Modelado de sistemas mediante regresión basada en un AG

Joel Artemio Morales Viscaya
Tecnológico Nacional de México
Instituto Tecnológico de La Paz
Baja California Sur, México
Email: iscviscaya@gmail.com

Carlos Antonio Olachea Hernández
Tecnológico Nacional de México
Instituto Tecnológico de La Paz
Baja California Sur, México
Email: antonio.mx.9605@gmail.com

Marco Antonio Castro Liera
Tecnológico Nacional de México
Instituto Tecnológico de La Paz
Baja California Sur, México
Email: mcastroliera@gmail.com

***Resumen*—En este trabajo se presenta una estrategia para el modelado de sistemas basada en un algoritmo genético, el algoritmo produce modelos para problemas de hasta tres variables independientes combinando linealmente funciones simples. Las funciones se escogen mediante una versión de un AG para una representación en que el cromosoma de cada individuo es una combinación, por lo que se presentan propuestas de operadores genéticos para dicha representación. Los coeficientes de las combinaciones lineales se producen mediante plantear el problema como uno de mínimos cuadrados. Se presenta también una interfaz gráfica para el algoritmo en Java así como pruebas de su desempeño en un par de conjuntos de datos.**

I. Introduccion

Con frecuencia es deseable describir en términos matemáticos el comportamiento de algunos sistemas o fenómenos de la vida real, ya sean físicos, sociológicos o incluso económicos. La descripción matemática de un sistema de fenómenos se llama modelo matemático y se construye con objetivos específicos. Por ejemplo, podemos desear controlar la dinámica de cierto sistema electromecánico al estudiar la relación entre sus variables de estado, o podemos analizar el decaimiento de una sustancia radiactiva ya sea en el fósil o en el estrato en que fué descubierto para datarlo [1].

El carácter empírico o teórico constituye la característica fundamental de un modelo. Un modelo teórico se basa en las leyes físicas que rigen los procesos, un modelo empírico se basa en las observaciones directas o los resultados de experimentos del fenómeno estudiado. Si bien en muchos casos los modelos teóricos son preferibles dado su mayor capacidad explicativa, modelos empíricos han demostrado poseer un mayor poder predictivo en sistemas complejos [2].

A la utilización de técnicas estadísticas para estimar las relaciones entre variables se le conoce como regresión, el principal problema de la regresión paramétrica es que presupone una forma explícita para la relación entre las variables en el modelo.

La estrategia que se propone es un algoritmo para construir modelos empíricos basado en las técnicas de regresión, pero sin presuponer una forma específica para el modelo, lo que algunos autores llaman regresión no paramétrica, con la particularidad de no producir un sistema donde las relaciones entre las variables sean complejas (casi uno de caja negra o gris), como ocurre con técnicas más avanzadas de modelación, como son las redes neuronales artificiales o los sistemas basados en lógica difusa.

Las expresiones matemáticas que produce el algoritmo propuesto son combinaciones lineales y productos de funciones simples escogidas utilizando una adaptación de un algoritmo genético (AG), cuyos coeficientes se calculan de manera que minimicen la norma cuadrada del error.

La principal ventaja de aproximar información discreta o relaciones complicadas entre variables, con funciones analíticas sencillas, radica en su mayor facilidad de evaluación y manipulación, situación necesaria en ingeniería.

Los algoritmos genéticos, surgen en los años 1970, de la mano de John Herny Holland [3], son llamados así porque se inspiran en la evolución biológica y su base genético-molecular, actualmente constituyen una de las líneas más prometedoras de la inteligencia artificial.

Estos algoritmos hacen evolucionar una población de individuos o soluciones del problema que se pretende resolver, sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

II. Representación de los individuos

Los cinco componentes necesarios para implementar un AG que resuelva un problema cualquiera, son [4]:

1. Una representación de soluciones potenciales al problema.
2. Una función de evaluación que juega el papel del ambiente, calificando a las soluciones producidas en términos de su “aptitud“.
3. Una forma de crear una población inicial de soluciones potenciales (esto se efectúa normalmente de manera aleatoria, pero también pueden usarse métodos determinísticos).
4. Operadores genéticos que alteran la composición de los descendientes (normalmente se usan la cruza y la mutación).
5. Valores para los diversos parámetros utilizados por el algoritmo genético (tamaño de la población, probabilidad

ISBN: 978-607-98174-6-6

de cruza y mutación, número máximo de generaciones, etc.)

Debido a que las funciones que se van a combinar linealmente para cada individuo se escogen de un banco de funciones candidatas, el cromosoma o la representación de las soluciones, es una combinación que toma n elementos de los N totales que hay en el banco, que se almacena como un arreglo de dimensión n, al que llamaremos $C = [c_1, c_2, \ldots, c_n]$.

Sin embargo, en el algoritmo propuesto cada combinación no es directamente una solución al problema de regresión, sino la estructura de la función que se utilizará para construir el modelo (el cromosoma C determina cuáles funciones se utilizarán).

Para representar a los individuos dentro de la población de soluciones del algoritmo genético se utilizó una estructura de datos. Esta consta de las propiedades C Cromosoma, con la estructura de la función escogida para hacer regresión, $\{a\}$, con las constantes particulares que producen el mejor ajuste de dicha estructura y AP con la aptitud o calidad de la solución, como se aprecia en la figura (1).

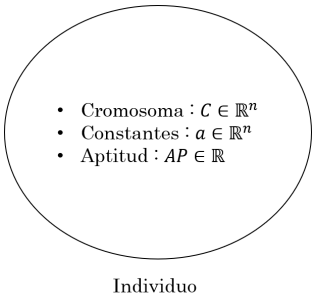

Figura 1. Representación gráfica de un individuo.

Las funciones que se pueden combinar linealmente para construir el modelo, es decir, el banco de funciones de la implementación del algoritmo, incluyen polinomios, exponenciales, gaussianas, sigmoideas así como funciones trigonométricas básicas y sus productos.

Cada función está asociada a un índice al que se hace referencia mediante los elementos que contiene la combinación (alelos, en terminología de AG); es importante remarcar que la representación tradicional de permutaciones no es adecuada para este problema, ya que el espacio vectorial de funciones que genera un conjunto de n funciones es el mismo sin importar en que orden se combinen linealmente estas.

El número de funciones N en el banco, que directamente es el número de funciones distintas que se pueden utilizar para la construcción del modelo, es un parámetro de diseño. El valor mínimo de N depende del número de variables independientes, once para una sola, 34 para dos variables y 79 para tres variables independientes. Al incrementar el parámetro se van agregando al banco de funciones seleccionables los productos, estiramientos y encogimientos de estas. De forma que no existe un valor máximo para N. Sin embargo, valores demasiado grandes para N implican un considerable incremento en el tiempo de ejecución del algoritmo.

El número de funciones que se van a combinar linealmente para construir los individuos, que es equivalente a la longitud del cromosoma, es otro parámetro de diseño al que denotamos como n. Más funciones producen un mejor ajuste pero al mismo tiempo modelos más complejos de analizar analíticamente. Determinar dicho parámetro lleva a la conocida discusión del compromiso entre buen ajuste a los datos y la simplicidad en el modelo.

Para el caso $N = 31$, $n = 5$, un individuo cuyo cromosoma sea como el de la figura (2).

$$C = \begin{array}{|c|c|c|c|c|} \hline 1 & 8 & 15 & 19 & 24 \\ \hline \end{array}$$

Figura 2. Representación de los alelos de un cromosoma.

representa a la función $\phi(x) = a_1\phi_1(x) + a_2\phi_8(x) + a_3\phi_{15} + a_4\phi_{19} + a_5\phi_{24}$, en este caso

$$\phi_1 = e^x \qquad \phi_8 = e^x \cos x \qquad \phi_{15} = \frac{1}{1+e^{-2x}}$$

$$\phi_{19} = e^{2x}\cos 2x \qquad \phi_{24} = e^{3x}x^3$$

Es decir, el cromosoma representa la función

$$\phi(x) = a_1 e^x + a_2 e^x \cos x + a_3 \frac{1}{1+e^{-2x}} + a_4 e^{2x}\cos 2x + a_5 e^{3x}x^3$$

III. Estructura general del algoritmo propuesto

El primer paso del algoritmo consiste en la carga y normalización de los datos con los que se llevará a cabo la regresión, debido a que es posible que exista una diferencia significativa en el orden de magnitud de los valores entre distintas variables independientes.

Para llevar a cabo la normalización de los datos, se calcula primero el valor máximo y mínimo de cada variable independiente. Posteriormente para cada valor x_i de la variable independiente x se produce $\bar{x_i}$ utilizando (1).

$$\bar{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Posteriormente se genera una población inicial de individuos, cuyos cromosomas se construyen aleatoriamente mediante combinaciones de números enteros sin repetición, cuyo rango va desde cero hasta el tamaño del banco N, como se puede apreciar en el siguiente pseudocódigo.

Algorithm Inicialización de los individuos.

1: $nL \leftarrow$ longitud de C
2: **for** $i = 0$ **to** nL **do**
3: Generar valor aleatorio $r \in \{0 \dots N \mid r \notin C\}$.
4: $c_i \leftarrow r$
5: **end for**

Los pasos restantes, la parte principal del algoritmo genético que se muestra en la figura (3), se llevan a cabo iterativamente en un ciclo que se repite tantas veces como un parámetro de diseño $Gmax$.

En este ciclo, a cada individuo de la población se le calcula su aptitud y se le aplica una sucesión de operadores genéticos (cruza y mutación) con el objetivo de generar nuevos individuos que sean potencialmente mejores en términos de la aptitud.

Estos nuevos individuos junto con algunos de los mejores de la población actual, pasarán a la siguiente población, en un símil con la selección natural.

Además se utiliza un enfoque conocido como elitismo, que conserva al mejor individuo de la población en cada una de las iteraciones $Gmax$, esto con el propósito de que el mejor individuo de la población (que representa la solución del AG al problema), en la iteración $k + 1$ sea al menos tan bueno como el de la iteración k [5].

IV. El cálculo de aptitud

La evolución de los individuos en un algoritmo genético depende directamente del cálculo de aptitud de los mismos, este permite evaluar que tan buena es una solución para resolver el problema en particular y, mediante un algoritmo de selección, preservar dicha solución en posteriores iteraciones.

En este caso al tratarse de un problema de regresión, y poseer los individuos solamente la estructura de la función a utilizar, el cálculo no es demasiado simple y puede ser dividido en dos partes.

1. Encontrar los coeficientes que permitan un máximo ajuste de la función de regresión a los datos.
2. Calcular un índice de desempeño para determinar que tan bueno es el ajuste de una función de regresión.

Formalmente la primera parte del problema puede plantearse como el de, dado un conjunto de m pares ordenados (x, y) y un conjunto de funciones de la forma

$$\phi_i(x), i \in \{1, 2, \dots, n\} \tag{2}$$

encontrar los valores de $a_1, a_2, \dots, a_n$ que hacen que la función de aproximación

$$\phi(x) = a_1\phi_1(x) + a_2\phi_2(x) + \cdots + a_n\phi_n(x) \tag{3}$$

sea tal que para toda $i \in \{1, 2, \dots, m\}$

$$\phi(x_i) \approx y_i. \tag{4}$$

Primero es necesario definir el índice de desempeño, que utilizaremos para determinar cuando decimos que $\phi(x) \approx y$.

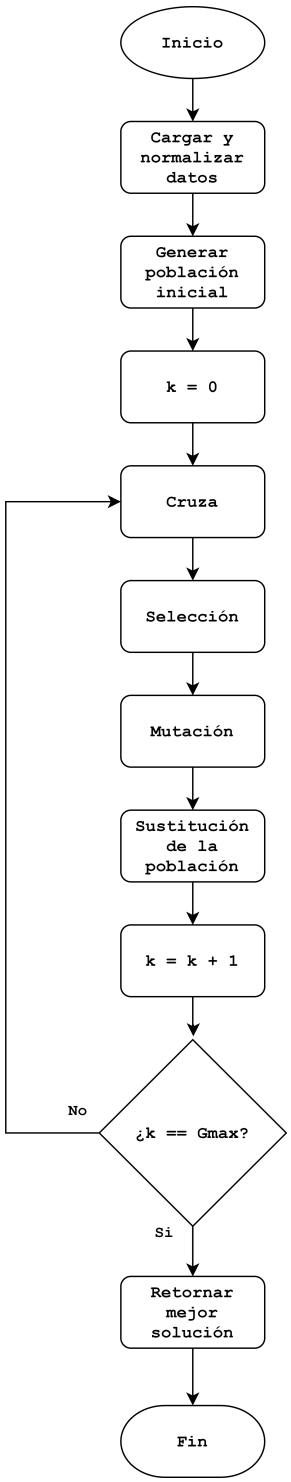

Figura 3. Diagrama del algoritmo genético.

Esta cercanía entre $\phi(x)$ y y la definimos en el sentido de mínimos cuadrados como

$$\min_{(a_1,a_2,\ldots,a_n)} E = \frac{1}{2}\sum_{k=1}^{m}[\phi(x_k) - y_k]^2 \quad (5)$$

esto debido principalmente a que es posible encontrar los coeficientes de máximo ajuste $a_1, a_2, \ldots, a_n$ a partir de este índice de desempeño y a que se utiliza con frecuencia en la literatura [6].

Sustituyendo (3) en (5) tenemos

$$E = \frac{1}{2}\sum_{k=1}^{m}[a_1\phi_1(x_k)+a_2\phi_2(x_k)+\cdots+a_m\phi_m(x_k)-y_k]^2 \quad (6)$$

es decir

$$E = \frac{1}{2}\sum_{k=1}^{m}[\sum_{j=1}^{n} a_j\phi_j(x_k) - y_k]^2. \quad (7)$$

Para resolver el problema de minimización (5) de manera analítica es necesario encontrar el vector gradiente de E respecto a $a_1, a_2, \ldots, a_n$.

Derivando E de (7) respecto a a_i se tiene

$$\frac{\partial E}{\partial a_i} = \sum_{k=1}^{m}[\sum_{j=1}^{n} a_j\phi_j(x_k) - y_k]\phi_i(x_k). \quad (8)$$

Para cada i, queremos los valores a_j que hacen que $\frac{\partial E}{\partial a_i} = 0$. Esto es

$$\sum_{j=1}^{n}[\sum_{k=1}^{m}\phi_j(x_k)\phi_i(x_k)]a_j = \sum_{k=1}^{m}\phi_i(x_k)y_k. \quad (9)$$

Para obtener directamente la expresión de la solución de (9) hay que notar que para cada i, se tiene que el error está dado por

$$e_i = \phi(x) - y_i = \sum_{j=1}^{n} a_j\phi_j(x_i) - y_i. \quad (10)$$

Si

$$\bar{\phi} = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_m) & \phi_2(x_m) & \cdots & \phi_n(x_m) \end{pmatrix},$$

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \text{ y } \vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

entonces el vector de errores es

$$\vec{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix}$$

Y el problema a resolver (5) es equivalente a minimizar

$$E = ||\vec{e}||^2 = \vec{e}^T\vec{e} = (\bar{\phi}\vec{a} - \vec{y})^T(\bar{\phi}\vec{a} - \vec{y}) \quad (11)$$

El cálculo del gradiente de E produce (9) que puede ser escrita en términos de $\bar{\phi}$, $\vec{a}$ y $\vec{y}$ como

$$\bar{\phi}^T\bar{\phi}\vec{a} = \bar{\phi}^T\vec{y} \quad (12)$$

cuya solución por mínimos cuadrados produce directamente los coeficientes $a_1, a_2, \ldots, a_n$ de mejor ajuste de la función de regresión para cada individuo. El valor de aptitud de cada individuo resulta en el valor de la función de error E (7) con los coeficientes encontrados.

V. Operadores genéticos

En los sistemas biológicos, la cruza es un proceso complejo que ocurre entre parejas de cromosomas. Estos cromosomas se alinean, luego se fraccionan en ciertas partes y posteriormente intercambian fragmentos entre sí [4].

En el cómputo evolutivo tradicional se simula la cruza intercambiando segmentos de cadenas lineales de longitud fija (los cromosomas padre), produciendo de esta manera nuevos individuos (hijos) con propiedades similares a las de los padres.

La técnica de cruza desarrollada en esta estrategia, mostrada en el Algorithm Cruza, está orientada a problemas de combinatoria donde el orden de los alelos es irrelevante, es decir, solo importan los valores en el cromosoma y no el orden en que aparezcan en el individuo.

Se seleccionan aleatoriamente dos individuos en la población, a los que llamaremos padres. Los alelos de los padres se concatenan y se ordenan, resultando en una lista, con la cuál se generan dos nuevos individuos (hijos) a partir de tomar, en cada caso, los elementos impares o pares de dicha lista. Esto garantiza, que los alelos de los padres se transmiten y que además, los hijos no tendrán alelos repetidos (y las características comunes de los padres, las tendrán ambos hijos).

Los hijos generados mediante la cruza, corresponderán a la mitad de individuos de la nueva población y el resto serán elegidos mediante un proceso de selección que se será descrito más adelante.

En el siguiente algoritmo se aprecia la cruza que se explicó anteriormente.

ISBN: 978-607-98174-6-6

Algorithm Cruza.

```
1: CP es la población actual.
2: NP es la población nueva.
3: PL ← longitud de CP.
4: for i ← 0 to PL/2 do
5:    Seleccionar dos padres F1 y F2 de CP.
6:    Concatenar las c's de F1 y F2 en L.
7:    Ordenar L.
8:    Crear hijos S1 y S2.
9:    Declarar LS ← longitud de L.
10:   Declarar indexador k ← 0.
11:   for j = 0 to LS do
12:      S1.cj ← Lk .
13:      S2.cj ← Lk+1 .
14:      k ← k + 2.
15:   end for
16:   if S1.C = F1.C or S1.C = F2.C then
17:      Copiar aptitud y constantes de F similar, a S1.
18:   else
19:      Evaluar S1
20:   end if
21:   if S2.C = F1.C or S2.C = F2.C then
22:      Copiar aptitud y constantes de F similar, a S2.
23:   else
24:      Evaluar S2
25:   end if
26:   NPi ← S1.
27:   NPi+1 ← S2.
28: end for
```

Para la selección, cuyo pseudocódigo se muestra en el Algorithm Selección, de los mejores individuos, se utilizó la estrategia conocida como selección por torneo, debido a que su complejidad es de orden menor a otras estrategias y además no produce una presión de selección muy alta [4].

Se considera un tamaño de torneo $t = 2$, donde se comparan las aptitudes AP de dos individuos seleccionados aleatoriamente de la población actual, y aquel con menor aptitud pasará a formar parte de la nueva población.

Algorithm Selección.

```
1: for i ← PL/2 to PL − 1 do
2:    Seleccionar dos individuos I1 y I2 de CP.
3:    if I1.AP < I2.AP then
4:       NPi ← I1.
5:    else
6:       NPi ← I2.
7:    end if
8: end for
```

Además se considera elitismo, lo cual garantiza la permanencia de la mejor solución en cada iteración, al colocarlo como el último elemento de la población siguiente NP en cada iteración.

Se denomina mutación a un operador que forma un nuevo cromosoma a través de alteraciones (usualmente pequeñas) de los valores de los genes de uno de los cromosomas.

La mutación se considera un operador secundario en algoritmos genéticos tradicionales [4]. Es decir, su uso es menos frecuente que el de la cruza. El objetivo de la mutación es el de evitar un estancamiento en óptimos locales; valores altos para la probabilidad de mutación se asocian a una búsqueda más en anchura que en profundidad y mantienen una mayor diversidad en las soluciones.

Debido al carácter secundario de la mutación en un AG, este operador solo se aplica a algunos individuos de la población, estos se escogen a partir de generar un número aleatorio y en caso de superar un parámetro de diseño PM, el individuo se mutará.

La estrategia utilizada para llevar a cabo la mutación, se muestra en el Algorithm Mutación. Consiste en tomar de forma aleatoria algún gen del cromosoma (una posición entre 0 y n del arreglo) e insertar un alelo distinto (un número entero entre 0 y N) que no se encuentre en el cromosoma, lo que se conoce como una mutación por inserción.

Algorithm Mutación.

```
1: for i ← 0 to PL do
2:    Generar valor aleatorio r1 ∈ (0, 1).
3:    if r1 ≤ PM then
4:       Generar valor aleatorio r2 ∈
         {0 . . . N | r2 ∉ NPi.C}.
5:       Seleccionar ck aleatoria de NPi.
6:       NPi.ck ← r2.
7:    end if
8: end for
```

VI. Implementación del algoritmo

El algoritmo desarrollado se implementó en Java 8 utilizando la IDE de NetBeans 8.2. Además se le dotó de una interfaz gráfica de usuario. Esta, inicialmente proporciona una pantalla en la cual se pueden ajustar los parámetros del algoritmo genético y la regresión como se ve en la figura (4).

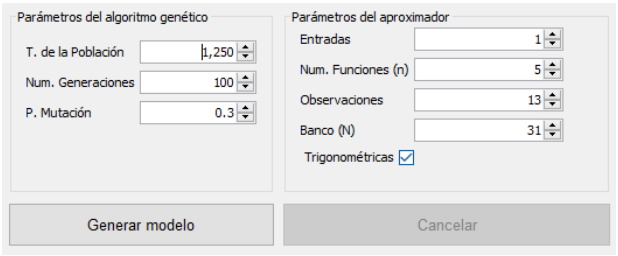

Figura 4. Menú de configuración del software.

Una vez introducidos los parámetros del algoritmo, es posible seleccionar en el menú file, la opción cargar datos, desde la cuál se selecciona un archivo csv con los valores que se utilizarán en la regresión. Además de cargarlos en memoria, los valores de las variables independientes se escalan entre 0

y 1 como se mencionó anteriormente y se muestran en forma de tabla en la parte derecha de la ventana, como se aprecia en la figura 5.

Datos del Archivo

C1	C2	C3
0.6471154401432173	0.9922202777590811	3.7121
0.6936995899824827	0.8414874582127393	3.9542
0.7403029894704422	0.7129565749029639	4.1919
0.7860594044158695	0.6033463462789707	4.42
0.8299678530866812	0.5098845101074692	4.6328
0.8709311055073244	0.43023210159072045	4.8237
0.9076786848640012	0.36241474276996255	4.986
0.9389208646942193	0.3047679544995625	5.1133
0.9634256674815684	0.2558880774494627	5.1997
0.9800573639531078	0.21458599762177205	5.2407
0.9879497199176116	0.17984227412443068	5.2335
0.9865637452116499	0.15078189854389626	5.1777
0.9758224412404473	0.12663783626124608	5.0751
0.9560723016804943	0.10673419936730162	4.9303
0.9281988103717107	0.09046381054946043	4.7493
0.8933569462357311	0.07728119180633149	4.5398
0.8530289322219868	0.0667011621906621	4.3098
0.8087547402259909	0.05829042426690001	4.0671

Figura 5. Tabla de los datos cargados.

Con los datos cargados, se habilita la opción de generar el modelo, la cuál no sólo lleva a cabo el algoritmo mencionado anteriormente, sino que además lo presenta de manera explícita y la despliega en una gráfica (en caso de ser una o dos variables independientes) en la que también pueden ver los datos como puntos rojos.

Si se trata de una función de una variable independiente se muestra la curva de regresión $\phi = f(x)$, como en la figura (6). En caso de dos variables independientes se muestra la superficie $\phi = f(x_1, x_2)$, como en la figura (7). Para funciones de tres variables independientes no se despliega ninguna gráfica (ya que su gráfica vive en un espacio de cuatro dimensiones).

El programa muestra adicionalmente la magnitud del error, así como el tiempo de ejecución del algoritmo, como se puede ver en la figura 8.

En la parte inferior se muestra la ecuación interpretada que mejor se ajusta al set de datos. Se dan las opciones de exportar la ecuación en formato **png** o copiar la ecuación en formato **LATEX** para poder ser insertada en documentos compatibles.

Finalmente en la figura 10, se muestra la pantalla completa del software y la ubicación de los componentes anteriormente descritos.

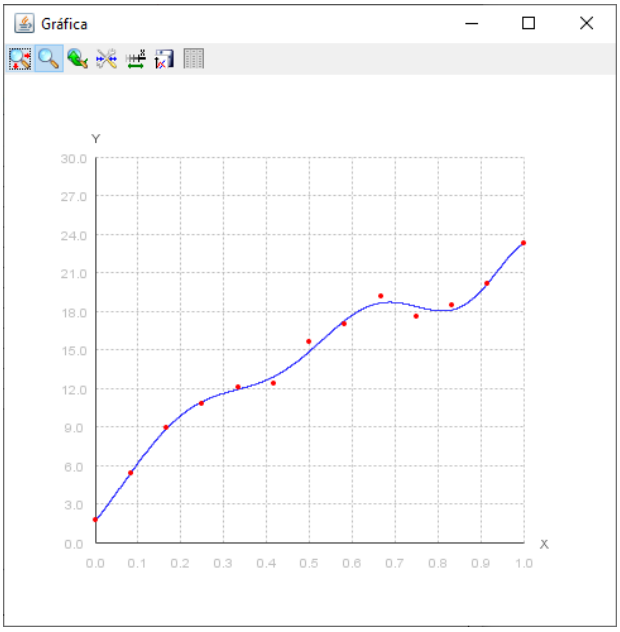

Figura 6. Ejemplo de una curva de regresión.

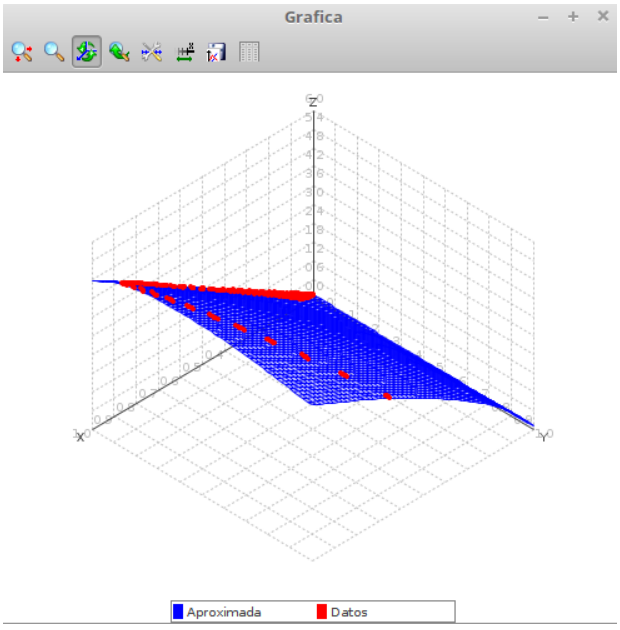

Figura 7. Ejemplo de una superficie de regresión.

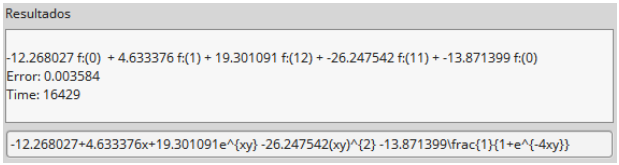

Figura 8. Resultados de la ejecución del algoritmo.

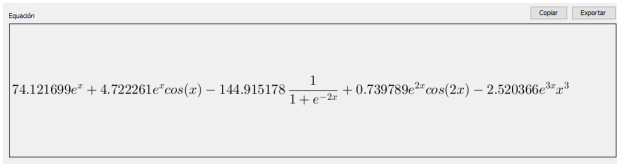

Figura 9. Ecuación obtenida del software.

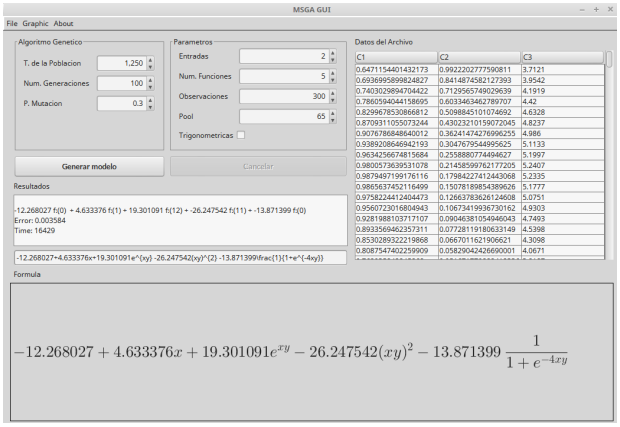

Figura 10. Pantalla completa del software.

VII. Resultados y trabajo futuro

La implementación desarrollada se probó en una computadora Aspire Acer modelo F5, con un procesador Intel Core i5-6200, 2.30 GHz y 8 GiB de RAM, bajo el sistema operativo Linux Mint 17 con dos conjuntos de datos. El primero con 490 registros del comportamiento de un bioreactor, en el cúal se calcula la biomasa en futura a partir de la biomasa actual y el sustrato [7], es decir, un problema con dos variables independientes.

El segundo conjunto de datos, consta de 1,562 registros acerca del comportamiento de un levitador neumático presentado en [8], con tres variables independientes: posición anterior, velocidad anterior y una variable de control relacionada con la velocidad del dispositivo que produce la levitación, la variable de salida es la posición siguiente.

En el caso del modelo del bioreactor, se consiguió un ajuste considerablemente bueno en un tiempo razonablemente pequeño, como se puede apreciar en el Cuadro 1, utilizando $N = 130$, $GMax = 100$, $PL = 100$ y $PM = 0{,}3$. Al incrementar n el tiempo de ejecución es mayor pero la magnitud del error disminuye como era de esperar.

Modelo	Suma de errores al cuadrado	Tiempo
MSAG $n = 3$	0.1524	1.924 s
MSAG $n = 5$	0.0242	3.546 s
MSAG $n = 7$	0.0006	5.108 s

Cuadro I

Modelo	Suma de errores al cuadrado	Tiempo
MSAG $n = 3$	76.715	7.777 s
MSAG $n = 5$	69.124	18.662 s
MSAG $n = 7$	17.640	27.705 s

Cuadro II

En el caso del modelo del levitador neumático, utilizando los mismos parámetros, los resultados se pueden observar en el Cuadro 2. Se aprecia que el ajuste no es demasiado bueno en términos del error al cuadrado, sin embargo, este archivo de datos se construyó concatenando los resultados de tres diferentes pruebas en el levitador; cada una con diferentes objetivos de control.

Es bastante interesante notar que si se utiliza cada conjunto por separado es posible conseguir un error de 0 utilizando $n = 3$ para el primer set, con la función $w(x, y, z) = 2{,}380952 - 4{,}05e^{xy} - 99{,}7642(xyz)^2$. Para los dos conjuntos restantes el mismo comportamiento se consigue con $n = 5$, por lo que un modelo con una función definida a pedazos parece una mejor idea.

Otra causa del error cuadrado tan alto, además de los diferentes objetivos de control, es que los registros incluyen cálculos de velocidad por diferencias finitas y estimaciones de posición mediante visión artificial. Además el conjunto de datos no ha sido preprocesado y podría contar con la presencia de datos anómalos u *outlayers*.

A pesar de todas las limitaciones mencionadas, un modelo producido con este algoritmo se utilizó en [8] para llevar a cabo simulaciones del prototipo levitador que permitieron el ajuste adecuado de los parámetros utilizados en el cálculo de la entrada de control, lo que se conoce como sintonía de controladores.

Queda como trabajo futuro, comparar el modelo producido por esta estrategia contra modelos más sofisticados, así como probar el desempeño del mismo con parámetros distintos o con otras estrategias de cruza, selección o mutación.

Referencias

[1] D. G. Zill. *Ecuaciones diferenciales con aplicaciones de modelado*. Ed. por Cengage Learning Editores. 9.ª ed. 2009.

[2] D. Livignstone, D. Manallack y I Tetko. “Data modelling with neural networks: Advantages and limitations”. En: *Journal of Computer-Aided Molecular Design* (1997), págs. 135-142. DOI: https : / / doi . org / 10 . 1023 / A : 1008074223811.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ed. por Michigan: MIT Press. 6.ª ed. 2001. 205 págs.

[4] C. A. Coello Coello. *Introducción a la Computación Evolutiva*. Notas de curso. México, D.F.: CINVESTAV-IPN Departamento de Computación, mayo de 2013.

[5] J. Koljonen y J. T. Alander. *Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms. In Proceedings of the ninth Scandinavian conference on artificial intelligence*. Reporte de conferencia. Oct. de 2006, págs. 54-60.

[6] Peña J. *Notas del curso de Métodos Numéricos*. Centro de Investigación en Matemáticas, 2013.

[7] M. A. Castro Liera. "Un algoritmo Genético Distribuido con aplicación en la identificación difusa de un proceso fermentativo". Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. Santa Clara, Cuba: Universidad Central "Marta Abreu" de las Villas, 2009.

[8] Geraldo Sánchez O. P. "Diseño, construcción y control de un sistema de levitación neumática". Tesis presentada en opción al grado de Ing. Electromecánico. La Paz, Baja California Sur: Instituto Tecnológico de La Paz, jun. de 2019.