Evolutionary development of tensegrity structures

Daniel Lobo*, Francisco J. Vico

Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga
Severo Ochoa 4, 29590 Málaga, Spain
Phone: +34 952137037

Abstract

Contributions from the emerging fields of molecular genetics and evo-devo (evolutionary developmental biology) are greatly benefiting the field of evolutionary computation, initiating a promise of renewal in the traditional methodology. While direct encoding has constituted a dominant paradigm, indirect ways to encode the solutions have been reported, yet little attention has been paid to the benefits of the proposed methods to real problems. In this work, we study the biological properties that emerge by means of using indirect encodings in the context of form-finding problems. A novel indirect encoding model for artificial development has been defined and applied to an engineering structural-design problem, specifically to the discovery of tensegrity structures. This model has been compared with a direct encoding scheme. While the direct encoding performs similarly well to the proposed method, indirect-based results typically outperform the direct-based results in aspects not directly linked to the nature of the problem itself, but to the emergence of properties found in biological organisms, like organicity, generalization capacity, or modularity – aspects which are highly valuable in engineering.

Keywords: Indirect encoding, Artificial development, Evolutionary computation, Tensegrity, Form-finding

Accepted Manuscript

Lobo D. and Vico F.J.
"Evolutionary development of tensegrity structures"
BioSystems 101(3), pp. 167-176, 2010.
DOI: 10.1016/j.biosystems.2010.06.005

*Corresponding author
Email addresses: dlobo@geb.uma.es (Daniel Lobo), fjvico@geb.uma.es (Francisco J. Vico)

1. Introduction

New insights in developmental biology are configuring a new computational paradigm that may dramatically change the methodology in evolutionary computing. Artificial development is a discipline that mimics the mechanisms of development, what is known as the embryological period, by means of which a group of living systems develop from one single cell into mature multicellular organisms. In search of methods to evolve phenotypes with a high structural complexity (resembling that of multicellular organisms), evolutionary algorithms have incorporated mechanisms inspired by embryology, such as gene expression, morphogenesis, and cellular differentiation (Stanley and Miikkulainen, 2003).

An important aspect in the implementation of an evolutionary algorithm is the genotype-to-phenotype encoding scheme (Komosinski and Rotaru-Varga, 2002). Traditional evolutionary algorithms use a direct encoding method (i.e., the individual is represented by a row of data that is explicitly linked to its properties). However, a direct mapping has been shown to not being effective in complex problems, due to limitations in scalability, adaptability, and evolvability (Hornby and Pollack, 2002). As a result, evolutionary algorithms with indirect encodings have been applied in a wide range of abstract problems (for a review see (Stanley and Miikkulainen, 2003)), yet the discovery of morphologies, or form-finding, that verify certain constraints is one of the most promising lines of application.

A diverse set of computational paradigms that make use of developmental mechanisms and indirect encodings has been proposed for the problem of form-finding. Cellular automata have been used as a model of development to generate simple shapes (de Garis, 1991, 1992), biological processes (e.g. gastrulation and limb budding) (Hogeweg, 2000a,b), and specific target two-dimensional patterns (Chavoya and Duthen, 2008) and three-dimensional patterns (Basanta et al., 2003, 2008). Rules that fire cell functions (as mitosis, apoptosis, or migration) have been implemented to design three-dimensional geometrical shapes (Taura and Nagasaka, 1999), tessellating tiles in a grid (Bentley and Kumar, 1999), and three-dimensional morphologies (Kumar and Bentley, 2000). Rules have been combined with gene regulatory networks that regulate their activation to evolve two-dimensional shapes (Eggenberger, 1996), two-dimensional patterns (Trefzer et al., 2009; Fernández-Blanco et al., 2007a,b), and three-dimensional multicellular organisms (Eggenberger, 1997, 2003; Joachimczak and Wróbel, 2008). Rules combined with diffusion of chemicals have also been proposed in (Haddow and Hoye, 2007) to develop simple shapes. Development models based on grammars have also been proposed, such as string grammars to develop house plants (Rosenman, 1997) and L-systems to encode plant morphologies (Prusinkiewicz and Lindenmayer, 1990; Kniemeyer et al., 2004), 3D branched organisms (Coates et al., 1999), objects made of voxels (Hornby et al., 2001; Hornby, 2004), and surfaces in 3D spaces (Hemberg and O'Reilly, 2007). Shape grammars (Stiny, 1980), a production system to generate geometric shapes, have been evolved for architectural designs (Schnier and Gero, 1996). Grammatical evolution, a method to evolve productions of a string grammar by encoding the rules of the production in a binary genome (O'Neill and Ryan, 2001), has been applied to shape grammars to evolve simple shapes (O'Neill et al., 2009). Other paradigms used in the literature are genetic programming for 2D patterns (Miller, 2004), instructions for a block builder based on turtle graphics (Rieffel and Pollack, 2004, 2006), neural networks to develop multicellular organisms (Devert et al., 2007, 2008), and functions to develop 2D images with structural motifs (Stanley, 2007). Finally, models that resemble in more detail biological processes, as proteins concentrations or cell chemical signaling, have been also proposed (Kumar and Bentley,

2003; Roggen et al., 2003; Roggen and Federici, 2004; Federici and Downing, 2006; Roggen et al., 2007; Andersen et al., 2009). While these models make use of developmental methods and indirect encodings, and constitute a great achievement in the field, they are not focused on the implementation of engineering structures; thus, their practical utility is limited.

A few researchers have applied developmental methods to the form-finding of engineering structures. Shea et al. (1997; 1997) used simulated annealing to evolve shape grammars for the automation of the design process of roof trusses and discrete structures. Rudolph and Alber (2002) proposed an evolutionary algorithm based on genetic programming to evolve node-based graph grammars that encoded structures that resemble transmission towers. Finally, Lobo et al. (2009) 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 form-finding of engineering structures.

Increased attention has been devoted to structures called tensegrity, a term coined by Buckminster Fuller (1975) to denominate structures consisting of a set of rigid elements (struts) connected by a set of tensile elements (strings). Tensegrities are characterized by their tensile integrity (Motro, 2006), a property that has been found in biological structures (Ingber, 1993) and used in multiple engineering problems (Tibert and Pellegrino, 2003). Consequently, it has been generated a natural interest in automated systems to find tensegrity structures.

Several methods have been proposed for exploring subsets of the entire search space of tensegrity structures (for a review see (Tibert and Pellegrino, 2003) and (Juan and Mirats Tur, 2008)); however, the problem of discovering new and complex tensegrity structures remains open (Rieffel et al., 2009). Bioinspired methods have been also applied to seek for tensegrity structures. Paul et al. (2005) proposed an evolutionary algorithm based on a direct encoding to evolve the connectivity pattern and parameter values of tensegrity structures with maximal volume. Rieffel et al. (2009) presented an evolutionary algorithm using a generative representation based on map L-systems, which produces large and complex irregular tensegrities with the goal of maximizing their volume. Therefore, while recent methods have been proposed to discover tensegrity structures that maximize their volume, a model that searches for complex tensegrity structures to optimize a complex engineering problem is yet to be undertaken.

In this work, we describe a novel artificial developmental model based on the formalism of regulated graph grammars, subscribed to the family of indirect encoding strategies in evolutionary computation. The performance of the proposed model has been tested in the well-known problem of form-finding. More precisely, the problem consists in designing three-dimensional tensegrity structures that perform well in a complex simulated scenario. In order to demonstrate the significance of the proposed indirect encoding, we have performed a comparison with a direct encoding method presented by Paul et al. (2005) for the generation of tensegrity structures. The comparison highlights interesting qualitative biological properties that emerge when the solutions are indirectly encoded. It is also worth noticing that such properties are highly valuable in an engineering context.

2. Developmental and evolutionary morphodynamics

We formulated the problem as to how a vehicle, configured as a mass-spring network, might land properly when falling from a given height. The physics of the simulation will be described in the next section; here we concentrate on how the structure of such a vehicle is encoded. Some indirect methods have been described in the

literature to encode networks of springs, and search for structures with particular properties, partially benefiting from the strategies that biology exploits in searching fitted structures. Below it is described a very simple model of genetic expression that allows modularity of substructures in the development of an organism. This model will be compared to a method for direct encoding in evolutionary form-finding of landing structures.

2.1. An indirect encoding scheme

Similarly to other methods for indirect encoding that have been proposed, this model has a grammatical nature. More precisely, an individual develops according to the information contained in its genome, by regulated rewriting of an initial graph under the control of a regulated graph grammar. The production system of the grammar is made of rules which 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 as genes, since they represent indivisible units of genetic expression.

Fig. 1 is a graphical interpretation of the proposed model in biological terms. If the organism is represented by a labeled graph, an edge corresponds to a cell in this scheme, and the two vertices that delimit the edge are points of adhesion to neighboring cells. As it is the case in multicellular organisms, cells can differentiate into specific cellular types by controlling its genetic expression in a particular way. Each cell stores a copy of the genome, plus additional information about what portion of it will be expressed (we will refer to this subset of genes as the domain of expression of a cell in a given time). This mechanism is implemented as a pair of indexes to the genes, indicating the domain's beginning and end. In addition to the morphological transformation that it may produce, the application of a rule always alters the domain of a cell. In each derivation step, every cell expresses simultaneously the first gene of its domain, afterwards the graph is rewritten, and the domains are updated in each cell of the resulting graph. If the domain of a cell is an empty string, then the cell has finished expressing the genome, entering a sort of stable house-keeping regime. The development of an organism stops when all cells are in such stable regime. The rules have been designed in such a way that the resulting graph is connected, and a final graph is always obtained after a derivation of a finite length (i.e., development completes in a given time, and entering infinite loops cannot occur, as it is shown below).

In order to obtain a mass-spring network, the developing graph is extended with geometric properties in a Euclidean space (each node is labeled with spatial coordinates), and edges are considered as springs (labeled with physical properties). The grammar's rules can affect these values in order to configure the morphology and functionality of the developed organism. Individuals always start development from an initial graph that is the axiom of the grammar (or the zygote), and which is made of two nodes connected by a single edge. Graph rewriting proceeds until the development stops. The developed organism is then ready to be simulated in a landing test, as it is described in the next section.

The set of possible morphologies (and consequently, possible functions) is constrained by the characteristics of the production system. The rules of the generative model are described first for a two-dimensional model (Fig. 2a). One rule to alter the length of a cell, and two different rules to implement cellular division are considered. 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 range [0.2,5], relative to the current one. If the value of the attribute is greater than one, then

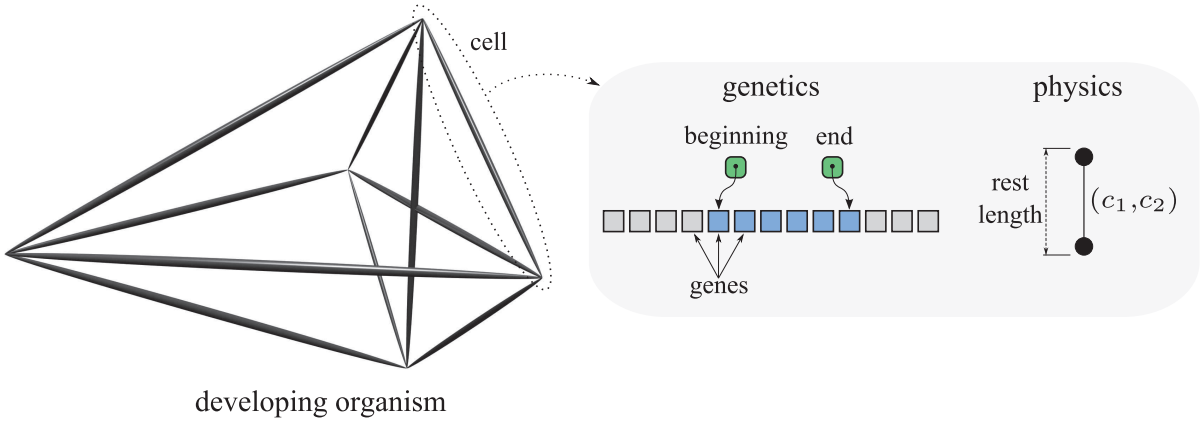


Figure 1: Diagram of a developing multicellular organism, representing the genetic (genome, and beginning and end of the cell domain) and physical (rest length, and connectivity properties c_1 and c_2) cellular information.

the rest length is increased, otherwise it decreases. 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, similarly 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 (one for each node) determine the activation state of the edges in the following way: a node will be active if at least one of 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 average over the rest lengths of the edges converging to the two nodes. The creation of these structural edges facilitates the emergence of tensile integrity by connecting nodes that are pushed later to opposite directions.

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 range [0,1]. These rules assign to the first descendant cell a domain that covers from the second gene of the current domain to an intermediate gene (as shown in Fig. 2a), being the rest of the domain 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 will be the same, and covers 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.

The described genetic expression model generates an infinite family of 2D connected graphs. Although not all connected graphs are represented with this regulated graph grammar, the diversity of forms and behaviors dis-

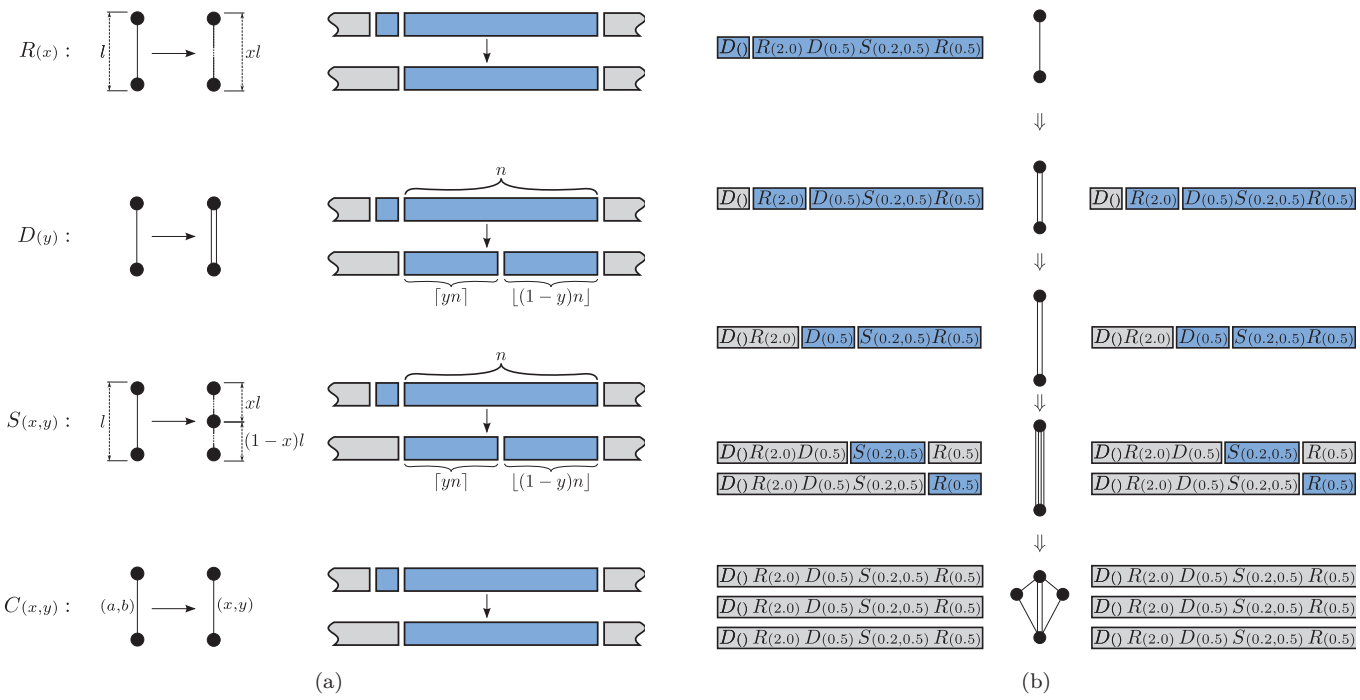


Figure 2: Production system of the regulated graph grammar. (a) Definition of the rewriting rules: transformation of an edge (left), and new domain(s) of expression (right). (b) Four derivation steps for the genome $D_{()}R_{(2.0)}D_{(0.5)}S_{(0.2,0.5)}R_{(0.5)}$.

played by the resulting springs networks is huge, and would cover the needs of most form-finding strategies. We will refer to this model as *ELSA* (a short for Expression by Limited Splitting Actions).

Fig. 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. From top to bottom, the initial graph contains a single cell with a domain that covers the entire genome. Expressing the first gene (duplicate, without parameters) results into two overlapping edges with the same domain. The second gene is then interpreted by the two edges, causing an increment of the rest length in both edges. The domains now start in the third gene, and a new duplication takes place, leaving the graph with four overlapping edges, although now the domains have been distributed between the descendant nodes. The following derivation step involves two different rules: two edges split, and the other two edges resize to half the original length. In this point development completes, since all the edges have an empty domain.

A variant of this model, *ELSA3*, incorporates necessary mechanisms for the morphologies to explore the third spatial dimension. On one hand, the spatial coordinates of the nodes are defined in a three dimensional space. On the other hand, the split rule defines a new parameter to set the inclination of the new edges in the original edge's normal plane.

The performance of *ELSA3* in form-finding will be compared to a direct encoding scheme, concretely to the model proposed in (Paul et al., 2005). This method is appropriate for comparison since it was tested in the evolutionary search of tensegrities. It represents structures as a connected graph with vertices located in a three dimensional space and edges labeled as either struts or cables. A genotype contains the initial position of the vertices, and the connectivity pattern of the struts and cables is

obtained through shuffle actions. The final form of a structure is obtained by applying a relaxation algorithm to the initial position, setting the rest length of all cables to 0, and the rest lengths of all struts to 1.

Finally, we demonstrate that the genetic expression derived from the proposed model ends for all possible genomes, since a given genome will always produce a derivation of finite length. Firstly, we see that the derivation starts with a single edge that has a finite domain of genes ($n > 0$). After expressing the first gene of the genome, we get a graph that either has one edge (if the gene corresponded to a resize or a connection rule) or two edges (duplicate or split rules). In both cases, we can see that the domains of the resulting edges contains a number of genes smaller than n. In the first case, the resulting edge contains a domain with $n-1$ genes. In the second case, either both edges contain a domain with $n-1$ edges if the duplicate or split rule did not specify a value for the domain division, or they contain x and y genes, with $x+y=n-1$, in the case that the domain apportion parameter was instantiated. From this argument it follows that, given a finite genome, the number of genes to be expressed by the cells of the organism as development progresses is strictly decreasing, which guarantees the development to stop for all possible genomes.

2.2. Physics of the model

As pointed out in the previous section, edges have been modeled as damped springs, while nodes are free movable joints. The dynamics of a spring is determined by the spring constant, the damping constant, and the rest length, being the first two global to all edges, and the last one particular of each edge. These values configure the compressive and tensile properties of the edge. The fact that an edge is stretched or compressed in a given moment during the simulation depends on the parameters and the forces applied on it by neighboring edges, and its interaction with the environment.

All the springs in the organism share the same spring and damping constants (k and c, respectively), while the rest length is settled during the development. Spring dynamics results from applying these equations to the nodes: $\vec{F}_k = -k\vec{l}$ (Hook's law), and $\vec{F}_c = -c\vec{l}$ (damping force), where $\vec{l}$ is the displacement vector of the edge (i.e., the distance and direction in which the edge is deformed). Node dynamics is also affected by the friction with the medium. This is implemented according to the expression $\vec{F}_m = -\mu_m\vec{v}$, where μ_m is the friction coefficient, and $\vec{v}$ is the velocity vector of the node. During development, springs' dynamics are evaluated, and contribute to the final shape of the structure (the phenotype). In the evaluation of the individual's fitness, apart from these forces, the simulation also includes (1) gravitation $\vec{F}_g = -g\vec{u}_z$, where g is gravity's acceleration constant, and $\vec{u}_z$ is the vector orthonormal to the Z-axis; (2) a force that is normal to the ground $\vec{F}_N$, modeled as a plane defined by the X- and Y-axis; (3) a force of the kinetic friction with the ground, modeled as $\vec{F}_f = -\mu_d \left|\vec{F}_N\right| \hat{x}$, being u_d the kinetic friction coefficient, and $\hat{x}$ the displacement unit vector of the node; and finally, (4) a coefficient of restitution ρ that is applied to the nodes after inelastic ground collisions. For the class of structures used in this work, a 4th-order Runge-Kutta integrator has been implemented to approximate solutions for the former equations of motion. Finally, collisions among edges have not been modeled. This reduces significantly the computational cost of the fitness function, while keeping a high level of realism. (Since the considered problem favors the generation of tensegrities in the organism, tensile balance in well-fitted solutions tends to avoid large excursions of the edges during landing, reducing considerably overall collisions among edges.)

2.3. Evolutionary search

ELSA3 and the direct scheme have been used as the encoding strategy in a genetic search of structures that

land properly when launched from a given height. Every individual in the population develops from its genetic information, and afterwards the freely fall on the ground is physically simulated (with a given initial speed vector). The fitness of an individual represents how well it managed landing, integrating how far it stays from a target point, and how the impact has been absorbed. The following equation combines these magnitudes in a single value:

$$f(s_k) = \frac{1}{d(c_1, c_2) \sum_{i \leq n} \left| \vec{j}_i \right|}$$

where $f(s_k)$ means the fitness value for the k-th structure of the population that has been simulated for n time steps. The first term in the denominator represents the distance traveled by the lander after the impact, measured as the Euclidean distance between the contact and the resting position. More precisely, c_1 is the point in the plane where the center of mass projects when the structure touches down for the first time, and c_2 is the center of mass projection in the plane when the structure has completed landing. On the other hand, the second term represents the impact disturbance of the structure, computed as the accumulated jerk in the sequence of points that the center of mass describes during landing, being $\vec{j}_i$ the jerk vector at time step i. The jerk (first derivative of the acceleration) behaves as a predictor for large accelerations of short duration (Schot, 1978), a magnitude that is to be minimized in vehicles which have to decelerate abruptly. As a consequence, the resulting function computes a higher fitness value for structures that smoothly slow down 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 contains genomes comprising between 2 and 8 genes, where the genes and their attribute values are randomly generated. The genomes of the individuals that form the new populations are obtained by mutating existing ones, according to four operators:

- Insertion: a new random gene is inserted in a random position.
- 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 to each operator are 0.2, 0.1, 0.1 and 0.6, for insertion, deletion, replacement, and attribute mutation, respectively. A new generation is obtained by mutating and replacing 90% of the current population. The selection operator was implemented as a roulette algorithm, with a probability of selection linearly proportional to the fitness. In order to minimize the disruption of mutations, they are more likely to occur on positions of the genome to the right side, since the genome is interpreted from left to right, and with this strategy the first stages of development tend to stay unaltered. Elitism of one individual has also been implemented.

3. Simulation results

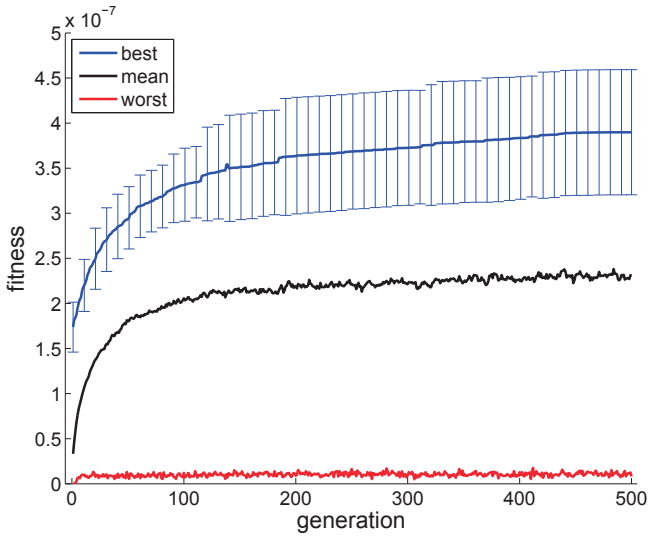
The developmental model, physical simulator, and evolutionary algorithm have been implemented in Matlab (The MathWorks, Inc.). The genetic search has been run 100 times to test the models (50 times for each encoding scheme). On average, simulation time for one run (initialization and evolution of the population) in a computer cluster of 52 CPUs (2 GHz) meant 4 hours.

Fig. 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 with error bars, representing the standard deviation of the fitness of the best-individuals. In both cases it is observed

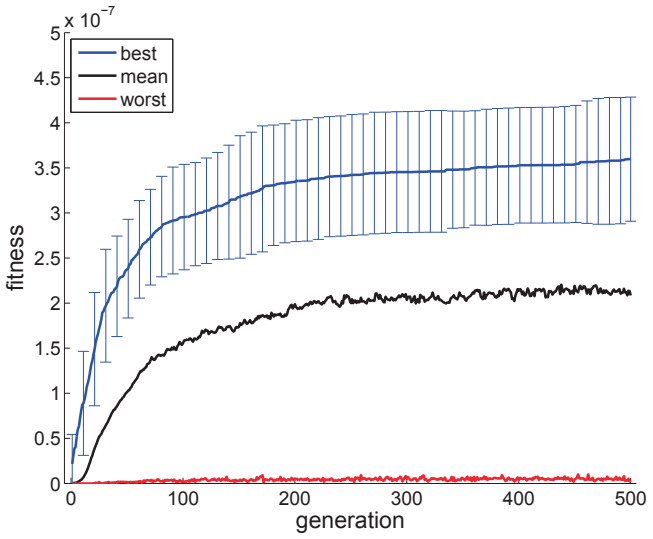
a quick fitness improvement in the first 100 generations, followed by a gradual refinement as the evolution progresses. However, the best individuals with direct encoding start with an average fitness value several times bigger than the best individuals with indirect encoding (7 times). This difference is explained by the morphological differences between early individuals in both encodings. While the random individuals coded by the direct method have between 8 and 100 edges to guarantee the diversity of the population, the individuals coded by the indirect method usually have between 1 and 10 edges, since their genomes are from 2 to 8 genes long. For this reason, the direct-encoding fitness curves generally reach slightly better values than the indirect-encoding fitness curves. The black and red curves represent the average mean fitness, and average worst fitness, respectively. In conclusion, we observe that both strategies perform equally well to the proposed landing test.

The best tensegrity structures evolved in six different evolutionary runs with the direct encoding method are presented in Fig. 4. The edges of the structures are colored according to its internal tensile state, being red edges compressed and blue edges stretched. They are characterized by its irregular shapeless organization and lack of modularity, in clear contrast to the evolved structures with the indirect encoding.

Fig. 5 shows the best tensegrity structures obtained by *ELSA3*, along with their phylogeny, in five different runs. Each row shows a selection of representative milestones in the evolution of the structure, starting from the first generation (left), and ending in the last generation (final structure, to the right). The edges of the structures are colored according to its internal tensile state: red meaning compressed edge, green for relaxed, and blue for stretched. Regularity and symmetry can be already found in some early structures, and the evolution proceeds by modifying, adding or substituting modules. For instance, structure (b) maintains its body plan from the



(a)



(b)

Figure 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.

second ancestor, and each of the four external edges is replaced as a module: first, a single edge is replaced by a flat tensegrity module (4th ancestor), then the module evolves to a three dimensional tensegrity (6th and 7th ancestors), and finally, it acquires a three dimensional tensegrity divided into three sections. Symmetries can also be found frequently in the structures, such as bilateral symmetry in structures (a), (b), and (d), and ra-

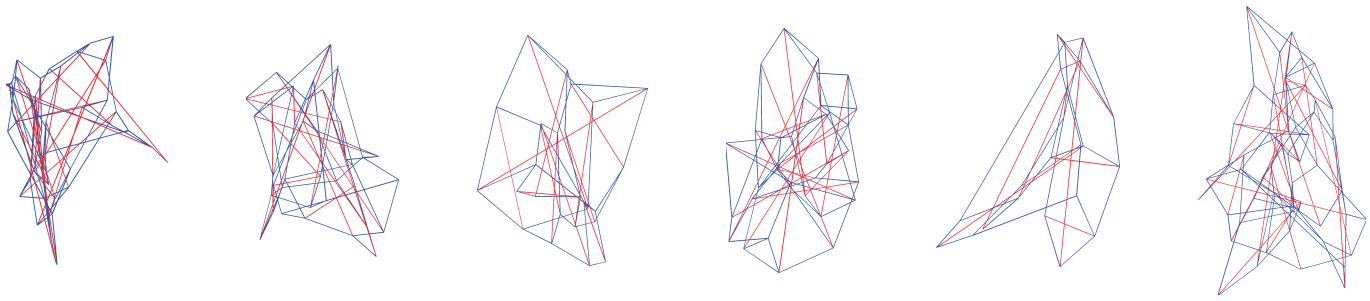


Figure 4: Best tensegrity structures evolved in six different evolutionary runs using direct encoding. Edges are colored according to its internal tensile state: compressed (red) and stretched (blue).

dial symmetry in structures (a), and (e). In general, the complexity of the structures, measured as the number of edges, increases monotonically; however, simplifications can also take place, as it is the case in structure (d), where the complex pattern shown by the third ancestor progressively evolves towards a simpler organization. In addition, a transitory simplified stage occurs in structure (a), evolving from a square pyramid (4th ancestor) to a triangular one (5th ancestor), and then going back to the square configuration, while keeping the size increase (6th ancestor). Similarly, the complex appendage in structure (c) was not always present.

Fig. 6 shows the ontogeny of the best tensegrity structures found (same runs than in Fig. 5). Each row corresponds to a representative selection of milestones during the developmental process of the structure, starting from the zygote (left) and ending in the mature phenotype (right). The edges of the structures are colored again accordingly to its internal tensile state. A zygote is made of a single edge (one cell), one unit length. Here, the complexity of the structures increases monotonically through development, as it generally does along evolution. This derives from the fact that the production system of the graph grammar does not include rules for edge deletion (cellular death, or apoptosis, has not been modeled). For instance, structure (a) first develops into a simple pyramid with a square base. Then a complexification of each of the four edges in the apex turns them into a triangular pyramid. Finally, a complex module develops in the base of the structure. Another developmental strategy that shows frequently means an enlargement of the structure by elongating the edges during the first stages of development, as it is the case in structures (b), (c), and (d), and this propagates through development. Another strategy that can be found is the modular repetition of structures, which is a consequence of reusing parts of the genome. For example, the four appendages in structure (a) develop simultaneously from the four edges in the apex of the pyramid, reusing the same genetic instructions. More precisely, two consecutive duplicate rules lacking the apportion attribute are responsible for the generation of the four edges in the apex; consequently, each lateral edge contains the same expression domain, which codes for the triangular pyramid appendage. Structure (b), makes use of a triangular bipyramid module that repeats four times. Interestingly, the size of a module depends on the seed edge that generates it, as can be seen in structure (b), where the left couple of modules are half size with respect to the right couple of modules, corresponding to the sizes of the edges that generated them. This process resembles the modularity described in biological development, where a module repeats itself with slight differences, such as fingers or limbs in animals. Finally, radial patterning does also emerge, as it is the case in structure (e). In summary, the indirect method makes use of a complicated and diverse portfolio of design strategies,

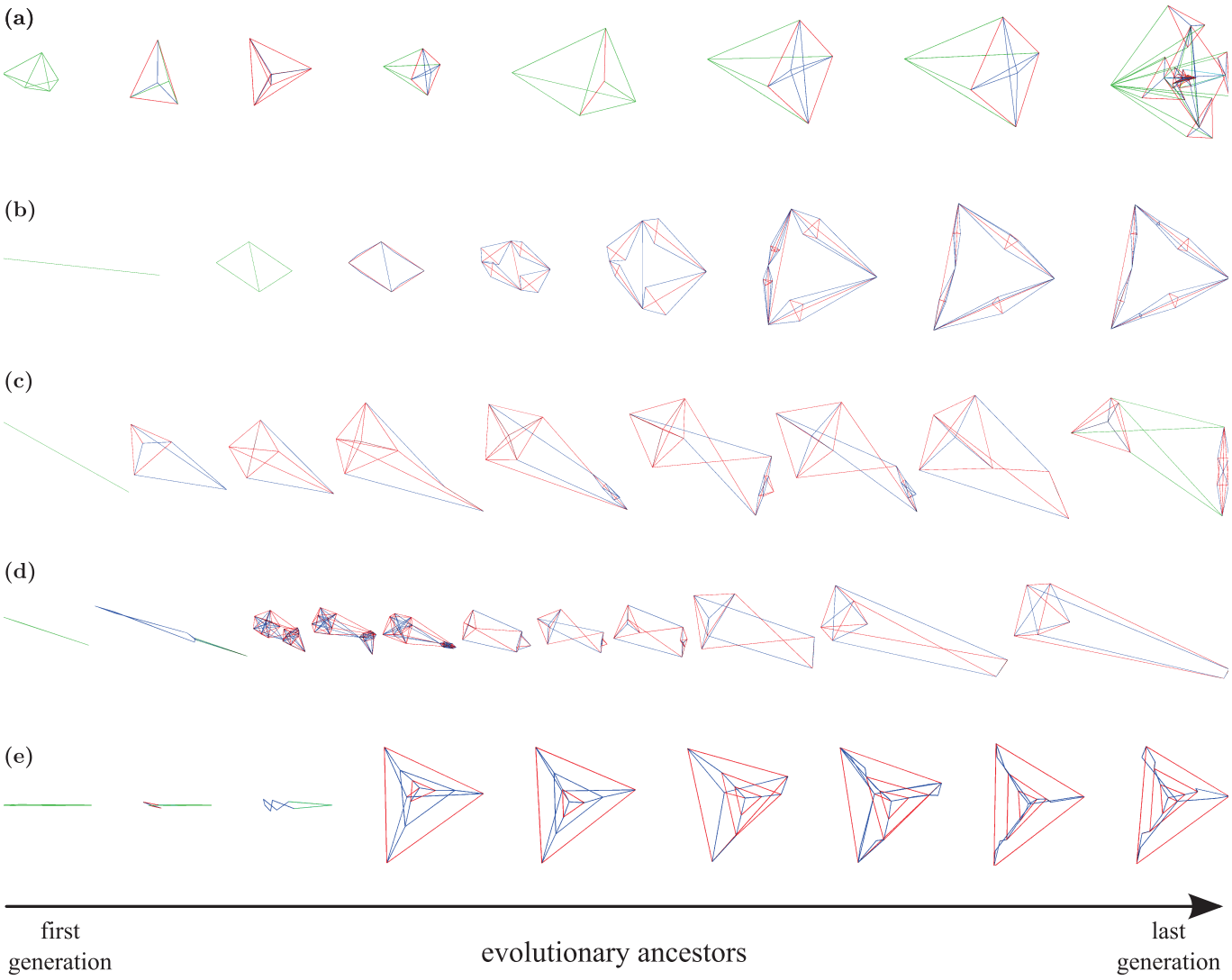


Figure 5: Phylogeny of evolved tensegrity structures using indirect encoding. Each row represents a representative selection of evolutionary steps of the best evolved tensegrity structures in five different evolutionary runs, ordered from the first generation (left) to the last generation (right). Edges are colored according to its internal tensile state: compressed (red), relaxed (green), and stretched (blue).

as compared to the direct method.

The best structures evolved with both encodings are drawn in Fig. 7 in their initial position after development (levitating structure centered in $(0, 0, 1)$), and final position after landing (structure on the ground). The deformation of the springs has been linearly mapped to the red-to-blue portion of the hue component in the *HSV* color model, to represent the compression (red), elongation (blue) or relaxation (green) state of the springs. A cross marks the projection on the ground of the center of mass when the structure touches down for the first time. A dot indicates the projection of the center of mass when the structure has landed. The distance between these two points is represented by a line on the ground. The trajectory of the center of mass during landing is shown as a black line spotted with circles that represent instantaneous jerk values at constant intervals. This value is coded with a hot color map, from white (minimum) to black (maximum), going through yellow, orange and red. In Fig. 7a it can be seen that the structure obtained

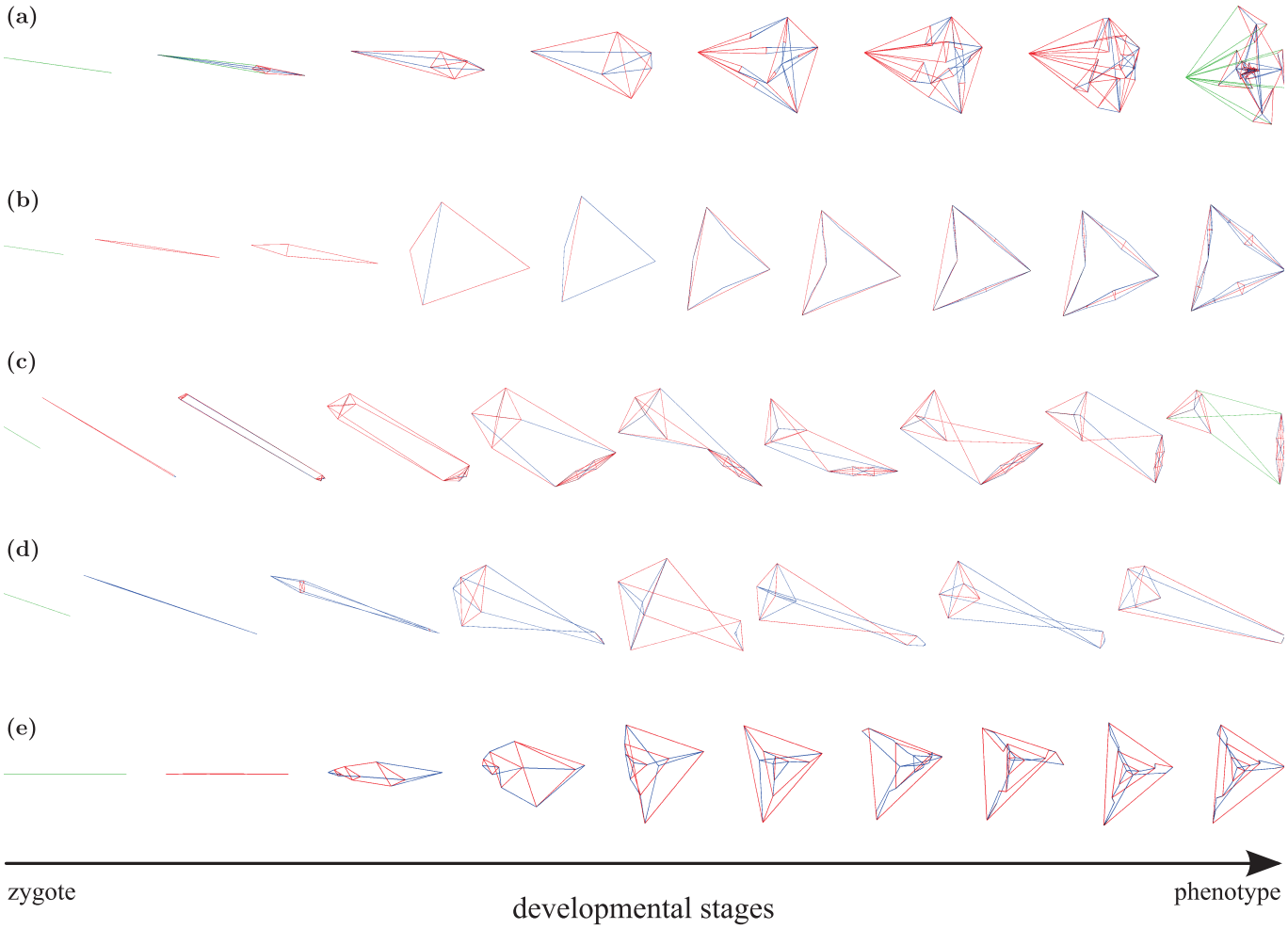
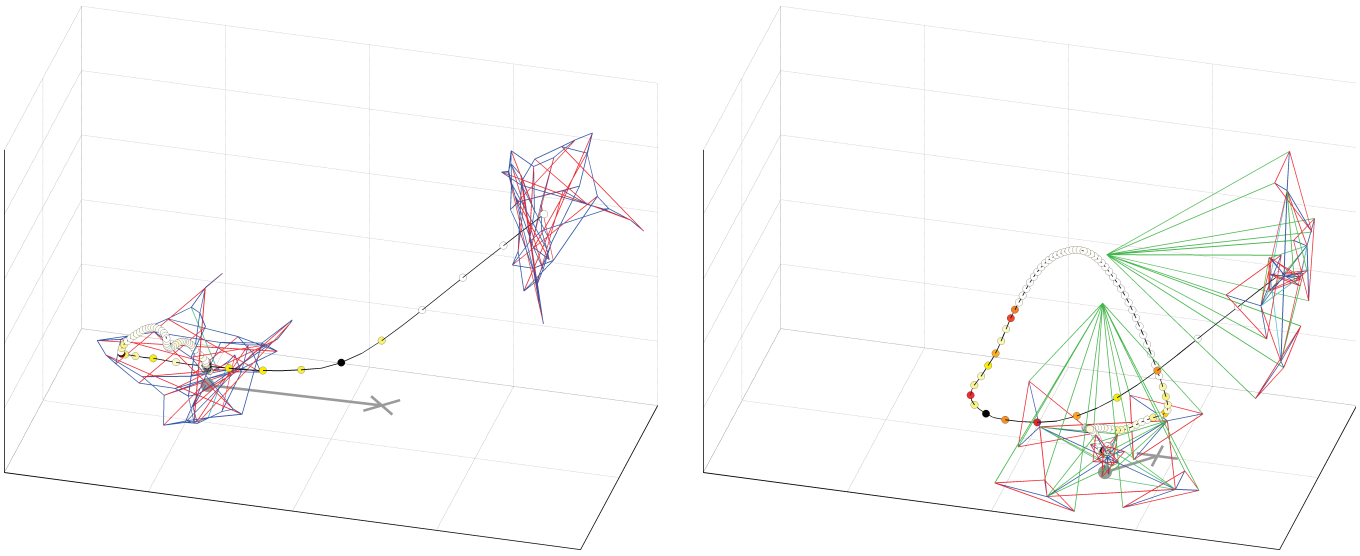


Figure 6: Ontogeny of evolved tensegrity structures using indirect encoding. Each row represents a representative selection of snapshots of the developmental process of a tensegrity structure (same structures that in Fig. 5), ordered from the zygote (left) to the final phenotype (right). Edges are colored according to its internal tensile state: compressed (red), relaxed (green), and stretched (blue).

with direct encoding adopts the form of an irregular tensegrity, which helps in the impact absorption (Fig. 8 and Movie 1). On the other hand, the particular morphology of this lander allows it to bounce in a way that keeps the position close to the impact point. Similarly, the structure evolved with indirect encoding (Fig. 7b) develops tensegrities, with some differences, since they: (1) show a regular geometry, (2) are organized modularly, and (3) are strategically distributed in the organism. This structure incorporates four lateral modules (with a tensegrity on their base) which help in stabilizing the position of the structure after the impact. Also, another module has evolved in the basement (as seen in the landed position) which seems to propel the strong rebound that brings the lander backwards (Fig. 9 and Movie 2), keeping it closer to the impact point (only 1.5 units apart, for 3.5 units of the direct method's best result).

4. Conclusions and discussion

The presented results demonstrate that both encoding strategies can find good solutions to the form-finding problem. The difference is more on their particular way to prospect candidate forms, mainly derived from the search-space that each type of encoding defines. Direct



(a) (b)

Figure 7: Best evolved tensegrity structures during the landing test. The structures in their initial and final positions in the test are shown. Edges are colored according to its internal tensile state: compressed (red), relaxed (green), and stretched (blue). The distance traveled by the structure from the contact point (cross) to the rest position (dot) is shown as a gray line. A black line represents the trajectory of the center of mass during the landing test, along with the jerk values (colored circles) at a constant interval. Jerk is represented with a hot color map: from white (minimum) to black (maximum), going through yellow, orange and red colors. (a) Best tensegrity structure with direct encoding. (b) Best tensegrity structure with indirect encoding.

encoding is based on a genome that defines the exact positions of every node in the structure. This independence of the nodes generates designs without any patterning or regularity (Fig. 4). Furthermore, mutations have a local effect on the structure, preventing the evolution of iterated modules. On the contrary, the proposed indirect encoding scheme *ELSA3*, based on regulated graph grammars, has some benefits. First, graph grammars generate structures that follow patterns and regularities due to their rewriting nature. Diverse examples have been presented in Fig. 6, such as the formation of triangular and square pyramids, bipyramids, and bilateral and rotational symmetries. Second, grammar productions are regulated by a string that is copied and transmitted during the edge duplication rules, analogously to the mechanisms of copy and transmission of the genome during the division of biological cells. This process causes the reuse of genetic information (during both, evolution and development), what allows the repetition of modules in the structures, as it has been shown in Fig 5 and Fig 6.

As a result, we have shown that the use of an indirect encoding facilitates the emergence of qualitative biological properties in the solutions, summarized in the following aspects where the direct and indirect encodings differ (summarized in Table I):

1. Regularity: while the direct encoding method explores irregular structures, *ELSA3* exploits mechanisms to obtain symmetric and modular configurations. In terms of the minimum description length principle, the landers in Fig. 4 can be described in a language of a higher level of abstraction than those in Fig. 5, like polyhedric blocks (e.g. pyramids) assembled in a particular manner.
2. Organicity: solutions evolved with the indirect en-

coding method have an organic appearance, in the sense that structural (and, possibly, functional) parts or modules can be segmented in the developed organism.

3. Generalization capacity: the regularities observed in indirectly encoded organisms are not futile; indeed symmetry permits 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, while the irregular forms showed to be highly sensitive to the initial conditions.
4. Tensegrities: both encodings can generate tensegrities (which is clearly rewarded in a landing task), but here, again, they differ in the methodology: directly encoded structures form global tensegrities, while indirectly encoded structures form simple and modular tensegrities that emerges locally.
5. Manufacturability: the properties of regularity and organicity favor industrial manufacturing of the final solution. Even that this was not the objective in this study, it is clear that the modularity of the landers in Fig.5 would make easier the independent construction and assembly of the different parts of the vehicles, allowing also serial production of components. This would not be the case with irregular forms (Fig. 4) , where the diversity of constituent parts hinders production and assembly.

Despite its simplicity, *ELSA* resembles some fundamental biological mechanisms, like cellular differentiation, patterning, symmetry, modularity, and differential gene expression. This is a consequence of the particular gene expression modeled, which proceeds by blocking genes in a similar way to the gene methylation process that takes place during embryogenesis. Concerning the evolvability, the fact that the domain is defined in an indexed fashion (i.e., it addresses gene positions) makes this model very sensitive to the insertion or deletion of genes. For example, inserting a gene at the beginning may alter completely the final result, not only because it could change the initial graph, but because the domains could be distributed differently during cell proliferation. In this sense, mutations made by the end of the genome will have a smaller effect (on average) than those that modify the beginning, since genomes are read from left to right. The performed simulations favored mutating the genome in this way, what significantly reduced the disruption.

Table I: Properties of the solutions found according to their encoding.

	direct encoding	*indirect encoding*
regularity	no	yes
organicity	no	yes
generalization	no	yes
tensegrities	global	local
manufacturability	complex	easy

In conclusion, we have presented an effective and novel method for the general indirect encoding of structures, particularly the tensegrity structures, which have demonstrated their usefulness in the evolutionary design of such structures. The results obtained qualitatively differ from those generated under a direct encoding scheme. These differences resemble properties found in biological organisms and emerge as a result of the developmental mechanism introduced in the evolution of the structures. The results demonstrate these biological properties, especially those regarding modularity, symmetry, and manufacturability, are particularly valuable in engineering in the class of form-finding problems, such as the design of tensegrity vehicles and robots (Paul et al., 2006; Graells Rovira and Mirats Tur, 2009).

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments, as well as José David

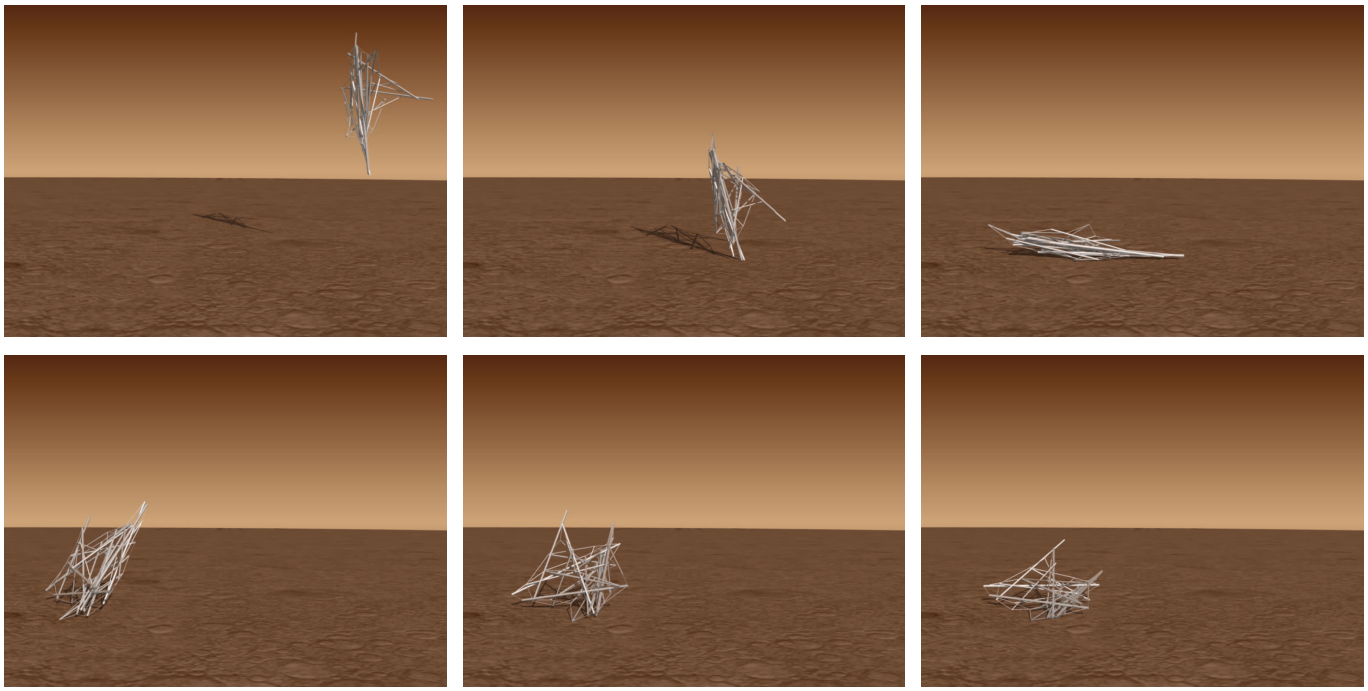


Figure 8: Landing sequence of the best tensegrity structure with direct encoding.

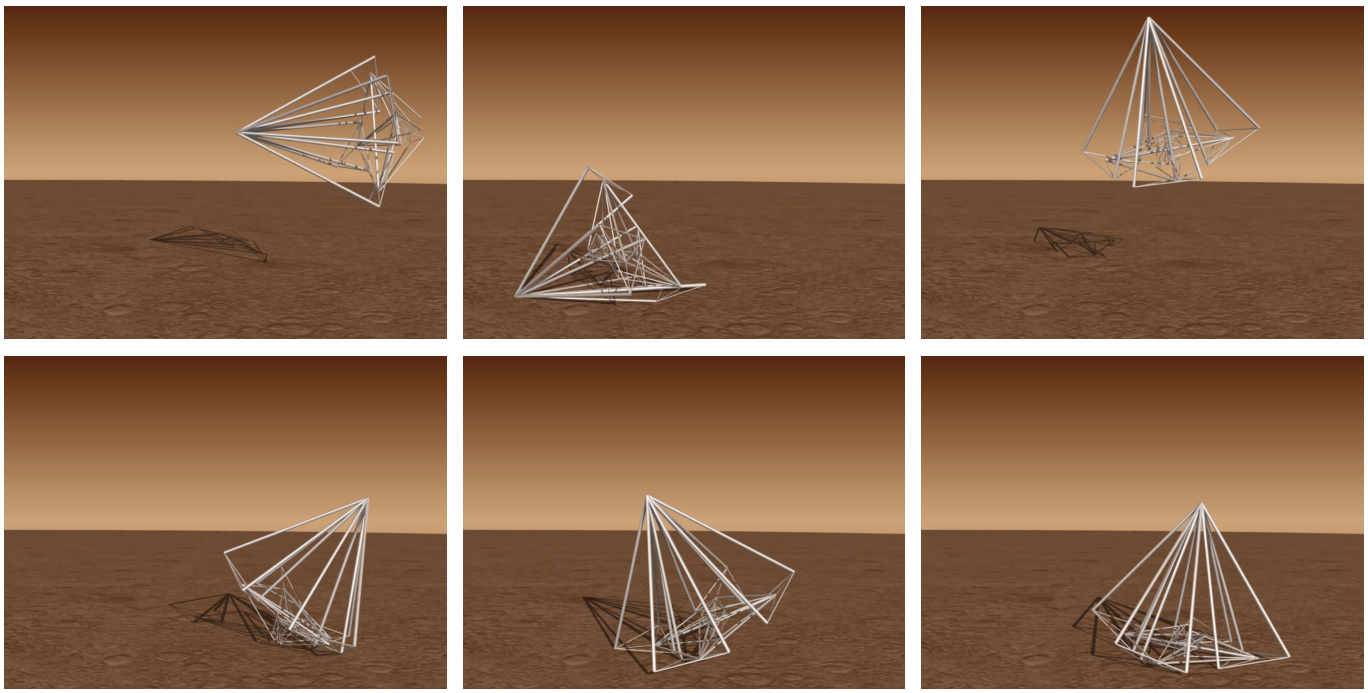


Figure 9: Landing sequence of the best tensegrity structure with indirect encoding.

Fernández for his useful suggestions.

References

Andersen, T., Newman, R., Otter, T., 2009. Shape homeostasis in virtual embryos. Artificial Life 15 (2), 161–183.

Basanta, D., Bentley, P., Miodownik, M., Holm, E., 2003. Evolving cellular automata to grow microstructures. In: Genetic Programming. pp. 77–130.

Basanta, D., Miodownik, M., Baum, B., 2008. The evolution of robust development and homeostasis in artificial organisms. PLoS Computational Biology 4 (3).

Bentley, P., Kumar, S., 1999. Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. In: Genetic and Evolutionary Computation Conference. Vol. 1. Morgan Kaufmann, pp. 35–43.

Buckminster Fuller, R., January 1975. Synergetics: Explorations in the Geometry of Thinking. Scribner.

Chavoya, A., Duthen, Y., 2008. A cell pattern generation model based on an extended artificial regulatory network. Biosystems 94 (1-2), 95–101.

Coates, P., Broughton, T., Jackson, H., 1999. Exploring three-dimensional design worlds using lindenmayer systems and genetic programming. In: Bentley, P. J. (Ed.), Evolutionary Design by Computers. Morgan Kaufmann, pp. 323–341.

de Garis, H., 1991. Genetic programming: Artificial nervous systems, artificial embryos and embryological electronics. In: Parallel Problem Solving from Nature.

de Garis, H., 1992. Artificial embryology: The genetic programming of cellular differentiation. In: Artificial Life III Workshop.

Devert, A., Bredeche, N., Schoenauer, M., 2007. Robust multi-cellular developmental design. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, New York, NY, USA, pp. 982–989.

Devert, A., Bredeche, N., Schoenauer, M., 2008. Unsupervised learning of echo state networks: A case study in artificial embryogeny. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (Eds.), Artificial Evolution. Vol. 4926 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, Ch. 24, pp. 278–290.

Eggenberger, P., 1996. Cell interactions as a control tool of developmental processes for evolutionary robotics. In: From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior. pp. 440–448.

Eggenberger, P., 1997. Evolving morphologies of simulated 3d organisms based on differential gene expression. In: the Fourth European Conference on Artificial Life. MIT Press, pp. 205–213.

Eggenberger, P., 2003. Genome-physics interaction as a new concept to reduce the number of genetic parameters in artificial evolution. In: The 2003 Confress on Evolutionary Computation. Canberra, Australia, pp. 191–198.

Federici, D., Downing, K., 2006. Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. Artificial Life 12 (3), 381–409.

Fernández-Blanco, E., Dorado, J., Rabuñal, J., Gestal, M., Pedreira, N., 2007a. A computational morphogenesis approach to simple structure development. Lecture Notes in Artificial Intelligence LNAI 4648, 825–834.

Fernández-Blanco, E., Dorado, J., Rabuñal, J. R., Gestal, M., Pedreira, N., 2007b. A new evolutionary computation technique for 2d morphogenesis and information processing. WSEAS Transactions on Information Science & Applications 4 (3), 600–607.

Graells Rovira, A., Mirats Tur, J. M., 2009. Control and simulation of a tensegrity-based mobile robot. Robotics and Autonomous Systems 57 (5), 526–535.

Haddow, P. C., Hoye, J., 2007. Achieving a simple development model for 3d shapes: are chemicals necessary? In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, New York, NY, USA, pp. 1013–1020.

Hemberg, M., O'Reilly, U. M., 2007. Integrating generative growth and evolutionary computation for form exploration. Genetic Programming and Evolvable Machines 8 (2), 163–186.

Hogeweg, P., 2000a. Evolving mechanisms of morphogenesis: on the interplay between differential adhesion and cell differentiation. Journal of Theoretical Biology 203 (4), 317–333.

Hogeweg, P., 2000b. Shapes in the shadow: evolutionary dynamics of morphogenesis. Artificial Life 6 (1), 85–101.

Hornby, G. S., 2004. Functional scalability through generative representations: the evolution of table designs. Environment and Planning B: Planning and Design 31 (4), 569–587.

Hornby, G. S., Lipson, H., Pollack, J. B., 2001. Evolution of generative design systems for modular physical robots. In: IEEE International Conference on Robotics and Automation (ICRA). Vol. 4. pp. 4146–4151 vol.4.

Hornby, G. S., Pollack, J. B., 2002. Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8 (3), 223–246.

Ingber, D., 1993. Cellular tensegrity: defining new rules of biological design that govern the cytoskeleton. Journal of Cell Science 104 (Pt 3), 613–627.

Joachimczak, M., Wróbel, B., 2008. Evo-devo *in silico*: a model of a gene network regulating multicellular development in 3D space with artificial physics. In: Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. MIT Press, Cambridge, MA, pp. 297–304.

Juan, S. H., Mirats Tur, J. M., 2008. Tensegrity frameworks: Static analysis review. Mechanism and Machine Theory 43 (7), 859–881.

Kniemeyer, O., Buck-Sorlin, G. H., Kurth, W., 2004. A graph grammar approach to artificial life. Artificial Life 10 (4), 413+.

Komosinski, M., Rotaru-Varga, A., 2002. Comparison of different genotype encodings for simulated three-dimensional agents. Artificial Life 7 (4), 395–418.

Kumar, S., Bentley, P. J., 2000. Implicit evolvability: An investigation into the evolvability of an embryogeny. In: Late Breaking Papers in The Second Genetic And Evolutionary Computation Conference (GECCO 2000).

Kumar, S., Bentley, P. J., 2003. Biologically inspired evolutionary development. In: Evolvable Systems: From Biology to Hardware. pp. 99–106.

Lobo, D., Hjelle, D. A., Lipson, H., 2009. Reconfiguration algorithms for robotically manipulatable structures. In: ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots, 2009. ReMAR 2009. pp. 13–22.

Miller, J. F., 2004. Evolving a self-repairing, self-regulating, french flag organism. In: Genetic and Evolutionary Computation, GECCO 2004. pp. 129–139.

Motro, R., 2006. Tensegrity: Structural Systems for the Future. Butterworth-Heinemann.

O'Neill, M., Ryan, C., 2001. Grammatical evolution. IEEE Transactions on Evolutionary Computation 5 (4), 349–358.

O'Neill, M., Swafford, J. M., McDermott, J., Byrne, J., Brabazon, A., Shotton, E., McNally, C., Hemberg, M., 2009. Shape grammars and grammatical evolution for evolutionary design. In: GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. ACM, Montreal, pp. 1035–1042.

Paul, C., Lipson, H., Valero-Cuevas, F., 2005. Evolutionary form-finding of tensegrity structures. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM, New York, NY, USA, pp. 3–10.

Paul, C., Valero-Cuevas, F., Lipson, H., 2006. Design and control of tensegrity robots for locomotion. IEEE Transactions on Robotics 22 (5), 944–957.

Prusinkiewicz, P., Lindenmayer, A., 1990. The algorithmic beauty of plants. Springer-Verlag, New York.

Rieffel, J., Pollack, J., 2004. The emergence of ontogenic scaffolding in a stochastic development environment. In: 2004 Genetic and Evolutionary Computation Conference, GECCO-2004. pp. 804–815.

Rieffel, J., Pollack, J., 2006. Crossing the fabrication gap: Evolving assembly plans to build 3-d objects. In: IEEE Congress on Evolutionary Computation.

Rieffel, J., Valero-Cuevas, F., Lipson, H., 2009. Automated discovery and optimization of large irregular tensegrity structures. Comput. Struct. 87 (5-6), 368–379.

Roggen, D., Federici, D., 2004. Multi-cellular development: Is there scalability and robustness to gain? pp. 391–400.

Roggen, D., Federici, D., Floreano, D., 2007. Evolutionary morphogenesis for multi-cellular systems. Genetic Programming and Evolvable Machines 8 (1), 61–96.

Roggen, D., Floreano, D., Mattiussi, C., 2003. A morphogenetic evolutionary system: Phylogenesis of the poetic circuit. In: Int. Conf. on Evolvable Systems (ICES 2003). pp. 153–164.

Rosenman, M. A., 1997. The generation of form using an evolutionary approach. Evolutionary Algorithms in Engineering Applications, 69–86.

Rudolph, S., Alber, R., 2002. An evolutionary approach to the inverse problem in rule-based design representations. In: 7th International Conference on Artificial Intelligence in Design (AID'02). Kluwer Academic Publishers, Cambridge University, Cambridge, UK.

Schnier, T., Gero, J., 1996. Learning genetic representations as alternative to hand-coded shape grammars. In: Artificial Intelligence in Design'96.

Schot, S. H., 1978. Jerk: The time rate of change of acceleration. American Journal of Physics 46 (11), 1090–1094.

Shea, K., Cagan, J., 1997. Innovative dome design: Applying geodesic patterns with shape annealing. Artificial intelligence for engineering design, analysis and manufacturing 11 (5), 379–394.

Shea, K., Cagan, J., Fenves, S. J., 1997. A shape annealing approach to optimal truss design with dynamic grouping of members. Journal of Mechanical Design 119 (3), 388–394.

Stanley, K., Miikkulainen, R., 2003. A taxonomy for artificial embryogeny. Artificial Life 9 (2), 93–130.

Stanley, K. O., 2007. Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines 8 (2), 131–162.

Stiny, G., 1980. Introduction to shape and shape grammars. Environment and Planning B 7 (3), 343–351.

Taura, T., Nagasaka, I., 1999. Adaptive-growth-type 3d representation for configuration design. Artificial intelligence for engineering design, analysis and manufacturing 13 (3), 171–184.

Tibert, A., Pellegrino, S., 2003. Review of form-finding methods for tensegrity structures. International Journal of Space Structures 18, 209–223.

Trefzer, M. A., Kuyucu, T., Miller, J. F., Tyrrell, A. M., 2009. A model for intrinsic artificial development featuring structural feedback and emergent growth. In: CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation. IEEE Press, Piscataway, NJ, USA, pp. 301–308.