Proceedings of IDETC/DAC 2014
The ASME 2014 International Design Engineering Technical Conferences &
Design Automation Conference
August 17–20, 2014, Buffalo, NY, USA

DETC2014–34256

LARGE-SCALE TOPOLOGY OPTIMIZATION USING PARAMETERIZED BOOLEAN NETWORKS

Ashish Khetan[*], **James T. Allison**[®]
University of Illinois at Urbana-Champaign
Urbana, IL 61801
Email: {khetan2, jtalliso}@illinois.edu

ABSTRACT

A novel parameterization concept for structural truss topology optimization is presented in this article 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. 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.

1 Introduction

Optimization of many engineering systems can be formulated mathematically as connectedness among a very large number of components. For example, structural truss, electrical power network, and automotive hybrid powertrain design problems can be formulated in this way. Use of a ground structure is a standard approach for truss topology optimization. In this approach, a ground structure is first defined that describes all possible components and connections. The optimization algorithm then determines which of these components should exist in the optimal design. An alternative approach is presented here where the process begins with the minimal number of truss members required to produce a viable structure, and then new members are added systematically to optimize the structure. In addition, conventional ground structure approaches are limited in that the size of the ground structure (i.e., number of potential members) is restricted by the size of the optimization vector since design variables correspond directly to truss members. The dimension of a binary truss topology representation increases quadratically with the number of ground structure nodes. The new approach presented here, based on Boolean random networks, significantly reduces design representation dimension, supporting the use of much larger initial ground structures. In this representation, design variables are parameters that control the generation of truss topologies, whereas design variables in conventional approaches

[*]Graduate Student, Industrial and Enterprise Systems Engineering
[®]Assistant Professor, Industrial and Enterprise Systems Engineering

design variables correspond directly to system elements. This indirect Boolean random network representation results in a design vector that increases in dimension only linearly in proportion to the number of ground structure nodes. This property allows scaling to large-dimension design problems with many nodes. The effectiveness of the proposed methodology is demonstrated using benchmark truss design problems involving optimal topology, geometry and size.

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 [1], Schmit and Farshi [2], and Dobbs and Nelson [3]. In addition, Goldberg and Samtani [4] and Rajeev and Krishnamoorthy [5] used evolutionary algorithms to solve the sizing optimization problem in truss design.

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

As the topology optimization using ground structures does not incorporate geometry optimization, an important next step was to extend this approach to include joint position optimization. This integrated geometry and topology design approach has been studied extensively, primarily using a hierarchical solution approach. Further, many GA-based approaches have also been explored to achieve integrated size, geometry and topology optimization. Rahami, Kevah, and Gholipour [9], Giger and Ermanni [10], Rajan [11], Balling, Briggs and Gillman [12], and Kaveh and Laknejadi [13] all used evolutionary algorithms[1] to find the optimal size, geometry and topology of trusses.

In this paper, we present a new methodology to solve the combined topology, geometry and size design problem for trusses using Boolean random networks. 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. Further, we present 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. Use of Boolean random networks allows us to consider 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, and to some extent ameliorates the need for shape optimization.

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, we optimize 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 rest of the paper is organized as follows. First, we re-

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

view how cellular automata has been used thus far in engineering system optimization. Then, we discuss the basics of cellular automata and Boolean random networks. Following that, we present a novel methodology using a modified representation of Boolean networks in combination with a genetic algorithm for truss topology, geometry and size optimization. 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.

3 Cellular Automata

The behavior of cellular automata (CA) is governed by a set of local rules that when applied iteratively produce complex global phenomena. The engineering implementation of CA is 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 [14]. It has been used frequently by physicists to describe 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 shape optimization has been presented by Inou et al. [15, 16], Kundu et al. [17, 18], Xie et al. [19–22], Zhao et al. [23, 24], Yang et al. [25], Young et al. [26] and Kim et al. [27]. 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

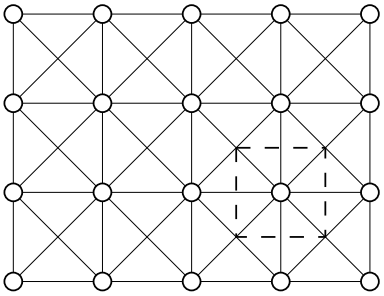

Figure 1: A ground truss lattice

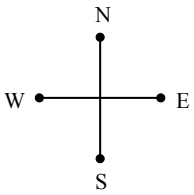

Figure 2: The Moore neighborhood

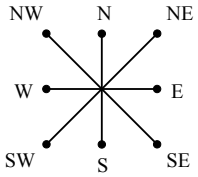

Figure 3: The Von Neumann neighborhood

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 shape and topology of the structures. Other studies involving the use of CA for structural design optimization include Kita et al. [28], Gürdal et al. [29], Tatting et al. [30], and Abdalla et al. [31].

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. 1–3.

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 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.

3.1 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 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 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 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 Boolean Random Networks 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.

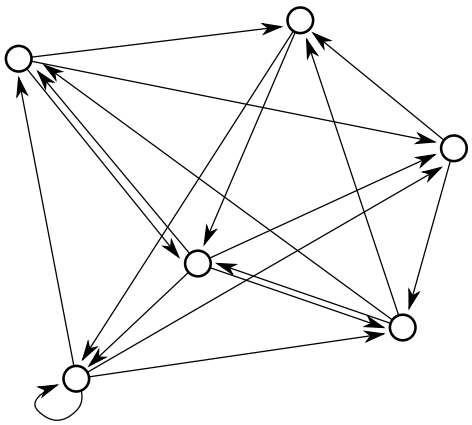

Figure 4: A random Boolean network

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 instead on BRN neighborhood definitions. 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. In direct GA encodings the genotype equals the phenotype.

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 support 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. 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. 6 and 7. In Fig. 6 a new node was connected to the starting truss from Fig. 5 through only two nodes. In Fig. 7 three connections to stable nodes are made. It 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 1. Adding a node with a triple connection leads to significantly lower mass. In the above example the overlapping members of the truss are removed in a systematic way that will be explained in a later section.

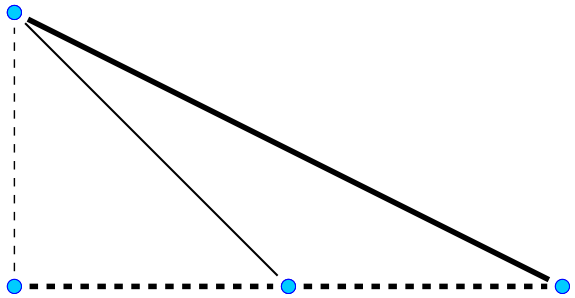

Figure 5: Ten bar minimally connected truss

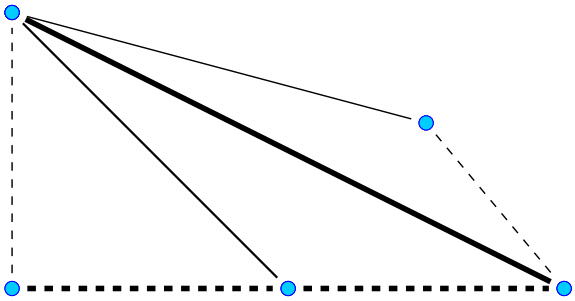

Figure 6: Ten bar truss- with two new connections

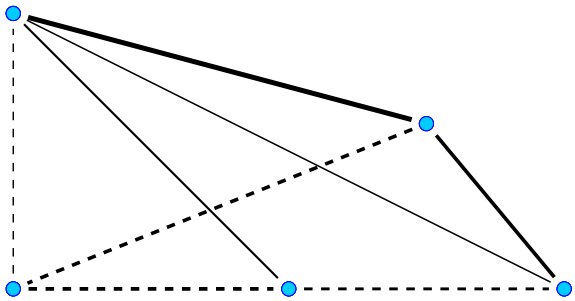

Figure 7: Ten bar truss- with three new connections

Table 1: Ten-bar Truss- Optimal Connections

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

4.1 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 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. 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 $\times$. 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. 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 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

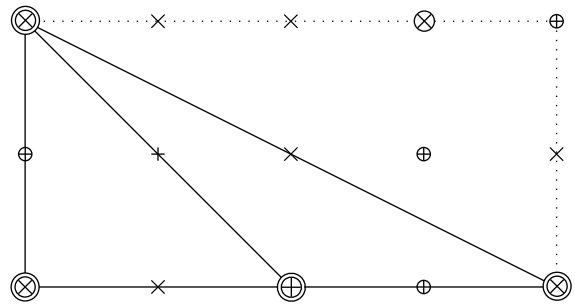

Figure 8: RBN Parametrization

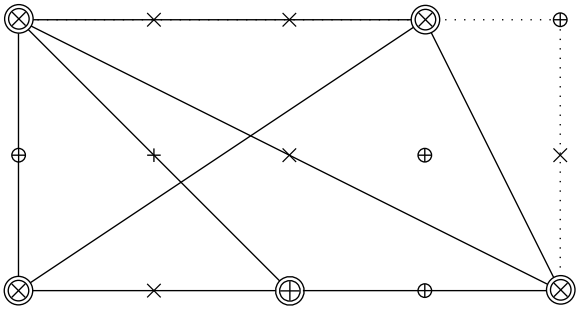

Figure 9: RBN Rule application

topology. Given an initial network topology, in 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 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 10 illustrates this multi-stage methodology. Additional details follow, including definition of

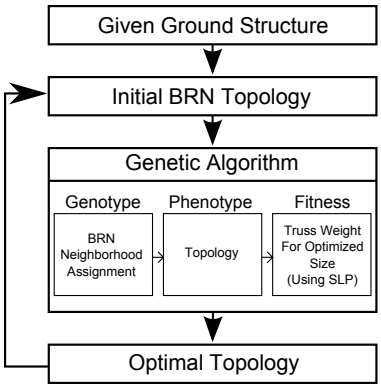

Figure 10: Multistage GA/BRN truss design methodology

the genotype (BRN neighborhood assignment), and a strategy for avoiding overlapping members in generated truss topologies.

4.2 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 (1) illustrates the structure of $\mathbf{x}$.

$$\mathbf{x} = \left[x_1^g, x_1^p, x_2^g, x_2^p, \ldots, x_N^g, x_N^p\right]^{\mathrm{T}} \tag{1}$$

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.

4.3 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 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

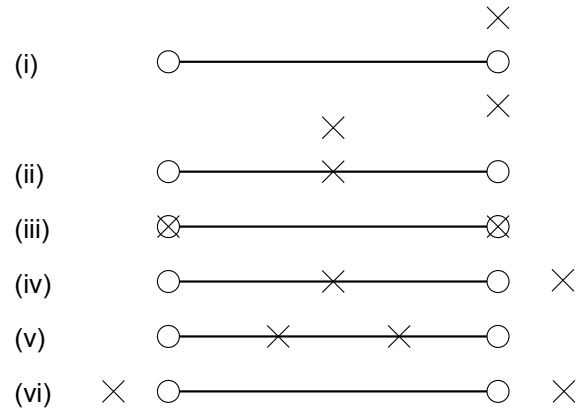

Figure 11: Rules to avoid overlapping members

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 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 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

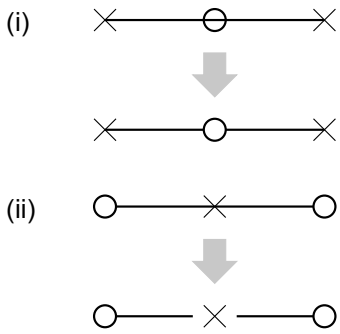

Figure 12: Rules to avoid overlapping members

all the three connections made to an earlier non-connected nodes are parallel the new connections are not allowed.

5 Results and Discussion

The methodology described above is demonstrated using an archetypal truss benchmark design problem. The truss design 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 and size optimization problem is represented mathematically using the equations below.

$$\min_{n,\mathbf{C},\mathbf{A},\mathbf{P}} \quad f = \sum_{\substack{0\le i\le n\\ 0\le j\le n}} \rho C_{i,j} A_{i,j} l_{i,j}$$

$$\text{Subject to:} \quad \sigma_{\min} \le \sigma_{i,j} \le \sigma_{\max} \qquad (2)$$

$$d_{\min} \le d_k \le d_{\max} \text{where} \qquad i,j,k \in \{1,2,\ldots,n\}, i \ne j$$

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 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 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 position vectors of the nodes and cross-sectional areas while satisfying maximum tensile and compressive stress limits of each bar member, and maximum nodal displacement of each node. Stress and displacement values are computed using finite element analysis based on the force method. In the inner-loop geometry and size problems are decoupled and solved using SLP for the same set of stress and displacement constraints. SLP is a recursive procedure where a series of linearly approximated sub-problems is formulated and solved. For individual SLP problem each intermediate solution is the starting point for the subsequent sub-problem. For size optimization problem 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. A lower and upper bound is imposed on cross sectional area values.

For geometry optimization the nodes are allowed to move only in their vicinity. The entire design domain is divided into rectangles where the space between two internal nodes is equally divided and it forms 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 13 illustrates the geometry optimization domain of points on a rectangular design domain which has a total of 24 points equally spaced horizontally and vertically. The concept of allowing the node to move only in their vicinity works because in the topology optimization a node is selected over the other nodes which are outside its geometry optimization domain. Together topology and geometry optimization allows for the search of the entire design domain.

The geometry and size problems are decoupled and solved in a loopy approach sequentially one after the other until the solution converges. However, it is observed that geometry optimization problem is required to be solved only once, i.e., the loop

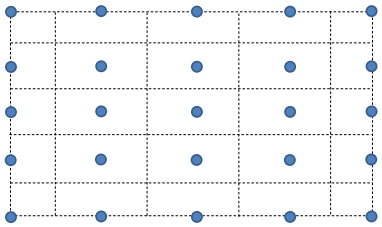

Figure 13: Localized Geometry Optimization Domain

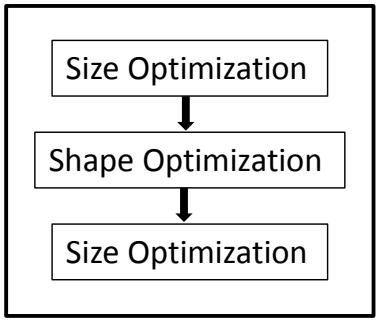

Figure 14: Geometry and Size Optimization

converges in one step. First size optimization problem is solved keeping the initial geometry constant. After that geometry optimization problem is solved and then again size optimization problem is solved. The loop converges in one step as the nodes are allowed to move only in their vicinity. Figure 14 illustrates the loopy approach explained above.

When both the stress and the displacement constraints are included in the formulation, basic SLP does not converge. One effective resolution is to use move limits, where the search domain of each subproblem is limited by optimization variable limits centered on the linearization point. The move limits must not be too large in order to avoid oscillations in numerical solutions, but excessively small move limits may result in slow or premature convergence. Renaud et al. [32] argued that a proper choice of the move limits should ensure that the objective function is always decreasing, the intermediate solutions are always feasible, and that the optimization variable movement is controlled in order to limit linear approximation error. Lamberti and Pappalettere's [33] partial modification of Chen's [34] Constraints Gradient based Move Limits (CGML) algorithm was used here. Scaled move limits are equal for all optimization variables. Lamberti recalculated the move limits until the intermediate solution improved meaningfully and then reduced it by means of an user supplied factor. Here we impose a maximum limit on the number of SLP iterations of 50. Each linear subproblem is solved using the MATLAB® `linprog` function using the interior point method, which ensures that subproblem solution is always feasible.

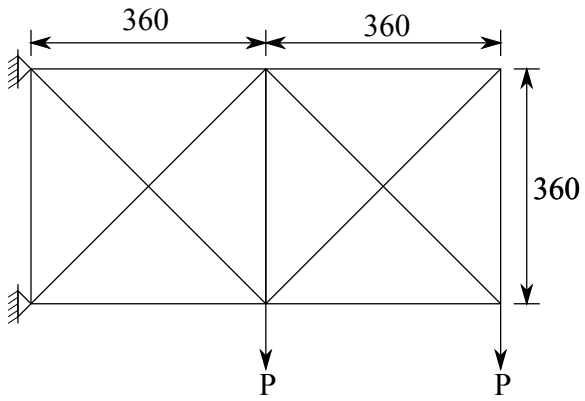

Figure 15: Topology and Geometry of Ten-bar Truss

5.1 Ten-bar Truss

In this example the ten-bar truss of Fig. 15 is optimized for size and topology. This truss was previously analyzed using various mathematical methods (see Refs. [1–3, 33, 35–44]). The results reported in these studies are for size optimization of the standard ten bar topology. The example used here is based on these earlier studies (same design domain, stress limits, etc.), but topology is allowed to vary. Table 2 presents input data for the example problem.

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. 16. 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 Fig. 10. Figures 17–20 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 5 (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 members for each truss development stage is shown in Table 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 lbs.

Table 2: Input Data for Ten-bar Truss

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

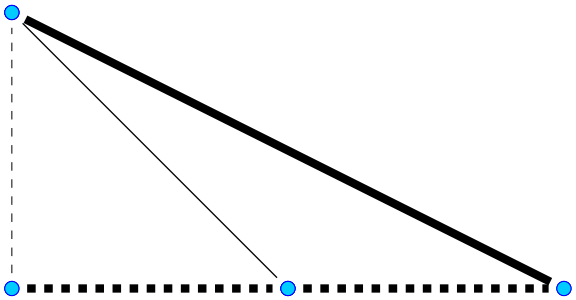

Figure 16: Ten-bar Truss Initial Design-0

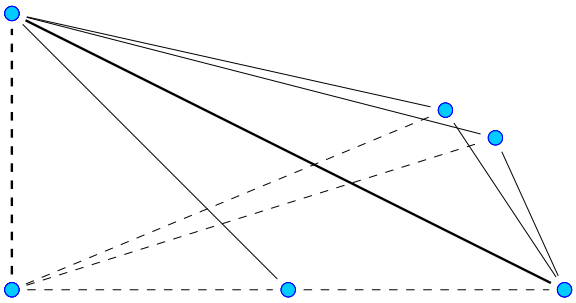

Figure 17: Ten-bar Truss Development-1

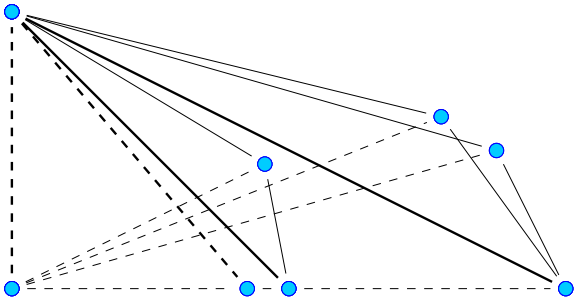

Figure 18: Ten-bar Truss Development-2

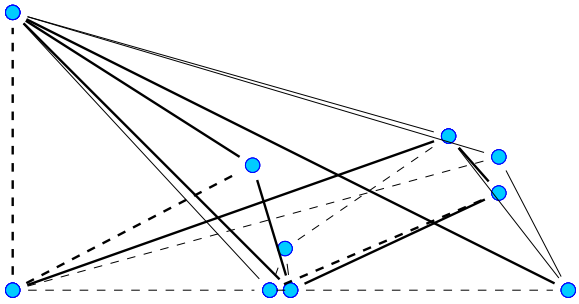

Figure 19: Ten-bar Truss Development-3

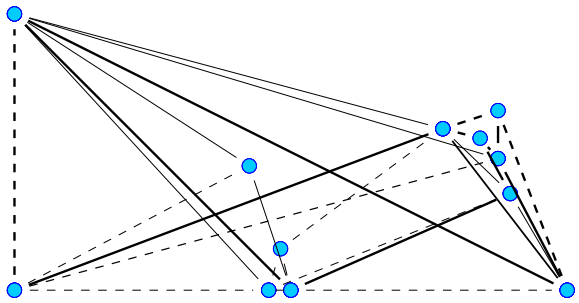

Figure 20: Ten-bar Truss Development-4

Table 3: Ten-bar Truss- Optimal Mass and No. of bars

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

6 Conclusion

A novel parameterization concept using Boolean random vectors was presented for structural truss topology optimization that facilitates the use of genetic algorithms in design of large scale structures. The proposed approach addresses the two shortcomings of direct GA encodes; one the inability to scale up to large-dimension problems and second the inefficiency of exploring infeasible design topologies. The Boolean random vectors based parameterization methodology increases linearly in optimization problem size with respect the number of nodes in the design domains whereas the direct GA encoding increases quadratically. Further a generative growth strategy was presented which exploits the basic concept of truss development, i.e. a new

bar member is added to redistribute load and hence to reduce optimal truss mass. A localized geometry optimization methodology was also presented which together with the proposed topology optimization allows for the search of entire design domain.

Opportunities for future work include studies of GA convergence with respect to the proposed methodology, as well as an investigation of how well the proposed algorithms cover (or access) different portions of the truss topology and geometry design space. In summary, utilizing 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.

Acknowledgements

The authors would like to acknowledge the assistance of Danny Lohan in the preparation of this article, in particular generation of figures used to illustrate the truss design method and example.

REFERENCES

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

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

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

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

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

[6] 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.

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

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] Wujek, B., and Renaud, J., 1998. "New adaptive move-limit man-

agement strategy for approximate optimization, part 1". *AIAA journal,* **36**(10), pp. 1911–1921.
[33] 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.
[34] 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.
[35] Gellatly, R. A., and Berke, L., 1971. Optimal structural design. Tech. rep., DTIC Document.
[36] Schmit, L. A., 1976. "Approximation concepts for efficient structural synthesis".
[37] 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.
[38] Khan, M., Willmert, K., and Thornton, W., 1979. "An optimality criterion method for large-scale structures". *AIAA Journal,* **17**(7), pp. 753–761.
[39] 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.
[40] 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.
[41] 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.
[42] Xu, S., and Grandhi, R. V., 1998. "Effective two-point function approximation for design optimization". *AIAA journal,* **36**(12), pp. 2269–2275.
[43] 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.
[44] 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.