DISEÑO OPTIMO DE VIGAS DE CONCRETO REFORZADO MEDIANTE ALGORITMOS GENÉTICOS

Carlos A. Coello Coello **Filiberto Santos Hernández** **Francisco A. Alonso Farrera**

Escuela de Ingeniería Civil
Universidad Autónoma de Chiapas
México

Teléfono : **52+(961) 5-03-22**
Fax : **52+(961) 5-05-27**

Dirección : **Escuela de Ingeniería Civil**
Boulevard Belisario Domínguez km. 1081
Apartado Postal 61
C.P. 29000
Tuxtla Gutiérrez, Chiapas
México

E-mail : **coello@eecs.tulane.edu**

DISEÑO OPTIMO DE VIGAS DE CONCRETO REFORZADO MEDIANTE ALGORITMOS GENÉTICOS

Resumen

En este artículo se aborda el diseño óptimo de columnas de concreto reforzado haciendo uso de una técnica de inteligencia artificial basada en los mecanismos de la selección natural, y que se conoce como el algoritmo genético. Se propone un modelo de optimización que parece adecuado para aplicaciones del mundo real, y que fue ideado en base a las necesidades y regulaciones específicas de nuestro país. Debido a que el algoritmo genético empleado hace uso de representación de punto flotante, y a la alta no-convexidad del espacio de búsqueda de este problema, garantizar convergencia se convirtió en una de las prioridades principales de esta investigación. De tal forma, se desarrolló una metodología que permite eliminar el ajuste de parámetros del algoritmo genético (i.e., tamaño de población, porcentajes de mutación y cruza, y número máximo de generaciones) y se incorporó en un sistema que genera diseños óptimos de vigas de concreto en tiempos muy cortos y con un alto grado de confiabilidad. Un prototipo de dicho sistema se encuentra actualmente en período de prueba, a fin de que pueda emplearse como una herramienta seria de diseño en aplicaciones del mundo real dentro de poco tiempo.

1. Introducción

El proceso tradicional de diseño en ingeniería suele ser muy tedioso: basado en la información que posee sobre el problema en estudio, el ingeniero propone un diseño inicial, el cual es posteriormente corroborado mediante las técnicas de análisis matemático disponible. Si el diseño es válido, entonces se adopta, y si no lo es, se procede a modificar ciertos parámetros del diseño y se vuelve a analizar. En este proceso de ensayo y error es donde el ingeniero gana experiencia, si bien es a un precio muy alto en términos de tiempo y esfuerzo (ver Figura 1). Debido a que el tiempo es siempre una limitante en el mundo real, suele adoptarse una solución sub-óptima en la mayor parte de los casos. El advenimiento de la computadora ha hecho posible ayudar a los ingenieros a automatizar este proceso. Sin embargo, su uso se ha concentrado principalmente en efectuar los tediosos cálculos matemáticos que se requieren, pero no en el proceso de diseño en sí.

Por otro lado, en los últimos años ha cobrado gran impulso una rama de la ingeniería conocida como "optimización", en la cual los problemas de diseño se reformulan en base a una o más funciones objetivo, la cual se quiere minimizar o maximizar, mientras se le sujeta a una serie de restricciones (ver Figura 2). Técnicas de programación matemática suelen ser la herramienta básica de los ingenieros que trabajan en esta área, y un gran número de heurísticas se han desarrollado para enfrentar la alta no-linealidad y no-convexidad de la mayoría de los problemas de diseño [1], que se caracterizan por tener un gran número de mínimos locales.

Este artículo se enfoca en el uso de una técnica de inteligencia artificial basada en los mecanismos de la selección natural, y que se conoce como el *algoritmo genético* (AG) [16][15]. El proceso de diseño basado en esta técnica es muy similar al proceso de diseño óptimo previamente mostrado (ver Figura 3). La diferencia principal es la noción de una *función de aptitud* que sustituye a la función objetivo, y al hecho de que las modificaciones realizadas al diseño no dependen ni del ingeniero ni del gradiente de la función objetivo, como en los dos casos anteriores. Un aspecto interesante del AG es que los diseños que genera inicialmente son completamente aleatorios, sin que haya intervención humana alguna, no obstante lo cual la técnica converge a un diseño óptimo en una cantidad razonable de tiempo.

El diseño de vigas de concreto reforzado juega un papel primordial en la ingeniería civil mexicana, por su uso tan extendido en nuestro país. Tradicionalmente este proceso se efectúa de forma

iterativa, tal y como se ilustra en la Figura 1, asumiendo el peso propio de la viga a la que se le desea diseñar la sección. Posteriormente, se determina su momento resistente para checar si corresponde al momento flexionante aplicado. Este proceso se repite una y otra vez, gastándose una cantidad considerable de tiempo, hasta que se encuentra una sección apropiada. En dicho proceso suele ser difícil hacer que el momento resistente de la sección corresponda con el momento actuante total aplicado, incluyendo el producido por el peso propio de la viga, el cual puede ser substancial en muchos casos. De tal forma, el diseño de una viga no sólo es lento, sino que además tiene una carencia total de economía, puesto que lo único que nos interesa es encontrar cualquier sección que se acomode a las condiciones establecidas, sin siquiera considerar la posibilidad de hacerla lo más barata posible.

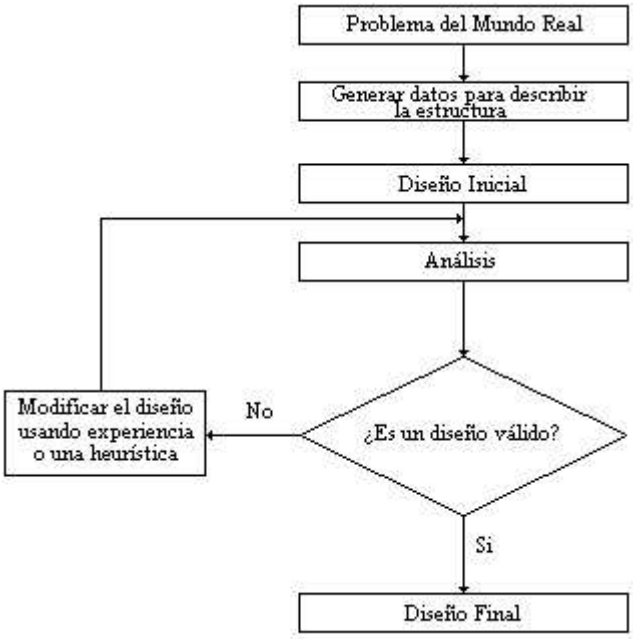

Figura 1 : *Proceso tradicional de diseño*

En este trabajo se presenta un modelo de diseño óptimo de vigas de concreto reforzado, el cual trata de minimizar el costo de una viga considerando no sólo los esfuerzos permisibles por el elemento, sino también los costos del concreto, el acero y el recubrimiento utilizados. Nuestro modelo de diseño óptimo se basa en el propuesto por Chakrabarty [2,3], aunque incluye ciertas modificaciones (i.e., restricciones adicionales) que lo hacen más apropiado para aplicaciones prácticas. En la siguiente sección introduciremos algunos conceptos generales de diseño de concreto reforzado. Después, mostraremos nuestro modelo y describiremos el uso del algoritmo genético. Finalmente, presentaremos los resultados reportados usando nuestro modelo al aplicarlo a ciertos problemas encontrados en la literatura, y discutiremos brevemente algunas de las dificultades encontradas al tratar de aplicar el algoritmo genético a este problema de diseño óptimo.

2. Generalidades

Para fines de este trabajo, se adoptó el método de diseño por resistencia, el cual presenta las siguientes ventajas [11]:

Predice mejor la resistencia de una sección debido al hecho de que reconoce la no linealidad del diagrama de esfuerzo-deformación en los niveles elevados de esfuerzo.

Debido a que las cargas muertas a las cuales se sujeta la estructura se pueden determinar con mayor certeza que las cargas vivas, es poco razonable aplicar el mismo factor de seguridad a ambas. Por lo tanto, este método permite el uso de factores de seguridad distintos para cada una de ellas.

Las hipótesis básicas que se toman al utilizar el método de diseño por resistencia son las siguientes [11] :

Las secciones planas antes de la flexión permanecen planas después de ella.

Trabajando a capacidad última, la deformación y el esfuerzo no son proporcionales.

La deformación en el concreto es proporcional a la distancia del eje neutro.

Se ignora la resistencia a tensión del concreto en los cálculos de la flexión.

La deformación última del concreto es 0.003.

El módulo de elasticidad del acero de refuerzo es 29'000,000 psi (200,000 MPa).

El esfuerzo a compresión medio en el concreto es $0.85f'_c$.

El esfuerzo a tensión medio en el refuerzo no excede el valor de f_y.

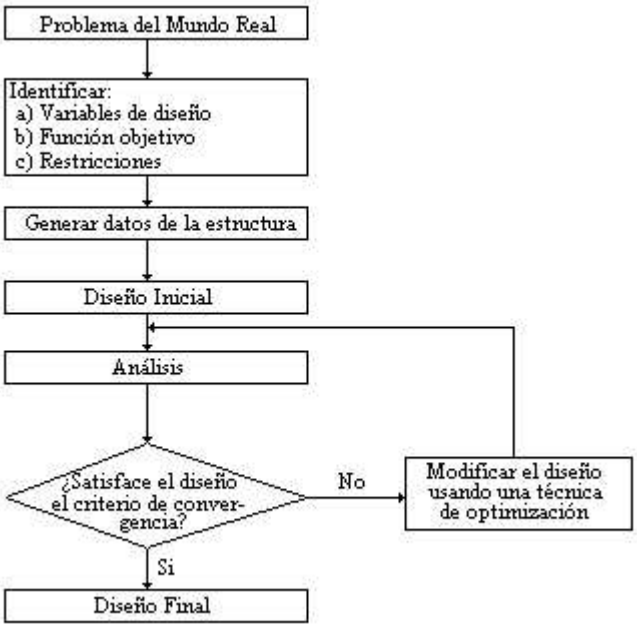

Figura 2 : *Proceso de diseño óptimo* [1]

De acuerdo a este método de diseño, la capacidad de Momento nominal M_n de una viga rectangular únicamente con refuerzo a tensión, está dado por [11]

$$M_n = bd^2 f'_c \, w \, (1 - 0.59 \, w) \qquad (1)$$

donde **b** es el ancho de la viga, **d** es la distancia de la fibra a compresión más alejada al centroide del refuerzo a tensión, $\mathbf{f'_c}$ es la resistencia a compresión del concreto, $\mathbf{w = (A_s f_y / bdf'_c)}$, $\mathbf{f_y}$ es la resistencia efectiva del refuerzo y $\mathbf{A_s}$ es el área del refuerzo a tensión.

Hay un número infinito de soluciones a la ecuación (1) que producen el mismo valor de M_n [4]. En el método tradicional, se asumen **b** y/o **d**, y en base a eso se calcula $\mathbf{A_s}$ y se itera hasta obtener una sección satisfactoria. Un problema obvio con esta metodología de diseño es que únicamente se puede evaluar un número limitado de secciones. Puesto que la ecuación (1) no incorpora ningún parámetro de costo, no hay forma de llegar a un diseño de costo mínimo. Eso plantea la necesidad de incluir parámetros que evalúen el costo de la viga en nuestro modelo de diseño óptimo.

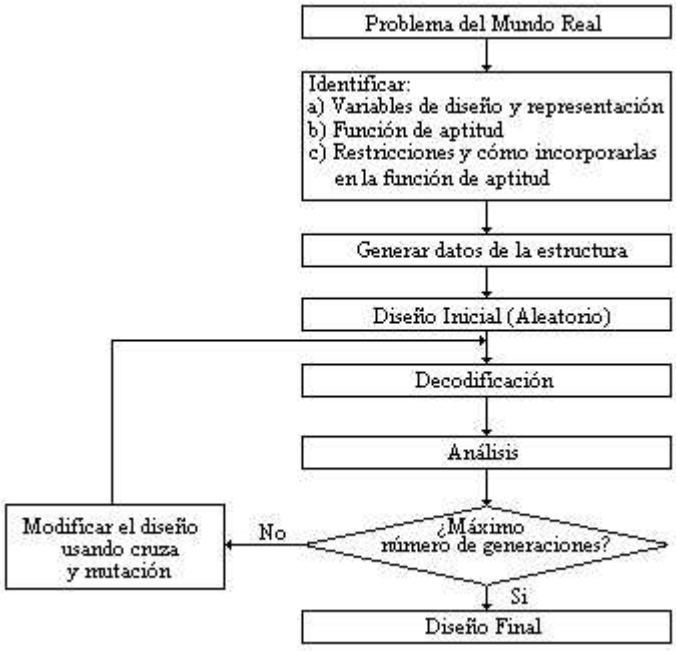

Figura 3 : *Proceso de diseño óptimo usando un Algoritmo Genético*

3. Trabajo Previo

El diseño óptimo de vigas ha sido abordado desde la época de Galileo [12], aunque no siempre con éxito. Al parecer, la tesis doctoral de E. J. Haug Jr. [10] (ver también [9]) en 1966 es uno de los primeros intentos modernos serios de aplicar equipo de cómputo para el diseño óptimo de vigas. El trabajo de Haug consistió en reducir el problema de diseño óptimo no lineal a uno de Lagrange en el Cálculo de Variaciones. Su modelo incluye restricciones y pretende minimizar el peso de la viga en un buen número de situaciones diferentes.

Venkayya [21] desarrolló un método basado en un criterio energético y un procedimiento de búsqueda para el diseño de estructuras sujetas a cargas estáticas. El argumenta que su método puede manejar muy eficientemente: (a) diseños para múltiples condiciones de carga, (b) condiciones de esfuerzo, (c) restricciones en los desplazamientos, (d) restricciones en las dimensiones de los elementos. Su método ha sido aplicado éxitosamente al diseño de armaduras, marcos y vigas.

Osyczka [18, 19] ha aplicado métodos de optimización con criterios múltiples a problemas de diseño de vigas. Asimismo, Prakash et al. [20] propusieron un modelo para optimizar el diseño de secciones de concreto reforzado en el cual consideraron los costos del acero, el concreto y el recubrimiento. El modelo de Chakrabarty [2,3] presenta algunas similitudes con el de Prakash et al.,

aunque el primero es más completo y detallado, lo que facilitó su uso. Sin embargo, cabe mencionar que nosotros modificamos este modelo ligeramente a fin de obtener resultados más coherentes con las normas estándares de diseño de elementos de concreto vigentes en México, ya que en su forma original nos arrojaba diseños con ciertas inconsistencias.

4. Modelo de Diseño Optimo

Una sección esquemática de una viga de concreto rectangular simplemente reforzada se muestra la Figura 4. El costo por unidad de longitud de la viga está dado por la siguiente expresión:

$$y(x) = c_1x_1 + c_2x_2x_3 + c_3x_2 + c_4x_3 \qquad (2)$$

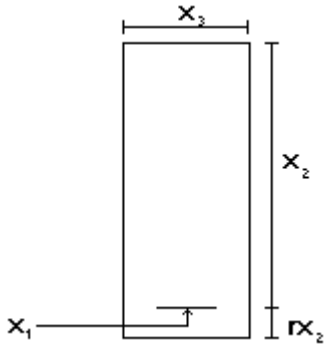

Figura 4 : *Sección esquemática de una viga rectangular simplemente reforzada.*

donde **y(x)** es el costo por unidad de longitud de la viga ($/cm), $\mathbf{c_1}$ es el coeficiente de costo debido al volumen de acero de refuerzo en la viga ($/cm^3), $\mathbf{c_2}$ es el coeficiente de costo debido al volumen de concreto en la viga ($/cm^3), $\mathbf{c_3}$ es el coeficiente de costo debido al recubrimiento a lo largo de la superficie horizontal inferior de la viga ($/cm^2), $\mathbf{x_1}$ es la variable que representa el área de acero de refuerzo tal y como se indica en la Figura 1 (cm^2), $\mathbf{x_2}$ es la variable que representa la altura de la viga, tal y como se indica en la Figura 1 (cm), y $\mathbf{x_3}$ es la variable que representa el ancho de la viga, tal y como se indica en la Figura 4 (cm). Las variables $\mathbf{x_1}$, $\mathbf{x_2}$ y $\mathbf{x_3}$ no sólo afectan el costo de la viga, sino que también determinan su momento resistente. Dado que $\mathbf{x_1}$ puede determinarse a partir de $\mathbf{x_2}$ y $\mathbf{x_3}$ [11], debemos proponer los valores de estas últimas 2 variables, de tal forma que se minimice el costo total de la viga al mismo tiempo que se obtiene un momento resistente adecuado. De tal forma, nuestro modelo de diseño óptimo será el siguiente:

minimizar: $f(x) = c_1x_1 + c_2x_2x_3 + c_3x_2 + c_4x_3$

sujeta a :

$a_1x_1^{-1}x_3x_5 < 1$ (restricción de equilibrio) (3)

$a_2x_4^{-1} + a_3x_2x_3x_4^{-1} < 1$ (restricción de compatibilidad de momentos) (4)

$0.25 \ x_3/x_2 \ 0.6$ (restricción de la relación de ancho y altura) (5)

$Q(x_2-a_5x_5)(f_rf'_cx_5x_3+x_1f_y)a_5/x_4 \ 1$ (restricción de momento actuante) (6)

$a_6/x_3 < 1$ (restricción de ancho mínimo de la viga) (7)

$$x_1, x_2, x_3, x_4, x_5 > 0 \quad \text{(restricción de no negatividad)} \qquad (8)$$

Aquí $\mathbf{x_4}$ es una variable que define el momento flexionante actuante total, incluyendo el momento flexionante debido al peso propio de la viga; $\mathbf{x_5}$ es una variable que define la altura del bloque rectangular de esfuerzo equivalente. Adicionalmente, tenemos las siguientes fórmulas:

$$c_1 = w_s \text{ x } c_s \ (\$/cm^3) \qquad (9)$$

donde $\mathbf{w_s}$ = peso unitario del acero de refuerzo (kg/cm^3) = 0.00785 kg/cm^3 (se asume este valor).
$\mathbf{c_s}$ = costo unitario del acero de refuerzo ($/kg).

$$c_2 = (1 + r)c_c \text{ x } 10^{-6} \ (\$/cm^3) \qquad (10)$$

donde $\mathbf{c_c}$ = costo unitario del concreto ($/$m^3$).
r = porcentaje de recubrimiento.

$$c_3 = 2(1 + r)c_r \text{ x } 10^{-4} \ (\$/cm^2) \qquad (11)$$

donde $\mathbf{c_r}$ = costo unitario del recubrimiento ($/$m^2$).

$$c_4 = c_r \text{ x } 10^{-4} \ (\$/cm^2) \qquad (12)$$

$$a_1 = 0.85f'_c/f_y \qquad (13)$$

donde $\mathbf{f_y}$ = resistencia a tensión del acero de refuerzo (N/cm^2)
$\mathbf{f'_c}$ = resistencia a compresión del concreto (N/cm^2)

a_2 = momento flexionante aplicado (N-cm)

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

donde **D** = factor de cargas muertas (se asume igual a 1.4)
$\mathbf{w_c}$ = peso unitario del concreto (N/cm^3)
k = coeficiente de momento para la sección de diseño
(= 1.8 para una viga simplemente apoyada)
L = longitud de la viga (cm)

$$a_4 = 1/(f_r\ Q\ f'_c) \qquad (15)$$

donde $\mathbf{f_r}$ = factor de reducción de la resistencia del concreto
Q = factor de la capacidad de reducción (= 0.90 para flexión)

a_5 = 1/2 (asumiendo que el centroide de la fuerza de compresión se encuentra a la mitad de la altura del bloque de esfuerzo rectangular equivalente)

a_6 = ancho mínimo aceptable de la viga

Para calcular x_4 (momento actuante total, que incluye el peso propio de la viga), se usa:

$$x_4 = a_2 + a_3\ x_2\ x_3 \qquad (16)$$

Para calcular x_1 (área de acero de refuerzo) se usa :

$$x_1 = \omega\, x_2\, x_3\, f'_c / f_y \qquad (17)$$

donde $$\omega = \frac{\left(1 - \sqrt{1 - \frac{4(0.59)\, x_4}{0.9\, x_3\, x_2^2\, f'c}}\right)}{1.18}$$

Esta última expresión puede derivarse de la Ecuación (1).

Finalmente, x_5 (profundidad del bloque de esfuerzo rectangular equivalente) está dada por:

$$x_5 = x_1/(a_1\, x_3) \qquad (18)$$

5. Empleo de los Algoritmos Genéticos para resolver el modelo óptimo

Para este problema se utilizó una implementación en Turbo Pascal del algoritmo genético simple propuesto por Goldberg [15], experimentando con varios esquemas de representación. En trabajos anteriores hemos utilizado representación binaria [7] y hemos experimentado con códigos de Gray [6] para los casos en que representamos un espacio de búsqueda continuo como éste. Para esta aplicación en particular decidimos experimentar con representación binaria con y sin códigos de Gray, y con representación de punto flotante. Omitiremos las generalidades de la técnica, ya que hemos incluído esa información en trabajos previos [4, 8, 5]. De tal forma, procederemos a discutir los detalles de los 3 esquemas de representación con los que experimentamos, y las modificaciones que se hicieron al algoritmo genético simple para darles cabida.

La representación tradicional utilizada en los algoritmos genéticos ha sido la binaria, de acuerdo a la cual un cromosoma es un número binario de la forma $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ donde $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_m$ son llamados alelos (ceros o unos). Debido a que el alfabeto binario ofrece el número máximo de cadenas posibles por bit de información de cualquier otra codificación [15], su uso se ha vuelto cosa común en la comunidad científica. Esta codificación también facilita el análisis teórico y permite el uso de operadores genéticos elegantes. Sin embargo, debido a que la propiedad de ' paralelismo implícito' con que cuentan los algoritmos genéticos no depende del uso de cadenas de bits [17] vale la pena experimentar con alfabetos más grandes y (posiblemente) con nuevos operadores genéticos. En particular, para los problemas de optimización de parámetros con variables sobre dominios continuos, podemos experimentar con una representación de punto flotante, en la que cada cromosoma es una cadena $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m$ donde $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m$ son dígitos (números entre cero y nueve). Una de las ventajas de la representación de punto flotante es que dos puntos cercanos en el espacio de representación deben estar también cercanos en el espacio del problema, y viceversa [17]. Esto no es necesariamente cierto cuando se usa la representación binaria, porque la distancia en una representación se define normalmente mediante el número de posiciones de bits diferentes. Sin embargo, es posible reducir tal discrepancia usando códigos de Gray.

Los procedimientos para convertir un número binario $\mathbf{b} = \mathbf{b}_1, \ldots, \mathbf{b}_m$ en un código de Gray $\mathbf{g} = \mathbf{g}_1, \ldots, \mathbf{g}_m$ y viceversa, se presentan a continuación [17]; el parámetro *m* denota el número de bits en estas representaciones.

```
procedure Binario-a-Gray
begin
        g1 = b1
        for k = 2 to m do
            gk = bk-1 XOR bk
```

```
end

procedure Gray-a-Binario
begin
        value = g1
        b1 = value
        for k = 2 to m do
        begin
            if gk = 1 then value = NOT value
            bk = value
        end
end
```

La Tabla 1 muestra los primeros 16 números binarios junto con sus códigos de Gray correspondientes. Advierta que la representación mediante códigos de Gray tiene la propiedad de que dos puntos cualquiera que se encuentren próximos uno a otro en el espacio del problema difieren únicamente en un bit [17]. En otras palabras, un incremento unitario del valor del parámetro corresponde a un cambio de un solo bit en el código. Advierta también que hay otros procedimientos equivalentes para realizar conversiones entre binarios y códigos de Gray. Por ejemplo (caso de $m = 4$), el par de matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 01 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

proporciona las siguientes transformaciones:

$\mathbf{g} = A\mathbf{b}$ y $\mathbf{b} = A^{-1}\mathbf{g}$,

donde las multiplicaciones se efectúan módulo 2.

Binario	Código de Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Tabla 1 : Equivalencias entre números binarios y códigos de Gray.

La representación mediante códigos de Gray ha sido muy utilizada para reducir la distancia de dos puntos en el espacio de búsqueda del problema, y se dice que trae algunos beneficios debido a su propiedad de adyacencia, y a la pequeña perturbación causada por muchas mutaciones individuales.

Finalmente, debemos mencionar que usamos una cruza de dos puntos, y selección mediante torneo binario en todas nuestras pruebas. El único operador que tuvo que redefinirse fue el de la *mutación*, el cual en el caso de la representación de punto flotante consistió en un número aleatorio entre 0 y 9. Nuestra función de aptitud estuvo dada por la Ecuación 2, usando una función de penalización de la forma *aptitud = 1/(costo*(v*500+1))* donde *v* depende del número de restricciones violadas, y 500 fue un valor derivado experimentalmente. Cuando el diseño no viola ninguna restricción, la función de aptitud es simplemente la inversa del costo (el AG sólo maximiza, y lo que se requería en este caso era una minimización).

6. Ejemplo de Aplicación

El siguiente ejemplo fue tomado de Everard y Tanner [11]:

Diseñar una viga de concreto reforzado simplemente apoyada de costo mínimo, cuya longitud es de 10 m, y soporta una carga muerta uniforme de 15 kN/m y una carga viva uniforme de 20 kN/m. La resistencia del concreto es f'_c = 30 MPa, y la del acero es f_y = 300 MPa. El costo unitario del acero es de \$0.72/kg, el del concreto \$64.5/m^3 y el del recubrimiento \$2.155/m^2. Asumir un espesor de recubrimiento r = 0.10. El peso unitario del concreto es de 2323 kg/m^3, y el factor de reducción de capacidad es de 0.90.

La carga uniforme última es

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

El momento actuante último es

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

Usando los datos anteriores, los valores de los coeficientes de costo y las otras constantes del modelo son:

c_1 = 0.0056520,	c_2 = 0.00007095
c_3 = 0.00047410,	c_4 = 0.00021550
a_1 = 0.08500,	a_2 = 68' 750,000.00
a_3 = 438' 233,950	a_4 = 0.00043573
a_5 = 0.50,	a_6 = 30.00

Los resultados obtenidos se muestran en la Tabla 2. Como puede verse la representación de punto flotante produjo los mejores resultados, y la de códigos de Gray, los peores. Nuestro diseño final para este ejemplo se muestra en la Figura 5, y tiene una altura total de 95.125, lo cual es alrededor del 1% más que la del diseño de Chakrabarty. Esta pequeña diferencia se debe al hecho de que el modelo de Chakrabarty considera el área de refuerzo de acero como una variable, pese a que éste es un parámetro que depende de la sección de la viga, y no puede tomar cualquier valor arbitrario. Por otra parte, los costos del acero, el concreto y el recubrimiento representan el 47.80%, 41.50% y 10.70%, que corresponden casi de forma exacta con los obtenidos por Chakrabarty. La representación de punto flotante fue la única utilizada en todos los demás experimentos, puesto que es la que proporcionó los mejores resultados en general.

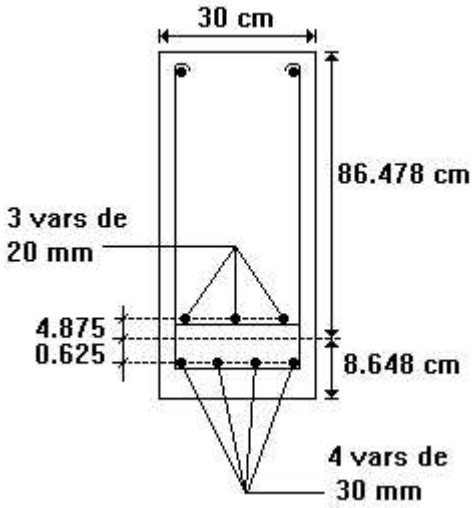

Figura 5 : *Diseño óptimo de la viga del primer ejemplo.*

Parámetro	Chakrabarty	AG (Binario)	AG (Gray)	AG (PF)
x_1 (cm²)	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
x_4 (N-cm)	80' 064,711.73	80' 540,242.0620	79' 111,846.5650	80' 131,661.9160
x_5 (cm)	14.7814	14.1842	16.2857	14.7128
Costo ($/cm)	0.4435	0.4442	0.4464	0.4436

Tabla 2 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) con representación binaria con y sin códigos de Gray, y con representación de punto flotante (PF).

Debe hacerse una importante observación antes de mostrar más ejemplos. Nuestro modelo tiene más restricciones que el de Chakrabarty para hacerlo más realista. Por ejemplo, nosotros requerimos que la relación de peraltes (x_3/x_2) se encuentre entre 0.25 y 0.6, lo cual es una práctica común usada por los ingenieros civiles. Esta restricción no es meramente empírica, sino que tiene un fundamento teórico. Estos límites nos permiten tener una cantidad "razonable" de acero de refuerzo en nuestros diseños, de tal forma que podamos garantizar una buena adherencia entre el acero y el concreto, y que podamos proporcionar un buen control de la deflexión de la viga. Puesto que Chakrabarty no impone esta restricción en su modelo, algunos de sus resultados mostrados a continuación la violarán.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	31.1267	39.5412
x_2 (cm)	101.5494	82.7043
x_3 (cm)	20.000	20.6825
Costo ($/cm)	0.3725	0.3885

Tabla 3 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF), para **b**=20 cm. Observe cómo el diseño de Chakrabarty no cumple con la restricción de relación de peraltes, porque x_3/x_2=0.20.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	43.6017	43.7644
x_2 (cm)	76.1499	75.9102
x_3 (cm)	40.000	40.0042
Costo ($/cm)	0.5073	0.5074

Tabla 4 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF), para **b**=40 cm.

Pasemos a efectuar parte del análisis que efectuó Chakrabarty para diferentes valores de **b**. Los resultados de nuestras pruebas se muestran en las Tabla 3, 4 y 5. Para el caso en el que **b**=62.50, el modelo de Chakrabarty produce un diseño que es un 42.98% más caro que cuando se usa una **b**=30. Nuestro diseño es 63.98% más costoso. Sin embargo, el diseño de Chakrabarty viola la restricción impuesta por la Ecuación 5. Por lo tanto, en la práctica un ingeniero prefiriría nuestro diseño, pese a ser más costoso, por las razones previamente expuestas. En todos los ejemplos restantes, será siempre el caso que cuando nuestros resultados no sean iguales a los producidos por el modelo de Chakrabarty (o casi iguales, debiéramos decir, puesto que hay siempre una diferencia en el último dígito debido a errores de redondeo) es porque su diseño está violando alguna restricción —normalmente la definida por la Ecuación (5)—.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	55.4435	35.7172
x_2 (cm)	62.5974	104.3223
x_3 (cm)	62.500	62.5010
Costo ($/cm)	0.6341	0.7274

Tabla 5 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=62.50 cm. Observe cómo en este caso la relación de peraltes producida por el diseño de Chakrabarty es casi 1.0, mientras que la nuestra es de 0.6, por lo que se ajusta a las especificaciones señaladas anteriormente.

Finalmente, experimentamos con una serie de costos diferentes para los costos del acero, el concreto y el recubrimiento. Los resultados pueden verse en las Tablas de la 6 a la 12. Podrá notarse cómo en los casos en que la solución de Chakrabarty se encuentra dentro de las restricciones de nuestro modelo, nuestros resultados son prácticamente los mismos, pero cuando no es así, nuestros diseños tienen un costo ligeramente superior aunque, por supuesto, poseen mayor valor práctico que los primeros.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	57.0072	50.2583
x_2 (cm)	59.8678	66.7029
x_3 (cm)	40.000	40.0033
Costo ($/cm)	0.3680	0.3716

Tabla 6 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=0.36, CC=64.5 y CSH=2.155. Nuevamente el diseño de Chakrabarty excede ligeramente la restricción de peraltes, y curiosamente tiene un área de acero mayor al nuestro, si bien su costo es 1% menor.

Parámetro	Chakrabarty	AG (PF)

x_1 (cm²)	37.2006	37.0318
x_2 (cm)	89.5455	90.0205
x_3 (cm)	40.000	40.0010
Costo ($/cm)	0.6206	0.6207

Tabla 7 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=1.08, CC=64.5 y CSH=2.155.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	33.2691	33.2279
x_2 (cm)	101.1724	101.3565
x_3 (cm)	40.000	40.0001
Costo ($/cm)	0.7198	0.7199

Tabla 8 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=1.44, CC=64.5 y CSH=2.155.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	33.0372	35.0698
x_2 (cm)	95.5279	95.4719
x_3 (cm)	40.000	40.0001
Costo ($/cm)	0.3875	0.3876

Tabla 9 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=0.72, CC=32.25 y CSH=2.155.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	55.4240	49.9278
x_2 (cm)	61.2698	67.0981
x_3 (cm)	40.000	40.0050
Costo ($/cm)	0.6987	0.7035

Tabla 10 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=0.72, CC=129.0 y CSH=2.155. La diferencia de costos se debe nuevamente a la restricción de relación de peraltes.

Parámetro	Chakrabarty	AG (PF)
x_1 (cm²)	42.3510	42.5568
x_2 (cm)	78.3650	78.0255
x_3 (cm)	40.000	40.0001
Costo ($/cm)	0.4847	0.4848

Tabla 11 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=0.72, CC=64.5 y CSH=1.0775.

Parámetro	Chakrabarty	AG (PF)

x_1 (cm²)	45.9454	45.9012
x_2 (cm)	72.4085	72.5103
x_3 (cm)	40.000	40.0003
Costo ($/cm)	0.5511	0.5512

Tabla 12 : Comparación del uso de programación geométrica (Chrakrabarty [2]) con el Algoritmo Genético (AG) usando representación de punto flotante (PF) para **b**=40 cm, CS=0.72, CC=64.5 y CSH=4.31.

7. Ajuste de los parámetros del Algoritmo Genético

Una de las cuestiones más difíciles de definir cuando se usa un algoritmo genético en un problema de optimización es la de cómo escoger el valor más adecuado de los parámetros que se utilizan (i.e., tamaño de la población, número máximo de generaciones, porcentaje de cruza y porcentaje de mutación). Este es normalmente un proceso que involucra cierto ensayo y error hasta completarse. Una de las experiencias más valiosas derivadas de este estudio es que descubrimos que resulta mucho más difícil ajustar los parámetros del AG cuando se usa la representación de punto flotante. Confrontamos, de tal manera, un dilema: la representación de punto flotante producía los mejores resultados, pero también era la que parecía más difícil de manejar en términos del ajuste de parámetros. Obviamente cualquier sistema de optimización no será de gran utilidad si sus salidas son completamente impredecibles. Después de una intensa experimentación, desarrollamos una metodología que parece ser capaz de generar soluciones óptimas (o al menos sub-óptimas) en un período de tiempo muy corto. Sin embargo, no tenemos todavía suficiente soporte teórico de su confiabilidad, aunque la evidencia empírica es bastante sólida. El método es el siguiente:

- Se escoge un valor fijo para la semilla de números aleatorios (0.35 en nuestro caso, aunque puede ser cualquiera).
- Se mantienen constantes el tamaño de la población y el número máximo de generaciones (400 cromosomas y 50 generaciones, respectivamente en nuestro caso).
- Ciclamos el porcentaje de mutación de 0.1 a 0.9 en incrementos de 0.1 dentro del ciclo del porcentaje de cruza que abarca exactamente los mismos valores. Esto implica realizar 81 corridas de prueba.
- Por cada corrida completa se generan 2 archivos. Uno contiene sólo los costos, y el otro contiene un resumen que incluye, además del costo, los parámetros correspondientes (i.e., las variables de diseño correspondientes) y los porcentajes de cruza y mutación en que ocurrió.
- Una vez concluído el proceso, el archivo de costos se ordena de forma ascendente, y el valor más pequeño es buscado en el archivo que contiene el resumen de las corridas. Los parámetros correspondientes son los del diseño óptimo.

En este esquema, alguien pudiera aducir que el tiempo puede volverse un factor a tomar en cuenta si el mismo proceso se tiene que repetir 81 veces. Sin embargo, si se analizan los pasos anteriores, cada ejecución es completamente independiente de las demás, por lo que pueden llevarse a cabo todas en paralelo, reduciéndose así el tiempo total al requerido por una sola corrida (aproximadamente 15 segundos en una PC DX/2 de 66 MHz con coprocesador matemático).

Nuestra metodología fue probada extensamente, y en todos los casos obtuvo el valor óptimo, pese a que en varias ocasiones sólo ocurrió una vez en todo el espectro de valores utilizados. Aunque no hay por el momento evidencia teórica que respalde nuestra premisa, todo parece indicar que nuestro sistema constituye una herramienta confiable para llevar a cabo diseños óptimos de vigas rectangulares de concreto reforzado.

Trabajo Futuro y Conclusiones

Todavía queda mucho trabajo por delante, pues tenemos que afinar nuestros resultados, y queremos explorar nuevas alternativas para ajustar los parámetros del algoritmo genético, como por ejemplo el empleo de lógica difusa. También se necesita más trabajo teórico y mayor análisis de los espacios de búsqueda de este tipo de problemas de optimización, a fin de poder determinar estrategias más efectivas de atacarlo usando esta técnica heurística. Sin embargo, hasta ahora los resultados obtenidos parecen alentadores, y la flexibilidad y generalidad de la técnica la hacen muy adecuada para abordar problemas de optimización estructural más grandes y complejos.

Tenemos planes a corto plazo de seguir aplicando esta técnica a otro tipo de vigas, y extenderla a marcos y parrillas planas. Nuestro objetivo a largo plazo es contar con un sistema integrado de diseño óptimo basado en los algoritmos genéticos, y ya hemos realizado trabajo en los últimos dos años en esa dirección [4, 5, 6, 7]. Finalmente, tenemos contemplado utilizar otras técnicas heurísticas que han estado causando conmoción en las esferas académicas, como la búsqueda tabú [13, 14]. De tal forma, pensamos incorporar en nuestro sistema las técnicas de programación matemática tradicionales, junto con los algoritmos genéticos y otras técnicas heurísticas, y posiblemente híbridos que podamos crear entre ellas. Esto deberá ser de gran utilidad para poder realizar experimentos en esta disciplina, y para poder derivar importantes resultados teóricos que den luz en el futuro sobre cómo incorporar estas técnicas en sistemas reales que puedan ser utilizados en la práctica cotidiana de la ingeniería.

Agradecimientos

Los autores desean expresar su agradecimiento a Ascensión Elizalde Molina y Carlos Narcía López por su tiempo y empeño al realizar las pruebas experimentales, ya que sin su ayuda este trabajo no habría sido posible.

Referencias

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

[2] B. K. Chakrabarty. A model for optimal design of reinforced concrete beam. *Journal of Structural Engineering*, 118(11):3238-3242, November 1992.

[3] B. K. Chakrabarty. Model for optimal design of reinforced concrete beams. *Computers and Structures*, 42(3):447-451, 1992.

[4] Carlos A. Coello. Uso de algoritmos genéticos para el diseño óptimo de armaduras. En el *Congreso Nacional de Informática - 1994 Herramientas para los Mercados Globales*, pp. 290-305, Cd. de México, México, Junio de 1994. Fundación Arturo Rosenblueth.

[5] Carlos A. Coello and Alan D. Christiansen. Optimization of truss designs using genetic algorithms. Technical Report TUTR-CS-94-102, Tulane University, November 1994.

[6] Carlos A. Coello and Alan D. Christiansen. Using genetic algorithms for optimal design of axially loaded non-prismatic columns. Technical Report TUTR-CS-95-101, Tulane University, January 1995.

[7] Carlos A. Coello Coello. Discrete optimization of trusses using genetic algorithms. En J.G. Chen, F. G. Attia, and D. L. Crabtree, editors, *EXPERSYS-94. Expert Systems Applications and Artificial Intelligence*, pp. 331-336, Houston, Texas, November 1994. I.I.T.T. International. Technology Transfer Series.

[8] Carlos A. Coello Coello. El algoritmo genético como alternativa a la programación dinámica. En *Actas del VIII Simposio Internacional en Aplicaciones de Informática*, pp. 151-157, Antofagasta, Chile, Noviembre de 1994. Universidad Católica del Norte.

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

[10] Edward Joseph Haug, Jr. *Minimum Weight Design of Beams with Inequality Constraints on Stress and Deflection*. PhD thesis, Department of Mechanical Engineering, Kansas State University, 1966.

[11] Noel J. Everard and John L. Tanner III. *Theory and Problems of Reinforced Concrete Design*. McGraw-Hill Book Company, second edition, 1987.

[12] Galileo Galilei. *Dialogues Concerning Two New Sciences*. Evanston, Ill. Northwestern University Press, 1950. (Publicado originalmente en 1665).

[13] Fred Glover. Tabu search - part I. *ORSA Journal on Computing*, 1(3):190-206, 1989.

[14] Fred Glover. Tabu search - part II. *ORSA Journal on Computing*, 2(4):4-32, 1990.

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

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

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

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

[19] Andrzej Osyczka. Multicriteria optimization for engineering design. En John S. Gero, editor, *Design Optimization*, pp. 193-227. Academic Press, 1985.

[20] Anand Prakash, S.K. Agarwala, & K. K. Singh. Optimum design of reinforced concrete sections. *Computers and Structures*, 30(4):1009-1011, 1988.

[21] V. B. Venkayya. Design of optimum structures. *Computers and Structures*, 1:265-309, 1971.