XV CONGRESO ESPAÑOL SOBRE TECNOLOGÍAS Y LÓGICA FUZZY

ESTYLF 2010 HUELVA

DEL 3 AL 5 DE FEBRERO · HOTEL BARCELÓ CONVENTION CENTER PUNTA UMBRÍA · HUELVA

ORGANIZAN

PATROCINAN

Gobierno de España · Ministerio de Ciencia e Innovación · Junta de Andalucía · consejosocial Universidad de Huelva · Fundación Cajasol

Universidad de Huelva · EUSFLAT · European Society for Fuzzy Logic and Technology

ORGANIZADORES:

Universidad de Huelva
European Society for Fuzzy Logic and Technology

EDITORES:

Antonio Peregrín Rubio
Jesús Alcalá Fernández
Francisco José Moreno Velo
Francisco Alfredo Márquez Hernández

ENTIDADES COLABORADORAS:

Ministerio de Ciencia e Innovación. Gobierno de España.
Consejería de Innovación, Ciencia y Empresa. Junta de Andalucía.
Vicerrectorado de Extensión Universitaria. Universidad de Huelva.
Consejo Social de la Universidad de Huelva.
Fundación Cajasol.

ISBN: 978-84-92944-02-6

Sesión Especial: Soft Computing en minería y resumen de datos

Organizadores:

Daniel Sánchez

María José Martín Bautista

1 .- Speeding Up Evolutionary Learning Algorithms using GPUs.....229-234
(A. Cano, A. Zafra, S. Ventura)

2 .- Determinando Automáticamente los Dominios de Competencia de un Sistema de Clasificación Basado en Reglas Difusas: Un Caso de Estudio con FH-GBML.....235-240
(J. Luengo, F. Herrera)

3 .- Minería de reglas de asociación con programación genética gramatical 241-248
(J. M. Luna, J. L. Olmo, J. R. Romero, S. Ventura)

4 .- Un Modelo de Clasificación basado en Reglas de Asociación Difusas para Problemas de Alta Dimensionalidad y Selección Evolutiva de Reglas.....249-254
(J. Alcalá-Fdez, R. Alcalá, F. Herrera)

5 .- Linguistic summarization of data with probabilistic fuzzy quantifiers.....255-260
(F. Díaz-Hermida, A. Bugarín)

6 .- Construcción de conjuntos de reglas difusas para la clasificación de objetos.....261-266
(C.J. Solana-Ciprés, J. Albusac, J.J. Castro-Schez, J. Moreno-García, L. Rodríguez-Benítez)

7 .- Linguistic comparison of time series: a fuzzy approach.....267-272
(R. Castillo-Ortega, N. Marín, D. Sánchez)

Sesión Especial: Funciones de agregación

Organizadores:

Tomasa Calvo Sánchez

Humberto Bustince Sola

Javier Montero de Juan

1 .- On the satisfaction of the functional equation A(x,N(x))=c 273-278
(A. Pradera)

2 .- Una generalización de la migratividad.....279-284
(H. Bustince, J. Fernández, B. De Baets, J. Montero, R. Mesiar)

3 .- Using heavy aggregations in a unified model between the weighted average and the OWA operator 285-290
(J. M. Merigó, M. Casanovas)

4 .- Decision making with distance measures, weighted averages and induced OWA operators 291-296
(J. M. Merigó, A. M. Gil-Lafuente)

Minería de reglas de asociación con programación genética gramatical

José María Luna Juan Luis Olmo José Raúl Romero Sebastián Ventura

Departamento de Informática y Análisis Numérico, Universidad de Córdoba
{i32luarj, juanluisolmo, jrromero, sventura}@uco.es

Resumen

En este trabajo presentamos un algoritmo de programación genética gramatical G3P (*Grammar Guided Genetic Programming*) para la extracción de reglas de asociación sobre conjuntos de datos. Para ello, proponemos dos versiones para la extracción de reglas de asociación: la primera de ellas es un algoritmo generacional simple, mientras que la segunda hace uso de una población auxiliar en la que se insertan individuos que sobrepasen una determinada calidad. Además, comparamos nuestra propuesta con el algoritmo *Apriori* mostrando cómo nuestro algoritmo obtiene unos resultados medios mejores con menor número de reglas.

Palabras Clave: Algoritmos Evolutivos, Programación Genética Gramatical, Minería de datos.

1 INTRODUCCIÓN

Las técnicas de minería de patrones están diseñadas para extraer datos de interés y útiles de las bases de datos. La mayoría de estas técnicas tienen su base en los conjuntos de patrones frecuentes, si bien en ocasiones resulta interesante encontrar asociaciones excepcionales mediante el análisis de los conjuntos de patrones no frecuentes [1]. Este análisis cobra especial relevancia en sectores como la medicina o la biología, así como en ámbitos comerciales, de finanzas o en transacciones económicas a través de la web. Sin embargo, para la mayoría de dominios, los patrones que aparecen con frecuencia son los más útiles y, por tanto, objeto de estudio.

El objetivo de la minería de asociación es la obtención de *reglas útiles* (o de interés) desde las que sea posible derivar nuevo conocimiento. Así pues, el papel de los sistemas de minería de asociación en este proceso es facilitar el descubrimiento de patrones y permitir la presentación de estas reglas o inferencias para su posterior interpretación por el usuario y así determinar su utilidad.

Uno de los principales algoritmos basados en patrones frecuentes de bases de datos fue el propuesto por *R. Agrawal and R. Srikant* [2] en 1994 y que ha servido como punto de partida de muchas investigaciones [3, 4, 12]. Este algoritmo, denominado *Apriori*, fue el primero utilizado satisfactoriamente para la extracción de reglas de asociación. En él, se extraen los conjuntos de elementos frecuentes y, a partir de estos, se obtiene reglas de asociación. Para reducir el coste computacional, el algoritmo *Apriori* mantiene que si un conjunto de elementos es frecuente, entonces todos sus subconjuntos son frecuentes. Si un conjunto no es frecuente, sus superconjuntos no serán frecuentes, por lo que se puede reducir el coste computacional eliminando dichas producciones. Con posterioridad al algoritmo *Apriori*, se presentaron distintas alternativas para la extracción de patrones frecuentes, como *Eclat* [14], *FP-Growth* [8] y *TM* [11].

Existen investigaciones que proponen algoritmos evolutivos [5] para la extracción de reglas de asociación de datos cuantitativos [9, 10], considerándose este tipo de algoritmos y, particularmente, los algoritmos genéticos [6], como una de las técnicas de búsqueda de mayor éxito para los problemas complejos, demostrando ser una técnica importante para el aprendizaje y la extracción de conocimiento. Los algoritmos genéticos son métodos de búsqueda robustos y flexibles, pues un mismo algoritmo genético puede ser usado con diferentes representaciones. Además, los algoritmos genéticos, permiten obtener soluciones viables dentro de unos límites de tiempo. Es por esto que, tanto los algoritmos evolutivos como los algoritmos genéticos, han experimentado un creciente interés en el campo de la minería de datos.

Regla = Antecedente, Consecuente ;
Antecedente = Comparación | "Y", Comparación, Antecedente ;
Consecuente = Comparación ;
Comparación = Comparador Categórico, Atributo Categórico ;
Comparador Categórico = "! =" | "=" ;
Atributo Categórico = "nombre", "valor" ;

Figura 1: Gramática de contexto libre.

En el presente trabajo proponemos un algoritmo de programación genética gramatical G3P (*Grammar Guided Genetic Programming*) para la extracción de reglas de asociación. G3P es una extensión de la programación genética que permite la obtención de programas (en este caso, reglas) válidos al definir los individuos utilizando una gramática de contexto libre, con la que estableceremos formalmente las restricciones sintácticas del problema a resolver. Por tanto, cada individuo generado por G3P es un árbol de derivación que genera y representa una solución perteneciente al lenguaje definido por la gramática. Proponemos dos versiones del algoritmo G3P: la primera de ellas es un algoritmo generacional simple, mientras que la segunda hace uso de una población auxiliar en la que se guardan individuos que sobrepasan una determinada calidad. Además, realizamos una primera comparación entre las versiones de nuestro algoritmo para determinar la que mejores resultados nos proporcione, para luego comparar nuestra propuesta con *Apriori*. Los resultados mostrarán que nuestro algoritmo obtiene menor número de reglas de asociación que *Apriori*, lo que implica una mayor eficiencia, y unos resultados medios mejores que *Apriori*.

El presente artículo está organizado como sigue: la Sección 2 describe el modelo desarrollado y sus principales características. En la Sección 3 se describen las bases de datos utilizadas para realizar la experimentación. Posteriormente, en la Sección 4 se describe tanto la ejecución como los resultados obtenidos. Finalmente, mostraremos las conclusiones alcanzadas y comentaremos las líneas de trabajo futuro que pretendemos abordar.

2 DESCRIPCIÓN DEL MODELO

En esta sección presentaremos nuestro modelo, así como sus características más relevantes: cómo se representan los individuos, los operadores genéticos utilizados, así como los evaluadores y mecanismos de control de la población auxiliar.

2.1 REPRESENTACIÓN DE LOS INDIVIDUOS

Cada individuo está formado por dos componentes diferenciados: (*a*) un *genotipo*, que se codifica mediante G3P con una estructura de árbol con profundidad limitada para evitar las infinitas derivaciones posibles, y (*b*) un *fenotipo*, que representa la regla completa formada por un antecedente y un consecuente. El antecedente de cada regla se integra por una serie de condiciones relativas a los valores de ciertos atributos que han de ser todas satisfechas. Por el contrario, el consecuente está compuesto por una sola condición. La Figura 1 muestra la gramática de contexto libre que representa las reglas codificadas por los individuos de la población. Hay que indicar que, el símbolo no terminal 'nombre' de la gramática vendrá determinado por los atributos del conjunto de datos utilizado en cada momento. Además, para cada atributo de la gramática, se asignará un valor determinado por el rango de valores del mismo atributo en el *dataset*.

2.2 OPERADORES GENÉTICOS

Los nuevos individuos de cada generación se obtienen por medio de dos operadores:

- *Cruce*. Realiza un intercambio de los subárboles de derivación de dos padres a partir de dos puntos compatibles seleccionados aleatoriamente en cada uno de ellos. Dos nodos son compatibles si pertenecen al mismo símbolo no terminal, evitando así producir un individuo erróneo según la gramática definida.
- *Mutación*. Selecciona aleatoriamente un nodo del árbol y, según el tipo de nodo, se actúa. Si el nodo seleccionado es un elemento terminal, se cambia el valor de dicho elemento terminal de forma aleatoria. Si, por el contrario, el nodo seleccionado es un elemento no terminal, se realiza una nueva derivación a partir de dicho nodo. Hay que tener en cuenta que si el elemento seleccionado es un elemento no terminal, y debido a que se podría realizar una derivación diferente a la realizada en una primera instancia, el número de derivaciones necesarias para llegar a un símbolo termi-

nal podría variar, por lo que hay que controlar que no se sobrepase el tamaño máximo de derivación impuesto en el algoritmo.

2.3 EVALUACIÓN

Antes de llevar a cabo la evaluación de un individuo, hemos de llevar a cabo su decodificación, es decir, obtener la regla de asociación que se corresponde con el árbol sintáctico de su genotipo.

El proceso de decodificación consiste en construir una expresión recorriendo el árbol sintáctico en profundidad, y eliminando los símbolos no terminales que aparecen en el mismo. Asimismo, se comprueba que los individuos no utilicen el mismo atributo en el antecedente y consecuente de la regla.

El proceso de evaluación de los individuos se realiza obteniendo el valor de su función de evaluación. Éste será el *soporte*, que se define como la proporción de transacciones en una base de datos D que contiene el conjunto de elementos X:

$$fitness(individuo) = sop(X) = \frac{|X|}{|D|} \quad (1)$$

Otra de las heurísticas que vamos a utilizar es la *confianza* de la regla obtenida mediante el proceso de decodificación. Ésta se define como un estimador de P(Y/X), la probabilidad de encontrar la parte derecha de una regla condicionada a que se encuentre también la parte izquierda:

$$conf(X \Rightarrow Y) = \frac{sop(X \bigcup Y)}{sop(X)} \quad (2)$$

Si el *fenotipo* del individuo es correcto, se procederá a su evaluación respecto al conjunto de instancias del *dataset*. Dicha evaluación consiste en el cálculo del *soporte*, que se realiza del siguiente modo: para cada atributo del antecedente, se compara con su homónimo de la instancia y se realiza un Y lógico con el valor lógico *verdadero* si es correcto y con el valor lógico *falso* si no lo es. Con el atributo del consecuente actuamos de forma análoga incrementando una variable si el antecedente es correcto y otra si tanto el antecedente como el consecuente son correctos.

2.4 ALGORITMO EVOLUTIVO

En esta sección describiremos las dos versiones del algoritmo evolutivo que hemos desarrollado para resolver el problema de asociación.

2.4.1 Primera versión: algoritmo generacional

Esta primera solución comienza obteniendo la población mediante la generación aleatoria de individuos a partir de la gramática de contexto libre definida en la sección 2.1 y cumpliendo el número máximo de derivaciones posibles. Para llevar a cabo la derivación de los diferentes símbolos que aparecen en la gramática se utiliza el concepto de cardinalidad. El número de cadenas que pueden ser generadas a partir de una gramática de contexto libre es infinito. Sin embargo, podemos agrupar las cadenas en conjuntos generados en d derivaciones, denominando cardinalidad al número de elementos del conjunto generado. Cada símbolo no terminal, tendrá una cardinalidad en base al conjunto generado en d derivaciones. Si un símbolo no terminal se puede derivar de varias maneras, la cardinalidad de dicho símbolo no terminal vendrá determinada por la suma de las cardinalidades de cada una de las derivaciones posibles a partir de dicho símbolo. Si una derivación posee más de un símbolo no terminal, la cardinalidad del conjunto formado por los símbolos vendrá determinada por el producto de las cardinalidades de cada símbolo no terminal de la derivación.

Cada individuo se genera a partir del símbolo inicial de la gramática aplicando reglas de producción de manera aleatoria hasta conseguir una cadena válida. El número de derivaciones vendrá determinado por el número máximo de derivaciones establecido en los parámetros de configuración del algoritmo. Para generar un individuo, el algoritmo se inicia con el símbolo inicial de la gramática y con el número máximo de derivaciones. A partir de este símbolo, se buscan las producciones que se pueden obtener en base al número de derivaciones y se elige una de manera aleatoria, teniendo en cuenta que cuanto mayor es la cardinalidad de un símbolo, mayor es la probabilidad de derivarse a partir de dicho símbolo. El algoritmo continúa de forma recursiva por cada símbolo no terminal, reduciendo, en cada iteración, el número máximo de derivaciones.

En base a la población se seleccionan individuos por medio de un torneo binario. Se seleccionan individuos que actuarán de padres para el cruce, que se realizará en base su probabilidad, siendo más probable que el cruce se realice cuanto mayor sea dicha probabilidad. El siguiente paso es realizar la mutación de los individuos seleccionados. Al igual que el cruce, la mutación dependerá de su probabilidad. Mediante el cruce y la mutación, obtenemos nuevos individuos que pasarán a formar parte de la nueva población mediante reemplazo directo. En la Figura 2 se muestra el pseudocódigo de esta primera versión.

1 Algoritmo generacional.
Entrada: $max\_generaciones, N$
Salida: P
1: $P_0 \leftarrow random(N)$
2: $A_0 \leftarrow \emptyset$
3: **mientras** $num\_generaciones < max\_generaciones$ **hacer**
4: Seleccionar padres $(P_t \cup A_t)$
5: Cruce (P')
6: Mutación (P')
7: $P \leftarrow P'$
8: Actualizar población auxiliar (A_t)
9: $num\_generaciones++$
10: **fin mientras**

Figura 2: Pseudocódigo del algoritmo generacional.

2.4.2 Segunda versión: uso de una población auxiliar

Proponemos una mejora del algoritmo generacional simple mediante una población auxiliar en la que guardaremos los individuos que cumplan una determinada condición. Esta propuesta comenzará, al igual que la primera versión, mediante la generación aleatoria de individuos a partir de la gramática de contexto libre definida y cumpliendo el número máximo de derivaciones posibles. En la generación inicial, la población auxiliar estará vacía.

Los padres o individuos seleccionados para el cruce se seleccionan de la unión de la población actual y de la población auxiliar. Mediante el cruce, se obtienen nuevos individuos que pasarán a formar la nueva población. El proceso continúa de la misma forma que la primera versión propuesta. Una vez que hemos obtenido la nueva población mediante el cruce y la mutación, pasaremos a actualizar la población auxiliar. Para actualizar dicha población, se realizará la unión de la población auxiliar de la generación anterior y la población actual. A continuación, se procede a la eliminación de los individuos que estén repetidos y con el conjunto resultante, se actúa según las propuestas: seleccionando los mejores individuos en base a su función de evaluación, seleccionando aquellos que sobrepasen un umbral de *soporte*, seleccionando aquellos que sobrepasen un umbral de *confianza* o seleccionando aquellos que sobrepasen un determinado umbral de *soporte* y *confianza*. En las figuras 3 y 4 se muestra el pseudocódigo de esta segunda versión.

2 Algoritmo con población auxiliar.
Entrada: $max\_generaciones, N$
Salida: A
1: $P_0 \leftarrow random(N)$
2: $A_0 \leftarrow \emptyset$
3: **mientras** $num\_generaciones < max\_generaciones$ **hacer**
4: Seleccionar padres $(P_t \cup A_t)$
5: Cruce (P')
6: Mutación (P')
7: $P \leftarrow P'$
8: Actualizar población auxiliar (A_t)
9: $num\_generaciones++$
10: **fin mientras**

Figura 3: Pseudocódigo del algoritmo con población auxiliar.

3 Actualizar población auxiliar.
Entrada: A
Salida: A
1: A' $\leftarrow$ P' + A_t
2: Ordenar (A')
3: Eliminar duplicados (A')
4: $A_t \leftarrow$ Umbral (A')

Figura 4: Pseudocódigo de la actualización de la población auxiliar.

3 EXPERIMENTACIÓN

La experimentación se ha realizado con 5 conjuntos de datos[1](véase Tabla 1). Para poder realizar comparaciones con el algoritmo *Apriori*, los datos numéricos han sido preprocesados mediante la técnica de discretización *equal-width binning*[2] [7] en 10 rangos.

Los *datasets WDBC*, *WPBC* y *WDatabaseBC*, se corresponden, respectivamente, con los conjuntos de datos: *Wisconsin Diagnostic Breast Cancer*, *Wisconsin Prognostic Breast Cancer* y *Wisconsin Breast Cancer Database*.

El algoritmo G3P presentado se ha desarrollado usando el software JCLEC (*Java Class Library for Evolutionary Computation*) [13], que sirve como *framework* para el desarrollo de aplicaciones de computación evolutiva. Se ha utilizado para la comparación la implementación de *Apriori* existente en el software WEKA [15].

Los parámetros de configuración empleados son: 100 individuos, 100 generaciones, probabilidad de cruce del 70%, probabilidad de mutación del 10% y número máximo de derivaciones de 24. La propuesta de utilizar una población auxiliar se realiza sobre un tamaño de población auxiliar de 60 individuos. Además, se tendrán en cuenta aquellos individuos por encima de los umbrales de *soporte* 0.5 y *confianza* 0.8. El al-

[1] *UCI Machine Learning Repository.*
`http://archive.ics.uci.edu/ml/datasets.html`

Mining Association Rules from Data Sets.
`http://www.cs.iastate.edu/~neeraj/cs572.html`

[2] Este método consiste en dividir en rango de valores en intervalos de tamaño constante.

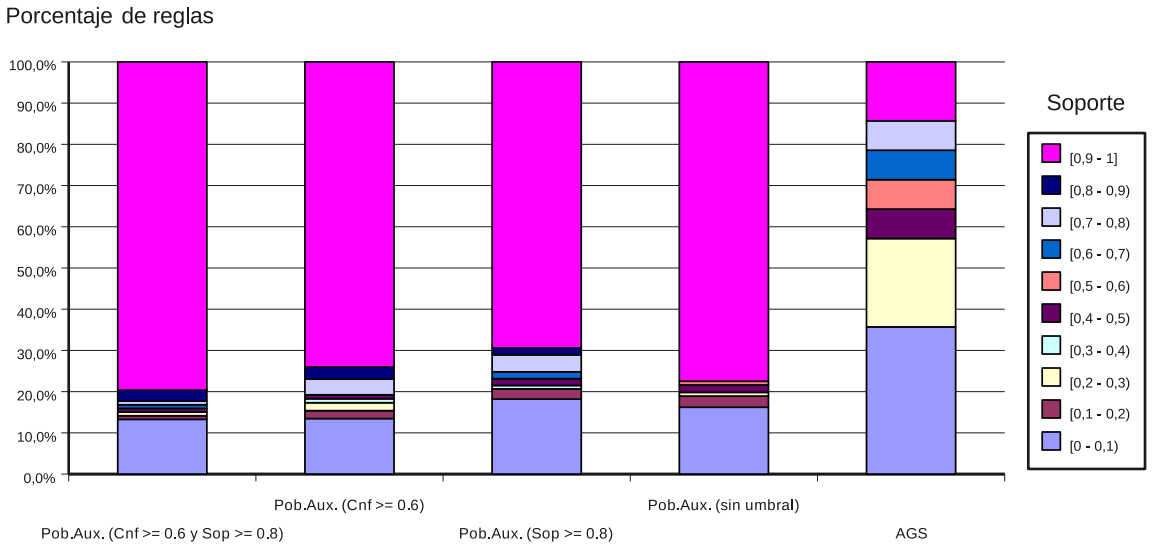

Figura 5: Comparativa entre formas de actualizar la población auxiliar.

Tabla 1: Conjuntos de datos utilizados.

Nombre	Instancias	Atributos	Valores
Soybean	683	36	116
Segment	1500	20	79
WDBC	569	31	302
WPBC	194	34	333
WDatabaseBC	683	11	102

goritmo Apriori, tendrá como umbrales de *soporte* y *confianza* los anteriormente indicados, y como número máximo de reglas a obtener el valor 300000.

4 RESULTADOS

En esta sección mostraremos los resultados obtenidos por las versiones propuestas, realizando una comparación entre las versiones y entre la mejor propuesta y el algoritmo *Apriori*.

4.1 COMPARATIVA ENTRE MODOS DE ACTUALIZAR LA POBLACIÓN AUXILIAR

En este apartado, realizaremos una comparativa entre el algoritmo generacional simple sin población auxiliar y las distintas propuestas de asignación de individuos a la población auxiliar: (a) Población auxiliar con individuos por encima de un umbral de *soporte* de 0.6 y *confianza* 0.8, (b) población auxiliar con individuos por encima de 0.6 como umbral de *confianza*, (c) población auxiliar con individuos por encima de 0.8 como umbral de *soporte*, (d) población auxiliar con los mejores individuos. Para realizar dicha comparativa, mostraremos el porcentaje de reglas obtenidas para los diferentes intervalos de *soporte*.

En vista de los resultados mostrados en la Figura 5, comprobamos que utilizando una población auxiliar en la que se vayan guardando los mejores individuos que se han ido obteniendo a lo largo de las generaciones, se obtienen mejores resultados que si hacemos uso del algoritmo generacional simple sin población auxiliar. Debido a que estamos buscando *items* frecuentes, entendemos por mejores resultados aquellos que poseen un *soporte* alto. Haciendo uso del algoritmo generacional simple, sólo cerca del 15% de las reglas obtenidas tienen un *soporte* entre el 90% y el 100%. Además, con esta versión del algoritmo, el número de individuos con *soporte* por debajo del 10% es muy elevado.

Descartando esta versión del algoritmo, nos centramos en los diferentes tipos de actualización de la población auxiliar. Comprobamos que la actualización de la población auxiliar con aquellos individuos cuyo *soporte* y *confianza* esté por encima de 0.6 y 0.8 respectivamente, son los que mejores resultados obtiene. Cerca del 80% de las reglas que se obtienen con este tipo de actualización de la población auxiliar tienen un soporte por encima del 90%. Además, el porcentaje de reglas cuyo soporte esté por debajo del 10% es muy bajo en esta versión.

4.2 COMPARATIVA CON *APRIORI*

En este apartado, realizaremos una comparativa, para cada uno de los *datasets*, entre los resultados obtenidos

con el algoritmo *Apriori* y la mejor propuesta obtenida (uso de una población auxiliar en la que se incluirán aquellos individuos cuyo *soporte* y *confianza* esté por encima de 0.6 y 0.8 respectivamente). Los resultados obtenidos por el algoritmo *Apriori* y por nuestra propuesta son los que se muestran en las tablas 2 y 3 respectivamente, donde *n_reglas* representa el número de reglas obtenido cuyo *soporte* es mayor de 0.5 y *confianza* mayor de 0.8; *Sop_global* representa el *soporte* del conjunto de reglas obtenido; *Sop_med* representa la media del *soporte* del conjunto de reglas; y *Conf_med* la media de la *confianza* del conjunto de reglas obtenido.

Tabla 2: Resultados obtenidos por *Apriori*.

Nombre	*n_reglas*	*Sop_global*	*Sop_med*	*Conf_med*
Soybean	300000	100%	0.6409	0.9280
Segment	169945	100%	0.5479	0.9615
WDBC	75	84.71%	0.5458	0.8957
WPBC	1	59.27%	0.5927	0.8273
WDatabaseBC	237	88.29%	0.5322	0.9204

Tabla 3: Resultados obtenidos por la mejor propuesta.

Nombre	*n_reglas*	*Sop_global*	*Sop_med*	*Conf_med*
Soybean	36	100%	0.9171	0.9795
Segment	33	100%	0.9955	0.9993
WDBC	54	100%	0.9997	0.9998
WPBC	51	100%	0.9892	0.9929
WDatabaseBC	50	100%	0.9884	0.9999

Analizando los resultados presentados podemos comprobar cómo, con mayor número de reglas obtenidas, *Apriori* obtiene un *soporte* global menor que el obtenido por nuestra propuesta. Además, nuestra propuesta obtiene un *soporte* y *confianza* promedios mejores que los del algoritmo *Apriori*.

5 CONCLUSIONES Y TRABAJOS FUTUROS

El presente artículo describe un primer intento de aplicar técnicas G3P para descubrir reglas de asociación en un conjunto de datos. Se han utilizado diversas formas de llevar a cabo el algoritmo: algoritmo generacional simple sin hacer uso de ninguna población auxiliar, y mejora del algoritmo generacional simple haciendo uso de una población auxiliar (así como diferentes requisitos que deben satisfacer los individuos para formar parte de la población auxiliar).

Como hemos mostrado, utilizar una población auxiliar en la que se guardan los mejores individuos obtenidos hasta el momento permite obtener un porcentaje mayor de individuos con *soporte* alto que en el caso de un algoritmo generacional simple sin población auxiliar. Además, de las distintas versiones propuestas para la asignación de individuos a esta población auxiliar, ordenar los individuos candidatos a formar parte de la población auxiliar por orden de *soporte* y que superen un umbral de *soporte* y *confianza* es la que mejores resultados ofrece. Así, en vista de los resultados obtenidos en la Sección 4, nuestra propuesta obtiene un *soporte* global mayor que el algoritmo *Apriori* para cada uno de los 5 conjuntos de datos utilizados. Además, obtenemos menos reglas, por lo que la eficiencia es mayor. A todo esto, añadimos que el *soporte* y *confianza* medios obtenidos son mayores que los conseguidos con el algoritmo *Apriori*.

En futuras investigaciones se abordarán nuevas formas de mutación o de cruce que nos permitan introducir mayor diversidad en el algoritmo y de esta forma ampliar el espacio de búsqueda. Además, se explorará sobre el uso de algoritmos multiobjetivo, en los que podemos maximizar el *soporte* y la *confianza*. Así, comprobamos si este tipo de algoritmos es eficaz en nuestro problema y si supone una mejora a las propuestas presentadas en el presente artículo. Para ello, podemos hacer uso de dos algoritmos multiobjetivo conocidos como son SPEA2 (*Strength Pareto Evolutionary Algorithm*) y NSGA2 (*Non-dominated Sorting Generational Algorithm*). También trataremos de estudiar qué formas de sustitución de individuos son más adecuadas. Obsérvese que para formar parte de la población auxiliar, los individuos deberán compararse según su genotipo y su cadena de derivación. Individuos con el mismo *soporte* o *confianza* no tienen porqué ser igual de 'importantes' y podría sustituirse un individuo por otro siendo, a priori, iguales; sin embargo, en generaciones sucesivas podría ocurrir que se demostrase que hubiese sido mejor no sustituirlos.

Finalmente, en futuros trabajos pretendemos explorar el uso de bases de datos multi-relacionales. La mayoría de las investigaciones de minería de datos están enfocadas a buscar patrones sobre simples tablas de datos, pero hoy en día, la mayoría de los datos comerciales están almacenados en bases de datos relacionales usando multiples tablas conectadas por un identificador. Además, centraremos nuestra investigación en realizar una versión de nuestro algoritmo que trate la incertidumbre y permita obtener reglas de asociación difusas. El enfoque cuantitativo permite a un elemento formar o no parte de un intervalo, lo que nos lleva a una subestimación o sobreestimación de los valores que están cerca de las fronteras de estos conjuntos. Este hecho lleva al desarrollo de reglas de asociación difusas.

Agradecimientos. Este trabajo ha sido financiado por los proyectos del Ministerio de Ciencia y Tec-

nología y de la Junta de Andalucía, TIN2008-06681-C06-03 y TIC-3720, respectivamente.

Referencias

[1] M. Adda, L. Wu and Y. Feng. *Rare Itemset Mining*. Sixth International Conference on Machine Learning and Applications. IEEE Computer Society (2007).

[2] R. Agrawal and R. Srikant. *Fast algorithms for mining association rules*, The International Conference on Very Large Databases (1994), 487-499.

[3] F. Bodon. *A tire-based APRIORI implementation for mining frequent item sequences*. In 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, Chicago, Illinois, USA. ACM Press, New York (2005), 56-65.

[4] C. Borgelt. *Efficient Implementations of Apriori and Eclat*. In Workshop on Frequent Itemset Mining Implementations. CEUR Workshop Proc. 90, Florida, USA (2003).

[5] A.E. Eiben, J.E. Smith. *Introduction to Evolutionary Computing*. In 1st Natural Computing Series. Springer, Heidelberg (2003).

[6] D.E. Golberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, New York (1998).

[7] J. Han and M. Kamber. *Data Mining - Concepts and Techniques*. Morgan Kaufmann, 2nd (2006).

[8] J. Han, J. Pei and Y. Yin. *Mining frequent patterns without candidate generation*. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (2000), 1-12.

[9] J. Mata, J.L. Alvarez and J.C. Riquelme. *Mining numeric association rules via Evolutionary Algorithm*. In 5th International Conference on Artificial Neural Networks and Genetic Algorithms, Prague (2001), 264-267.

[10] J. Mata, J.L. Alvarez and J.C. Riquelme. *Discovering Numeric Association Rules via Evolutionary Algorithm*. In Chen, M.-S., P.S., Liu, B. PAKDD 2002. Springer, Heidelberg (2002), 40-51.

[11] M. Song and R. Sanguthevar. *A transaction mapping algorithm for frequent itemsets mining*, IEEE Transactions on Knowledge and Data Engineering (2006), 472-481.

[12] R. Srikant and R. Agrawal. *Mining quantitative association rules in large relational tables*. ACM SIGMOD International Conference on Management of Data, Canada (1996).

[13] S. Ventura, C. Romero, A. Zafra, J.A. Delgado and C. Hervás. JCLEC: *A Java Framework for Evolutionary Computation*. SoftComputing (2008), 381-392.

[14] M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li. *New algorithms for fast discovery of association rules*. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (1997), 283-296.

[15] *Weka Machine Learning Project*. `http://www.cs.waikato.ac.nz/~ml/index.html`