Uso de algoritmos genéticos para el diseño óptimo de armaduras

M.C. Carlos Artemio Coello Coello

Introducción

Al parecer, Galileo fue el primero en interesarse en el estudio de la optimización de estructuras, como se refleja en sus trabajos sobre la deflexión de vigas [Galilei, 1950]. Bernoulli, Lagrange y Navier, son sólo algunos de los tantos científicos que se interesaron también en encontrar la "mejor" forma geométrica que pudiera resistir ciertos esfuerzos deseados. Con el paso de los años, esta disciplina evolucionó hasta convertirse en una rama más de la ingeniería llamada *Optimización Estructural*, que se ocupa precisamente de encontrar las formas geométricas que resulten más económicas, pero que a la vez sean capaces de resistir los esfuerzos requeridos por nuestro diseño.

Este trabajo se enfoca en el uso de un procedimiento de búsqueda basado en la mecánica de la selección natural, conocido como *algoritmo genético*, para el diseño óptimo de armaduras planas y tridimensionales. Primero, explicaremos de forma general en qué consiste la técnica, así como la forma en que se aplicó a este problema en particular, para después pasar al análisis de los resultados obtenidos, comparándolos con las técnicas existentes para optimizar este tipo de estructuras.

1. Nociones de algoritmos genéticos

A fines de los 60's, John Holland y un grupo de estudiantes de la Universidad de Michigan investigaban cómo construir máquinas que pudieran aprender. Holland notó que el aprendizaje podía ocurrir no sólo mediante la adaptación de un solo organismo, sino también mediante la adaptación evolutiva de muchas generaciones de una especie. Su trabajo se inspiró en la noción Darwiniana de la evolución, de acuerdo a la cual sólo el más apto sobrevive. Holland propuso que la búsqueda de una máquina que fuera capaz de aprender debía manejarse a través de los "descendientes" de muchas estrategias en una población de candidatos, en vez de plantearla como la construcción y el refinamiento de una sola estrategia. Holland y sus estudiantes llamaron a su técnica "planes reproductivos", pero el término **algoritmos genéticos** se volvió popular tras la publicación de su libro [Holland, 1975].

Podemos definir un algoritmo genético de la siguiente manera [Koza, 1992]:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto (*población*) de objetos matemáticos individuales (típicamente cadenas de caracteres de longitud fija que se adaptan al modelo de las cadenas de cromosomas), cada uno de los cuales se asocia con una *aptitud*, en una población nueva (i.e. la siguiente *generación*) usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas (notablemente la recombinación sexual)".

A principios de los 80's los algoritmos genéticos parecían altamente prometedores. Los líderes del campo empezaron a dar conferencias regulares cada año. En 1989, David Goldberg, de la Univesidad de Alabama publicó un libro [Goldberg, 1989] que proporcionó una sólida base científica para el campo y citó no menos de 73 aplicaciones exitosas de los algoritmos genéticos. En 1991, Lawrence Davis de Tica Asociados publicó un manual de algoritmos genéticos [Davis, 1991] que se volvió también un clásico instantáneo. Desde entonces, el área se ha vuelto una verdadera mina de oro para desarrollar aplicaciones en campos como optimización, búsqueda y aprendizaje de máquinas.

1.1. ¿Cómo funcionan los algoritmos genéticos?

La operación básica de un algoritmo genético se muestra en el siguiente segmento de pseudo-código [Buckles & Petry, 1992]:

```
generar población inicial, G(0);
evaluar G(0);
t:=0;
repetir
        t:=t+1;
        generar G(t) usando G(t-1);
        evaluar G(t);
hasta encontrar una solución
```

Básicamente lo que se hace es buscar una representación de las posibles soluciones de nuestro problema (normalmente se usan cadenas binarias). A cada una de estas representaciones se le llama *cromosoma*, y al conjunto de ellas se le conoce como *población*. La primera generación es creada usando números aleatorios, pero en las siguientes los cromosomas más aptos (i.e. los más cercanos a la solución deseada) tendrán más

probabilidades de sobrevivir. Los cromosomas que logran sobrevivir se *cruzan* entre sí, y sus descendientes originan la siguiente generación.

Las principales operaciones asociadas con los algoritmos genéticos son: selección, cruza y mutación.

La **selección** consiste en tomar de una cierta población a algunos de sus individuos (presumiblemente, los "mejores"), de acuerdo a una función de aptitud definida previamente. La definición de tal función suele ser, precisamente, la parte más difícil de resolver cuando se aplica la técnica a un cierto problema. Una de las cosas más interesantes de esta técnica es que, el hecho de que un individuo sea muy apto no necesariamente implica que será seleccionado, pues este operador suele ser aleatorio. Sin embargo, la aptitud guía fuertemente la búsqueda, por lo que los más aptos tienen mucha mayor probabilidad de ser seleccionados. Las 2 técnicas más comunes de selección son : la rueda de la fortuna y el torneo.

En la selección con uso de la **rueda de la fortuna**, se asigna una probabilidad *i* a cada organismo, que se calcula de acuerdo a [Buckles & Petry, 1992] :

$$F_i = \frac{f_i}{\sum_j f_j}$$

Esta fórmula se usa para seleccionar aleatoriamente un individuo, basándose en su probabilidad. Puede verse claramente que los individuos con mayor aptitud (la f representa la aptitud) tienen mayores probabilidades de ser seleccionados.

La selección mediante **torneo** consiste en hacer competir a los individuos en grupos de un cierto tamaño predefinido, y los sobrevivientes son aquellos cuya aptitud es mayor. Antes del torneo la población debe barajarse de forma aleatoria.

La **cruza** es el operador mediante el cual los individuos seleccionados previamente intercambian material cromosómico, y son sus descendientes los que formarán la población de la siguiente generación. Las 2 formas más populares de cruzar 2 cadenas (asumiendo una representación binaria) son : uso de un punto único de cruza y uso de 2 puntos de cruza.

Cuando se usa *un solo punto de cruza*, éste se escoge de forma aleatoria, y a partir de él se realiza el intercambio de material de los 2 individuos, tal y como se muestra en la Figura 1.

Cuando se usan *2 puntos de cruza*, se procede de manera similar, pero en este caso, el intercambio se realiza en la forma que se muestra en la Figura 2.

Una **mutación** es un cambio aleatorio realizado a uno de los genes de un cromosoma. Cuando se usa una representación binaria, el gene seleccionado se sustituye por su complemento (un cero cambia en uno y viceversa). Este operador permite la introducción de

nuevo material genético en la población, y desde el punto de vista teórico asegura que el espacio de búsqueda se encuentre totalmente conectado [Buckles & Petry, 1992].

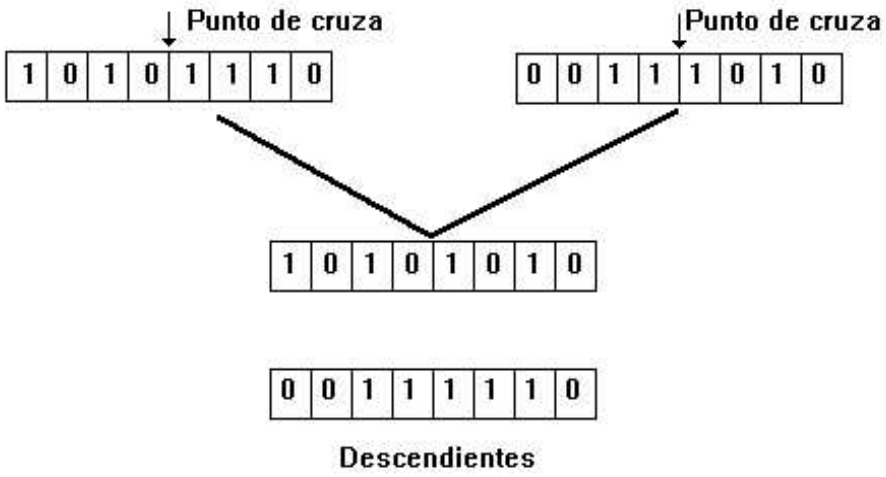

Figura 1 : Uso de un solo punto de cruza entre 2 individuos. Observe que cada pareja de cromosomas da origen a 2 descendientes para la siguiente generación. El punto de cruza puede ser cualquiera de los 2 extremos de la cadena, en cuyo caso no se realiza la cruza.

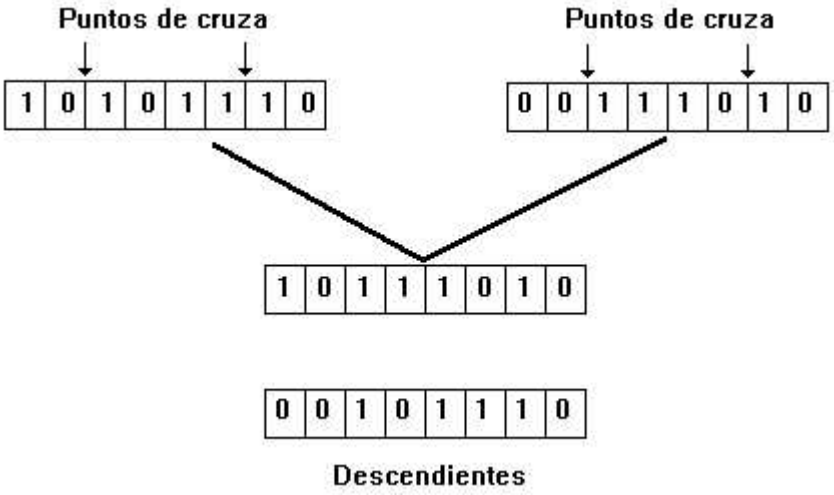

Figura 2 : Uso de 2 puntos de cruza entre 2 individuos. Note como en este caso se mantienen los genes de los extremos, y se intercambian los del centro. También aquí existe la posibilidad de que uno o ambos puntos de cruza se encuentren en los extremos de la cadena, en cuyo caso se hará una cruza usando un solo punto, o ninguna cruza, según corresponda.

Si supiéramos la respuesta a la que debemos llegar, entonces detener el algoritmo genético sería algo trivial. Sin embargo, eso casi nunca es posible, por lo que normalmente se usan 2 criterios principales para detenerlo : correrlo durante un número máximo de generaciones

o detenerse cuando la población se haya estabilizado (i.e. cuando todos o la mayoría de los individuos tengan la misma aptitud).

1.2. Ventajas de los algoritmos genéticos

Los algoritmos genéticos poseen varias características que los hacen altamente deseables para problemas de optimización :

No requieren conocimientos específicos del problema para llevar a cabo la búsqueda.

Usan operadores aleatorios en vez de operadores determinísticos, lo que hace que la convergencia de la técnica varíe con respecto al tiempo.

Operan de forma simultánea con varias soluciones, tomando información de varios puntos del espacio de búsqueda como guía.

Resultan menos afectados por los máximos locales que las técnicas de búsqueda tradicionales.

2. Trabajo previo

Goldberg parece haber sido el primero en advertir el potencial de los algoritmos genéticos en el diseño óptimo de estructuras, en un artículo que publicó hace 8 años [Goldberg & Samtani, 1986] sobre el uso de un algoritmo genético para optimizar una armadura plana de 10 miembros. Después de él, algunos cuantos más se atrevieron a seguir sus pasos, y podemos encontrar en la literatura aplicaciones de la técnica a vigas soldadas [Debb, 1991], marcos planos [Jenkins, 1991] y armaduras en general [Rajeev & Krishnamoorthy, 1992].

[Pham & Yang, 1993] presentan también un trabajo muy interesante sobre la optimización de funciones discretas multi-modales usando algoritmos genéticos. [Powell et al., 1989] hablan sobre una herramienta para optimizar diseños, de carácter general (i.e. independiente del problema), dirigida a los ingenieros, a la que llaman EnGENEous. Este programa usa un sistema experto y algoritmos genéticos para moverse de un sistema general, del que carecemos de conocimiento, a uno que depende de su dominio, pero del cual se tiene información. El prototipo ha sido usado en el diseño de ventiladores, turbinas de aviones y con estructuras moleculares, y los autores afirman que ha incrementado la productividad de los ingenieros en un factor de diez.

[Powell & Skolnick, 1993] muestra los resultados de examinar 10 problemas de optimización de diseños ingenieriles con respecto al desempeño comparativo de los algoritmos genéticos versus la optimización numérica tradicional (normalmente, la programación lineal).

[Schoenauer & Xanthakis, 1993] presentan un método general para el manejo de restricciones en la optimización vía algoritmos genéticos, basada en el paradigma conocido como *Behavioural Memory*. En vez de requerir operadores (proyección de la región factible) o una función de penalización (suma ponderada de las violaciones a las restricciones y la función objetivo) que dependan del problema en particular, muestrearon la región factible mediante la evolución desde una población aleatoria inicial, aplicando sucesivamente una serie de diferentes funciones de aptitud que involucran la satisfacción de las restricciones propias del problema. El éxito de este proceso depende altamente en la diversidad genética mantenida durante los primeros pasos, asegurando un muestreo uniforme de la región factible. Este esquema fue aplicado a problemas de optimización de estructuras : una armadura plana de 10 barras y una armadura tridimensional de 25 barras.

[Louis & Rawlins, 1990] también proporcionan una discusión del uso de algoritmos genéticos para diseñar estructuras, aunque se enfocan en el problema del diseño de circuitos combinatorios : dado un conjunto de compuertas lógicas, se quiere diseñar un circuito que lleve a cabo una cierta función deseable.

3. Uso de la técnica

Para ilustrar el uso de los algoritmos genéticos en la optimización de estructuras, consideremos primero la armadura plana de 10 elementos [Rajeev & Krishnamoorthy, 1992] que se muestra en la Figura 3.

Se asumen los siguientes datos : E = $1x10^4$ ksi (6.89×10^4 MPa), = 0.10 lb/in^3 (2,770 kg/m^3), y cargas verticales hacia abajo de 100 kips (445.374 kN) en los nudos 2 y 4.

Puesto que la función objetivo es minimizar el peso, f(x), para este caso en particular puede escribirse como :

$$f(x) = \sum_{j=1}^{10} \rho A_j L_j$$

Donde A_j es el área de la sección transversal del miembro jésimo; L_j es la longitud del miembro jésimo y es la densidad del material. Adicionalmente, la armadura está sujeta al siguiente conjunto de restricciones:

$$\sigma_j \leq \sigma_a \quad \text{para j} = 1 \text{ a } 10$$

$$u_j \leq u_a$$

Donde $_j$ es el esfuerzo en cada miembro; $_a$ es el máximo esfuerzo permisible; u_j es el desplazamiento de cada nudo (horizontal y vertical) y u_a es el máximo desplazamiento permisible.

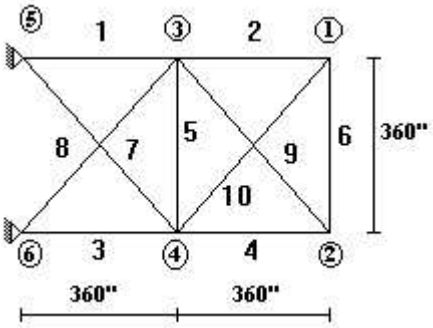

Figura 3 : Armadura plana de 10 miembros usada como primer ejemplo de optimización.

Las restricciones puede expresarse en forma normalizada:

$$\frac{\sigma_j}{\sigma_a} - 1 \leq 0 \; ; \quad \frac{u_j}{u_a} - 1 \leq 0$$

La función de aptitud propuesta fue : $\frac{1}{f(x) * (1000 * viol + 1)}$. La variable **viol** es un contador del número de restricciones que fueron violadas con la solución obtenida. Con esta función puede verse claramente que cuando no se viola ninguna restricción la aptitud es simplemente la inversa del peso. Por el contrario, entre más restricciones se violen, más se reduce el valor de la aptitud del individuo. El número 1000 fue determinado experimentalmente.

Las restricciones propias de este problema son las siguientes : el máximo desplazamiento permisible está limitado a 2 pulgadas (50.8 mm) en cualquier dirección, y los esfuerzos a 25 ksi (172.25 MPa).

Además, disponemos de una lista de valores discretos considerados para la solución del problema. Esta lista fue tomada del Manual del Instituto Americano de la Construcción con Acero [Arora, 1989]. S = {1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5} ($pulg^2$).

Puesto que hay 10 variables de diseño y cada una de ellas puede asumir cualquiera de las 42 secciones disponibles, entonces el espacio total de búsqueda es 42^{10}. Debido a que

estamos usando una representación binaria, el número 42 es un problema, porque no podemos usar sólo 5 dígitos para representarlo (serían insuficientes) y usando 6 tenemos valores de sobra. Por razones obvias se decidió usar 6 dígitos y se rellenaron los espacios sobrantes con los últimos valores de la lista, aunque pudo usarse cualquier valor dentro de S. Como tenemos 10 miembros, los cromosomas deberán tener 60 caracteres de longitud. La forma de representar un cromosoma es mediante un arreglo de dígitos binarios tal y como se muestra en la Figura 4.

Figura 4 : Representación binaria correspondiente a un cromosoma.

De tal forma, la cadena compuesta completamente por ceros representa al primer elemento del conjunto S (i.e. 1.62 pulg²).

3.1 Implementación

Para fines de este trabajo se utilizó una versión modificada del SGA (Simple Genetic Algorithm) escrito en Turbo Pascal que presenta [Goldberg, 1989]. Los cambios más importantes fueron en torno al uso de asignación dinámica de memoria, en vez de los arreglos estáticos del programa original, usando una técnica propuesta por [Porter, 1988], y en cuanto al uso del coprocesador matemático. Asimismo, utilizamos selección mediante torneo binario en vez de la rueda de la fortuna del programa de Goldberg, y se efectuaron varios cambios menores en algunas de sus funciones a fin de hacerlas más comprensibles. El criterio de detención fue determinado a través de un número máximo de generaciones proporcionado por el usuario. La probabilidad de cruza se manejó en el orden del 80%, mientras que la de mutación fue siempre muy pequeña (1%), pero sin desactivarla como propone [Rajeev & Krishnamoorthy, 1992]. Para analizar la estructura se utilizaron los programas en Turbo Pascal contenidos en [Coello, 1991]. La operación básica del programa es muy simple : se alimentan los valores de las áreas posibles (i.e. el contenido de la lista S). Posteriormente se incluyen las restricciones a las que está sometida la estructura. El programa de análisis es ejecutado por separado para generar la matriz de rigideces con la que se trabajará, y sus resultados se almacenan en un archivo. El algoritmo genético es ejecutado y se preguntan al usuario las probabilidades de cruza y mutación, el tamaño de la población, la longitud de cada cromosoma y el número máximo de generaciones

deseadas. Después, el programa empieza a trabajar, leyendo el archivo mencionado anteriormente, y modificándolo con los valores decodificados de los cromosomas de cada generación. Conforme se avanza, se va enviando a la impresora un resumen de cada generación que incluye información tal como quién fue el mejor individuo en ella, cuál es el mejor que se ha encontrado hasta el momento y cuál es el valor de su aptitud, y una decodificación de su cadena, para saber cuáles son las áreas que le corresponden.

3.2. Optimización de armaduras tridimensionales

Una de las bondades de los algoritmos genéticos se refleja en el hecho de que prácticamente sin cambios en el código, pudimos aplicarlo a armaduras tridimensionales, cambiando simplemente el módulo para analizar la estructura. El siguiente ejemplo que utilizamos es la armadura tridimensional de 25 miembros que se muestra en la Figura 5.

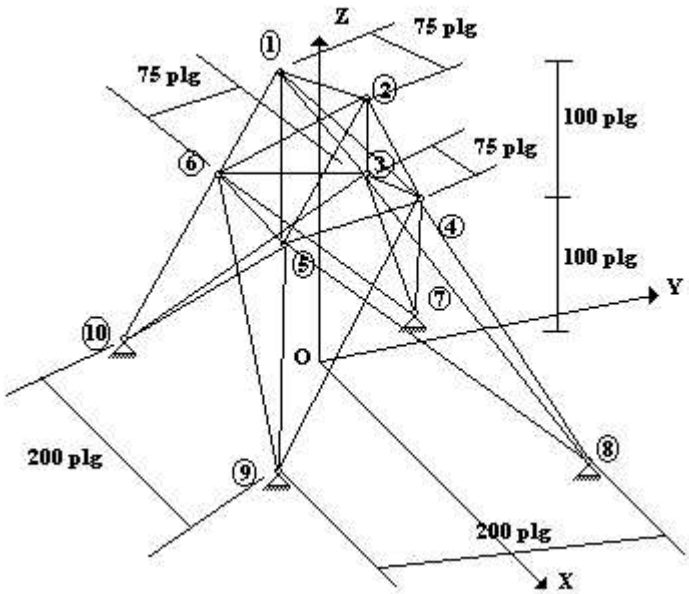

Figura 5 : Armadura tridimensional de 25 miembros utilizada como ejemplo.

Se asumen los siguientes datos : E = $1x10^7$ ksi, = 0.10 lb/in^3. Los desplazamientos en los nudos 1 y 2 no deben ser mayores de 0.35 pulgadas. Los esfuerzos están limitados a 40000 psi. Las cargas a las que está sometida la armadura se muestran en la Tabla 1.

El conjunto de secciones disponibles para este problema es [Rajeev & Krishnamoorthy, 1992] : S = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4} (pulg²).

Nudo	Fx (lbs)	Fy (lbs)	Fz (lbs)
1	1000	-10000	-10000
2	0	-10000	-10000
3	500	0	0
6	600	0	0

Tabla 1 : Condiciones de carga de la armadura tridimensional de 25 miembros mostrada en la Figura 5.

Esta armadura tridimensional, pese a tener 25 miembros, tiene sólo 8 tipos diferentes de áreas, los cuales se muestran en la Tabla 2, junto con la información de las conexiones de los miembros.

Grupo Número	Miembros
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

Tabla 2 : Grupos en que se encuentran conformadas las áreas de la armadura tridimensional mostrada en la Figura 5, así como los miembros que se incluyen en cada uno.

Los resultados obtenidos en este problema y el de la armadura plana planteado anteriormente se muestran en la siguiente sección.

4. Comparación de resultados

Tradicionalmente, el diseño óptimo de estructuras se ha realizado usando métodos que tratan al problema como si su espacio de búsqueda fuera continuo, y se han aplicado desde el método simplex hasta métodos gradientes y modificaciones al método de Newton. [Arora, 1989], [Brandt, 1984] y [Belegundu, 1982] contienen una buena revisión de estos métodos.

En los últimos años se han reportado, sin embargo, una cierta cantidad de métodos para el diseño óptimo de sistemas estructurales discretos. [Toskley, 1968] propuso un modelo de

optimización para armaduras estáticamente determinadas usando la lista discreta de los tamaños de sus miembros. [Templeman & Yates, 1983] retomaron la idea y propusieron un método para resolver armaduras hiperestáticas que toma en cuenta las restricciones de los esfuerzos y los desplazamientos de sus miembros. [Zhu, 1986] introdujo una modificación a este método que mejoró notablemente su desempeño. [John & Ramakrishnan, 1987] propusieron el uso de programación lineal secuencial con un algoritmo de ramificación y límite (branch & bound) para la optimización de estructuras discretas. [Saka & Ulker, 1992] presentan un algoritmo de optimización estructural para armaduras tridimensionales estáticamente indeterminadas, sujetas a restricciones de esfuerzo, desplazamiento y dimensiones de sus secciones. Dicho algoritmo se basa en el uso de un análisis lineal iterativo de la estructura. Finalmente, resulta de gran interés leer el trabajo de [Rao et al., 1992] sobre la optimización con múltiples objetivos de sistemas borrosos (fuzzy systems), en el cual ilustra esta técnica usando una armadura plana y una armadura espacial.

Pese a la cantidad de trabajo realizado en torno a este problema, existe mucho por hacer todavía en esta área. Para empezar, las técnicas tradicionales tratan al problema como si fuera continuo, cuando en realidad es discreto, lo que trae como consecuencia que las soluciones halladas no sean del todo adecuadas. Por su parte, los métodos que consideran al problema correctamente, suelen resultar imprácticos cuando intentan aplicarse a estructuras muy complejas debido al tiempo de ejecución requerido, o a lo difícil de la implementación. El uso de algoritmos genéticos parece tener aquí perfecta cabida, pues resultan idóneos para problemas de este tipo, y aunque su tiempo de ejecución no sea precisamente el más corto, no suelen hacerlo tan mal.

Para corroborar nuestras palabras, veamos los resultados obtenidos para la armadura de 10 miembros que se muestra en la Figura 3, así como la comparación de nuestros resultados con los obtenidos usando otros métodos encontrados en la literatura [Rajeev & Krishnamoorthy, 1992] [Belegundu, 1982] en la Tabla 3.

Puede verse que el algoritmo genético no lo hizo tan mal, sobre todo si tomamos en cuenta lo fácil que resultó su implementación. Lo que es mejor aún, es que existe la posibilidad de encontrar mejores soluciones simplemente mediante el ajuste conveniente de los parámetros proporcionados al programa.

Método	Peso	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Rajeev	5613.84	33.50	1.62	22.00	15.50	1.62	1.62	14.20	19.90	19.90	2.62
CONMIN	5563.00	25.20	1.89	24.87	15.83	0.10	1.75	16.76	19.73	20.98	2.51
OPTDYN	5472.00	25.70	0.10	25.11	19.39	0.10	0.10	15.40	20.32	20.74	1.14

LINRM	6249.00	21.57	10.98	22.08	14.95	0.10	10.98	18.91	18.42	18.40	13.51
SUMT	5932.00	30.69	2.37	31.62	11.66	0.10	3.71	21.71	20.90	13.97	3.26
M-3	5719.00	25.84	3.07	26.42	12.77	0.10	3.43	19.34	19.17	18.76	4.42
M-5	5725.00	25.83	2.88	26.45	12.75	0.10	3.77	19.37	19.18	18.77	4.38
GRP-UI	5727.00	24.78	4.17	24.78	14.45	0.10	4.17	17.46	19.26	19.27	5.26
GENETICO	5681.82	26.50	2.13	22.90	13.50	1.62	2.63	15.50	22.90	22.90	1.62

Tabla 3 : Resultados obtenidos con nuestra implementación (GENETICO) y su comparación con otros métodos reportados en la literatura. Rajeev utiliza también algoritmos genéticos, aunque limita considerablemente el espacio de búsqueda haciendo uso de ciertas premisas sobre la estructura y utiliza menos generaciones que nosotros. Para información detallada de cada uno de los métodos presentados, refiérase a [Belegundu, 1982] y [Rajeev & Krishnamoorthy,1 992].

Analicemos ahora los resultados obtenidos para la armadura tridimensional de 25 miembros que se muestra en la Figura 5, los cuales se encuentran en la Tabla 4.

Método	Peso	A1	A2	A3	A4	A5	A6	A7	A8
Zhu	562.93	0.1	1.9	2.6	0.1	0.1	0.8	2.1	2.6
Rizz	545.16	0.01	1.988	2.991	0.01	0.01	0.684	1.676	2.662
Schmit	545.22	0.01	1.964	3.033	0.01	0.01	0.67	1.68	2.67
Rajeev	546.01	0.1	1.8	2.3	0.2	0.1	0.8	1.8	3.0
GENETICO	539.78	1.50	0.70	3.40	0.70	0.40	0.70	1.50	3.20

Tabla 4 : Resultados obtenidos con nuestra implementación (GENETICO) y su comparación con otros métodos reportados en la literatura. Para mayor información sobre los otros métodos presentados, refiérase a [Zhu, 1986] y [Rajeev & Krishnamoorthy,1 992].

Como habrá podido notar, en esta ocasión nuestra implementación arrojó el mejor resultado de todos, pese a haber competido con métodos continuos. Nuevamente, debemos aclarar que [Rajeev & Krishnamoorthy, 1992] utilizan un algoritmo genético, pero reducen el espacio de búsqueda estableciendo ciertas premisas acerca de la estructura, y utilizan menos generaciones que nosotros.

5. Trabajo futuro

Todavía queda mucho trabajo por realizarse en torno a este tema. Actualmente, nos encontramos trabajando en un estudio detallado de los parámetros que se proporcionan al algoritmo genético, a fin de ser capaces de aconsejar a los usuarios cómo ajustarlos en caso de que sea necesario. Este es un problema importante, pues en algunos casos el método puede no converger debido a que la población es muy reducida, o al hecho de que se usaron muy pocas generaciones.

Por otro lado, estamos también interesados en funciones de aptitud más complejas que la presentada en este trabajo, las cuales deberán reflejar de manera más realista lo "buena" que

es una cierta solución. [Rajeev & Krishnamoorthy, 1992] proponen una interesante función de aptitud que, sin embargo, no funcionó muy bien en las pruebas que corrimos.

También queremos intentar usar una función de aptitud dinámica (i.e. que varíe conforme nos acerquemos o nos alejemos de la solución deseada).

Por otro lado, seguimos corriendo pruebas con estructuras más grandes y complejas, aunque resulta difícil encontrar información en la literatura que nos permita determinar qué tan bien lo estamos haciendo.

Finalmente, si tenemos el éxito esperado, nuestros planes a largo plazo incluyen la aplicación de la técnica a las estructuras reticulares restantes (i.e. marcos planos y tridimensionales, vigas y parrillas), a fin de poder elaborar un sistema de optimización de estructuras reticulares usando algoritmos genéticos.

6. Conclusiones

Los algoritmos genéticos parecen ser una técnica muy apropiada para resolver problemas de optimización con espacios de búsqueda discretos, debido a la forma en que operan. Hemos visto lo fácil que resulta aplicarlos para resolver problemas de alta complejidad matemática, así como la flexibilidad que presentan para poder aplicarse a problemas totalmente diferentes.

En términos de la implementación, ésta resulta muy sencilla, y no se requiere ningún tipo de técnica de programación sofisticada. Además, el programa puede correrse en cualquier tipo de equipo, si bien es recomendable contar con un coprocesador matemático para disminuir los tiempos de ejecución.

Aunque los resultados que hemos encontrado hasta ahora parecen altamente satisfactorios, todavía queda mucho trabajo por hacer antes de poder asegurar que la técnica es la piedra filosofal que los ingenieros han estado buscando desde hace casi 2 siglos para la optimización de estructuras. Sin embargo, el potencial de los algoritmos genéticos está ahí, esperando a ser explotado no sólo en este tipo de aplicaciones, sino en muchas otras dentro de la misma ingeniería civil, y en un sinfín más de disciplinas.

Referencias Bibliográficas

[Arora, 1989] Arora, Jasbir S. **Introduction to Optimum Design**. McGraw-Hill Book Company. 625 p.

[Belegundu, 1982] Belegundu, Ashok Dhondu. **A Study of Mathematical Programming Methods for Structural Optimization**. PhD Thesis. University of Iowa. Iowa City, Iowa. 362 p.

[Brandt et al. 1984] Brandt, Andrzej M., Dzieniszewski, Wojciech, Jendo, Stefan, Marks, Wojciech, Owczarek, Stefan & Wasiutynski, Zbigniew. **Criteria and Methods of Structural Optimization**. 538 p.

[Brousse, 1988] Brousse, Pierre. **Optimization in Mechanics : Problems and Methods**. North-Holland. 279 p.

[Buckles & Petry, 1992] Buckles, Bill P. and Petry, Frederick E. **Genetic Algorithms**. IEEE Computer Society Press Technology Series. 109 p.

[Cheng, 1986] Cheng, Franklin Y. (Editor). **Recent Developments in Structural Optimization**. American Society of Civil Engineers. 102 p.

[Coello, 1991] Coello, Coello Carlos A. **Análisis de Estructuras Reticulares por Computadora (Método de Rigideces)**. Tesis de Licenciatura. Universidad Autónoma de Chiapas. Marzo.

[Davis, 1991] Davis, Lawrence (Editor). **Handbook of Genetic Algorithms**. New York : Van Nostrand Reinhold.

[Deb, 1991] Deb, Kalyanmoy. **Optimal Design of a Welded Beam via Genetic Algorithms**. *AIAA Journal* 29 : 2013-15. November.

[Galilei, 1950] Galilei, Galileo. **Dialogues Concerning Two New Sciences**. Evanston, Ill. Northwestern University Press. 1950. 288 p.

[Goldberg & Samtani, 1986] Goldberg, David E. and Samtani, Manohar P. **Engineering Optimization via Genetic Algorithm**. *Proceedings of the Ninth Conference on Electronic Computation*. ASCE, New York, N.Y. pp. 471-482.

[Goldberg, 1989] Goldberg, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley Publishing Co., Inc., Reading, Mass. 412 p.

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

[Holland, 1992] Holland, John H. **Genetic Algorithms**. *Scientific American*. Vol. 267. July. pp. 66-72.

[Jenkins, 1992] Jenkins, W. M. **Plane Frame Optimum Design Environment Based on Genetic Algorithm**. *Journal of Structural Engineering*. Vol. 118. No. 11. November. pp. 3103-3113.

[John & Ramakrishnan, 1987] John, K. V. and Ramakrishnan, C. V. **Minimum Weight Design of Trusses using Improved Move Limit Method of Sequential Linear Programming**. *Computers and Structures*. Vol. 27. No. 5. pp. 583-91.

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

[Louis & Rawlins, 1990] Louis, Sushil J. and Rawlins, Gregory J. E. **Designer Genetic Algorithms: Genetic Algorithms in Structure Design**. *Proceedings of the Second International Conference on Genetic Algorithms*. pp. 53-60.

[Pham & Yang, 1993] Pham, D. T. and Yang, Y. **Optimization of Multi-Modal Discrete Functions using Genetic Algorithms**. *Proceedings of the Institute of Mechanical Engineers.* Vol. 207.

[Porter, 1988] Porter, Kent. **Handling Huge Arrays**. *Dr. Dobb's Journal of Software Tools for the Professional Programmer*. March. Volume 13. Issue 3. pp. 60-3.

[Powell et al., 1989] Powell, D. J., Skolnick, M. M. and Tong S. S. **EnGENEous : domain independent, machine learning for design optimization**. *Proceedings of the First International Conference on Genetic Algorithms*.

[Powell & Skolnick, 1993] Powell, David and Skolnick, Michael M. **Using Genetic Algorithms in Engineering Design Optimization via Non-linear Constraints**. *Proceedings of the Fifth International Conference on Genetic Algorithms*. University of Illinois at Urbana-Champaign. July 17-21. Morgan Kauffman Publishers. pp. 424-31.

[Rajeev & Krishnamoorthy, 1992] Rajeev, S. and Krishnamoorthy, C. S. **Discrete Optimization of Structures Using Genetic Algorithms**. *Journal of Structural Engineering*. Vol. 118. No. 5. May. pp. 1233-50.

[Rao et al., 1992] Rao, S. S., Sundararaju, K., Prakash, B. G. and Balakrishna, C. **Multiobjective Fuzzy Optimization Techniques for Engineering Design**. *Computers & Structures*. Vol. 42. No. 1. pp. 37-44.

[Saka & Ulker, 1992] Saka, M. P. and Ulker, M. **Optimum Design of Geometrically Nonlinear Space Trusses**. *Computers and Structures*. Vol. 42. No. 3. pp. 289-99.

[Schoenauer & Xanthakis, 1993] Schoenauer, Marc and Xanthakis, Spyros. **Constrained GA Optimization**. *Proceedings of the Fifth International Conference on Genetic Algorithms*. University of Illinois at Urbana-Champaign. July 17-21. Morgan Kauffman Publishers. pp. 573-80.

[Templeman & Yates, 1983] Templeman, A. B. and Yates, D. F. **A Linear Programming Approach to Discrete Optimum Design of Trusses. Optimization Methods in Structural Design**. H. Eschenauer and N. Olholff, eds., BI Wissenschaftsverlag. Mannheim, Germany. pp. 133-9.

[Toakley, 1968] Toakley, A. R. **Optimum Design Using Available Sections**. *Journal of the Structural Division*. ASCE. Vol. 94. (ST 5). pp. 1219-33.

[Zhu, 1986] Zhu, D. M. **An Improved Templeman's Algorithm for Optimum Design of Trusses with Discrete Member Sizes**. *Engrg. Opt*. Vol. 9. pp. 303-12.