LARGE SCALE STRUCTURAL OPTIMIZATION USING GENETIC AND GENERATIVE ALGORITHMS WITH SEQUENTIAL LINEAR PROGRAMMING

BY

ASHISH KUMAR KHETAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Assistant Professor James T. Allison

Abstract

This thesis explores novel parameterization concepts for large scale topology optimization that enables the use of evolutionary algorithms in large-scale structural design. Specifically, two novel parameterization concepts based on generative algorithms and Boolean random networks are proposed that facilitate systematic exploration of the design space while limiting the number of design variables. The presented methodology is demonstrated on classical planar and space truss optimization problems. A nested optimization methodology using genetic algorithms and sequential linear programming is also proposed to solve truss optimization problems. Further, a number of heuristics are also presented to perform the parameterization efficiently. The results obtained on solving the standard truss optimization problems are very encouraging.

To my parents and my sisters, for their love and support.

Acknowledgments

I wish to express my sincere appreciation and gratitude toward Professor James T. Allison for his invaluable guidance and motivation though-out this thesis work. In addition, special thanks to my family and my friends who have helped me through the sometimes trying process of earning my Master's Degree. I also wish to thank my friends Vamsi Talla and Gaurav Chadha who encouraged me to pursue graduate studies.

I am also thankful to Anand, Jeff, and Dan for the wonderful discussions at the Transportation Building.

Last but not the least, I am immensly grateful to the Department of Industrial & Enterprise Systems Engineering and Professor Allison for the continued financial support through Teaching and Research Assistantships, without which this work would not have come to fruition.

Table of Contents

List of Tables

List of Figures

Chapter 1

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, the dimension of the set of continuous design variables changes. This complicates design optimization. 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 design cannot always be defined before problem solution. One well-known solution is to use a ground-structure approach, where a large number of available system elements and their relationships are pre-defined, and the optimization vector specifies the existence (and in some cases geometry and size) 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 [1], are similar in that the number of potential system elements and the available relationships between them are predefined. This thesis presents two new approaches, based on generative algorithms and Boolean random vectors, that overcome these limitations by accommodating variable design dimension problems and allowing the exploration of design alternatives not prescribed a priori. The effectiveness of the proposed methodologies for solving truss design problems with respect to size, geometry, and topology is demonstrated using several archetypal truss design optimization problems.

The first approach provides a novel abstraction concept for truss topology and geometry optimization by means of generative algorithms. Abstraction of topology and geometry implies parameterizing them using an abstract representation. Generative algorithms are a class of representation algorithms that when decoded output a design. The idea is to represent truss topology and geometry using rules of generative algorithms, and to operate on the generative algorithm rules using a genetic algorithm instead of on the design

description directly. A new way of implementing the generative algorithm is also presented that leverages principles of optimal truss development. Further, truss size optimization is performed using a nested optimization routine based on sequential linear programming. The generative algorithm abstraction and nested optimization strategy support concurrent optimization of truss topology, geometry, and size. In addition, this new design strategy 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 hindering innovative design solutions). The generative algorithm abstraction layer also supports structural designs of variable dimension as variable-dimension structures can be generated from the same fixed-dimension rule set. Finally, the effectiveness of the new methodology is demonstrated by examining archetypal two- and three-dimensional truss design optimization problems.

The second approach provides a novel parameterization concept for structural truss topology optimization that enables the use of evolutionary algorithms in design of large-scale structures. The representational power of Boolean networks is used here to parameterize truss topology. A genetic algorithm then operates on parameters that govern the generation of truss topologies using this random network instead of operating directly on design variables. A genetic algorithm implementation is also presented that is congruent with the local rule application of the random network. The primary advantage of using a Boolean random network representation is that a relatively large number of ground structure nodes can be used, enabling successful exploration of a large-scale design space. In the classical binary representation of ground structures, the number of optimization variables increases quadratically with the number of nodes, restricting the maximum number of nodes that can be considered using a ground structure approach. The Boolean random network representation proposed here allows for the exploration of the entire topology space in a systematic way using only a linear number of variables. The number of nodes in the design domain, therefore, can be increased significantly. Truss member geometry and size optimization is performed here in a nested manner where an inner-loop size optimization problem is solved for every candidate topology using sequential linear programming with move-limits. Further, geometry and size optimization are performed sequentially in an iterative manner. Geometry optimization is

performed in a localized region which allows efficient optimization of geometry and size together. The Boolean random network and nested inner-loop optimization allows for the concurrent optimization of truss topology, geometry and size. The effectiveness of this method is demonstrated using a planar truss design optimization benchmark problem.

Chapter 2

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 size 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 kept fixed. In geometric optimization of truss structures, nodal coordinate location are treated as design variables. In truss topology optimization, parameters that govern truss member connectivity are design variables while nodal coordinates are held fixed. All three categories of truss design optimization have been studied extensively. Early efforts in truss size optimization were carried out by Venkayya [2], Schmit and Farshi [3], and Dobbs and Nelson [4]. In addition, Goldberg and Samtani [5] and Rajeev and Krishnamoorthy [6] 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 [7] as well as and Hagishita and Ohsaki [8] using genetic algorithms (GAs). Hajela, Lee, and Lin [9] 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 topology optimization using ground structures does not incorporate ge-

ometry 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 have also been explored to achieve integrated size, geometry and topology optimization. Rahami, Kevah, and Gholipour [10], Giger and Ermanni [11], Rajan [12], Balling, Briggs and Gillman [13], and Kaveh and Laknejadi [14] all used evolutionary algorithms[1] to find the optimal size, geometry and topology of trusses.

2.1 Map L-system Based Approach

In Chapter 3, a new methodology is presented 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 geometry design problem, and an inner-loop sequential linear programming (SLP) implementation that solves the size optimization problem (referred to as Algorithm 1 in later sections). The combined problem of topology, geometry and size optimization is decoupled in two parts. The first part, referred to here as the outer-loop, searches for an optimal topology and geometry using a GA. The second part, referred to here as the inner-loop, solves for optimal truss member sizes using gradient-based optimization algorithm. The outer loop GA uses the optimal value of the inner loop problem as its fitness function for each design candidate as it searches for the optimal topology and geometry. A new way of implementing the GA with the design abstraction is also presented that exploits the underlying methodology of the generative algorithm and properties of the truss optimization problem (referred to as Algorithm 2).

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 [15]. 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 a generative algo-

[1] GAs belong to the larger class of evolutionary algorithms.

rithm that outputs truss topology and geometry based on a set of rules. For each candidate topology, an inner loop algorithm solves the size optimization problem using sequential linear programming. Solving the inner-loop problem permits a fair comparison between candidate topologies because it determines the ultimate utility of each given topology. Adjusting the generative algorithms rules results in the generation of 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, optimization is performed 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 can be solved using existing methods.

The generative algorithm used here is a modified version of cellular division using map L-systems. These algorithms will be explained in more detail in Chapter 3. After a discussion of the basics of these algorithms, a new type of truss design methodology using a modified cellular division method in combination with a genetic algorithm to solve for optimal truss topology, geometry and size will be presented in Chapter 3. This methodology can be used to design a truss for a given arbitrary design boundary. The methodology leverages a basic concept of optimal truss development, namely, truss members are added strategically to redistribute load such that the overall structural mass required to support a given load is reduced. Truss design is developed starting with an initial sparse (but stable) design, and at each stage of development the design domain interior is explored by adding new truss bar members in a way that reduces structural mass, while satisfying stress and displacement constraints. The results of the proposed methodology are demonstrated in Chapter 5, using a pair of two-dimensional benchmark truss design optimization problems, and the extension of this methodology to three-dimensions is demonstrated using a well-known space truss design problem.

2.2 Boolean Random Network Based Approach

In Chapter 4, a new methodology using Boolean random networks (BRN) is presented to solve the combined topology, geometry and size design problem for trusses. Boolean networks, along with their local rules, provide an abstract representation of candidate truss topologies. A nested approach is used here where truss topology is optimized in an outer loop with respect to Boolean network parameters using a genetic algorithm. An inner-loop sequential linear programming (SLP) method solves the geometry and size optimization problem for every candidate topology. In the map L-systems based methodology, the inner-loop solves only the size optimization problem. However, in the new BRN methodology presented in Chapter 4, geometry and size both are solved in the inner-loop using a gradient-based optimization algorithm. A localized geometry optimization method is proposed to solve the inner loop problem efficiently. Further, a new GA implementation for solving the topology design problem by exploiting the underlying local rule application of the Boolean networks and the characteristics of the truss optimization problem is presented. Use of Boolean random networks allows consideration of relatively large numbers of points in the ground structure. This supports the extension to large-scale truss design problems beyond the capabilities of direct design representations.

Boolean random networks are an extension of cellular automata wherein each cell becomes a node that is connected to arbitrary nodes, not necessarily geometrically close neighboring nodes. Cellular automata algorithms have been used in structural layout optimization to achieve global system equilibrium by iteratively updating node states based on a defined nodal neighborhood and a set of local rules. Here Boolean networks are used in a completely different way; the representational power of Boolean networks are used to reduce optimization problem dimension (i.e., number of optimization variables) for a given truss design problem. The Boolean network representation based on a defined neighborhood and a set of fixed local rules outputs truss topology, and an inner loop solves the geometry and size optimization problem for each candidate topology using a sequential linear programming approach. Adjusting the Boolean network neighborhood results in different topologies on a given number of an initial set of nodes. The Boolean network is used as an abstract representation of truss topology. Instead of

optimizing with respect to binary variables as in classical truss topology optimization, optimization is performed with respect to the Boolean network parameters, reducing optimization problem dimension significantly. The binary representation results in an optimization problem dimension that increases quadratically with the number of ground structure nodes, whereas the proposed Boolean representation results in a linear increase. This representation approach supports scaling up to much larger topology optimization problems than what can be solved using the existing direct ground structure representations.

The Chapter 4 is organized as follows. First, a review of how cellular automata has been used thus far in engineering system optimization is provided. Then, the basics of cellular automata and Boolean random networks has been discussed. Following that, a novel methodology using a modified representation of Boolean networks in combination with a genetic algorithm for truss topology, geometry and size optimization is presented. The proposed methodology aims to reduce significantly the number of design variables needed in ground structure methods and supports the use of a large number of initial ground structure nodes. The methodology exploits a basic concept of optimal truss development; each truss member is added to redistribute load such that it reduces the overall mass required to support a given load. Truss topology is developed starting with an initial minimum number of bar members. At each stage of development all nodes are explored in a systematic way to add new bar members to reduce mass while satisfying stress and displacement constraints. The initial set of bar members is selected in a way that connects loads to fixed nodes though a stable truss structure. The results of the proposed methodology are demonstrated in Chapter 4, using one benchmark truss design problem.

Chapter 3

Map L-system Extension to Truss Design

3.1 Map L-systems

Lindenmayer systems, or L-systems, represent a novel type of string rewriting where the rewriting is carried out in parallel. The L-systems were developed by the eminent biologist Aristid Lindenmayer [16]. 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 [17]. The method is binary because cells divide into two during the cell division process. 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.

Mathematically, a map L-system consists of an alphabet Σ, an axiom ω (the initial string), 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 $A \rightarrow \alpha$, 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. Symbols that have pairs of matching brackets around them—i.e., [and]—specify locations for possible cell-dividing walls. These symbols are called markers. 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: $X \rightarrow X$. Examples of rules and their effects on edges are illustrated in Fig. 3.1.

Left: $\overleftarrow{A} \rightarrow \overrightarrow{B} X \overrightarrow{C} [+ \overleftarrow{D}] \overleftarrow{E}$

Right: $\overrightarrow{A} \rightarrow \overrightarrow{D} \overleftarrow{B} X [+ \overrightarrow{D}] \overleftarrow{F}$

Figure 3.1: Examples of rewriting rules

3.1.1 Cellular Division Algorithm

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, 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 processes 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. Figure 3.2 shows the first two development stages of the cellular division process for a non-oriented 'Cartesian' map L-systems defined by:

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

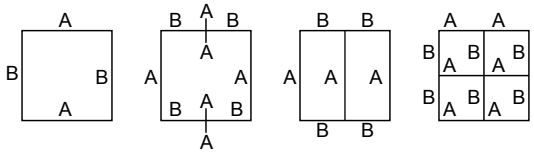

Figure 3.2: Example of cellular division in map L-system

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 horizontal edges A are transformed according to the rule, p_1: $A \rightarrow B[-A][+A]B$; and the vertical edges B are rewritten to edges A according to the rule, p_2: $B \rightarrow A$. We use a global counterclockwise orientation to decide right and left for this representation. Thus the lower edge A is first subdivided into 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. 3.2. 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. 3.2. 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. 3.2. This completes the second development stage.

3.2 Extension to Truss Design

In the proposed methodology a GA operates on cellular division algorithm rules to improve 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). Solving the inner-loop problem helps to determine how good each candidate truss topology is, allowing for a fair comparison between design alternatives [18]. In this section two new algorithms are presented for truss design optimization. In the first algorithm, several design issues that are unique to structural trusses are addressed, and the generative algorithm is executed completely for each individual in a GA population. In the second algorithm, a sequence of GA problems are solved where each problem involves the addition of just one set of bar members to the truss using a single generative algorithm development stage.

3.2.1 Algorithm 1: One-Step Generative Algorithm

The cellular division algorithm outputs a developed map given an initial map, a set of rules, and a specified 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 a link must be established between elements of the generated map (cells, edges and vertices) and the physical system topology. For truss design optimization, cellular division provides an intuitive link where the edges of the topology represent truss bars, and edge intersections represent pin joints. This provides an abstraction for topology and geometry of truss design via cellular division algorithm rules.

While a clear connection exists between maps generated using cellular division and truss design, a randomly selected 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 edge intersections as pin joints, may correspond to a mechanism instead of a stable truss structure. One standard remedy would be to evaluate stability of each generated design (e.g., checking for singular stiffness matrices), and then penalizing unstable designs in the genetic algorithm implementation. Initial studies performed

by the authors revealed that a large portion of designs generated using a standard cellular division algorithm resulted in mostly unstable designs. This was not an issue in previous studies that addressed frame design since joints could resist moments [19], but attempting to explore truss designs using standard cellular division algorithms that produce unstable structures is very inefficient. The new methodology presented here takes an alternative approach where a modified cellular division method based on map L-systems ensures that generated maps automatically satisfy truss stability requirements (i.e., no mechanical degrees of freedom). In other words, any arbitrary rule-set will output a stable truss topology. This implicit stability requirement satisfaction supports efficient truss design space exploration, whereas standard cellular division is impractical for use in truss design as it concentrates exploration efforts on infeasible designs. The details of the implicit stability requirement satisfaction strategy are discussed later in this section.

In the new methodology, the GA genotype is the rule-set, and the phenotype is the truss topology and geometry. The GA operates on the genotype, and the generative algorithm maps the genotype to phenotype. Design fitness (objective) and constraints are evaluated based on the phenotype. This approach for using a GA where the design is indirectly encoded more closely mimics evolution of real biological systems than conventional GA implementations with directly encoded designs. More specifically, an organism's DNA is not the organism; rather, it contains compactly encoded instructions for the growth of an organism. A variety of biological mechanisms produce fantastic complexity based on the information encoded in an organism's DNA. To illustrate this concept, consider the human genome. It has approximately 30,000 genes, but these genes produce more than one million human gene products (such as proteins). The biological mechanisms of transcription and translation increase the complexity of gene products beyond the level of complexity found in the human genome. If organism genes were direct encodings of the corresponding phenotype, the genotype would need to be magnitudes larger. In essence, every detail of an organism would need to be micromanaged through its genomic encoding, and the resulting genomic complexity would limit organism sophistication and the power of design exploration via evolution.

The mechanisms that map biological genotype to phenotype are pivotal to the success of biological evolution; without them, the richness and diversity

of life on Earth would not be possible [20]. Similarly, without a sophisticated mapping from genotype to phenotype for engineering designs, artificial evolution would be limited in the complexity of designs that could be explored. Put another way, using direct encoding fundamentally limits the complexity of designs that can be explored an optimized using GAs. Most GA implementations to date have used direct encodings. As a result, engineers have largely been unable to realize the full power of GAs. When attempting to apply directly-encoded GAs to problems of significant complexity, the process often breaks down, and these algorithms are unable to converge to meaningful results [21]. Utilizing a mapping between genotype and phenotype, however, results in an evolutionary process that is closer to what is found in biological systems, and is more successful when attempting to solve large-scale design problems.

The objective in the design problem here is to minimize system mass, subject to stress and displacement constraints. For every candidate topology considered by the GA, the inner-loop optimization problem solves for truss member cross section areas that minimize structural mass, while satisfying stress and displacement constraints. The optimal mass as computed by the inner-loop is used in calculating the fitness for the GA. In the inner loop, each individual is solved for a fixed number of SLP iterations. Each linear programming problem resulting from SLP formulation is solved using the MATLAB® interior point method, which ensures that each GA individual and its associated fitness value represents a truss design that satisfies all design constraints. At termination, the genetic algorithm produces the truss topology and geometry that has the (approximately) minimal mass when the inner-loop is applied to identify optimal truss member cross section areas.

This procedure is summarized in the form of pseudocode in Algorithm 1. An initial truss topology and geometry description, which is denoted by the variable $\mathbf{TS^0}$, is the algorithm input. The algorithm output is the optimal topology and geometry ($\mathbf{TS}^*$), and the optimal member cross section areas ($\mathbf{A}^*$). The objective function $f(\cdot)$ calculates truss mass given topology, geometry, and size specifications. Given $\mathbf{TS^0}$, the algorithm produces $\mathbf{TS}^*$ after k developments (including observance of stress and displacement constraints). In other words, the genetic algorithm outputs the optimal rule set which when applied for k developments on the initial topology and geometry will produce truss topology and shape that yields the minimal mass design.

As explained above, the size optimization is performed in a nested approach using an SLP strategy. This inner-loop (SLP) provides the GA fitness value for each truss topology and geometry in a GA population, as represented by a candidate rule set.

Algorithm 1: Truss design using One-step Generative Algorithm

Input: Initial topology and geometry $\mathbf{TS^0}$
Output: Optimal topology and geometry, $\mathbf{TS}^*$; and size, $\mathbf{A}^*$
1: $[\mathbf{TS}^*, \mathbf{A}^*] = \arg\min_{\{(\mathbf{TS}, k-developments|\mathbf{TS^0}), \mathbf{A}\}} f(\mathbf{TS}, \mathbf{A})$, Subject to stress and displacement constraints

Initial studies utilizing Algorithm 1 revealed that completing all development stages for each individual in a GA population lead to many designs that were self-similar. This limited how effectively the design space could be explored. A variant on this design methodology was developed to address these limitations, where only one development stage was utilized at a time, but a sequence of GA problems was solved. This second methodology, referred to here as Algorithm 2, performed much better in practice and resulted in more systematic design space exploration where each truss development stage led to mass reduction. Algorithm 2 is described in detail in the next subsection.

3.2.2 Algorithm 2: Multi-Step Generative Algorithm

An alternative strategy was created and investigated where the generative algorithm is used for only one development stage within the GA, but multiple GA problems are solved in sequence. Each new GA problem in the sequence uses the solution of the last GA problem as its starting map. This strategy harnesses a basic principle of optimal truss development—each truss member is added to redistribute load such that it reduces the overall structural mass required to support a given load—to frame an entirely novel methodology of truss development. In addition, this is a completely new way of utilizing generative algorithms for engineering design. Because the number of cellular division development stages is restricted to one, each GA solution adds at most one bar member to each cell of the map with the objective of reducing the mass of the overall structure. The next truss development stage is carried out by solving the next GA problem, which takes the optimal topology and geometry obtained in the previous stage as the initial map, and repeats the

process to further reduce mass. In this manner this methodology explores the design space systematically to reduce mass by adding bars in optimal locations. The truss design is developed with each subsequent development stage (GA solution) until mass cannot be reduced further by adding new bar members. Figure 4.10 illustrates this methodology graphically.

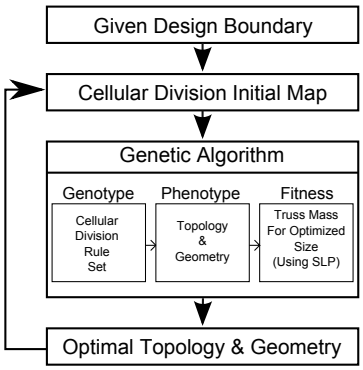

Figure 3.3: The proposed sequential GA methodology

This procedure is summarized in the form of pseudocode in Algorithm 2. As with Algorithm 1, the input is also initial topology and geometry ($\mathbf{TS^0}$), and the output is the optimal topology and geometry ($\mathbf{TS}^*$) and optimal size values ($\mathbf{A}^*$). In Algorithm 1, a single GA was solved. This algorithm, however, develops topology and geometry in successive GA solutions that each develop the truss design only partially (i.e., the generative algorithm is executed for just one development stage when solving the GA). The stepwise addition of new bar members is performed until the minimum mass design from successive GA solutions stops decreasing (i.e., $(f(\mathbf{TS^{k+1}}, \mathbf{A^{k+1}}) \geq f(\mathbf{TS^k}, \mathbf{A^k})))$. Adding members at strategic locations at each step reduces mass initially due to load redistribution, but due to lower size limits on members the mass will at some point begin to increase.

Algorithm 2: Truss design using Multi-step Generative Algorithm

Input: Initial topology and geometry $\mathbf{TS^0}$
Output: Optimal topology and geometry, $\mathbf{TS}^*$; and size, $\mathbf{A}^*$
1: Set $k = 0$
repeat
  1.1: $k \leftarrow k + 1$
  1.2: $[\mathbf{TS^{k+1}}, \mathbf{A^{k+1}}] = \arg\min_{\{(\mathbf{TS}, 1-development|\mathbf{TS^k}), A\}} f(\mathbf{TS}, \mathbf{A})$,
  Subject to stress, displacement constraints
until $(f(\mathbf{TS^{k+1}}, \mathbf{A^{k+1}}) \geq f(\mathbf{TS^k}, \mathbf{A^k}))$

Table 3.1: Genomic encoding vector

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

The following subsection explains mathematical genotype representation of the rule set and modifications to Map-L systems for truss design. In addition to ensuring stability, these modifications facilitate load redistribution that leads to optimal mass reduction.

3.3 Genomic Encoding of Cellular Division Rules

The GA used here acts upon the individual genes that encode all the information required for generating 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 mixed vector of reals and integers, X, whose real elements are in the interval $[0, 1]$, whose integer elements are in the set $\{1, 2..., z\}$, where z is the number of letters in the alphabet. Table 3.1 exemplifies the structure of X, where superscripts indicate what axiom or production rule element the genome component corresponds to.

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. As explained above, letters of the alphabets are encoded as integers $\{1, 2.., z\}$, where z is the cardinality of the alphabet Σ. A similar approach is followed for production rule encoding, where each rule is encoded according to a master rule of the form:

$$\sigma_i \rightarrow X_1^{P_i} X_2^{P_i} ... X_{m-1}^{P_i} X_m^{P_i}, \tag{3.1}$$

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 only an alphabet or an alphabet and a marker. To incorporate all possibilities in a token, all tokens are encoded as a vector of two real and two integer 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

Table 3.2: Production rule vector $X_1^{P_i}$

Blank Token	Edge-Letter	Read-Marker	Marker-Letter

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. Table 3.2 shows structure of the vector $X_1^{P_i}$.

The first element encodes whether or not the token is a blank space. The remaining three 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. The first and third elements—blank token and read-marker—are encoded on an interval of $[0, 1]$. If the value of the first element is less than or equal to 0.2, the token is blank (otherwise it is non-blank). Thus, blank tokens are less likely than non-blank. A marker follows an edge alphabet if the third element of this vector is less than or equal to 0.8. The second and fourth elements—edge-letter and marker-letter—are encoded as integers from the set $\{1, 2..., z\}$, where z is the number of letters in the alphabet. Integer coding of edge and marker letters greatly improves GA exploration effectiveness.

After decoding, the genome can be seen as a partitioned array of symbols whose first part is occupied by n letters that compose the 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.

The method presented here uses non-oriented map L-systems, whereas standard map L-systems are oriented (see the arrows in Fig. 3.1). Another significant modification used here relates to how markers are read. In standard L-systems, markers are labeled with $+$ or $-$ symbols to indicated whether marker is placed to the right or to the left of the predecessor edge. This restricts how cellular divisions may be made. Here a marker on an edge may be used for cell division on either side of the edge. This modification increases the overall number of cell divisions. Figure 3.4 shows the first development stage of the cellular division process in modified map L-systems

defined by:

$$\Sigma = \{A, B, C, D\}$$
$$\omega = BCBDA$$
$$P = \begin{cases} A \to B[D]C[B]D, B \to B \\ C \to C[B]D, D \to A[D]C \end{cases}$$

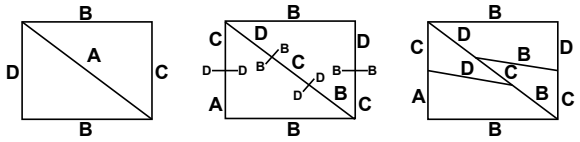

Figure 3.4: Example of cellular division in modified map L-system

3.4 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 for an arbitrary axiom and set of production rules. To illustrate this concept, consider the ten randomly generated truss topologies shown in Fig. 3.5. Most of them do not represent a stable truss design, i.e., they are mechanisms. These topologies were generated using four development stages on a set of four alphabets with six tokens and an axiom of four edges. The intervals used 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 that were always stable regardless of topology as long as load and support nodes were connected [19]. The assurance of stability greatly simplified the design problem. This motivates a modification to the cellular division algorithm to permit application to truss design by ensuring the generation of stable trusses.

Stability of an arbitrary truss topology and geometry can be fully ascertained by checking the singularity of the corresponding stiffness matrix. This metric, however, cannot be incorporated into a generative algorithm. One

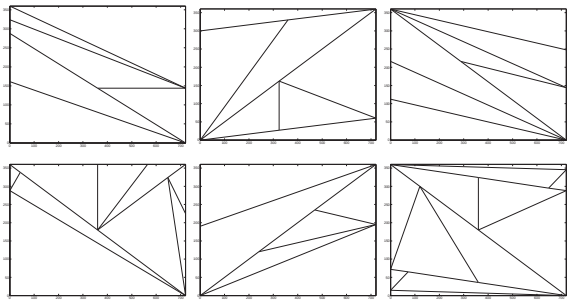

Figure 3.5: Randomly Generated Truss Topologies

could check trusses after generation using their stiffness matrices, and either 'repair' unstable trusses, or penalize the fitness of unstable truss topologies to guide the design exploration away from them. 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. A more efficient strategy is used here where only stable designs are explored. 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 a set of constraints is introduced that, when enforced upon the cellular division process, ensures stability of resulting truss topologies after each stage of map development. The axiom map is interpreted as a tessellation of triangles, and the cellular division process is restricted such that a division takes place only if it divides the cell into two triangles. The stability of such trusses is guaranteed and need not be checked using the stiffness matrix. For this purpose, two matching markers are connected only if one of them is at a vertex. Figure 3.6 depicts a set of randomly generated truss topologies using the modified cell division process, all of which are stable. These topologies were generated using the same algorithm parameters used for the (unstable) topologies illustrated in Fig. 3.5.

After exploring stable truss designs generated using this modified algorithm, an additional observation was made that led to another useful algorithm modification. Often a cell was divided such that it created a bar that it connected to the middle of another bar (see the downward-sloping bar just below the top bar in Fig. 3.7). If a pin joint is created at this connection, the newly added bar cannot exert much force at the joint because the other two bars are 180° apart. This limits overall load redistribution, and limits the

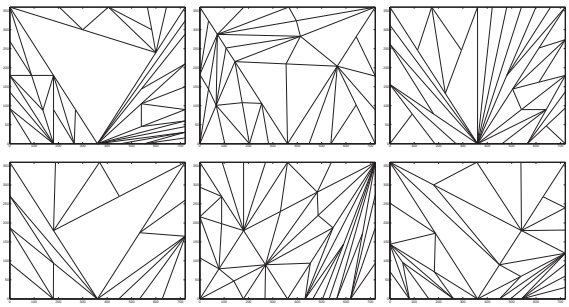

Figure 3.6: Randomly Generated Stable Truss Topologies

benefit of adding this new bar. However, if the adjoining cell is also divided at the same marker location by connecting with its opposite vertex (as in Fig. 3.8), it leads to substantial load redistribution and mass reduction after inner-loop optimization. The optimal mass for the design in Fig. 3.7 is 7099 lbm. (after applying the SLP size optimization strategy), whereas the optimal mass for the design in Fig. 3.8 it is 5476 lbm. The solid lines represent bars under tension, and the dotted lines represent bars under compression. Line thickness is a linear function of the cross-section area.

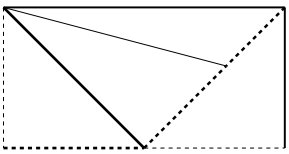

Figure 3.7: Ten Bar Truss- Adjoining Cell Not Divided

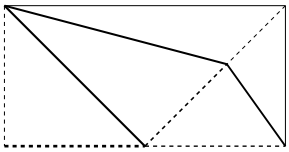

Figure 3.8: Ten Bar Truss- Adjoining Cell Divided

3.5 Extension to 3D Truss Design

A core advantage of generative design strategies is the ability to scale-up to the design of highly complex systems. This is primarily due to generative algorithm abstractions that support exploration of complex designs using only a small set of rules as design variables. This section details how the

methodology introduced in this article can be extended to the design of three-dimensional trusses, which involve additional complexities not present in two-dimensional trusses.

The starting point is an initial design with labeled edges (i.e., the axiom). The production rules are then applied to divide edges and add markers, and then matching edges are joined to produce cell divisions. However, to ensure truss stability upon cell division, only divisions that produce tetrahedral structures are permitted. A truss that is composed of tetrahedra is guaranteed to be stable. The initial map is a composition of tetrahedra, and new tetrahedra are formed by taking an existing tetrahedron, and connecting an interior point of one of its edges with its two opposite vertices. This type of division occurs if a marker at one of its edges matches markers on opposite vertices. In addition, design exploration is guided toward trusses with better load distribution by mirroring divisions in the adjoining tetrahedra (similar to what is done in the 2D case). More precisely, if the edge being divided is not on the external boundary of the truss, the adjoining tetrahedra are also divided at the same marker location with their opposite vertices being connected to that marker location. Figure 3.9 illustrates the division of a tetrahedron by joining an internal point of one of its edges with the opposite vertices (the mirroring operation is not illustrated).

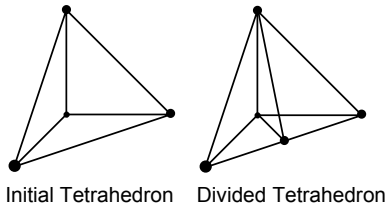

Figure 3.9: Tetrahedron Sub-division

3.6 Truss Optimization Problem Formulation

The new methodology is formulated mathematically to solve truss design problems. The problems involve finding the optimal topology, geometry and size of the circular cross-section bars in a given design space boundary. Each truss member size (cross-sectional area) is a continuous variable with lower and upper bounds. A general problem formulation for the concurrent optimization of size, geometry, and topology can be represented mathematically

as:

$$\begin{aligned} \min_{n,\mathbf{C},\mathbf{A},\mathbf{P}} \quad & f = \sum_{\substack{0\le i\le n \\ 0\le j\le n}} \rho C_{i,j} A_{i,j} l_{i,j} \\ \text{Subject to:} \quad & \sigma_{\min} \le \sigma_{i,j} \le \sigma_{\max} \\ & d_{\min} \le d_k \le d_{\max} \end{aligned} \tag{3.2}$$

where n is the number of nodes (joints), $C_{i,j} \in \{0, 1\}$ indicates whether nodes i and j are connected, $\mathbf{P}_k = [x_k, y_k, z_k]^{\mathrm{T}}$ is the position vector for each node $k = 1, 2, \ldots, n$, and $A_{i,j}$ is the cross-sectional area of the bar that connects nodes i and j. The material density is ρ, and several quantities are functions of design variables, including the length of the bar that connects nodes i and j ($l_{i,j} = \|\mathbf{P}_i - \mathbf{P}_j\|_2$), the axial stress of the member connecting nodes i and j ($\sigma_{i,j}$), and the displacement of each node ($d_k, k \in \{1, 2, \ldots, n\}$) (computed here using the force method). The minimum allowable stress is $\sigma_{\min}$ ($\sigma_{\min} < 0$, indicating compression), and the maximum allowable stress is $\sigma_{\max}$ ($\sigma_{\max} > 0$, indicating tension). Similarly, the minimum and maximum allowable node displacements are $d_{\min}$ and $d_{\max}$, respectively. The objective of the design problem is to minimize overall structural mass.

The above optimization problem is equivalent to finding the optimal topology, geometry, and size of a truss given the load, design space boundary, stress, and displacement constraints. Please note that while this formulation implies simultaneous solution of topology, geometry, and size, the problem was solved here using the nested generative algorithm approach introduced in the previous sections. The outer loop searches for the best topology and geometry, while the inner loop solves the size optimization problem in a way that satisfies stress and displacement constraints. The generative algorithm rule set is optimized using the MATLAB® genetic algorithm (`ga`) function. The inner-loop is solved using a custom SLP implementation based on the the MATLAB® function `linprog`, described in more detail below.

The initial truss (map) for each design problem is created by assuming a truss bar member exists for each edge of the boundary, and then adding new members inside the design boundary such that it forms a summation of triangles (two-dimensional truss) or a summation of tetrahedra (three-

dimensional truss). At each development stage, all the edges of the initial map are labeled according to the axiom letters. Production rules are applied on each edge. Then, in the case of the two-dimensional truss problem, the vertices of each triangular cell is scanned for a matching markers on opposite edges for a possible division. To avoid skew angles between truss members, a division is considered to be feasible only if the none of the resulting cells has area that is less than one-fifth of the original cell.

For the division of a non-internal edge, no further conditions are checked and the cell is divided in two. However, if the edge to be divided is an internal edge, the adjacent cell is also divided at the same marker location with its opposite vertex (this is the mirroring operation described above). For this to happen, the adjacent cell must not have been divided already in that stage of development since a basic rule of cellular division is that a cell can only be divided once in a development step. If more than one division is possible, ties are broken arbitrarily. Specifically, the algorithm searches for matching markers that satisfy the above conditions to identify edges that can be divided. If more than one edge exists, the first one found is chosen. A different tie-breaking strategy could be used if desired.

Similarly, in the case of three-dimensional truss design, each pair of vertices in each tetrahedron is checked for a matching marker on opposite edges, and a division is considered to be possible only if none of the resulting tetrahedra has volume less than one fifth of that of the original tetrahedron. In three dimensions more than one tetrahedron can be adjacent to an edge. If this is the case, all adjacent tetrahedra are divided in the mirroring operation (for divisions involving non-internal edges). If any adjacent tetrahedra have already been divided, the division is considered to be infeasible. Finally, the following map L-system parameters were used in the generation of truss topology and geometry:

- The number of development stages $n = 1$.
- The alphabet Σ contains 6 letters.
- The production rules have 12 tokens.
- The blank token threshold is 0.2.
- The marker after a division-edge threshold is 0.8.

The inner-loop problem is solved using sequential linear programming (SLP), which is a recursive procedure that involves the formulation and solution of a series of linearly approximated sub-problems, where each intermediate solution is the starting point for the subsequent sub-problem. The inverse of cross-sectional areas is used as the size optimization variable; this introduces a nonlinearity into the objective function, but reduces the nonlinearity of the constraint equations. Cross-sectional area values are bounded above and below.

When a basic SLP strategy is used, and when both stress and displacement constraints are included, SLP often did not coverage during initial tests. This was remedied by using move limits. In SLP, the linear approximation of the nonlinear size optimization problem is only accurate close to the linearization point. Move limits reduce potential error, improving convergence properties. A simple move limit strategy was used where at each SLP iteration, the search domain is limited to the intersection between the linearized constraint domain and a parallelepiped around the linearization point.

Large move limits may result in oscillations during numerical problem solution, and move limits that are too small will yield a slow convergence rate, and may cause the algorithm to accumulate to a local optimum. Wujek [22] argued that a proper move limit choice should ensure that the objective function is always decreasing, that the intermediate solutions are always feasible, and that the optimization variable movement is controlled to maintain approximation error at a reasonable level. Lamberti and Pappalettere's [23] partial modification of Chen's [24] Constraint Gradient-Based Move Limit (CGML) algorithm is used in the case studies presented here. In particular, scaled move limits that are equal for all optimization variables are used. Lamberti recalculated the move limits until the intermediate solution improved meaningfully, and then they reduced using a user-supplied factor. The maximum number of SLP iterations used here is 50. Each linear sub-problem is solved using MATLAB® `linprog` function using the interior point method (ensuring each sub-problem the solution is feasible). It should be noted that all the results reported in the sections below are generated using Algorithm 2. In Chapter 5, results of three archetypal truss design example problems solved using the above formulation is presented.

3.7 Discussion

In summary, this abstraction concept using map L-system gives a set of parameters that, when decoded using a cellular division algorithm, outputs both truss topology and geometry. The coupling of topology and geometry where both are represented together by a set of parameters restricts design space exploration. To overcome this limitation of the above map L-system approach, the next chapter introduces an alternative abstraction concept using Boolean random networks that represents topology alone. In this alternative formulation, geometry is optimized along with size in the inner loop, while the outer loop optimizes only topology. It is expected that the alternative formulation will cover a larger design space and give better results in terms of optimal truss design.

Chapter 4

Boolean Random Networks Extension to Truss Design

This chapter presents an alternative truss topology abstraction using random Boolean networks. Random Boolean networks are an extension of cellular automata. The section below explains the basics of cellular automata, followed by an introduction to random Boolean networks. The next section introduces a methodology for truss topology abstraction using random Boolean networks. Finally, a parameterization of random Boolean networks is presented that supports truss topology optimization by adjusting the parameters that govern network generation using random Boolean networks.

4.1 Cellular Automata

The behavior of cellular automata (CA) is governed by a set of local rules that when applied iteratively produce complex global phenomena. Typical engineering implementations of CA are based on the decomposition of a domain governed by physical laws into a set of regular cells that form a uniform lattice. Decision-making is implemented by rules at the cell level. Rules are functions of the neighboring cells, and the cell itself. The information used to update each cell is local by nature and is taken from its neighborhood only. By repetitively and simultaneously applying the local rules to each cell to update the associated physical quantity, the CA process converges to a global description of the system. Since CA relies only on local information for updating system state, system-level governing equations are not required. CA, therefore, is considered to be very effective for simulating physical phenomena whose governing equations are unknown.

The introduction of the CA is generally attributed to the works of Von Neumann in the early 1950s. More recently, cellular automata was revisited in 1994 by Wolfram [25]. It has been used frequently by physicists to describe

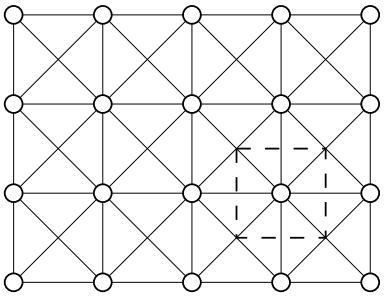

Figure 4.1: A ground truss lattice

systems of particles. There have been many studies of its application to engineering systems and structural mechanics as well. The application of the cellular automaton to geometry optimization has been presented by Inou et al. [26, 27], Kundu et al. [28, 29], Xie et al. [30–33], Zhao et al. [34, 35], Yang et al. [36], Young et al. [37] and Kim et al. [38]. In these studies, the design domain is divided into many small cells and the von Mises equivalent stress distribution on the whole domain is estimated using a finite element method (FEM) approximation. Then, the reference stress at each cell is updated by applying a local rule to the stress distribution. The Young's modulus for each cell is treated as a design variable. It is modified so that the equivalent stress at the updated cell is to be equal to the reference stress. The cells with relatively small Young's modulus are removed which leads to the modification of the geometry and topology of the structures. Other studies involving the use of CA for structural design optimization include Kita et al. [39], Gürdal et al. [40], Tatting et al. [41], and Abdalla et al. [42].

Cellular automata is based on the decomposition of a physical domain into regular cells forming a lattice. This domain can be a ground truss or a continuum domain in structural mechanics. As mentioned above, the cells in cellular automata receive information from their neighboring cells only. This leads to the definition of several possible neighborhoods characterized by the number and the location of the surrounding cells. The neighboring cells are typically located along the eight cardinal coordinates (N, S, E, W, NE, NW, SE, and SW). Moore and Von Neumann neighborhoods are the two most commonly used neighborhoods. For example, for a ground truss lattice description of a single cell, the Moore and the Von Neumann neighborhoods are illustrated in Figs. 4.1–4.3.

The state of a cell represents values of the response quantities of the physical domain associated with that cell (e.g., stress, displacement). Cell state

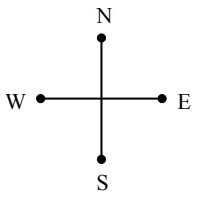

Figure 4.2: The Moore neighborhood

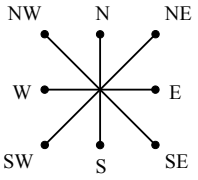

Figure 4.3: The Von Neumann neighborhood

may also involve domain properties, such as thickness and cross-sectional area in the case of trusses. The state of a particular cell at the center of a neighborhood is therefore represented by a set of quantities, which is a function of the neighboring cells states and the external forces applied to that cell. The state of all the cells is updated simultaneously by local rules which are derived based on the physical laws that govern the system or are heuristic in nature.

4.2 Random Boolean Networks

Rather than implement cellular automata in the usual one dimensional or two dimensional array format, it is possible to consider a network where each cell is a node that is connected to arbitrary other nodes, not geometrically close neighboring nodes. Similar to the cells in automata, where each cell is associated to a set of quantities which represent its state, the nodes here can be considered in various states. The state of the nodes is updated synchronously at each time step by a local rule in accordance with the other nodes to which an individual node is connected. If each node is allowed to be in only two possible states, 0 and 1 (off and on - Boolean), such a network is known as a Boolean network. To enhance the dynamics of the network, if each node is allowed to operate under its own rule picked at random, the generalization of cellular automaton is known as random Boolean network. Figure 4.4 depicts a random Boolean network that has 6 nodes, where each node is connected to 3 other nodes. The state of each node is updated based

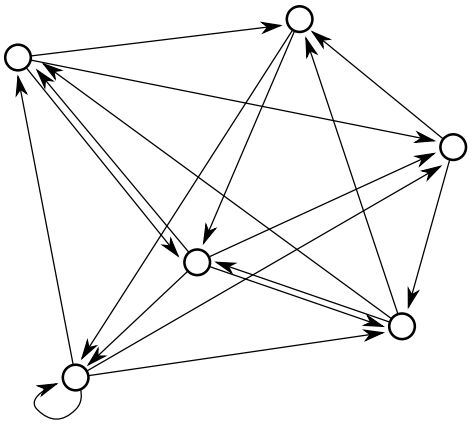

Figure 4.4: A random Boolean network

on the states of the other nodes from which it has incoming arrows. In cellular automata the state of a cell depends upon the states of the cells which are in its physical neighborhood, whereas in random Boolean networks the concept of physical neighborhood is not used. Node states instead can depend on any other node in the network as long as a correctly-oriented connection exists. In other words, neighborhood is not based on the geometric closeness of the nodes, but can be assigned in an arbitrary manner.

If each node has K neighbors, then there exist 2^K possible neighbor state combinations, and hence 2^{K^K} possible Boolean rules can be formed. For example, if each node receives inputs from $K = 2$ other nodes, there are $2^{2^2} = 16$ possible functions to choose from for each node. Further, with two possible states of 0 and 1 for each node, and if there are N nodes, then the number of possible unique network states is 2^N; this is the size of the state space. These models are also known as NK (or Kaufmann) networks. It has been demonstrated that if $K \geq 3$, networks exhibit chaotic behavior.

4.3 Extension to Truss Topology Design

In standard implementations of random Boolean networks, node states are updated simultaneously at each time step by applying local rules that depend on the state of connected nodes. This iterative node update strategy supports the efficient exploration of an otherwise exponentially large state space.

Some existing strategies for generating a variety of different network states (topologies) is to vary either the number of local rule iterations, or to adjust the local rules. Here an alternative method for network topology exploration is proposed where neighborhood definitions are changed instead of adjusting the set of local rules or number of iterations. We hypothesize that the exponentially large network state space can be explored by varying the neighborhood of each node while keeping the set of local rules fixed. Further, we seek to optimize the selection of the neighborhood for each node using a genetic algorithm to obtain the optimal truss topology. We propose a set of local rules and a parametrization of the Boolean random network that can be efficiently optimized using the genetic algorithm. In standard random Boolean networks, the rule set is selected randomly for each node in the network. Here the neighborhood of each node is varied instead, and thus use the term Boolean random network (BRN) instead of random Boolean network (RBN).

The standard ground structure method for truss topology optimization begins with the definition of a densely-connected ground structure topology. A discrete optimization algorithm, such as a genetic algorithm (GA), is then used to determine which of the available truss elements defined in the ground structure are used in the final optimal design. In most GA implementations a direct binary encoding is used where $x_i = 1$ indicates that the ith element exists, and $x_i = 0$ indicates that it does not. This direct encoding results in a quadratic increase in optimization problem size with the number of nodes, making design of large-scale structures impractical. In addition, this direct encoding does not prevent the GA from exploring truss designs that are topologically infeasible (e.g., disconnected or structurally unstable).

The approach proposed here addresses the shortcomings of direct GA encodings operating on ground structures, more specifically, 1) the inability to scale up to large-dimension problems, and 2) the inefficiency of exploring infeasible design topologies. This is accomplished using a generative growth strategy, and by embedding design requirements within the generative algorithm. Instead of starting with a densely-connected ground structure, the method begins with the definition of a minimally connected stable structure. A BRN algorithm is then used to add new members in a way that guarantees stability. Instead of operating directly on design variables that describe the existence of truss members, the GA operates on BRN neighborhood defini-

tions. This abstraction supports scaling up to larger-dimension problems, and is an indirect GA encoding where the genotype (the BRN neighborhoods) is mapped to the phenotype (truss topology definition) through the application of the BRN algorithm. When using direct GA encodings, the genotype equals the phenotype.

This approach is similar to the map L-systems based methodology explained in the previous chapter as it also provides an abstraction of truss design variables. However, this approach decouples the abstraction of topology and geometry. This coupling was a limitation of the map L-system based abstraction. Decoupling of geometry and topology allows for improved coverage of the design space, and hence gives better results in terms of truss design performance. In this approach, truss geometry is optimized along with size in the inner loop.

Any node in a truss design is stable (or fixed) if it is connected to two or more fixed nodes. A node is fixed if its motion is fully constrained by direct or indirect connections to stable nodes. If all the nodes in a truss design are stable than the truss is stable. Using these concepts, a minimally connected truss is framed by connecting all the loading nodes to at least two support nodes. An example of such a truss is illustrated in Fig. 4.5, where the two nodes on the left are support (anchor) nodes, and the bottom center and right nodes are loaded with downward forces. This truss is a starting point for a design problem modeled after a canonical 10-bar truss design problem. Please note that this problem will be referred to as the '10-bar' problem due to the problem it is based on, but design solutions may not necessarily have 10 bars since topology optimization is being performed.

New truss designs are generated by adding new members to the initial minimally connected stable truss. New members can be connected between nodes that are either already part of the existing truss, or are nodes defined on a grid not already connected to the truss. If a non-connected node is connected to at least two stable nodes, a stable truss results. While connecting a non-connected node to exactly two stable nodes will produce a stable truss, it does not enhance truss design as it does not redistribute load (which is required to reduce truss mass). The solution used here is to always connect a new node to three stable nodes. This is illustrated in Figs. 4.6 and 4.7. In Fig. 4.6 a new node was connected to the starting truss from Fig. 4.5 through only two nodes. In Fig. 4.7 three connections to stable nodes are made. It

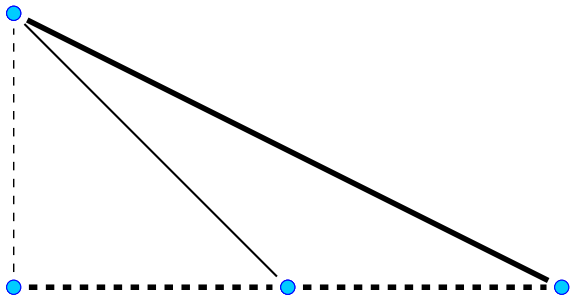

Figure 4.5: Ten bar minimally connected truss

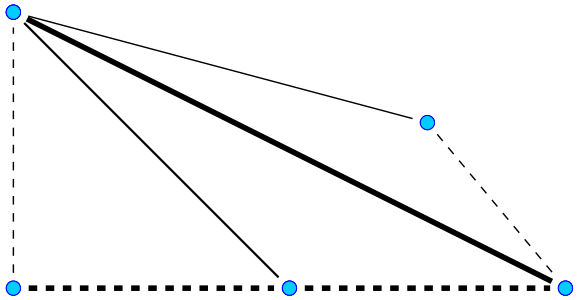

Figure 4.6: Ten bar truss- with two new connections

is clear that the doubly-connected new node will not help transfer any force from the loaded nodes to the support loads, whereas the new node with a triple connection will. The importance of load redistribution in mass reduction can be verified by solving the inner-loop problem (mass minimization with respect to bar sizes using SLP) for each of these three topologies. The results are summarized in Table 4.1. Adding a node with a triple connection leads to significantly lower mass after re-optimizing truss member sizes. In the above example the overlapping members of the truss are removed in a systematic way that will be explained in a later section.

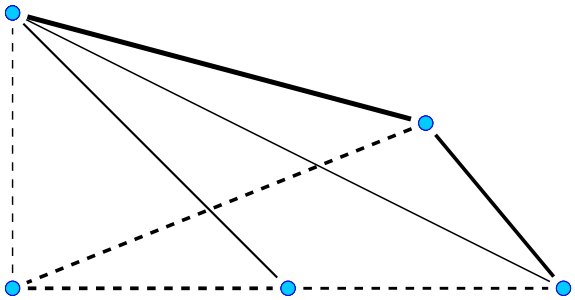

Figure 4.7: Ten bar truss- with three new connections

Table 4.1: Ten-bar Truss- Optimal Connections

Configuration	Optimal Mass (lbm.)
Minimally connected design	5689
Additional node- 2 Connections	5689
Additional node- 3 Connections	4841

4.3.1 Parametrization of Boolean Random Networks

The method described conceptually above can be formulated systematically using Boolean random networks. BRN as explained above has a set of nodes, where each node has its own neighborhood, a set of nodes to which it is connected, and a set of local rules. Local rules are applied simultaneously to all nodes to update network state. For the purpose of topology optimization here, the neighborhood of each node needs to be encoded in a way such that it can be varied efficiently. If nodes are numbered from 1 to N and K neighbors are defined for each node, then a total of NK variables are required. An alternative strategy is to partition the N nodes into K groups, where nodes with a group are considered to be neighbors. All the nodes in the network, including connected and non-connected, are assigned to one of the K groups. This encoding strategy requires only N variables, and is the approach used here due to its representational efficiency.

The local rule is defined here such that if a node has more than three stable nodes in its neighborhood, then the node is connected to those stable nodes. If the node is not connected to the network prior to applying the rules, the node becomes connected and hence stable. If the node is already connected, it makes new connections and remains stable. In other words, if a group has a minimum of three stable nodes, then all the non-connected nodes of the group would be connected to the three stable nodes and made stable. If a group has more than three stable nodes in it then all the nodes, including the earlier stable nodes, of the group would be connected to any of the three stable nodes and made stable. The first three stable nodes (according to node numbering sequence) are selected to make new member-connections. The maximum number of nodes that can be made stable from one group are restricted to a fixed number based on the configuration of the network. To enhance exploration further, all nodes are defined as alive or dead. While applying the local rules, only the alive nodes are considered for making new

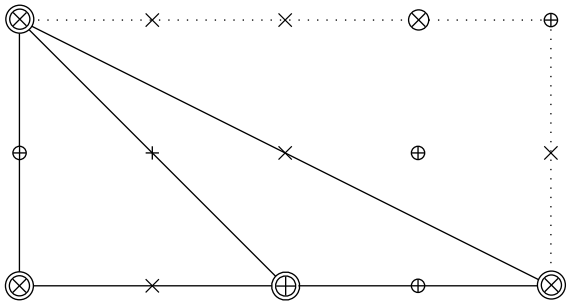

Figure 4.8: RBN Parametrization

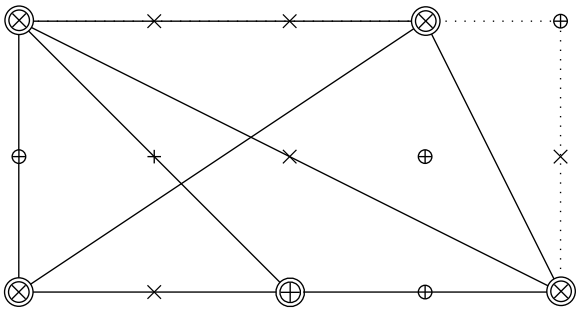

Figure 4.9: RBN Rule application

connections to the other nodes and are made stable. However, for the purpose of identifying the already stable nodes, to which a node is connected to, all the stable nodes are considered alive.

This BRN parameterization is illustrated in Fig. 4.8; it shows one particular network configuration for a minimally-connected initial truss design based on the classical 10-bar truss example. All the nodes are partitioned in two groups. The two groups are denoted by + and ×. A single ring around a symbol indicates that a node is alive, and a double ring indicates that the node is stable. Stability can also be observed from nodal connections. A symbol without a ring means that the node is dead. There are a total of 4 stable nodes in Fig. 4.8, out of which 3 are in one group. Therefore, while applying the local rules, all the alive nodes of that group are connected to the three stable nodes and made stable. For the purpose of illustration only one node is kept alive from that group. Figure 4.9 shows state of the nodes after applying the local rules.

The proposed methodology uses a GA to operate on BRN neighborhood assignments to optimize truss topology generated by BRNs. The genotype for the genetic algorithm is the neighborhood assignment of the BRN, and the phenotype is the truss topology. Given an initial network topology, in

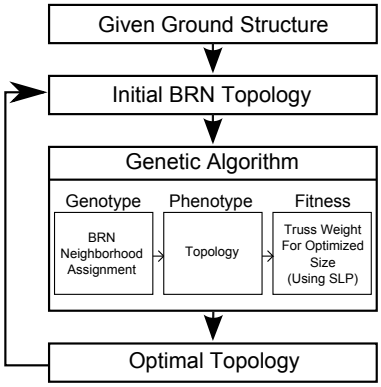

Figure 4.10: Multistage GA/BRN truss design methodology

accordance with the neighborhood assignment represented by the genotype, the local rules are applied on each node to obtain the corresponding phenotype. Fitness for each individual in the population is the minimal weight obtained by solving the inner loop geometry and size optimization problem. In the inner loop, each individual is solved for a fixed number of iterations using SLP with respect to stress and displacement constraints. The sequence of linear programming (LP) problems is solved using the MATLAB® interior point method. This ensures that each individual design in the GA population and its associated fitness value represents a truss design with minimal weight for the given topology, and satisfies all design constraints.

The approach of adding members to redistribute load and reduce mass is extended to an iterative multi-stage methodology. More specifically, starting with a minimally connected truss topology, the above methodology is applied in stages. At each stage of topology development a limited number of new members are added in an optimal way using the GA. In the next stage of development the optimal topology from the previous stage is used as the initial topology configuration. The GA/BRN strategy is then applied again to add members and further reduce mass. This methodology allows for a systematic exploration of topological designs. Truss topology is developed with each subsequent development stage until adding new members does not reduce truss mass further. Figure 4.10 illustrates this multi-stage methodology. Additional details follow, including definition of the genotype (BRN neighborhood assignment), and a strategy for avoiding overlapping members in generated truss topologies.

4.4 Genomic Encoding of Boolean Random Networks Representation

The GA used here acts upon the individual genes that encode the neighborhood assignments of a Boolean random network given a fixed number of nodes in the design space. As explained above, the genotype needs to encode the group identifier for each node, as well as an identifier that indicates whether the node is alive or dead. The genotype is a mixed vector of reals and integers: $\mathbf{x}$, whose elements are either in the interval $[0, 1]$, or are in the set $\{1, 2..., K\}$, where K is the number of groups. Equation (4.1) illustrates the structure of $\mathbf{x}$.

$$\mathbf{x} = [x_1^g, x_1^p, x_2^g, x_2^p, \ldots, x_N^g, x_N^p]^{\mathrm{T}} \tag{4.1}$$

The genotype has $2N$ elements, two for each node in the ground structure. The first element for each node ($x_i^g \in \{1, 2..., K\}$) encodes its group number. The second element for each node (x_i^p) is a real number between $[0, 1]$ that is used to determine whether the node is alive or dead. A fixed parameter p is chosen before problem solution that, along with the value of x_i^p, determines whether a node is alive or dead; if $x_i^p < p$, the node is alive, and if $x_i^p \geq p$ it is dead. After decoding the genome $\mathbf{x}$, new members can be deterministically added to the existing structure based on the methodology described above.

4.5 Resolving Overlapping Members

When the local rules are applied on a BRN to generate a truss topology, members may fully or partially overlap. Overlapping members and other undesirable connections should be avoided. Figure 4.11 illustrates a categorization of all six undesirable truss connection types that may occur based on the BRN rules defined here. The solid line with circular nodes at the each end represents an existing member between the two stable nodes. The two $\times$ symbols represent the nodes that are to be connected as a result of application of local rules. Below is the list describing the 6 types of erroneous connections.

1. The new member passes over an existing stable node without the two

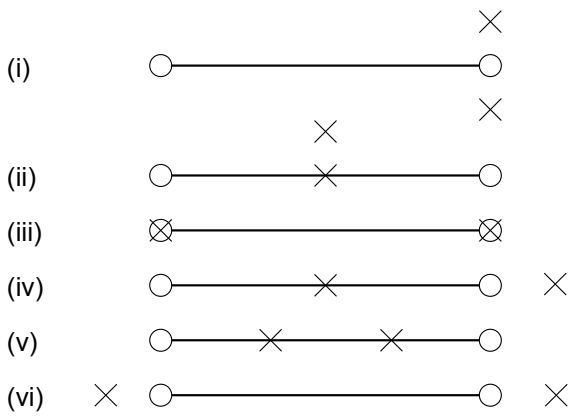

Figure 4.11: Rules to avoid overlapping members

members connecting.

2. One of the new nodes has an existing member passing over it without the two members connecting.

3. The new member exactly overlaps an existing member with both the nodes coinciding.

4. The new member partially overlaps an existing member with one new node falling over the existing member.

5. The new member fully overlaps an existing member with both the new nodes falling over the existing member.

6. The new member partially overlaps an existing member with none of the new nodes falling over the existing member.

It is clear that these connection errors need to be rectified. Below, two rules are proposed that rectify all six of these erroneous connections.

1. If a new member passes through a stable node then the new member should be replaced by two smaller members connecting the end nodes of the new member with the existing stable node.

2. If a new node being made stable falls on an existing member then the existing member should be replaced by two smaller members connecting the end nodes of the existing member with the new node.

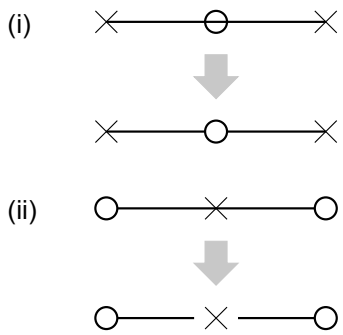

Figure 4.12: Rules to avoid overlapping members

Figure 4.12 graphically shows application of the above defined two rules. Application of rules 1 and 2 rectifies error types 1 and 2, respectively. Application of the rules 1 and 2 converts the error type 4 into error type 3. Similarly, two applications of rule 1 on error type 5, and two applications of rule 2 on error type 6 converts both into error type 3. Finally overlapping members of error type 3 are removed easily (they are redundant). Also, if all the three connections made to an earlier non-connected nodes are parallel the new connections are not allowed.

4.6 Truss Optimization Problem Formulation

The new methodology is formulated mathematically to solve truss design problems. The problem is formulated such that it finds the optimal topology geometry and size of the circular cross-sectional bars for a given design space. Truss geometry refers to the position of the nodes, specifically their x and y coordinates in the Cartesian coordinate system. Truss member size (cross-sectional area) is a continuous variable with lower and upper bounds. The concurrent topology, geometry and size optimization problem is represented mathematically using the equations below.

$$
\begin{aligned}
\min_{n,\mathbf{C},\mathbf{A},\mathbf{P}} \quad & f = \sum_{\substack{0\leq i\leq n \\ 0\leq j\leq n}} \rho C_{i,j} A_{i,j} l_{i,j} \\
\text{Subject to:} \quad & \sigma_{\min} \leq \sigma_{i,j} \leq \sigma_{\max} \\
& d_{\min} \leq d_k \leq d_{\max} \\
\text{where} \quad & i,j,k \in \{1,2,\ldots,n\}, i \neq j
\end{aligned}
\tag{4.2}
$$

where n is the number of nodes (joints) in the truss, $C_{i,j} \in \{0,1\}$ indicates whether nodes i and j are connected, $\mathbf{P}_k = [x_k, y_k]^{\mathrm{T}}$ is the position vector for each node $k = 1, 2, \ldots, n$, and $A_{i,j}$ is the cross-sectional area of the bar that connects nodes i and j. The material density is ρ, and several quantities are functions of design variables, including the length of the bar that connects nodes i and j ($l_{i,j} = \|\mathbf{P}_i - \mathbf{P}_j\|_2$), the axial stress of the member connecting nodes i and j ($\sigma_{i,j}$), and the displacement of each node ($d_k, k \in \{1,2,\ldots,n\}$) (computed here using the force method). The minimum allowable stress is $\sigma_{\min}$ ($\sigma_{\min} < 0$, indicating compression), and the maximum allowable stress is $\sigma_{\max}$ ($\sigma_{\max} > 0$, indicating tension). Similarly, the minimum and maximum allowable node displacements are $d_{\min}$ and $d_{\max}$, respectively. The objective of the design problem is to minimize overall structural mass.

The above stated mathematical problem is equivalent to finding the optimal topology, geometry and member sizes given a ground set of nodes and the stress and the displacement constraints. This problem is solved here using the proposed nested methodology where a BRN genotype representation is used with a GA to optimize truss topology, and an inner-loop geometry and size optimization problem is solved using SLP for every topology considered by the GA. The outer loop was solved using the MATLAB® `ga` function. The following BRN parameters were used:

- The mesh of ground set nodes is 21×11.
- The number of partition groups, K, for first development stage is 2.
- The number of partition groups is increased by 1 in each subsequent development stage.
- The maximum number of nodes that can be made stable in a group is 2.

- The probability of a node being alive is 0.2.

In the inner-loop problem, mass is minimized with respect to node position vectors and cross-sectional areas, while satisfying maximum tensile and compressive stress limits for each bar member, and satisfying maximum nodal displacement of each node. Stress and displacement values are computed using finite element analysis based on the force method. It can be noted that in the map L-systems based methodology, geometry is optimized in the outer loop as the L-system abstraction represents both geometry and topology. Boolean random networks, in contrast, represent only topology. Geometric design is instead managed along with size optimization by the inner loop.

In general, solving a simultaneous geometry and size optimization problem is not easy. There are many studies where genetic algorithms have been used in an attempt to solve this combined size and geometry design problem. Rahami, Kevah, and Gholipour [10], Giger and Ermanni [11], Rajan [12], Balling, Briggs and Gillman [13] studied methods to solve the combined optimization problem using various evolutionary algorithms. Allison and Papalambros [43] used decomposition-based design optimization methods to solve the same problem. Solving this combined problem using gradient based algorithms is very difficult because of the different nature of size and geometry design variables. It is desirable to solve the problem using gradient-based algorithms instead of using evolutionary algorithms to reduce computational expense. This is especially important since the combined size and geometry design problem must be solved in the inner loop for every candidate topology considered by the outer-loop GA. A new methodology is used here where geometry and size problems are decoupled and solved using SLP for the same set of stress and displacement constraints. Further, geometry optimization is solved in a novel way in a localized domain, making the problem much easier to solve.

The geometry and size problems are decoupled and solved in a iterative approach sequentially one after the other until the solution converges. However, it is observed that localized geometry optimization problem is required to be solved only once, i.e., the iteration converges in one step. First, the size optimization problem is solved keeping the initial geometry constant. After that the geometry optimization problem is solved, and then again the size optimization problem is solved. The iteration converges in one step as the

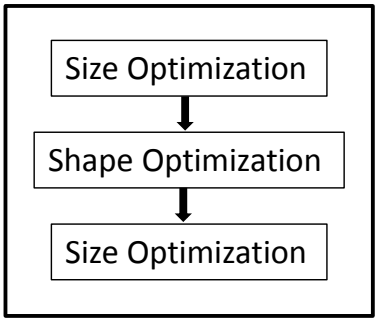

Figure 4.13: Geometry and Size Optimization

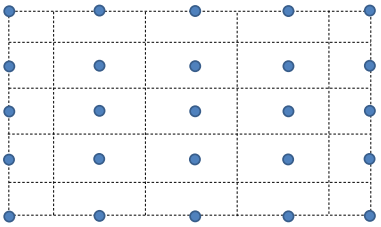

Figure 4.14: Localized Geometry Optimization Domain

nodes are allowed to move only in their vicinity. Figure 4.13 illustrates the iterative approach explained above.

For geometry optimization, the nodes are allowed to move only in a small domain near the original node location. The entire design domain is divided into rectangles where the space between two internal nodes is equally divided, forming a rectangle around each node. A node is allowed to move only inside the rectangle around it. The nodes on the boundary of the design domain have half the space of the internal nodes to move. Figure 4.14 illustrates the geometry optimization domain of the points within a rectangular design domain that has a total of 25 points equally spaced horizontally and vertically. The concept of allowing a node to move only in a local vicinity is an appropriate strategy because in the topology optimization a node is selected over the other nodes that are outside its geometry optimization domain. In other words, if moving a node outside its local domain is desirable, the outer loop algorithm can instead connect to the appropriate node that is adjacent to the current node. Topology, geometry, and size optimization allows for the search of the entire design domain. For size optimization problem, the inverse of cross sectional areas are used as optimization variables instead of the areas; this results in a non-linear objective function, but the non-linearity of the constraint equations is reduced significantly. A lower and upper bound is imposed on cross sectional area values.

As explained in Chapter 3, when both the stress and the displacement constraints are included in the formulation, basic SLP does not converge unless move limits are used that limit the error resulting from the linear approximation. The move limit formulation of Lamberti and Pappalettere's [23] described in Chapter 3 is used here. Further, in the same manner, each linear subproblem is solved using the MATLAB® `linprog` function using the interior point method. This ensures that subproblem solutions are always feasible. In Chapter 5, the results of one archetypal truss design example problem is solved using the above formulation, and the results are presented and discussed.

In summary, truss topology is represented using a parameterized neighborhood assignment of Boolean random networks. The BRN parameters form the genotype of the genetic algorithm. The GA phenotype is generated by applying local rules to the Boolean random networks in accordance with the structural stability of the truss topology design. The fitness of each candidate GA is calculated by the inner-loop solver that optimizes both truss geometry and size. A new strategy for combined geometry and size optimization, based on SLP, is presented that optimizes the position vector for each node within a local domain. The localized geometry optimization leads to fast convergence of the inner-loop problem. The Boolean random network is different from map L-systems methodology in fundamental ways as it optimizes only topology in the outer loop (the map L-systems methodology optimizes topology and geometry simultaneously in the outer loop). This strategy of decoupling the topology and geometry optimization allows for improved design space coverage, and is expected to give better result in terms of optimal truss design. The next chapter presents the results from solving three standard truss design problems using the two methodologies presented in this work. In particular, a 'ten-bar' truss design problem is solved using both the map L-system based methodology and Boolean random networks. The optimization results are compared. Next, an extended 'ten-bar' truss problem and a 'twenty-five' bar space truss problem are both solved using the map L-systems methodology. Finally, a conclusion of the work presented in this thesis is given.

Chapter 5

Results and Discussion

This section presents results of solving three classical truss design optimization problems using the two methodologies proposed in Chapters 3 and 4. The first example used is a ten-bar truss problem that is solved using both the methodologies: 1) the method based on a map L-systems design representation, and 2) a method based on a Boolean random network design representation. The second example used is an extended ten bar truss that is solved using only the map L-systems approach. The third example is of twenty-five bar space truss that is also solved using only map L-systems.

5.1 Ten-bar Truss

The first example used here is based on a ten-bar truss design problem from the structural optimization literature, illustrated in Fig. 5.1. Previous treatments of this problem dealt primarily with the size optimization problem, and assumed that the truss topology was fixed [2–4, 23, 44–53]. While the term 'ten-bar' is used here to refer to this example problem, this is primarily a historical reference, and is not meant to imply that solutions will composed of exactly ten bars. The problem used here involves combined size, geometry and topology optimization, but the design parameters (design boundary, load, material, etc) is based on the classical ten-bar truss design problem. Table 5.1 summarizes problem data.

5.1.1 Ten-bar Truss Optimization Using Map L-systems

To solve the problem using the proposed methodology explained in Chapter 3 an initial design domain is created. The initial truss design (map) is shown in Fig. 5.2; this includes bars on the edge of the design domain, as well as a

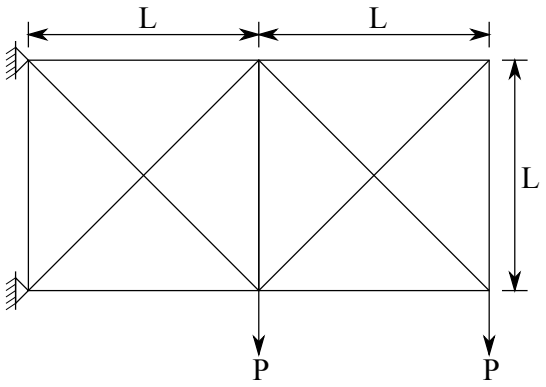

Figure 5.1: Topology and Geometry of Ten-bar Truss

Table 5.1: Input Data for Ten-bar Truss

Parameter	Value
L (design domain height)	360 in
P (load magnitude)	100 kips
Stress limits	$\pm$25 ksi
Maximum displacement	$\pm$2.0 in
Modulus of Elasticity	10^4 ksi
Material density	0.1 lbm/in^3
Section areas lower limit	0.1 in^2
Section areas upper limit	35 in^2
Number of loading conditions	Single loading

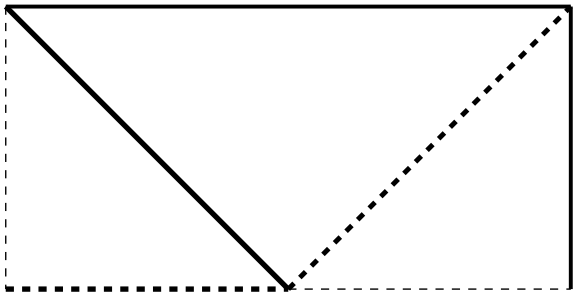

Figure 5.2: Ten-bar Truss Initial Design-0

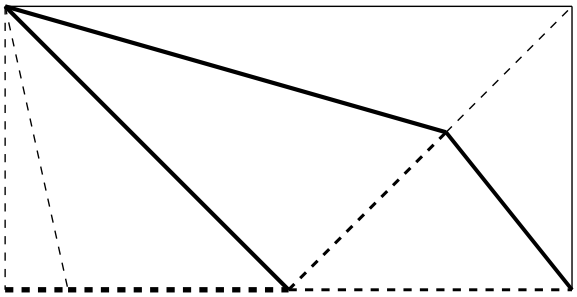

Figure 5.3: Ten-bar Truss Development-1

minimum number of additional bars required to provide structural stability using a composition of triangular cells. Figures 5.3–5.8 show the truss designs after each stage of development using Algorithm 2 where the GA is solved completely using one development stage at a time. The solid lines represent bars under tension, and the dotted lines represent bars under compression. Line thickness is proportional to cross-sectional area.

Going beyond six development stages does not reduce system mass, indicating that adding more truss members does not help re-distribute load in a way that permits mass reduction. The optimal mass and number of bar members for each truss development stage is shown in Table 5.2. Please observe that as new truss members are added in a strategic manner using Algorithm 2, substantial load redistribution and mass reduction occurs. The minimal mass obtained at stage six is significantly lower than the the results reported in the literature cited above (the minimum mass reported in previous articles is more than 5000 lbm.). It should be noted, however, that these past studies used a fixed topology, and the result here includes many more than ten bars, so a direct comparison cannot be made. These results, however, do indicate the effectiveness of this new combined topology, geometry, and size optimization methodology.

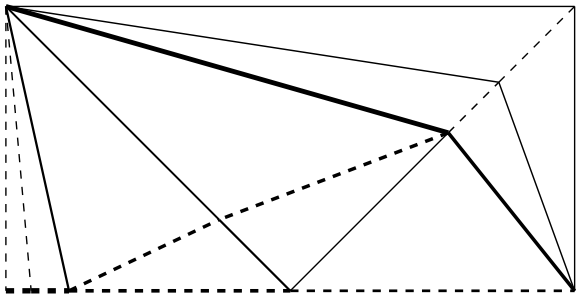

Figure 5.4: Ten-bar Truss Development-2

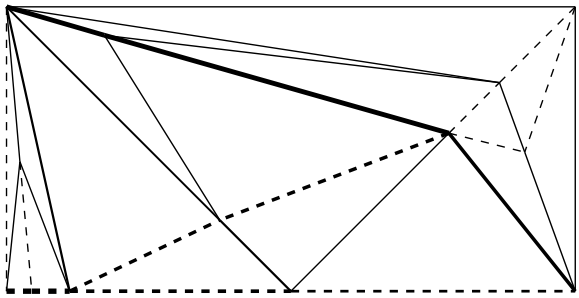

Figure 5.5: Ten-bar Truss Development-3

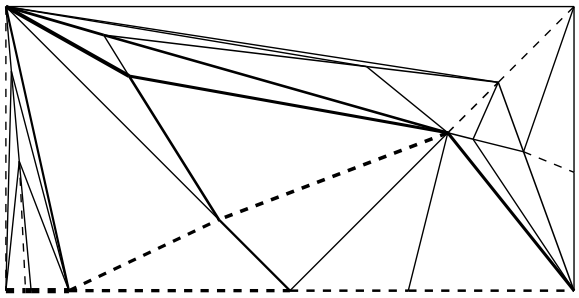

Figure 5.6: Ten-bar Truss Development-4

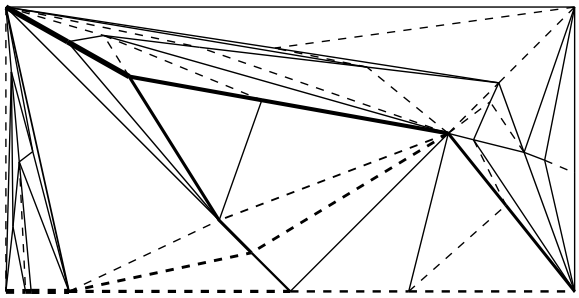

Figure 5.7: Ten-bar Truss Development-5

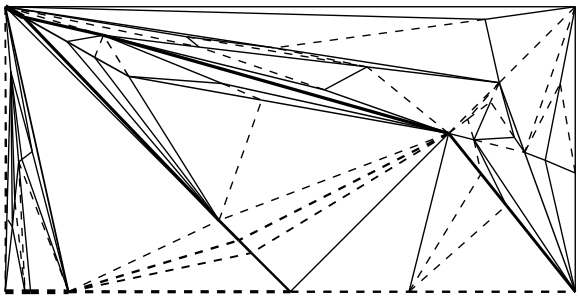

Figure 5.8: Ten-bar Truss Development-6

Table 5.2: Ten-bar Truss- Optimal Mass and No. of bars

Development Stage	Number of Bars	Optimal Mass (lbm.)
0	7	7097
1	12	5484
2	20	5147
3	29	5134
4	47	4913
5	77	4642
6	119	4241

5.1.2 Ten-bar Truss Optimization Using Boolean Random Networks

The same ten bar truss problem is further solved using the Boolean random networks methodology explained in Chapter 4. A ground set of 21×11 nodes is defined here based on the design domain of the earlier studies. The nodes are evenly distributed in a grid. The initial minimally connected truss topology is shown in Fig. 5.9. Note that the initial minimally connected truss design cannot satisfy displacement constraints at all the nodes due to bounds on cross-sectional areas. Additional members are required to satisfy displacement constraints completely. This initial topology is used to start the multi-stage optimization process illustrated in Chapter 4. Figures 5.10–5.13 show the truss designs after each stage of development. The circles represent nodes. The solid lines represent bars under tensile stress, and the dotted lines represent bars under compressive stress. Line thickness is proportional to cross-sectional area. No further mass reduction can be achieved beyond development stage 4 (i.e., the method has converged after five development stages). Adding more bar members does not help re-distribute load, and hence does not reduce truss mass. The optimal mass and number of bar

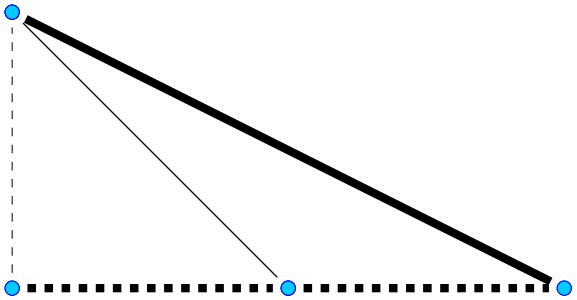

Figure 5.9: Ten-bar Truss Initial Design-0

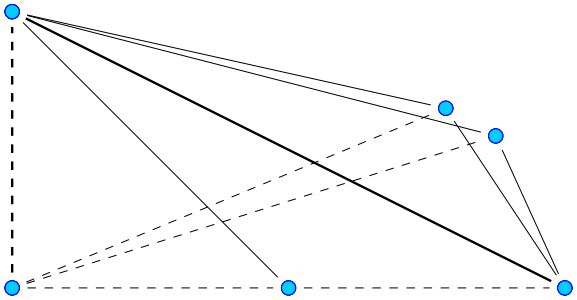

Figure 5.10: Ten-bar Truss Development-1

members for each truss development stage is shown in Table 5.3. The results obtained show that as new bar members are added there is substantial load re-distribution, and a corresponding reduction in truss mass. The minimal mass obtained at the end of the last development stage is significantly lower than the the results reported in the literature cited above, although this design has far more truss members. The minimum mass reported in the above cited literature is above 5000 lbm.

It can be seen that the methodology based on Boolean random networks gives significantly better results in comparison to the map L-systems based methodology. This methodology gives the optimal truss design of mass 4040 lbm with just 29 bars. However the map L-systems methodology gives the

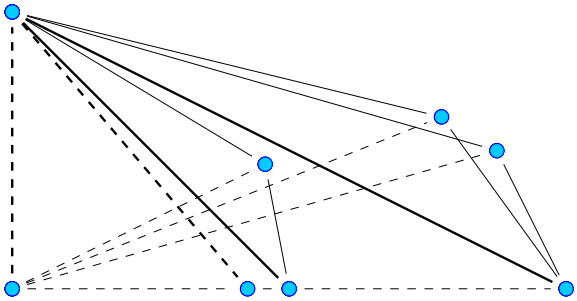

Figure 5.11: Ten-bar Truss Development-2

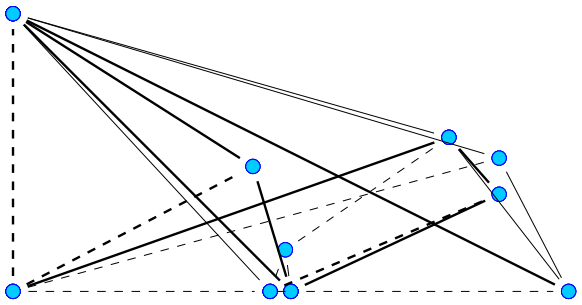

Figure 5.12: Ten-bar Truss Development-3

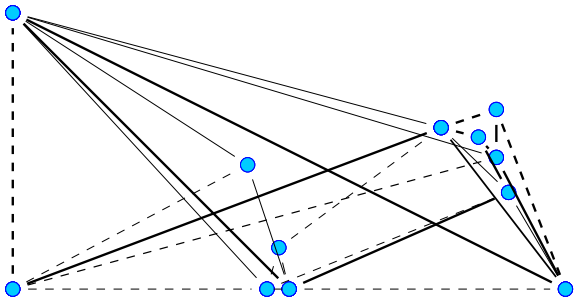

Figure 5.13: Ten-bar Truss Development-4

Table 5.3: Ten-bar Truss- Optimal Mass and No. of bars

Development Stage	Number of Bars	Optimal Mass (lbm.)
0	5	5689
1	11	4918
2	16	4710
3	22	4238
4	29	4040

optimal truss design of mass 4241 lbm with 119 bars which is far more than 29 bars used in the Boolean random networks. The map L-systems methodology uses abstraction concept for both truss topology and geometry while the Boolean random networks methodology uses abstraction concept only for topology and optimizes geometry in the inner loop. This difference in the two methodologies gives the latter one higher degree of freedom to search the design space for the optimal design. Further, the Boolean random networks methodology uses a relatively direct abstraction approach to search the optimal topology which also gives it a higher degree of freedom and hence the ability to identify a lower-mass design.

5.2 Extended Ten-bar Truss

This problem is solved using only map L-systems methodology explained in Chapter 3. An extended design boundary for the standard ten-bar truss problem is considered as shown in Fig. 5.14. The motivation for considering this problem is to compare the results with the literature where topology and geometry optimization is also performed along with size. The ten-bar truss design problem with topology, geometry, and size optimization was previously studied by Rahami, Kevah, and Gholipour [10], Rajan [12], and Tang, Tong and Gu [54]. In these previous studies, truss topology was managed using a ground structure approach. The strategy introduced here, however, does not require a ground structure (just a design boundary). To provide a fair comparison, the design boundary is expanded such that it encompasses the optimized topology and geometry reported in the previous studies cited above. The design domain here is a square with sides equal to the width of the standard ten-bar design domain used above (width $= 2L = 720$ in.), and the same problem data is used (see Table 5.1).

Figure 5.15 shows the initial stable truss design (map). Optimization is performed starting with this design and proceeding using Algorithm 2. Figures 5.16–5.19 show the truss designs after each stage of development. Additional development stages do not result in further mass reduction. Table 5.4 presents the optimal mass and number of truss members for each development stage. As with the previous example, this design methodology reduces mass at each stage by adding members strategically to redistribute load.

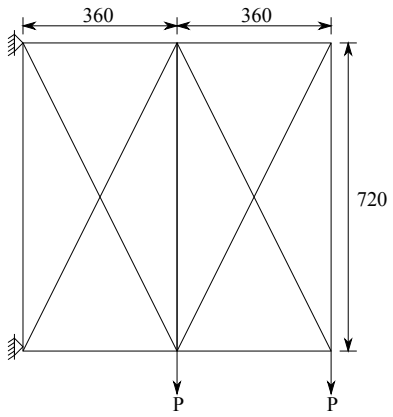

Figure 5.14: Topology and Geometry of Extended Ten-bar Truss

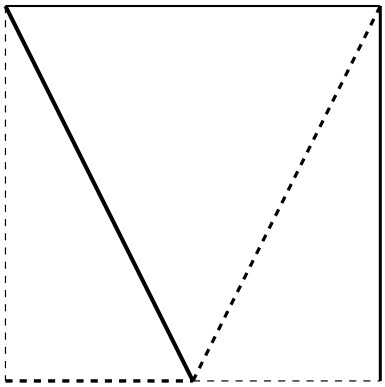

Figure 5.15: Extended Ten-bar Truss Initial Design-0

The minimal mass obtained at the end of the last development stage is significantly lower than the the results reported in the literature cited above, although it has many more truss members. The minimum mass reported in the above cited literature is above 2700 lbm. for a design domain with approximately the same size.

Table 5.4: Extended Ten-bar Truss- Optimal Mass and No. of bars

Development Stage	Number of Bars	Optimal Mass (lbm.)
0	7	6832
1	10	3039
2	18	2786
3	30	2645
4	41	2588

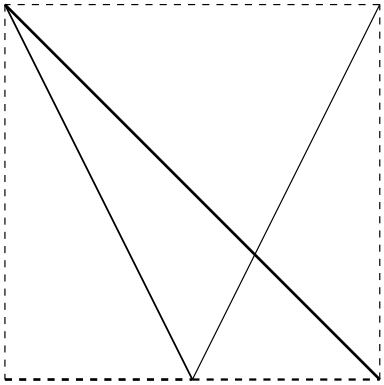

Figure 5.16: Extended Ten-bar Truss Development-1

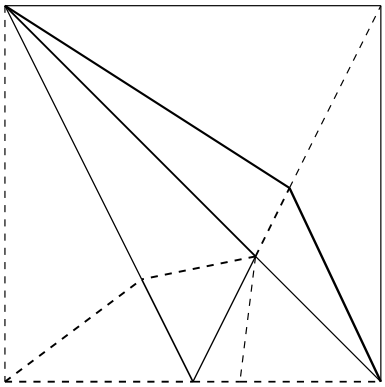

Figure 5.17: Extended Ten-bar Truss Development-2

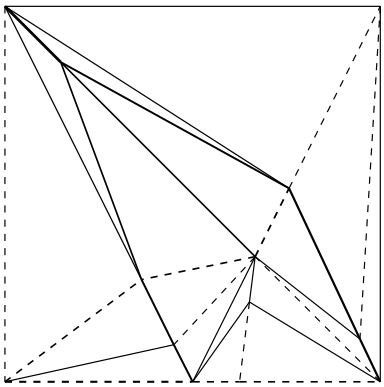

Figure 5.18: Extended Ten-bar Truss Development-3

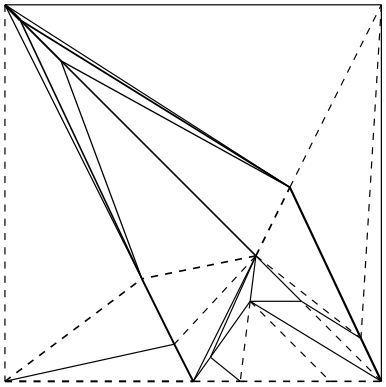

Figure 5.19: Extended Ten-bar Truss Development-4

5.3 Twenty-five-bar Space Truss

This problem is solved using only map L-systems methodology explained in Chapter 3. A space truss design problem is presented here to demonstrate the extension of the generative algorithm truss design methodology to three-dimensional problems. The twenty-five bar transmission tower space truss design problem, illustrated in Fig. 5.14, has been studied extensively [2, 3, 23, 44–48, 50–53], but has been limited to size optimization for a fixed twenty-five bar topology. The problem studied here includes also topology and geometry optimization, based on the same design domain boundary and problem data (see Table 5.5). This problem involves two loading conditions that are specified in Table 5.6. These loading cases are identical to the ones used in the references cited above.

Figure 5.21 illustrates the initial truss design (map); this initial design includes the minimum number of bars required to produce a stable truss using a composition of tetrahedra. The space between the nodes 3–10 (node numbering based on Fig. 5.20) has been divided into five non-intersecting tetrahedra. Nodes 1 and 2 has been joined with nodes 3–6 (equivalent to creating to four tetrahedra). These four tetrahedra are created to maintain symmetry of the structure and to cover the same space as covered by the standard design. In total, the initial design has 26 bars that form 9 tetrahedra.

Figures 5.22 and 5.23 show the truss designs after each stage of development using Algorithm 2. Going beyond two development stages does not reduce truss mass further. The optimal mass and number of bar members for each truss development stage is listed in Table 5.7. The minimal mass obtained at the end of the last development stage is marginally lower than the

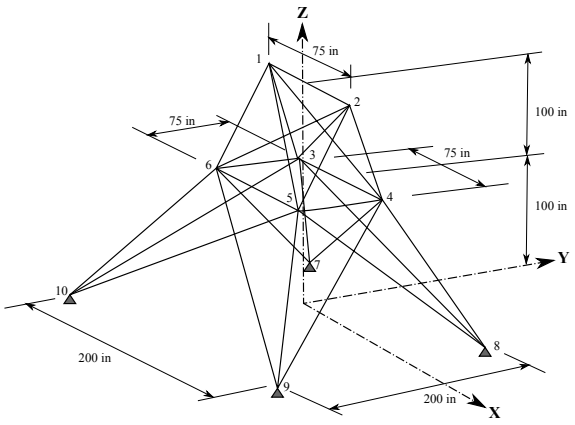

Figure 5.20: Topology and Geometry of Twenty-five-bar Truss

Table 5.5: Input Data for Twenty-five-bar Truss

Parameter	Value
Stress limits	± 40 ksi
Maximum displacement	± 0.35 in
Modulus of Elasticity	10^4 ksi
Material density	0.1 lbm/in^3
Section areas lower limit	0.1 in^2
Section areas upper limit	10 in^2
Number of loading conditions	Two Cases

the results reported in the literature cited above, although it has many more bar members. The minimum mass reported in the above cited literature is above 540 lbm.

As explained in the previous sections, Algorithm 2 exploits the ability to add new bars at each stage in a way that produces substantial load redistribution. Continuing with a new stage of the algorithm is beneficial only if the load redistribution from adding bars enables enough mass reduction to offset the addition of new bars (new bars have a lower bound on cross section areas, so adding new bars does have a mass penalty). This is unlikely to occur with the initial map unless the number of members in the initial truss design is relatively high. It was discovered that increasing load in the space truss design problem resulted in a larger number of development stages and more noticeable mass reductions as the algorithm proceeded. The generative algorithm approach to truss design appears to be particularly beneficial in cases where loading conditions demand more complex topologies.

Table 5.6: Loading conditions, in kips, for Twenty-five-bar Truss

Node	Condition 1			Condition 2		
	P_X	P_Y	P_Z	P_X	P_Y	P_Z
1	0.0	20.0	-5.0	1.0	10.0	-5.0
2	0.0	-20.0	-5.0	0.0	10.0	-5.0
3	0.0	0.0	0.0	0.5	0.0	0.0
6	0.0	0.0	0.0	0.5	0.0	0.0

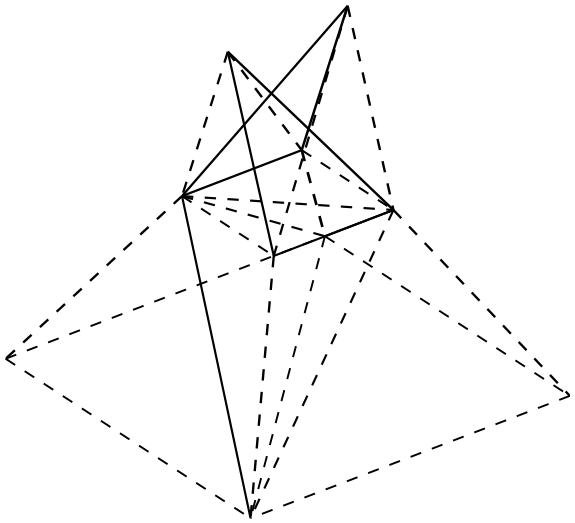

Figure 5.21: Twenty-five-bar Truss Initial Design-0

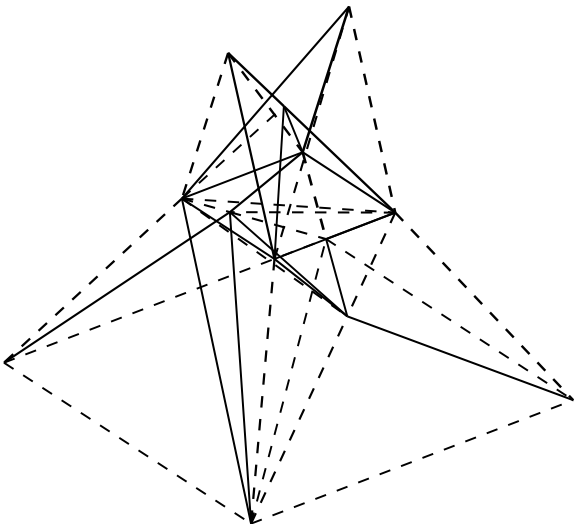

Figure 5.22: Twenty-five-bar Truss Development-1

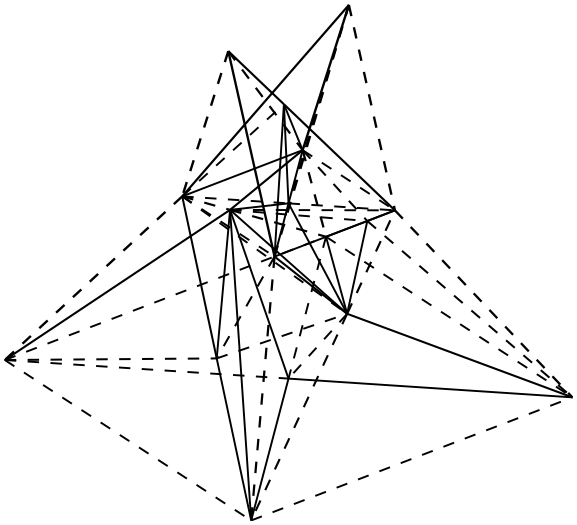

Figure 5.23: Twenty-five-bar Truss Development-2

Table 5.7: Twenty-five-bar Truss- Optimal Mass and No. of bars

Development Stage	Number of Bars	Optimal Mass (lbm.)
0	26	631
1	41	573
2	63	524

Chapter 6

Conclusion

Two new approaches, based on generative algorithms and Boolean random vectors, were presented that accommodate variable dimension design problems and allow the exploration of topological design alternatives that are not from a set that is defined a priori (as in the case with ground structure methods). The effectiveness of the proposed methodologies for solving truss design problems with respect to size, geometry, and topology was demonstrated using several archetypal truss design optimization problems.

A new generative algorithm strategy based on map L-systems was presented that accounts for several unique design requirements that are specific to structural truss design, namely stability and load redistribution. The generative algorithm serves as a design abstraction that supports the efficient topological design of complicated structures. Instead of operating on physical design variables directly, designers can adjust generative algorithm rules that govern design generation. A small set of parameters with fixed dimension can therefore control the design of complex structures with variable dimension. The new generative algorithm automatically produces stable trusses, contributing to efficient design space exploration. An inner loop is used to solve the size optimization problem so that a fair comparison can be made between candidate topologies. The generative algorithm is used as a sophisticated mapping between genetic algorithm (GA) genotype and phenotype, which results in a process that more closely mimics the morphogenetic process present in the evolution of real biological systems when compared to directly-encoded GAs. Two new algorithms were introduced. Algorithm 1 involves a single GA solution, and a generative algorithm that is executed for multiple development stages to produce a topology and geometry for each individual in a GA population. Algorithm 2 involves a series of GA solutions where only one development stage is used for each individual in the population, and the best design from the previous stage is used as the initial map

for the next stage. This methodology was introduced first for planar trusses, and then an extension to space trusses was presented. Algorithm 2 was demonstrated using three case studies, and the resulting designs exhibited significant mass reduction.

In the map L-system formulation, topology and geometry of the truss design are represented together by a cellular division algorithm abstraction layer. In other words, the same parameters control topology and geometry, and hence the design degrees of freedom in selecting topology and geometry are limited. If any parameter is changed in the map L-system, then it changes both resulting topology and geometry. The coupling of topology and geometry restricts exploration of design space. To overcome this limitation of the map L-system based abstraction, another novel representation algorithm using Boolean random networks was created. The Boolean random network based parameterization represents topology alone, and leaves geometry to be optimized in the inner-loop together with size optimization. To optimize size and geometry design together, a localized geometry optimization methodology was presented that, together with the outer-loop topology optimization, allows for an efficient search of the complete design domain.

The Boolean random network based methodology is similar to the ground structure methodology as it requires the definition of an initial set of a fixed number of nodes (and their locations) that serve as the basis for topology generation using a set of iterative rules. This approach using the Boolean random network addresses two shortcomings of ground structure methods that utilize direct GA encodings, namely, the inability to scale up to large-dimension problems, and the inefficiency of exploring infeasible design topologies. The Boolean random network approach utilize vector parameterizations where the representation dimension (and optimization problem size) increases linearly with the number of nodes in the design domain. Ground structure methods with direct GA encoding, however, have optimization problem sizes that increase quadratically with the number of nodes. The phenotype to genotype conversion effected by the Boolean random network algorithm (and also the map L-systems algorithm) can include design requirements embedded within this algorithm. In other words, these generative algorithms produce designs that automatically satisfy design requirements, such as structural stability for truss design. This property supports significantly more efficient topological design space exploration compared to ground structure methods. With

conventional ground structure methods there is no guarantee that a given design will be structurally stable. A large portion of designs are in fact unstable, resulting in very inefficient truss design space exploration. The BRN and map L-systems methods avoid this inefficiency by evaluating only structurally stable designs due to the design requirements that are embedded within the generative algorithms.

Further, both the proposed methodologies exploits a fundamental principle of truss development, namely, that new bar members are added incrementally to the existing truss design to redistribute load, and hence to reduce optimal truss mass. In accordance with this concept, both the methodologies develop trusses starting with an initial design that has a minimal number of bar members that provide a stable design for a given set of loading and boundary conditions. At each development stage these methodologies optimally add a set of bar members to redistribute load and enable mass reduction through reduced bar member areas. Subsequent stages develop upon the optimal design obtained from previous stages. Truss design is developed in stages until the addition of new bar members cannot reduce mass any further through load redistribution an size re-optimization.

Finally, in Chapter 5, both methodologies were demonstrated through the solution of three standard truss design problems. The ten-bar truss design problem was solved using both the methodologies, and the results demonstrated that the BRN methodology produces design with significantly lower mass and fewer bar members. This result was expected due to the decoupling of topology and geometry. The BRN methodology can explore the design space more effectively, and hence produces better result in terms of the optimal truss design. An extension of the ten-bar truss problem and a twenty-five bar space truss problem were also solved using map L-system based methodology. All the example problems show that the proposed methodologies explores design space efficiently, and produce better results than previous methods (with the acknowledgement that the trusses produces here have many more bars than the trusses used as comparison problems). In summary, utilizing generative algorithms and Boolean random networks as a design abstraction is a promising strategy for optimizing increasingly complex engineering systems, particularly when unique design considerations (such as structural stability) can be embedded into the generative algorithm, and when a range of problem dimensions and complexities must be explored.

Opportunities for future work include studies of GA convergence with respect to the two methodologies, as well as an analytical investigation of how well the proposed algorithms cover (or access) different portions of the truss topology and geometry design spaces. Also, the proposed methodologies should be applied to more complex problems to assess their performance. An immediate next step would be to extend the Boolean random network based methodology to three-dimensional space truss problems.

References

[1] Bendsoe, M. P., and Sigmund, O., 2003. *Topology optimization: theory, methods and applications*. Springer.

[2] Venkayya, V., 1971. "Design of optimum structures". *Computers & Structures,* **1**(1), pp. 265–309.

[3] Schmit, L. A., and Farshi, B., 1974. "Some approximation concepts for structural synthesis". *AIAA journal,* **12**(5), pp. 692–699.

[4] Dobbs, M., and Nelson, R., 1976. "Application of optimality criteria to automated structural design". *AIAA Journal,* **14**(10), pp. 1436–1443.

[5] Goldberg, D., and Samtni, M., 1991. "Engineering optimization via the genetic algorithms". *Computers and Structures,* **40**, pp. 1321–1327.

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

[7] Deb, K., and Gulati, S., 2001. "Design of truss-structures for minimum weight using genetic algorithms". *Finite elements in analysis and design,* **37**(5), pp. 447–465.

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

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

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

[11] Giger, M., and Ermanni, P., 2006. "Evolutionary truss topology optimization using a graph-based parameterization concept". *Structural and Multidisciplinary Optimization,* **32**(4), pp. 313–326.

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

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

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

[15] Reas, C., and Fry, B., 2006. "Processing: Programming for the Media Arts". *AI & Society,* **20**, pp. 526–538.

[16] Prusinkiewicz, P., Lindenmayer, A., Hanan, J. S., Fracchia, F. D., Fowler, D. R., de Boer, M. J., and Mercer, L., 1990. *The algorithmic beauty of plants*, Vol. 2. Springer-Verlag New York.

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

[18] Karbowski, D., Pagerit, S., Kwon, J., Rousseau, A., and Freiherr von Pechmann, K., 2009. "Fair Comparison of Powertrain Configurations for Plug-In Hybrid Operation Using Global Optimization". *SAE Technical Paper Number 2009-01-1334*.

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

[20] Futuyma, D., 1997. *Evolutionary Biology*, third ed. Sinauer Associates.

[21] Cheney, N., MacCurdy, R., Clune, J., and Lipson, H., 2013. "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding". In Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference, ACM, pp. 167–174.

[22] Wujek, B., and Renaud, J., 1998. "New adaptive move-limit management strategy for approximate optimization, part 1". *AIAA journal,* **36**(10), pp. 1911–1921.

[23] Lamberti, L., and Pappalettere, C., 2000. "Comparison of the numerical efficiency of different sequential linear programming based algorithms for structural optimisation problems". *Computers & Structures,* **76**(6), pp. 713–728.

[24] Chen, T.-Y., 1993. "Calculation of the move limits for the sequential linear programming method". *International Journal for Numerical Methods in Engineering,* **36**(15), pp. 2661–2679.

[25] Wolfram, S., 1994. *Cellular automata and complexity: collected papers*, Vol. 1. Addison-Wesley Reading.

[26] Inoue, N., Shimotai, N., and Uesugi, T., 1994. "Cellular automaton generating topological structures". In Smart Structures and Materials: Second European Conference, International Society for Optics and Photonics, pp. 47–50.

[27] Inou, N., Uesugi, T., Iwasaki, A., and Ujihashi, S., 1997. "Self-organization of mechanical structure by cellular automata". *Key Engineering Materials,* **145**, pp. 1115–1120.

[28] Kundu, S., Oda, J., and Koishi, T., 1997. "Design computation of discrete systems using evolutionary learning". In Proc. WCSMO-2, Second World Congress on Structural and Multidisciplinary Optimization (held in Zakopane, Poland), pp. 173–180.

[29] Kundu, S., Oda, J., and Koishi, T., 1997. "A self-organizing approach to optimization of structural plates using cellular automata". In Second World Congress on Structural and Multidisciplinary Optimization (WCSMO-2)(Gutkowski W. and Mroz Z., eds.), Polish Academy of Science, Zakopane, Poland, pp. 173–180.

[30] Xie, Y., and Steven, G. P., 1993. "A simple evolutionary procedure for structural optimization". *Computers & structures,* **49**(5), pp. 885–896.

[31] Xie, Y., and Steven, G. P., 1994. "Optimal design of multiple load case structures using an evolutionary procedure". *Engineering computations,* **11**(4), pp. 295–302.

[32] Xie, Y., and Steven, G., 1994. "A simple approach to structural frequency optimization". *Computers & structures,* **53**(6), pp. 1487–1491.

[33] Xie, Y., and Steven, G., 1996. "Evolutionary structural optimization for dynamic problems". *Computers & Structures,* **58**(6), pp. 1067–1073.

[34] Zhao, C., Steven, G., and Xie, Y., 1997. "Effect of initial nondesign domain on optimal topologies of structures during natural frequency optimization". *Computers & structures,* **62**(1), pp. 119–131.

[35] Zhao, C., Steven, G., and Xie, Y., 1998. "A generalized evolutionary method for natural frequency optimization of membrane vibration problems in finite element analysis". *Computers & Structures,* **66**(2), pp. 353–364.

[36] Yang, X., Xie, Y., Steven, G., and Querin, O., 1998. "Bi-directional evolutionary method for frequency optimisation". In Australasian Conference on Structural Optimisation, pp. 231–237.

[37] Young, V., Querin, O., Steven, G., and Xie, Y., 1999. "3d and multiple load case bi-directional evolutionary structural optimization (beso)". *Structural optimization,* **18**(2-3), pp. 183–192.

[38] Kim, H. A., Querin, O., Steven, G., and Xie, Y., 1998. "Development of an intelligent cavity creation (icc) algorithm for evolutionary structural optimisation". In The Australasian Conference on Structural Optimisation, University of Bath, pp. 241–250.

[39] Kita, E., and Toyoda, T., 2000. "Structural design using cellular automata". *Structural and Multidisciplinary Optimization,* **19**(1), pp. 64–73.

[40] Gürdal, Z., and Tatting, B., 2000. "Cellular automata for design of truss structures with linear and nonlinear response". In Proc. 41st AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conf., AIAA Paper, Vol. 1580.

[41] Tatting, B., and Gürdal, Z., 2000. "Cellular automata for design of two-dimensional continuum structures". In Proceedings of 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

[42] Abdalla, M. M., and Gürdal, Z., 2002. "Structural design using optimality based cellular automata". In Proceedings of 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference.

[43] Allison, J. T., and Papalambros, P. Y., 2007. "Optimal partitioning and coordination decisions in system design using an evolutionary algorithm". In Proceedings of the Seventh World Conference on Structural and Multidisciplinary Optimization, Seoul, South Korea, May, pp. 21–25.

[44] Gellatly, R. A., and Berke, L., 1971. Optimal Structural Design. Tech. rep., DTIC Document.

[45] Schmit, L. A., 1976. "Approximation concepts for efficient structural synthesis".

[46] Rizzi, P., 1976. "Optimization of multiconstrained structures based on optimality criteria". In Proc. AIAA/ASME/SAE 17th Structures, Structural Dynamics and Materials Conference, pp. 448–462.

[47] Khan, M., Willmert, K., and Thornton, W., 1979. "An optimality criterion method for large-scale structures". *AIAA Journal,* **17**(7), pp. 753–761.

[48] John, K., Ramakrishnan, C., and Sharma, K., 1987. "Minimum weight design of trusses using improved move limit method of sequential linear programming". *Computers & structures,* **27**(5), pp. 583–591.

[49] Sunar, M., and Belegundu, A., 1991. "Trust region methods for structural optimization using exact second order sensitivity". *International journal for numerical methods in engineering,* **32**(2), pp. 275–293.

[50] Stander, N., Snyman, J., and Coster, J., 1995. "On the robustness and efficiency of the sam algorithm for structural optimization". *International Journal for Numerical Methods in Engineering,* **38**(1), pp. 119–135.

[51] Xu, S., and Grandhi, R. V., 1998. "Effective two-point function approximation for design optimization". *AIAA journal,* **36**(12), pp. 2269–2275.

[52] Lamberti, L., and Pappalettere, C., 2003. "Move limits definition in structural optimization with sequential linear programming. part ii: Numerical examples". *Computers & structures,* **81**(4), pp. 215–238.

[53] Lee, K. S., and Geem, Z. W., 2004. "A new structural optimization method based on the harmony search algorithm". *Computers & Structures,* **82**(9), pp. 781–798.

[54] Tang, W., Tong, L., and Gu, Y., 2005. "Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables". *International Journal for Numerical Methods in Engineering,* **62**(13), pp. 1737–1762.