Algoritmos Genéticos y sus Aplicaciones[1]

Carlos A. Coello Coello

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad alrededor del mundo durante los últimos años. Se presentarán aquí los conceptos básicos que se requieren para abordarla, así como un sencillo ejemplo que permita a los lectores comprender cómo aplicarla al problema de su elección. Adicionalmente, se hablará acerca de los diversos ambientes de programación actuales basados en algoritmos genéticos y de las áreas abiertas de investigación.

Orígenes

En los últimos años, la comunidad científica internacional ha mostrado un creciente interés en una nueva técnica de búsqueda basada en la teoría de la evolución y que se conoce como el **algoritmo genético**. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes (unidad básica de codificación de cada uno de los atributos de un ser vivo) de un individuo, y que los atributos más deseables (i.e., los que le permiten a un individuo adaptarse mejor a su entorno) del mismo se transmiten a sus descendientes, cuando éste se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland estaba consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla en un programa de computadora. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su ibro [1] en 1975.

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza [2]:

Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo 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 de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.

¿Cómo saber si es posible usar el Algoritmo Genético?

La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y se recomienda en general tomar en cuenta las siguientes características del mismo antes de intentar usarla:

Su espacio de búsqueda (i.e., sus posibles soluciones) debe estar delimitado dentro de un cierto rango.

---

[1] Coello Coello, Carlos A. "Introducción a los Algoritmos Genéticos", *Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios*, Año 3, No. 17, Enero de 1995, pp. 5-11.

Debe poderse definir una función de aptitud que nos indique qué tan buena o mala es una cierta respuesta.

Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en la computadora.

El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos -aunque éstos sean muy grandes-. Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.

La **función de aptitud** no es más que la función objetivo de nuestro problema de optimización. El algoritmo genético únicamente maximiza, pero la minimización puede realizarse fácilmente utilizando el recíproco de la función maximizante (debe cuidarse, por supuesto, que el recíproco de la función no genere una división por cero). Una característica que debe tener esta función es que debe ser capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

La codificación más común de las respuestas es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero de estos esquemas ha gozado de mucha popularidad debido a que es el que propuso originalmente Holland, y además porque resulta muy sencillo de implementar.

Funcionamiento de un Algoritmo Genético Simple

La operación de un algoritmo genético simple puede ilustrarse con el siguiente segmento de pseudo-código [3]:

```
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;
```

Primero, se genera aleatoriamente la población inicial, que estará constituída por un conjunto de **cromosomas**, o cadenas de caracteres[2] que representan las soluciones posibles del problema. A cada uno de los cromosomas de esta población se le aplicará la función de aptitud a fin de saber qué tan buena es la solución que está codificando.

Sabiendo la aptitud de cada cromosoma, se procede a la **selección** de los que se cruzarán en la siguiente generación (presumiblemente, se escogerá a los "mejores"). Dos son los métodos de selección más comunes:

---

[2] En la práctica suelen usarse cadenas binarias para representar a los cromosomas, por lo que éstas suelen implementarse como cadenas de bits (usando el tipo *boolean* de Pascal o el modificador *unsigned* de C), pero también puede recurrirse (y se ha hecho) al uso de letras y/o números naturales.

1) La Ruleta : Es el usado por Goldberg en su libro [4]. Este método es muy simple, y consiste en crear una ruleta en la que cada cromosoma tiene asignada una fracción proporcional a su aptitud. Sin que nos refiramos a una función de aptitud en particular, supongamos que se tiene una población de 5 cromosomas cuyas aptitudes están dadas por los valores mostrados en la Tabla 1.

Cromosoma No.	Cadena	Aptitud	% del Total
1	11010110	254	24.5
2	10100111	47	4.5
3	00110110	457	44.1
4	01110010	194	18.7
5	11110010	85	8.2
Total		1037	100.0

Tabla 1 : Valores de ejemplo para ilustrar la selección mediante ruleta

Con los porcentajes mostrados en la cuarta columna de la Tabla 1 podemos elaborar la ruleta de la Figura 1. Esta ruleta se gira 5 veces para determinar qué individuos se seleccionarán. Debido a que a los individuos más aptos se les asignó un área mayor de la ruleta, se espera que sean seleccionados más veces que los menos aptos.

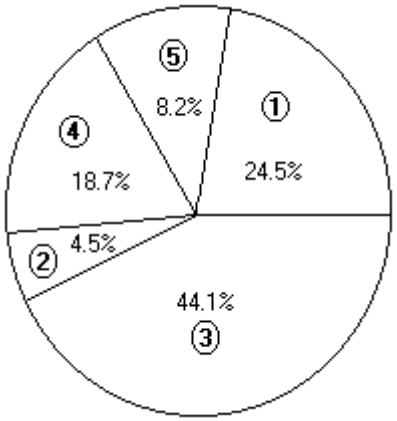

Figura 1 : Ruleta que representa los valores de aptitud de la Tabla 1

2) El torneo : La idea de este método es muy simple. Se baraja la población y después se hace competir a los cromosomas que la integran en grupos de tamaño predefinido (normalmente compiten en parejas) en un torneo del que resultarán ganadores aquéllos que tengan valores de aptitud más altos. Si se efectúa un torneo binario (i.e., competencia por parejas), entonces la población se debe barajar 2 veces. Nótese que esta técnica garantiza la obtención de múltiples copias del mejor individuo entre los progenitores de la siguiente generación (si se efectúa un torneo binario, el mejor individuo será seleccionado 2 veces).

Una vez realizada la selección, se procede a la **reproducción sexual** o **cruza** de los individuos seleccionados. En esta etapa, los sobrevivientes intercambiarán material cromosómico y sus descendientes formarán la población de la siguiente generación. Las 2 formas más comunes de reproducción sexual 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 sobre la longitud de la cadena que representa el cromosoma, y a partir de él se realiza el intercambio de material de los 2 individuos, tal y como se muestra en la Figura 2.

Cuando se usan **2 puntos de cruza**, se procede de manera similar, pero en este caso el intercambio se realiza en la forma mostrada en la Figura 3.

Normalmente la cruza se maneja dentro de la implementación del algoritmo genético como un porcentaje que indica con qué frecuencia se efectuará. Esto significa que no todas las parejas de cromosomas se cruzarán, sino que habrán algunas que pasarán intactas a la siguiente generación. De hecho existe una técnica desarrollada hace algunos años en la que el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie, y se mantiene intacto hasta que surge otro individuo mejor que él, que lo desplazará. Dicha técnica es llamada **elitismo**, y no debe sorprendernos el hecho de que haya sido desarrollada en Alemania.

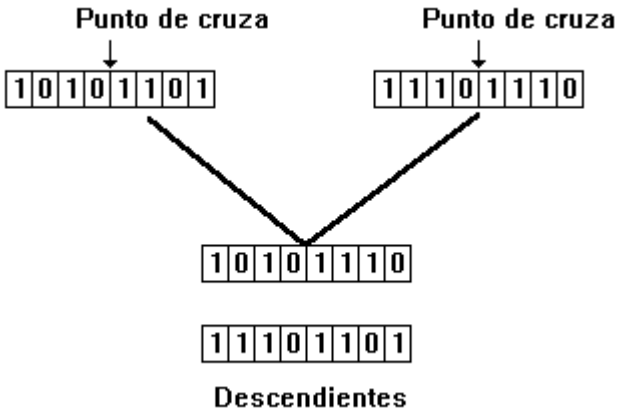

Figura 2 : Uso de un solo punto de cruza entre 2 individuos. Observe que cada pareja de cromosomas da origen a 2 descendientes para la siguiente generación. El punto de cruza puede ser cualquiera de los 2 extremos de la cadena, en cuyo caso no se realiza la cruza.

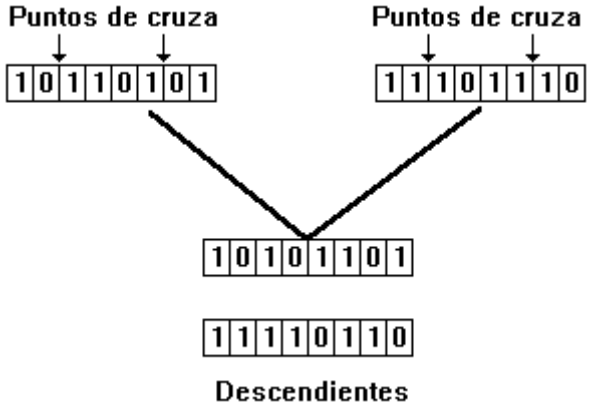

Figura 3 : Uso de 2 puntos de cruza entre 2 individuos. Note como en este caso se mantienen los genes de los extremos, y se intercambian los del centro. También aquí existe la posibilidad de que uno o ambos puntos de cruza se encuentren en los extremos de la cadena, en cuyo caso se hará una cruza usando un solo punto, o ninguna cruza, según corresponda.

Además de la selección y la cruza, existe otro operador llamado **mutación**, el cual realiza un cambio a uno de los genes de un cromosoma elegido aleatoriamente. 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 cromosómico en la población, tal y como sucede con sus equivalentes biológicos.

Al igual que la cruza, la mutación se maneja como un porcentaje que indica con qué frecuencia se efectuará, aunque se distingue de la primera por ocurrir mucho más esporádicamente (el porcentaje de cruza normalmente es de más del 60%, mientras que el de mutación normalmente nunca supera el 5%).

Si supiéramos la respuesta a la que debemos llegar de antemano, entonces detener el algoritmo genético sería algo trivial. Sin embargo, eso casi nunca es posible, por lo que normalmente se usan 2 criterios principales de detención : correr el algoritmo genético durante un número máximo de generaciones o detenerlo cuando la población se haya estabilizado (i.e., cuando todos o la mayoría de los individuos tengan la misma aptitud).

¿Qué Ventajas y Desventajas tienen con respecto a otras técnicas de búsqueda?

No necesitan conocimientos específicos sobre el problema que intentan resolver.

Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.

Cuando se usan para problemas de optimización -maximizar una función objetivo- resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales.

Resulta sumamente fácil ejecutarlos en las modernas arquitecturas masivas en paralelo.

Usan operadores probabilísticos, en vez de los típicos operadores determinísticos de las otras técnicas.

Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen -tamaño de la población, número de generaciones, etc.-.

Pueden converger prematuramente debido a una serie de problemas de diversa índole.

Un Ejemplo de Uso

Con la finalidad de aclarar mejor los conceptos cubiertos previamente, presentaremos ahora una sencilla aplicación del algoritmo genético a un problema de optimización tomado de [13]:

Un grupo de financieros mexicanos ha resuelto invertir 10 millones de pesos en la nueva marca de vino "Carta Nueva".

Así pues, en 4 ciudades de las principales de México se decide iniciar una vigorosa campaña comercial: México en el centro, Monterrey en el noroeste, Guadalajara en el occidente y Veracruz en el oriente. A esas 4 ciudades van a corresponder las zonas comerciales I, II, III y IV. Un estudio de mercado ha sido realizado en cada una de las zonas citadas y han sido establecidas curvas de ganancias medias en función de las inversiones totales (almacenes, tiendas de venta, representantes, publicidad, etc.) Estos datos se ilustran en la Tabla 2 y en la Figura 5.

Para simplificar los cálculos, supondremos que las asignaciones de créditos o de inversiones deben hacerse por unidades de 1 millón de pesos. La pregunta es: ¿en dónde se deben de asignar los 10 millones de pesos de los que se dispone para que la ganancia total sea máxima?

1) Representación : Lo primero que necesitamos determinar para poder aplicar el algoritmo genético, es cuál será el esquema a utilizarse para representar las posibles soluciones del problema. En este caso necesitamos 4 bits ($2^4 = 16$) para representar cada solución, porque cada una admite 11 valores posibles (de 0 a 10). Como existen 4 valores independientes (uno por cada

zona de estudio), se requieren entonces 16 bits (4 x 4) por cada cromosoma. De tal forma, una cadena representativa de un cromosoma será como se muestra en la Figura 4. Es importante hacer notar que se requiere una función de codificación (i.e., que transforme el valor de la inversión a binario) y una de decodificación (i.e., que realice el proceso inverso). Debido a que en este caso los 4 bits utilizados para representar una solución pueden producir más valores de los que se necesitan, se usará una función de ajuste que haga que los resultados producidos siempre se encuentren en el rango válido.

Inversión (en millones)	Beneficio I	Beneficio II	Beneficio III	Beneficio IV
0	0	0	0	0
1	0.28	0.25	0.15	0.20
2	0.45	0.41	0.25	0.33
3	0.65	0.55	0.40	0.42
4	0.78	0.65	0.50	0.48
5	0.90	0.75	0.62	0.53
6	1.02	0.80	0.73	0.56
7	1.13	0.85	0.82	0.58
8	1.23	0.88	0.90	0.60
9	1.32	0.90	0.96	0.60
10	1.38	0.90	1.00	0.60

Tabla 2 : Datos obtenidos con la investigación de mercado en cada una de las regiones en estudio.

2) Función de Aptitud : Dado que el objetivo es obtener las inversiones que sumen 10, y que tengan un beneficio máximo, podemos usar la siguiente función de aptitud penalizada:

$$F(x) = \frac{c1 + c2 + c3 + c4}{500 * v + 1}$$

donde **c1**, **c2**, **c3** y **c4** son las ganancias por zona, que se calculan de acuerdo a los valores de la Tabla 2, y **v** es el valor absoluto de la diferencia entre la suma obtenida y 10. Nótese que cuando no se viole ninguna restricción (i.e., cuando la suma de inversiones sea exactamente 10) la función de aptitud no será "castigada".

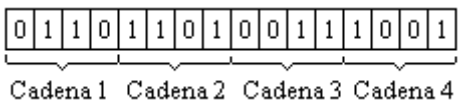

Figura 4 : Cadena representativa de un cromosoma de los utilizados en nuestro ejemplo de optimización. Las cadenas 1, 2, 3 y 4 corresponden a las cantidades invertidas en las zonas económicas respectivas.

3) Operadores : Se usará una cruza de 2 puntos, la cual se efectúa de la forma que se indica en la Figura 3. La probabilidad que se dará a la misma será del 80%. En cuanto a la mutación, se le asignará una probabilidad baja, tal y como sugiere Goldberg [4], por lo que será del orden del 1%. El tamaño de población manejado para este ejemplo será de 50 cromosomas, y se correrá el algoritmo genético durante 20 generaciones.

4) Resultados : El resultado obtenido en una corrida típica es de $1.81 (en millones de pesos), correspondiente a los valores de C_1=4 millones, C_2=3 millones, C_3=1 millón y C_4=2 millones. Esta es la solución óptima, la cual se obtuvo originalmente mediante programación dinámica [13]. El tiempo que le tomó al algoritmo genético encontrar este valor fue de sólo 13 segundos[3]. Debe hacerse notar que, en este caso, si deseáramos analizar inversiones que sumen otra cantidad, y en unidades menores al millón, el algoritmo genético tendría que modificarse de manera mínima, mientras que la programación dinámica requeriría una cantidad tal de trabajo que prácticamente se volvería inoperante.

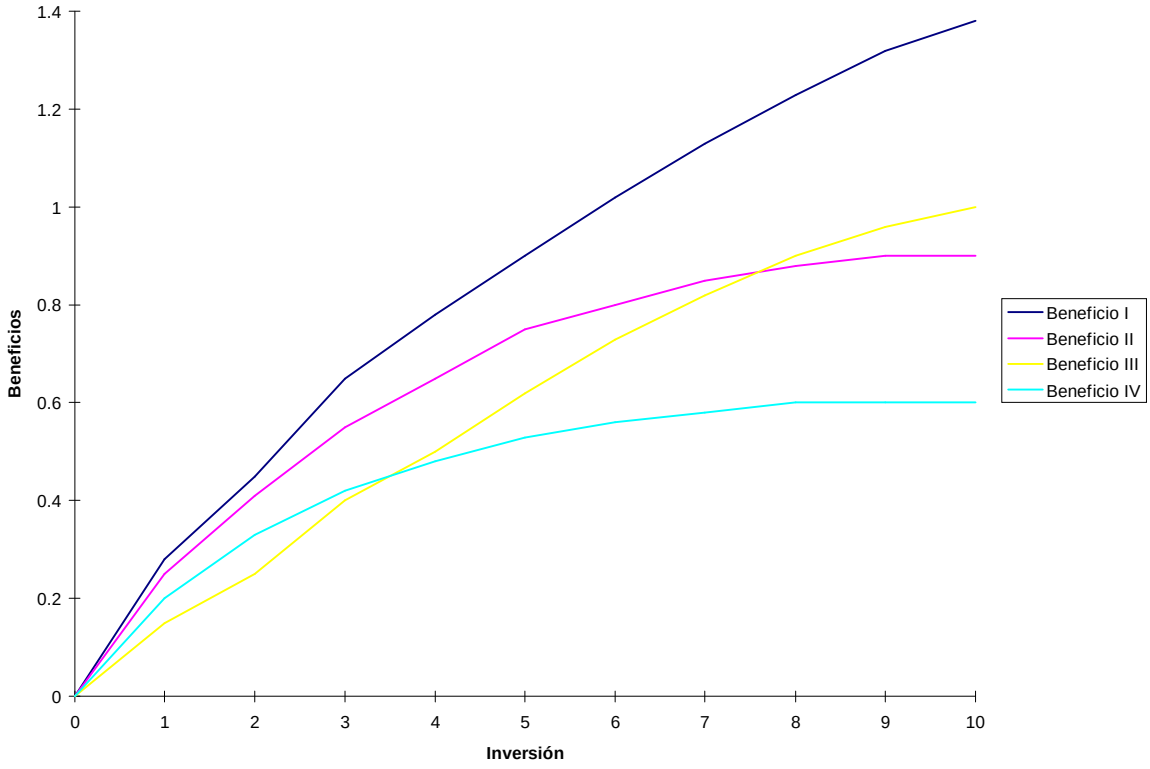

Figura 5 : Gráfica de los valores de la Tabla 2.

Aunque éste es sólo un sencillo ejemplo del uso del algoritmo genético para resolver problemas de optimización con restricciones, puede alcanzar a percibirse el poder de la técnica en comparación con los métodos tradicionales de búsqueda. Para saber más acerca de las múltiples aplicaciones que se les han encontrado a los algoritmos genéticos se recomienda consultar [4], [5] y [6]. Para aprender todos los detalles de implementación elementales, resulta de gran utilidad analizar el SGA (*Simple Genetic Algorithm*) incluído en [4]. Este programa está escrito en Turbo Pascal, aunque también existe una versión de dominio público en C, disponible a través del Internet[4] [8]. El autor ha proporcionado a los editores de esta revista el código completo en Turbo Pascal 7.0 de un algoritmo genético basado en el SGA de Goldberg, el cual se utilizó

---

[3] Las pruebas se corrieron en una computadora IBM PC Compatible 486DX/2 de 66 MHz con coprocesador matemático.

[4] Puede obtenerse usando **ftp** con '*anonymous*' en el nodo **ftp.aic.nrl.navy.mil**, bajo el directorio **/pub/galist/src/ga**.

para resolver el ejemplo incluído en esta sección. Este programa, a diferencia del presentado en [4], consta de un módulo único que hace uso de manejo dinámico de memoria, valiéndose de una técnica propuesta por Ken Porter [14], y contiene las modificaciones necesarias para incorporar cruza de 2 puntos y selección mediante torneo binario, entre otras cosas. El análisis de este código bien puede ser el punto de partida para los interesados en implementar un algoritmo genético.

Ambientes de Programación

En la actualidad existe un gran número de ambientes de programación disponibles en el mercado para experimentar con los algoritmos genéticos. De acuerdo a la taxonomía sugerida en [9], pueden distinguirse 3 clases de ambientes de programación:

1) Sistemas Orientados a las aplicaciones : Son esencialmente "cajas negras" para el usuario, pues ocultan todos los detalles de implementación. Sus usuarios -normalmente neófitos en el área- los utilizan para un cierto rango de aplicaciones diversas, pero no se interesan en conocer la forma en qué éstos operan. Ejemplos de este tipo de sistemas son: **Evolver** (Axcelis, Inc.) y **XpertRule GenAsys** (Attar Software).

2) Sistemas Orientados a los algoritmos : Soportan algoritmos genéticos específicos, y suelen subdividirse en:

Sistemas de uso específico : Contienen un solo algoritmo genético, y se dirigen a una aplicación en particular. Algunos ejemplos son: **Escapade** (Frank Hoffmeister), **GAGA** (Jon Crowcroft) y **Genesis** (John Grefenstette).

Bibliotecas : Agrupan varios tipos de algoritmos genéticos, y diversos operadores (e.g. distintas formas de realizar la cruza y la selección). **Evolution Machine** (H. M. Voigt y J. Born) y **OOGA** (Lawrence Davis) constituyen 2 ejemplos representativos de este grupo.

En estos sistemas se proporciona el código fuente para que el usuario -normalmente un programador- pueda incluir el algoritmo genético en sus propias aplicaciones.

3) Cajas de Herramientas : Proporcionan muchas herramientas de programación, algoritmos y operadores genéticos que pueden aplicarse en una enorme gama de problemas. Normalmente se subdividen en:

Sistemas Educativos : Ayudan a los usuarios novatos a introducirse de forma amigable a los conceptos de los algoritmos genéticos. **GA Workbench** (Mark Hughes) es un buen ejemplo de este tipo de ambiente.

Sistemas de Propósito General : Proporcionan un conjunto de herramientas para programar cualquier algoritmo genético y desarrollar cualquier aplicación. Tal vez el sistema más conocido de este tipo es **Splicer** (NASA).

Areas de Investigación

Durante los últimos años una gran parte de la investigación en esta área se ha concentrado en el desarrollo de mejoras al desempeño de los algoritmos genéticos. Se han propuesto nuevas técnicas de representación, selección y cruza, con resultados muy alentadores. Por ejemplo, el uso de los códigos de Gray y la codificación dinámica han superado algunos de los problemas asociados con la representación de valores reales mediante cadenas binarias. También se han propuesto técnicas adaptativas que varían dinámicamente los parámetros de control (porcentajes de mutación y cruza), en contraposición con el esquema estático tradicional. Otras innovaciones notables son los algoritmos genéticos distribuidos y los algoritmos genéticos paralelos. Una buena recopilación de este trabajo puede encontrarse en [10]

Por otra parte, el fundamento matemático de los algoritmos genéticos sigue siendo otra área abierta de investigación. Resulta de especial interés el desarrollo de modelos de la dinámica de los algoritmos genéticos, el análisis de problemas que resultan difíciles para la técnica, pruebas de convergencia y en general, el tratar de comprender mejor cómo funcionan. Para los interesados en saber qué se ha logrado en estas áreas teóricas, se recomienda consultar [11] y [12].

Probablemente uno de los trabajos de investigación más notables en el área sea el realizado por John Koza, que desarrolló una técnica de cruza que permite evolucionar expresiones-S de un programa en LISP. Esta innovadora forma de programación automática puede aplicarse a un enorme número de problemas, y por tal razón Koza -que por cierto fue discípulo de Holland, al igual que Goldberg- se apresuró a patentar sus algoritmos y a publicar un libro [2] que reporta 70 problemas de prueba muy populares entre los expertos en computación. Su obra incluye código en Common LISP para que los interesados puedan experimentar con lo que ha dado en llamarse **Programación Genética**. Yendo todavía más lejos, John Koza nos muestra en un libro de más reciente publicación [15] la forma en que su técnica puede aplicarse a problemas de mayor complejidad, y da testimonio de implementaciones realizadas en lenguajes procedurales como C y C++, rompiendo así el mito de que sólo un lenguaje de las características propias de LISP podría utilizarse para la programación genética. La propuesta de Koza ha sido acogida con beneplácito por un gran número de investigadores alrededor del mundo, sobre todo en el área de aprendizaje máquina.

Conclusiones

Se ha tratado de proporcionar de forma breve y concisa un panorama general de lo que son los algoritmos genéticos y su utilidad, sin que se llegara a profundizar mucho en ninguno de sus aspectos en particular. Las referencias bibliográficas proporcionadas deberán servir para que el lector interesado pueda averiguar más detalles por su cuenta. De cualquier forma, se recomienda a todo aquel interesado en trabajar en esta área contar al menos con las referencias [1] y [4]. Estos 2 libros se complementan, pues el primero proporciona una sólida base matemática de la técnica, y el segundo ahonda en los detalles de implementación, y explica muchos aspectos de su funcionamiento de una manera sencilla y agradable.

El éxito actual de los algoritmos genéticos se refleja en tres conferencias bianuales, una nueva revista internacional dedicada al tema[5] y un sinnúmero de publicaciones alrededor del

---

[5] *Evolutionary Computation*, MIT Press, Boston, Mass.

mundo. Este interés se ha visto ya reflejado en nuestro país, puesto que centros de investigación como LANIA (Laboratorios Nacionales de Informática Avanzada), así como científicos independientes han publicado varios trabajos sobre el tema. Es de esperarse que en los años siguientes más gente se sentirá atraída por esta novedosa técnica, por lo que tal vez lo más conveniente resulte saber aunque sea lo mínimo necesario sobre ella. De esa forma no nos sentiremos tan fuera de sitio en la siguiente charla que tengamos con otros profesionales de la computación.

Referencias

[1] Holland, John H. "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

[2] Koza, John R. "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

[3] Buckles, Bill P. y Petry, Frederick E. "Genetic Algorithms", IEEE Computer Society Press, 1992, 109 p.

[4] Goldberg, David E. "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Publishing Company, 1989, 412 p.

[5] Booker, L. B.; Goldberg, D. E. y Holland, J. H. "Classifier Systems and Genetic Algorithms", en *Artificial Intelligence*, 40, 1989, pp. 235-282.

[6] Davis, Lawrence (Editor). "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991, 385 p.

[7] Forrest, Stephanie. "Genetic Algorithms: Principles of Natural Selection Applied to Computation", en *Science*, Vol. 261, No. 5123, Agosto 13 de 1993, pp. 872-878.

[8] Smith, Robert E.; Goldberg, David, E. y Earickson, Jeff A. "SGA-C : A C-language Implementation of a Simple Genetic Algorithm", TCGA Report No. 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Mayo 14 de 1991.

[9] Filho, José L. Ribeiro; Treleaven, Philip C. y Alippi, Cesare. "Genetic-Algorithm Programming Environments", en *IEEE Computer*, Junio de 1994, pp. 28-43.

[10] Srinivas, M. y Patnaik, Lalit M. "Genetic Algorithms : A Survey". en *IEEE Computer*, Junio de 1994, pp. 17-26.

[11] Rawlins, Gregory J. E. (Editor). "Foundations of Genetic Algorithms", Morgan Kaufmann Publishers, 1991, 341 p.

[12] Whitley, L. Darrell (Editor). "Foundations of Genetic Algorithms 2", Morgan Kaufmann Publishers, 1993, 322 p.

[13] Kaufmann, A. y Faure, R. "Invitación a la Investigación de Operaciones", Segunda Edición, CECSA, México, 1977, 311 p.

[14] Porter, Kent. "Handling Huge Arrays", en Dr. Dobb' s Journal of Software Tools for the Professional Programmer, Vol. 13, No. 3, 1988, pp. 60-3.

[15] Koza, John, "Genetic Programming II : Automatic Discovery of Reusable Programs", The MIT Press, 1992, 746 p.

Lecturas Adicionales

- Holland, John H. "Genetic Algorithms", en *Scientific American* 267, Julio de 1992, pp. 66-72.
- Michalewicz, Zbigniew. "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, 1992, 250 p.
- Denning, Peter J. "Genetic Algorithms", en *American Scientist*, Volume 80, January-February 1992, pp. 12-14.