Oct 19th, 12:00 AM

# Automated Design of Steel Trusses

B. Mobasher

S. D. Rajan

S. Y. Chen

Fifteenth International Specialty Conference on Cold-Formed Steel Structures
St. Louis, Missouri U.S.A., October 19-20, 2000

# Automated Design of Steel Trusses

S-Y.Chen[1], B. Mobasher and S. D. Rajan
Department of Civil and Environmental Engineering
Arizona State University
Tempe, AZ 85287

**Abstract**: Designing an automated procedure for the optimal design of any structural system poses special challenges. Converting this methodology into a practical tool is even more challenging. In this research, a point-and-click software system is developed for the optimal design of roof truss systems. The starting point is a roof template containing minimal user input - outline of the truss, truss spacing, load information, and cost figures. A ground structure is constructed as the starting point of the design iterations. The Genetic Algorithm (GA) is used as the optimization tool to drive the design changes. Using the database of available sections, the member cross-sections are selected for the top and bottom chords, and the webs. In addition, the number and layout of the web members is also determined. The final design is obtained so that the truss has the lowest cost and also satisfies AISI-LRFD design specifications. Numerical experience using the developed methodology and the software system on an Intel-based PC running Microsoft Windows OS shows that optimal designs can be obtained in a few minutes.

## Introduction

The cost effectiveness of using steel roof systems for residential buildings is becoming increasingly apparent with the decrease in manufacturing cost of steel components, reliability and efficiency in construction practices, and the economic pressure on alternate building materials. While steel has been one of the primary materials for structural systems, it is only recently that its use for residential buildings is being explored.

The challenge in turning any design methodology into a practical tool is to find a way to translate the theory into a robust, efficient and accurate computer program that is also easy to use. From a user perspective, the software system must require the user to input minimal amount of information with the rest of the required data being automatically computed with reasonable accuracy and fit. On the other hand, for experienced users, the software system must provide controls that the user can use to guide the design process and results. We will attempt to address some of these issues in this paper.

The paper is divided into four parts. We first list the basic requirements in the design of steel trusses. The next section examines these requirements in greater detail. The paper

[1] Research Assistant (currently with Honeywell Engines & Systems, Phoenix)

concludes with details of two numerical examples that illustrate our design philosophy and a section on possibilities for future research.

## Steel Truss Design Requirements

The designed trusses must not only satisfy safety and serviceability requirements as suggested by design codes but they also need to be easy to manufacture, construct, and design, and they must be economical.

### *AISI-related requirements*

The AISI-LRFD design code (AISI, 1986) was used to compute the corresponding values of the different cross-sections. The details of the code provisions and the relevant calculations are not shown here since they are not the primary focus of the current study. They are however available in a research report [Mobasher and Situ, 1996].

### *Other considerations - manufacturing and cost issues*

When dealing with the problems of practical structural design, the total weight of the structure is usually a good first estimate of the cost. However, there are other considerations that must be met.

1. A truss with a smaller number of joints is usually preferable.
2. Similarly, a design involving the least number of cuts to be made to obtain the different truss members is also preferable.
3. The design should use members with available cross-section. Customization of the cross section (e.g. built-up section) is an expensive option.
4. Crisscrossing members are not allowed simply because of construction constraints.

It should be noted that these requirements usually cannot be easily satisfied by traditional NLP techniques. On the other hand, GAs are easily adapted to handle such requirements.

### *Design Software System*

When developing a robust and user-friendly software for practical design of steel truss, the above stated factors need to be carefully addressed and implemented. This paper focuses on these issues as well as theoretically aspect of numerical optimization techniques to achieve the purpose of automated design of steel trusses.

## GA as a Design Tool

When the truss design problem is posed as a structural optimization problem, the tasks usually involve sizing, shaping and topology optimization. This problem can be stated as follows.

$$\begin{array}{lll} \text{Find :} & \mathbf{x} = \left\lfloor {}^b x_1, \ldots, {}^b x_{NBDV}; {}^i x_1, \ldots, {}^i x_{NIDV}; {}^s x_1, \ldots, {}^s x_{NSDV} \right\rfloor & \\ \text{Minimize :} & f(\mathbf{x}) & \\ \text{Subject to :} & & \\ & g_i(\mathbf{x}) \le 0 & i = 1, \ldots, NEQC \\ & h_j(\mathbf{x}) = 0 & j = 1, \ldots, NINEQC \\ & {}^b x_p \in \{ 0,\ 1 \} & p = 1, \ldots, NBDV \\ & {}^i x_q \in \{ x_q^1, x_q^2, \ldots\ldots\ldots, x_q^{Nq} \} & q = 1, \ldots, NIDV \\ & {}^s x_r^L \le {}^s x_r \le {}^s x_r^U & r = 1, \ldots, NSDV \end{array} \qquad (1)$$

where $\mathbf{x}$ is the design variable vector, $f(\mathbf{x})$ is the objective function, *NEQC* is the number of equality constraints, *NINEQC* is the number of inequality constraints, *NBDV* is the number of binary design variables, *NIDV* is the number of integer design variables, and *NSDV* is the number of real design variables.

Traditional optimization algorithms perform remarkably well with engineering design problems where the objective and constraint functions are well-behaved (usually unimodal functions defined in a continuous design space). The attractiveness of the Genetic Algorithms to solve engineering design problems arises when the design space is disjointed, where the existence of multiple local minima is a distinct possibility and where the computational challenges with traditional optimization techniques are too great (e.g. computation of gradient vectors to be used with gradient-based methods). When sizing, shape and topology design variables are introduced into a structural design problem simultaneously, numerous special situations are created that cannot be handled in the context of traditional design optimization. GAs then become an attractive solution methodology.

Genetic algorithms are based on the principles of natural genetics and originated with the work by John Holland at the University of Michigan in 1975. The simple GA while powerful, is perhaps too general to be efficient and robust for structural design problems. First, function (or, fitness) evaluations are computationally expensive since they involve finite element analysis. Second, the design space is at times, disjointed with multiple local minima, and is a function of boolean, discrete and continuous design variables. In this section, we show how our proposed improvements to the simple GA are implemented.

*Adaptive Penalty Function for Constraints*

GAs were developed to solve unconstrained optimization problems. However, engineering design problems are usually constrained. They are solved by transforming the problem to an unconstrained problem. The transformation is not unique and one possibility is to use the following strategy.

$$\begin{array}{ll}\text{Find} & \mathbf{x} = \left\lfloor {}^{b}x_1,\ldots,{}^{b}x_{NBDV};\ {}^{i}x_1,\ldots,{}^{i}x_{NIDV};\ {}^{s}x_1,\ldots,{}^{s}x_{NSDV} \right\rfloor \\ \text{Minimize} & f(\mathbf{x}) + \sum_i c_i \cdot \max(0,\ g_i) + \sum_j c_j \cdot \left|h_j\right| \end{array} \tag{2}$$

where $c_i$ and $c_j$ are penalty parameters used with inequality and equality constraints. Determining the appropriate penalty weights $c_i$ and $c_j$ is always problematic. We propose an algorithm here where the penalty weight is computed automatically and adjusted in an adaptive manner. First the objective function is modified as follows.

$$f(\mathrm{x}) + c_a \cdot \left( \sum_i \max(0, g_i) + \sum_j \left|h_j\right| \right) \tag{3}$$

The following rules are used to select $c_a$.

(1) If there are feasible designs in the current generation, $c_a$ is set as the minimum $f$ among all feasible designs in the current generation. The rationale is that for the design with minor violations and smaller objective value, the probability of survival is not eliminated. If, on the other hand, the maximum $f$ among all feasible designs is used, infeasible designs will have a smaller probability to survive even if the constraint violations are small.

(2) If there is no feasible design, $c_a$ is set as the $f$ that has the least constraint violation. The motivation idea has the effect of both pushing the design into feasible domain as well as preserving the design with the smallest fitness.

*Improving Crossover Operators Using the Association String*

As discussed by some researchers, the one-point crossover is preferred for continuous domains, and the uniform crossover for discrete domains. However, schema representation still plays a pivotal role in the efficiency of the GA. If one uses a one-point crossover then it is obvious that the ordering of the design variables is an important issue. Since the characteristic of one-point crossover is that the shorter schema has a better chance to survive, if two variables that have less of an interdependency are placed adjacent to each other, or two variables with a strong relationship are placed far away from each other, the crossover operation will make it more difficult for the GA to search the design space efficiently. To implement this strategy, we introduce an additional string called the *association string*. The details of this scheme can be found in a previous publication by Chen and Rajan [2000, 1998, 1997]. Results show that the association string improves the robustness of the solution process.

*Mating Pool Selection*

The selection scheme (for generating the mating pool) together with the penalty function dictate the probability of survival of each string. While it is very important to preserve the diversity in each generation, researchers have also found that sometimes it may be profitable to bias certain schema [Jong, 1975]. However, results from most of the selection rules, like roulette wheel, depend heavily on the mapping of fitness function. In this paper, the tournament selection [Jong, 1975] is used. There are at least two reasons for this choice. First, tournament selection increases the probability of survival of better strings. Second, only the relative fitness values are relevant when comparing two strings. In other words, the selection depends on individual fitness rather than ratio of fitness values. This is attractive since in this research, the fitness value contains the penalty function and does not represent the true objective function.

*Elitist Approach*
The elitist approach was proposed by De Jong [Jong, 1975]. Our research has shown GA with the incorporation of the elitist approach can be more reliable and efficient than the ones without. This approach is used in the current research.

*Repeating Chromosome*
It was found that, during the evolutionary process, the same chromosomes at times are repeatedly generated [Rajan, 1995]. Since the fitness evaluation in structural design involves finite element analysis, a computationally expensive step, all generated chromosome and the associated fitness information are saved in memory. In this way, if a chromosome is repeated, a finite element analysis is not necessary. Saved chromosomes may also be helpful for further processing of the design history.

*Convergence Property, Population Size and Stopping Criteria*
Convergence property of GAs can be proven to be applicable under certain assumptions [Chen and Rajan, 2000]. The result can be used to estimate the population size. Consider a good initial population containing uniformly distributed alleles. By this, it is meant that no chromosome pattern is missed. Each chromosome is represented by $n$ bits with each bit being either 1 or 0. If the distribution of 1's in each bit location is to be uniform, the initial population size should be at least $n$. During the evolution, it is expected that that the chromosome converges to some special pattern with the (0-1) choice decided for $n$ locations.

Assume that the choice of each bit is independent of all the other bits. Since the population size is $n$ in each generation, after every generation from the statistical viewpoint we can expect to learn about at least one bit. Ideally then after $n$ generations, one can expect to learn about all the $n$ bits forming the chromosome. However, since each bit is not independent of the others, more than $n$ generations are perhaps necessary to obtain a good solution. This suggests that the population size and the number of generations should be *at least* n. Our previous work suggests that using population and generation size of $2n$ leads to reasonable results efficiently.

*The Improved GA Optimizer*
As mentioned before selective improvement can be made to obtain a more robust solution methodology for a class of problems. Table 1 shows the proposed improvements.

*Other Considerations: Design Variable Linking*

As shown in Eqn. (1), GAs essentially can handle three types of design variables - discrete or integer, real, and boolean. These design variables capture all the possible structural design parameters. The sizing design variables considered in this dissertation are either cross-sectional dimensions or available cross-section. The former can be described using real design variables since these dimensions can vary continuously. The latter is described in terms of integers (an integer index that points to a row in a table of available cross-sections). The table search is carried out by using a table of ordered available cross-sections with the lower and upper bound candidate cross-sections specified by the user. The shape design variables are the nodal locations. These are real design variables. The topology design variables can be structural parameters such as the presence or absence of members, and presence or absence of fixity conditions at supports. Table 2 shows the linking of the design variables.

**Development of the Software System**

The algorithms mentioned above were implemented in a computer program. The software was developed under MS Windows 95/NT due to its user-friendly graphical user interface and excellent performance per unit cost that can be achieved. Figure 3 shows a snapshot of the graphical user interface for the design system. The software system offers the following features.

(1) The design process is initiated by specifying the geometrical and loading parameters. These include the span, height of the King Post (or, the pitch of the roof), the dead and live loads acting on the top and bottom chords, heel heights and support conditions, the overhangs, and the truss spacing. Finally, the different types of cross-sections to consider for the members are specified. Table 3 shows the typical cost values of the candidate sections, and Table 4 shows the use of the cross section in each member. Figures 1 and 2 show the typical cross-section of the members.

(2) The truss structure is defined by the specification of the panel points, the maximum unbraced length of a bottom chord member or top chord members. Once the panel points are identified, the elements and nodes of the model are defined by connection of all the nodes to adjacent nodes. This creates what is popularly known as the ground structure.

(3) With the truss completely defined in terms of the topology (all the members with their cross-sectional properties and the member end nodal coordinates known), a materially linear, small displacement, small strain finite element analysis is carried out. The structure is assumed to be a planar frame with **rigid** connections. No second-order effects are considered.

(4) Design checks based on the AISI code are carried out on the finite element results.

(5) The genetic algorithm attempts to remove the elements that have a low stress magnitude while the elements with the stress level exceeding the allowable stress are penalized, increased in size and/or repositioned. Step three is repeated again for the newly updated shape and geometry.

(6) At the end of the design process, one would obtain the cross-sections for the top and bottom chord, heels, King Post and the webs. The number of web sections and their location (in terms of the coordinates of the web members) are also determined.

When the implementation includes the improvement of the GAs as mentioned above, the user just needs to specify the design model and no user-adjustment of the design parameters or algorithm is necessary.

## Numerical Examples

Two numerical examples are considered in this section. In the first example, a commonly used flat-bottom truss was designed, constructed and was a full-scale testing was carried out under carefully monitored conditions. The second example is used to test the applicability of the developed methodology for different types of trusses.

*Example 1*

The example considered is a flat bottomed symmetric truss with a span of 6.10 m (20 ft) and 2 panel points on each side. The height of the heels is 15.24 cm (6") with 30.48 cm (1') overhangs, and the height of the ridge is 1.524 m (5'). The loading on the truss include 957.6 N/m$^2$ (20 psf) live load and 478.8 N/m$^2$ (10 psf) dead load on the top chord, and 239.4 N/m$^2$ (5 psf) dead load on the bottom chord. Figure 4 shows the ground structure of this example. The final cost is $39.13. The final connectivity and layout of the structure is shown in Figure 5.

*Example 2*

This example is a flat bottomed symmetric truss with a span of 9.14m (30 ft). The height of the heels is 22.86 cm (9") with 45.7 cm (18") overhangs, and the height of the ridge is 2.29 m (7.5'). The loading on the truss include 957.6 N/m$^2$ (20 psf) live load and 478.8 N/m$^2$ (10 psf) dead load on the top chord, and 5 psf dead load on the bottom chord. There are 5 panel points on each side. Figure 6 shows the ground structure. The final cost is $73.0. The final connectivity and layout of the structure is shown in Figure 7. The problem took about 1900 seconds CPU time on an Intel Pentium 75 machine with Windows 95! Clearly, with today's faster computers, a much reduced time is possible.

## Concluding Remarks

A comprehensive system has been developed to design residential steel roof truss systems. The AISI-LRFD design code is used in the design process. A GA-based design methodology has been developed that uses minimal input to automatically size, shape and configure the truss. The summary of the research accomplishments is as follows.

(1) Development an automated design procedure to design the lowest cost truss. The design procedure includes (a) the planar frame structural analysis carried out to

compute the response of the individual members subjected to the design loads, (b) use of the response values in an AISI-based design checks to ensure the adequacy of the individual members, and (c) the procedure to redesign in order to minimize the cost of the truss.

(2) The validity of the structural analysis is established by comparing the linear and elasto-plastic FEA strain and deflection values against the experimentally obtained values from the full-scale test.

(3) The procedure to redesign the truss in order to obtain the lowest cost truss is validated by comparing the cost of the truss to industry norms. The cost of the designed truss is $51 translating to about $2.50 per linear foot. This is about the best estimated cost as per industry norm. These final designs are obtained with minimal user input and in a reasonable amount of computer time.

A good number of the tasks that a design engineer normally deals with when translating an architectural drawing into code-conforming final design can be automated. This task is certainly worthy of further investigation.

**References**

1. American Institute of Steel Construction, *Load and resistance factor design specification for structural steel building*, 1986.
2. S-Y. Chen and S. D. Rajan, "A Robust Genetic Algorithm for Structural Optimization", Accepted for publication by *Structural Engineering & Mechanics Journal*, Feb 2000.
3. S-Y. Chen and S.D. Rajan, "Improving the Efficiency of Genetic Algorithms for Frame Designs", *Engineering Optimization,* Vol. 30, pp. 281-307, 1998.
4. S-Y. Chen, *Using Genetic Algorithms for the Optimal Design of Structural Systems*, Dissertation for Doctor of Philosophy, Department of Civil Engineering, Arizona State University, December 1997.
5. K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Dissertation, The University of Michigan, 1975.
6. D. E. Grierson and W. H. Lee, Optimal Synthesis of Frameworks Using Standard Sections, *Journal of Structural Mechanics*, Vol. 12, No. 3, pp. 335-370, 1984.
7. B. Mobasher and J. Situ, *ICBO Test Report of the American Studco, Inc., Residential Roof Truss Chord Components*, *Report No. CEE-AS-95-1*, Dept. of Civil Engineering, Arizona State University, 1996.
8. S. D. Rajan, "Sizing, Shape, and Topology Design Optimization of Trusses Using Genetic Algorithm", *ASCE Journal of Structural Engineering*, Vol.121, No. 10, pp.1480-1487, 1995.

Table 1 Differences Between Traditional and Proposed GA

| | **Traditional GA** | **Proposed GA** |
|---|---|---|
| Penalty Function | ad hoc | Automatic |
| Schema | ad hoc | Ordered |
| Cross-over Probability | ad hoc | Adaptive |
| Population/Max Generation Size | ad hoc | Suggested as *2n* |

Table 2 Linking of Design Variables and the Physical Meaning

| Optimization | Physical Meaning | Design Variable Type in GA | Note |
|---|---|---|---|
| Topology | Element Existence | Boolean | |
| Sizing | Cross-sectional selection | Integer | Search inside a given table |
| Shaping | Nodal Coordinates | Real | Varies inside upper and lower bound |

Table 3 Typical Cost Values

| **Item** | **Cost ($)** | **Item** | **Cost ($)** |
|---|---|---|---|
| 3.5 Chord 16 GA | 0.84/ft | 3.5 Chord 18 GA | 0.72/ft |
| 3.5 Chord 20 GA | 0.60/ft | 3.5 Chord 22 GA | 0.468/ft |
| 2.5 Chord 16 GA | 0.73/ft | 2.5 Chord 18 GA | 0.996/ft |
| 2.5 Chord 20 GA | 0.48/ft | 2.5 Chord 22 GA | 0.372/ft |
| 1.5 SQWEB 20 GA | 0.34/ft | 1.5 SQWEB 18 GA | 0.44/ft |
| 1.5 SQWEB 16 GA | 0.68/ft | 1.5 CEWEB 20 GA | 0.34/ft |
| Screw | 0.02 | Labor per screw | 0.08 |
| Cut Cost | 0.35 | | |

Table 4 Usage of Cross-sections

| | Top Chord | Bottom | King Post | Webs | Heel |
|---|---|---|---|---|---|
| 3.5 Chord 16 GA | YES | YES | NO | NO | NO |
| 3.5 Chord 18 GA | YES | YES | NO | NO | NO |
| 3.5 Chord 20 GA | YES | YES | NO | NO | NO |
| 3.5 Chord 22 GA | YES | YES | NO | NO | NO |
| 2.5 Chord 16 GA | YES | YES | NO | NO | NO |
| 2.5 Chord 18 GA | YES | YES | NO | NO | NO |
| 2.5 Chord 20 GA | YES | YES | NO | NO | NO |
| 2.5 Chord 22 GA | YES | YES | NO | NO | NO |
| 1.5 SQWEB 20 GA | NO | NO | NO | NO | NO |
| 1.5 SQWEB 18 GA | NO | NO | YES | YES | YES |
| 1.5 SQWEB 16 GA | NO | NO | YES | YES | YES |
| 1.5 CEWEB 20 GA | NO | NO | NO | YES | NO |

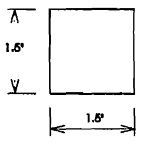

Figure 1 Typical web and heel section (designation: *1.5 WEB 18GA)*

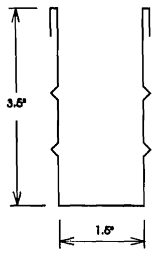

Figure 2 Typical chord section (designation: *3.5 CHORD 20GA)*

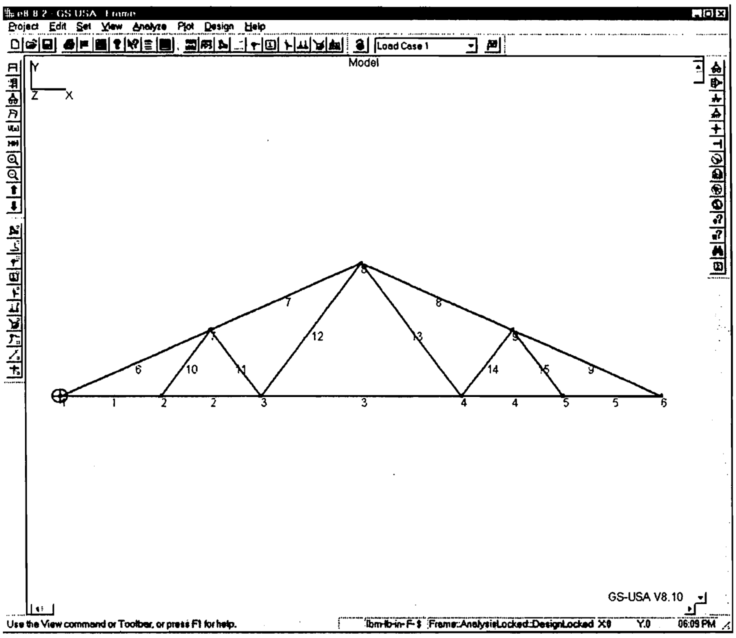
GS USA Frame
Project Edit Set View Analyze Plot Design Help
Load Case 1
Model
Y
Z X
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
GS-USA V8.10
Use the View command or Toolbar, or press F1 for help.
lbm-lb-in-F-$ Frame:AnalysisLocked:DesignLocked X0 Y0 06:09 PM

Figure 3 Graphical User Interface for the Roof Truss Design System

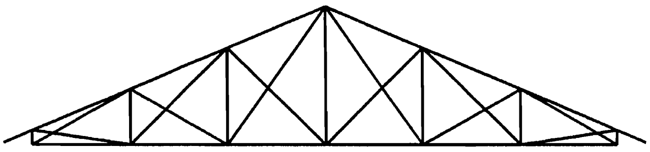

Figure 4 The Ground Structure of Example 1

Figure 5 Final Design of the Example 1 ($39.13)

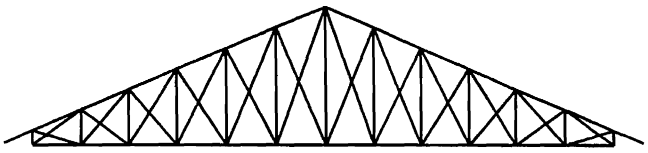
Figure 6 The Ground Structure of Example 2

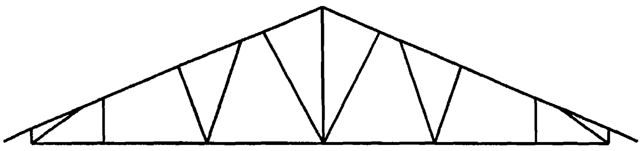
Figure 7 Final Design of the Example 2 ($72.00)