Analog Genetic Encoding for the Evolution of Circuits and Networks

Claudio Mattiussi and Dario Floreano, *Senior Member, IEEE*

***Abstract*—This paper describes a new kind of genetic representation called analog genetic encoding (AGE). The representation is aimed at the evolutionary synthesis and reverse engineering of circuits and networks such as analog electronic circuits, neural networks, and genetic regulatory networks. AGE permits the simultaneous evolution of the topology and sizing of the networks. The establishment of the links between the devices that form the network is based on an implicit definition of the interaction between different parts of the genome. This reduces the amount of information that must be carried by the genome, relatively to a direct encoding of the links. The application of AGE is illustrated with examples of analog electronic circuit and neural network synthesis. The performance of the representation and the quality of the results obtained with AGE are compared with those produced by genetic programming.**

***Index Terms*—Analog circuit synthesis, analog genetic encoding (AGE), analog network synthesis, evolutionary computation, genetic representation, neural network synthesis.**

I. Introduction

MANY SYTEMS of technical and scientific interest can be seen as collections of devices connected by links characterized by a numeric value. We will call these systems *analog networks* (Fig. 1). Examples of analog networks are analog electronic circuits—where the devices are the electronic components that are not resistors and the link values correspond to the conductance between the terminals of the devices—artificial neural networks—where the devices are the neurons and the values correspond to the weights associated with the neuron inputs—and genetic regulatory networks (GRNs)—where the devices are the genes and the link values represent the effect of one gene on the activation of another.

In some cases, the focus of the engineering activity is the synthesis or design of analog networks. In other cases, it is their reverse engineering, that is, the determination of the characteristics of existing networks, taking into account some information on their behavior. When tackled by a human expert, both the synthesis and reverse engineering of analog networks are recognized as knowledge-intensive activities, where few systematic techniques exist. For this reason, there is a founded interest in the development of automatic techniques capable of handling both problems. Evolutionary methods appear as one of the most promising approaches for the fulfillment of this objective [1]–[3].

Manuscript received August 22, 2005; revised March 14, 2005 and August 23, 2005. This work was supported in part by the Swiss National Science Foundation under Grant 620-58049.

The authors are with the Laboratory of Intelligent Systems, Institute of Systems Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland (e-mail: Claudio.mattiussi@epfl.ch; dario.floreano@epfl.ch).

Digital Object Identifier 10.1109/TEVC.2006.886801

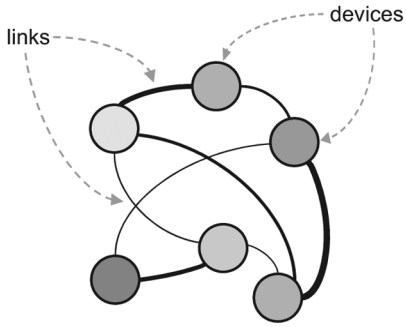


Fig. 1. An analog network is composed by a collection of devices connected by links characterized by a numeric value (represented here graphically by the thickness of the lines).

The synthesis or reverse engineering of an analog network requires the specification of the topology of the network, that is, the specification of the devices that compose the network, their connectivity, and the specification of the sizing of the network, that is, the specification of the values of the device parameters and the specification of the values that characterize the links. This implies that a genetic encoding for artificial evolution of analog networks must be capable of representing both the topology and the sizing of the network.

A. Direct Encoding

The most straightforward approach to the evolution of the topology and sizing of analog networks is the use of a variable length representation where the genome is composed by a list of genes that can represent the devices existing in the network or the links between the devices. Each gene representing a device specifies the nature of the device and the value of its parameters. Each gene representing a link specifies the value characterizing the link and specifies also the terminals of the devices that it connects. The direct encoding has been applied to the synthesis of electronic circuits [4]–[6] and neural networks [7]–[11]. This representation has the advantage of simplicity in the decoding of the genome, but the need of explicitly representing each connection results in a rapid growth in genome length with the complexity of the network.

B. Developmental Encoding

Another popular approach to the genetic representation of analog networks is based on the use of a genome that directs a developmental process, leading to the construction of the network. Examples of evolutionary developmental systems are [12]–[15] for the case of electronic circuits, [16]–[22] for the case of artificial neural networks, and [23], [24] for GRNs.

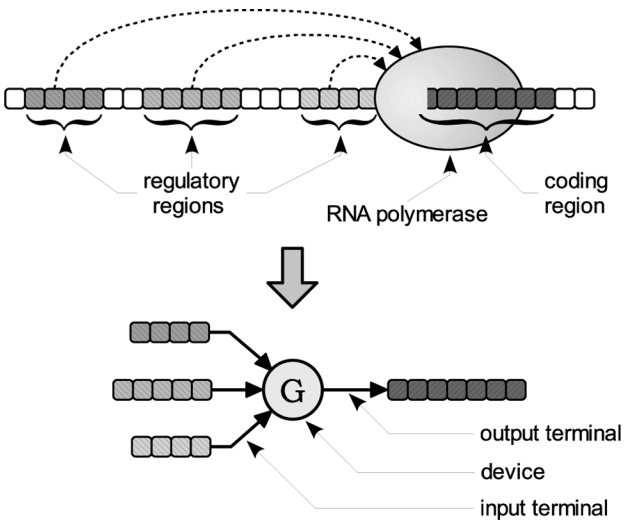


Fig. 2. A gene contained in a fragment of chromosome (represented schematically by the sequence of rounded boxes in the top part of the figure) can be interpreted as a device with input and output terminals, and with sequences of nucleotides associated with the terminals (bottom). The regulatory regions of the gene correspond to the sequences associated with the input terminals, the coding regions correspond to the sequence associated with the output terminals.

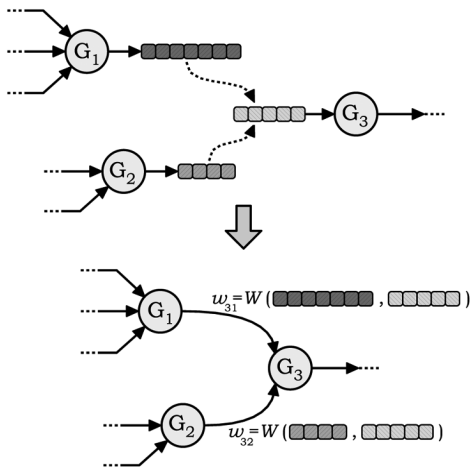


Fig. 3. Using for the genes the representation of Fig. 2, the strength of the interaction between two genes in a GRN can be assumed as obtained through a device interaction map $W(s_j, s_i)$ from pairs of sequences (s_j, s_i) to values w_{ij} of interaction strength (bottom).

This approach permits a much more compact description of the network than the direct encoding. In general, the adoption of a developmental representation requires special care in the definition of the genetic operators, to ensure that the resulting genomes define a valid developmental process. Moreover, it is very difficult to define a developmental process that displays evolvability. The complex relationship linking the genome to the developed network is typically opaque and can lead to systems where the effects of the genetic operators are either insignificant or catastrophic in terms of network functionality.

C. Implicit Interaction

An alternative genetic representation for the topology and sizing of analog networks can be derived from the observation of biological GRNs. Biological GRNs are composed by genes that interact through the production of gene regulatory proteins. Each gene contains one or more sequences of nucleotides (i.e., characters of the genetic alphabet) called coding regions and one or more sequences called regulatory regions. The coding regions specify—via a molecular machine called RNA polymerase—the production of regulatory proteins that can potentially interact with all the regulatory regions existing in the genome. The actual influence of one gene on another is determined by the interaction between the regulatory proteins produced by the first and the regulatory regions of the second. In other words, the strength of the interaction between two genes is not explicitly encoded in the genome but follows implicitly from the characteristics of the sequences of characters constituting the regulatory and coding regions and from the characteristics of the environment in which the genome is immersed. It is, therefore, possible to interpret genes as devices with input and output terminals and with sequences of nucleotides associated with the terminals (Fig. 2), and to represent the interaction between genes in terms of a *device interaction map*, which transforms pairs of character sequences associated with two distinct device terminals into a numeric value that characterizes the link connecting the two terminals (Fig. 3).

In this representation of biological GRNS, we can identify two aspects that are crucial to the possibility of their evolutionary synthesis. The first is the possibility to change the number and type of devices that are represented in the genome. The second is the possibility to change the strength of the interaction that characterizes the connections between pairs of devices. In biological genomes, these changes are brought about by the action of genetic operators that go beyond the simple operators of mutation and crossover that are typically used in genetic algorithms. In particular, it is largely recognized that the duplication, deletion, and transpositions of fragments of genome play a fundamental role in the evolution and complexification of biological organisms.

Several artificial evolutionary systems exist that use an implicit representation of the interactions inspired from biological GRNs. For example, [25]–[27] study the possibility of defining a simple model capable of exhibiting a biologically plausible dynamics of gene activation. The genome is constituted by a string of characters from a finite alphabet and the device interaction map is defined in terms of matching between substrings of the genome constituting the coding regions and substrings constituting the regulatory regions. Other approaches described in the literature use more sophisticated device interaction maps [28]–[33]. However, all the existing approaches are either limited to the representation of GRNs or define the device interaction map in terms of some simple kind of sequence matching that leads to a poor tolerance for genome reorganizations or limits the possibility of representing and evolving the topology or the sizing of the network.

II. Analog Genetic Encoding (AGE)

In this paper, we describe a new genetic representation called analog genetic encoding (AGE) which is inspired by the working of biological GRNs. AGE applies to the representation and evolution of the topology and sizing of all kinds of analog networks, not only to GRNs. In particular, the device interaction map is defined so as to permit application of all the genetic

operators mentioned above which are instrumental to the evolutionary complexification of the networks. We will show that the resulting evolutionary system displays state-of-the-art performance in the evolutionary synthesis of analog electronic circuits.

A. Overview

The basic idea is to define a genetic representation that admits the interpretation illustrated in Figs. 2 and 3. The AGE genome is constituted by one or more strings of characters (called chromosomes) from a finite genetic alphabet $\mathcal{G}$. The experimenter defines a *device set* which specifies the kind of devices that can appear in the network. For example, the device set of an evolutionary experiment aimed at the synthesis of an analog electronic circuit could contain a few types of transistors, and the device set of an evolutionary experiment aimed at the synthesis of a neural network could contain a few types of artificial neuron models. The experimenter also specifies the number of terminals of each kind of device. For example, a bipolar transistor has three terminals, a capacitor has two terminals, and an artificial neuron could be specified as having one output terminal and one input terminal, or, alternatively as having one output terminal and several input terminals. The experimenter also specifies the number of evolvable parameters of each device. For example, in evolving electronic circuits one could specify completely predefined bipolar transistors (no evolvable parameters) and specify capacitors with evolvable capacitance value (one evolvable parameter).

Fig. 4 represents schematically the modus operandi of AGE. The AGE genome contains one gene for each device that will appear in the network decoded from the genome. Each gene contains one region for each terminal and one region for each evolvable parameter specified for that kind of device (Fig. 4, top). In an initial stage of the decoding process, the devices are "extracted" from the genome, and the sequences of characters corresponding to the gene regions are associated with the terminals and the evolvable parameters of the devices (Fig. 4, center). Then, the device interaction map (and the parameter map, discussed below) is applied to this collection of devices to connect them and finally produce an analog network (Fig. 4, bottom).

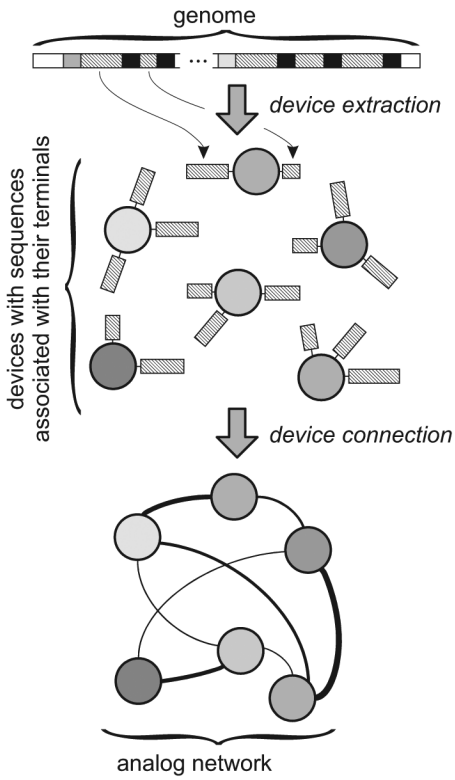


Fig. 4. Simplified representation of the workings of AGE. The encoding of parameters is not illustrated here.

B. Device Representation

In order to define the representation of the regions of the genome which correspond to the devices and to their terminals and parameters, we define a collection of specific sequences of characters that we call tokens. One specific *device token* is defined by the experimenter for each element of the device set. The device token signals the start of a fragment of genome that encodes an instance of the corresponding device. The experimenter also defines a *terminal token* and a *parameter token* whose role is to delimit the sequences of characters that must be extracted from the genome and associated with the terminals and with the parameters of the devices (Fig. 5, top). When the terminals of a device are not interchangeable, the order of association must be specified by the experimenter.

C. Device Extraction

To decode the devices encoded in the genome, each chromosome is scanned in search of a device token. If one is found, the fragment of genome starting after the token is scanned in search of all the terminal and parameter tokens required by the corresponding device. If all the required tokens are found before the next device token or before the end of the chromosome, a device—for the moment unconnected—is created and the sequences of characters delimited by the tokens are associated with the terminals and parameters of the device. Then, another device token is searched in the remaining genome, until the entire genome has been examined. Fig. 5 shows an example of device extraction.

D. Device Connection and Parameter Assignment

The result of the process of device extraction is a collection of unconnected devices that have sequences of characters associated with their terminals and with their evolvable parameters. To turn this collection of devices into an actual analog network, we need to connect the devices and assign actual values to their evolvable parameters.

1) Device Interaction Map: In order to connect the devices, the device interaction map is applied to all pairs of sequences associated with distinct terminals. This gives a value for each possible link between the devices extracted from the genome (Fig. 6). Note that the device interaction map can produce a value that corresponds to the absence of direct interaction. In this case, no link will be established between the corresponding terminals. The device interaction map must be at least in part specific of the kind of analog network considered. For example, in the case of electronic circuits, the device interaction map will produce values of conductance, whereas for neural networks it will produce weight values, which are typically dimensionless.

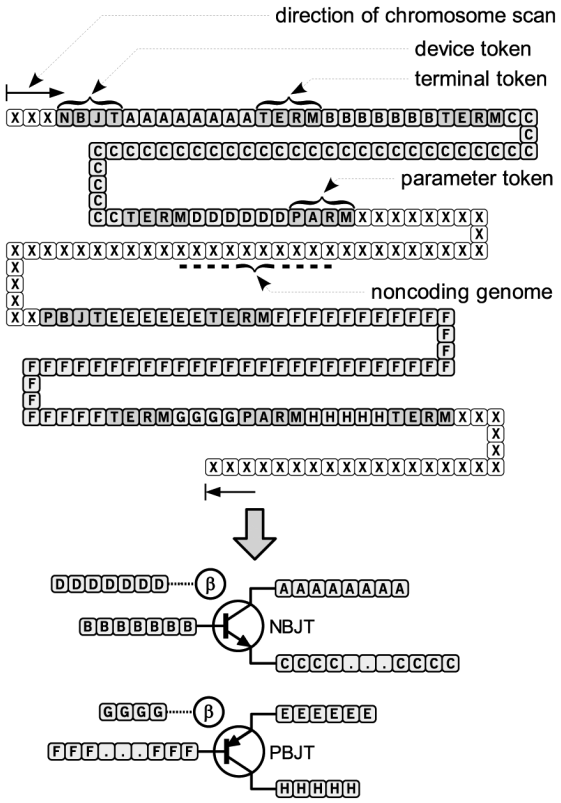


Fig. 5. (Top) A chromosome encoding two three-terminal devices with one evolvable parameter. The NBJT and PBJT device tokens signal the start of the fragments of genome coding for the devices. The terminal token TERM and the parameter token PARM signal the end of a sequence of characters that is associated with a terminal or with a parameter of the device and, possibly, the start of another sequence associated with a terminal or with a parameter. The fragments that do not correspond to a token or to a sequence associated with a terminal or with a parameter, constitute noncoding fragments of the genome. (Bottom) The devices are extracted from the genome following the procedure described in Section II-C. Note that terminal and parameter tokens are left free to appear and mix in any order in the fragment of genome that codes for a device, making the representation more tolerant of genome reorganizations.

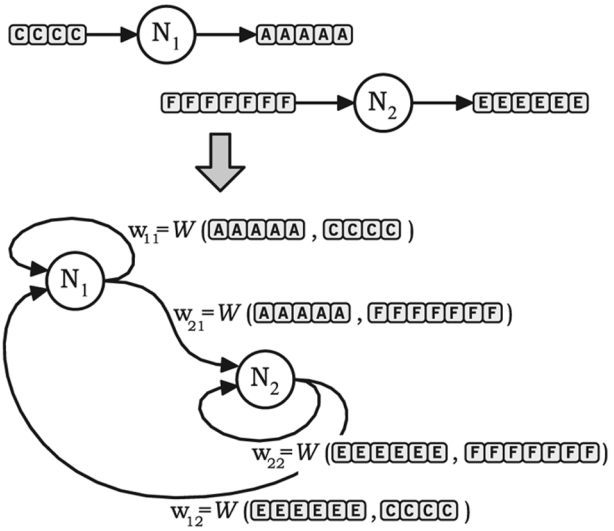


Fig. 6. Two artificial neurons extracted from a genome and having sequences of characters associated with their input and output terminals (top) are connected, and thus transformed into a neural network (bottom) by applying the device interaction map $W(s_i, s_j)$ that associates link weights w_{ij} with pairs of character sequences.

To make AGE a representation that can be adapted with minimal effort to the various kinds of analog networks, it is expedient to distinguish the network-specific component of this map from a generic component. In this way, it is possible to reuse this generic component for any kind of analog network. For this reason, we write the device interaction map as a composed map $N(L(s_1, s_2))$ formed by a generic *sequence interaction map* $L(s_1, s_2)$ that transforms pairs of sequences into abstract sequence interaction values i, and by a *network-specific interaction map* $N(i)$ that transforms sequence interaction values into network-specific numeric values characterizing the links.

Examples of network-specific interaction maps will be presented later, in the context of specific evolutionary experiments. In these experiments, the value of the sequence interaction map $L(s_1, s_2)$ corresponds to the value of the local alignment score between the two sequences s_1 and s_2 [34]. The local alignment of sequences has many advantages with respect to simpler techniques of sequence comparison such as exact matching or Hamming distance: it operates on pairs of sequences of different and arbitrary length; it permits the generation of a whole range of values of interaction, which can be easily changed through the parameters that define the alignment; the effect of mutations of the sequences on the value of interaction can also be changed through the parameters that define the alignment; finally, it can be made highly redundant, in the sense that many different pairs of sequences correspond to the same value of interaction: this facilitates the evolutionary generation of a given value of interaction. The reader is referred to [35] for a more extensive discussion of these properties in relation to the choice of the size of the genetic alphabet and of the other parameters of the local alignment.

2) Parameter Map: The assignment of a value to the evolvable parameters of the devices is based on the definition of a *parameter map* which transforms the sequence s of characters extracted from the genome and associated with the parameter into the value of the parameter. A simple way to define the parameter map is to interpret s as an integer written in base $|\mathcal{G}|$ and to map this integer into the actual value of the parameter. Using this approach, however, the nature of the parameter map would be very different from that of the device interaction map described above, which acts instead on pairs of sequences. In particular, this would be true for the consequences of mutations and reorganizations of the genome on the value of interaction strength and parameter values. For this reason, it is worth considering the possibility of defining the parameter map as a function that produces a parameter value given the pair constituted by the sequence s and a fixed sequence of characters s_f. In this way, both the device interaction map and the parameter map correspond to functions that take as arguments pairs of sequences and produce numeric values.

E. External Connections

To perform its function an analog network must be connected to external devices that provide input signals and accept output signals. The external devices are defined by the experimenter according to the kind of application. For example, the external devices for an analog electronic circuit might be a power supply

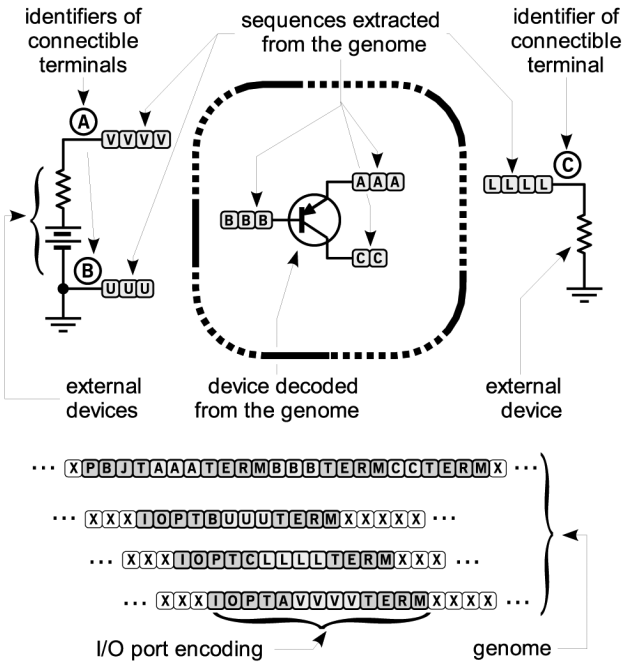


Fig. 7. An example of use of I/O ports for the connection of external devices to the evolving circuit The external devices defined by the experimenter are drawn outside the dotted box. The genome contains the tokens `IOPTA`, `IOPTB`, and `IOPTC` corresponding to the three connectible external terminals, which determine the association of sequences extracted from the genome with the terminals of the external devices.

and a resistive load, whereas the external devices for the evolution of an artificial neural network controlling a robot might be the sensors and the actuators of the robot.

In AGE, the connections between the evolved network and the preassigned external devices is obtained by associating sequences of characters with the terminals of the external devices that might possibly be connected to the evolving network. In this way, the use of a sequence-based device interaction map can be extended to the evolution of the interactions between these external devices and the evolved network. In the experiments reported below, the association of sequences with the terminals of external devices is based on the definition of a specialized device called *I/O port*. The experimenter assigns a unique identifier to each terminal of external device which must be potentially connected to the evolved network. Using this identifier, a different I/O port token is defined for each external terminal. These tokens signal the presence of fragments of genome that encode the I/O ports, as illustrated in Fig. 7.

F. Genetic Operators

Many kinds of reorganizations can be applied to the AGE genome without compromising its decodability. In AGE, the fragments of the genome encoding the devices can be of variable length and be located anywhere within the genome. The genome itself is obviously a variable length genome. Note that from the point of view of most genetic operators, the tokens for devices, terminals, and parameters have no special meaning. In particular, the tokens are not protected from the action of the genetic operators, which can invalidate them.

Below, we describe the genetic operators that are applied to the AGE genome in the experiments reported in this paper. To avoid repetition, only the action of the operator is described and it is implicitly assumed that these operators are applied probabilistically to randomly chosen parts of the genome and using for insertions and substitutions elements randomly chosen in the suitable set.

- *Character deletion, insertion, and substitution*. A character is removed, inserted, or substituted in the genome.
- *Chromosome fragment deletion, transposition, and duplication*. Two points are chosen in a chromosome and the intervening genome fragment is deleted, or transferred at another point of the genome, leaving in place the original fragment in the case of duplication.
- *Device insertion*. The descriptor of a device is inserted in the genome. The sequences of characters associated with the terminals and the evolvable parameters of the devices can be randomly generated or can be obtained by sampling the sequences associated with the terminals of the devices existing in the genome.
- *Chromosome deletion and duplication*. A chromosome is deleted or duplicated. The duplication can either append to the chromosome a copy of itself, or create a new chromosome.
- *Homologous crossover*. A tentative crossover point is chosen within the first of the two chromosomes that are candidates for recombination. Starting at the chosen point, a sequence of characters of predefined length (but short with respect to the typical chromosome length) is taken as template from the first chromosome. A sequence of characters sufficiently similar to the template is then searched in the second chromosome. If a sequence of sufficient similarity is found, its starting point becomes the crossover point in the second chromosome; otherwise, no recombination occurs. When there is recombination, the fragments of chromosomes starting at the crossover points are swapped to generate the recombined chromosomes. In the experiments reported below, the criterion of similarity is the existence in the second chromosome of a sequence that matches exactly the template taken from the first chromosome. Other techniques such as local alignment of sequences could be used as well. The reason for defining homologous crossover is that, since the AGE genome does not have a fixed structure and length, the traditional crossover can be expected to produce just macromutations rather than recombination of structures. The homologous crossover operator favors the recombination of fragments of chromosome that can be considered homologous and can be expected to represent homologous features of the encoded analog network.
- *Genome duplication*. The whole genome is duplicated. The duplication can either append to each existing chromosome a copy of itself, or create a new chromosome for each existing one.
- *Genome trimming*. The application of the genetic operators described so far can result in the transformation of fragments of genome previously coding for devices into noncoding genome. The presence of this noncoding genome does not prevent the decoding of the remaining genome, and can even conceivably play the role of an evolutionary

substitution matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2
B	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1

⋮

indel vector

	A	B	⋯
–	-3	-3	

Fig. 8. The substitution matrix (top) and indel vector (bottom) used for the calculation of the local alignment score in the experiments reported in this paper. The matrix is circulant and the indel score is the same for all characters of the genetic alphabet.

useful repository of genetic fragments and pseudogenes. However, to prevent the excessive increase of the genome size, it is useful to also have the possibility to free the genome from most of the noncoding genome. The genome trimming operation removes from the genome all the noncoding genome, except possibly short padding fragments between the coding regions.

- *Generation of initial population*. Although not strictly a genetic operator, the generation of a population is an operation that is traditionally required to start the evolutionary process. A first possibility is to produce the initial genomes by performing one or more operations of device insertion on a background of random characters. Another possibility is to start from a previously evolved network, or from a related network and evolve a collection of sequences that can be used to build the AGE representation for it [35]. This permits the incorporation of expert knowledge in the evolutionary process.

III. Experiments

In the series of examples presented in this section, AGE is applied to the evolutionary synthesis of analog networks. First, a series of three experiments of evolution of analog electronic circuits is presented, followed by an example of evolution of a neural network. The performance of the evolved electronic circuits was evaluated in simulation using SPICE [36]. To avoid the simulation problems potentially caused by dangling device terminals, a 1 GΩ resistor was inserted between any dangling terminal of the decoded circuit and the ground node. In all the experiments, the size of the genetic alphabet is $|\mathcal{G}| = 26$ and the scoring matrices are those shown in Fig. 8.

A. Voltage Reference

The first network evolution problem is aimed at the synthesis of a voltage reference electronic circuit. Fig. 9 shows the devices of the predefined external circuit. The goal of the evolutionary experiment is the synthesis of a circuit producing a fixed output voltage $V_o^* = 2$ V on the load resistor when the source voltage V_c varies in the range $4\text{ V} \leq V_c \leq 6\text{ V}$ and the circuit temperature T varies in the range $0^\circ\text{ C} \leq T \leq 100^\circ\text{ C}$.

1) Fitness Function: The input voltage range is discretized into intervals of width $\Delta V_c = 0.1$ V, resulting in 21 discrete values $\{V_{c_i}\}$. The temperature range is discretized into the five

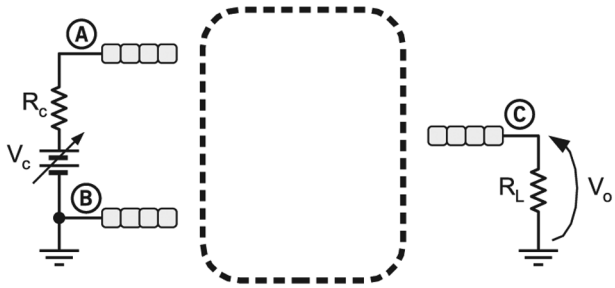


Fig. 9. The devices of the external network in the experiment of evolution of a voltage reference. The external network is an electronic circuit composed of a variable voltage source V_c with a series resistance $R_c = 1\text{ k}\Omega$, and by a load resistor $R_L = 10\text{ k}\Omega$ across which the output voltage V_o of the evolved network is measured.

values of the set $\{T_j\} = \{0^\circ\text{ C}, 25^\circ\text{ C}, 50^\circ\text{ C}, 75^\circ\text{ C}, 100^\circ\text{ C}\}$. The output voltage $V_{o_{ij}}$ is computed in correspondence of each input voltage value V_{c_i} and circuit temperature T_j pair. The fitness function is defined as $f = -\sum_{i,j} \varepsilon_{ij}$, where

$$\varepsilon_{ij} = \begin{cases} \left(V_{o_{ij}} - V_o^*\right)^2, & \text{if } \left|V_{o_{ij}} - V_o^*\right| \geq 0.01\text{ V} \\ 0, & \text{if } \left|V_{o_{ij}} - V_o^*\right| < 0.01\text{ V} \end{cases}.$$

2) Network-Specific Interaction Map: The network-specific interaction map corresponds to a logarithmic quantization of the conductance values g. The sequence interaction score $i_{\min} = 20$ is mapped to the value of conductance $g_{\min} = 10^{-6}$ S. The sequence interaction score $i_{\max} = 68$ is mapped to a value of conductance $g_{\max} = 1$ S. All values of sequence interaction below $i_{\min}$ are mapped to the value of conductance $g_o = 0$ S, and all values above $i_{\max}$ are mapped to the value $g_\infty = \infty$. The values of sequence interaction between $i_{\min}$ and $i_{\max}$ are orderly mapped on a set of logarithmically distributed resistance values in the range from 1 Ω to 1 MΩ, with $n_d = 8$ values of resistance per decade.

3) Evolutionary Algorithm and Parameters: The evolutionary algorithm used in the current experiment is a standard generational genetic algorithm that uses tournament selection and elitism. The values of the parameters are listed below:

- population size 100;
- tournament size 5;
- elite size 1;
- prob. of character substitution 0.001;
- prob. of character insertion 0.001;

- prob. of character deletion 0.001;
- prob. of fragment duplication 0.01;
- prob. of fragment deletion 0.01;
- prob. of fragment transposition 0.01;
- prob. of chromosome duplication 0.001;
- prob. of chromosome deletion 0.001;
- prob. of genome duplication 0.001;
- prob. of individual genome trimming 0.01;
- prob. of population genome trimming 0.01;
- length of spacing genome in trimming 20;
- prob. of device insertion 0.01;
- prob. of homologous crossover 0.1;
- num. of matching chars for hom. cross. 10.

The probabilities of application of the genetic operators listed above are not critical. They were chosen heuristically based on the information collected on a series of test runs, with the objective of keeping reasonably low the disruptive effect of the operators.

4) Device Set: The device set for the current experiment contains one NPN (BC846B) and one PNP (BC856B) bipolar junction transistor. The listings of the SPICE models of the transistors can be found in [35].

5) Initial Population and Device Insertion: Each individual of the initial population is obtained by generating a genome composed of one chromosome containing the representation of five devices randomly chosen in the device set and the representation of two copies of each kind of I/O port associated with the external circuit. The sequences associated with the terminals are randomly generated sequences of characters of length 20. Fragments of genome representing distinct devices are separated in the genome by randomly generated spacer sequences of length 20. Spacer sequences are also inserted at the start and at the end of the chromosome.

6) Results: Fig. 10 shows the evolutionary graphs for five repetitions of the experiment. Fig. 11 shows an example of voltage reference circuit evolved in these runs. Fig. 12 illustrates the output of this circuit. Each run shown in Fig. 10 took on average about nine days to complete on a single-processor Pentium 4 machine. The SPICE simulation time averaged over all runs and generations is about 4.2 s per generation and the genome decoding time is about 3.6 s per generation. These values of time are also representative of the other experiments of electronic circuit evolution considered in this paper.

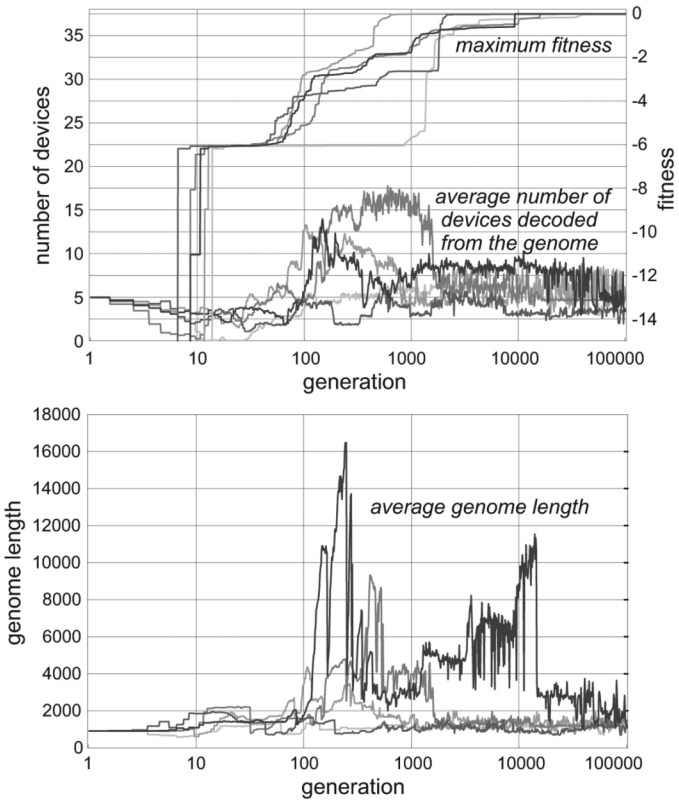


Fig. 10. The evolutionary graphs for five runs of the experiment aimed at the evolution of a voltage reference electronic circuit.

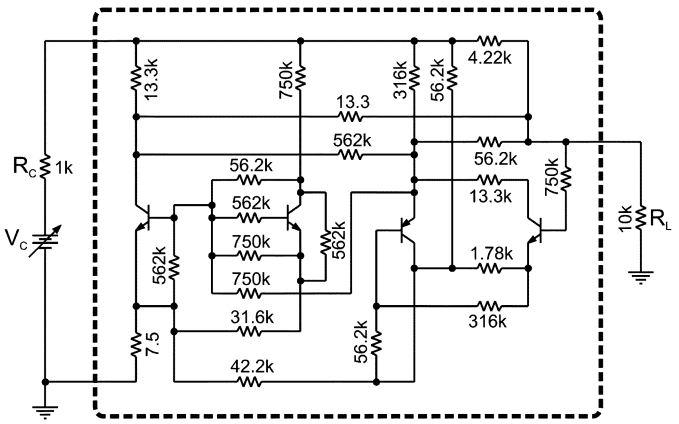


Fig. 11. Example of evolved voltage reference circuit.

B. Temperature Sensor

The second network evolution problem is aimed at the synthesis of a temperature sensing electronic circuit. Fig. 13 shows the devices of the predefined external circuit. The goal of the evolutionary experiment is the synthesis of a circuit producing across the load resistor an output voltage V_o that is proportional to the circuit temperature T in the range $0^\circ\,\mathrm{C} \leq T \leq 100^\circ\,\mathrm{C}$, with a proportionality coefficient of $\gamma = 0.1\mathrm{V}/^\circ\,\mathrm{C}$.

1) Fitness Function: The output voltage V_{o_i} of the evolved circuits is evaluated in correspondence of the discrete set $\{T_i\} = \{0^\circ\,\mathrm{C}, 5^\circ\,\mathrm{C}, 10^\circ\,\mathrm{C}, 15^\circ\,\mathrm{C}, \ldots, 100^\circ\,\mathrm{C}\}$ of 21 equispaced temperature values. The fitness function f is defined as $f = -\sum_i \left(V_{o_i} - V_{o_i}^*\right)^2$, where $V_{o_i}^* = \gamma T_i$ is the desired output voltage at the temperature T_i.

2) Maps and Parameters: The network specific interaction map, the device set, the initial population, and the parameters of the evolutionary algorithm are the same as in previous experiment.

3) Results: Fig. 14 shows the fitness graphs for five repetitions of the experiment. Fig. 15 shows an example of evolved temperature sensing circuit. The behavior of this circuit is illustrated in Fig. 16.

C. Gaussian Function Generator

The third circuit evolution problem is aimed at the synthesis of a Gaussian function generator. Fig. 17 shows the devices of

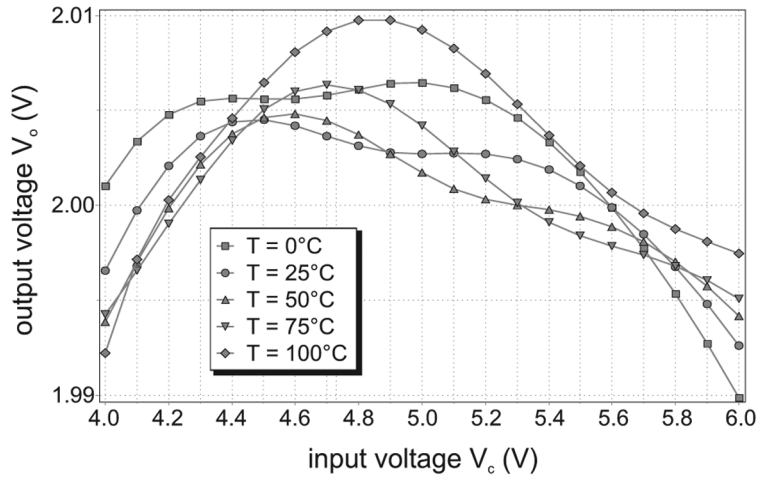


Fig. 12. The output voltage V_o of the evolved voltage reference circuit shown in Fig. 11, plotted as a function of the input voltage V_c.

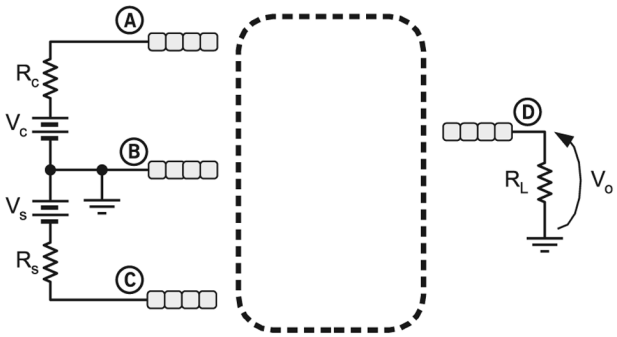


Fig. 13. The devices of the external network in the experiment of evolution of a temperature sensing electronic circuit. The external network is composed of two fixed voltage sources $V_c = 15$ V and $V_s = 5$ V, two series resistance $R_c = 1\,\mathrm{k}\Omega$ and $R_s = 1\,\mathrm{k}\Omega$, and by a load resistor $R_L = 10\,\mathrm{k}\Omega$ across which the output voltage V_o of the evolved network is measured.

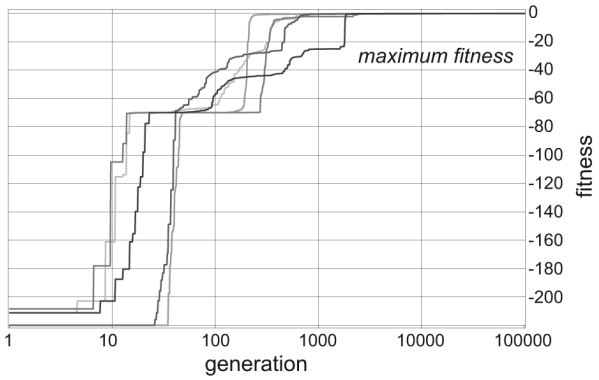


Fig. 14. The fitness graphs for five runs of the experiment aimed at the evolution of a temperature sensor circuit.

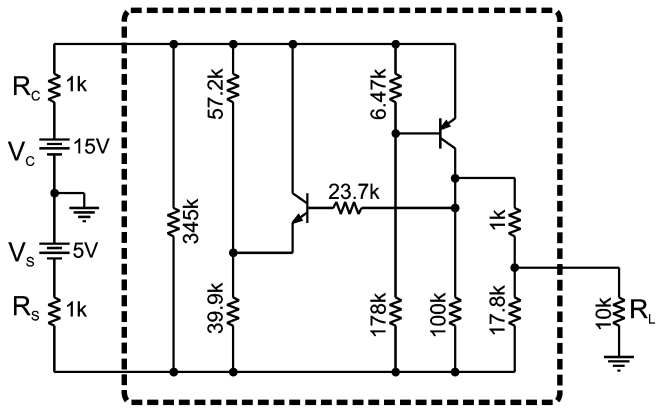


Fig. 15. Example of evolved temperature sensing circuit.

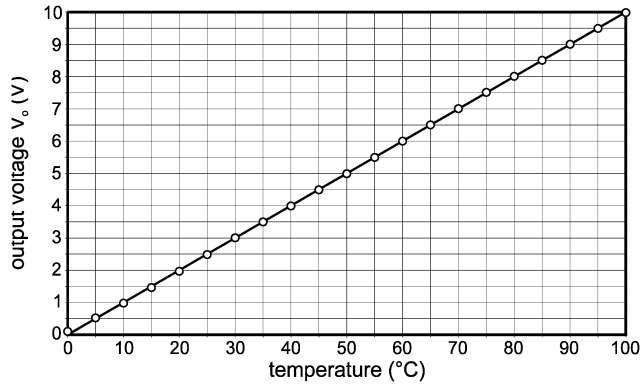


Fig. 16. The graph of the output voltage V_o of the evolved temperature sensing circuit shown in Fig. 15 plotted as a function of the circuit temperature T. The background line shows the ideal behavior.

the predefined external circuit. The goal of the evolutionary experiment is the synthesis of a circuit producing through the load voltage source V_L an output current I_o that is a (non-normalized) Gaussian function of the variable input voltage V_c in the range $2\ \mathrm{V} \leq V_c \leq 3\ \mathrm{V}$, with a peak value $I_{o_{\max}} = 80\ \mathrm{nA}$ in correspondence of $\overline{V}_c = 2.5\ \mathrm{V}$, and $\sigma = 0.1\ \mathrm{V}$.

1) Fitness Function: The output current I_{o_i} of the evolved circuits is evaluated in correspondence of the discrete set $\{V_{c_i}\} = \{2\ \mathrm{V}, 2.01\ \mathrm{V}, 2.02\ \mathrm{V}, \ldots, 3\ \mathrm{V}\}$ of 101 equispaced values of voltage. The fitness function f is defined as $f = -10^{14} * \sum_i \left(I_{o_i} - I^*_{o_i}\right)^2$, where $I^*_{o_i} = I_o(V_c)$ is the target output current.

2) Device Set: The device set for the current experiment contains two MOSFET transistors: a PMOS and an NMOS.

3) Maps and Parameters: The network specific interaction map, the device set, the initial population, and the parameters

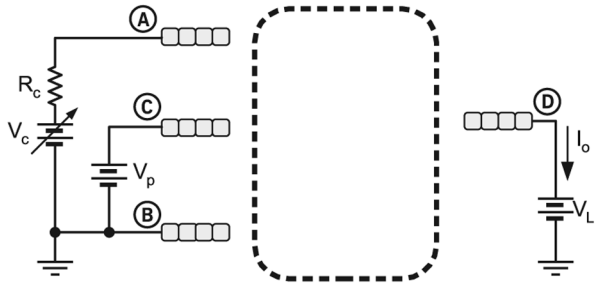


Fig. 17. The devices of the external network in the experiment of evolution of a Gaussian function generator circuit. The external network is composed of a variable voltage source V_c connected to a series resistance $R_c = 1\ \Omega$, a fixed voltage source $V_p = 5\ \mathrm{V}$, and another fixed voltage source $V_L = 2.5\ \mathrm{V}$ through which the output current I_0 of the evolved network is measured.

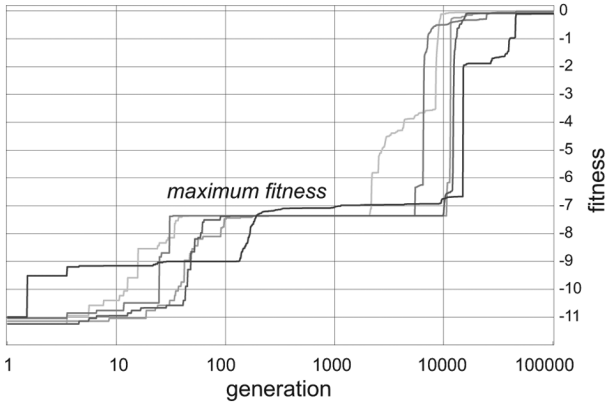


Fig. 18. The fitness graphs for five runs of the experiment aimed at the evolution of a Gaussian function generator circuit.

of the evolutionary algorithm are unchanged from the previous experiments, except for the number of devices in the initial genome which is set to 10.

4) Results: Fig. 18 shows the fitness graphs for five repetitions of the experiment. Fig. 19 shows an example of evolved Gaussian function generator. The input–output behavior of this circuit is illustrated in Fig. 20.

D. XOR Neural Network

The goal of this final experiment of network evolution is the synthesis of a neural network realizing the two-input XOR function. Fig. 21 shows the predefined external devices.

1) Fitness Function: The fitness is defines as $f = -(\varepsilon(0,0) + \varepsilon(1,0) + \varepsilon(0,1) + \varepsilon(1,1))$, as shown in the equation at the bottom of the page, where $Y(X_0, X_1)$ is the function realized by the network and $Y^*(X_0, X_1)$ is the XOR function.

2) Network-Specific Interaction Map: The network-specific interaction map realizes a logarithmic quantization of the connection weights. Using the symbolism of the quantization of

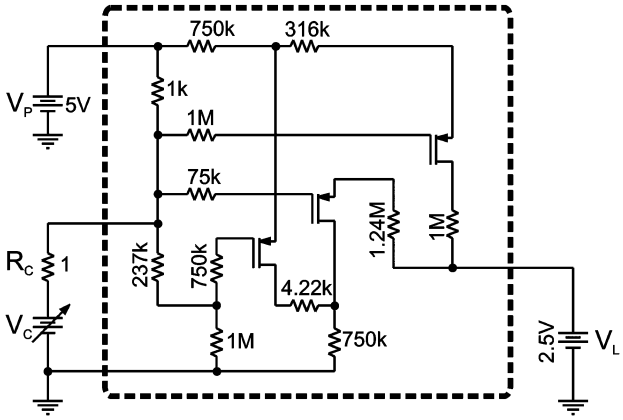


Fig. 19. Example of evolved Gaussian function generator.

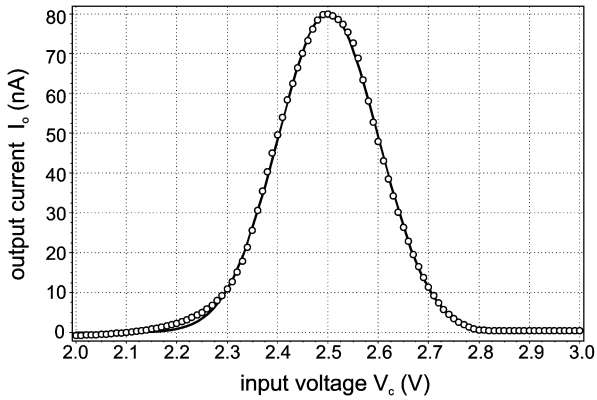


Fig. 20. The output current I_0 of the evolved Gaussian function generator circuit shown in Fig. 19 plotted as a function of the input voltage V_c. The background line represents the ideal relationship between V_c and I_o.

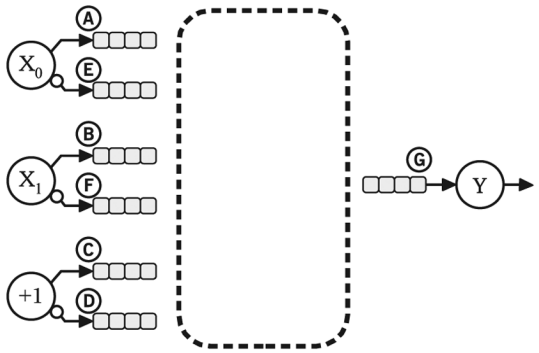


Fig. 21. The devices of the external network in the experiment of evolution of a neural network realizing the XOR function. X_0 and X_1 are the two input neurons, the "+1" neuron is the fixed input bias neuron, and Y is the output neuron. In order to simplify the quantization of the connection weights, all the input neurons produce also the inverted signal.

conductance values described in Section III-A, we have $w_o = 0$ (absence of link), $w_{\min} = 0.001$, $w_{\max} = 1000$, $w_\infty = 1000$, $i_{\min} = 1$, $n_d = 6$, and $i_{\max} = 37$.

$$\varepsilon(X_0, X_1) = \begin{cases} 0, & \text{if } |Y(X_0, X_1) - Y^*(X_0, X_1)| < 0.001 \\ (Y(X_0, X_1) - Y^*(X_0, X_1))^2, & \text{otherwise} \end{cases}$$

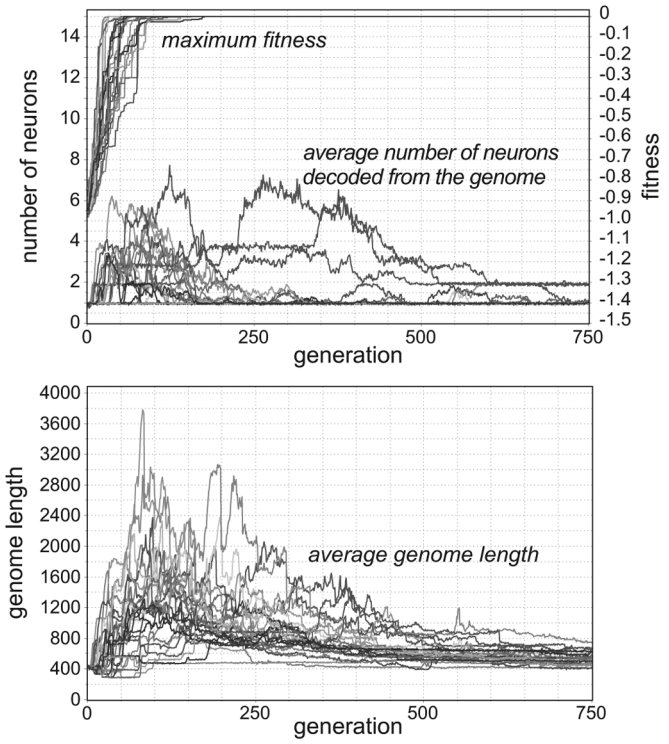


Fig. 22. The fitness graphs obtained in 25 runs of the experiment aimed at the synthesis of a neural network realizing the XOR function.

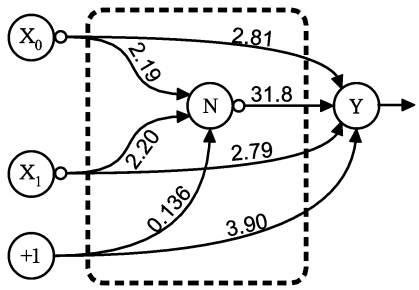


Fig. 23. Example of evolved neural network realizing the XOR function.

3) Device Set: The devices available to the evolving circuit are two kinds of sigmoid neurons—one excitatory and the other inhibitory—implementing the function $y = 1/\left(1+\exp\left(-5\sum_j w_j x_j\right)\right)$.

4) Maps and Parameters: The network specific interaction map, the device set, and the parameters of the evolutionary algorithm are the same as in the experiment of evolution of a voltage reference. The genome of the individuals of the initial population encodes one neuron.

5) Results: Fig. 22 shows the evolutionary graphs for 25 repetitions of the experiment. The curves of the maximum fitness show that all the runs evolved, within 200 generations, at least one network realizing the required function with the given tolerance. Fig. 23 shows an example of evolved neural network.

IV. Discussion

1) Performance: There exist no established benchmarks for the evaluation of experiments of evolution of analog circuits. However, the three problems to which AGE has been applied in this paper are inspired from problems that were tackled with genetic programming (GP) in [12]. The problems considered here and those described in [12] have the same target functionality but differ in the transistor models used in the simulator and in the values of some of the resistors of the external circuits. Despite these differences, and despite the availability of just one GP result for each problem, it is useful to consider side by side the results obtained with the two methods. To this end, the circuits evolved with AGE were evaluated with the fitness function used in the GP experiments, which consists of the sum of the absolute value of the deviations from the target output value weighted by a factor of 10 if the deviation is greater than a preassigned limit value (so that smaller fitness values correspond to better circuit performance). Each output value falling within the limit is defined as a hit. For both AGE and the GP experiments, the number of evaluations required to produce the best circuit of the run is considered.

TABLE I
Comparison of GP and AGE for Voltage Reference

Method	Evaluations	Hits	Fitness	Trans.	Res.
GP (1 run)	$5.12 \cdot 10^7$	85.7%	6.6	54	13
AGE (avg. 5 runs)	$5.6 \cdot 10^6$	93.5%	2.64	6.8	63.4

TABLE II
Comparison of GP and AGE for Temperature Sensor

Method	Evaluations	Hits	Fitness	Trans.	Res.
GP (1 run)	$1.6 \cdot 10^7$	76.2%	26.4	48	6
AGE (avg. 5 runs)	$6.5 \cdot 10^6$	97.1%	1.13	4.8	23

TABLE III
Comparison of GP and AGE for Gaussian Function Generator

Method	Evaluations	Hits	Fitness	Trans.	Res.
GP (1 run)	$2.3 \cdot 10^7$	100%	0.094	13	1
AGE (avg. 5 runs)	$4.3 \cdot 10^6$	97.4%	0.3	7.2	28.8

Tables I –III show the average number of circuit evaluations, the number of hits, the fitness value, and the number of transistors and resistors of the best evolved circuits obtained with GP and AGE. The tables show that for the voltage reference and temperature sensor problems, AGE produced better results in terms of fitness and number of hits with a tenfold and twofold reduction, respectively, of the number of circuit evaluations with respect to GP. In the Gaussian function case, the unique GP circuit achieved a slightly better performance than the average AGE circuit, but using about an order of magnitude more circuit evaluations. The circuits produced by AGE contain on average less transistors and more resistors than those produced by GP. Some of the AGE runs produced circuits of the kind shown in Figs. 11, 15, and 19, which are much more compact than the typical GP product that performs the same function. Summing up, in the evolution of electronic circuits, AGE compares well with GP in terms of performance of the evolved circuits and computational effort required to evolve them. In all the experiments with AGE, the length of the genome remained within reasonable limits and did not manifest any phenomenon of bloat, as exemplified by Figs. 10 and 22. The absence of bloat is presumably due to the combined effect of the genome trimming operator and of the possibility of invalidation of the device tokens by genetic

operators mentioned in Section II-F. Since there is no evolutionary pressure to preserve from invalidation the tokens of the devices that do not contribute to the circuit functionality, these devices are likely to be invalidated by some genome mutation and then trimmed from the genome.

In the experiments of circuit evolution, the decoding of the genome took almost the same time as the circuit simulations. However, in this experiment, the evaluation of fitness merely requires the calculation of the operating point of the evolved circuits. In experiments requiring more complex circuit simulations such as time transients the decoding time can be expected to become negligible with respect to the simulation time. Note that the 100 individuals population size with which AGE can work is much smaller than the several hundred thousand individuals of the typical population size required by GP to solve the same problems.

2) Complexification: The AGE runs display a variation in the number of devices encoded in the genome. For example, Fig. 10 shows an initial decrease in the number of devices followed by an increase after a few tens of generations, which later subsides. The initial decrease corresponds to the elimination of most of the randomly generated initial devices. The subsequent increase corresponds to the introduction of devices produced by the genetic operators during the evolutionary process. This phenomenon was observed in the solution of all the electronic circuit problems and is apparent also in the XOR runs documented in Fig. 22. These latter runs were continued for several hundreds of generations after the attainment of the required solution and illustrate the eventual reduction in most runs of the number of devices to the minimum that is known to be required to solve the XOR problem. This shows that the AGE representation permits the complexification and decomplexification of the network according to the needs of the evolutionary process.

3) Automatic Feature Selection: The comparison of the evolved XOR network of Fig. 23 with the predefined external network of Fig. 21 shows that the evolutionary process has selected the inverted input signals for X_0 and X_1 and the directed bias output as input signals disregarding the directed input and inverted bias. This shows that the AGE representation permits the automatic selection of the subset of inputs (and outputs) required to solve the given problem. In machine learning parlance AGE is intrinsically capable of solving the feature selection problem.

V. Future Work

1) Scalability: The evolution of large sparsely connected networks could be hindered by the randomly generated interactions between the sequences associated with the terminals. Although this problem never materialized in the experiments performed so far, its possible appearance can be prevented in several ways. A first possibility consists in the introduction of techniques for the modularization of the evolved network. A complementary approach consists in the implementation of a mechanism of interaction silencing similar to the mechanism of gene silencing by RNA interference observed in eukaryotic cells.

2) Real-World Problems: A limitation of the series of network synthesis problems considered in the previous section is that they are only in part representative of real-world network synthesis problems. The XOR problem is quite remote from actual neural network applications and electronic circuits must perform correctly in an operational envelope that is typically more complex than those considered in the previous section. The consideration of a more realistic operational envelope requires the formulation of design synthesis as a multiobjective problem with constraints, the consideration of which would have entailed the additional complications constituted by multiobjective evolutionary algorithms and prevented our focusing on the issue of the newly defined genetic encoding. Experiments are under way to assess the performance of AGE in more complex and realistic contexts and for the synthesis and reverse engineering of GRNs. The results obtained so far show that AGE compares well with the most powerful algorithms for neuroevolution existing in the literature [37].

VI. Conclusion

AGE is a new approach to the representation and evolution of generic analog networks. The structure of the AGE genome and that of the genetic operators it tolerates permits the evolutionary complexification and decomplexification of the network during the evolutionary process. We have shown that AGE is a powerful method for the synthesis of analog electronic circuits, with performances that compare well with those of a state-of-the-art evolutionary method like GP.

Acknowledgment

The authors thank D. Marbach, M. Waibel, and P. Dürr for their comments and suggestions on a preliminary version of the manuscript.

References

[1] R. S. Zebulum, M. A. C. Pacheco, and M. M. B. R. Vellasco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. Boca Raton, FL: CRC Press, 2002.

[2] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, pp. 1423–1447, Sep. 1999.

[3] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita, "Dynamic modeling of genetic networks using genetic algorithm and S-system," *Bioinformatics*, vol. 19, no. 5, pp. 643–650, Mar. 2003.

[4] J. B. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," *Proc. IEE —Circuits, Devices, Syst.*, vol. 147, no. 6, pp. 319–323, Dec. 2000.

[5] R. Zebulum, M. Vellasco, and M. Pacheco, "Variable length representation in evolutionary electronics," *Evol. Comput.*, vol. 8, no. 1, pp. 93–120, 2000.

[6] S. Ando, M. Ishizuka, and H. Iba, "Evolving analog circuits by variable length chromosomes," in *Advances in evolutionary computing*, A. Ghosh and S. Tsutsui, Eds. New York: Springer, 2003, pp. 643–662.

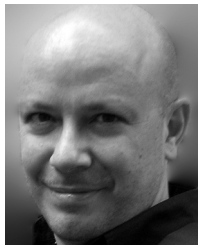
[7] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Netw.*, vol. 5, pp. 39–53, Jan. 1994.

[8] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, pp. 694–713, May 1997.

[9] J. Pujol and R. Poli, "Evolving the topology and the weights of neural networks using a dual representation," *Appl. Intell.*, vol. 8, no. 1, pp. 73–84, 1998.

[10] K. Kobayashi and M. Ohbayashi, "A new indirect encoding method with variable length gene code to optimize neural network structures," in *Proc. Int. Joint Conf. Neural Netw.*, Piscataway, NJ, 1999, pp. 4409–4412.

[11] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.
[12] J. R. Koza, F. H. Bennet, III, D. Andre, and M. A. Keane, *Genetic Programming III*. San Mateo, CA: Morgan Kaufmann, 1999.
[13] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Norwell, MA: Kluwer, 2003.
[14] J. D. Lohn and S. P. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 205–219, Sep. 1999.
[15] S.-J. Chang, H.-S. Hou, and Y.-K. Su, "Automated passive filter synthesis using a novel tree representation and genetic programming," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 93–100, Feb. 2006.
[16] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Syst.*, vol. 4, no. 4, pp. 461–476, 1990.
[17] R. Belew, "Interposing an ontogenic model between genetic algorithms and neural networks," in *Advances in Neural Information Processing, NIPS5*, S. Hanson, J. Cowan, and C. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1993, pp. 99–106.
[18] A. Cangelosi, D. Parisi, and S. Nolfi, "Cell division and migration in a 'genotype' for neural networks," *Network: Computation in Neural Systems*, vol. 5, no. 4, pp. 497–515, Nov. 1994.
[19] F. Gruau, "Automatic definition of modular neural networks," *Adaptive Behaviour*, vol. 3, no. 2, pp. 151–183, 1995.
[20] S. Nolfi and D. Parisi, "'Genotypes' for neural networks," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 431–434.
[21] P. Eggenberger, "Creation of neural networks based on developmental and evolutionary principles," in *Proc. Int. Conf. Artif. Neural Netw.*, W. Gerstner, A. Germond, M. Hasler, and J. Nicoud, Eds., Lausanne, Berlin, Switzerland, 1997, vol. 1327, pp. 337–342. [Online]. Available: http://www.ifi.unizh.ch/ailab/people/eggen/publications/97.21.pdf
[22] J. Astor and C. Adami, "A developmental model for the evolution of artificial neural networks," *Artif. Life*, vol. 6, no. 3, pp. 189–218, 2000.
[23] J. Bongard, "Evolving modular genetic regulatory networks," in *Proc. IEEE Congr. Evol. Comput.*, Piscataway, NJ, 2002, pp. 1872–1877.
[24] N. Jakobi, "Harnessing morphogenesis," in *On Growth, Form and Computers*, P. B. S. Kumar, Ed. New York: Academic Press, 2003, pp. 392–404.
[25] T. Reil, "Dynamics of gene expression in an artificial genome—implications for biological and artificial ontogeny," in *Proc. 5th Eur. Conf. Artif. Life*, D. Floreano, F. Mondada, and J. Nicoud, Eds., Berlin, Germany, 1999, pp. 457–466.
[26] J. Watson, J. Wiles, and J. Hanan, "Towards more relevant evolutionary models: Integrating an artificial genome with a developmental phenotype," in *Proc. Australian Conf. Artif. Life*, H. Abbass and J. Wiles, Eds., 2003, pp. 288–298.
[27] J. Hallinan and J. Wiles, "Evolving genetic regulatory networks using an artificial genome," in *Proc. 2nd Asia-Pacific Bioinformatics Conf.*, Y. Chen, Ed., Dunedin, New Zealand, 2004, pp. 291–296.
[28] P. Kennedy and T. Osborn, "A model of gene expression and regulation in an artificial cellular organism," *Complex Syst.*, vol. 13, no. 1, pp. 2001–2001.
[29] N. Geard and J. Wiles, "Structure and dynamics of a gene network model incorporating small RNAs," in *Proc. Congr. Evol. Comput.*, R. Sarker, R. Reynolds, H. Abbass, K.-C. Tan, R. McKay, D. Essam, and T. Gedeon, Eds., 2003, pp. 199–206.
[30] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice*, R. Riolo and B. Worzel, Eds. Norwell, MA: Kluwer, 2003, pp. 43–62.
[31] T. Quick, C. L. Nehaniv, K. Dautenhahn, and G. Roberts, "Evolving embodied genetic regulatory network-driven control systems," in *Proc. Eur. Conf. Artif. Life*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Zieglre, Eds., Berlin, Germany, 2003, vol. 2801, pp. 266–277.
[32] P. J. Bentley, "Fractal proteins," *Genetic Program. Evolv. Mach.*, vol. 5, no. 1, pp. 71–101, 2004.
[33] M. Lones and A. Tyrrell, "Enzyme genetic programming," in *Cellular Computing*, M. Amos, Ed. Oxford, U.K.: Oxford Univ. Press, 2004, pp. 18–42.
[34] G. Gusfield, *Algorithms on Strings, Trees, and Sequences*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
[35] C. Mattiussi, "Evolutionary Synthesis of Analog Networks," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2005.
[36] A. Vladimirescu, *The SPICE Book*. New York: Wiley, 1994.
[37] P. Dürr, C. Mattiussi, and D. Floreano, "Neuroevolution with analog genetic encoding," in *Proc. 9th Int. Conf. Parallel Problem Solving from Nature (PPSN IX), Reykjavik, Iceland*, Berlin, Germany, Sep. 9–13, 2006, vol. 4193, pp. 671–680.



Claudio Mattiussi is a Research Scientist with the Swiss Federal Institute of Technology, Lausanne (EPFL), Lausanne, Switzerland, in the Laboratory of Intelligent Systems, where he conducts work on evolutionary computation, neural networks, and machine learning. His research interests include bioinspired engineering, probabilistic robotics, and the numerical formulation of physical field problems.

Dario Floreano (SM'05) is an Associate Professor in the School of Engineering, Swiss Federal Institute of Technology, Lausanne (EPFL), Lausanne, Switzerland, where he is Director of the Laboratory of Intelligent Systems and of the Institute of Systems Engineering. He has published more than 100 peer-reviewed papers, authored two books, and edited three other books. His research activities span bioinspired and evolutionary robotics, biomimetic electronics, neural computation, self-organizing systems, and biology reverse engineering. He co-organized nine international conferences, and joined the program committee of more than 80 other conferences.

Dr. Floreano is cofounder and member of the Board of Directors of the International Society for Artificial Life, Inc., and a member of the Board of Governors of the International Society for Neural Networks. He is on the Editorial Board of eight international journals: *Neural Networks*, *Genetic Programming and Evolvable Machines*, *Adaptive Behavior*, *Artificial Life*, *Connection Science*, *Evolutionary Computation*, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and *Autonomous Robots*.