Computers and Structures 87 (2009) 368–379

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Automated discovery and optimization of large irregular tensegrity structures

John Rieffel *, Francisco Valero-Cuevas, Hod Lipson

Department of Mechanical and Aerospace Engineering, Cornell University, 138 Upson Hall, Ithaca, NY 14853, United States

ARTICLE INFO

Article history:
Received 26 November 2007
Accepted 20 November 2008
Available online 4 January 2009

Keywords:
Tensegrity
Smart structures
Genetic algorithm
Map L-system

ABSTRACT

Tensegrities consist of disjoint struts connected by tensile strings which maintain shape due to pre-stress stability. Because of their rigidity, foldability and deployability, tensegrities are becoming increasingly popular in engineering. Unfortunately few effective analytical methods for discovering tensegrity geometries exist. We introduce an evolutionary algorithm which produces large tensegrity structures, and demonstrate its efficacy and scalability relative to previous methods. A generative representation allows the discovery of underlying structural patterns. These techniques have produced the largest and most complex irregular tensegrities known in the field, paving the way toward novel solutions ranging from space antennas to soft robotics.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

The word *tensegrity*, a concatenation of *tensile integrity* was coined by Buckminster Fuller to describe structures popularized by the sculptor Kenneth Snelson in 1948 [1]. A tensegrity structure is a self-supporting structure consisting of a set of disjoint rigid elements (struts) whose endpoints are connected by a set of continuous tensile elements (strings), and which maintains its shape due to the self-stressed equilibrium imposed by compression of struts and tension of strings [2], as illustrated by Fig. 1. Such structures are *pre-stress stable* in the sense that in equilibrium, each element is under compression and each tensile element is under tension. As such, the structure has a tendency to return to its stable configuration when subjected to any temporary perturbation [3].

These principles of tensegrity can be found in a variety of man-made and natural objects, ranging from free-standing camping tents and geodesic domes to the cellular cytoskeleton and even the structure of proteins [4]. In engineering and architecture, tensegrities are popular because of their efficient combination of simplicity with rigidity, and their ability to collapse and re-deploy by changing string tension.This mutability has led to increased attention for space applications [5,6] and robotics [7].

With this increased attention comes increased interest in new tensegrity structures, and a corresponding interest in methods of discovering them. While many regular tensegrity towers can be created either compositionally, by stacking smaller tensegrities atop each other, as in Skelton et al.'s work [5], or by intertwining spiraling frustum modules, as in Murakami et al.'s work [8], the broader problem of discovering new and increasingly complex tensegrity structures, particularly irregular ones, remains open. While several methods exist for exploring subsets of the entire search space, no purely analytical method exists for producing novel, stable tensegrity structures.

This problem of *form finding* is far from trivial. The first aspect of the problem lies in determining a set of connections struts and tensile elements which is capable of structural stability. Once given such a set of connections, a second problem lies in determining the geometric and spatial properties of the structure, such as the exact position of each strut and the exact rest length of each string. Tibert [9] and Juan and Tur [10] both provide comprehensive reviews of form finding. Our focus in this work is primarily in the former aspect of form finding: namely, developing a novel means of describing and generating the topologies of large, irregular tensegrities.

Originally, Fuller and Snelson both used heuristics and interactive methods to produce the earliest tensegrities[11]. More advanced mathematical approaches exist, such as those using group symmetries [3], but are often limited to *regular* tensegrities, in which all struts share the same fixed length, and all tensile elements share a single, or small set of, fixed rest lengths. Other mathematical approaches, such as those employing force density, are not limited to regular structures, but tend to suffer when scaling to large, irregular structures [11]. More recently, Micheletti et al. have used a marching procedure for form finding of irregular tensegrities, albeit for a given topology [12] and Zhang et al. have used dynamic relaxation mixed with kinetic damping to discover more irregularly shaped tensegrities [13], but relied on an interactive and compositional approach for free-form tensegrities, adding a new element and connecting strings to an already stable tensegrity. Therefore the need remains for an automated, open-ended method capable of *scalably* discovering large and increasingly

* Corresponding author.
E-mail address: jrieffel@gmail.com (J. Rieffel).

0045-7949/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2008.11.010

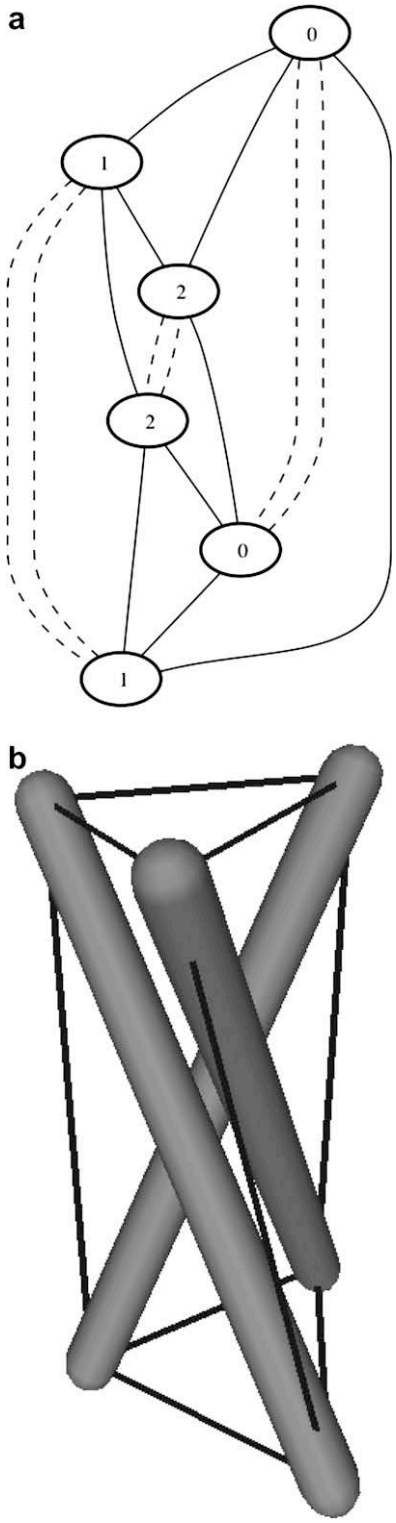

Fig. 1. A 3-Strut tensegrity (left), and its corresponding graph. Each vertex in the graph is the endpoint of a rigid element, and the solid edges are tensile elements. Rigid elements are represented as a double dashed line connecting nodes with matching numbers. In all future graphs, dashed lines will be omitted for clarity.

complex irregular tensegrity structures. Given the scope and nature of the problem – a large and loosely structured search space – tensegrity form finding lends itself to evolutionary search. In earlier work, colleagues of ours evolved 6, 8, 10 and 12-strut tensegrities using a direct encoding method [14]. Each tensegrity's genotype was represented by a linear string containing, in sequence, the location of each vertex, and a series of vertices between which to "swap" strings and struts. While they were able to produce several interesting structures, they noted issues with the scalability and evolvability of the system. In particular, the direct encoding grows linearly as $(14s + 8c)$ where s is the number of struts and c the number of cables. Furthermore, these efforts treated the emergent tensegrities as purely mathematical entities, and made no effort to eliminate inter-element collisions within the structure.

In this paper we describe an alternative method of representing and generating tensegrity structures, one that employs a *developmental representation* – in particular a map L-system – to grammatically "grow" a graph representing tensegrity structure. As such, issues of scalability, both in terms of representation and of performance, are addressed. In addition, by adding a layer of refinement to the system we can produce collision-free structures. Finally we utilize rapid prototyping technologies in order to easily print, in 3-D, models of the evolved tensegrities in order to validate the designs, thereby skipping what would otherwise be an extremely complicated process of assembly.

2. Automated discovery of tensegrities

Given the lack of analytical methods of describing large irregular tensegrities, as well as an incredibly large space of possible structures to explore, we follow the lead of [14] in using an evolutionary search technique. The advantage of automated evolutionary approaches lies in rapidly exploring a large design space while limiting the role of human bias – as a result, these methods can lead to products which possess unanticipated novelty, acting as an "automated invention machine" [15] capable equaling or even surpassing human designs of products such as sorting networks [16], photonic crystals [17], optical lens systems [18] and quantum Fourier transforms [19]. Essential to this ability to operate orthogonal to human intuition is minimizing the bias of human-generated domain knowledge required to produce an answer – thus allowing for the emergent discovery of broader underlying principles.

2.1. Tensegrities as graphs

Central to our approach is the use of graphs as a means of representing tensegrity structures. In his book on the subject, Motro illustrates several tensegrities in this manner [11]. This is also the basis of de Guzman's form finding methodology [20]. Specifically, a tensegrity can be represented as a graph as follows: each vertex corresponds to the endpoint of a strut and each edge corresponds to a tensile connection between endpoints. Fig. 1 shows a tensegrity and its corresponding graph.

With this representation in hand, we can proceed to address the issue of representational scalability by introducing a grammar-based developmental process to incrementally grow graphs representing tensegrities.

2.2. A map L-system to grow planar graphs

Inspired by the biological processes of growth and development, grammar-based representations, such as Lindenmeyer Systems (L-Systems), are often used to grow large complex objects from a simple set of rules. Map L-systems are a special type of L-System which, instead of operating on strings of characters, operates on the edges and nodes of graphs [21]. As such, they can be used for more geometric purposes than string L-Systems. Hemberg and O'Reilly, for instance, use a map L-system to generate architec-

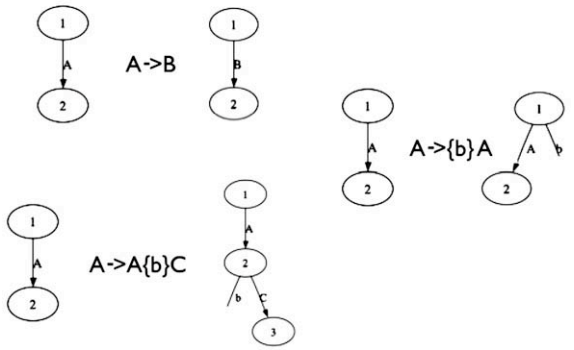

Fig. 2. Examples of map L-system rules.

tural surfaces [22], Schmidt et al. used a graph grammar to generate mechanical structures [23], and Shea and Cagan have utilized grammars to describe three-dimensional geodesic domes, trusses, and other discrete structures [24,25]. More broadly, Kaveh has developed the use of topological transformations and expansions to represent structures [26–28].

In our case, we can generate a set of rules which, when iteratively re-applied to a starting graph, gradually grow the graph.

The particular rules we use for this purpose are as follows:

- *Relabel*: $E_X \rightarrow E_Y$ – replace edges labeled X with label Y.
- *Pre-Branch Stub*: $E_X \rightarrow \{z\}E_Y$ – For each edge labeled X, source node sprouts a stub with label z and edge is relabeled Y.
- *Post-Branch Stub*: $E_X \rightarrow E_Y\{z\}$ – For each edge labeled X, destination node sprouts a stub with label z and edge is relabeled Y.
- *Split Edge*: $E_X \rightarrow E_W\{z\}E_Y$ – For each edge labeled X, edge splits into edges labeled W and Y joined by a node with stub z.

These rules are most easily demonstrated with Fig. 2.

In order to grow a graph with a desired number of nodes, the progression of the map L-system is as follows: in the first stage,

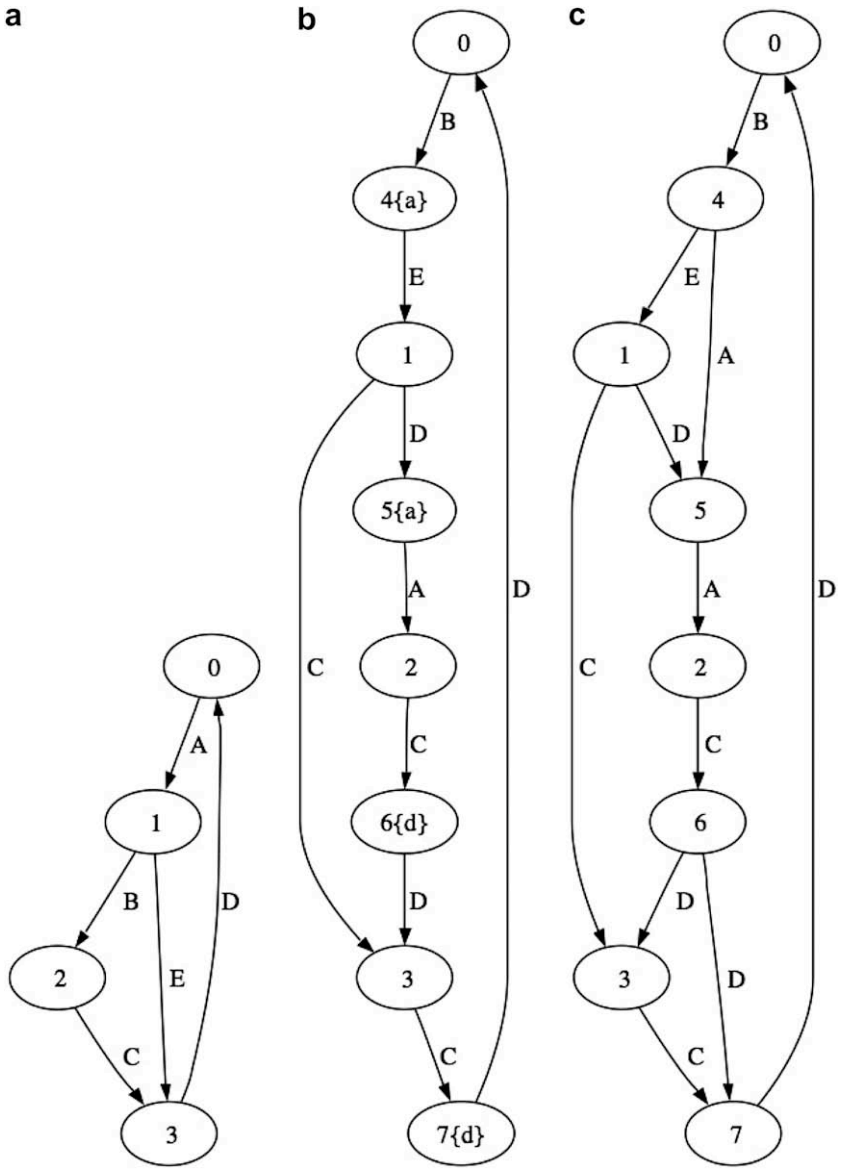

Fig. 3. The growth of a planar graph. Between steps 1 and 2, rules are applied to each edge, adding nodes and relabeling edges. For instance, the rule $A \rightarrow B\{a\}E$ applied to the edge between nodes 0 and 1 creates a new node, 4, with a stub a. In the subsequent stage, matching stubs are found and become new edges. Note that the a stubs on nodes 4 and 5 are matched, creating a new edge with label A.

rules are applied to each edge in turn, relabeling and creating new nodes and stubs accordingly. In the second stage, nodes with matching stubs which share a common face are joined with a new edge. In the event that multiple stubs exist along a face, they are matched using a simple greedy algorithm. These steps are repeated until the desired number of three-connected nodes are created. As a final step, the graph is simplified by removing redundant edges and nodes of degree less than three.

Fig. 3, for instance, shows the progression the following grammar on the initial graph:

$A \rightarrow B\{a\}E$
$B \rightarrow D\{a\}A$
$C \rightarrow D$
$D \rightarrow C\{d\}D$
$E \rightarrow C$

2.3. From planar graph to tensegrity

The map L-system described above is sufficient for generating 3-connected graphs, but subsequent computation is necessary before a generated graph can correspond to a candidate tensegrity structure – namely, the vertices must be paired into sets representing end points of structural elements. Returning to our graph in Fig. 1, the grammatical system described above will generate the subgraph composed of vertices and solid lines, but not the *dashed lines* corresponding to element pairs.

In our case, a candidate pairing of vertices is computed by using a greedy depth-first algorithm which matches two pairs provided they are not already joined either by an edge or by an existing rigid element. Note that this greedy algorithm will find a candidate pairing of nodes, but will not necessarily find an optimal pairing: each developed graph often contains multiple possible different pairings. Finding *which* particular pairing produces the best tensegrity is the domain of the evolutionary algorithm which we describe below.

Table 1
Left: A single-point mutation of a map L-system production Rule. *Right:* Crossover between two parents. Relevant changes are highlighted in bold face.

Parent	Child	Parent 1	Parent 2	Child
$A \rightarrow B\{a\}E$	$A \rightarrow B\{a\}E$	$\mathbf{A \rightarrow B\{a\}E}$	$A \rightarrow C\{b\}C$	$\mathbf{A \rightarrow B\{a\}E}$
$B \rightarrow D\{a\}A$	$B \rightarrow D\{a\}A$	$\mathbf{B \rightarrow D\{a\}A}$	$B \rightarrow D$	$\mathbf{B \rightarrow D\{a\}A}$
$\mathbf{C \rightarrow D}$	$\mathbf{C \rightarrow A\{c\}A}$	$C \rightarrow D$	$\mathbf{C \rightarrow A\{b\}A}$	$\mathbf{C \rightarrow A\{b\}A}$
$D \rightarrow C\{d\}D$	$D \rightarrow C\{d\}D$	$D \rightarrow C\{d\}D$	$\mathbf{D \rightarrow E}$	$\mathbf{D \rightarrow E}$
$E \rightarrow C$	$E \rightarrow C$	$E \rightarrow C$	$\mathbf{E \rightarrow B\{c\}A}$	$\mathbf{E \rightarrow B\{c\}A}$
Mutation		Crossover		

2.4. Evolving tensegrities

The map L-system described in Section 2.2 and the vertex-matching method described in Section 2.3 provide nearly all of the pieces needed to perform an evolutionary search for large irregular tensegrity structures.

2.4.1. Primary and secondary mutations

As a developmental encoding, map L-systems offer the ability produce large tensegrity graphs using only a seed structure and a few grammatical rules. As Hornby demonstrated in his work on evolving robots and tables, the nature of L-systems allows for symmetric growth, and large-scale co-ordinated changes in the final graph can be achieved with a single small change to the grammar [29]. We refer to this type of wholesale mutation of an L-system production rule as a *primary* mutation. Mutation and crossover of a map L-system is shown in Table 1.

However, not every graph generated with a map L-system corresponds to an optimally form-filling structure, and sometimes, as noted in Paul et al.'s work [14], smaller mutations to a tensegrity, such as re-arranging two strings, or interchanging two vertex pairs corresponding to strut endpoints, may be necessary to more fully

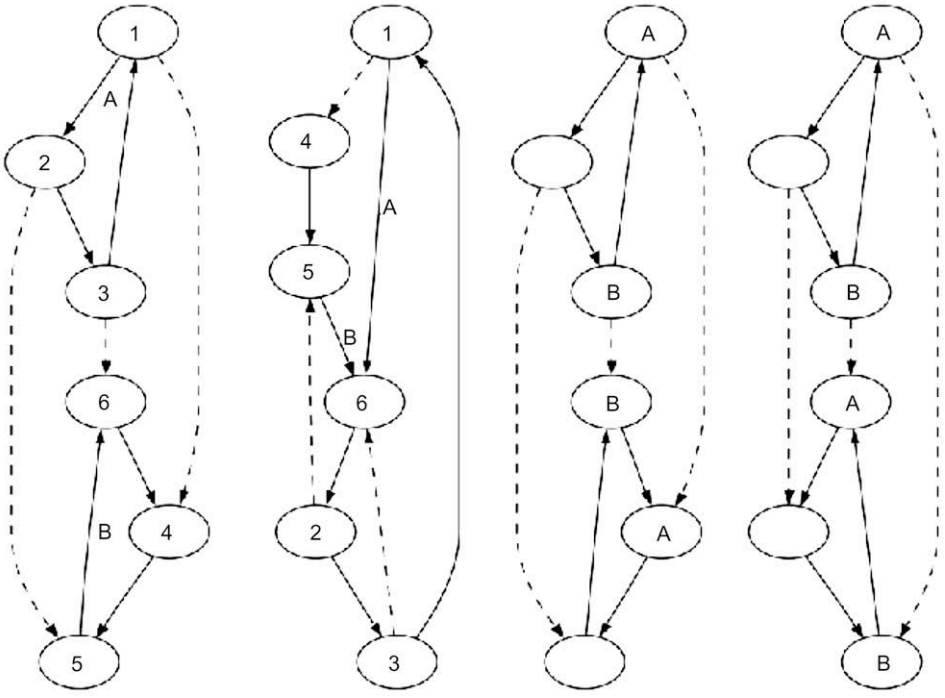

Fig. 4. Left: A *secondary mutation* in which the endpoints of two strings (solid edges marked A and B) are swapped. Right: A *secondary mutation* in which the endpoints of two elements (dotted lines) are swapped.

and smoothly explore the space of structures. Furthermore, no aspect of the grammar described above specifies which of the nodes are to be matched to produce tensegrity elements – this is an aspect of the search beyond the scope of the grammar alone.

Therefore, to allow for smaller localized changes to a tensegrity we also allow two *secondary* mutations. The first picks two graph edges at random and swaps their destination nodes, as demonstrated by the Fig. 4. The second acts similarly, but operates on strut endpoints, as shown in the right hand side of Fig. 4.

2.4.2. Solving tensegrity structures

Each generated tensegrity was then reproduced within the Open Dynamic Engine (ODE) [30], the widely used open-source physics engine which provides high-performance simulations of

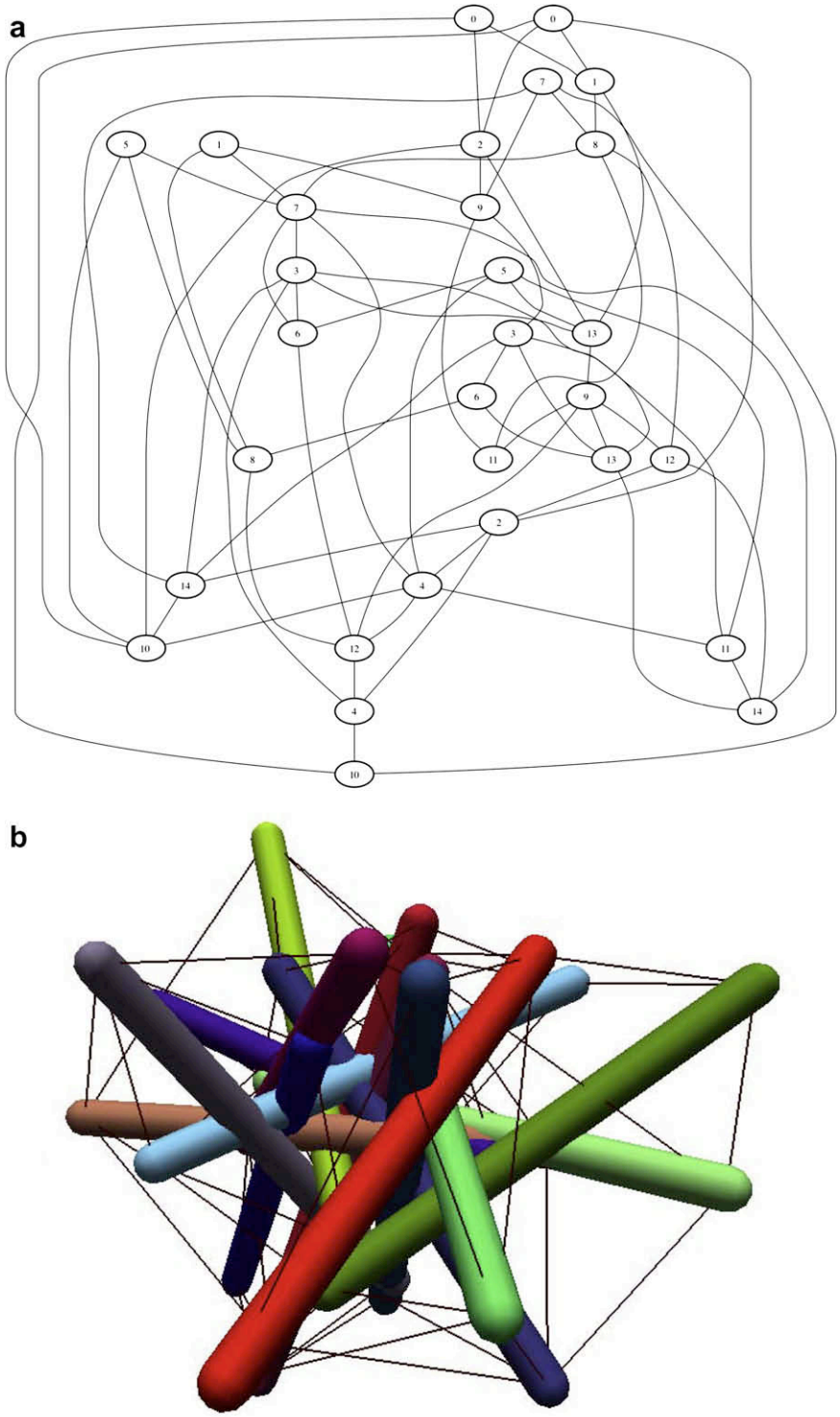

Fig. 5. A 15-strut tensegrity and associated graph evolved using a direct encoding. Tensegrities produced with the direct encoding tend to be irregular, and have correspondingly irregular underlying graphs. Nodes with matching numbers correspond to endpoints of a single strut.

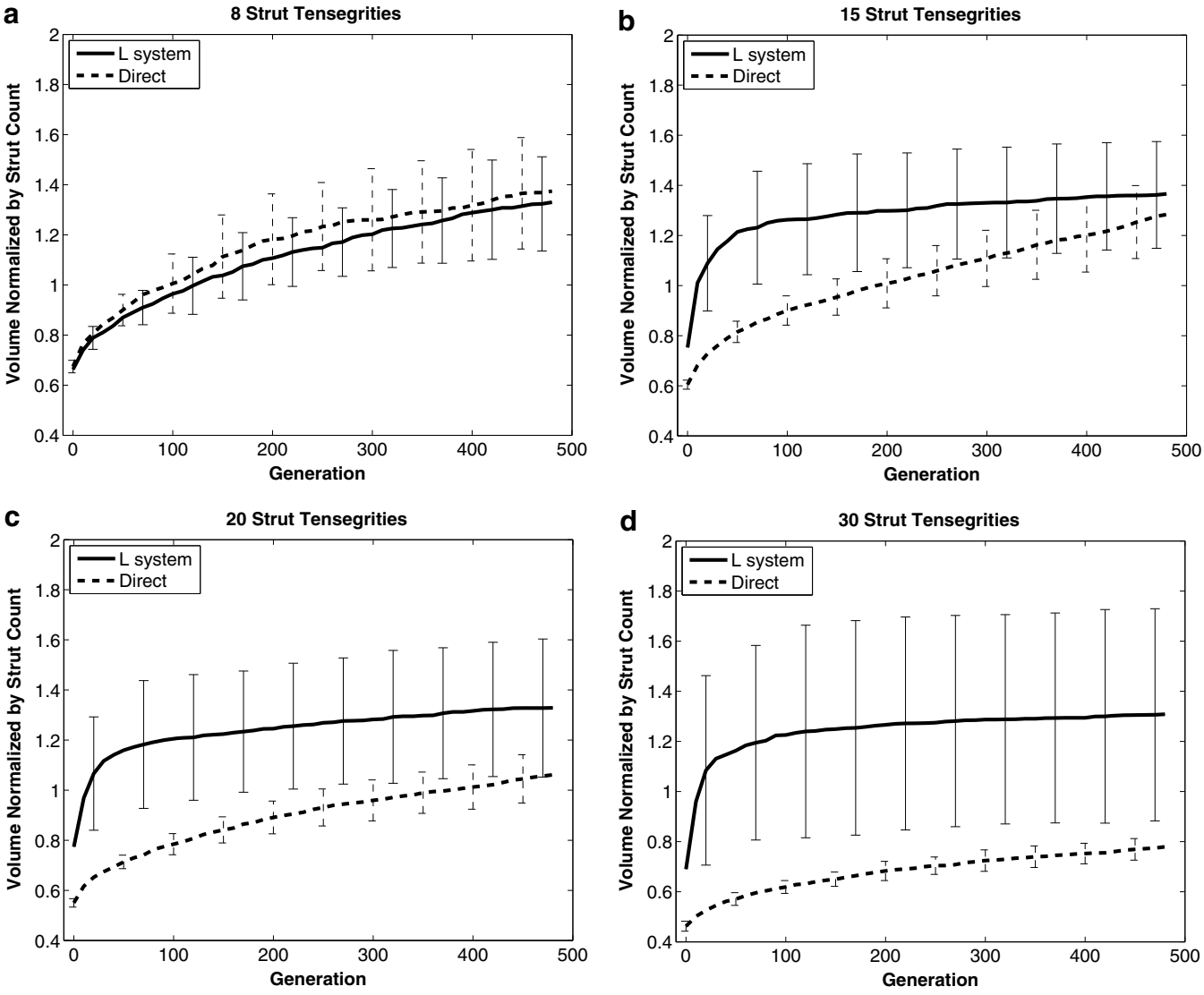

Fig. 6. Comparison of direct and developmental methods.

3D rigid body dynamics. While methods such as dynamic relaxation are well proven for this task, our choice of a dynamics package is driven by the requirements of future research, in which we will exploit the dynamics of large tensegrity structures in order to produce motion. Following Pellegrino and Calladine [31], we will assume that evolved tensegrities have infinitesimal mechanisms.

Rigid elements were represented as solid capped cylinders of fixed length with a length-to-radius ratio of 24:1 and unit mass.Tensile elements were represented as spring-like forces acting upon the cylinder ends. A given string s_i with length L_i, rest length L_0, and spring constant K produces a force $\hat{F}_i$:

$$\hat{F}_i = \begin{cases} K * (\hat{L}_i - \hat{L}_0) & \text{if } \hat{L}_i > \hat{L}_0 \\ 0 & \text{if } \hat{L}_i \leqslant \hat{L}_0 \end{cases} \quad (1)$$

The length of all struts, and the rest length and spring constant of all tensile elements, remained constant throughout evolution. Although each of these could conceivably also be individually varied over the course of evolution, it would induce a significantly larger search space and correspondingly decrease the rate of evolutionary discovery.

In order to find a stable form of the tensegrity the elements were then randomly oriented within the space and the structure was set free to move until it reached dynamic equilibrium. Although it has been shown that tensegrity structures can contain multiple equilibrium states with different potential energies [32], it is unlikely that a tensegrity would continuously equilibrate to the same higher energy shape over the course of multiple randomized strut starting locations and orientations. Small damping drag forces we added to each element's linear and angular velocity in order to reduce oscillations. Collisions between struts were disabled, allowing struts to pass through each other, and intersect if necessary. Once the structure has stabilized such that none of the elements are in motion, the simulation was completed and the fitness of the finalized structure was measured. Details of the simulation of tensegrities within ODE can be found in [7].

2.4.3. Evaluation and fitness

One property desired in tensegrity structures is their ability to encompass as large a volume as possible, with as few struts as possible. There are of several other criteria by which the value of a tensegrity can be judged, such as overall rigidity, clearance between struts, or orthogonality of struts. Our choice of the volume of the convex hull as a fitness metric is consistent with earlier work on evolved tensegrities [14], discourages pathological solutions in

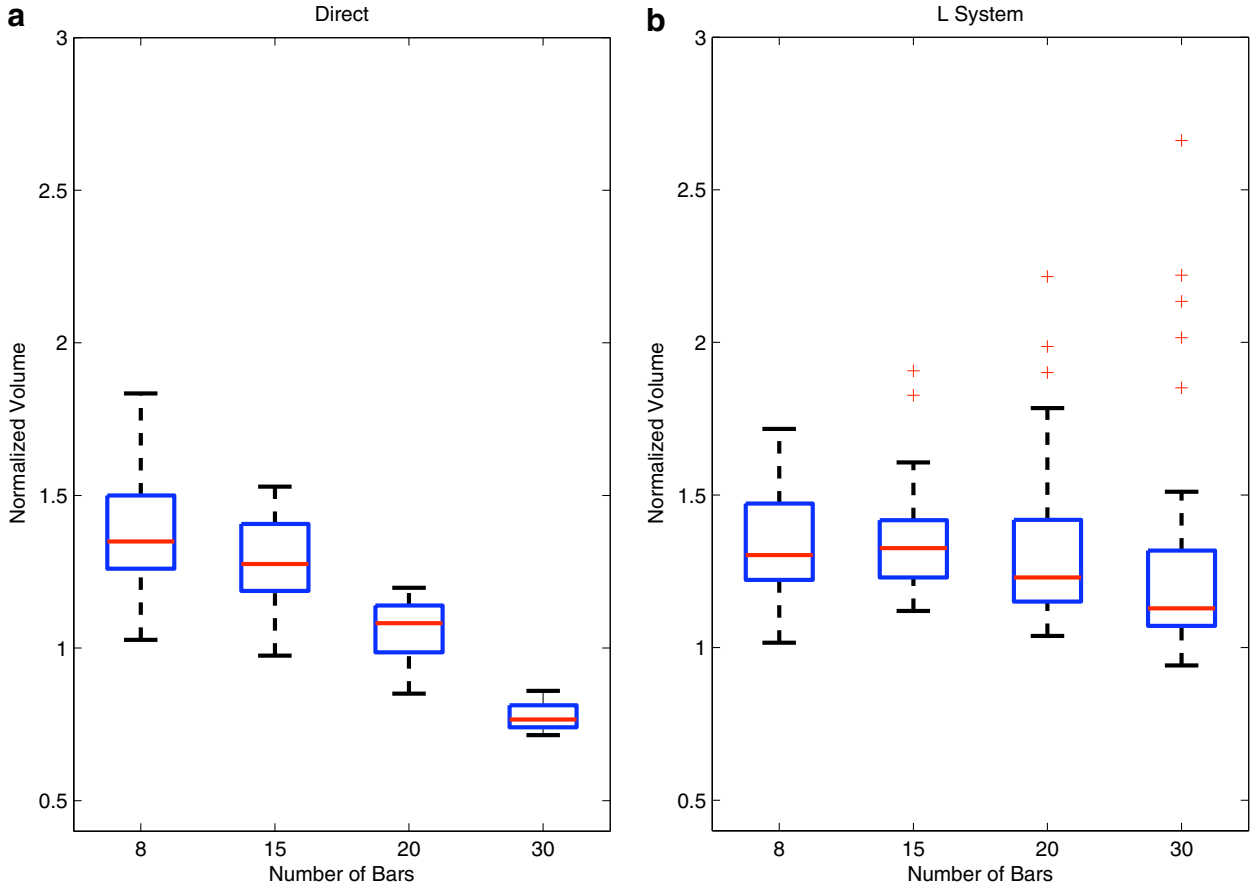

Fig. 7. As the number of elements in the tensegrity increases, the direct encoding (left) suffers in performance, whereas the performance of the developmental map L-system (right) remains stable.

which a structure collapses into a flat or linear, albeit structurally rigid, shape. The volume of the convex hull described by the final location of rigid element endpoints was as calculated by the Quickhull algorithm [33]. Measured volume was normalized by the total number of struts in order to provide a suitable metric regardless of strut count.

3. Experiments and results

An Evolutionary Algorithm was then performed using a population size of 100, 20% crossover rate, 40% primary mutation, 40% secondary mutation, with fitness proportional selection and elitism. In order to reduce the likelihood of any particular graph dominating the population with isomorphs, children produced via secondary mutation were only added to the population if their fitness was greater than that of their parent. Complete runs of 500 generations required on the order of four hours to complete on a 3.2 GHz Intel Xeon processor.

The population was initialized with random 5-rule L-systems, all operating on the starting graph shown in Fig. 3. Graphs were grown until they had the desired number of nodes of degree 3. Those which failed to grow, contained an odd number or nodes, or which did not contain a permutation of vertex pairings corresponding to element endpoints (as described in Section 2.3) were discarded. For comparison we also ran an identical GA using a direct representation similar to that of our earlier work [14].

Fig. 5 shows a representative evolved tensegrity 8-strut structure and its corresponding graph.

As shown in Fig. 6, which compares the progress of the map L-system method against the direct encoding method, the map L-system significantly outperforms the direct encoding across all runs. Furthermore, as the complexity of the structure increases, so does the performance of the map L-system, as shown by the box plots in Fig. 7, which compares final fitness values across all runs.

4. Discussion

In order to explain this marked difference in performance between direct and map L-system encodings we can compare the fitness distribution of random tensegrities generated by each representation. As indicated by Fig. 8, while both populations have similar mean fitness, the tensegrities generated by L-systems have much higher variation in fitness, suggesting that it is capable of exploring a wider swath of the search space.

4.1. Finding semi-regular structures

A further explanation of the advantage of the generative map L-system is produced by comparing the graphs underlying tensegrities evolved under each scheme. As shown by the graph and tensegrity in Fig. 5, the tensegrities produced with the direct encoding tend to be irregular, and have correspondingly irregular underlying graphs. This is because there is nothing inherent in the direct encoding which is capable of discovering, or exploiting, the underlying structural properties or patterns of tensegrity.

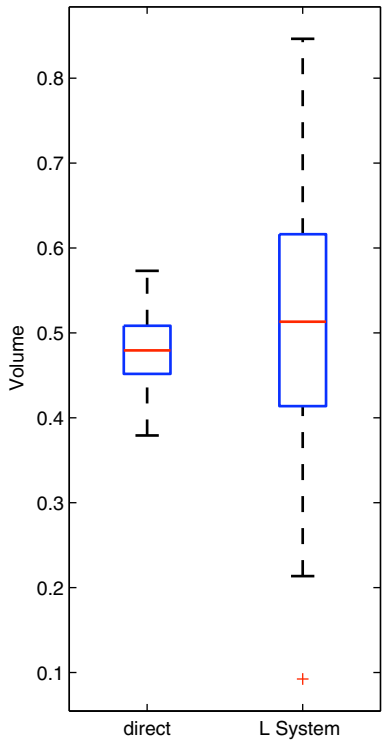

Fig. 8. A comparison of the fitness values of randomly generated tensegrities using the direct encoding (left) and map L-system (right). While the distributions have similar means, the map L-system produces wider variation, leading to a more evolvable system.

Those tensegrities evolved using the map L-system, on the other hand, tend to exhibit a much higher degree of pattern and regularity, as demonstrated by those shown in Fig. 9. The grammatical nature of the map L-system is capable of generating repeated patterns, such as those which can be clearly seen in the underlying graphs of the evolved tensegrities. Patterns such as the repeated chain of nodes in the first graph of Fig. 9 correspond to repeated structural elements in the tensegrities which they represent. This *implicit* patterning allows the map L-system to discover and exploit useful structural properties of tensegrity. These chains can be seen as an emergent and automated analog to the explicit, and human-derived repeated patterning of tensegrity towers such as Snelson's sculptures and the Cyclic Frustum Tensegrity Modules produced by Nishimura et al. [8].

4.2. Open-ended growth of tensegrities

A final exploration of the power of this grammatical approach lies in the potential for open-ended generation of tensegrities. In the experiments above, we terminate graph growth once the desired number of element-pairs has been produced, and select for fitness of the corresponding tensegrity. The growth of those same graphs, however, can iterated several more steps in order to produce increasingly large tensegrity structures. Because of the ability of these grammars to find underlying patterns of tensegrity structure, iteration of the grammars can result in iterations of those structural patterns.

The most striking example of this generative growth is shown in the sequence of images in Fig. 10. The same grammar which grows a linear tensegrity tower of ten struts in the top picture, continues the pattern when iterated through, 20, 30, 40 and even 50 struts. This points to a *generality* in the underlying grammar – it has discovered not just a tensegrity, but an entire class of tensegrities.

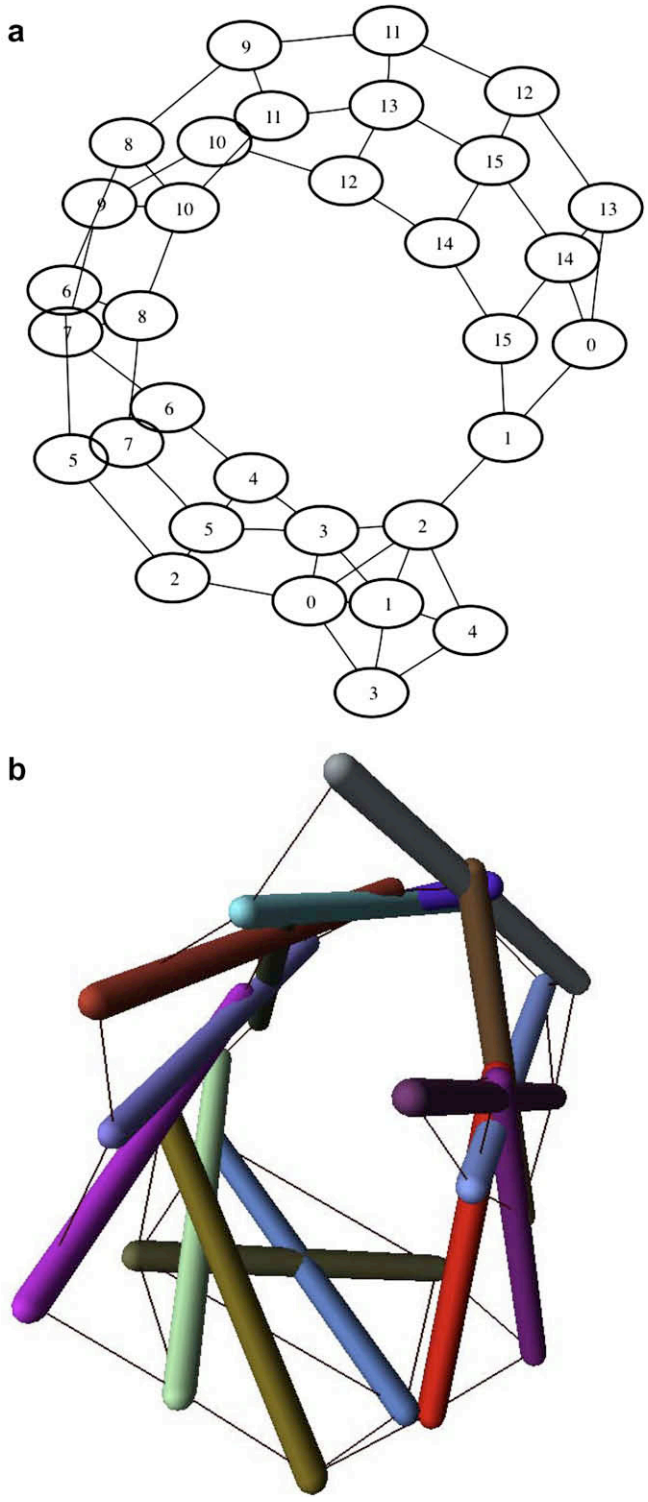

Fig. 9. A 16-strut tensegrity evolved using map L-systems along with its corresponding graph. As can be seen, the graph contains regularities which directly translate into structural patterns in the tensegrity.

5. Tensegrity optimization for fabrication

Note that although the structures evolved in the above sections were tensegrities in the mathematical sense, struts were allowed to intersect and penetrate, and so they are less than ideal from an engineering standpoint. Although collisions between struts is also significantly reduced by allowing each strut to have an infin-

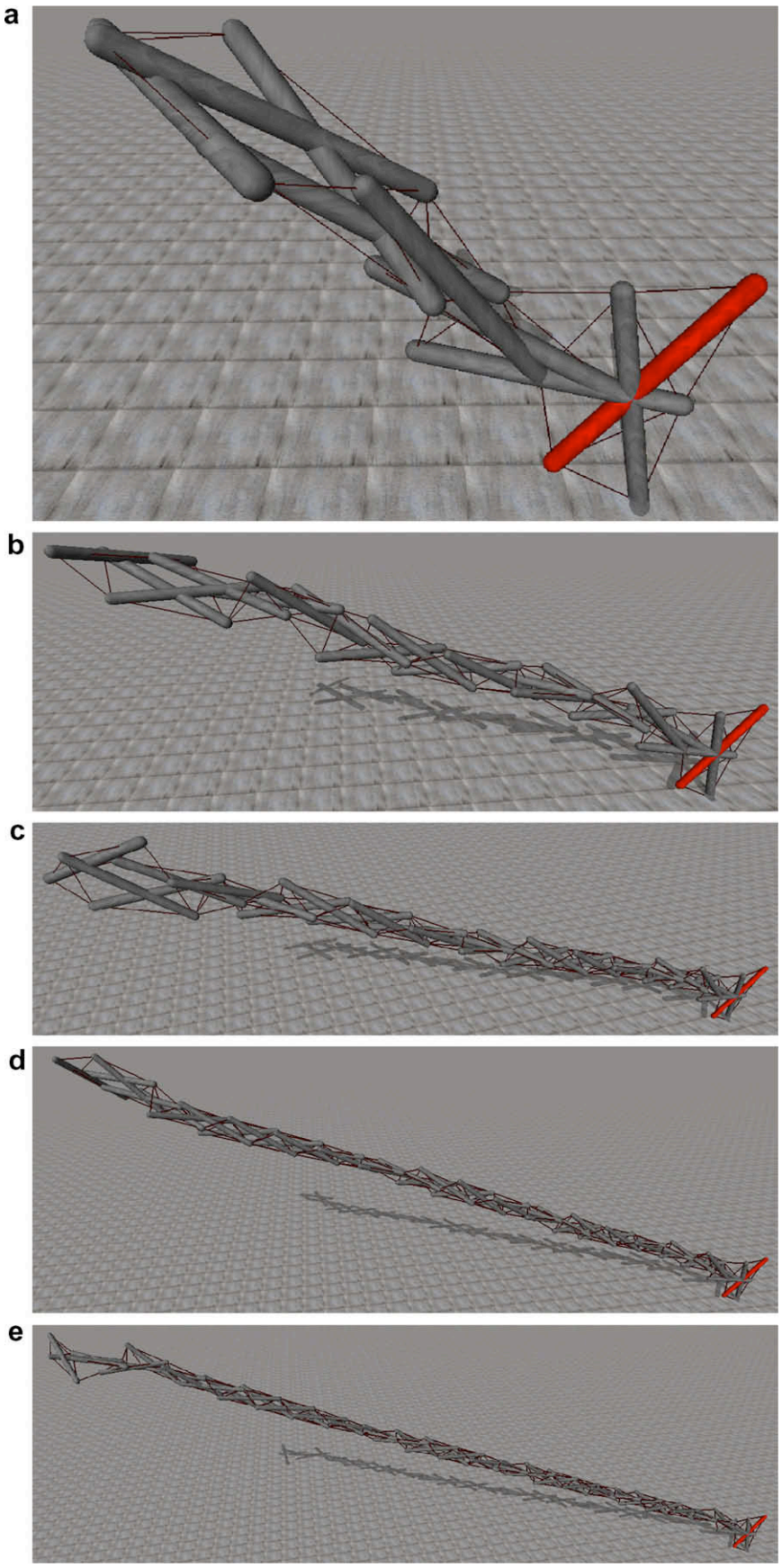

Fig. 10. Five tensegrity towers produced by the same grammar. As the number of iterations of the grammar increases, the tower grows from ten struts to 20, 30, 40 and 50, repeating the same pattern of twisting struts as it grows.

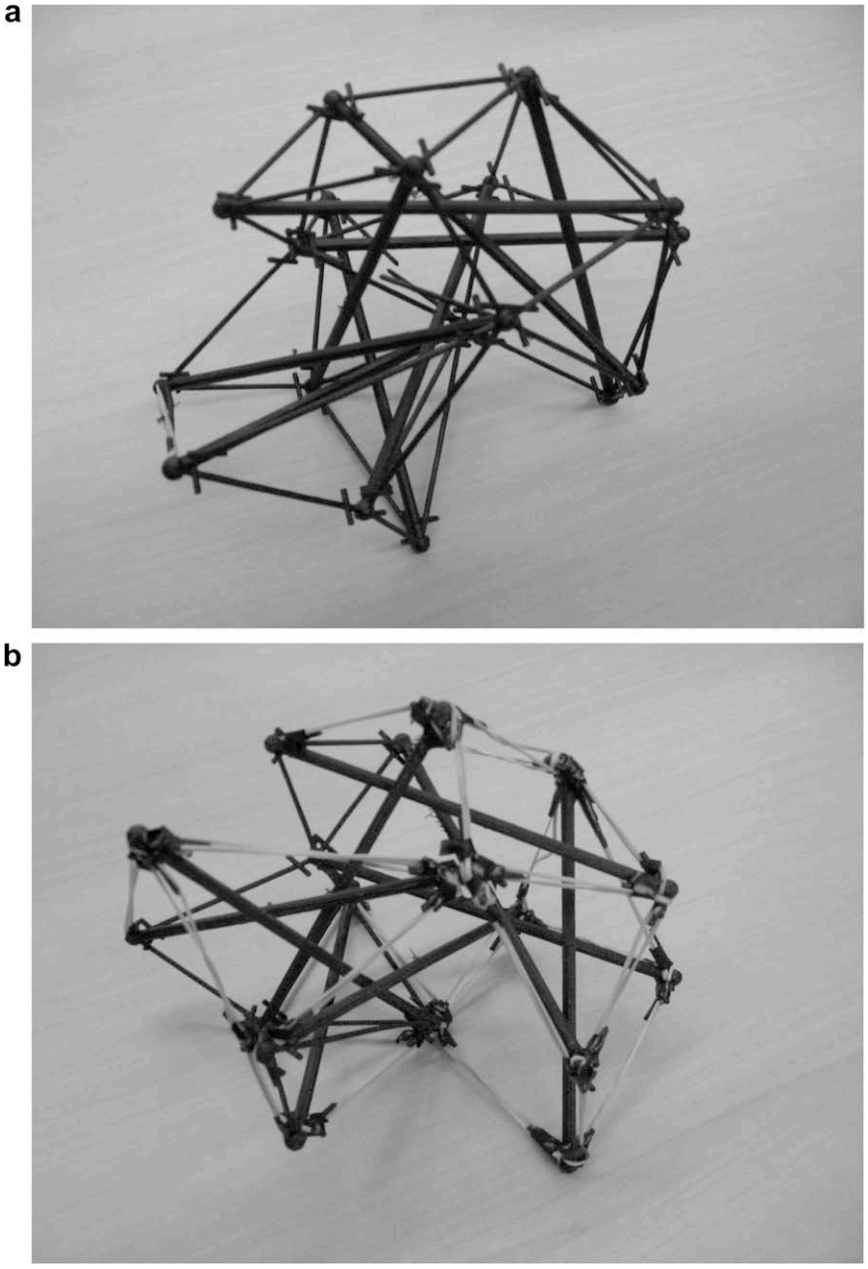

Fig. 11. A physical manifestation of an evolved tensegrity printed using a rapid prototyping machine, before (left) and after (right) printed strings are replaced with elastic.

itesimal cross sectional diameter, this is also not a practical engineering solution. Instead, in order to eliminate internal collisions between elements, tensegrities were post-processed by performing a Random Mutation Hill Climber (RHC) on each string's rest length. Candidate solutions were selected if they either reduced the number of collisions, or reduced the penetration depth. Search was terminated when the collisions were eliminated. This approach produces structures with non-colliding struts, but does not guarantee that struts do not subsequently collide when the structure is subjected to loading. This would be a valuable criterion to add in the future.

As anyone who has attempted to assemble tensegrities by hand can attest, even modest regular four strut structures can be incredibly vexing to produce. Needless to say, the task of assembling our evolved, irregular tensegrities by hand is even more difficult. Care must be taken to preserve each element's 3-D orientation and position, as well as each individual string's unique rest length. We overcome this challenge by directly reproducing the structures, *in situ*, with a rapid prototyping machine.

5.1. Fabrication

Once optimized to eliminate collisions, a tensegrity can be automatically reproduced in 3-D using a rapid prototyping machine. Because it is impossible to print elements under tension, rigid place-holder strings are printed in the place of tensile elements. Once printing is complete these place holder strings are replaced with elastic elements, producing the final tensegrity. Fig. 11 shows several stages of this process.

As the size of evolved tensegrities increased, they became less practical to print in one piece. Instead they were carefully constructed by hand from laser-cut struts and elastic bands, as shown in Fig. 12. Even with the original 3-D model was used as a reference guide, assembly took several hours.

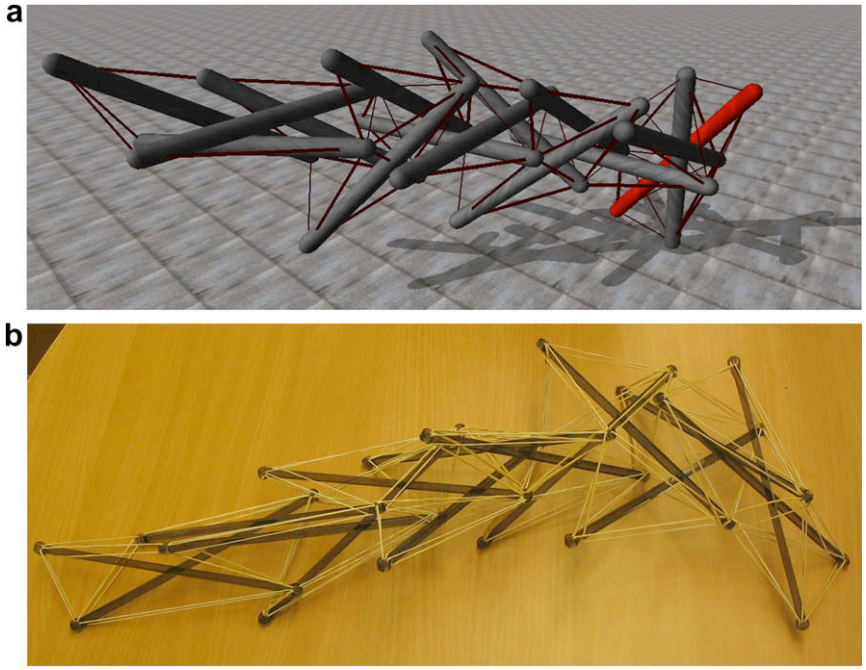

Fig. 12. Tensegrities too large to be printed out in one piece by the rapid prototyping machine, such as this 15 strut tensegrity, were built by hand from unit struts and elastic bands.

6. Conclusion

Conventional mathematical approaches to tensegrity form finding tend either to be limited to regular structures or to scale poorly. An early evolutionary approach of our own produced promising results, but also suffered scalability issues. In this work we have introduced a generative and grammatical graph-based approach to representing and evolving large structures. This representation allows for the discovery of large, complex and *irregular* tensegrity structures, and scales very well as the size and complexity of the structures increases. Furthermore, the generative aspect of our representation allows a grammar to describe not just a single tensegrity, but an entire *class* of tensegrities, as demonstrated by the towers in Fig. 10.

Once evolved in simulation, the task of evaluating and testing them is rendered significantly easier by our ability to directly print the structures in their final form. As our discoveries scale into dozens of struts, they become too large to be printable, and must instead be built slowly and carefully by hand. A remaining challenge raised by our research is the discovery of a more understandable and *prescriptive* representation of these complex structures – one which describes not just the object of assembly, but the process of assembly as well.

The irregular tensegrities produced in this work are the significantly larger and more complex than any others, both man-made and computer-generated, that we have seen in the literature. We have, therefore opened up a vast new slice of the space of tensegrity structures to exploration by engineers and architects alike. Given the growing popularity of tensegrities in engineering and architecture, as well as a growing understanding of the role of tensegrity in biological systems, our methodology addresses a growing need for methods of generating novel large complex tensegrity structures.

Furthermore, because of their collapsibility and deployability, tensegrity structures offer a robust robotic platform which can rapidly and drastically change shapes by altering string tensions, for instance collapsing flat for portability and then fully re-deploying the press of a button. As such, tensegrity structures would be an ideal means of exploring the budding field of *soft robotics*.

Acknowledgements

The authors would like to thank Jonathan Hiller for his diligent effort in the manual construction of several of the tensegrities shown. This research was supported in part by the United States DCI Postdoctoral Research Fellowship Program, Award Number NMA501-03-1-2013.

References

[1] Buckminster Fuller R. Synergetics—explorations in the geometry of thinking. Macmillan Publishing Co.; 1975.
[2] Wang Bin-Bang. Cable-strut systems: Part i – tensegrity. J Construct Steel Res 1998;45(3).
[3] Connelly Robert, Back Allen. Mathematics and tensegrity. Am Sci 1998;86.
[4] Ingber Donald E. The architecture of life. Sci Am 1998(January).
[5] Masic Milenko, Skelton Robert E. Open-loop control of class-2 tensegrity towers. In: Proceedings of SPIE 11th annual international symposium on smart structures and materials; 2004.
[6] Chan Wai Leung, Arbelaez Diego, Bossens Frederic, Skelton Robert E. Active vibration control of a three-stage tensegrity structure. In: Proceedings of SPIE 11th annual international symposium on smart structures and materials; 2004.
[7] Paul Chandana, Valero-Cuevas Francisco J, Lipson Hod. Design and control of tensegrity robots for locomotion. IEEE Trans Robot 2006;22(5).
[8] Nishimura Yoshitaka, Murakami Hidenori. Initial shape-finding and modal analyses of cyclic frustum tensegrity modules. Comput Meth Appl Mech Eng 2001:5795–818.
[9] Tibert AG, Pellegrino S. Review of form-finding methods for tensegrity structures. Int J Space Struct 2003;18(4):209–23.
[10] Juan Sergi Hernandez, Tur Josep M Mirats. Tensegrity frameworks: static analysis review. Int J Mech Machine Theory 2007.
[11] Rene Motro. Tensegrity: structural systems for the future. Kogan 2003.
[12] Micheletti A, Williams WO. A marching procedure for form-finding for tensegrity structures. J Mech Mater Struct 2007;2(5).
[13] Zhang Li, Maurin Bernard, Motro Rene. Form-finding of nonregular tensegrity systems. J Struct Eng 2006;132(9).
[14] Paul Chandana, Lipson Hod, Valero-Cuevas Francisco J. Evolutionary form-finding of tensegrity structures. In: Proceedings of the 2005 genetic and evolutionary computation conference (GECCO'05); 2005.

[15] Koza John R, Keane Martin A, Streeter Matthew J, Mydlowec William, Yu Jessen, Lanza Guido. Genetic programming IV: routine human-competitive machine intelligence. Kluwer Academic Publishers; 2003.
[16] Hugues J. Evolution of non-deterministic incremental algorithms as a new approach for search in state spaces. In: Eshelman LJ, editor. Proceedings of the sixth international conference on genetic algorithms; 1995.
[17] Preble S, Lipson H, Lipson M. Two-dimensional photonic crystals designed by evolutionary algorithms. Appl Phys Lett 2005;86.
[18] Al-Sakran Sameer H, Koza John R, Jones Lee W. Automated re-invention of a previously patented optical lens system using genetic programming. In: Keijzer Maarten, Tettamanzi Andrea, Collet Pierre, van Hemert Jano I, Tomassini Marco, editors. Proceedings of the eigth European conference on genetic programming. Lecture notes in computer science, vol. 3447. Lausanne, Switzerland: Springer; 2005. p. 25–37.
[19] Massey Paul, Clark John A, Stepney Susan. Evolution of a human-competitive Quantum Fourier Transform algorithm using genetic programming. p. 1657–64.
[20] de Guzman Miguel, Orden David. From graphs to tensegrity structures: geometric and symbolic approaches. Publ Mat 2006;50.
[21] Prusinkiewicz P, Lindenmayer A. The algorithmic beauty of plants. New York, USA: Springer-Verlag; 1990.
[22] Hemberg Martin, O'Reilly Una-May. Extending grammatical evolution to evolve digital surfaces with genr8. In: EuroGP; 2004.
[23] Schmidt Linda C, Shetty Harshawardhan, Chase Scott C. A graph grammar approach for structure synthesis of mechanisms. J Mech Design 2000;122(4):371–6.
[24] Shea Kristina, Cagan Jonathan. Innovative dome design: applying geodesic patterns with shape annealing. Artif Int Eng Design Anal Manuf 1997;11(379–394).
[25] Shea Kristina, Cagan Jonathan. Languages and semantics of grammatical discrete structures. Artif Int Eng Design Anal Manuf 1999;13:241–51.
[26] Kaveh A. Topological transformations applied to structural mechanics. Comput Struct 1997;63:709–18.
[27] Kaveh A. Structural mechanics: graph and matrix methods. Research Studies Press (John Wiley); 2004.
[28] Kaveh A. Optimal structural analysis. John Wiley; 2006.
[29] Hornby Gregory S, Lipson Hod, Pollack Jordan. Generative encodings for the automated design of modular physical robots. IEEE Trans Robot Automat 2003;19(4):703–19.
[30] Smith R. Open dynamics engine. <http://www.ode.org>.
[31] Pellegrino S, Calladine CR. Matrix analysis of statically and kinematically indeterminate frameworks. Int J Solids Struct 1986;22(4):409–28.
[32] Defossez Marc. Shape memory effect in tensegrity structures. Mech Res Commun 2003;30:311–6.
[33] Bradford Barber C, David P Dobkin, Hannu Huhdanpaa. The quickhull algorithm for convex hulls. ACM Trans Math Softw 1996;22(4):469–83.