UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

Unidad Académica de los Ciclos Profesional
y de Posgrado del C. C. H.

ALGORITMOS GENETICOS AUTOADAPTABLES

T E S I S

Que para obtener el Grado de

MAESTRO EN CIENCIAS DE LA COMPUTACION

p r e s e n t a

JOSE DE JESUS GALAVIZ CASAS

Director de Tesis: DR. ANGEL KURI MORALES

México, D. F. [redacted]

1997

ALGORITMOS GENÉTICOS AUTOADAPTABLES

José Galavíz Casas

AGRADECIMIENTOS

Este trabajo no hubiera sido posible sin la ayuda y/o apoyo de muchos seres a los que mencionaré por orden de aparición:

La primer alga verde-azul del oceano primitivo
Algunos animales arcaicos hoy extintos que contribuyeron al progreso con sus genes
Darwin, Mendel y todos los precursores de las teorias de la evolución y la genética
Holland, que tuvo la previsión de inventar justo aquello que yo necesitaba.

Más cercanamente a mí debo agradecer a mi abuelo por permitirme ser un parásito hasta hace poco. A mi madre por su programación genética (cualquier *bug* es de origen posterior). A Mónica mi esposa, que curiosamente tenía sueño en las noches en que yo tenía que trabajar en esta tesis. A mis amigos: Amparo y Salvador, Ari, Arturo, Beto, Carlos, Chucho, Luis de la Vega y La Mancha, quienes curiosamente tenían sueño siempre que yo hablaba de este trabajo. A mi director de tesis y tutor, Angel Kuri, quien curiosamente nunca tenía sueño. A mis sinodales, quienes podrían haber tenido más horas de sueño en vez de revisar esta cosa.

José Galaviz
C.U. noviembre 1996

Contenido

Capítulo 1

Introducción

Los algoritmos genéticos (AG's) son métodos probabilísticos de búsqueda basados en la simulación parcial de la evolución y selección naturales. Fueron inventados por Holland [11] en la década de los 70's y han sido utilizados exitosamente en la optimización de funciones y en la definición de mecanismos de aprendizaje de máquinas [7, 15]. En este trabajo se utilizarán en la primera de estas vertientes.

Normalmente y en términos generales, los algoritmos genéticos simulan el proceso de la evolución biológica de la siguiente manera [9, 18]:

1. Se codifica el dominio del problema (p.ej. el dominio de la función a optimizar).
2. Se genera una población inicial de tamaño N de posibles soluciones. Esta es una muestra aleatoria de elementos del dominio. Cada uno de ellos está representado por un código binario (cadena genética).
3. Se evalúa el desempeño de cada individuo en la muestra. La función evaluada en cada individuo es llamada su grado de adaptación (*fitness*).
4. Se selecciona un subconjunto de los individuos de la población con probabilidad directamente proporcional a su grado de adaptación.
5. Con probabilidad p_c se combinan por pares los códigos de los individuos seleccionados (cruzamiento).
6. Con probabilidad p_m se cambia el valor guardado en la cadena genética de los individuos seleccionados y cruzados. Dado que normalmente los individuos están codificados en binario p_m es llamada la probabilidad de mutación de bit.
7. Los individuos seleccionados, cruzados y mutados constituyen una nueva generación de posibles soluciones y el proceso es iterado de (3) a (7) hasta que se satisface algún criterio de detención.

Es posible describir este proceso de forma más precisa:

PROCESS *Algoritmo Genético*
begin
  {*Generar una población inicial de individuos* }
  repeat
    {*Calcular el grado de adaptación de cada individuo de la población*}
    for $i := 1$ **to** (*tamaño de la población*) **step** 2 **do**
      {*Seleccionar dos individuos de la población*
      con probabilidad proporcional a su grado de adaptación}
      {*Con probabilidad* p_c *cruzar los dos individuos para formar dos híbridos*}
      {*Con probabilidad* p_m *invertir el valor de los bits de los híbridos*}
      {*Incluir a los dos híbridos en una nueva población*}
    endfor
  until (*criterio de convergencia se satisface*)
end

Durante este proceso intervienen ciertas variables tales como la probabilidad de cruza p_c, la probabilidad de mutación p_m, el tamaño de la población y el mecanismo utilizado para combinar dos cadenas genéticas (cromosomas) para producir cadenas híbridas. Normalmente estas variables (parámetros de control) son dadas *a priori* por el usuario o programador del algoritmo y gobiernan el comportamiento de éste durante toda su ejecución. Es decir son parámetros globales y constantes.

Los parámetros de control del algoritmo genético gobiernan su comportamiento y por tanto influyen en su desempeño [15]. Ya ha sido demostrado que los dos operadores genéticos (el cruzamiento y la mutación) son esenciales para que los AG's funcionen [17]. Cada operador cumple con una parte importante del proceso de búsqueda. El cruzamiento y la selección permiten *explotar* el conocimiento ya obtenido, al cruzar individuos bien adaptados se pretende aprovechar las características convenientes de ellos para obtener descendientes mejor adaptados aún. La mutación permite *explorar* regiones de dominio del problema que quizás aun no se hayan muestreado. Un algoritmo genético robusto y con buen desempeño debe lograr un equilibrio entre la explotación y la exploración. Si la probabilidad de cruza es muy alta y la de mutación muy baja puede ser que solo se encuentren soluciones sub-óptimas (máximos o mínimos locales en el contexto de optimización de funciones) y si la probabilidad de cruza es baja y la de mutación muy alta el algoritmo se la pasará saltando bruscamente en una búsqueda prácticamente aleatoria.

Se han hecho trabajos tendientes a buscar las mejores combinaciones de valores para los parámetros de un algoritmo genético. Algunos autores proponen la utilización de un meta-algoritmo genético que encuentre la combinación óptima de los valores de los parámetros para un AG particular [10], otros proponen la utilización de lógica difusa [14] y otros más la inclusión de los parámetros de control en el código genético de los individuos de la población [1, 16]. Aparentemente esta última es la aproximación más razonable a la

solución del problema, dado que se ha demostrado [2, 3] que los valores óptimos (al menos en el caso de la probabilidad de mutación) no permanecen constantes durante toda la ejecución del algoritmo, sino que varían generación con generación proporcionalmente al grado de adaptación promedio de la población.

Siguiendo con la última línea de investigación mencionada, en esta tesis se pretende demostrar la hipótesis de que un algoritmo genético con sus parámetros de control codificados en los individuos de la población, exhibe un comportamiento cercano al comportamiento promedio exhibido por el algoritmo genético tradicional, ejecutado con una variedad amplia de valores para sus parámetros de control estáticos y globales. Para probar esta hipótesis se seleccionó un conjunto heterogéneo de funciones sobre las que se midió el comportamiento de ambos algoritmos.

Más adelante se aclarará qué se pretende decir con "comportamiento" y con "cercano al promedio". Por ahora baste decir que para medir objetivamente el comportamiento se tomarán algunas medidas del desempeño de los algoritmos.

En el capítulo 2 se describirá detalladamente como se codificó el dominio de la funciones de prueba, cuáles parámetros de control del algoritmo genético se codificaron y cómo se hizo esto. En el capítulo 3 se explica algo del diseño del programa que implanta ambos algoritmos genéticos: el tradicional, al que llamaremos AGT y el autoadaptable (con parámetros de control codificados en los individuos) al que llamaremos AGAA. En el capítulo 4 se presentan los resultados obtenidos y su análisis estadístico. Finalmente, en el capítulo 5 se presentan las conclusiones.

Capítulo 2

Genotipo de los Individuos

Como se mencionó en el capítulo anterior, existen ciertos parámetros que controlan el comportamiento de un algoritmo genético que normalmente son dados por el usuario del mismo. Estos parámetros se mantienen estáticos y actúan globalmente sobre todos los individuos de la población a lo largo de todas las generaciones que corra el algoritmo. En la naturaleza las cosas no funcionan así. Ninguna inteligencia superior se encarga de definir cuál será la razón de cruza de los individuos de una población de animales ni cuántas mutaciones ocurrirán en cada individuo de cada generación. Un ecosistema funciona por sí solo: se autoadapta. Los individuos que habitan en él están perfectamente adaptados y esta adaptación es el resultado de muchas generaciones de ensayos por obtener organismos cada vez más capaces de sobrevivir. Los códigos genéticos de estos individuos se han ido modificando a lo largo de eones hasta lograr configuraciones óptimas y los parámetros que gobiernan estas modificaciones están dados por el ecosistema mismo.

El objetivo es imitar a la naturaleza en la medida de lo posible. Para esto es necesario minimizar la cantidad de parámetros que determinan el comportamiento del AG dados *a priori*. Los más evidentes son la probabilidad de cruza p_c, la probabilidad de mutación p_m y el número de descendientes que puede tener una pareja de individuos, que en el algoritmo genético tradicional siempre son dos. Además de estos parámetros existe otro que, aún sin ser dado de antemano por el usuario, no es determinado de una manera "inteligente" sino solamente aleatoria: el punto de cruza.

Existen varios mecanismos en los AG's para mezclar los códigos genéticos de individuos que se aparean para dar lugar a nuevos individuos. Para tener un panorama de estos mecanismos véanse las referencias [4, 8, 18]. Tradicionalmente el más usado de estos mecanismos es el cruzamiento de 1 punto (*1-point crossover*). En este, dadas dos cadenas genéticas de longitud ℓ se escoge aleatoriamente un punto de corte de las mismas $x_c \in \{1, .. \ell - 1\}$ y se procede a combinar los segmentos resultantes. Como se ha dicho, el punto de corte es determinado aleatoriamente, pero en la población puede haber individuos que ya tengan en su código parte de la cadena genética óptima y que esta se pierda

al momento de cruzarlos con otros. Entonces ¿por qué no mejor dejar que los individuos a aparear determinen su propio punto de cruza y así, con el tiempo, se llegue a un punto de cruza "óptimo"?

Los parámetros que se decidió codificar en el genotipo de los individuos de la población del AGAA son: probabilidad de cruza p_c, probabilidad de mutación p_m, número de descendientes y punto de cruza. Surge entonces un nuevo problema a resolver: dados dos individuos, cada uno con sus propios valores de parámetros, ¿cómo determinar los valores efectivos de los parámetros que serán usados para cruzarlos y mutarlos y cuáles heredarán sus hijos?

En cuanto a la herencia el problema está resuelto si consideramos los códigos de los valores de los parámetros como cadenas genéticas también y los sometemos a los mismos operadores genéticos (cruza y mutación) que el resto del cromosoma. De esta manera los valores de los parámetros para los hijos serán generados automáticamente a través de la cruza y mutación de los valores de los padres. Con la salvedad de no mezclar códigos de cosas diferentes (p. ej. no cruzar el código de probabilidad de mutación con el de probabilidad de cruza).

Por otra parte, para codificar y determinar los valores efectivos de los parámetros que serán usados para cruzar y mutar a los individuos se hizo lo que se describe a continuación. Se asumirá que todos los enteros que aparezcan están en el conjunto $\{0, 1, .., K\}$ donde K es de la forma $2^n - 1$, suponiendo que el algoritmo se implanta en una computadora cuyos enteros miden n bits.

2.1 Codificación del dominio de las funciones de prueba

Como se mencionó en la introducción, para comparar los comportamientos del AGT y del AGAA se seleccionó un conjunto de funciones de prueba. A todas ellas se les restringió su dominio al conjunto $[0, 1] \times [0, 1]$, por lo tanto los elementos del dominio de las funciones están constituidos por parejas ordenadas de números reales (a, b) donde $a, b \in [0, 1]$. Cada coordenada se codificó en los individuos de la población como un entero de n bits de longitud, para obtener el código a partir del valor real de cada coordenada éste se multiplica por $2^n - 1$ y se trunca al entero menor más cercano, luego se concatenan los dos enteros correspondientes a las coordenadas del punto y el resultado constituye el código genético del elemento del dominio deseado.

En el AGT el genotipo de los individuos de la población es únicamente el código del punto del dominio que representan, mientras en el AGAA además de este código se tienen los correspondientes a los parámetros de control.

2.2 Probabilidad de cruza

Sea p_c el valor de la probabilidad de cruza de un individuo dado. El código para p_c contenido en el individuo será el correspondiente al entero c_c donde:

$$c_c = \lfloor K \cdot p_c \rfloor$$

Ahora bien, dos individuos con probabilidades de cruza p_{c1} y p_{c2}, son cruzados con probabilidad:

$$p_{c12} = \frac{p_{c1} + p_{c2}}{2}$$

sencillamente la media aritmética de las probabilidades de ambos.

2.3 Probabilidad de mutación

Sea p_m el valor de la probabilidad de mutación de un individuo dado. El código para p_m contenido en el individuo será el correspondiente al entero c_m donde:

$$c_m = \lfloor K\, p_m \rfloor$$

Para aplicar el operador de mutación a un individuo dado, la probabilidad de mutación estará dada por el procedimiento propuesto por Bäck [1, 2].

1. Se aplica el operador de mutación al código correspondiente a la probabilidad de mutación, usando la probabilidad codificada en él.
2. Se considera este nuevo valor como la probabilidad de mutación que será usada para aplicar el operador al resto del cromosoma y a los códigos de los demás parámetros.

2.4 Punto de cruza

El código del punto de cruza se especifica como el porcentaje de cadena genética que se intercambiará entre los individuos. Se hizo de esta forma porque no es posible poner el número correspondiente al punto de corte directamente, dado que se aplicará el cruzamiento a varios códigos (los de los valores de los parámetros y el del genotipo del individuo) y estos tienen longitudes diferentes. Sean ℓ la longitud de algún código y $x_1, x_2 \in \{0, 1, ..., 100\}$ los puntos de cruza de dos individuos dados. El punto de cruza efectivo usado para combinar los códigos de ambos individuos será:

$$x_{ef} = \left(\frac{x_1 + x_2}{200}\right) \ell$$

2.5 Número de descendientes

Cada individuo en el AGAA tiene codificado entre sus parámetros el número de descendientes que puede tener. Sean c_1 y c_2 los números de descendientes de dos individuos dados. Cuando estos individuos se cruzan generan

$$d = \left(\frac{c_1 + c_2}{2}\right)\left(\frac{f_{1,2}}{f_{max}}\right)$$

donde $\overline{f}_{1,2}$ es el grado de adaptación medio para los dos individuos y f_{max} es el máximo grado de adaptación obtenido en la generación actual.

El número de descendientes determinado de esta manera asegura que las estructuras con grado de adaptación bajo tengan menos oprtunidad de reproducción que las que son mejores. Bajo estas condiciones es posible, en principio, que el tamaño de la población crezca indiscriminadamente conforme los individuos vayan mejorando al paso del tiempo. Para evitar este fenómeno se fijó el tamaño de la población durante todo el proceso del algoritmo. Este es el único parámetro fijado *a priori* en el AGAA.

Capítulo 3

Diseño e Implantación

3.1 Identificación de clases y relaciones

Los programas de cómputo que implantan los algoritmos genéticos tradicional y auto-adaptable se diseñaron usando objetos mediante el método descrito por Booch [5]. Era necesario hacer un programa que permitiera manipular *poblaciones* de *individuos* que evolucionaran en el tiempo. De aquí que las clases necesarias sean *población* e *individuo*. Sin embargo los individuos y las poblaciones manipulados en el AGT no son iguales a los del AGAA.

Haciendo un análisis más a fondo es claro que los individuos manejados en el AGAA son una especialización de los manejados en el AGT. Es decir, aquellos tienen lo mismo que éstos y un poco más. Ese "poco más" son justamente los códigos para los valores de los parámetros. Así que se puede pensar en los individuos del AGAA como una subespecie de los del AGT. Para los individuos se definen entonces dos clases: `Individuo` de la cual son instancias los individuos del AGT e `IndividuoP` (individuo con parámetros) de la que son instancias los individuos del AGAA. Esta última es una clase derivada de la primera y por tanto sus objetos heredan todas las características de `Individuo` y poseen algunas más que de hecho son los parámetros.

Los objetos de `Individuo` poseen su información genética (`genotipo`), el elemento del dominio del problema que está siendo codificado en ésta (`fenotipo`) y su grado de adaptación (`fitness`). Los objetos de la clase `IndividuoP` poseen lo mismo que los de `Individuo` y además el código para la probabilidad de cruza (`pcross`), el código para la probabilidad de mutación (`pmut`), el código para el punto de cruza (`ptox`) y el código para el número de descendientes (`ndes`).

Por otra parte es necesario definir una clase `Poblacion` que permita agrupar individuos y manipularlos así como obtener los parámetros poblacionales (adaptación promedio, adaptación máxima, etc.) y la correspondiente clase `PoblacionAO` (población auto-organizada) que agrupe individuos de tipo `individuop`.

En la fig 1 se muestra un diagrama que, en notación de Booch, expresa las

relaciones existentes entre las clases mencionadas.

La implantación se hizo en lenguaje C++ en computadoras PC compatibles con IBM con sistema operativo MS-DOS. Las poblaciones contienen un arreglo de tamaño fijo de individuos y éstos un arreglo dinámico de bytes para su genotipo. Cada población mantiene, entre sus características, información acerca del desempeño promedio de sus individuos, el desempeño máximo en la generación actual y otros datos que permiten calcular promedios y desviaciones estándar de genotipos, fenotipos y *fitness*. Esta información es actualizada cada vez que entra o sale de la población un individuo nuevo y permite conocer datos estadísticos rápidamente sin tener que calcularlos en el momento.

Con estas clases se elaboraron dos programas, uno que implanta el AGT y otro que implanta el AGAA.

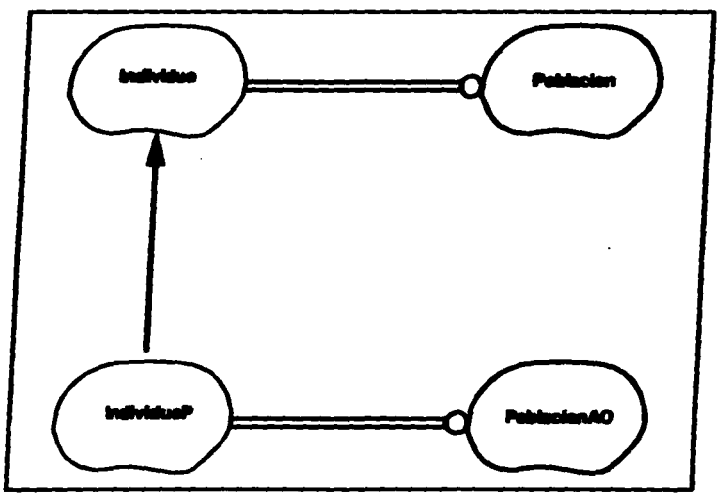

Fig 1: Relaciones entre las clases.

3.2 Esquemas de clases

Siguiendo con el método de diseño propuesto por Booch se presentan a continuación los esquemas de las cuatro clases mencionadas.

Nombre: `Individuo`

Documentación: individuos simples

Visibilidad: exportado

Cardinalidad: n

Superclases: ninguna

Usa: servicios implantados en utilerías

Campos: `genotipo, fenotipo, fitness`

Operaciones: funciones para establecer y extraer la información de cada campo, funciones para realizar mutaciones y para crear y destruir dinámicamente instancias, operador de asignación, función amiga para cruzamiento, etc.

Complejidad de espacio: tamaño(genotipo)+2*tamaño(`real`)+tamaño(`real`)

Nombre: `IndividuoP`

Documentación: individuo con parámetros

Visibilidad: exportado

Cardinalidad: n

Superclases: `Individuo`

Usa: servicios implantados en utilerías

Campos: `pcross, pmut, ptor, ndes`

Operaciones: funciones para establecer y extraer la información de cada campo, funciones para realizar mutaciones y para crear y destruir dinámicamente instancias, operador de asignación, función amiga para cruzamiento, etc. (Hereda las operaciones de `individuo`)

Complejidad de espacio: tamaño(`Individuo`)+ 4*tamaño(`entero`)

Nombre: `Poblacion`

Documentación: conjunto de elementos del tipo `Individuo`

Visibilidad: exportado

Cardinalidad: 1

Superclases: ninguna

Usa: `Individuo`

Campos: arreglo de individuos, `suma_fitness`, `suma_fenotipos`, `suma_fitness_cuadrados`, `suma_fenotipos_cuadrados`, `indice_fit_max`

Operaciones: para extraer información, meter y sacar individuos, para seleccionar parejas de individuos, para extraer información estadística de la población

Complejidad de espacio: variable

Nombre: `PoblacionAO`

Documentación: población auto-organizada, conjunto de elementos del tipo `IndividuoP`

Visibilidad: exportado

Cardinalidad: 1

Superclases: ninguna

Usa: `IndividuoP`

Campos: arreglo de individuos (con parámetros), `suma_fitness`, `suma_fenotipos`, `suma_fitness_cuadrados`, `suma_fenotipos_cuadrados`, `indice_fit_max`, suma de cada parámetro y suma de cuadrados de cada parámetro

Operaciones: para extraer información, meter y sacar individuos, para seleccionar parejas de individuos, para extraer información estadística de la población

Complejidad de espacio: variable

3.3 Utilerías

Además de las clases descritas fue necesario implantar diversos servicios en un módulo de utilerías. Aquí se programó la generación de números aleatorios con distribución uniforme en el intervalo [0, 1]. Previamente se efectuaron dos pruebas estadísticas para evaluar la conveniencia de tres diferentes métodos de generación de números aleatorios. Estas pruebas fueron χ^2 y Kolmogorov-Smirnoff. Dos métodos obtuvieron resultados satisfactorios en ambas pruebas (método randu [12, 13] y generador del compilador). En los experimentos realizados se utilizó el generador interconstruido en el compilador, pero en las utilerías aparecen ambos métodos disponibles para su uso.

Con base en el generador de números aleatorios se programaron otras rutinas para generar variables aleatorias de Bernoulli y uniformemente distribuídas en un rango específico. También entre las rutinas de utilería se hicieron funciones para codificar enteros y reales en cadenas de caracteres y para cruzarlos.

Capítulo 4

Experimentos y Resultados

4.1 El conjunto de funciones

Con el fin de probar el AGAA en contraste con el AGT se definió, al estilo de De Jong [6], un conjunto de veinte funciones de $\mathbb{R}^2$ en $\mathbb{R}$ para ser optimizadas (maximizadas) por ambos mecanismos. Las reglas de correspondencia de estas funciones aparecen en la tabla 1. A todas ellas se les restringió el dominio al conjunto $[0,1] \times [0,1]$. Las gráficas de las funciones en este conjunto son mostradas en las figuras 2-6.

Al definir este conjunto de funciones se procuró que fuera lo más representativo posible, en el sentido de exhibir una amplia variedad de comportamientos. En el conjunto hay funciones suaves y no suaves, continuas y discontinuas, convexas y no convexas, unimodales y multimodales. Algunas de las funciones usadas son versiones tridimensionales de las usadas por De Jong.

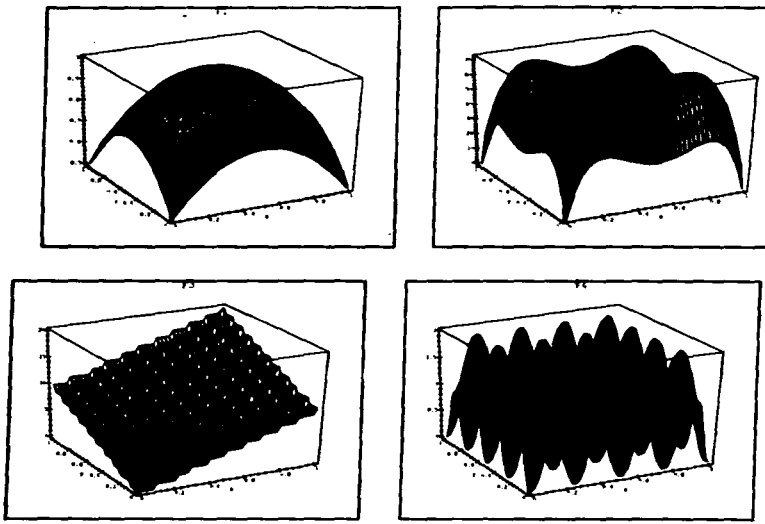

Fig 2: Gráfica de las funciones 1-4 listadas en la tabla 1.

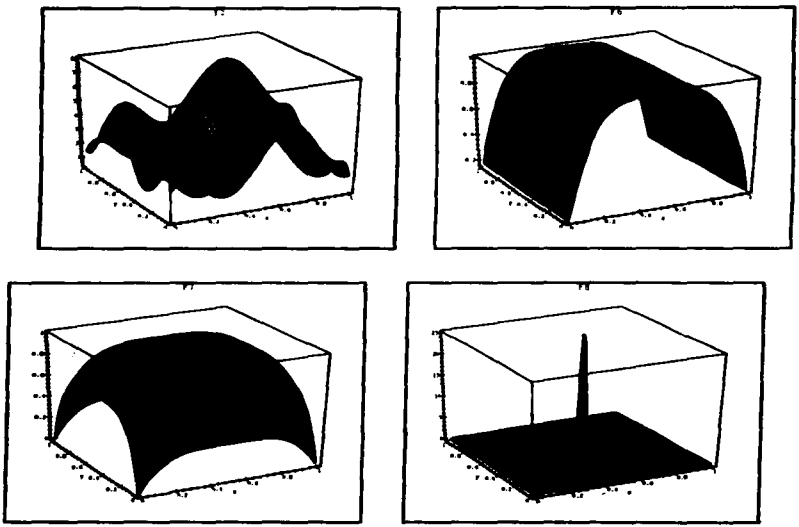

Fig 3: Gráfica de las funciones 5-8 listadas en la tabla 1.

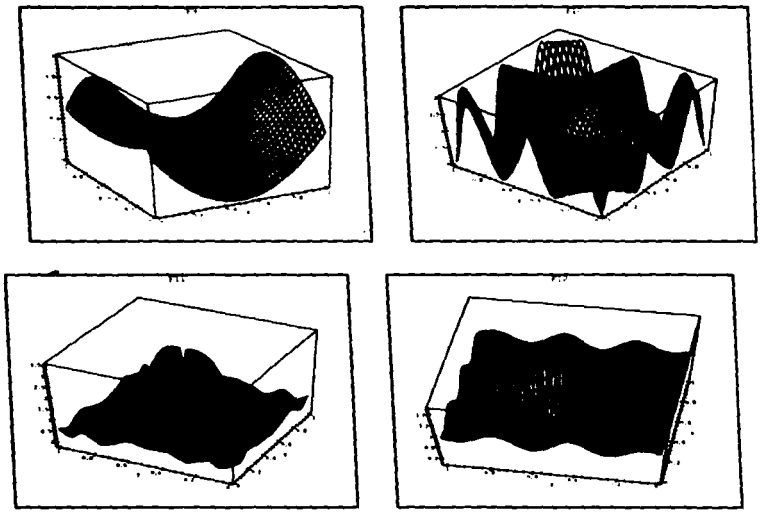

Fig 4: Gráfica de las funciones 9-12 listadas en la tabla 1.

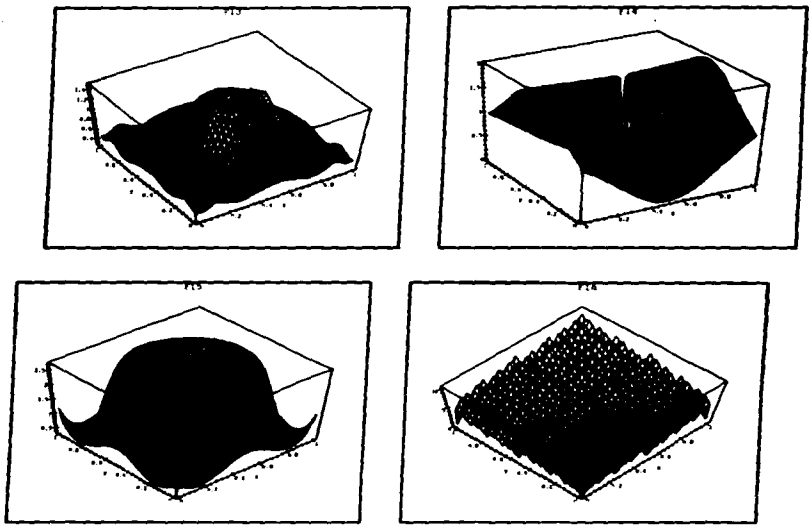

Fig 5: Gráfica de las funciones 13-16 listadas en la tabla 1.

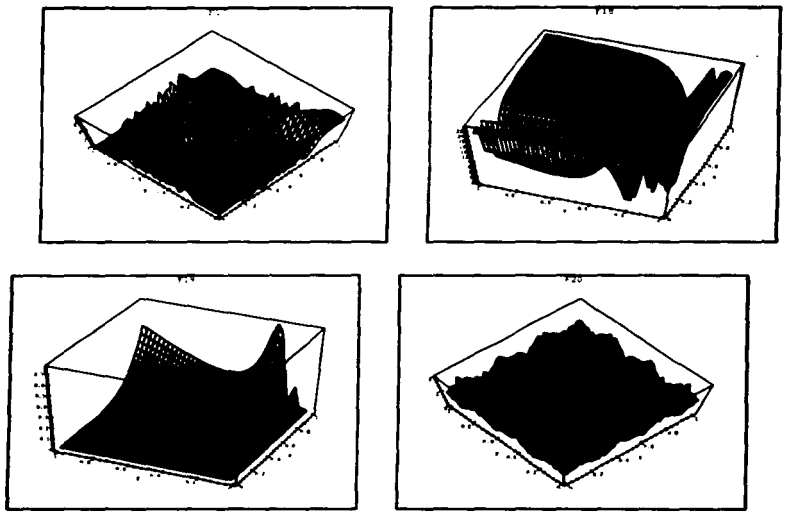

Fig 6: Gráfica de las funciones 17-20 listadas en la tabla 1.

4.2 El conjunto de parámetros

El algoritmo genético tradicional fue ejecutado 36 veces para cada una de las veinte funciones mencionadas durante 100 generaciones y en cada una de estas ejecuciones se usaron valores distintos de probabilidad de mutación (p_m) y probabilidad de cruza (p_c). p_c en el intervalo [0.5, 1] en incrementos de 0.1 y p_m.

en el intervalo $[0.005, 0.01]$ tomando incrementos de 0.001. Estos intervalos son considerados como óptimos [10].

El algoritmo genético autoadaptable fue ejecutado para cada función una única vez también durante 100 generaciones. El tamaño de la población para ambos algoritmos fue fijado en 80 individuos por generación.

Para cada ejecución del AGT(p_c, p_m) durante 100 generaciones con una instancia de sus parámetros fueron tomadas diversas medidas de desempeño que serán mencionadas más adelante. Estas mismas medidas se tomaron para la única ejecución el AGAA. Ambos algoritmos utilizaron una estrategia elitista: siempre en la generación n se conserva el individuo mejor adaptado de la generación $n-1$.

Número	Función
1	$1-(x-1/2)^2-(y-1/2)^2$
2	$72-\left((6x-3)^2-3\right)^2-\left((6y-3)^2-3\right)^2$
3	$\lfloor 10x \rfloor + \lfloor 10y \rfloor$
4	$(\sin(4\pi x))^2 + (\sin(4\pi y))^2$
5	$\left(1-(5.5x-3.5)^2\right)(\cos(5.5x-3.5)+1)+4$ $+\left(1-(5.5y-3.5)^2\right)(\cos(5.5y-3.5)+1)$
6	$-\frac{((20x-10)^2-y)^2}{12000} - \frac{(11-20x)^2}{1200000} + 1$
7	$-\frac{(20x-10)^4}{30000} - \frac{(20y-10)^4}{15000} + 1$
8	$\frac{1}{((20x-12)^2+(20y-13)^2+0.0001)}$
9	$\frac{(200x-100)^2}{20000} - \frac{(200y-100)^2}{200000} + 1/2$
10	$\cos((6x-3)(6y-3))+1$
11	$\frac{\sin(2(4x-2)^2+3(4y-2)^2)}{(4x-2)^2+(4y-2)^2+0.0001} + 0.5$
12	$\frac{\sin((6x-3)(6y-3))}{(6x-3)^2+(6y-3)^2+0.0001} + 1$
13	$\frac{\sin((6x-3)^2+(6y-3)^2)}{(6x-3)^2+(6y-3)^2+0.0001} + 0.5$
14	$\frac{(2x-1)^2-(2y-1)^2}{(2x-1)^2+(2y-1)^2+0.0001} + 1$
15	$e^{\sin((3.6x-1.8)^2+(3.6y-1.8)^2)}$
16	$32x-\lfloor 16x \rfloor - \lfloor 8x \rfloor - \lfloor 4x \rfloor - \lfloor 2x \rfloor - \lfloor x \rfloor$ $+32y-\lfloor 16y \rfloor - \lfloor 8y \rfloor - \lfloor 4y \rfloor - \lfloor 2y \rfloor - \lfloor y \rfloor$
17	$\frac{e^{\sin\left(\frac{2x-1}{(2y-1)^2+0.0001}\right)}}{e^{\sin\left(\frac{2y-1}{(2x-1)^2+0.0001}\right)}}$
18	$\frac{3x^2y+2x^2y^2+x^2y-9}{e^{\sin(\frac{x}{2})\sin(\frac{y}{2})}} + 25$
19	$\frac{3x^2y}{e^{\sin(\frac{x}{2})+\sin(\frac{y}{2})}}$
20	$-\sin(20x-10)\left(\sin\left(\frac{(20x-10)^2}{\pi}\right)\right)^2$ $-\sin(20y-10)\left(\sin\left(\frac{(20y-10)^2}{\pi}\right)\right)^2 + 2$

Tabla 1: El conjunto de funciones de prueba.

4.3 Resultados

Después de ejecutar durante 100 generaciones el AGT con unos valores dados de p_c y p_m se obtuvo un grado de adaptación (*fitness*) promedio de los individuos de la generación 100 (FitP) y su desviación estándar (Des), el grado de adaptación máximo obtenido durante la ejecución del algoritmo (FitM), el desempeño *offline* (OffIP) [9] y el número de la generación en la que se obtuvo por primera vez el mejor grado de adaptación (GenM).

Los datos mencionados en el párrafo anterior fueron obtenidos para cada una de las 36 ejecuciones que se efectuaron para cada función, así que al final se tenían 36 medidas de cada uno de ellos para cada función. Luego estos valores fueron promediados. Los valores medios de las medidas para cada función aparecen en la tabla 2.

FitP	*Des*	*FitM*	*OffIP*	*GenM*
0.96666586	0.05770856	1.0	0.99998561	85.5
69.3172662	4.1223945	71.9999526	71.9971044	78.1388889
17.3072917	1.41609994	18.3611111	18.2124018	15.1388889
1.90392431	0.21367044	1.99999994	1.99881633	76.4166667
7.59497275	0.76841847	7.99999625	7.98998072	79.9444444
0.98548044	0.05605603	1.0	1.0	89.3333333
0.97173894	0.08094989	1.0	1.0	76.6944444
3.75237753	1.63087144	4.70176172	3.87103275	42.8333333
0.93861858	0.11357353	0.99999997	0.99920886	72.4166667
1.87746228	0.27929353	2.0	1.99998536	48.25
3.24395714	0.56711233	3.48605178	3.48453097	72.3611111
1.38899444	0.16193189	1.49877589	1.49842897	84.5555556
1.42010583	0.21303939	1.497763	1.49774847	48.7222222
1.92785025	0.21820211	1.9999	1.99989164	82.3055556
2.57419717	0.33136022	2.718282	2.71827239	49.5
11.4174322	0.97864533	11.9989688	11.9825123	38.9444444
6.78651606	1.24904272	7.38427347	7.36451553	78.8333333
22.7508129	1.11108189	23.5217079	23.4775751	82.7777778
0.69552894	0.1380715	0.76318425	0.75498358	70.5555556
3.58804192	0.52293617	3.96646122	3.95220894	65.7222222

Tabla 2: Valores medios obtenidos para el AGT.

Las mismas datos fueron medidos para una única ejecución del AGAA para cada una de las funciones. Los valores obtenidos se muestran en la tabla 3.

FitP	Des	FitM	OffIP	GenM
0.984554	0.043212	1.0	0.999979	62
64.072835	6.764878	71.973576	71.947449	72
14.975	2.863455	18.0	17.98	2
1.437668	0.397415	2.0	1.999998	64
6.403574	1.663187	8.0	7.991276	78
0.950197	0.13541	1.0	1.0	92
0.961209	0.057005	1.0	0.999998	73
0.000126	0.000653	0.004202	0.004165	2
0.817622	0.2003	0.999802	0.999051	37
1.854739	0.243739	2.0	1.999991	79
1.701703	0.833925	3.420772	3.420772	1
1.023707	0.218746	1.499101	1.498376	27
1.186233	0.439572	1.497763	1.497729	37
1.799526	0.344303	1.9999	1.999897	89
2.115787	0.722542	2.718282	2.718277	41
8.721888	1.950765	11.999023	11.964952	47
7.242745	0.914653	9.509728	9.509728	1
21.318638	2.579965	23.520363	23.452592	86
0.663067	0.095108	0.6878	0.678179	96
2.568771	0.775243	3.952474	3.946541	92

Tabla 3: Valores obtenidos para el AGAA.

4.4 Análisis estadístico

El objetivo de este trabajo es probar que el comportamiento, medido en términos de los datos mencionados en la sección anterior, del AGAA es un promedio del comportamiento del AGT. Esto es, debemos mostrar que los resultados obtenidos con la única ejecución del AGAA "concuerdan" de alguna manera con los resultados medios del AGT. Es posible formular esto en términos estadísticos. Se requiere probar la hipótesis H_0: *los valores medios obtenidos para el AGT y los valores obtenidos para el AGAA pertenecen a la misma población.*

Para probar esta hipótesis se utilizó la prueba de bondad de ajuste de χ^2 con 19 grados de libertad, dado que se tienen 20 datos (uno para cada función). Para cada medida m (FitM, FitP, OffIP) se calculó:

$$R_m = \sum_{i=1}^{20} \frac{(m_{a,i} - m_{t,i})^2}{m_{t,i}}$$

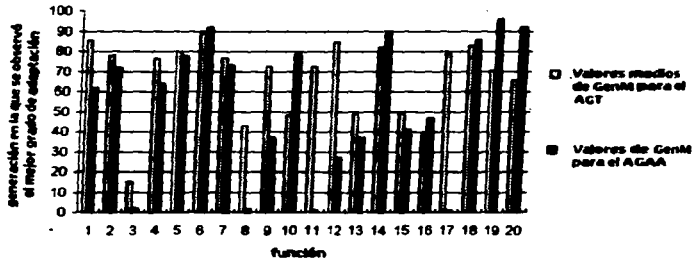

Fig 7: El número de generaciones hasta que el mejor grado de adaptación es alcanzado.

donde $m_{a,i}$ es el valor medio de la medida obtenido para el AGT en la función i y $m_{a,i}$ es el valor de la misma medida obtenido para el AGAA en la función i.

Los valores de R_m obtenidos son los siguientes:

Adaptación promedio (FitP): $R_{\bar{f}} = 6.788$

Mejor adaptación (FitM): $R_{f_{max}} = 5.32$

Desempeño *Off-line* (OffP): $R_{off} = 4.499$

Todos estos valores son menores que $\chi^2_{19,0.005} = 6.884$ por lo que se acepta la hipótesis con un nivel de significación del 0.995.

Además de este resultado es también importante mencionar que en la mayoría de los casos (60%) el AGAA alcanza su mejor grado de adaptación en menos generaciones (41%) que la media de los AGT's probados (ver figura 7).

Capítulo 5

Conclusiones

Los resultados obtenidos en el capítulo anterior permiten concluir que el AGAA es capaz de autoadaptarse. Exhibe un comportamiento cercano al promedio de su contraparte ejecutado con el mejor conjunto posible de valores. Esto fue observado y demostrado estadísticamente al optimizar un conjunto particularmente heterogéneo de funciones.

A pesar de que el concepto de autoadaptación en algoritmos genéticos ya ha sido estudiado antes, los resultados mostrados son un tanto más generales dado que casi todos los parámetros de control del algoritmo fueron codificados, incluidos en la cadena genética de los individuos y sometidos a los operadores genéticos.

El presente trabajo abre nuevas interrogantes y líneas de investigación. Algunas de éstas son:

1. Sería interesante probar el AGAA con otro tipo de problemas que no sean estáticos como en el caso de la optimización de funciones, sino que varíen con el tiempo. Esto con el fin de observar si el AGAA es capaz de autoadaptarse a condiciones "ambientales" variables y si lo es con qué rapidez se adapta.

2. Averiguar cómo se comportan los parámetros codificados generación con generación. Por ejemplo, observar si la probabilidad de mutación promedio de la población crece o decrece conforme la población es cada vez más apta.

3. Investigar en qué tipo de problemas vale la pena hacer una búsqueda exhaustiva en el espacio de parámetros del AGT con el fin de obtener un desempeño óptimo en vez de obtener un desempeño promedio ejecutando el AGAA.

4. Establecer comparaciones bidireccionales entre el AGAA y el comportamiento de poblaciones de organismos vivos.

5. Investigar si es posible y en qué condiciones el AGAA exhibe un comportamiento caótico.

6. **Finalmente, sería interesante utilizar el AGAA en el contexto de aprendizaje de máquinas.**

Referencias

[1] Bäck, Thomas. *Self-Adaptation in Genetic Algorithms.* Proceedings of the First European Conference on Artificial Life. Paris. France. 1991. pp. 263-271.

[2] Bäck, Thomas. *The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm.* Paralell Problem Solving from Nature 2. Elsevier, Amsterdam. 1992. pp. 85-94.

[3] Bäck, Thomas. *Optimal Mutation Rates in Genetic Search.* ENCORE (192.12.12.99).

[4] Beasley, D., Bull D. y Martin R. *An Overview of Genetic Algorithms: Part 1, Fundamentals.* University Computing. 1993.

[5] Booch, Grady. *Object-Oriented Design with Applications.* Benjamin/Cummings Pub. Co. 1991.

[6] DeJong, K. A., *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* Doctoral Dissertation. University of Michigan. 1975.

[7] DeJong, K. A., *Learning with Genetic Algorithms: An Overview.* Genetic Algorithms. IEEE Computer Society Press. 1992. pp. 30-47.

[8] Eshelman, L., Schaffer, D. y Caruana, R., *Biases in the Crossover Landscape.* Proceedings of the Third International Conference on Genetic Algorithms. George Mason University. 1989. pp. 10-19.

[9] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co. 1989.

[10] Grefenstette, John J. *Optimization of Control Parameters for Genetic Algorithms.* Genetic Algorithms. IEEE Computer Society Press. 1992.

[11] Holland, J.H. *Adaptation in Natural and Artificial Systems.* Ann Arbor, The University of Michigan Press. 1975.

[12] Jain, R. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons. 1991.

[13] Knuth, D. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms*. Addison Wesley. 1981.

[14] Lee, M. y Takagi, H. *Dynamic Control of Genetic Algorithms Using Fuzzy Logic Techniques*. First Online Workshop on Evolutionary Computing. http://www.bioele.nuee.nagoya-u.ac.jp

[15] Schaffer, D., Caruana, R., Eshelman, L. y Das, L., *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization*. Proceedings of the 3rd. International Conference on Genetic Algorithms. George Mason University. 1989. pp. 51-60.

[16] Spears, William M. *Adapting Crossover in Evolutionary Algorithms*. Por aparecer en Proceedings of the Fourth Annual Conference on Evolutionary Programming, San Diego, CA. o en ftp://ftp.aic.nrl.navy.mil/pub/spears/ep95.ps.Z

[17] Spears, William M. *Crossover or Mutation?* Navy Center for Applied Research in Artificial Intelligence. ftp://ftp.aic.nrl.navy.mil/pub/spears

[18] Whitley, Darrell. *A Genetic Algorithm Tutorial*. Colorado State University. 1993.