

# ***ALGORITMOS GENÉTICOS CON CODIFICACIÓN REAL EN LA OPTIMIZACIÓN DE FUNCIONES DE UNA VARIABLE REAL***

**Autor:** Víctor Hermides Pino de los Reyes

Master en Matemática Aplicada e Informática para la Administración.

Profesor Auxiliar del Departamento de Informática de la Facultad de Ciencias Técnicas del Centro Universitario de Las Tunas.

E – mail: pino@ult.edu.cu

**Coautor:** Alan Oscar Mondol Romero

Estudiante del 4to año de la Carrera de Ingeniería Informática.

E- mail: alanmrinf@estudiantes.ult.edu.cu

Centro Universitario de Las Tunas (CULT)

Las Tunas, mayo de 2009

# INDICE

RESUMEN .....	3
INTRODUCCION .....	4
DESARROLLO.....	4
En qué consisten y cómo funcionan los AG.....	5
Algoritmos genéticos con codificación real .....	7
Operadores de cruce: .....	7
1. Cruzamiento plano .....	8
2. Cruzamiento simple.....	8
3. Cruzamiento aritmético .....	8
4. Cruzamiento $BLX - \alpha$ .....	8
5. Cruzamiento lineal.....	9
6. Cruzamiento discreto .....	9
Operadores de Mutación: .....	10
1. Mutación aleatoria .....	10
2. Mutación no uniforme.....	10
CONCLUSIONES.....	13
TRABAJOS FUTUROS .....	13
BIBLIOGRAFIA .....	13

## RESUMEN

Los Algoritmos Genéticos (AG) se basan en los procesos hereditarios de los organismos biológicos y el principio de la evolución natural de las especies. Procesan poblaciones de cromosomas en función de su aptitud, los cuales representan soluciones candidatas en un espacio de búsqueda de soluciones, usando operadores de selección, cruzamiento y mutación.

La bibliografía referencia mayoritariamente el alfabeto binario para la codificación de los cromosomas y la aplicación de los operadores genéticos.

La codificación real es natural para representar problemas con parámetros en dominios continuos.

Se presenta un software para la optimización de funciones de una variable empleando AG con codificación real, de modo que, en la representación de los cromosomas coinciden genotipo y fenotipo, evadiendo el proceso de transformar las soluciones candidatas de una representación real a una cadena binaria y su proceso inverso, disminuyendo la pérdida de precisión al trabajar con herramientas discretas problemas continuos.

El Software fue desarrollado en la plataforma Visual Studio 2005, en el lenguaje Microsoft Visual C# 2005. Fue diseñado siguiendo el paradigma orientado a objetos.

Se implementan la selección por la ruleta y torneo, mutación aleatoria y no uniforme, cruzamientos aritméticos y BLX- $\alpha$  y una estrategia evolutiva elitista. El criterio de terminación es alcanzar una cantidad de generaciones.

Pueden tratarse tanto problemas de máximo como de mínimo, funciones polinómicas, trigonométricas, exponenciales, y/o la composición de sumas, restas, productos o cocientes entre ellas. Se brindan estadísticas del proceso generacional.

Se observan mejores resultados empleando cruzamiento BLX- $\alpha$ , mutación no uniforme y selección por la ruleta.

## INTRODUCCION

Los Algoritmos Genéticos (AG<sup>1</sup>) surgen como herramientas para la solución de complejos problemas de búsqueda y optimización, producto del análisis de los sistemas adaptativos en la naturaleza, y como resultado de abstraer la esencia de su funcionamiento.

Los métodos de búsqueda y optimización han sido objeto de estudio desde los primeros años de la computación extendiéndose desde los métodos basados en el cálculo, pasando por los métodos enumerativos, hasta llegar a los algoritmos de búsqueda aleatoria. Los métodos de búsqueda y optimización tradicionales, los basados en el cálculo, enumerativos y aleatorios puros son analizados y criticados en términos de robustez, ello no significa que no sean útiles; pudiendo servir de complemento a esquemas más robustos para la creación de híbridos.

El término Algoritmo Genético se usa por el hecho de que estos simulan los procesos de la evolución darwiniana a través del uso de operadores genéticos que operan sobre una población de individuos que “evoluciona” de una generación a otra.

## DESARROLLO

Los Algoritmos Genéticos (AG) son métodos de búsqueda de propósito general basados en los principios de la genética natural, es decir, son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética.

Los Algoritmos Genéticos son un ejemplo de método que explota la búsqueda aleatoria “guiada” que ha ganado popularidad en los últimos años debido a la posibilidad de aplicarlos en una gran gama de campos y a las pocas exigencias que impone al problema.

En el trabajo con AG se maneja una serie de términos “importados” de la genética natural. No siempre es adecuada la analogía, pero estos son comúnmente aceptados:

---

<sup>1</sup> En el texto se utilizará AG tanto para el plural como para el singular.

Población	Conjunto de individuos o cromosomas. Equivale a una muestra aleatoria del espacio de solución o un conjunto de soluciones alternativas.
Cromosoma	Un cromosoma es un portador de la información genética que transmite cada uno de sus genes. Una posible solución.
Gen	Cada uno de los rasgos o características que conforman el cromosoma. También se les llama parámetros o aspectos. Cada gen equivale a una variable del problema.
Genotipo	En biología se le llama al “paquete” genético total en su forma interna. En la terminología de AG será la información genética de todo el cromosoma en forma codificada.
Fenotipo	Se le llama en genética al paquete genético tal y como interactúa con el medio exterior. En los AG artificiales serían los aspectos del cromosoma decodificados.
Locus	Es la posición de un gen el cromosoma
Alelo	Es el valor asociado a un gen

### ***En qué consisten y cómo funcionan los AG***

Los AG trabajan a partir de una población inicial de estructuras artificiales que van modificando repetidamente a través de la aplicación de los siguientes operadores genéticos:

- Operador de Selección o Darwiniano
- Operador de Cruzamiento o Mendeliano
- Operador de Mutación

Para utilizar los AG es necesario encontrar una posible estructura para representar las soluciones. Se busca en un espacio de estados, una instancia de esta

estructura representa un punto o un estado en el espacio de búsqueda de todas las posibles soluciones. Así, una estructura de datos en el AG consistirá en uno o más cromosomas (frecuentemente uno), el cual se representa comúnmente como una cadena de bits (existen otras representaciones).

Cada cromosoma (cadena) es una concatenación de un número de subcomponentes llamados genes. La posición de un gen en el cromosoma se conoce como locus y sus valores como alelos. En la representación como cadena de bits, un gen es un bit o una cadena de bits, un locus es su posición en la cadena y un alelo es su valor (0 ó 1 si es un bit).

Al optimizar una estructura usando un AG se necesita una medida de la calidad de cada estructura en el espacio de búsqueda. La función de adaptabilidad es la encargada de esta tarea. En una maximización de funciones, la función objetivo frecuentemente actúa como la función de adaptabilidad.

Los AG realizan una maximización por defecto, para los problemas de minimización los valores de la función objetivo pueden ser negados y trasladados con vistas a tomar valores positivos para producir así la adaptabilidad.

El mecanismo de un AG simple es como sigue:

- El AG simple genera aleatoriamente una población de  $n$  estructuras (cadenas, cromosomas o individuos)
- Sobre la población actúan los operadores de selección, cruce y mutación transformando la población. Una vez completada la acción de los tres operadores se dice que ha transcurrido un ciclo generacional.
- Luego se repite el paso anterior mientras no se garantice el criterio de parada del AG.

El **operador de selección o Darwiniano** realiza la selección de las cadenas de acuerdo a su adaptabilidad para el posterior apareamiento.

El **operador de cruzamiento o Mendeliano** realiza la recombinación del material genético de dos cadenas padres.

El **operador de Mutación** al estilo del operador natural realiza la mutación de un gen dentro de un cromosoma o cadena a sus diferentes formas alelomorfos.

Para cada uno de estos operadores está asociado el uso de probabilidades y la generación de números aleatorios.

El AG ejecuta para un número fijo de generaciones o hasta que se satisface algún criterio de parada.

### ***Algoritmos genéticos con codificación real***

En la formulación original de AG, las soluciones se han codificado usando el alfabeto binario. Sin embargo, se han aplicado codificaciones no binarias más naturales para los problemas particulares de aplicación. Entre ellas, destaca la codificación real, particularmente adecuada para problemas con parámetros en dominios continuos. Un cromosoma es un vector de números reales con un tamaño igual a la longitud del vector solución del problema. Los AG basados en este tipo de codificación se denominan AG con codificación real (AGCR).

En la actualidad, existe un creciente interés en la resolución de problemas de optimización reales usando AGCR, en campos tan diversos como la es ingeniería industrial, biotecnología, economía, control, diseño aeroespacial, y otros.

El operador de cruce juega un papel central en los AGCR. De hecho puede considerarse como una de las características que definen a estos algoritmos y es uno de los componentes a tener en cuenta para mejorar su eficacia. En el caso de los AGCR, este operador influye decisivamente sobre el nivel de diversidad en la población, y por ello, es un factor determinante para evitar el problema de la convergencia prematura. Esto explica que el principal esfuerzo en la investigación desarrollada para mejorar los AGCR se centre en la propuesta y estudio de nuevos operadores de cruce.

### ***Operadores de cruce:***

Sean los dos cromosomas  $X = (x_1, x_2, \dots, x_n)$  y  $Y = (y_1, y_2, \dots, y_n)$  padres.

### 1. Cruzamiento plano (Radcliffe, 1991)

Se selecciona aleatoriamente una posición  $i \in \{1, 2, 3, \dots, n\}$ , entonces se obtiene un hijo  $H = (h_1, h_2, \dots, h_i, \dots, h_n)$  seleccionando aleatoriamente  $h_i$  del intervalo  $[x_i, y_i]$ , el resto de los componentes del hijo se toman de cualquiera de los dos padres.

### 2. Cruzamiento simple (Wright, 1991; Michalewics, 1992)

Se selecciona aleatoriamente una posición  $i \in \{1, 2, 3, \dots, n-1\}$  y se construyen dos nuevos hijos:

$$H_1 = (x_1, x_2, \dots, x_i, y_{i+1}, y_{i+2}, \dots, y_n)$$

$$H_2 = (y_1, y_2, \dots, y_i, x_{i+1}, x_{i+2}, \dots, x_n)$$

### 3. Cruzamiento aritmético (Michalewics, 1992)

Se selecciona aleatoriamente una constante  $\lambda \in [0, 1]$  y una posición  $i \in \{1, 2, 3, \dots, n\}$  y se obtienen dos hijos de la siguiente forma:

$$H_1 = (x_1, x_2, \dots, x_{i-1}, \lambda x_i + (1 - \lambda) y_i, x_{i+1}, \dots, x_n)$$

$$H_2 = (y_1, y_2, \dots, y_{i-1}, \lambda y_i + (1 - \lambda) x_i, y_{i+1}, \dots, y_n)$$

La constante  $\lambda$  puede no cambiar en durante el todo el proceso, en ese caso, se trata de un cruzamiento aritmético uniforme, en caso contrario, es un cruzamiento aritmético no uniforme.

### 4. Cruzamiento $BLX - \alpha$ (Eshelman y otros, 1993)

Se obtiene un hijo  $H = (h_1, h_2, \dots, h_i, \dots, h_n)$  donde  $h_i$  se selecciona aleatoriamente del intervalo  $[XY_{\min} - I * \alpha, XY_{\max} + I * \alpha]$  siendo

$$XY_{\max} = \max(x_i, y_i) \text{ y } XY_{\min} = \min(x_i, y_i), I = XY_{\max} - XY_{\min},$$

$\alpha \in [-0.25, 1.25]$  y la posición  $i \in \{1, 2, 3, \dots, n\}$  se seleccionan también aleatoriamente. El resto de los componentes del hijo se toman de cualquiera de los dos padres.

Nota: El cruzamiento  $BLX - 0.0$  es igual al cruzamiento plano.

### 5. Cruzamiento lineal (Wright, 1991)

Se toma una posición  $i \in \{1, 2, 3, \dots, n\}$  aleatoriamente y se obtienen tres hijos con valores en el componente seleccionado iguales a:

$$xy_i^1 = \frac{1}{2}x_i + \frac{1}{2}y_i$$

$$xy_i^2 = \frac{3}{2}x_i - \frac{1}{2}y_i$$

$$xy_i^3 = -\frac{1}{2}x_i + \frac{3}{2}y_i$$

El resto de los componentes de los hijos se toman de cualquiera de los dos padres.

En este tipo de cruzamiento, de los tres hijos obtenidos se seleccionan los dos con mejor aptitud.

### 6. Cruzamiento discreto (Muhlenbein y otros, 1993)

Se toma una posición  $i \in \{1, 2, 3, \dots, n\}$  aleatoriamente y se obtienen un hijo que tendrá en la posición seleccionada un número aleatorio seleccionado del conjunto formado por  $\{x_i, y_i\}$ . El resto de los componentes del hijo se toman de cualquiera de los dos padres.

### **Operadores de Mutación:**

Sea  $X = (x_1, x_2, \dots, x_n)$  el cromosoma seleccionado para mutar y  $x_i \in [a_i, b_i]$  el gen que mutará. En lo que sigue  $x'_i$  será el resultado de la aplicación del operador de mutación a ese gen.

#### **1. Mutación aleatoria (Michalewics, 1992)**

$x'_i$  se toma aleatoriamente del intervalo  $[a_i, b_i]$

#### **2. Mutación no uniforme**

Este operador depende del número de la iteración (generación) actual  $t$  en que se aplique y  $G_{\max}$  es el número máximo de iteraciones (generaciones).

$$x'_i = \begin{cases} x_i + \nabla(t, b_i - x_i) & \text{if } \tau = 0 \\ x_i + \nabla(t, x_i - a_i) & \text{if } \tau = 1 \end{cases}$$

Donde  $\tau$  es un número aleatorio escogido del conjunto  $\{0,1\}$  y:

$$\nabla(t, y) = y \left( 1 - r \left( \frac{1 - \frac{t}{G_{\max}}}{1 - \frac{1}{G_{\max}}} \right)^b \right)$$

Donde se toma aleatoriamente  $r \in [0,1]$  y  $b$  es un parámetro seleccionado por el usuario, el cual determina el grado de dependencia respecto al número de iteraciones. Esta función da un valor en el rango  $[0, y]$  tal que la probabilidad de retornar un valor cercano a cero se incrementa a medida que el algoritmo avanza. El tamaño del intervalo de generación del gen debe hacerse pequeño con el paso de las iteraciones (generaciones)

En el software elaborado para la optimización de funciones de una variable real, en esta, su versión 1.0, en la representación de los cromosomas coinciden genotipo y fenotipo, evadiendo el proceso de transformar las soluciones

candidatas de una representación real a una cadena binaria y su proceso inverso, disminuyendo la pérdida de precisión al trabajar con herramientas discretas problemas continuos.

El Software fue desarrollado en la plataforma Visual Studio 2005, en el lenguaje Microsoft Visual C# 2005. Fue diseñado siguiendo el paradigma orientado a objetos.

Se implementan la selección por la ruleta y torneo, mutación aleatoria y no uniforme, cruzamientos aritméticos y BLX- $\alpha$  y una estrategia evolutiva elitista. El criterio de terminación es alcanzar una cantidad de generaciones.

Pueden tratarse tanto problemas de máximo como de mínimo, funciones polinómicas, trigonométricas, exponenciales, y/o la composición de sumas, restas, productos o cocientes entre ellas. Se brindan estadísticas del proceso generacional. Esta es la ventana principal:

**Algoritmos genéticos con codificación real para la optimización de funciones de una variable**

Archivo Calcular Ayuda

Escriba la función:

Extremo izquierdo del intervalo:

Extremo derecho del intervalo:

Probabilidad de cruzamiento:

Probabilidad de mutación:

Tamaño de la población:

Cantidad de generaciones:

Tipo de problema:

Maximizar o minimizar

Máximo

Mínimo

Método de cruzamiento:

Aritmético

BLX-alfa

Método de mutación:

Aleatoria

No Uniforme

Método de selección:

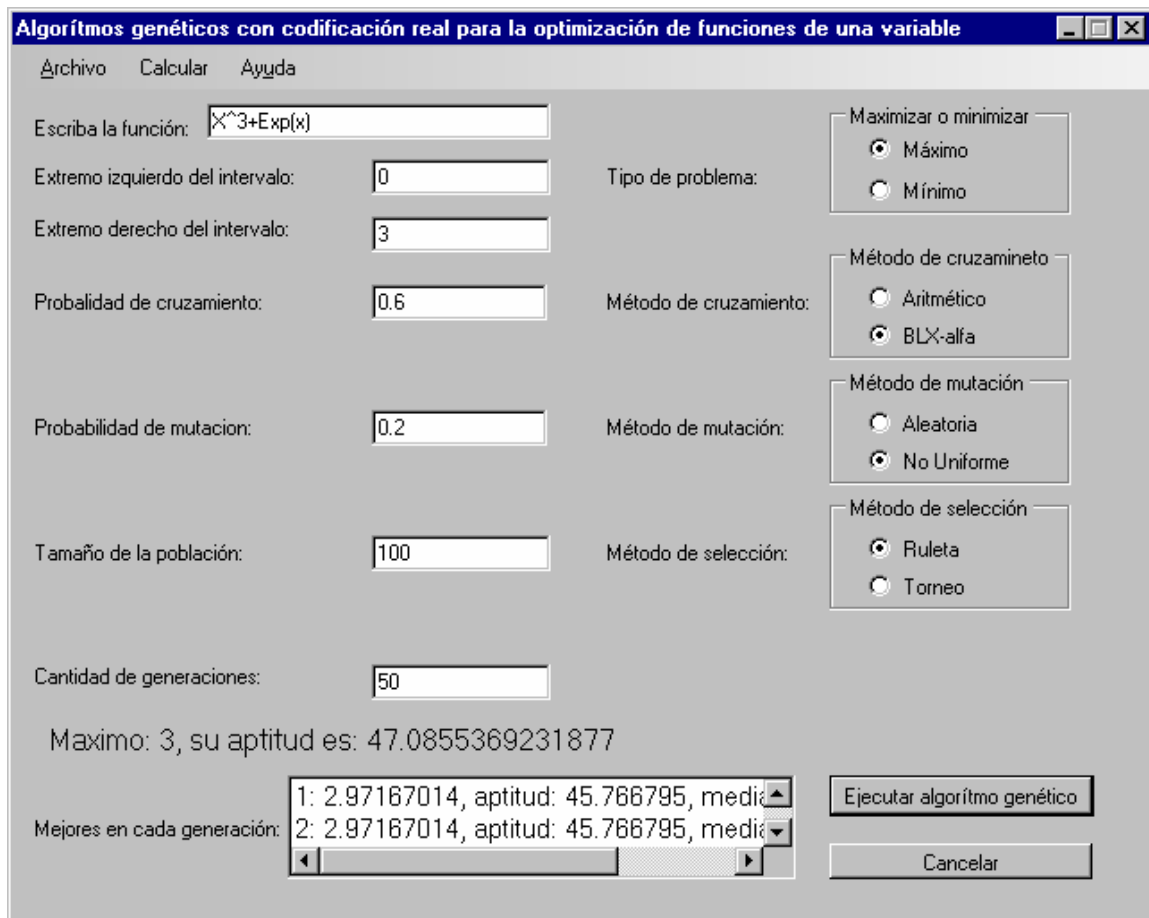
Ruleta

Torneo

Ejecutar algoritmo genético

Cancelar

La siguiente figura muestra una ejecución:



Se observan mejores resultados empleando cruzamiento BLX- $\alpha$ , mutación no uniforme y selección por la ruleta.

Se ha empleado una estrategia evolutiva elitista, sembrando el mejor individuo de cada generación en la siguiente.

## CONCLUSIONES

1. Es natural y no es complejo desde el punto de vista computacional la aplicación de los AGCR en problemas de optimización de funciones de una variable real.
2. La combinación de los operadores de selección por la Ruleta, cruzamiento BLX- $\alpha$  y mutación No Uniforme muestran mejor convergencia que otras combinaciones entre los operadores empleados.

## TRABAJOS FUTUROS

1. Mejorar el analizador sintáctico de expresiones empleado, de modo que se amplíe la gama de funciones que se puedan procesar y su ingreso sea más cercano a la forma matemática usual.
2. Extender el análisis al caso n dimensional.
3. Implementar otros operadores genéticos y validar estadísticamente su eficiencia.

## BIBLIOGRAFIA

1. Francisco Herrera, Manuel Lozano, Ana M. Sánchez. Algoritmos Genéticos con Codificación Real: Operadores de Cruce Híbridos Basados en Entornos con Múltiples Descendientes.
2. F. Herrera, M. Lozano, J.L. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behavioural analysis. *Artificial Intelligence Reviews* 12(4): 265-319.
3. Daniel Gálvez Lio. Algoritmos Genéticos.
4. Manuel Lozano. Algoritmos genéticos con codificación real.
5. MSDN Library para Visual Studio 2005.