Gene Regulatory Network Evolution Through Augmenting Topologies

Sylvain Cussat-Blanc, Kyle Harrington, and Jordan Pollack

***Abstract*—Artificial gene regulatory networks (GRNs) are biologically inspired dynamical systems used to control various kinds of agents, from the cells in developmental models to embodied robot swarms. Most recent work uses a genetic algorithm (GA) or an evolution strategy in order to optimize the network for a specific task. However, the empirical performances of these algorithms are unsatisfactory. This paper presents an algorithm that primarily exploits a network distance metric, which allows genetic similarity to be used for speciation and variation of GRNs. This algorithm, inspired by the successful neuroevolution of augmenting topologies algorithm's use in evolving neural networks and compositional pattern-producing networks, is based on a specific initialization method, a crossover operator based on gene alignment, and speciation based upon GRN structures. We demonstrate the effectiveness of this new algorithm by comparing our approach both to a standard GA and to evolutionary programming on four different experiments from three distinct problem domains, where the proposed algorithm excels on all experiments.**

***Index Terms*—Evolution, gene regulatory networks (GRNs), genetic algorithm (GA), speciation.**

I. Introduction

ARTIFICIAL gene regulatory networks (GRNs) are a class of biologically inspired algorithms. In living systems, GRNs are used within the cell to control DNA transcription and, correspondingly, the phenotypic gene expression. Although the inner workings of the cell are governed by a large collection of complex machines, simplified models of cells as entities with protein sensors and actuators both exhibit complex behavior and offer insights into natural systems [1]. These protein sensors represent receptor molecules localized to the cellular membrane, which transduce external activity into excitatory and/or inhibitory regulatory signals. Cells use external signals collected from protein sensors localized on the membrane to activate or inhibit the transcription of the genes. A schematic of the artificial GRN model of gene regulation is shown in Fig. 1.

Computational models of GRNs were first introduced in 1960s [2] as random Boolean networks. Early studies of GRNs were focused on understanding the complexity and behavior of random networks. Since the late 1990s, there has been a resurgence of interest in GRNs as the application of evolutionary algorithms has revealed their power as a problem-solving tool. Now, GRNs are becoming ubiquitous models in artificial life and robotics. GRNs are the basis of a number of developmental models [3]–[5], as controllers of virtual and real robots [6]–[10], and neuromodulators of learning behavior [11], [12]. Other related methods of encoding reaction networks are commonly applied to similar problem domains [13]–[15].

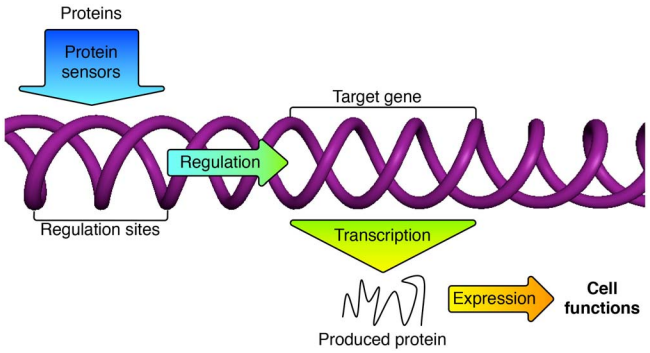

Fig. 1. Functioning of a GRN.

Most current research on the evolution of artificial GRNs [16]–[19] employ a standard genetic algorithm (GA) [20]–[22] or mutation-based evolution to discover and optimize networks. The evolutionary algorithm is often inefficient and many generations are necessary to converge to a near optimal solution. Other evolutionary algorithms have also been used to evolve GRNs such as CMA-ES [7], [9], but they experience similar issues. In this paper, we propose an algorithm for the evolution of GRNs that is loosely based upon the successful neuroevolution of augmenting topologies (NEATs) algorithm [23]. Specifically, we initialize with a bias for smaller genotypes, incorporate a speciation mechanism for the maintenance of diversity, and introduce a gene alignment-based crossover operator. In this paper, we present the GRN evolution by augmenting topology (GRNEAT) algorithm for the evolution of GRNs.

This paper is organized as follows. First, we present a discussion of the state-of-the-art in GRNs and our GRN implementation. Then, Section III introduces our adaptation of the NEAT algorithm to specifically evolve GRNs. It subsequently presents the initialization process of the algorithm, the

Manuscript received October 14, 2013; revised May 23, 2014 and November 12, 2014; accepted January 2, 2015. Date of publication January 23, 2015; date of current version November 25, 2015. This work was supported by the National Science Foundation under Grant 0757452.

S. Cussat-Blanc is with University of Toulouse, Toulouse 31042, France (e-mail: sylvain.cussat-blanc@ut-capitole.fr).

K. Harrington and J. Pollack are with DEMO Laboratory, Department of Computer Science, Brandeis University, Waltham, MA 02454 USA.

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

Digital Object Identifier 10.1109/TEVC.2015.2396199

subdivision of the population into species, and the aligning crossover. Section IV introduces four experiments to compare this approach to both a standard GA and evolutionary programming [(EP) evolution with mutation only] [24]. We demonstrate that this method excels on three distinct problem domains: 1) signal processing; 2) classification; and 3) robot navigation. We conclude with a discussion of extensions of the evolution by augmenting topology algorithm to alternative genetic representations and considerations for future work.

II. GRN Models

Research on the evolution of bio-inspired artificial regulation network models began to appear in the late 1990s. In 1999, Reil [25] was one of the first to present a biologically plausible model. He defined his GRN as a vector of numbers. The vector length is not defined in advance. Each gene starts with the particular sequence 0101, which represents a "promoter." This kind of structure is present in living systems in which the *TATA*[1] box plays the part of a promoter in real DNA [26]. Reil uses a graph visualization to observe gene activation and inhibition over time with randomly generated networks. Several classes of patterns can be generated, such as gene activation cascades, chaotic dynamics, and the settling into point or cyclic attractors. Reil also points out that after random genome mutations, the system can recover to the same pattern after a period of oscillation.

In 2003, Banzhaf [27] formulated an artificial regulation network model heavily inspired by real gene regulation. In his work, the genome is composed of multiple 32-bit integers encoded as bit strings. Each gene in the genome starts with a promoter coded by the sequence "XYZ01010101" where XYZ is any bit sequence. The combination "01010101" occurs with a 2^{-8} probability, that is to say about 0.39%. The gene coded after the 32-bit promoter has a fixed size of five integers (160 bits). The first two integers code for two regulatory kinetic components of protein production, activation, and inhibition. In this model, all aspects of DNA transcription (for example, mRNA production) are ignored in favor of focusing on the mechanism of gene regulation. This model can produce a wide distribution of protein dynamics over time.

From these seminal models, many researchers have developed their own artificial regulation networks for specific problems. In a particularly comprehensive example, Bongard and Pfeifer [28] used a model close to Reil's model to develop a modular robot. This robot has a neural network that controls each module (rotations, elongation, etc.). The genetic expression of the GRN allows the activation or the inhibition of 23 predefined phenotypic transformations such as module size growth, split a module in two, parameter modifications, neural network topology, etc.

The French flag problem is also a common benchmark for this kind of regulatory network. Introduced by Wolpert [29] at the end of the 1960s, this problem consists of developing a French flag with its three colors (blue, white, and red) starting from a single cell in the center. Lindenmayer [30] uses the problem to show the capacities of his L-systems to generate user-defined shapes. Miller [31] also solves it with Cartesian genetic programming (GP), adding at the same time the property of self-repair in his system. Bowers [32] finally used this problem to test his embryonic developmental model. The French flag problem is also a common benchmark for evo-devo models based on GRNs [3], [17], [33].

GRNs are now used to control multiple kinds of agents. They are used to control the cells of developmental models [3]–[5] or virtual agents [6]–[8], [10]. Two kinds of encoding exist in these works: GRN can be encoded with a bitstring as presented previously in Banzhaf's model [27] or with a more abstract model based on a network of proteins. However, a standard GA is often used to evolve this encoding, regardless to its structure. In this paper, we argue that a GA specially designed to optimize GRNs will both reduce the computational effort and improve the quality of the solution, by increasing evolvability. Although a number of researchers have differing definitions of evolvability, we use Kirschner and Gerhart's definition [34], where evolvability means the capacity to generate heritable phenotypic properties by reducing the potential lethality of variations, and/or reducing the number of reproductive events required to reach a novel trait. Our underlying hypothesis is that the proposed algorithm can improve evolution and evolvability of GRNs similarly to how the NEAT algorithm improves neural network evolution.

A. Our Model

Our GRN model describes an interaction network of abstract proteins. We have based our regulatory network on Cussat-Blanc *et al.*'s [5] version of Banzhaf's GRN model [27]. The Banzhaf model [27] extends GA-based matrix methods in [35] by introducing transcriptional regulation. In the GA-based matrix method, a matrix of production rates encode the kinetic contributions of each protein to each other protein. While this representation of a GRN is quite powerful, it does not account for the evolvability of the representation underlying transcriptional regulation. Transcriptional regulation provides an indirect mechanism for the enhancement and inhibition of the kinetic contributions of our abstracted proteins. In the Banzhaf model [27],the kinetics of transcriptional regulation is rate-controlled by the differential of two binary strings. The Cussat-Blanc model reduces this transcriptional interaction to a purely numerical interaction, which simplifies the evolution by eliminating complications from noncoding data and improves the model computational performances. We show that when our GRN model, an evolvable representation of the interactions of abstract proteins, is encoded into a problem domain it can be evolutionarily optimized to discover problem solutions. We first describe how our GRN interfaces with arbitrary problem domains before detailing the GRN model itself.

We consider problems where a desirable solution maps a set of inputs, I, onto a set of outputs, O. While I and O are not confined to particular domains and ranges, in this discussion, we assume both sets only contain elements in $\mathbb{Q}^{+}_{[0,1]}$. However, there must be a surjective input mapping

$$i \in I, \exists f : f(i) = \mathbb{Q}^{+}_{[0,1]} \tag{1}$$

where I is the domain of dependence and f is a function that maps values from the domain of dependence onto the rationals

[1] T = Thymine and A = Adenine.

from 0 to 1, $\mathbb{Q}^+_{[0,1]}$. There must also be a surjective output mapping

$$o \in O, \exists g : g\left(\mathbb{Q}^+_{[0,1]}\right) = o \tag{2}$$

where O is the domain of effect and g is a function that maps values from the natural numbers onto the domain of effect. The choice of the range [0, 1] is a matter of mathematical convenience, and the constraint to the positive values is a matter of physical plausibility. A task/problem is presented to a GRN by the mapping of input values onto protein concentrations [here, expressed with the function $f(i)$]. Similarly, solutions are retrieved by mapping protein concentrations onto output values. Using this approach, Nicolau *et al*. [7] applied an evolution strategy to evolve a GRN for control of a pole-balancing cart. Though Nicolau's experiment behaved consistently, the evolvability of the GRN has been a limitation.

We adapt the encoding of the regulatory network and its dynamics. In our model, a GRN is defined as a set of proteins. Each protein has the following properties.

1) The protein tag is encoded as an integer between 0 and p. The upper bound, p, of the domain can be tuned to control the precision of the GRN.
2) The enhancer tag coded as an integer between 0 and p. The enhancer tag is used to calculate the enhancing matching factor between two proteins.
3) The inhibitor tag coded as an integer between 0 and p. The inhibitor tag is used to calculate the inhibiting matching factor between two proteins.
4) The type determines if the protein is an input protein, the concentration of which is given by the environment of the GRN and which regulates other proteins but is not regulated, an output protein, the concentration of which is used as output of the network and which is regulated but does not regulate other proteins, or a regulatory protein, an internal protein that regulates and is regulated by other proteins.

The dynamics of the GRN are computed by using protein tag affinity to determine kinetic rates. These rates control the productivity of the pairwise interactions between the abstracted proteins. The affinity of a protein, a, for another protein, b, is given by the enhancing factor, u^+_{ab}, and the inhibiting factor, u^-_{ab},

$$u^+_{ab} = p - \left|t^+_a - t^*_b\right|; \quad u^-_{ab} = p - \left|t^-_a - t^*_b\right| \tag{3}$$

where t^*_x is the protein tag, t^+_x is the enhancer tag, and t^-_x is the inhibiting tag of protein x.

The GRNs kinetics are computed as a pairwise comparison of the enhancing and the inhibiting matching factors of all proteins and the source protein's concentration. For each protein i in the network, both the total enhancement and inhibition are given by

$$g_i = \frac{1}{N}\sum_{j=1}^{N} c_j e^{\beta u^+_{ij} - u^+_{\max}}; \quad h_i = \frac{1}{N}\sum_{j=1}^{N} c_j e^{\beta u^-_{ij} - u^-_{\max}} \tag{4}$$

where g_i is the total enhancing factor for a protein i, h_i is the total inhibiting factor for protein i, N is the number of protein species in the network, c_j is the concentration of protein j, $u^+_{\max}$ is the overall maximum enhancing factor, $u^-_{\max}$ is the overall maximum inhibiting factor, and β is a control parameter which we will shortly describe in greater detail. The change in concentration of protein i is given by

$$\frac{dc_i}{dt} = \frac{\delta(g_i - h_i)}{\Phi} \tag{5}$$

where Φ is a normalization factor to ensure the sum total of all output and regulatory protein concentrations is unity. β and δ are two constants that influence the reaction rates of the regulatory network. β affects the affinity distances between the proteins: the higher β, the more the proteins can affect each other. δ affects the level of production of the protein in the differential equation. The lower δ, the more slowly a gene responds to regulation. Higher values cause the proteins to respond more rapidly to regulation.

III. Evolving GRNs by Augmenting Topology

The focus of this paper is an algorithm that facilitates the evolution of GRNs. Given the many parallels between the current state of GRN research and the first neural networks renaissance, we draw inspiration from one of the most successful applications of evolutionary computation to neural networks, the NEAT algorithm [23]. A key contribution of the NEAT algorithm is its solution to the problem of genetic crossover for neural networks. The integrative nature of neural networks can cause small changes in weights and structure to have a significant influence on overall network behavior. For this reason, the splicing of two arbitrarily chosen neural networks will generally not produce a network with behavior similar to either of the two choices. Given this phenomena, early algorithms for neuroevolution [36] focused on evolving networks via mutation instead of recombination. In this paper, we focus on three key elements of the NEAT algorithm.

1) The initialization of the algorithm—as opposed to initializing with individuals randomly sampled from the complete distribution, only small networks are used in the initial population so as to allow for a more progressive complexification.
2) The speciation protects newly appeared solutions by giving them some time to optimize their structures before competing with the whole population.
3) The alignment crossover with the use of a historical marker to protect ancestral genes during a crossover operation between two genomes.

In this paper, we apply these elements to the evolution of GRNs. Of course, they must be adapted to the evolution of the GRN structure. The adaptations of each of these elements are described in the next subsections.

A. Initialization of the Evolution

Instead of initializing the population with GRNs of random sizes and protein structures, the population is inoculated with small GRNs. According to our GRN model, the population is initialized as follows. The genome first contains one input protein per sensor in the problem and one output protein per actuator in the problem. When created, the input proteins are linked to a given sensor and the output proteins are linked to a given actuator. The parameters (protein tag, enhancing

tag, and inhibiting tag) of these proteins are randomly generated. One random regulatory protein is added to the network in order to create the minimal structure required by the GRN. In order to create viable species, each genome is duplicated two times the minimum species size. The duplication is made by mutating one protein of the first genome. This insures both similarity between GRNs in the initial species, and minimal initial genome sizes. Moreover, maintaining a minimal species size is crucial for reproduction via crossover: always crossing the same genomes might be counter-productive in terms of diversity because the offspring generated by such crossovers will have very similar genetic material. Therefore, we think it is important to have species with enough individuals to produce diverse offspring. Having too small species could lead to premature extinction.

B. Speciation

While speciation is subject to ongoing research in evolutionary biology [37], [38], here, we use the term to mean clusters of similar genotypes in a GA. Even this topic has been explored to a great degree under a number of names, in particular niching [39], [40]. We draw upon Stanley's observation that speciation can be an effective mechanism for maintaining diversity within populations of graph-based individuals [23]. In the NEAT algorithm, speciation is made possible by the use of historical marker annotations for each gene. The historical marker is used to track the divergence of genes, and also to calculate the distance between genomes. Genetic distance is used to cluster individuals into species. While historical markers were an essential component in allowing the meaningful speciation of evolving neural networks, the tags which determine the affinity between genes offer a natural measure of genetic distance. Moreover, this measure has the advantage of directly working on the network structure instead of the network nodes' lineage (edges are not directly subject to evolution in a GRN since they are not encoded in the genome), which might improve alignment during crossing over.

1) Genetic Distance: The distance between two GRNs is calculated as an accumulation of distances between proteins in both networks. The distance between two proteins A and B is given by

$$D_{\text{prot}}(A,B) = \frac{a\left|t_A^* - id_B\right| + b\left|t_A^+ - t_B^+\right| + c\left|t_A^- - t_B^-\right|}{p} \tag{6}$$

where t_x^* is the tag, t_x^+ is the enhancer tag, t_x^- is the inhibiter tag of protein x, and p is the precision of the GRN (see Section II-A). a, b, and c are constants that weight each part of the protein properties. The sum of these constants must be equal to 1. In this paper, they are set up to $a = 0.75$ and $b = c = 0.125$. These values have been chosen by the means of a parameter survey.

To calculate the global distance between two genomes, the distance between the matching input and output proteins (the inputs linked to the same sensor and the output linked to the same actuator) are summed. Then, for each regulatory protein R_i^1 of the first genome, the global genome's distance is augmented with the minimum distance of R_i^1 to all the regulatory protein of second genome (with replacement). The dynamics coefficients β and δ are also taken into account when computing the genetic distance. The distance between both coefficients is divided by the maximum possible distance and contributes additively. Finally, the genome distance is normalized by dividing it by the larger number of proteins of both genomes augmented by the number of coefficients taken into account in the distance calculation. The distance between two genomes can be formalized as

$$\begin{aligned} D(G_1,G_2) &= \frac{D_{\text{in}} + D_{\text{out}} + D_{\text{reg}} + D_\beta + D_\delta}{\max(N_1,N_2)+2} \\ D_{\text{in}} &= \sum_{x=0}^{\#\text{in}} D_{\text{prot}}(I_1(x), I_2(x)) \\ D_{\text{out}} &= \sum_{x=0}^{\#\text{out}} D_{\text{prot}}(O_1(x), O_2(x)) \\ D_{\text{reg}} &= \sum_{R_i \in G_1} \min_{R_j \in G_2} D_{\text{prot}}\left(R_i, R_j\right) \\ D_\beta &= \frac{|\beta_1 - \beta_2|}{\beta_{\max} - \beta_{\min}} \\ D_\delta &= \frac{|\delta_1 - \delta_2|}{\delta_{\max} - \delta_{\min}} \end{aligned} \tag{7}$$

where

G_i	genome;
N_i	number of proteins contained in the genome G_i;
$I_i(x)$ [respectively, $O_i(x)$]	G_i's input (respectively, output) protein linked to the sensor (respectively, actuator) x;
#in (respectively, #out)	number of input (respectively, output) proteins;
R_*	regulatory protein in a genome;
β_i and δ_i	dynamics coefficient of G_i;
$\beta_{\min}$, $\beta_{\max}$, $\delta_{\min}$, and $\delta_{\max}$	bounds of the variation intervals of the dynamics coefficient β and δ;
$\max(N_1,N_2)+2$	number of genes the distance is computed: the number of proteins of the biggest network plus two dynamics coefficients (β and δ).

2) Organization of Species: With this distance function, the population can be organized into species. A genome G is classified as a member of species s in which the distance $D(G, G_s^{\text{rep}})$ of the genome to a representative genome is the lowest and only if this distance is lower than a speciation threshold σ_s given to each species s. In order to reduce the number of parameters, the user must set up before using the algorithm, this threshold is automatically tuned according to the average size of all species: if the species size is less than the average species size, the speciation threshold of the species is incremented by 0.01; if the species size is greater than the average, the speciation threshold is decremented by 0.01. Every speciation threshold is bounded by the interval $[0.03, 0.5]$. The representative genome is a genome randomly selected in the previous generation. This method is less expensive than comparing the distance of the genome with all the species genomes and has been shown to give positive

results [23]. When a new species is created, the first individual that creates the species serves as the representative for the current generation.

This method insures a subdivision of the entire population into species that have comparable structure. Then, the genomes are evaluated and compete within each species. The number of offspring that each species generates is adjusted according to the performance of the species. Every genome is evaluated with the general fitness function of the problem. Fitness sharing [20], [41] is then applied by dividing the fitness of each genome by the number of members of its species: the bigger the species, the stronger the fitness penalty should be because the species has more search power.

The number of offspring $o(s)$ of a species s is then given by

$$o(s) = \frac{\sum_{G_i \in s} f'(G_i)}{\sum_{t \in S} \sum_{G_j \in t} f'(G_j)} \tag{8}$$

where S is the set of the existing species, G_x is a genome, and $f'(G_x)$ is the adjusted fitness of the genome G_x.

Reproduction is then standard within each species. The offspring are created by first copying the best genome of each species (elistism). For each species, the global crossover and mutation rates give the number of offspring to generate with each variation method based on the number of offspring allowed by species. In the case where the sum of both rates are lower than one, the remaining offspring are selected and added to the species without variation. The selection operator used in this paper is a standard three-player tournament.

Once the offspring are generated, they are organized into species with the procedure detailed previously. When a species is too small (under a given minimum size), this species and its genomes are deleted and as many genomes are randomly created by mutation in the other species to prevent the population size from changing abruptly. When expanding the number of members of a species, mutation, and rejection sampling are used in order to maximize the probability of generating a compatible genome. The aim is to produce new genome structurally close to the one existing in this species in order to keep a certain consistency within the species. The parent genome is selected by a tournament and a candidate is created by mutating this parent. However, if the candidate does not fit into the species, then the candidate is rejected and a new genome is generated by mutation until it fits the speciation requirements.

C. Reproduction Operators

Genetic variation of GRNs is accomplished with three standard mutation operators and a new crossover operator based on genetic alignment of protein tags. This section presents the reproduction operators.

1) Mutation Operators: When a mutation has to be applied to a genome, one of the three following mutation operators can be used.

a) *Mutate a Protein:* Randomly modify one of the tags of a protein. Both the tag and the protein are selected randomly from the genome.

b) *Add a Protein:* Generate a new random regulatory protein (its protein tag, its enhancer tag and its inhibiter tag are generated randomly) and add it to the genome.

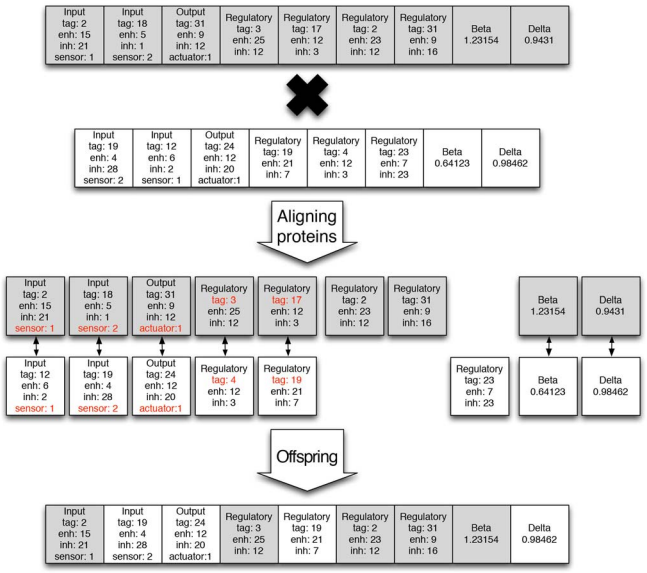

Fig. 2. Aligning crossover tries to align the protein before crossing the genomes. The input and output proteins are aligned according to the sensors and actuators they are linked to. The regulatory proteins are aligned by the mean of the distance between the proteins.

c) *Delete a Protein:* Remove one randomly selected regulatory protein from the genome. This operator can be applied only if there is more than one regulatory protein in the genome. If not, one of the two other operators is applied.

When a mutation must be applied in the evolutionary process, one of these three operators is selected. Independent of the global mutation rate of the algorithm, each mutation operator has a specific selection probability that can be parameterized. After a parameter survey, we found that a probability of 0.5 for adding a protein and 0.25 for the two other operators is the best compromise for slowly increasing the GRN size and maintaining a sufficient level of exploration. Furthermore, the genome size is unbounded since the aim of the algorithm is to explore the GRNs search space starting from minimal networks and slowly increasing the network size iteration after iteration.

2) Crossover Operator: In the standard evolution of a GRN, one-point crossover is commonly used to cross two genomes. One-point crossover of two parent GRNs is very similar to the crossover of binary string genomes. First, a point, P, is randomly chosen on the genome of the smaller parent. Then, genetic material from one parent up to P and genetic material from the other parent from P onward are joined to create a child genome. In this paper, we have designed a new crossover operator inspired by the gene aligning crossover operator utilized in the NEAT algorithm. A schematic of the crossover operator is shown in Fig. 2.

Each type of protein is processed separately. Both the input and the output proteins are treated with the same method. One of each input (or output) protein linked to a sensor (or an actuator) is randomly selected from one of the parents. The regulatory proteins are then aligned before being crossed: first, the regulatory proteins of each parent are randomly shuffled (in order to avoid historical biases). Then, for each regulatory protein p_i^1 from the first parent, the closest regulatory

TABLE I
PARAMETERS OF THE EVOLUTIONARY ALGORITHMS

Parameter name	Parameter values					
Algorithm	GRNEAT	GA	EP	GA with aligning crossover	GA with speciation	EP with speciation
Population size	500	500	500	500	500	500
Selection	3-player tournament					
Initial σ_s	0.15	–	–	–	0.15	0.15
Min species size	15	–	–	–	15	15
# Reg proteins	[1, +inf]	[3, 50]	[3, 50]	[1, +inf]	[3, 50]	[3, 50]
Crossover rate	0.25	0.6	0.0	0.25	0.25	0.0
Mutation rate	0.75	0.3	1.0	0.75	0.75	1.0
Crossovers	Aligning (σ_a=0.15)	1-point	–	Aligning (σ_a=0.15)	1-point	–
Mutations (probability)	Add (p=0.5), Del (p=0.25), Change (p=0.25)					

protein p_j^2 not yet aligned is selected from the second parent. If the distance $d_{\text{prot}}(p_i^1, p_j^2)$ is lower than a given alignment threshold σ_a, both proteins are aligned. An aligned protein cannot be aligned anymore. Once alignment of all proteins has been attempted, one protein of each aligned pair is randomly selected and added to the offspring. The remaining unaligned proteins from a single parent are then added to the offspring. The source parent is selected with a probability based on the quantity of proteins the offspring inherited from each parent

$$P(P_1) = \frac{N_{P_1}}{N_{P_1} + N_{P_2}}; \quad P(P_2) = \frac{N_{P_2}}{N_{P_1} + N_{P_2}} \tag{9}$$

where $P(P_i)$ and $P(P_2)$ are the respective probabilities of selecting proteins from the first and second parent and N_{P_i} is the number of proteins inherited by the offspring from a given parent. This ensures that no crucial genetic material is deleted during the crossover and that the added material best fit to the aligned material. Finally, the dynamics coefficient are also crossed. One of the β and the δ coefficients are randomly selected from the parent genomes and used in the offspring genome.

IV. EXPERIMENTS

We have implemented this algorithm in Java. It is available on github at the following address: https://github.com/scussatb/GRNEAT. With this implementation, we have evaluated the GRNEAT evolutionary algorithm on four different problems. The first two are signal processing experiments inspired by [19]. They consists of doubling an input frequency and generating a low-pass frequency filter. The third is the intertwined spirals in which the GRN has to classify 194 2-D points into two intertwined spirals. The last experiment is more complex, the coverage control problem (CCP), which involves an agent controlled by a GRN that has to cover a 2-D environment with obstacles, much like a robotic vacuum cleaner.

In these experiments, GRNEAT is compared to a standard GA and EP, where only mutation is used. These methods are the most common techniques used in the GRN community to evolve the networks. In order to evaluate the impact of each component of GRNEAT, three more comparison are made.

1) GA with aligning crossover instead of the one-point crossover.
2) GA with one-point crossover plus GRNEAT speciation's method.
3) EP with GRNEAT speciation's method.

Table I summarizes the parameters that used these three algorithms. These parameters have been found to be the best for each algorithm via a parameter sampling survey.[2] The results presented in the experiment are averaged over multiple runs (varying depending on the experiment). During each run, an identical random seed is used for all three algorithms.

In all the following experiments, each GRN is initially stabilized for 25 steps with zero concentration on all its input proteins. This mechanism is used to avoid the chaotic stabilization period of the GRN: since all the protein concentrations are initialized with the same value (1 over the number of proteins), the GRN needs some step to stabilize the concentrations. Once stabilized, the input concentrations are updated according to the problem. These inputs are detailed in the following problem descriptions.

A. Doubling Input Frequency

1) Problem Description: In this experiment, the target solution is a GRN which doubles the frequency of a sine curve provided as input. The input is described by

$$\text{in}(t) = \frac{1}{2}\sin\left(\frac{2\pi t}{p} - \frac{\pi}{2}\right) + \frac{1}{2} \tag{10}$$

where $\text{in}(t)$ is the input provided to the GRN at timestep t and p is the (constant) period of the sinusoidal curve.

After the warm up of the GRN, the input concentration is updated with the input sinusoidal value and the GRN is run for one step. The output protein concentration is recorded and used to evaluate the GRN. This last step is repeated as many times as required by the experiment's duration (number of timesteps). The GRN is trained for 1000 timesteps on two different input periods: $p = 250$ and $p = 1000$. In these cases, the target output signals that the GRN must generate is the sine functions of periods $p = 125$ and $p = 500$. A constant zero-function is added to the training set. Joachimczak and Wróbel [19] noted that this zero function helps by ensuring that an output signal is only generated when an input signal is present. The fitnesses of each training signal are summed in order to obtain the GRN fitness.

The error of a signal is the difference between the output signal generated by the GRN o_t and the desired signal d_t at

[2]EP population size might appear high for a mutation-based evolution but parameter sampling shows that smaller population sizes lead to early convergence to local optimum, probably due to lack of diversity.

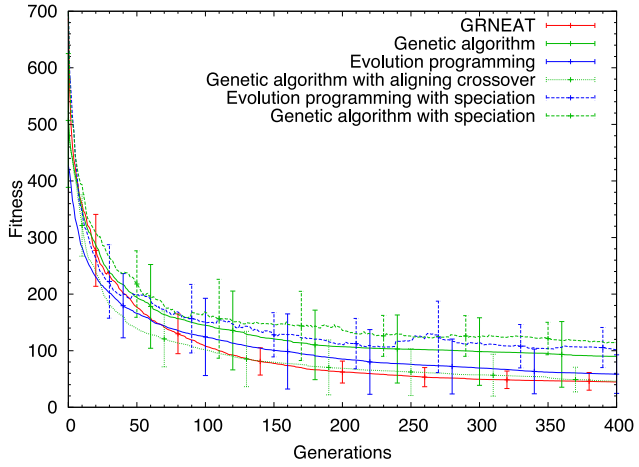

Fig. 3. Comparison of the GRN evolutions on the doubling frequency problem. The averaged best fitness (plus standard deviation) is represented. Here, the lower the fitness, the better is the minimization of the error.

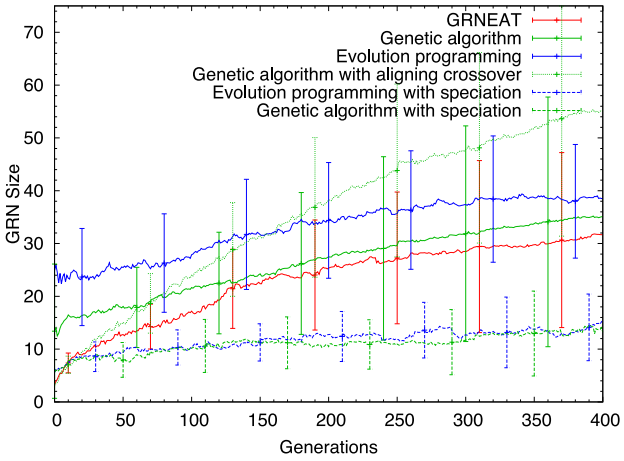

Fig. 4. Comparison of the median GRN sizes trained with GRNEAT (in red), GA (in green), and EP (in blue) and GRNEAT components (dashed and dotted lines) on the doubling frequency problem.

each timestep t. This error is weighted by the desired signal: when a signal must be generated, it is crucial that the GRN effectively produce an output. Finally, the summed error is weighted by the number of observed events E_o during the GRNs evaluation and the number of awaited events E_d. An event is defined by the fact that the output signal is crossing the 0.5 value (ascending or descending). This helps the evolution to promote oscillating GRNs and therefore reduces the computational effort required to converge to good solutions. The fitness can be formalized as

$$\frac{1}{1+|E_o-E_d|/E_d}\sum_{t=0}^{\#\text{steps}}|o_t-d_t|(1+d_t). \tag{11}$$

2) Evolution: Fig. 3 compares the best fitness of both evolutionary algorithms averaged over 26 runs. GRNEAT converges faster toward better solutions than the standard GA, but GRNEAT is a slow starter. A paired student t-test shows GRNEAT is significantly better than GA with a confidence interval of 95% (t-value $= -9.64$ and p-value $< 10^{-5}$). However, recall that, while the GA is initialized with 500 totally random genomes, each random genome of GRNEAT is duplicated multiple times in order to generate species of minimum size. Furthermore, the genomes generated by GRNEAT are initially smaller. This suggests that the initial exploration of the search space by GRNEAT may not be optimal. Yet, this initialization effect is corrected after 70 generations, beyond which the average best fitness of GRNEAT is always better than the average best fitness of the GA.

In comparison to EP, GRNEAT has similar performance but with a lower standard deviation. This suggests that GRNEAT is more likely to converge to a better solution than EP, which is more likely to get stuck in a local optimum. A student's t-test shows that GRNEAT is almost significantly better than EP with a t-value of -1.93 and a p-value of 0.065.

Fig. 4 compares the median size of the best GRNs generated by the three approaches. GRNEAT is initialized with small networks and redundancy within species, which explains its slow start on the convergence curves. However, the algorithm slowly increases the size while optimizing the network

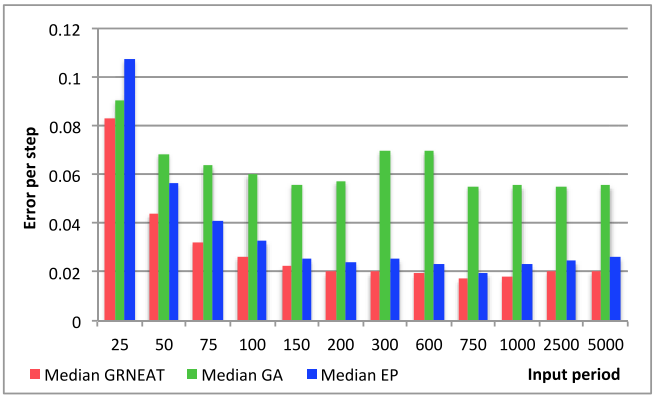

Fig. 5. Comparison of the median GRNs trained with GRNEAT (in red), with a GA (in green), and with EP (in blue) on a multiple frequencies of the doubling frequency problem.

performance. As a matter of fact, the final solutions generated by GRNEAT contain fewer proteins than the ones generated by EP. This shows the capacity of the algorithm to complexify the network structure up to the best size. However, the large standard deviation of network sizes on this simple problem prevent a statistically significant conclusion regarding the difference in network sizes on this problem.

When comparing the influence of each component of GRNEAT (see dashed and dotted lines on Figs. 3 and 4), it can be noticed that GRNEAT and GA with aligning crossover produce very comparable results in term of fitness. However, when comparing the sizes of the GRNs generated by both approaches, we can notice that all the GRNs generated by method with speciation are smaller than the one generated by algorithm without speciation. This component clearly helps the algorithm to cluster comparable genomes and therefore cross networks with comparable structures and keeps the size from increasing excessively.

3) Generalization: To verify that GRNEAT generates GRNs that have comparable generalization properties to the ones generated by a standard GA and EP, all the best genomes generated with both approaches have been run on eleven other input periods. The median fitness error per simulation step is presented on Fig. 5: the lower the error,

TABLE II
INPUT AND TARGET SIGNALS USED FOR TRAINING LOW-PASS FILTER GRNS

#	Input signal	Target Signal
1	a 250-timestep-period sinusoid	a 250-timestep-period sinusoid
2	a 500-timestep-period sinusoid	a 500-timestep-period sinusoid
3	a 150-timestep-period sinusoid	a constant zero-signal
4	a 62.5-timestep-period sinusoid	a constant zero-signal
5	the combination of a 125-timestep-period and a 62.5-timestep-period sinusoids	a 125-timestep-period sinusoid
6	the combination of a 250-timestep-period and a 62.5-timestep-period sinusoids	a 250-timestep-period sinusoid

the better the GRN. Each bar of the histogram presents the median error value per simulation step of the 26 best GRNs generated by GRNEAT (in red), GA (in green), and EP (in blue). Twelve different period values of the sinusoid input function have been tested. The GRNs evolved with GRNEAT give better result on generalization than GA and EP. Actually, the difference between these three algorithms on nontraining cases is comparable to the difference on the training set. This suggests that GRNEAT may have equivalent generalization properties to the GA and the EP. The obtained outputs of the best GRNs generated by GRNEAT, GA, and EP are presented in Supplementary material 1.

B. Low-Pass Frequency Filter

1) Problem Description: In the low-pass frequency filter experiment, the target GRN behavior is to attenuate all signals above a certain frequency and ignore lower frequencies. If the period of the input signal is below a given threshold (here, equal to a 200 timestep), the output of the GRN should be zero. For signals above this threshold, the GRN has to reproduce the input signal exactly. When two signals are combined with two different frequencies, if one of them is under the threshold, the GRN must allow only low frequency signal to pass. The method for signal processing with a GRN is the same as the method presented in the previous frequency doubling experiment.

The GRN is trained on six different input signals that consist of two signals which are completely above the threshold, two below the threshold, and two which contain both a signal above and below the low-pass threshold. The zero-signal function presented in the previous experiment is also added to the training set. The training signals are detailed in Table II. The fitness used to evaluate the GRN is exactly the same as the one presented in (11).

2) Evolution: Fig. 6 presents the results of this experiment. Both evolutionary algorithms have been run 30 times. The figure presents the averaged best fitness with each algorithm. The same convergence observed in the previous problem is obtained with this problem. GRNEAT starts slower because of its initialization method but quickly finds better solutions. Here, 70 iterations are necessary for GRNEAT to find better solutions. A student's t-test shows GRNEAT is significantly better than GA with a t-value of -5.69 and a p-value less than 10^{-5}.

When comparing GRNEAT to EP, the same observations that were observed for the GA can be applied. GRNEAT converges to better solutions with a lower standard deviation, as in the previous experiment, although not to a statistically significant degree. This again shows the capacity of this algorithm to avoid local optima and to converge to better solutions in general.

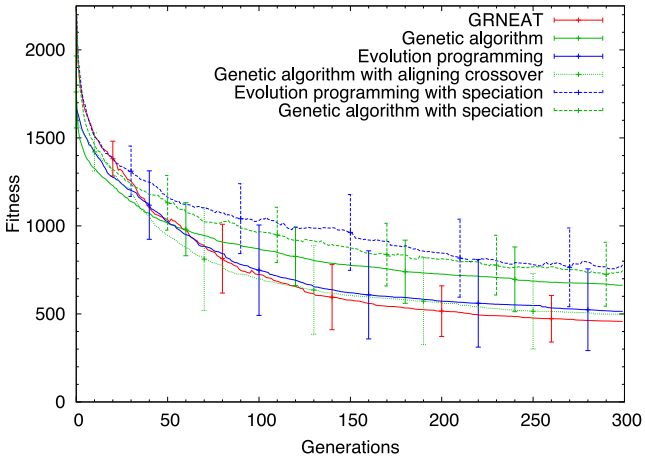

Fig. 6. Comparison of the GRN evolution on the low-pass frequency filter problem. The averaged best fitness (plus standard deviation) is represented. Here, the lower the fitness, the better is the minimization of the error.

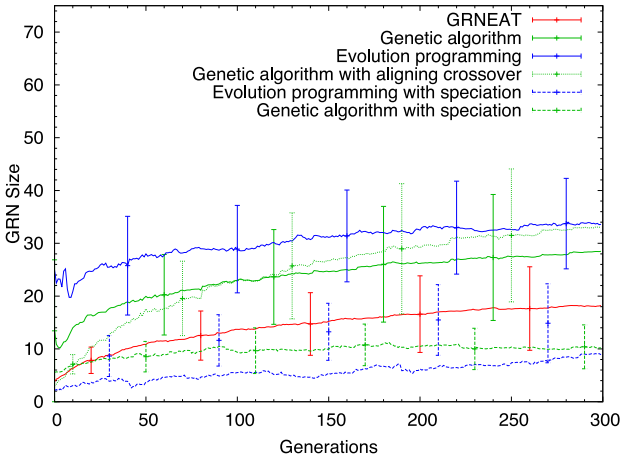

Fig. 7. Comparison of the size of the GRNs generated by GRNEAT, GA, and EP on the low-pass filter problem.

Fig. 7 compares the sizes of the GRNs generated by GRNEAT, GA, and EP. GRNEAT converges to smaller networks than EP with statistical significance (t-value $= -3.69$ and p-value $= 0.001$), but not significantly smaller networks than GA (t-value $= -0.61$ and p-value $= 0.54$). However, recall that GRNEAT has similar performance to EP, and now we see that GRNEAT produces significantly smaller networks. Furthermore, while the difference in network size under GA and GRNEAT is not statistically significant, GRNEAT yields better performing networks than GA.

When comparing GRNEAT components, the same observations as in the previous experiment can be pointed out: the aligning crossover used with a GA leads to the evolution of some of the best fit networks, yet the resulting networks are markedly larger. On the other hand, when considering only EP with speciation, smaller network sizes are maintained, but

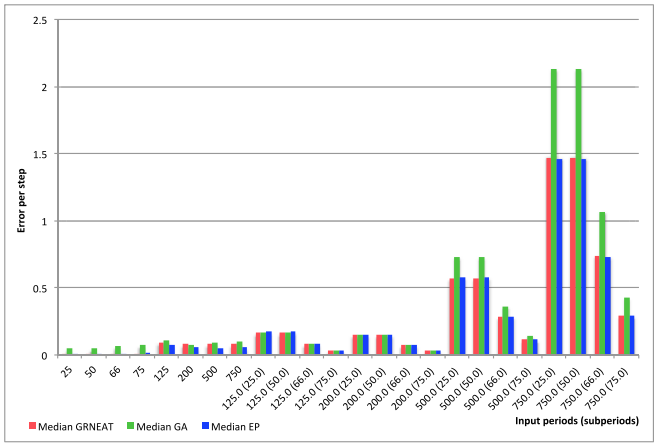

Fig. 8. Comparison of the generalization capacities of GRNs trained with GRNEAT (in red), GA (in green), and EP (in blue) on the low-pass filter problem. Bars represent the median error.

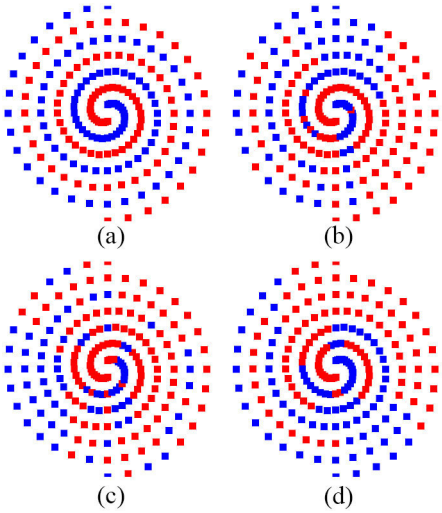

Fig. 9. Result of the best GRN obtained with GRNEAT, GA, and EP on the intertwined spirals.

networks do not achieve the same fitness that is reached when aligning crossover is utilized.

3) Generalization: The GRNs generalization capacities have also been tested on this problem. Fig. 8 presents the median generalization score of the 30 genomes obtained with GRNEAT, GA, and EP after 300 generations. It presents the error per simulation step on four frequencies under the period threshold, four over the threshold, and a combination of the periods the GRNs have to filter. GRNs generated by GRNEAT still generalize better than those produced by the GA. GRNEAT-generated GRNs perform comparably to the ones generated by EP. Some of the curves of the low-pass frequency filter training and generalization are presented in Supplementary material 2.

C. Intertwined Spirals

1) Problem Description: The intertwined spiral problem is a classification problem, where a set of 2-D points organized as two intertwining spirals must be correctly classified. Each of the two spirals represents a class. Spirals start from the center

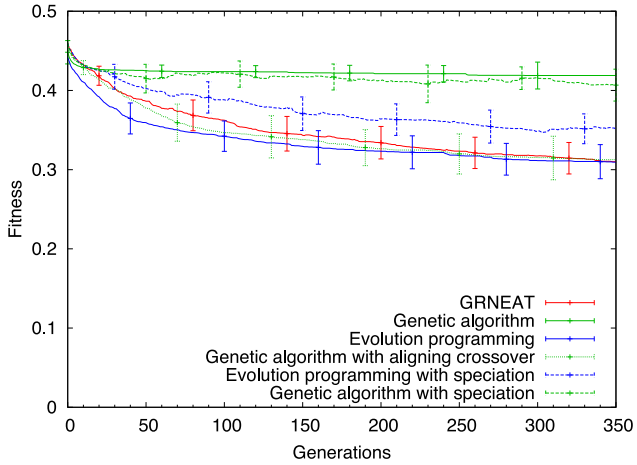

Fig. 10. Comparison of the GRN evolution on the intertwined spirals. The averaged best fitness (plus standard deviation) is represented. The GRNEAT algorithm evolves faster the GRN than a standard GA. GRNEAT produces comparable fitnesses to EP. Here, the lower the fitness, the better is the quantity of errors made by GRN.

of the space and are offset by a 180° rotation. Fig. 9 represents the training points and their classification provided to the GRN. This problem is a common benchmark in the genetic programming [42], [43] and the neural network [44] communities. To the best of our knowledge, this problem has never been solved with a GRN. However, Cussat-Blanc and Pollack [45] introduced the use of GRNs for the generation of images. The implementation of the fitness function is similar to this approach to GRN-based image synthesis.

In this problem, the inputs of the GRN are the (x, y) coordinates of each point that must be classified. Two outputs are used to classify the point into one of the two categories by the following method: if the first output protein concentration is greater than the second output protein concentration, the point is classified in the first category and the opposite concentration difference classifies a point into the second category. The GRN is first stabilized during 25 steps before the inputs are set up. Then, input is provided to the GRN and it is evaluated for 25 more steps before the classification is computed. The GRN is rerun for the 194 points to be classified. This 50 step evaluation provides the GRN with the computation and stabilization time necessary for classification.

2) Evolution: The evaluation function for this problem consists of maximizing the number of well-classified points from 194 samples that describe two spirals. Fig. 10 shows the evolution of the GRN with GRNEAT, GA, and EP on 350 iterations. This figure presents the average best fitness and its standard deviation of 20 runs. Fig. 9 presents the best classification obtained with GRNEAT along all the 20 runs. This classification achieved a 72.68% accuracy. While visual inspection of the generalization pattern in Fig. 9 shows some chaotic patterning, this also supports the variability of patterns producible by a GRN. Concerning the comparison, the results are very comparable between GRNEAT and EP. At the opposite, GRNEAT does significantly better than GA (t-value $= -25.72$ and p-value $< 10^{-5}$). GRNEATs initialization problem is still present on this experiment but is able to reach EPs performance after 250 generations.

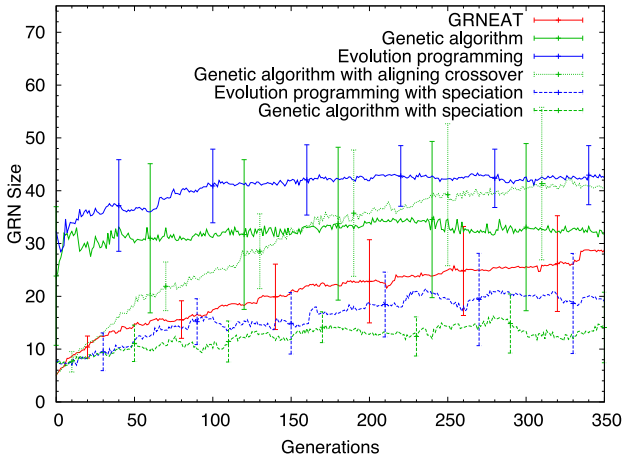

Fig. 11. Comparison between the sizes of the GRNs generated by GRNEAT, GA, and EP on the intertwined spirals problem.

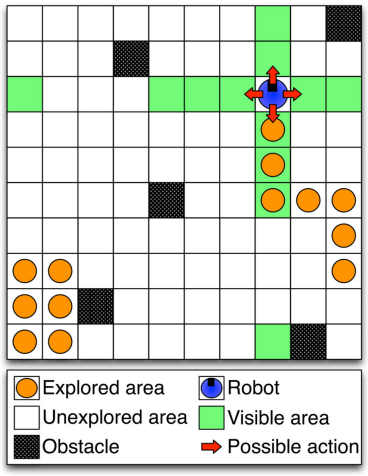

Fig. 12. Environment for the CCP.

The main advantage of GRNEAT over EP is again the capacity to generate smaller-sized GRNs. As presented in Fig. 11, the GRNs generated by GRNEAT are significantly smaller than those generated by EP (t-value $=$ 5.50 and p-value $< 10^{-4}$). On average, the best GRNs generated by GRNEAT after 350 generations have 28.8 proteins ($\sigma_{\text{dev}} =$ 10.67) whereas GRNs generated by EP have 42.45 proteins ($\sigma_{\text{dev}} =$ 5.03), that is to say EP led to networks 47% larger than GRNEAT networks.

Concerning GRNEAT components, once again, GA with aligning crossover produces solutions that compete in term of fitness with GRNEAT but with larger networks. Speciation helps again to keep the genome size smaller.

D. Coverage Control

1) Problem Description: Robot coverage problems have a long standing history in evolutionary computation. One of the first GP studies was on what has come to be called the "lawnmower" problem [46]. The lawnmower problem is an instance of the CCP in a uniform environment, often with turn-based navigation and the ability to jump. The use of modular structures have been shown to facilitate solving the lawnmower problem in GP [47]. The CCP consists of finding a path that visits every node at least once. In this paper, our representation of problems is related to a graph traversal problem, similar to the classic traveling salesman problem, where robotic agents are not required to turn and may only move to adjacent nodes.

In this last experiment, the CCP consists of a robot having to cover a discrete environment. The environment is toroidal and can contain obstacles. The robot perceives the surroundings cells in its Von Neumann neighborhood with a range of 3. For each cell, the robot knows if the cell has already been explored. At each timestep, the robot has four possible actions of movement along the Von Neumann neighborhood (North, South, East, or West). When the robot explores a previously unexplored cell, it receives a reward of 1 and otherwise receives no reward. An illustration of the environment is shown in Fig. 12. Here, the GRN is used to control the robot. The GRN has nine inputs proteins that correspond to the following.

1) The summed reward three steps away in the North direction.

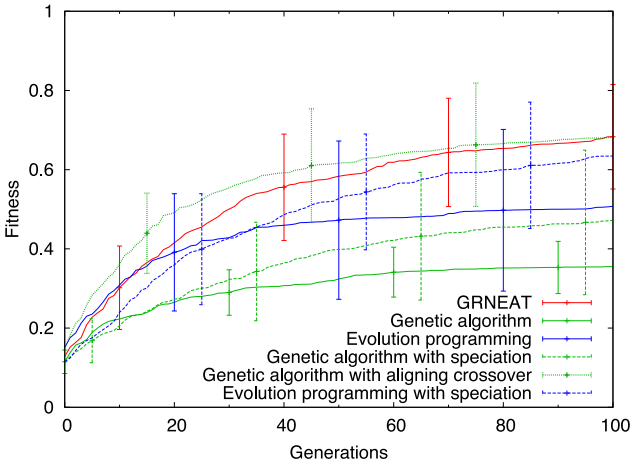

Fig. 13. Comparison of the GRN evolution on the CCP. The averaged best fitness (plus standard deviation) is represented. The GRNEAT algorithm evolves faster the GRN than both a standard GA and evolution programming. Here, the higher the fitness, the better (quantity of cells explored by the robot). This figure also presents the gain of the aligning crossover operator (dotted green line) to the GA and the speciation associated to GA (dashed green line) and to evolution programming (dashed blue line).

2) The summed reward three steps away in the South direction.
3) The summed reward three steps away in the East direction.
4) The summed reward three steps away in the West direction.
5) The number of obstacles in a three-cell range in the North direction.
6) The number of obstacles in a three-cell range in the South direction.
7) The number of obstacles in a three-cell range in the East direction.
8) The number of obstacles in a three-cell range in the West direction.
9) The current reward obtained by the robot.

In addition to these nine inputs, four output proteins corresponding to the four possible movements are used. The GRN is used to compute a movement action at every timestep.

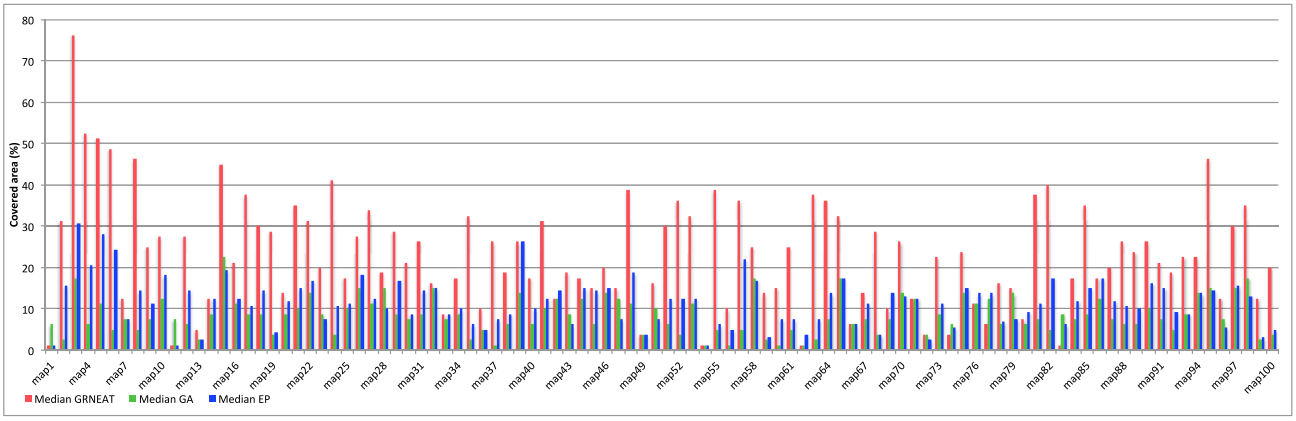

Fig. 14. Coverage control generalization results on 100 maps: median fitness of GRNs evolved with GRNEAT (in red), with GA (in green), and with EP (in blue).

The GRN is evaluated 25 steps (a necessity for the stabilization of the GRN in a discrete environment), then the action that corresponds to the output protein with the largest concentration is selected. If this action is valid (i.e., if there are no obstacles in the way), then the robot moves in the desired direction. The fitness function is calculated after 1000 simulation steps and the fitness value is the ratio of covered cells to uncovered cells (excluding obstacles).

2) Evolution: The GRN is trained on three maps with 20 obstacles randomly placed obstacles. Obstacles are placed such that every remaining square is still accessible by the robot to avoid situations where a location is completely surrounded by obstacles. The map size is 10×10. For each evolutionary run for all evolutionary algorithms tested, the GRNs are trained with the same three environments. This ensures that the results are comparable between all methods.

Fig. 13 presents the result of evolution averaged on 24 runs. GRNEAT evolves the GRN faster to better solutions than both GA and EP. The final best averaged fitness is 0.68 with GRNEAT, 0.35 with GA, and 0.51 with EP. The environment is covered significantly better by GRNs optimized with GRNEAT than with networks optimized with both GA (t-value $= -10.19$ and p-value $< 10^{-5}$) and EP (t-value $= -3.69$ and p-value $= 0.001$). Fig. 14 presents the median score of the GRNs evolved with GRNEAT (in red), GA (in green), and EP (in blue) on 100 random maps. Here again, GRNs evolved with GRNEAT present better results on the generalization tasks than those evolved by both GA and EP. The gain is comparable to the gain observed on the training set. Thus, GRNs evolved by GRNEAT seem to maintain the same performance during generalization as opposed to GRNs evolved with either GA or EP. Some of the results of both the training set and the generalization set are presented in Supplementary material 3.

Fig. 15 presents the average sizes of the GRNs generated by GRNEAT, GA, and EP. As in the previous two experiments, GRNEAT produces GRNs of significantly smaller sizes: when comparing GRNEAT and GA, the paired student t-test gives a t-value of 2.47 with a p-value equal to 0.02 and when comparing GRNEAT to EP, the t-value is 2.16 with p-value $= 0.04$. Therefore, on this problem, GRNEAT converges to both better

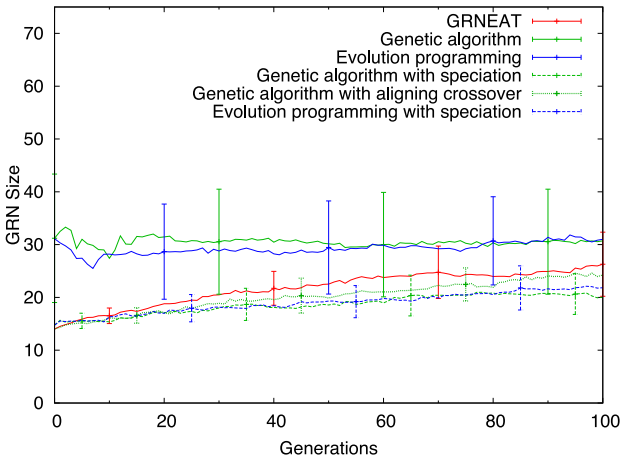

Fig. 15. Comparison of the sizes of the GRNs generated by GRNEAT, GA, and EP. The size of the GRNs generated by a GA with aligning crossover and GA, and evolution programming speciation are also plotted on this graph.

solutions in terms of performance, and to smaller, simpler networks.

When comparing each of GRNEATs components, aligning crossover again produces GRNs very comparable to GRNEAT in term of fitness. This time, the size of the networks are also comparable between both approaches. Speciation still produces GRNs of smaller sizes but the difference is less evident. In our opinion, this is due to the smaller size of GRNs involved in the evolution for this particular problem. It makes the increase of network size less critical than in other problems.

3) Phylogenetics: In order to understand the speciation dynamics of the GRNEAT algorithm, we investigate the structure of a representative phylogenetic tree. In Fig. 16, we present a dendrogram from one of the best trials of GRNEAT on the CCP. The tree was constructed for generation samples taken every 15 generations. The construction process involves two steps: 1) computing a distance matrix between all species in adjacent generations and 2) linking species to their most similar ancestral species. The distance matrix contains a pairwise genetic distances between the respective species of the 11 sampled generations. Pairwise distance was computed as the

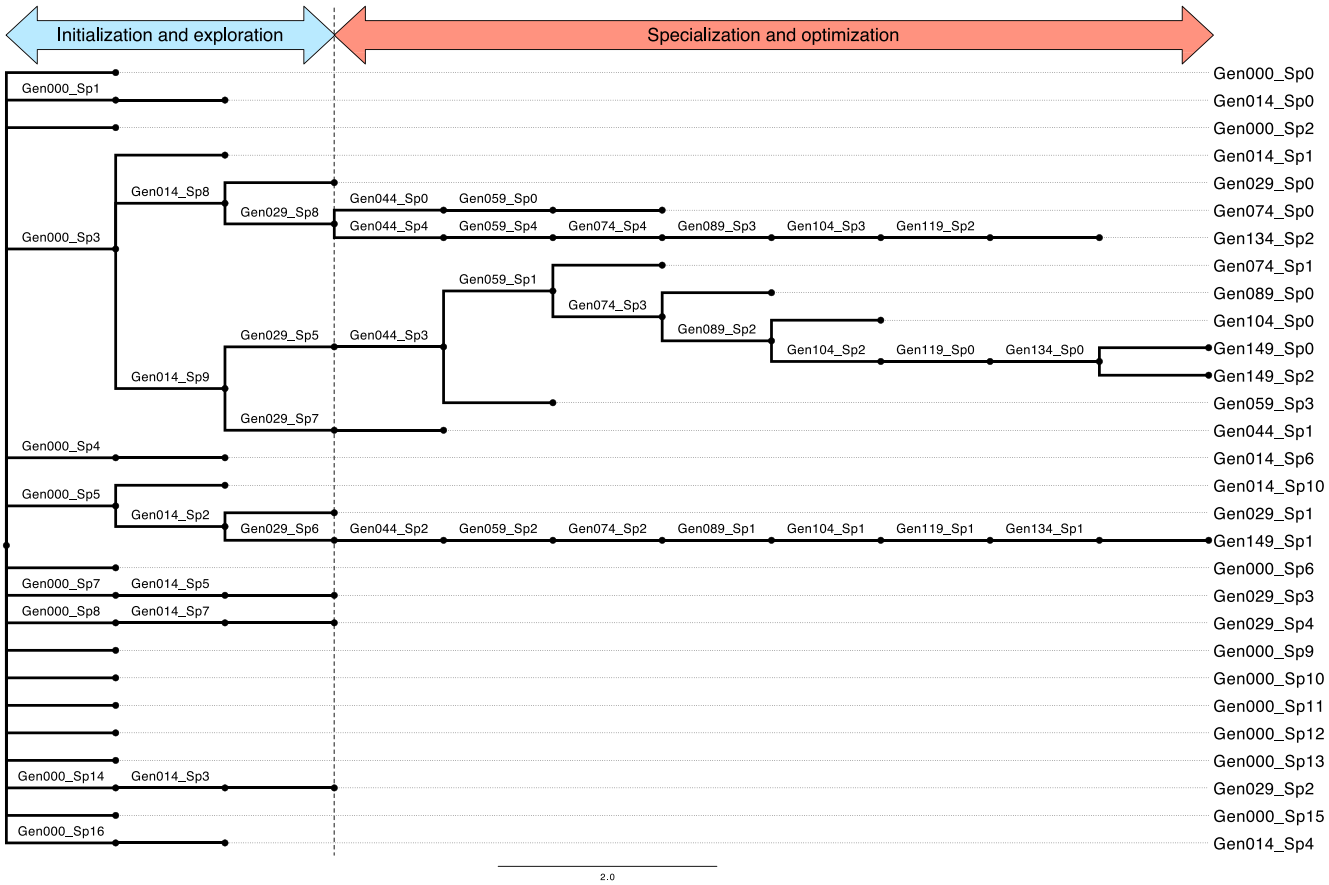

Fig. 16. Phylogenetic tree of species evolution in GRNEAT at intervals of ten generations for the first 100 generations.

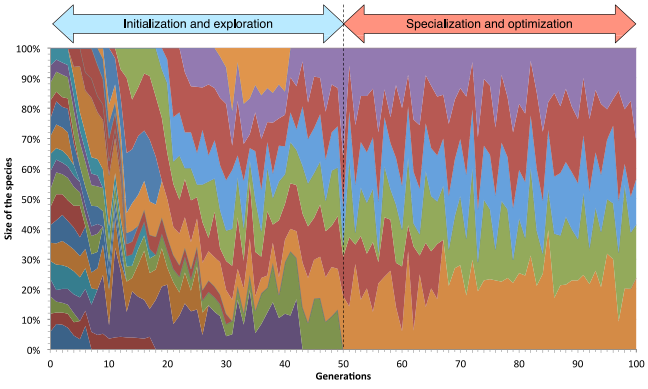

Fig. 17. Ratio of the species sizes among the evolution of the best trials of the GRNEAT on the CCP.

average distance of 10 000 randomly selected representatives from two species. In this case, we have additional information that is not often incorporated into phylogenetic algorithms, such as the neighbor-joining method [48], [49], specifically exact generation numbers. The tree is constructed by creating an edge to each species in generation 14 from the species in generation 0 with the smallest genetic distance. This is repeated for each pair of adjacent generations, creating forward edges from ancestral generations to the most recent.

We also present a plot of the ratio of species size in Fig. 17. Let us begin by discussing the species size plots, as they provide a clear overall picture of GRNEATs behavior. The population is initialized with a large number of small species, most of which are extinguished within the first 45 generations. During this initial period the allocation of species size via the fitness sharing mechanism is likely one of the mechanisms dominating the speciation dynamics. We can see a number of instances in these early generations, where some species expand and others shrink to extinction. As evolution reaches generation 50, only six species remain, all with similar sizes. Upon reaching generation 70, only five species remain, and this number is sustained for the remainder of the trial.

Now, we return to the corresponding dendrogram. Our nodes indicate species of a given generation. Nodes are placed in generational order and edges are unidirectionally forward in evolutionary time. Note that edges represent genetic similarity, and do not necessarily correspond to a direct ancestral relationship. The majority of species from the initial generation are not similar to subsequent species. We see that species 3 of the initial generation is likely the source of genetic material for 30% of the species in generation 14. In fact, this lineage is one of the most characteristic of the GRNEAT algorithm. Later in evolution, we can see that some species within this lineage branch but go extinct before the simulation is complete. However, we can see an increasing specialization occurring along this lineage. While some species branch and go extinct, others survive and continue to lead to new species and branching events. Ultimately two of the remaining three species have

arisen from the lineage of generation 0s species 3, while the other remaining species have descended along a minimally branching lineage from generation 0s species 5. Within this tree, we can see the core dynamics of speciation in the GRNEAT algorithm, such as branching, diversification, and specialization. Although a complete phylogenetic analysis of the GRNEAT algorithm would warrant an independent study, we can see from this example that the phylogenetic trends can provide information about evolutionary dynamics which may facilitate an understanding of the relationship between evolutionary algorithms and fitness landscapes.

V. Conclusion

In this paper, we have presented a novel approach to evolving GRNs. The method employs a number of features that improve the evolvability of a population of GRNs. The foremost feature is on an efficient tag-aligning crossover operator that maintains the network structures. A genetic distance metric is used to compare GRNs and speciate the population. The subdivision of the population into species facilitates diversity and helps the algorithm to keep small-sized networks. Finally, we inoculate the population with small genomes as opposed to the most common current practice of using a uniform distribution of genome sizes. This allows the algorithm to direct the evolutionary search toward growing the complexity of networks. The overall algorithm is inspired by the NEAT algorithm, but instead of requiring artificial historical markers as genetic annotations, we use the existing regulatory tags of GRNs to measure genetic distance. In this sense, the GRN representation is a particularly parsimonious fit for evolution by augmenting topology. We have shown that tag-aligning crossover is a significant piece of our evolutionary algorithm, but it is also further improved by speciation. On the other hand, the initialization method increases the general applicability of the algorithm by reducing the necessary parameter tuning: unlike a genetic algorithm, GRNEAT does not need any bounds on the maximum network size; network size is regulated by the algorithm itself.

The results obtained with this approach are consistently positive. Our evolutionary algorithm converges faster and to better solutions on most problems we have tested. In addition to that the networks generated by this new algorithm are smaller than the one generated by usually used methods. Although further experiments on each of the three key components of GRNEAT (aligned crossover, speciation, and initialization) may lead to even greater improvements, we find that the inclusion of these three features in our algorithm provide a consistent improvement over the current algorithms used to evolve GRNs. Future research is encouraged to focus on the speciation algorithm and, in particular, the offspring adjustment method. We also have introduced some modifications to the speciation process in comparison to the original speciation method used in NEAT. First, we used a self-tuning speciation threshold instead of a global fixed threshold to reduce the number of parameters the user has to set up when using this algorithm, which was used in later iterations of the NEAT algorithm. Another modification concerns the deletion of unviable species: deleting species can be dangerous because it can lead to the deletion of potentially good solutions but, in our opinion, this extinction is an important drive of the population dynamics. This also implies the use of a species extension method in order to keep the population size constant. In this paper, we utilize a rejection sampling method when adding new individuals to replace deleted species in order to keep the intraspecies genomes structurally consistent. Currently, species are compared with a standard fitness adjustment mechanism. However, alternative methods of species management, such as crowding, have recently been used in particle swarm optimization to evaluate the improvement of a species over multiple generations based on the quantity of genomes that improves their parents fitnesses [50]. Statistical classifications such as k-means clustering [51] could also improve speciation. The impact of all these modifications and newly introduced mechanisms must now be analyzed in detail in order to validate their benefits. Each of them must be taken separately to evaluate their impact on the algorithm's convergence and the population diversity. Some of them could also be introduced to the NEAT algorithm for a comparison of their impact on neuroevolution.

Considering multiobjective optimization in addition to these mechanisms could improve the evolvability of the GRN on complex problems but, in our opinion, the use of multiple objectives should be restrained to the problem specifically and not to the network structure itself in order to keep the evolution manageable. Actually, this approach has already been successfully used in [52]. Objectives such as minimizing the size of networks can be directly tackled by the genetic operators (crossover and mutation) as is shown in this paper.

The original NEAT algorithm has been shown to be an efficient method for the evolution of neural networks in a number of works [53]–[56]. We have now shown that some of the key principles of the NEAT algorithm can be used to evolve GRNs with the GRNEAT algorithm. This result thus enables a broader use of GRNs for tasks in simulation, control, and problem-solving than heretofore. This also suggests that the same mechanisms could be applied to other kinds of networks (Boolean, Bayesian, etc.) to obtain similar results. The challenge in designing an algorithm for evolution by augmenting topology is the implementation of a metric between networks that measures the structural differences between networks. Once this metric is defined, other mechanisms of the algorithm will closely resemble those employed by the NEAT and GRNEAT algorithms.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] E. H. Davidson, *The Regulatory Genome: Gene Regulatory Networks in Development and Evolution*. Amsterdam, The Netherlands: Academic Press, 2006.

[2] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *J. Theor. Biol.*, vol. 22, no. 3, pp. 437–467, 1969.

[3] M. Joachimczak and B. Wróbel, "Evo-devo *in silico*: A model of a gene network regulating multicellular development in 3D space with artificial physics," in *Proc. Alife XI Conf.*, Winchester, U.K., 2008, pp. 297–304.

[4] R. Doursat, "Facilitating evolutionary innovation by developmental modularity and variability," in *Proc. 11th Annu. Conf. Genet. Evol. Comput.*, Montreal, QC, Canada, 2009, pp. 683–690.

[5] S. Cussat-Blanc, J. Pascalie, S. Mazac, H. Luga, and Y. Duthen, "A synthesis of the cell2organ developmental model," in *Morphogenetic Engineering*. Berlin, Germany: Springer, 2012, pp. 353–381.

[6] J. Ziegler and W. Banzhaf, "Evolving control metabolisms for a robot," *Artif. Life*, vol. 7, no. 2, pp. 171–190, 2001.

[7] M. Nicolau, M. Schoenauer, and W. Banzhaf, "Evolving genes to balance a pole," in *Proc. 13th Eur. Conf. Genet. Program. (EuroGP)*, Istanbul, Turkey, 2010, pp. 196–207.

[8] M. Joachimczak and B. Wróbel, "Evolving gene regulatory networks for real time control of foraging behaviours," in *Proc. Alife XII Conf.*, Odense, Denmark, 2010, pp. 348–355.

[9] Y. Jin, H. Guo, and Y. Meng, "A hierarchical gene regulatory network for adaptive multirobot pattern formation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 805–816, Jun. 2012.

[10] S. Cussat-Blanc, S. Sanchez, and Y. Duthen, "Controlling cooperative and conflicting continuous actions with a gene regulatory network," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Granada, Spain, 2012, pp. 187–194.

[11] L. Benuskova, V. Jain, S. G. Wysoski, and N. K. Kasabov, "Computational neurogenetic modelling: A pathway to new discoveries in genetic neuroscience," *Int. J. Neural Syst.*, vol. 16, no. 3, pp. 215–226, 2006.

[12] K. I. Harrington, E. Awa, S. Cussat-Blanc, and J. Pollack, "Robot coverage control by evolved neuromodulation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Dallas, TX, USA, 2013, pp. 1–8.

[13] J. C. Bongard and R. Pfeifer, "Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny," in *Proc. Genet. Evol. Comput. Conf.*, 2001, pp. 829–836.

[14] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," *Artif. Life*, vol. 9, no. 2, pp. 93–130, 2003.

[15] F. Roth, H. Siegelmann, and R. Douglas, "The self-construction and -repair of a foraging organism by explicitly specified development from a single cell," *Artif. Life*, vol. 13, no. 4, pp. 347–368, 2007.

[16] P. J. Bentley, "Evolving beyond perfection: An investigation of the effects of long-term evolution on fractal gene regulatory networks," *Biosystems*, vol. 76, no. 1, pp. 291–302, 2004.

[17] J. F. Knabe, C. L. Nehaniv, and M. J. Schilstra, "Genetic regulatory network models of biological clocks: Evolutionary history matters," *Artif. Life*, vol. 14, no. 1, pp. 135–148, 2008.

[18] S. Cussat-Blanc, S. Sanchez, and Y. Duthen, "Simultaneous cooperative and conflicting behaviors handled by a gene regulatory network," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–8.

[19] M. Joachimczak and B. Wróbel, "Processing signals with evolving artificial gene regulatory networks," in *Proc. Alife XII Conf.*, Odense, Denmark, 2010, pp. 203–210.

[20] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[21] S. Forrest, "Genetic algorithms: Principles of natural selection applied to computation," *Science*, vol. 261, no. 5123, pp. 872–878, 1993.

[22] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996.

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

[24] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, NY, USA: Wiley, 1966.

[25] T. Reil, "Dynamics of gene expression in an artificial genome-implications for biological and artificial ontogeny," in *Advances in Artificial Life* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 1999, pp. 457–466.

[26] R. Lifton, M. Goldberg, R. Karp, and D. Hogness, "The organization of the histone genes in drosophila melanogaster: Functional and evolutionary implications," in *Cold Spring Harbor Symposia on Quantitative Biology*, vol. 42. New York, NY, USA: Cold Spring Harbor Lab. Press, 1978, pp. 1047–1051.

[27] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice*, R. L. Riolo and B. Worzel, Eds. New York, NY, USA: Springer, 2003, ch. 4, pp. 43–62.

[28] J. Bongard and R. Pfeifer, "Evolving complete agents using artificial ontogeny," in *Morpho-Functional Machines: The New Species (Designing Embodied Intelligence)*. Tokyo, Japan: Springer, 2003, pp. 237–258.

[29] L. Wolpert, "The French flag problem: A contribution to the discussion on pattern development and regulation," in *Towards a Theoretical Biology*, vol. 1. Edinburgh, U.K.: Edinburgh Univ. Press, 1968, pp. 125–133.

[30] A. Lindenmayer, "Developmental systems without cellular interactions, their languages and grammars," *J. Theor. Biol.*, vol. 30, no. 3, p. 455, 1971.

[31] J. Miller, "Evolving developmental programs for adaptation, morphogenesis, and self-repair," in *Proc. Eur. Conf. Artif. Life (ECAL)*, Dortmund, Germany, 2003, pp. 256–265.

[32] C. Bowers, "Simulating evolution with a computational model of embryogeny: Obtaining robustness from evolved individuals," in *Advances in Artificial Life* (Lecture Notes in Computer Science), vol. 3630. Berlin, Germany: Springer, 2005, p. 149.

[33] S. Cussat-Blanc, N. Bredeche, H. Luga, Y. Duthen, and M. Schoenauer, "Artificial gene regulatory networks and spatial computation: A case study," in *Proc. Eur. Conf. Artif. Life (ECAL)*, Paris, France, 2011, pp. 192–199.

[34] M. Kirschner and J. Gerhart, "Evolvability," *Proc. Nat. Acad. Sci.*, vol. 95, no. 15, pp. 8420–8427, 1998.

[35] S. Ando and H. Iba, "Inference of gene regulatory model by genetic algorithms," in *Proc. Congr. Evol. Comput.*, vol. 1. Seoul, Korea, 2001, pp. 712–719.

[36] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.

[37] C. Darwin, *On the Origin of Species*. New York, NY, USA: Modern Library, 1859.

[38] S. Wright, "Breeding structure of populations in relation to speciation," *Amer. Nat.*, vol. 74, no. 752, pp. 232–248, 1940.

[39] C. Oei, D. Goldberg, and S. Chang, "Tournament selection, niching, and the preservation of diversity," Illinois Genet. Algorithms Lab., Univ. Illinois, Urbana, IL, USA, Tech. Rep. 91011, 1991.

[40] S. Mahfoud, "Niching methods for genetic algorithms," Illinois Genet. Algorithms Lab., Univ. Illinois, Urbana, IL, USA, Tech. Rep. 95001, 1995.

[41] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genet. Algorithms Appl.*, 1987, pp. 41–49.

[42] J. R. Koza, "A genetic approach to the truck backer upper problem and the inter-twined spiral problem," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 4. Baltimore, MD, USA, 1992, pp. 310–318.

[43] H. Juille, J. B. Pollack, L. J. Fogel, P. J. Angeline, and T. Baeck, "Co-evolving intertwined spirals," in *Proc. 5th Evol. Program.*, 1996, pp. 461–467.

[44] K. J. Lang, "Learning to tell two spirals apart," in *Proc. Connection. Models Summer School*, Pittsburgh, PA, USA, 1988, pp. 52–59.

[45] S. Cussat-Blanc and J. Pollack, "Using pictures to visualize the complexity of gene regulatory networks," in *Proc. Alife XIII Conf.*, East Lansing, MI, USA, Jul. 2012, pp. 491–498.

[46] D. Dickmanns, J. Schmidhuber, and A. Winklhofer, "Der genetische algorithmus: Eine implementierung in prolog," Fortgeschrittenenpraktikum, Institut f ur Informatik, Technische Universität München, München, Germany, 1987.

[47] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, vol. 1. Cambridge, MA, USA: MIT Press, 1994.

[48] J. A. Studier and K. J. Keppler, "A note on the neighbor-joining algorithm of Saitou and Nei," *Mol. Biol. Evol.*, vol. 5, no. 6, pp. 729–731, 1988.

[49] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees," *Mol. Biol. Evol.*, vol. 4, no. 4, pp. 406–425, 1987.

[50] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance," in *Evol. Multi-Criterion Optim.*, Guanajuato, Mexico, 2005, pp. 505–519.

[51] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 1. Berkeley, CA, USA, 1967, pp. 281–297.

[52] H. Guo, Y. Meng, and Y. Jin, "A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network," *Biosystems*, vol. 98, no. 3, pp. 193–203, 2009.

[53] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 653–668, Dec. 2005.

[54] K. O. Stanley and R. Miikkulainen, "Evolving a roving eye for go," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Seattle, WA, USA, 2004, pp. 1226–1238.

[55] J. E. Auerbach and J. C. Bongard, "Dynamic resolution in the co-evolution of morphology and control," in *Proc. Alife XII Conf.*, Odense, Denmark, 2010, pp. 451–458.
[56] E. Haasdijk, A. A. Rusu, and A. Eiben, "HyperNEAT for locomotion control in modular robots," in *Evolvable Systems: From Biology to Hardware*. Berlin, Germany: Springer, 2010, pp. 169–180.

Sylvain Cussat-Blanc received the Ph.D. degree from the University of Toulouse, Toulouse, France, in 2009, specializing in a cell-based developmental model to produce artificial creatures.

He was a Post-Doctoral Fellow with J. Pollack in 2011, where he applied his research to evolutionary robotics with the aim to automatically design the real-modular robots' morphologies. He was a Researcher and a Teacher with University of Toulouse. His research interests include developmental models, gene regulatory networks, evolutionary robotics, artificial life, and evolutionary computation in general.

Dr. Cussat-Blanc is a member of the Institute of Research in Computer Science of Toulouse, Toulouse, which is a research unit of the French National Center for Research.

Kyle Harrington received the B.A. degree in artificial life from Hampshire College, Amherst, MA, USA, and the Ph.D. degree in computer science from Brandeis University, Waltham, MA, USA, in 2007 and 2014, respectively.

He was a Visiting Scholar with BINDS Laboratory, University of Massachusetts–Amherst, Amherst, from 2010 to 2011, and was a Visiting Scientist with Janelia Farm Research Campus, Howard Hughes Medical Institute, Chevy Chase, MD, USA, in 2014. He is currently a Post-Doctoral Scientist with the Computational Biology Laboratory, Department of Vascular Biology, Beth Israel Deaconess Medical Center, Harvard Medical School, Boston, MA, USA. His research interests include artificial life, evolutionary and dynamical systems, physical systems, neural and machine learning, and game theory.

Jordan Pollack received the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, Urbana, IL, USA, in 1987.

He is a Professor of Computer Science and Complex Systems, the Chair of Computer Science Department, and the Director of DEMO Laboratory, Brandeis University, Waltham, MA, USA. He was involved with several startup companies, including Abuzz, Watertown, MA, USA; Affinnova, Waltham, MA, USA; Nannon Technology, Wayland, MA, USA; and Thinmail, Sudbury, MA, USA. His research interests include artificial intelligence, artificial life, neural and evolutionary computing, robotics, complex and dynamic systems, educational technology, and intellectual property law.

Dr. Pollack was named as one of the Massachusetts Institute of Technology's Technology Review "TR 10" in 2001 for his laboratory work on automatically evolved and manufactured robots.