Abstract: *In this paper, we propose an extension to an optimization model for design of reinforced concrete beams subject to a specified set of constraints. The new model is more suitable for real-world applications, because it better follows reinforced concrete design regulations and practical recipes used by experienced engineers. To solve this new model, we used an artificial intelligence technique based on the mechanics of natural selection, called the genetic algorithm. In order to deal with the parameter tuning problem that characterizes genetic algorithms applications (i.e., how to choose the population size, crossover and mutation rates, and maximum number of generations), we developed a very simple methodology that provides good results in a reasonable amount of time when using floating point representation. A prototype of our system is currently being tested so that we can have a reliable design tool that can be used to solve real-world problems in a short period of time.*

Keywords*:* genetic algorithms, design optimization, structural optimization, artificial intelligence.

1 Introduction

In the traditional design methodology, a solution is proposed and then corroborated by mathematical analysis in order to verify that the problem requirements are satisfied. If the requirements are not satisfied, a new solution is proposed. In this trial and error process the engineer gains experience, but at a very high cost in terms of time and effort. As time is always a constraint in real design, a reasonable sub-optimal solution is normally adopted. Computers have been recently used to help engineers automate this process. However, their use has concentrated mainly in performing the tedious mathematical calculations that are required. Alternatively, the optimal design approach consists of changing the design based on a certain "optimality condition". However, the general optimal design problem is highly nonlinear and nonconvex [1]. As a result, structural optimization problems are characterized by having multiple local optima. This paper focuses on the use of an artificial intelligence (AI) technique based on the mechanics of natural selection, called the *genetic algorithm* (GA) [2] [3]. The design process based on this technique is very similar to the optimal design process previously mentioned. The main difference is the notion of a *fitness function* instead of a cost function, and the fact that the adaptation of the design is dependent upon neither (a) the engineer nor (b) the gradient of the cost function, as in the two previous cases. Even more

interesting is that initial designs are randomly generated, without any human intervention, and nevertheless the technique converges to an optimal design in a reasonable amount of time.

The design of a reinforced concrete beam is normally an interactive process in which the engineer assumes the self-weight of the beam beforehand, and a trial section is chosen. Then, the moment of resistance of this section is determined, to check its suitability against the given applied bending moment. This process is repeated until a trial section is found suitable. This procedure often creates difficulty in exactly matching the moment of resistance of the section with the total applied bending moment due to the self-weight of the beam, which may be quite substantial in many cases. Therefore, the design process of a beam is not only slow, but also has a complete lack of economics, since the only concern is to find any section suitable for the given conditions, without even considering the possibility of making it as cheap as possible.

In this paper, we'll present a model for optimal design which minimizes the cost of a reinforced concrete beam based not only on the allowable stresses of the element, but also in the costs of concrete, steel and shuttering. Our model follows that proposed by Chakrabarty [4] [5], with certain modifications (i.e., additional constraints) that makes it suitable for practical applications. In the next section, we'll introduce some general concepts from reinforced concrete design. Then, our model will be shown and the genetic algorithm approach will be described. Finally, we'll present the results found by our model when solving some problems found in the literature, and we'll briefly discuss some of the issues that arose when using genetic algorithms in this kind of application.

2 Basic Concepts

According to the strength design method adopted for this work, the nominal moment capacity M_n of a rectangular beam with tension reinforcement only is given by [6]:

$$M_n = bd^2 f_c^{'} w(1 - 0.59w) \tag{1}$$

where b is the width of the beam, d is the distance from the extreme compressive fibre to the centroid of tension reinforcement, $f_c^{'}$ is the compressive strength of concrete, $w = (A_s f_y / bdf_c^{'})$, f_y is the yield strength of reinforcement and A_s is the area of tension reinforcement. There is an infinite amount of solutions to equation (1) that yield the same value of M_n [6]. In the traditional design process, the values of b and/or d are assumed, and the remaining parameters are

calculated based on them, iterating until a suitable section is found. An obvious restriction of this approach is that only a few sections can be evaluated in this manner. Since equation (1) does not incorporate any cost parameter, there is no way of achieving a least-cost design. Therefore, we need to include certain cost parameters combined with the design parameters in our optimal design model, so that we can produce least-cost suitable designs.

3 Previous Work

The optimal design of beams was first proposed by Galileo [7], even though his calculations were wrong. Apparently, the doctoral dissertation by E. J. Haug Jr. [8] (see also [9]) in 1966 is one of the first modern attempts to use a digital computer as a tool for the optimal design of this structural element. Haug reduced the non-linear optimal design problem to a Lagrange problem in the Calculus of Variations. His model includes restrictions and tries to minimize the weight of the beam in several different situations. Venkayya [10] developed a method based on an energy criteria and a search procedure for design of structures subjected to static loading. He argues that his method can handle very efficiently: (a) designs for multiple load conditions, (b) stress constraints, (c) constraints on displacements, (d) constraints on sizes of the elements. His method has been successfuly applied to the design of trusses, frames and beams. Osyczka [11] [12] has applied multi-objective optimization techniques to beam design problems. Also, Prakash et al. [13] proposed a model for optimal design of reinforced concrete sections in which the costs of steel, concrete and shuttering were included. Chakrabarty's model [4] [5] has some similarities with Prakash's model, but the former is more complete and detailed. That's the main reason why we decided to use Chakrabarty's model as a basis for our implementation, even though we had to slightly modify it in order to produce designs that fall into Mexico's standard regulations for reinforced concrete design, since the original model led in some cases to inconsistent designs.

4 The Optimal Design Model

A schematic section of a rectangular singly reinforced concrete beam is shown in Figure 1. The cost per unit length of the beam will be given by the following expression [5]:

$$y(x) = c_1 x_1 + c_2 x_2 x_3 + c_3 x_2 + c_4 x_3 \qquad (2)$$

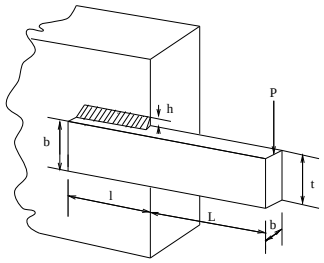

Figure 1: Schematic section of a singly reinforced rectangular beam. Taken from [4].

where $y(x)$ is the cost per unit length of the beam ($\$/cm$), c_1 is the cost coefficient due to volume of tensile steel reinforcement in the beam ($\$/cm^3$), c_2 is the cost coefficient due to volume of concrete in the beam ($\$/cm^3$), c_3 is the cost coefficient due to shuttering along the vertical surfaces of the beam ($\$/cm^2$), c_4 is the cost coefficient due to shuttering along the bottom horizontal surface of the beam ($\$/cm^2$), x_1 is the variable giving the area of tensile steel reinforcement as shown in Figure 1 (cm^2), x_2 is the variable giving the depth of the beam as shown in Figure 1 (cm) and x_3 is the variable giving the width of the beam, as shown in Figure 1 (cm).

The variables x_1, x_2 and x_3 not only affect the cost of a beam, but will also determine its moment of resistance. Since x_1 may be calculated if we know x_2 and x_3 [6], we'll propose different values for these two variables so that the total cost of the beam is minimum, verifying at the same time that our section has a proper resistant moment. Then, our optimal design model is the following:

minimize: $f(x) = c_1 x_1 + c_2 x_2 x_3 + c_3 x_2 + c_4 x_3$
subject to:

$$a_1 x_1^{-1} x_3 x_5 < 1 \quad (equilibrium\ constraint) \tag{3}$$

$$a_2 x_4^{-1} + a_3 x_2 x_3 x_4^{-1} < 1$$
$$(bending\ moment\ compatibility\ constraint) \tag{4}$$

$$0.25 \leq x_3/x_2 \leq 0.6$$
$$(width - height\ ratio\ constraint) \tag{5}$$

$$Q(x_2 - a_5 x_5)(f_r f_c^{'} x_5 x_3 + x_1 f_y) a_5 / x_4 \geq 1$$
$$(acting\ moment\ constraint) \tag{6}$$

$$a_6 / x_3 < 1 \quad (minimum\ width\ constraint) \tag{7}$$

$$x_1, x_2, x_3, x_4, x_5 > 0 \quad (non - negativity\ constraint) \tag{8}$$

Here x_4 is a variable defining the total applied bending moment including the bending moment due to self-weight of the beam; x_5 is a variable defining the depth of the equivalent rectangular stress block. Additionally, we have the following formulas:

$$c_1 = w_s \times c_s \quad (\$/cm^3) \tag{9}$$

where $w_s = 0.00785\ kg/cm^3$ (assumed value) is the unit weight of steel reinforcement, and c_s is the unit cost of steel reinforcement ($\$/kg$).

$$c_2 = (1 + r) c_c \times 10^{-6} \quad (\$/cm^3) \tag{10}$$

where c_c is the unit cost of concrete ($\$/m^3$) and r is the cover ratio.

$$c_3 = 2(1 + r) c_r \times 10^{-4} \quad (\$/cm^2) \tag{11}$$

where c_r is the unit cost of shuttering ($\$/m^2$).

$$c_4 = c_r \times 10^{-4} \quad (\$/cm^2) \tag{12}$$

$$a_1 = 0.85 f_c^{'} / f_y \tag{13}$$

where f_y is the yield strength of steel reinforcement (N/cm^2) and $f_c^{'}$ is the compressive strength of concrete (N/cm^2).

$$a_3 = D(1 + r) w_c k L^2 \tag{14}$$

where $D = 1.4$ (assumed) is the load factor for dead load, $w_c = 0.0228\ N/cm^3$ is the unit weight force of concrete, k is the moment coefficient for the design section ($= 1.8$ for simply supported beam) and L is the span of the beam (cm).

$$a_4 = 1 / (f_r Q f_c^{'}) \tag{15}$$

where Q is the capacity reduction factor ($= 0.90$ for flexure) and $f_r = 0.85$ (assumed) is the reduction factor of concrete. Also, a_2 is the applied bending moment ($N - cm$), $a_5 = \frac{1}{2}$ (assuming the centroid of

compressive force at half the depth of equivalent rectangular stress block), and a_6 is the minimum acceptable width of the beam.

To determine x_4 (total bending moment, including self-weight of the beam), we use:

$$x_4 = a_2 + a_3 x_2 x_3 \tag{16}$$

To calculate x_1 (area of reinforcement steel), we use:

$$x_1 = \omega x_2 x_3 f_c^{'} / f_y \tag{17}$$

where

$$\omega = \frac{1 - \sqrt{1 - \frac{4(0.59)x_4}{0.9 x_3 x_2^2 f_c^{'}}}}{1.18} \tag{18}$$

This last expression can be derived from equation (1). Finally, x_5 (depth of the equivalent stress block) is given by:

$$x_5 = x_1 / (a_1 x_3) \tag{19}$$

5 Use of Genetic Algorithms

To solve this optimization problem, we used a Turbo Pascal implementation of the Simple Genetic Algorithm (SGA) proposed by David Goldberg [3], and we experimented with several representation schemes. We have previously used binary representation [14] and we have tried Gray coding [15] for structural optimization problems with a continuous search space like this one. For this particular application, we decided to experiment also with floating point representation. We won't talk much about the genetic algorithm (GA), since we have done so in previous publications [16] [17] [18]. Instead, we'll give some details about the different representation schemes that we used in our experiments.

The traditional representation used by the genetic algorithms community is the binary scheme according to which a chromosome is a string of the form $\langle b_1, b_2, \ldots, b_m \rangle$, where $b_1, b_2, \ldots, b_m$ are called *allele* (either zeros or ones). Since the binary alphabet offers the maximum number of schemata per bit of information of any coding [3], 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 [19] it may be 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. Accoding 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). 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 [19]. 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. However, it is possible to reduce such discrepancy by using Gray coding. The Gray code representation has the property that any two points next to each other in the problem space differ by only one bit [19]. 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. Finally, we should mention that we used a two-point crossover, and binary tournament selection in all our tests. The only operator that had to be redefined was *mutation*, which in the floating point representation consisted on selecting a random number between 0 and 9. Our fitness function was given by equation (2), using a penalty function of the form $fitness = 1/(cost * (v * 500 + 1))$ where v depends on the number of constraints violated, and 500 was a value derived experimentally. Whenever the design doesn't violate any constraint, the fitness function is just the inverse of the cost (the GA only maximizes, and we required a minimization in this case).

6 Examples

The following example was taken from Everard and Tanner [6]: Design a least-cost reinforced concrete rectangular beam simply supported over a span of 10 m supporting a uniform dead load of 15 kN/m and a uniform live load of 20 kN/m. The concrete strength $f_c^{'} = 30\ MPa$ and the steel yield strength $f_y = 300\ MPa$. The unit cost of steel (CS), concrete (CC) and shuttering (CSH) are \$ $0.72/kg$, \$ $64.5/m^3$ and \$ $2.155/m^2$, respectively. Assume a cover ratio (r) of 0.10, unit weight of concrete of 2323 kg/m^3 and capacity reduction factor as 0.90.

The ultimate uniform load is

$$= 1.4 \times 15 + 1.7 \times 20 = 55\ kN/m.$$

The ultimate applied bending moment is

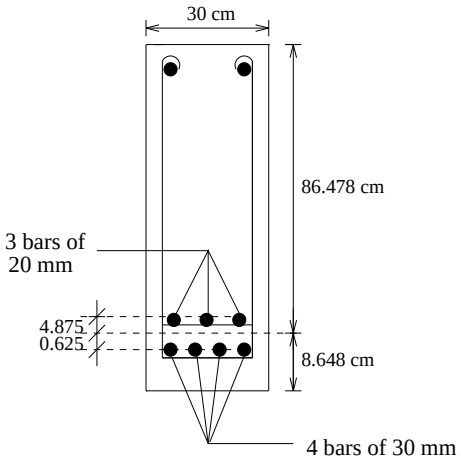

Figure 2: Optimum design of the beam of the first example.

Parameter	**Chakrabarty**	**GA (B)**	**GA (G)**	**GA (FP)**
x_1 (cm^2)	37.6926	36.1893	41.5905	37.5205
x_2 (cm)	86.0629	89.5402	78.6177	86.4776
x_3 (cm)	30.0000	30.0162	30.0447	30.0022
cost $(\$/cm)$	0.4435	0.4442	0.4464	0.4436

Table 1: Comparison of the geometric programming approach used by Chakrabarty [4] and the GA using binary (B), Gray coding (G) and floating point (FP) representation.

$$= 55 \times 10^2/8 = 687.5\ kN, \quad m = 687.5 \times 10^5\ N - cm.$$

Using this information, we can get the values of the cost coefficients and the other model constants:

$$\begin{array}{ll} c_1 = 0.0056520 & c_2 = 0.00007095 \\ c_3 = 0.00047410 & c_4 = 0.00021550 \\ a_1 = 0.08500 & a_2 = 68'750,000 \\ a_3 = 438'233,950 & a_4 = 0.00043573 \\ a_5 = 0.50 & a_6 = 30.00 \end{array}$$

Our results and their comparison with the geometric programming method used by Chakrabarty [5] are shown in Table 1. As we can see, the floating point representation produced the best results and Gray coding the worst. Our final design for this problem is shown in Figure 2, and has a total height of 95.125, which is about 1% more than Chakrabarty's design. This slight difference is due to the fact

Parameter	Chakrabarty	GA (FP)
b (cm)	20	20
x_1 (cm^2)	31.1267	39.5412
x_2 (cm)	101.5494	82.7043
x_3 (cm)	20.000	20.6825
cost $(\$/cm)$	0.3725	0.3885
b (cm)	40	40
x_1 (cm^2)	43.6017	43.7644
x_2 (cm)	76.1499	75.9102
x_3 (cm)	40.000	40.0042
cost $(\$/cm)$	0.5073	0.5074
b (cm)	62.5	62.5
x_1 (cm^2)	55.4435	35.7172
x_2 (cm)	62.5974	104.3223
x_3 (cm)	62.500	62.5010
cost $(\$/cm)$	0.6341	0.7274

Table 2: Comparison of the geometric programming approach used by Chakrabarty [4] and the GA using floating point representation.

that Chakrabarty's model considers the area of reinforcement steel as a variable, even when this is a parameter that depends on the beam section, and can't take any arbitraty value. On the other hand, our costs of steel, concrete and shuttering represent the 47.80%, 41.50% and 10.70% of the total cost, which corresponds almost exactly to the costs obtained by Chakrabarty. Floating point represention was used in all the further experiments, since it provided the best results overall. It should be noticed that our model has more constraints than Chakrabarty's model, in order to make it more realistic. For example, we require the relation x_3/x^2 to be between 0.25 and 0.60 which is a common recipe used by civil engineers in practice. The reason for this is not purely empirical. These limits allow us to have a "reasonable" amount of reinforcement steel in our designs, so that we can guarantee a good adherence between steel and concrete, and we can provide a good control of the beam's deflection. Since Chakrabarty doesn't impose this constraint in his model, some of the results shown next will violate it.

First, we'll perform an analysis similar to that conducted by Chakrabarty, experimenting with different values of b. The results of our tests are shown in Table 2. For the case in which $b = 62.50$

Chakrabarty's model produces a design 42.98% more expensive than when $b = 30$. Our design is 63.98% more expensive. However, Chakrabarty's design violates the restriction imposed by equation (5). Therefore, in practice an engineer would prefer our design even when it's more expensive, for the reasons previously exposed. In all the remaining examples, it will always be the case that when our results are not equals to those produced by Chakrabarty's model (or almost equal, should we say, since there is always a difference in the last digit due to rounding-off errors) it's because his design is violating some constraint—normally that defined by equation (5)—.

Finally, we tested different values for the costs of reinforcement steel, concrete and shuttering. The results are shown in Table 3. Again, the discrepancies between our results and those produced by Chakrabarty's method will indicate some violation of the constraints imposed by our model.

7 Selecting the Parameters of the GA

One of the main problems when using GAs in optimization problems is how to choose the most appropriate parameter values (i.e., population size, maximum number of generations, and 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 turned out to be much harder to fine tune the parameters of the GA when a floating point representation scheme was used. To deal with this problem, we used a systematic process that seems to be able to generate optimal (or at least sub-optimal) solutions in a very short period of time and with minimal human intervention. We set the population size and the maximum number of generations as constants (400 chromosomes and 50 generations). Then, we ran a nested loop in which the crossover rate and the mutation rate went from 0.1 to 0.9 at increments of 0.1. For each run, we updated 2 files. One contained only the final costs, and the other had a summary that included, besides the cost, the corresponding values of the design parameters and the mutation and crossover rates used. Also, a flag indicating if any constraint was violated or not. When the whole process ended up, the file with the costs was sorted in ascending order, and the smallest value was searched in the other file. If that cost didn't violate any constraint, then the design parameters corresponding to that cost were printed as the final answer. Otherwise, the next cost was taken and searched in the auxiliary file, repeating the same process until a cost that didn't violate any constraint was found. In practice, we normally never

Parameter	**Chakrabarty**	**GA (FP)**
b (cm)	40	40
$CS = 0.36$	$CC = 64.5$	$CSH = 2.155$
x_1 (cm^2)	57.0072	50.2583
x_2 (cm)	59.8678	66.7029
x_3 (cm)	40.000	40.0033
$cost$ $(\$/cm)$	0.3680	0.3716
$CS = 1.08$	$CC = 64.5$	$CSH = 2.155$
x_1 (cm^2)	37.2006	37.0318
x_2 (cm)	89.5455	90.0205
x_3 (cm)	40.000	40.0010
$cost$ $(\$/cm)$	0.6206	0.6207
$CS = 1.44$	$CC = 64.5$	$CSH = 2.155$
x_1 (cm^2)	33.2691	33.2279
x_2 (cm)	101.1724	101.3565
x_3 (cm)	40.000	40.0001
$cost$ $(\$/cm)$	0.7198	0.7199
$CS = 0.72$	$CC = 32.25$	$CSH = 2.155$
x_1 (cm^2)	33.0372	35.0698
x_2 (cm)	95.5279	95.4719
x_3 (cm)	40.000	40.0001
$cost$ $(\$/cm)$	0.3875	0.3876
$CS = 0.72$	$CC = 129.0$	$CSH = 2.155$
x_1 (cm^2)	55.4240	49.9278
x_2 (cm)	61.2698	67.0981
x_3 (cm)	40.000	40.0050
$cost$ $(\$/cm)$	0.6987	0.7035
$CS = 0.72$	$CC = 64.5$	$CSH = 1.0775$
x_1 (cm^2)	42.3510	42.5568
x_2 (cm)	78.3650	78.0625
x_3 (cm)	40.000	40.0001
$cost$ $(\$/cm)$	0.4847	0.4848
$CS = 0.72$	$CC = 64.5$	$CSH = 4.31$
x_1 (cm^2)	45.9454	45.9012
x_2 (cm)	72.4085	72.5103
x_3 (cm)	40.000	40.0003
$cost$ $(\$/cm)$	0.5511	0.5512

Table 3: Comparison of the geometric programming approach used by Chakrabarty [4] and the GA using floating point representation.

had to search more than twice for any given minimum cost. 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 15 seconds in a PC DX/2 running at 66 MHz and with a mathematical coprocessor).

Future Work and Conclusions

Even when we already have some concrete results in this research, a lot of work remains to be done. For example, we are currently exploring other techniques for adjusting the parameters of the GA, such as fuzzy logic. Also, we are interested on doing a theoretical analysis of the search space of this optimization problem, so that we can devise some strategies to solve it more efficiently. Nevertheless, our current results are very promising, and the system has called the attention of more than one engineer both in the academia and the building industry in México.

We have been working in the use of GAs for structural optimization problems during the last two years, and so far, we have implemented systems to generate optimal designs of beams, columns and plane and space trusses. However, our final goal is to develop a complete structural optimization system that uses GAs, and that probably incorporates also the traditional mathematical programming techniques available, together with some other powerful heuristics such as tabu search and simulated annealing. Such a system intends to be a very powerful tool for computer aided structural design that will allow to reduce costs without sacrificing safety.

References

[1] Belegundu, A. D., *A Study of Mathematical Programming Methods for Structural Optimization.* PhD dissertation, University of Iowa, Dept. of Civil and Environmental Engineering, 1982.

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

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

[4] Chakrabarty, B. K., "A model for optimal design of reinforced concrete beam," *Journal of Structural Engineering*, vol. 118, pp. 3238–3242, nov 1992.

[5] Chakrabarty, B. K., "Model for optimal design of reinforced concrete beams," *Computers and Structures*, vol. 42, no. 3, pp. 447–451, 1992.

[6] Everard, N. J. and III, J. L. T., *Theory and Problems of Reinforced Concrete Design*. McGraw-Hill Book Company, second ed., 1987.

[7] Galilei, G., *Dialogues Concerning Two New Sciences*. Evanston, Ill. Northwestern University Press, 1950. Originally published in 1665.

[8] Haug, E. J., *Minimum Weight Design of Beams with Inequality Constraints on Stress and Deflection*. Department of mechanical engineering, Kansas State University, 1966.

[9] Haug, E. J. and Kirmser, P. G., "Minimum weight design of beams with inequality constraints on stress and deflection," *Journal of Applied Mechanics. Transactions of the ASME*, pp. 999–1004, dec 1967.

[10] Venkayya, V. B., "Design of optimum structures," *Computers and Structures*, vol. 1, pp. 265–309, 1971.

[11] Osyczka, A., *Multicriterion Optimization in Engineering with FORTRAN programs*. Ellis Horwood Limited, 1984.

[12] Osyczka, A., "Multicriteria optimization for engineering design," in *Design Optimization* (Gero, J. S., ed.), pp. 193–227, Academic Press, 1985.

[13] Prakash, A., Agarwala, S. K., and Singh, K. K., "Optimum design of reinforced concrete sections," *Computers and Structures*, vol. 30, no. 4, pp. 1009–1011, 1988.

[14] Coello, C. A. C., "Discrete optimization of trusses using genetic algorithms," in *EXPERSYS-94. Expert Systems Applications and Artificial Intelligence* (Chen, J., Attia, F. G., and Crabtree, D. L., eds.), (Houston, Texas), pp. 331–336, I.I.T.T. International. Technology Transfer Series, nov 1994.

[15] Coello, C. A. and Christiansen, A. D., "Using genetic algorithms for optimal design of axially loaded non-prismatic columns," Tech. Rep. TUTR-CS-95-101, Tulane University, jan 1995.

[16] Coello, C. A., "Uso de algoritmos genéticos para el diseño óptimo de armaduras," in *Congreso Nacional de Informática - 1994 Herramientas para los Mercados Globales*, (Mexico City, México), pp. 290–305, Fundación Arturo Rosenblueth, jun 1994. (in Spanish).

[17] Coello, C. A. C., "El algoritmo genético como alternativa a la programación dinámica," in *Actas del VIII Simposio Internacional en Aplicaciones de Informática*, (Antofagasta, Chile), pp. 151–157, Universidad Católica del Norte, nov 1994. (in Spanish).

[18] Coello, C. A. and Christiansen, A. D., "Optimization of truss designs using genetic algorithms," Tech. Rep. TUTR-CS-94-102, Tulane University, nov 1994.

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