ABSTRACT

TSCHIDA, COLIN E. A Novel Generative Approach for User Directed Design of Space Trusses. (Under the direction of Larry Silverberg.)

In this dissertation we introduce a method for the shape optimization of trusses, as well as a design method for the generation of trusses.

Chapter two introduces a novel approach to truss structure shape design, drawing inspiration from biological processes, that treats each joint or member of a truss as a cell and the entire truss as a multicellular organism. The approach mimics plant tropisms in which, through cellular growth, cells change their size and shape in response to scripted gene commands. Examples illustrate the effectiveness of the approach; it is shown to produce arching and double-arching of pinnedpinned trusses, bulging of cantilever trusses and a plant-like tropic response.

Chapter three introduces a new engineering design algorithm, called the multicellular organism design (MOD) algorithm. The MOD algorithm is well-suited to the early design phase during which there is an interest in generating different topologies. The MOD algorithm switches between topology modification and design parameter optimization to generate feasible designs. The topology modification either adds or subtracts nodes and edges and the design parameter optimization is performed by any suitable optimization method. The MOD algorithm is applied to truss design and several illustrations demonstrate its ability to readily generate common trusses, an optimal truss, and interesting solutions with little computational effort.

Finally, in chapter four we extend the generative procedure outlined in chapter three to produce space trusses. This requires the development of fast optimization routines, parametric representations of truss structures, and refinements of the MOD algorithm. Examples illustrate its ability to produce a tower, a bridge, and explore multiple objectives. A final example demonstrates two trusses generated via user direction.

A Novel Generative Approach for User Directed Design of Space Trusses

by
Colin E. Tschida

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Mechanical Engineering

Raleigh, North Carolina

2013

APPROVED BY:

Scott Ferguson	Pierre Gremaud
Kevin Lyons	Larry Silverberg Chair of Advisory Committee

DEDICATION

To Katie and Dagny.

BIOGRAPHY

The author was born - or was supposed to have been born - in Duluth Minnesota. After ten years, his parents decided that the weather was not going to break and moved to a relatively balmier locale: Rapid City, South Dakota, which provided an ideal location for rock climbing and the development of bad ideas.

He attended Grinnell College where he received degrees in Physics and French and never saw Herbie Hancock (a notable alumnus). After college, his then-girlfriend, now wife, convinced him to move to the Bull City, where he promptly found a job banging on metal and playing the trombone. Due to the short-lived nature of both of these pursuits, he found himself falling amongst a strange people at North Carolina State where he attempted to pursue a career in sustainable energy. Because of the vagaries of the hiring processes he is now completing a PhD in mechanical engineering.

ACKNOWLEDGEMENTS

I would like to thank Larry Silverberg for his advice, direction, and coffee. I would like to thank Chad for even more coffee, although had he not offered it I may have defended months earlier. This dissertation would never have been written had it not been for the support that I provided to Bean Traders and Cup-A-Joe, coffee shops in Durham and Raleigh, respectively. For the distraction they provided I would like to thank my co-workers: a dancer, a welder, a philosophy professor, an IT semi-pro, and Carl the very important novelist.

I would also like to recognize my first job after college for being intolerable. Had it been different, I may have remained a metal-worker.

My wife, Katie, and family have provided necessary distractions and support, without which I would likely have thrown in the towel. I would be remiss if I did not thank my parents for putting up with (and even encouraging) the projects that could have burned down our house. These were probably a necessary step towards my pursuit of a science and engineering education.

I would also like to thank my sister, who subjected my teenage self to countless Fred Astaire and Ginger Rogers movies (in addition to Die Hard and Lethal Weapon). I'm sure this affected me in important unquantifiable ways. Speaking of which, we begin...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Engineering design is a complex process, the arc of which encompasses problem identification, ideation, and rigorous analysis using computational models. Although the later design stages have benefited enormously from advances in computational power and numerical methods, developing computational aids for the early stages of design has proven to be a difficult nut to crack. Problem identification and ideation will remain a human endeavor but it has been noted recently that, despite the potential utility, "there is a void of relevant digital support when it comes to generating new ideas" during the ideation process [15]. Or, to state it differently, "In the design process, computers have become...instruments of rhetoric left to represent anew already designed ideas" [16]. These early stages, when the gross features take form and many different alternatives are considered, is largely the domain of freehand sketches and rough physical models.

In recognition of this, a growing number of researchers have developed software that is intended to compliment the tools used in the early design process, for example Dorta [15] describes an immersive design environment while Krish describes an algorithmic process for incorporating user feedback [32].

Much of this work focuses on form-factor design, for example, Krish et al. (2011) design the case of a cellphone based on user preference [32]. However, in many aspects of engineering design, there is the added difficulty that a novel design must also satisfy some intended function and constraints [47] which is made more difficult because the constraints are implicit (not explicit) functions of the design variables. To further complicate matters, many engineering systems require the integration of many interacting parts and Papalambros, among others, has noted that the toughest design problems involve the determination of a reasonable configuration (topology) of many parts [40].

To overcome the complexities of difficult problems designers often create "external representations" [58] and it is the unexpected results of "reflective conversations with the situation" which often enriches the design process [46]. Because engineering systems are complex, a successful exploratory design tool will likely need to "bury" some of the lower level design choices, allowing the designer to focus on the gross features of the system. To wit, when designing a house an architect might begin by sketching the floor-plan without attention to the plumbing. The pen and paper did not require him to include these features and, at any rate, doing so would have encumbered this early stage of the design process.

Unfortunately, many systems do require attention to details, even early in the design process. The potential exists that a designer may not be able to "see the forest for the trees" when working with such systems. Clearly a method which aids in the exploration of design configurations of complicated engineering systems would be a useful tool for the design engineer.

In this dissertation we attempt to develop a method to aid in the design of trusses, which being nearly ubiquitous, have been the subject of much investigation. The literature on the generation of *optimal* trusses (or truss-like structures) is extensive, however, the literature on their *design* is less well developed. Building on the idea that designers make use of "external representations," we develop a ground structure method, where the ground structure embodies the designer's initial solution concepts, and the initial solution is iteratively modified by the proposed method.

Since the principal aim of this research is to develop methods that aid in truss design, emphasis has been placed on developing (when necessary) or using (when available) existing fast optimization methods that can be used within the context of this goal. To this end, Chapter 2 describes a novel shape optimization procedure. Chapter 3 describes a novel design paradigm which, using methods from Chapter 2, produces novel planar trusses. Chapter 4 extends the methods presented in Chapter 3 to produce space trusses. Finally, we summarize our findings in Chapter 5.

Before we present original work we offer a brief discussion of truss optimization, a survey of the existing literature, and a discussion of the state-of-the-art of truss design.

1.1 Problem Description

The design and optimization of trusses is rooted in analytical methods that treat relatively simple problems and simple cost functions. The ability to formulate these problems was initially limited by a lack of appropriate analysis methodology. Very early work necessarily focused on strength of materials and was pursued by such luminaries Galileo Galilei [22], however, it was not until contributions by Claude-Louis Navier (of Navier-Stokes fame) that analysis methods were mature enough to allow for analytical optimization. Soon after, building on the work of

Maxwell [20] the Australian inventor Michell developed methods to determine optimal truss-like structures [36] marking the beginning of modern analytical methods in 1904.

Because of the complexity of many structural optimization problems (e.g. they may be non-convex, discontinuous, or have constraints which are implicit functions of the design variables) it is common to use heuristic techniques. As such, methods in structural optimization can be divided into mathematical programming (MP) techniques and heuristic techniques. A general overview of the heuristic techniques is given by, among many others, Hare [26] and a review of the mathematical programming methods is given by Vanderplaats [57], [56]. Broadly speaking, derivative based methods should be used when gradient information is available and continuous. Because the method we develop is heuristic, we will focus our attention on heuristic techniques, discussing MP techniques when appropriate.

Finally, truss optimization is often divided into three subproblems: topology optimization, shape optimization, and material optimization. Rozvany [43] notes that if the problem is to be tackled in this fashion the order of operations is important and that "the order of the three ...[should be], logically, topology, geometry, and sizing." In the simplest case, one set of attributes is optimized while the others are fixed. For example, materials may be optimized for a fixed shape and topology. Although the underlying problem is not decoupled this "divide and conquer approach" is common and our discussion of techniques is divided along similar lines.

For convenience, the generic problem may be stated as

$$min \quad f(X) \tag{1.1}$$

$$s.t. \quad h_i(X) \leq 0 \tag{1.2}$$

where the vector X is a vector of components of node locations, $h_i(X)$ is the i^{th} nonlinear inequality constraint, and $i = 1, \ldots, k$ for k nonlinear inequality constraints. Typically, $f(X)$ is the weight of the structure, its compliance, or total strain energy and $h(X)$ consists of maximum stress and node displacement constraints. Another common constraint is a frequency constraint, which generally defines a lower bound on the natural frequencies of the structure, so that the structure will be more robust with respect to natural excitation such as wind or earthquakes.

1.1.1 Shape Optimization

An important subset of structural optimization by heuristic methods is shape optimization. GAs were applied here by, among others, Deb et al. [14] and more recently Fraternali et al. [19]. The PSO and its variants were applied to shape optimization by Fourie [18] and Jansen [29], SA was used by Shim [50], and CMLPSA was recently developed by Lamberti [33]. These methods, from the simple analytical methods to the multi-objective, evolutionary algorithms, use a well-defined

performance function when, in fact, structural design by optimizing a well-defined performance function is not always possible. In many structural design problems, it is impractical to find an analytical function that captures all of the components of a desired performance. Whereas certain components of the performance can be captured analytically, like the desire to be strong or stiff, other components are difficult to quantify, like aesthetic qualities and those features of a solution that are initially unknown [39]. Such is the case in learning algorithms, like the one developed for a mechanism described by Silverberg and Gardner [51]. Indeed, there is a need for an approach that overcomes the difficulty in structural design problems with requiring, in advance, a well-defined performance measure.

Recently, more complex problems were treated numerically by heuristic approaches that draw inspiration from natural processes, like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). A comprehensive review of heuristic methods can be found in [54] and the development of new approaches continues, for example, the Charge System Search [30]. Applications to structural design are summarized by Ohsaki [39] and Bendsøe [8] and the state-of-the-art is summarized by Saka [45] and Kicinger et al. [31].

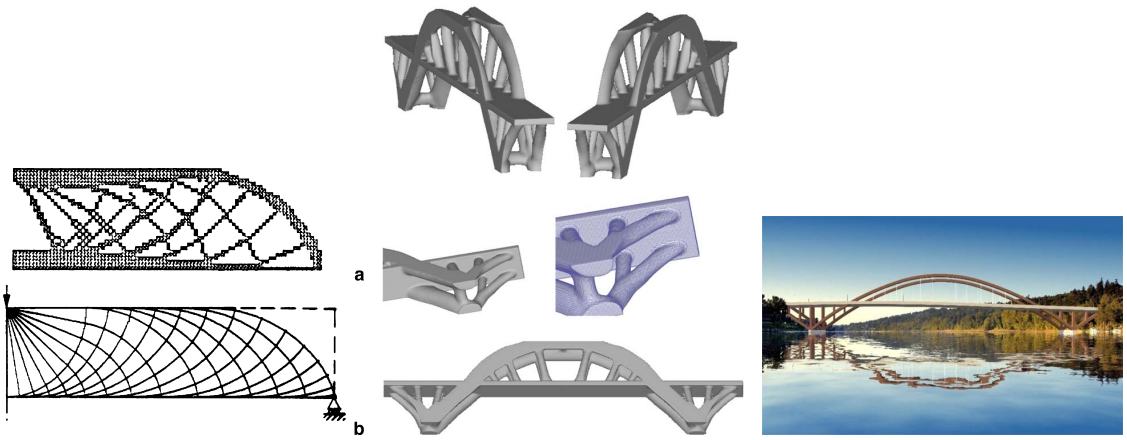
In chapter 2 we introduce an approach to truss structure shape design that is motivated by tropisms a biological feature prevalent in plants. In plant structures, tropisms ". . . are growth curvatures of plant structures. . . in response to environmental stimuli. . . [where] the curvatures result from differential elongation of the cells" [25]. Tropisms are a type of cellular growth consisting of the adaptive process that a cell undergoes when it changes its size and shape in reaction to stimuli. The process alternates between globally coordinated cell changes and locally coordinated cell property changes. The sequencing of these global and local changes is unknown in advance since the plant is only aware of its current state and may encounter environmental stimuli or constraints that call for modification. By analogy, the approach introduced in this chapter recognizes that at times it is desirable to effect global changes in node positions and at other times to make local refinements, and that the sequencing of these algorithms may be accomplished by a suitable algorithm or by inspection of the optimization process. Two cellular algorithms are developed in this chapter. They represent a local building block and a global building block and the mixing of the two building blocks is a kin to the sequencing of algorithms.

1.1.2 Material or Sizing Optimization

Material optimization is perhaps the simplest of the three subproblems; determine the composition of each member - material, cross-section, etc. - that minimizes $f(X)$. The design variables, X, represent the material properties. As such X may be continuous or discrete. If it is continuous it may simply represent a cross-sectional area for a fixed material. If it is discrete it may represent possible cross sections (circular, annular, I-beam, etc), or commonly available building

materials. This problem has been thoroughly investigated and it is tackled by both derivative based and heuristic approaches. For example, Vanderplaats [56] discusses gradient techniques for material optimization while Perez [41] discusses application of a Particle Swarm Optimization algorithm (PSO). As with many other problems, the choice of method (derivative-based or heuristic) depends on its size and complexity. Generally speaking, if the only goal is material optimization and material properties may be varied continuously, than derivative methods are preferable, and as Vanderplaats notes, the derivatives are relatively easy to obtain analytically. That being said, if the problem requires a discrete material palette than a heuristic method is the logical choice. The methods discussed in the previous section are generally applicable to this problem and many papers that consider shape optimization of truss structures also consider material optimization, albeit in separate examples.

1.1.3 Topology Optimization



(a) (b) (c)

Figure 1.1: (a) Analytical solution for a Michell beam (top) and an early SIMP solution (bottom). (b) A related solution by Nguyen, using a modern SIMP method and (c) an artist's rendition of a bridge over a river.

As noted earlier, topology optimization of truss structures has its origins with the pioneering work of Michell. He developed a paradigm for truss design in which the weight of the structure is minimized subjected to stress constraints, and material selection is limited to a single material type for compression and one for tension. These methods generate fascinating structures but their use is not widespread. This is because the optimal designs often have difficult-to-manufacture topologies or require an infinite number of members. Another handicap of these

methods is their use of a simplified material palette.

Research into analytical topology optimization is ongoing, e.g. Prager and Rozvany extended Michell's work in the seventies and eighties and Melchers (2005) summarizes recent developments [35]. However, the limitations of these techniques has led to interest on computational methods, a description of which is given by Rozvany [43] where he focuses his attention on those methods which currently find application in industry.

The "material removal" approach was pioneered by Bendsøe [7] and later christened SIMP - Solid Isotropic Microstructure/Material with Penalization (for intermediate densities). This works as follows: imagine we wish to design a cantilevered beam. We would begin with a rectangular plate (subject to boundary conditions), perform FE analysis, and then remove material where stress is low. This method does not generate trusses but perforated plates and, as with any FE procedure, a large number of elements may be required to provide reasonable results. An example of this method is shown in Figure 1.1(a) (from [43])and research into this method is ongoing. For example, Suresh [53] and Nguyen [37] develop 3D methods which reduce the computational effort. As an example of the state of the art, Nguyen produces truss-like structures in 3D Figure 1.1(b). Finally, it is interesting to note that Bendsøe has shown that solving the compliance-minimization problem for plates in plane stress produces optimal, least-weight trusses (in the Michell sense) [9].

A method similar to SIMP is termed ESO (Evolutionary Structural Optimization), also known as SERA (Sequential Element Rejection and Admission). Similar to SIMP, ESO works by pixelating the domain and removing material where stress is low, the primary difference being in the complete removal of an element as opposed to the thickness minimization method employed in SIMP. This results in a number of differences in implementation, use and results, a brief discussion of which is given by [43].

A class of algorithms which has been used in configuration exploration is called a *generative system* [34]. These are algorithms that use iterative application of a simple rule set to evolve. Examples of generative systems include L-systems and cellular automata and they have already been used for topological reasoning, optimization, and exploration [52]. For example, Caldas [10] applies a generative system with a GA to design buildings while Hornby [27] designs a table via a generative process.

In Chapter 3 we present a simple, generative system that is intended to aid the engineer in exploring alternative topological forms early in the design process. At this stage, an engineer often has a rough but reasonable notion of the system constraints. The described method allows the engineer to "sketch" a rough form of a design, subject to any constraints, and watch the unique development of that design under a generative process. The intent is that the resulting system or its generative path will provide fresh ideas which may inform the design process. The method is motivated by a biological metaphor and is referred to as Multicellular Organism

Design (MOD).

1.1.4 Simultaneous Optimization

In recent years it has become more common to treat the truss optimization problem more holistically, combining some number of the three sub-problems described above. When all three are considered together it is often termed the "layout" problem. Rozvany notes that it is becoming common to consider topology optimization and shape/sizing separately [44] (pp 77 and 122). Moreover, shape and sizing optimization are more readily combined since both may be continuous optimization problems. However, there are potential difficulties in implementation since the shape and sizing variables may be orders of magnitude apart, potentially resulting in numerical ill-conditioning. An interesting approach is provided by Wang et al. [60] where they also include frequency constraints. However, it is more difficult to include the topology optimization problem without a little more ingenuity. A relatively early algorithm which accomplishes this is due to Rozvany and Zhou [62] who present an iterative, continuum based method. One method that is featured heavily in the literature is the material removal problem, which we discussed in the previous section; the state of the art of this method is show-cased by Nguyen et al. [37] wherein they develop sophisticated meshing algorithms to allow for computationally tractable solutions to a relatively large 3D example. Examples of methods that treat all sub-problems with discrete (true) trusses is given by Bailey et al. [5] and Agarwal et al [2].

As with any problem, when increasing the complexity you generally increase the computational cost, and this is certainly the case with the layout problem. This may be alleviated, to some extent, by noticing that although many structural optimization problems are globally discrete and/or non-convex they are locally smooth, continuous, and convex. To this end, there are interesting examples of "hybrid" methods that use a heuristic method to "ballpark" the solution and then use a derivative-based method to refine the design. For example, Plevris et al [42] develop a hybrid method with Sequential Quadratic Programming and PSO to tackle small truss optimization problems.

1.1.5 Algorithmic Design

As noted earlier, the literature on design of trusses (incorporating user preference) is limited. Of the published methods, the most prominent fall into the category of Interactive Evolutionary Computing (IEC), wherein user preference is incorporated into an Evolutionary Algorithm, such as a GA. An early example of this is provided by Furuta (1995) wherein he designs arch bridges [21] with pre-determined user preferences (e.g. on aspect ratio). Currently a popular method is to have a user choose designs at the selection step in a GA.

Other authors attempt to develop new GA operators that include user preference, e.g.

Astudillo et al develop a cross-over operator for the design of long-span roof-trusses [4]. Another example of IEC is provided by Bailey et al., who present a method to determine and then incorporate user preference into a MOGA with which they generate roof trusses [5]. (Bailey also includes a good summary of the application of GA to other design problems). Significantly, Bailey notes that

> Although easily quantified, structural performance is not the only requirement placed upon large-span roof truss systems during the conceptual design phase. Constraints such as conformance to the architectural program, constructibility, and overall economic feasibility may play equally important roles in selecting the final design. However, although such intangible criteria are often apparent to a human user based upon visual inspection of the design alternatives, formulating these constraints in terms accessible to the GA is mathematically intractable.

Of course, nothing is unique about a GA; inability to define what Bailey terms *intangible criteria* is a common problem which must be addressed for any computational design methodology. Unfortunately, the ambiguity inherent in defining the intangible criteria means that the performance of the design can be difficult to quantify. Perhaps this is why the engineering optimization community has largely shied away from this problem?

Another interesting set of approaches use what are termed *shape grammars*: Essentially, these methods iteratively apply a simple rule-set to a design representation to obtain new designs and their application to architectural and engineering design is discussed by Antonsson et al. [3]. Shea [48] [49] uses a modification of this approach called shape annealing to design roof trusses while "meeting the needs of designers with varying intent...presenting spatially intriguing, yet functional, structures that expand the range of designs considered in the conceptual design stage."

Off in left field, an example making use of advances in computational power and mobile computing is provided by Aage et al.[1] wherein they develop a topology optimization "app" for tablets. Their stated goal is to develop "fast and efficient optimization code such that the user will experience that the structure will respond quickly, i.e. become alive, when loads or supports are moved around, added or removed." This work is interesting not just because they choose to put the designer in the loop with an optimization process but they choose to do so in a novel environment to facilitate more hands-on exploration of the design space. This is the sort of "reflexive conversation" of which Schon [46] would approve.

Finally, it is clear that some pure optimization algorithms may be used for design. The SIMP methods, for example, might be adapted for design by starting with odd-shaped domains. We see the hand of a designer in the work by Nguyen et al: It is no accident that their method produced the bridge in Figure 1.1(b) that is so similar to the example in Figure 1.1(c). The

designer had to choose the appropriate boundary conditions and domain to allow their algorithm to produce that example.

Notice that there are two broad categories of design methods. Methods that require user interaction during the solution process (e.g. Aage) and methods that require user interaction only for problem specification (e.g. Nguyen). Let us term those methods that allow continued interaction "dynamic" and those that do not "static." Thus, hand sketching is a dynamic process while traditional optimization is static. In order to encourage a "reflexive conversation" (Schon, [46]) it seems likely that a dynamic method is preferable.

Finally, note that I have used the word *direction* in the title to this dissertation while in the previous paragraph I have used the word *interaction* in reference to design methods. The difference between "direction" and "interaction" is simple, yet subtle: you *interact* with an algorithm to *direct* the course of the design. If the designer chooses to modify a design in an computational design environment they *direct* the algorithm to make that change; the act of making the change is *interactive* but the choice of change indicates intent and *direction*. In other words, *direction* is the result of the user's choices imposed via *interaction*.

1.2 Summary and conclusions

Truss design and optimization is common to mechanical, aerospace, structural, and civil engineering. Although the optimization problem may be succinctly stated (e.g. minimize weight subject to constraints), this simple formulation is not always completely representative of the design problem. However, continued advances in structural optimization techniques, widespread availability of computational resources, and a general acknowledgment that simple cost functions do not necessarily represent all of the important aspects of the underlying problem have led to the development of computational methods for *design* as well as *optimization*.

This raises the following question: **can an appropriate computational aid be developed that allows a designer to direct and dynamically explore engineering systems?**

In order to facilitate the design process, we contend that computers should be an exploration tool, not just an analysis tool. It is not practical or desirable for design exploration to be handled exclusively by a computer since the *designer* is the one interested in the unquantifiable aspects of the design, not the computer. As a result, user preference is a necessary component of a realistic design process and allowing the user to interact with a computational design aid provides an effective method for incorporating their preferences while allowing enhanced exploration of the design space. As a natural result of incorporating cost-external information (in the form of user preferences) it must be acknowledged that design problems are multi-objective. We will show that generative systems provide a format that incorporates all of these goals.

The remainder of this dissertation presents truss design methods that attempt to answer

this question.

CHAPTER 2

CG ALGORITHM

In Chapter 1 we reviewed the available literature on truss optimization. A limitation of current methods, which occurs in high-dimensional design spaces common in structural design, is concerned with the ability to take advantage of the nature of the design space. Algorithms like GA, and SA, since they are general purpose, only take partial advantage of the characteristics of a design space. To tailor to design spaces of particular problem types, algorithms containing customized local and global building blocks are needed, as Lamberti [33] and Fawaz et al. [17] show.

In this chapter we present a novel method for shape optimization of trusses which uses unique attributes of trusses to help steer the optimization process.

2.1 Methods

A truss structure is a set of m pin-connected, one-dimensional members of length L, cross-sectional area A, and elastic modulus E that traditionally functions as a means of supporting external loads at its pin connections. The truss is designed to be strong, stiff, and sometimes to serve as a dynamical mechanism. The truss is governed by a system of nonlinear equation described in Appendix A. Solution of these equations may be accomplished via standard, derivative based optimization procedures. However, the system may be linearized to admit quick solution permitting more rapid system evaluation. The set of the linear algebraic equations

$$A\mathbf{x} = \mathbf{b} \tag{2.1}$$

where

$$A = \begin{bmatrix} 0 & 0 & -B(\mathbf{r}) & -C(\mathbf{r}) \\ 0 & I & -D & 0 \\ B^T(\mathbf{r}) & I & 0 & 0 \\ C^T(\mathbf{r}) & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{pmatrix} U \\ u \\ P \\ R \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} F \\ v \\ 0 \\ 0 \end{pmatrix} \tag{2.2}$$

and $r = [r_1, r_2, \ldots r_n]^T$ is the $3n \times 1$ vector of initial joint positions in which $r_i = [x_i, y_i, z_i]^T$ $(i = 1, 2, \ldots, n)$ is the initial position vector of the i-th joint, and where F is a $3n \times 1$ vector of force loads at joints, v is a $m \times 1$ vector of thermal displacement loads in the members, U is a $3n \times 1$ vector of joint displacements, u is a $m \times 1$ vector of member displacements, P is a $m \times 1$ vector of internal forces, and R is a $p \times 1$ vector of external force reactions. The first set of equations enforces $3n$ equilibrium equations, the second set gives the constitutive equations, the third set represent m member-joint displacement equations, and the fourth set gives p external constraints. In Eq. 2.1, B is a $3n \times m$ transformation matrix between global and local coordinates, C is a $3n \times p$ constraint matrix associated with the reactions, D is a diagonal matrix of member compliances, and I is the identity matrix. The solution is $\mathbf{x} = A^{-1}\mathbf{b}$ in which A is invertible when the truss is stable.

Inspired by biological systems, each member or joint is analogous to a cell and the structure to a multicellular organism. The algorithms introduced in this chapter, which will be referred to as cellular growth (CG) algorithms, mimic the general features of cellular growth and plant tropisms: (1) Joints may move locally akin to a cellular maturation process, (2) Members may change their length, akin to a plant tropism, (3) Changes occur subject to constraints on cell size, and (4) Joint and member mechanisms are triggered by organism performance in its environment. In this chapter the quantifiable component of the cost reflects the desire to maximize strength. The cost functional is given by

$$J = \sum_{1}^{m} \mathbf{P}^T \mathbf{P} \tag{2.3}$$

This measure results in shape changes that tend to reduce internal loads in the members as a whole. This cost function is identical to the weighted sum of member strain energies:

$$J = \sum_{1}^{m} w_i U_i = \sum_{1}^{m} w_i \left(\frac{A_i E_i}{2L_i} \right) u_i^2) \tag{2.4}$$

where $w_i = 2A_iE_i/L_i$. Other measures that exhibit similar tendencies, although not considered

in this chapter, are

$$J = \sum_{1}^{m} |P_i| \quad \text{and} \quad J = max(P_i) \tag{2.5}$$

The feasible design space is constrained by upper bounds on member lengths, minimum proximities of joints to each other, and by obstacles, written as inequality constraints in the general form $\mathbf{h} \leq 0$ which are further discussed in Appendix B.

Two basic CG algorithms are developed. The first algorithm, which changes joint locations, is referred to as the CG joint algorithm and the second, which changes the lengths of a set of members, is called the CG member algorithm. The joint and member algorithms can be used in isolation or mixed. For simplicity, only shape design is considered in this chapter. The members mass per unit length, cross-sectional area, and elastic modulus will be fixed. This problem reflects the limited yet important class of shape design problems that arises in systems for which the materials have already been selected, or can be viewed as one step in multilevel design optimization. We now describe in detail CG joint and member algorithms.

2.1.1 CG joint algorithm

Using the CG joint algorithm, a single joint is selected at random. The initial vector of joint positions $\mathbf{r}^0$ produces the ground structure. At the k^{th} step the location of a selected joint is changed thereby changing the lengths of neighboring members. The candidate structures are selected by considering a cloud of p points surrounding the selected joint. The radii R of the points are uniformly distributed in the interval $[0, R_{max}]$ and their polar angles h are uniformly distributed in the interval $[0, 2\pi]$. As shown in Fig. 1, the lengths of the non-neighboring members are unchanged. The CG joint algorithm is written as

$$\mathbf{r}^{k+1} = \mathbf{r}^k + {}^J\mathbf{U}^{k+1} \tag{2.6}$$

where

$${}^J\mathbf{U}^{k+1} = \begin{cases} R[\cos\theta, \sin\theta]^T \delta_{ij}, & J_{k+1} < J_k \ \text{ and } \ \mathbf{h}(\mathbf{r}^{k+1}) \leq 0 \\ 0, \text{ otherwise} \end{cases} \tag{2.7}$$

so that ${}^j\mathbf{U}^k$ is the joint-displacement vector for the j^{th} joint at the k^{th} step in which j is the index of the randomly selected joint and δ_{ij} is the Kronecker-delta function ($\delta_{ii} = 0$ and $\delta_{ij} = 1$

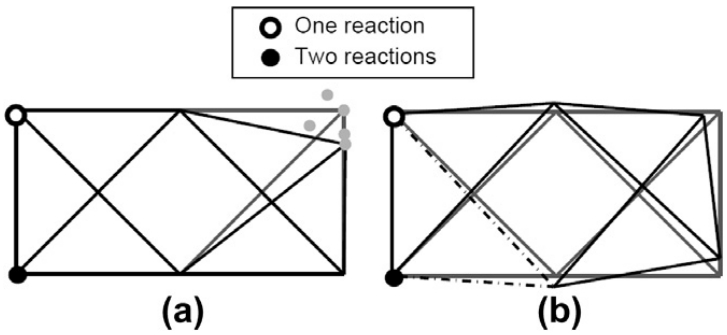


Figure 2.1: (a) CG joint algorithm and (b) CG member algorithm. (original structure in gray).

for $i \neq j$). In Eq. 2.6 we see that the joint algorithm has the form of a standard update procedure in which R is a random search magnitude and $[\cos\theta, \sin\theta]$ is a random update direction. Notice in Eq. 2.6 that the joint algorithm, for simplicity, handles infeasible solutions by rejecting them, after-the-fact. This penalty method is discussed in a review by Coello [12] where he refers to it as the "death-penalty." He notes that despite it's simplicity it can be useful in problems where the feasible region is a significant fraction of the search space and the performance landscape is convex within that region.

2.1.2 CG member algorithm

Using the CG member algorithm, a set of q members is selected at random at the k^{th} step. The selected members undergo thermal expansion. The level of thermal expansion of the i^{th} member is expressed in terms of a growth factor g_i^k as $v_i^k = g_i^k L_i^k$ $(i = 1, \ldots, m)$ in which L_i^k is the length of the i^{th} member in the unloaded structure. The selected members are assigned different (unshared) growth factors or one (shared) growth factor, and the members not selected are assigned a growth factor of 0. Whether shared or unshared, the growth factors of the selected members are uniformly distributed over the range $[g_{min}, g_{max}]$ and one candidate structure is produced. The growth factors produce the load ${}^M\mathbf{b}^{k+1}$ and the resulting solution $\mathbf{x}^{k+1} = A^{-1\,M}\mathbf{b}^{k+1}$ which, in turn determines the joint displacement vector ${}^M\mathbf{U}^{k+1}$ and then the updated position $\mathbf{r}^{k+1} = \mathbf{x}^{k\,M}\mathbf{U}^{k+1}$. The CG member algorithm is written as

$$\mathbf{r}^{k+1} = \mathbf{r}^k + {}^M\mathbf{U}^{k+1} \tag{2.8}$$

where

$$ {}^{M}\mathbf{U}^{k+1} = \begin{cases} \text{from } \mathbf{x}^{k+1} = A^{-1\,M}\mathbf{b}^{k+1} \;\; J_{k+1} < J_k \;\; \text{and} \;\; \mathbf{h}(\mathbf{r}^{k+1}) \leq 0 \\ 0, \text{ otherwise.} \end{cases} \tag{2.9} $$

The growth factor is typically close to zero causing a small change in member length. In Fig. 1 the member algorithm considered $q = 2$ members, shown as dashed lines. The most obvious difference between the joint algorithm and the member algorithm is that the joint algorithm causes local changes in the structure whereas the member algorithm causes global changes throughout the structure. The results will later demonstrate that the joint algorithm (since it causes local changes) can get the solution caught in a local minimum and that the member algorithm (since it causes global changes) can dislodge the solution from its local minimum.

2.1.3 Triggering CG joint and member algorithms

The unquantifiable component of the performance is guided by a set of flat curves or surfaces, called obstacles, reminiscent of the method described by Ohsaki [38]. None of the members are allowed to intersect an obstacle. In two-dimensional analysis, the obstacle is simply a set of line segments and in three-dimensional analysis the obstacle is a set of triangular surfaces. The obstacle can be used to (1) mimic physical obstacles, or (2) to direct the design away from undesirable regions and toward desirable design regions while the algorithm is optimizing for strength. The triggering mechanism can either terminate the changes in an algorithm all at once (global) or can terminate them member by member (local). If terminated all at once, the triggering mechanism could simply be a lower bound on the cost functional or the termination could be performed at the discretion of a designer. If terminating locally, the triggering mechanism could simply be a lower bound on the contribution of the member to an overall change in the cost functional. Members contributing less to the optimization could be considered to have locally converged. This chapter considered global triggering consisting of random mixing between the CG joint and member algorithms. The mixed joint and member algorithm is expressed as

$$ \mathbf{r}^{k+1} = \mathbf{r}^{k} + \beta^{J}\mathbf{U}^{k+1} + (1-\beta)^{M}\mathbf{U}^{k+1} \tag{2.10} $$

where $\beta = 1$ or 0. When $b = 0$ the member algorithm updates r and when $b = 1$ the joint algorithm updates r. In summary, the optimization problem is to minimize J subject to the constraints $\mathbf{h} \leq 0$ (described in detail in Appendix B). The optimization is performed using a mixture of the CG joint and member algorithms according to Eq. 2.10.

2.2 Results and discussion

The members of the trusses have an elastic modulus of $210,000$ GPa, an outer radius of 0.219 m, an inner radius of 0.203 m, and a mass per unit length $\mu = 42.40$ kg/m. The triggering mechanism terminated the optimization globally when the average cost J over the last 50 steps changed by less than $\epsilon = 10^{-8}\ N^2$. The CG algorithms were developed by studying pinned-pinned trusses and cantilever trusses - common systems used also by Ohsaki [39] and Bendse [8]. In all of the cases, unless otherwise noted, the growth factor was set to $g_{min} = -0.01$, $g_{max} = 0.01$, $R_{max} = 0.05$ m, and $p = 2$. Feasible solutions required a maximum member length of L_{max} and a minimum distance L_{min}. In all of the examples we set L_{min} to 0.8 times the smallest original member length. The results below show the different truss configurations that can be obtained, algorithm tendencies, and illustrate how designs are directed. We let $p = 2$ since no gain was observed by selecting larger p. Larger values of R lead to instabilities when p was not further increased. The relatively small values of p and R essentially preserved the local nature of the joint algorithm. L_{max} and L_{min} were likewise selected to preserve algorithm stability. For example, without these constraints, the pinned-pinned truss slowly converges to a very thin catenary-shaped truss.

2.2.1 Pinned-pinned trusses

In all of the pinned-pinned truss examples the member lengths were limited to a maximum length L_{max} of 1.2 times greater than the largest original member length. In Fig. 2.2, each arrow refers to an optimization pathway. Referring to pathways (a) and (b), when starting with a ground truss ($C0 = 1.71010\ N^2$), the joint algorithm produces an M-shaped truss ($C/C0 = 0.54$) and the member algorithm produces an arch-shaped truss ($C/C0 = 0.16$). When starting with the M-shaped truss (produced by the joint algorithm), the truss shape does not change ($C/C0 = 0.54$, pathway (c)). When starting with the arch-shaped truss (produced by the member algorithm), the joint algorithm does not change the shape of the truss either ($C/C0 = 0.14$, pathway (d)). For simplicity, we used a 50% mixture between the member algorithm and the joint algorithm although convergence can be accelerated by changing the mixture (not performed here). The 50% mixture of joint and member algorithms changes the truss from the ground truss to an arch-shaped truss and it has the lowest cost ($C/C0 = 0.13$, pathway (e)). This study shows that CG algorithms can produce two types of pinned-pinned trusses, that the joint algorithm can get caught in a restrictive design space, and that the member algorithm can get the truss out of a restrictive design space. The ability of the joint algorithm to get caught in a restrictive design space and the ability of the member algorithm to get a solution out of a restrictive design space (whether in isolation or in concert with the joint algorithm) is consistent with the joint algorithm providing local changes and the

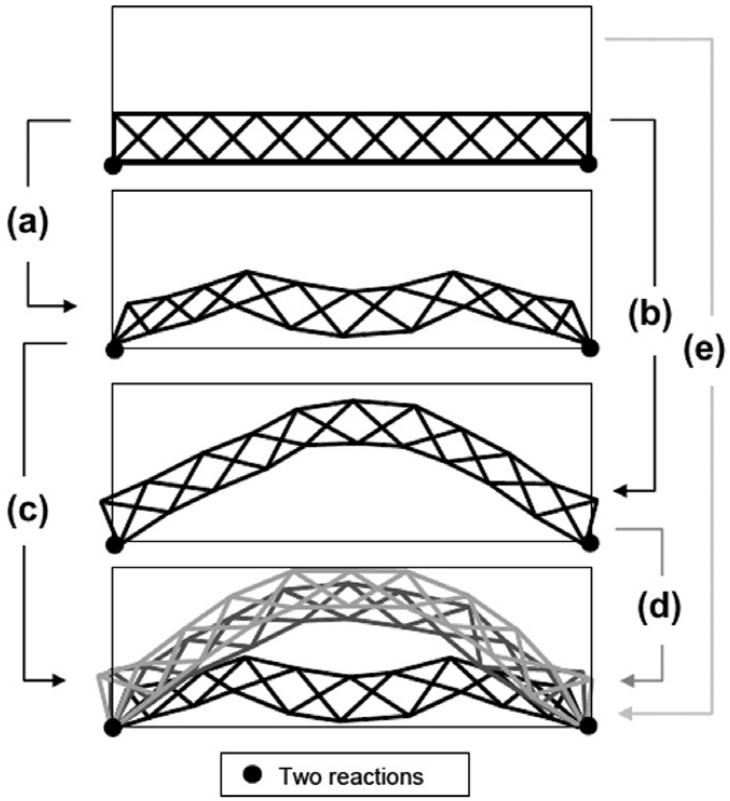


Figure 2.2: Pinned-pinned truss (a) joint algorithm, (b) member algorithm, (c) member algorithm starting with a as an initial condition (black), (d) joint algorithm starting with b as an initial condition (dark gray), (e) 50% mixture of joint and member algorithms (light gray).

member algorithm with providing global changes. In Fig. 2.3, the same ground truss is shown with geometries now optimized under two different loading conditions. Loading condition (a) consists of a distributed weight totaling 4×10^5 N and 3 equal point loads totaling 4×10^5 N and loading condition (b) consists of a distributed weight totaling 4×10^5 N and 9 equal point loads totaling 4×10^5 N. The optimized shape noticeably changed when the structure was acted on by 3 point loads. On the other hand, the optimized shape, when acted on by 9 point loads, is very similar in appearance to the structure acted on by just the distributed weight (shown in Fig. 2.2d) because the 9 point loads are distributed along the bottom of the chord. These results compliment the results obtained by Ohsaki [39] in a study of a pinned-pinned beam.

Next, an obstacle is placed above the mid-portion of the pinned-pinned truss, as shown in Fig. 2.4. The left column shows the truss and the right column shows the corresponding history of the normalized cost ($C/C0$).

As shown in Fig. 2.4b, the joint algorithm converges when the truss reaches the obstacle (See the performance history on the right side). Fig. 2.4c shows that the member algorithm produces

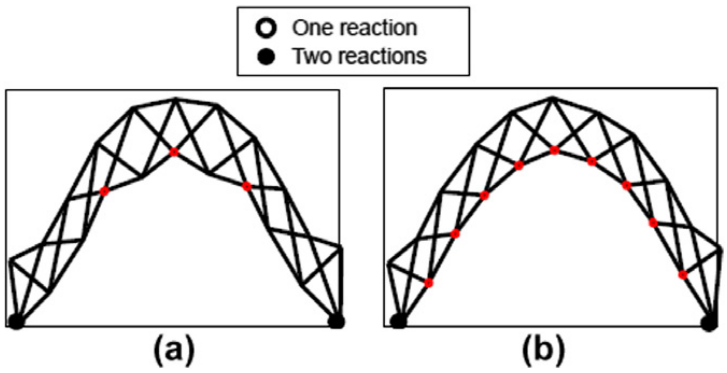


Figure 2.3: Pinned-pinned truss optimized by 50% mixture: (a) loading consists of member weights and three point loads, (b) Loading consists of member weights and nine point loads.

a reasonably symmetric arch that retains the basic features of the initial truss. Its performance history shows two jumps in improvement, resulting from steps that had particularly effective member combinations. Notice that the member algorithm finds the constraint very quickly (within 200 steps) but has difficulty in making local changes so it reaches a cost of 0.54 whereas those examples in Fig. 2.4(b-e) reach a value of at least 0.37. The differences lie in the way the obstacle interacts with the truss when the truss reaches the obstacle. In Fig. 2.4d, the obstacle constraint is enforced at each step by simply removing the constraint violators from the set of candidate structures. This produces similar results as the joint algorithm. Referring to the performance history, we see that the largest improvement occurred in the beginning of the algorithm. After that, the improvement is gradual as the joint algorithm refines shapes locally. In Fig. 2.4e the joint that made contact with the obstacle is constrained when that joint reaches the obstacle. The truss flexes around the obstacle producing a double-arch-shaped truss. Its performance history is similar to the performance history in Fig. 2.4d. In Fig. 2.4f, the joint that makes contact with the obstacle is vertically supported by the obstacle when that joint reaches the obstacle. The vertical support clearly changes the trusss shape and the problem. Finally notice in all of the cases that the member algorithm accelerates convergence.

2.2.2 Cantilevered trusses

In the cantilever truss examples below the member lengths were limited to a maximum length L_{max} of 1.5 times greater than the largest original member length. First consider the cantilevered truss pinned at the lower left joint and sliding at the upper left joint, as shown in Fig. 2.5. Line obstacles were placed below and above the truss. The joint algorithm produces a symmetric truss that bulges on the left, as shown. The member algorithm produces an awkward looking structure that lazily slumped against the bottom boundary. The final pane in Fig. 2.5 compares

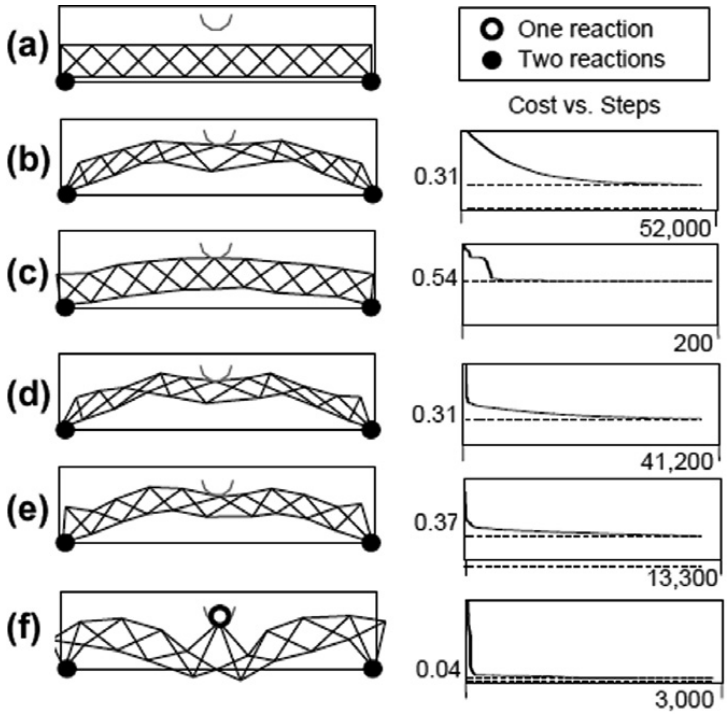


Figure 2.4: Pinned-pinned truss with obstacle: (a) initial truss and obstacle (gray). (b) Joint algorithm, (c) member algorithm, (d) 50% mixture of joint and member algorithms, (e) 50% mixture of joint and member algorithms with joint in contact with the obstacle constrained after reaching the obstacle, (f) 50% mixture of joint and member algorithms with joint in contact with obstacle sticking (provided vertical support) after reaching the obstacle.

three different composite pathways. Letting the joint algorithm follow the member algorithm produces a truss that retains the bulge feature but also slumps against the bottom boundary. On the other hand, when the member algorithm was allowed to follow the joint algorithm the truss retains all of its features from the joint algorithm. Finally, when a 50% mixture of the joint and member algorithms is considered, we get the same features as the truss optimized by the joint algorithm followed by the member algorithm. The study of the cantilever truss revealed the same properties as those found for the pinned-pinned truss. In both studies the joint algorithm and the member algorithm could converge to different truss configurations, the joint algorithm converged in a restrictive design space and the member algorithm was capable of pushing the truss out of a restrictive design space. Fig. 2.6 examines the cantilever truss in more detail, in particular the growth factor. The growth factor refers to the percentage of growth assigned to a member. The growth factor is chosen randomly from within a predetermined range (uniform distribution). The member(s) to be grown are either assigned growth factors individually (unshared) or are assigned the same growth factor (shared). Each lettered image

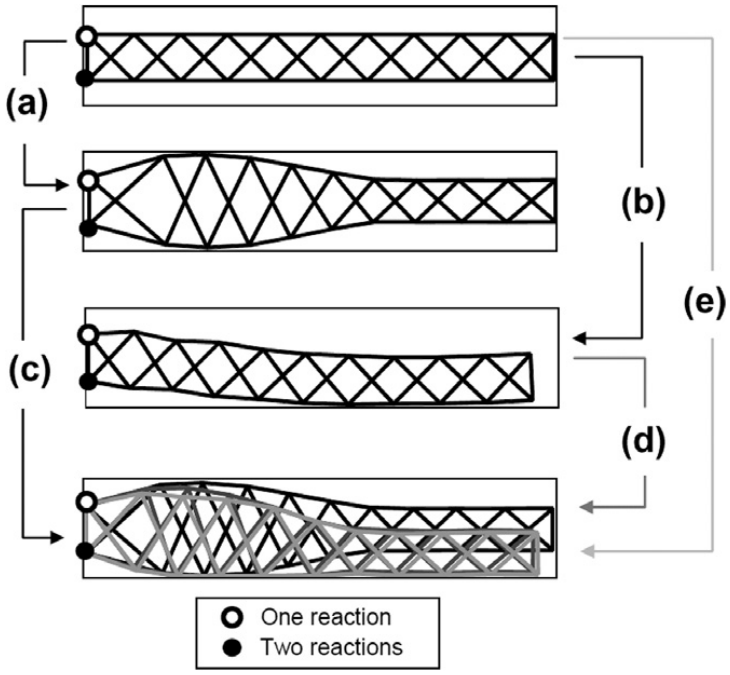


Figure 2.5: Cantilevered truss: (a) joint algorithm, (b) member algorithm, (c) member algorithm starting with a as an initial condition (black), (d) joint algorithm starting with b as an initial condition (dark gray), and (e) 50% mixture of joint and member algorithms (light gray).

consists of five overlaid trusses resulting from separate optimization runs. The overlaying of the runs shows their stability (repeatability). As shown, the member algorithm with the shared growth factor is the least stable of the sets of cases run. Interestingly, the randomization and the incorporation of the joint algorithm stabilize the results. We also see in Fig. 2.6e and f, where the top left joint was allowed to move, that it moved up until it reached the top obstacle. The gravitropic response is illustrated in Fig. 2.7a and b in which the black line represents the ground. Fig. 2.7c is the ground structure and Fig. 2.7d, which is generated using a 50% mixture of the joint and member algorithms, shows that the ground structure turns upward, like the potted plant. This result was obtained by removing the upper constraint and by increasing the growth factor from 0.01 to 0.1 in order to hasten the search of the design space. Notice that truss-like appearance was preserved while it turned upward.

2.2.3 Comparison

The application of meta-heuristic methods to truss problems is commonly considered in the literature, e.g. Perez et al. [41]. Other optimization methods designed specifically for truss optimization can offer improved performance over generic algorithms and a comparison with

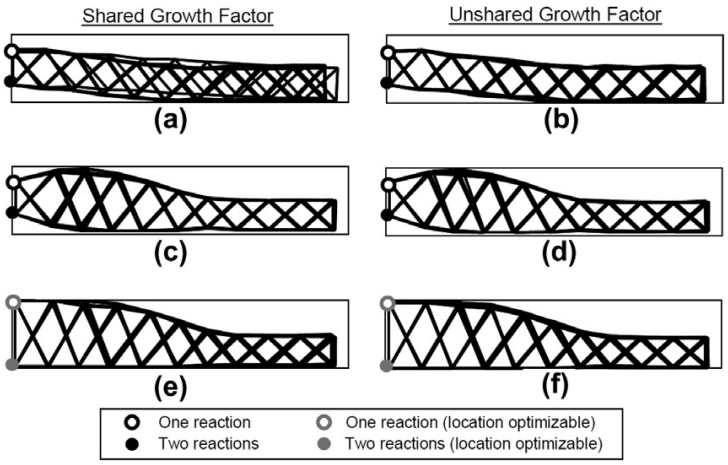


Figure 2.6: Cantilevered trussgrowth factor study: (a) member algorithm, shared, (b) member algorithm, unshared, (c) 50% mixture of joint and member algorithm, shared, (d) 50% mixture of joint and member algorithms, unshared, (e) 50% mixture of joint and member algorithms with location of top right joint optimized, shared, (f) 50% mixture of joint and member algorithm with location of top right joint optimized.

these methods would be interesting. However, unlike the proposed method these schemes usually require gradient information. Since the method we develop is fully heuristic we provide a simple comparison with other algorithms which are commonly applied to this problem in the literature.

In order to facilitate this comparison we consider a 42 bar, pinned-pinned truss investigated above. In particular, one implementation of a GA and two separate implementations of a PSO were evaluated. It was found that both the GA and the PSO converged prematurely to non-optimal solutions and in both cases this problem was overcome by re-starting the optimization routine with the previous solution as an initial condition.

The precise implementation of the various algorithms is described below, in each case the representation used is given in B.3.1. For brevity, n_{dvs} is the number of design variables and the

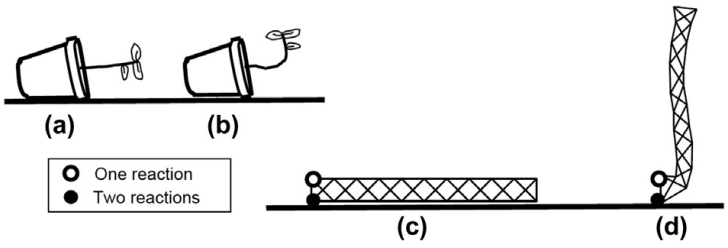


Figure 2.7: Cantilever truss: (a) ground potted plant, (b) gravitropic response of potted plant, (c) ground truss, (d) gravitropic-like response of truss.

Table 2.1: PSO parameter values. (ϵ denotes tolerance value for a termination condition.)

Parameter	C_1	C_2	$\epsilon_{h(X)}$	$\epsilon_{f(X)}$	gen. limit	pop. size
Value	1.5	1.75	1×10^{-6}	1×10^{-9}	1500	n_{dvs}

Table 2.2: GA parameter values. (ϵ denotes tolerance value for a termination condition.)

Parameter	Crossover	Mutation	$\epsilon_{h(X)}$	$\epsilon_{f(X)}$	gen. limit	pop. size
Value	0.4	4%	1×10^{-6}	1×10^{-6}	none	$10n_{dvs}$

truss used as an initial condition will be referred to as X_0 which was the same as used earlier for the CG algorithms. The CG algorithms use the parameters described in section 2.2.1.

1. PSO (variant 1): Version 20100818 of a standard PSO written by S. Chen [11], freely available from the Matlab Central File Exchange under the BSD license. Soft boundary conditions were employed and the initial population range was set to be $[X-0.2, X+0.2]$. Parameters were tuned as suggested by A.E. Perez et al [41] and are provided in Table 2.1, all other parameters were left as default.

2. PSO (variant 2): The same as variant 1 with the modification that the social attraction parameter was set to zero.

3. GA: the Genetic Optimization Toolbox, available with Matlab 2011b. Parameters that were not left as default are provided in Table 2.2. The function "mutationadaptfeasible" was used for mutation, the design was represented with real-valued encoding was and the optimization procedure was started with the unmodified individual.

Now, as can be seen in the Figure 2.8(a), the PSO (variant 1) and those algorithms which incorporate the CG member algorithm converge most quickly and yield the lowest cost solutions.

The GA, for its part, performed well, in so far as it obtained a low-cost solution in one iteration. However, that single iteration took more function evaluations than either the PSO or most variants of the CG algorithms, and the GA did not improve much upon its initial solution. Figure 1(b) compares the performance of the CG joint algorithm and a PSO without the benefit of social attraction. This demonstrates that the local random search is similar to the PSO without global information.

Clearly, the worst performers were PSO variant 2 and the CG joint algorithms. In other words, both the handicapped PSO and the joint algorithm converged the slowest because they both lack the capacity to perform any variety of global search. The fact that both routines yield reasonable final values for the cost seems to indicate that, at least in the region of interest, the design space is convex.

As discussed earlier, the CG algorithms incorporate a number of novel features that allow them to perform well on this particular problem. The fact that the GA and the PSO are seen to not converge as quickly as the algorithms suggested in this chapter is not necessarily surprising since both the GA and PSO are general optimization routines not developed around a specific problem.

In particular, the global and local aspects of the CG algorithms are effectively "split-up," allowing them to search a large portion of the design space while still making local refinements and naturally incorporating the geometric constraints. To wit, the joint algorithm will not generate new shapes that grossly violate geometry constraints and the member algorithm can not readily violate geometry constraints because of the physical constraints imposed by the truss. Standard optimization techniques, designed for general problems, do not naturally incorporate this variety of constraints and must resort to penalty functions or other methods. Although there are certainly many approaches for incorporating constraints (and there is extensive literature on the subject of shape optimization of structures), rejecting infeasible solutions was chosen because it produced reasonable results and kept the focus of the chapter on the development of the CG algorithms, and not on our choice of penalty function. That said, it is clear that the tailored nature of the CG algorithms allowed them to perform well despite not utilizing a more sophisticated method for maintaining the feasibility of solutions. In sum, all algorithms produced reasonable final solutions to the particular problem studied. The PSO and CG algorithms required the least number of cost function evaluations although all perform similarly in order of magnitude.

2.3 Summary and conclusions

This chapter introduced a new heuristic approach to design of truss structures that mimics plant tropisms and incorporates its features of local actuation, global stimulus, constraints, randomness, and triggers. The approach was referred to in this chapter as cellular growth (CG). In particular, a CG joint algorithm, CG member algorithm, and mixing were developed. The joint algorithm was shown to optimize in local regions of the design space and the member algorithm to move the structure out of local regions of the design space. The CG algorithms were studied on a pinned-pinned truss and a cantilevered truss. It was found that the joint algorithm could get caught in a restrictive design space, and that the member algorithm and the mixture

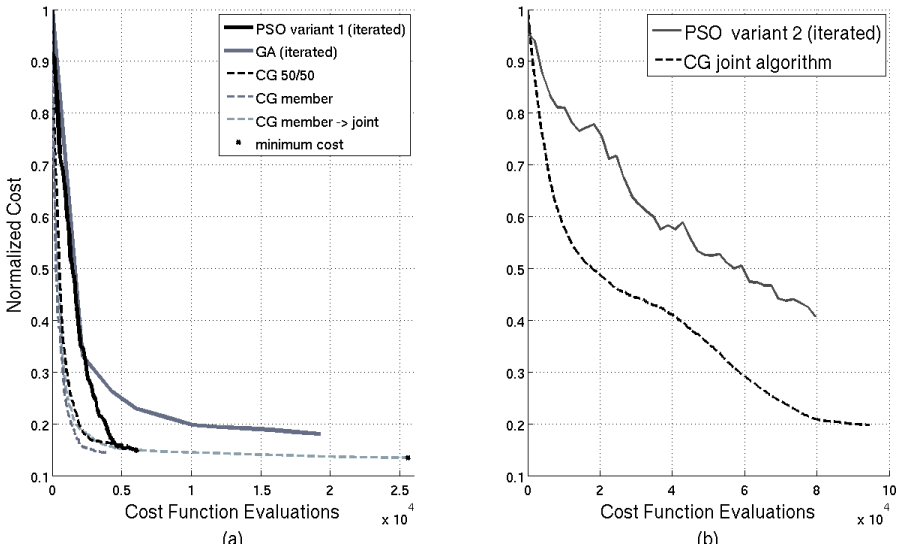


Figure 2.8: (a) CG algorithms are indicated with dashed lines while a PSO and GA are indicated with solid lines. (b) The CG joint algorithm is compared with the same PSO except that the social attraction parameter has been set to zero (PSO variant 2). Note the difference in scale between figures (a) and (b).

of joint and member algorithms could get the solutions out of the restrictive design space. The member algorithm and the mixture of joint and member algorithms were also shown to be more stable when the growth factors are unshared. In other words, randomization increased the repeatability of the optimized results. The joint algorithm in itself can be replaced with any other optimization algorithm that changes a subset of the parameters of a design space and, in this regard, is not unique. The member algorithm and mixing, on the other hand, are novel is the way they tailor global searches of the design space to truss structures.

With respect to behaviors, the optimizations resulted in the arching and double-arching of pinned-pinned trusses and the gravitropic response and bulging of cantilever trusses. The differentiation of the objectives makes the approach well-suited at treating non-well-defined cost functions, in which a designer is allowed to interact with the iterations by triggering specific cost functions at different stages of the optimization and place constraints on the system, as desired, to direct the design.

Future extensions to the work are plentiful. Future studies can optimize material parameters, consider larger-order three-dimensional trusses, new cost measures can be developed, the algorithms could be compared in detail with other heuristic approaches, incorporation of the member algorithm into other hybrid evolutionary algorithms could be studied, particular types of popular trusses could be advanced (like classical tower, bridge and roof designs) and the

approach could be extended to frame structures, and mechanism design.

CHAPTER 3

MOD ALGORITHM

Topological optimization is a difficult problem, due to its discrete, combinatorial nature. Moreover, strict optimization based on minimization of a performance functional is not always desirable because of unquantified aspects of the design. In this chapter we present a method that is designed to overcome this difficulty in the design of truss structures by creating a *prescriptive* heuristic for the generation of planar trusses. By beginning from a ground structure the method incorporates user preference in a feed forward manner with further opportunities for user direction available.

The description of the method is followed by three examples which illustrate the ability of the proposed system to generate common trusses, an optimal truss, and a set of interesting solutions.

3.1 Method

Optimization procedures use an iterative scheme: $D_{i+1} = A_i D_i$ where D_i is a vector of design variables at the i^{th} step and A_i is the operator for the scheme. The operator may be a simple linear update procedure or it could be a more complicated algorithm. As noted earlier, many engineering systems have both topological and non-topological properties. A common divide and conquer simplification separates the properties so that the operator A is essentially divided into an update procedure for the topological properties and an update procedure for the design parameters. Letting the system's topology be expressed by a graph characterized by an adjacency matrix G and the design parameters expressed by the vector X, the update is written as

$$G_{i+1} = PG_i \tag{3.1}$$
$$X_{i+1} = QX_i$$

where P and Q are operators. Equation (3.1) indicates that step i consists of two updates; first of the topology and second of the design parameters. In the MOD algorithm, the operator P performs either a graph addition or a graph subtraction and the operator Q performs either design parameter initialization or design parameter optimization, written

$$P = \{P_A \text{ or } P_S\} \quad Q = \{Q_I \text{ or } Q_O\} \tag{3.2}$$

where P_A is the addition operator, P_S is the subtraction operator, Q_I is the design parameter initialization operator, and Q_O is the design parameter optimization operator. The decision to perform P_A or P_S and the decision to perform Q_I or Q_O are determined by a rule set.

Equations (3.1) and (3.2) reflect behaviors found in multicellular organisms. The multicellular organism is an assembly of cells with a topology (adjacency) and design parameters (edge properties). Graph addition is akin to cellular division, graph subtraction to cellular atrophy, design parameter initialization to initializing (parent) cellular properties, and design parameter optimization is akin to cellular growth. Cellular differentiation provides the rule set wherein cells are divided (or die) until triggered to stop dividing (or die). They grow until triggered to stop growing. The triggers stop any one of these processes when it is no longer advantageous to continue. The MOD algorithm performs addition (or subtraction) until a performance measure ceases to improve. The performance measure is assessed when the design parameters are being updated. Addition switches to subtraction or subtraction switches to addition when a performance measure ceases to improve through design parameter optimization. Before discussing the different operations and the rule set in detail, note that the graph addition operator P_A adds a node to the graph which accompanies the introduction of new design parameters, so it must be followed by the design parameter initialization operator Q_I. Also note that the MOD algorithm is individual and not population based. This reduces computational time so that a designer can readily see a generative process based on a sketch of initial conditions and constraints. The flow chart of the MOD algorithm is given in Fig. 3.1.

3.1.1 Topology and design parameter operators

The multicellular organism is represented by a graph in which each cell is represented by a node and the adjacency of two cells is represented by an edge. The topology of the graph

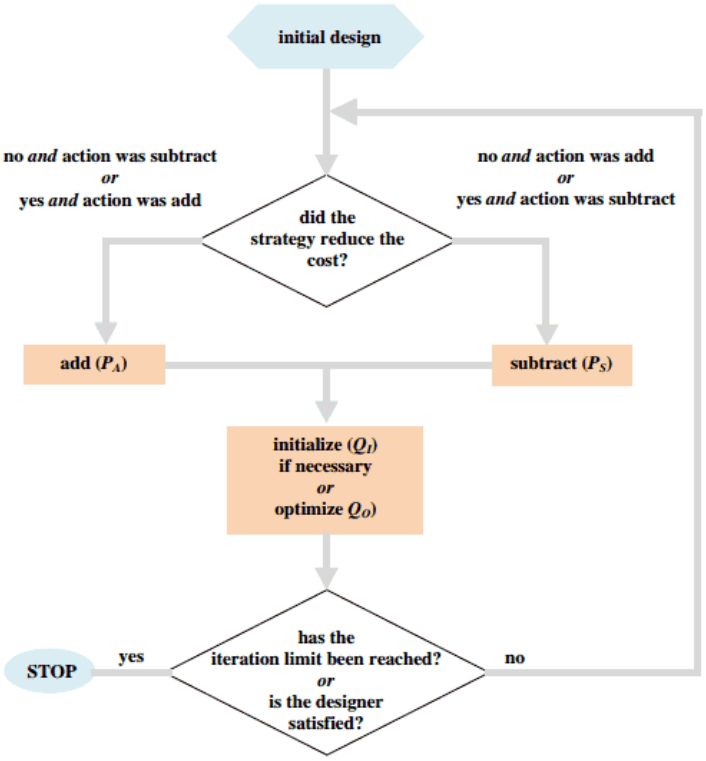


Figure 3.1: MOD algorithm flow chart.

changes when the organism undergoes addition or subtraction. The modification is performed at a cellular level, that is, edge by edge. Each edge has a cost that represents some feature of the interaction between its cells. The selection of an edge depends on its cost. Assuming that there are m edges with costs c_j $(j = 1, \ldots, m)$, the k^{th} edge is selected to undergo addition (or subtraction) in which k is the index of $c_k = max\{c_j\}1^m$ (or $c_k = min\{c_j\}_1^m$). In cases of multiple extrema, each of the edges with the same cost undergoes addition (or subtraction). We now consider the individual operators P_A, P_S, Q_I, and Q_O.

The operator P_A adds a new node to the graph and connects it to appropriate existing nodes. The addition of the new node is determined by the selected edge. The new node is placed between the two nodes on each side of the edge.

The operator P_S subtracts an edge. This is accomplished by either removing an edge from the graph while not removing any nodes, or by contracting the edge, in which case the two nodes on each side of the edge contract to one node. In general, the graph resulting from contraction may not be a viable topology (e.g., produce an unstable truss) so the following rule set is adopted for subtraction:

- Form two graphs, G_R and G_C resulting from the removal and contraction of an edge from

G.

- Evaluate the viability of G_R and G_C.
- If G_R and/or G_C. is viable, select the viable graph that later has the minimum cost.

If neither is viable, skip this step.

Figure 3.1 does not provide the details for the case of multiple edges having the same cost. Multiple edges do not depend on order of removal but do depend on order of contraction. Therefore the different combinations of contraction order are considered and the one that produces the lowest cost is selected. Only a subset of the combinations is considered when the number of edges is prohibitively large.

The operator Q_I initializes the design parameters associated with any new edge. Thus, Q_I is selected immediately after P_A is performed. Otherwise Q_O is performed. The two nodes on the sides of the edge can be regarded as the parent nodes and the new node as the daughter node. The operator Q_I assigns initial design parameters to the new edges that are inherited from the parent edge. The value assigned to the edge is only important in so far as a reasonable value will allow for faster convergence during the subsequent optimization. Throughout this chapter the design parameters of the new edges take on the value of the parent edge.

The operator Q_O performs parameter optimization of a given topology. It is analogous to cellular growth or the interphase which occurs during the cellular division (or atrophy) phases. Parameter optimization can be accomplished with any number of optimization methods such as traditional, gradient based methods or meta-heuristic optimization methods, such as a GA or PSO. Q_O is the most computationally expensive part of the method, like in a real cell where the interphase occupies the bulk of a cell's life.

The rule set governs the sequencing of the operations. In terms of the multicellular analogy, it determines how much division (or atrophy) a cell undergoes before entering interphase. The trigger depends on the global measure of the cost, written

$$C_i = \sum_{j=1}^{m} c_{ij} \tag{3.3}$$

where c_{ij} is the cost of the j^{th} edge at the i^{th} step. The trigger at the i^{th} step is $\Delta C_i = C_i - C_{i-1}$. The parameter optimization continues until the trigger is below a threshold ϵ. The effect of this sequencing is to switch tactics between addition and subtraction when the parameter optimization begins to converge. The sequencing also aims to prevent cancerous growth. Cells that divide without a limit will die necessitating a trigger that terminates cellular growth.

Randomness can be added to this step to generate a wider variety of possibilities, although not performed here.

3.2 Results and discussion

The MOD algorithm is applied to truss design and three examples are explored. The cost function is based on the strain energy in a truss member, written

$$c_j = \frac{A_i E_i}{2L}(\Delta L)^2, \qquad \Delta L = L_i - \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{3.4}$$

where A_i is cross-sectional area, E_i is elastic modulus, L_i is member length, and where a and b designate the two joints at the ends of the member. The members have an elastic modulus of 210,000 GPa, an outer radius of 0.219 m, an inner radius of 0.203 m, and a mass per unit length of 42.40 kg/m. The parameter optimization modifies the lengths of the members by a cellular growth algorithm described in detail in section 2 and in [55]. The truss members are subject to nonlinear inequality constraints placed on member length, joint proximity, and obstacle avoidance. The member length constraints are $\mathbf{h}_1(\mathbf{r}) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} - L_{max} \leq 0$ where r is a $m \times 1$ vector of joint locations, and where a and b designate the joints on each end of a member. The joint proximity constraints are $\mathbf{h}_2(\mathbf{r})(r) = L_{min} - \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \leq 0$ where a and b designate any two distinct joints. The obstacle avoidance constraints allow the user to constrain the truss to specified regions; a detailed description of which is found in [55] and the appendix. Material optimization is not considered here, but will be considered in the following chapter.

In the context of our additive design strategy it is natural to consider the strain minimization problem because the initial structure is (by design) minimal and subject to large deformations, hence strain energy. Additional members stiffen the structure so that the strain energy should decrease as the topology is appropriately modified. Note that if we consider the weight minimization problem we will see the weight/cost increase initially as more members are added, not decrease. In Eq. A.12 we show that the strain minimization problem (Eq. 3.4) is equivalent to minimizing the work done on the truss. Ben-Tal et al. [6] note that this is equivalent to minimizing the compliance, a common objective for structural optimization.

The three examples considered below examine the effectiveness of the topology modification piece of the MOD algorithm. The effectiveness of the design parameter optimization piece is not examined here since this piece is not being advanced. The primary focus of the examination of the topology modification piece is on the ability to produce different trusses.

Figure 3.2 (a) shows an initial truss composed of three pinned members arranged in the

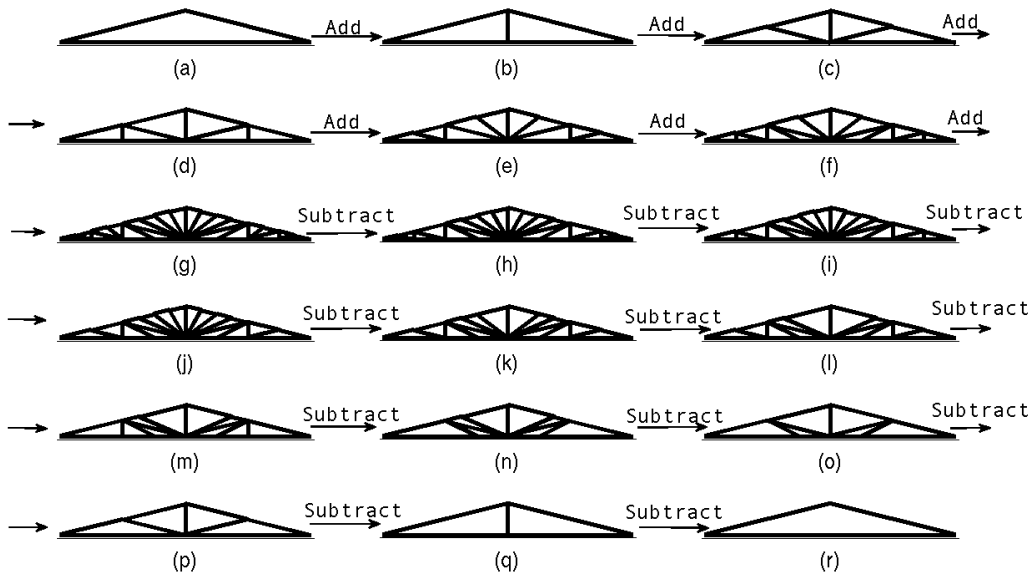
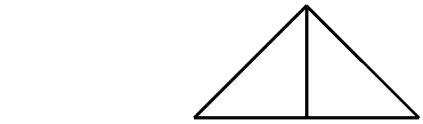


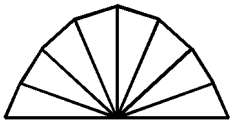
Figure 3.2: Division produce recognizable trusses and atrophy recovers the initial truss.

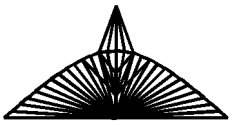
shape of a triangle. The loading is produced by the members weights and a central load (at the top joint) of 10^6N. The bottom member is pinned to the ground. The first question is whether addition by itself is capable of producing different trusses. Toward this end, addition was applied consecutively six times. As shown, addition produced the following trusses: a Kings Post in (b), a Queens Post in (c), a Howe Truss in (d), and then a truss reminiscent of a Double Howe in (e). As the complexity further increased, a fan-like pattern of Michel-like topologies emerged in (f) and (g) on top of the Howe-type truss. Subtraction is first applied to obtain (h). After a few iterations we obtain (k) where we begin to recognize the Double Howe in which the diagonal members were replaced by a structure serving the same purpose. A diagonal structure remains in some form until (o) while we produce trusses similar to a single Howe in (m), and a Queen's Post in (n) and (o). Finally, the last three iterations produce a normal Queen's Post in (p), a King's Post in (q), until the original structure is recovered in (r).

The production of a Michell-like topology beginning from a ground structure is a common problem considered in the literature, for example, Wang et al. [59] offer a solution. We also consider this problem with the proposed method in Figure 3.3 where Figure 3.3 (a) shows an initial truss composed of four pinned members acted on by loading provided by member weights and a central load of 10^3N on the central, bottom joint. This example addresses the question of whether the iterations can lead to optimal features found in trusses. The final truss is shown in Fig. 3.3 (b). It was produced after 5 steps, generating the Michel-like shape and topology. Note



(a)





(b) (c)

(d) (e)

Figure 3.3: (a) The ground structure proposed for the Michell bridge problem and (b) the generated solution has topology reminiscent of the Michell bridge. (c) The same problem with a central load of 10^6N. (d)-(e) Alternative solutions with randomization (the top node has been allowed to move in the y-direction).

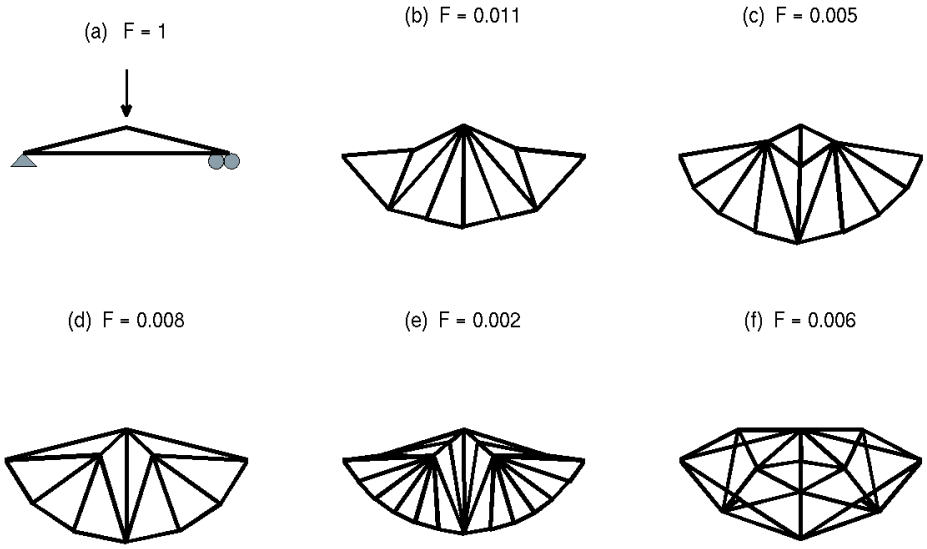


Figure 3.4: MOD algorithm generates different designs. Each design is generated in 10 iterations or less.

that as more members were added (not shown) the cost function would decrease at a slower rate and eventually stop. The limit is associated with an increasing weight that approaches a sizable fraction of the point load. This is demonstrated in Figure 3.3(c) where increasing the load decreased the penalty for additional members. Finally, Figure 3.3(d)-(e) demonstrate alternative solutions produced by adding randomization to the group selection process, and allowing the top node to move.

Figure 3.4 (a) shows an initial triangular truss composed of three pinned members acted on by a point load of 10^6 N at the top. The truss is pinned to the ground on the lower left end and sliding on the right end. This example addresses the question of whether the iterations can generate a rich family of truss types for the early design stage. Each truss (b through f) begins with the initial truss shown in (a) and is generated in 10 iterations or less. The different trusses are produced by adding randomness in the group selection process. Members of the same cost are placed together in a group, creating k groups g_i $(i = 1, 2, \ldots, k)$. Then, the i^{th} group is chosen by

$$i = \begin{cases} max\{g_i\} & \text{if} \tau < \epsilon \\ rand\{1, 2, \ldots, k\}, & \text{otherwise} \end{cases} \tag{3.5}$$

where $max\{g_i\}$ returns the group index of the maximum value of g_i, τ is a uniformly distributed random number between 0 and 1, ϵ is a constant between 0 and 1, and $rand$ returns a random integer. As shown, the different trusses share the catenary-like bottom chord. The topology of the interior is apparently less critical, as indicated by the reasonable performances of the different trusses, particularly the truss shown in (f). The interesting topologies of the resulting designs, the presence of local minima, the reasonable performances, and the speed at which solutions are generated, suggest that the MOD algorithm is a valuable tool for the initial stages of truss design.

3.3 Summary and conclusions

This chapter developed the multicellular organism design (MOD) algorithm for engineering design and optimization. The algorithm is particularly well-suited for the early stages of design where the examination of different topologies plays an important role. The method mimics the sequencing in cellular differentiation of cellular division and atrophy (topology) and growth (optimization). Specific examples illustrated the ability of the division to produce popular trusses, the ability of the algorithm to produce an optimal truss, and the ability of the algorithm to produce a rich family of solutions. The different solutions were found without resorting to a combinatorial search of the design space which is computationally expensive. In fact, the computational time for the topological modifications was small compared to the computational time for the shape optimization. The extension of this method to operate on populations should be straight-forward and would be an interesting subject of future study. The MOD algorithm could also be applied to truss-like problems like frames and simple mechanisms as well as problems from further afield like the disk-packing problem (considered in A.2), which also require a topological and non-topological search.

CHAPTER 4

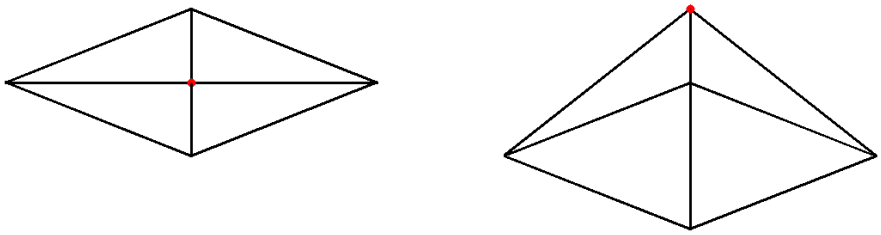
SPACE TRUSS GENERATION

In the previous chapter we introduced the MOD algorithm which we applied to the generation of planar trusses. In this chapter we present an extension for generation of space trusses and we investigate two examples that illustrate how a designer can dynamically interact with the method to direct problem solution.

The extension of the MOD algorithm to space trusses seems straight-forward, unfortunately, the simple division rules that were used in 2D do not necessarily generate rigid trusses in 3D and require more careful attention. To illustrate this point, Figure 4.1 shows a truss which is made rigid by moving a single node where the topology of the structure did not change. In fact, developing methods to determine the rigidity of structures in 3 dimensions is the subject of current mathematical research. For example, as recently as 2009 Connely et al. [13] notes that "In 3D, there is no known counting characterization of generically isostatic frameworks." Because the methods used in Chapter 3 were "counting methods" (they did not incorporate geometric information) they cannot guarantee the stability of a new joint.

In the context of this research, the effect of this constraint is that a simple topological condition is not sufficient to generate stable trusses. When a new node is added to the structure a successful edge addition rule must incorporate both topological and geometric information in order to produce a stable configuration. A rule that guarantees the rigidity of the structure is presented, and related rules are discussed; iterative application of these rules to an example structure then demonstrate topology generation and optimization is carried out for both the shape of the structure (as in chapter 3) and the sizing of its members.

Finally the proposed heuristics are used within the context of the MOD algorithm to generate a tower and a bridge. A final example considers the ability of the method (in its current form) to explore competing objectives.



(a) (b)

Figure 4.1: (a) An unstable truss and, (b) the same truss which is made stable by moving a node.

4.1 Introduction

As noted in 1.1.5, there is a relative paucity of literature on the *design* of truss structures. This is evident in 2D and it is all the more apparent when considering 3D. In any event, the state of the art of truss design is typified by the work of Nguyen et al [37] who use SIMP methods to consider the generations of truss-like structures in 3D.

Methods that use ground structures are commonly criticized because the search space is limited to designs which are obtainable based on modifications of the ground structure. An example of a ground structure truss design method is provided by Shea et al who apply shape grammars to planar trusses [48]. Although it is true that this constraint is limiting in the event that the cost is well-defined, in the context of design we do not believe that this is a limitation but a strength. To wit, when designing a bridge the engineer may already have in mind what general "look" he wants but desires to examine related options. In this light, the ground structure embodies aspects of the desired solution and the power of the method is only limited by its ability to search the design space and generate interesting solutions.

4.2 Methods

The general framework of the MOD algorithm is described in 3.1, the following sections aim to describe those aspects of the method which are specific to 3D.

As in the previous chapter, we alternate between topology generation and shape optimization. In addition, we will also optimize member size. As discussed in 1.1 the order of these operations should be: topology generation, shape optimization, then size optimization. As such,

we consider the sizing optimization step following shape optimization.

4.2.1 Topology generation

A review of topology generation is provided in 1.1.3 and the basic concept behind our approach is discussed in 3.1. It is the goal of this section to revisit the heuristics developed for node addition in the plane and consider their relevance in 3D. Specifically, we need to address: (1) Where the node should be added and (2) How the node can be made stable.

Where should the node be added?

The heuristic which we describe in Chapter 3 works as follows: Beginning with a structure with a reasonable layout (topology, shape, and sizing) we determine the stress of each member. If a member has high stress relative to other members in the truss we divide it, allowing the truss to redistribute forces. This simple idea will not be modified.

How should we connect the node?

The MOD algorithm dictates that we modifies a structure's topology by 'dividing a member,' effectively adding a joint and several members. In order for that joint to be stable in three dimensions, that joint must be stable in *two* planes. Moreover, just adding members that stabilize the joint might not result in preservation of symmetries in the structure. To this end, the new joints are added by a simple counting rule that ensures the new joint(s) will be stable. The outcome of this procedure yields "good" structures, although it may add more than the minimum number of edges required to preserve symmetry and ensure stability. These edges may be removed later if they are seen to perform poorly (as defined in Chapter 3), or using methods we discuss. We may formulate a simple rule that only adds local connections as follows:

Consider a node n_i and let the k joints that are adjacent to n_i be denoted by the set $N_i = \{n_i^1, \ldots, n_i^k\}$ where $k \neq i$. The set of unique vectors which describe the edges are $\mathbf{e}_i^k = \mathbf{n}_i - \mathbf{n}_i^k$ (boldface indicates vectors). Then,

Definition 1. *Node n_i is said to be stable IFF if there exists $\{n_i^1, n_i^2, n_i^3\} \subset N_i$ such that $\{\mathbf{e}_i^1, \mathbf{e}_i^2, \mathbf{e}_i^3\}$ is a spanning set of $\mathbb{R}^3$.*

We may now define a simple manner to add nodes:

Theorem 1. *Let two stable nodes n_1 and n_2 be initially connected by a member e_1^2. The sets of nodes to which n_1 and n_2 are adjacent are $N_1 = \{n_1^1, n_1^2, \ldots, n_1^j\}$ and $N_2 = \{n_2^1, n_2^2, \ldots, n_2^k\}$, respectively. Now, let e_1^2 be bisected by a new node, $\mathbf{n}_i = t\mathbf{e}_1^2 + \mathbf{n}_1$ ($t = 0.5$) to form two new members e_i^1 and e_i^2. If the new node is connected to either set N_1 or N_2 then the new node will also be stable.*

Proof. Let's consider the edges formed by joining n_i with those joints in N_1 to form the new members

$$\begin{aligned}\mathbf{E_i} &= \{\mathbf{e}_i^1, \ldots, \mathbf{e}_i^j\} \\ &= \{\mathbf{n}_i - \mathbf{n}_1, \ldots, \mathbf{n}_i - \mathbf{n}_1^j\} \\ &= \mathbf{n}_i - \{\mathbf{e}_1^1, \ldots, \mathbf{e}_1^j\}.\end{aligned}$$

Now, since n_1 is assumed to be stable we know that its edge set $\mathbf{E}_1 = \{\mathbf{e}_1^1, \ldots, \mathbf{e}_1^j\}$ spans $\mathbb{R}^3$. Since $\mathbf{E_i}$ contains $\mathbf{E}_1$ it must also span $\mathbb{R}^3$ unless $\mathbf{n}_i = \mathbf{e}_1^j$ and $\mathbf{e}_1^j$ is a member of the spanning set of $\mathbf{E}_1$. Since $\mathbf{E}_1$ contains only unique vectors (pairs of nodes are not connected to each other multiple times) then n_i is stable according to Def. 1.

□

Thus, if we add a joint as described by Theorem 1 we are guaranteed to maintain the stability of the truss. However, because the choice to connect n_i with N_1 was arbitrary we might just as well have chosen to connect n_i with N_2 and the same argument holds. If either choice produces a stable truss then which set should we choose? Without additional information it would be arbitrary to choose one over the other. Moreover, as we saw above it is only necessary to connect to a subset of N_1 that consists of three nodes and contains the basis for $\mathbb{R}^3$. Operating under the assumption that the joint is only minimally stable (it *just* satisfies Def. 1) we will connect the new joint to $N_1 \cup N_2$, possibly adding a few additional members but guaranteeing that symmetries are preserved.

Three different rules have been tried and are summarized below:

1. Set union (as described above) nodes are added such that $\mathbf{E}_i = \mathbf{n}_i - \{\mathbf{E}_1 \cup \mathbf{E}_2\}$
    - Guarantees stability.
2. Set difference, nodes are added such that $\mathbf{E}_i = \mathbf{n}_i - \{\mathbf{E}_1 \cap \mathbf{E}_2\}$
    - Does not guarantee stability, although it often produces stable structure and results in fewer extraneous members.
3. All paths of length k. This method works as follows: starting with node n_1 find all paths of length two that reach n_2, connect n_i to the nodes in along these paths. If the truss is still unstable, find all paths of length three and connect n_i to the nodes along these paths.
    - Guarantees stability, while being more conservative than "set union" and more aggressive than "set difference." More computationally intensive.

Examples of these heuristics applied to a simple tetrahedral tower are provided in B.2.

4.2.2 Shape Optimization

This section details the shape optimization procedure used to generate the remaining examples; a review of shape optimization as it applies to truss structures is given in 1.1.1. For convenience, the general optimization problem is stated below:

$$\begin{aligned} min \;\; & f(X) \\ s.t. \;\; & h_i(X) \leq 0 \end{aligned}$$

where the vector X is the vector of design variables, $h_i(X)$ is the i^{th} nonlinear inequality constraint, and $i = 1, \ldots, k$ for k nonlinear inequality constraints. The cost is taken to be the total strain energy

$$f(X) = \frac{1}{2} \sum_{i}^{m} \frac{E_i A_i}{L_{i,0}} (L_{i,0} - L_i)^2 \tag{4.1}$$

where the sum is taken over the members $i = 1, \ldots, m$, E_i is the modulus, A_i is the cross sectional area, $L_{i,0}$ is the unloaded length, and L_i is the loaded length. The inequality constraints are maximum permissible stress on each member

$$h_i(X) = |\sigma| - \sigma_{max} \tag{4.2}$$

where $\sigma = F_i/A_i$, F_i is the member force, A_i is the member area, $\sigma_{max} = 200$MPa, and $i = 1, \ldots, m$ for m members. Other constraints could be included, such as limiting maximum nodal displacement, or Euler buckling, but were omitted in this study.

To find the minimum of this problem we tried a number of constrained non-linear optimization techniques: derivative approaches with a simple mapping (described in B.3.1), a 3D implementation of the CG algorithms (described in Chapter 2), as well as GA and PSO (with simple mapping). Of these attempts, the best results were initially obtained with a two part approach: use a PSO with large bounds to get "in the ballpark" and then refine the design using a traditional gradient-based algorithm with tight bounds on the output of the PSO. For convenience, we will refer to this as the hybrid approach. However, computational time remained large and the solutions were found to reliably preserve structural symmetries only for small numbers of design variables. As a result, an approach is adopted that parameterizes

Table 4.1: PSO parameter values. (ϵ denotes tolerance value for a termination condition.)

Parameter	C_1	C_2	$\epsilon_{h(X)}$	$\epsilon_{f(X)}$	gen. limit	pop. size
Value	0.5	1.6	1×10^{-3}	1×10^{-6}	2500	$3n_{dvs}$

the truss in such a way that symmetry is guaranteed, and the number of design variables and computational time decrease.

Representation

The vector X is the vector of node locations, however, the node locations are initially a $n \times 3$ matrix, V, so that a mapping is required to generate X. A simple representation can be constructed by mapping the x, y, and z values of each free node into a vector of length $n_{dvs} = 3n - n_r$, where n_r is the number of reactions (3 if the joint is pinned, and 1 or 2 if the joint may roll in one or two directions). It was found that the ability to iteratively generate "clean" topologies relied significantly on the ability to accurately reproduce the symmetries present in a particular topology. For this reason more complex representations were developed and are discussed in detail in Appendix B.3. One of the examples uses the "orbital" mapping described in B.3.3, while the other uses the simple representation.

PSO

Fourie et al. [18] has shown that a PSO can be as efficient as gradient based methods (RQP) when applied to shape and sizing optimization of truss structures. Our implementation uses what Chen [11] refers to as "soft" boundary conditions, a discussion of which is given by Xu et al. [61] where he refers to the same boundary conditions as "damping." Note that the use of boundary conditions precludes the necessity of a penalty function. Finally, the parameter values for the optimization steps are given in Table 4.1, where the values for the *social* parameter, C_1, and the *cognitive* parameter, C_2, are those suggested by Fourie [18]. The remainder of the values were left as default.

Derivative-based search

For the derivative-based search we use Sequential Quadratic Programming (SQP) implemented as the *active-set* algorithm in the 2011b version of Matlab's constrained nonlinear optimization routine *fmincon*. An excellent discussion of SQP is provided by Vanderplaats [56] and the non-default parameter values used are given in Table 4.2.

Table 4.2: SQP parameter values. (ϵ denotes tolerance value for a termination condition.)

Parameter	$\epsilon_{h(X)}$	$\epsilon_{f(X)}$
Value	1×10^{-3}	1×10^{-3}

4.2.3 Material optimization

Material optimization may be included in truss optimization in a variety of ways, a brief summary of which is available in 1.1.2. In this study, they are circular in cross-section with radius allowed to vary so that $0.005\text{m} \leq r \leq 0.1\text{m}$. The modulus of elasticity is $E = 200\text{GPa}$ and the density is $\rho = 7830\text{kg/m}^3$.

The representation used for the design variables is given in B.3.2.

4.3 Results and discussion

For the remainder of this section we will consider structures that have "generated" topologies and have been subject to sizing and geometry optimization.

4.3.1 Bridge

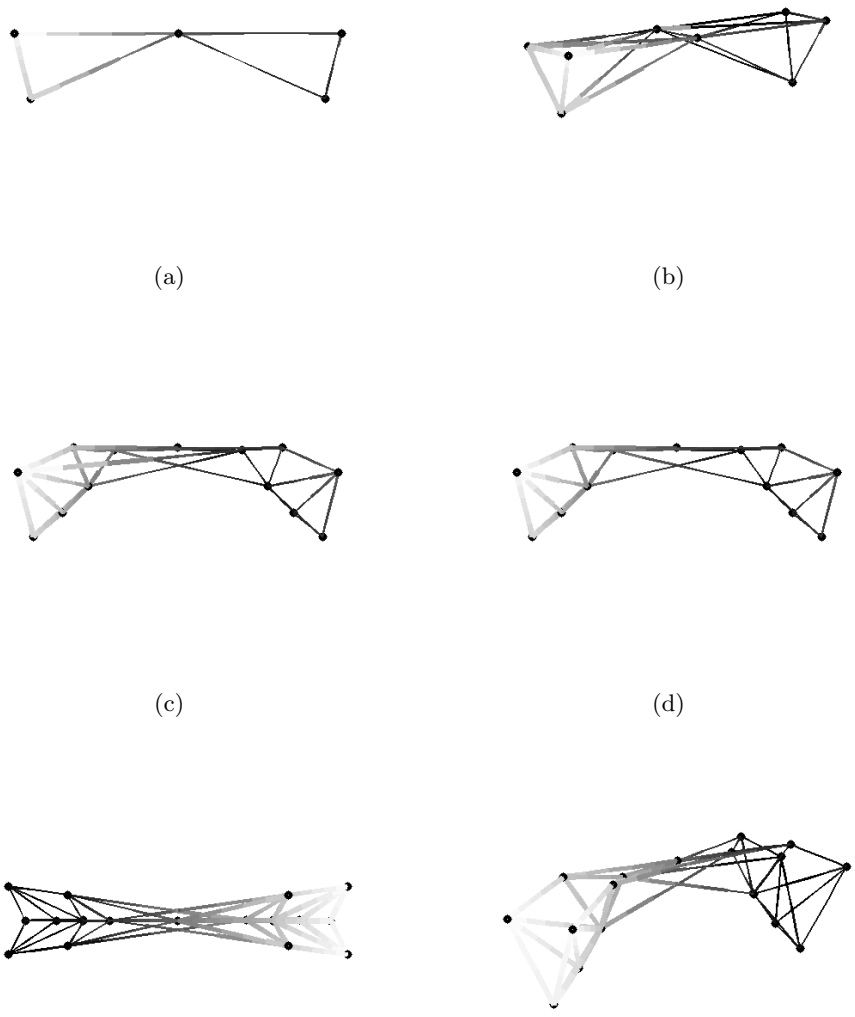
In this example we consider a bridge with a 100 meter span. Initially, it is 10 meters wide has pin supports located 20 meters below the level of its "deck" with roller supports (in the z-direction) located at the top (see Figure 4.2). It is further subject to a maximum height constraint of ($z \leq 30\text{m}$) and two central loads of $10\times^6\text{N}$. In this example, the following additional heuristics were adopted:

If the shape optimization procedure produces a structure with converging nodes those nodes are merged (removing a single member from the structure if they are connected). After the nodes are merged the structure is evaluated: if it is no longer rigid the change is rejected.

If the material optimization procedure produces members whose radius is at the lower bound of the permitted range they are removed from the structure. After the members are removed the structure is evaluated: if it is no longer rigid the change is rejected.

4.3.2 Multiple objectives

Although MOD is not formulated for multi-objective optimization its generative scheme does produce designs that may then be compared against multiple objectives. For example, consider total weight and total strain energy. On the one hand, increasing the number of elements in



(a) (b)

(c) (d)

(e) (f)

Figure 4.2: (a) A profile view of the initial structure for the long bridge problem. The joints in the lower left and lower rights corners are pinned, while the joints at the top left and top right corners are rollers providing support in the vertical direction. (b) The same structure shown from off axis. (c) A solution to the bridge problem posed in section 4.3.1. MOD failed to remove a single member which is shown removed manually in (d). (e)-(f) Alternative view of the same truss.

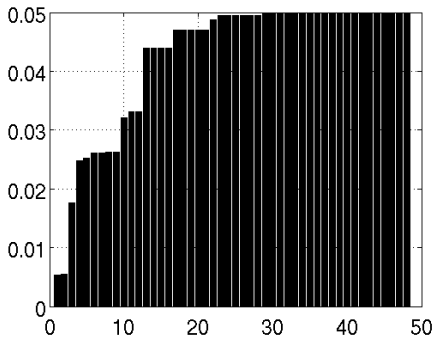


Figure 4.3: The optimized member radii for Figure 4.2(c).

the structure will tend to increase its rigidity, decreasing the total strain. On the other hand, increasing the number of elements will tend to increase the total weight. In this way, weight minimization and strain minimization are competing objectives and decreasing one will likely increase the other.

A comparison of the generated design for a 16 iteration run of MOD is given in Figure 4.4. The truss is subject to its own weight, two large central loads totalling 2×10^6N, stress constraints, and a maximum height constraint ($z \leq 30$m). Feasible designs are shown with red circles, infeasible designs are shown with green diamonds, and those designs (regardless of feasibility) that are non-dominated are overlaid with black x's. The initial design (the least weight design) is non-dominated but infeasible. Feasible, non-dominated designs are produced at iterations 4, 6, 8, 12, 16. Note that the initial structure was infeasible because stresses exceeded permissible values. The addition of members stiffen the structure and yield a feasible design by iteration 5, which was subsequently optimized and produced the first feasible, non-dominated solution at iteration 6.

4.3.3 Tower

In this section we consider the generation of towers. Interestingly, generating towers was relatively difficult and required the adoption of additional rules for division. In the examples presented, the generative steps were user directed and the shape and material optimization used the simple mapping described in section B.3.1. The division routine used was 'set difference,' as described in section 4.2.1 and two additional heuristics were adopted: (1) members that have more than two crossings are removed from the structure and (2) when new nodes are added they are connected to each other. Both of these steps were required to generate the repeated elements seen in Figure 4.5a. (See Appendix B.1 for examples of towers produced without the aid of these additional heuristics). Note that the first of these two additional heuristics (remov-

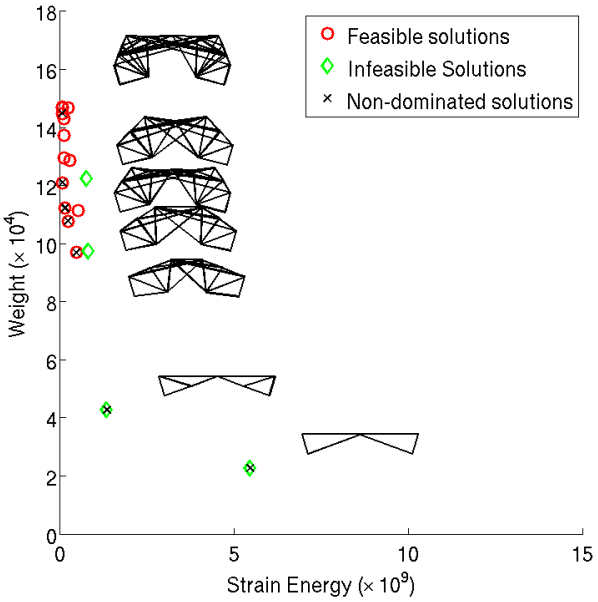


Figure 4.4: The performance of bridges produced by 16 iterations of MOD are plotted for total strain energy and weight. Dominant solutions are shown with a black 'x' and the associated truss is plotted immediately to the right.

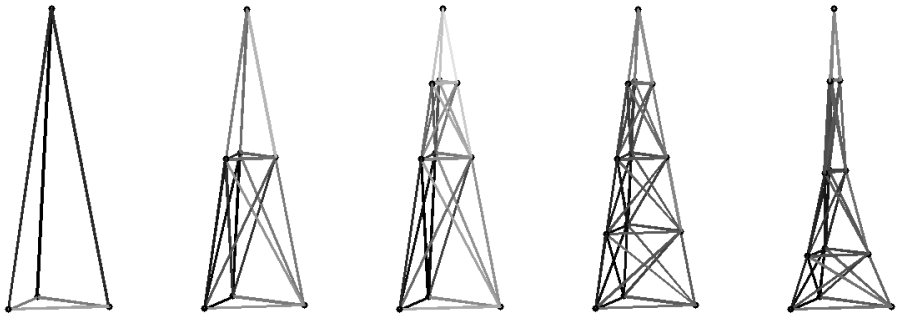
ing members with more than two crossings) is effectively a "cleaning" step performed after the division adds members to the truss. The necessity of this is *visually* evident to the designer and suggests that the designer may be able to play important role in the design process which we will investigate in the next section.

4.3.4 User Direction

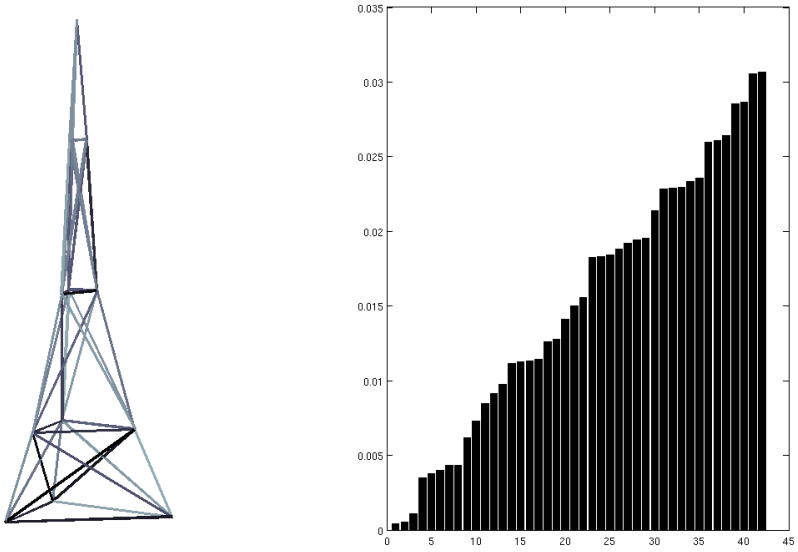
A goal of this dissertation is to develop a truss design method which allows a user to interact with a design and optimization process, taking inspiration from free-hand sketching and incorporating elements of optimization. To accomplish this, the designer must obviously interact with the proposed algorithm and, to this end, a GUI is presented in Figure 4.6. The designer is free to modify a space truss, confident that the generated design is rigid and may quickly execute an optimization process so that the early design can benefit from their own intuition as well as analytical aids, hopefully encouraging a "reflexive conversation."

In the following figures we demonstrate two user-directed designs. In the first example a truss is generated over the course of eight steps (Figures 4.7-4.8) and the final truss shown in Figure 4.8(f) has a cost of 0.0076.

In the second example, a more lengthy process is demonstrated in Figures 4.9-4.11. The



(a)



(b)

Figure 4.5: (a) A tower is generated with three division steps followed by optimization. (b) The final tower from (a) is shown with a (sorted) bar graph of optimized cross-sectional area.

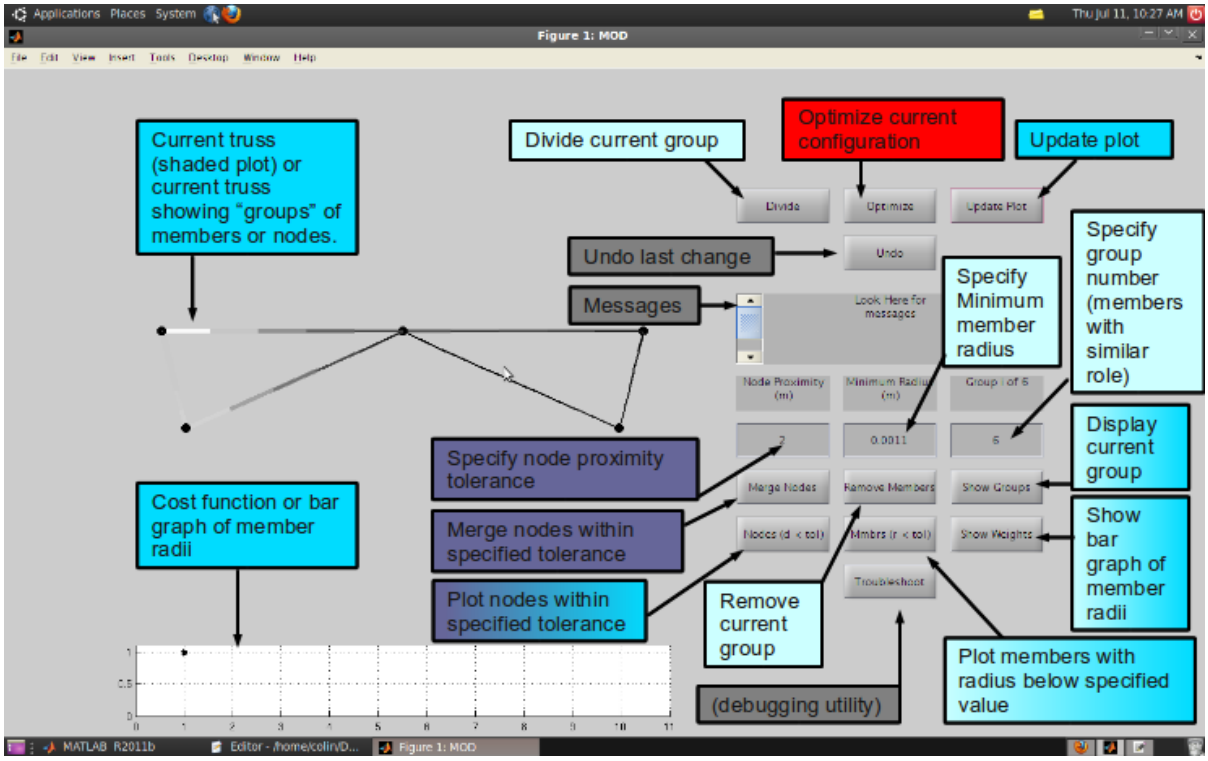
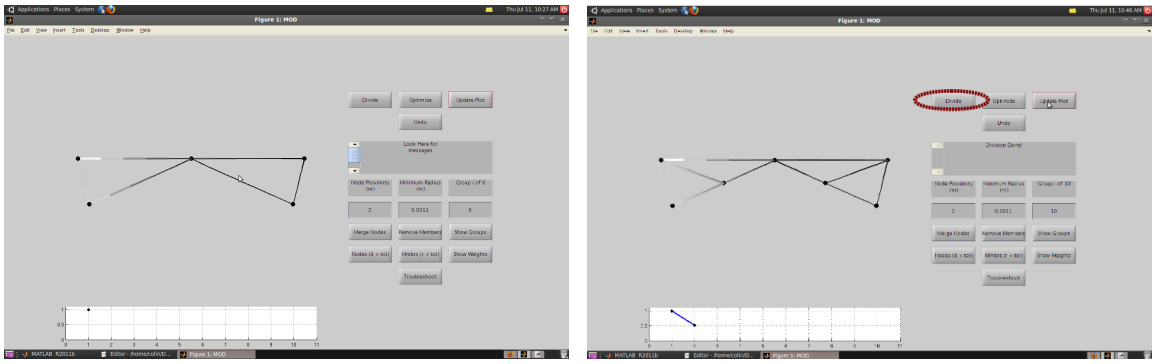


Figure 4.6: A screenshot of the GUI used in section 4.3.4. The buttons of the GUI are labeled with call-out boxes and their color indicates the category of their function: Light green boxes execute member-related commands, purple boxes execute node related commands, turquoise boxes execute plotting functions, the red box executes shape and sizing optimization, and gray boxes have miscellaneous functions. Boxes with a two-color gradient have functions that span two categories.

process is terminated after twenty steps with a cost of 0.0076. Although an intermediate truss obtains a lower cost (0.0041) this truss lacks the fully developed arch shape of the final example, which may represent a more desirable design for unquantified reasons.

4.4 Summary and conclusions

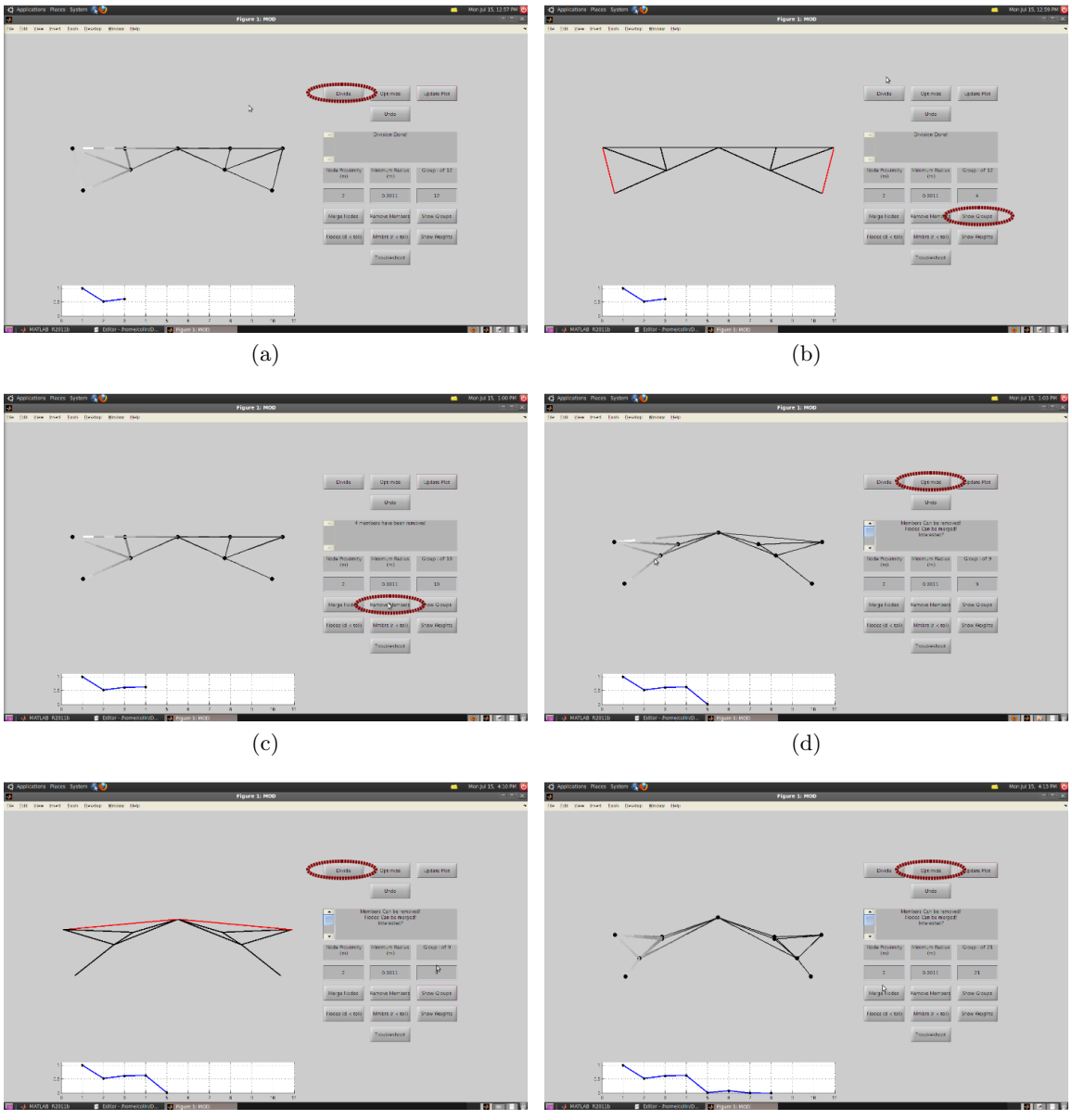
In this chapter we extended the MOD algorithm to generate space trusses. This was facilitated by the development of new heuristics for division, heuristics for the removal of members and nodes, and the development of symmetric representations that allow for fast optimization. Examples demonstrate the ability of the proposed method to generate feasible designs for a tower, and a 100 meter bridge. The bridge example was shown to produce a variety of solutions which were evaluated against two objectives, providing the user with more insight into the



(a) (b)

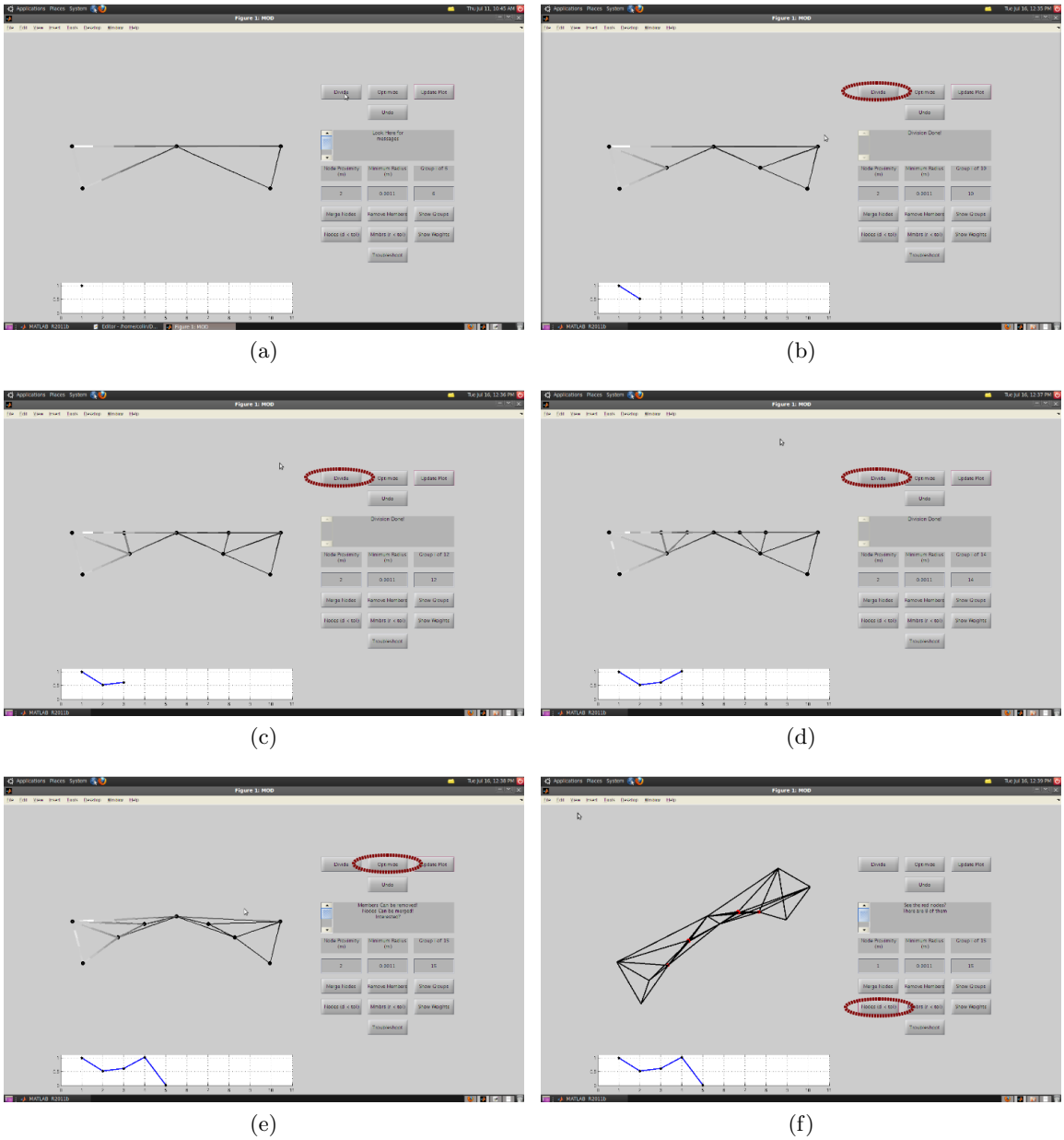
Figure 4.7: An example of a user directed design process, red-dashed circles indicate the action that updated the design from the previous iteration. This example is continued in Figure 4.8.

generated designs. Finally, a GUI that provides a means for convenient user interaction was described. Using this GUI two examples illustrate a directed design process, generating two different solutions to the initial problem. Because of the necessity of maintaining rigidity, space trusses can be complicated structures to easily explore without appropriate aids. These final two examples demonstrate the ability of the method to divorce the designer from this problem and show that a directed exploration process is a promising approach for space truss generation. Further development is needed to assess the impact of such a tool on design exploration.



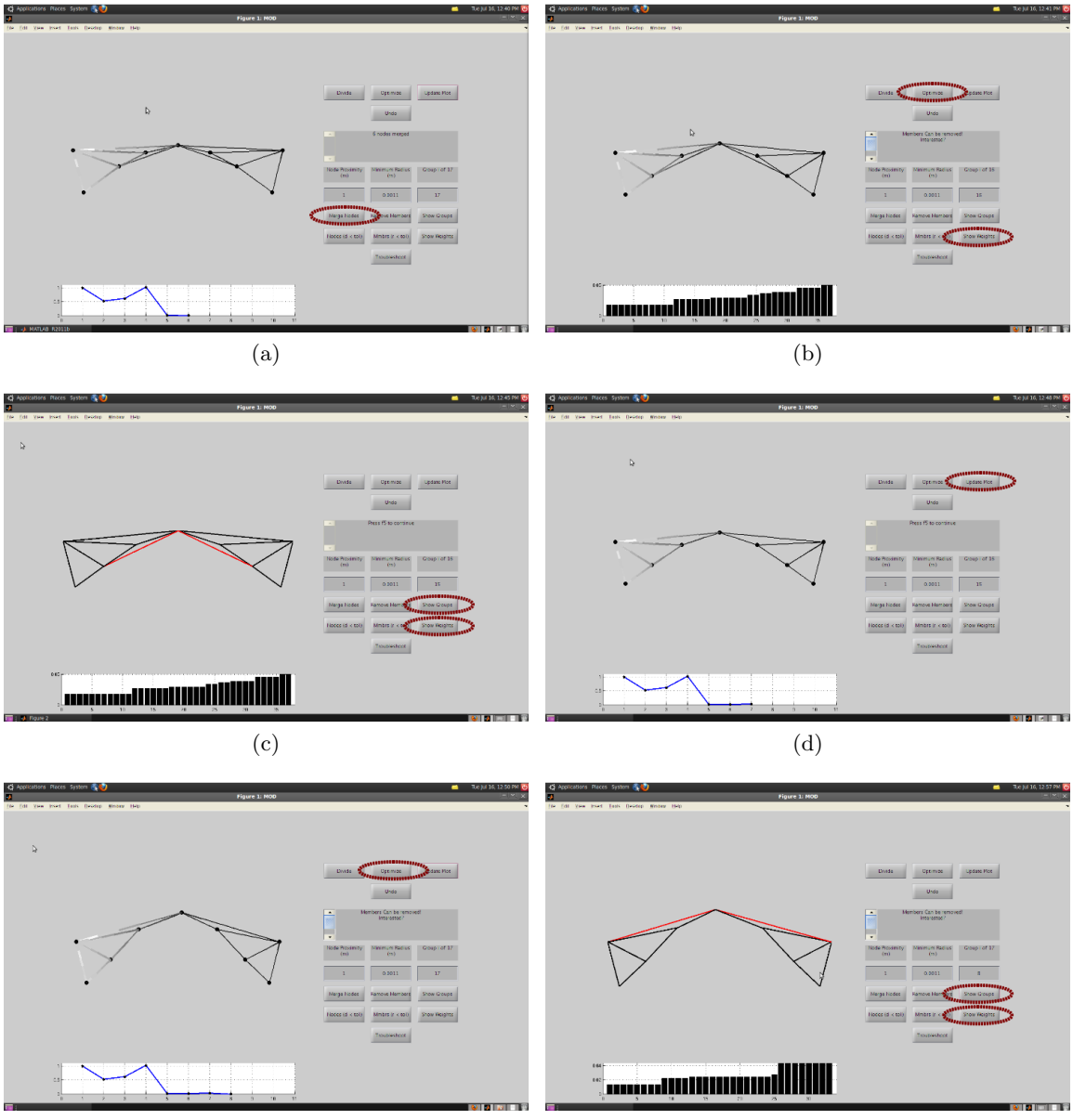
(a) (b) (c) (d) (e) (f)

Figure 4.8: This figure is a continuation of the design process begun in Figure 4.7 The final design has a normalized cost of 0.0076.



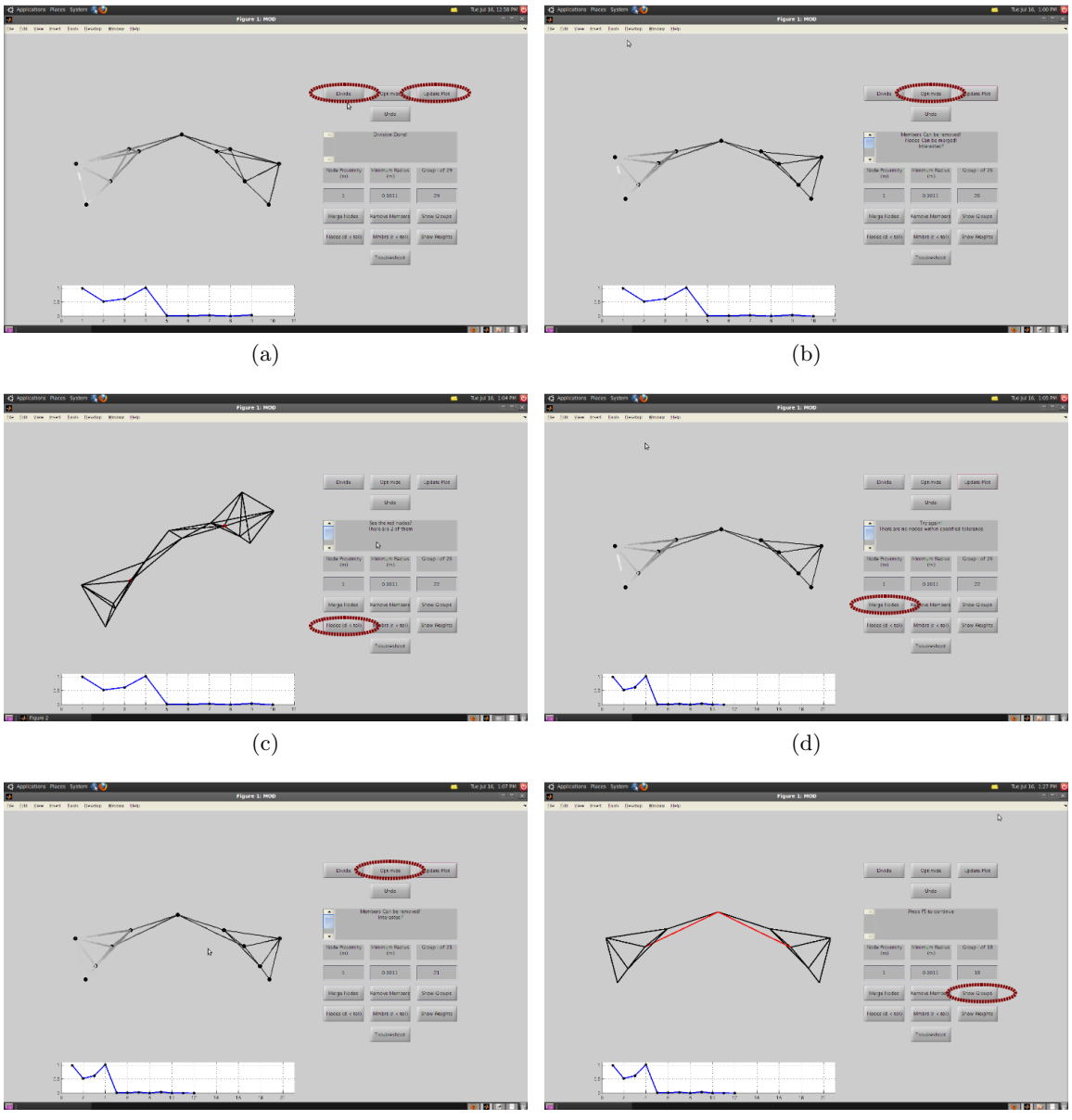
(a) (b) (c) (d) (e) (f)

Figure 4.9: Another example of a user directed design process, red-dashed circles indicate the action that updated the design from the previous iteration. This figure is continued in Figure 4.10.



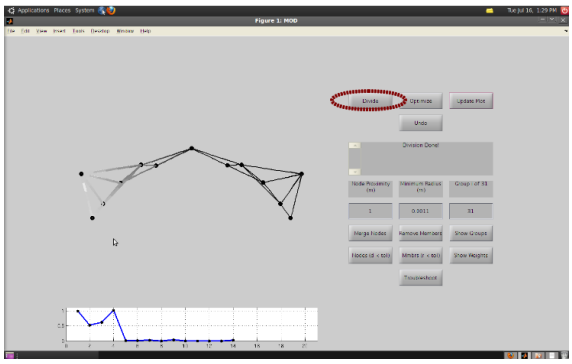
(a) (b) (c) (d) (e) (f)

Figure 4.10: This figure is a continuation of the design process begun in Figure 4.9

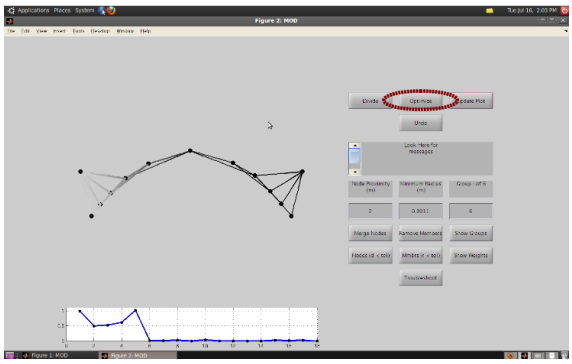


(a) (b) (c) (d) (e) (f)

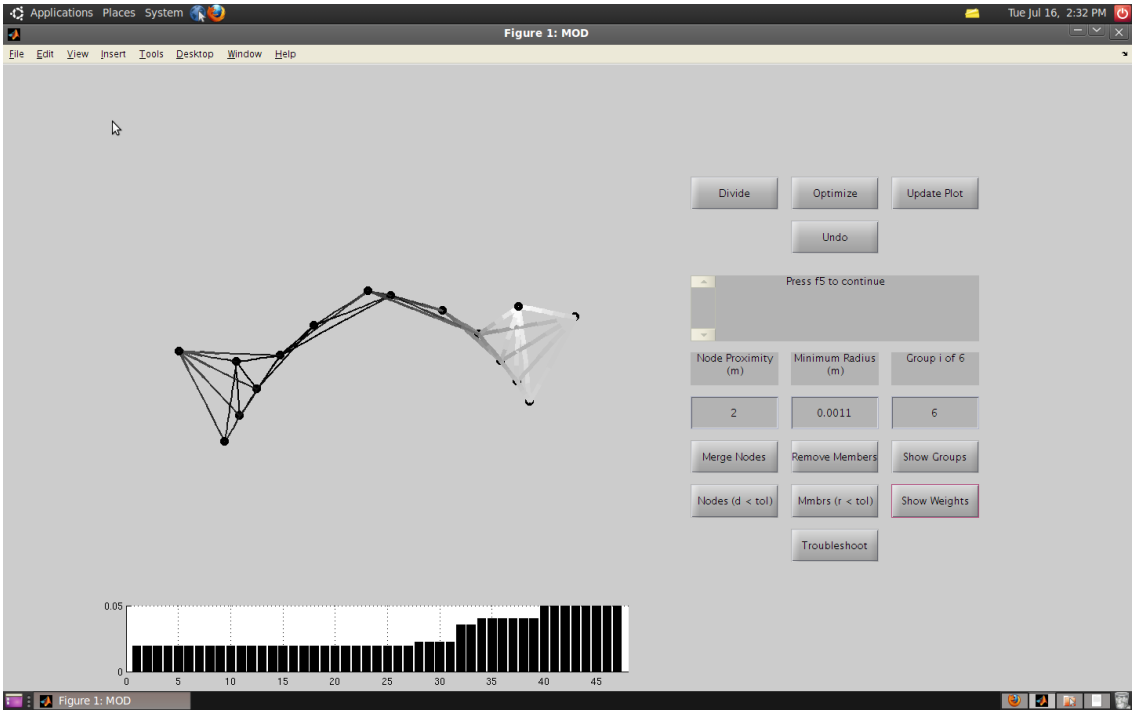
Figure 4.11: The continuation of the design process shown from Figure 4.10. The minimum value for this process occurs in (e) with F = 0.0041.



(a)



(b)



(c)

Figure 4.12: The continuation of the design process shown from Figure 4.11. The final truss has a cost of 0.0076 and is shown with a profile view (b) and a skew view (c).

CHAPTER 5

CONCLUSION

As noted in section 1.1.5, there has been a limited amount of work dedicated to the *design* of trusses; most effort has been focused on developing optimization methodologies. Although optimization is a pillar of structural design the community has often ignored that many problems require more than rote optimization to produce "optimal" results. Since some aspects of design are intangible, methods that address this would be beneficial. We consider this problem and present examples of user-informed, generated designs of planar and space trusses.

That being said, there are certainly additional features that could add utility to the method and would be interesting to incorporate. For example, improved rules that provide robust truss generation in 3D are necessary. The methods that are developed provide a useful starting point, however, there is room for improvement. In addition to refining the operation of the existing algorithm, two other possibilities are briefly discussed in the following section.

5.1 Future directions

5.1.1 User feedback

Currently, the amount of user involvement is limited to the specification of a ground structure, deciding sequencing (e.g. 4.3.3), and specification of obstacle constraints (with CG algorithms). Fortunately, the nature of the method means there are opportunities for more interaction.

Design methods that incorporate user feedback currently exist and were briefly discussed in 1.1.5. One of the popular methods places the user in the loop, commonly with a GA, making the user the selection operator. In the context of MOD, a simple possibility would be to give the user more control over the various subroutines, for example allowing the user to select the member(s) to be divided, to specify areas of the structure to undergo shape optimization, or

to actively add obstacle constraints to guide the optimization process (see section 2.1.3 for a description of these constraints). An example of the possibilities that may be generated by deviating from the rote selection process are given in 3.4. Allowing the user to interact with the division or "death" process would perhaps allow for creativity, encouraging the "reflexive" conversation that Schon insists is the hallmark of an engaging design process, currently obtained with pen and paper yet elusive with computational aids.

Accomplishing this would require development of a reasonable method to allow the designer to explore and engage with the design; a GUI might help to facilitate this. Such a tool is demonstrated in section 4.3.4 where two examples illustrate the generation of different designs using MOD. However, further work needs to be carried out to determine the efficacy of the methods and interface. The development of a study to assess a user's ability to generate designs would prove useful and this would necessitate the development of a metric to judge the efficacy of the tool in aiding design exploration. In the introduction, we asked if it was possible to develop an appropriate computational aid that allows a designer to direct and dynamically explore engineering systems. We believe that, for the special case of truss design, we have shown that the answer to this question is *yes*.

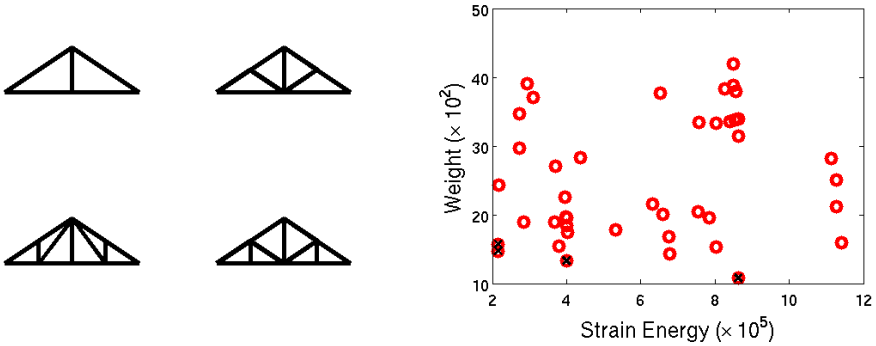
5.1.2 Multiple objectives

We may extend the MOD algorithm to handle multiple objectives in several ways, the simplest of which would be to define an aggregate objective function. This involves no modification of the algorithm but also does not allow for convenient exploration of the Pareto frontier. Most algorithms that develop the Pareto frontier are population based, which is not the case with MOD. Thus, it seems desirable to consider ways in which the algorithm might be modified to incorporate multiple individuals.

First, recall that the algorithm described in Chapter 3 begins each iteration with a generative process (which involves the addition of a node or nodes to the graph) followed by an optimization process. The new node is placed along an existing edge and its location is determined via a simple heuristic. The generative process is responsible for updating the topology according to a heuristic and the optimization process refines the current design.

During the generation step, if more than one edge has the same (or very similar) weight, then the same division process will be performed on each edge of equal weight. A simple possibility is the following: instead of selecting *one* set of edges for the division process it could be applied to all sets. If there are m_i sets at iteration i, then there will be m_i designs generated. This has the benefit of eliminating the seemingly arbitrary choice of heuristic for set selection but has other drawbacks. Previously, we required that each generative step be followed by shape optimization before being evaluated. If we keep this convention then this approach results in

ballooning computational effort. Furthermore, since the generative step usually adds complexity the branches located farther from the root require more computational time to optimize. A demonstration of the results produced by following this branching procedure (without optimization) are shown in Figure 5.1 where the designs are shown evaluated for weight and strain energy. Interestingly, three of the four non-dominated designs are common styles which and are reproduced by following the ordinary selection procedure as shown in Figure 3.2.



(a) (b)

Figure 5.1: (a) Non-dominated trusses in the population produced from four successive iterations of division (on all groups) and, (b) dominated designs are indicated with a red circle, non-dominated solutions are marked with a circumscribed black X.

To make this approach tractable, it is likely necessary to minimize computational effort by reducing the number of designs at each step. This could be done by applying the generative step to the non-dominated solutions (and setting aside these solutions in an "elite" pool). Initially this is not possible, since there are only a few solutions but this could be dealt with by waiting to discard solutions until a minimum number have been generated.

Experience from the traditional multi-objective optimization community indicates that maintenance and generation of population diversity is a difficult and critical aspect of defining a successful multi-objective algorithm. Both PSO and GA do this by defining a distance metric used in the selection process. This is used to drive a preferential selection process, steering designs away from crowded areas of the design space. Solving this problem in the context of MOD would be key to creating a reasonable generative, multiobjective algorithm using MOD.

In other words, generating a population of individuals does not mean that they will be dispersed along the Pareto frontier. Successful multiobjective algorithms must also ensure that they do not generate solutions in "niches." Algorithms such as NSGA-II (a popular MOGA)

use a distance metric to measure the proximity of solutions to each other, and use this measure to limit the number of solutions within a defined area. It would be simple to adapt one of these methods to prevent niching, however, just because two trusses have similar performance does not mean that one would not be preferred over the other for unquantified reasons. It would seem that elimination of solutions needs to be performed by the designer, or an algorithmic proxy.

An interesting possibility that might help maintain the visual diversity of the solutions would be to employ a measure of topological similarity which could be used as one of the objectives or as a metric to evaluate niching. There are many possible measures, one such measure is called "edit distance" which essentially quantifies the number of changes required to produce one graph from another.

As a concluding remark, we might note that the complexities inherent in creating a multi-objective generative system are apparent. An obvious question is, "why bother?" Hornby [27] provides an interesting response: "One of the applications of evolutionary algorithms is the automatic creation of designs. For evolutionary techniques to scale to the complexities necessary for actual engineering problems, it has been argued that generative systems, where the genotype is an algorithm for constructing the final design, should be used as the encoding."

In other words, when you start from scratch the number of possible designs is hopelessly large. Asking an EA to design a building with only knowledge of material properties and constraints is akin to asking it to design the building *and* the construction techniques required to build it. We see this when we look at designs produced by the SIMP methods, e.g. Figure 1.1. Developing a constructible design from the examples Nguyen provides is not a trivial problem and would require significant interpolation. On the other hand, constructing a design produced by MOD would be comparatively trivial since the methods incorporates a crude knowledge of construction practices, reducing both the computational overhead to produce the design as well as reducing the difficulty in translating the design into constructible trusses.

5.1.3 Summary of suggested directions

The general theme of the previous two sections might be summarized as, "How do I give the designer tools to explore the design space?" We suggested two methods: a static method and a dynamic method. Developing MOD to handle multiple objectives would give the designer a *static* sample solutions along the Pareto frontier from which to choose, and there are many example of authors who have pursued this method. Alternatively, providing the user with a way to interact with and modify the design during the generation process would provide a *dynamic* experience, similar to the work by Aage et al [1]. It seems likely that any truly "engaging" method should somehow incorporate both: allowing users to interact with the generation process while also

still providing information about the trade-offs of the different designs.

5.2 Conclusion

An admirable goal of design or optimization methods is generality, yet throughout this dissertation we have focused on trusses. Further development may allow the proposed method to be applied to other problems, a taste of which we provide in A.2 where we consider the disk-packing problem.

In this dissertation we developed a biologically motivated, heuristic optimization technique for shape optimization of planar trusses. Following this, we presented a design algorithm which generates trusses starting from a ground structure, with the preceding technique being incorporated into the process. We then extended the method to generate space trusses.

The shape optimization method of chapter 2 was seen to preserve aspects of user preference (truss shape) while exploring distant regions of the design space; it was also demonstrated to perform comparably to other heuristic techniques on an example problem. The truss generation method, MOD, was developed and shown to reproduce an near-optimal topology for a planar truss, and novel trusses for common loading scenarios. The same method was shown to produce interesting structures in three dimensions where it generated feasible solutions to tower and bridge problems. Finally, an example of a graphical interface used these tools to allow a designer (the author) to direct the generation two different structures. To our knowledge, there is not yet another truss *design* method that provides a unified approach to truss design which incorporates user preferences.

Computational analysis techniques have reached a level of maturity that they are no longer a limiting factor in the design process: the lack of computational design tools is. To be useful these tools must advance beyond repetitive analysis and incorporate the user in an exploration process, augmenting their instincts and preferences, helping them to discover design ideas that they themselves would not have discovered. It is our belief that practical computational design aids will require user interaction and by necessity address multiple objectives. Generative systems are well suited to producing the rich variety required to explore a design space while still incorporating user interaction. The methods presented here incorporate all of these features, resulting in a novel design methodology and a stride towards accessible computational design.

REFERENCES

[1] Niels Aage, Morten Nobel-Jørgensen, Casper S Andreasen, and Ole Sigmund. Interactive topology optimization. In *6th European Congress on Computational Methods in Applied Sciences and Engineering*.

[2] P Agarwal and AM Raich. Optimal design of bridge and roof truss systems using multi-objective genetic algorithms. In *Computing in Civil Engineering (2005)*, pages 1–12. ASCE, 2005.

[3] Erik K. Antonsson and Jonathan Cagan. Engineering shape grammars. *Formal engineering design synthesis*, page 65, 2001.

[4] Nayar Cuitláhuac Gutiérrez Astudillo, Rebeca del Rocío Peniche Vera, Gilberto Herrera Ruiz, Roberto Alvarado Cardenas, and Francisco J Carrión Viramontes. A long span bridge and a greenhouse roof truss structure optimized by means of a consistent genetic algorithm with a natural crossover. *Engineering Computations*, 30(1):49–73, 2013.

[5] Breanna Bailey and Anne Raich. Capturing user aesthetic design preferences during multi-objective roof truss optimization. In *Joint int. conference on computing and decision making in civil and building engineering (ICCCBE), Montreal*, pages 3374–3384, 2006.

[6] Aharon Ben-Tal and Martin P Bendsøe. A new method for optimal truss topology design. *SIAM Journal on Optimization*, 3(2):322–358, 1993.

[7] Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering*, 71(2):197–224, 1988.

[8] Martin Philip Bendsøe and Ole Sigmund. *Topology optimization: theory, methods and applications*. Springer Verlag, 2003.

[9] MP Bendsøe and RB Haber. The michell layout problem as a low volume fraction limit of the perforated plate topology optimization problem: an asymptotic study. *Structural optimization*, 6(4):263–267, 1993.

[10] L. Caldas. Generation of energy-efficient architecture solutions applying gene_arch: An evolution-based generative design system. *Advanced Engineering Informatics*, 22(1):59–70, 2008.

[11] S. Chen. Another particle swarm toolbox (version 20100818). Matlab Central File Exchange. `http://www.mathworks.com/matlabcentral/fileexchange/25986-another-particle-swarm-toolbox`.

[12] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002.

[13] Robert Connelly, PW Fowler, SD Guest, Bernd Schulze, and WJ Whiteley. When is a symmetric pin-jointed framework isostatic? *International Journal of Solids and Structures*, 46(3):762–773, 2009.

[14] Kalyanmoy Deb and Tushar Goel. A hybrid multi-objective evolutionary approach to engineering shape design. *Lecture notes in computer science*, 1993:385–399, 2001.

[15] T. Dorta. Design flow and ideation. *International Journal of Architectural Computing*, 6(3):299–316, 2008.

[16] T. Dorta, E. Perez, and A. Lesage. The ideation gap:: hybrid tools, design flow and practice. *Design Studies*, 29(2):121–141, 2008.

[17] Z Fawaz, YG Xu, and K Behdinan. Hybrid evolutionary algorithm and application to structural optimization. *Structural and Multidisciplinary Optimization*, 30(3):219–226, 2005.

[18] PC Fourie and AA Groenwold. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, 23(4):259–267, 2002.

[19] Fernando Fraternali, Andrea Marino, Tamer El Sayed, and Antonio Della Cioppa. On the structural shape optimization through variational methods and evolutionary algorithms. *Mechanics of Advanced Materials and Structures*, 18(4):225–243, 2011.

[20] J.C.M. FRS. L. on the calculation of the equilibrium and stiffness of frames. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(182):294–299, 1864.

[21] Hitoshi Furuta, Kenji Maeda, and Eiichi Watanabe. Application of genetic algorithm to aesthetic design of bridge structures. *Computer-Aided Civil and Infrastructure Engineering*, 10(6):415–421, 1995.

[22] Galileo Galilei. Discorsi e dimostrazioni matematiche intorno a due nuove scienze (leiden, 1638), 98.

[23] R.L. Graham. Sets of points with given minimum separation (solution to problem el921). *American Mathematical Monthly*, 75:192–193, 1968.

[24] R.L. Graham, BD Lubachevsky, K.J. Nurmela, and PRJ Östergård. Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181(1):139–154, 1998.

[25] Victor A Greulach. *Plant function and structure*. Macmillan Co.: New York, 1973.

[26] Warren Hare, Julie Nutini, and Solomon Tesfamariam. A survey of non-gradient optimization methods in structural engineering. *Advances in Engineering Software*, 59:19–28, 2013.

[27] G.S. Hornby and J.B. Pollack. The advantages of generative grammatical encodings for physical design. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 600–607. IEEE, 2001.

[28] W. Huang and T. Ye. Global optimization method for finding dense packings of equal circles in a circle. *European Journal of Operational Research*, 210(3):474–481, 2011.

[29] PW Jansen and RE Perez. Constrained structural design optimization via a parallel augmented lagrangian particle swarm optimization approach. *Computers & Structures*, 89(13):1352–1366, 2011.

[30] Ali Kaveh and Siamak Talatahari. Optimal design of skeletal structures via the charged system search algorithm. *Structural and Multidisciplinary Optimization*, 41(6):893–911, 2010.

[31] Rafal Kicinger, Tomasz Arciszewski, and Kenneth De Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23):1943–1978, 2005.

[32] S. Krish. A practical generative design method. *Computer-Aided Design*, 43(1):88–100, 2011.

[33] L Lamberti. An efficient simulated annealing algorithm for design optimization of truss structures. *Computers & Structures*, 86(19):1936–1953, 2008.

[34] J. McCormack, A. Dorin, and T. Innocent. Generative design: a paradigm for design research. *Proceedings of Future ground, Design Research Society, Melbourne*, 2004.

[35] RE Melchers. On extending the range of michell-like optimal topology structures. *Structural and Multidisciplinary Optimization*, 29(2):85–92, 2005.

[36] A.G.M. Michell. The limits of economy of material in frame structures. *Phil. Mag*, 8(47):589–597, 1904.

[37] Tam H Nguyen, Glaucio H Paulino, Junho Song, and Chau H Le. A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, 41(4):525–539, 2010.

[38] M Ohsaki, Tsuneyoshi Nakamura, and Y Isshiki. Shape-size optimization of plane trusses with designer's preference. *Journal of structural engineering*, 124(11):1323–1330, 1998.

[39] Makoto Ohsaki. *Optimization of finite dimensional structures*. CRC Press, 2010.

[40] Panos Y. Papalambros. The optimization paradigm in engineering design:promises and challenges. *Computer-Aided Design*, 2002.

[41] RE Perez and K Behdinan. Particle swarm approach for structural design optimization. *Computers & Structures*, 85(19):1579–1588, 2007.

[42] Vagelis Plevris and Manolis Papadrakakis. A hybrid particle swarmgradient algorithm for global structural optimization. *Computer-Aided Civil and Infrastructure Engineering*, 26(1):48–68, 2011.

[43] George IN Rozvany. A critical review of established methods of structural topology optimization. *Structural and Multidisciplinary Optimization*, 37(3):217–237, 2009.

[44] GIN Rozvany and M Zhou. Layout and generalized shape optimization by iterative coc methods. *NATO ASI SERIES E APPLIED SCIENCES*, 231:103–103, 1993.

[45] MP Saka. Optimum design of steel frames using stochastic search techniques based on natural phenomena: a review. *Civil engineering computations: tools and techniques*, pages 105–147, 2007.

[46] D.A. Schön. Designing as reflective conversation with the materials of a design situation. *Knowledge-Based Systems*, 5(1):3–14, 1992.

[47] Jamie J Shah and Noe Vargas-Hernandex. Metrics for measuring ideation effectivness. *Design Studies*, 24:111–134, 2003.

[48] Kristina Shea and Jonathan Cagan. Generating structural essays from languages of discrete structures. In *Artificial Intelligence in Design98*, pages 365–384. Springer, 1998.

[49] Kristina Shea and Jonathan Cagan. The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent. *Design Studies*, 20(1):3–23, 1999.

[50] Patrick Y Shim and Souran Manoochehri. Generating optimal configurations in structural design using simulated annealing. *International journal for numerical methods in engineering*, 40(6):1053–1069, 1997.

[51] Larry Silverberg and John Richard Gardiner. Simulation of the mechanics of water-skier motion. *Mathematics and computers in simulation*, 34(2):163–181, 1992.

[52] V. Singh and N. Gu. Towards an integrated generative design framework. *Design Studies*, 2011.

[53] Krishnan Suresh. Efficient generation of large-scale pareto-optimal topologies. *Structural and Multidisciplinary Optimization*, 47(1):49–61, 2013.

[54] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. Wiley, 2009.

[55] C.E. Tschida and L.M. Silverberg. Cellular growth algorithms for shape design of truss structures. *Computers & Structures*, 116:1–6, 2013.

[56] G.N. Vanderplaats. *Numerical optimization techniques for engineering design: with applications*. McGraw-Hill New York, 1984.

[57] GN Vanderplaats. Structural design optimization status and direction. *Journal of Aircraft*, 36(1):11–20, 1999.

[58] W. Visser. Designing as construction of representations: A dynamic viewpoint in cognitive design research. *Human-Computer Interaction*, 21(1):103–152, 2006.

[59] D Wang, WH Zhang, and JS Jiang. Combined shape and sizing optimization of truss structures. *Computational mechanics*, 29(4-5):307–312, 2002.

[60] D Wang, WH Zhang, and JS Jiang. Truss optimization on shape and sizing with frequency constraints. *AIAA journal*, 42(3):622–630, 2004.

[61] Shenheng Xu and Yahya Rahmat-Samii. Boundary conditions in particle swarm optimization revisited. *Antennas and Propagation, IEEE Transactions on*, 55(3):760–765, 2007.

[62] M Zhou and GIN Rozvany. The coc algorithm, part ii: topological, geometrical and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1):309–336, 1991.

APPENDICES

APPENDIX A

TRUSS ANALYSIS

A.1 Nonlinear truss analysis

A truss is composed of members connected via pinned joints. Let the list of joint locations be contained in an nxD matrix called J where D is the dimension of the truss ($D = 2$ for planar trusses $D = 3$ for space trusses). At static equilibrium the forces at each joint must sum to zero such that

$$\begin{aligned} \vec{F}_i &= 0 \\ \vec{P}_i + \sum_{j=1}^{m} \eta_{i,j} \vec{T}_j + \vec{F}_i^{ext} &= 0 \end{aligned} \tag{A.1}$$

where $\vec{P}_i$ are the reactions, $\vec{T}_j$ are the forces on the joint due to the jth member, $\vec{F}_i^{ext}$ are the loads on the ith joint, and

$$\eta_{i,j} = \begin{cases} 1 \text{ if member j is incident on joint i,} \\ \\ 0 \text{ otherwise.} \end{cases}$$

The member forces are given by

$$\begin{aligned}\vec{T}_j &= k_j \left(|\vec{r}_j| - g_j L_j\right) \frac{\vec{r}_j}{|\vec{r}_j|} \\ &= k_j \left(1 - g_j \frac{L_j}{|\vec{r}_j|}\right) \vec{r}_j \end{aligned} \tag{A.2}$$

where the vector $\vec{r}_j$ is the member vector, L_j is the unloaded length of the member, and g_j is a growth factor (due to thermal loads) such that $g_j \neq 1$ causes the member to apply a "growth" force on the structure. Finally, the reactions are given by

$$\vec{P}_i = -\left(\sum_{j=1}^{m} \eta_{i,j} \vec{T}_j + \vec{F}_i^{ext}\right) \cdot \hat{p}_i \tag{A.3}$$

where $\hat{p}_i$ is the unit vector indicating the direction of the reaction such that

$$\hat{p}_i = \begin{cases} \hat{p}_i \cdot \hat{p}_i = 1 \text{ if the i}^{th} \text{ joint is acted on by a reaction} \\ \hat{p}_i \cdot \hat{p}_i = 0 \text{ otherwise.} \end{cases}$$

Equation (A.1) represents nD equations with $nD + p$ unknowns where there are n joints, and there are p reactions. Upon substitution of equation (A.2) and equation (A.3) into equation (A.1) we have nD equations with nD unknowns so that

$$\vec{F}_i = \sum_{j=1}^{m} \left[\eta_{i,j} k_j \left(1 - g_j \frac{L_j}{|\vec{r}_j|}\right) (\vec{r}_j - \vec{r}_j \cdot \hat{p}_i)\right] + \left(\vec{F}_i^{ext} - \vec{F}_i^{ext} \cdot \hat{p}_i\right) = 0 \tag{A.4}$$

for $i = 1, \ldots, n$. Notice that the resulting system is nonlinear due to $|\vec{r}_j|$ and must be solved via a nonlinear optimization technique. It is also possible to develop linearized model although the response under large deformations (such as large growth factors) may be inaccurate.

Finally, consider the following case: when a reaction is directed along the x axis of the ith joint the method prescribed above results in the equation $0 = 0$ for the x component of the ith joint. This equation is always satisfied and may be excluded when formulating the solution routine.

A.1.1 Energy Analysis

Imagine a system boundary that encapsulates a truss. This system is closed but may have heat or work interactions as follows: When a load is applied to the truss, work is done on the truss. If a member "expands" (or grows) then we may think of this as heat being transferred to the truss. Now, the truss does not do work on its surroundings and if the members are all simple elastic bars (or ties) than no heat is generated by the truss. Thus, we may summarize the first law for the truss as follows:

$$E_{in} - E_{out} = E_{final} - E_{initial}$$
$$W_{in} + Q_{in} = U_{final} - U_{initial}$$

where it remains to determine W_{in}, Q_{in}, U_{final}, and $U_{initial}$.

Now, the initial state of the truss may be taken as the unloaded state. This implies that none of the members are compressed and $U_{initial} = 0$. U_{final} may be found by integration of equation (A.2) which yields the stored energy, $U_j = \frac{1}{2}k\left(|\vec{r}_j| - g_j L_j\right)^2$. And the total stored energy in the truss is

$$U_{final} = \frac{1}{2}\sum_{j=1}^{m} k_j \left(|\vec{r}_j| - g_j L_j\right)^2. \tag{A.5}$$

To understand this expression in the context of the thermodynamics we must draw a comparison between the growth factor g_j and thermodynamic quantities. Consider the thermal expansion coefficient

$$\alpha = \frac{1}{L}\frac{dL}{dT}$$

which is approximately

$$\alpha = \frac{1}{L}\frac{\Delta L}{\Delta T}.$$

Now, the earlier g notation is formulated in terms of percentage growth. We may also write

this as

$$\begin{aligned} g_j L_j &= L_j + \Delta L_j \\ &= L_j + L_j \alpha_j \Delta T_j \\ &= L_j \left(1 + \alpha_j \Delta T_j\right) \end{aligned}$$

so that

$$g_j = 1 + \alpha_j \Delta T_j. \tag{A.6}$$

where we essentially may choose ΔT_j to generate growth in a member. Thus we may write equation (A.5) as

$$U_{final} = \frac{1}{2} \sum_{j=1}^{m} k_j \left(|\vec{r}_j| - \left(1 + \alpha_j \Delta T_j\right) L_j\right)^2. \tag{A.7}$$

Turning our attention to the left hand side of the energy balance, it remains to determine W_{in} and Q_{in}.

The work that is done on the truss is simple mechanical work so that the work done on a single joint is

$$\begin{aligned} W_i &= Fd \\ &= F_i^{ext} \sqrt{(x_i - x_i^0)^2 + (y_i - y_i^0)^2} \end{aligned}$$

where F_i^{ext} is the load on the ith joint, $[x_i, y_i]$ is the displaced joint location, and $[x_i^0, y_i^0]$ is the unloaded joint location. The total work done on the truss is

$$W_{in} = \sum_{i=1}^{n} F_i^{ext} \sqrt{(x_i - x_i^0)^2 + (y_i - y_i^0)^2}. \tag{A.8}$$

Heat transfer to a single member is $Q_j = m_j c_j \Delta T_j$ so that the total heat transferred to the truss is

$$Q_{in} = \sum_{j=1}^{m} m_j c_j \Delta T_j. \tag{A.9}$$

Finally, we may use equations (A.7), (A.8), and (A.9) to express the energy balance as

$$\sum_{i=1}^{n} F_i^{ext} \sqrt{(x_i - x_i^0)^2 + (y_i - y_i^0)^2} + \sum_{j=1}^{m} m_j c_j \Delta T_j = \frac{1}{2} \sum_{j=1}^{m} k_j \left(|\vec{r}_j| - (1 + \alpha_j \Delta T_j) L_j \right)^2.$$

Grouping like terms we may separate joint and member sums to yield

$$\sum_{i=1}^{n} F_i^{ext} \sqrt{(x_i - x_i^0)^2 + (y_i - y_i^0)^2} = \sum_{j=1}^{m} \left(\frac{1}{2} k_j \left(|\vec{r}_j| - (1 + \alpha_j \Delta T_j) L_j \right)^2 - m_j c_j \Delta T_j \right). \tag{A.10}$$

To the extent that our models of thermal expansion and elastic deformation are correct it is important to note that equation (A.10) is an identity since it follows directly from energy conservation. Thus, this equation may be used to verify the validity of a solution since any solution must satisfy the above equation. Figure (A.1) shows that equation (A.10) is satisfied by a linear and nonlinear solver.

For reference, let's write equation (A.10) in terms of the g notation. From equation (A.6) we have that

$$\Delta T_j = \frac{g_j - 1}{\alpha_j}$$

so that

$$\sum_{i=1}^{n} F_i^{ext} \sqrt{(x_i - x_i^0)^2 + (y_i - y_i^0)^2} = \sum_{j=1}^{m} \left(\frac{1}{2} k_j \left(|\vec{r}_j| - g_j L_j \right)^2 - m_j c_j \frac{g_j - 1}{\alpha_j} \right). \tag{A.11}$$

or

$$\sum_{i=1}^{n} \mathbf{F}_i^{ext} \cdot \mathbf{x}_{disp} = \sum_{j=1}^{m} \frac{1}{2} k_j \Delta L_j^2. \tag{A.12}$$

where $\mathbf{x}_{disp}$ is the node displacement vector and I let $g_j = 1$ for the case where there are no

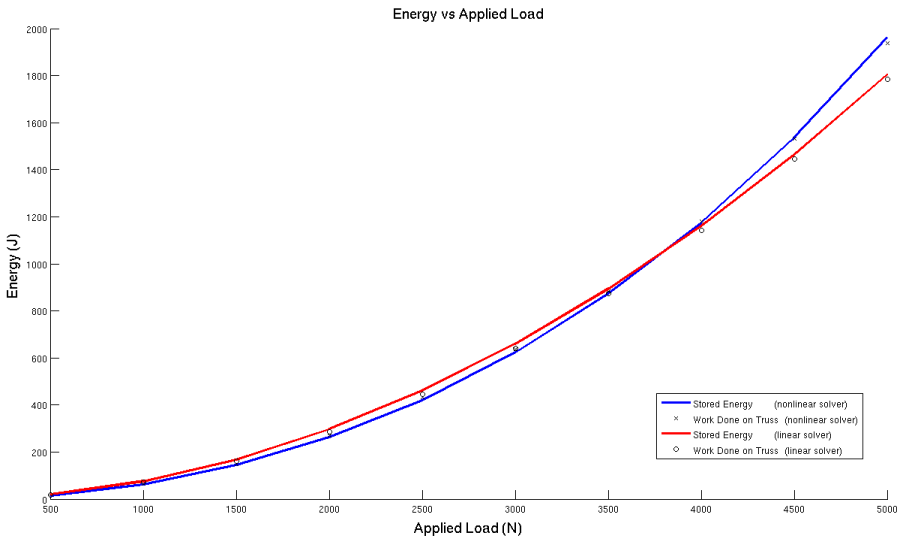


Figure A.1: Curves showing stored energy and input work vs load for a prototype 42 bar truss. The truss is a pin-pin planar truss with a single load at one location in the center. Note that the stored energy is equivalent to the input work and that there is a difference between the linear and nonlinear solvers predicted response.

thermal/growth loads.

A.1.2 Discussion

The equilibrium position of the truss is the position that minimizes the stored energy subject to any constraints.

If there exists a lower energy configuration for a system than that system will relax to that configuration, provided that the new configuration is immediately accessible. That is to say, that the "relaxation" path to the new configuration is permissible according to the system's constraints. The constraints on the system are material properties, external forces, *and* its initial configuration. Thus, when we load a truss it deforms to minimize energy.

Now, the energy of the structure is given by equation (A.7) or equivalently equation (A.8). Looking at equation (A.7) we see that

$$
\begin{aligned}
U_{final} &= \frac{1}{2}\sum_{j=1}^{m} k_j \left(|\vec{r}_j| - g_j L_j\right)^2 \\
&= \frac{1}{2}\sum_{j=1}^{m} \frac{1}{k_j}\Big(k_j \left(|\vec{r}_j| - g_j L_j\right) \cdot k_j \left(|\vec{r}_j| - g_j L_j\right)\Big) \\
U_{final} &= \sum_{j=1}^{m} \frac{1}{k_j}|T_j|^2
\end{aligned} \tag{A.13}
$$

Now, if $k_j = k$, so that the stiffness is the same for all members then

$$
U_{final} \propto \sum_{j=1}^{m} |T_j|^2. \tag{A.14}
$$

Therefore, for a structure composed of similar members minimizing the sum of the forces in the structure is equivalent to minimizing the strain energy of the structure.

It is also worth noting the following: for a cross sectional area A, length L, and modulus of elasticity E, the stiffness is defined as $k = EA/L$. By fixing stiffness we implicitly allow material properties and dimensions to vary so that ratio of EA to L remains constant.

If we further choose a particular material (with a fixed elasticity) then increasing a member's length requires a commensurate increase in its cross sectional area so that $A/L = constant$.

If the structure is not being "heated" then the total strain energy is equivalent to the work done on the structure. In other words, minimizing the U_{final} is equivalent to minimizing the displacement of the loaded joint.

A.2 Disk packing

In chapter three the MOD algorithm was introduced and applied to the generation of trusses. As it is described, the algorithm may also be applied to similar problems, one example being the disk packing problem.

Determining the most efficient packing of disks in a given planar volume is a common optimization problem and has applications in engineering. The simplicity of the problem statement belies its difficulty; finding optimal packings for more than a small number of disks is a difficult task and remains an area of active research. Initial progress was made with analytical techniques but researchers have since turned to optimization methods, for example [24] [28].

The standard problem may be stated as "Given a circular container, determine the configuration of N equal sized disks such that the container's radius is minimized." In order to

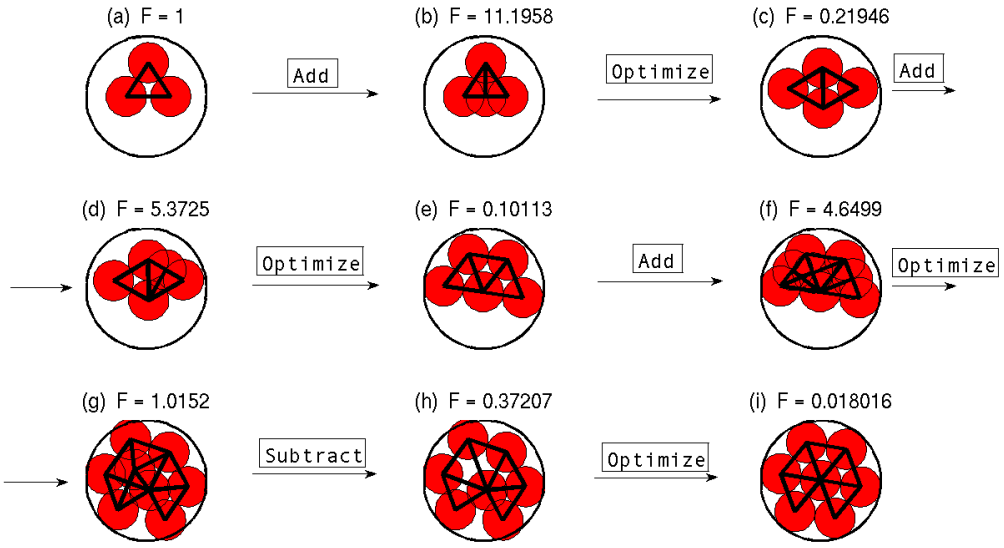


Figure A.2: The first nine iterations of the solution process. Note that the value of the cost function has been normalized relative to the initial value.

demonstrate features of the MOD algorithm we will tackle a twist on this problem: Given a container of fixed radius, and disks of fixed radius, determine the configuration of the greatest number of non-overlapping disks that can fit in this volume.

Cost function and minimization

We consider the packing problem for seven disks. The solution is well known and was first obtained by Graham in 1968 [23]. To obtain this solution we adopt a cost measure based on a quasi-physical model of the system, allowing demonstration of the relevant features of the MOD algorithm.

Consider the following: when two disks are compressed against one another they contribute the compression energy $E_{i,j} = \frac{1}{4}k(d - r_i - r_j)^2$ where k is the stiffness, d is the distance between them, and r denotes the radius of the i^{th} or j^{th} disk. (The extra factor of $1/2$ comes from adding springs in series where each spring has stiffness k). Note that a pair of disks is in compression when $d < r_i + r_j$.

Similarly, if a disk is compressed against the boundary of the circular container it contributes the energy $E_i = \frac{1}{2}k(p_i + r_i - R)^2$ where p_i is location of the center of the $i^{th} disk$ and R is the radius is the container. The disk is compressed against the container when $d + r_i > R$.

Now, let's also allow certain disks to be "linked" together. These disks will also contribute

energy in tension such that $\tilde{E}_{i,j} = \frac{1}{4}k(d - r_i - r_j)^2$ when $d > r_i + r_j$. Although in reality the disks are not linked together, optimal configurations require that certain disks are in contact. Adding this feature creates persistent structures in the graph which we may use to keep track of performance of the graph.

Finally, the total energy is $E_{tot} = \sum E_{i,j} + \sum E_i + \sum \tilde{E}_{i,j}$.

From this model we adopt the following costs measures:

$$\mathbf{C} = \tilde{E}_{i,j} \tag{A.15}$$

and

$$C = [E_{tot} + 1]\left[(n-7)^2 + 1\right]. \tag{A.16}$$

where n is the number of disks.

The small size of this problem allows the minimum value of C to be efficiently found with Matlab's $fsolve$ command using the Levenberg-Marquardt algorithm.

Equation (A.15) allows us to "put numbers on" how well a particular configuration is at a local level. Equation (A.16) is a contrived aggregate objective function that has the appropriate minimum at $n = 7$ and incorporates the penalty information of disk compression. It is adopted for the following reasons:

1. It allows for configuration optimization since it keeps track of compression via the incorporation of the term E_{tot}, which is configuration independent.

2. It has a minimum at $n = 7$. This is a well known problem with an optimal configuration consisting of 7 disks. The minimum is made explicit in the objective function in order to illustrate certain features of the MOD algorithm.

3. Both bracketed terms have minimums of one, not zero, in order to prevent artificial minimums (Optimal configurations of less than 7 disks will have zero compressive energy despite not efficiently "packing" the available space).

Finally, the stiffness coefficient was chosen to be $K = 50$.

Results

The MOD algorithm is conceived to operate in cycles of topology modification and optimization. Since the initial structure is minimal, the default is to begin with the division cycle. Initially, this

causes the number of nodes and edges (which represent the disks and their enforced interactions) to increase.

An example of this process can be seen in Figure (A.2). First, the initial structure, (a), is caused to divide producing (b). This is followed by an optimization step yielding (c). Note that the cost initially increases before decreasing to the value given shown in (c). Because the previous division/optimization cycle yielded a reduction in cost functional the next cycle is also a division cycle, which produces (d-e). Again, the cost at the end of the cycle (e) is lower than the cost of configuration in the beginning, (c), which prompts another division cycle, (f-g). This cycle yields (g), which has a higher cost than (e), triggering the subtraction cycle (h-i). This cycle yields the optimal configuration (although noise in the optimization step means that the cost function has only obtained a local minimum).

Note that if one strategy yields an increase in the cost functional after a cycle the other strategy is chosen. This means that, in the previous example, the next cycle would have been a subtraction cycle, potentially driving the configuration away from the optimal configuration. The dynamic nature of the sequencing algorithm means that convergence is not determined by reaching a "fixed point" but rather a stable cycle. Since the goal of this procedure is to aid in ideation, this dynamism is seen a benefit of the algorithm.

APPENDIX B

CONSTRAINTS, REPRESENTATION, AND HEURISTICS

B.1 Truss shape constraints

The optimization problem which we consider in Chapter 2 and in [55] is defined below:

$$\begin{aligned} &\text{minimize} && J(\mathbf{r}) \\ &\text{such that} && h_i(\mathbf{r}) \leq 0 \end{aligned}$$

where J is defined in the paper and their are q nonlinear inequality constraints. The vector $\mathbf{r}$ contains the node locations of the structure and the constraints are defined rigorously below.

1. **Maximum length constraints**

   There are m length constraints of the form

   $$h_i(\mathbf{r}) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} - L_{max} \tag{B.1}$$

   where the indices indicate the a and b joints of the ith member and m is the number of members.

2. **Minimum proximity constraints**

   There are

$$\frac{n!}{2!(n-2)!} = \binom{n}{2}$$

proximity constraints of the form

$$h_i(\mathbf{r}) = L_{min} - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{B.2}$$

where the indices indicate the ith and jth nodes of the structure and n is the number of nodes.

3. **Obstacle avoidance constraints**

   Although constraints on a design space may be modeled may different ways, the model was developed here, as follows:

   Both the truss and the obstacles may be considered to be a collection of line segments. The truss is considered to violate a constraint when one of its members (line segments) intersects one of the obstacles. A line segment may be parameterized as

$$\begin{aligned} \mathbf{r}_p &= \mathbf{r}_b + t_k(\mathbf{r}_b - \mathbf{r}_a) \\ &= \mathbf{r}_b + t_k\mathbf{v_1} \end{aligned} \tag{B.3}$$

   where $\mathbf{r}_a$ and $\mathbf{r}_b$ are the vectors that locate the ends of a line segment and t_k is a scalar. The vector $\mathbf{r}_p$ will lie in the line segment when $0 \leq t_k \leq 1$. Unless they are parallel, two lines must intersect. The intersection of two line segments may be found by parameterizing two lines with equation (B.3) and setting them equal so that

$$\mathbf{r}_b + t_k\mathbf{v_1} = \mathbf{r}_1 + t_l\mathbf{v_2}$$

   where $\mathbf{v}_2 = \mathbf{r}_1 - \mathbf{r}_0$. Solving for t_k and t_l we find

$$t_k = \frac{|(\mathbf{r}_1 - \mathbf{r}_b) \times \mathbf{v}_2|}{|\mathbf{v}_1 \times \mathbf{v}_2|} \qquad\qquad t_l = \frac{|(\mathbf{r}_1 - \mathbf{r}_b) \times \mathbf{v}_1|}{|\mathbf{v}_1 \times \mathbf{v}_2|}. \tag{B.4}$$

Thus, the lines are intersecting when the following four conditions are met:

$$\begin{aligned} h_i &= -t_k \leq 1 \\ h_{i+1} &= \quad t_k \leq 0 \\ h_{i+2} &= -t_l \leq 1 \\ h_{i+3} &= \quad t_l \leq 0 \end{aligned}$$

where the subscript indicates the k^{th} member and the l^{th} line segment. For t line segments there are an additional $4mt$ nonlinear inequality constraints.

In summary, the number of inequality constraints is

$$q = m + \binom{n}{2} + 4mt \tag{B.5}$$

where there are m members, n nodes, and t obstacles.

B.2 Example figures

This section includes examples of division heuristics referred to in Chapter 4. Comparison with the example given in Figure 4.5a shows that the additional heuristics used in that section (for connecting new nodes and discarding members with multiple crossings) produce "cleaner," more recognizable structures than those shown here.

B.3 Representation

The representation of space trusses was found to be very important for shape optimization; proper representation provides faster convergence, insures symmetry, and makes increases the likliehood of identifying an optimal truss.

B.3.1 Simple Representation

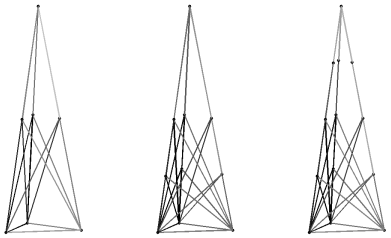
A simple representation can be constructed as follows: let $V = [\mathbf{x}, \mathbf{y}, \mathbf{z}]$ where $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are the column vectors of V. Then $X = [\mathbf{x}^T, \mathbf{y}^T, \mathbf{z}^T]$ where ^{T} is the transpose and X is formed by concatenating the columns of V. If joints (or components of joints) are not to be optimized than we may exclude them to form the truncated vector of design variables $\tilde{X} \subset X$. This method is straight forward to implement but its performance was found to be inferior to the representations discussed in the following sections. In particular, it did not reliably preserve symmetries and will not be discussed further. Good examples of this method are shown in 4.5b and 4.5a.

B.3.2 Thermal Representation

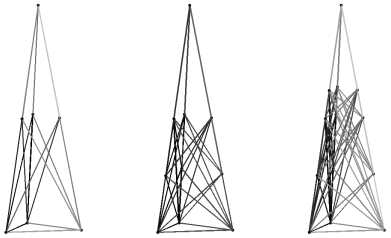
Building on the work described in Chapter 2 and we may parameterize the node locations of the truss in terms of the thermal properties of its members.

Begin by loading the truss and obtaining the member forces, F_i and determine the "groups" of like members. (If members carry the same force under load, they are in the same group). If performing sizing optimization, simply vary the radius of the group members together.

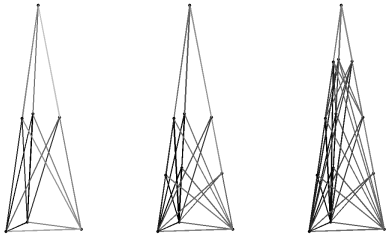
If optimizing shape, let the design variables be the thermal displacements of the lead member of each group. Now, determine the shape of the structure by applying the lead member's thermal displacement to the entire group. The resulting displacement in the structure defines the its new shape. The structure must now be re-evaluated (with this new shape) to determine the performance of the structure without the influence of the artificial thermal loads. In this manner the number of design variable is reduced to n_g and the resulting structure preserves symmetries present in the original.



(a)



(b)



(c)

Figure B.1: A tower is generated with three division steps using different methods: (a) 'set-difference' Note that the final truss is unstable due to the unsupported nodes at the top. (b) 'set-union', and (c) 'N-cycles.'

B.3.3 Orbital Representation

This mapping works by finding sets of joints which are symmetric about the structure's midpoint. The optimization is then carried out for a single member of each set while the other joints are recovered by pre-determined rotations about the origin.

Begin by translating to a new coordinate systems $\tilde{\mathbf{v}}_i = \mathbf{v}_i - [\mu_x, \mu_y, \mu_z]$ where $\mu_{x,y,z}$ is the mean of $\mathbf{x}, \mathbf{y}$ or $\mathbf{z}$ and i denotes the i^{th} row (a joint) of V or $\tilde{\mathbf{v}}$. Now let $R = \{r_1, \ldots, r_n\}$ where $r_i = ||\tilde{\mathbf{v}}_i||$ for $i = 1, \ldots, n$. Calculate the percent difference between the element of R and every other element of R to find n *choose* 2 distances. Where the percent difference is less than a tolerance, τ, "link" these two elements in set so that we may decompose R into connected subsets so that $R = \{R_1, R_2, ...R_{n_g}\}$ where each R_{n_g} is a set of joints of $\tilde{\mathbf{v}}$ that are the same distance from the origin (they occupy the same orbit) and there are n_g groups.

Let the first group be denoted $R_1 = \{\tilde{\mathbf{v}}_i, ..., \tilde{\mathbf{v}}_k\}$ where i and k indicate the i or k component of $\tilde{\mathbf{v}}$ that belongs to R_1 and there are n_g^1 vectors in the set. For convenience, relabel the group elements $R_1 = \{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, ...\tilde{\mathbf{v}}_{n_g^1}\}$ and refer to the first member of the set as the "principle" vector. We may express the other members of the group in terms of a rotation, A, from the principle so that

$$\tilde{\mathbf{v}}_j = A\tilde{\mathbf{v}}_1. \tag{B.6}$$

where $j \in [1, 2, \ldots, n_g^1]$.

Now, there are many possible rotations that could be considered, each one providing different results. For this work we used the following: If no elements of the principle vector, $\tilde{\mathbf{v}}_1$, are zero then the elements of A are

$$A = \begin{bmatrix} x_j/x_1 & 0 & 0 \\ 0 & y_j/y_1 & 0 \\ 0 & 0 & z_j/z_1 \end{bmatrix}. \tag{B.7}$$

If only one element of the principle is zero, say $y_1 = 0$, then

$$A = \begin{bmatrix} x_j/x_1 & 0 & 0 \\ 0 & 1 & y_j/z_j \\ 0 & 0 & z_j/z_1 \end{bmatrix}. \tag{B.8}$$

In this way A preserves proportional distance from the origin of each element of the set. In this way we can reduce the number of design variables to $3n_g$ and the resulting truss preserves

symmetries present in the original. In principle it is possible that nodes might orbit at the same radial distance but at different inclinations and in this case our method would cause them to be grouped together. However, this was not found to be a problem and this case was not addressed.