

Evolutionary morphogenesis for multi-cellular systems

Daniel Roggen · Diego Federici · Dario Floreano

Received: 29 August 2005 / Revised: 27 October 2006 /
Published online: 28 December 2006
© Springer Science + Business Media, LLC 2007

Abstract With a gene required for each phenotypic trait, direct genetic encodings may show poor scalability to increasing phenotype length. Developmental systems may alleviate this problem by providing more efficient indirect genotype to phenotype mappings. A novel classification of multi-cellular developmental systems in evolvable hardware is introduced. It shows a category of developmental systems that up to now has rarely been explored. We argue that this category is where most of the benefits of developmental systems lie (e.g. speed, scalability, robustness, inter-cellular and environmental interactions that allow fault-tolerance or adaptivity). This article describes a very simple genetic encoding and developmental system designed for multi-cellular circuits that belongs to this category. We refer to it as the *morphogenetic system*. The morphogenetic system is inspired by gene expression and cellular differentiation. It focuses on low computational requirements which allows fast execution and a compact hardware implementation. The morphogenetic system shows better scalability compared to a direct genetic encoding in the evolution of structures of differentiated cells, and its dynamics provides fault-tolerance up to high fault rates. It outperforms a direct genetic encoding when evolving spiking neural networks for

D. Roggen (✉) · D. Floreano
Laboratory of Intelligent Systems, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
e-mail: droggen@gmail.com

D. Floreano
e-mail: dario.floreano@epfl.ch

D. Federici
Department of Computer and Information Sciences, Norwegian University of Science and Technology,
Trondheim, Norway
e-mail: diego.federici@gmail.com

Present address:

D. Roggen
Wearable Computing Laboratory, ETH Zürich, Zürich, Switzerland

D. Federici
Google's Zürich European Engineering Centre, Switzerland

pattern recognition and robot navigation. The results obtained with the morphogenetic system indicate that this “minimalist” approach to developmental systems merits further study.

Keywords Evolutionary computation · Developmental system · Genotype to phenotype mapping · Evolvable hardware · Neural network

1 Introduction

Evolvable hardware (EHW) is a new approach to the creation of electronic circuits that has been explored in the last 10 years [34]. It consists in using evolutionary algorithms (e.g. genetic algorithms) to create electronic circuits. This approach showed that novel or more efficient circuits than those obtained with traditional techniques can be found [9, 79, 86]. EHW is believed to have a lot of potential [9], for instance in adaptive hardware [39], or in fault-tolerant hardware [43], and it gains support in the industrial community [35].

However, the complexity of evolved circuits remains limited when using direct genetic encodings because the size of the genetic string grows with the size of the phenotype and leads to large search spaces [32, 40, 42, 87]. This problem is also encountered when evolving neural networks [9].¹

One possible solution may be to use an indirect genetic encoding which takes the form of a developmental process [45, 91]. A small number of “instructions” in the genotype encodes for a larger phenotype and by consequence the size of the search space is reduced.

Evolvability may be improved if the encoding is biased toward locations of the search space that are more likely to contain good solutions, or if the encoding and the genetic operators generate a fitness landscape better suited for search by evolutionary algorithms. Furthermore, a developmental process may interact with the environment to provide the adaptivity and robustness seen in biological organisms [59].

Such genetic encodings are referred to as embryogenics [45], embryogeny [6], artificial ontogeny [8], morphogenic evolutionary computation [2], or artificial embryology [11]. We refer to them simply as *developmental systems*.

In this article we describe a very simple (trying to be “minimalist”) genetic encoding and developmental system for multi-cellular systems called the *morphogenetic system*. It is inspired by gene expression and cellular differentiation and focuses on low computational complexity. This is important in evolutionary computation, that requires the evaluation of a lot of candidate solutions, and in hardware applications, that have limited available resources. The morphogenetic system is suited for compact hardware implementation. It was developed initially to suit the dynamic reconfiguration needs of a bio-inspired multi-cellular² reconfigurable electronic circuit called POEtic [84]. However, the intention is for

¹ The challenge of growth of complexity is usually associated with genetic representation [88], but other aspects may limit the complexity of evolved circuits. An important one is fitness evaluation, that may take a lot of time when complex behaviors are desired (e.g. when evolving a robot controller, sufficient time must be allowed for the robot to perform its task before assigning the fitness). Fostering environmental interactions or emergence are also key issues. Arguments in favor of genetic encodings that include developmental mechanisms were evidenced in the context of genetic programming [4] but are also relevant here.

² A multi-cellular circuit is a circuit composed of a number of interconnected cells that are the elementary functional blocks of the circuit. For example cells can implement the functionality of a logic gate, of a signal processing element, of a neuron, etc.

it to be generic. It is not tied to a specific hardware or software platform and it can be used to evolve any type of multi-cellular phenotypes.

A short review and a new classification of developmental systems in evolvable hardware is given in Section 2. Section 3 describes the morphogenetic system. Section 4 investigates its capacity to evolve 2D structures of various complexity and compares it to a direct encoding. Scalability is tested in Section 5 by evolving phenotypes of different size, and the morphogenetic system is shown to outperform a direct encoding. The capacity of the dynamics of the morphogenetic system to withstand faults is investigated in Section 6, where it is shown that it is able to recover differentiated patterns of cells even at high fault-rates. The morphogenetic system is then used to evolve spiking neural networks for pattern recognition and robot control. Results are presented in Section 7 and shown to outperform those obtained with a direct genetic encoding. The morphogenetic system is analyzed in Section 8. In particular the effect of the parameters of the system on the best fitness and on the resulting phenotypic complexity is extensively discussed. Also the morphology of phenotypes obtained by the developmental process is studied. Results are discussed in Section 9 before concluding in Section 10.

2 Developmental systems

Over the last 40 years, several scientists have proposed mathematical models of development, notably Turing's morphogenesis [83], Lindenmayer's L-Systems [52] and Kauffman's random Boolean networks [41]. Models of development were introduced in the genotype to phenotype decoding since the 1990s, with the objective of reducing the size of the search space, or providing more biologically plausible evolutionary systems. De Garis, in his 1992 PhD thesis, predicted that the combination of evolution and development would be applied to electronic circuits [11].

Developmental systems in evolvable hardware can be grouped in two categories: those that control the development of a multi-cellular circuit (i.e. a circuit whose functionality is given by interconnected elementary cells), and those that control the development of an abstract representation of the circuit (e.g. the decoding of a genetic programming tree into a circuit made of discrete components [48]). Since the morphogenetic system applies to multi-cellular systems we consider the former category. In this case a developmental system controls the behavior of cells, notably their growth, division, differentiation and possibly death.

Developmental systems can consist of biologically inspired gene regulatory networks that control the cell behavior [27, 47]. Gene regulatory networks act as programs within the cells. However these cell programs need not necessarily mimic genes and proteins; they can be more general programs. For example the cell program can also be a cellular automata [14] or a simple electronic circuit as in Miller's Developmental Cartesian Genetic Programming [59]. The cell program can even be a neural network [19] or an interpreted programming language [30], although these last two systems were only applied to the development of neural networks in software. Since simulating gene regulatory networks or cell programs may be computationally intensive, more abstract developmental systems such as L-Systems can be used [31, 82]. Developmental systems can also provide fault-tolerance in electronics by letting the circuit develop on spare cells in case of faults as in Embryonics [56, 57] or by letting the developmental system repair or grow new cells [47, 53, 54, 59]. Highly reorganizable genomes were considered for the evolution of analog networks to allow for complexity growth [58]. Developmental systems for evolvable hardware are further reviewed in [67].

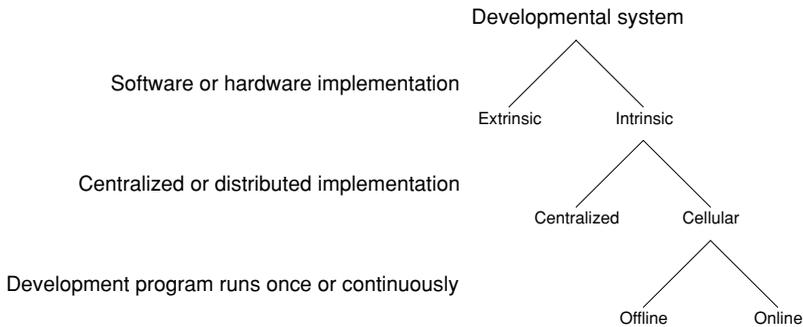


Fig. 1 Classification of developmental systems used in evolvable hardware. We distinguish between developmental systems executed in software or hardware. For those executed in hardware, their implementation can be either centralized (e.g. in a dedicated coprocessor) or cellular. In the latter case environmental or inter-cellular interactions may continuously influence the developmental process, in which case development is online. On the other hand if development is executed only once to obtain the phenotype then development is offline

We introduce here a new classification of developmental systems employed in evolvable hardware based on key characteristics of their hardware implementation (Fig. 1).

Inspired by the difference between intrinsic and extrinsic evolution in evolvable hardware [14], that distinguishes the physical implementation of an evolved circuit from its software simulation, we distinguish similarly between the execution of the developmental system in software or in hardware. *Extrinsic developmental system* means that the developmental mechanism is executed in software (e.g. on a desktop computer). The resulting circuit is then implemented physically or simulated. *Intrinsic developmental system* means that the developmental system is implemented in hardware, generally the same hardware as the one where the circuit that is evolved is evaluated (e.g. the same chip).³

In the case of an intrinsic developmental system we distinguish between a *centralized implementation* or a *distributed* or *cellular implementation*. In a centralized implementation a single hardware unit (e.g. a CPU or a dedicated coprocessor) runs the developmental mechanism in the same device as the evolved circuit. In a cellular approach the developmental system is executed by many independent but communicating units or cells that implement the developmental process in addition to their normal functionality.

Finally, in *online development* the developmental process runs continuously to decode the genotype into the phenotype. Development may thus react to inter-cellular or environmental signals while the circuit operates. This may allow for characteristics that are seen in living organisms such as self-repair or adaptation. In *offline development* the developmental process decodes the genotype into the phenotype in one step. Once the phenotype is obtained the developmental process stops. Although this does not allow inter-cellular or environmental interactions, this may save hardware resources.

Up to now, few developmental systems used in evolvable hardware are intrinsic [13, 53, 56, 82]. Among the intrinsic developmental models, Embryonics is a system that provide

³ This terminology does not indicate whether the developmental system “grows” directly the configuration string of physical hardware or of “virtual hardware” where e.g. an FPGA is configured to implement another circuit, which then undergoes development and evolution. In other words, intrinsic developmental system does not indicate whether evolution is constrained or unconstrained [80], although this may be a further distinction (e.g. intrinsic developmental system for constrained or unconstrained evolution).

self-repair capabilities to circuits, however the genetic code of the circuit is hand-coded and not evolved [56]. To the author's knowledge the only intrinsic evolutionary developmental systems are the L-System-based developmental system of Haddow and Tufte [82], the hardware growth of neural networks with a developmental system implemented in a cellular automata of De Garis et al. [13], and Liu, Miller and Tyrrell's evolutionary development of fault-tolerant arithmetic functions [53]. The first system uses a centralized implementation, while the other two use a cellular implementation of the developmental system. Gordon et al. devised a developmental system for the evolutionary growth of arithmetic functions. Although their developmental system might be efficiently implemented in hardware, it is presently software simulated, while resulting circuits are implemented and evaluated in hardware [27, 29].

The majority of developmental systems for hardware operate offline, with the notable exception of cellular programs devised by Miller and extended by Liu et al. that showed that online development could lead to fault tolerant and adaptive development of cell patterns [54, 59], Embryonics development that allows self-repairing electronic circuits [56], and Liu, Miller and Tyrrell's work on an intrinsic developmental system for fault-tolerant arithmetic functions [53].

There are few intrinsic, online and cellular developmental systems, even though this is where we believe most of the benefits of developmental systems lie. Intrinsic development means fast genotype to phenotype mapping and close interaction of the developing circuit with its environment. Together with online development this may allow adaptation to the environment and fault-tolerance. Finally cellular implementations may be faster, more scalable, more biologically plausible, and possibly more robust than centralized implementations, at the expense of more space. For these reasons, the morphogenetic system described in Section 3 is designed to allow cellular, intrinsic and online implementations.

Developmental systems can also be used in software to grow neural networks or morphologies and they are further reviewed in [2, 46, 50, 76]. Developmental systems can build neural networks using L-Systems [7, 44, 85], growth equations [60], biologically inspired growth processes [16, 17, 19, 38, 63, 70], cell languages which control the growth of the network [3, 30, 55], or incremental topology growth [75]. Developmental systems can build agent or robot morphologies and the appropriate neural controller [8, 11, 36, 73], or the morphology of 3D organisms using biologically plausible developmental systems [18, 49]. The intrinsic dynamics of developmental systems can also be used as a control system [64]. Developmental systems were proposed in the context of genetic programming too [4].

3 Morphogenetic system

Here we describe a new morphogenetic evolutionary system for multi-cellular circuits capable of dynamic reconfiguration. Reconfiguration occurs for instance when errors are detected, when the environment changes, or when the circuit is expanded with new cells [84]. Direct genetic encodings are not well suited for this kind of circuits since the number of elements in the system must be known in advance and cannot change throughout the life of the system. Furthermore, as we previously mentioned, direct genetic encodings may have problems of scalability that limit the size of practical solutions.

The morphogenetic system is a minimalist developmental system that seeks to address these issues while focusing on low computational cost, compact hardware implementation, and fast execution. It assumes that circuits consist of a regular 2D array of cells that communicate locally with their topological neighbors. Therefore the morphogenetic system can be

applied to circuits regardless of their size, and cells can be added or removed from the circuit during development. The morphogenetic system is designed for cellular implementation, which allows fast development and close interaction between the development mechanism and the environment and it allows online development that can provide fault-tolerance.

The system is inspired by the mechanisms of gene expression and cell differentiation of living organism, notably by the fact that concentrations of proteins and inter-cellular chemical signaling regulate the functionality of cells [10, 90]. It assumes that the cells of the circuit can implement a function from a set of predefined functionalities (something akin to skin, muscle, neuron cells, etc. in living organisms).

The process works in two phases. A *signaling* phase relies on local communication in the cellular circuit to exchange signals among adjacent cells to implement a diffusion process. In parallel, the *expression* phase finds the functionality to be expressed at each cell by matching the signal intensities in each cell with a corresponding functionality stored in an expression table. The genetic code contains the position of diffusing cells (*diffusers*) and the signal-function matching of the expression table. It is evolved using a genetic algorithm.

3.1 Family of functions

The morphogenetic system relies on a set of predefined functionalities which we refer to as a family of functions. The family of functions must include a sufficiently rich repertoire of functionalities to realize the desired circuit. For instance, when evolving neural networks, neurons with different connectivity patterns, and excitatory or inhibitory characteristics may be used (see Section 7). However the morphogenetic system is essentially independent of the phenotype: any functionalities can be used as long as they fit in cells. For instance pixels of different colors may be used to evolve patterns or logic gates may be used to evolve circuits.

3.2 Signaling phase

Inter-cellular communication allows the exchange of signals between adjacent cells. A signal is a simple numerical value (the signal intensity) that the cell owns, and that adjacent cells are able to read, akin to a chemical concentration. Signals may be of different types (i.e. of a different chemical nature). Signaling starts from *diffusers* placed in cells. There are diffusers for the different type of signals. When a cell contains a diffuser for a particular signal type, the intensity of this signal in the cell is always set to the maximum intensity.

We refer to the intensity of a signal of type s in cell i as C_s^i . The signaling algorithm (see below) only sets signals which have not yet been initialized (i.e. which have not yet been set by the signaling algorithm). For this reason, each signal in each cell has a flag which indicates if it is initialized (or valid). When $V_s^i = 1$ the signal of type s in cell i is initialized, otherwise $V_s^i = 0$.

Signals of each type are processed independently, without interactions among them, as if they were in different chemical layers. Initially, except for diffusers, all the signals are uninitialized. Signals are then set up by the signaling algorithm. The algorithm is illustrated in Table 1.

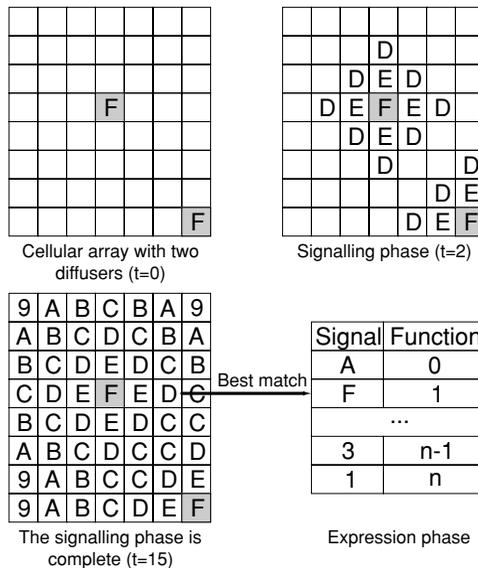
The signaling algorithm ensures that signal intensities decrease linearly with the Manhattan distance to the diffusers.⁴ All the signals are updated synchronously at the end of step

⁴ The choice of the Manhattan distance is motivated by its efficient translation in hardware. Alternatively one may interpret C_s^i not directly as the chemical concentration but as a non-linearly scaled measure of it.

Table 1 Algorithm of the signaling phase of the morphogenetic system. V_s^i is one when cell i diffuses signal s . C_s^i is the intensity of signal s in cell i . The default reset-state of cells (step 1) is for them to have maximum signal intensities (15). The signals are then updated in the developmental steps

1	Decode the chromosome to find the location of the diffusers. For all the diffusers of type t in cells j set $V_t^j = 1$ and $C_t^j = 15$, for all the other signals of type s in cells i set $V_s^i = 0$ and $C_s^i = 15$
2	For each signal s and each cell i do steps 3 to 5:
3	If $V_s^i = 1$ then skip this cell or signal
4	Compute the intensity of signal s . It is $C_s^i = \max(C_s^j - 1, 0)$. j is any of the four neighbors of cell i for which $V_s^j = 1$. Set $V_s^i = 1$. If not such cell j exists, then $V_s^i = 0$
5	Perform the expression phase to obtain the cell functionalities according to the signal intensities in the cells
6	Repeat step 2 to 5 15 times to complete the signaling phase. Each iteration corresponds to a developmental step

Fig. 2 The three first arrays are snapshots of the signaling phase with one type of signal and two diffusers (gray cells) at the start of the signaling phase (top left), after two time steps (top right) and when the signaling is complete (bottom left). The number inside the cells indicates the intensity of the signal in hexadecimal. The expression table used in the expression phase is shown on the right. In this example the signal D matches the second entry of the table with signal F (smallest Hamming distance), thus expressing function 1



4 in the table. Step 2 to 5 are one *developmental step*. Each additional developmental step expands the signals around the diffusers. Figure 2 illustrates this in the case of a single type of signal, with two diffusers placed on the cellular circuit. In the current implementation signal intensities are represented by a 4-bit number. Therefore, after $2^4 = 16$ steps the signaling phase is completed.⁵

3.3 Expression phase

The expression phase assigns a function to each cell by matching the signal intensities inside that cell with the entries of an expression table T stored in the genetic code (Fig. 2, bottom

⁵ Step 4 is an optimized implementation of $C_s^i = (\sum_{j, V_s^j=1} (C_s^j - 1)) / (\sum_j V_s^j)$ that does not require additions or divisions but gives the same result according to the properties of the signaling process.

Fig. 3 The expression table T of the morphogenetic system, here with n entries and 4 type of signals

	Signal intensities				Fcn
Entry 1	T_1^1	T_2^1	T_3^1	T_4^1	F_1
...
Entry n	T_1^n	T_2^n	T_3^n	T_4^n	F_n

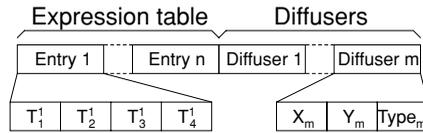


Fig. 4 The genetic code contains two parts. The first is the expression table T , here with n entries. In this example there are four type of signals therefore each entry is 16 bits long. The second part contains the location and type of the diffusers. The number of bits for the X and Y coordinates depends on the size of the network. The number of bits for the type of the diffuser depends on the number of type of signals (e.g. 2 bits when 4 type of signals are used)

right). Figure 3 shows an expression table with n entries and $S = 4$ types of signals. Each entry of the table contains the intensities of the signals and the function to express in case of match. The intensity of signal s in the entry j of the table is noted by T_s^j . A cell i is said to match an entry j of the expression table when the distance $d = \sum_{s=1}^S DOp(C_s^i, T_s^j)$ is the smallest among all entries of the expression table. The distance operator DOp is the bitwise Hamming distance.⁶

3.4 Genetic encoding and evolution

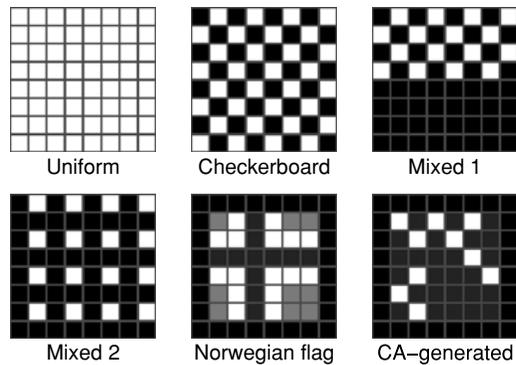
The genetic code contains the expression table T , and the location of the diffusers (Fig. 4). The genetic code therefore affects the pattern of diffusion and the expression rules of the cells in the circuit. In most of the experiments described here, we use 4 types of signals. Therefore each entry of the expression table is 16 bits (4 signals coded on 4 bits each). The functions are not encoded and evolved in these experiments. The locations of the diffusers are stored as pairs of X, Y Gray-coded coordinates, plus two bits (in the case of 4 signal types) indicating the type of the diffuser (i.e. $2^2 = 4$ type of signals). A population of genetic strings is randomly initialized and evolved using a standard genetic algorithm [25].

3.5 Computational requirements

The morphogenetic system is implemented using only additions, subtractions, comparisons and logic operations. There are no floating point operations and none of the costly operations of multiplication or division are required. This allows compact hardware implementations and fast software execution. The time required for complete development in a sequential implementation is of order $O(X \cdot Y \cdot S \cdot n)$ for an X by Y multi-cellular system with S type

⁶ The evolvability of the morphogenetic system was tested with three distance operators DOp on the task described in Section 4: the Hamming distance, the absolute difference, and the square difference. Overall, performance was best when DOp was the Hamming distance, presumably because of its discontinuous nature which allows signals with very different intensities to map to identical entries in the expression table. Furthermore the Hamming distance can be implemented in hardware in a compact way which is why it was used.

Fig. 5 The pattern coverage experiment consists in evolving an array of 8×8 cells with a specific target pattern. There are six target pattern. The first four are binary patterns (*uniform*, *checkerboard*, *mixed1* and *mixed2*) and the last two use 4 colors (*Norwegian flag*, *CA-generated* pattern)



of signals and n entries in the expression table. In a cellular (distributed) implementation where each cell implements the signaling and expression mechanisms the time is of order $O(S \cdot n)$. The number of diffusers influences only the initialization time (i.e. setting the signal intensities to the maximum value after decoding of the genetic string).

4 Evolvability

A prerequisite for evolvability⁷ is the ability of genetic encodings to generate phenotypes of various structure and complexity. This capacity is assessed by evolving phenotypes to resemble various 2D patterns [59]. In this case cell functionalities correspond to pixel colors. The six 8×8 target patterns illustrated in Fig. 5 are considered. The first four patterns (*uniform*, *checkerboard*, *mixed1* and *mixed2*) are black and white (family of functions: black, white) and test whether the morphogenetic system is capable of generating uniform structures and different kinds of diversified structures. Those patterns were initially proposed in [68]. The remaining two patterns (*Norwegian flag* and *CA-generated* pattern) are composed of four colors (family of functions: black, white, blue, red, shown in grayscale in this paper). They were initially proposed in [20]. The *Norwegian flag* has symmetries which may be exploited by a developmental system as proposed in [76]. The *CA-generated* pattern is generated with a cellular automata (CA) using Wolfram's rule 90 starting from a random initial line. This rule tends to generate patterns of high complexity which may be difficult for a developmental system to evolve.

The parameters of the morphogenetic system are the following: 2 or 4 entries in the expression table (depending on the size of the function family), 16 diffusers and 4 types of signals. The number of diffusers has been selected from preliminary tests which showed this value to be adequate to generate a wide range of target patterns. The chromosome size is 160 and 192 bits for the 2- and 4-color patterns respectively: 2 or 4 entries in the expression table * 16 bits + 16 diffusers * 8 bits (6 bits for the coordinates and 2 bits for the type of the diffuser).

The population is composed of 400 individuals, selection is rank selection of the 300 best individuals (the first 100 best individuals are reproduced twice, the following 200 are copied once), the mutation rate is 0.5% per bit, one-point crossover rate is 20% and elitism is used by copying the 5 best individuals without modifications into the new generation.

⁷ Evolvability is understood here as the capacity of the genetic representation and operators to produce offsprings with fitness higher than their parents [1]. See [62] for an overview on evolvability in different disciplines.

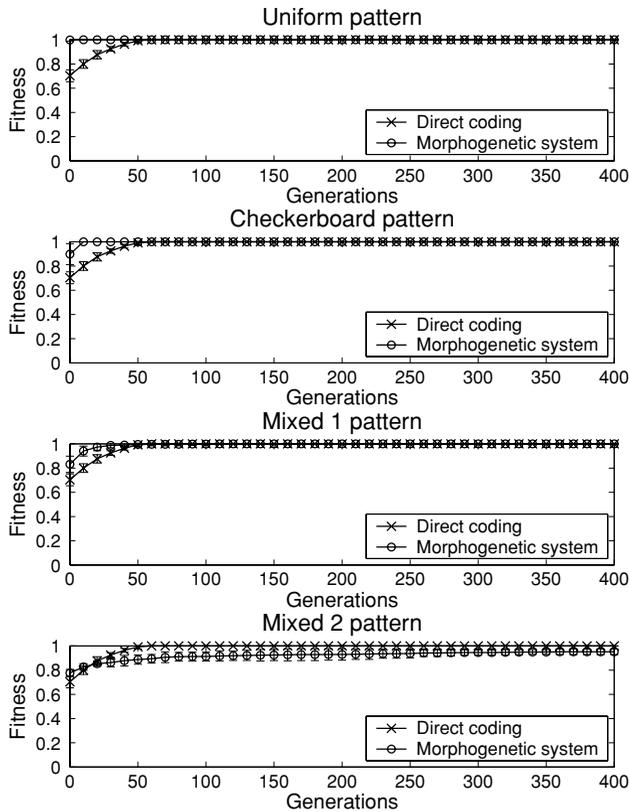


Fig. 6 Evolution of the maximum fitness for the black and white patterns (average of 10 runs). Vertical lines indicate standard deviation. For visualization purposes the standard deviation is magnified 2 times. With the exception of the *mixed2* pattern evolved with the morphogenetic system, the standard deviation tends to zero as generations increases because runs obtain identical maximum fitness scores

The fitness is proportional to the resemblance of the phenotype to the target. In order to maintain diversity, phenotypic traits that occur several times in the population contribute in decreasing amount to individuals that own them (i.e. they contribute fully to the first individual owning them, then less and less to subsequent individuals until a minimal contribution is reached) [20]. In this way we prevent the spread in the population of identical individual, and thus limit premature convergence.

The morphogenetic system is compared to a direct genetic encoding where each pixel is encoded by 1 or 2 bits for the 2- and 4-color patterns, leading to a chromosome of 64 and 128 bits respectively. The parameters of the genetic algorithm are the same as those used with the morphogenetic system.

Evolution comprises 2000 generations. Figures 6 and 7 show the evolution of the maximum fitness (average of 10 runs) for the first 400 generations for all the target patterns.

The morphogenetic system always reaches the maximum fitness with the *uniform*, *checkerboard* and *mixed1* patterns, on average in 1, 8 and 49 generations respectively. The morphogenetic system cannot fully cover the *mixed2* pattern, but it comes very close (fitness higher than 0.95).

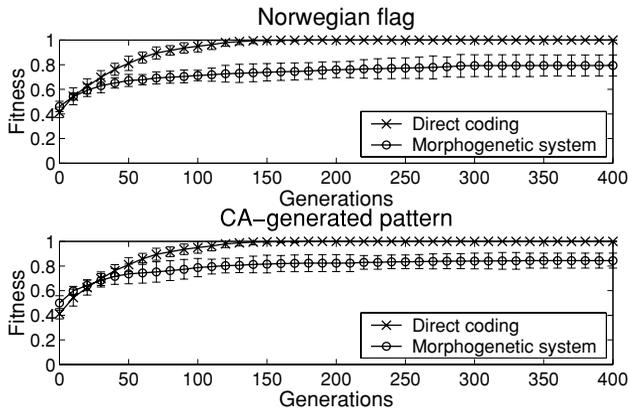


Fig. 7 Evolution of the maximum fitness for the *Norwegian flag* and *CA-generated* patterns (average of 10 runs). Vertical lines indicate standard deviation. For visualization purposes the standard deviation is magnified 2 times. The standard deviation obtained with the direct encoding tends to zero as generations increases because runs obtain identical maximum fitness scores

The *Norwegian flag* and *CA-generated* patterns, which display more complex structure and four instead of two colors, cannot be covered completely within 2000 generations by the morphogenetic system. However the fitness is still very high (average maximum fitness of 0.84 and 0.88 respectively).

The direct genetic encoding manages to reach the maximum fitness with all the patterns: the small search space and the simple structure of the fitness landscape favor that (i.e. the problem is linearly decomposable). It takes on average 51 generations to cover the patterns of two colors, and 136 generations for the pattern of 4 colors.

Genetic representations influence structures that are more likely to be generated from random genetic strings. This may be considered as a form of *bias*. When a target pattern belongs to structures that are more likely to be generated by one genetic representation, it is more likely to be found in a set of randomly generated genetic strings than if a genetic representation imposing another bias were used. Some pattern types are very easily evolved with the morphogenetic system, in particular the *uniform* pattern that is found in the first generation. In this case evolution seems unnecessary, since among 400 randomly generated genetic strings at least one achieves the maximum fitness. However, among another 400 randomly generated strings, the direct encoding does not manage to achieve the same results. This evidences a bias of the morphogenetic system toward the generation of very regular structures (and of checkerboard-type of structures which are also relatively easily evolved). Another bias of the morphogenetic system is the generation of diamond-shaped patterns centered on the diffusers (see Fig. 8). In the following sections the more complex *Norwegian flag* and *CA-generated* pattern are considered since they are not trivial to evolve with both encodings.

In the problem considered here, indirect encodings tends to have more complex fitness landscape than a direct encoding because of epistatic interactions among genes, which may generate more rugged fitness landscapes (see Section 8.5 for a preliminary analysis). Furthermore some phenotypes may not be expressible because the genotype to phenotype mapping may not allow some regions of the phenotypic space to be encoded.

Still, the morphogenetic system is capable of generating patterns of various complexity with regular and irregular structures with a relatively high fitness. The fine details of the

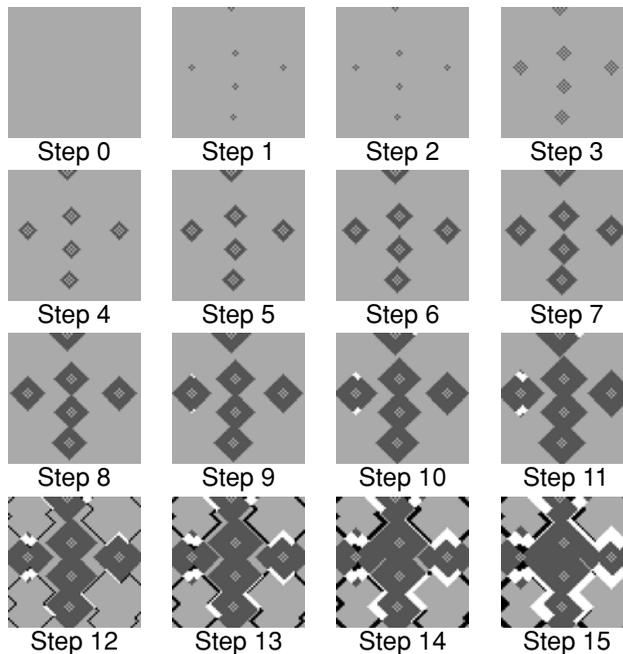


Fig. 8 Development of the 64×64 *Norwegian flag*. Each picture represents with gray levels the type of all the cells expressed in the multi-cellular circuit. Each picture is taken after a development step. After 16 steps the development is completed. The top-left picture shows the type of all the cells of the multi-cellular circuit at the beginning of the developmental process. At this stage most cells (with the exception of those having diffusers) have signals whose intensities are uninitialized (i.e. the signal intensities correspond to the reset state of the cell). The evolutionary process yet adapts the expression table so that these cells express the background color of the flag, which happens to be the most common color in the target pattern. Visually the *Norwegian flag* forms from the growth and interaction of diamond-shaped structures. These structures are caused by diffusers and are centered on them. They illustrate one bias of the morphogenetic system

phenotype may be left to an epigenetic mechanism, such as local hill climbing or Hebbian learning for neural networks, while the phylogenetic mechanism deals with structures at a coarser level. In this case the fact that specific phenotypes cannot be exactly evolved may not be an issue. Furthermore, even if evolution fails to achieve maximum fitness scores with some of the target patterns, this does not necessarily imply that real-world circuits cannot meet their specifications. In particular several circuit configurations may lead to adequate performance in real-world problems (e.g. the evolution of neural networks in Section 7). Evolvability is however easier to study in the context of patterns since their size and shape can be seamlessly varied.

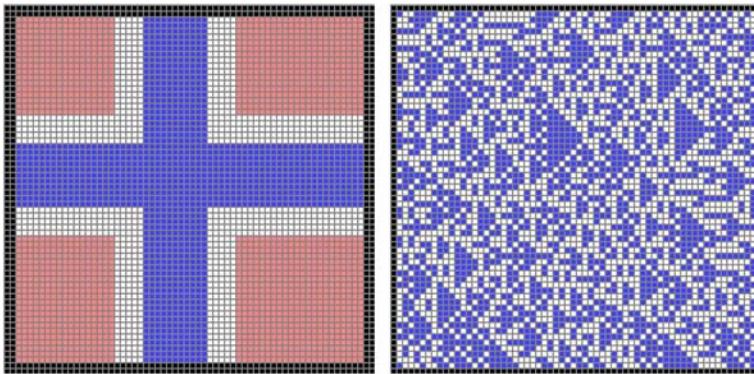
The process of development is illustrated in Fig. 8 for a larger 64×64 *Norwegian flag* (the 8×8 flag develops in very few steps, larger phenotypes make a better illustration of development).

5 Scalability

The scalability of a developmental system relates to its capacity to attain more complex solutions. The scalability of the morphogenetic system is assessed and compared to a direct genetic encoding by evolving phenotypes to resemble target patterns of increasing size. Two

Table 2 Size of the genetic code for the direct encoding and the morphogenetic system for the various phenotype sizes

Phenotype size	Genetic encoding	
	Direct encoding	Morphogenetic system
8×8	128	192
16×16	512	224
32×32	2048	256
64×64	8192	288
96×96	18432	320
128×128	32768	320
256×256	131072	352

**Fig. 9** The patterns used to assess the scalability of the morphogenetic system are the *Norwegian flag* and the *CA-Generated* pattern, here illustrated with size 64×64

phenotypes are considered, the *Norwegian flag* and the *CA-generated* pattern. Phenotype sizes are: 8×8 , 16×16 , 32×32 , 64×64 , 96×96 , 128×128 and 256×256 . The *Norwegian flag* is a scaled version of the 8×8 flag. The *CA-generated* pattern is computed using the CA rule from a wider initial line.⁸

The CA rule generates highly irregular patterns thus the complexity of the target pattern increases with its size. Figure 9 illustrates the target patterns for phenotypes of size 64×64 .

The size of genetic string with the direct encoding scales with the size of the array. The size of the genetic string of the morphogenetic system scales only with the logarithm (base 2) of the size of the array. The lengths of the genetic strings for the direct coding and the morphogenetic system are listed in Table 2.

Evolution is performed with the same parameters as in Section 4 with the GA effort limited to 2000 generations. The maximum fitness scores are measured at the last generations and

⁸ With the exception of the patterns smaller than 32×32 for which the border has a fixed width of one pixel, the *Norwegian flag* scales in length with the size of the pattern: the width of the border and the widths of the crosses inside the flag are proportional to the width of the pattern. The pattern width, the width of the border, the width of the outer cross (white) and the width of the inner cross (blue) are listed in this order in the following tuples: (8,1,1,1), (16,1,1,3), (32,1,2,5), (64,2,5,11), (96,3,7,15), (128,4,10,21), (256,8,20,41). The border of the *CA-generated* pattern is always one pixel wide. The first line of this pattern is randomly generated (each pixel takes randomly one of two colors), and the following lines are computed from the line above them using cellular automata rule 90.

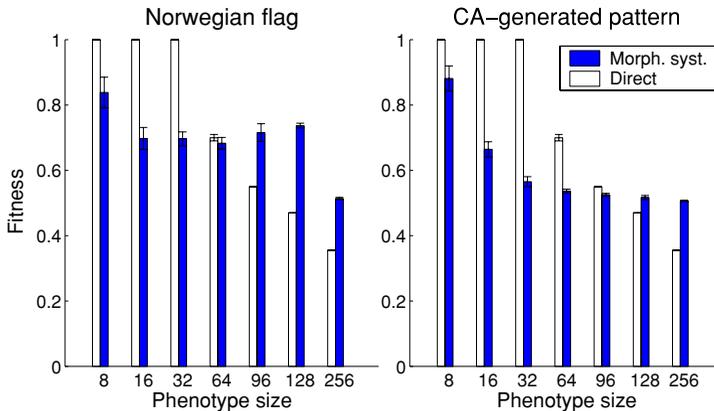


Fig. 10 Scalability of the morphogenetic system and direct encoding when evolving the *Norwegian flag* and the *CA-generated pattern*. The bars indicate the average of the maximum fitness scores measured at the last generations of 20 runs (for the 128×128 and 256×256 pattern 5 runs were done because of the long time required for the runs to complete). The bars indicate the average of the maximum fitness obtained in all the runs. At the top of the bar, the vertical line indicates the standard deviation. It tends to be very small or even null with the direct encoding, thus in this case the vertical line do not show up

averaged on 20 runs (for the 128×128 and 256×256 pattern 5 runs were done because of the long time required for the runs to complete). Figure 10 illustrates the scalability of the direct genetic encoding and the morphogenetic system. With the largest phenotypes (256×256) the morphogenetic system achieves a higher fitness than the direct genetic encoding, which indicates that it scales better than the direct encoding on this problem.

The direct encoding reaches the maximum fitness for arrays up to size 32×32 . The larger search space then limits the fitness scores for larger arrays. The morphogenetic system achieves lower fitness scores than the direct encoding on the smaller patterns (up to size 64×64 and 96×96 for the *Norwegian flag* and the *CA-generated pattern* respectively). However with larger phenotypes the morphogenetic system outperforms the direct encoding with both patterns.

The morphogenetic system tends to exploit the frequency of the colors to maximize the fitness because it can easily generate large patches with a uniform color. This happens, even without any diffusers, by expressing the most common color when signal intensities are uninitialized. As a consequence the morphogenetic system starts with higher maximum fitness than the direct coding in the first generation. It achieves a maximum fitness close to 0.5 in the first generation with the *CA-generated pattern* which mostly contains an equivalent distribution of two colors, even though each cell can take one of 4 different colors. With the *Norwegian flag* it expresses by default the background color of the flag because it is the most frequent color. It then places diffusers on the branches of the flag to generate locally the correct colors (Fig. 8). Therefore it is likely that the fitness obtained with the morphogenetic system does not drop below the normalized area that the most common color in the phenotype covers, even for larger phenotypes. In comparison the direct genetic encoding starts with a maximum fitness score that is close to 0.25 because each pixel is randomly assigned to one of the four possible colors and has a $\frac{1}{4}$ probability of matching the target pixel color.

For comparison purposes the number of diffusers (16) remained identical for all the phenotype sizes. This is not enough diffusers to entirely cover the larger multi-cellular

arrays with signals and fitness may be improved by increasing the number of diffusers. This is investigated in Section 8.

Obviously scalability cannot be achieved for all problems: it depends on the interplay between the structure of the target pattern, and the intrinsic *bias* of any genetic representation (see Section 8.3 for an analysis of one aspect of genetic bias). One extreme case is when the target pattern is random. Such pattern was not considered since it precludes any scalability possibility regardless of the developmental system. It may however serve as a reference or “worst-case” problem in future benchmarks.

6 Fault-tolerance

In this section we show that the dynamics of development may be exploited to provide fault-tolerance to evolved patterns of cells.

We consider transient events that damage the state of the cell. This can happen for instance when radiations corrupt memory elements. As development continues to operate normally, the cell functionality may be recovered. We assume that the necessary circuitry is available to detect corrupted states (e.g. by doing periodical checksums of the cell’s memories) and that a reset of the cell ensues in such cases. Therefore in case of faults all the variables describing the state of the cell take the default reset value. The cell forgets whether it was diffusing signals, and all the signal intensities are flagged as uninitialized. Here we look how the intrinsic dynamics of the morphogenetic system can recover the cell functionality upon a reset, not actually on how to design the fault detection mechanism.

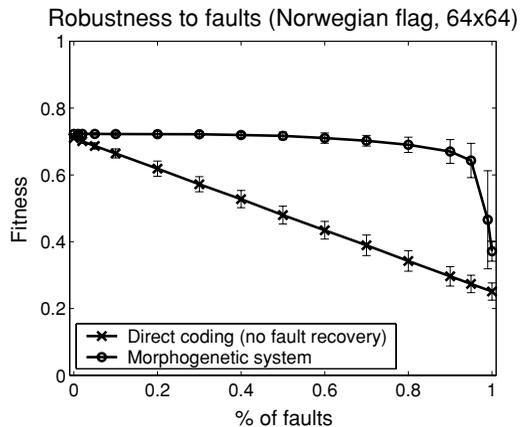
The state of a cell depends on the expression table and on the signal intensities. Both are stored in memories, and thus they may both be disrupted by faults. The expression table is part of the genome and identical in all cells. Therefore circuitry can be designed to recover the expression table from neighboring cells with a majority voting scheme. Hence we consider that such a fault can always be recovered, as long as at least one cell is intact in the system. The signal intensities however do differ in each cell, so they cannot be recovered in the same way. Nonetheless, there is a strong relationship between signal intensities in neighboring cells as the signal intensities decrease linearly with the Manhattan distance to the diffusers. Therefore signal intensities can be approximately reconstructed from the neighborhood.

To predict the signal in a cell from that of its neighbors the morphogenetic system is slightly modified. Instead of setting a signal by taking the value of the first initialized signal in a neighboring cell decremented by one, the diffusion rules assign the smallest value for which the signal gradient with all the initialized neighbors is -1 , 0 or 1 . This gives the same result as the original rules in fault-free conditions but allows better recovery in case of faults. Recovery is not always possible as there are cases where this rule does not predict correctly the signal intensities, in particular when faults occur on diffusers.

In order to highlight the effect of the dynamics of the morphogenetic system in case of faults, it is compared, as a reference, to a direct genetic encoding which has no developmental dynamics and thus no fault recovery mechanism. A fault alters randomly the color of a pixel in the case of the direct encoding.

To ensure that fault-tolerance is provided by the developmental process and not by evolution, individuals are evolved in fault-free conditions before being tested. The best evolved phenotype of the *Norwegian flag* on the 64×64 array is used for testing. This pattern and size is selected because the fitness of the morphogenetic system and the direct genetic encoding are very similar and higher than the trivial solution consisting of exploiting only the frequency of colors (the fitness would be 0.37 if the morphogenetic system initialized

Fig. 11 Fitness obtained on the Norwegian flag (size 64×64) with the morphogenetic system after recovery from different fault rates. The fitness obtained with the direct genetic encoding, which has no fault tolerance mechanism, is indicated as a reference for the same fault rates. Vertical lines indicate standard deviation, which for better visualization is magnified 4 times



all the cells with the most common color). The damage rate (percentage of faulty cells) is varied between 0% and 100%. The damage process is repeated 100 times for each damage rate on the same evolved, fault-free phenotype.

Figure 11 illustrates the results. While without developmental dynamics (i.e. with the direct encoding) a linear decrease in fitness is evidenced, the morphogenetic system shows a superior resistance to faults. The morphogenetic system benefits from the fact that signal intensities vary with continuity, and can be easily reconstructed. Also, evolution assigned the most frequent color in the target to the default cell type, which explains the fitness value of 0.37 with 100% of faults. A fitness of 0.25 is obtained in this case with the direct genetic encoding, because pixels are assigned randomly one of the four possible colours.

Figure 12 illustrates the process of recovery after faults in the case of the evolved 64×64 Norwegian flag (95% of faults). Figure 13 illustrates the 64×64 Norwegian flag recovered from different fault rates. The recovered pattern is very similar to the original one up to high fault rates.

7 Evolution of spiking neural networks

In the previous section we showed that the morphogenetic system could generate circuits resembling various 2D patterns, but these circuits did not have any functionality. In this section we demonstrate that the morphogenetic system can be used to evolve functional circuits. We describe the evolution of spiking neural networks for pattern recognition and robot navigation.

Spiking neurons depart from traditional connectionist models in that information is transmitted by the mean of pulses (or spikes) [24], rather than by firing rates. Therefore spiking neurons may have rich temporal dynamics and may exploit the temporal domain to encode information in the exchanged spikes. Spiking neurons are suited for efficient analog implementations [37], or very fast digital implementations [33, 72], and they may be optimized for use with limited resources [23, 81].

Spiking neurons have been used as controllers in evolutionary robotics, e.g. to perform vision-based obstacle avoidance [21, 23] and phototaxis [15].

Designing functional spiking networks may be challenging therefore we resort to evolution. Each cell of the multi-cellular circuit can implement one type of spiking neuron. The

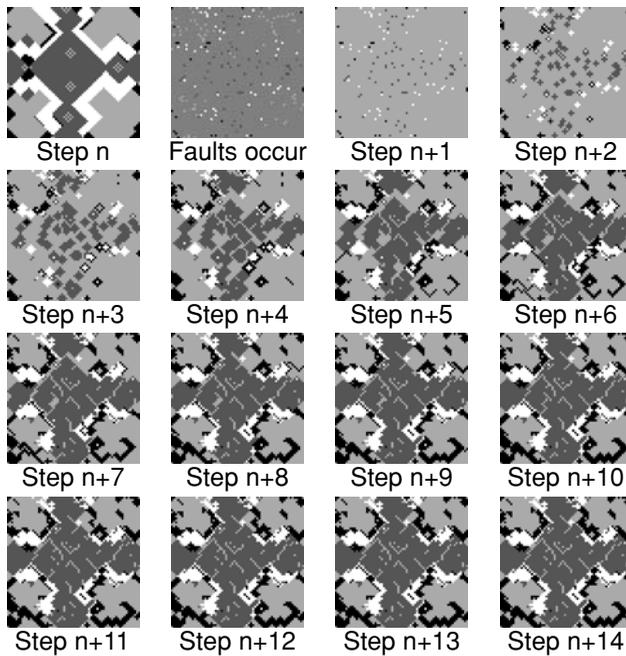


Fig. 12 Recovery of functionality after 95% of faults. The first picture is the evolved pattern. The second show the pattern after damage. The remaining pictures illustrate the pattern after each additional developmental step

family of functionalities consists of neurons with different patterns of incoming connections, and either excitatory or inhibitory characteristics (Fig. 14). In these experiments there are 12 different functionalities that can describe a large number of complex and recurrent neural architectures, depending on circuit size and genetic code. When evolving the network each cell implements a neuron selected in this family of functionalities, with the corresponding pattern of incoming connections and excitatory/inhibitory characteristics. The morphogenetic system thus uses this family of 12 functionalities (12 entries in the expression table). In the direct encoding 4 bits are assigned to each cell in the genetic string. They indicate which of the 12 functionalities in the function family the cell should implement.⁹

The spiking neuron model used in the following experiments is a discrete-time, integrate-and-fire model with leakage and a refractory period. Each neuron has weighted inputs (+ 2 or -2 depending on whether the presynaptic neuron is excitatory or inhibitory) from connected neurons, according to the connectivity patterns shown in Fig. 14. It has one more connection from an external input, e.g. to connect from a sensor, with fixed weight of + 10. The neuron integrates the incoming spikes in the membrane potential, according to the weights of the connections. Once the membrane potential reaches a threshold (fixed to 4), the neuron fires (emits a spike), resets its membrane potential to 0 and enters a refractory period where it does not integrate incoming spikes for one time step. After the integration phase and if the neuron has not fired, leakage decrements the membrane potential by 1 (or incrementing it

⁹ Since 4 bits can encode 16 values but there are only 12 cell functionalities, the cell functionalities F0 and F1 with excitatory and inhibitory characteristics are encoded twice by different binary codes.

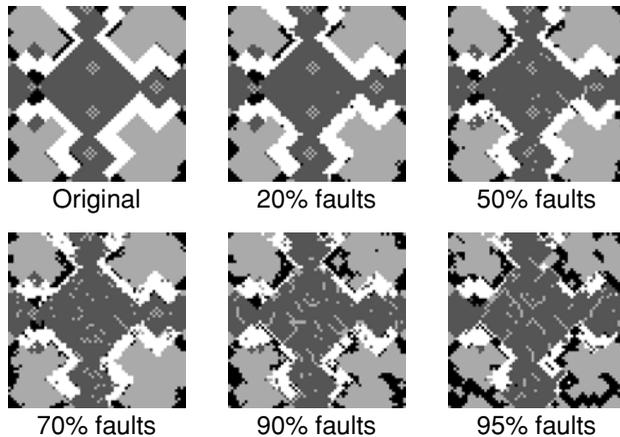


Fig. 13 Patterns recovered by development with different fault rates in the case of the evolved 64×64 Norwegian flag

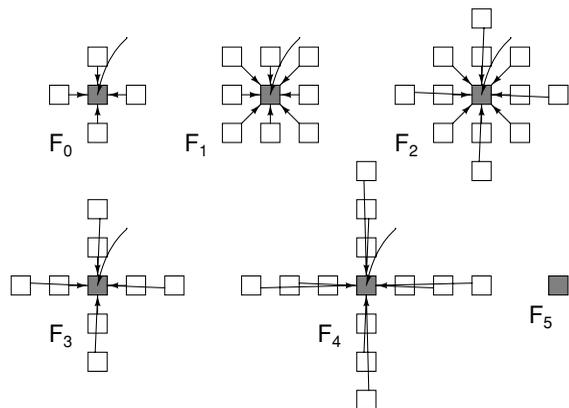


Fig. 14 A family composed of 12 functions is used when evolving neural networks. The functions are spiking neurons with different connectivities (the 6 types of connectivity shown in the figure), and with either excitatory or inhibitory characteristics. Each cell of the multi-cellular circuit implements a single neuron, shown in gray. It receives inputs from neighboring neurons (outlined), which are implemented in neighboring cells. Each neuron has an extra external input (curved arrow). Neuron F_5 is equivalent to a void cell. At the boundary of the cellular array the connectivity is truncated (no periodic boundary condition)

if the potential is below 0), so that the asymptotic potential is 0. This neural model is well suited for compact hardware implementation [69].¹⁰

In all of the experiments of this section the population is composed of 50 individuals. Selection consists of rank selection of the 15 best individuals each breeding 3 children. Finally elitism copies the 5 best individuals without modification in the new population. The other evolutionary parameters are those indicated in Section 4. Evolution with the morphogenetic system with 16 diffusers and 12 entries in the expression table (one for each type of neuron) is compared to a direct genetic encoding. The size of the morphogenetic

¹⁰ More complex neural models capable of learning can also be evolved with the morphogenetic system [69].

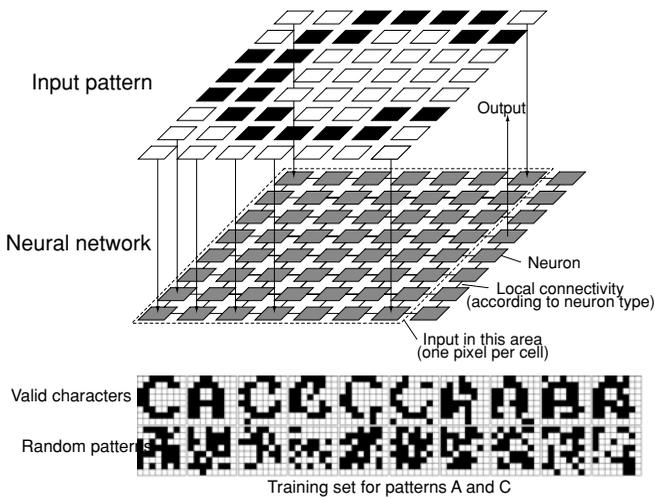


Fig. 15 *Top*: spiking network doing pattern recognition. The input pattern is applied on an array of 7×8 neurons on the left of the network. Each neuron receives the input from one pixel of the pattern. The activity of the output neuron indicates whether a character is recognized. *Bottom*: training set for the recognition of the patterns A and C. It is composed of 20 patterns. The upper 10 are the patterns to recognize which are the letters A and C. The lower 10 are random patterns that the network must reject

coding is 320 bits: 12 entries in the expression table \times 16 bits + 16 diffusers \times 8 bits (6 bits for the coordinates and 2 bits for the type of diffuser). The size of the direct coding is 256 bits (12 type of neurons, thus 4 bits/cells \times 64 cells).

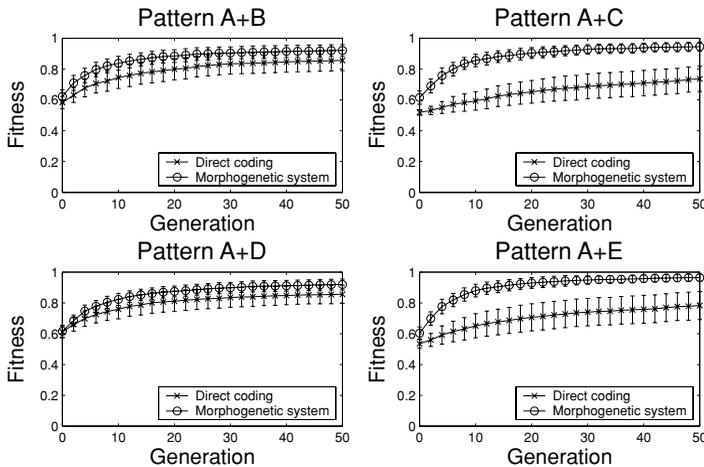
7.1 Pattern recognition

The circuit of 8×8 neurons illustrated in Fig. 15 is evolved to recognize characters (any other pattern could be used) using two training sets: one set contains corrupted versions of two characters, the other set contains random patterns which have on average the same percentage of black pixels as the characters. The circuit must tell whether the current pattern is one of the two characters or not by raising the firing rate of one specific neuron in the circuit. The input pattern is applied on the subset of neurons illustrated in Fig. 15 through their external input. Each neuron receives one pixel of the pattern: if the pixel is black, then it receives a spike every two time steps, otherwise it receives no spikes. The network is run for 100 time steps with the input applied to it, afterwards the activity of the output neuron is read. The activity (number of spikes) of that neuron indicates whether the input pattern is a character (threshold = 50% of maximum spike number).

The bottom of Fig. 15 shows the training set for the recognition of characters A and C (noted as A + C). The upper line shows the subset of patterns to recognize. The second line contains random patterns that must be rejected. The fitness of the network is evaluated by presenting successively all the patterns of the training set. It is the number of times it correctly classifies the input pattern. The maximum normalized fitness is 1, corresponding to the successful recognition of the 20 patterns in the training set. The experiments are performed with four different training sets, for the recognition of characters A + B, A + C, A + D and A + E.

Table 3 Number of runs, out of 100 performed for each training set, reaching the maximum fitness

Training set	8 × 8 network	
	Direct	Morph
A + B	18	27
A + C	8	34
A + D	19	20
A + E	14	54
Total (max is 400):	59	135

**Fig. 16** Evolution of best fitness (average of 100 runs) for pattern recognition with the four training sets. The vertical lines indicate the standard deviation of the results

The circuit is evolved one hundred times for each of the four training sets (Fig. 16). On the training sets A + C and A + E the morphogenetic system clearly outperforms the direct coding. On the other two sets the morphogenetic system achieves higher fitness than the direct coding, although the fitness difference between the two encodings is close to the standard deviation of the results. Still the morphogenetic system outperforms the direct encoding when comparing the number of runs that reach the maximum fitness after 50 generations. Table 3 reports the number of runs (on the 100 runs performed) where the maximum fitness is reached. Averaged over the four training sets, runs finding a maximum fitness using the morphogenetic encoding are more than twice as frequent than when using the direct encoding.

7.2 Robot controller

A spiking network is evolved as a controller for a Khepera miniature autonomous robot [61]. The objective is to navigate while avoiding obstacles using the sensory information coming from the proximity sensors of the robot.

Figure 17 illustrates the robot and the neural controller. There are eight input neurons organized as four groups of two neurons (S0 to S3). These input neurons are connected to the proximity sensors. Each group has one “low activity” neuron and one “high activity” neuron. When further than about 5 cm from the obstacles, none of the inputs are stimulated.

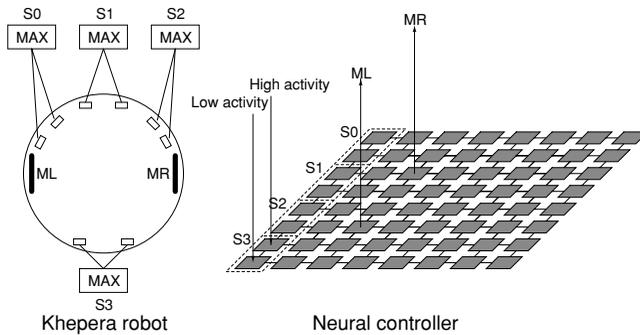


Fig. 17 The Khepera robot (*left*) and the neural controller (*right*). The Khepera has 8 proximity sensors. They are grouped by two, taking value of the most active sensors, to have 4 sensory inputs S0 to S3. The circuit receives S0 to S3 as sensory inputs. Each input is coded on two neurons. The neurons ML and MR control the speed of the wheels according to their activity

Between 5 cm and about 1 cm the “low” activity input is stimulated. When closer both the “low” and “high” activity inputs are stimulated. A stimulated input receives a spike train of period 2 (one spike every two time steps).

Neurons ML and MR are used to set the speed of the wheels, which is inversely proportional to their activity. When the neurons do not fire the speed of the wheel is +80 mm/s. With maximum activity the speed is –80 mm/s. The speed of the wheels scales linearly in between. This allows the robot to move forward when no obstacles are sensed and thus when there is potentially no activity in the network.

The robot has a sensory-motor period of 100 ms. During that period, the network is updated 20 times. At the end of the sensory-motor period, the speed of the wheels is updated and the proximity sensors are read to compute the spike trains for the next sensory-motor cycle. Noise is introduced by varying the period of the input spike trains.

The spiking neuron model used here is the same as used in the pattern recognition experiment, with the addition of random variations in the threshold value to avoid locked oscillations. For each neuron and at each time step the threshold value is incremented or decremented by 1 with a probability of 5%.

The fitness of the robot is measured on two tests of 30 seconds in a rectangular arena (40 × 65 cm). It is the average of the fitness computed at each sensory-motor step using the following equation [22]: $f = \bar{v} \cdot (1 - \Delta v) \cdot (1 - p)$, where \bar{v} is the average speed of the two wheels, Δv is the absolute value of the difference of speed of the wheels, and p is the value of the most active sensor (\bar{v} , Δv and p are in the range [0;1]). The three parts of this function aim to (1) maximize the speed of the robot, (2) minimize the rotation, and (3) maximize the distance to the obstacles.

Ten runs are performed with the morphogenetic system and a direct genetic encoding using the real robot. The morphogenetic system performs better than the direct coding (Fig. 18, left): it clearly achieves a higher maximum fitness than the direct coding. Moreover, after 15 generations, only five runs managed to find individuals displaying obstacle avoidance behavior with the direct coding, whereas with the morphogenetic system individuals were found displaying this behavior in all ten runs. Figure 18 (right) shows the typical behavior of a robot in the arena.

Observation of the controllers evolved with the morphogenetic system show that patches of interconnected excitatory neurons arise, which connect the sensors to the motor neuron

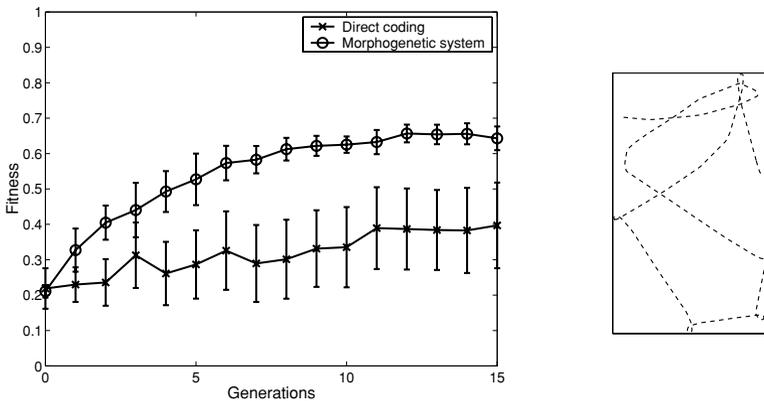


Fig. 18 *Left*: Evolution of the best fitness in the obstacle avoidance experiment (average of 10 runs on a physical robot). Vertical lines indicate the standard deviation. *Right*: Typical trajectory of the robot in the arena

and therefore cause a reversal of the wheel speeds when the robot gets close to obstacles. With the direct encoding such patterns do not seem to arise as frequently.

8 Analysis

In this section we investigate the performance of the morphogenetic system at generating specific 2D phenotypes as a function of the number of diffusers and the number of signal types. We then explore how those parameters influence the phenotypic complexity in terms of numbers of connected areas of identical colors. Finally we compare one morphological characteristic, the size of connected areas of identical colors obtained with the morphogenetic system, with those obtained with a direct genetic encoding, and we do a preliminary analysis of the fitness landscape generated by both encodings. The fitness function, target patterns and genetic algorithm parameters are the same as those used in Sections 4 and 5. Results presented below are 5 run averages unless otherwise noted.

8.1 Number of diffusers

Diffusers affect a limited area of the phenotype around their position (a 31×31 diamond centered on the diffuser). Therefore larger phenotypes tend to require more diffusers so that all the cells of the system have the chance to receive the required signal intensities to express the correct functionality, as illustrated in Fig. 19.

The effect of the number of diffusers in relation with the phenotype size is especially visible in the case of the checkerboard pattern: with a single diffuser the maximum fitness is reached for patterns up to size of 16×16 . With larger patterns the fitness decreases because diffusers do not manage to set the signals in all the cells.

The combination of signals allows the expression of complex or irregular patterns. With no diffusers or when sufficiently far from them, cells will all express the same default functionality (i.e. signal intensities are all to the default value). With more diffusers more complex phenotypes can be expressed. This is evidenced in the case of the *mixed2*, the

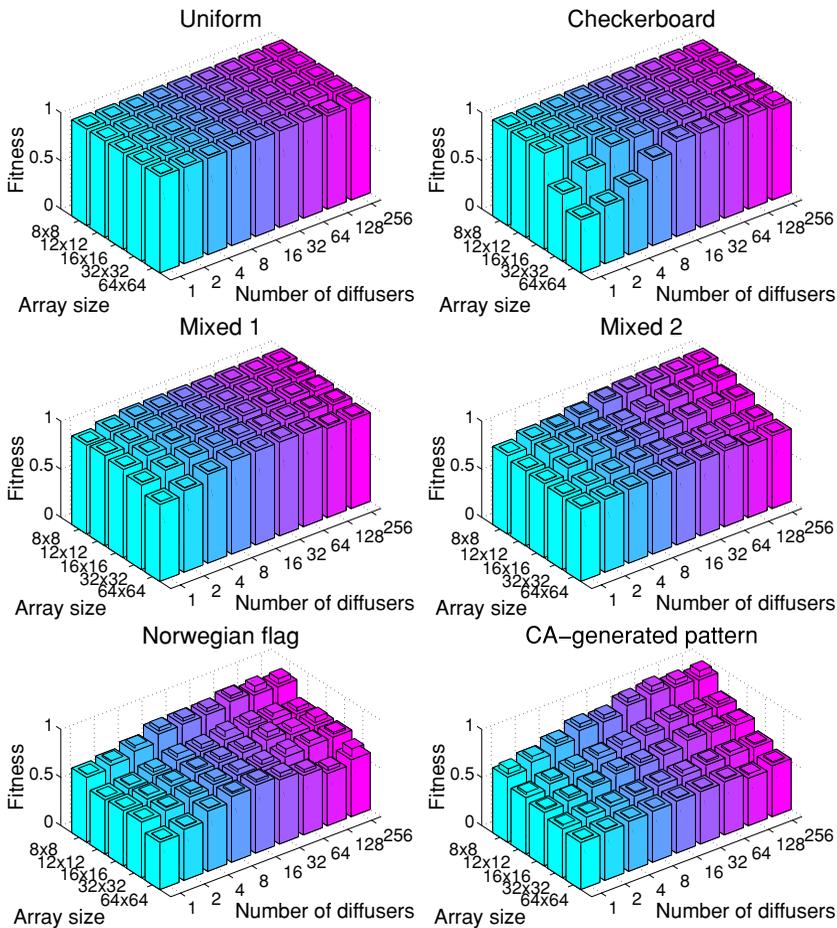


Fig. 19 Influence of the number of diffusers on the fitness obtained after 2000 generations. The wide bars represent the average of the maximum fitness obtained in 5 runs. The difference of height between the wide and thin bars represents the standard deviation. With larger or more complex phenotypes increasing the number of diffusers tends to lead to an increase in the maximum fitness score

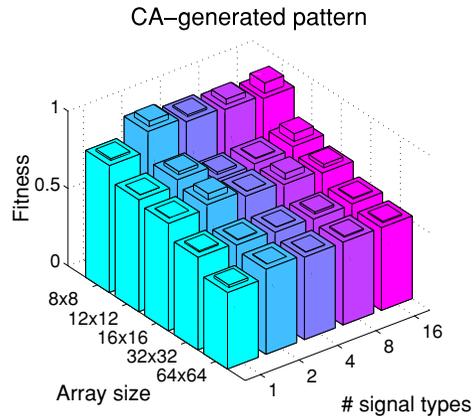
CA-generated and the *Norwegian flag* patterns: by increasing the number of diffusers the phenotypes generated by the morphogenetic system match more closely the target phenotype.

Although adding more diffusers increases the size of the genetic string, the morphogenetic system does not seem to show a degradation of the performance which is typically associated with larger search spaces.

8.2 Number of signal types

In the previous section the morphogenetic system used four type of signals. The number of signal types affects the maximum number of functionalities which can be expressed by the morphogenetic system. With n signal types, each encoded on m bits, there are $2^{n \cdot m}$

Fig. 20 Influence of the number of signals on the fitness which is obtained after 2000 generations when evolving the CA-generated pattern. The wide bars represent the average of the maximum fitness obtained in 5 runs. The difference of height between the wide and thin bars represents the standard deviation. The number of diffusers is always 16. Note that changing the number of type of signals has few influence on the fitness which is obtained



different possible states of signals in cells and therefore up to $2^{n \cdot m}$ different functions can be expressed.

Empirical tests showed that a number of 4 signal types, which are encoded on 4 bits, are adequate for the applications described in this paper. This corresponds to a maximum of $2^{4 \cdot 4} = 65536$ different functionalities which is more than the number of functionalities which are actually used. Therefore several combinations of signals may express the same cell functionalities.

Adding more signal types has little influence on the performance of the morphogenetic coding in these experiments, as illustrated by Fig. 20 which shows the fitness scores obtained after evolving the *CA-generated* pattern with different number of signal types. Results obtained with the evolution of other target patterns are similar.

While the minimum number of signal types should allow the expression of all the functionalities required in the system, selecting a too high number of signal types does not seem to affect the performance of the morphogenetic system, even if this increases the size of the search space. There is however an influence associated with the number signals when the number of diffusers is changed. This is discussed in Section 5.3.

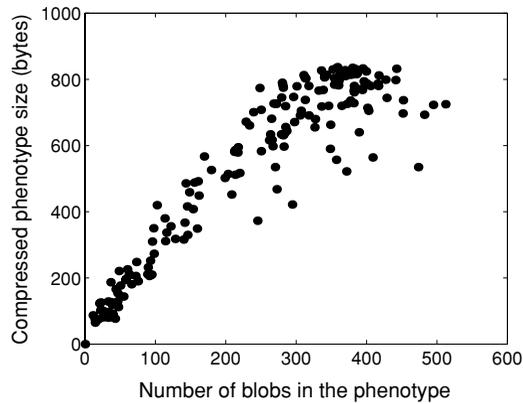
8.3 Phenotypic complexity

Evolvability may be improved if a genetic encoding is biased toward locations of the search space which are more likely to contain solutions. This bias depends on the nature of the morphogenetic system and on its parameters. An important factor when evolving phenotypes to resemble specific 2D patterns is the capacity of the genetic encoding to generate phenotypes of different structural complexity. We investigate the effect of the parameters of the morphogenetic system on this aspect of the genetic bias. Knowing the effect of these parameters on the phenotypic complexity might help selecting appropriate parameters for evolution.

The parameters which affect the phenotypic complexity include the number of diffusers, number of signal types and number of entries in the expression table (i.e. several entries in the expression table may map to the same functionality). This section deals with binary phenotypes (family of two functions, e.g. black and white cells).

As we are dealing with the evolution of specific 2D patterns, the phenotypic complexity is understood as a measure of the irregularity of the phenotype. A quantitative measure of

Fig. 21 The phenotypic complexity in bytes is plotted against the number of blobs in the phenotype for many randomly generated phenotypes. The figure illustrates the correlation between these two measures of phenotypic complexity



the complexity is the Kolmogorov measure of complexity, that can be approximated by the compressibility [51]. The latter measure consists in running a compression algorithm on the phenotype. The size of the compressed phenotype is an indication of its complexity (irregular phenotypes tend to be less compressible).

Here we consider compressibility with the Lempel-Ziv algorithm [71] as a measure of complexity.¹¹ To verify that it is related to the structural complexity in terms of size and shapes of patterns in the phenotype, we compare it with another measure that is based on structural characteristics of the phenotype. This measure is the number of *blobs* in the phenotype. A blob is a computer vision term which describes a connected region (or object) in an image. Two pixels of same color that touch along the horizontal or vertical axis are considered part of the same blob [26].

The number of blobs and the size of the compressed phenotypes is measured for a large number of randomly generated binary phenotypes of various size (from 8×8 up to 64×64) with various parameters of the morphogenetic system (from 1 to 1024 diffusers, from 2 to 8 entries in the expression table and from 1 to 16 type of signals). Figure 21 shows that there is a significant correlation¹² between the number of blobs in a phenotype and its compressed size. Therefore compressibility is a reasonable indication of the structural complexity of the phenotypes.

The bias of the morphogenetic system toward phenotypes of different complexity is measured by generating random binary phenotypes with different parameters of the morphogenetic system and measuring the complexity of the phenotype by using the compression algorithm. All the data presented below are averages obtained on 25 random binary phenotypes.

Figure 22 shows the effect of the number of diffusers on the relative complexity of phenotypes of different sizes. A small number of diffusers limits the complexity of the phenotype. With increased number of diffusers the complexity also increases. With ever increasing diffusers, cells all tend to have signals of maximum intensities and therefore they express the same functionality, thereby reducing the complexity. The number of diffusers for which the complexity is highest depends on the phenotype size.

¹¹ A custom implementation of the algorithm was used which has an identical behavior regardless of the input size.

¹² Linear correlation coefficient $r = 0.976$ with standard deviation 0.003 using bootstrap method.

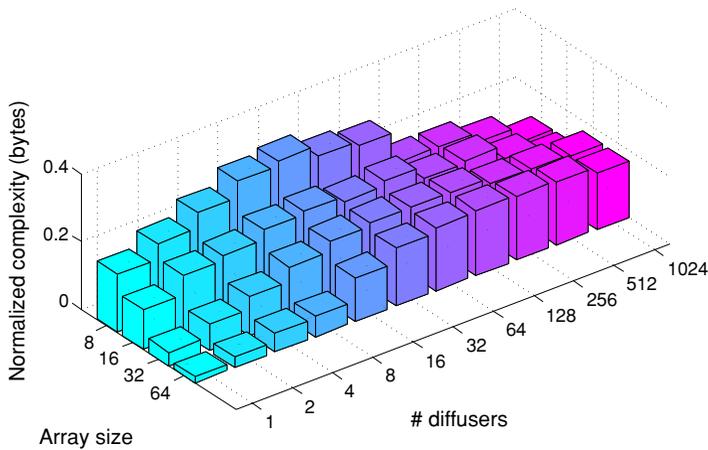


Fig. 22 Relative complexity in function of the number of diffusers for different array sizes. The complexity is normalized by the phenotype area for better scaling in the figure. The other parameters of the morphogenetic system are: 4 types of signals and 2 entries in the expression table (one for each cell functionality). The number of diffusers influences the phenotypic complexity: too few or too many number of diffusers lead to low complexity (i.e. cells will mostly contain either uninitialized signals or signals of maximum intensity), whereas with an average number of diffusers the complexity is maximized

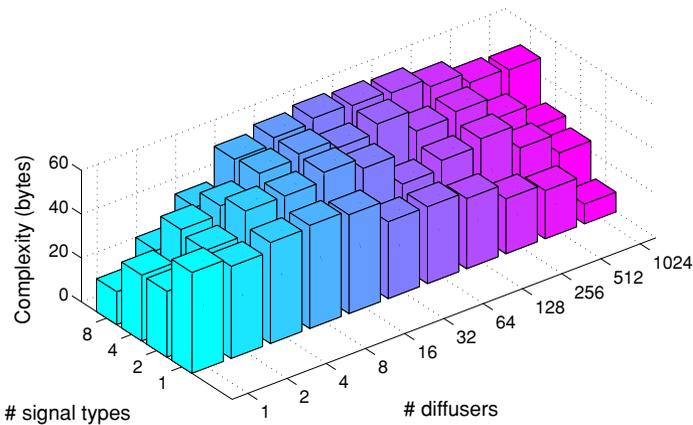


Fig. 23 Complexity in function of the number of signal types and number of diffusers in a 16×16 array with two functionalities (2 entries in the expression table). With few diffusers the complexity decreases when increasing the number of signal types because cells with uninitialized signals predominate, and thereby they express the same functionality. With many diffusers increasing the number of signal types increases the complexity because more combinations of signal intensities are possible

Figure 23 illustrates the effect of the number of signal types and number of diffusers on the complexity in the case of a 16×16 array (results with other array sizes are similar). For low number of diffusers, increasing the number of signals decreases the complexity (see the figure with a single diffuser). This happens because chemical layers tend to have fewer or no diffusers, hence signal intensities are more likely to be uninitialized. Therefore cells will express with higher probability identical functionalities corresponding to uninitialized signals. When increasing the number of diffusers, having more signal types allows more

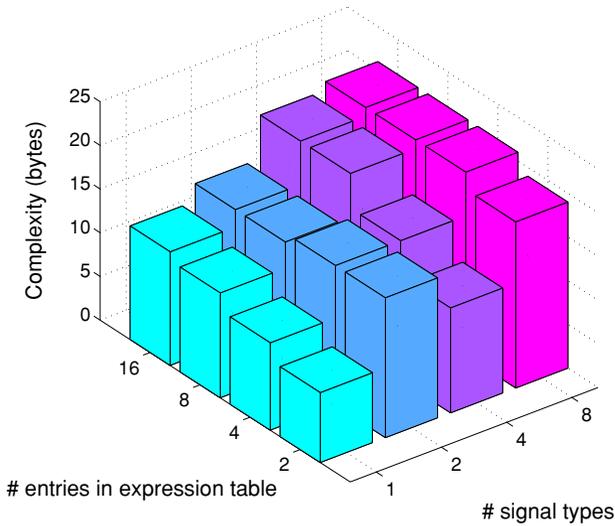


Fig. 24 Complexity in function of the number of types of signal types and number entries in the expression table. The phenotype is an array of 8×8 cells; the number of diffusers is 128. The number of entries in the expression table varies from 2 to 16 (i.e. there are from 1 to 8 entries in the expression table for each cell functionality). The phenotypic complexity tends to increase with more entries in the expression table

complex combinations of signals in the cells and this generates higher structural complexity in the phenotypes (see the figure with 1024 diffusers). In this case, if the number of signal types is restricted, large parts of the phenotype are saturated with signals of maximum intensity. Therefore functionalities corresponding to saturated signals are expressed with higher probabilities, which reduces the complexity. This is the case for 16 and more diffusers in the figure.

There must be at least one entry in the expression table for each functionality in the family that is used, but it is possible to have several entries in the expression table that correspond to the same functionality. Figure 24 shows that on average the complexity of the phenotypes tends to increase when several entries in the expression table map to the same functionality.

In summary the number of diffusers, array size, number of signal and number of entries in the expression table have a coupled effect on the complexity. We speculate that these parameters might be selected according to the foreseen complexity of the target phenotypes. This however must be further investigated since the complexity of random phenotypes (i.e. those of the first random generation) may not necessarily be helpful to bootstrap evolution toward the useful complexity (i.e. functional) necessary to solve the problem at hand. Alternatively these parameters may be subject to evolution.

8.4 Morphological characteristics

The morphogenetic system generates blobs whose size have a different distribution compared to a direct genetic encoding. We assess this by generating 100 random 8×8 binary phenotypes with a direct genetic encoding and the morphogenetic system (16 diffusers, 2 functionalities, 4 signal types). For each phenotype, we detect the blobs and we count the number of cells belonging to each blob. From this we estimate the probability for a cell to belong to a connected area in function of its size (Fig. 25). In comparison to the direct

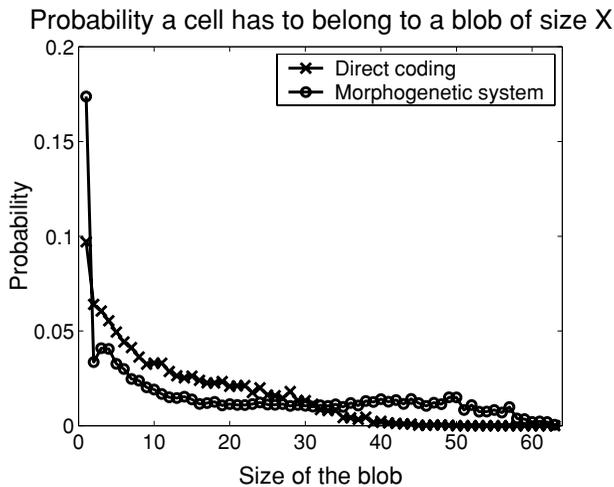


Fig. 25 Probability for a cell to belong to a blob in function of its size. With the morphogenetic system, cells tend to belong more to blobs of size 1 than with the direct coding; cells also belong more to blobs of larger size (30 and more cells), at the expense of blobs of size in the range 2 to 20

genetic encoding, cells tend to belong more to larger blobs and less to smaller ones with the morphogenetic system.

This supports the results obtained when evolving 2D phenotypes to resemble to the *uniform* pattern. It also explains the better performance of the morphogenetic system in the robotic experiment. In that experiment we observed that the morphogenetic system generates easily large patches of interconnected excitatory neurons that link sensors to motor neurons, causing a reversal of wheel speeds when the robot approaches an obstacle. The direct genetic encoding is slower at doing this.

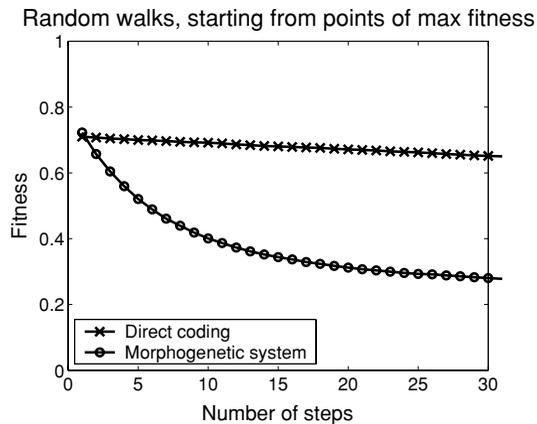
8.5 Fitness landscape

The ruggedness of the fitness landscape is often linked to the search difficulty when genetic algorithms are used [74]. The ruggedness is investigated by performing random walks using the mutation operator starting from points of maximum fitness in the evolved 64×64 *Norwegian flag*. The genetic encoding and mutation rate are the same as used during evolution in Section 5. Taking the best evolved individual, 3000 random walks of 30 steps are performed with the morphogenetic system and the direct genetic encoding. Figure 26 shows that the fitness drops faster with the morphogenetic system when moving away from a point of maximum fitness, which seems to imply a more rugged fitness landscape in the case of the morphogenetic system. The better performance of the morphogenetic system observed in previous experiments thus does not seem to come from a smoother fitness landscape. This tends to support that the morphogenetic system benefits essentially from its smaller search space size in comparison to the direct genetic encoding.

9 Discussion

Combining a genetic encoding with a developmental system in an evolutionary system may provide dynamic reorganization capabilities for multi-cellular circuits [84]. For instance

Fig. 26 Comparison of random walks. The fitness drops faster with the morphogenetic system when going farther away from points of maximum fitness. This seems to indicate that the fitness landscape is more rugged with the morphogenetic system



reorganization may occur when the circuit is expanded with new cells, when the environment changes, or when sensors or actuators are connected to the circuit. With the morphogenetic system, a cell newly connected to the circuit would differentiate according to the signals emitted from its neighbors. Dynamic reorganization can be mediated by “chemicals” (i.e. signals produced by diffusers) injected in the multi-cellular circuit from the environment or special cells (e.g. sensors or actuators). The morphogenetic system is designed to allow these mechanisms, however at the moment dynamic reorganization has not yet been investigated.

The morphogenetic system is based on *exogenous* structuring factors which are the diffusers, as opposed to *endogenous* self-organization where development emerges from purely local cell interactions [77]. Diffusers are similar to *morphogens*, the chemicals which convey positional information in the development of biological organisms [90]. This approach has the disadvantage of requiring more diffusers, hence a longer genetic string, when the size or complexity of the phenotype increases, but it may be compensated by the less drastic “compression ratio” that the exogenous approach allows. Indeed diffusers can be placed locally in areas of higher complexity in the phenotype. Endogenous approaches on the other hand need to encode the entire phenotype in a cell program of limited size.

Genetic encodings tend to impose a bias on the phenotypes they generate. The bias of the morphogenetic system is partly controlled by the expression table which is evolved. It indicates cell functionalities expressed under default conditions (no signals). As a consequence the expression of cell types may be partly adjusted to the phenotype statistics. This is evident in the evolution of phenotypes to resemble specific 2D patterns. Evolution tends to assign the most common color (e.g. the background of the *Norwegian flag*, Fig. 8) to uninitialized cells and further refinement is modulated by the diffusers.

The genetic bias is also partly controlled by the parameters of the morphogenetic system (e.g. number of diffusers or number of signal types). We have considered one type of bias which is the phenotypic complexity. It is measured by the size of the phenotype compressed by the Lempel-Ziv algorithm as suggested by [51], that is related to the structural complexity in terms of number of connected areas of identical colors in the phenotype. We also showed that morphogenetic system parameters influence the obtained fitness.

The morphogenetic system employs hard-coded signaling and expression mechanisms that tend to generate diamond-shaped patterns (Fig. 8). This is another bias that may limit the maximum fitness that can be achieved by the morphogenetic system. However these hard-coded mechanisms are also one of the reasons for the simplicity of the morphogenetic system.

More general developmental systems use evolved cell programs (e.g. a gene regulatory network or a neural network) to control the production of chemicals and the differentiation of cells [17, 59]. Another key difference between these systems and the morphogenetic system is that in the latter the production of chemicals is not modulated during development by the chemicals present in the cells (i.e. new diffusers are not introduced, or existing diffusers are not shut off). This makes the morphogenetic system intrinsically simpler compared to developmental systems using evolved cell programs (e.g. gene regulatory networks).

A developmental system may impose a trade-off between the quick generation of large structures and its ability to fine-tune a solution [2]. The morphogenetic system seems to allow both large structures to emerge (i.e. the initialization of large areas of cells of the same type) and local refinement modulated by diffusers. The expression mechanism also seems to allow the expression of irregular structures, while not preventing the evolution of more regular ones.

The morphogenetic system has been optimized for low computational complexity. It does not use multiplications, divisions, nor floating point operations. The signaling and expression mechanism can be implemented with increments, decrements, logic operations and comparisons. In a software implementation, development and evaluation of the fitness of an 8×8 phenotype with a family of 4 functionalities (experiments of Section 4) proceeds at the speed of approximately 10000 individuals/sec on a 2.08 GHz AMD Athlon XP CPU. The speed scales down with the size of the phenotypes: phenotypes of 64×64 are evaluated at 90 individuals/second. As a reference, a direct encoding evaluates 60000 individuals/sec with an 8×8 phenotype. In comparison, the artificial embryogeny of Federici [19], which relies on a neural network controlling the growth of a multi-cellular system, achieves approximately 1000 individuals/sec in the same experiment with an 8×8 phenotype. While this developmental system is slower than the morphogenetic system, it is more biologically plausible as development starts from a single cell and cell growth or death is physically located (i.e. cell duplication “pushes” neighboring cells). Still, on a common task, the scalability and fault-tolerance of the artificial embryogeny and the morphogenetic system was shown to be similar [66]. Therefore the morphogenetic system fulfills its objective of low computational complexity and at the same time it performs well compared to a more complex developmental system. Gordon et al. investigated the evolvability and scalability of a developmental system akin to a minimalistic gene regulatory network against (among others) the one introduced in this paper on the *Norwegian flag* [28]. The authors showed promising results and argued that developmental systems biased toward generating symmetric patterns may be a key to evolvability according to biological insight, but they did not analyze the computational cost of the developmental system nor its suitability for hardware implementation. A developmental system based on fractal proteins evaluates the temporal behavior of protein concentrations at a rate of 20 individuals/sec on a 1GHz PC in what would be the equivalent of a single cell in the application discussed here [5]. Eggenberger’s biologically inspired model of development which is used to create 3D neural networks from local genetic interactions also seems relatively complex as it requires multiplications, divisions and exponentials to compute gene regulation [16, 17].

The morphogenetic system is capable of *online* development. We showed in software that online development allows the recovery from faults up to high fault rates by exploiting the dynamics of the developmental process. These results bear some similarities with those shown by Miller [59] and Federici [66]. In these two cases fault-tolerance is emergent (there is no selective pressure for it): regeneration appears as a side effect of the evolution of fit individuals. In the morphogenetic system fault-tolerance is an *intrinsic* property of the diffusion process (i.e. the fault-tolerant diffusion rules are designed to approximate the

signals in a cell based on the signals in neighboring cells). Since regeneration is a property of the diffusion process, it is in principle possible to predict the amount of robustness that can be obtained. This is an advantage compared to the other two systems where robustness cannot be predicted and has to be tested.

The low computational complexity of the morphogenetic system allows compact implementation in reconfigurable hardware, such as the one that we carried out in the reconfigurable POEtic circuit [67, 65]. According to the classification of Section 2 this is an *intrinsic, online and cellular* implementation. The intrinsic cellular implementation of the morphogenetic system is fast and possibly more robust than centralized sequential implementations such as those done in specialized hardware or in software. In particular, development time is constant regardless of the number of cells in the system. Online development is exploited in the software model where we showed that it brings fault-tolerance, although the hardware implementation does not yet use the fault-tolerant diffusion rules of Section 6.

10 Conclusion

We introduced a new classification of developmental systems for evolvable hardware. This classification highlighted one category of developmental systems, *intrinsic, online and cellular*, that has rarely been investigated, even though we argued that this category is where most of the benefits of developmental systems lie (i.e. speed, implementation scalability and robustness, inter-cellular and environmental interactions that allow fault-tolerance or adaptivity).

Subsequently we introduced a very simple developmental system and genetic encoding suited for multi-cellular circuits called the morphogenetic system. This morphogenetic system allows intrinsic, online and cellular implementations. It is inspired by gene expression and cell differentiation. It can be implemented in a fully distributed way, and it assumes only local communication between immediate neighboring cells, which allows it to be applied to circuits regardless of their size. In addition it achieves low computational complexity: it does not use multiplications, divisions, nor floating point operations. This makes it suited for compact hardware implementation, and in particular we implemented it in a dynamically reconfigurable bio-inspired electronic circuit. It was developed originally as the evolutionary mechanism of this circuit, but it is suited for any hardware or software platform, and it can evolve any kind of multi-cellular systems.

We found that the morphogenetic system was capable of evolving patterns of diversified structures with relatively high fitness scores, and that it scaled better than a direct genetic encoding on the patterns which were considered. We showed that the dynamics of the developmental system could recover structures of differentiated cells up to high fault rates. In applications which consisted of evolving networks of spiking neurons, the morphogenetic system outperformed a direct genetic encoding in tasks of pattern recognition and robot control.

Future work may consider a number of improvements. Variable diffusion ranges may allow more efficient evolution of phenotypic structures by letting long range signals shape large structures while signals of shorter range take care of local details. Evolution may be used to adapt the number of diffusers to the size and complexity of the phenotype. The full potential of development is still to be explored. Development may take into account environmental interactions to provide adaptation to new operating conditions, for instance if cells are added or removed from the system. Also the current system relies on pre-defined functionalities. It remains to be explored how new functionalities can be created or modified by the evolutionary process.

There seem to be two approaches to developmental systems. On the one hand supporters of biologically plausible models of development claim that scalability and evolvability can only be improved this way [49, 50]. However the level of biological realism is limited by our still partial understanding of biological development [89] and the trade-offs between biological plausibility and implementation constraints (e.g. limited silicon resources) and evolutionary needs (e.g. fast evaluation of many candidate solutions). On the other hand there are developmental systems which may be inspired by biology, but do not seek biological plausibility. This second approach is a more pragmatic view of development [78].

The morphogenetic system is inspired by biology, but it is very far from any model of biological development and it does not seek biological plausibility. Instead it focuses on low computational cost and efficient hardware implementation. Still, its simplicity and its performance in various tasks indicate that this “minimalist” approach does produce interesting results and is worth further exploration.

Acknowledgments This project is funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community’s opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to this project are funded by the Swiss government grant 00.0529-1.

References

1. L. Altenberg, “The evolution of evolvability,” in *Advances in Genetic Programming*, K. Kinneer (ed.), MIT Press: Cambridge, MA, 1994, pp. 47–74.
2. P. J. Angeline, “Morphogenic evolutionary computations: Introduction, issues and examples,” in *The Fourth Annual Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds and D. B. Fogel (eds.), MIT Press: Cambridge, MA, 1995, pp. 387–401.
3. J. C. Astor and C. Adami, “A developmental model for the evolution of artificial neural networks,” *Artificial Life*, vol. 6, pp. 189–218, 2000.
4. W. Banzhaf and J. Miller, “The challenge of complexity,” in *Frontiers of Evolutionary Computation*, volume 11 of *Genetic Algorithms And Evolutionary Computation Series*, A. Menon (ed.), Kluwer Academic Publishers: Boston, MA, USA, chapter 11, 2004, pp. 73–99.
5. P. J. Bentley, “Fractal proteins,” *Genetic Programming and Evolvable Machines*, vol. 5, pp.71–101, 2004.
6. P. J. Bentley and S. Kumar, “Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem,” in *Proceeding of Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (eds.), Morgan Kaufmann: San Francisco, CA, 1999, pp. 35–43.
7. E. J. W. Boers and H. Kuiper, “Biological metaphors and the design of modular artificial neural networks,” Master’s thesis, Department of Computer Science and Experimental and Theoretical Psychology, Leiden University: The Netherlands, 1992.
8. J. C. Bongard and R. Pfeifer, “Evolving complete agents using artificial ontogeny,” in *Morpho-Functional Machines: The New Species (Designing Embodied Intelligence)*, F. Hara and R. Pfeifer, (eds.), Springer-Verlag: Heidelberg, 2003, pp. 237–258.
9. C. A. Coello, A. H. Aguirre, and B. P. Buckles, “Evolutionary multiobjective design of combinational logic circuits,” in *2nd NASA/DoD Workshop on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2000, pp. 161–170.
10. E. Coen, *The Art of Genes*, Oxford University Press: Oxford, 1999.
11. H. de Garis, *Genetic Programming: GenNets, Artificial Nervous Systems, Artificial Embryos*. PhD thesis, Brussels University, 1992.
12. H. de Garis, “Growing an artificial brain with a million neural net modules inside a trillion cell cellular automaton machine,” in *Proceedings of the Fourth International Symposium on Micro Machine and Computer Science*, 1993, pp. 211–214.
13. H. de Garis, L. de Penning, A. Buller, and D. Decesare, “Early experiments on the CAM-brain machine (CBM),” in *1st NASA/DoD Workshop on Evolvable Hardware*, A. Stoica et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2001, pp. 211–219.

14. F. Dellaert and R. Beer, “A developmental model for the evolution of complete autonomous agents,” in *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack and S. Wilson (eds.), MIT Press-Bradford Books: Cambridge, MA, 1996, pp. 393–401.
15. E. A. Di Paolo, “Spike timing dependent plasticity for evolved robots,” *Adaptive Behavior*, vol. 10, pp. 243–263, 2002.
16. P. Eggenberger, “Cell interactions as a control tool of developmental processes for evolutionary robotics,” in *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson (eds.), MIT Press-Bradford Books: Cambridge, MA, 1996, pp. 440–448.
17. P. Eggenberger, “Creation of neural networks based on developmental and evolutionary principles,” in *Proceedings of the International Conference on Artificial Neural Networks ICANN’97*, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (eds.), Springer-Verlag: Heidelberg, 1997, pp. 337–342.
18. P. Eggenberger, “Evolving morphologies of simulated 3D organisms based on differential gene expression,” in *Proceedings of the 4th European Conference on Artificial Life (ECAL97)*, P. Husbands and I. Harvey (eds.), MIT Press: Cambridge, MA, 1997, pp. 205–213.
19. D. Federici, “Evolving a neurocontroller through a process of embryogeny,” in *Proceedings of the 8th International Conference on Simulation of Adaptive Behavior*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer (eds.), MIT Press-Bradford Book: Cambridge, MA, 2004, pp. 373–382.
20. D. Federici, “Using embryonic stages to increase the evolvability of development,” in *Proceedings of WORLDS Workshop at GECCO 2004*, 2004.
21. D. Floreano and C. Mattiussi, “Evolution of spiking neural controllers for autonomous vision-based robots,” in *Evolutionary Robotics IV*, T. Gomi (ed.), Springer-Verlag: Heidelberg, 2001, pp. 38–61.
22. D. Floreano and F. Mondada, “Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot,” in *Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior*, D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson (eds.), MIT Press-Bradford Books: Cambridge, MA, 1994, pp. 421–430.
23. D. Floreano, N. Schoeni, G. Caprari, and J. Blynel, “Evolutionary bits’n’spikes,” in *Proceedings of Artificial Life VIII*, R. K. Standish, M. A. Bedau, and H. A. Abbass (eds.), MIT Press: Cambridge, MA, 2002, pp. 335–344.
24. W. Gerstner and W. Kistler, *Spiking Neuron Models*, Cambridge University Press, 2002.
25. D. E. Goldberg, *Genetic Algorithms in Search Optimization & Machine Learning*, Addison-Wesley: Reading, MA, 1989.
26. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley: Reading, MA, 1993.
27. T. Gordon, “Exploring models of development for evolutionary circuit design,” in *Congress on Evolutionary Computation (CEC2003)*, IEEE Press, 2003, pp. 2050–2057.
28. T. Gordon and P. Bentley, “Bias and scalability in evolutionary development,” in *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, ACM Press, 2005, pp. 83–90.
29. T. Gordon and P. Bentley, “Development brings scalability to hardware evolution,” in *2005 NASA/DoD Conference on Evolvable Hardware*, J. Lohn et al. (ed.), IEEE Computer Society Press: Los Alamitos, CA, 2005, pp. 272–279.
30. F. Gruau, *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Laboratoire de l’Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, France, 1994.
31. P. C. Haddow, G. Tufte, and P. van Remortel, “Shrinking the Genotype: L-systems for EHW?,” in *Proceedings of the 4th International Conference on Evolvable Systems (ICES 2001)*, Y. Liu et al. (eds.), Springer-Verlag: Heidelberg, 2001, pp. 128–139.
32. P. C. Haddow and P. van Remortel, “From here to there: Future robust EHW technologies for large digital designs,” in *3rd NASA/DoD Workshop on Evolvable Hardware*, D. Keymeulen et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2001, pp. 232–239.
33. G. Hartmann, G. Frank, G. Schäfer, and M. Wolff, “SPIKE128K-An accelerator for dynamic simulation of large pulse-coded networks,” in *Proceedings of MicroNeuro 97*, Dresden, 1997, pp. 130–139.
34. T. Higuchi, et al., “Evolving hardware with genetic learning: A first step towards building a Darwin machine,” in *Proceedings of the 2nd International Conference on Simulation of Adaptive Behaviour*, J.-A. Meyer, H. Roitblat, and S. Wilson (eds.), MIT Press-Bradford Books: Cambridge, MA, 1993, pp. 417–424.
35. T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, M. Salami, N. Kajihara, and N. Otsu, “Real-world applications of analog and digital evolvable hardware,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 220–235, 1999.

36. G. S. Hornby and J. B. Pollack, "Evolving L-systems to generate virtual creatures," *Computers and Graphics*, vol. 25, pp. 1041–1048, 2001.
37. G. Indiveri and R. Douglas, "ROBOTIC VISION: Neuromorphic vision sensors," *Science*, vol. 288, pp. 1189–1190, 2000.
38. N. Jakobi, *Harnessing morphogenesis*. Technical Report CSRP 423, School of Cognitive and Computing Sciences, University of Sussex, 1995.
39. I. Kajitani, T. Hoshino, D. Nishikawa, H. Yokoi, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, M. Iwata, D. Keymeulen, and T. Higuchi, "A gate-level EHW chip: implementing GA operations and reconfigurable hardware on a single LSI," in *Proceedings of the 2nd International Conference on Evolvable Systems (ICES 98)*, M. Sipper et al. (eds.), Springer-Verlag, Heidelberg, 1998, pp. 1–12.
40. T. Kalganova, "Bidirectional incremental evolution in extrinsic evolvable hardware," in *2nd NASA/DoD Workshop on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2000, pp. 65–74.
41. S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.
42. S. Kazadi, Y. Qi, I. Park, N. Huang, P. Hwu, B. Kwan, W. Lue, and H. Li, "Insufficiency of piecewise evolution," in *3rd NASA/DoD Workshop on Evolvable Hardware*, D. Keymeulen et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2001, pp. 223–231.
43. D. Keymeulen, R. Zebulum, Y. Jin, and A. Stoica, "Fault-tolerant evolvable hardware using field-programmable transistor arrays," *IEEE Transactions on Reliability*, vol. 49, pp. 305–316, 2000.
44. H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, vol. 4, pp. 461–476, 1990.
45. H. Kitano, "Challenges of evolvable systems: Analysis and future directions," in *Proceedings of the 1st International Conference on Evolvable Systems (ICES 96)*, T. Higuchi et al. (eds.), Springer-Verlag: Heidelberg, 1996, pp. 125–135.
46. J. Kodjacobian and J.-A. Meyer, "Evolution and development of control architectures in animats," *Robotics and Autonomous Systems*, vol. 16, pp. 161–182, 1995.
47. A. Koopman and D. Roggen, "Evolving genetic regulatory networks for hardware fault tolerance," in *Proceedings of Parallel Problem Solving from Nature VIII*, X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. Tiño, A. Kabñ, and H.-P. Schwefel (eds.), Springer-Verlag: Heidelberg, 2004, pp. 561–570.
48. J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.
49. S. Kumar and P. J. Bentley, "Biologically inspired evolutionary development," in *Proceedings of the 5th International Conference on Evolvable Systems (ICES 2003)*, A. M. Tyrrell et al. (eds.), Springer-Verlag: Heidelberg, 2003, pp. 57–68.
50. S. Kumar and P. J. Bentley, *On Growth, Form and Computers*, Academic Press: London, UK, 2003.
51. P. K. Lehre and P. C. Haddow, "Developmental mappings and phenotypic complexity," in *Congress on Evolutionary Computation (CEC2003)*, IEEE Press, 2003, pp. 62–68.
52. A. Lindenmayer, "Mathematical models for cellular interactions in development," *Journal of Theoretical Biology*, vol. 18, pp. 280–299, 1968.
53. H. Liu, J. Miller, and A. Tyrrell, "A biological development model for the design of robust multiplier," in *Applications of Evolutionary Computing: EvoHot*, G. Drechsler, R. Squillero (eds.), Springer-Verlag: Heidelberg, 2005, pp. 195–204.
54. H. Liu, J. F. Miller, and A. M. Tyrrell, "An intrinsic robust transient fault-tolerant developmental model for digital systems," in *Proceedings of WORLDS workshop at GECCO 2004*, 2004.
55. S. Luke and L. Spector, "Evolving graphs and networks with edge encoding: Preliminary report," in *Late Breaking Papers at the Genetic Programming 1996 Conference*, J. R. Koza (ed.), 1996, pp. 117–124.
56. D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, "Toward robust integrated circuits: The embryonics approach," in *Proceedings of the IEEE*, vol. 88, pp. 516–541, 2000.
57. P. Marchal, C. Piguet, D. Mange, A. Stauffer, and S. Durand, "Embryological development on silicon," in *Proceedings of Artificial Life IV*, R. Brooks and P. Maes (eds.), MIT Press, 1994, pp. 365–370.
58. C. Mattiussi and D. Floreano, "Evolution of analog networks using local string alignment on highly reorganizable genomes," in *2004 NASA/DoD Conference on Evolvable Hardware*, R. Zebulum et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2004, pp. 30–37.
59. J. F. Miller, "Evolving developmental programs for adaptation, morphogenesis, and self-repair," in *Proceedings of 7th European Conference on Artificial Life*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (eds.), Springer-Verlag: Heidelberg, 2003, pp. 256–265.
60. E. Mjolsness, D. H. Sharp, and B. K. Alpert, "Scaling, machine learning, and genetic neural nets," *Advances in Applied Mathematics*, vol. 10, pp. 137–163, 1989.

61. F. Mondada, E. Franzi, and P. Jenne, “Mobile robot miniaturisation: A tool for investigation in control algorithms,” in *Proceedings of the 3rd International Symposium on Experimental Robotics*, Springer-Verlag: Heidelberg, 1994, pp. 501–513.
62. C. L. Nehaniv, “Evolvability,” *BioSystems*, vol. 69, pp. 77–81, 2003.
63. S. Nolfi and D. Parisi, *Growing neural networks*. Technical report, Institute of Psychology, Rome, 1992.
64. T. Quick, C. L. Nehaniv, K. Dautenhahn, and G. Roberts, “Evolving embodied genetic regulatory network-driven control systems,” in *Proceedings of the 7th European Conference on Artificial Life (ECAL2003)*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (eds.), Springer-Verlag: Heidelberg, 2003, pp. 266–277.
65. D. Roggen, *Multi-Cellular Reconfigurable Circuits: Evolution, Morphogenesis and Learning*, PhD thesis, EPFL, Lausanne, Switzerland, 2005.
66. D. Roggen and D. Federici, “Multi-cellular development: is there scalability and robustness to gain?,” in *Proceedings of Parallel Problem Solving from Nature VIII*, X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. Tiño, A. Kabán, and H.-P. Schwefel (eds.), Springer-Verlag: Heidelberg, 2004, pp. 391–400.
67. D. Roggen and D. Floreano, “Hardware morphogenetic developmental system,” Technical report, Laboratory of Intelligent Systems, EPFL, Lausanne, Switzerland, <http://lis.epfl.ch/publications.php>, 2004.
68. D. Roggen, D. Floreano, and C. Mattiussi, “A morphogenetic evolutionary system: Phylogenesis of the POETic tissue,” in *Proceedings of the 5th International Conference on Evolvable Systems (ICES 2003)*, A. M. Tyrrell et al. (eds.), Springer-Verlag: Heidelberg, 2003, pp. 153–164.
69. D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano, “Hardware spiking neural network with runtime reconfigurable connectivity in an autonomous robot,” in *2003 NASA/DoD Conference on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2003, pp. 189–198.
70. A. G. Rust, *Developmental Self-Organisation in Artificial Neural Networks*. PhD thesis, Dept. of Computer Science, University of Hertfordshire, 1998.
71. D. Salomon, *Data Compression: The Complete Reference*, Springer-Verlag: Heidelberg, 2004.
72. T. Schoenauer, S. Atasoy, N. Mehrtaş, and H. Klar, “NeuroPipe-Chip: A digital neuro-processor for spiking neural networks,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 205–213, 2002.
73. K. Sims, “Evolving 3D morphology and behavior by competition,” in *Proceedings of the Artificial Life IV*, R. Brooks and P. Maes (eds.), MIT Press: Cambridge, MA, 1994, pp. 28–39.
74. T. Smith, P. Husbands, and M. O’Shea, “Not measuring evolvability: Initial investigation of an evolutionary robotics search space,” in *Congress on Evolutionary Computation (CEC2001)*, IEEE Press, 2001, pp. 9–16.
75. K. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol. 10, pp. 99–127, 2002.
76. K. Stanley and R. Miikkulainen, “A taxonomy for artificial embryogeny,” *Artificial Life*, vol. 9, pp. 93–130, 2003.
77. F. Streichert, C. Spieth, H. Ulmer, and A. Zell, “How to evolve the head-tail pattern from reaction-diffusion systems,” in *2004 NASA/DoD Conference on Evolvable Hardware*, R. Zebulum et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2004, pp. 261–268.
78. G. Tempesti, D. Mange, E. Petraglio, A. Stauffer, and Y. Thoma, “Developmental processes in silicon: An engineering perspective,” in *2003 NASA/DoD Conference on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2003, pp. 255–264.
79. A. Thompson, “An evolved circuit, intrinsic in silicon, entwined with physics,” in *Proceedings of the 1st International Conference on Evolvable Systems (ICES 96)*, T. Higuchi et al. (eds.), Springer-Verlag: Heidelberg, 1996, pp. 390–405.
80. A. Thompson, I. Harvey, and P. Husbands, “Unconstrained evolution and hard consequences,” in *Towards Evolvable Hardware*, E. Sanchez and M. Tomassini (eds.), Springer-Verlag: Heidelberg, 1996, pp. 136–165.
81. O. Torres, J. Eriksson, J. M. Moreno, and A. Villa, “Hardware optimization of a novel spiking neuron model for the POETic tissue,” in *Proceedings of the IWANN’03*, J. Mira (ed.), Springer-Verlag: Heidelberg, 2003, pp. 113–120.
82. G. Tufte and P. C. Haddow, “Biologically-inspired: A rule-based self-reconfiguration of a virtex chip,” in *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, M. Bubak, G. D. van Albada, P. M. A. Sloot, et al. (eds.), Springer-Verlag: Heidelberg, 2004, pp. 1249–125.
83. A. M. Turing, “The chemical basis of morphogenesis,” *Philosophical Transactions of the Royal Society of London*, B, vol. 237, pp. 37–72, 1952.
84. A. M. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.-M. Moreno, J. Rosenberg, and A. Villa, “POETic tissue: An integrated architecture for bio-inspired hardware,” in *Proceedings of the 5th*

- International Conference on Evolvable Systems (ICES 2003)*, A. M. Tyrrell et al. (eds.), Springer-Verlag: Heidelberg, 2003, pp. 129–140.
85. J. Vaario, S. Ohsuga, and K. Hori, “Connectionist modeling using Lindenmayer systems,” in *Information Modeling and Knowledge Bases: Foundations, Theory, and Applications*, Ohsuga et al. (eds.), IOS Press, 1991, pp. 496–510.
 86. V. K. Vassilev, D. Job, and J. F. Miller, “Towards the automatic design of more efficient digital circuits,” in *2nd NASA/DoD Workshop on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2000, pp. 151–160.
 87. V. K. Vassilev and J. F. Miller, “Scalability problems of digital circuit evolution: Evolvability and efficient designs,” in *2nd NASA/DoD Workshop on Evolvable Hardware*, J. Lohn et al. (eds.), IEEE Computer Society Press: Los Alamitos, CA, 2000, pp. 55–64.
 88. G. P. Wagner and L. Altenberg, “Complex adaptations and the evolution of evolvability,” *Evolution*, vol. 50, pp. 967–976, 1996.
 89. L. Wolpert, “Do we understand development?,” *Science*, vol. 266, pp. 571–572, 1994.
 90. L. Wolpert, *Principles of Development*, Oxford University Press: Oxford, 1998.
 91. X. Yao, “A review of evolutionary artificial neural networks,” *International Journal of Intelligent Systems*, vol. 4, pp. 203–222, 1993.
 92. X. Yao and T. Higuchi, “Promises and challenges of evolvable hardware,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, pp. 87–97, 1999.