Chapter 1
Behavior-Finding: Morphogenetic Designs Shaped by Function

Daniel Lobo, Jose David Fernández, and Francisco J. Vico

Abstract Evolution has shaped an incredible diversity of multicellular living organisms, whose complex forms are self-made through a robust developmental process. This fundamental combination of biological evolution and development has served as an inspiration for novel engineering design methodologies, with the goal to overcome the scalability problems suffered by classical top-down approaches. Top-down methodologies are based on the manual decomposition of the design into modular, independent subunits. In contrast, recent computational morphogenetic techniques have shown that they were able to automatically generate truly complex innovative designs. Algorithms based on evolutionary computation and artificial development have been proposed to automatically design both the structures, within certain constraints, and the controllers that optimize their function. However, the driving force of biological evolution does not resemble an enumeration of design requirements, but much rather relies on the interaction of organisms within the environment. Similarly, controllers do not evolve nor develop separately, but are woven into the organism's morphology. In this chapter, we discuss evolutionary morphogenetic algorithms inspired by these important aspects of biological evolution. The proposed methodologies could contribute to the automation of processes that design "organic" structures, whose morphologies and controllers are intended to solve a functional problem. The performance of the algorithms is tested on a class of optimization problems that we call *behavior-finding*. These challenges are not explicitly based on morphology or controller constraints, but only on the solving abilities and efficacy of the design. Our results show that morphogenetic algorithms are well suited to behavior-finding.

Daniel Lobo
Biology Department and Tufts Center for Regenerative and Developmental Biology, Tufts University, 200 Boston Ave., Suite 4600, Medford, MA, 02155, USA. e-mail: daniel.lobo@tufts.edu

Jose David Fernández and Francisco J. Vico
Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga Severo Ochoa 4, 29590 Málaga, Spain. e-mail: {josedavid,fjv}@geb.uma.es

1.1 Introduction

Engineers have made remarkable progress in their ability to design complex products. However, current engineering practice still favors a top-down approach, where the main problem is manually divided into smaller ones in order to maintain the overall complexity reasonably manageable. This procedure is rather loosely defined and ultimately relies on human expertise and creativity, which are skills that typically involve high costs, are unreliable and are difficult to formalize. Moreover, the ever increasing complexity of current engineered systems and robustness requirements is reaching the feasibility limits of the current paradigm, forcing the implementation of new engineering methodologies.

Inspired by the biological evolution and morphogenesis of organisms, recent advances in the discipline of evolutionary computation propose a radically different approach. Genetic algorithms combined with artificial development mechanisms operate over a population of individuals encoded by *genomes* that govern a morphogenetic process producing self-constructed designs [74]. That is, the genome is not a blueprint of the design, but the set of instructions that indirectly build it. The evolutionary operations (mutations and crossover) are applied to the developmental generative process that build the design, not to the design itself. This approach has been shown to overcome the issues of scalability, adaptability, and evolvability present in traditional evolutionary algorithms (based on a genomic representation that encodes the design in a explicit way) when applied to complex problems [30]. As a result, evolutionary developmental algorithms have been tried in a wide range of design problems, including the structure and controller of robots [20], digital creatures in Artificial Life studies [4, 29, 35, 56, 73], and computer animated characters [71]. In almost all models, however, the control system is fairly complex (often based on some kind of recurrent neural network), and in many cases, we believe, unnecessarily so.

In a seminal work [53], Chandana Paul demonstrated that a whole body-control system is able to perform more complex computations than the control system alone. This observation spawned the concept of *morphological computation*—a design methodology for robotic-like agents to exploit the dynamics of interaction between the body and the control system, resulting in minimal control systems. Several applications have been proposed in the field of robotics, including the design of semi-passive bipedal robots [48], tensegrity robots whose complex, coupled non-linear dynamics are harnessed to generate a gait pattern with minimal control [53], and robots with open-loop control systems and minimal numbers of degrees of freedom that can self-stabilize into fast gait patterns and generate diverse behaviors through the interaction between body and control system [55].

In a line similar to morphological computation, we present here a set of methodologies for the evolution of designs optimized to solve a functional problem. We call this class of problems *behavior-finding*, in contrast to form-finding problems focused on the optimization of a structure or morphology according to a set of constraints. The morphogenetic techniques that we present here do not assume a clear-cut boundary between structure and controller. As the encoding of the agents

has no provision for the complexification of their control systems in an explicit way, innovative designs arise by evolutionary optimization, together with their control systems implicitly contained in their structure. The results obtained from the application of the proposed methodologies to the set of behavior-finding problems reflect the potential of these algorithms to find novel and diverse designs, which are characterized by qualitative biological properties that would be otherwise hard to reproduce by traditional engineering approaches.

In the following sections, we present three behavior-finding studies carried out in our group: an evolutionary developmental model based on graph grammars (Section 1.2), a more refined version of the same model adding complex regulatory mechanisms to the development (Section 1.3), and a model solving a biologically inspired behavior-finding problem, in which the agents do not undergo artificial development per se, but an innovative mutation operator provides some of the advantages of artificial development in the sense of evolutionary optimization (Section 1.4).

1.2 A graph-grammar-based developmental model for behavior-finding

New insights in developmental biology are configuring a new algorithmic paradigm that may dramatically change the tenets of evolutionary computation. Artificial development is a discipline that mimics the mechanisms of natural development, or "embryogenesis", by which a group of living systems develop from one single cell into complete multicellular organisms. In search of methods to produce phenotypes with high structural complexity, evolutionary algorithms have incorporated mechanisms inspired by embryonic development, such as gene expression, morphogenesis, and cellular differentiation [74].

An important aspect in the implementation of an evolutionary algorithm is the genotype-to-phenotype encoding scheme [35]. Traditional evolutionary algorithms use a "direct" encoding method, meaning that the individual is represented by a list of data explicitly linked to its properties. However, direct mappings have been shown to lack effectiveness in complex problems, due to their limitations in scalability, adaptability, and evolvability [30]. As a result, evolutionary algorithms with "indirect" encodings have started to be applied to a wide range of problems (for a review see [74]).

Various computational paradigms that rely on developmental mechanisms and indirect encodings have been proposed for the design of structures and robotic agents. Cellular automata have been used as a model of development to generate simple shapes [21, 22], biological processes (e.g., gastrulation and limb budding) [26], or specific 2D [6] and 3D [2] target patterns. Rules that fire cell functions, such as mitosis, apoptosis, or migration, have been implemented to design 3D geometrical shapes [78], tessellating tiles in a grid [3], and 3D morphologies [38]. Rules have also been combined with gene regulatory networks to control their activation in the evolution of 2D shapes [14], 2D patterns [81, 19], and 3D multicellu-

lar organisms [15, 16, 31]. An alternative combination of rules with diffusion of chemicals has been proposed to develop simple shapes [24]. Other development models have taken a "generative" approach, such as string grammars to develop house plants [64], or L-systems to encode plant morphologies [57, 34], 3D branched organisms [7], objects made of voxels [28, 27], and surfaces in 3D spaces [25]. Shape grammars [76], a production system to generate geometric shapes, have been evolved for architectural designs [67]. Grammatical evolution, a method to evolve productions of a string grammar by encoding the rules of the production in a binary genome [51], has been applied to shape grammars to evolve simple shapes [52]. Other paradigms used in the literature are genetic programming for 2D patterns [49], instructions for a block builder based on turtle graphics [59, 60], neural networks to develop multicellular organisms [12, 13], and functions to develop 2D images with structural motifs [75]. Finally, models that resemble in more detail biological processes, as proteins concentrations or cell chemical signaling, have been also proposed [39, 63, 61, 17, 62, 1]. While these models make use of developmental methods and indirect encodings, and undoubtedly constitute a great achievement in the field, they are not focused on the implementation of engineering structures, thus their practical utility is still limited.

Little work has been done in designing *engineering* structures with developmental methods. Shea et al. [69, 70] used simulated annealing to evolve shape grammars for the automation of the design process of roof trusses and discrete structures. Rudolph and Alber [65] proposed an evolutionary algorithm based on genetic programming to evolve node-based graph grammars that encoded structures resembling transmission towers. Kowaliw et al. [36] used a cellular growth method to generate truss designs, controlling developed morphology via a variety of fitness functions. Finally, Lobo et al. [43] proposed an encoding based on a construction tree to evolve the sequence of modifications that can transform a given truss structure into a new one that serves a different function. In summary, these works represent promising examples of the application of developmental methods to the design of engineering structures.

1.2.1 The model

In this section we present a model of development (published in [45]) of artificial organisms (agents) based on string-regulated graph grammars [46]. The organisms are connected and directed geometrical graphs. In these graphs, edges can be interpreted as cells (black lines in Fig. 1.1), and the nodes are junctions where cells get attached to each other. In physical simulations, edges are considered as damped springs, all with the same stiffness and damping constants, while nodes are free movable joints, all with the same mass and friction with the medium. The connectivity and rest lengths of the springs are determined during development, as described in the following paragraphs.

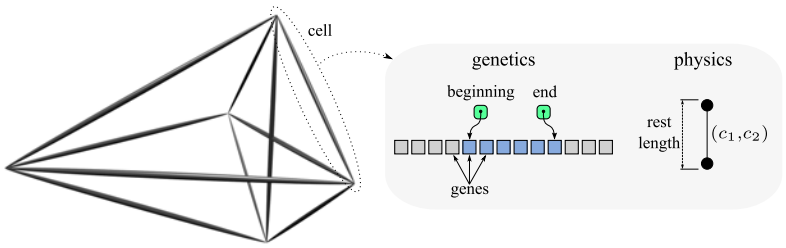


Fig. 1.1 Diagram of a developing multicellular organism, representing the genetic information (genome with beginning and end of the cell domain) and physical information (rest length with connectivity properties; see text) of a cell.

Development starts with a graph of a single edge, resembling the zygote in living organisms. Edges, like cells, perform developmental actions during embryogenesis, which consist of a sequence of regulated applications of rewriting rules that can (1) alter the properties of one edge in a graph, (2) replace the edge by two new edges, and (3) affect the future regulation of the resulting edge or edges. The genome of an individual is implemented as a string that results from the concatenation of substrings, each containing an index to a rule, and numeric values to instantiate the rule's attributes. These substrings will be referred to as *genes*, since they represent indivisible units of genetic expression.

The rewriting rules are described first for a 2D model (Fig. 1.2a):

- The resize rule (*R*) may affect the rest length of an edge. It includes an attribute determining the new length of the edge, in the $[0.2, 5]$ range, relative to the current one. If the value of the attribute is greater than 1, then the rest length is increased, otherwise it is decreased.
- The duplicate rule (*D*) replaces an edge by two edges connecting the same nodes of the original edge. It also defines an attribute that is used to apportion the expression domain of the original edge.
- The split rule (*S*) replaces an edge by two new edges connecting the original nodes in a sequence. It defines two attributes, the first one determines the proportional length of the new edges, with respect the length of the original edge, and the second one concerns the distribution of the domain of expression, similar to the duplicate rule. The new node created in a split rule is slightly misaligned to the right with respect to the direction of the original edge, provoking a bias towards a given direction in case of compression.
- Finally, the connection rule (*C*) is included to affect the connectivity properties of the nodes that delimit an edge. Two attributes (c_1 and c_2 in Fig. 1.1, one for each node of the edge) determine the activation state of the edges in the following way: a node will be active if at least one of the edges converging on it has its corresponding attribute set active. Two nodes will be connected by a new edge if both nodes are active and the distance between them is below a certain threshold. This new edge will be inserted as a structural element, with an empty expression domain, connection attributes set to inactive, and a rest length equal to the av-

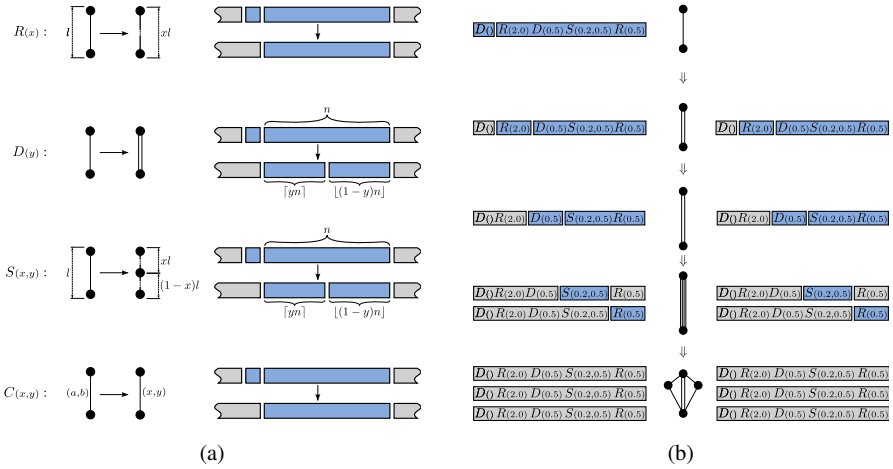


Fig. 1.2 Production system of the regulated graph grammar. (a) Definition of the rewriting rules: transformation of an edge (left), and new domain(s) of expression (right). (b) Four derivation steps for the genome $D()R(2.0)D(0.5)S(0.2,0.5)R(0.5)$.

erage of the rest lengths of the edges converging to the two nodes. The creation of these structural edges facilitates the emergence of *tensile integrity*, or "tensegrity" [50], by connecting nodes that are later pushed toward opposite directions. Tensegrity is a useful property in several engineering problems [79].

All rules have an effect on the domain of the cell. In the case of the resize rule, only the beginning of the domain is altered, in a way that discards the first gene of the current domain. Duplicate and split rules generate two domains by splitting the current one into two, according to the value of their apportion attribute, in the $[0,1]$ range. These rules assign to the first descendant cell a domain extending from the second gene of the current domain to an intermediate gene (as shown in Fig. 1.2a), the rest of the domain being assigned to the second cell. As a special case, when the gene does not specify a value for this attribute, the domain for both descendants is the same, and extends from the second gene to the end of the domain in the progenitor cell. As a consequence of this, concrete regions of the genome can be expressed simultaneously in different parts of the organism, allowing modularity.

We will refer to this model as *ELSA* (standing for Expression by Limited Splitting Actions). Fig. 1.2b illustrates how an example genome ($D()R(2.0)D(0.5)S(0.2,0.5)R(0.5)$) develops into a final organism under the *ELSA* scheme. A variant of this model, *ELSA3*, incorporates necessary mechanisms for the morphologies to explore the third spatial dimension: the geometric graphs are defined in three dimensions and the split rule defines a new parameter to set the inclination of the new edges in the original edge's normal plane.

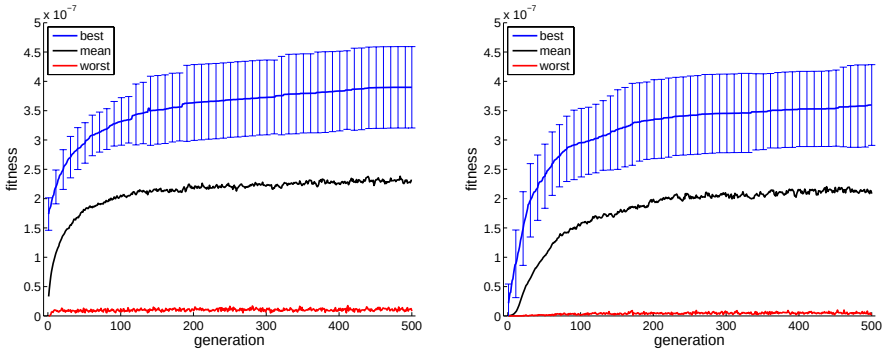


Fig. 1.3 Average best, mean, and worst fitness for each generation of 50 evolutionary runs using (a) direct encoding and (b) indirect encoding. The length of the error bars represents the standard deviation of the best fitness.

1.2.2 Evolutionary search

We test the potential of *ELSA3* in an evolutionary search with the following problem: how a vehicle, configured as a mass-spring tensegrity structure, can land properly when falling from a given height. In tensegrity structures, some edges bear compressive forces while others bear stretching forces, such that the sum of forces incident in each node of the structure is zero. The performance of *ELSA3* is also compared to a direct encoding scheme for tensegrity structures, such as the model proposed in [54].

Each individual in the population develops from its genetic information, and afterwards the free fall to the ground is physically simulated (with a given initial speed vector). The fitness of an individual represents how "well" it managed landing, by calculating how far it moved from a target point, and how the impact was absorbed. The following equation combines these measures into a single value:

$$f(s_k) = \frac{1}{d(c_1, c_2) \sum\limits_{i \leq n} |\mathbf{j}_i|}$$

where $f(s_k)$ means the fitness value of the k-th individual, simulated for n time steps. The first divider represents the Euclidean distance traveled by the lander after the impact, i.e., between the first contact and the resting site. More precisely, c_1 is the projection of the center of mass on the plane when the structure touches down for the first time, and c_2 is the projection of the center of mass on the plane when the structure has completed landing. The second divider represents the "impact disturbance" of the structure, computed as the accumulated jerk (first derivative of the acceleration) in the sequence of points that the center of mass describes during landing, where $\mathbf{j}_i$ denotes the jerk vector at time step i. The jerk behaves as a predictor for large accelerations of short duration [68], a magnitude that must be minimized

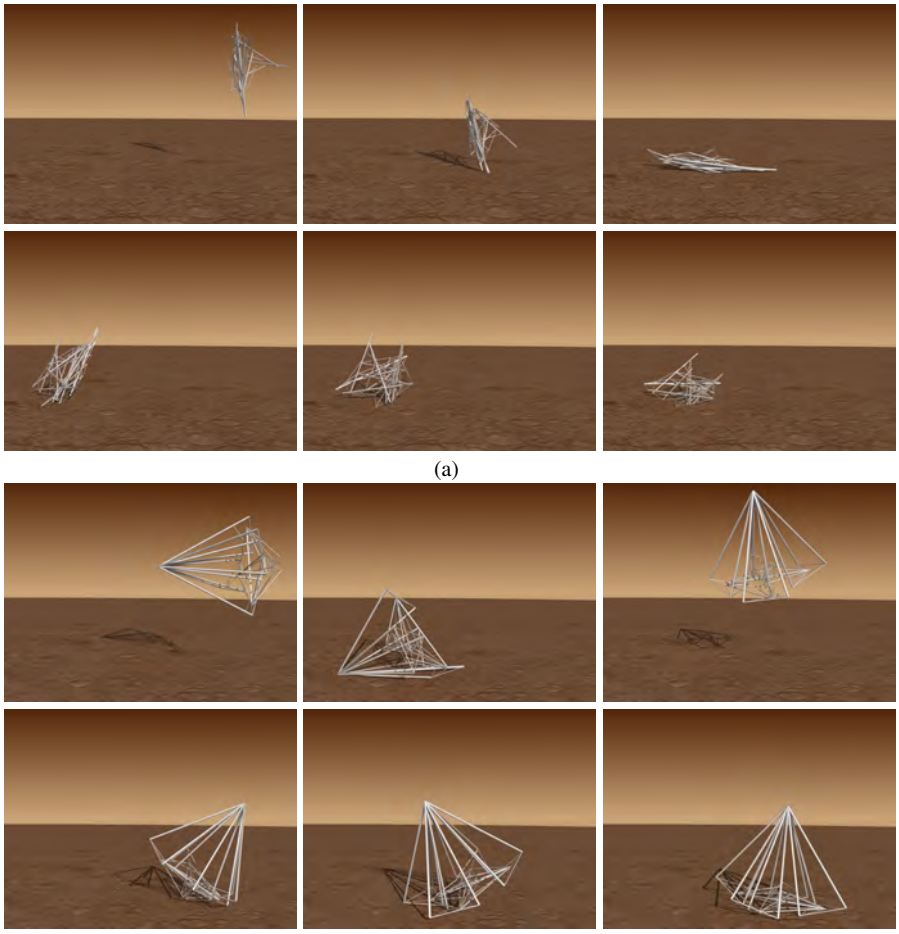
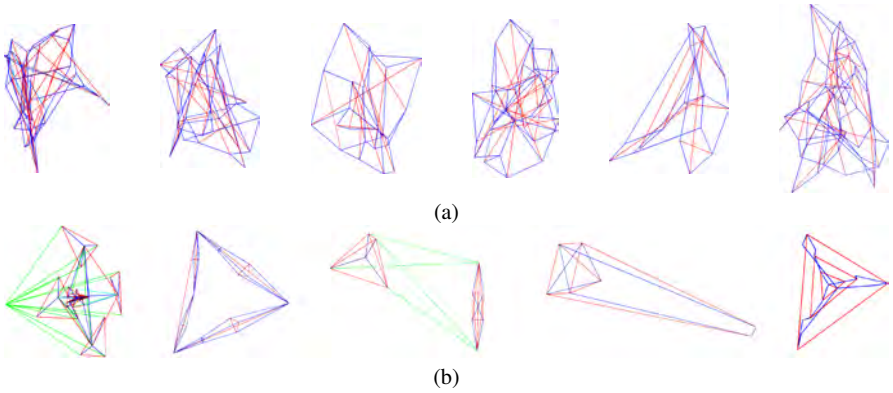


Fig. 1.4 From left to right, and from top to bottom: landing sequences of the best structures with direct (a) and indirect (b) encodings.

in vehicles having to decelerate abruptly. As a consequence, the resulting function computes a higher fitness value for structures that slow down smoothly and remain close to the impact point.

A standard experiment consisted of simulating the evolution of a fix-sized population of 250 structures during 500 generations. The initial population contained genomes made of a variable number of genes (from 2 to 8), where genes and their attribute values were randomly generated. The genomes of the individuals forming the new populations were obtained by mutating existing ones, according to four operators:

- Insertion: a new random gene is inserted at a random position.



(a)

(b)

Fig. 1.5 Best tensegrity structures evolved in several different evolutionary runs using (a) direct and (b) indirect encodings. Edges are colored according to their state: compressed (red), stretched (blue), and relaxed (green).

- Deletion: a gene is randomly chosen and removed.
- Replacement: a gene is randomly chosen and replaced by a new random gene.
- Single-attribute: an attribute of a randomly chosen gene is replaced by a new random attribute.

The probabilities of occurrence associated with each operator were 0.2, 0.1, 0.1 and 0.6, respectively. A new generation was obtained by mutating and replacing 90% of the current population. Selection was implemented as a roulette algorithm, in which the probability of selection was linearly proportional to the fitness. In order to minimize the disruption of mutations, they were more likely to occur at positions of the genome toward the right side. Since the genome was interpreted from left to right, this strategy had a tendency to keep the first stages of development unaltered. Elitism of one individual was also implemented.

1.2.3 Evolution of modular designs

The evolutionary search was run 100 times to test the models (50 times for *ELSA3*, 50 times for the direct coding scheme of [54]). Fig. 1.3 shows the average fitness curves over 50 runs using the direct (a), and indirect (b) encoding methods. The best-individual curves (blue) are accompanied by error bars, representing the standard deviation. In both cases we observed a fast fitness improvement in the first 100 generations, followed by a gradual refinement as evolution progressed. However, the best individuals with direct encoding started with an average fitness value several times larger than the best individuals with indirect encoding, due to the morphological differences between early individuals in both encodings. For this reason, the direct-encoding fitness curves generally reached slightly better values than the

indirect-encoding fitness curves. The black and red curves represent the average mean fitness, and average worst fitness, respectively. We observed that both strategies performed equally well in the proposed landing test. However, as we will see, structures with indirect encoding are better suited to engineering.

In Fig. 1.5, the best structures evolved by direct and indirect encodings are shown. The best structures obtained from evolutionary runs with direct encoding are characterized by their irregular, shapeless organization and lack of modularity (Fig. 1.5a). In clear contrast, the best structures evolved with indirect encoding present bilateral and radial symmetries, and clear modularity (Fig. 1.5b). See [45] for an extended discussion on the ontogeny and phylogeny of these structures. Similarly, Fig. 1.4 shows the landing sequence of the best structures with direct and indirect encodings. In the first case (Fig. 1.4a), the irregular shape helps during impact absorption, landing about 3.5 units away from the impact point. In the second case (Fig. 1.4b), the structure shows a regular geometry and four modular subunits that help stabilizing the position of the structure after impact, landing about 1.5 units away from the impact point.

1.2.4 Advantages of the model

These results demonstrate that both encoding strategies can find good solutions to the proposed problem. The differences concern their particular way of prospecting candidate forms, mainly derived from the search space that each type of encoding defines. Direct encoding is based on a genome that describes the exact positions of every node in the structure. This independence of the nodes generates designs without any patterning or regularity (Fig. 1.5a). Furthermore, mutations have only a local effect on the structure, preventing the evolution of iterated modules. In contrast, the proposed indirect encoding scheme *ELSA3*, based on regulated graph grammars, generates structures that follow patterns and regularities due to their rewriting nature, such as triangular and square pyramids, bipyramids, and bilateral and rotational symmetries (see [45]). Moreover, grammar productions are regulated by a string that is copied and transmitted through edge duplication, in a way analog to the mechanisms of copy and transmission of the genome during division of biological cells. This process causes the reuse of genetic information (both across development and evolution), which in turn allows the repetition of modules in the structures.

In summary, we have shown that the use of an indirect encoding scheme facilitated the emergence of qualitative biological properties in the solutions. Direct and indirect encodings differed in the following aspects:

1. **Regularity:** while the direct encoding method explores irregular structures, *ELSA3* exploits mechanisms to obtain symmetric and modular configurations.
2. **Organicity:** solutions evolved with the indirect encoding method have an "organic" appearance, in the sense the developed organism can be segmented into structural (and, possibly, functional) parts or modules.

3. **Generalization capability:** the regularities observed in indirectly encoded organisms are not superficial. Symmetry allows the structure to display a similar behavior when landing conditions vary slightly. The landing problem was solved by the indirect encoding structures for a wide range of initial angles and speeds, whereas the irregular forms created by direct encoding proved to be highly sensitive to initial conditions.
4. **Manufacturability:** properties of regularity and organicity are well suited to future industrial manufacturing of solution individuals. Even though it was not the objective of this study, the modularity of landers obtained with indirect encoding (see Fig. 1.5b) facilitates the independent construction and assembly of the different parts of the vehicles, in particular allowing the serial production of components. This would not be the case with irregular forms (Fig. 1.5a), where the diversity of constituent parts hinders production and assembly.

1.3 A gene regulatory developmental model for behavior-finding

In the model presented in Section 1.2, diverse organisms developed by the application of rewriting rules, which can be considered actions performed by the cells. Since the genome is a sequence of the rules to be applied, regulation of rules is encoded in the sequence. In nature, however, the actions (the rules) performed by the cells of a developing organism are controlled by its gene regulatory system. In fact, the evolution of diverse phenotypes may very well have to do with a complexification of the *regulation* of gene expression [41], in addition to direct gene mutation. In this sense, it has been pointed out that evolutionary change in body plans could be the result of change in the architecture of regulatory developmental programs [9], suggesting that diversity could be better explained by variation in the regulation of gene expression than the structural genes themselves [10]. Because designing new structures and robotic agents require a model able to generate novel morphologies and controlling systems, it is interesting to explore models closer to biological development, including some kind of regulatory network to drive the developmental process.

Several theoretical models and formalisms have been proposed to describe gene regulatory systems (see [32] for a review). Among them, the Boolean networks proposed by Kauffman [33] allow the simulation of large regulatory networks [32] and have been extensively used. Furthermore, a recent study [8] has demonstrated a good correspondence between Boolean networks and more realistic models based on differential equations of chemical kinetics. Similarly to Boolean networks, other network-level models focus on a statistical analysis of network properties and patterns. In order to apply realistic mutation operators in network-level models, an encoding of the network in a sequence-based genome is needed. Among such models, the Artificial Genome proposed in [58] has attracted much attention. An Artificial Genome encodes a regulatory network in a sequence of digits, making the regulatory

dynamics equivalent to a Boolean network with limitations in the possible Boolean functions carried by its nodes [84].

Similarly, a great number of theoretical models have been proposed to model biological development, creating an expanding subfield within evolutionary computation. In this emerging discipline, some models addressed the network-level for developmental regulation. Sims [72] presented a system for the evolution of physically simulated virtual creatures made of articulated rigid parts, effectors, and sensors, and controlled by an extended neural network. However, the morphology and controller were encoded separately in two recurrent directed graphs, an aspect that did not correspond to real biological development. Eggenberger [15] described a growing phenotype made of spherical modules, connected by articulated joints. A parametric regulatory network model was used, including diffusion of gene products from particular concentration source sites. The evolved forms presented only limited variability and the emergence of bilaterality. Bongard and Pfeifer [5] extended this type of model by adding a neural controller, intended to favor agents that developed directed locomotion and block pushing. The evolved agents managed to perform the assigned tasks, although, there too, with limited variability in their characteristics. Kumar and Bentley [39] proposed a computational model of development in which a regulatory network controlled the synthesis of proteins, and embryos with spherical forms were evolved. Once more, the evolved phenotypes remained rather simple. Andersen et al. [1] proposed a model of developmental cellular systems in 3D based on chemical signaling and gene regulatory networks. Their evolved embryos showed particular stable shapes and a high capacity for self-repairing. Yet again, the shapes presented by the phenotypes were too primitive, mostly rectangular or spherical. Finally, Zhan et al. [86] presented an evolutionary developmental system based on cell signaling and artificial gene regulatory networks, which was focused on the engineering challenge of electronic circuits design.

In summary, most theoretical developmental models based on genetic regulation found in the literature are not fully adequate for the morphogenetic engineering of structures and robotic agents. In this section, we describe a computational model (published in [44]) in which an artificial gene regulatory network controls the development of locomotive multicellular organisms through a fixed set of simple structural genes. An evolutionary algorithm optimizes morphologies, based on a blend of structures and controllers, which are well adapted to a basic behavior-finding problem: following a path. Our results show that, despite the simple fixed set of structural genes, the evolution of gene regulation yields a rich diversity of body plans and original behaviors.

1.3.1 The model

The model presented in this section is based on a regulated graph grammar that abstracts a developmental process in the same way as the model presented in Section 1.2. In this case, however, a gene regulatory network controls the sequence of

rules that are applied during the developmental process of the design. The main characteristics of this model, and differences with the previous model, are the following:

- The evolutionary search selects for active structures able to follow a path.
- The graphs are 2D, and their edges are oriented. This new attribute of the model will be used in the locomotion of the structures.
- The first three rules (resize *r*, duplicate *d*, and split *s*) remain unchanged, but the connection rule is omitted, while a new rule is added, the swap rule *w*, which simply reverses the direction of an edge.
- Instead of a sequence of rules, each cell (edge) contains a genome consisting of a regulatory network that controls rule firing.

The last change is the most significative from the point of view of morphogenetic engineering. The genome is now a Boolean network encoded in an Artificial Genome [58]. This artificial genome is represented by a vector of digits (Fig. 1.6a) between 0 and 3, which correspond to bases in a real DNA strand. Each gene is a sequence of four digits (for example the word 0320) preceded by a promoter (always 0101). A word placed between the promoter of a gene and the previous gene plays the role of regulatory region of the former gene. The product of a gene (a "protein") is also a sequence of four digits, obtained by increasing every digit in a gene by 1, modulo 4 (thus 1031 in the above example). Each protein regulates those genes that have a matching regulatory regions. A protein can act as an enhancer, activating the gene, or as an inhibitor, blocking its activation. Similarly to previous works, proteins ending with the base 0 are inhibitory, otherwise they behave as enhancers. While the presence of a single enhancer is enough to activate a gene, inhibition blocks enhancement.

This Artificial Genome is expressed in each cell (edge) of a developing organism as a corresponding Boolean network (Fig. 1.6b). Although all edges are governed by the same Boolean network, each one has its own expression state during development, thus allowing cell differentiation. This Boolean network controls the application of the rewriting rules: each rule is triggered by a specific node in the Boolean network (Fig. 1.6c). The genome specifies an order for the nodes, that are then mapped to the rules. During development, one edge is transformed according to a rule when the corresponding node of its Boolean network is activated. In this way, a step of development in an organism comprises the following actions: (1) updating the state of the Boolean network in each edge, and (2) transforming the edges according to their active nodes. Edge updates are done sequentially, from the oldest to the newest. Also, if several nodes are active in a Boolean network, the application of the rules is done in a sequential manner, following the ordering of the nodes.

The first node of the Boolean network is assigned a role of differentiation. When split or duplication rules are applied (division rules), one of the descendant edges will set this node to an active state, while the other resets it. The fixed mapping between nodes and rules is made starting from the second node. This asymmetry introduces a slight difference in the future expression patterns of both cells, allowing cell differentiation.

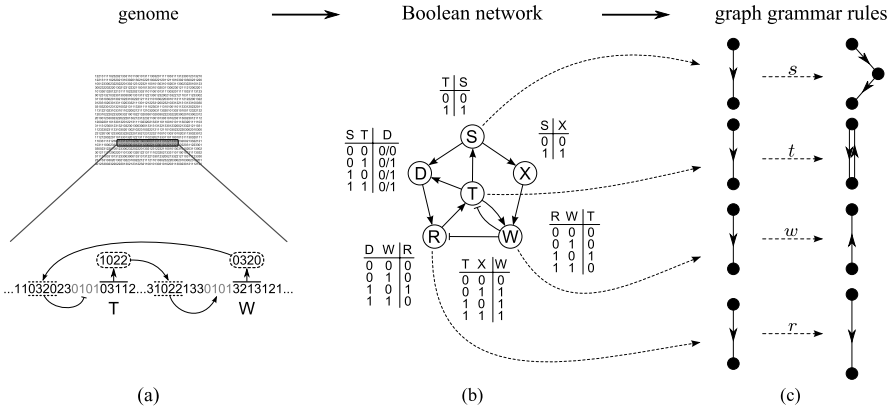


Fig. 1.6 Diagram of the morphogenetic model consisting on a derivation of a graph grammar regulated by a Boolean network encoded in a sequential genome. (a) The genome is represented by a sequence of digits. A detail of the sequence, with the regulation between genes T and W, is shown below. Gene T is an enhancer of gene W while gene W is an inhibitor of gene T. (b) Boolean network encoded in the genome. Node D is the differentiation gene. Nodes S, T, W, and R are genes mapped to split, duplication, swap, and resize rules, respectively. Finally, node X is a regular gene (type node omitted for clarity). (c) Graph grammar rules set, being *s* the split rule, *t* the duplication rule, *w* the swap rule, and *r* the resize rule.

Similarly to differentiation in biological multicellular organisms, the model includes a cellular type that determines how the edge will behave in the physical simulation: *motor edges*, *sensor edges*, and *structural edges*. This is implemented again with a special node in the Boolean network, the "type node", which determines the type of the edge. Each edge embeds a counter that stores the number of times this node has been active during development. This counter acts as a signaler that induces the differentiation of the edge. One edge becomes a sensor if it has accumulated more than three quarters of the maximum activations of an edge in the organism; it becomes a motor edge if it accumulates less than a quarter of that amount (and has been active at least once); and it becomes a structural edge otherwise. In this way, the function and the form of an organism are implemented by edges. Consequently, the model makes no distinction between the control (i.e., how the function is commanded) and the morphology of the organism.

Development starts with a graph containing a single edge (resembling the zygote of living organisms). In grammatical terms, this graph is the "axiom" of the Boolean-network-regulated graph grammar, from where the resulting graph derives after a number of productions. The nodes of its Boolean network are initially inactive, except for the first node, which initiates the dynamics of the network. The developmental process ends when one of the following conditions is verified: (1) all edges have ended their expression (i.e., all nodes of the Boolean network inside every edge are inactive), (2) the expression of an edge enters in a loop without division rules, or (3) the organism has exceeded a given number of edges (our simulations are limited to 20 edges). Finally, mimicking biological competition and cellular death

processes, only one edge is allowed between each pair of nodes. If edges of different types connect the same pair of nodes, they are deleted in this order: structural, then sensor, then motor edges.

1.3.2 Problem statement and evolutionary search

After development has completed, an organism is physically simulated in a flat world where it has to follow a path as much as possible in constant time. An organism interacts with the environment by sensing and acting: it is propelled by its motor edges, and senses the path borders with its sensor edges, in a chemotactic way. The physical simulation is similar to the model described in Section 1.3: edges are modeled as damped springs, and nodes are free movable joints that have friction with the medium. All springs and joints have the same physical parameters, and the connectivity and rest length of the springs are determined by the final developed graph. Motor edges implement an additional force $\overrightarrow{F_{t+1}}$ that pushes the edge in the direction that it defines. The magnitude of this force is proportional to the actual length of the edge, accordingly to $\overrightarrow{F_{t+1}} = \alpha L_t \overrightarrow{u}$, where $\alpha > 0$ is the motor strength parameter, L_t the length of the motor edge in the current time step, and $\overrightarrow{u}$ the unitary vector of the direction. Consequently, the whole organism moves as a result of the motor edges pushing forward in a continuous way. Edges differentiated as sensors transduce the physical world information to the organism. They have the regular forces of a damped spring, but their spring rest length l_t is dynamically upscaled accordingly to the following equation: $l_{t+1} = (r_t(\beta - 1) + 1) l_0$, where $\beta > 1$ is a continuous gain parameter that regulates the upscaling factor, $r_t \in [0,1]$ is the proportion of the edge that falls outside of the path in the current time step, and l_o is the original rest length of that edge in the graph that results from morphogenesis. In this way, if a sensor edge is completely inside of the path, its rest length equals its original rest length. On the contrary, when it falls completely outside the path, the rest length equals the original rest length scaled by the gain parameter β (thus it gets longer). Intermediate situations are linearly scaled by the portion of edge falling outside the path. Notice that a sensor edge transduces sensory information (how much it falls outside the path) to mechanical information (its rest length). This mechanical information is propagated to its adjacent edges, in the same way as muscular cells propagate a change in length to adjacent cells. Finally, structural edges are normal springs without any particular effect.

The paths used in the simulations are made of two equal curves but in opposite directions discretized by a closed polygon. Each path curve is formed by two circular arcs that form the left and right path borders respectively. A path is defined by three parameters: $\gamma \in [0,1]$ is its difficulty, ω is its width, and λ its length. The difficulty determines the sharpness of the bend, a path with $\gamma = 0$ being a straight line, and a path with $\gamma = 1$ the most twisted (Fig. 1.7). The actual curves have an angle $a = 3\pi\gamma/2$. The segments needed to build the whole path will have positive angle if the curve is to the left, and negative if it is to the right. A radius of $\lambda/2a + \omega/2$ units

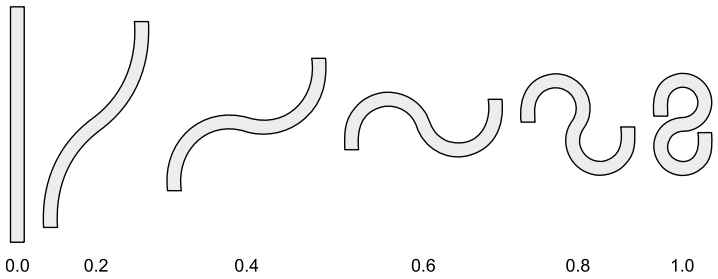


Fig. 1.7 The behavior-finding problem consists in following a path, which difficulty is given by the value of parameter γ (below).

is applied to the external border, and $\lambda/2a - \omega/2$ to the internal border. Finally, the extremes of the path are extended with a beginning and an end (straight segments of length 2ω).

The goal of the evolutionary search is to obtain efficient path-followers, i.e., organisms that, when placed at the beginning of the path, can follow it until the end. The fitness of an individual is determined by the length of path traveled in a constant simulation time. The path is divided into consecutive sections (similar to tiles) in order to quantify how well and how far an organism has moved along it. A simulation starts by developing the individual from its genomic information, and placing the resulting organism at the beginning of the path. The physics is then run for a fixed number of steps and stops if the organism reaches the path end. During the simulation, a new section of the path is labeled as "visited" if the centroid of the organism (the average position of its nodes) steps on it. In order to prevent high scores in organisms that do not interact with the environment (e.g., by starting with any trajectory that happens to fit the path), the fitness is the minimum of two runs, where in the second simulation the path is flipped along the horizontal axis.

A genetic algorithm has been implemented to evolve the structure and function of organisms. The initial population is made of 200 random organisms with short genomes (256 bases). On average, 256 bases only contain a single gene. In each generation only 25% of the population is mutated. The Artificial Genome enables the use of bio-inspired mutation operators, instead of the network-level mutations typical in other works using Boolean networks. In this way, mutations give rise to a wide variety of network-level changes, which eventually project onto the morphology [83]. Five biologically inspired sequence-level mutation operators are used:

- Single-point: a single nucleotide is replaced by another nucleotide.
- Duplication: a segment of the genome is randomly chosen and copied immediately after the original (tandem duplication).
- Transposition: a segment is deleted and copied at a random location.
- Deletion: a segment is randomly chosen and removed.
- Inversion: a segment is randomly chosen and re-written in reverse order.

In all cases, the size of the segment to be mutated was fixed at 256 bases. However, genomes could vary their lengths during evolution, as a direct consequence

of mutations. The mutant individuals obtained are added to the population, and the next generation is obtained by deterministic tournament selection with size 2. This scheme induces a low selection pressure that is compensated by elitism of one individual. The result is a good balance between exploitation and exploration, which favors the evolution of different strategies of locomotion.

1.3.3 Evolution of novel behaviors

In order to evolve a variety of path followers, the genetic algorithm was run 21 times, comprising 7 evolutionary runs for each different path ($\gamma = 0.2$, $\gamma = 0.4$, and $\gamma = 0.6$). In spite of the simple building blocks available for the organisms, 4 clearly different steering behaviors have evolved (classified by hand). For a more complete discussion on the performance of the evolutionary algorithm and the ontogeny of the organisms, see [44]. Below, we present a brief description of a representative organism for each resulting behavior:

Behavior *A*: emergence of bilateral sensors. The simplest path-follower that one could think of would include sensors in both sides to correct the direction, and a motor inbetween. This type of structure effectively evolved in about half of the evolutionary runs. Fig. 1.8a illustrates the steering behavior with a sequence of snapshots. When the organism was on the path (in gray), the forces of its two motor edges compensated, resulting in a straight motion. When one of the sides exited the path, the sensor became longer, transmitting a positional change to the motor edges. This corrected the direction of movement, pointing now to the interior of the path. This process repeated every time the organism transgressed a path border, allowing it to stay inside the path.

Behavior *B*: emergence of turning by friction. This was an interesting behavior that exploited a completely different aspect of the physics, emerging in about 10% of the evolutionary runs. The morphology of the organism integrated a more sophisticated sensory system (8 sensor edges), only one motor edge, and it showed symmetry with respect to the motion direction axis. It moved straightforward while inside the path. When the organism started exiting the path, the external skeleton of structural edges forced the sensors to reconfigure internally, and the symmetry broke down due to the elongation of some of the sensors. In this asymmetrical configuration, more nodes concentrated on the side opposite to the exit border, producing a higher overall friction on that side, which generated a bent movement towards the path. When the organism went back inside the path, it recovered its initial symmetry.

Behavior *C*: emergent spinning. Contrary to expectation, the second behavior preferred by evolution (emerging in about 30% of the evolutionary runs) had to do with spinning organisms. A combination of sensor and motor edges arranged in a sort of quadrilateral pattern favored a rotational motion. Typically, a small asymmetry was important in this type of organisms, since it allowed them to actually start moving and reach a path border, preventing endless rotation around the starting point. In this way, when a path border was transgressed for the first time, the

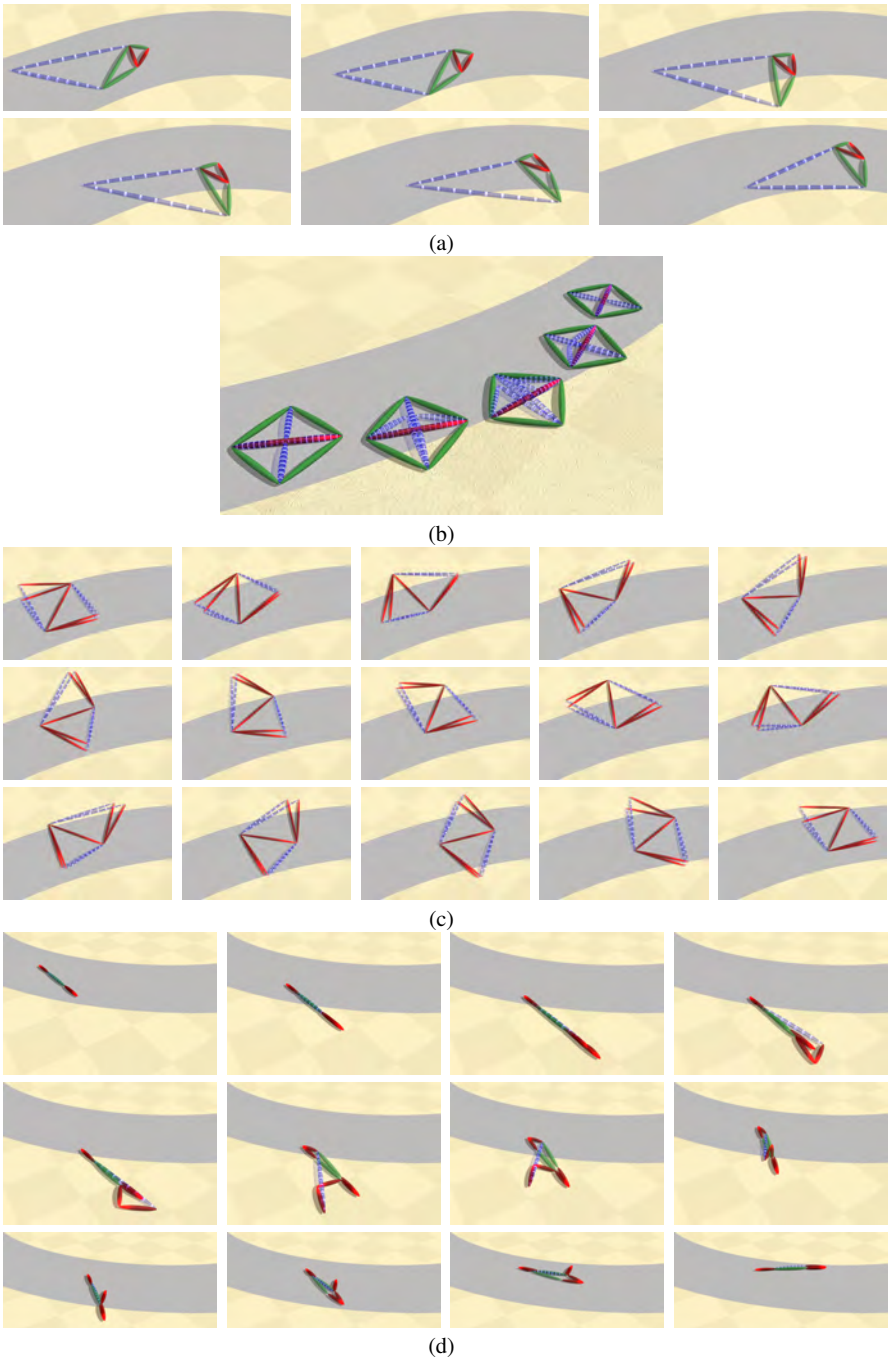


Fig. 1.8 From left to right, and from top to bottom: sequences of snapshots illustrating the steering behaviors *A*, *B*, *C* and *D*. Motor edges are in red, sensor edges in cyan (with white bands to compare relative lengths), and structural edges in green. The path is the area in gray.

organism followed it due to an iterative elongation of its sensors during the rotation, while keeping its centroid inside the path most of the time.

Organisms showing a spinning behavior described a typical cyclic trajectory along one of the path borders during the simulation. Snapshots of an organism performing a complete spin are shown in Fig. 1.8c. Notice how in snapshot 1 the exterior motor edges were aligned at roughly 45° with respect to the interior motor edges. In snapshot 3 this angle increased to about 90° due to the elongation of the sensor edges that fell outside the path. Repetitive transition between these two configurations allowed the organism to steer following the path's border.

Behavior *D*: emergent rectification. Finally, some organisms revealed a much more elaborate behavior. Remarkably, this behavior emerged from the simplest possible sensory system: one sensor edge. While the organism was inside the path, and as a result of balanced motor actions, all the edges were arranged in a single line, and the organism followed a straightforward movement. When the sensor exited the path, its elongation broke the previous configuration, initiating a long sequence of actions (the rectification) which forced the organism to go backwards, return to the path, and start another trajectory, shifted by a few degrees (around 40°) with respect to the original one.

This organism moved comparatively much slower than the others, and maneuvered in a complex way to correct the direction. The structure of the organism included two pairs of motor edges that pushed forward, and another pair that pushed backward (hence the overall slowing down). The net effect was an alignment of the edges, and a straight movement in the direction of the two pairs of motor edges. When the organism exited the path, the sensor elongated, forcing one of the leading pairs of motor edges to rotate and push backwards. In this configuration, the net movement was backwards, taking the organism back to the path. As the sensor entered the path again, it shortened, provoking the pair of motor edges to return to its original aligned arrangement. While this happened, the organism tilted to one side, correcting the original direction. Finally, the organism kept moving straightforward.

1.3.4 Advantages of network-regulated morphogenesis

Despite the simple and fixed set of structural genes implemented (one for each rewriting rule), a rich variety of body plans evolved, providing the organisms with appropriate steering strategies. The strategies obtained are diverse and complex, and successfully exploit very different aspects of the model: sensory systems adapted to the geometry of the problem (behaviors *A* and *B*), physical aspects of the environment (behavior *B*), symmetry (behaviors *B* and *C*), or complex arrangements of edges (behaviors *C* and *D*). This diversity is remarkable considering the simplicity of the resulting graphs (7 edges for behavior *A*, 17 for *B*, and 9 for *C* and *D*). Some strategies show generalization capabilities (see [44] for a discussion), being able to function in altered environments with different paths. Considering the very limited functionality of the cellular types that have been modeled, the performance and

generalization capacity of the evolved organisms result from the expressive power of the genetic model and the high degree of adaptation to the environment reached by the organisms. The fact that behavior *D* can be obtained with only 9 edges and a single sensor edge is remarkable given the efficiency and generalization capacity demonstrated by this organism.

In the field of autonomous agents, an embodiment is employed to allow agents to interact with the environment. Such embodiments have been traditionally split in morphology and controller [23, 72, 37, 11, 4, 35, 29]: the controller is typically implemented in a neural network and adjusted separately from the morphology. In contrast, in the proposed model there is no clear separation between the body and the brain that controls its behavior. A sensor is implemented as an excitable element that alters its rest length depending on its position relative to the path. In this way, sensors transduce information of the environment by introducing a change in the geometrical state of the organism. This change propagates along the organism to adapt the response in a proper way, so as to accomplish what is favored by evolution: steering to keep following the path. Furthermore, since the controller and morphology are merged in the model, both of them develop seamlessly during the same process, simplifying the model as a reliable abstraction of biological development.

1.4 Behavior-finding in a non-developmental model

In evolutionary optimization, the genomes of the individuals being optimized are mutated, and the best ranked ones are selected. However, in rugged fitness landscapes, many coordinated changes may be needed in order to transform a given individual into a better one, so the evolutionary search is likely to get trapped in local optima. Morphogenetic techniques provide the means to solve this problem: when genomes indirectly encode the structure and the behavior of the agents, in component-level specifications or instructions on how to drive a self-organization or self-assembly process, a small mutation in the genome is frequently translated as many (small or big) correlated changes in the phenotype, potentially smoothing the fitness landscape. In this way, the evolutionary search of new designs becomes more efficient than with direct genotype-phenotype encodings, as in the models examined in Sections 1.2 and 1.3.

However, instead of the genotype-phenotype mapping, a similar morphogenetic effect can also be obtained at the evolutionary scale if the mutation operators are able to bring many coordinated changes to the phenotype directly, even when the mapping is mainly direct.

In this section, we present a third and last model of morphogenetic design based on behavior-finding (published in [18]). As in the previous studies, agents are represented by mass-spring networks, and their structure is the result of an evolutionary search for agents able to perform a specified behavior. Here, however, agents do not self-assemble through a developmental process, but their morphologies are sculpted by an evolutionary optimization with a simple yet powerful mutation operator. Their

resultant behaviors are the product of a controller indirectly encoded in their morphologies and the consequent reactive interactions with the environment.

1.4.1 An agent-based model of molecular motors

The model is motivated by biological molecular motors, such as the enzymes myosin, kinesin and dynein, capable of transforming chemical energy into mechanical work. Breaking down *ATP* molecules for power, these molecular motors can effectively *walk* along cellular filaments [66]. They are composed of one or two *motor heads*, each comprising a *catalytic core* (the site where ATP molecules attach) and a *docking site* (the site where the motor attaches to the filament). Each motor head undergoes a cycle (*working cycle*) of shape changes (*conformational changes*), powered by the energy stored in ATP molecules. The docking site cyclically attaches and detaches from the filament in a coordinated way, allowing the motor to advance through the filament.

Molecular motors can be construed as nanoscale robotic agents. The control system is implicitly defined in the specific biochemical interactions between the molecular motor, the ATP molecules, and the filament; in this way, their morphologies canalize the movements and the function of the motors [82]. Indeed, molecular motors represent a clear example of morphological computation. Taking inspiration from this observation, we have built a framework based on evolutionary optimization to design robotic agents that function in a way similar to molecular motors. We call these agents molecular motor *templates*. An agent, or template, is defined as a mass-spring network, which represents the structure of a plausible protein [80]. While modeling molecular motors with mass-spring networks may seem simplistic, it can be justified theoretically: for most proteins, including many molecular motors, the dynamics is mainly dictated by their overall structure rather than their specific biochemical compositions [47].

A template has two motor heads, each one endowed with a catalytic core and a docking site. The catalytic core is defined as a set of two nodes in the network. When an ATP molecule binds to the core, it is placed exactly in the middle of the two vertices, connected by a spring to each vertex in the pair. These springs are stretched to model the change in potential energy brought by the ATP molecule (this mechanism has been used in other studies, as [80]). The docking site is modeled as a set of nodes that can attach and detach from the filament. The working cycle of a motor head can be described as a reactive finite-state machine with four states:

1. *Sticky* state: The docking site is not in contact with the filament, and the catalytic core is empty (Fig. 1.9a). This state ends when any node of the docking site touches the filament: the node becomes fixed to the filament, and an ATP molecule is bound to the catalytic core with stretched springs (Fig. 1.9b). Then, the motor head transitions to the next state.
2. *Bound* state: The stretched springs introduced in the transition to this state induce a conformational change (Fig. 1.9c), while the docking site remains firmly at-

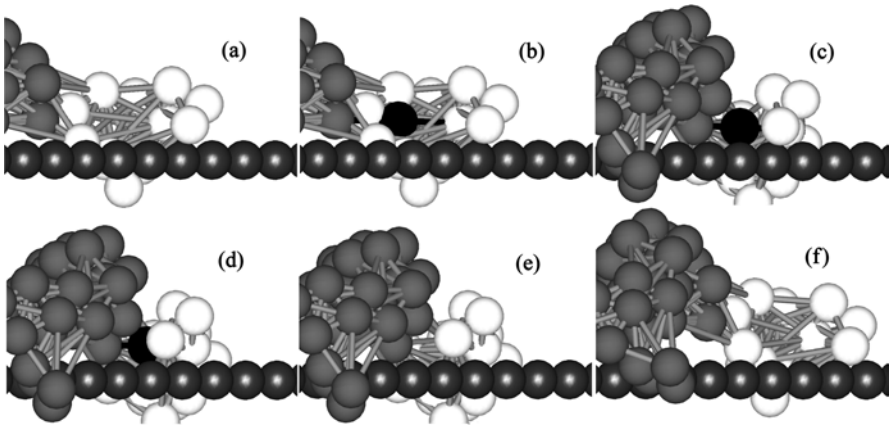


Fig. 1.9 Working cycle of a molecular motor template.

tached to the filament, resulting in a conformational change. After a fixed amount of time passes, the motor head transitions to the next state.

3. *Nonsticky* state: the nodes of the docking site detach from the filament, but remain in contact with it. If the activity of the other motor head or residual elastic forces drive the docking site out of contact with the filament (Fig. 1.9d), the ATP is expulsed from the catalytic core, deleting the associated springs (Fig. 1.9e). Then, the motor head transitions to the next state.
4. *Relaxing* state: When the catalytic core becomes empty, the absence of the associated springs triggers another conformational change. After a fixed amount of time passes, the vertices of the docking site regain the ability to get fixed to the filament, and the motor head transitions to the initial state (Fig. 1.9f), completing the cycle. A motor head has *completed* a working cycle when it has passed through all states and is back to the initial one: 1-2-3-4-1.

Simple and elegant theoretical tools that consider proteins as mass-spring networks, such as the Gaussian Network Model (GNM), use normal mode analysis to predict their structural and dynamical properties, and can do so to a surprising extent, including their unfolding pathways [77], their domain decomposition [40], and, in particular, their conformational changes and the position of their catalytic cores [85]. We use a heuristic based in GNM to determine the placement of the docking sites and catalytic cores, which are indirectly encoded in the morphology of the structure. Specifically, to define the two motor heads (each one with a catalytic core and a docking site) in the mass-spring network of a template (Fig. 1.10a), we segment the network using the normal mode associated to the third eigenvector of its Kirchhoff matrix [85]. This eigenvector assigns a vibrational amplitude to each node in the network, which can be either positive or negative. In Fig. 1.10b, each node's size and color represent the amplitude and sign, respectively (white is positive, gray is negative). Grouping neighboring nodes with same-sign vibrational amplitudes, three clusters can be defined in most mass-spring networks. There are two inter-

(a) (b) (c)

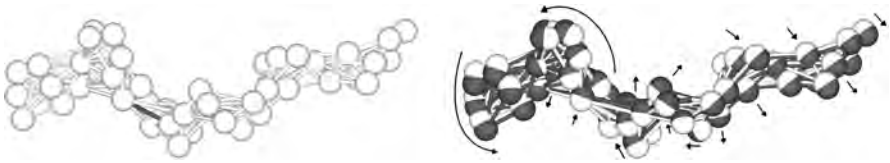
Fig. 1.10 (a) A mass-spring network is processed to determine its catalytic cores and docking sites. (b) The normal mode associated to the third eigenvector of its Kirchhoff matrix is shown. Each vertex is associated to a component of the eigenvector, whose magnitude (size) and sign (white: positive, gray: negative) conveys information about the vibration of the vertex in that normal mode, splitting the structure into three clusters. (c) The resulting molecular motor template has two motor heads, each one composed of a catalytic core (ATP and binding springs shown in black) placed between a distal cluster and the central cluster, and a docking site (white).

faces (hinges) between the clusters, such that two of the clusters are distal while the other one is central. As the third eigenvector is associated to a low-frequency normal mode, the interfaces heuristically indicate the places where the structure may bend easily in a conformational change [85]. Each catalytic core is in one interface, defined as a pair of nodes where ATP can bind (In Fig. 1.10c, the ATP and its binding springs to the nodes of the core are shown in black), one node in a distal cluster and the other in the central one. As many pairs of vertices may exist, a heuristic is applied to select one of them. Each docking site is defined as the nodes of one of the distal clusters (Fig. 1.10c, white nodes), and is associated to the catalytic core in the interface of that cluster.

1.4.2 Evolutionary search

The genotype-phenotype mapping is direct at the morphological level: the genome *is* the 3D structure. At the functional level, however, the configuration of the motor heads is indirectly encoded by the structure, as described in the previous subsection.

To start an evolutionary optimization, the agents in the initial generation are generated as randomly folded chains of 50 nodes, defining relaxed springs between all neighboring nodes. Then, agents are evaluated in the following simulation: they are placed above a straight filament (made of nodes of the same size as the nodes of the structure), such that both docking sites touch it. One of the motor heads is set in the *sticky* state, while the other is set in the beginning of the *relaxing* state. If the structure and the configuration of the motor heads is adequate, the molecule is able to "walk" along the filament, as the motor heads undergo coordinated working cycles (that is to say, their states change in a coordinated and cyclic way). After a preset amount of time passes, the simulation is stopped and the fitness is calculated



(a) (b)

Fig. 1.11 (a) A mass-spring network structure is mutated by enlarging the rest length of a spring (dark gray). (b) The resulting structure after relaxation is shown along with the original structure, in dark gray. Arrows point towards the main direction of displacement in each part of the structure.

to be the displacement of the agent's center of mass in the direction of the filament, plus the number of completed working cycles by both motor heads.

For some structures, the heuristic cannot properly define the configuration of the motor heads (docking sites and catalytic cores). In this case, they are tagged as *nonevaluable* and are not subject to selection (they are eliminated from the evolutionary competition).

After the evaluation is done, a new population of agents is generated from the previous one by preferentially selecting agents with higher fitness. Finally, the mutation operator is applied (Fig. 1.11). As the genotype-phenotype mapping is direct at the morphological level, the mutation operator must be able to bring many coordinated changes to the structure. This can be accomplished by using a physics-based mutation: as each network is a spatial configuration of vertices connected by springs in resting state (neither compressed nor stretched), a mutation consists of changing the rest length of one or several springs, each one by an independent, random amount. These perturbations introduce potential energy in the mass-spring network. If it is allowed to relax through a physics simulation, the relative positions of many vertices will change in a coordinated manner (just as originally intended) to relieve the stress. After the relaxation process, the rest lengths of the springs are set to the new distances between nodes, and springs may be added (resp. deleted) if nodes become (resp. cease to be) neighbors. In each evolutionary run, a population of 100 templates undergoes the evaluation-selection-mutation cycle for 200 generations.

1.4.3 Coevolution of form and function

The model has been tested in 30 evolutionary runs. In each run, 100 random mass-spring networks were generated to compose the corresponding initial population, 3000 in total. Almost all of them either walked a negligible distance or were nonevaluable (Fig. 1.12). However, taking as a reference the distance walked by the best individual in each evolutionary run, significantly improved individuals have evolved, too. In many cases, relatively minor modifications to the mass-spring network triggered a significant increase in the distance covered by the corresponding

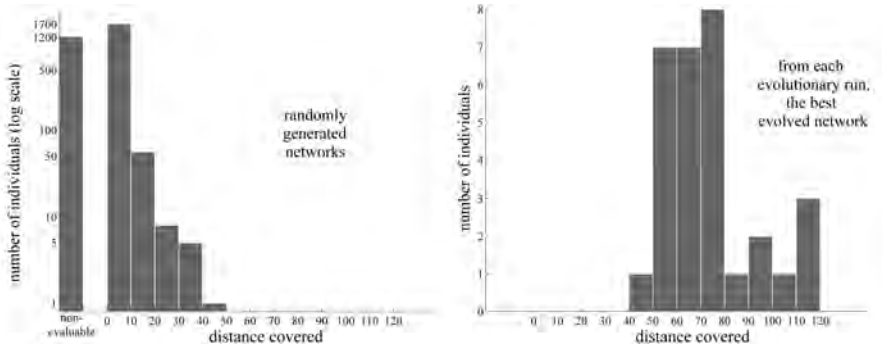


Fig. 1.12 Histograms comparing the performance of 3000 randomly generated templates and the best evolved templates in 30 evolutionary runs. In the first histogram, a significant fraction of the templates (~ 1200) were nonevaluable.

motor templates, suggesting that good templates needed to be precisely tuned to the working cycle and the details of the simulation.

Several walking strategies have evolved, associated to a diverse collection of shapes (see three examples in Fig. 1.13). In some instances, the evolutionary algorithm has produced templates using just one catalytic core and one docking site to move, as is the case for some subtypes of molecular motors in biology [42]. In our results, templates that used both motor heads did not do so symmetrically, i.e., in most instances, one motor head was responsible for the motion, while the other was used to secure the template to the filament and/or to maintain a correct orientation. Three examples are presented here:

- An example using only the leading motor head (Fig. 1.13a). After completing a single working cycle, the rear motor head lost contact with the filament for the rest of the simulation, effectively becoming useless. This specific example also rotated around the filament as it progressed.
- An example using both motor heads for conformational changes (Fig. 1.13b). The leading motor head moved forward the template, while the rear motor head attached itself to the filament, inducing a conformational change in the template, and reorienting it at each working cycle of the leading head.
- An example using both motor heads, but only one of them induced a conformational change (Fig. 1.13c). It was almost functionally equivalent to the one that used one motor head, but here the leading motor head pushed the template against the filament, leading the rear motor head to remain attached to the filament. This happened even if no effective conformational change was generated because of the geometry of the template.

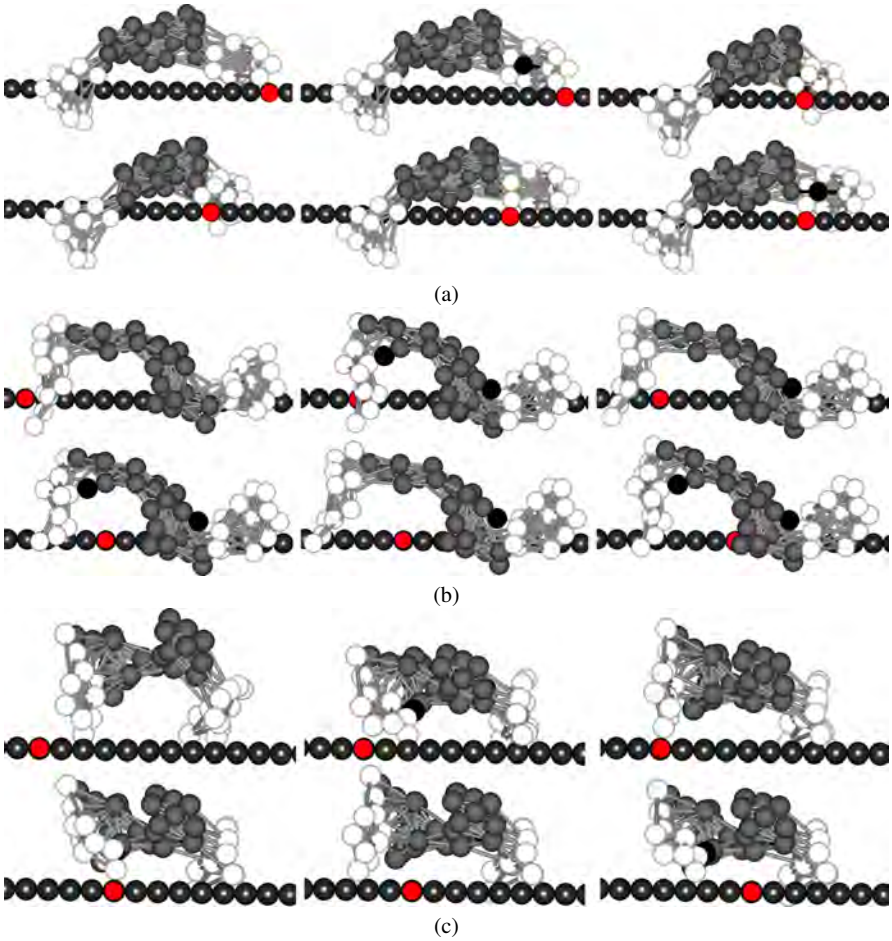


Fig. 1.13 Sequences of snapshots illustrating the gait patterns of three evolved molecular motor templates (in each case from left to right and from top to bottom). A node in the filament is marked in red to provide a point of reference.

1.4.4 A different approach to morphogenetic design

Many aspects of the model were specifically designed to be as simple as possible. The genome is minimal: it is only a fixed-width sequence of nodes in 3D space with springs between neighboring nodes, and the evolutionary algorithm is also very simple, including a single mutation operator and no crossover. Viable gait patterns could still be found in a high-dimensional space because the search was canalized in two ways:

- The working cycle (a simple reactive model) is hard-wired, and the configuration of the motor heads is indirectly encoded in the morphology of the agent.

- The mutation operator is based on physical relaxation after the application of perturbations to the structure, so it induces a fitness landscape that is more correlated to the physical characteristics of the structure, which plays a key role in the configuration of gait patterns.

However, these features of the model are relatively low-level and did not constrain in any precise way the gait patterns of the templates. Thus the diversity of shapes and gait patterns was only enabled, not determined, by these characteristics and by the fact that the individuals competed in a 3D virtual world, coevolving their morphologies and behaviors (gait patterns). Morphogenesis arose by repeated application of a complex mutation operator through evolutionary time, instead of leveraging a complex genotype-phenotype mapping. As an example of morphological computation, gaits lacked any specific control subsystem: gait patterns emerged from the interaction between the properties, the physics, and the geometry of the templates and filament.

The mutation operator can also be considered as a mode of morphological computation. Instead of using heuristics based on the analysis of the characteristics of the structures, the mutation operator only perturbed the rest length of one or more springs in the structure. The new structure was then calculated by simulating physical relaxation, which naturally induced many coordinated changes into the mutated structure.

1.5 Concluding remarks

In this chapter, we have examined three morphogenetic algorithms automatically producing designs optimized for behavior-finding problems. The various methodologies that we followed were based on genetic algorithms enhanced either by artificial development, network regulation, or dynamic mutation operators, all of which enabled the evolution of complex morphologies and behaviors. Inspired by biological organisms, the morphologies produced by this common approach were not governed by any separate controller; instead, the morphologies *were* the controllers. In this way, the structure and behavior of the agents coevolved to produce designs optimized for specific functional problems. This approach is able to generate novel and diverse designs with useful qualitative biological properties, but at a bigger computational cost than traditional evolutionary algorithms. Yet, removing the necessity of implementing an independent controller brings overall a valuable benefit from an engineering perspective, as it decreases manufacture and maintenance costs and increases robustness and reliability of the product.

The results demonstrate the versatility of our approach. Depending on the requirements of the behavioral problem and the properties of the available structural building blocks, the algorithms could produce either complex or minimalistic designs. In the examples using building blocks with limited functionality, such as the simple tensegrity structures (Section 1.2) and molecular motors (Section 1.4), the evolutionary search exploited elaborated morphologies to obtain the adequate struc-

tural and behavioral complexity in order to solve the problem. In contrast, when more elaborated building blocks were available, as in the path-follower organisms (Section 1.3), the search algorithm was able to find minimalistic designs that could produce novel and complex emergent behaviors. This adaptability of the presented morphogenetic approach also represents an attractive advantage in an engineering setting.

As engineering products become more and more complex, the potential of the traditional engineering process is bounded by the ability of engineers to fully assimilate and globally comprehend all the aspects of the product. In this context, evolutionary morphogenetic methods, such as those presented in this chapter, have the potential to become a valuable computer-aided tool for the design of new products. This approach has the capacity to solve engineering problems in innovative ways by exploiting the characteristics of the available structural and functional building blocks in truly clever designs.

References

1. Andersen, T., Newman, R., Otter, T.: Shape homeostasis in virtual embryos. Artif. Life **15**(2), 161–183 (2009)
2. Basanta, D., Miodownik, M., Baum, B.: The evolution of robust development and homeostasis in artificial organisms. PLoS Comput. Biol. **4**(3), e1000,030 (2008)
3. Bentley, P., Kumar, S.: Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. In: Proc. Genetic Evol. Comput. Conf. (GECCO), vol. 1, pp. 35–43. Morgan Kaufmann (1999)
4. Bongard, J.C., Pfeifer, R.: Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 829–836. Morgan Kaufmann (2001)
5. Bongard, J.C., Pfeifer, R.: Morpho-functional Machines: The New Species, chap. Evolving Complete Agents Using Artificial Ontogeny, pp. 237–258. Springer (2003)
6. Chavoya, A., Duthen, Y.: A cell pattern generation model based on an extended artificial regulatory network. Biosystems **94**(1-2), 95–101 (2008)
7. Coates, P., Broughton, T., Jackson, H.: Evolutionary Design by Computers, chap. Exploring three-dimensional design worlds using Lindenmayer systems and genetic programming, pp. 323–341. Morgan Kaufmann (1999)
8. Davidich, M., Bornholdt, S.: The transition from differential equations to boolean networks: A case study in simplifying a regulatory network model. J. Theor. Biol. **255**(3), 269–277 (2008)
9. Davidson, E.H.: The Regulatory Genome: Gene Regulatory Networks In Development And Evolution, 1 edn. Academic Press (2006)
10. Davidson, E.H., Erwin, D.H.: Gene regulatory networks and the evolution of animal body plans. Science **311**(5762), 796–800 (2006)
11. Dellaert, F., Beer, R.D.: A developmental model for the evolution of complete autonomous agents. In: Proc. From Animals To Animats: Internatl. Conf. Simul. Adaptive Behav. (ISAB), pp. 393–401. MIT Press (1996)
12. Devert, A., Bredeche, N., Schoenauer, M.: Robust multi-cellular developmental design. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 982–989. ACM (2007)
13. Devert, A., Bredeche, N., Schoenauer, M.: Unsupervised learning of echo state networks: A case study in artificial embryogeny. In: Proc. Internatl. Conf. Artif. Evol. (EA). Springer (2008)

14. Eggenberger, P.: Cell interactions as a control tool of developmental processes for evolutionary robotics. In: Proc. From Animals To Animats: Internatl. Conf. Simul. Adaptive Behav. (ISAB), pp. 440–448. MIT Press (1996)
15. Eggenberger, P.: Evolving morphologies of simulated 3D organisms based on differential gene expression. In: Proc. Eur. Conf. Artif. Life (ECAL), pp. 205–213. MIT Press (1997)
16. Eggenberger, P.: Genome-physics interaction as a new concept to reduce the number of genetic parameters in artificial evolution. In: Proc. IEEE Congress Evol. Comput. (CEC), pp. 191–198. IEEE-Press (2003)
17. Federici, D., Downing, K.: Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. Artif. Life **12**(3), 381–409 (2006)
18. Fernández, J.D., Vico, F.J.: Automating the search of molecular motor templates by evolutionary methods. Biosystems **106**, 82–93 (2011)
19. Fernández-Blanco, E., Dorado, J., Rabuñal, J.R., Gestal, M., Pedreira, N.: A new evolutionary computation technique for 2D morphogenesis and information processing. WSEAS T. Inf. Sci. Appl. **4**(3), 600–607 (2007)
20. Floreano, D., Keller, L.: Evolution of adaptive behaviour in robots by means of Darwinian selection. PLoS Biol. **8**(1), e1000,292 (2010)
21. de Garis, H.: Genetic programming: Artificial nervous systems, artificial embryos and embryological electronics. In: Proc. Parallel Problem Solving Nature (PPSN). Springer (1991)
22. de Garis, H.: Artificial embryology: The genetic programming of cellular differentiation. In: Proc. Workshop Artif. Life (ARTIFICIAL LIFE). Addison-Wesley (1992)
23. Gruau, F.: Advances in genetic programming, chap. Genetic micro programming of neural networks, pp. 495–518. MIT Press (1994)
24. Haddow, P.C., Hoye, J.: Achieving a simple development model for 3D shapes: are chemicals necessary? In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 1013–1020. ACM (2007)
25. Hemberg, M., O'Reilly, U.M.: Integrating generative growth and evolutionary computation for form exploration. Genet. Program. Evol. M. **8**(2), 163–186 (2007)
26. Hogeweg, P.: Evolving mechanisms of morphogenesis: on the interplay between differential adhesion and cell differentiation. J. Theor. Biol. **203**(4), 317–333 (2000)
27. Hornby, G.S.: Functional scalability through generative representations: the evolution of table designs. Environ. Plan. B **31**(4), 569–587 (2004)
28. Hornby, G.S., Lipson, H., Pollack, J.B.: Evolution of generative design systems for modular physical robots. In: Proc. IEEE Internatl. Conf. Robot. Autom. (ICRA), vol. 4, pp. 4146–4151 vol.4. IEEE-Press (2001)
29. Hornby, G.S., Lipson, H., Pollack, J.B.: Generative representations for the automated design of modular physical robots. IEEE T. Robot. Autom. **19**(4), 703–719 (2003)
30. Hornby, G.S., Pollack, J.B.: Creating high-level components with a generative representation for body-brain evolution. Artif. Life **8**(3), 223–246 (2002)
31. Joachimczak, M., Wróbel, B.: Evo-devo *in silico*: a model of a gene network regulating multicellular development in 3D space with artificial physics. In: Proc. Internatl. Conf. Simul. Synth. Living Systems (ARTIFICIAL LIFE), pp. 297–304. MIT Press (2008)
32. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. **9**(1), 67–103 (2002)
33. Kauffman, S.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**(3), 437–467 (1969)
34. Kniemeyer, O., Buck-Sorlin, G.H., Kurth, W.: A graph grammar approach to artificial life. Artif. Life **10**(4), 413+ (2004)
35. Komosinski, M., Rotaru-Varga, A.: Comparison of different genotype encodings for simulated 3D agents. Artif. Life **7**(4), 395–418 (2002)
36. Kowaliw, T., Grogono, P., Kharma, N.: The evolution of structural design through artificial embryogeny. In: IEEE Symposium on Artificial Life (IEEE-ALIFE), pp. 425–432 (2007). DOI 10.1109/ALIFE.2007.367826
37. Koza, J.R.: Gene duplication to enable genetic programming to concurrently evolve both the architecture and work-performing steps of a computer program. In: Proc. Internatl. Joint Conf. Artif. Int. (IJCAI), vol. 1, pp. 734–740. Morgan Kaufmann (1995)

38. Kumar, S., Bentley, P.J.: Implicit evolvability: An investigation into the evolvability of an embryogeny. In: Proc. Genetic Evol. Comput. Conf. (GECCO). Morgan Kaufmann (2000)
39. Kumar, S., Bentley, P.J.: Biologically inspired evolutionary development. In: Proc. Internatl. Conf. Evolvable Systems (ICES), pp. 99–106. Springer (2003)
40. Kundu, S., Sorensen, D.C., Phillips, G.N., Jr: Automatic domain decomposition of proteins by a gaussian network model. Proteins Struct. Funct. Bioinf. **57**(4), 725–733 (2004)
41. Levine, M., Tjian, R.: Transcription regulation and animal diversity. Nature **424**(6945), 147–151 (2003)
42. Lister, I., Schmitz, S., Walker, M., Trinick, J., Buss, F., Veigel, C., Kendrick-Jones, J.: A monomeric myosin VI with a large working stroke. EMBO J. **23**(8), 1729–1738 (2004)
43. Lobo, D., Hjelle, D.A., Lipson, H.: Reconfiguration algorithms for robotically manipulatable structures. In: Proc. ASME/IFToMM Internatl. Conf. Reconfigurable Mechanisms Robots (ReMAR), pp. 13–22. IEEE-Press (2009)
44. Lobo, D., Vico, F.J.: Evolution of form and function in a model of differentiated multicellular organisms with gene regulatory networks. Biosystems **102**(2-3), 112–123 (2010)
45. Lobo, D., Vico, F.J.: Evolutionary development of tensegrity structures. Biosystems **101**(3), 167–176 (2010)
46. Lobo, D., Vico, F.J., Dassow, J.: Graph grammars with string-regulated rewriting. Theor. Comput. Sci. **412**(43), 6101–6111 (2011)
47. Lu, M.: The role of shape in determining molecular motions. Biophys. J. **89**(4), 2395–2401 (2005)
48. Matsushita, K., Lungarella, M., Paul, C., Yokoi, H.: Locomoting with less computation but more morphology. In: Proc. IEEE Internatl. Conf. Robot. Autom. (ICRA), pp. 2008–2013. IEEE-Press (2005)
49. Miller, J.F.: Evolving a self-repairing, self-regulating, french flag organism. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 129–139. Springer (2004)
50. Motro, R.: Tensegrity: Structural Systems for the Future. Butterworth-Heinemann (2006)
51. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE T. Evolut. Comput. **5**(4), 349–358 (2001)
52. O'Neill, M., Swafford, J.M., McDermott, J., Byrne, J., Brabazon, A., Shotton, E., McNally, C., Hemberg, M.: Shape grammars and grammatical evolution for evolutionary design. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 1035–1042. ACM (2009)
53. Paul, C.: Morphological computation: a basis for the analysis of morphology and control requirements. Robot. Auton. Syst. **54**(8), 619–630 (2006)
54. Paul, C., Lipson, H., Valero-Cuevas, F.: Evolutionary form-finding of tensegrity structures. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 3–10. ACM (2005)
55. Pfeifer, R., Iida, F., Gomez, G.: Morphological computation for adaptive behavior and cognition. Internatl. Congress Series **1291**, 22–29 (2006)
56. Pollack, J., Lipson, H., Hornby, G., Funes, P.: Three generations of automatically designed robots. Artif. Life **7**, 215–223 (2001)
57. Prusinkiewicz, P., Lindenmayer, A.: The algorithmic beauty of plants. Springer (1990)
58. Reil, T.: Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In: Proc. Eur. Conf. Artif. Life (ECAL), pp. 457–466. Springer (1999)
59. Rieffel, J., Pollack, J.: The emergence of ontogenic scaffolding in a stochastic development environment. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 804–815. Springer (2004)
60. Rieffel, J., Pollack, J.: Crossing the fabrication gap: Evolving assembly plans to build 3D objects. In: Proc. IEEE Congress Evol. Comput. (CEC). IEEE-Press (2006)
61. Roggen, D., Federici, D.: Multi-cellular development: Is there scalability and robustness to gain? In: Proc. Parallel Problem Solving Nature (PPSN), pp. 391–400. Springer (2004)
62. Roggen, D., Federici, D., Floreano, D.: Evolutionary morphogenesis for multi-cellular systems. Genet. Program. Evol. M. **8**(1), 61–96 (2007)
63. Roggen, D., Floreano, D., Mattiussi, C.: A morphogenetic evolutionary system: Phylogenesis of the poetic circuit. In: Proc. Internatl. Conf. Evolvable Systems (ICES), pp. 153–164. Springer (2003)
64. Rosenman, M.A.: Evolutionary Algorithms in Engineering Applications, chap. The generation of form using an evolutionary approach, pp. 69–86. Springer (1997)

65. Rudolph, S., Alber, R.: An evolutionary approach to the inverse problem in rule-based design representations. In: Proc. Internatl. Conf. Artif. Int. Design (AID). Kluwer Publishers (2002)
66. Schliwa, M., Woehlke, G.: Molecular motors. Nature **422**(6933), 759–765 (2003)
67. Schnier, T., Gero, J.: Learning genetic representations as alternative to hand-coded shape grammars. In: Proc. Internatl. Conf. Artif. Int. Design (AID). Kluwer Publishers (1996)
68. Schot, S.H.: Jerk: The time rate of change of acceleration. Am. J. Phys. **46**(11), 1090–1094 (1978)
69. Shea, K., Cagan, J.: Innovative dome design: Applying geodesic patterns with shape annealing. Artif. Int. Eng. Design Anal. Manuf. **11**(5), 379–394 (1997)
70. Shea, K., Cagan, J., Fenves, S.J.: A shape annealing approach to optimal truss design with dynamic grouping of members. J. Mec. Design **119**(3), 388–394 (1997)
71. Shim, Y.S., Kim, C.H.: Generating flying creatures using body-brain co-evolution. In: Proc. Symp. Comput. Animation (SCA), pp. 276–285. Eurographics Association (2003)
72. Sims, K.: Evolving 3D morphology and behavior by competition. Artif. Life **1**(4), 353–372 (1994)
73. Spector, L., Klein, J., Feinstein, M.: Division blocks and the open-ended evolution of development, form, and behavior. In: Proc. Genetic Evol. Comput. Conf. (GECCO), pp. 316–323. ACM (2007)
74. Stanley, K., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artif. Life **9**(2), 93–130 (2003)
75. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genet. Program. Evol. M. **8**(2), 131–162 (2007)
76. Stiny, G.: Introduction to shape and shape grammars. Environ. Plan. B **7**(3), 343–351 (1980)
77. Su, J.: Protein unfolding behavior studied by elastic network model. Biophys. J. **94**(12), 4586–4596 (2008)
78. Taura, T., Nagasaka, I.: Adaptive-growth-type 3D representation for configuration design. Artif. Int. Eng. Design Anal. Manuf. **13**(3), 171–184 (1999)
79. Tibert, A., Pellegrino, S.: Review of form-finding methods for tensegrity structures. Internatl. J. Space Struct. **18**, 209–223 (2003)
80. Togashi, Y., Mikhailov, A.S.: Nonlinear relaxation dynamics in elastic networks and design principles of molecular machines. Proc. Natl. Acad. Sci. USA **104**(21), 8697–8702 (2007)
81. Trefzer, M.A., Kuyucu, T., Miller, J.F., Tyrrell, A.M.: A model for intrinsic artificial development featuring structural feedback and emergent growth. In: Proc. IEEE Congress Evol. Comput. (CEC), pp. 301–308. IEEE-Press (2009)
82. Vale, R.D., Milligan, R.A.: The way things move: Looking under the hood of molecular motor proteins. Science **288**(5463), 88–95 (2000)
83. Watson, J., Geard, N., Wiles, J.: Towards more biological mutation operators in gene regulation studies. Biosystems **76**(1-3), 239–248 (2004)
84. Willadsen, K., Wiles, J.: Dynamics of gene expression in an artificial genome. In: Proc. IEEE Congress Evol. Comput. (CEC), pp. 185–190. IEEE-Press (2003)
85. Yang, L.W.W., Bahar, I.: Coupling between catalytic site and collective dynamics: a requirement for mechanochemical activity of enzymes. Structure **13**(6), 893–904 (2005)
86. Zhan, S., Miller, J.F., Tyrrell, A.M.: An evolutionary system using development and artificial genetic regulatory networks for electronic circuit design. Biosystems **98**(3), 176–192 (2009)