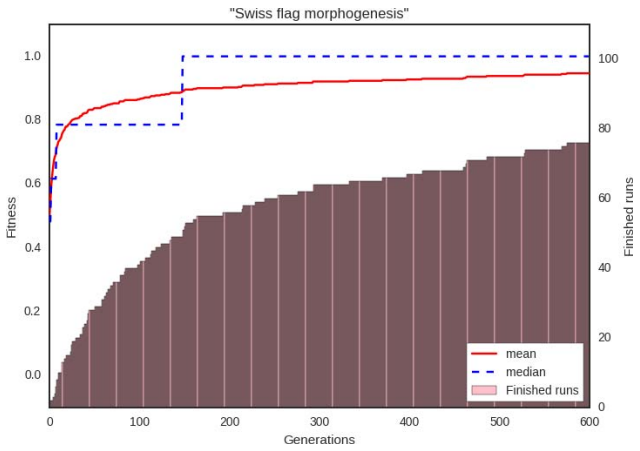


Fig. 9. *Swiss* flag pattern morphogenesis, 600 first generations.

is found for a fairly long time. The populations find local maxima solutions which have only one incorrect cell, but struggle to find the global maxima that gives the 100% correct patterns. At ca. 150 generations the median value is equal to the fitness given to a solution with one incorrect cell. After 270 generations one of the populations is successful, but after 500 generations no other population has succeeded, and the experiment is terminated.

Fig. 9 shows the results from the *Swiss* morphogenesis. From both mean line and success histogram we can see that there is an initially rapid increase that gradually diminishes. By 150 generations 50 of the runs have succeeded,

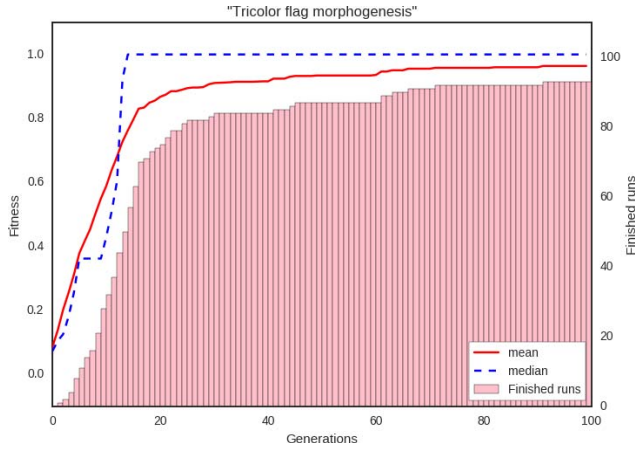


Fig. 10. *Tricolor* flag pattern morphogenesis, 100 first generations.

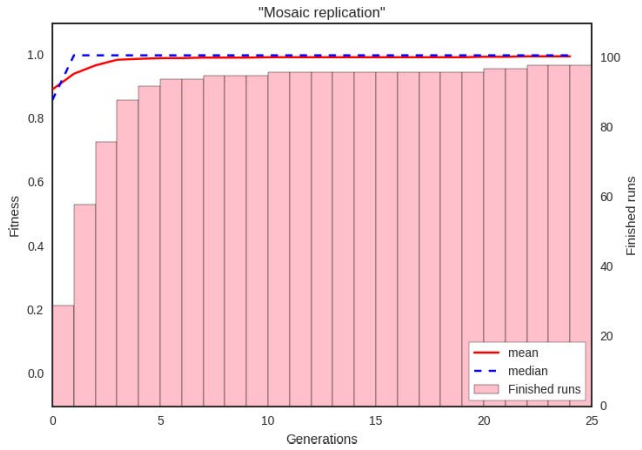


Fig. 11. *Mosaic* pattern replication, 25 first generations.

but by 300 generations only 11 more have finished, and by 600 generations the number of successful runs is 76. At this point the experiment is terminated.

Fig. 10 shows the results of the *Tricolor* morphogenesis. There is a large number of runs that complete in the first 15 generations, after which the rate of completion slows down. At 25 generations 80 runs have completed, but by 100 generations only 13 more have finished. This experiment was allowed to run to completion, but results after the first 100 generations are omitted from the figure. The last runs succeeded at generations 117, 149, 177, 302, 534, 607, and 2189.

B. Replication

Fig. 11 shows the results of the *Mosaic* replication. By 25 generations 98 runs have completed. The last two finished at 38 and 99 generations.

Fig. 12 shows the results of the *Swiss* replication. Again we see the number of successful runs increase quickly at first, before slowing down. In five generations, 50 of the runs have completed, but the last 50 succeed over the next 15 generations.

Fig. 13 show the results of the *Tricolor* replication. In this case, the first results started appearing after five generations. Like other cases there is a quick rise in finished runs early on, which drops off gradually. By 100 generations, 45 runs have

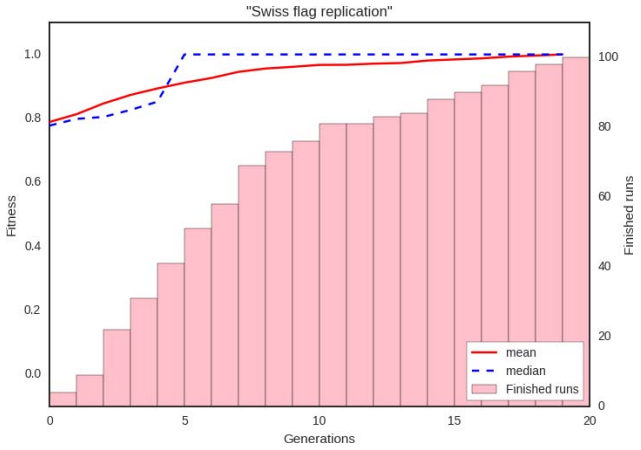


Fig. 12. *Swiss* flag pattern replication, all generations.

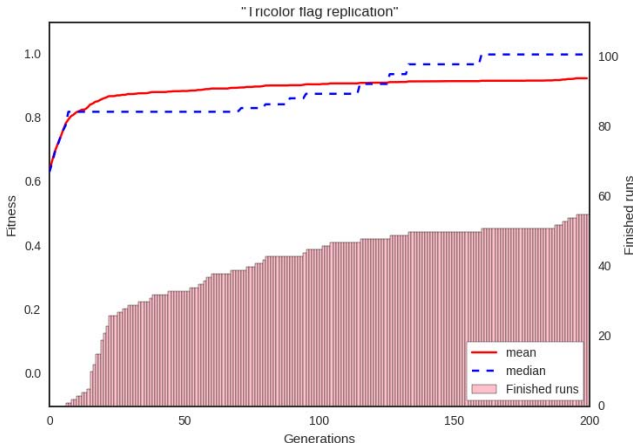


Fig. 13. *Tricolor* flag pattern replication, 200 first generations.

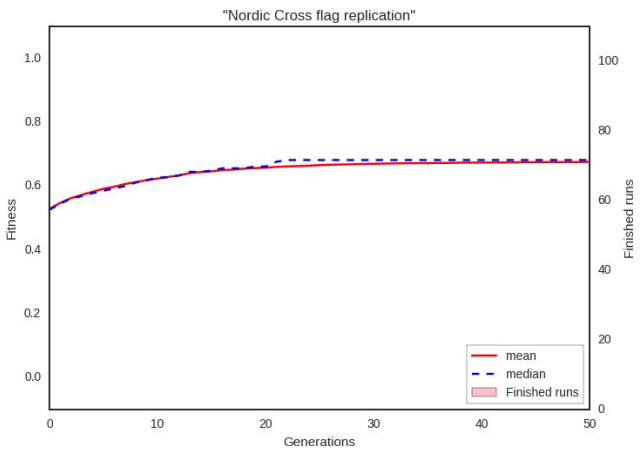


Fig. 14. *Nordic* cross pattern replication, 50 first generations. Further generations up to 200 did not have any significant change in the mean or median lines.

succeeded, and by 200 generations there are 55 finished runs, at which point the experiment is ended.

Fig. 14 shows the results of the *Nordic* replication. The populations quite quickly reach local maxima at 0.7, but are not able to find their way out of there to the global maximum. The experiment was run up to 200 generations without any further change.

TABLE IV
SIZES OF GENOMES OF OPTIMAL SOLUTIONS. GENOMES CONSIST OF NODE GENES AND CONNECTION GENES, WHICH MAY BE COUNTED CONSIDERED SEPARATELY OR COMBINED. EACH GENOME HAS A FIXED NUMBER $N + K$ INPUT AND OUTPUT NODES, PLUS SOME NUMBER (POSSIBLY 0) OF HIDDEN NODES. WHEN CONSIDERING GENOME SIZE, ONLY THE HIDDEN NODES ARE COUNTED

		Min	Max	Mean	Median	Mode(s)	σ
Mosaic morphogenesis (241 results)	Hidden nodes	0	1	0.1	0	0 (213 occurrences)	0.3
	Connections	4	11	7.1	7	6 (53 occurrences)	1.5
	Both	4	12	7.2	7	6 (49 occurrences)	1.6
Border morphogenesis (1 result)	Hidden nodes	7	7	7	7	7 (1 occurrence)	0
	Connections	16	16	16	16	16 (1 occurrence)	0
	Both	23	23	23	23	23 (1 occurrence)	0
Tricolor morphogenesis (119 results)	Hidden nodes	0	14	2	2	1 (31 occurrences)	2.1
	Connections	6	32	15.1	15	16 (20 occurrences)	4.3
	Both	6	46	17.1	17	13, 14, 18 (11 occurrences)	6
Swiss morphogenesis (61 results)	Hidden nodes	0	13	2.9	2	2 (19 occurrences)	3.1
	Connections	5	22	10.2	9.5	9, 10 (13 occurrences)	3.7
	Both	6	32	13.1	11	11 (12 occurrences)	6.5
Mosaic replication (136 results)	Hidden nodes	0	10	0.6	0	0 (81 occurrences)	1.2
	Connections	4	21	7.6	7	7 (41 occurrences)	2
	Both	4	31	8.2	8	7 (33 occurrences)	3
Swiss replication (114 results)	Hidden nodes	0	3	0.5	0	0 (63 occurrences)	0.7
	Connections	7	14	9.5	9	9 (32 occurrences)	1.5
	Both	7	16	10	10	9 (28 occurrences)	2
Tricolor replication (47 results)	Hidden nodes	0	20	4.8	4	2 (14 occurrences)	4.2
	Connections	8	41	14.7	14	8 (7 occurrences)	5.7
	Both	8	61	19.5	17	17 (5 occurrences)	9.5

C. Size of Genomes

In addition to the number of completed runs, another interesting result is the sizes of the genotypes of optimal solutions. NEAT genotypes consists of a fixed number of input and output nodes $N+K$, 0 or more hidden nodes, and some number of connections between nodes. When evaluating NEAT genotypes, it is interesting to consider these numbers both separately and combined.

Table IV shows measures of the optimal genotype sizes for each experiment. Since some runs finish with a generation where there is more than one optimal solution present, the number of optimal genotypes may be higher than the number of finished runs.

V. DISCUSSION

In some cases we observe that there is at least one optimal solution among the individuals generated as part of initial populations, e.g., *Mosaic* morphogenesis and *Mosaic* replication. This means that there exists a simple solution consisting of only the input and output layers with connections. In all cases where there exists many solutions early, we can also notice a pattern in the cumulative number of completed runs that follows a cumulative chi-squared distribution [65], with a rapid rise that gradually diminishes. The most extreme of these cases is the *Mosaic* morphogenesis where 80 runs complete in the initial generation and the last 20 in the second generation. This result can be explained by the symmetry and repetitiveness in the target pattern, and CPPN has been shown particularly successful when targeting morphologies with such properties. However, CPPNs usually exploit topological information to achieve symmetry and regularity. In the work herein, the same result is achieved without any available topological information, and only through developmental processes based on local interactions, i.e., a self-organizing morphogenetic behavior. For more complex patterns, more generations are obviously required in order to evolve and complexify the networks in the initial population.

It is somewhat surprising which problems are easily solved and which ones are not. We can observe that the *Swiss* replication is much easier than the *Swiss* morphogenesis. This is in line with results in [55] and [60], when instruction-based development or conditionally matching rules are used. However, the *Tricolor* morphogenesis is easier than the replication of the same pattern, as it is often the case when CA transition functions are used (obviously with worst results). The fact that the *Border* morphogenesis is more difficult than the *Tricolor* morphogenesis might seem not intuitive, since the *Border* pattern has both fewer colors (states) and one more symmetry. Fig. 15 shows an example of the imperfect patterns produced by CA-NEAT for the *Border* morphogenesis. One plausible explanation is that the symmetry is deceptive and leads to a local maxima, while the *Tricolor* experiment avoids this, since symmetry in solutions will not give better scores in such case. Novelty search [50] might be explored for deceptive tasks. This is further discussed in Section VI.

CA-NEAT does quite well for three out of four replication problems, but does not succeed at the *Nordic* replication. This problem was expected to be difficult, since the pattern is rather complex (shifted symmetries). However, instruction-based encoding in [58] found solutions for this task. One possible solution for this problem would be to exploit CPPN

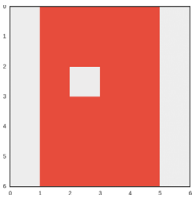


Fig. 15. *Border* pattern with only one wrong cell. CA-NEAT manages to find a pattern like this for the majority of the runs by 150 generations, but struggles to find the 100% correct pattern.

ability to produce such patterns when topological information is available to the evolved networks. This is discussed further in Section VI.

As mentioned earlier, optimization of NEAT parameters was outside the scope of this experimentation. As such, populations were initialized without hidden nodes. Alternatives could be to initialize the population with one or more hidden nodes, a larger CA neighborhood, and an increased NEAT mutation rate to encourage innovation. This is also discussed further in Section VI.

A. Comparison With Literature

The patterns and structures investigated herein are widely used benchmarks in [66] and [67]. For example, in [58] similar problems are investigated with an instruction-based encoding, as well as a table-based encoding for comparison.

When comparing results, it is important to consider the differences in the experimental setups. In particular, the populations in NEAT have to be much larger to allow speciation. This gives CA-NEAT an advantage in cases such as the *Mosaic* morphogenesis where a large initial population is likely to contain an optimal solution. Conversely, it means it takes longer for each generation of CA-NEAT, so results that require development over generations may be found *faster* with smaller populations. In [58], all populations ran up to 10 000 generations before being terminated.

1) Morphogenesis: For the *Mosaic* morphogenesis, table-based evolution has a success rate of 58% and instruction-based 98%. With CA-NEAT this rate was 100%. The table-based and instruction-based evolutions took on average 1336 and 1257 generations, respectively, while the NEAT search had found all solutions in two generations. This is mostly explained by the large NEAT population and relatively simple target pattern.

For the *Swiss* morphogenesis, table-based evolution had a success rate of 23% and instruction-based 100%. The average generations until solution was 2668 for table-based and 285 for instruction-based. CA-NEAT results seems comparable to the instruction-based results, with a 76% success rate by generation 600, and an average generations of 147.7 so far, in the same order of magnitude as the instruction-based result.

There is a stark contrast between the results of the *Border* morphogenesis, where table-based evolution has a 69% success rate and instruction-based 98%. CA-NEAT has a 1% success rate at 500 generations.

For the *Tricolor* morphogenesis, table-based evolution has a 19% success rate and instruction-based evolution has a 46% success rate. CA-NEAT improves both of these with a large margin, with a success rate of 92% at 100 generations, 99% at 607 generations, and finally 100% at 2189 generations. CA-NEAT has an average number of generations at 56.5, compared to 5002 for table-based and 6424 for instruction-based.

2) Replication: For the *Mosaic* replication, table-based evolution has a success rate of 85% and instruction-based 100%. CA-NEAT also has a 100% success rate and average generations of 4.2, compared to 39.7 for instruction-based encoding, a significantly better result.

For the *Swiss* replication, table-based evolution has a success rate of 1% and instruction-based a 100% rate. CA-NEAT has a 100% success rate too, with 7.7 average generations, compared to the 41.8 of the instruction-based evolution, also a significant difference.

For the *Tricolor* replication, table-based evolution has a success rate of 8%, and instruction-based a 100% rate. CA-NEAT has a 45% success rate at 100 generations, and an average generations of 34.6 at that point. Compared to the instruction-based average of 41.8 generations after all runs, the performance of CA-NEAT seems lower in this case.

B. Size and Topology of CPPN Phenotypes

In all the experiments where multiple solutions were found, there existed at least one solution with no hidden nodes. This indicates that a CPPN-based encoding *can* be efficient at those particular problems. However, it has not been determined whether these small CPPNs encode solutions that are fast at morphogenesis, fast at replicating, or how many replicas they produce, nor how many steps of CA development they require.

Some of the found CPPNs were visually inspected to try to understand their topologies. Fig. 16 shows two of these visualizations. In solutions with many hidden nodes, the structure seems disorderly to a human, and it is difficult to understand the relationship between input and output based on topology alone.

In solutions with hidden nodes, it is often possible to see some nodes that are not connected to the output nodes. These are a kind of *vestigial structure* [68]. In a final solution these could be pruned away to reduce the genome size without changing CA behavior. During evolution these should probably be left in place, as they could be reconnected by mutations and possibly produce positive effects.

VI. Future Work

There are many aspects of the CA-NEAT model that have not yet been tested or analyzed. In the remainder of this sections, possible directions for future work are analyzed and discussed.

A. Other Morphogenetic Engineering Problems

The framework developed for this project is quite generic and can handle a variety of problems already. Code that is specific to a problem is contained in the fitness function for that particular problem. In order to test a new class of problems, it would be enough to use a new fitness evaluation function.

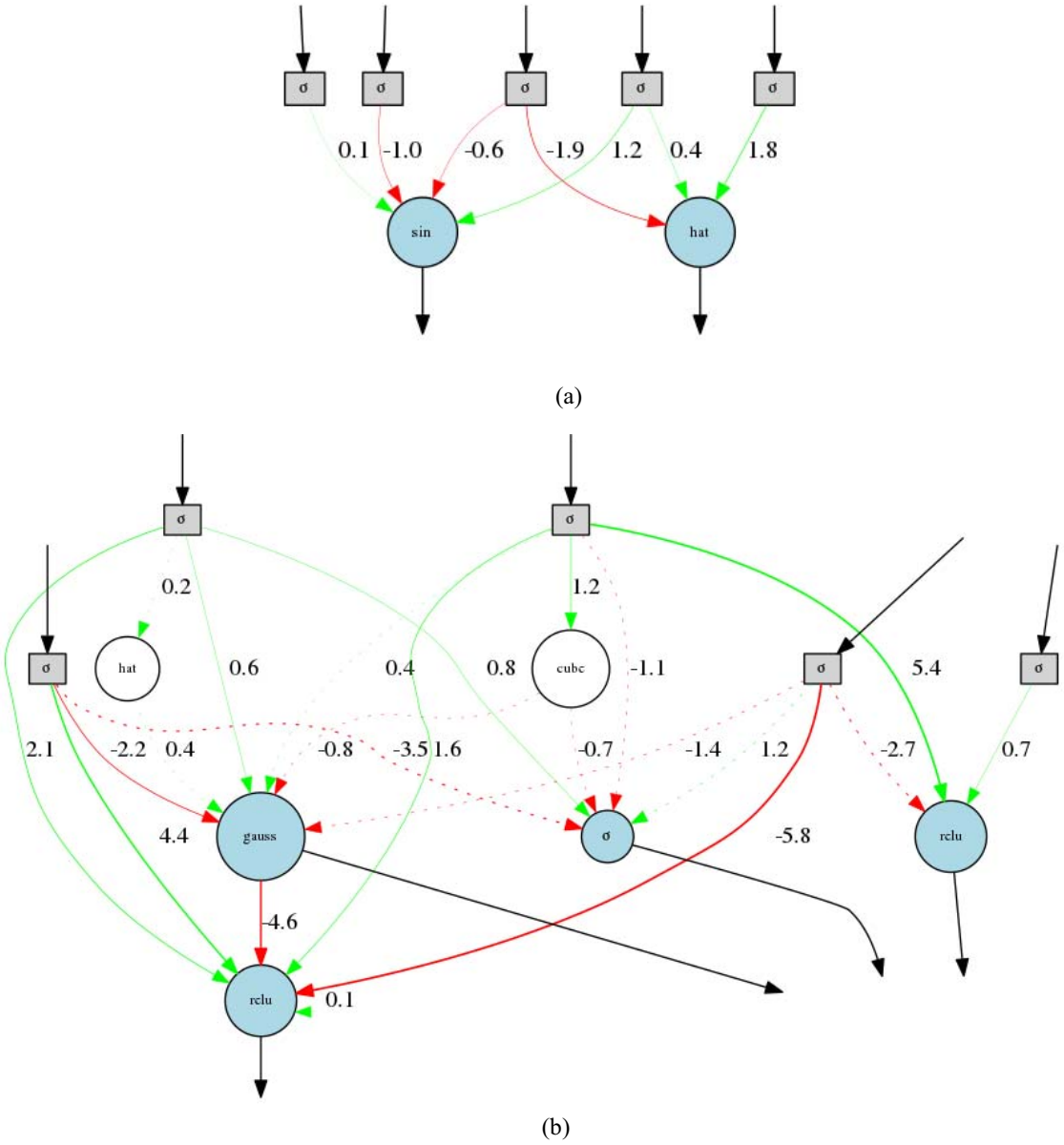


Fig. 16. Examples of found CPPN encodings. Dashed lines represent disabled connections. The visualisation library optimizes the figures to have few crossing connections, and does not care about presenting nodes as structured layers. (a) *Mosaic* replication along one axis, like Fig. A.14, in the supplementary material. (b) *Tricolor* morphogenesis that reaches a point attractor equal to the target pattern. The two hidden nodes are not connected to output nodes and are thus "vestigial." See Fig. A.10, in the supplementary material, for the CA development of this phenotype.

The experiments presented in this paper concern development and replication of 2-D patterns, as proof of concept that CA-NEAT is a promising avenue for morphogenetic engineering and self-assembly of complex structures and morphologies through local interactions. Morphogenetic systems can be considered very powerful and decentralized computing machines, where computation and memory are totally distributed and the actual computation is a result of self-organization and emergence. There are many other morphogenetic computational problems that could be explored, e.g., the majority problem [69], the firing squad synchronization problem [70], or mathematical and algorithmic problems, such as square calculation [71]. Another variation of replication problem, known as *replicating loops* [72], as well as morphogenesis in 3-D space [73] are target problems for future research.

B. Fitness Evaluation Variations

Over time NEAT will keep expanding genomes, increasing the number of nodes and connections of the CPPN topology. When comparing two CPPNs that produce the same CA rules, it is the less complex CPPN that is most desirable, due to its lower memory footprint and running time. The fitness evaluation function could be amended to reward smaller genomes, encouraging this.

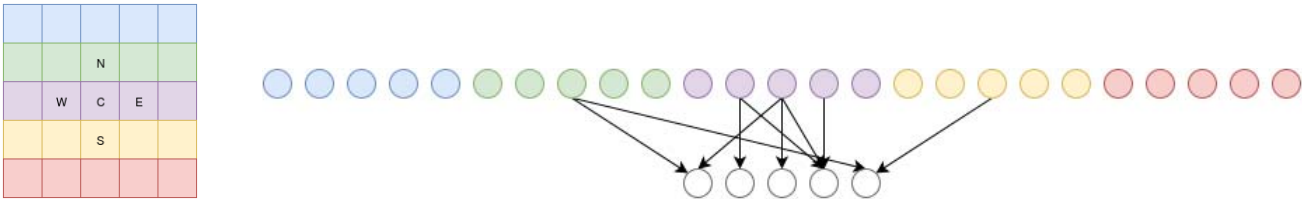


Fig. 17. Example CPPN with a large available input neighborhood, but only the Von Neumann subneighborhood connected to the output layer.

Another aspect that could be considered for certain problems, such as morphogenesis, is that of attractors. For the morphogenesis problem, the arguably best result is to find the shortest possible attractor that contains the target state, preferably a point attractor. One option could be to introduce a penalizing factor inversely proportional to the length of the cycle. In a replication problem, one might consider the emergence of a new copy of the original pattern as the repeating of a cycle, and thus reward a quicker replication in such case.

C. Variations of Cellular Model

The CA-NEAT experiments herein have shown different degrees of success. However, the NEAT parameters have not been optimized and the same settings have been used for all the experiments. An exploration of various parameters and relative performance is desired in future research, as to be able to pinpoint suitable setting for such morphogenetic engineering systems.

An interesting idea for further work is to allow evolution to optimize the neighborhood definition. The neighborhood radius could be expanded easily with CPPN, as it would simply be an addition of new input nodes. This would create much more diversity in the initial population, and possibly create genotypes that are more complex for harder problems. Fig. 17 proposes one implementation of larger neighborhoods, where there are many input nodes available, but only the closest ones are connected in the initial population. When mutating new genotypes, new inputs could be reconnected, and over time evolution would determine if such innovation was beneficial and worth to be retained.

It would be also trivial to include environmental information in the form of one or more additional CPPN input nodes. For example, each cell could know something about the physics of the CA world. e.g., its position (coordinates or distance from the origin coordinate), or chemicals in the environment. The addition of an abstraction layer on top of the CA layer (e.g., chemicals layer) has been shown beneficial for cellular systems. In [74], a French flag organism was shown to possess beneficial properties for morphogenetic systems, such as self-repair and self-regulation, as a result of local interactions with neighboring cells as well as with the environment (chemicals). Biological cells have different means of receiving positional information [75]. Topological information has been shown to be a key contributor to CPPN-based solutions, even without local interactions. It is therefore reasonable to imagine that CA-NEAT would also benefit by including positional information as available input to the CPPN, in the form of coordinates or gravity (direction). We envision that complex solution will be easily achieved with CA-NEAT through development based on local interactions with the addition of topological information.

D. Novelty Search

In [48], it was shown that novelty search in combination with NEAT produced some interesting results where objective search did not. Considering that the objective-based search did not entirely succeed at finding solutions to some of the deceptive problems in this paper, novelty search could provide a different strategy of exploring the solution space without getting trapped in local maxima. In particular, one challenge is giving fitness scores to intermediate nonoptimal solutions in such a way as to reward the solutions that will eventually lead to optimal solutions and avoid rewarding "dead ends." Novelty search [50] attempts to solve the problem by disregarding the objective score and instead rewarding phenotypes that exhibit previously unseen behavior.

VII. Conclusion

In this paper we presented a novel method for self-organization of morphogenetic cellular systems based on development through CPPNs. CPPNs are used as developmental mappings that take advantage of local interactions. CPPNs have been evolved with the NEATs algorithm and CA have been used as the experimental platform. One of the main issues of morphogenetic engineering systems, and CA in particular, is the difficulty of programmability and control when the number of components, their types (or states) and their local interaction neighborhoods become larger (toward systems at complexity levels found in nature). CPPNs provide an appropriate mapping that scales well in all these cases, e.g., linear increase of input CPPN nodes when the neighborhood is increased as opposed to CA transition functions that would grow exponentially, or linear increase of CPPN output nodes when the number of states is increased as opposed to CA transition functions that would grow exponentially. The CA-NEAT morphogenetic framework has been tested on two different problems, the development of structures from a seed and the replication of structures of increasing complexities. The presented results have shown promise in most of the experiments, considering that NEAT parameters were not optimized as it was outside the scope of this paper. We suggest that the natural way forward is to incorporate topological/positional information in CA-NEAT, as CPPNs have

been proven successful even with development without local interactions. We argue that CA-NEAT could provide a valuable EvoDevo approach to self-organizing decentralized control and programmability of morphogenetic systems.

REFERENCES

[1] T. Kowaliw and W. Banzhaf, "Mechanisms for complex systems engineering through artificial development," in *Morphogenetic Engineering*. Heidelberg, Germany: Springer, 2012, pp. 331–351.

[2] W. Banzhaf and N. Pillay, "Why complex systems engineering needs biological development," *Complexity*, vol. 13, no. 2, pp. 12–21, 2007.

[3] B. K. Hall, "Unlocking the black box between genotype and phenotype: Cell condensations as morphogenetic (modular) units," *Biol. Philos.*, vol. 18, no. 2, pp. 219–247, 2003.

[4] K. O. Stanley and R. Miikkulainen, "Achieving high-level functionality through complexification," in *Proc. AAAI Spring Symp. Comput. Synth.*, 2003, pp. 226–232.

[5] J. von Neumann, *Theory of Self-Reproducing Automata*, A. W. Burks, Ed. Champaign, IL, USA: Univ. Illinois Press, 1966.

[6] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Phys. D Nonlin. Phenomena*, vol. 42, nos. 1–3, pp. 12–37, 1990.

[7] M. J. Flynn, "Parallel processors were the future and may yet be," *IEEE Comput.*, vol. 29, no. 12, p. 152, 1996.

[8] M. Cook, "Universality in elementary cellular automata," *Complex Syst.*, vol. 15, no. 1, pp. 1–40, 2004.

[9] M. Sipper, "The emergence of cellular computing," *Comput. Mag.*, vol. 32, no. 7, pp. 18–26, 1999.

[10] C. Gershenson, "Introduction to random Boolean networks," in *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, M. Bedau, P. Husbands, T. Hutton, S. Kumar, and H. Suzuki, Eds. Cambridge, MA, USA: MIT Press, 2004, pp. 160–173.

[11] S. Wolfram *et al.*, *Theory and Applications of Cellular Automata*, vol. 1. Singapore: World Sci., 1986.

[12] G. Tufte and S. Nichele, "On the correlations between developmental diversity and genomic composition," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, Dublin, Ireland, 2011, pp. 1507–1514.

[13] Y. Bengio, I. J. Goodfellow, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2015. [Online]. Available: http://www.iro.umontreal.ca/bengioy/dlbook

[14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608089900208

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[17] R. Collobert *et al.*, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Aug. 2011.

[18] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genet. Program. Evol. Mach.*, vol. 8, no. 2, pp. 131–162, 2007.

[19] K. O. Stanley, "Exploiting regularity without development," in *Proc. AAAI Fall Symp. Develop. Syst.*, Menlo Park, CA, USA, 2006, p. 37.

[20] J. Secretan *et al.*, "Picbreeder: A case study in collaborative evolutionary exploration of design space," *Evol. Comput.*, vol. 19, no. 3, pp. 373–403, 2011.

[21] J. Clune and H. Lipson, "Evolving three-dimensional objects with a generative encoding inspired by developmental biology," in *Proc. Eur. Conf. Artif. Life*, 2011, pp. 144–148. [Online]. Available: http://EndlessForms.com

[22] A. K. Hoover *et al.*, "Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding," in *Proc. Int. Conf. Comput. Creativity*, 2012, p. 111.

[23] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Petalz: Search-based procedural content generation for the casual gamer," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 3, pp. 244–255, Sep. 2016.

[24] D. B. D'Ambrosio and K. O. Stanley, "Generative encoding for multiagent learning," in *Proc. 10th Annu. Conf. Genet. Evol. Comput.*, Atlanta, GA, USA, 2008, pp. 819–826.

[25] D. B. D'Ambrosio and K. O. Stanley, "Scalable multiagent learning through indirect encoding of policy geometry," *Evol. Intell.*, vol. 6, no. 1, pp. 1–26, 2013.

[26] J. Drchal, J. Koutník, and M. Snorek, "HyperNEAT controlled robots learn how to drive on roads in simulated environment," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Trondheim, Norway, 2009, pp. 1087–1092.

[27] S. Risi and K. O. Stanley, "A unified approach to evolving plasticity and neural geometry," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Brisbane, QLD, Australia, 2012, pp. 1–8.

[28] S. Risi and K. O. Stanley, "Confronting the challenge of learning a flexible neural controller for a diversity of morphologies," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, Amsterdam, The Netherlands, 2013, pp. 255–262.

[29] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria, "On the performance of indirect encoding across the continuum of regularity," *IEEE Trans. Evol. Comput.*, vol. 15, no. 3, pp. 346–367, Jun. 2011.

[30] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the hyperneat generative encoding," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Trondheim, Norway, 2009, pp. 2764–2771.

[31] C. Darwin, *On the Origin of Species by Means of Natural Selection*. London, U.K.: John Murray, 1859.

[32] L. Wolpert, *Principles of Development*, 2nd ed. Oxford, U.K.: Oxford Univ. Press, 2002.

[33] D. E. Goldberg, *Genetic Algorithms in Search Optimization & Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

[34] J. R. Koza, *Genetic Programming*. Cambridge, MA, USA: MIT Press, 1993.

[35] M. Lantin and F. D. Fracchia, "Generalized context-sensitive cell systems," in *Proc. 1st Int. Workshop Inf. Process. Cells Tissues*, 1995, pp. 1–14.

[36] H. Kitano, "Building complex systems using developmental process: An engineering approach," in *Evolvable Systems: From Biology to Hardware* (Lecture Notes in Computer Science). Heidelberg, Germany: Springer, 1998, pp. 218–229.

[37] L. Sekanina and M. Bidlo, "Evolutionary design of arbitrarily large sorting networks using development," *Genet. Program. Evol. Mach.*, vol. 6, no. 3, pp. 319–347, 2005.

[38] W. Banzhaf and J. Miller, "The challenge of complexity," in *Frontiers of Evolutionary Computation*. Boston, MA, USA: Kluwer Acad., 2004, pp. 243–260.

[39] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[40] T. Aaltonen *et al.*, "Measurement of the top-quark mass with dilepton events selected using neuroevolution at CDF," *Phys. Rev. Lett.*, vol. 102, no. 15, 2009, Art. no. 152001.

[41] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *J. Artif. Intell. Res.*, vol. 21, pp. 63–100, Feb. 2004.

[42] J. E. Darnell and W. E. Doolittle, "Speculations on the early course of evolution," *Proc. Nat. Acad. Sci. USA*, vol. 83, no. 5, pp. 1271–1275, 1986.

[43] A. Pross, "On the emergence of biological complexity: Life as a kinetic state of matter," *Origins Life Evol. Biospheres*, vol. 35, no. 2, pp. 151–166, 2005.

[44] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, pp. 185–212, 2009.

[45] J. Gauci and K. O. Stanley, "Indirect encoding of neural networks for scalable go," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, Kraków, Poland, 2010, pp. 354–363.

[46] B. G. Woolley and K. O. Stanley, "Evolving a single scalable controller for an octopus arm with a variable number of segments," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, Kraków, Poland, 2010, pp. 270–279.

[47] A. Spicher, O. Michel, and J.-L. Giavitto, "A topological framework for the specification and the simulation of discrete dynamical systems," in *Proc. Int. Conf. Cellular Automata*, Amsterdam, The Netherlands, 2004, pp. 238–247.

[48] J. Wolper and G. Abraham, "Evolving novel cellular automaton seeds using compositional pattern producing networks (CPPN)," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2016, pp. 27–28.

[49] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways, for Your Mathematical Plays: Games in Particular*, vol. 2. London, U.K.: Academic Press, 1982,

[50] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proc. ALIFE*, 2008, pp. 329–336.
[51] M. Bidlo and Z. Vasicek, "Evolution of cellular automata with conditionally matching rules," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 1178–1185.
[52] M. Bidlo, "Investigation of replicating tiles in cellular automata designed by evolution using conditionally matching rules," in *Proc. IEEE Symp. Comput. Intell.*, Cape Town, South Africa, 2015, pp. 1506–1513.
[53] M. Bidlo, "On routine evolution of new replicating structures in cellular automata," in *Proc. 7th Int. Conf. Evol. Comput. Theory Appl. (SCITEPRESS)*, Lisbon, Portugal, 2015, pp. 28–38.
[54] M. Bidlo and J. Škarvada, "Instruction-based development: From evolution to generic structures of digital circuits," *Int. J. Knowl. Based Intell. Eng. Syst.*, vol. 12, no. 3, pp. 221–236, 2008.
[55] M. Bidlo and Z. Vasicek, "Evolution of cellular automata using instruction-based approach," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[56] S. L. Harding, J. F. Miller, and W. Banzhaf, "Self-modifying Cartesian genetic programming," in *Cartesian Genetic Programming*. Heidelberg, Germany: Springer, 2011, pp. 101–124.
[57] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell, "On the advantages of variable length GRNs for the evolution of multicellular developmental systems," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp. 100–121, Feb. 2013.
[58] S. Nichele and G. Tufte, "Evolutionary growth of genomes for the development and replication of multicellular organisms with indirect encoding," in *Proc. IEEE Int. Conf. Evol. Syst. (ICES)*, Orlando, FL, USA, 2014, pp. 141–148.
[59] S. Nichele, A. Giskeødegård, and G. Tufte, "Evolutionary growth of genome representations on artificial cellular organisms with indirect encodings," *Artif. Life*, vol. 22, no. 1, pp. 76–111, 2016.
[60] S. Nichele, T. E. Glover, and G. Tufte, "Genotype regulation by self-modifying instruction-based development on cellular automata," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, Edinburgh, U.K., 2016, pp. 14–25.
[61] N. Cheney and H. Lipson, "Topological evolution for embodied cellular automata," *Theor. Comput. Sci.*, vol. 633, pp. 19–27, Jun. 2016.
[62] C. A. Hutchison *et al.*, "Design and synthesis of a minimal bacterial genome," *Science*, vol. 351, no. 6280, Mar. 2016. [Online]. Available: http://science.sciencemag.org/content/351/6280/aad6253
[63] A. W. Burks and J. Von Neumann, *Theory of Self-Reproducing Automata*. Urbana, IL, USA: Univ. Illinois Press, 1966.
[64] P. J. B. Hancock, "An empirical comparison of selection methods in evolutionary algorithms," in *Proc. AISB Workshop Evol. Comput.*, Leeds, U.K., 1994, pp. 80–94.
[65] H. O. Lancaster and E. Seneta, *Chi-Square Distribution*. Hoboken, NJ, USA: Wiley, 1969.
[66] D. Federici and K. Downing, "Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification," *Artif. Life*, vol. 12, no. 3, pp. 381–409, 2006.
[67] J. F. Miller and W. Banzhaf, "Evolving the program for a cell: From French flag to Boolean circuits," in *On Growth, Form and Computers*, S. Kumar and P. J. Bentley, Eds. Oxford U.K.: Elsevier, 2003, pp. 278–301.
[68] G. Muller, "Vestigial organs and structures," in *Encyclopedia of Evolution*, vol. 2. Oxford, U.K.: Oxford Univ. Press, 2002, pp. 1131–1133.
[69] M. Mitchell, J. P. Crutchfield, and R. Das, "Evolving cellular automata with genetic algorithms: A review of recent work," in *Proc. 1st Int. Conf. Evol. Comput. Appl. (EvCA)*, Moscow, Russia, 1996, pp. 42–55.
[70] A. Waksman, "An optimum solution to the firing squad synchronization problem," *Inf. Control*, vol. 9, no. 1, pp. 66–78, 1966.
[71] M. Bidlo, "On routine evolution of complex cellular automata," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 742–754, Oct. 2016.
[72] C. G. Langton, "Self-reproduction in cellular automata," *Phys. D Nonlin. Phenomena*, vol. 10, nos. 1–2, pp. 135–144, 1984.
[73] S. K. Semwal and K. Chandrashekhar, "Cellular automata for 3D morphing of volume data," in *Proc. Int. Conf. Central Europe Comput. Graph. Visual. Comput. Vis.*, 2005, pp. 195–202.
[74] J. F. Miller, "Evolving a self-repairing, self-regulating, French flag organism," in *Proc. Genet. Evol. Comput. Conf.*, Seattle, WA, USA, 2004, pp. 129–139.
[75] A. D. Lander, "How cells know where they are," *Science*, vol. 339, no. 6122, pp. 923–927, 2013.

Stefano Nichele received the master's degree from the University of Insubria, Varese, Italy, in 2009, and the Ph.D. degree from the Norwegian University of Science and Technology, Trondheim, Norway, in 2015, both in computer science.

He is an Associate Professor with the Department of Computer Science, Oslo and Akershus University College of Applied Sciences, Oslo, Norway. He was a Post-Doctoral Researcher with the Norwegian University of Science and Technology, and a Visiting Researcher with the IT University of Copenhagen, Copenhagen, Denmark. His current research interests include complex systems, evolutionary and developmental systems, artificial life, biological and artificial neural networks, and Internet of Things.

Mathias Berild Ose received the master's degree in computer science from the Norwegian University of Science and Technology, Trondheim, Norway, in 2017.

He is currently with the private sector of natural language processing and machine learning development at convertelligence.com, Oslo, Norway.

Sebastian Risi received the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA.

He is an Associate Professor with the Department of Computer Science, IT University of Copenhagen, Copenhagen, Denmark, where he is part of the Center for Computer Games Research and the Robotics, Evolution and Art Laboratory. He was the Co-Founder of FinchBeak, Orlando, FL, USA, a company that focused on casual and educational social games enabled by AI technology, and a Consultant with Uber AI Laboratories, San Francisco, CA, USA. His current research interests include neuroevolution, evolutionary robotics, and human computation.

Dr. Risi was a recipient of several best paper awards at the Genetic and Evolutionary Computation Conference, International Conference on Computational Intelligence in Music, Sound, Art and Design, International Joint Conference on Neural Networks, and the Continual Learning Workshop at Neural Information Processing Systems Conference for his work on adaptive systems, the HyperNEAT algorithm for evolving complex artificial neural networks, and music generation.

Gunnar Tufte received the master's and Ph.D. degrees in computer and information science from the Norwegian University of Science and Technology, Trondheim, Norway, in 1999 and 2004, respectively.

He is a Professor with the Department of Computer and Information Science, Norwegian University of Science and Technology, where he leads the NTNU morphogenetic engineering enabling technology initiative. His current research interests include biologically inspired machines, artificial life, evolutionary and developmental systems, cellular computing, and evolution-in-materio.