Paper

S-Y. Chen and S.D. Rajan, 1998, “Improving the Efficiency of Genetic Algorithms for Frame Designs”, *Engineering Optimization*, Vol. 30, pp281-307.

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

IMPROVING THE EFFICIENCY OF GENETIC ALGORITHMS FOR FRAME DESIGNS

S-Y.Chen[1] and S. D. Rajan[2]
Department of Civil Engineering
Arizona State University
Tempe, Arizona, USA

ABSTRACT

The focus of this paper is on the development of a design software system that has enough flexibility and capability to search for the most economical steel roof truss design in a reasonable amount of time. This objective is achieved by improving the efficiency and robustness of the genetic algorithm (GA) methodology developed earlier. The effect of schema representation, schema survival, type of crossover, problem definition, the size of the population, and the number of design iterations on the computational expense and the value of the objective function are studied. The research results show that while the final AISI (American Iron and Steel Institute) code-conforming 'best' designs are very close to each other when different starting designs (ground structures) are used, the use of some GA strategies can lead to either highly non-optimal designs or design processes that are computationally expensive. The results also show some other interesting conclusions. The size of the population and the maximum number of design iterations (or, generations) need to be at least the size of the chromosome. The schema representation is perhaps one of the most important factors. Depending on the complexity of the initial design (density of the ground structure) and the size of the chromosome, a newly developed Association String strategy has led to a computationally effective GA process when combined with the elitist, one-point and uniform crossover strategies.

Keywords: Roof truss, genetic algorithm, optimal design, AISI, frame design.

[1] Research Assistant
[2] Professor and Member, ASCE, ASME

INTRODUCTION

While steel has been one of the primary material for structural systems, it is only recently that its use for residential buildings is being explored. The allowable stress design (ASD) method has traditionally been used for the design of steel buildings. More recently, the load and resistance factor design (LRFD) criteria have been developed for both hot-rolled[1] and cold-formed steel structural members[8]. Guidelines for the AISI-LRFD design of cold-formed members is discussed in the latter reference in terms of the yield point, tension members, flexural members (bending strength, lateral buckling strength, shear strength, web crippling strength, combined bending and web crippling), concentrically loaded compression members, and combined axial load and bending. Additional work, especially experimental, has been carried out by Weng and Pekoz[15] to characterize the compression tests of cold-formed steel columns.

Practical design situations have special requirements. In the case of residential steel roofs, the members are selected from a list of available sections and there are constraints on what cross-sections can be used as the chord members, king post, heels and webs. The roof pitch and truss spacing are restricted to builder-specified values, and generally speaking, the outline of the truss has one of only a few options. The labor cost must be factored into the overall cost of the truss. Under these constraints and requirements, traditional design optimization techniques such as nonlinear programming techniques cannot be used to obtain the optimal solution. The design space is nonconvex and disjointed. Genetic algorithms (GA) have been used for solving a variety of structural design problems. Recently, the possibility of making the GA solution a practical design methodology was developed and presented[11,16]. It was shown that sizing, shape and topology optimization can be handled simultaneously in an effective manner. The simple GA strategies used in the design of truss and frame structures are discussed in these publications. The strategies are variations, refinements and improvements over those proposed by several other researchers[5,7,9,10,12,13].

The paper is divided into four sections. First, the design problem formulation is discussed. The second section covers the effect of schema representation, schema survival, type of crossover, problem definition, the size of the population, and the number of design iterations on the computational expense and the value of the objective function. This is followed by a brief discussion of the design software system and the implementation details. In the last section, several numerical experiments are run with typical residential roof systems to illustrate the developed methodology and draw pertinent conclusions.

FORMULATION OF THE DESIGN PROBLEM

The optimal design of structural systems can be classified as sizing optimal design, shape optimal design or topology optimal design problem. The nature of the design variables determines the type of the optimal design problem. Traditionally, the optimal design of structural systems has been formulated as

$$\begin{aligned} &\text{Find} \quad \mathbf{x} \in R^k \quad \text{to} \\ &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad g_i(\mathbf{x}) \le 0 \quad i = 1,2,...,n \\ &\qquad\qquad x_j^L \le x_j \le x_j^U \quad j = 1,2,...,k \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the vector of design variables, $f(\mathbf{x})$ is the objective function, $g_i(\mathbf{x})$ are the performance constraints and x_j^L and x_j^U refer to the lower and upper bounds on the design variables. The objective function can be the weight or a structural response such as maximum displacement. The performance constraints may include concern for axial stress, nodal displacements, local buckling, structural frequency etc. More details on the problem formulation can be found in a recent publication[16].

GENETIC ALGORITHM

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. 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 GA's 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. GA's then become an attractive solution methodology. In the context of genetic algorithms, the roof truss design problem is posed as follows.

$$\begin{aligned} &\text{Find} \ (\mathbf{x} \Rightarrow b_1 b_2 .....b_s) \ \text{to} \\ &\text{minimize} \quad f(\mathbf{x}) + f_{penalty} \end{aligned} \tag{2}$$

where $\mathbf{x} \Rightarrow b_1 b_2 ....b_s$ is used to denote the representation of the design variables $\mathbf{x}$ as a single chromosome or string $b_1 b_2 ....b_s$ of length *s* and the second term is the penalty term resulting from designs that are unacceptable. For a design to be classified as a feasible or the est' design requires that $f_{penalty}$ be zero.

One of the drawbacks of a GA-based approach is the large number of function (or, fitness) evaluations over the evolutionary process. It should also be noted that GA lack a consistent convergence criterion. In our present work, we have attempted to address both these issues.

Design Variables

The design variables used in this study are divided into three types - boolean, integer (or, discrete) and real (or, continuous). *Standard* (or, 1-bit) boolean design variables require a string of length one. Hence they are either 0 or 1. A real design variable is specified as lying between x^L and x^U and of precision *p*. An *integer* design variable is similar to a real design variable except that the precision *p* is 1. In the present research, boolean variables are used to represent the presence or absence of a member. A zero value indicates that a member should not be considered during structural analysis and evaluation of the objective function. Similarly, the support condition along a particular degree of freedom at a joint is also represented by a boolean variable. A zero value is used to indicate that the degree of freedom is unrestrained. The nodal locations (that control the shape of the roof truss) are represented as real, while the member cross-sections are represented as discrete (or, integer) design variables with the member selected from the list of available sections.

Crossover for Sizing Optimization

Crossover is probably the most important operator in GA. Researchers have tried many different crossover strategies. One of the conclusions is that the operation should yield as different a chromosome as possible so as to search the design space more thoroughly[3]. Hence, the uniform crossover is preferred by some researchers[4]. It is also very well recognized that the efficacy of the crossover operator is problem-dependent. The current research has shown that the uniform crossover does not offer a better solution, either in efficiency, stability, or in obtaining the best value of the objective function. This may be due to the fact that while the feasible domain for the entire design space is discrete, the design space for the sizing design variables alone is still continuous and locally smooth. Using uniform crossover with sizing design variables makes the search inefficient. Therefore, for problems involving continuous design variables, one-point crossover is recommended.

Schema Representation

While 1-bit boolean design variables provide a compact manner to represent the 0-1 design variable, there are important considerations about the survival of the design

variable. For example, if bit *q* in the chromosome represents a boolean design variable and its value is zero for the entire population, then the only way for that bit to change to a 1 is through mutation. The implication is that a critical dimension of the design space may not be included in the search process. To overcome this, a 3-bit representation for boolean design variables is used as explained below.

Let the boolean design variable be represented as a 3-bit string, $b_1 b_2 b_3$. If $2 \le \sum b_i \le 3$, the value of the mapped variable is 1, and 0 otherwise. Let us compare two scenarios involving 1-bit and 3-bit representations. Consider the one-bit representation of a variable involving two members of a population. Let the two values be 1 and 1. The probability of obtaining a 0 by taking a one-point crossover of these two members is zero. Now consider the same two members of the population but represented in the 3-bit form as 110 and 101. The probability of obtaining a 0 (i.e. 100) value by taking a one-point crossover is now 1/8. One can see that the probability of producing off-spring of value 0 depends on the number of zeros in the 3-bit strings, which strongly depends on the history of the population. One can propose the use of larger length representations; however this must be weighed against the increase in storage requirements needed by such representations.

Crossover for Boolean Design Variables

Goldberg[6] has shown that proper schema representation allows the GA to develop the process of searching for the best designs in a discrete and non-monotonic space. The design space corresponding to boolean design variables is discrete and nonconvex. To handle this situation, the uniform crossover operator is used with boolean design variables.

Schema Representation: Design Variable Association String

As discussed before, one-point crossover is preferred for a continuous domain, and 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 on the string, then it is obvious that the placement or ordering of the design variables is an important issue. Goldberg[6] has shown that with the one-point crossover scheme, a shorter schema has a better chance of survival. In other words, 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), better designs are less likely to yield from the crossover operation. One strategy to overcome this problem (with longer schema) is to segment the chromosome based on the nature of the design variables. Then depending on the type of design variable, the

appropriate crossover operator can be used. In any case, segmenting the chromosome alone will not increase the efficiency of the search process unless the association between the design variables (or, their interdependencies) is established.

Example 1: Consider an example of a chromosome that represents five design variables (dv) as follows

Table 1 Example 1

0011	0101	0101	1001	0100
dv1	dv2	dv3	dv4	dv5

Assume that the probability of crossover is p and that it is desirable to change design variables 2 and 4 but not 1, 3, and 5. With a one-point crossover taken with the entire chromosome, the probability of achieving the above objective is zero. However, with the same one-point crossover applied to each design variable individually, the probability of achieving the objective is $p^2(1-p)^3$. To implement this strategy, we introduce an additional string called the *association string*. This string is formed for each member of the population with three bits assigned to each design variable. The following procedure describes the implementation.

Let the segments for the *j*th design variable associated with the two chromosomes, selected using the selection criterion, be denoted as ${}^1_j\mathrm{y}$ and ${}^2_j\mathrm{y}$. Implement the following two steps.

Step 1: For each design variable implement the following.

Assume ${}^1_j\mathbf{q} = \sum_{i=1}^{3} {}^1_j\mathbf{q}_i$ is the segment of the *association string* 1 associated with the *j*th design variable. Let ${}^2_j\mathbf{q}$ be the corresponding value for the second string. Let ${}^1f$ and ${}^2f$ be the fitness value for chromosomes one and two respectively. Calculate the crossover parameter

$$\rho = \vartheta \left\{ \mathrm{int}\left[\frac{3-{}^1_j\mathbf{q}}{3} \right] \frac{{}^2f}{{}^1f+{}^2f} + \mathrm{int}\left[\frac{3-{}^2_j\mathbf{q}}{3} \right] \frac{{}^1f}{{}^1f+{}^2f} \right\} \tag{3}$$

where ϑ is a random number between 0 and 1. If $\rho > 0.5$ take the one-point crossover on strings ${}^1_j\mathbf{y}$ and ${}^2_j\mathbf{y}$. Otherwise, skip the crossover operation for the *j*th design variable. The major objective is to reduce the number of disruptive crossovers.

Step 2: Take uniform crossover on the two *association strings* to generate the *association string* for the next generation.

We will illustrate the above procedure with an example.

Example 2: Consider 2 members (designated *pop1* and *pop2*) of a population that have the fitness values given as ${}^1f = 10$ and ${}^2f = 30$, and have been selected for mating. Let the chromosome represent six design variables with the first three as real or integers, and the last three as boolean. Let their current representation be as shown in the table.

Table 2 Example 2

pop1	0011	0101	1001	0	1	0
AS1	000	110	111	000	110	111
pop2	1110	0100	0010	1	0	1
AS2	110	000	100	100	011	110

The procedure for producing offspring from these two members is as follows.

(a) Generate the random numbers ϑ for each AS and use Eqn. (3) to calculate the crossover parameter for each design variable. The results are shown below (using arbitrarily generated values).

Table 3 Example 2

ϑ	0.8	0.2	0.2	0.9	0.7	0.4
ρ	0.6	0.05	0	0.675	0	0

(b) If the crossover parameter is greater than 0.5, then take the one-point crossover for real or integer design variables and exchange the 2 bits for boolean design variables. With the current example, crossover takes place for the first and the fourth design variables only. While mutation can take place, the new chromosomes are shown below without any mutation.

Table 4 Example 2

pop1	**0010**	0101	1001	**1**	1	0
AS1	000	110	111	000	110	111
pop2	**1111**	0100	0010	**0**	0	1

AS2	110	000	100	100	011	110

(c) Now implement step (2) with the association strings (AS). No mutation is allowed in this step. The results are shown below.

Table 5 Example 2

pop1	0010	0101	1001	1	1	0
AS1	010	010	111	100	010	111
pop2	1111	0100	0010	0	0	1
AS2	100	100	100	000	111	110

The purpose of the association string is to enable the GA to use and establish the inter-dependency between the different design variables without any prior knowledge of the problem characteristics. Once the chromosome is segmented based on the nature of the design variable (sizing, shape or topology), the somewhat disruptive nature of the crossover operation is controlled by the use of the Association String. We believe that this is the key in retaining the right genetic material for the subsequent generations in a frame design scenario.

Population Size and the Number of Generations

While assigning values to the GA parameters is problem-dependent, arriving at the recommended values for the size of the population or the number of generations is even more difficult. In this paper, we propose the following guidelines.

Consider a string $\mathbf{x}_1 = \left[b_1 b_2 .... b_s\right]_1$. The string $\mathbf{x}_1$ can be obtained by a linear combination of the basis e_i, that has 1 in the *i*th position and 0 for all the other positions. That is $\mathrm{x}_1 = \sum_{i=1}^{s} a_i \, \mathrm{e}_i$ where a_i are 0 or 1. When crossover is to take place between $\mathbf{x}_1$ and another string $\mathbf{x}_2$ (assuming no mutation), the resulting string is still a combination of the old strings $\mathbf{x}_1$ and $\mathbf{x}_2$. The dimension of the design space due to the chromosomes in the current (new) generation is less than or equal to the previous

generation. As the evolutionary process progresses, this dimensionality decreases as a result of the survival rules in GA.

To span the entire design space, the population size should be 2^s which can be expressed as

$$2^s = C_0^s + C_1^s + C_2^s + \cdots \quad + C_s^s$$

Hence, if one wants to keep the search dimension complete, the population size should at least be equal to the chromosome size s.

IMPLEMENTATION OF THE DESIGN SYSTEM

The roof truss design problem is formulated as follows. The objective function is taken as the cost of the truss as

$$f(\mathbf{x}) = \sum_{j=1}^{ne} c_j L_j + \sum_{k=1}^{nj} d_k + \sum_{l=1}^{nc} e$$

(4)

where *ne* is the number of elements, *nj* is the number of joints, *nc* is the number of cuts made to obtain the truss members, L_j is the length of the element j, c_j is the cost per unit length, d_k is the cost of the connection (a function of the number of screws needed to construct the connection), and e is the cost of making a cut in a cold-rolled specimen so as to obtain a specified length member. Typical cross-sections used in the truss design are shown in Figs. 1 and 2. The typical cost values are shown in Table 6.

The LRFD strength requirements for each member type in seven failure modes were calculated. This data was recorded as linear approximations to a strength envelope curve (in most cases as a function of member length). The forces in the member and its length were compared to the strength envelope data and the distance from the envelope curve was calculated. If the point corresponding to the member forces and length fell inside the curve, no penalty was assessed. If it fell outside the curve, the penalty term was calculated as

$$f_{penalty} = c\,(\frac{v}{v_a} - 1)$$

(5)

where the term in the parenthesis represents the normalized distance to the curve and c is the penalty parameter. For each of the numerical examples discussed in the next section, the following steps are followed. It should be noted that further details are available in a recent publication[16].

(1) The design process is initiated by specifying the span of the truss, the height of the King Post (or, the pitch of the roof), the dead and the live loads acting on the top and bottom chords, the heel heights, the heel support conditions, the overhangs, the truss spacing, the number of panel points, the cost figures (see Table 6) and finally, the different types of cross-sections to consider for the members. A very dense ground structure is automatically constructed as the initial design.

(2) With the truss completely defined, a finite element analysis is carried out. The structure is assumed to be a planar frame with rigid connections.

(3) Using the finite element results, design checks based on the AISI code are carried out. The cost of the truss is computed and the cost penalized if the design is found to be unacceptable.

(4) At the end of the design process the cross-sections for the top chord, bottom chord, heels, King Post and the webs, the number of web sections to use, and the locations of these web members (in terms of the coordinates of the ends of the web members) are known.

NUMERICAL EXAMPLES

The computations were performed on two Intel x86 machines running MS-DOS. Runs pertaining to one set of parameters (or, group) however took place on the same hardware so as to obtain consistent CPU times.

The GA used in the design system is based on tournament selection. The other options include: one-point crossover, two-point crossover, uniform crossover, elitist approach, 1-bit boolean (0-1) design variables, 3-bit boolean (0-1) design variables, and the 3-bit association string. The numerical examples were solved with the probability of crossover as 0.9 and mutation as 0.03. The same random number seed value was used for all the runs.

The example considered exhaustively is a flat bottomed symmetric truss with a span of 6.10m (20 ft) (see Fig. 3). 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.

A total of 156 runs were made. The following nomenclature is used to identify the different runs made. Each run (implementing a strategy) is designated Txy-abc where

x is linked to the number of panel points on the top and bottom chords

y is Set A or B or C

a is 1 if the elitist strategy is used, 0 otherwise

b is the crossover type (0:uniform, 1:one point, 2:two point)

c is the schema representation (0:1 bit boolean dv, 1:3 bit boolean dv,
2: 1 bit boolean dv with 3 bit AS)

The number of panel points dictates the member density in the ground structure, which then affects the chromosome length. In Set A, the population size and the number of generations are less than the chromosome size whereas in Set B they are equal and in Set C they are roughly twice the chromosome size. The results for the three sets are shown in Tables 7, 8 and 9. In each group, two to three designs corresponding to the lowest objective function, best computational time and the least number of function evaluations are identified. Their results are shown in Fig. 3(a)-(w). Based on the results, the following observations and conclusions can be made.

(1) Fig. 3 shows the objective function to vary between \$39 and \$47. It is likely that the best obtained design is very close to the global minimum. The cost value (for all the runs made) varied between \$106.64 (T3B-101) and \$39.13 (T0C-112), a ratio of 2.73. Within a particular group, the ratio of the worst computational time to the best time is 2.33 (T0C-121 versus T0C-112).

(2) It is interesting to note that the four different ground structures lead to final cost values that are very close to each other. The only anomaly is T3B-101 that had the least compute time within the group but the worst cost value overall. The designs in Fig. 2 can be roughly categorized into two groups - one with one panel point per side (5 designs) and the other with two panel points per side (17 designs). In the latter group, the majority of the designs are similar to Fig. 3(c) where the two panel points are spaced approximately equally along the top and bottom chords with the web members slightly inclined or straight.

(3) Increasing the population size and number of generations (compared to the chromosome size) is generally beneficial. More often than not, corresponding Set C and Set B results are superior to those from Set A. This reiterates our belief that the minimum value of the population size and number of generations should be at least the size of the chromosome.

(4) The efficiency of the different strategies are compared in Table 10. Column 2 is computed as follows

$$Efficiency = \frac{\sum_{i=1}^{12} Normalized\ Time * Fitness}{12} \tag{6}$$

where 12 denotes the number of runs made per strategy. Generally speaking, the following comments can be made. The elitist option performs better than the no elitist option. The crossover operator by itself has little effect. Having drawn those conclusions, it should be noted that some combinations fare much worse than others. The 101 (elitist, uniform crossover, and 3-bit boolean) and 111 (elitist, 1 point

crossover and 3-bit boolean) combinations have the worst performance. The solution obtained when the elitist approach is combined with the 3-bit boolean design variable tends to stagnate after a few generations (see (6)).

(5) The 112 combination (elitist, 1 point crossover and 1 bit boolean dv with 3 bit AS) is computed to have the best efficiency. While this strategy found the lowest cost solution in this numerical study, it is a generally more compute efficient method than a strategy that always find the lowest cost.

(6) Figs. 4(a)-(b) shows the plot of Homogenity Index and Fitness Value versus Generation Number. The average Homogenity Index is defined as

$$Homogenity\ Index_{avg} = \frac{\left[\sum_{j=1}^{s} \max[\sum_{i=1}^{p}\left(b_j^i = 0\right), \sum_{i=1}^{p}\left(b_j^i = 1\right)]\right]}{ps} \quad (7)$$

where p is the population size, s is the chromosome length, and b_j^i is the bit at location j for member i of the population. The larger the average value, the more homogenous the population. The value for a completely heterogeneous population is 0.5. The run T0B-101 has one of the worst cost while T0C-112 has the best solution. In both cases, it is interesting to note that the population is diverse initially and becomes homogenous as the generations progress. The relationship between the (lowest) cost and the homogenity index is nebulous from the two plots. However, it should be noted that with the use of the elitist strategy, it is possible for the results to stagnate if the best chromosome lacks the right genetic material that is found in the best solution. One way of jump starting the solution would be to increase the probability of mutation once the population becomes too homogenous.

This index along with other criterion such as the change in the fitness value over a predetermined number of generations can be used as an effective termination criteria.

In order to validate the conclusions drawn here, another truss was designed. The new truss has a span of 9.14 m (30'). The height of the heels is 22.86 cm (9") with 45.72 cm (18") overhangs, and the height of the ridge is 2.13m (7'). The loading on the truss include 766.1 N/m^2 (16 psf) live load and 670 N/m^2 (14 psf) dead load on the top chord, and 478.8 N/m^2 (10 psf) dead load on the bottom chord. Four operators were selected based on their performance with the earlier problem. They are the 112, 110, 100 and 120 operators. The new results are shown in Table 11 and Figs. 5(a)-(p). Once again, we summarize our findings.

(1) Computational Effort: Except for operator 100, the other operators have similar performance.
(2) Reliability and Stability: Operators 112, 100 and 120 have similar average cost values. However, operator 112 has the best result. Moreover, the difference between the worst and the best normalized cost values for operators 110 and 120 are 20% and 14% respectively, while for operator 112 it is only 3%. This is significant because ideally, a GA methodology should work well irrespective of the quality of the initial population.

CONCLUDING REMARKS

The use of cold-formed structural steel members in residential buildings is investigated. Experimental techniques were used to evaluate the properties of individual members. The design requirements and guidelines are incorporated in an automated design software system. The system uses a genetic algorithm to search the design space for the most economical design that also meets the performance requirements. The result shows that, in general, only the options that use elitist, one-point crossover with *association string* performed efficiently and generated acceptable final cost, while other combinations of options usually work well on special problems only. In some cases, the savings in compute cost can be as high as 200%. The 3-bit *association string* enhances the probability that different patterns will survive through the crossover operation not through mutation alone. The methodology is efficient, reliable and stable. Some guidelines for computing the size of the population, the number of generations, and termination criteria are also presented.

ACKNOWLDGEMENTS

The assistance provided by American Studco, Inc., Phoenix, AZ during the course of the research is greatly appreciated.

REFERENCES

1. American Institute of Steel Construction (1986) *Load and resistance factor design specification for structural steel building*.

2. Budiman, J. and Rajan, S.D. (1993) Shape optimal design methodology - The hybrid natural approach, *Proc. 34th AIAA/ASCE/ASME/AHS SDM Conference*, San Diego, CA, pp. 544-554.

3. Crossley, W.A. (1995) Using genetic algorithms as an automated methodology for conceptual design of rotorcrafts, *Ph.D. Thesis*, Department of Mechanical Engineering, Arizona State University, Tempe, AZ.

4. Eshelman, L.J. and Schaffer, J.D. (1991) Preventing premature convergence in genetic algorithms by preventing incest, *Proc. Fourth Intl. Conference on Genetic Algorithms, University of California*, San Diego, Morgan Kaufmann Publishers, San Mateo, CA, pp. 115-122

5. Goldberg, D.E. and Samtani, M.P. (1986) Engineering optimization via genetic algorithms, *Proc. 9th Conf. Electronic Computation*, ASCE, New York, NY, pp. 471-482.

6. Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley.

7. Grierson, D.E. and Pak, W.H. (1993) Optimal sizing, geometrical and topological design using a genetic algorithm, *Structural Optimization*, 6, pp.151-159.

8. Hsiao, L-E, Yu, W-W and Galambos, V. (1990) AISI LRFD method for cold-formed steel structural members, *ASCE J of Structural Engineering*, 116, pp.500-517.

9. Lin, C.Y. and Hajela, P. (1993), Genetic search strategies in large scale optimization, *Proc. 34th AIAA/ASCE/ASME/AHS SDM Conference*, San Diego, CA, pp. 2437-2447.

10. Jenkins, W.M. (1991) Towards structural optimization via the genetic algorithm, *Computer and Structures*, 40, pp. 1321-1327.

11. Rajan, S. D. (1995) Sizing, shape and topology design optimization of trusses using a genetic algorithm, *ASCE J Structural Engineering,* 121-10, pp. 1480-1487.

12. Rajeev, S. and Krishnamoorthy, C.S. (1992) Discrete optimization of structures using genetic algorithms, *ASCE J of Structural Engineering*, 118, pp. 1233-1250.

13. Sakamoto, J. and Oda, J. (1993), A technique of optimal layout design for truss structures using genetic algorithm, *Proc. 35th AIAA/ASCE/ASME/AHS SDM Conference*, San Diego, CA, pp. 2402-2408.

14. Salmon, C. and Johnson, J. (1990) *Steel Structures: Design and Behavior*. Harper Collins Publishers, New York.

15. Weng, C.C. and Pekoz, T. (1990) Compression tests of cold-formed steel columns, *ASCE J of Structural Engineering*, 116, pp.1230-1246.

16. Wright, D., Situ, J., Mobasher, B. and Rajan, S.D. (1995) Development and implementation of an automated design system for steel roof trusses*, Proc. Research Transformed Into Practice: Implementation of NSF Research*, Eds. Colville and Amde, ASCE Press, Washington, D.C., 245-256.

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Table 6 Typical Cost Values

Item	Unit Cost ($)
3.5 Chord 16 GA	0.84
3.5 Chord 20 GA	0.60
2.5 Chord 16 GA	0.73
1.5 SQWEB 18 GA	0.44
1.5 CEWEB 20 GA	0.34
Screw	0.02
Labor per screw	0.08
Cut Cost	0.35

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

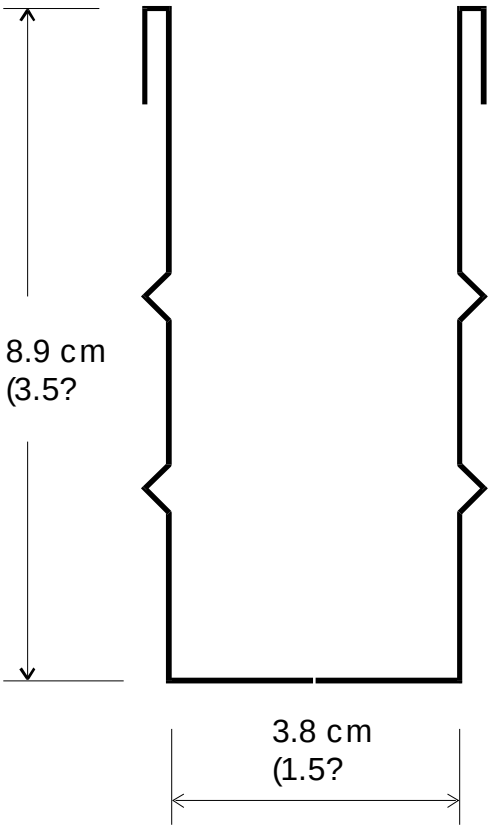


Fig. 1 Typical chord section

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

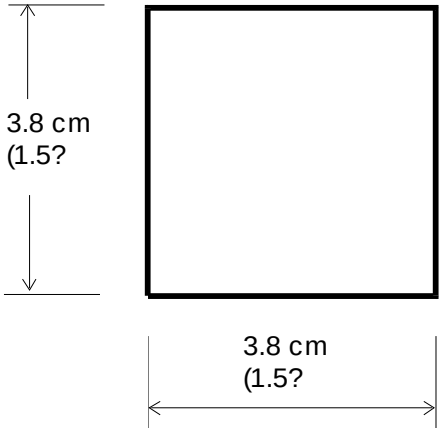
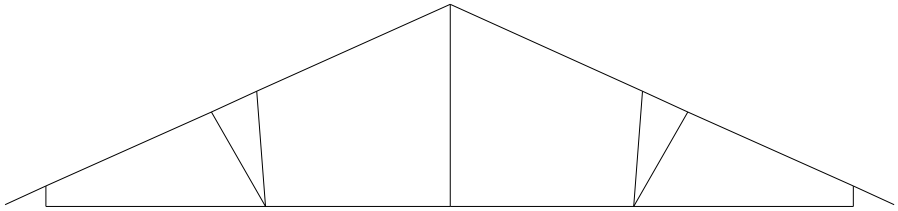
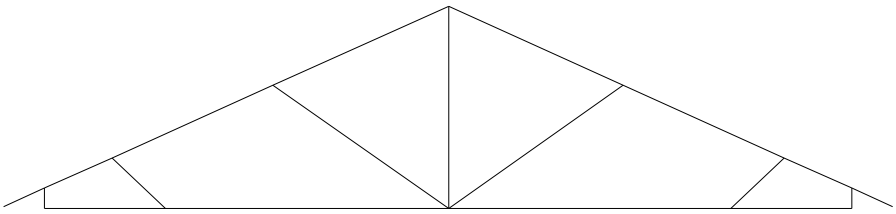


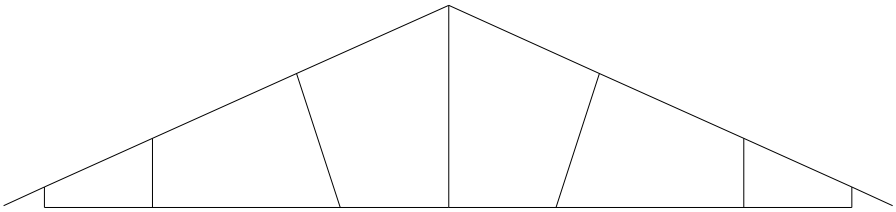
Fig. 2 Typical web and heel section



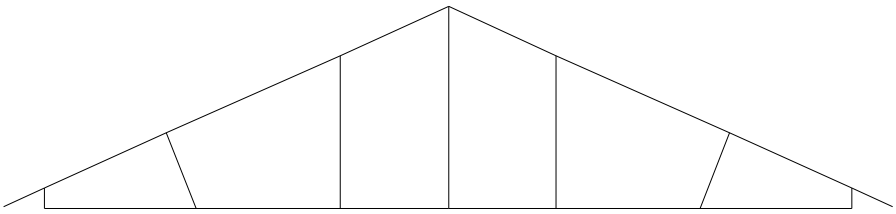
(a) T0A-110 ($41.10)



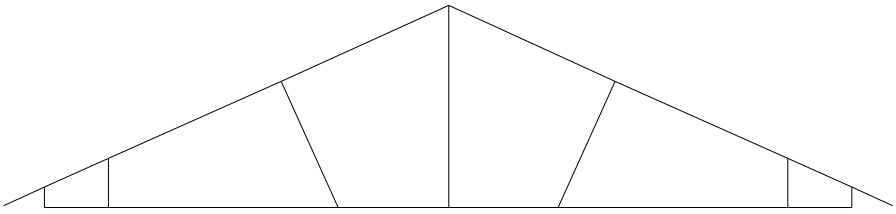
(b) T0A-112 ($42.63)



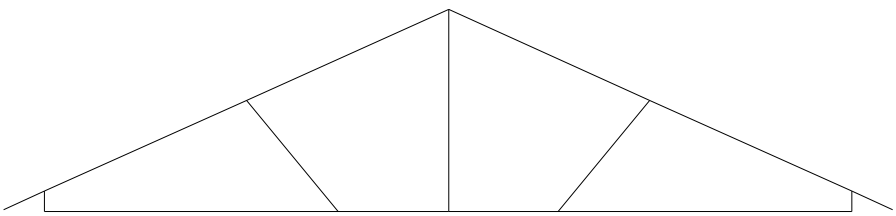
(c) T1A-110 ($40.85)



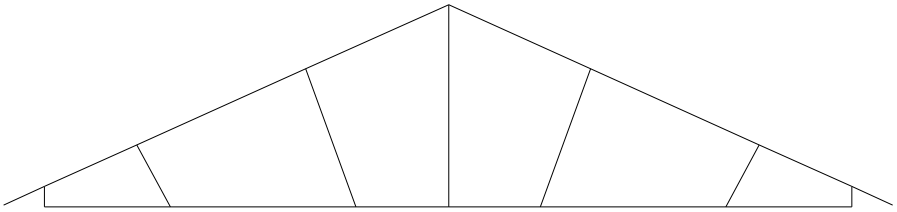
(d) T1A-112 ($44.82)



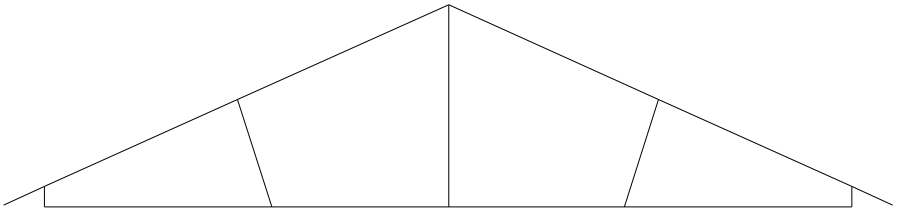
(e) T2A-110 ($40.48)



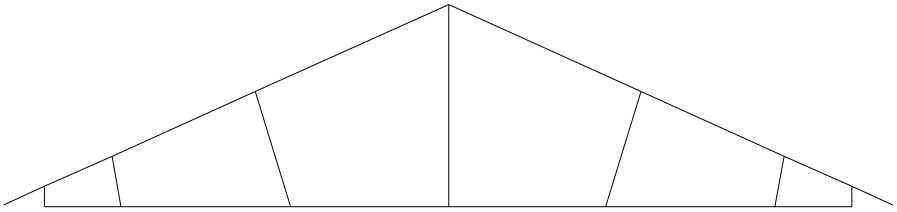
(f) T2A-111 ($47.44)



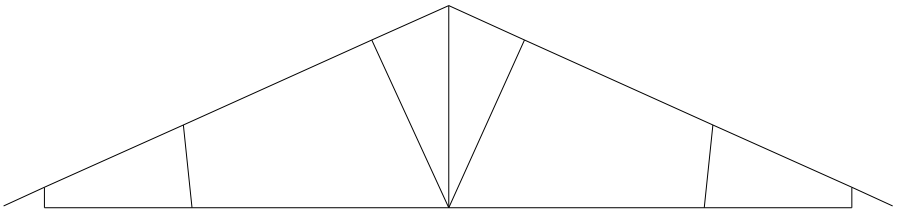
(g) T3A-110 ($40.97)



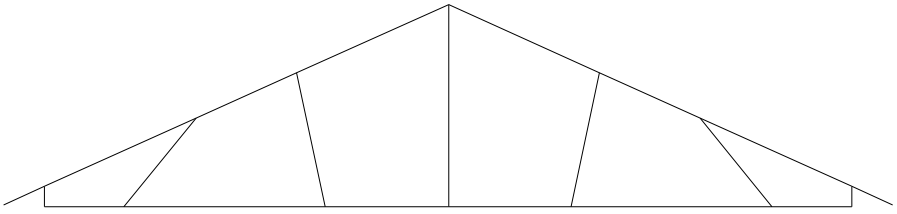
(h) T3A-121 ($42.23)



(i) T0B-100 ($40.21)

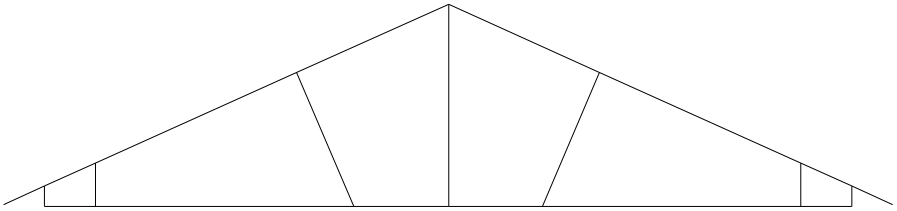


(j) T0B-112 ($41.82)

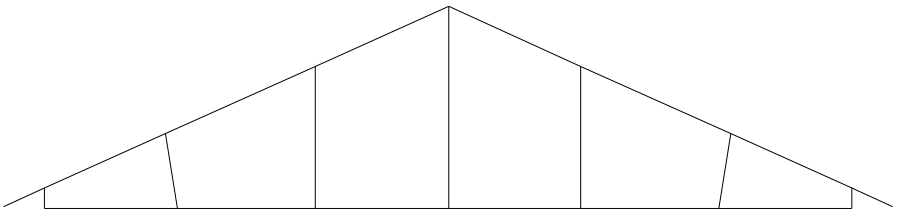


(k) T1B-011 ($41.82)

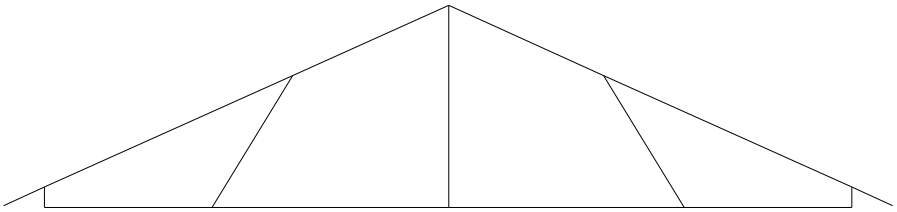
Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com



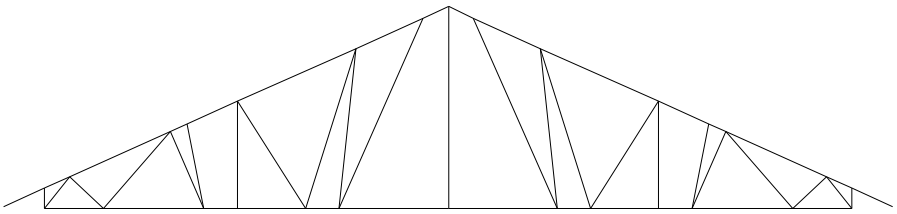
(l) T1B-100 ($40.73)



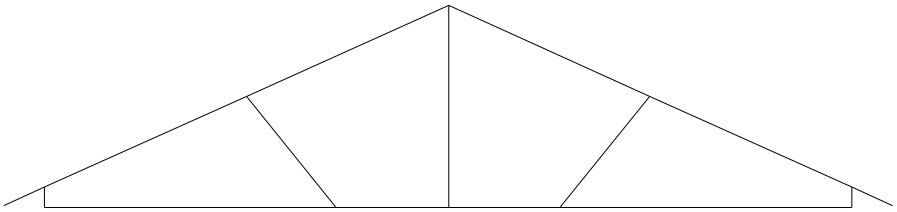
(m) T2B-100 ($40.99)



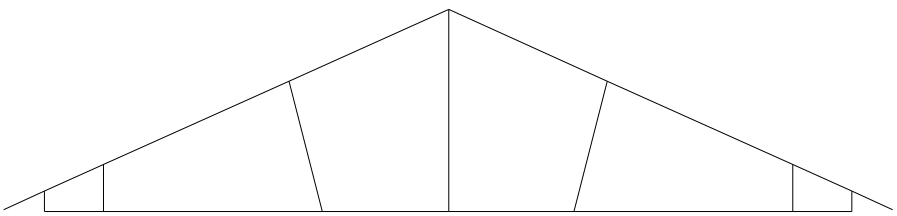
(n) T2B-021 ($46.54)



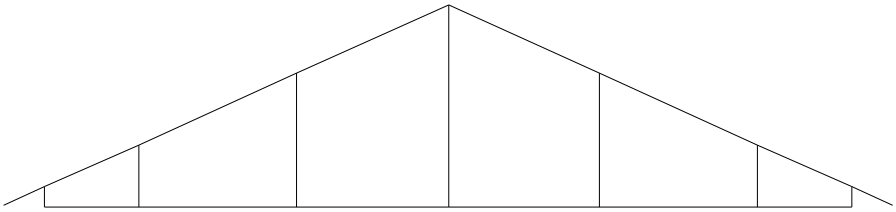
(o) T3B-101 ($106.64)



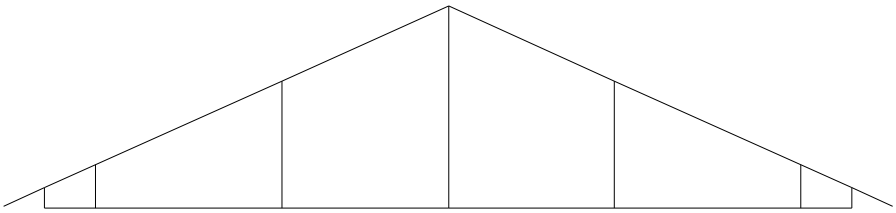
(p) T0C-112 ($39.13)



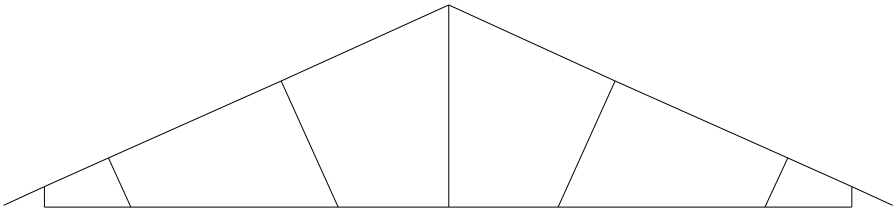
(q) T3B-110 ($40.37)



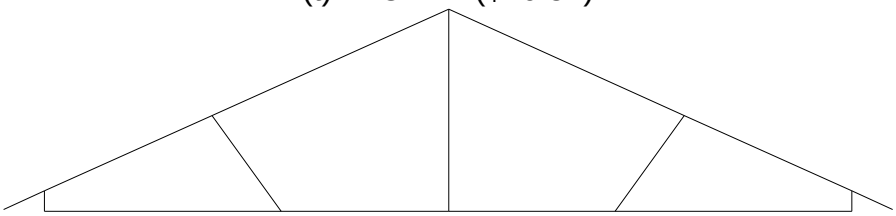
(r) T1C-010 ($40.62)



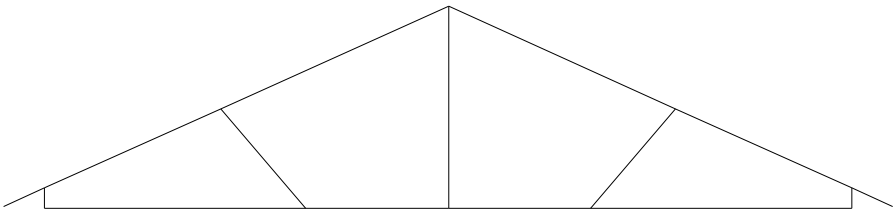
(s) T1C-112 ($42.59)



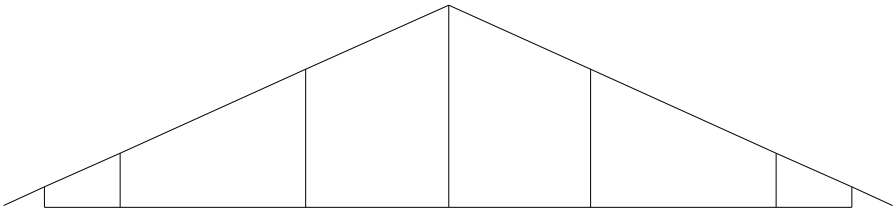
(t) T2C-112 ($40.57)



(u) T2C-121 ($45.15)



(v) T3C-001 ($45.97)



(w) T3C-112 ($40.54)

Fig. 3 Best designs for the 20’ span truss

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

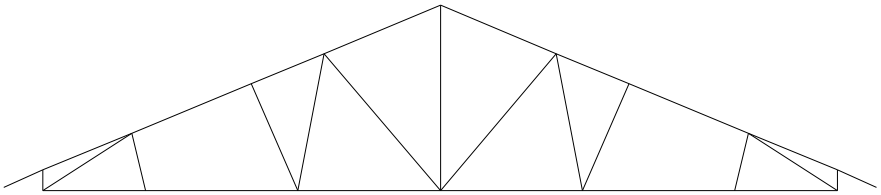
(a)

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

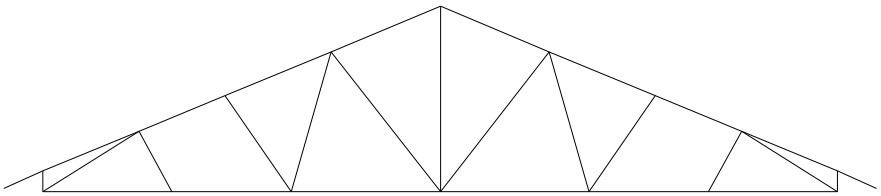
(b)

Fig. 4 Objective Function and Homogenity Index Versus Generation Number (a) T0B-101 (b) T0C-112

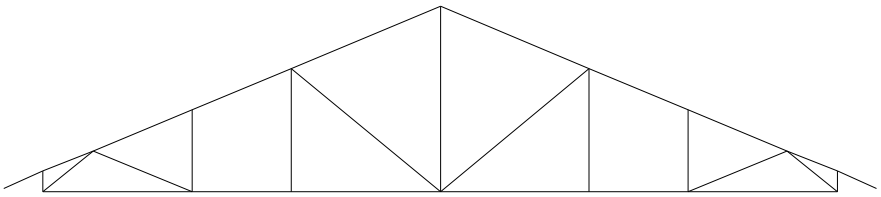
Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com



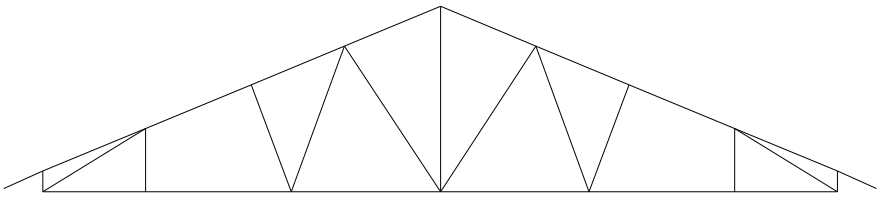
(a) T0H30-112 ($73.5)



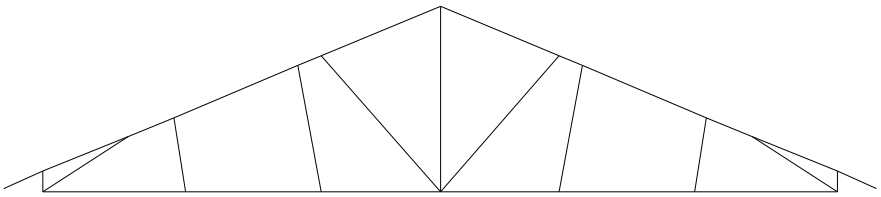
(b) T0H30-110 ($74.0)



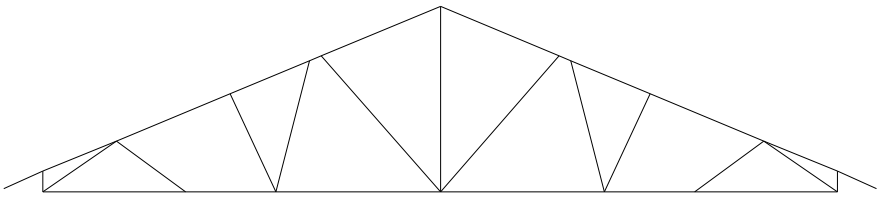
(c) T0H30-100 ($78.1)



(d) T0H30-120 ($74.2)

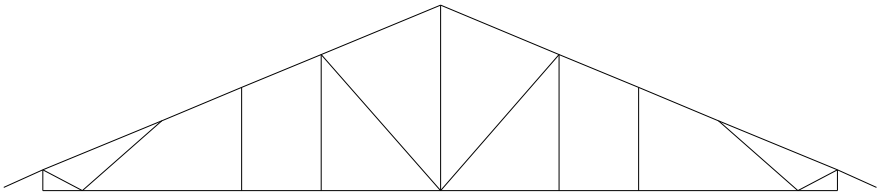


(e) T1H30-112 ($72.9)

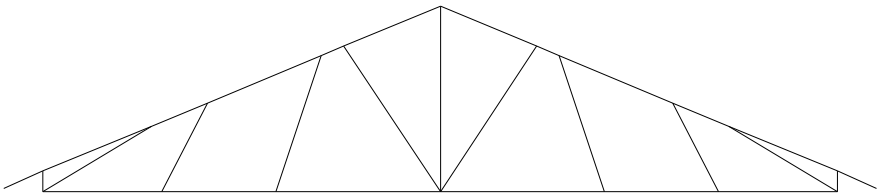


(f) T1H30-110 ($73.5)

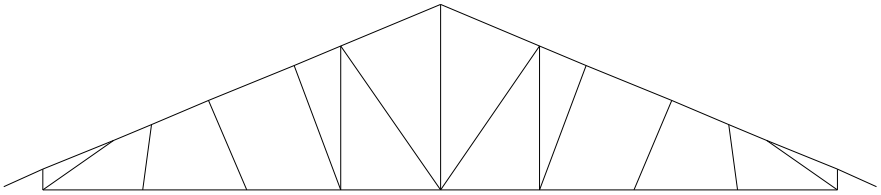
Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com



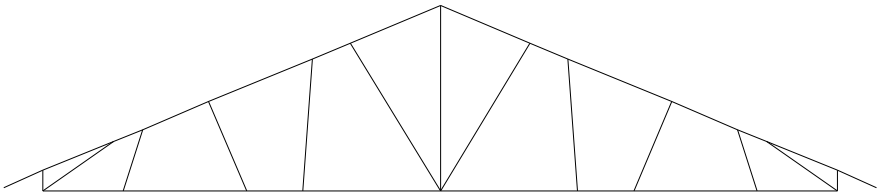
(g) T1H30-100 ($85.6)



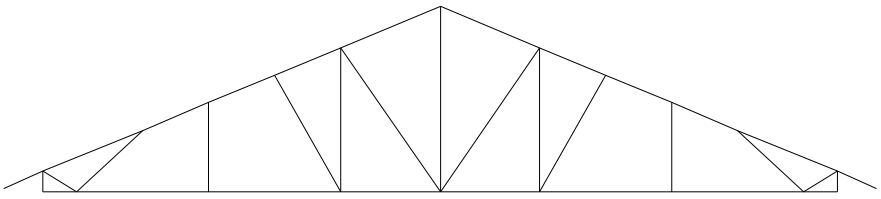
(h) T1H30-120 (74.4)



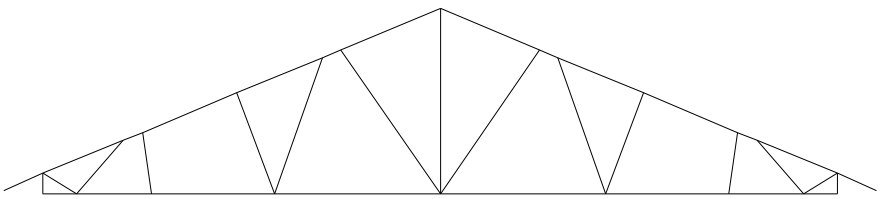
(i) T2H30-112 ($79.1)



(j) T2H30-110 ($80.9)

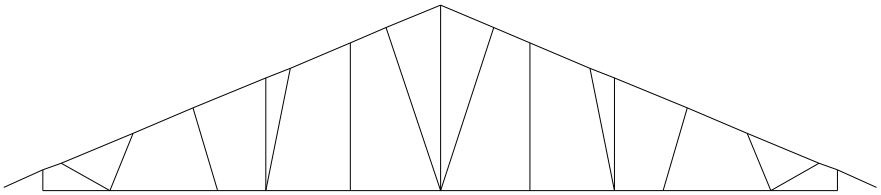


(k) T2H30-100 ($78.3)

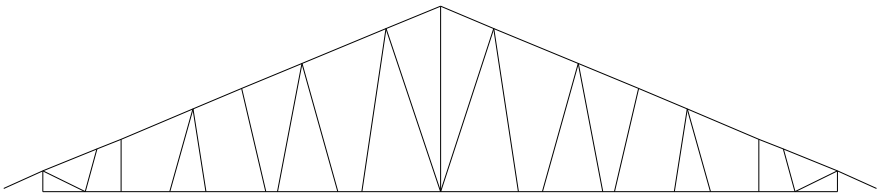


(l) T2H30-120 ($76.5)

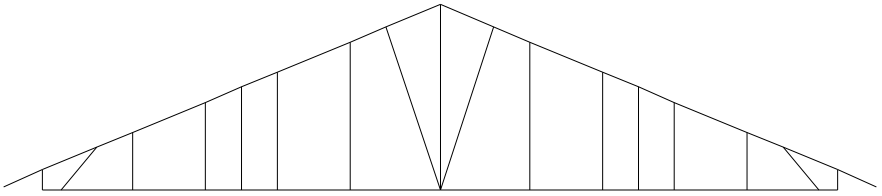
Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com



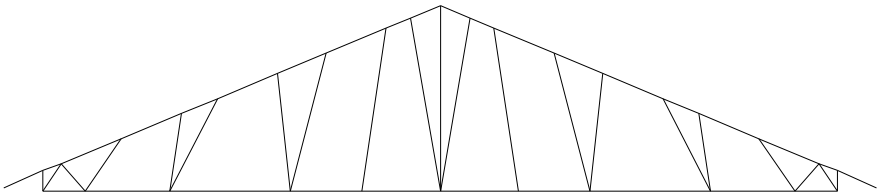
(m) T3H30-112 ($89.0)



(n) T3H30-110 ($108.0)



(o) T3H30-100 ($105.0)



(p) T3H30-120 ($102.0)

Fig. 5 Best designs for the 30’ span truss

Table 7 Set A Results

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Table 8 Set B Results

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Table 9 Set C Results

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Table 10 Comparison of Operator Efficiency

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

Table 11 Results from the 30’ Truss Example

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com

List of Figures

Best designs for the 30’ span truss

Shen-Yeh Chen, Ph.D.
S-Y.Chen@FEA-Optimization.com