EVFUZZYSYSTEM: EVOLUCIÓN DE SISTEMAS DIFUSOS PARA PROBLEMAS DE REGRESIÓN MULTI-DIMENSIONALES

M. Martínez-Ballesteros[1] **Víctor M. Rivas**[2] **M.J. del Jesus**[2]

[1] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, mariamartinez@us.es
[2] Departamento de Informática, Universidad de Sevilla,{vrivas, mjjesus}@ujaen.es

Resumen

Este paper introduce la versión multidimensional de EvFuzzySystems, y un método evolutivo que permite el diseño simultáneo de funciones miembro y conjunto de reglas apropiados. La versión publicada previamente, mostraba buen comportamiento pero sólo podría ser aplicada en problemas con dos entradas y una salida. Extenderlo para ser aplicado a problemas con entradas multidimensionales no ha sido una tarea trivial, desde el punto de vista computacional. Este trabajo muestra los resultados de un nuevo método para problemas de regresión.

Palabras Clave: Sistemas difusos, Algoritmos genéticos, Algoritmos evolutivos, Métodos híbridos, Función de aproximación.

1 INTRODUCCIÓN

En 2003, Rivas et al. [1] presentó un algoritmo genético nuevo diseñado para evolucionar sistemas difusos de dos dimensiones. El paper describió un algoritmo evolutivo capaz de diseñar simultáneamente las funciones miembro así como el conjunto de reglas necesarias para construir sistemas de lógica difusa (SLD). Sin embargo, el algoritmo, fue centrado en SLD de dos dimensiones, por ejemplo, sistemas para problemas con 2 entradas y 1 salida. Pero en el paper, se deja abierta la extensión futura del algoritmo, es decir, que podría ser aplicado a problemas multi-dimensionales. Desde un punto de vista computacional, esta nueva característica del algoritmo no fue trivial. Representó cambios profundos tanto en la representación usada para los individuos tanto como en los mecanismos para manejar el modelo, principalmente, los operadores genéticos diseñados específicamente para el algoritmo.

Este paper introduce el nuevo algoritmo EvFuzzySystem, el cual puede ser usado para diseñar SLD en problemas de regresión de un número indeterminado de entradas. Para esto, han sido usadas estructuras de datos adecuadas para representar los individuos y se han desarrollado nuevos operadores genéticos para evolucionar dichas estructuras.

2 EL ALGORITMO EVOLUTIVO

Las siguientes subsecciones detallan las arquitecturas tanto del SLD como del AE que lo optimiza, así como la función fitness y los operadores genéticos usados.

2.1 CODIFICACIÓN DE LOS INDIVIDUOS: EL SISTEMA DE LÓGICA DIFUSA

El esquema de representación usado para codificar cada una de las soluciones, sigue el enfoque Pittsburg, en el cual se tiene que "Cromosoma = Base de Reglas?, puesto que cada individuo va a representar un conjunto de reglas, es decir, una solución al problema.

En el algoritmo implementado, para representar conjuntos difusos de las variables del antecedente de la regla, se han usado conjuntos difusos triangulares, es decir, la funciones de pertenencia de los términos lingísticos será una función triangular como muestra la figura 1.

Un cromosoma, es decir, la solución de un individuo, va a representar un SLD, un Sistema de Lógica Difusa, que será implementado como una estructura que almacena distintos elementos: los centros de las particiones triangulares de las funciones de pertenencia de las variables de entrada, y los valores de la variable de salida, es decir, dicha estructura contiene tanto los precedentes como los consecuentes de las reglas del

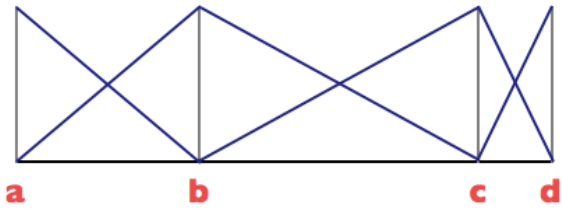

Figura 1: Conjuntos difusos de una variable del algoritmo implementado.

SLD.

De modo más amplio, dicha estructura está compuesta por otras dos estructuras, las cuales representan el conjunto de variables de entrada de n variables con un número m de términos, que puede variar en cada variable y el conjunto de consecuentes cuyo tamaño será igual al producto de los términos de cada variable respectivamente. Cada gen, es decir, cada uno de los elementos que forman tanto el conjunto de variables de entrada como el conjunto de consecuentes, serán codificados mediante valores reales, los cuales tendrán por alelos valores reales comprendidos entre un máximo y un mínimo establecido. Del mismo modo, la función de evaluación asociará a la estructura en cuestión, un valor real de evaluación que identifica el valor de aptitud de la solución obtenida, que cuanto mayor sea, mejor será la solución.

El conjunto de variables de entrada es implementado mediante un vector de variables. Cada variable a su vez está formado por un vector de reales donde son almacenados los centros de las particiones triangulares de las funciones de pertenencia constituyendo el valor de los precedentes, además de contener el rango que comprende dichos centros y el nombre de la misma. Por lo cual resulta que el conjunto de variables de entrada es implementado como una matriz bidimensional de antecedentes.

Para un conjunto de variables de entrada con un número de n=2 variables y un número máximo de 5 términos, resulta ser como se muestra en la figura 2.

$$MV(n, mmax) = \begin{vmatrix} X_1 & X_2 & X_3 & X_4 & X_5 \\ Y_1 & Y_2 & Y_3 & Y_4 & Y_5 \end{vmatrix}$$

Figura 2: Conjunto de variables del algoritmo implementado.

Un ejemplo de una variable con un número m=5 de términos sería como se muestra en la figura 3.

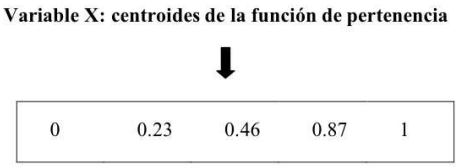

Figura 3: Centroides de una variable del algoritmo implementado.

En cada variable el primer centroide y el último corresponden al valor mínimo y máximo establecido respectivamente, determinando así el rango del universo de discurso de los términos pertenecientes a cada variable. El número mínimo de términos en cada variable ha de ser igual a 3, mientras que el número máximo de términos viene dado mediante uno de los parámetros de entrada del algoritmo evolutivo. Por tanto, el universo de cada variable se particiona en tantos términos como existan. Para ello, para cada variable se determina un número aleatorio de términos comprendido entre 3 y el máximo número especificado.

Posteriormente el universo se particiona de la siguiente forma:

- En primer lugar se hacen tantas particiones como número de términos corresponda.
- El primer centroide corresponde con el valor mínimo especificado para dicha variable.
- A continuación se genera un valor entre el valor mínimo y el valor del límite superior de la primera partición.
- Para el resto de centroides, se genera un aleatorio entre el valor del centroide anterior y el valor del límite superior de la partición en la que nos encontremos.
- El valor para el último centroide coincide con el máximo valor para la variable.

Los valores se generan de esta forma para que exista uniformidad, pero a la vez se introduzca una pequeña diversidad. Se debe tener en cuenta que el número de términos entre variables no tiene porqué ser el mismo, ya que el tamaño de los mismos se genera aleatoriamente entre el rango establecido.

Cada matriz 2-D, como muestra la figura 2, que implementa el conjunto de variables de entrada de un SLD está dividido en:

De $MV[0,0]$ a $MV[0,m]$ (Primera fila) almacena los centros de las particiones triangulares de las funciones de pertenencia de la primera variable de entrada, por ejemplo X.

La variable X toma valores en el rango $[x_{min}, x_{min}]$, teniéndose que cumplir lo siguiente en cualquier momento:

$x_{min} = MV[0,0] < MV[0,1] < ... < MV[0,m] = x_{max}$ con $m \geq 3$, siendo m el número de términos.

El resto de filas corresponden a cada una de las variables restantes cumpliéndose en cada momento lo mencionado anteriormente. De forma generalizada: $x_{imin} = MV[i,0] < MV[i,1] < ... < MV[i,m] = x_{imax}$

$\forall i$, 1≤i≤n siendo n el número de variables y $m \geq 3$ siendo el número de términos. El valor de m puede cambiar en cada variable puesto que es generado de forma aleatoria.

El conjunto de valores de la variable de salida es implementado mediante un vector de consecuentes. Cada consecuente es un valor real representando un valor de salida para una combinación de valores de entrada. Esta estructura además contiene el rango que comprende dichos valores y el número de consecuentes a generar. Este número viene dado por el producto del número de términos de cada una de las variables, puesto que cada consecuente es una salida para cada una de las posibles combinaciones de los términos que constituyen a cada variable. Para una variable de salida con un número de consecuentes igual a 12, resulta ser como se muestra en la figura 4:

Figura 4: Estructura de consecuentes del algoritmo implementado.

Cada celda de este vector almacena un valor para Z, la variable de salida, donde: $z_{min} \leq MC[i] \leq z_{max}$ $\forall i$, 1≤i≤n siendo n el número de consecuentes.

Los valores del conjunto de consecuentes son generados aleatoriamente entre un valor mínimo y un máximo dados sin conservar ningún tipo de orden entre los mismos.

En este ejemplo tenemos dos variables de entrada, suponemos que son$x$ e y, y que la variable de salida es z. Para este caso, la fórmula para acceder a la posición del consecuente que corresponde a una combinación de valores de entrada conociendo la posición de los mismos dentro del conjunto de variables de entrada sería:

$$vz = vx \cdot ny + vy \tag{1}$$

donde vx y vy serían las posiciones de los términos correspondientes a los valores de cada una de las variables de entrada una vez aplicado el motor de inferencia, el cual será detallado más adelante, y ny el número de términos que tiene la segunda variable. El resultado obtenido sería la posición del consecuente correspondiente a dichos valores de entrada. En general, para obtener la posición del consecuente correspondiente a una combinación de valores de entrada teniendo n variables, sería la suma de los productos de la posición del término correspondiente a cada variable por el producto del número de términos que tienen cada una de las variables siguientes, menos en el caso de la última variable que se multiplica por 1 al no haber variables sucesivas.

$$F(v_0, ..., v_n) = \sum (v_i \prod_{j=i+1} n_j) \tag{2}$$

El Sistema de Lógica Difusa trabaja usando reglas de la siguiente forma:

SI X es $MV[0,i]$ y Y es $MV[1,j]$ ENTONCES Z es $MC[k]$

Siendo i la posición del término correspondiente a la variable X, j la posición correspondiente a la variable Y, y k la posición del consecuente correspondiente resultante de $F(i,j)$.

Los valores x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, z_{min}, son parámetros que son proporcionados al algoritmo.

2.2 GENERACIÓN DE LA POBLACIÓN INICIAL

Para la creación de la población de soluciones inicial, en cualquier caso, se hará de forma aleatoria tanto para el conjunto de variables de entrada como el conjunto de valores de la variable de salida, así como el número de términos para cada variable de entrada según un máximo dado.

2.3 OPERADORES GENÉTICOS

2.3.1 Recombinación

Este operador une valores de un SLD a otro. Teniendo en cuenta que estos dos SLD no necesariamente tienen

la mismas dimensiones, la recombinación no puede ser realizada de manera trivial.

El operador trabaja de la siguiente manera:

1. Aleatoriamente se elige un SLD para ser modificado (será llamado M_r, o SLD destinatario), y otro que proporcionará los genes para ser recombinados (M_d, o SLD donante). Solo la primera M_r será modificada.

2. Se elige un bloque aleatorio de celdas de la estructura de consecuentes correspondiente a M_r. Para especificar este bloque sólo se seleccionarán dos celdas aleatoriamente correspondientes a los dos extremos o esquinas del bloque que son capaces de determinar el bloque entero. Para seleccionar aleatoriamente estas dos celdas, se crearán dos vectores con las posiciones correspondientes a un término de cada variable, las cuales serán seleccionadas de manera aleatoria. Posteriormente se codificaran dichos vectores para obtener las posiciones correspondientes en el conjunto de consecuentes. Llámense estas dos celdas $M_r[r_1]$ y $M_r[r_2]$. Imaginando espacialmente la estructura multidimensional que constituyen los consecuentes, las celdas a modificar con valores procedentes de M_d, serán las incluidas en el bloque espacial que forman las dos celdas seleccionadas aleatoriamente, como extremos de bloque, es decir, aquellas celdas del consecuente cuya posición al ser decodificada (aplicándole una función de decodificación para conocer la combinación de posiciones de términos pertenecientes a las variables) contiene las posiciones de los términos de cada variable dentro del rango establecido por las posiciones elegidas aleatoriamente como esquinas del bloque.

3. Para cada celda $M_r[r_i]$, decodificada tal que así, para saber la posición del término correspondiente en cada variable a la que se refiere la celda en cuestión, $M_r[r_{1i}, r_{2j}, ...r_{nm}]$, (donde r_{1i} va de r_{11} a r_{12}, r_{2j} va de r_{21} a r_{22} ...y... r_{nm} va de r_{n1} a r_{n2}) el operador hace lo siguiente:

4. Del
conjunto de variables de entrada perteneciente al SLD donante, M_d, de cada variable se selecciona el término cuyo valor se acerca más al término de la misma variable correspondiente al SLD receptor M_r.

5. Finalmente, se modifica el valor almacenado en $M_r[r_i]$, por el almacenado en la celda perteneciente a M_d que resulta tras codificar las posiciones de los términos de cada variable obtenidos en el punto anterior.

2.3.2 Adición de funciones de pertenencia (Mutador de incremento de tamaño)

Este operador modifica la estructura de un SLD aadiendo una función de pertenencia a las variables de entrada. El operador trabaja de la siguiente manera:

1. Aleatoriamente se elige un SLD para ser modificado.

2. Se selecciona aleatoriamente la variable dentro del conjunto de variables de entrada que va a ser modificada.

3. Se elige aleatoriamente la posición del nuevo término dentro de la variable seleccionada. Esta posición ha de ser distinta de la primera y la última y obviamente encontrarse dentro de dicho rango.

4. Se obtiene un valor aleatorio para el nuevo término entre el valor del término situado justo en la posición anterior y la posición posterior.

5. El siguiente paso es calcular dentro del conjunto de consecuentes las posiciones que van a ser modificadas, tras la inserción de un nuevo término o función de pertenencia dentro de una de las variables del conjunto de variables de entrada. Para ello, se calculan las posiciones dentro de los consecuentes que al ser decodificadas contienen la posición del nuevo término. Tras esto, habrá que calcular el desplazamiento que sufre cada una de ellas al insertarse una nueva. Para evitar problemas de incosistencia en las posiciones calculadas a la hora de insertar consecuentes, esto se hará de forma creciente.

6. Una vez calculadas dichas posiciones se inserta el nuevo término en la variable seleccionada dentro del conjunto de variables de entrada con el valor aleatorio obtenido.

7. Por último, dentro del conjunto de consecuentes se insertarán nuevos valores en las posiciones calculadas en el paso 5. Estos, serán valores aleatorios generados por una función Gaussiana centrada en el punto medio entre el valor del consecuente situado encima del actual espacialmente, es decir, el correspondiente a aquel cuya posición decodificada contiene la misma pero con la posición del término nuevo decrementado en una unidad, de la variable que ha sido modificada nuevos valores, y aquel situado por debajo espacialmente, análogamente, con la posición del término modificado incrementado en una unidad.

La manera en que este operador trabaja asegura que el SLD resultante es siempre válido, porque los valores del nuevo centroide y de los nuevos consecuentes son establecidos de forma que los resultados siguen un determinado orden o se encuentran dentro de un rango válido (por ejemplo, el valor para el nuevo centroide es mayor que el precedente y menor que el siguiente).

El algoritmo para el operador de adición de un nuevo término, centroide o función de pertenencia opera de la misma forma para cualquier variable.

2.3.3 Eliminación de funciones de pertenencia (Mutador de decremento de tamaño)

Este operador modifica la estructura de un SLD decrementando el número de funciones de pertenencia en las variables de entrada. El operador trabaja como sigue:

1. Aleatoriamente se elige un SLD para ser modificado.
2. Se selecciona aleatoriamente la variable dentro del conjunto de variables de entrada que va a ser modificada.
3. Se elige aleatoriamente la posición del término a eliminar dentro de la variable seleccionada. Esta posición ha de ser distinta de la primera y la última y obviamente encontrarse dentro de dicho rango. En el caso de que el número de términos sea el mínimo permitido, es decir, que tenga un número de términos menor o igual a 3, no se eliminará ninguno.
4. El siguiente paso es calcular dentro del conjunto de consecuentes, las posiciones que van a ser modificadas tras la eliminación del término o función de pertenencia dentro de una de las variables del conjunto de variables de entrada. Para ello, se calculan las posiciones dentro de los consecuentes que al ser decodificadas contienen en la variable seleccionada, la posición del término a eliminar.
5. Una vez calculadas dichas posiciones, tendrá lugar el borrado de los consecuentes situados en las mismas. Para evitar problemas de inconsistencia en las posiciones en cuanto a eliminar consecuentes, esta tarea será realizada en orden decreciente, es decir, empezando por detrás.
6. Por último, se elimina el término elegido en la variable seleccionada dentro del conjunto de variables de entrada.

Al igual que el operador anterior, este operador trabaja de forma que asegura que el SLD resultante es siempre válido, puesto que no puede eliminar términos de una variable de un SLD cuyo número de términos es el mínimo permitido, además de no poder elegir tanto el primero como el último para ser eliminados y obviamente, tras eliminar un término el SLD permanece ordenado.

El algoritmo para el operador de eliminación de un término, centroide o función de pertenencia opera de la misma forma para cualquier variable.

2.3.4 Modificación de un valor centroide (Mutador de precedentes)

Este operador modifica el valor almacenado en un término de una de las variables del conjunto de variables de entrada.

El operador trabaja así:

1. Aleatoriamente se elige un SLD para ser modificado
2. Se selecciona aleatoriamente la variable dentro del conjunto de variables de entrada que va a ser modificada.
3. Se elige aleatoriamente la posición del término dentro de la variable seleccionada al cual va a modificarse su valor. Esta posición ha de ser distinta de la primera y la última y obviamente encontrarse dentro de dicho rango.
4. Se obtiene un nuevo valor calculado de forma aleatoria para el término seleccionado entre el valor del término anterior y posterior.
5. Por último modificamos el valor del término con el nuevo valor calculado.

Tal y como los operadores anteriores, este operador también trabaja de forma que asegura que el SLD resultante es siempre válido, ya que no se pueden modificar tanto el primer como el último término y obviamente, tras modificar el término, el SLD permanece ordenado, puesto que siempre se cálcula entre el valor precedente que es menor y el siguiente que es mayor. El algoritmo para el operador de eliminación de un término, centroide o función de pertenencia opera de la misma forma para cualquier variable.

2.3.5 Modificación de un valor consecuente (Mutador de consecuentes)

Este operador modifica el valor almacenado en un consecuente del conjunto de consecuentes de un determinado SLD.

El operador trabaja de la siguiente manera:

1. Aleatoriamente se elige un SLD para ser modificado.

2. Se selecciona aleatoriamente la posición del consecuente cuyo valor va a ser modificado.

3. Se obtiene un nuevo valor calculado de forma aleatoria determinado por una función Gaussiana centrada en el valor existente para el consecuente seleccionado, y variando entre el valor mínimo zmin y el valor máximo zmax de los consecuentes.

4. Por último modificamos el valor del consecuente con el nuevo valor calculado.

Tal y como los operadores anteriores, este operador también trabaja de forma que asegura que el SLD resultante es siempre válido, ya que se generan los valores dentro de un rango determinado. El algoritmo para el operador de eliminación de un término, centroide o función de pertenencia opera de la misma forma para cualquier variable.

2.4 Función fitness

La función fitness u objetivo es una función de evaluación de la medida de calidad de un cromosoma. Para este problema será un valor real que se obtendrá mediante el uso directo de la función a maximizar. El cálculo del fitness de cada individuo se realiza en base a un conjunto (llamado conjunto de entrenamiento) de pares de entrada-salida de una función conocida. El fitness asignado al SLD es la inversa de la distancia del Error Cuadrático Medio Normalizado (Normalized MSE) de las salidas correctas conocidas a las salidas producidas por el SLD, calculadas usando el conjunto de entrenamiento.

$$F_j = \frac{N}{\sum_{i=1}^{N}((z_i - z_{ij})/(z_{max} - z_{min}))^2)} \qquad (3)$$

Dónde F_j es el fitness para el individuo número j, N es el número de ejemplos el conjunto de entrenamiento (training set), z_i es la salida correcta, y z_{ij} es la salida proporcionada por el SLD. Un caso especial es cuando $z_i = z_{ij}$, $\forall i$ $1 \leq$ i $\leq$ m, porque F_j sería igual a $N/0$; en este caso, F_j es asignado con un valor muy alto. Se debe tener en cuenta que el tamaño del SLD no interviene en la computación o cálculo del fitness.

2.5 PROCESO EVOLUTIVO

El algoritmo evolutivo usado consiste en la selección elitista, tamaño de población fijado y los operadores descritos. He aquí los principales pasos:

1. Crear la primera población, compuesta de p individuos generados aleatoriamente de tamaño aleatorio (con un límite máximo del número de términos dado por el problema como parámetro de entrada, solo para esta primera generación). Se establece el contador de generaciones a 1. p es también un parámetro de entrada del problema (tamaño de la población).

2. Computar el valor del fitness para cada individuo.

3. Ordenar los individuos desde el valor de fitness más alto al más bajo.

4. Seleccionar los q mejores individuos (subconjunto élite de la población). Estos formarán parte del subconjunto élite de la población, a la cual van a aplicarse los operadores. q viene dado indirectamente como parámetro de entrada del problema, ya que es proporcionado un parámetro "non elite rate" , que indica el porcentaje de la población total que ha de ser sustituido por nuevos individuos. Por tanto q es igual (1 - non elite rate) por el tamaño de la población.

5. Generar p - q nuevos SLDs, para después incrementar la población a p. Cada nuevo individuo es creado mediante el duplicado de un individuo del subconjunto élite de la población, y después de eso, aplicando uno de los operadores genéticos a la copia. La probabilidad de uno de los individuos del subconjunto élite de ser seleccionado para la reproducción, es relativo a su valor fitness. El método de selección empleado es la selección por ruleta.

6. Evaluar los nuevos individuos.

7. Reemplazar los q peores individuos (es decir, los q individuos cuyo fitness es menor al resto) de la población por los individuos de la subpoblación resultantes de haber elegido los q mejores individuo de la población y posteriormente aplicarles un operador genético.

8. Ordenar la población tras el reemplazamiento de dicha subpoblación.

9. Incrementar el contador de generaciones, e ir al paso 3, a menos que se haya alcanzado el número de generaciones especificado.

10. Finalmente, el mejor SLD encontrado es el primer individuo de la actual (o última) población ya que los individuos están ordenados desde el más alto al más bajo fitness.

Este algoritmo se ejecuta con los siguientes parámetros:

- Tamao de la pobiación.
- Proporción de individuos que son eliminados en cada nueva generación (non-elite-rate).
- Probabilidad de cada operador.
- Máximo número de términos para la primera generación.
- Máximo número de generaciones.
- Numbero de variables.

Cada operador tiene asociado una probabilidad de aplicación que no cambia durante la aplicación del algoritmo. Además, cada nuevo individuo es creado mediante la aplicación de uno, operador de copia o duplicación de sus padres, operador de recombinación o cualquiera de los mutadores. Usualmente, el operador de recombinación es aplicado con mayor probabilidad para permitir la mezcla de bloques construidos, mientras que los mutadores son aplicados con pequeñas probabilidades para escapar de búsquedas aleatorias. De otro lado, todos los operadores de mutación comparten el mismo radio o probabilidad para balancear el incremento o decremento del tamaño de los SLDs. Se utilizará como criterio de parada el alcance de un número máximo de generaciones realizadas. Es decir, el programa finalizará cuando el número de veces que se ejecuta el algoritmo completo supera un n número de veces dado por el problema como parámetro de entrada. El algoritmo completo se ejecutará 1 vez para cada instancia del problema.

3 EXPERIMENTOS Y RESULTADOS

En esta seccin se tiene como objetivo comprobar el funcionamiento del algoritmo desarrollado, as como validar el comportamiento de los operadores creados. Para lo cual, se han configurado y ejecutado una serie de experimentos tanto con el EvFuzzySystem as como otros mtodos y algoritmos de regresin ya existentes.

La experimentacin y anlisis de resultados que han sido realizados est dividido en:

- Experimentacin con el algoritmo evolutivo desarrollado segn las funciones del artculo ?Evolving two-dimensional fuzzy system?.
- Experimentacin con el algoritmo evolutivo desarrollado y con algoritmos contenidos dentro de la herramienta KEEL.

3.1 Descripcin de las fuentes de datos utilizadas

Para realizar el proceso de toma de resultados y de anlisis del algoritmo evolutivo implementado, se han utilizado a partir de diferentes bases de datos provenientes de fuentes distintas, que podran catalogarse en dos categoras distintas: funciones propuestas del artculo inicial y fuentes de datos provenientes de la herramienta KEEL.

3.2 Fuentes de datos del artculo "Evolving two-dimensional fuzzy systems"

En primer lugar, tenemos las bases de datos referentes al artculo que extiende el algoritmo, las cuales provienen de una serie de funciones de dos variables. Las funciones empleadas han sido:

$$F(x,y) = sin(xy) \tag{4}$$

$$F(x,y) = e^{xsin(xy)} \tag{5}$$

$$F(x,y) = \frac{40e^{8(x-0.5)^2+(y-0.5)^2}}{e^{8((x-2)^2+(y-0.7)^2)} + e^{((x-0.7)^2+(y-0.2)^2)}} \tag{6}$$

$$F(x,y) = sin(2\pi\sqrt{x^2+y^2}) \tag{7}$$

3.3 Fuentes de datos de la herramienta KEEL

En segundo lugar, se han utilizado 4 conjuntos de datos obtenidas de la tarea de regresin de la herramienta KEEL, con tal de poder realizar una comparativa con el resto de los algoritmos de regresin integrados. Dichos datos se encuentran disponibles en la seccin KEEL-dataset dentro del sitio www.keel.es. Los conjuntos de datos con los cuales se han desarrollado los experimentos son las siguientes:

- Daily-electric-energy average cost in Spain in 2003
- Electrical-Length
- Friedman Benchmark problem
- MachineCPU

3.4 Experimentacin y resultados para el artculo "Evolving two-dimensional fuzzy systems""

Para llevar acabo los experimentos relacionados con el artculo, se han creado ejemplos con 3 valores: 2 datos de entrada (X, Y) y una salida (Z). Por tanto, se ha generado un fichero por funcin con 400 ejemplos, donde los datos de entrada X e Y tienen valores igualmente espaciados, en un rango dado para cada una, el cual en este caso, es el mismo para ambas. Es decir, se han generado 400 puntos tomando 20 valores distribuidos uniformemente para X en un rango dado, y anlogamente para la Y, haciendo todas las posibles combinaciones entre ambas entradas. Los valores que toma la salida Z, son los resultantes tras aplicar la frmula en cada caso.

Para crear los conjuntos de entrenamiento y test de los experimentos, se han generado 5 particiones distintas para cada funcin, sin usar ningn tipo de particionador especfico. Para cada particin, fueron elegidos aleatoriamente 320 puntos para ser usados como conjunto de entrenamiento (80% del total de puntos), mientras que los 80 restantes fueron como conjunto de test (20% del total de puntos) para examinar las aptitudes del mejor SLD obtenido. Por tanto, para conseguir valores medios y desviaciones tpicas, el algoritmo fue ejecutado 5 veces, cada vez con una particin diferente de las generadas y con diferentes semillas aleatorias procedentes de la funcin getTime() dentro de la clase Date de Java, la cual devuelve el nmero de milisegundos desde el 1 de Enero de 1970, a las 00:00:00 GMT. Segn el momento en el que se ejecute el algoritmo, esta funcin devolver valores diferentes.

Los parmetros utilizados en el algoritmo evolutivo implementado son los mismos que se emplean en el algoritmo propuesto en el artculo.

Los valores utilizados en los parmetros durante las distintas ejecuciones del algoritmo son: 500 para el tamao de la poblacin, 300 para el nmero de generaciones, 40 mximo nmero de trminos, 0.1 probabilidad de recombinacin, 0.001 probabilidad del mutador de simensin, 0.01 probabilidad del mutador de precedentes, 0.01 probabilidad del mutador de consecuentes y 0.35 como porcentaje de individuos que no van a pertenecer a la poblacin lite. Cada experimento consta de 5 ejecuciones para cada funcin sumando un total de 4 experimentos de 5 ejecuciones cada uno suponiendo por tanto 20 ejecuciones realizadas.

A continuacin se muestran grficamente las salidas de cada funcin original as como la mejor solucin obtenida durante las cinco ejecuciones del algoritmo en cada problema.

//METER grafica 1

//METER grafica 2

//METER grafica 3

//METER grafica 4

Tabla 1: Resultados del algoritmo implementado para cada funcin del artculo.

ID	FITNESS	ERROR	TAMAÑO
1	39.18 (6.17)	0.02984 (0.0049)	95.4 (82.97)
2	124.72 (37.72)	0.01383 (0.00488)	99.4 (39.17)
3	209.3 (165.23)	0.00875 (0.00348)	99.8 (71.32)
4	32.75 (7.27)	0.05635 (0.01523)	159.2 (95.64)

4 CONCLUSIONES

Los experimentos incluidos, han sido realizados para validar el comportamiento de los operadores creados. Los relativos a las funciones del artículo "Evolving two-dimensional fuzzy system" demuestran que siguiendo las mismas características del algoritmo propuesto en el artículo, se obtienen resultados con diferencias no muy significativas en cuanto al Error de Generalización, sólo consiguiendo peores valores en cuanto al tamao. Por tanto para funciones de dos variables de entrada el EvFuzzySystem presenta un comportamiento adecuado.

Los experimentos realizados con KEEL se han diseado mayormente para comprobar el funcionamiento de EvFuzzySystem con más de dos variables de entrada. Como hemos podido ver, los resultados obtenidos no son los más óptimos pero sin embargo, no son los peores. Además presenta un comportamiento similar para cada uno de los problemas con los que hemos trabajado, mientras que otros algoritmos son muy buenos para algunos problemas y muy pésimos para otros. El hecho de no obtener resultados más óptimos, puede ser consecuencia de varios motivos, entre ellos, el valor de los parámetros que igual no son los más adecuados, ya que no conocemos la posible equivalencia existente con respecto a los parámetros del resto de algoritmos. No obstante, se ha verificado que el algoritmo funciona para más de 2 variables de entrada el cual era el principal objetivo a cumplir en este trabajo.

5 TRABAJOS FUTUROS

Como mejoras y futuras tareas que se puedan aplicar al algoritmo desarrollado, cabe destacar:

- Uso de algún tipo de búsqueda local en orden

de afinar las soluciones encontradas por el algoritmo genético, consiguiendo valores más precisos para los consecuentes. De este modo, un algoritmo genético realiza una búsqueda global, mientras que el operador local de búsqueda, trata de mejorar el individuo en algún pequeo vecindario.

- Encontrar alguna otra forma de computar el fitness para reducir considerablemente el tiempo necesario para la ejecución del algoritmo.
- Realizar variaciones en los actuales operadores, como por ejemplo, en la recombinación, permitir también el intercambio de términos dentro de las variables. En la selección, se podría optar por usar otro tipo de métodos, como por ejemplo selección por torneo.
- Una vez que el proceso evolutivo ha terminado, se podría llevar a cabo una etapa de post-procesamiento para eliminar reglas que no sean significativas para el problema en concreto del cual se esté tratando.
- Aplicar otro tipo de algoritmos genéticos en vez de uno básico, como por ejemplo uno del tipo CHC.
- Aplicar algoritmos genéticos multiobjetivo, puesto que tanto el algoritmo CHC como el AG tienen grandes limitaciones cuando los problemas a los que nos enfrentamos tienen más de una variable de decisión.

5.1 TABLAS

Todas las tablas deberán estar referenciadas en el texto, como se ha hecho con la tabla 2, centradas y claras. El número y título siempre aparecerán en la parte superior de la tabla. Las tablas deben ser numeradas de manera correlativa.

Tabla 2: Ejemplo de título de una tabla.

NOMBRE	**DESCRIPCIÓN**
A	Subconjunto difuso de X
μ_A	Función de pertenencia de A

Agradecimientos

La palabra agradecimientos debe ir alineada a la izquierda, no numerada y en negrita. Todos los agradecimientos deben figurar al final del trabajo.

Referencias

[1] V.M. Rivas, J.J. Merelo, I. Rojas, G. Romero, P.A. Castillo, J. Carpio. Evolving two-dimensional fuzzy systems. *Fuzzy Sets and Systems* 138 Pág. 381-398, 2003.