From analysis to design: A new computational strategy for structural creativity

Caitlin Mueller* and John Ochsendorf
caitlinm@mit.edu, jao@mit.edu
Building Technology Program, Massachusetts Institute of Technology, United States

Abstract: Since the introduction of finite element analysis software in the 1970s, structural engineers have become increasingly reliant on computational tools to carry out sophisticated simulations of structural performance. However, most structural analysis tools can only be used once there is a structure to be analyzed; they are not directly applicable in the design or synthesis of a new structural solution. This paper presents new research that expands the applicability of computation from structural analysis to structural design, with an emphasis on conceptual design applications. Specifically, this paper introduces a new interactive evolutionary framework implemented in a web-based structural design tool, structureFIT. This approach enables users to explore structural design options through an interactive evolutionary algorithm, and to further refine designs through a real-time analysis mode. This paper includes a critical background on optimization and its applications in structural design, an overview of the original interactive evolutionary framework, a description of the design tool, and a discussion of potential applications.

Keywords: conceptual structural design, structural optimization, computation, evolutionary algorithms

Introduction

Conceptual Design of Architecture and Structures

In building design disciplines, including architecture and structural engineering, the design process is conventionally divided into four sequential phases: Conceptual Design, Schematic Design, Design Development, and Construction Documents. In practice today, major decisions regarding the building's geometry, massing, and overall form are usually made during the first phase, Conceptual Design (Hsu and Liu 2000; Wang *et al.* 2002). This phase is typically carried out by the architecture team alone, before strong involvement of engineering consultants.

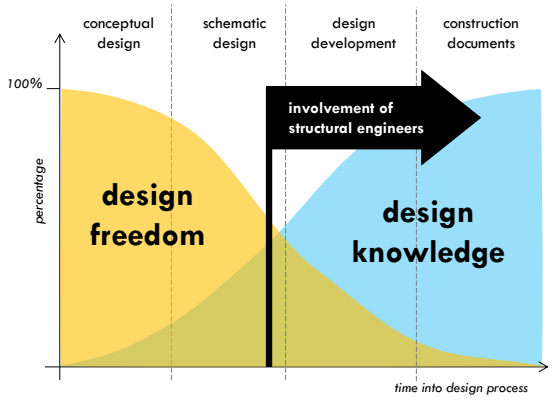

Figure 1. Relationship between design freedom and design knowledge in building design projects. After Fabrycky and Blanchard (1991) and Paulson (1976).

After the project has already taken shape, structural engineers and other consultants typically begin work, with the task of developing engineering strategies to enable the conceptual design vision, as illustrated in Fig. 1. This means that in standard practice, structural considerations are often subservient to architectural goals (Macdonald 2001). The design process is necessarily linear and unidirectional, and there are few opportunities for structural input to inform or improve the initial concept in significant ways (Holgate 1986).

Significance of Structural Form

History, theory, and nature show that for structural performance, overall form matters much more than material, member sizing, or internal topology (Thompson 1942; Zalewski *et al.* 1998; Larsen and Tyas 2003; Allen and Zalewski 2010). The geometry of a building's structure directly determines the distribution and magnitude of the forces it must resist (Macdonald 2001). Uruguayan structural designer Eladio Dieste (1917 – 2000) is quoted in an elegant expression of this point: "The resistant virtues of the structures that we seek depend on their form; it is through their form that they are stable, not because of an awkward accumulation of material. There is nothing more noble and elegant from an intellectual viewpoint than this: to resist through form" (Zalewski *et al.* 1998).

Today, with advances in a broad range of technologies, it is possible to design, analyze, and build forms regardless of their structural performance (Addis 1994). In fact, there is a recognized ingenuity

in meeting the challenge of making a structurally poor forms work in spite of their inefficiencies (Macdonald 2001). However, this does not mean that this is the best way forward. This paper argues for and presents an alternate paradigm in which structural considerations are integrated into the form-making phase of the design process, conceptual design.

Existing Computational Design Tools

Today's architecture and engineering practices make widespread use of computational tools throughout the design process, and currently available tools both reflect and enforce existing design paradigms (Hsu and Liu 2000; Wang *et al.* 2002).

Geometry-based Tools for Architects

Architecture tools, starting with Computer-Aided Drafting programs in the 1980s, allow users to thoroughly document, and more recently generate, both conceptual and detailed designs. An increasing interest in complex geometry has led to powerful 3D modeling software which, coupled with scripting capabilities, enables the development of impressively complex forms.

Analysis-based Tools for Engineers

Computational tools for structural analysis mirror architecture tools in their power and capacity for complexity, and yet also maintain existing design roles. Finite element analysis (or FEA) programs are capable of determining stresses, deflections, and dynamic behavior for highly complicated geometry using very sophisticated techniques. Recent developments focus on increased accuracy and speed under a range of conditions. However, these tools are of little use in conceptual design; they require a geometry be provided to be analyzed, and are incapable of assisting with geometry generation. Again, these tools relegate engineers to the tasks of verifying the form and sizing the members, thus limiting or eliminating their involvement in conceptual design.

Key Structural Design Tool Features

The emerging research area of computational structural design tools seeks to bridge the gap between these existing computational approaches, enabling a true integration of structural input during conceptual design. This paper identifies two key types of features for such tools, feedback and guidance.

Feedback Features

A clear remedy for the lack of performance evaluation in geometry-generation tools is to integrate structural analysis capabilities into such software. It is critical that such analysis be fast, or ideally real-time, to allow for an interactive user experience. This type of feature shows users how design changes will affect structural performance according to metrics such as required material volume, structural stiffness, or estimated construction costs. This has been implemented in a number of applications both in research and practice, but is limited by the speed of computational structural analysis.

Guidance Features

To shift engineering software from the existing analysis and verification focus, tools for structural design should include form-guiding capabilities. This type of feature enables the software to suggest new geometries to the user in order to improve the structural performance of a design concept. While the field of optimization offers insight into ways to achieve this, there has been little progress in developing guidance-based tools for conceptual design both in research and practice. To truly encourage integrated conceptual structural design through modern computational tools, it is critical that methodologies that achieve this functionality be further developed.

Optimization in Structural Design

Structural optimization is a promising field with a rich history, but has nevertheless yet to make a significant impact on structural design in practice. This section explains the development of structural optimization theory and discusses the reasons for its disconnect with design.

The history of structural optimization can be traced back to Galileo Galilei (1564 – 1642), who in 1638 determined the optimal shape of a cantilevered beam subjected to a point load at its free end (Timoshenko 1953; Heyman 1998). By finding the parabolic profile, as illustrated in Fig. 2, Galileo showed that mathematics can be used to find forms that use material as efficiently as possible to support a given load. For many years since, this has been the goal of structural optimization.

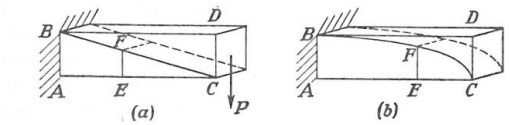

Figure 2. Drawings from Galileo's *Dialogues Concerning Two New Sciences* (1638), showing in (a) an incorrect linearly varying solution for the optimal shape of a cantilevered constant-width beam supporting a point load at its tip, along with (b), the correct parabolically varying solution (Timoshenko 1953).

Since Galileo, scholars have solved a steady stream of increasingly complex structural optimization problems (Wasiutynski and Brandt 1963). One of the most well-known contributions comes from Anthony G. M. Michell's work on another cantilever problem almost three hundred years after Galileo's original work. Michell showed how to find an optimal truss solution for the point-loaded cantilever problem (and a few others) in his seminal 1904 paper, "The Limits of Economy of Material in Frame-structures." Like Galileo, Michell was looking for minimal-material analytical solutions for key canonical problems, rather than offering a general approach for optimization of any structure. (Timoshenko 1953; Heyman 1998).

A more general approach that resembles methods in use today was developed in the 1960s, with critical work by Schmit (1960). A cohesive overview of work since is given by Spillers and MacBain (2009). In contrast with the analytical methods of scholars like Galileo and Michell, the new numerical methods attempted to find the optimum by iterating through potential solutions in a systematic way (Kirsch 1981). While iterative approaches were practically impossible in the days of manual calculation, the newly developed computers brought rapid calculations for large problems to reality.

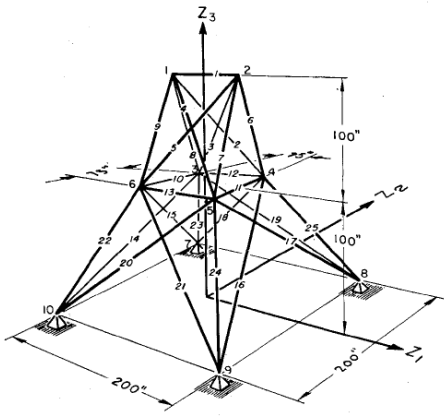

Figure 3. 25-bar trussed tower with member cross sectional diameters and wall thicknesses chosen by an optimization algorithm (Fox and Schmidt 1966).

Importantly, structural optimization researchers in the 1960s referred to their discipline as *structural synthesis* (Schmit 1981; Vanderplaats 2010), revealing the early aspirations of the field and evoking ideas of design in its truest sense: creating something new. However, the work actually dealt with choosing member cross sections for predetermined geometries and member configurations (Fox and Schmit 1966). For example, Fig. 3 shows a three-dimensional truss tower with 25 elements, whose cross sections were selected using a numerical weight minimization algorithm. This type of problem is referred to as size optimization. While improvements since the 1960s have broadened the reach of structural optimization strategies, the general disconnect between the goals and reality of structural optimization persist still today. In short, although structural optimization aims to generate new and exciting forms, most applications are limited to rather narrow problem spaces.

An important step forward in structural optimization was the development of shape optimization, or the determination of overall structural form as opposed to element sizes (Vanderplaats 1982; Bennett and Botkin 1986; Haftka and Grandhi 1986). Most applications of this early work were in structural design of components in the automotive and aerospace industries, where an improved part would be used hundreds or thousands of times, yielding extensive savings, although there are also examples of shape optimization for trusses, sometimes called geometry optimization. Because it deals with overall form, shape optimization is more relevant to conceptual design than size optimization.

The third type of structural optimization used today is topology optimization, or the optimal connective arrangement of elements in a structure, developed numerically in the late 1980s (Bendsøe and Kikuchi 1988; Rozvany 2001; Rozvany 2007). This type of optimization can also be integrated with shape optimization and size optimization.

Specific methods have been developed to address each of the three classes of structural optimization problems, but in general they share a common formulation, described in the following subsection.

Optimization Problem Formulation

Formally, structural optimization is a numerical method of finding the best solution according to mathematically formulated functional requirements, or objectives, while conforming to mathematically formulated constraints. The solution is expressed in the form of numerical values for a design vector, x, which represents a list of design decisions to be made – for example, nodal positions, material selections, cross sections – called design variables.

The objective function, $f(x)$, is often a calculation of the weight or volume of the structure, such that a minimal-weight structure can be found. However, this function can also consider stiffness, strain energy, deflection, or other quantitative goals, structural or otherwise. The constraints, $g(x) \leq 0$ and $h(x) = 0$, and the variable bounds, $x_{i,lb}$ and $x_{i,ub}$, restrict the solutions according to design or behavioral requirements. More specifically, design constraints can represent geometric or spatial requirements, constructability or fabrication limitations, or other functional considerations. Behavioral constraints set limitations on structural

behavior, and include restrictions on performance metrics like internal stresses, deflections, or buckling capacity (Kirsch 1981).

Together, the design vector, constraints, variable bounds, and objective function define a design space, or solution space, for a given problem. The dimension of this space is given as one more than the number of design variables, to represent the space of possible design vector values and their resulting objective, or performance, values. Structural design problems often have design spaces that are large and complex, although the exact nature of the design space depends on the specifics of the problem.

Limitations of Optimization in Design

Despite the rich academic history of structural optimization, it has had relatively little impact on structural engineering in practice. Fundamentally, this can be attributed to an inherent difference in goals between optimization and the design of buildings. While optimization is necessarily a convergent process, or one in which an iterative and systematic algorithm converges upon a single solution, design is decidedly divergent. In design, it is recognized that a variety of significantly different yet suitable solutions can be found from a single starting point.

Moreover, the exercise of mathematically formulating objectives and constraints is difficult or impossible in the design of buildings. Many goals and requirements are qualitative, or even subjective, such as visual impact, spatial experience, contextual fit, and overall architectural value. Since most structural design cannot occur in the absence of architectural goals, this presents a significant challenge.

In addition, the design process for buildings is often one of discovery: designers do not know all of their objectives and constraints at the beginning of the process, but develop them as they explore design possibilities. The designer's interaction with the process of evaluation and iteration is key. In contrast, standard optimization is a relatively rigid and automated process in which goals and requirements must be enumerated completely at the start. Unlike the human design process, optimization on its own cannot handle unformulated objectives and constraints.

Finally, most structural designers lack intensive training in optimization, and there are few tools or approaches available that make optimization accessible to non-experts. Furthermore, optimization tools that do exist are often text-based or severely limited in their graphical displays, and often rely on piecing several pieces of software together. Human designers are necessarily highly visual, and can process and evaluate information much more quickly and fully when it is presented graphically. Therefore, in order to be useful for designers in practice, tools that use optimization should be easy to use, integrated, and strongly graphical.

Interactive Design Space Exploration

Given the issues with standard optimization in conceptual structural design, it is necessary to look beyond the established approaches to find ways to bring computational design guidance to conceptual design tools.

Interactive optimization addresses this issue in a simple but compelling way: the designer is allowed to interact with the computer algorithm in deciding which designs to pursue in the iterative optimization process. The exact mechanics of the interaction depend on the specific algorithm chosen. In general, the interactive element allows the user to only partially formulate the design problem in a quantitative way, and to use unformulated or newly discovered objectives and constraints to make design selections.

Interactive Evolutionary Algorithms

Evolutionary algorithms are a general class of optimization strategies that use the principles of Darwinian natural selection to grow and evolve populations of designs. They have the advantages of being robust and well-suited to complicated engineering problems. Because they incorporate randomness, they avoid getting stuck in local optima, and can effectively hop around the design space in search of better solutions.

Furthermore, because they work with populations of candidate designs, evolutionary algorithms are especially useful in promoting design diversity. Unlike algorithms that focus on improving singular solutions, these algorithms improve a group of alternative options as they iterate. The general procedure is to randomly initialize a first generation, evaluate the fitness of each member of the generation, identify the top performers, and use those to create a subsequent generation by combining and mutating them. In standard evolutionary algorithms, the process runs automatically until preset criteria are reached, and a single solution is presented as the optimum. However, it is also possible to take better advantage of the design diversity created by this approach by incorporating human interaction.

On their own, evolutionary algorithms are subject to the same criticisms as other standard optimization approaches, as detailed previously. However, because of their population-based approach and selection mechanics, evolutionary algorithms lend themselves particularly well to human interaction. Interactive evolutionary algorithms are a subclass of optimization algorithms that use principles of evolution combined with human input to drive design space exploration. The general iterative

process for this type of algorithm is illustrated in the diagram in Fig. 4. The cycle differs from standard evolutionary algorithms at the design selection step. The algorithm identifies top performers, but solicits input from the user to make final choices about which designs to proceed with to form the subsequent generation. This key difference allows the designer to adjust the optimization process based on unformulated goals, such as visual impact or constructability requirements. Furthermore, the user may adapt goals across generations, based on newly realized design criteria discovered in the explorative process.

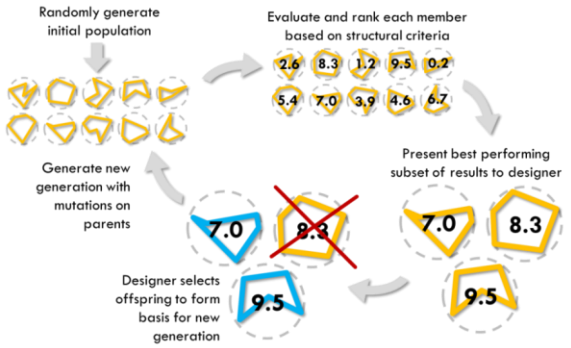

Figure 4. General diagram of an interactive evolutionary algorithm, including the interactive step highlighted in blue.

The first interactive evolutionary algorithms were developed in Sims (1992) for the purpose of finding visually interesting cellular automata. In this early case, selection was entirely based on user preferences, rather than on a combination of user preferences with calculated objective functions. The literature includes many subsequent examples of this strict type of interactive evolutionary algorithm, including for the design of web pages (Oliver *et al.*, 2002) and coffee blends (Herdy 1997).

Contributions from Parmee and collaborators led to some of the first interactive evolutionary algorithms that used both computation and human input to drive selection (Parmee 1997; Parmee and Bonham 2000; Parmee 2001). Unlike the earlier examples, which focused on design problems with highly subjective performance metrics, this work is in the realm of engineering, which has both quantitative and qualitative goals. This work laid the foundations for further research in the applications of interactive evolutionary computation to structural design.

More recently, some progress has been made in applying interactive evolutionary computation specifically to the realm of structural design. Most notably, von Buelow has proposed an interactive genetic design tool for creative exploration of design spaces, including for the design of trusses (2008) and folded plate structures (2011).

Specific Needs

Existing work suggests specific challenges to be addressed by a new interactive evolutionary framework. First, existing approaches implement interactivity in limited ways. Interactive features should be expanded to allow more incorporation of requirements and criteria from the designer. These features can also help the designer direct exploration of the design space in a more precise way, further improving the effectiveness of an interactive evolutionary approach.

Additionally, existing research treats interactive evolutionary algorithms as a stand-alone approach without considering the broader user design experience. There is a need to incorporate general problem setup strategies and design refinement functionalities into an expanded approach, along with the evolutionary approach itself.

The framework presented in this paper is a novel holistic approach that generalizes the use of interactive evolutionary algorithms in conceptual structural design, and also addresses these specific needs.

Interactive Evolutionary Framework

This section introduces a novel framework that adapts a generalized interactive evolutionary algorithm for conceptual structural design, as well as its implementation as a software tool. Detailed descriptions of specific original features of the framework are discussed more fully in subsequent sections.

Framework and Software Architecture

Existing work suggests specific challenges to be addressed by a new interactive evolutionary framework. First, existing approaches implement interactivity in limited ways. Interactive features should be expanded to allow more incorporation of requirements and criteria from the designer. These features can also help the designer direct exploration of the design space in a more precise way, further improving the effectiveness of an interactive evolutionary approach.

The software implementation of this framework reflects its generalized nature. The program is written in C#/.NET (Microsoft 2012), an object-oriented programming language, and is designed to be modular and extensible. There are four general types of backend classes: variables, design models, structural analysis engines, and the interactive evolutionary algorithm population generator. The population generator connects with a graphical user interface to allow input from the user. The interaction of these parts is illustrated in Fig. 5.

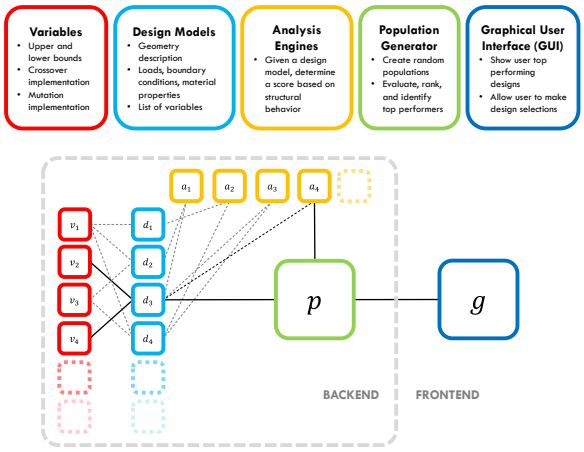

Figure 5. Software architecture diagram for the interactive evolutionary framework, illustrating main class types and interactions.

This diagram shows the versatile nature of the framework. Variables, design models, and analysis engines are all designed using interfaces, meaning that each can be implemented as a variety of types. For example, variable types can be horizontal and vertical nodal positions, but they could also be material properties, joint fixities, member topologies, or other design decisions. Design models can be truss structures, again as introduced previously, but they could also be frame structures, continuous solid structures, or other structural types. A design model type must have one or more analysis engine type that can apply to it. For example, truss structures are associated with a truss analysis engine, but could also be analyzed by more detailed analysis engine types. Examples of variable types, design model types, and analysis engine types are presented in the following subsections.

The population generator works with a particular design model type and a particular associated analysis engine type. Using the design model and its variables, it creates a generation through crossover and mutation. Using the analysis engine, it applies a fitness score to each candidate design. It then presents the best designs to the user through the graphical user interface, which also allows the user to make selections. These selections are sent back to the population generator, which produces a new generation.

Variables and Design Models

As discussed in the previous subsection, the interactive evolutionary framework supports multiple variable types and design model types. To illustrate how these classes work, the example of a truss design model with variable nodal positions will be used. Fig. 6 shows a seven-bar truss with three design variables. The truss model is defined by its nodes and members. Nodes are defined by degrees of freedom, which have coordinates, loads, and supports. In this two-dimensional case, nodes have two degrees of freedom. Members are defined by their start and end nodes and their material properties. Like all design model types, the truss model also has a vector of variables. This is the model's design vector, or parametric representation.

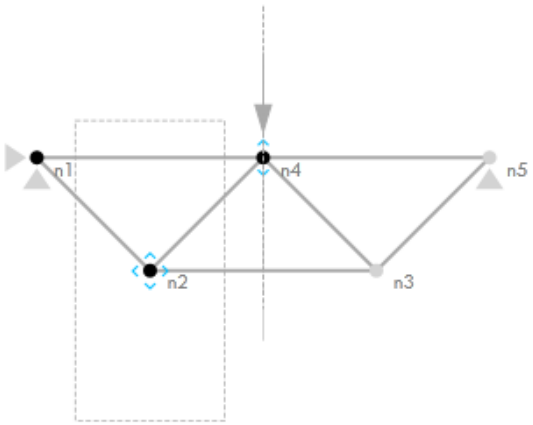

Figure 6. A planar seven-bar truss design problem with three design variables: the horizontal and vertical positions of the lower left node (n2), and the vertical position of the central node (n4). This truss is simply supported, has a central point load, and is bilaterally symmetrical.

In this type of design problem, the coordinate of each degree of freedom can be a variable. Any variable type must have defined upper and lower bounds. In this case, the upper and lower bounds are the allowable range for the coordinate, illustrated in Fig. 6 with the dashed rectangle for node 2 and line for node 4.

Additionally, any variable type must implement analogues of the biological concepts of crossover and mutation. Conceptually, crossover combines encoded information from more than one parent to create offspring that have traits from each of them. Mutation then randomly perturbs the newly formed offspring in order to encourage diversity. For this example, the implementations of mutation and crossover are given in Eqs. (1-6), and apply to continuous variables in general beyond the degree of freedom coordinate. Crossover is accomplished through a weighted average of seed variable values with random weights. Mutation updates a variable value with a random variable from a normal distribution with a standard deviation related to the variable's allowable range and set mutation rate. For discrete or integer variables, these same approaches can be used with minor modifications.

$$\text{Crossover: } x_{\text{crossed}} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} \quad (1)$$

$$\text{Mutation: } \mu = x_{\text{crossed}} \quad (2)$$

$$\sigma = \frac{|x_{\text{ub}} - x_{\text{lb}}|}{2} r_{\text{mutation}} \quad (3)$$

$$\text{Normal distribution: } f(x;\, \mu, \sigma^2) \quad (4)$$

$$x_{\text{rand}} = \text{rand}(f) \quad (5)$$

$$x_{\text{mutated}} = \min(\max(x_{\text{rand}}, x_{\text{lb}}), x_{\text{ub}}) \quad (6)$$

The framework also supports parametric relationships between variables and non-variables. For example, the truss design model presented here allows for mirror and offset relationships between degree of freedom coordinates. The former is illustrated in the problem shown in Fig. 6, which uses bilateral symmetry to define the position of the lower right node (n3) based on the position of the lower left node (n2).

Analysis Engines

Design model types must be associated with at least one analysis engine type, although the framework supports the use of multiple analysis engines. Any analysis engine must determine a quantitative fitness score for a given design model, based on structural criteria. For example, in the case of the truss model, a truss analysis engine can find the required volume of a structure with a given geometry, loading, and support conditions. The engine calculates this metric as follows: compute the forces in each member using the direct stiffness method, assign required cross sectional areas to each member based on allowable stress and buckling considerations, and find the sum of the area lengths times their required areas.

The code for this truss analysis engine was written from scratch, using the open-source Math.NET numerical analysis library for matrix operations (Math.NET Project 2012). However, analysis engines could also make use of commercial structural analysis codes. An important note is that for statically indeterminate structures, this particular process is affected by initial member sizes used to compute forces. In this case, optimal member sizing can be computed through iteration, or an approximate result found through initial equal member sizing can be accepted.

Population Generator

The population generator in this framework implements a simple and flexible interactive evolutionary algorithm that can be easily controlled by the user and adapted to a wide range of variable, design model, and analysis engine types. As explained previously, the interactive evolutionary algorithm is an iterative approach that can be repeated until the user is satisfied.

The first step of the algorithm is to generate a random population of a preset number of candidate designs. For the first generation, this is based on random perturbations from an initial structure defined by the user. Specifically, for each candidate design in the new generation, each design variable is mutated from initial values from the user-defined initial structure. Mutation is carried out in the manner previously discussed, and illustrated in Eqs. (2-6) for the example of continuous variables.

Next, the algorithm uses the analysis engine to assign a fitness score to each candidate design. The algorithm then sorts the designs according to this score and presents a top-performing subset of designs to the user through the graphical user interface. The user is then able to visually evaluate the designs and choose those that best meet the qualitative or otherwise unformulated goals for the design process. The designs that the user chooses are used as seeds for creating the next generation in the iterative process.

The seeds are used to form a new generation using the previously discussed crossover and mutation functionalities. The newly formed generation of new candidate designs is then evaluated, sorted, and presented again, and this process can continue as long as the user wishes. There are also several ways for the user to interrupt the process. If the user does not like any of the presented designs, or wishes to make changes to designs previously selected, the user can return to a previous generation, adjust selections, and rerun the algorithm from that point. Also, the user can choose to select no designs, and the algorithm will reset and start with the previously defined initial structure once again.

User Experience and Interface

The framework described above has been implemented in an interactive proof-of-concept design tool called structureFIT (Mueller 2013). The following sections describe the graphical user interface and general user experience.

Graphical User Interface (GUI)

The graphical user interface (GUI) enables the interactive step of the interactive evolutionary algorithm by showing the user top-performing designs graphically and allowing the user to make selections. The GUI is implemented using Silverlight, a platform-agnostic technology that supports interactive user applications that run in a web browser (Microsoft 2012). There are several advantages to this approach, in comparison with traditional desktop applications or integration into

existing software. First, the program is highly accessible: anyone with a web browser can use it, regardless of operating system, and there is no need to download or install it. Second, there is no need for the user to own other commercial software, such as Rhino or AutoCAD, to run the program, and the program is not tied to software trends, which tend to change relatively quickly in the architectural computation realm. Finally, the web-based interface lends itself naturally to analysis calculations on remote servers. While all calculations are currently executed on the client-side, or on the user's computer, future use of server-side calculations through remote resources or cloud computing could significantly improve performance.

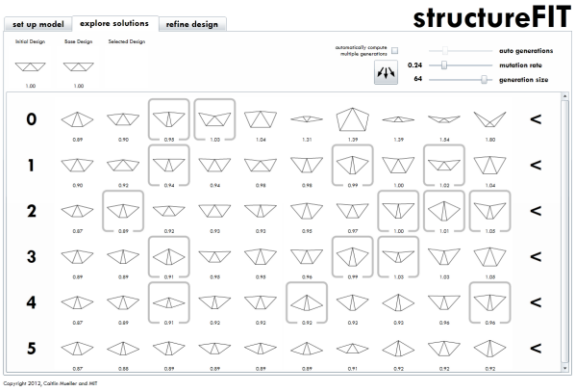

Figure 7. Screenshot of the web-based graphical user interface, showing the evolution of solutions for the design problem presented in Fig. 6.

Figure 8. A closer view of several candidate designs created by the population generator and presented to the user, with scores normalized by a base design's score shown underneath each.

A screenshot of the GUI is shown in Fig. 7. It is designed to be simple and user-friendly, while still allowing for powerful user control. The main feature of the interface is the matrix of designs, shown in numbered rows. Each row represents a generation created by the population generator, and the designs shown are the top ten performers. The number under each design corresponds to its score, normalized by the score of a base design, which is shown, along with the initial design, in the upper left-hand corner of the interface. Designs with scores less than 1.00 perform better than the base design, and those with scores higher than 1.00 perform worse. A closer view of generated designs and their scores is shown in Fig. 8. After each generation is produced, the user is able to select zero, one, or more designs by clicking on them, and selected designs are indicated with a gray square. The user then clicks the main "generate" button to produce a new generation.

The user can return to a previous generation by clicking the "<" button next to the corresponding row. This will erase the designs generated since, and the user can change the selected designs and rerun the computation. The user can also adjust the mutation rate and population size for each generation, and can choose to turn on a hybrid approach that automatically computes several generations in a row. These features are discussed in more detail in subsequent sections.

Expanded User Experience

In addition to the interactive evolutionary design experience, this framework includes original functionality that can be used before and after. Before evolutionary design exploration, the user can set up the design problem by drawing in a graphical and intuitive user interface. This makes the framework general beyond specific examples. After the evolutionary design evaluation, the user can refine an evolved design using real-time performance feedback. These additional features help bring this framework beyond an algorithm and toward an approach usable for real design problems.

The design setup mode allows the user to define a design problem by building a structural model and identifying variables. The user can draw a structure by clicking and dragging to create nodes and members on a canvas, or by modifying entries in an adjacent spreadsheet. The user can then assign loads and supports to defined notes, and define variables, including upper and lower bounds. Finally, the user can define planes of symmetry and parametric relationships, including mirror and offset relations. The information entered by the user is updated dynamically in the graphical view of the structural model. This functionality is illustrated in Fig. 9.

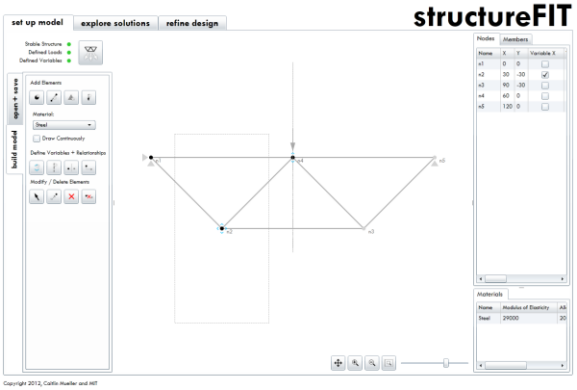

Figure 9. Screenshot of the model setup mode, in which the user can input a design problem, specified by structural geometry, loads, materials, boundary conditions, and variable definitions.

The user may also choose to open one of a range of preset design examples that can be run directly, or modified to adapt to new problems. Additionally, the user can choose to save a custom setup structure that can be opened again later in the design session. Once the setup structure has been finalized, the user can click the button in the upper left of the screen to set it as the initial design for the interactive evolutionary mode. If the structure is not stable, or contains no loads or variable definitions, the program will identify these issues for the user to correct.

This setup mode is important because it makes the interactive evolutionary framework both highly flexible and easy to use. The framework is not tied to any particular example or case study, and can be used by designers for real design problems. Additionally, the GUI for design input is powerful and user friendly, so that designers can define problems quickly and move on to exploring solutions in the interactive evolutionary mode.

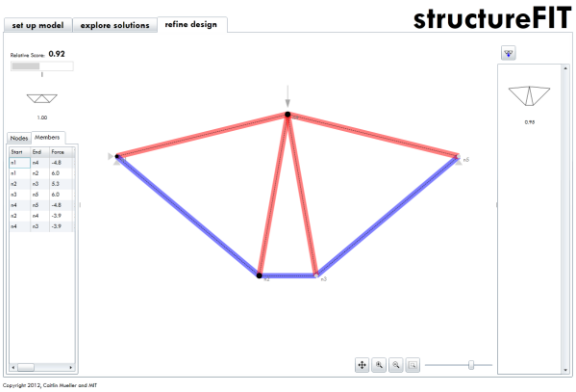

Figure 10. Screenshot of the design refinement mode, in which the user can adjust designs found in the interactive evolutionary exploration with real-time performance feedback in terms of the overall score and individual member sizing. The members are drawn with required thicknesses shown to scale, with blue indicating tension and red compression.

Once the user has found an interesting design, it can be studied and refined further in the design refinement mode. This mode allows the user to graphically adjust variable settings for a selected design to fine-tune its appearance, while also receiving real-time feedback on the performance implications of the adjustments. In the case of nodal coordinate variables, the user is able to adjust the nodal positions by clicking and dragging, and note the change in the overall design score. The program also instantly updates the required thickness of individual members, shown graphically on the members themselves and numerically in a spreadsheet. The user is able to save particular designs found in this design refinement mode and return to them for comparison. Once an attractive solution is found, the user can export it for use in more advanced modeling and analysis software. A screenshot of this design mode is shown in Fig. 10.

Like the model setup mode, the design refinement mode adds crucial functionality to the interactive evolutionary framework. By combining a guidance-based approach with a feedback-based post-processing step, the framework is able to expand design freedom for users.

Conclusions

This paper has presented a general framework for using interactive evolutionary optimization in conceptual structural design. This work is important because it helps enable a guided exploration of structural design spaces, while still allowing for creativity and freedom, addressing the issues found in standard optimization previously identified.

This framework builds upon existing work in interactive evolutionary algorithms and in structural design tools, addressing specific issues that remain unresolved in previous literature. The specific contributions include the generalized approach for interactive structural design as well as its graphical and interactive implementation in the form of the structureFIT design tool.

Applications

The framework and tool introduced here could significantly improve conceptual design exercises in practice, as a way to generate and compare a wide range of design ideas quickly and easily. An architect with basic structural knowledge could use the tool alone or as a supplement to working with a creative structural engineer early in the design process. A structural designer could also use the tool to develop innovative structural concepts to discuss with the architect for further development. In a more integrated approach, a team of architects and engineers could use the tool together during conceptual design, collaboratively developing design alternatives that perform well structurally and achieve architectural design goals. Finally, the tool could be useful in facilitating discussions between designers and clients, helping clients understand tradeoffs between options and cost implications of design ideas at the earliest stages.

Possible applications in the classroom mirror those in practice: architecture students could use a tool implementing this approach for exploring early design options for studio projects, and engineering students could use such a tool for engineering design projects. However, the design tool also has additional didactic potential for developing intuition for structural behavior in architecture and engineering students, a very important and increasingly neglected

aspect of education in both disciplines. For engineering students, this tool could also offer a way to encourage design creativity, another significant but overlooked area. Furthermore, a tool used by students from both disciplines together would foster collaboration and improve students' cross-disciplinary communication skills, which are much needed in practice.

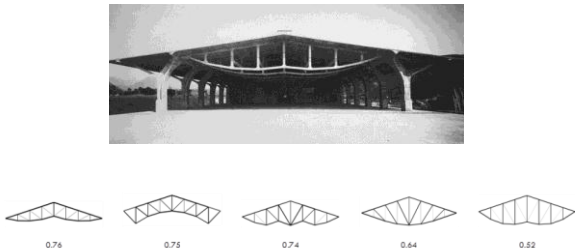

Figure 11. Robert Maillart's 1924 design for a shed roof in Chiasso, compared with designs discovered using the computational approach presented in this paper.

In addition to discovering design possibilities for new projects, this approach could also be useful in studying existing work within the context of a formal design space. Most architectural history research does not include detailed analyses on structural performance, which can be of value in evaluating success and identifying lessons to move forward with. The design space strategies used in this approach allow researchers to consider a historical work as a point in a space of alternatives of varying structural performance and formal attributes, potentially gaining insights on design decisions and process. For example, Robert Maillart's concrete shed roof in Chiasso, Switzerland, designed in 1924, is shown in Fig. 11, along with related design alternatives explored using the approach presented in this paper. It is evident that there is a family of solutions of varying performance, some of which share more in common with Maillart's design, which achieves a constant force in the gable elements, and some less. Such a study could provide a new context through which designs could be analyzed, understood, and revisited in the future.

References

Addis, B. (1994) The Art of the Structural Engineer. London: Artemis.

Allen, E., and Zalewski, W. (2010) Form and Forces: Designing Efficient, Expressive Structures. Hoboken, NJ: John Wiley and Sons.

Bendsøe, M. P., and Kikuchi, N. (1988) Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering, 71(2), pp. 197-224.

Bennett, J. A., and Botkin, M. E. (Eds.) (1986). The Optimum Shape: Automated Structural Design. New York: Plenum Press.

Computers and Structures. (2008). SAP2000 Version 12.0.0 Release Notes.

Fabrycky, W. J., and Blanchard, B. S. (1991) Life-Cycle Cost and Economic Analysis. Lebanon, IN: Prentice-Hall.

Fox, R. L., and Schmit, L. A. (1966) Advances in the Integrated Approach to Structural Synthesis. Journal of Spacecraft and Rockets, 3(6), pp. 858-866.

Haftka, R. T., and Grandhi, R. V. (1986) Structural shape optimization--A survey. Computer Methods in Applied Mechanics and Engineering, 57, pp. 91-106.

Herdy, M. (1997) Evolutionary Optimization Based on Subjective Selection - Evolving Blends of Coffee. Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing , pp. 640-644.

Heyman, J. (1998) Structural Analysis: A Historical Approach. Cambridge, UK: Cambridge University Press.

Holgate, A. (1986) The Art in Structural Design: An Introduction and Sourcebook. Oxford: Clarendon Press.

Hsu, W., and Liu, B. (2000) Conceptual design: issues and challenges. Computer-Aided Design, 32, pp. 849-850.

Kirsch, U. (1981) Optimum Structural Design. New York: McGraw-Hill.

Macdonald, A. J. (2001) Structure and Architecture (2nd ed.). Oxford: Architectural Press.

Math.NET Project. (2012) Math.NET Numerics. Retrieved from http://numerics.mathdotnet.com/

Michell, A. G. (1904) The limits of economy of material in frame-structures. Philosophical Magazine, 8(47), pp. 589-597.

Microsoft. (2012) Introduction to the C# Language and the .NET Framework. Retrieved from Microsoft Developer Network (MSDN): http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

Microsoft. (2012) Silverlight. Retrieved from http://www.silverlight.net

Mueller, C. (2013) structureFIT Design Tool. Retrieved from www.caitlinmueller.com/structurefit

Oliver, A., Monmarché, N., and Venturini, G. (2002) Interactive design of web sites with a genetic algorithm. IADIS International Conference WWW/Internet, pp. 355-362.

Parmee, I. C. (1997) Evolutionary and adaptive strategies for engineering design-an overall framework. IEEE International Conference on Evolutionary Computation, 1997, pp. 373-378.

Parmee, I. C. (2001) Evolutionary and Adaptive Computing in Engineering Design. Springer.

Parmee, I. C., and Bonham, C. (2000) Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies. Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, 14, pp. 3-16.

Paulson, B. (1976) Designing to reduce construction costs. Journal of the Construction Division, 102(4), pp. 587-592.

Rozvany, G. I. (2001) Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. Structural and Multidisciplinary Optimization, 21, pp. 90-108.

Rozvany, G. I. (2007) A critical review of established methods of structural topology optimization. Structural and Multidisciplinary Optimization, 37(3), pp. 217-237.

Schmit, L. A. (1960) Structural design by systematic synthesis. Proceedings of the Second Conference on Electronic Computation, pp. 105-122.

Schmit, L. A. (1981) Structural sythesis–Its genesis and development. AIAA Journal, 19(10), pp. 1249-1263.

Sims, K. (1992) Interactive Evolution of Dynamical Systems. Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 171-178.

Spillers, W. R., and MacBain, K. M. (2009) Structural Optimization. New York: Springer.

Timoshenko, S. P. (1953) History of Strength of Materials. New York: McGraw-Hill.

Vanderplaats, G. N. (1982) Structural optimization—Past, present, and future. AIAA Journal, pp. 992-1000.

Vanderplaats, G. N. (2010) Fifty years of structural synthesis: Some musings from a disciple of Schmit. 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, pp. 1-8.

von Buelow, P. (2008) Suitability of genetic based exploration in the creative design process. Digital Creativity, 19(1), pp. 51-61.

von Buelow, P. (2011) Genetically Enhanced Parametric Design for Performance Optimization. 7th International Seminar of the IASS Structural Morphology Group. London.

Wang, L., Shen, W., Xie, H., Neelamkavil, J., and Pardasani, A. (2002) Collaborative conceptual design -- state of the art and future trends. Computer-Aided Design, 34, pp. 981-996.

Wasiutynski, Z., and Brandt, A. (1963). The present state of knowledge in the field of optimum design of structures. Applied Mechanics Reviews, 16(5), pp. 341-350.

Zalewski, W., Allen, E., and Iano, J. (1998). Shaping Structures: Statics. New York: John Wiley and Sons.