La selección natural como generadora de orden espontáneo a partir del caos, y sus aplicaciones en ingeniería

Natural selection as generating factor of spontaneous order from chaos, and its applications in engineering

COLCIENCIAS TIPO 2. ARTÍCULO DE REFLEXIÓN
RECIBIDO: MAYO 17; ACEPTADO: JUNIO 16, 2012

Julio César Millán Barco, M.Sc
juliocesarmillanbarco@yahoo.es

Universidad Santiago de Cali, Colombia

Resumen

Este artículo muestra una introducción a los algoritmos genéticos, las características de los mismos y las ventajas de su utilización. El objetivo del mismo es exponer la forma en que agentes individuales pueden constituir un sistema organizado si siguen unas reglas específicas. Se plantea como ejemplo la investigación hecha para producir una frase coherente a partir de letras en desorden por medio de los principios de la selección natural utilizando los citados algoritmos. Se indican los principios de este tipo de algoritmos, la manera en que fueron implementados y sus resultados, además de algunas conclusiones con respecto a los alcances de este tipo de técnicas en diferentes campos de la ciencia.

Palabras Clave
Algoritmos genéticos, selección natural, sistemas de orden espontáneo.

Abstract

This paper shows an introduction to genetic algorithms, the same characteristics and advantages of their use. Its objective is to show how individual agents can be an organized system if they follow specific rules. Example is presented as research done to produce a coherent sentence from letters in disorder through the principles of natural selection using the above algorithms. Indicates the beginning of such algorithms, the way they were implemented and their results, along with some conclusions regarding the scope of these techniques in different fields of science.

Keywords
Genetic algorithms, natural selection, systems of spontaneous order.

I. INTRODUCCIÓN

La aplicación de sistemas *bioinspirados* es uno de los campos en que el ser humano ha intentado imitar las maravillas de la naturaleza con el fin de dominarla. Eso hace parte de la ingeniería que toma los conocimientos de las llamadas ciencias puras y los convierte en tecnología útil para resolver problemas específicos. Es posible ver ejemplos de sistemas bioinspirados en el sonar, el radar, los aviones o la vista telescópica.

Para muchas personas es difícil entender la complejidad de ciertos aspectos de la naturaleza como originadas a partir del azar. En realidad, esa complejidad no proviene de la simple aleatoriedad. Hace más de 150 años el naturalista británico Charles Darwin descubrió la manera en que la gran cantidad de especies de seres vivos se han especializado para sobrevivir a diferentes ambientes. Su trabajo fue fruto de un viaje épico de cinco años alrededor del mundo y de la acumulación de evidencia desde que regresó de esa odisea. La genialidad de Darwin se ha demostrado con el paso del tiempo, pues su teoría de la selección natural, que explica la biodiversidad terrestre, fue enunciada en un tiempo en que no se conocía la genética, ni el ADN, ni la deriva continental, ni la tectónica de placas, ni los conceptos de la geología moderna.

En la ciencia, una teoría es mejor si utiliza el llamado concepto de la cuchilla de Okham, es decir, si explica la mayor cantidad de cosas con el menor número de enunciados. La teoría de la selección natural, que explica el complejo hecho de la evolución, es simple. En la naturaleza sobreviven los individuos que logran adaptarse mejor a un ambiente específico, aumentando su probabilidad de reproducirse y dejar como herencia a sus descendientes precisamente aquello que los hace vivir más tiempo y replicarse de manera más efectiva que otros.

El trabajo realizado para este artículo consiste en un experimento simple pero concreto, para lograr orden a partir de unas piezas en desorden. Se construye una frase coherente del español a partir del conjunto de letras del alfabeto colocadas en diferentes posiciones sin orden alguno. Para ello se utiliza la técnica de los algoritmos genéticos que reúne métodos inspirados en la selección natural para alcanzar soluciones a problemas específicos sin emplear complejas operaciones matemáticas. La idea central que rodea al artículo es exponer una forma en que la adaptabilidad genera orden espontáneo tal como lo afirma Holland (2004), considerado el creador de los algoritmos genéticos.

En los primeros capítulos se dan a conocer generalidades sobre los algoritmos genéticos para luego mostrar la forma en que se implementó uno para organizar un conjunto de letras de tal manera que formaran una frase coherente. Al final se hacen algunas conclusiones sobre el trabajo realizado y especialmente se extrapolan algunas ideas sobre el uso de sistemas bioinspirados y sobre la creación de un orden espontáneo.

II. LOS MECANISMOS DE LA EVOLUCIÓN Y LOS ALGORITMOS GENÉTICOS

Los mecanismos de la evolución son básicamente tres: primero, la selección de los individuos. La lucha por la supervivencia es una descarnada aventura por los paisajes más hostiles. La naturaleza es amoral. En ella se observan hongos y parásitos que crecen a expensas de la vida de otros, depredadores implacables como los felinos que cazan mamíferos, hembras que se alimentan de la cabeza del macho que las fecunda, plantas que transforman la energía solar en alimento para las demás variedades de seres vivos, hormigas o abejas que se sacrifican para que su comunidad sobreviva, pingüinos que arrojan al mar a sus semejantes para que los leones marinos que los acechan calmen su hambre, osos polares blancos como el hielo, ranas verdes como una hoja selvática, mariposas negras como el carbón, miles de espermatozoides para una cantidad limitada de óvulos. Todo esto lo explica la selección natural.

El siguiente mecanismo es la reproducción que logran los que no mueren antes. Hoy la teoría sintética de la selección natural explica por qué los hijos, si tienen suerte, heredan las cualidades que hacen sobrevivir a sus padres. Gracias a la genética sabemos que este mecanismo tiene su origen en cada célula de un ser vivo que cuenta con su propio ADN, la secuencia de la vida.

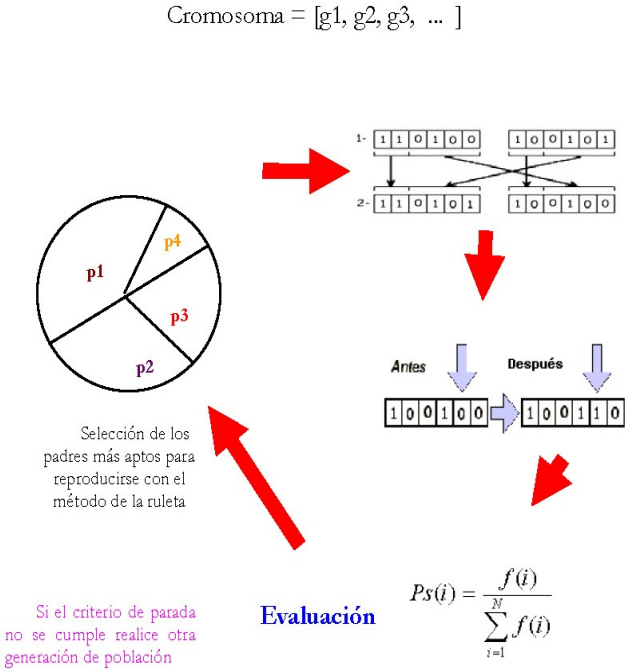
En principio, los genes tienden a inmortalizarse a través de los individuos en que habitan y al que configuran. Sin embargo, esta copia en la reproducción no es perfecta. Aquí sucede el tercer mecanismo de la evolución: la mutación. Se tiende a relacionar la mutación con una malformación. No obstante, es precisamente la mutación la que hace posible la variabilidad de la vida y por tanto su supervivencia y evolución. Lo que al principio puede parecer un defecto se convierte en una característica que juega un papel importante en el desarrollo del individuo

dentro de medios ambientes cambiantes. La Figura 1 muestra un procedimiento clásico de algoritmos genéticos, aunque no de carácter obligatorio para la implementación de todo algoritmo evolutivo, ya que es factible cambiar la forma en que se hace la selección, la reproducción y la mutación.

Entendidos estos tres mecanismos, es posible asimilarlos para su uso en aplicaciones para ingeniería. Los algoritmos genéticos son técnicas bioinspiradas basadas en la selección natural que se dirigen, en general, a la optimización de funciones cuya complejidad impida hacer los cálculos necesarios para utilizar los métodos clásicos –que se apoyan en el gradiente– ya que este es difícil de calcular en muchísimos casos. Los algoritmos genéticos no utilizan las derivadas de la función a optimizar, por lo que su aplicabilidad se extiende de manera más amplia.

Los algoritmos genéticos empiezan definiendo cromosomas de individuos de una población. El cromosoma será un vector de parámetros, valores, caracteres o lo que sea necesario para la aplicación específica.

Figura 1. Operaciones genéticas básicas



Se llena una población de determinados individuos con esos cromosomas. Al inicio sería aleatorio. Luego, se hace una selección mediante una función de adaptación que es un indicador de que tan cerca se encuentra de la solución del problema.

Adaptación = fevaluacion(cromosoma)

Luego se saca una función de aptitud que es la función de evaluación dividida entre lo que se espera de ella:

$$Fitness = fevaluación \;/\; Fmáx$$

Donde: *Fmáx* es lo que se espera de la solución que da el cromosoma evaluador.

A. Algoritmos genéticos binarios

A veces se hace necesario que las soluciones a un determinado problema sean representadas en algoritmos genéticos con números binarios (Haupt & Haupt, 1998). En este caso la representación de cada cromosoma se divide en genes o bits. La codificación sería así:

Dado el parámetro p_n se puede codificar

$$pnorm = \frac{p_n - p_{min}}{p_{max} - p_{min}}$$

$$gene[m] = redondeo\left\{p_{norm} - 2^{-m} - \sum_{p=1}^{m-1} gene[m]2^{-p}\right\}$$

Para decodificar:

$$p_{cuant} = \sum_{m=1}^{Ngene} gene[m]2^{-m} + 2^{-(M+1)}$$

$$q_n = p_{norm}(p_{máx} - p_{mín}) + p_{mín}$$

Donde:

p_{norm} = *parámetro normalizado* $0 \leq p_{norm} \leq 1$

pmín = valor del parámetro más bajo

pmáx = valor del parámetro más alto

Gene[m] = versión binaria de p_n

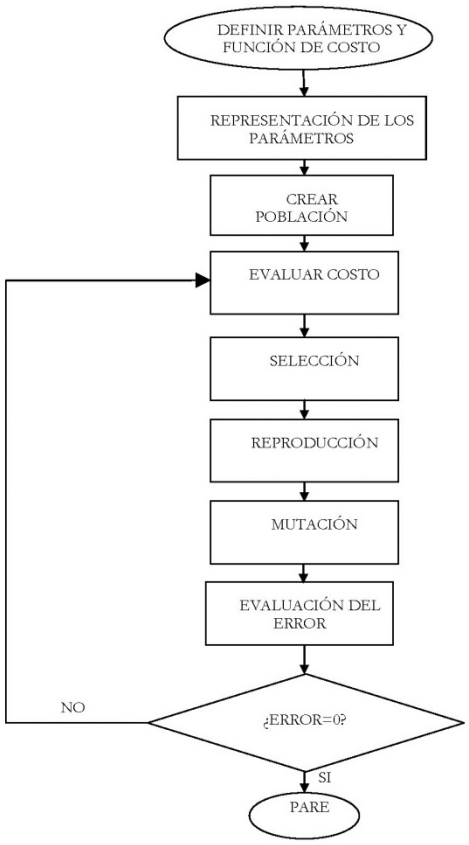
redondeo{.} = redondeo al entero más cercano

Ngene = número de bits en el gen

p_{cuant} = *versión cuantizada de* p_{norm}

q_n = *versión cuantizada de* p_n

Figura 2. Diagrama de flujo Algoritmo Genético Binario



B. *Selección*

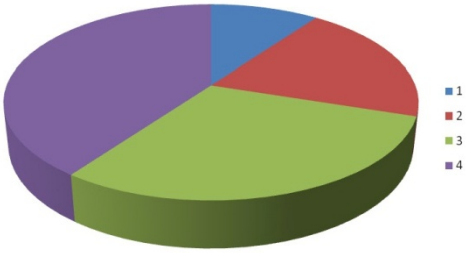
La selección de los mejores individuos se puede hacer de diferentes maneras, una de ellas, tal vez la más usada, es el mecanismo de ruleta. A cada individuo se le asigna una probabilidad de ser seleccionado de acuerdo a la función de adaptabilidad y en un siguiente proceso se van seleccionando individuos donde, por supuesto, los de mejor *fitness* tienen mayor probabilidad de ser elegidos.

Hay otros métodos de selección como el elitismo, donde solo se eligen los mejores. La ventaja del caso de la ruleta es que en un individuo de poca probabilidad, puede haber escondida una solución ingeniosa a un problema concreto, la convergencia puede demorar más pero puede lograrse al final de suficientes generaciones, en lugar de que el algoritmo se quede atrapado en un mínimo local.

Tabla 1. Asignación de probabilidad de selección

Individuo	Probabilidad de selección
1	20
2	15
3	35
4	30

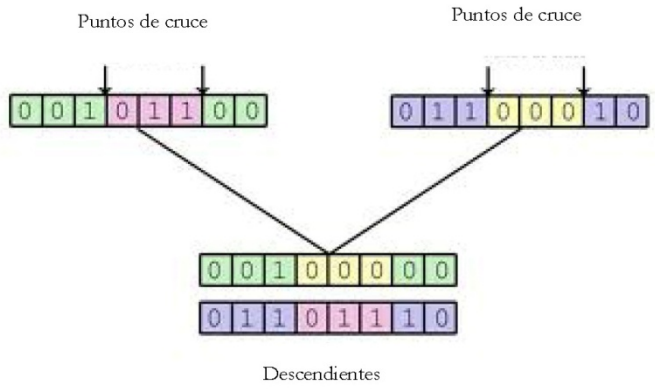
Figura 3. Clasificación por el mecanismo de la ruleta



C. *Cruce*

En la reproducción se elige una parte de un cromosoma, para que sea parte de un cromosoma hijo que se complementa con la parte faltante de otro cromosoma padre. Los puntos de cruce pueden ser elegidos al azar. Este mecanismo se inspira en que en la reproducción biológica cada padre aporta algo de su genotipo al nuevo individuo.

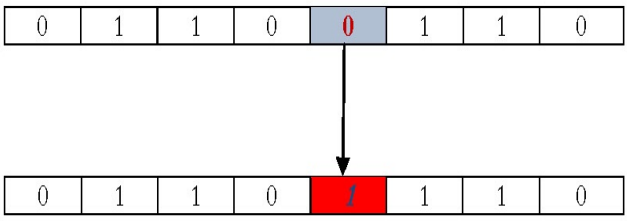
Figura 4. Explicación del cruce genético binario



D. *Mutación*

Una mutación es un cambio en la información genética que produce cambios en el individuo y que puede ser heredada. Las copias que se hacen después de una reproducción pueden sufrir mutaciones que a veces generan cambios favorables, aunque en general son perjudiciales y se notan mucho en el fenotipo del individuo.

Figura 5. Representación de la operación mutación genética

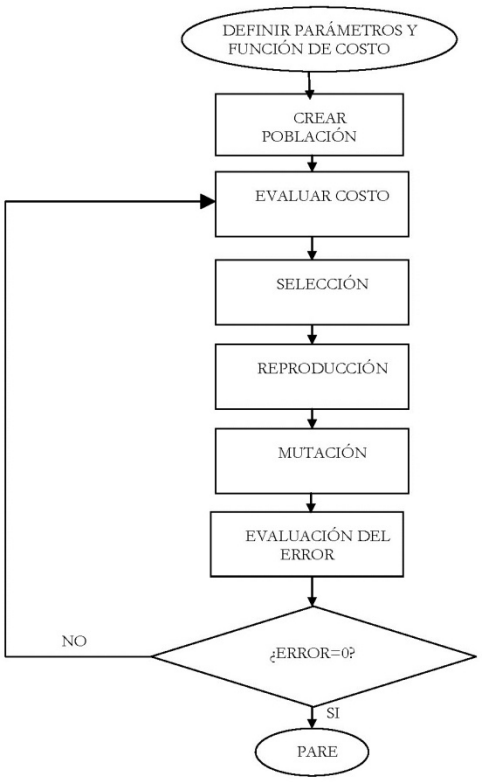


En este caso, para algoritmos genéticos binarios, hay cambio en un gen representado por un bit. En el algoritmo, estas mutaciones se hacen aleatoriamente para dar cabida a posibles soluciones gracias a la variabilidad genética.

E. Algoritmos genéticos continuos

Los algoritmos genéticos funcionan con códigos binarios, pero las aplicaciones requieren usualmente funciones de aptitud o de adaptabilidad que sean parámetros continuos. La limitación de la cuantificación hace que se usen con mayor aplicabilidad algoritmos genéticos con individuos que estén representados con números reales continuos. Con la introducción de la representación continua se puede alcanzar mayor precisión. El diagrama de flujo representa un algoritmo genético continuo.

Figura 6. Diagrama de flujo Algoritmo Genético Continuo



Se puede notar que no es necesario hacer la cuantificación, lo que permite al algoritmo funcionar con mayor precisión, especialmente ayudando a encontrar los valores mínimos en funciones de difícil convergencia.

F. Selección

La selección en algoritmos genéticos continuos se hace tal como en los binarios. Se puede hacer una ruleta, un torneo o simplemente aplicar elitismo. A cada individuo se le aplica un costo de acuerdo a su adaptabilidad; es decir a la forma en que se acerca a la solución del problema.

G. Cruce

La reproducción o cruce en algoritmos continuos se puede hacer por apareamiento de parámetros. Por ejemplo, si un padre tiene un cromosoma como el siguiente:

$$Padre1=[pm1, pm2, pm3,\ldots, pmNpar]$$

$$Padre2=[pd1,pd2,pd3,\ldots, pdNpar]$$

Los descendientes podrían ser:

$$Hijo1=[pm1, pd2,pm3,\ldots,pmNpar]$$

$$Hijo2=[pd1,pm2,pd3,\ldots,pdNpar]$$

Y los siguientes descendientes podrían hacerse con diferentes puntos de cruce para favorecer la variabilidad genética.

El problema con este tipo de cruces es que estos métodos no agregan información nueva, lo cual se puede resolver con un cambio en los parámetros, estableciendo una ecuación como la siguiente:

$$Pnew= \beta pmn + (1- \beta)pdn$$

Donde β es un número aleatorio en el intervalo de 0 a 1

pmn es el parámetro n en el cromosoma de padre1

pdn es el parámetro m en el cromosoma de padre2

H. Mutación

Los algoritmos genéticos continuos pueden converger rápidamente a soluciones adecuadas a problemas precisos. Sin embargo, una de las cosas que logra que funcionen es la mutación. En algoritmos con cuantificaciones continuas se aplica una rata de mutación para cambiar algunos parámetros y generar mayor variabilidad genética. Un algoritmo de mutación bien podría seguir el siguiente código:

```
Num_Mut=ceil(Num_Cromo*Num_Par*Rata_mutacion);
Poblacion_Mut=Poblacion;
for in=1:Num_Mut
     fila    = round((Num_Cromo-1)*rand + 1);
```

```
        columna = round((Num_Par-1)*rand + 1);

Poblacion_Mut(fila,columna)=(Rango(2,columna)-
Rango(1,columna))*rand + Rango(1,columna);
   End
```

Se nota que se elige al azar el cromosoma de la población a mutar; asimismo, el parámetro del cromosoma a cambiar. El número de mutaciones está regido por la rata de mutación.

I. Algoritmos genéticos con caracteres

Para este artículo se implementó un programa en MATLAB que utiliza algoritmos genéticos con el objetivo de obtener una frase coherente a partir de las letras del alfabeto mezcladas al azar. Se parte de una frase aleatoria, sin sentido, y se llega a una oración con significado. Por supuesto, hay que aclarar que en la naturaleza no hay funciones objetivo predeterminadas, en ella simplemente se da la lucha cotidiana por sobrevivir sin un direccionamiento establecido.

Los métodos clásicos de optimización tienen la ventaja de que son computacionalmente rápidos de implementar siempre y cuando conozcamos la función a evaluar y que esta sea derivable en el tiempo para obtener su gradiente y con él ir minimizándola. Una desventaja es que eso hace que sea susceptible de ser atrapada en un mínimo local. Esto es especialmente preocupante si la curva de la función objetivo contiene muchos valles aislados o si tiene demasiadas variables a ser evaluadas.

Los algoritmos genéticos tienen características atractivas para los problemas de optimización porque ellos no requieren modelos específicos o linealidades como en las aproximaciones clásicas y pueden explorar todas las partes del espacio de los parámetros (probablemente incierto). En criptografía, por ejemplo, los algoritmos genéticos pueden ayudar a desencriptar mensajes a partir de frases sin sentido; también, dado que otro de los rasgos importantes de los algoritmos genéticos es la búsqueda que hacen sin utilizar técnicas avanzadas de cálculo diferencial (ubicándolos dentro de las técnicas de gradiente libre que más utilidad tienen en los procesos de optimización de funciones), son capaces de llegar a soluciones efectivas al problema del *agente viajero* (Haupt & Haupt, 1998) o al de la asignación de tareas en un *sistema embebido multinúcleo* (Ezzatti & Nesmachnow, S/F; Jhumka, Klaus, & Huss, 2005).

Comparado con los métodos de optimización no lineales convencionales, los algoritmos genéticos ofrecen hipotéticamente, según Ji y Zhang (2001) las siguientes ventajas claves:

1) Autonomía

Los algoritmos genéticos no requieren estimaciones iniciales. El conjunto de parámetros iniciales es generado aleatoriamente en el dominio de parámetros ya predefinido.

2) Robustez

Conceptualmente, los algoritmos genéticos trabajan con unas poblaciones ricas y simultáneamente suben muchos picos en paralelo durante el proceso de búsqueda. Esto reduce significativamente la probabilidad de quedar atrapado en un mínimo local.

3) Inmunidad al ruido

Los algoritmos genéticos buscan un conjunto de parámetros adecuados y se mueven hacia el óptimo global por la reducción gradual de la posibilidad de reproducirse de los conjuntos de parámetros no adecuados (o con mal *fitness*). Esto además tiene un potencial de alta exactitud en situaciones de ruido.

En este trabajo de investigación se parte del conocimiento de estas tres hipotéticas ventajas y se muestra que simulando artificialmente las leyes de la selección natural de manera computacional se puede llegar a crear sistemas complejos a partir de un caos primigenio. Es lo que John Holland (2004) llamaría la complejidad que crea la adaptabilidad en un orden oculto: cada uno de los agentes que intervienen no sabe exactamente cómo funcionan juntos, pero lo logran.

III. IMPLEMENTACIÓN COMPUTACIONAL DE LAS LEYES DE SELECCIÓN NATURAL

El programa en MATLAB realiza las operaciones básicas de la selección natural que intentamos imitar. Primero se le indica al programa las variables a usar, es decir, las letras del alfabeto y los signos de puntuación, enseguida se establece la función objetivo, es decir, la frase a la que queremos llegar. En este caso la opción es por una frase muy recordada en la literatura colombiana (aunque el programa está adaptado para llegar a cualquier frase objetivo). La frase en cuestión fue tomada del inicio de la obra *Cien años de soledad*. Es *[…] Muchos años después, frente al pelotón de fusilamiento (sic) el coronel Aureliano Buendía había de recordar aquella tarde remota en que su padre lo llevó a conocer el hielo…"*

```
clc

clear all;

Vector_Alfabeto                                    =
'AÁBCDEÉFGHIÍJKLMNÑOÓPQRSTUÚVXYZaábcdeéfghiíjklmnñoó
pqrstuúvwxyz ,.'

Vector_Frase = 'Muchos años después, frente al
pelotón de fusilamiento el coronel Aureliano
Buendía había de recordar aquella tarde remota en
que su padre lo llevó a conocer el hielo';

NumerodeletrasAlfabeto = size(Vector_Alfabeto,2;

NumerodeletrasFrase = size(Vector_Frase,2);
```

A continuación, se crea de manera aleatoria una población de varias frases, cada una con el mismo número de caracteres que la frase objetivo. En este punto es importante aclarar que se eligieron 500 frases aleatorias porque es el tamaño de la población adecuado para brindar la mejor optimización. Las poblaciones grandes tienen la ventaja de otorgar mayor variabilidad y, por tal motivo, mayor posibilidad de evitar caer atrapados en mínimos locales. Entre más individuos, mejor simulación de un procesamiento en paralelo y mayor oportunidad de que alguno de los fenotipos de la cantidad generada se adapte mejor. La desventaja es que se requiere mayor capacidad de cálculo en cada generación, algo que al principio puede parecer más lento pero que al final llega a la convergencia en menor número de poblaciones, mejorando a un algoritmo que tenga una población pequeña, que corre el riesgo de demorarse muchas generaciones en encontrar el objetivo o incluso no hallarlo.

```
Numerodeindividuos=500;

for j=1:1:Numerodeindividuos

  for i=1:1:NumerodeletrasFrase

      genes=randperm(NumerodeletrasAlfabeto);

      gen=genes(1);

      Vector_Frase_Nueva(j,i)=
(Vector_Alfabeto(gen));

  end

end
```

A. Selección

Luego se pasa a una evaluación de cada una de las frases de la población comparándola con la frase objetivo. El segundo paso es la asignación de la adaptabilidad de cada frase o su *fitness* de acuerdo al número de coincidencias con la frase asignada. Basándose en el número de aciertos, se ubican las dos primeras frases con mayores números de coincidencias. Esas dos primeras frases se convierten en padres de una siguiente población.

Esta técnica de seleccionar directamente a los mejores para que se reproduzcan es conocida como elitismo, pues solo la élite de individuos más adaptados propaga sus características a sus hijos. En general, la manera de selección de individuos puede variar de aplicación a aplicación de acuerdo a la necesidad. Es posible utilizar otros métodos, como el de la ruleta, en el que los individuos que sobreviven tienen una probabilidad más alta de reproducirse, aunque esta no es igual a uno. En el elitismo esta probabilidad si es igual a uno y son los hijos de los "mejores" individuos los que llenan de nuevo la población en la siguiente generación (Chang, 2003).

```
for x=1:Numero_de_Generaciones
 x;
Vector_Frase_Nueva
   for i=1:1:Numerodeindividuos
        if Vector_Frase_Nueva(i,:)==Vector_Frase
        Ganador =  Vector_Frase_Nueva(i,:)
        break;
   end
end


aciertos(1:1:Numerodeindividuos,1)=zeros(Numerode
individuos,1);
for i=1:1:Numerodeindividuos
    for j=1:1:NumerodeletrasFrase
        if
Vector_Frase_Nueva(i,j)==Vector_Frase(1,j)
         aciertos(i)=aciertos(i)+ 1;
        end
    end
end
[max(aciertos) x]
pause(0.1)
%Selección de individuos
% Cromosoma_Ganador


for i=1:1:Numerodeindividuos
    Vector_Frase_Nueva;
        if Vector_Frase_Nueva(i,:)==Vector_Frase
% %         Ganador =  Vector_Frase_Nueva;
        break;
        end
```

```
    end


   [Primer_Numero_mayor_de_aciertos,Ubicacion_Cromos
oma_Ganador_Hembra]= max(aciertos);

   Cromosoma_Ganador_Hembra                          =
Vector_Frase_Nueva(Ubicacion_Cromosoma_Ganador_Hembr
a,:);

     if Cromosoma_Ganador_Hembra==Vector_Frase;

       break;

     end

   Vector_Frase_Nueva(Ubicacion_Cromosoma_Ganador_He
mbra,:)= zeros(1,NumerodeletrasFrase);

    aciertos(Ubicacion_Cromosoma_Ganador_Hembra,1)  =
0;

   [Segundo_Numero_mayor_de_aciertos,Ubicacion_Cromo
soma_Ganador_Macho]= max(aciertos);

   Cromosoma_Ganador_Macho                           =
Vector_Frase_Nueva(Ubicacion_Cromosoma_Ganador_Macho
,:);
```

B. Cruce

El cruce entre dos individuos para reproducirse también queda a disposición del programador (Herrera, 1997) en cuanto al punto –o puntos– de cruce que desea como aportante de cada individuo. Para este artículo se programó un punto de cruce justo en la mitad del cromosoma de cada progenitor; esto, aunque arrojó buenos resultados, es susceptible de mejorar (Zhang, Sakamoto, & Furutani, 1998).

```
   Punto_de_Cruce = floor(NumerodeletrasFrase/2);

   Cromosoma_Ganador_Hijo1_Parte1
=Cromosoma_Ganador_Macho(1,1:Punto_de_Cruce);

   Cromosoma_Ganador_Hijo1_Parte2=
   Cromosoma_Ganador_Hembra(1,Punto_de_Cruce+1:Numer
   odeletrasFrase);


   Cromosoma_Ganador_Hijo2_Parte1
=Cromosoma_Ganador_Macho(1,1:Punto_de_Cruce);

   Cromosoma_Ganador_Hijo2_Parte2=
   Cromosoma_Ganador_Hembra(1,Punto_de_Cruce+1:Numer
   odeletrasFrase);

   Cromosoma_Ganador_Hijo1                           =
[Cromosoma_Ganador_Hijo1_Parte1
Cromosoma_Ganador_Hijo2_Parte2];

   Cromosoma_Ganador_Hijo2                           =
[Cromosoma_Ganador_Hijo2_Parte1
Cromosoma_Ganador_Hijo1_Parte2];

   mitad_de_individuos=floor(Numerodeindividuos/2);

   Vector_Frase_Nueva(1,:)=Cromosoma_Ganador_Hijo1;

   Vector_Frase_Nueva(2,:)=Cromosoma_Ganador_Hijo2;

   %%Se crean  copias idénticas

   MITAD=floor((Numerodeindividuos-2)/2);

   for i=3:1:MITAD

       for j=1:NumerodeletrasFrase

               Vector_Frase_Nueva(i,j)=
Cromosoma_Ganador_Hijo1(1,j);

       end

   end

   Cromosoma_Ganador_Hijo1(1,1:NumerodeletrasFrase)

   SIGUIENTE=MITAD+1;

   for i=SIGUIENTE:1:Numerodeindividuos

       for j=1:NumerodeletrasFrase

               Vector_Frase_Nueva(i,j)=
Cromosoma_Ganador_Hijo2(1,j);

       end

   end
```

C. Mutación

Estos hijos pueblan la siguiente generación de individuos, no sin antes pasar por una etapa de mutación de alguno de sus genes hecha de manera aleatoria, algo que le da variabilidad a la población, pues si no se hace, toda estaría llena de los mismos individuos. La mutación es la clave de por qué cada individuo es distinto a sus padres y a sus hermanos, si no todos serían clones. La mutación genera adaptabilidad, explora nuevos espacios y por situaciones relacionadas con el medio ambiente, hace que aquello que en principio parece un error se convierta en una ventaja para la supervivencia del individuo. En el caso del programa hecho para este artículo la mutación siempre se hace (Tzung, 1996).

```
   %MUTACIONES

   genes=randperm(NumerodeletrasAlfabeto);

         for j=1:1:Numerodeindividuos

            genes=randperm(NumerodeletrasAlfabeto);

            gen=genes(1);

            genes2=randperm(NumerodeletrasFrase);

            gen2=genes2(1);

            Vector_Frase_Nueva(j,gen2)=
(Vector_Alfabeto(gen));

          end
```

Con estos individuos hijos se genera la población y se inicia de nuevo el ciclo de selección, descarte, reproducción y mutación. El programa implementado

requiere de menos de 1000 generaciones para llegar a la frase objetivo.

```
for i=1:1:Numerodeindividuos
    Vector_Frase_Nueva;
        if Vector_Frase_Nueva(i,:)==Vector_Frase
                Ganador =  Vector_Frase_Nueva;
        break;
end
```

En conclusión, es posible generar orden a partir del caos imitando artificialmente las leyes de la selección natural. Un sistema bioinspirado con multitud de aplicaciones en ingeniería, especialmente en criptografía para descifrar mensajes codificados, en optimización de funciones, en asignación de tareas y en problemas de economía relacionados con el equilibrio de la demanda y la oferta y en teoría de juegos. Este algoritmo genético en particular puede tener aplicaciones en criptografía para la decodificación de mensajes cifrados.

IV. PRUEBAS Y RESULTADOS

Tabla 2. Comparación de tamaño de población contra tiempo empleado e iteraciones necesarias

Número de individuos	Número de iteraciones	Tiempo de CPU empleado(segundos)
200	917	155.0477
500	304	67.9852
1000	190	59.9093
2000	180	110.4672

En las siguientes gráficas se muestra el comportamiento del número de aciertos comparado contra el número de iteraciones. Para el caso en cuestión el número máximo de aciertos es de 166.

Figura 7. Número de aciertos contra número de iteraciones Población= 200 individuos

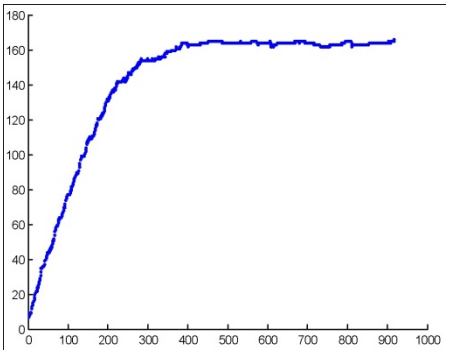


Figura 8. Número de aciertos contra número de iteraciones Población= 500 individuos

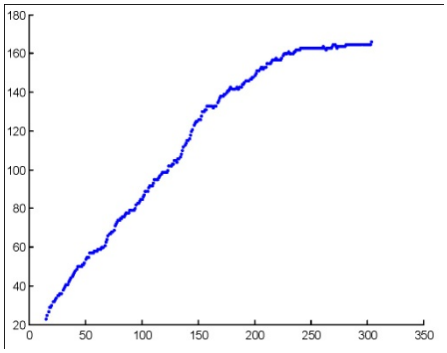


Figura 9. Número de aciertos contra número de iteraciones Población= 1000 individuos

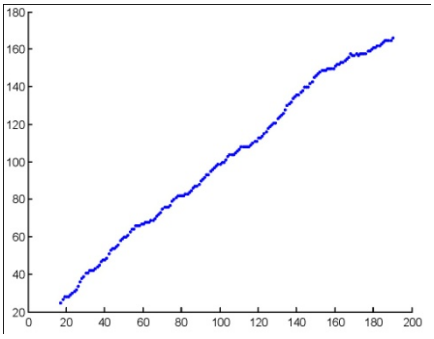
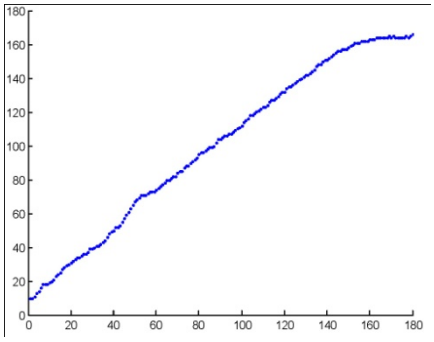


Figura 10. Número de aciertos contra número de iteraciones Población= 2000 individuos



El algoritmo genético fue probado para diferentes características y se alcanzaron resultados interesantes. Uno de ellos, saber que ante poblaciones más grandes el algoritmo converge en menos iteraciones, pero demora más en cada iteración, lo que tiene su equivalente biológico al apreciar que un individuo que quiera reproducirse más debe vivir una vida más larga. En la Tabla 2 se puede observar que, a pesar de que con más individuos se obtiene

la frase correcta en menor número de iteraciones, si se aumenta mucho la población puede que el tiempo empleado sea mayor debido a que requiere más cálculo en cada iteración. En el caso de algoritmos genéticos depende de la intuición propia del programador determinar qué cantidad es propicia a ser cambiada para obtener mejor rendimiento y mejores soluciones, dependiendo de la capacidad de cómputo de la máquina utilizada. Para profundizar en la influencia del tamaño de la población y la tasa de mutación en los resultados de algoritmos genéticos se recomienda consultar a Zhang (2008).

A medida que el programa va corriendo, es posible observar que las letras se van colocando en la posición "correcta" de la frase. En este punto se hace pertinente recordar que en la evolución biológica no existen los términos correcto o incorrecto. La evolución sigue ciegamente unas leyes fundamentales y no muestra evidencia de ser dirigida hacia un objetivo específico, cosa que si hace el algoritmo genético en cuestión, y en general todos los algoritmos evolutivos que buscan optimización.

En estos casos hay que observar muy bien el avance de la ejecución del algoritmo genético sobre el que se esté trabajando, pues es como una evolución en "cámara rápida". La atenta observación podrá determinar qué tasa de reproducción es adecuada, qué tanto influye el número de individuos sobre el desempeño computacional y, en general, qué variables se pueden escoger en los procesos de selección, reproducción y mutación.

V. OTROS SISTEMAS DE GENERACIÓN ESPONTÁNEA DE COMPLEJIDAD

En ciertos sistemas, existen agentes que actúan movidos por su propio interés o instinto, que realizan funciones básicas en un entorno cohabitado por agentes de la misma clase. Estos sistemas logran desarrollar, en conjunto, una complejidad cercana al concepto de inteligencia. Ejemplos de esos sistemas son los hormigueros, los enjambres de abejas, la economía de libre mercado, lo que en el siglo XVIII Adam Smith metafóricamente llamó *la mano invisible* (Smith, 1776) y el más importante: el cerebro humano. En él, habitan unas células llamadas neuronas que individualmente no realizan mayores procesos pero que en conjunto y en una altísima interconectividad desarrollan un grado de complejidad tal que producen nuestra mente. En el cerebro de cada individuo habita su ser, lo que es; todo el universo que modela alrededor suyo, lo hace a través de su cerebro. El cerebro permite la adaptación del ser humano y propaga sus genes de generación en generación. Es posible incluso afirmar que el ser humano es un invento del cerebro en evolución para mantener inmortales los genes que constituyen el ADN de ese ser humano, como expresan Antonio Damasio (2010) o Richard Dawkins (1985).

VI. CONCLUSIONES

Existen aplicaciones en ingeniería que pueden utilizar algoritmos genéticos, tales como la criptografía, la teoría de juegos, las negociaciones diplomáticas, la economía o la ingeniería industrial. Los algoritmos genéticos incluso pueden servir para obtener el mejor ruteado en las pistas de conexión de un circuito impreso. Un rasgo importante de los algoritmos genéticos es su capacidad de realizar búsquedas sin utilizar técnicas avanzadas de cálculo diferencial, por lo que se le ubica dentro de las técnicas de gradiente libre, que son los de mayor utilidad en los procesos de optimización de funciones. Son capaces, por ejemplo, de llegar a soluciones efectivas a problemas donde la búsqueda sea multiobjetivo, es decir, aquellos donde es necesario cumplir con varias condiciones. Algunos problemas en que se pueden utilizar algoritmos genéticos son, por ejemplo, en el dilema del agente viajero y los generados por la teoría de juegos aplicable a la economía. La clave está en saber desarrollar la función de selección adecuada. También se ha utilizado en el modelamiento de antenas terminales para sistemas de comunicación móviles.

En el caso del problema planteado, se puede extrapolar al campo biológico al afirmar que los mecanismos de la evolución no se dan *al azar* sino que siguen leyes que vienen insertadas en el código genético de cada individuo.

El resultado alcanzado corresponde a lo esperado, pues se logró la meta de organizar un sistema caótico. Esta experiencia puede aportar elementos para entender la selección natural en el campo de la biología y para entender ciertos modelos en áreas como la ingeniería y la economía. Por ejemplo, el Estado recoge cierta cantidad de dinero en impuestos y no sabe cómo gastarlos para que los contribuyentes se sientan, al menos, medianamente satisfechos. Tiene varias opciones. Puede construir hospitales de atención gratuita, puede sentirse amenazado e invertir en defensa o puede levantar un puente innecesario pero que genere empleos. Buenas o malas decisiones pueden hacerse y los algoritmos genéticos pueden ayudar a tomar las mejores medidas.

VII. REFERENCIAS

Damasio A. (2010). *Y el cerebro creó al hombre*. Bogotá DC, Colombia: Planeta

Dawkins, R. (1985). *El gen egoísta. Las bases biológicas de nuestra conducta*. Barcelona, España: Salvat

Ezzatti, P. & Nesmachnow, S. (2003). *Un algoritmo evolutivo simple para el problema de asignación de tareas a procesadores* [Ponencia en IX Congreso Argentino de Ciencias de la Computación (CACIC), La Plata, 2003]. Recuperado de http://www.fing.edu.uy/~sergion/gp/documentos/propios/MOSES.pdf

Haupt, R. & Haupt, S. (1998). *Practical Genetic Algorithms*. New York, NY: John Wiley & Sons

Herrera, F. (1997). Heterogeneous distributed genetic algorithms based on the crossover operator. En *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications,1997, GALESIA 97.* (pp.203-208). Inglaterra: IET Digital Library. http://dx.doi.org/10.1049/cp:19971181

Holland, J. (2004). *El Orden oculto. De cómo la adaptación crea la complejidad*. México DF, México: Fondo de Cultura Económica

Ji Q. & Zhang Y. (2001). Camera calibration with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 31*(2)

Jhumka, A.; Klaus, S.; Huss, S. (2005). Dependability-Driven System-Level Design Approach for Embedded Systems. Recuperado de http://hal.archives-ouvertes.fr/docs/00/18/16/46/PDF/228810372.pdf

Smith, A. (1776 / 1985). *Una investigación sobre la naturaleza y causas de la riqueza de las naciones*. Madrid, España: Orbis

Tzung-Pei, H. (1996). A dynamic mutation genetic algorithm. En *IEEE International Conference on Systems, Man, and Cybernetics*, [Vol.3], (pp.2000-2005). Piscataway, NY: IEEE. doi: 10.1109/ICSMC.1996.565436

Zhang,Y., Sakamoto, M., & Furutani, H. (2008). Effects of Population size and mutation rate on results of Genetic Algorithm. *ICNC '08, Proceedings of the 2008 Fourth International Conference on Natural Computation* [Vol.1], (pp. 70-75). Whasington, DC: IEEE Computer Society

VIII. CURRÍCULO

Julio Cesar Millán Barco. Ingeniero Electrónico de la Universidad del Valle (Colombia, 1999) con Maestría en Automática de la misma Universidad (2005). Docente investigador de la Universidad Santiago de Cali.