Extracción de Conocimiento Mediante un Algoritmo Genético Multiobjetivo Estilo Pittsburgh y Reglas Lingüísticas Tipo DNF

Oscar Delgado, Jorge Casillas
Departamento de Ciencias de la Computación e Inteligencia Artificial,
Universidad de Granada, 18071 Granada, España

Resumen— **Los algoritmos genéticos son una de las formas más comunes de generar de forma automática la base de reglas de los Sistemas Basados en Reglas Difusas (SBRDs). Sin embargo, este proceso adolece de algunos problemas importantes, como son la generación de reglas redundantes, inconsistentes o sobregenerales. Estas reglas reducen la interpretabilidad de la solución y, según el tipo de problema que se trata de resolver, pueden llegar a ser inadmisibles. Nuestro objetivo en este trabajo se centra en diseñar un algoritmo que alcance una interpretabilidad muy alta sin renunciar al grado de aproximación. Para ello, proponemos un método de aprendizaje de la base de reglas que mantiene intacta la definición de las funciones de pertenencia (evitando así pérdida de interpretabilidad). Consideraremos la representación de reglas DNF (forma normal disyuntiva) para conseguir una alta capacidad de compactación, favoreciendo de esta forma la interpretabilidad.**

Palabras clave— **extracción de conocimiento, algoritmos genéticos, reglas difusas DNF.**

I. Introducción

En el proceso de descubrimiento de conocimiento en bases de datos podemos distinguir entre dos enfoques distintos [1]: inducción *predictiva* e inducción *descriptiva*. La diferencia radica en el principal objetivo que se persigue y, por tanto, en el método de aprendizaje empleado para conseguirlo. Por un lado, la inducción predictiva intenta generar modelos legibles que describan con la mayor fiabilidad el conjunto de datos que representa el sistema analizado. En este caso, la meta es usar el modelo aprendido para simular el sistema, obteniendo así un explicación del su comportamiento complejo. Por otro lado, la inducción descriptiva se centra en obtener patrones particulares que resulten de interés. En este segundo caso, no se obtiene una visión global de las relaciones entre las variables sino que descubrimos una serie de reglas concretas (suficientemente diferentes entre ellas) estadísticamente significativas.

Este trabajo se centra en la primera vía, la inducción predictiva, para abordar problemas cuyas variables de entrada y salida toman valores reales y donde el conocimiento extraído resulta de interés para comprender mejor el sistema analizado. Para representar el conocimiento, y con la intención de generar modelos suficientemente legibles (que, sin duda, es una de las premisas fundamentales en todo proceso de extracción de conocimiento), proponemos el uso de sistemas basados en reglas difusas (SBRD). Estos sistemas emplean reglas difusas del tipo SI-ENTONCES para representar el conocimiento sobre el problema.

La extracción automática de SBRD se puede realizar con diferentes técnicas de aprendizaje, ya sea con algoritmos voraces sencillos [2], [3] o con técnicas de optimización tales como las redes neuronales [4], [5] y los algoritmos genéticos (GA) [6]. Debido a la orientación seguida en este trabajo hacia generación de conocimiento con alto grado de interpretabilidad, proponemos el uso de algoritmos genéticos ya que creemos que presenta una serie de ventajas útiles para el objetivo de este trabajo. Por un lado, tienen una potente capacidad de búsqueda que permite trabajar con optimización multiobjetivo. Por otro, pueden manejar estructuras de representación flexibles mezclando esquemas de codificación o incluyendo restricciones. Desde principios de los 90 se ha investigado en el uso de algoritmos evolutivos para diseñar o aprender automáticamente partes de estos SBRD [7], [8], [9]. A estos sistemas de aprendizaje se les suele conocer con el nombre de sistemas difusos genéticos o, más genéricamente, *sistemas difusos evolutivos*.

Independientemente del método utilizado para obtener la base de conocimiento nos encontramos con un problema esencial, el de generar un modelo que sea, simultáneamente, todo lo preciso e interpretable que sea posible. En efecto, cualquier modelo difuso cuenta con estas dos características contradictorias: *interpretabilidad*, es decir, la capacidad de expresar el comportamiento del sistema real de una forma comprensible, y la *precisión*, la capacidad de representar con exactitud el sistema modelado. Por supuesto, lo ideal sería satisfacer ambos criterios simultáneamente con un alto grado, pero como son contradictorios, esto no es posible normalmente. Esta área, el compromiso entre interpretabilidad y precisión, es objeto de numerosos trabajos de investigación hoy en día [10], [11].

El algoritmo que proponemos se enfrenta a este compromiso entre interpretabilidad y precisión. Para ello, la propuesta se centra sólo en aprender el conjunto de reglas, sin modificar las funciones de perte-

Trabajo soportado en parte por el Proyecto TIN2005-08386-C05-01 del MEC y fondos FEDER. E-mail del autor: casillas@decsai.ugr.es

nencia asociadas a los distintos términos lingüísticos. Además, emplea una representación de regla difusa con un alto grado de compacidad y síntesis de conocimiento. Por último, considera un proceso de optimización multiobjetivo que permite generar una gama amplia de soluciones con diferente balance entre interpretabilidad y precisión.

El resto del artículo se organiza de la siguiente manera. En la sección II hablaremos sobre el balance interpretabilidad-precisión y algunos métodos existentes para alcanzarlo. En la sección III comentaremos la representación de las reglas difusas utilizada en nuestro algoritmo, que es presentado en la sección IV, estudiando en detalle cada uno de sus pasos, los problemas que pueden presentarse y cómo resolverlos. En la sección V estudiaremos los resultados obtenidos utilizando dos problemas reales. Acabaremos en la sección VI con unas conclusiones sobre el trabajo y los logros conseguidos.

II. Balance Interpretabilidad-Precisión en Modelización Difusa

Como hemos mencionado, los modelos difusos comparten dos objetivos contradictorios, interpretabilidad y precisión. Como la mejora de uno de ellos habitualmente provoca el degradado del otro, existen dos aproximaciones en función del criterio principal:

- *Modelos lingüísticos*, principalmente desarrollados con reglas tipo Mamdani y dirigido a obtener interpretabilidad.
- *Modelos precisos*, obtenidos habitualmente con SBRD tipo Takagi-Sugeno y focalizados en la precisión de sus soluciones.

Independientemente del modelo elegido, es habitual utilizar el mismo método para obtener el deseado balance entre interpretabilidad y precisión:

1. El objetivo principal define el modelo a utilizar (lingüístico o preciso).
2. Los componentes del modelo, como su estructura o procesos, se mejoran con mecanismos diseñados para compensar la diferencia inicial entre ambos objetivos. Así, se buscarán mejoras en la precisión en los modelos lingüísticos y mejoras en la interpretabilidad en los modelos precisos.

Podemos encontrar algunos ejemplos de estos mecanismos en la literatura:

- *Modelos lingüísticos con precisión mejorada*: La mejora se lleva a cabo modificando las funciones de pertenencia, a través de sus formas [8], [12], [13], [14], [15], [16], [17], sus tipos (triangular, trapezoidal, etc.) [18], o su contexto (definiendo toda la semántica) [19], ajustando el número de términos lingüísticos de las particiones difusas [20], extendiendo la estructura del modelo utilizando modificadores lingüísticos [14], [21], o utilizando pesos (factores de importancia para cada regla) [22], entre otros.

Por otra parte, a pesar de que la reducción de la base de reglas [23], [24] y la selección de las variables de entrada [25], [26] mejoran la interpretabilidad, también pueden verse como mejoras en la precisión si se consideran criterios como la redundancia e inconsistencia entre reglas.

- *Modelos precisos con interpretabilidad mejorada*: En este caso la mejora se obtiene reduciendo el conjunto de reglas difusas [27], [28], [29], [30], reduciendo el número de conjuntos difusos, con la consecuente fusión de reglas [31], [32], [33], o reduciendo el número de variables de entrada [34].

Nuestra propuesta comparte varios de estos mecanismos, concretamente la extensión de la estructura del modelo, la reducción de la base de reglas y la selección de variables de entrada, para conseguir soluciones altamente interpretables sin renunciar a un grado de precisión admisible.

III. Representación de las Reglas Difusas

Para garantizar un alto grado de legibilidad hemos optado por una descripción más compacta de las relaciones difusas basada en la *forma normal disyuntiva* (DNF). Este tipo de regla difusa tiene la siguiente estructura:

SI X_1 es $\widetilde{A_1}$ y ... y X_n es $\widetilde{A_n}$ **ENTONCES** Y es B

donde cada variable de entrada X_i, $i \in \{1, \ldots, n\}$, toma valores de un conjunto de términos lingüísticos $\widetilde{A_i} = \{A_{i1} \vee \ldots \vee A_{il_i}\}$, cuyos miembros se unen mediante un operador de disyunción (en nuestro caso, la T-conorma de la *suma ponderada*, $min\{1, a+b\}$). Esta estructura es un soporte natural para permitir la ausencia de variables de entrada en cada regla simplemente haciendo que $\widetilde{A_i}$ sea el conjunto de términos lingüísticos completo.

Sin embargo, existen algunos inconvenientes con el uso de reglas DNF que aparecen cuando se intenta aprender una base de reglas completa debido a que surgen colisiones con mucha facilidad. Básicamente, estas colisiones son de dos tipos.

El primero de ellos es la *inconsistencia*. Se dice que dos reglas son inconsistentes entre sí cuando sus antecedentes se solapan (es decir, sus antecedentes son iguales, uno contiene a otro o coinciden en algunas etiquetas de todas las variables de entrada con otro) y tienen distinto consecuente. Por ejemplo, las dos reglas siguientes son inconsistentes, porque la segunda variable de la primera regla contiene a la de las segunda y, sin embargo, sus consecuentes son diferentes:

```
SI X1 es A1 y X2 es {A1 o A2} ENTONCES Y es B3
SI X1 es A1 y X2 es A1 ENTONCES Y es B2
```

Otro problema que puede presentar esta representación de reglas es la *redundancia*. Dos reglas se consideran redundantes si tienen el mismo antecedente o éste está solapado y cuentan con el mismo consecuente. Veámos un ejemplo:

```
SI X1 es A1 y X2 es {A1 o A2} ENTONCES Y es B2
SI X1 es A1 y X2 es A1 ENTONCES Y es B2
```

La redundancia puede darse por subsumisión, como en el caso anterior, o por solapamiento, como en el siguiente:

```
SI X1 es A1 y X2 es {A1 o A2} ENTONCES Y es B2
SI X1 es A1 y X2 es {A2 o A3} ENTONCES Y es B2
```

El método de aprendizaje propuesto en este trabajo intenta dar respuesta a estos problemas mediante operadores concretos que garantizan la consistencia y falta de redundancia por subsumisión en las soluciones obtenidas.

IV. Descripción del Algoritmo

A continuación mostramos una descripción en pseudo-código del algoritmo:

```
Inicializacion(P)
Evaluacion(P)
HM = CalculoHipermatriz()
Mientras (no condicion de parada)
    P1 = Seleccion(P)
    P2 = Cruce(P1)
    P3 = Mutacion Antecedente(P2)
    P4 = Mutacion Consecuente(P3)
    P5 = Operador de Completitud(P4)
Fin-Mientras
```

La población está compuesta por individuos cuyos cromosomas son de longitud variable y codifican un conjunto de reglas difusas tipo DNF. Cada regla se codifica de forma binaria en el antecedente (donde '1' significa que se usa el correspondiente término lingüístico), siendo éste de tamaño igual al número de etiquetas de todas las variables de entrada.

Dado que no se permite más de una etiqueta en una variable de salida, el consecuente se puede codificar con valores enteros (donde cada gen contiene el índice de la etiqueta usada en la variable correspondiente), siendo esta parte de tamaño igual al número de variables de salida. Sin embargo, para mantener la coherencia con la codificación utilizada en el antecedente, usamos también consecuentes binarios.

Finalmente, reseñar que uno de los requisitos del algoritmo es que, durante el proceso de evolución, todos los individuos deben contener al menos un '1' en cada una de sus variables de entrada. Si no fuera así, significaría que esa variable no se usa, y esta circunstancia ya está contemplada cuando todas las etiquetas están a '1' (y, por tanto, tendríamos dos individuos genotípicamente cercanos pero fenotípicamente muy alejados).

A. Inicialización

Dado que buscamos completitud total, necesitamos partir con reglas que cubran todos los ejemplos. Por esta razón, utilizamos el conocido algoritmo de Wang y Mendel [2]. Concretamente, cada cromosoma se genera con el mínimo número de reglas que cubren los ejemplos según este algoritmo y con consecuente aleatorio para cada regla. De esta forma, todos los cromosomas parten con el mismo número de reglas, siendo éstas lo más específicas posible.

B. Cálculo de la Hipermatriz de Completitud

El objetivo de este paso, novedoso respecto a los sistemas difusos genéticos tradicionales, es generar una estructura de datos que será utilizada en varias ocasiones a lo largo de cada iteración del algoritmo. Esta estructura, que hemos llamado *hipermatriz de completitud*, almacena las combinaciones de etiquetas del antecedente que cubren cada uno de los ejemplos del conjunto de datos.

Su forma es la de una hipermatriz de dimensión igual al número de variables, conteniendo en cada posición un '1' si esa combinación concreta corresponde a un ejemplo (n-dimensional) y un '0' en caso contrario. De esta forma la hipermatriz permitirá conocer si una determinada regla corresponde a algún ejemplo. Esto hace posible diseñar operadores modificados (como el de mutación) o nuevos (como el de completitud) que evitan generar reglas "problemáticas" que, en cualquier caso, tendrían que ser eliminadas con posterioridad.

La implementación de esta estructura ha de realizarse de forma especialmente eficiente, debido a sus exigentes requisitos de tiempo de acceso a la información. En este trabajo se decidió implementar la hipermatriz utilizando una tabla *hash*, cuya claves se forman con la concatenación de las etiquetas de las reglas que contiene. Para optimizar el rendimiento de la tabla en espacio y tiempo de recuperación de información, las combinaciones '0' no se almacenan. Consideramos que si una determinada clave no existe, su valor es '0'.

C. Operador de Cruce

Un operador de cruce tradicional genera hijos cuyo cromosomas son una combinación de los cromosomas de los padres. En nuestro caso, el operador de cruce sólo intercambia reglas entres los dos conjuntos, pero no las modifica. Por otra parte, también garantiza que los descendientes no contienen inconsistencias ni redundancias.

Un pseudo-código del funcionamiento del operador puede encontrarse en la figura 1.

D. Operador de Mutación del Antecedente

Este operador será una de las fuentes de nuevas reglas, que exploran otras zonas del espacio de búsqueda. Como su nombre indica, actúa sobre las variables de los antecedentes, con probabilidad proporcional al número total de variables de entrada de la población.

En el funcionamiento de este operador tendremos en cuenta que sólo permitiremos añadir un '1' en aquellos genes de reglas que sabemos que cubren, al menos, un ejemplo (teniendo en cuenta, también, el resto de variables). Para ello, utilizaremos la hipermatriz de completitud y buscaremos la combinación de variables de entrada de la regla a mutar. Si esta combinación existe, permitiremos la mutación. Si no, intentaremos otra combinación, si es posible.

Algoritmo: Operador de Cruce.
Entrada: Dos individuos (dos padres).
Salida: Dos individuos (dos hijos).
Precondiciones: Los individuos recibidos no tienen inconsistencias internas.
Postcondiciones: Los individuos generados no tienen inconsistencias ni redundancias por subsumisión, aunque sí pueden contener redundancias por solapamiento.

1. Meter todas las reglas procedentes de los dos padres en un conjunto, S.
2. Analizar aquellas reglas inconsistentes entre sí (que siempre provendrán de los dos padres, ya que en cada padre no hay inconsistencias). Además, las reglas no tienen porqué ser inconsistentes por parejas. Por ejemplo, una regla del padre 1 puede ser inconsistente con dos reglas del padre 2.
3. Separar cada grupo de reglas inconsistentes en dos subconjuntos en función del padre del que provienen y sacar estas reglas de S. Asignar cada subconjunto a un hijo distinto.
4. Lanzar un número aleatorio en r$\sim$U[1,$|S|$] que definirá el número de reglas de S que se asignan al hijo 1, siendo el resto ($|S|$-r) el número de reglas asignadas al hijo 2.
5. Elegir aleatoriamente r reglas de S para asignarlas al hijo 1.
6. Asignar el resto al hijo 2.
7. Este proceso puede haber generado redundancias en los hijos. Para evitarlas, para cada hijo, comprobar si un antecedente está contenido en otro; si es así, eliminar la regla más específica.

Fig. 1. Operador de cruce

Algoritmo: Operador de Mutación del Antecedente.
Entrada: Un individuo.
Salida: Un individuo mutado en su antecedente.
Precondiciones: El individuo recibido no tiene inconsistencias internas.
Postcondiciones: El individuo generado no tiene inconsistencias ni redundancias por subsumisión, aunque sí puede contener redundancias por solapamiento.

1. Elegir la variable de entrada concreta sobre la que se va a actuar. Para ello, se escoge aleatoriamente una variable de toda la población, que corresponderá a una regla concreta de un cromosoma concreto.
2. Elegir aleatoriamente entre alguna de las siguientes operaciones: *contracción*, *expansión* o *desplazamiento*. Cada una de ellas tienen unas restricciones para poder ser aplicadas, así que no siempre todas serán posibles, en cuyo caso se elegirá entre las que queden disponibles.

Fig. 2. Operador de Mutación del Antecedente

Un pseudo-código del funcionamiento del operador puede encontrarse en la figura 2.

D.1 Operación de Contracción

Esta operación hace a la regla mutada más específica, eligiendo un gen de esa variable que contenga un '1' y poniéndolo a '0'. Claramente, la contracción sólo se puede aplicar cuando hay, al menos, dos '1', pues de otro modo todos los genes de esta variable quedarían a '0'.

Esta operación nunca causará problemas de consistencia o redundancia (pues, como hemos dicho, aumenta el grado de especificidad de la regla, lo que evita que entre en conflico con otras).

D.2 Operación de Expansión

Esta operación realiza el proceso inverso a la contracción, disminuyendo el grado de especificidad de la regla o, lo que esa lo mismo, haciéndola más general. Para ello, se escoge un gen con alelo '0' y se pone a '1'. En este caso, la restricción es que sólo se puede aplicar cuando hay, al menos, un '0' en los genes de la variable.

Desafortunadamente, esta operación sí que puede causar problemas de colisión con otras reglas. Para detectarlos y evitarlos procederemos de esta forma:

- Se elige aleatoriamente un gen a '0' de entre todos los disponibles y se pone a '1'.
- Si la expansión anterior provoca que la regla mutada subsuma a otra, se eliminan las reglas subsumidas (esto es, las más específicas).

D.3 Operación de Desplazamiento

En este caso, la regla no se hace más o menos específica, sino que la movemos en el espacio de búsqueda. El funcionamiento de esta operación es sencillo: se escoge un gen con alelo '1', se pone a '0', y se pone a '1' el gen inmediatamente anterior o inmediatamente posterior al gen mutado dentro de la variable correspondiente. La decisión de desplazar a izquierda o derecha se realiza de forma aleatoria, teniendo en cuenta que si se trata del primer gen de la variable, claramente sólo tendremos un movimiento posible, a la derecha. Por otro lado, en el caso de que sea el último, el único desplazamiento posible será a la izquierda.

Al igual que en la expansión, podemos encontrarnos con problemas de colisión. Seguiremos el mismo procedimiento para evitarlos, de forma que, en primer lugar, se analizan los movimientos válidos que no causan colisión, y se escoge aleatoriamente uno de ellos. En caso de que todos los movimientos causen colisión, se analiza si algún movimiento causa redundancia. Si es así, se escoge una de estas opciones y se opera como en la expansión.

E. Operador de Mutación del Consecuente

La probabilidad de mutación de cada individuo será ser proporcional al número total de variables de salida (número de reglas multiplicado por el número de variables de entrada) de la población. El primer paso, como siempre, es la elección de una variable a mutar, que se hará de forma aleatoria entre todas las variables de salida de la población.

Dado que en nuestros consecuentes no permitimos más de una etiqueta activa simultáneamente, el número de operaciones que se pueden realizar se restringe. De hecho, la única operación posible es el desplazamiento que, en este caso, equivale a incrementar o decrementar en una unidad el valor del consecuente. Si el gen ya toma el valor mínimo o máximo permitido (que estará entre 1 y el número de etiquetas de la variable de salida correspondiente), sólo se lleva a cabo la opción posible.

Algoritmo: Operador de Completitud.
Entrada: Toda la población de indiviudos.
Salida: Población modificada.
Precondiciones: -
Postcondiciones: La base de reglas codificada en cada uno de los individuos de la población cubre todos los ejemplos de entrenamiento.

Mientras (haya cromosomas)
1. Dado un cromosoma, calcular el *matching* de su base de reglas con el conjunto de ejemplos, para descubrir aquellos que han sido 'olvidados' y no son cubiertos por el conjunto de reglas.
2. Sobre el conjunto de ejemplos no cubiertos anterior, aplicar el algoritmo de Wang y Mendel para generar el conjunto de reglas mínimo (y de máxima especificidad) que los representa.
3. Añadir estas reglas al cromosoma.
Fin-Mientras

Fig. 3. Operador de Completitud

De nuevo, tras esta mutación es posible que se produzcan colisiones con el resto de reglas. Operamos igual que en la mutación del antecedente por expansión. Es decir, si se provoca redundancia se eliminan las reglas más específicas (las contenidas en otra más general), si hay inconsistencia separar el antecedente de las reglas haciendo que para cada gen, sólo una de las reglas tenga el valor '1'.

F. Operador de Completitud

Dado que el intercambio de reglas del cruce o la mutación del antecedente por contracción pueden provocar que una zona de datos con ejemplos quede sin cubrir, este operador tendrá por objetivo evitar este problema.

Un pseudo-código del funcionamiento del operador puede encontrarse en la figura 3.

G. Esquema de Selección y Funciones Objetivo

Consideramos un enfoque generacional con la estrategia de reemplazo elitista multiobjetivo de NSGA-II [35] y selección mediante torneo binario basado en la distancia de concentración en el espacio de las funciones objetivo.

Utilizaremos dos funciones objetivo para medir la calidad de los modelos generados: el *error cuadrático medio* (ECM) para mejorar la *precisión* y la *complejidad lingüística* para mejorar la *interpretabilidad*.

- **ECM**: Se define como

$$ECM(S) = \sum_{i=1}^{N} \frac{(S(x^i) - y^i)^2}{N},$$

donde S es el sistema difuso evaluado, N el tamaño del conjunto de datos y (x^i, y^i) el i-ésimo par entrada-salida del conjunto de datos. El objetivo será minimizar esta función.

- **Complejidad lingüística**: Se estimará mediante el número de reglas DNF. Dado que el algoritmo garantiza que no se generan reglas en aquellas zonas del espacio de entrada (definida en la hipermatriz de completitud) donde no existen ejemplos (gracias al operador de mutación del antecedente) ni se dejan zonas sin cubrir (gracias al operador de completitud), la complejidad de cada regla DNF, es decir, el número de reglas de tipo Mamdani equivalentes, no es relevante en nuestro caso pues bastará con perseguir el mínimo número de reglas DNF que cubre de forma óptima (ni por exceso ni por defecto) el espacio de entrada.

H. Mecanismo de inferencia

El mecanismo de inferencia que utilizaremos en el algoritmo será una combinación de operadores Max-Min, es decir, agregación mediante la T-conorma del máximo e implicación mediante la T-norma del mínimo, siguiendo el enfoque FATI (que primero construye el conjunto difuso agregado y luego lo defuzzifica). De esta forma conseguimos equivalencia numérica entre bases de reglas con equivalencia lingüística, lo que resulta de vital importancia al considerar reglas difusas tipo DNF. Finalmente, como mecanismo de defuzzificación empleamos el centro de gravedad.

V. Resultados

En este apartado analizamos el comportamiento de nuestro algoritmo, comprobando si han sido efectivas las medidas tomadas en el diseño de los operadores.

A. Problemas Utilizados

Para ello utilizaremos dos problemas reales en los que las variables de entrada y salida son continuas. Estos datos fueron extraídos con mediciones reales para su uso en redes de distribución eléctrica. En la tabla I se resumen las características más importantes de ambos problemas, con nombres *ELE1* y *ELE2*. El primero de ellos hace referencia al problema del cálculo de longitudes de líneas de baja tensión en zonas rurales y el segundo al cálculo del coste de mantenimiento de líneas de media tensión en zonas urbanas.

TABLA I
Problemas utilizados

Problema	*Núm. variables*	*Núm. ejemplos*	*Núm. etiquetas lingüísticas*
ELE1	2	495	7
ELE2	4	1056	5

Los conjuntos de datos originales han sido divididos de forma aleatoria en 5 subconjuntos diferentes. Los cuatro primeros han sido unidos en uno sólo y conforma el subconjunto de entrenamiento. El quinto, el de prueba. De esta forma, se realiza una validación cruzada de 5 particiones. Los datos y las particiones empleadas, junto con más información de interés sobre ellos, pueden encontrarse en la siguiente URL: http://decsai.ugr.es/∼casillas/fmlib/index.html.

A.1 Estimación de la Longitud de Tendido Eléctrico de Bajo Voltaje en Zonas Rurales

En ocasiones existe la necesidad de calcular la longitud de las líneas eléctricas que una compañía posee. Esta medida puede ser útil en diversos aspectos como la estimación de los costes de mantenimiento de la red. Las líneas de bajo voltaje son las que instalan en los pueblos y es muy caro medir su longitud, pues se trata de un proceso manual. Por ello, es necesario un método indirecto que consiga resultados precisos.

El problema consiste en encontrar un modelo que relacione la longitud de la línea con el número de habitantes y la media de las distancias desde el centro de la ciudad a los tres clientes más alejados del mismo. Este modelo podría ser usado para estimar la longitud total de la línea. Además, sería preferible que las soluciones fueran no sólo numéricamente precisas, sino también fácilmente interpretables por humanos.

En este problema, el conjunto de datos está compuesto de 495 ejemplos reales obtenidos de medidas directas en una serie de pueblos de Asturias.

A.2 Estimación del Coste de Mantenimiento del Tendido de Voltaje Medio en Ciudades

En este caso, el problema consiste en estimar los costes mínimos de mantenimiento del tendido eléctrico. El problema tiene cuatro variables de entrada: suma de las longitudes de las calles de la ciudad, área total de la ciudad, área ocupada por edificios y energía que se suministra a la ciudad.

El objetivo es generar un modelo que relacione los costes de mantenimiento con estas características de forma que puedan estudiarse instalaciones óptimas para el futuro.

B. Resultados

En la experimentación realizada, hemos incluido también los resultados obtenidos por otros algoritmos para poder observar el comportamiento de la propuesta frente a éstos:

- **Wang y Mendel** [2]: Es un algoritmo simple que, a pesar de que no obtiene resultados precisos, se considera ya una referencia tradicional en el área.
- **Thrift** [7]: Clásico método de aprendizaje de reglas difusas basado en algoritmos genéticos. Aprende reglas de tipo Mamdani.
- **COR-BWAS** [36]: Algoritmo de aprendizaje basado en Colonias de Hormigas, con un gran rendimiento en interpretabilidad y precisión. Hemos considerado la versión que no elimina reglas ya que este algoritmo no garantiza cubrimiento total y sus resultados nos serían comparables con los de nuestra propuesta.
- **Pittsburgh** [37]: Algoritmo evolutivo estilo Pittsburgh que también aprende reglas difusas tipo DNF. Se sigue un enfoque generacional con reemplazo directo (los descendientes sustituyen a los padres). Considera un esquema de elitismo que asegura que el mejor cromosoma de la generación anterior sobrevive en la nueva. El *fitness* es el error cuadrático medio. La población se inicializa aleatoriamente y se emplea torneo binario como mecanismo de selección. El esquema de codificación es el mismo que el empleado en este trabajo.

En la tabla II se resumen los valores de los parámetros considerados en nuestro algoritmo, que hemos nombrado Pitts-DNF.

TABLA II
PARÁMETROS DEL ALGORITMO PITTS-DNF

Parámetro	*Valor*
Número máximo de iteraciones	300
Tamaño de la población	60
Probabilidad de cruce	0,7
Probabilidad de mutación	0,1

Las tablas III y IV resumen los resultados obtenidos. En ellas, #R representa el número de reglas y ECM_{entr} y ECM_{prue} los valores del error cuadrático medio en los conjuntos de entrenamiento y prueba, respectivamente, redondeados al valor entero más cercano. $\bar{x}$ representa la media aritmética de cada valor y σ la desviación típica. El mejor resultado para cada problema está resaltado en negrita.

TABLA III
RESULTADOS OBTENIDOS EN EL PROBLEMA ELE1

	$\#R$		ECM_{entr}		ECM_{prue}	
Método	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
WM	22,0	1,4	423.466	8.069	455.262	19.943
Thrift	48,3	0,8	**333.062**	2.804	419.408	34.806
COR-BWAS	22,0	4,6	354.304	7.065	417.142	9.823
Pittsburgh	15,4	3,4	374.595	22.114	460.404	67.597
Pitts-DNF	**15,2**	1,4	335.666	3.396	**397.774**	11.305

TABLA IV
RESULTADOS OBTENIDOS EN EL PROBLEMA ELE2

	$\#R$		ECM_{entr}		ECM_{prue}	
Método	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
WM	65,0	0,0	112.270	1.498	112.718	4.685
Thrift	565,3	1,8	**62.456**	1.108	75.158	7.279
COR-BWAS	65,0	4,6	102.664	1.080	102.740	4.321
Pittsburgh	270,0	29,9	174.399	22.237	238.413	47.063
Pitts-DNF	**50,4**	2,33	67.468	3.396	**74.508**	5.292

En las figuras 4 y 5 se muestran los frentes Pareto obtenidos por el algoritmo propuesto.

B.1 Análisis de los Resultados

A la vista de los resultados, creemos que el algoritmo propuesto muestra un buen comportamiento. Obtiene grados de precisión muy buenos en ambos problemas con un número sensiblemente menor de reglas que el resto de algoritmos en el primer problema. Este resultado, sin duda, mejora notablemente la interpretabilidad de la solución, uno de los objetivos de este trabajo.

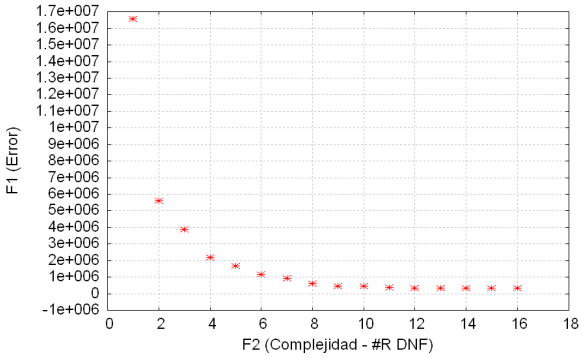

Fig. 4. Frente Pareto obtenido en el problema ELE1

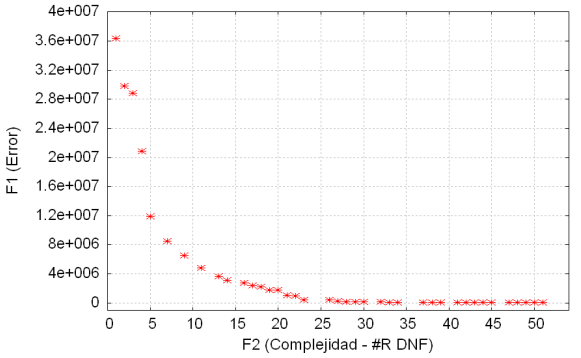

Fig. 5. Frente Pareto obtenido en el problema ELE2

En el primer problema la diferencia en el ECM con el método ganador, *Thrift*, es de sólo el 0,7 %. Sin embargo, este resultado se obtiene con un número de reglas muy inferior, 15,2 frente a 48,3. Por otra parte, en el segundo problema la situación es similar: en números absolutos el algoritmo *Thrift* consigue un ECM_{entr} de 62.456 (frente a los 67.468 de nuestro método), pero con una media exageradamente alta de 563,5 reglas, resultado poco útil en la práctica.

También desde el punto de vista de la interpretabilidad los resultados son buenos. De hecho, obtenemos los mejores resultados de todos los algoritmos comparados. En el primer problema mejoramos ligeramente a la siguiente mejor solución, *Pittsburgh*, y en un 217 % a la peor, *Thrift*. También en el segundo problema el algoritmo se comporta ligeramente mejor que el segundo mejor método (COR-BWAS), generando 50,4 reglas frente a 65 (una mejora del 20 %).

Finalmente, también en lo relativo a la robustez del algoritmo se obtienen resultados satisfactorios. La desviación típica muestra valores sensiblemente inferiores a la de los otros algoritmos en la mayoría de los casos o sólo un poco superiores (un máximo de un 15 %) en únicamente dos casos (desviación del ECM_{prue} respecto al algoritmo *COR-BWAS*).

VI. Conclusiones

En este trabajo hemos presentado un algoritmo difuso evolutivo enfocado a obtener una alta interpretabilidad en sus resultados sin renunciar por ello al grado de aproximación. Para ello, se han redefinido los operadores clásicos de cruce y mutación y se ha añadido uno nuevo, llamado de completitud, cuya misión es garantizar, en todo momento, que no quedan ejemplos sin cubrir por alguna regla.

A la vista de los resultados, nuestro algoritmo parece comportarse bien resolviendo dos problemas reales. Comparado con otros métodos de aprendizaje, obtenemos modelos más precisos en prácticamente todos los casos y manteniendo una interpretabilidad muy buena. Podemos concluir, por tanto, que el algoritmo parece manejar bien el balance entre interpretabilidad y precisión, obteniendo modelos lingüísticos compactos y aproximados.

Referencias

[1] N. Lavrac, B. Cestnik, D. Gamberger, and P. Flach, "Decision support through subgroup discovery: three case studies and the lessons learned", *Machine Learning*, vol. 57, núm. 1-2, págs. 115–143, 2004.

[2] L.-X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, núm. 6, págs. 1414–1427, 1992.

[3] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data", *Fuzzy Sets and Systems*, vol. 86, núm. 3, págs. 251–270, 1997.

[4] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of neuro-fuzzy systems*, John Wiley & Sons, New York, NY, EE.UU., 1997.

[5] R. Fullér, *Introduction to neuro-fuzzy systems*, Physica-Verlag, Heidelberg, Alemania, 2000.

[6] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*, World Scientific, Singapore, 2001.

[7] P. Thrift, "Fuzzy logic synthesis with genetic algorithms", en *Proceedings of the 4th International Conference on Genetic Algorithms*, R.K. Belew and L.B. Booker, Eds., San Mateo, CA, EE.UU., 1991, págs. 509–513, Morgan Kaufmann Publishers.

[8] C.L. Karr, "Genetic algorithms for fuzzy controllers", *AI Expert*, vol. 6, núm. 2, págs. 26–33, 1991.

[9] M. Valenzuela-Rendón, "The fuzzy classifier system: a classifier system for continuously varying variables", en *Proceedings of the 4th International Conference on Genetic Algorithms*, San Mateo, CA, EE.UU., 1991, págs. 346–353, Morgan Kaufmann Publishers.

[10] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds., *Interpretability issues in fuzzy modeling*, Springer, Heidelberg, Alemania, 2003.

[11] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds., *Accuracy improvements in linguistic fuzzy modeling*, Springer, Heidelberg, Alemania, 2003.

[12] F. Herrera, M. Lozano, and J.L. Verdegay, "Tuning fuzzy controllers by genetic algorithms", *International Journal of Approximate Reasoning*, vol. 12, págs. 299–315, 1995.

[13] O. Cordón and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases from examples", *International Journal of Approximate Reasoning*, vol. 17, núm. 4, págs. 369–407, 1997.

[14] O. Cordón, M.J. del Jesus, and F. Herrera, "Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods", *International Journal of Intelligent Systems*, vol. 13, págs. 1025–1053, 1998.

[15] Y. Jin, W. von Seelen, and B. Sendhoff, "On generating FC^3 fuzzy rule systems from data using evolution strategies", *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 29, núm. 4, págs. 829–845, 1999.

[16] D. Nauck and R. Kruse, "Neuro-fuzzy systems for function approximaton", *Fuzzy Sets and Systems*, vol. 101, núm. 2, págs. 261–271, 1999.

[17] B.-D. Liu, C.-Y. Chen, and J.-Y. Tsao, "Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms", *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 31, núm. 1, págs. 32–53, 2001.

[18] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems", *IEEE Transactions on Fuzzy Systems*, vol. 7, núm. 2, págs. 109–119, 1999.

[19] W. Pedrycz, R.R. Gudwin, and F.A.C. Gomide, "Nonlinear context adaptation in the calibration of fuzzy sets", *Fuzzy Sets and Systems*, vol. 88, núm. 1, págs. 91–97, 1997.

[20] J. Espinosa and J. Vandewalle, "Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm", *IEEE Transactions on Fuzzy Systems*, vol. 8, núm. 5, págs. 591–600, 2000.

[21] A. González and R. Pérez, "A study about the inclusion of linguistic hedges in a fuzzy rule learning algorithm", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 7, núm. 3, págs. 257–266, 1999.

[22] N.R. Pal and K. Pal, "Handling of inconsistent rules with an extended model of fuzzy reasoning", *Journal of Intelligent and Fuzzy Systems*, vol. 7, págs. 55–73, 1999.

[23] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms", *IEEE Transactions on Fuzzy Systems*, vol. 3, núm. 3, págs. 260–270, 1995.

[24] T.-P. Hong and C.-Y. Lee, "Effect of merging order on performance of fuzzy induction", *Intelligent Data Analysis*, vol. 3, núm. 2, págs. 139–151, 1999.

[25] T.-P. Hong and J.-B. Chen, "Finding relevant attributes and membership functions", *Fuzzy Sets and Systems*, vol. 103, núm. 3, págs. 389–404, 1999.

[26] H.-M. Lee, C.-M. Chen, J.-M. Chen, and Y.-L. Jou, "An efficient fuzzy classifier with feature selection based on fuzzy entropy", *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 31, núm. 3, págs. 426–432, 2001.

[27] J. Yen and L. Wang, "An SVD-based fuzzy model reduction strategy", en *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, New Orleans, LA, EE.UU., 1996, págs. 835–841.

[28] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods", *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 29, núm. 1, págs. 13–24, 1999.

[29] Y. Yam, P. Baranyi, and C.-T. Yang, "Reduction of fuzzy rule base via singular value decomposition", *IEEE Transactions on Fuzzy Systems*, vol. 7, núm. 2, págs. 120–132, 1999.

[30] T. Taniguchi, K. Tanaka, H. Ohtake, and H.O. Wang, "Model construction, rule reduction, and robust compensation for generalized form of Takagi-Sugeno fuzzy systems", *IEEE Transactions on Fuzzy Systems*, vol. 9, núm. 4, págs. 525–538, 2001.

[31] M. Setnes, R. Babuška, U. Kaymak, and H.R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification", *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 28, núm. 3, págs. 376–386, 1998.

[32] M. Setnes and J.A. Roubos, "GA-fuzzy modeling and classification: complexity and performance", *IEEE Transactions on Fuzzy Systems*, vol. 8, núm. 5, págs. 509–522, 2000.

[33] J.A. Roubos and M. Setnes, "Compact and transparent fuzzy models and classifiers through iterative complexity reduction", *IEEE Transactions on Fuzzy Systems*, vol. 9, núm. 4, págs. 516–524, 2001.

[34] M.L. Hadjili and V. Wertz, "Takagi-Sugeno fuzzy modeling incorporating input variables selection", *IEEE Transactions on Fuzzy Systems*, vol. 10, núm. 6, págs. 728–742, 2002.

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarevian, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, núm. 2, págs. 182–197, 2002.

[36] J. Casillas, O. Cordón, I. Fernández de Viana, and F. Herrera, "Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm", *International Journal of Intelligent Systems*, vol. 20, págs. 433–452, 2005.

[37] J. Casillas, O. Delgado, and F.J. Martínez-López, "Predictive knowledge discovery by multiobjective genetic fuzzy systems for estimating consumer behavior models", en *Proceedings of the 4th International Conference in Fuzzy Logic and Technology*, Barcelona, España, 2005, págs. 272–278.