10th World Congress on Structural and Multidisciplinary Optimization
May 19 - 24, 2013, Orlando, Florida, USA

Managing Variable-Dimension Structural Optimization Problems using Generative Algorithms

James T. Allison, Ashish Khetan, Danny Lohan

Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building, 104 S. Mathews Ave.
Urbana, Illinois, 61801, USA
{jtalliso,khetan2,dlohan2}@illinois.edu

1. Abstract
A novel abstraction concept for the structural topology and shape optimization by means of evolutionary algorithms is presented. The main idea is to represent structural topology and shape using rules of generative algorithms and operate on the generative algorithm rules using the genetic algorithm instead of on the design description directly. Further, to optimize size in truss design optimization a nested optimization routine is implemented using sequential linear programming. The generative algorithm abstraction and nested loop optimization allows for the concurrent optimization of topology, geometry and size of the truss structures. Furthermore, it is completely independent from any kind of ground structure; this avoids the limitations inherent to ground structure approaches that define a priori what topologies may be considered (potentially preventing innovative design solutions). A further advantage of using this abstraction layer is the ability to manage structural designs of variable dimension, while providing a fixed-dimension genotype for the generic algorithm to operate on. Finally, the effectiveness of the concept is demonstrated by examining one archetypal truss design optimization benchmark problem.

2. Keywords: Truss topology optimization, Generative algorithms, Structural optimization

3. Introduction
Design dimension can vary during the development of many engineering systems. For example, in truss design or automotive powertrain design, as system elements are added or removed dimension of the set of continuous design variables changes. One significant challenge here is that the number of system elements in the optimal design is not known a priori, so a design vector that permits description of the optimal system topology cannot always be defined. One well-known solution is to use a ground-structure approach, where a large number of available system elements are pre-defined, and the optimization vector specifies the existence (and in some cases size and shape) of these elements. This approach is fundamentally limited, as the number and relationship of elements cannot deviate from what is allowed by the ground structure. Established approaches that discretize a given design domain, such as SIMP, are similar in that the number of potential system elements and the available relationships between them are predefined. Here we present a new approach, based on generative algorithms, that overcomes these limitations by accommodating variable design dimension problems and allowing the exploration of design alternatives not prescribed a priori. We demonstrate effectiveness of the proposed methodology on solving truss design problems for optimal size, geometry and topology.

3.1. Truss Optimization Problems
Truss design optimization is a classical subject in structural design optimization, and can be classified into three main categories: (i) sizing,(ii) geometry, and (iii) topology. In sizing optimization of trusses, cross sectional areas of members are considered as design variables and the coordinates of the nodes and connectivity among various members are fixed. In geometric optimization of trusses the change in nodal coordinates are treated as design variables. In truss topology optimization, parameters that govern truss member connectivity are design variables, while node coordinates are held fixed. All the three categories of truss design optimization have been studied widely. Early efforts in truss size optimization were carried out by Venkayya [1], Schmit and Farsh [2], and Dobs and Nelson [3]. Further, Goldberg and Samtani [4] and Rajeev and Krishnamoorthy [5] used evolutionary algorithms to solve the sizing optimization problem in truss design.

One of the most used methods for topology optimization is the Ground Structure approach. This method consists of generating a fixed grid of joints and adding members in some or all of the possible connections between the joints as potential structural or vanishing members. The optimum structure for the imposed boundary conditions and applied loads is found using the cross-sectional areas as design variables, including the possibility of zero-area members. The number of joints is not a design variable. In the classical formulation of the problem, the positions of the joints are fixed, so a high number of joints are used to increase the variety of possible designs. The classical formulation of size and topology optimization has been solved by Deb and Gulati [6] as well as and Hagishita and Ohsaki [7] using genetic algorithms (GAs). Hejela, Lee, and Lin [8] used a two-level optimization scheme of first finding multiple optimal topologies and then finding the optimal member areas for each of the truss topologies.

As the topology optimization using ground structures does not incorporate geometry optimization, an important next step was to extend this approach to include joint position optimization. This integrated geometry and topology design approach has been studied extensively, primarily using a hierarchical solution approach. Further, many GA-based approaches has also been explored to achieve integrated size, geometry and topology optimization. Rahami, Kevah, and Gholipour [9], Giger and Ermanni [10], Rajan [11], Balling, Briggs and Gillman [12], and Kaveh and Laknejadi [13] all used evolutionary algorithms* to find the optimal size, geometry and topology of trusses. In this paper, we present a new methodology to solve the combined size, geometry, and topology truss design problem using generative algorithms as an abstraction layer between an outer-loop GA that solves the topology and shape design problem and an inner-loop sequential linear programming (SLP) implementation that solves the size optimization problem.

Generative algorithms produce output based on a set of rules that is applied iteratively. This class of algorithms has been used widely in the fields of generative art and architecture. Recently one type of generative algorithm, cellular division, has been applied to structural topology optimization, but these early implementations have been limited to predefined design domains, and have not been applied to truss design. Here we use a generative algorithm that outputs truss topology and shape based on a set of rules, and an inner loop solves the size optimization problem for each candidate topology using sequential linear programming. Adjusting the generative algorithms rules results in different topologies (often with different numbers of system elements). The generative algorithm is used as an abstracted representation of truss topology. Instead of optimizing in the topology design space directly, we optimize in the rule space. Design space dimension varies, whereas the rule space dimension is constant. The generative algorithm maps a design in the rule space to the design space. Since the rule space dimension does not grow with system dimension, this approach provides the potential for scaling up to much larger system design problems than what can be solved using existing methods.

The rest of the paper is organized as follows. First, we review how generative algorithms have been used thus far in topology optimization. Then we discuss the basics of cellular division in the map L-system, the generative algorithm used in this work. Following that we present a novel methodology using a modified cellular division method in combination with a genetic algorithm to solve for optimal truss topology, geometry and size. We demonstrate the results of our methodology on benchmark ten-bar truss design optimization problem. Finally we close the paper with a summary of the main findings of the work and possible future explorations of generative algorithms for topology optimization.

4. Generative Algorithms in Topology Optimization

Here we introduce generative algorithms—a class of algorithms that make decisions based on system state and a predefined set of rules—as a potential solution for the challenging requirements in general topology design optimization. While generative algorithms have been used extensively in artistic expression [14–16], recently they have been applied also to engineering design (e.g., structural optimization [17–19] and robotics [20–22]). In engineering design applications, the rules that govern generative algorithms are parameterized such that they can be tuned by an outer-loop optimization algorithm (e.g., and evolutionary algorithm [23]). In other words, designs are represented abstractly using generative algorithms. Generative algorithms do not make design decisions, but their rule parameters represent the design, avoiding the need to represent every element of a complex design directly.

Using generative algorithms to represent topology design allows outer-loop optimization algorithms to operate in the algorithm 'rule space' instead of directly in the design space. This approach offers several advantages. Algorithm rules are a reduced-dimension representation of designs. The number of adjustable parameters associated with a set of rules is the representation dimension, which is typically much smaller

*GAs belong to the larger class of evolutionary algorithms.

than the dimension of the actual design. This allows engineers to scale up to designing much larger systems [21]. In addition, rule parameters with a particular dimension are capable of generating designs with a range of design dimensions. This property is especially desirable for topology optimization, where the number of system elements can vary between design candidates. Design dimension can vary widely in topology design, as the number of system elements are not known a priori. Standard mixed-integer nonlinear programming (MINLP) algorithms cannot solve these variable-dimension problems directly [24], so some type of abstraction layer is required. Generative algorithms serve as this abstraction layer, keeping the dimension of the optimization space constant and enabling the use of standard optimization algorithms.

Generative algorithms may also be classified according to design representation type (i.e., direct vs. indirect). For example, cellular division algorithms are a natural analogy to truss topology and geometry since edges represent bar members and intersection of two or more edges represents pin joints, so these algorithms would be considered a direct approach. Graph evolution algorithms [25, 26] (another class of generative algorithms), however, would be more naturally suited to operate on the dual graph. The following subsection reviews the current state of generative algorithms, paying particular attention to how specific algorithms relate to topology design optimization.

4.1. Survey of Generative Algorithms

The visual nature of truss geometry and topology provides another link to existing generative algorithms that are traditionally used for visual art and design. Here we review a sampling of generative algorithms developed for artistic expression, and discuss their relationship with topology design.

The first set of algorithms reviewed here have the potential for direct representation of geometry and topology (i.e., primal methods). These include algorithms that perform Delaunay triangulation or generate Voronoi diagrams, including the Bowyer-Watson algorithm [27, 28] and Fortune's algorithm [29]. The Voronoi diagram is an established tool in art and design; e.g., it is the basis of the Adobe Photoshop Stained Glass filter [30], and is a key method in more complex visual works, such as Levin's *Segmentation and Symptom* [31] and the archaeological data visualization methods of Varinlioglu et al. [32]. A related class of algorithms, Venation, may have application as either a primal or dual topology representation method [33].

Another class of algorithms that could serve as primal algorithms for geometry and topology design are tree-layout algorithms that are often used to generate fractals and fractal-like imagery. These algorithms are either most concerned with the tree hierarchy, such as the Reingold-Tilford algorithm, or with tree scalability [34]. Lindenmayer (or L-) systems were originally developed to simulate the growth of plants and trees [35,36]. While L-systems have been used recently in structural design as described above, they have been used more extensively in visual arts (e.g., Thorp's *tree.growth* [37] and Coyne's *Context Free* project [38].

Particle systems involve a collection of autonomous agents that can be influenced by other agents, and have been used to model both natural processes and physical behaviors [39–41]. Notable examples include the generative design works by Maddox, such as *Ribbons, Filaments, and Anemone*, as well as in his interactive games *Oasis and Pulsus* [42]. Particle systems have the potential as either primal or dual algorithms for topology design, depending on the specific algorithm and whether agents represent vertices or edges.

Many seemingly unique algorithms used in visual generative art and design practice are, in fact, not unique in terms of the actual algorithms that are used; rather, it could be argued that it is the combination and representation of existing algorithms that is a novel pursuit. Casey Reas, a preeminent visual generative artist and designer and co-founder of Processing [43], describes the constituent generative works of his Process Compendium using terminology such as form, behavior, and element to establish a pseudocode-like language [44]. Many of his works that demonstrate this language not only exhibit characteristics of particle systems and an L-system-like grammar, but their use of circles and lines as primitive forms suggest the potential for use in indirectly generating graphs.

5. Cellular division in Map L-systems

Lindenmayer systems, or L systems, represent a novel type of rewriting where the rewriting is carried out in parallel. The L-systems were developed by the eminent biologist Aristid Lindenmayer [45]. Map L-systems extend the parallel rewriting in L-systems to planar graphs with cycles called maps. The maps are evolved according to cellular division rules. Formally, a map is defined as a finite set of regions. Each region is bounded by a sequence of edges and the edges intersect at vertices. Every edge is part of the

boundary of a region and the regions are simply connected. These maps are analogous to cellular layers, where the regions represent the cells and the edges their walls.
There are numerous variants of map L-systems. This work uses one of the most powerful L-systems, the Binary Propagating Map OL-systems with markers, or mBPMOL-systems, proposed by Nakamura [46]. The method is binary because in the cell division process the cells divide into two. It is propagating since cells cannot fuse or vanish. The designation OL system refers to context-free parallel rewriting systems that do not allow for region interactions. Finally, markers specify juncture points at the edges where the cell can divide. Hereafter, mBPMOL-systems are referred to as map L-systems.

5.1. Map L-systems Implementation
Mathematically, the map L-system consists of an alphabet Σ, an axiom ω, a finite set of rewriting rules P, and any additional special symbols or constants (they are called constants because they are not affected by the rewriting). The alphabet is a finite, non-empty set Σ, whose elements are called letters. Each rule is of the form $A \to \alpha$, where the edge $A \in \Sigma$ is called the predecessor, whereas the string α, composed of symbols from Σ and special symbols, [,], + and −, is called the successor. Pairs of matching brackets [and] specify markers for possible cell-dividing walls. Symbols outside of the square brackets specify the edge subdivisions; each subdivision has the same length. Inside the brackets the first symbol is either + or −, and this symbol defines whether the marker is placed to the left or to the right of the predecessor edge, respectively. The second symbol within brackets is always a letter. Letters can carry an arrow over them to represent the local edge orientation of the successor edges relative to the predecessor edge. A rule is assigned to all letters in Σ. The letters $A, B, \ldots$ are called non-terminal symbols, whereas x is called a terminal symbol—the rule for x is omitted since it is always the identity rule $x \to x$. Examples of rules and their effects on edges are illustrated in the Fig. 1.

Figure 1: Examples of rewriting rules

Left: $\overleftarrow{A} \to \overrightarrow{B}x\overrightarrow{C}[+\overleftarrow{D}]\overleftarrow{E}$ Right: $\overrightarrow{A} \to [-\overleftarrow{C}]\overrightarrow{D}\overleftarrow{B}[+\overrightarrow{A}]\overleftarrow{F}$

5.2. Cellular Division in Map L-Systems
The cellular division process is preceded by the derivation phase where, at first, the production rules are applied to all edges in the map, and second, all the cell edges are scanned for matching markers. If in a cell there exist two markers that carry the same letter and are directed to each other, then they are matching markers. Depending upon the division criteria explained later, a cell division can be formed by connecting these two markers. It is possible for more than one pair of matching markers to be found in a cell. In this case, and owing to the binary character of the method, only the first pair of markers that satisfy the division criteria is selected and the remaining markers are discarded. Once this set of operations is complete a new map has been generated. The derivation and cell division process are then repeated as many times as required, or until all edges are labeled with the terminal symbol x. The number of times this process is repeated is specified beforehand and is called the number of development stages. Fig. 2 shows the first two development stages of the cellular division process for non-oriented 'Cartesian' map L-systems defined by:

$$\Sigma = \{A, B\}$$
$$\omega = ABAB$$
$$P = \{A \to B[-A][+A]B, B \to A\}$$

The initial map has the edge labels defined by the axiom $\omega = ABAB$: starting with the label A at bottom and proceeding counterclockwise. The cellular division proceeds by simultaneously rewriting all its edges: both vertical edges B are rewritten to edges A according to the rule, $p_2 : B \to A$, and the horizontal edges A are transformed according to the rule, $p_1 : A \to B[-A][+A]B$. We use a global counterclockwise orientation to decide right and left for this representation. Thus the lower edge A is first subdivided into

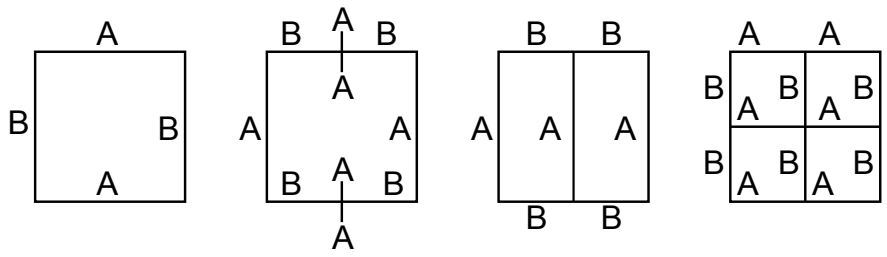

Figure 2: Example of cellular division in map L-system

two equal segments corresponding to the number of non-bracketed letter in the rule. The first segment is labeled B. This is followed by two markers: a marker of type A is placed to the right of the initial edge according to the $[-A]$ command, and a marker of type A placed to the left according to the next command $[+A]$. The last letter in the rule label as segment B. A similar procedure is applied to the top edge resulting in the intermediate stage depicted in the second schematic from left in Fig. 2. At this stage, all edges have been rewritten and then matching markers are searched in the cell. The two matching markers of type A are connected and the resulting edge is labeled as A, the third schematic in Fig. 2. This completes the first development stage and the same process is repeated on all the edges and matching markers are connected of the two cells which results the fourth schematic of Fig. 2. This completes the second development stage.

6. L-System Extension to Truss Design

The proposed methodology uses a GA to operate on rules of the cellular division algorithm to optimize truss topology and geometry. For each individual candidate topology and geometry, an inner loop optimization problem solves for optimal member sizes using sequential linear programming (SLP).

The cellular division algorithm outputs a developed map given an initial map, a set of rules, and number of development stages. However, the map generated by the cellular division in map L-systems holds no canonical physical meaning; therefore, for each optimization problem an identification between the elements of the topology (cells, edges and vertices) must be established. For the truss design optimization problem at hand, the cellular division method provides an intuitive identification where the edges of the topology represent truss bars and their intersection represents pin joints. This provides an abstraction for topology and geometry of truss design via cellular division algorithm rules.

A randomly generated cellular division rule-set, however, may not represent a valid truss topology. The map developed by cellular division, when identified with edges as truss bars and intersection of edges as pin joints, may render a mechanism instead. This work presents a set of suitable modifications to the original cellular division method in map L-systems that ensures resulting maps satisfy truss stability requirements (i.e., no mechanical degrees of freedom). In other words, any randomly generated rule-set will output a stable truss topology design.

The genotype for the genetic algorithm is the rule-set, and the phenotype is the truss topology and geometry. Fitness for each individual in the population is the minimal weight obtained by solving the inner loop size optimization problem. In the inner loop, each individual is solved for a fixed number of iterations using SLP methods under the constraints of the design problem. Further, the degree to which constraints are not satisfied by the inner loop are enforced by genetic algorithm through penalization. In other words, the genetic algorithm gives the optimal topology and geometry which, when solved for optimal cross section areas of bar members, has the minimal weight. Figure 3 illustrates the methodology graphically. The section below explains mathematical genotype representation. Further, we explain a set of modifications in the original cellular division method which ensures stability of truss topology designs.

6.1. Genomic Encoding of Cellular Division Rules

The GA used here acts upon the individual genes that encode all the information required for the topology and geometry represented by a cellular division process, namely the map L-system axiom ω and the production rules P. For the purposes of the GA used here, the genome is encoded as a vector of reals, X, whose elements are in the interval $[0, 1]$. The structure below exemplifies the structure of X.

X_1^ω	X_2^ω	...	...	X_n^ω	$X_1^{P_1}$	$X_2^{P_1}$	...	...	$X_m^{P_1}$	$X_1^{P_2}$	$X_2^{P_2}$	...	...	$X_t^{P_2}$	...	...

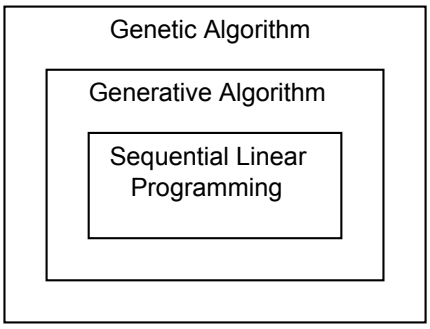

Figure 3: The proposed methodology

The first n elements of X encode the axiom word. The length of the axiom (n) is equal to the number of edges in the initial map. To convert from reals to the letters, the interval $[0, 1]$ is divided evenly in as many segments as the number of symbols in the alphabet Σ. For the encoding of the production rules of the map L-system, a similar approach is followed where each rule is encoded according to a master rule of the form:

$$\sigma_i \rightarrow X_1^{P_i} X_2^{P_i} ... X_{m-1}^{P_i} X_m^{P_i}, \tag{1}$$

where σ_i is the ith letter in Σ and $X_1^{P_i}$ is a token that encodes an independent entity of the production rule of an alphabet. This work uses non-oriented map L-systems and a modified production rule wherein each token represents an alphabet only or an alphabet and a marker. To incorporate all possibilities in a token, all tokens are encoded as vector of four real values that are capable of representing any possible token. A token may also represent a blank space. With the inclusion of the blank space, the size of the production rules can vary; however, the maximum length is dictated by the number of slots in the rules, m, which is decided by the user and is the same for all rules. The structure of the vector $X_1^{P_i}$ is:

Blank Space	Edge Alphabet	Read-Marker	Marker-Alphabet

The first element encodes whether or not the token is a blank space. The following codes are ignored if this element encodes a blank space. The second element encodes the edge alphabet $\{A, B, C, ...\}$. The third element encodes whether or not the edge division is followed by a marker. The fourth element encodes the marker alphabet. If there is a marker, it is considered to be on both the sides of the edge, i.e., it can be used for division of cells on either side of the edge. All the elements of the production rule are encoded on an interval of $[0, 1]$ and are decoded based on the number of total possible values of that element, being evenly divided on the range of encoding interval. The probability of having a blank token is smaller than having a non-blank token, and therefore the range is unevenly divided between $[0, 0.2]$ and $[0.2, 1]$ for blank and non-blank tokens, respectively. The probability of having a marker following an edge alphabet used here is 0.8; therefore range is accordingly divided between read-marker true and false.
In summary, after decoding, the genome can be seen as a partitioned array of symbols whose first part is occupied by the n letters of axiom and the second part by the production rules for each alphabet. The number of tokens for each production rule may vary as there is some possibility of a token being blank.

6.2. Modification of Cellular Division for Truss Optimization
The map developed by cellular division in map L-systems—where edges correspond to truss bars and edge intersections correspond to pin joints—may not result in a stable truss design due to the nature of the cellular division algorithm. To illustrate this concept, consider the ten randomly generated truss topologies shown in Fig. 4. None of them represent a stable truss design, i.e., they are all mechanisms. These topologies were generated using four development stages on a set of four alphabets with six tokens and an axiom of seven edges. The probabilities for blank space and markers are the same as mentioned above. Generation of a stable truss is in fact rare when using the standard map L-system algorithm. Previous work in structural topology optimization generated truss-like systems, but these were in fact frames, greatly simplifying the design problem [18]. This motivates a modification to the cellular division algorithm to permit application to truss design.
Stability of a randomly generated truss topology and geometry can be fully ascertained by checking the singularity of the corresponding stiffness matrix. This approach, however, would necessitate either a

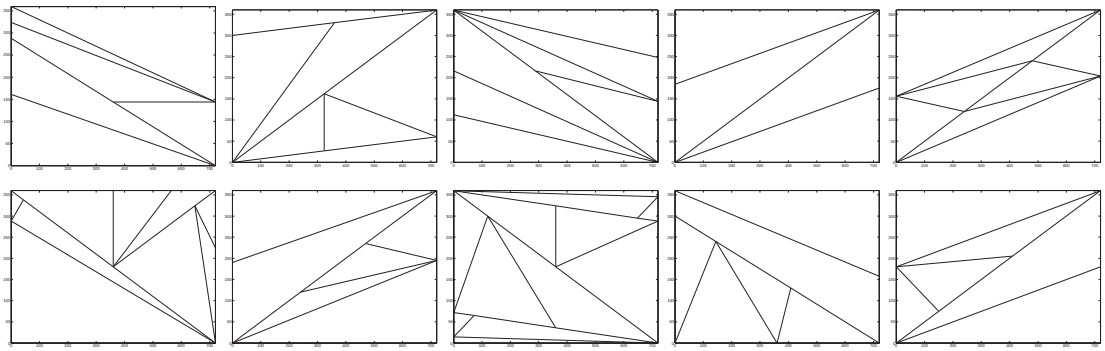

Figure 4: Randomly Generated Truss Topologies

penalty function or a repair mechanism since stiffness matrix singularity cannot be incorporated directly into the cellular division algorithm. Using a repair mechanism would result in sub-optimal designs, and both approaches would be inefficient since a large number of unstable structures would be generated during the solution process. To ensure that the output of the cellular division process for any set of randomly generated rules is a stable truss design, a set of modifications in cell division process has been developed.

Here we present a set of constraints which, when enforced upon the cellular division process, ensures stability of resulting truss topologies after each stage of map development. We endeavor to design the constraints which lead to a truss design which is a summation of triangles, as stability of such trusses is guaranteed and need not be checked by the stiffness matrix. We take the axiom map as a summation of triangles and restrict the cellular division process such that a cell division takes place only if it divides the cell into two triangles. This results in the generation of designs which are a summation of triangles. For this purpose, two matching markers are connected only if one of them is at a vertex. Further, if the non-vertex marker is located at an internal edge it would lead to creation of a quadrilateral on the other adjoining cell of the dividing edge. A quadrilateral may lead to an instability. To avoid quadrilaterals, the adjoining cell is divided at the same marker location with the opposite vertex of that cell. If the adjoining cell has already been divided in that stage of development the first cell is not divided at this marker.

Since these constraints reduce the probability of a cell division, high marker probability is needed in the randomly generated production rule. Here markers are placed with a high probability after every edge-division alphabet in the production rule. In addition, this modified algorithm uses non-oriented map L-systems, and a new production rule where a marker at any edge may be used for division of the cells on either side of the edge. When a cell has more than one pair of matching markers, the division which leads to a more even cell division is preferred. If the cell cannot be divided at that marker due to the triangle generation constraint, the next best pair of matching markers are checked. To avoid a small angle between two truss members a cell is not divided if it leads to skewed division of the cell. Figure 5 depicts a set of randomly generated truss topologies using the modified cell division process. All systems produced by this modified algorithm are stable truss designs. These topologies are generated with the same parameters used for the random (unstable) topologies illustrated in Fig. 4. This modified cellular division process leads to a higher number of cell divisions for the same number of development stages due to the higher probability of markers and forced division of adjoining cells. The cellular division process in map L-system grows exponentially with the number of edges. This is also a reason for higher number of cell divisions in modified cell division process.

7. Results and Discussion

The methodology described above is demonstrated on the archetypal ten-bar truss benchmark design problem. The initial topology and geometry of the ten-bar truss considered here is shown in Fig. 7, and the problem data is given in Table 6. Here we formulate the problem such that it finds the optimal topology and geometry in the given design space boundary of the ten-bar truss. The size of the truss members is considered to be a continuous variable with a given lower bound. The concurrent optimization problem of finding optimal size, geometry and topology is represented mathematically using the equations below. The design variables are n (no. of nodes), $C_{i,j} \in \{0, 1\}$ (indicates whether nodes i and j are

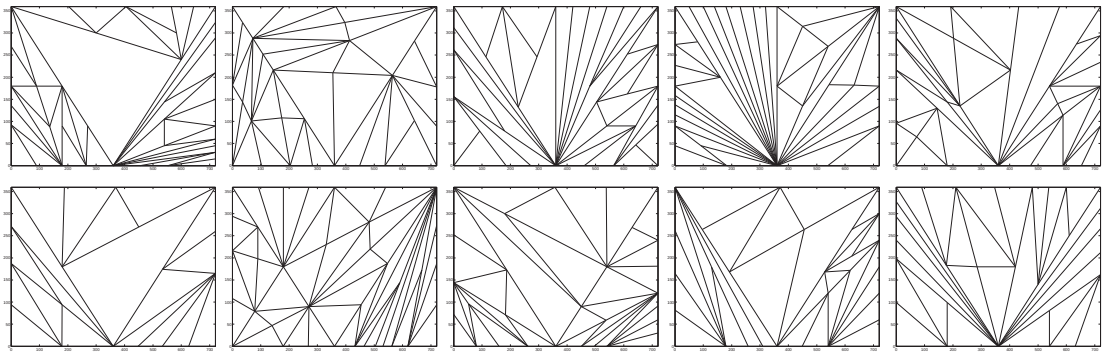

Figure 5: Randomly Generated Stable Truss Topologies

Parameter	Value
L	360 in
P	100 kips
Stress Limits	±25 ksi
Modulus of Elasticity	104 ksi
Material density	0.1 lb/in^3
Section areas lower limit	0.1 in^2
Section areas upper limit	None
Number of loading conditions	Single loading

Figure 6: Input Data for Ten-bar Truss

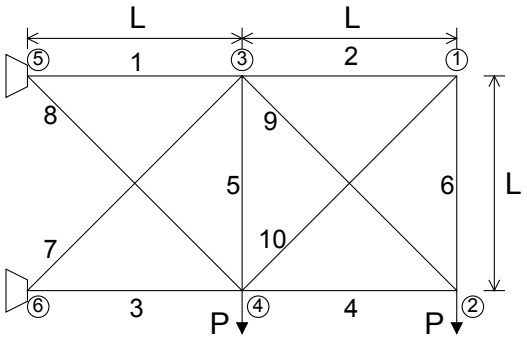

Figure 7: Geometry of Ten-bar Truss

connected), $A_{i,j}$ (cross-sectional area of the bar connecting nodes i and j), and $l_{i,j}$ (length of bar that connects nodes i and j). The variable n and $C_{i,j}$ define topology of the design and coordinates of the nodes i,j represent geometry of the design. The objective is to minimize weight of the truss design subject to stress constraints on each bar. For simplicity of solving inner loop optimization using the SLP approach, only stress constraints are considered. However, the same methodology can be extended to include displacement constraints as well.

$$minf = \sum_{\substack{0 \le i \le n \\ 0 \le j \le n}} \rho C_{i,j} A_{i,j} l_{i,j} \tag{2}$$

Subject to

$$\sigma_{min} \le \sigma_{i,j} \le \sigma_{max} \tag{3}$$

The map L-system used in the generation of the topology and geometry is defined as follows:

- The initial map is bounded by the rectangle $[0, 720] \times [0, 360]$. In addition, nodes (1,4) and nodes (4,8) are also connected to make the initial map a summation of triangles and to ensure that coordinate [0,360] is a pin joint the lower boundary is encoded as two edges in the initial map Fig. 8.
- The initial map has 5 edges on boundaries and 2 in the internal map area as depicted in the Fig. 8.
- The number of development stages is kept 2.
- The alphabet Σ contains 4 letters.
- The production rules have 6 tokens.

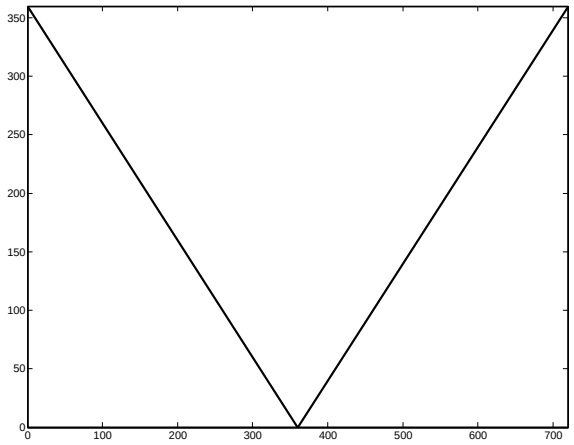

Figure 8: Initial Map

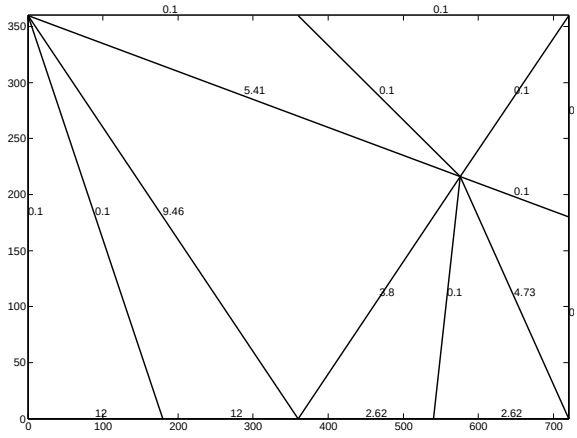

Figure 9: Optimal Design-1

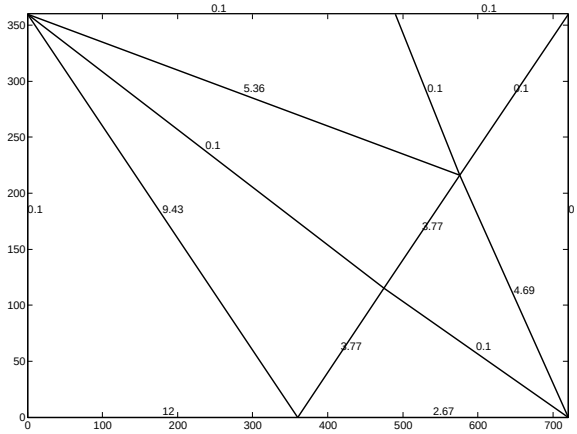

Figure 10: Optimal Design-2

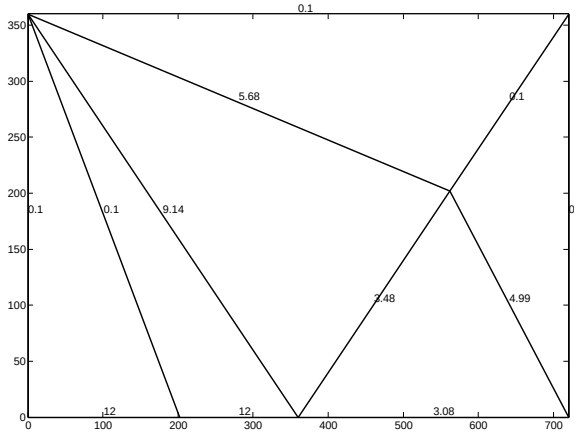

Figure 11: Optimal Design-3

- The probability of a blank token is 0.2 and the probability of a marker after a division-edge is 0.8.

The inner loop optimization is solved using the SLP approach with the use of *linprog* function of MATLAB for 100 iterations. Given area values, stress values of the bar members are computed using force method. The genetic algorithm to optimize the rules of cellular division is implemented using *ga* function of MATLAB for 10 generations.
The 5 optimal truss designs obtained through this methodology are shown in Fig. **??**. The cross sectional area values of the bar members are shown adjacent to the bar members. The weight of the 5 optimal designs and the cross sectional areas of the members carrying load is shown in Table 1. The optimal weight obtained using this methodology is comparable to the results reported in literature [1–3].

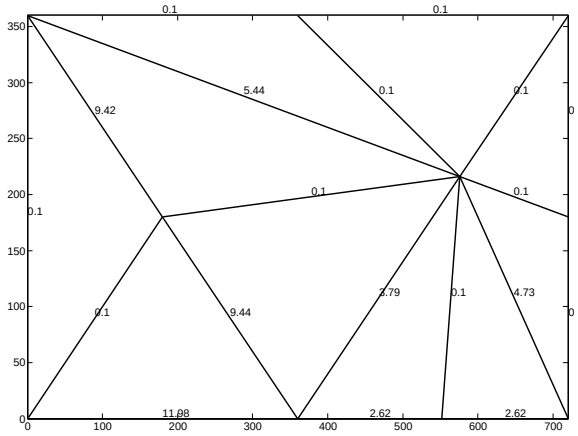

Figure 12: Optimal Design-4

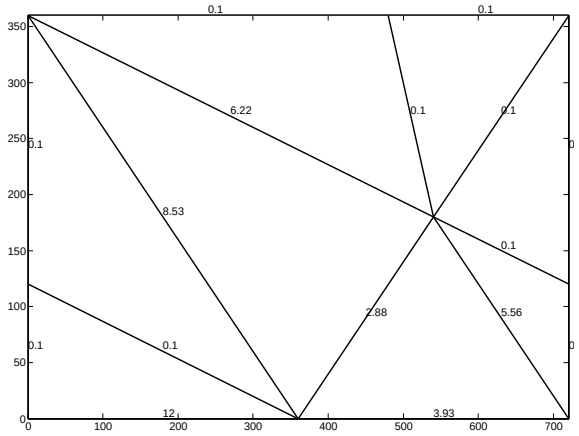

Figure 13: Optimal Design-5

Table 1: Optimal Weight and Area values for the 10 bar Truss Problem

	Area(in^2)				
	Design 1	Design 2	Design 3	Design 4	Design 5
	12	12	12	11.98	12
	2.62	2.67	3.07	2.62	3.93
	3.80	3.76	3.48	3.78	2.87
	9.45	9.42	9.13	9.44	8.53
	5.40	5.36	5.68	5.43	6.21
	4.72	4.68	4.99	4.72	5.56
	2.62	3.77	12	9.41	
	12	2.62			
Weight(lbs.)	1594	1589	1588	1596	1601

8. Acknowledgment
The authors would like to acknowledge the helpful insights offered by Brad Tober (UIUC School of Art and Design) regarding the use of generative algorithms in artistic expression.

9. References

[1] VB Venkayya. Design of optimum structures. *Computers & Structures*, 1(1):265–309, 1971.

[2] Lucien A Schmit and B Farshi. Some approximation concepts for structural synthesis. *AIAA journal*, 12(5):692–699, 1974.

[3] MWi Dobbs and RB Nelson. Application of optimality criteria to automated structural design. *AIAA Journal*, 14(10):1436–1443, 1976.

[4] DE Goldberg and MP Samtni. Engineering optimization via the genetic algorithms. *Computers and Structures*, 40:1321–1327, 1991.

[5] S Rajeev and CS Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–1250, 1992.

[6] Kalyanmoy Deb and Surendra Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite elements in analysis and design*, 37(5):447–465, 2001.

[7] T Hagishita and M Ohsaki. Topology optimization of trusses by growing ground structure method. *Structural and Multidisciplinary Optimization*, 37(4):377–393, 2009.

[8] Prabhat Hajela, E Lee, and C-Y Lin. Genetic algorithms in structural topology optimization. In *Topology design of structures*, pages 117–133. Springer, 1993.

[9] H Rahami, A Kaveh, and Y Gholipour. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures*, 30(9):2360–2369, 2008.

[10] Mathias Giger and Paolo Ermanni. Evolutionary truss topology optimization using a graph-based parameterization concept. *Structural and Multidisciplinary Optimization*, 32(4):313–326, 2006.

[11] SD Rajan. Sizing, shape, and topology design optimization of trusses using genetic algorithm. *Journal of Structural Engineering*, 121(10):1480–1487, 1995.

[12] Richard J Balling, Ryan R Briggs, and Kevin Gillman. Multiple optimum size/shape/topology designs for skeletal structures using a genetic algorithm. *Journal of Structural Engineering*, 132(7):1158–1165, 2006.

[13] A Kaveh and K Laknejadi. A hybrid evolutionary graph-based multi-objective algorithm for layout optimization of truss structures. *Acta Mechanica*, 224(2):343–364, 2013.

[14] M.A. Boden and E.A. Edmonds. What is Generative Art? *Digital Creativity*, 20(1-2):21–46, 2009.

[15] M. Pearson. *Generative Art.* Manning Publications, 2011.

[16] Sivam Krish. A Practical Generative Design Method. *Computer-Aided Design*, 43(1):88–100, 2011.

[17] J. Rieffel, F. Valero-Cuevas, and H. Lipson. Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5):368–379, 2009.

[18] H.C. Pedro and M.H. Kobayashi. On a Cellular Division Method for Topology Optimization. *International Journal for Numerical Methods in Engineering*, 88(11):1175–1197, 2011.

[19] B. Stanford, P. Beran, and M.H. Kobayashi. Simultaneous Topology Optimization of Membrane Wings and Their Compliant Flapping Mechanisms. In *the Proceedings of the 8th AIAA Multidisciplinary Design Optimization Specialist Conference.* AIAA, 2012.

[20] G.S. Hornby, H. Lipson, and J.B. Pollack. Evolution of Generative Design Systems for Modular Physical Robots. In *the Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, volume 4, pages 4146–4151, 2001.

[21] G.S. Hornby, H. Lipson, and J.B. Pollack. Generative Representations for the Automated Design of Modular Physical Robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719, 2003.

[22] J.D. Hiller and H. Lipson. Evolving Amorphous Robots. *Artificial Life XII*, pages 717–724, 2010.

[23] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Springer, 1999.

[24] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., 1999.

[25] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *Knowledge Discovery in Databases: PKDD 2005*, volume 3721 of *Lecture Notes in Computer Science*, pages 133–145. Springer Berlin / Heidelberg, 2005.

[26] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. Mining Graph Evolution Rules. In *the Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD).* Springer, 2009.

[27] A. Bowyer. Computing Dirichlet Tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[28] D.F. Watson. Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[29] S. Fortune. A Sweepline Algorithm for Voronoi Diagrams. In *the Proceedings of the Second Annual Symposium on Computational Geometry.* ACM, 1986.

[30] S. Brooks. Image–based stained glass. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1547–1558, 2006.

[31] Levin, G. Segmentation and Symptom, 2000. http://www.flong.com/projects/zoo/.

[32] G. Varinlioglu, Y. Ipek, O. Balaban, and G. Cagdas. Visualisation of Archaeological Data Using Voronoi Diagrams. In *the Proceedings of the 15th Generative Art Conference*, 2012.

[33] A. Runions, M. Fuhrer, B. Lane, P. Federl, A. Rolland-Lagan, and P. Prusinkiewicz. Modeling and Visualization of Leaf Venation Patterns. *ACM Trans. Graph.*, 24(3):702–711, 2005.

[34] S. Grivet, D. Auber, J.P. Domenger, and G. Melancon. Bubble Tree Drawing Algorithm. *Computer Vision and Graphics*, pages 633–641, 2006.

[35] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer, 1991.

[36] J. Romero and P. Machado. *The Art of Artificial Evolution: a Handbook on Evolutionary Art and Music.* Springer, 2008.

[37] Thorpe, J. tree.growth, 2013. http://www.blprnt.com/treegrowth/.

[38] Coyne, C. Context Free, 2013. http://www.contextfreeart.org/.

[39] T.M. Liggett. *Interacting Particle Systems*. Springer, 2004.

[40] P. Del Moral, L. Kallel, and J. Rowe. Modeling Genetic Algorithms with Interacting Particle Systems. *Revista de Matemática: Teoría y Aplicaciones*, 8(2):19–77, 2012.

[41] D. Shiffman. *The Nature of Code*. Self-published, 2012.

[42] A. Maddox. Interaction Design + Digital Art, 2011. http://archive.anthonymattox.com/.

[43] C. Reas and B. Fry. Processing: Programming for the media arts. *AI & Society*, 20:526–538, 2006.

[44] Reas, C. Process Compendium, 2010. http://reas.com/texts/processcompendium.html.

[45] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S Hanan, F David Fracchia, Deborah R Fowler, Martin JM de Boer, and Lynn Mercer. *The algorithmic beauty of plants*, volume 2. Springer-Verlag New York, 1990.

[46] A Nakamura, A Lindenmayer, and K Aizawa. Some systems for map generation. In *The Book of L*, pages 323–332. Springer, 1986.