A Genetic Algorithm for the Optimal Design of Axially Loaded Non-Prismatic Columns

By C. A. Coello Coello* **A. D. Christiansen†** **F. Alonso Farrera ‡**

Abstract: *This paper addresses the optimum design of columns, in which the objective is to minimize their volume under a given load by changing their shape, and assuming that are subject to buckling and strength constraints. We use a genetic algorithm (GA) to move through the search space of possible column designs, and choose the best one. Several issues arise when using the GA, such as how to decide which is the most appropriate representation scheme and how to fine tune its parameters. Both floating point and binary representation (with and without Gray coding) were used and compared to a more traditional optimization technique based on the generalized reduced gradient method. Our results show that the floating point representation provides the best solutions overall. Furthermore, we propose a very simple methodology to fine tune the GA parameters when using this kind of representation, and we show how the GA is able to find designs that reduce the column volume up to 30% in some cases, with respect to more traditional techniques.*

Keywords: structural optimization, genetic algorithms, design optimization, column design, non-prismatic columns, design automation, computer-aided design, column optimization, artificial intelligence.

Introduction

The optimization of structural members subject to compression forces has been of great interest for a long time. Cox [1992] gives a good historical review of the attempts to provide a solid mathematical analysis of the column. As he indicates, the aesthetic ideal was devised in the ancient Greece, and was originally reported by Vitruvius (circa 25 B.C.). This design consisted of a subtle variation on the cylindrical shape, with a bulge at approximately one third of the column's height and a diminution near its top. In 1738, Daniel Bernoulli sent a letter to Leonhard Euler in which he posed the problem of finding a curved that described the bending of a column. In 1743, Euler derived the formula for the critical buckling load of an ideal slender column (Euler 1743). He was also the first to solve the problem of inextensible elastica (i.e., for constant modulus of elasticity and modulus of inertia, he found the curve of prescribed length with prescribed terminal displacements and slopes and minimum stored energy) in 1757. Euler solved the case of a column that has the lower end fixed and the

*Graduate Student, Department of Computer Science, Tulane University, New Orleans, LA, 70118, USA
†Assistant Professor, Department of Computer Science, Tulane University, New Orleans, LA, 70118, USA
‡Assistant Professor, Escuela de Ingeniería Civil, UN. A. CH., Apartado Postal 61, Tuxtla Gutiérrez, Chiapas, México

upper end free. Later he extended (Euler 1757) his work on columns, and even today it has a great influence in every strength of materials textbook. In 1773, Lagrange [1867] unsuccessfully attempted to determine the optimum shape for a column, based on Euler's scientific derivations rather than only the physical appearance of the element. However, several missteps in his calculations led him to the mistaken conclusion that the cylinder was the strongest hinged column. There were very few contributions to Euler's work during several years, until Lamarle [1845] noticed that Euler's formula should be used only for slenderness ratios over a certain limit, and that the experimental data should be applied only to small ratios. T. Clausen [1851] was the first to find the optimal shape of a column of circular cross section hinged (pinned) at the end points, but didn't get much attention from the scientific community. In 1889 the French engineer Considère [1891] performed a set of 32 tests on columns, establishing the so-called *theory of the reduced module*. During that same year, the German engineer F. Engesser [1889] independently suggested the *theory of tangential module*. From these two theories, the first one dominated until 1946, when American professor F. R. Shanley indicated the logical paradoxes of both theories. In a remarkable paper of only one page [Shanley, 1946] he explained not only what was wrong with both theories, but he also proposed his own theory that solved the paradoxes.

The problem of non-prismatic columns (i.e., those with a variable cross-section area) has been studied more recently. A. N. Dinnik [1932] discussed the design of columns in which the moment of inertia of the cross-section areas varies according to a power of the distance along the member axis. Keller [1960] and later Tadjbakhsh and Keller [1962] derived optimal solutions to the strongest-column problem, which was characterized in the following way: "For a column of given length and volume of material, determine the column shape for which the Euler buckling load is maximum". In their analysis, Tadjbakhsh and Keller established the necessary conditions for a maximum by performing variations on the differential equations of equilibrium and associated boundary conditions, and the constraint of constant volume. Keller showed that of all twisted columns, hinged at the end points, with arbitrary convex cross sections, the strongest has all its cross sections equilateral triangles, and is not uniform, but is thickest at its center and thinnest at its ends.

J. Taylor [1967] studied the same problem using an energy approach, and presented a method to calculate a lower bound to the maximum eigenvalue. Spillers and Levy extended Keller's solution for the optimal design of columns to the case of plates [Spillers and Levy, 1990] and later, for axisymmetric cylindrical shells [Spillers and Levy, 1991]. However, in all these works, only the constraint of constant volume was considered and, as Fu and Ren [1992] point out, in a practical design, material strength constraints are equally important. With that in mind, these two last authors added such constraint to the optimization problem and used an algorithm called the generalized reduced gradient method [Reklaitis *et al.*, 1983] to select the design variables at nodal points. The results that they obtained are very reasonable and verifiable. The generalized reduced gradient method linearizes the non-linear constraints of this problem, and uses the convex simplex method to select the best direction of search from all the candidate directions which are both feasible and descent. Then the search for the optimum is started from the feasible initial point. Newton iteration is employed to adjust the basic variable to maintain feasibility. The convergence is accelerated by incorporating conjugate direction or quasi-Newton constructions. Naturally, the existence of continuous differentiability of the problem functions is a fundamental requirement for using the method.

Our work consisted in following this last approach, and apply the Genetic Algorithm (GA) instead of the generalized reduced gradient method. Some problems had to be faced, though, namely the

representation scheme and the optimal parameters to be used. However, our results are practically as good as those found by Fu and Ren [1992], and in at least one case we got a better solution than them. In fact, we'll see how in a particular example, the GA was able to generate a design that shows a reduction of the cross section in its center, introducing savings of about 32% with respect to the optimum design produced by the reduced gradient method used by Fu and Ren. Since this problem has a continuous search space, we experimented with both binary and floating point representation schemes. As the GA requires a discretization of the search space, we used an algorithm to map it in a form suitable for the GA. This turns out to be an easy task, since in real designs there is always a lower and an upper bound on the dimensions of the column. Finally, we also provide a very simple methodology that can be used to adjust the parameters of the GA, so that we can guarantee finding at least sub-optimal designs in a reasonable amount of time when using floating point representation.

Statement of the Problem

Given a column subject to axial load along the horizontal direction, the governing differential equation is

$$EIy'' + Py = 0 \tag{1}$$

where E is the modulus of elasticity, I is the moment of inertia, P is the axial load, and y is the function that represents the slenderness of the column. Let's assume that the column that we are going to study has the shape shown in Figure 1, where it has been divided into 6 equal-sized segments throughout its length. Then, Equation (1) may be expressed in a finite difference form, as [Fu and Ren, 1992]:

$$\frac{E}{h^2}\begin{bmatrix} -2I_2 & I_2 & 0 \\ I_3 & -2I_3 & I_3 \\ 0 & 2I_4 & -2I_4 \end{bmatrix}\begin{bmatrix} y_2 \\ y_3 \\ y_4 \end{bmatrix} + P\begin{bmatrix} y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

For non-trivial solution, the determinant must vanish, namely

$$\begin{vmatrix} \left(-2 + \frac{Ph^2}{EI_2}\right) & 1 & 0 \\ 1 & \left(-2 + \frac{Ph^2}{EI_3}\right) & 1 \\ 0 & 2 & \left(-2 + \frac{Ph^2}{EI_4}\right) \end{vmatrix} = 0 \tag{3}$$

or, in linear form:

$$\frac{P^3h^6}{E^3I_2I_3I_4} - 2\frac{P^2h^4}{E^2}\left(\frac{1}{I_2I_3} + \frac{1}{I_3I_4} + \frac{1}{I_2I_4}\right) + \frac{Ph^2}{E}\left(\frac{2}{I_2} + \frac{4}{I_3} + \frac{3}{I_4}\right) - 2 = 0 \tag{4}$$

where, for round and regular polygonal sections, the moments of inertia will be given by

$$I_i = \alpha D_i^4 \tag{5}$$

D_i is the diameter for round sections, or the side length for regular polygonal sections. Table 1 shows the values of α for the most commonly used cross-sections.

In general, for an n-sided regular polygon, α may be derived as

$$\alpha = \frac{n}{192} Cot\frac{\pi}{n}\left(3Cot^2\frac{\pi}{n} + 1\right) \tag{6}$$

For rectangular sections where width b is assumed to be constant throughout the length of the column,

$$I_i = \frac{bD_i^3}{12}, i = 2, 3, 4. \tag{7}$$

In a column design, Equation (4) represents a buckling constraint. Furthermore, a compressive strength constraint must also be satisfied, that is

$$\frac{P}{A_1} \leq \sigma_y, \tag{8}$$

where A_1 is a function of D_1. Since P and σ_y are always given values, we can compute the minimum D_1 or A_1 for each particular problem. This means that we may express the compressive strength constraint only in terms of D_1 or A_1.

Now, we have all the necessary elements to express the column design problem as an optimization problem. If we assume that P, h and σ_y are given, the objective is to minimize the volume of the column. Therefore, we'll consider 2 cases:

(1) Square or Round Columns: The objective function may be stated as [Fu and Ren, 1992]

Minimize:

$$V_s = K(D_1^2 + 2D_2^2 + 2D_3^2 + D_4^2 + D_1D_2 + D_2D_3 + D_3D_4) \tag{9}$$

where K is a constant defined according to Table 2, and V_c is the volume of the round or square column.

The objective function is subjected to the equality constraint defined by Equation (4), and the following additional inequality constraints:

$$C_1 < D_i < C_\mu, i = 1, 2, 3, 4 \tag{10}$$

where D_i are the design variables; C_1 and C_μ are, respectively, the lower and upper bounds of the design variables.

(2) Rectangular Columns: The objective function will be [Fu and Ren, 1992]:

Minimize:

$$V_r = \frac{bl}{9}(D_1 + 2D_2 + 2D_3 + D_4 + \sqrt{D_1D_2} + \sqrt{D_2D_3} + \sqrt{D_3D_4}) \tag{11}$$

where V_r is the volume of the rectangular column.

The objective function is subjected to the equality constraint defined by Equation (4), and the following similar equation that is derived on the basis of buckling in the orthogonal direction:

$$\frac{P^3h^6}{E^3I_2'I_3'I_4'} - 2\frac{P^2h^4}{E^2}\left(\frac{1}{I_2'I_3'} + \frac{1}{I_3'I_4'} + \frac{1}{I_2'I_4'}\right) + \frac{Ph^2}{E}\left(\frac{2}{I_2'} + \frac{4}{I_3'} + \frac{3}{I_4'}\right) - 2 = 0 \tag{12}$$

where

$$I_i^{'} = \frac{D_i b^3}{12} i = 1, 2, 3, 4 \tag{13}$$

Furthermore, an additional set of inequality constraints have to be satisfied

$$\left.\begin{array}{c} b \times D_1 > A_1 \\ C_l < D_i < C_\mu \\ C_l < b < C_\mu \end{array}\right\} i = 1, 2, 3, 4 \tag{14}$$

where $A_1 = P/\sigma_y$.

Notions of Genetic Algorithms

The famous naturalist Charles Darwin defined *Natural Selection* or *Survival of the Fittest* in his book [Darwin, 1929] as the *preservation of favorable individual differences and variations, and the destruction of those that are injurious.* In nature, individuals have to adapt to their environment in order to survive in a process called *evolution*, in which those features that make an individual more suited to compete are preserved when it reproduces, and those features that make it weaker are eliminated. Such features are controlled by units called *genes* which form sets called *chromosomes.* Over subsequent generations not only the fittest individuals survive, but also their fittest genes which are transmitted to their descendants during the sexual recombination process which is called *crossover.*

John H. Holland became interested in the application of natural selection to machine learning, and in the late 60s, while working at the University of Michigan, he developed a technique that allowed computer programs to mimic the process of evolution. Originally, this technique was called *reproductive plans*, but the term *genetic algorithm* became popular after the publication of his book [Holland, 1975] [Holland, 1992].

In 1989, Goldberg published a book [Goldberg, 1989] that provided a solid scientific basis for this area, and cited no less than 73 successful applications of the genetic algorithm. In the last few years the growing interest on this technique is reflected in a larger number of conferences, a new international journal, and an increasing amount of software and literature devoted to this subject.

Koza [Koza, 1992] provides a good definition of a GA:

> The *genetic algorithm* is a highly parallel mathematical algorithm that transforms a set (*population* of individual mathematical objects (typically fixed-length character strings patterned after chromosome strings), each with an associated *fitness* value, into a new population (i.e., the next *generation*) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (notably sexual recombination).

Actually, the genetic algorithm derives its behavior from a metaphor of one of the mechanisms of evolution in nature which is called *hard selection* [Heitkoetter and Beasley, 1995]. Under this scheme, only the best available individuals are retained for generating descendants. This contrasts with *soft selection*, which offers a probabilistic mechanism for maintaining individuals to be parents of future progeny despite possessing relatively poorer objective values.

It has been argued [Heitkoetter and Beasley, 1995] that the term *genetic algorithm* (GA) is misleading, since natural selection is only one of the mechanisms of evolution, and it would be more appropriate to call them *hard selection* (HS) algorithms to reflect the fact that they deal with only that particular selection scheme. However, the term is so common today, that a change does not seem feasible, at least in the near future.

A genetic algorithm for a particular problem must have the following five components [Michalewicz, 1992]:

1. A representation for potential solutions to the problem.
2. A way to create an initial population of potential solutions.
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness".
4. Genetic operators that alter the composition of children.
5. Values for various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

Some of the basic terminology used by the genetic algorithms (GAs) community is the following [Heitkoetter and Beasley, 1995]:

- A *chromosome* is a data structure that holds a "string" of task parameters, or genes. This string may be stored, for example, as a binary bit-string (binary representation) or as an array of integers (floating point o real-coded representation) that represent a floating point number. This chromosome is analogous to the base-4 chromosomes present in our own DNA. Normally, in the GA community, the haploid model of a cell is assumed (one-chromosome individuals). However, diploids have also been used in the past [Goldberg, 1989].
- A *gene* is a subsection of a chromosome that usually encodes the value of a single parameter.
- An *allele* is the value of a gene. For example, for a binary representation each gene may have an allele of 0 or 1, and for a floating point representation, each gene may have an allele from 0 to 9.
- A *schema* (plural *schemata*) is a pattern of gene values in a chromosome, which may include "do not care" states (represented by a # symbol). Thus in a binary chromosome, each schema can be specified by a string of the same length as the chromosome, with each character being one of { 0, 1, # }. A particular chromosome is said to "contain" a particular schema if it matches the scheme (e.g. chromosome 01101 matches schema #1#0#).

- If the solution of a problem can be represented by a set of N real-valued parameters, then the job of finding this solution can be thought of as a search in an N-dimensional space. This region is simply referred as the *search space* of the problem.

- The *fitness* of an individual is a value that reflects its performance (i.e., how well solves a certain task). A *fitness function* is a mapping of the chromosomes in a population to their corresponding fitness values. A *fitness landscape* is the hypersurface obtained by applying the fitness function to every point in the search space.

- A *building block* is a small, tightly clustered group of genes which have co-evolved in such a way that their introduction into any chromosome will be likely to give increased fitness to that chromosome. The *building block hypothesis* [Goldberg, 1989] states that GAs generate their solutions by first finding as many building blocks as possible, and then combining them together to give the highest fitness.

- *Deception* a condition under which the combination of good building blocks leads to reduced fitness, rather than increased fitness. This condition was proposed by Goldberg [Goldberg, 1989] as a reason for the failure of GAs on certain tasks.

- *Elitism* (or an elitist strategy) is a mechanism which ensures that the chromosomes of the highly fit member(s) of the population are passed on to the next generation without being altered by any genetic operator. The use of elitism guarantess that the maximum fitness of the population never decreases from one generation to the next, and it normally produces a faster convergence of the population.

- *Epistasis* is the interaction between different genes in a chromosome. It is the extent to which the contribution to fitness of one gene depends on the values of other genes. Geneticists use this term to refer to a "masking" or "switching" effect among genes, and a gene is considered to be "epistatic" if its presence suppresss the effect of a gene at another locus. This concept is closely related to deception, since a problem with high degree of epistasis is deceptive, since building blocks can not be formed. On the other hand, problems with little or no epistasis are trivial to solve (hillclimbing is sufficient).

- *Exploitation* is the process of using information gathered from previously visited points in the search space to determine which places might be profitable to visit next. Hillclimbing is an example of exploitation, because it investigates adjacent points in the search space, and moves in the direction giving the greatest increase in fitness. Exploitation techniques are good at finding local minima (or maxima). The GA uses crossover as an exploitation mechanism.

- *Exploration* is the process of visiting entirely new regions of a search space, to see if anything promising may be found there. Unlike exploitation, exploration involves leaps into unknown regions. Random search is an example of exploration. Problems which have many local minima (or maxima) can sometimes only be solved using exploration techniques such as random search. The GA uses mutation as an exploration mechanism.

- A *genotype* represents a potential solution to a problem, and is basically the string of values chosen by the user, also called chromosome.

- A *phenotype* is the meaning of a particular chromosome, defined externally by the user.
- Genetic drift is the name given to the changes in gene/allele frequencies in a population over many generations, resulting from chance rather than from selection. It occurs most rapidly in small populations and can lead to some alleles to become extinct, thus reducing the genetic variability in the population.
- A em niche is a group of individuals which have similar fitness. Normally in multiobjective and multimodal optimization, a technique called *sharing* is used to reduce the fitness of those individuals who are in the same niche, in order to prevent the population to converge to a single solution, so that stable sub-populations can be formed, each one corresponding to a different objective or peak (in a multimodal optimization problem) of the function.

The basic operation of a Genetic Algorithm is illustrated in the following segment of pseudo-code [Buckles and Petry, 1992]:

```
generate initial population, G(0);
evaluate G(0);
t:=0;
repeat
   t:=t+1;
   generate G(t) using G(t-1);
   evaluate G(t);
until a solution is found
```

First, an initial population is randomly generated. The individuals of this population will be a set of chromosomes or strings of characters (letters and/or numbers) that represent all the possible solutions to the problem. We apply a *fitness function* to each one of these chromosomes in order to measure the quality of the solution encoded by the chromosome. Knowing each chromosome's fitness, a *selection* process takes place to choose the individuals (presumably, the fittest) that will be the parents of the following generation. The most commonly used selection schemes are the following [Goldberg and Deb, 1991]:

- **Proportionate Reproduction:** This term is used generically to describe several selection schemes that choose individuals for birth according to their objective function values f. In these schemes, the probability of selection p of an individual from the ith class in the tth generation is calculated as

$$p_{i,t} = \frac{f_i}{\sum_{j=1}^{k} m_{j,t} f_j} \tag{15}$$

  where k classes exist and the total number of individuals sums to n. Several methods have been suggested for sampling this probability distribution, including Monte Carlo or *roulette wheel* selection [Jong, 1975], *stochastic remainder* selection [Booker, 1982] [Brindle, 1981], and *stochastic universal* selection [Baker, 1987] [Grefenstette and Baker, 1989].

- **Ranking Selection:** In this scheme, proposed by Baker [Baker, 1985] the population is sorted from best to worst, and each individual is copied as many times as it can, according to a non-increasing assignment function, and then proportionate selection is performed according to that assignment.

- **Tournament Selection:** The population is shuffled and then is divided into groups of k elements from which the best individual (i.e., the fittest) will be chosen. This process has to be repeated k times because on each iteration only m parents are selected, where

$$m = \frac{population\ size}{k}$$

  For example, if we use binary tournament selection ($k = 2$), then we have to shuffle the population twice, since in each stage half of the parents required will be selected. The interesting property of this selection scheme is that we can guarantee multiple copies of the fittest individual among the parents of the next generation.

- **Steady State Selection:** This is the technique used in Genitor [Whitley, 1989], which works individual by individual, choosing an offspring for birth according to linear ranking, and choosing the currently worst individual for replacement.

After being selected, *crossover* takes place. During this stage, the genetic material of a pair of individuals is exchanged in order to create the population of the next generation. The two main ways of performing crossover are called single-point and two-point crossover. When a *single-point* crossover scheme is used, a position of the chromosome is randomly selected as the crossover point as indicated in Figure 2. When a *two-point* crossover scheme is used, two positions of the chromosome are randomly selected as indicated in Figure 3.

Mutation is another important genetic operator that randomly changes a gene of a chromosome. If we use a binary representation, a mutation changes a 0 to 1 and viceversa. This operator allows the introduction of new chromosomic material to the population and, from the theoretical perspective, it assures that—given any population—the entire search space is connected [Buckles and Petry, 1992].

If we knew in advance the final solution, it would be trivial to determine how to stop a genetic algorithm. However, as this is not normally the case, we have to use one of the two following criteria to stop the GA: either give a fixed number of generations in advance, or verify when the population has stabilized (i.e., all or most of the individuals have the same fitness).

GAs differ from traditional search techniques in several ways [Buckles and Petry, 1992]:

- GAs don't require problem specific knowledge to carry out a search.

- GAs use stochastic instead of deterministic operators and appear to be robust in noisy environments.

- GAs operate on multiple partial solutions simultaneously (sometimes called *implicit parallelism*), gathering information from a population of search points to direct subsequent search efforts. Their ability to maintain multiple partial solutions concurrently helps make GAs less susceptible to the problems of local maxima and noise.

The traditional representation used by the genetic algorithms community is the binary scheme according to which a chromosome is a string the form $\langle b_1, b_2, \ldots, b_m \rangle$, where $b_1, b_2, \ldots, b_m$ are called *alleles* (either zeros or ones). Since the binary alphabet offers the maximum number of schemata per bit of information of any coding [Goldberg, 1989], its use has became very popular among scientists. This coding also facilitates theoretical analysis of the technique and allows elegant genetic operators. However, since the "implicit parallelism" property of GAs does not depend on using bit strings [Michalewicz, 1992] it is worthwhile to experiment with larger alphabets, and even with new genetic operators. In particular, for optimization problems in which the parameters to be adjusted are continuous, a floating point representation scheme seems a logical choice. According to this representation, a chromosome is a string of the form $\langle d_1, d_2, \ldots, d_m \rangle$, where $d_1, d_2, \ldots, d_m$ are digits (numbers between zero and nine). Consider the examples shown in Figure 4, in which the same value is represented using binary and floating point encoding.

The term "floating" may seem misleading since the position of the implied decimal point is at a fixed position, and the term "fixed point representation" seems more appropriate. However, the reason that the term "floating point" is preferred is because in this representation each variable (representing a parameter to be optimized) may have the point at any position along the string. This means that even when the point is fixed for each gene, is not necessarily fixed along the chromosome. Therefore, some variables could have a precision of 3 decimal places, while others are integers, and still they could all be represented with the same string. Nevertheless, the term *real-coded* GAs is also used in the literature [Goldberg, 1990] [Wright, 1991].

Floating point representation is faster and easier to implement, and provides a higher precision than its binary counterpart, particularly in large domains, where binary strings would be prohibitively long. One of the advantages of floating point representation is that it has the property that two points close to each other in the representation space must also be close in the problem space, and vice versa [Michalewicz, 1992]. This is not generally true in the binary approach, where the distance in a representation is normally defined by the number of different bit positions.

Goldberg [Goldberg, 1990] has presented a theory of convergence for real-coded or floating-point GAs, and also real numbers and other alphabets have been proposed [Wright, 1991], particularly for numerical optimization, in a resemblance of the power of evolutionary strategies [Schwefel, 1981] in this domain.

Use of the Genetic Algorithm

To solve this problem, we used the Simple Genetic Algorithm (SGA) proposed by Goldberg [1989]. An issue in this application is the representation scheme, because we are dealing with real-valued parameters, and therefore it is necessary to use some kind of discretization, so that we can apply a binary representation scheme. The algorithm that we used for the discretization of the search space was the following:

$difference = L_s - L_i$
Determine necessary number of bits using *difference*
If $dv >$ *difference* then $dv =$ *difference*
$dv = (L_i + dv)/1000.0$

Here, dv stands for *decoded value*, and L_s and L_i are respectively the upper and lower bounds multiplied by one thousand (we considered only a 3 decimals precision, but this value can be modified as desired). As we can see in this algorithm, we first determine how many bits we require to represent the total amount of responses–i.e., by rounding up to a certain fixed amount of decimals, we discretize the search space–. As this value will hardly be an exact power of two, then we use the next immediate value. The decision showed after that is necessary to adjust the responses to valid values. The last line of the algorithm allows us to get the real value that we want, since the results always will be shifted L_i positions with respect to the starting point, and they also have to be divided by a thousand.

We also tried to use Gray codes as suggested by Goldberg [1989]. Consider a binary number $\mathbf{b}= \langle b_1, \ldots, b_m \rangle$ that we want to convert into a Gray code number $\mathbf{g}= \langle g_1, \ldots, g_m \rangle$, where m denotes the number of bits. The following two procedures can be used to convert a binary number into Gray coding and viceversa [Michalewicz, 1992]:

procedure Binary-to-Gray
begin
  $g_1 = b_1$
  for $k = 2$ **to** m **do**
    $g_k = b_{k-1}\ XOR\ b_k$
end

procedure Gray-to-Binary
begin
  $value = g_1$
  $b_1 = value$
  for $k = 2$ **to** m **do**
  begin
    if $g_k = 1$ **then** $value = NOT\ value$
    $b_k = value$
  end
end

The Gray code representation has the property that any two points next to each other in the problem space differ by only one bit [Michalewicz, 1992]. In other words, an increase of one step in the parameter value corresponds to a change of a single bit in the code. This is a well known technique used to reduce the distance of two points in the problem space, and it is argued to bring some benefit because of their adjacency property, and the small perturbation caused by many single mutations. Nevertheless, the use of Gray codes didn't improve much the performance of the GA in this particular application, as we'll see later on.

Finally, we used a floating point representation, since it is conceptually closest to the problem space [Michalewicz, 1992], and allows the easy and efficient implementation of closed and dynamic operators.

We'll see how this last approach provided the best results, both in terms of the precision obtained and in terms of the computation time needed.

The fitness function that we used is illustrated by the following algorithm:

$$
\begin{aligned}
&\textit{check1} = \text{Error in Equation (4)} \\
&\text{If } P/(A_1 \times \sigma_y) - 1.0 > 0.0 \text{ then } \textit{check2} = 1.0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{else } \textit{check2} = 0.0 \\
&\textit{fitness} = 1.0/(\textit{vol} \times 800 \times (\textit{check1} + \textit{check2}) + 1.0)
\end{aligned}
$$

As we can see, if our answer violates the constraint imposed by Equation (4) then the fitness function will be penalized by the error produced. On the other hand, if it violates the stress constraint (i.e. $P/A_1 \leq \sigma_y$), then the penalty is 1.0. For rectangular columns, the constraint is that $b \times D_1 > P/\sigma_y$. In this last case, we must also check the orthogonal direction, so that we have three penalty values instead of two. These values are added and the result is multiplied by 800–we magnify the error– so that we "punish" our result. Note that when there are no violations to any of the constraints the fitness function returns the inverse of the volume (we had to use the inverse, because GAs only maximize, and this is a minimization problem).

We used a simple genetic algorithm (SGA) as that described by Goldberg [1989], but with some modifications: we used two-point crossover and binary tournament selection. The four diameters that we want to find were represented by consecutive strings of the same length. The halting criteria was through a maximum number of generations. The GA was implemented in Turbo Pascal 7.0 using the technique proposed by Porter [1988] for dynamic memory management. We found experimentally that the following parameters seem to give the best results when using binary representation:

Population size	$\Rightarrow 400$
Crossover probability	$\Rightarrow 0.80$
Mutation probability	$\Rightarrow 0.01$

We had to derive our own methodology to fine tune the GA parameters when using floating point representation, since the chromosomes were more sensitive to changes in this case, because the mutation operator produced larger jumps in the search space.

Fine Tuning the GA Parameters

One of the main problems when using GAs is how to choose the most appropriate parameter values (i.e., population size, maximum number of generations, mutation and crossover rate). This is normally a trial and error process which takes some time. One of the experiences derived from this research was the fact that it turns out to be much harder to fine tune the parameters of the GA when a floating point representation scheme is used. We faced a dilemma: the floating point representation gave the best results, but was also the hardest to deal with in terms of finding the most appropriate parameters.

Obviously any optimization system won't be very useful if its outcomes are completely unpredictable. After a lot of experimentation, we came out with a systematic empirical process that seems to be able to generate optimal (or at least sub-optimal) solutions in a very short period of time. However, we don't have yet any theoretical support of its reliability, even though the empirical evidence is quite solid. The method is the following:

- Choose a certain value for the random numbers seed and make it a constant.
- Make constants also the population size and the maximum number of generations (we used 400 chromosomes and 50 generations, respectively).
- Loop the mutation and crossover rates from 0.1 to 0.9 at increments of 0.1 (this is actually a nested loop). This implies that 81 runs are necessary.
- For each run, update 2 files. One contains only the final costs, and the other has a summary that includes, besides the cost, the corresponding values of the design parameters and the mutation and crossover rates used. These files only contain information about relevant designs—i.e., designs that don't violate any constraints in a greater degree than the optimization technique with which the GA was compared—.
- When the whole process ends up, the file with the costs is sorted in ascending order, and the smallest value is searched in the other file, returning the corresponding design parameters as the final answer.

Since each run is completely independent from the others, we can perform all this process in parallel, so that the total execution time will be practically the same required for a single run (approximately 10 seconds in a PC DX/2 running at 66 MHz and with a mathematical coprocessor).

Examples

The following examples were taken from Fu and Ren [1992]:

(1) Example 1: Select the best diameters at nodal points for a steel round column of 10' length which is subjected to an axial load of 400 kips. The modulus of elasticity is $E = 30 \times 10^6$ psi and the yield strength, σ_y is $60,000$ psi. Thus, the minimum diameter may be computed as 2.914". The design variables are the diameters at nodal point, D_1, D_2, D_3 and D_4. The lower and upper bounds, C_l and C_μ are 2.914" and 20", respectively. The size of the search space for this problem is $(20000 - 2914)^4 \cong 8.52 \times 10^{16}$. The results are shown on Table 3. It is interesting to observe how our approach generates a design that has a slightly higher volume than Fu and Ren's method, but it violates in a lower degree the constraints imposed by Equation (4). This is because of the penalty applied to the fitness function, that drives the GA towards solutions that minimize all the constraints at the same time. Also, notice how the Gray coding produced a poorer solution, both in terms of the volume and in terms of the constraints. Figures 5, 6 and 7 show the convergence graph of the genetic algorithm for this example using binary representation with and without Gray encoding, and using floating point representation, respectively.

(2) Example 2: Select the best side-widths at nodal points for a steel square column of 10' length which is subjected to an axial load of 400 kips. The modulus of elasticity is $E = 30 \times 10^6$ psi

and the yield strength, σ_y is 60,000 psi. Thus, the minimum diameter may be computed as 2.582". The design variables are the diameters at nodal point, D_1, D_2, D_3 and D_4. The lower and upper bounds, C_l and C_μ are 2.582" and 20", respectively. The size of the search space for this problem is $(20000 - 2582)^4 \cong 9.20 \times 10^{16}$. The results are shown on Table 4. In this case we find a solution that produces a greater volume that Fu and Ren's method, but violating fewer of the constraints imposed by Equation (4). In this case the use of Gray coding produced a poorer solution in terms of the volume but a better one in terms of the constraints. Figures 8, 9 and 10 show the convergence graph of the genetic algorithm for this example using binary representation with and without Gray encoding, and using floating point representation, respectively.

(3) Example 3: Select the best side-widths at nodal points for a steel equilateral triangle column of 10' length which is subjected to an axial load of 400 kips. The modulus of elasticity is $E = 30 \times 10^6$ psi and the yield strength, σ_y is 60,000 psi. Thus, the minimum diameter may be computed as 3.924". The design variables are the diameters at nodal point, D_1, D_2, D_3 and D_4. The lower and upper bounds, C_l and C_μ are 3.924" and 20", respectively. The size of the search space for this problem is $(20000 - 3924)^4 \cong 6.70 \times 10^{16}$. The results are shown on Table 5. In this problem, the GA behave as in the previous example, finding a solution less optimal in terms of the volume, but that violates fewer of the constraints imposed by Equation (4). Also in this case the use of Gray coding produced a poorer solution in terms of the volume but a slightly better one in terms of the constraints. Figures 11, 12 and 13 show the convergence graph of the genetic algorithm for this example using binary representation with and without Gray encoding, and using floating point representation, respectively.

(4) Example 4: Select the best side-widths at nodal points for a steel rectangular column of 10' length which is subjected to an axial load of 400 kips. The modulus of elasticity is $E = 30 \times 10^6$ psi and the yield strength, σ_y is 60,000 psi. Thus, the minimum diameter may be computed as 1.500". The design variables are the diameters at nodal point, D_1, D_2, D_3 and D_4. The lower and upper bounds, C_l and C_μ are 1.500" and 20", respectively. The size of the search space for this problem is $(20000 - 1500)^4 \cong 1.20 \times 10^{17}$. The results are shown in Table 6. In this problem we had to use a larger population than in the others (500 chromosomes as compared to the 400 used in the others), and we ran the algorithm for 100 generations, instead of the 50 generations used before. The reason for this change of parameters was the extra length added to our chromosomic strings in this case (we have 15 extra bits) because of the extra parameter needed (the width b of the column). We can see how in this case we found a design that has 32% less volume than the solution produced by the generalized reduced gradient method. Given the values of the diameters, we can see that the shape of this column is thinnest at its center, as oppossed to the previous designs in which the center is precisely the thickest part of the element. It's also interesting to notice that, even though our design introduces such savings of material, the constraints imposed by Equations (4) and (14) are violated in a lower degree than Fu and Ren's design. In fact, strictly speaking, they are slightly violating one of the constraints imposed by Equation (14), because $P/\sigma_y > b \times D_1$. Also in this case the use of Gray coding produced a poorer solution in terms of the volume but a slightly better one in terms of the constraints. Figures 14, 15 and 16 show the convergence graph of the genetic algorithm for this example using binary representation with and without Gray encoding, and using floating point representation, respectively.

Future Work

We are now working in a generalization of this problem, so that columns of any material (probably even of composite materials) can be considered. This will complicate the analysis a bit more. Also, we are trying to approach the problem of considering more than one objective function at a time (multiobjective optimization), in which we could have conflicting objectives. This will introduce some changes in the way in which the GA has to be applied, but it will drive us towards a more realistic view of structural design. Our final goal is to have a complete computer-aided structural design system for framed structures. So far, our work with columns [Coello and Christiansen, 1995], plane and space trusses [Coello and Christiansen, 1994] [Coello *et al.*, 1994], and reinforced concrete beams [Coello *et al.*, 1995] has provided us with very encouraging results, and we expect to develop prototypes for the remaining framed structures (i.e., plane and space frames and plane grids) in the next few months.

Conclusions

We have shown another successful application of the genetic algorithm to an engineering optimization problem. In this case we saw how a floating point representation worked better when dealing with a continuous search space. This representation not only beats the binary representation scheme (with or without Gray encoding) in terms of the precision obtained, but also in terms of the speed, because we can use the same genetic operators with only slight modifications, and the convergence will be faster since the chromosomes are of a considerable shorter length. This problem is an interesting one because, even when its analysis is very simple, it normally has fairly large search spaces and several constraints. The penalty technique that we used has proved to be useful in incorporating the constraints into the fitness function for this particular application, as can be seen from our results. The selection mechanism of the GA was able to find interesting solutions that escape to the search capabilities of more conventional techniques. Finally, we also proposed a methodology to fine tune the parameters of the GA, so that the designer may generate an optimal (or at least sub-optimal) solution in a reasonable amount of time.

The problem treated here is one that still challenges mathematicians and engineers all over the world, because it refers to the design of the optimal geometrical shape of a column. Even if the precise mathematical answer of our method may not be suitable for practical civil engineering applications, it may be very useful in industry, mainly for mass-production of structural elements.

References

[Baker, 1985] Baker, J. E. (1985) Adaptive selection methods for genetic algorithms. In Grefenstette, J. J., editor 1985, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale, New Jersey. Lawrence Erlbaum. 100–111.

[Baker, 1987] Baker, J. E. (1987) Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J., editor 1987, *Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, New Jersey. Lawrence Erlbaum Associates. 14–21.

[Booker, 1982] Booker, L. B. (1982) Intelligent behavior as an adaptation to the task environment. Technical Report 243, University of Michigan at Ann Arbor, Ann Arbor, Michigan.

[Brindle, 1981] Brindle, A. (1981) *Genetic Algorithms for Function Optimization.* Ph.D. Dissertation, Department of Computer Science of the University of Alberta, Alberta, Canada.

[Buckles and Petry, 1992] Buckles, B. P. and Petry, F. E. (1992) *Genetic Algorithms.* Technology Series. IEEE Computer Society Press.

[Clausen, 1851] Clausen, T. (1851) Über die form architektonischer säulen. *Bulletin physico-mathématique de l'Académie* 9:368–379.

[Coello and Christiansen, 1994] Coello, C. A. and Christiansen, A. D. (1994) Optimization of truss designs using genetic algorithms. Technical Report TUTR-CS-94-102, Tulane University.

[Coello and Christiansen, 1995] Coello, C. A. and Christiansen, A. D. (1995) Using genetic algorithms for optimal design of axially loaded non-prismatic columns. In Pearson, D. W.; Steele, N. C.; and Albrecht, R. F., editors 1995, *International Conference on Artificial Neural Nets and Genetic Algorithms*, Ecole des Mines d'Alès, France. Springer-Verlag. 460–463.

[Coello *et al.*, 1994] Coello, C. A.; Rudnick, M.; and Christiansen, A. D. (1994) Using genetic algorithms for optimal design of trusses. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, New Orleans, LA. IEEE Computer Society Press. 88–94.

[Coello *et al.*, 1995] Coello, C. A.; Hernández, F. S.; and Farrera, F. A. (1995) An approach to optimal design of reinforced concrete beams using genetic algorithms. In *IASTED International Conference on Applied Modelling, Simulation and Optimization*, Cancún, México. IASTED. (Accepted for publication).

[Considère, 1891] Considère, A. (1891) Résistance des pièces comprimées. In *Congrès International des Procédés de Construction*, volume 3. Libraire Polytechnique, Paris. 371.

[Cox, 1992] Cox, S. J. (1992) The shape of the ideal column. *The Mathematical Intelligencer* 14(1):16–24.

[Darwin, 1929] Darwin, C. (1929) *The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life.* The Book League of America. Originally published in 1859.

[Dinnik, 1932] Dinnik, A. N. (1932) Design of columns of varying cross-section. *Transactions ASME* 54.

[Engesser, 1889] Engesser, F. (1889) Uber die knickfestigkeit gerader stäbe. *Zeischrift für Architektur und Ingenieurwesen* 35(4):455–562.

[Euler, 1960a] Euler, L. (1960) Euler's calculation of buckling loads for columns of non uniform section. In *The Rational Mechanics of Flexible or Elastic Bodies 1638-1788.* Orell Füssli Turici, Societatis Scientiarum Naturalium Helveticae. 345–347. Originally published in 1757.

[Euler, 1960b] Euler, L. (1960) Euler's treatise on elastic curves. In *The Rational Mechanics of Flexible or Elastic Bodies 1638-1788.* Orell Füssli Turici, Societatis Scientiarum Naturalium Helveticae. 199–219. Originally published in 1743.

[Fu and Ren, 1992] Fu, K. C. and Ren, D. (1992) Optimization of axially loaded non-prismatic column. *Computers and Structures* 43(1):159–62.

[Goldberg and Deb, 1991] Goldberg, D. E. and Deb, K. (1991) A comparison of selection schemes used in genetic algorithms. In Rawlins, G. E., editor 1991, *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, California. 69–93.

[Goldberg, 1989] Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Mass. : Addison-Wesley Publishing Co.

[Goldberg, 1990] Goldberg, D. E. (1990) Real-coded genetic algorithms, virtual alphabets and blocking. Technical Report 90001, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[Grefenstette and Baker, 1989] Grefenstette, J. J. and Baker, J. E. (1989) How genetic algorithms work: A critical look at implicit parallelism. In Schaffer, J. D., editor 1989, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, California. George Mason University, Morgan Kaufmann Publishers. 20–27.

[Heitkoetter and Beasley, 1995] Heitkoetter, J. and Beasley, D. The hitch-hiker's guide to evolutionary computation (faq in comp.ai.genetic). USENET. (Version 3.3).

[Holland, 1975] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Harbor : University of Michigan Press.

[Holland, 1992] Holland, J. H. (1992) *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Massachusetts.

[Jong, 1975] Jong, A. K. D. (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. Dissertation, University of Michigan.

[Keller, 1960] Keller, J. B. (1960) The shape of the strongest column. *Archive for Rational Mechanics and Analysis* 5:275–85.

[Koza, 1992] Koza, J. R. (1992) *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. The MIT Press.

[Lamarle, 1845] Lamarle, A. H. E. (1845) *Mémoire sur la flexion du bois*, volume 3. France. 1–64.

[Michalewicz, 1992] Michalewicz, Z. (1992) *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition.

[Porter, 1988] Porter, K. (1988) Handling huge arrays. *Dr. Dobb's Journal of Software Tools for the Professional Programmer* 13(3):60–3.

[Reklaitis *et al.*, 1983] Reklaitis, G. V.; Ravindran, A.; and Ragsdell, K. M. (1983) *Engineering Optimization. Methods and Applications*. John Wiley and Sons.

[Schwefel, 1981] Schwefel, H. P. (1981) *Numerical Optimization of Computer Models*. John Wiley and sons, Great Britain.

[Serret, 1867] Serret, J. A., editor 1867 (1867) *Oeuvres de Lagrange*. Gauthier-Villars, Paris, France.

[Shanley, 1946] Shanley, F. R. (1946) The column paradox. *Journal of the Aeronautical Sciences* 13(12):261.

[Spillers and Levy, 1990] Spillers, W. R. and Levy, R. (1990) Optimal design for plate buckling. *Journal of Structural Engineering* 116(3):850–8.

[Spillers and Levy, 1991] Spillers, W. R. and Levy, R. (1991) Optimal design for axisymmetric cylindrical shell buckling. *Journal of Engng Mech. ASCE* 115:1683–90.

[Tadjbakhsh and Keller, 1962] Tadjbakhsh, I. and Keller, J. B. (1962) Strongest columns and isoperimetric inequalities for eigenvalues. *Journal of Applied Mechanics* 29(1):159–64. Transactions ASME Series E.

[Taylor, 1967] Taylor, J. E. (1967) The strongest column: an energy approach. *Journal of Applied Mechanics* 34:486–7. Transactions of the ASME.

[Vitruvius, 1931] Vitruvius, P. (1931) *On Architecture*. G. P. Putnam's sons, New York. F. Granger (translator).

[Whitley, 1989] Whitley, D. (1989) The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Schaffer, J. D., editor 1989, *Proceedings of the Third Conference on Genetic Algorithms*, San Mateo, California. George Mason University, Morgan Kaufmann Publishers. 116–121.

[Wright, 1991] Wright, A. H. (1991) Genetic algorithms for real parameter optimization. In Rawlins, G. J. E., editor 1991, *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers, San Mateo, California. 205–218.