Transactions on the Built Environment vol 13, © 1995 WIT Press, www.witpress.com, ISSN 1743-3509

# Multiple-objective, shape optimal design via genetic optimization

E. Sandgren
*TRW Steering and Suspension Systems, PO Box 8008, Sterling Heights, MI 48311-8008, USA*

## Abstract

A multiple objective, goal programming formulation is coupled to a hybrid genetic optimization - nonlinear programming algorithm and applied to the design of structures. The combination provides a unique design environment which can generate solutions not attainable by a traditional approach to structural optimization. Crossectional, geometric and topological change are considered in the formulation. Design goals include weight, cost and robust character. Both hard and soft constraints are included. The genetic optimizer controls the topology of the structure, while a gradient based, nonlinear programming method refines the local geometry and crossectional properties. A specific example involving a ten bar truss is presented which highlights the benefit of the approach over previously reported results.

## 1 Introduction

Every "good" design is in some sense an optimal design. This statement, while being somewhat obvious, is very important, as it suggests that design optimization may well represent a unifying methodology for computer aided design. The difficulty is in finding a way to implement the process at an early enough stage in the design process where significant benefit can be realized. Work in the area of simultaneous engineering has documented the fact that the majority of a product's cost is established in the initial design stage. This means that any analysis or optimization tool that simply hones an existing design concept will have limited impact on the final result. The need is to move from design analysis and refinement into the arena of design synthesis.

From a structural design point of view, design synthesis requires more than geometric and crossectional change. Supporting conceptual design necessitates the consideration of topology. Traditional approaches to topological modification have yet to yield much in the way of useful tools for "real" problems. Design tradeoff analysis must also be dealt with which involves consideration of many separate design criteria. Weight, safety margins for stress

and buckling, natural frequencies, cost and ease of assembly must all be considered. The treatment of design criteria as hard constraints in a nonlinear programming formulation is very restrictive. A multi-objective formulation is a much more general approach. The issue of computational time requirements is and will remain an issue. Any approach built on analysis performed via the finite element technique will consume significant computational resources. This fact places a heavy emphasis on efficiency.

The approach taken herein, to provide a means of intelligent decision support for structural design, involves a hybrid combination of genetic and nonlinear programming methods with a multi-objective design optimization formulation. The genetic algorithm controls topological issues, the gradient based nonlinear programming method performs efficient design refinement, and the goal programming formulation accounts for multiple design objectives. Each of these three elements will be reviewed separately, followed by a description of the integration for structural design synthesis and finally by the application to a simple structural system composed of truss elements. The solution technique is not limited to structures composed of truss and beam elements but the topological issues are much easier to describe at this level.

## 2 Genetic Optimization

Genetic optimization parallels the process of the natural evolution of species. In nature, the fittest of the population survive to produce offspring which embody the important traits of the parents. Over a period of time, the species either adapt to better survive in the surrounding environment or perish. Put very simply, the progression is dominated by the concept of the survival of the fittest. The conceptualization of this process may be used to create a fundamental design paradigm which can be implemented to perform computer aided structural design. The progression of designs created by such a process actually demonstrates some rudimentary form of intelligent behavior. This means that the process is more than just design refinement and therefore is applicable to the initial design stage where the most freedom as well as benefit is available.

A design representation to a genetic algorithm is an encoded string of information which is analogous to a chromosome in a living organism. Each position or gene in the string represents a specific design characteristic such as the number of structural elements (topology) or the position of connecting nodes (geometry). The collection of all possible gene states represents the number of possible designs in the population. Several key distinctions separate a genetic algorithm from a conventional nonlinear programming approach. Among these distinctions are:

i) The genetic algorithm operates by manipulating the coding of the set of gene values composing the chromosome.

ii) A population of designs is considered rather than a single design point.

iii) The rules which govern the transition from one set of designs to the next are probabilistic rather than deterministic.

iv) The algorithm operates in a discrete instead of a continuous design space.

These differences may not seem that dramatic, but they produce an algorithm which is more globally oriented in its search and is not trapped by local minima.

The overall suitability of a chromosome, or the performance of a specific design, is termed its fitness. The property of fitness may be related to any function or functions of the design and is used to determine the probability of a particular design chromosome becoming a parent for the next generation of designs. The chromosomes possessing the greatest fitness have the highest probability of becoming parents but even the least fit chromosome has a finite chance of being selected. The chromosome strings are combined using various genetic operators in order to produce the next generation of designs. The process is continually repeated with the expectation that both the average fitness of the population as well as the maximum fitness contained in the population will improve.

In order to initiate the algorithm, an initial population is generated randomly and the rules which specify how parents are selected and combined to form offspring must be defined. Special operators such as mutation are introduced in order to guard against the loss of important design information which is particularly significant in the structural design application where the population size is fairly small due to the computational time required to evaluate each individual design. The search requires no gradient information and produces a number of design options rather than a single design. It relies on the randomness present in natural selection, but quickly exploits information gathered in order to produce an efficient design procedure. Additional speed may be accessed due to the parallel nature of the algorithm. Details of genetic algorithms are described by Goldburg [1] and Davis [2].

## 3 Goal Programming

A goal programming formulation allows any number of design objectives to be considered, each with a relative priority and/or weighting factor. The solution process attempts to satisfy as many of the objectives or goals as possible, starting with the highest priority. This approach removes much of the difficulty of problem formulation and is more closely aligned with a manual design process. It handles both hard (must be exactly satisfied) and soft (must be approximately satisfied) constraints and fits in well under the umbrella of genetic optimization as the driver for the solution process. The application of goal programming to structural design has been demonstrated by several investigators including Shupe and Mistree [3], El Sayed [4] and Sandgren [5].

The formulation of a goal programming problem contains no exact counterpart to the objective function in a nonlinear programming formulation. Design

constraints may be included as goals or as hard constraints. The solution process revolves around the minimization of the positive and/or negative deviations from the specified goals. The goals themselves are formulated as goal constraints and may take the form of inequality constraints (either $\geq 0$ or $\leq 0$) or equality constraints. These forms are listed below:

$$G_j(x) + d_i^- = b_i \quad (1)$$

$$G_j(x) - d_i^+ = b_i \quad (2)$$

and

$$G_j(x) + d_i^- - d_i^+ = b_i \quad (3)$$

The $x$ vector contains the design variables including crossectonal, geometrical and topological effects and $G_j$ are nonlinear functions relating the design variables to the desired goal values $b_i$. The $d_i^-$ and $d_i^+$ are the under and over achievement deviational variables which may be thought of as a form of slack variables. With this analogy, the first two goal constraint forms are seen to represent less than or equal to and greater than or equal to specifications, respectively. The third form is an equality specification. Only the deviational variables are allowed to appear in the objective function.

The objective function may, like the goal specifications, take on a number of different forms. The most general form allows weighting factors to be assigned within individually specified priority levels. This form can be expressed as:

$$\text{minimize } f(x) = \sum_{i=1}^{I} W_{kj} P_k \{d_i^- + d_i^+\} \text{ for } k = 1,2,\ldots,K;\ j = 1,2,\ldots,J \quad (4)$$

Here, for a priority level k, a total of j individual weights ($W_{kj}$) may be assigned within that priority. The solution process compares solutions by evaluating goals by priority level assignment which is well suited to the genetic algorithm which uncovers local solutions throughout the design space which have fundamentally different relative goal satisfaction.

## 4 Integration

The solution process is summarized in the flowchart presented in Figure 1. The process begins with the user defining spatial regions where the structure can exist, surfaces where ground points can be located and the minimum and maximum number of nodes and elements to be considered. The loads are specified as well as material properties and design goals or objectives are defined and prioritized. This input is all that is required in order to initiate the process. No initial design configuration must be specified. The second step is to create

the design encoding for the genetic optimizer which drives all of the topological considerations. Many different approaches can be taken in the definition of an encoding for structural design. The chromosome for an individual design must, however, contain the following design information:

1) The number of truss elements contained in the design
2) The number of ground attachment nodes (at least 2 are required)
3) The number of free nodes contained in the design
4) Initial location of each node point (within allowable region for free nodes and along allowable surface for ground nodes)
5) Initial estimates of crossectional areas for each truss member
6) Beginning and ending node for each truss element

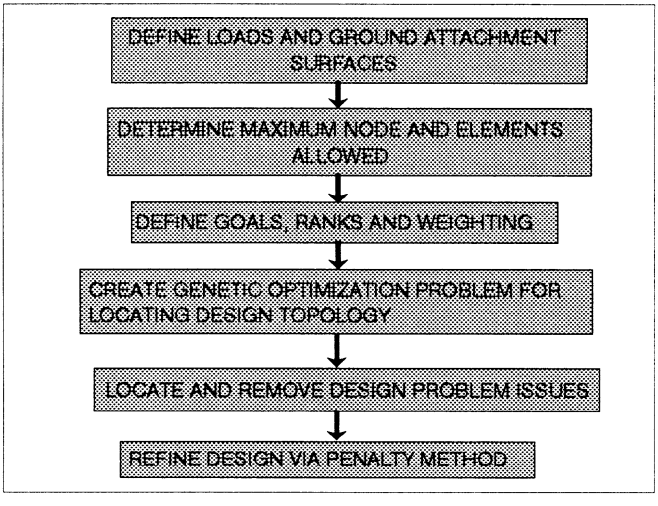


Figure 1 Flowchart for Hybrid Design Optimization Procedure

The specification of these design details defines the topology and provides an initial starting point for the geometric and crossectional optimization which follows. Before this refinement can take place, however, it is necessary to deal with the very real possibility that the structure defined by the chromosomal representation is not a structure at all. Truss elements may be unattached, ground points may not have a connected path to the load(s) and the number of truss elements may define a mechanism with free degrees of freedom instead of a rigid structure. To deal with these issues, a define refinement routine is inserted which adds dummy members to remedy all of the above situations. These dummy elements have a very small but finite crossectional area and keep displacements in unconnected members within finite limits. The stresses in these members are not included in the constraint or goal set.

Once the refinement routine has been executed, a penalty based nonlinear programming algorithm is used to refine the geometry (node locations) and crossectional areas. Each design chromosome is converted to a finite element input file and a finite element analysis is executed. This, along with information concerning weight and cost is used to generate the objective function in the goal formulation. A gradient based penalty function with loose convergence criteria was found to be more efficient in design refinement than allowing the genetic optimizer to generate the final geometry and crossectional areas. This is due in a large part to the discrete nature of the design encoding in the genetic solution. This process is repeated for each design defined by the genetic algorithm. An initial population of designs is produced randomly and these designs are refined in future generations. The process runs for a set number of generations and a selection of the best designs are retained for evaluation and post processing. The entire process is probably best understood by considering an example.

## 5 Example

Many examples of optimal truss design occur in the published literature. Several have appeared a number of times and have become defacto benchmarks for proving out new solution methodologies. One such structure is the ten bar truss pictured in Figure 2. The original structure has node points separated by 360 units in both the horizontal or vertical direction. Two side ground points are employed and two separate loads of 10,000 units must be supported. A maximum allowable stress of 25,000 is allowed either in tension or compression and both load application nodes must not undergo more than a 2 unit displacement. The modulus of elasticity for the truss members is 10.0E+06. The generally accepted solution for the minimum weight subject to the stress and deflection constraints is in the vicinity of 5060.

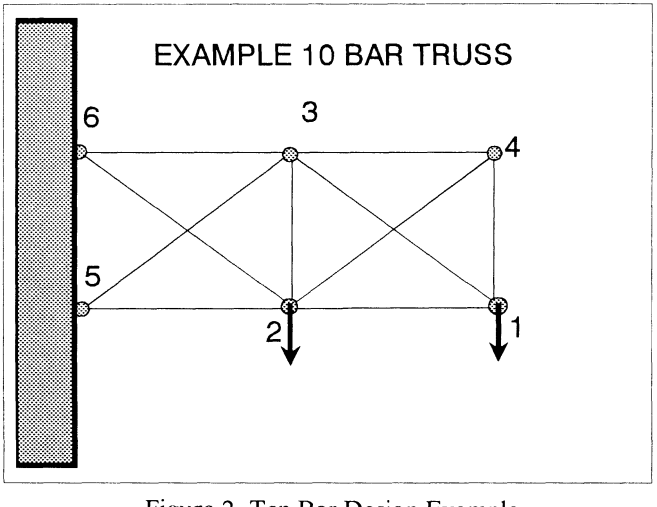


Figure 2 Ten Bar Design Example

In order to apply the new approach, a less restrictive view is taken of the problem. The load application points remain fixed in space but the ground attachment points are allowed to move along the vertical wall. The number and location of node points are allowed to vary as well as the number of truss members employed. The goals for this example are specified in priority order as follows:

1) Maximum stress in any member to be less than 25000
2) Maximum deflection of load supporting node to be less than 2
3) Weight should be as small as possible (equivalent to goal of zero)

Additional goals concerning common sized members or insensitivity to load magnitude and direction could be included as presented by Sandgren [5], but were not considered in order to keep the solution comparable to that generated by a conventional nonlinear programming approach.

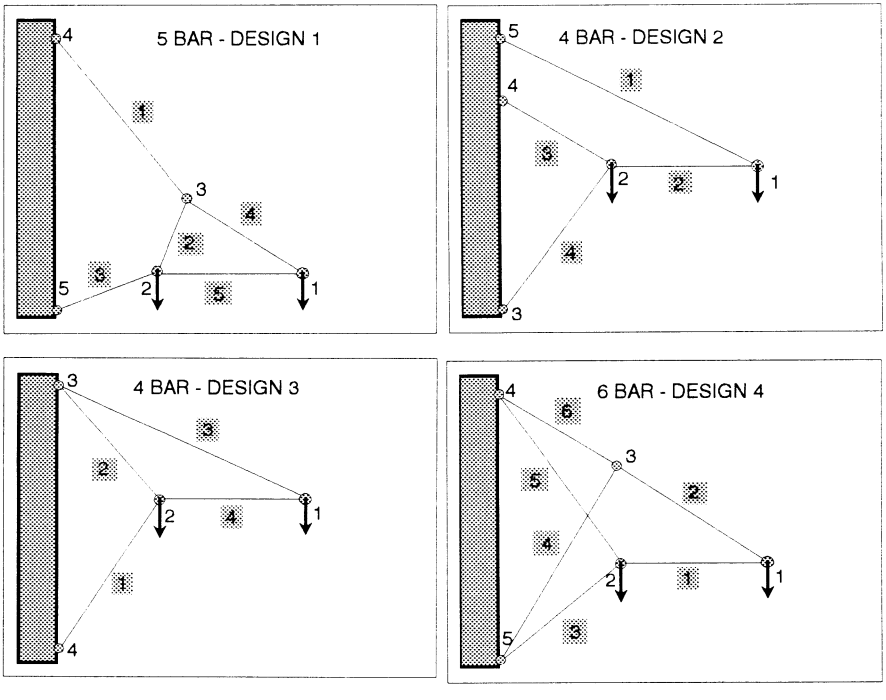


Figure 3 Selected Solutions for the Design Example

The problem was executed with an initial population size of one hundred designs. After fifty generations, several interesting designs were located. Among these designs are the four pictured in Figure 3 with critical dimensions as specified in Table 1. Note the variety in number of truss elements and ground points as well as the satisfaction of the various goals. Design 1 is actually a mechanism which locks at the design position. The most important point, however, is that the stress and deflection goals can be met with a fraction of the weight required for

the ten bar solution. Of course, considerations such as buckling and natural frequency may be important, and they can be included as additional goals.

| DESIGN | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| ELEMENTS | 5 | 4 | 4 | 6 |
| AREA 1 | 15.09 | 12.31 | 8.95 | 9.89 |
| AREA 2 | 19.20 | 9.22 | 14.38 | 13.30 |
| AREA 3 | 11.96 | 1.33 | 11.75 | 10.59 |
| AREA 4 | 29.29 | 9.28 | 8.36 | 3.84 |
| AREA 5 | 12.41 | ------- | ------- | 2.01 |
| AREA 6 | ------- | ------- | ------- | 17.70 |
| P3X | 429.61 | 0.0 | 0.0 | 367.38 |
| P3Y | 172.19 | -189.45 | 654.90 | 317.34 |
| P4X | 0.0 | 0.0 | 0.0 | 0.0 |
| P4Y | 654.07 | 471.99 | -189.84 | 510.24 |
| P5X | 0.0 | 0.0 | ------- | 0.0 |
| P5Y | -88.71 | 655.45 | ------- | -151.72 |
| WEIGHT | 3210 | 1987 | 1916 | 2317 |
| MAX DISP | -2.00 | -2.00 | -2.00 | -2.00 |
| MAX STRESS | 14211 | 24989 | 13206 | 24940 |

Table 1 Summary of Critical Dimensions for Design Example Solutions

## 6 Summary and Conclusions

A goal programming, multi-objective problem formulation with a hybrid optimization solution methodology has been presented and demonstrated on a design example. The key attributes of the procedure are the ability to handle multiple, competing objectives and that it is applicable at the earliest stage of the design process. The hybrid approach of a genetic algorithm for handling topological choice and a traditional nonlinear programming algorithm for local refinement is shown to be an effective design synthesis tool. The methodology presented allows for a formulation which is natural and parallels the manual design process better than conventional optimization methods.

## References

1. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA., 1989.
2. Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
3. Shupe, J.A. and Mistree, F., "Compromise: an Effective Approach for the Design of Damage Tolerant Structural Systems," *Computers and Structures*, 27(3), 1987, pp 407-415.
4. El Sayed, M., Ridgley, B. and Sandgren, E., "Structural Design by Nonlinear Goal Programming," *Computers and Structures*, 32, 1989, pp 69-73.
5. Sandgren, E., "Multicriteria Design Optimization by Goal Programming", *Advances in Design Optimization*, ed. H. Adeli, Chapman & Hall, London, 1994, pp 225-265.