MOPNAR-BigData: un diseño *MapReduce* para la extracción de reglas de asociación cuantitativas en problemas de *big data*

D. Martín[1], M. Martínez-Ballesteros[2], S. Río[3], J. Alcalá-Fdez[3], J. Riquelme[2], and F. Herrera[3]

[1] Dpto. de Inteligencia Artificial e Infraestructura de Sistemas,
Instituto Superior Politécnico J.A Echeverría, Cujae, La Habana, Cuba
{dmartin}@ceis.cujae.edu.cu
[2] Dpto. de Lenguajes y Sistemas Informáticos,
Universidad de Sevilla, Sevilla, España
{mariamartinez,riquelme}@us.es
[3] Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada, Granada, España
{srio,jalcala,herrera}@decsai.ugr.es

Resumen El término *big data* se ha extendido rápidamente en el área de la minería de datos debido a que las grandes cantidades de datos que se generan hoy en día no pueden ser procesadas o analizadas por las técnicas tradicionales para extraer conocimiento. Durante los últimos años, han sido propuestos algoritmos evolutivos multiobjetivo para extraer reglas de asociación a partir de conjuntos de datos. Sin embargo, estos algoritmos presentan problemas cuando el tamaño del problema aumenta considerablemente. Por esta razón, en este trabajo proponemos MOPNAR-BigData, un nuevo algoritmo iterativo para extraer reglas de asociación cuantitativas positivas y negativas a partir de grandes cantidades de datos. Esta propuesta se basa en el algoritmo MOPNAR y su diseño sigue el paradigma *MapReduce*. Los resultados obtenidos en el estudio experimental realizado sobre tres problemas de *big data* muestran cómo esta propuesta es capaz de obtener conjuntos reducidos de reglas de buena calidad en un tiempo razonable.

Palabras clave: Reglas de Asociación Cuantitativas, Algoritmos Evolutivos Multiobjetivo, *big data*, *MapReduce*

1. Introducción

El uso de técnicas de procesamiento computacionales para análisis y gestión de datos masivos ha revolucionado la investigación científica en los últimos años. En concreto, el proceso de extracción de conocimiento se está convirtiendo en una tarea difícil y compleja ya que en muchos casos, las cantidades de datos generados exceden las capacidades de procesamiento de los sistemas convencionales. Estas cantidades inmensas de datos reciben el nombre de *big data* y el diseño

de algoritmos eficientes que sean capaces de procesar y analizar estas cantidades de datos se ha convertido en un gran desafío para los investigadores. Uno de los enfoques más utilizado para ello es el modelo de programación *MapReduce* [1], el cual es un paradigma computacional para el procesamiento de grandes conjunto de datos en entornos distribuidos.

El descubrimiento de reglas de asociación es una de las técnicas de minería de datos más utilizada para extraer conocimiento interesante a partir de conjuntos de datos. Estas son expresiones del tipo $X \longrightarrow Y$, donde X y Y son conjuntos de items (parejas atributo-valor) y cumplen que $X \cap Y = \emptyset$. Muchos estudios han sido utilizados para extraer reglas de asociación cuantitativas (RACs) a partir de conjuntos de datos con valores numéricos, sin embargo, muchos de ellos no han puesto especial atención en las reglas de asociación negativas. Estas reglas representan la presencia de ciertos ítems ante la ausencia de otros, lo cual puede ser también de gran interés para el usuario. Esto ha provocado que durante los últimos años se hayan propuesto algoritmos para extraer reglas de asociación cuantitativas positivas y negativas (RACPNs) [2].

Muchos algoritmos evolutivos (AEs) han sido propuestos en la literatura para extraer RACs debido a que se consideran unas de las técnicas de búsqueda y optimización de más éxito para problemas complejos. Estos métodos suelen considerar un único objetivo para medir la calidad de las reglas. Recientemente se ha planteado la extracción de reglas de asociación como un problema multiobjetivo. Varios autores han presentado algoritmos evolutivos multiobjetivos para optimizar al mismo tiempo varios objetivos y proporcionar al usuario un conjunto de reglas del frente de Pareto, donde cada solución representa una regla con un grado diferente de equilibrio entre los distintos objetivos optimizados [3,4,5].

Sin embargo, estos algoritmos pueden presentar varias limitaciones para tratar con grandes cantidades de datos debido a que el proceso de evaluación es muy costoso. La evaluación de cada solución requiere el procesamiento de todos los registros del conjunto de datos, por lo tanto, estos algoritmos presentan muchos problemas cuando la cantidad de datos del conjunto de datos aumenta considerablemente [6]. Por este motivo, es necesario rediseñar las técnicas existentes para poder abordar problemas de *big data*. Entre las diferentes soluciones para abordar estos problemas, se encuentra el modelo de programación *MapReduce* [7] que ofrece un paradigma robusto y efectivo para abordar el procesamiento de grandes conjuntos de datos. En este modelo un problema se divide en subproblemas más pequeños y fáciles de abordar, combinando posteriormente las soluciones parciales para obtener el resultado final.

En este trabajo presentamos MOPNAR-BigData, un nuevo algoritmo iterativo basado en el paradigma *MapReduce* para extraer RACPNs de buena calidad a partir de problemas de *big data*. Para ello esta propuesta se basa en el reciente AE multiobjetivo MOPNAR que permite realizar un aprendizaje evolutivo de RACPNs maximizando tres objetivos: rendimiento, interés y comprensibilidad. Para evaluar la calidad de la propuesta, se ha llevado a cabo un estudio experimental sobre tres problemas de *big data*.

Este trabajo se organiza de la siguiente forma. En la Sección 2 se introducen los conceptos básicos sobre las RACs, *big data* y el paradigma de programación *MapReduce*. En la Sección 3 se detalla el algoritmo propuesto para obtener RACPNs a partir de problemas de *big data*. En la Sección 4 se muestran los resultados obtenidos sobre tres problemas de *big data*. Por último, en la sección 5 se proporcionan algunas conclusiones.

2. Preliminares

En esta sección proporcionamos una breve descripción de las RACs (Sección 2.1) y, a continuación, una introducción a *big data* y al modelo de programación *MapReduce* (Sección 2.2).

2.1. Reglas de Asociación Cuantitativas

El proceso de extracción de reglas de asociación, consiste en descubrir conjunciones de pares (atributo - valor) que aparecen en un conjunto de datos con una cierta frecuencia con objeto de obtener reglas que muestren la relación existente entre los atributos [8]. Las reglas de asociación son representaciones del tipo $X \Rightarrow Y$, donde X e Y son conjuntos de elementos y $X \cap Y = \emptyset$. Por tanto, si los elementos de X existen en un ejemplo, entonces, es muy probable que los elementos de Y también aparezcan en el ejemplo. Además, X e Y no deben tener elementos en común. En las RACs [9], cada elemento es un par atributo-intervalo.

La mayoría de los algoritmos existentes suelen extraer RACs positivas sin tener en cuenta RACs negativas. Sin embargo, también son interesantes las reglas del tipo $X \Longrightarrow \neg Y$, ya que relacionan la presencia de X con la ausencia de Y. Las reglas de asociación negativas consideran el mismo conjunto de elementos que las reglas de asociación positivas pero además, pueden incluir elementos negados en el antecedente ($\neg X \Longrightarrow Y$), en el consecuente ($X \Longrightarrow \neg Y$) o en ambos ($\neg X \Longrightarrow \neg Y$).

En cuanto a las medidas de calidad, el soporte y la confianza son las más utilizadas por los algoritmos existentes. Sin embargo, la confianza no tiene en cuenta el soporte del consecuente de la regla, por tanto, no es capaz de detectar dependencias negativas entre los atributos [10]. Por esta razón, en la literatura se han propuesto otras medidas de calidad para evaluar diferentes propiedades de las RACs [11]. Algunas de las más populares son: interés [12], factor de certeza [13] y netconf [14].

2.2. *Big Data* y el Modelo de Programación *MapReduce*

El proceso de extracción de conocimiento se ha convertido en un inmenso desafío para los algoritmos de minería de datos y aprendizaje automático pues no son capaces de abordar de manera sencilla las cantidades de datos que se generan hoy en día. Por este motivo, es necesario rediseñar dichos algoritmos de forma

que puedan ser aplicados a problemas del mundo real. Por ejemplo, las redes sociales como Twitter y Facebook se enfrentan a un crecimiento exponencial de datos diario generando nuevos TB cada día [15]. Y es aquí donde emerge el concepto de *big data*, que hace referencia a cantidades de datos que no pueden ser procesadas o analizadas utilizando herramientas o técnicas tradicionales [1][16].

Una de las soluciones más populares para abordar problemas de *big data* es el modelo de programación *MapReduce* [7]. Se trata de un paradigma computacional que fue diseñado para el procesamiento de grandes conjuntos de datos en entornos distribuidos. Este paradigma procesa datos estructurados en pares (clave-valor) y consta de dos fases: *Map* y *Reduce*. En términos generales, los datos se dividen en subproblemas más pequeños que son distribuidos por los nodos de un clúster para ser procesados de forma independiente en la fase *Map*. Las soluciones parciales obtenidas por cada proceso *Map* se combinan de alguna forma en la fase *Reduce* para obtener una solución final. Apache Hadoop [16] es una de las implementaciones más populares de *MapReduce*. Se trata de un proyecto de código abierto escrito en Java que facilita la escritura de aplicaciones distribuidas, confiables y escalables. Hadoop cuenta con un sistema de almacenamiento distribuido denominado *HDFS* (*Hadoop Distributed File System*), que es altamente tolerante a fallos y está diseñado para usarse con *commodity hardware*.

En la literatura existen técnicas basadas en el paradigma *MapReduce* para la extracción de reglas de asociación, sin embargo, la mayoría de ellas se basan en algoritmos clásicos como Apriori [17], FP-Growth [18] o combinaciones de ambos [19].

Cabe destacar que estas propuestas se centran en datos binarios o discretos, cuando el dominio de la mayoría de las aplicaciones de datos del mundo real es continuo.

3. MOPNAR-BigData: un diseño *MapReduce* para extraer RACs en *big data*

Esta sección describe nuestra propuesta, una implementación *MapReduce* basada en el algoritmo MOPNAR para la extracción de RACPNs en problemas de *big data*: MOPNAR-BigData.

MOPNAR [5] es un nuevo AE multiobjetivo diseñado para extraer un conjunto reducido de RACPNs a un bajo coste computacional con un buen equilibrio entre el número de reglas, soporte y cubrimiento del conjunto de datos. Este algoritmo extiende el AE multiobjetivo basado en descomposición MOEA/D-DE [20] para realizar un aprendizaje evolutivo de los intervalos de los atributos y una selección de las condiciones para cada regla. MOPNAR introduce una población externa y un proceso de reinicialización de la población para almacenar todas las reglas no dominadas encontradas, mejorar la diversidad y mejorar el cubrimiento del conjunto de datos. Este método maximiza tres objetivos: comprensibilidad [21], interés y rendimiento, donde el rendimiento se define como el producto entre el soporte y el factor de certeza [13].

Nuestra propuesta, MOPNAR-BigData, consta de cuatro fases donde tres de ellas siguen un esquema *MapReduce*. La primera fase, que sigue un diseño *MapReduce*, es la encargada de ejecutar el algoritmo MOPNAR y obtener un conjunto de RACPNs (*RuleSet*) para cada uno de los bloques en los que se divide el conjunto de datos de entrada (Sección 3.1). La segunda fase, que también sigue un esquema *MapReduce*, es la responsable de calcular el soporte de las RACPNs obtenidas en la fase anterior teniendo en cuenta todo el conjunto de datos (Sección 3.2). La tercera fase evalúa de manera secuencial este conjunto de reglas y actualiza una lista de reglas globales que hemos denominado *GlobalRulePool* con el fin de almacenar las soluciones no dominadas encontradas en el conjunto de datos completo. La cuarta fase construye un nuevo conjunto de datos a partir de los ejemplos no cubiertos por las reglas que se encuentran en *GlobalRulePool* siguiendo un esquema *MapReduce* (Sección 3.3). Estas cuatro fases forman parte de un proceso iterativo que se repite hasta alcanzar un número de evaluaciones globales dado ($N_{eval-Total}$). El número de evaluaciones se actualiza sumando el número de evaluaciones del subproblema más lento, esto es, el número de evaluaciones de la partición del conjunto de datos original que requiere un mayor número de evaluaciones para ser procesado por MOPNAR en la fase descrita en la Sección 3.1. La Figura 1 presenta un esquema general del método propuesto. Las siguientes secciones describen el diseño de las tres fases *MapReduce*.

3.1. Fase de ejecución del AE para extraer RACs

En esta fase se obtiene un conjunto de RACs (*RuleSet*). Se trata de un proceso *MapReduce* donde cada proceso *Map* se encarga de ejecutar MOPNAR haciendo uso de los datos disponibles en su partición (Fase MR 1 en la Figura 1). Este proceso *MapReduce* consta de tres etapas: **inicial**, ***map*** y **final**.

La etapa **inicial** lleva a cabo una segmentación del conjunto de datos de entrada en bloques independientes que son distribuidos por los nodos de procesamiento. En la etapa ***map***, cada proceso *Map* ejecuta el algoritmo MOPNAR utilizando el bloque de datos de su partición y genera un fichero como el *RuleSet* obtenido. Cada ejecución finaliza cuando se alcanza un número de evaluaciones determinado ($N_{eval-MOPNAR}$). La etapa **final** combina los ficheros generados por cada uno de los procesos *Map* para construir un fichero con todos los *RuleSet* obtenidos. Este proceso *MapReduce* no cuenta con fase *Reduce* ya que las salidas de cada proceso *Map* se combinan de forma directa. La agregación de los resultados obtenidos por los procesos *Map* también podría llevarse a cabo mediante una fase *Reduce* pero al ser innecesaria sólo provocaría que el algoritmo consumiese más tiempo de ejecución.

3.2. Fase de cálculo del soporte de las RACs

Esta fase ejecuta un nuevo proceso *MapReduce* (Fase MR 2 en la Figura 1) que calcula el soporte de la regla, el soporte del antecedente y el soporte del

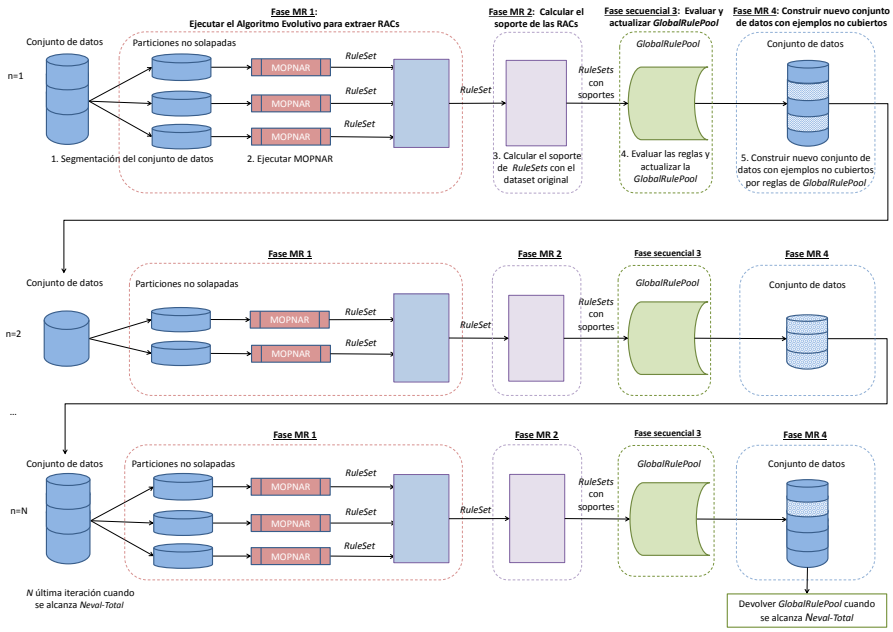

Figura 1. Esquema general de MOPNAR-BigData

consecuente para cada una de las RACs del *RuleSet* obtenido en la fase anterior. Esta fase consta de cuatro etapas: **inicial**, ***map***, ***reduce*** y **final**.

La etapa **inicial** obtiene todas las RACs obtenidas en la fase anterior. En la etapa ***map***, cada proceso *Map* calcula el soporte de las RACs para cada uno de los registros del bloque de datos de su partición y genera un fichero que contiene para cada regla analizada los soportes (antecedente, consecuente y de la regla) obtenido en cada registro. En la etapa ***reduce***, cada proceso *Reduce* combina todos los soportes obtenidos para cada regla para generar el soporte global de cada una en el conjunto de datos original. La etapa **final** genera un fichero con cada una de las RACs y el soporte global obtenido.

3.3. Fase de construcción del nuevo conjunto de datos basado en ejemplos no cubiertos

Esta fase ejecuta un nuevo proceso *MapReduce* (Fase MR 3 en la Figura 1) que se encarga de construir un nuevo conjunto de datos a partir de los registros no cubiertos por las RACs que se encuentran en *GlobalRulePool*. Esta fase consta de tres etapas: **inicial**, ***map*** y **final**. La etapa **inicial** realiza una segmentación del conjunto de datos de entrada en bloques independientes que son distribuidos por los nodos de procesamiento. En la etapa ***map***, cada proceso *Map* analiza para cada uno de los registros del bloque de datos de su partición si alguna

Tabla 1. Problemas de *big data* considerados en el estudio experimental

Conjunto de datos	*#Atrib(R/E/N)*	*#Reg*
poker	(0/6/5)	1025010
susy	(18/0/1)	5000000
higgs	(28/0/1)	11000000

Tabla 2. Parámetros considerados en la comparación de los métodos

Algoritmos	*Parámetros*
MOPNAR	N_{eval}=100000, H=13, m=3, PopSize=N_{H+m-1}^{m-1}, T=10, δ=0.9, η_r=2, γ=2, P_{mut}= 0.1, α = 5 %
MOPNAR-BigData	$N_{eval-Total}$=100000, $N_{eval-MOPNAR}$=25000, H=9, trNotCover=2 %, m=3, PopSize=N_{H+m-1}^{m-1}, T=7, δ=0.9, η_r=2, γ=2, P_{mut}= 0.1, α = 5 %

de las RACs de *GlobalRulePool* lo cubre, generando un fichero que contiene los registros que no han sido cubiertos por ninguna de las RACs de *GlobalRulePool*. La etapa **final** construye un nuevo fichero que contiene la agregación de los ficheros generados por cada uno de los procesos *Map*. En el caso de que el número de registros del nuevo conjunto de entrada sea menor que un umbral mínimo de cubrimiento ($trNotCover$), se añadirán de nuevo todos los registros del conjunto de datos original. Este nuevo conjunto de datos se utilizará como entrada de la fase descrita en la Sección 3.1. Este proceso *MapReduce* tampoco cuenta con fase *Reduce* ya que las salidas de cada proceso *Map*, registros no cubiertos, se combinan de forma directa.

4. Estudio Experimental

En esta sección se describe el estudio experimental realizado para analizar el rendimiento de MOPNAR-BigData. Describimos las principales características de los tres problemas *big data* seleccionados, así como los métodos y la configuración de parámetros utilizados en el estudio. Presentamos un estudio de escalabilidad del algoritmo MOPNAR respecto a MOPNAR-BigData con el objetivo de analizar su comportamiento en problemas de *big data*. Evaluamos el rendimiento del método propuesto respecto a la versión secuencial y por último analizamos la calidad de las reglas obtenidas por MOPNAR-BigData en estos problemas.

En la Tabla 1 se describen las principales características de los tres conjuntos de datos seleccionados, los cuales están disponibles en el repositorio UCI [22]. Para cada conjunto de datos se muestra el número de registros ("#Reg") y el número de atributos reales, enteros o nominales ("#Atrib(R/E/N)").

Los parámetros de los métodos analizados se muestran en la Tabla 2. Con estos valores, hemos tratado de facilitar las comparaciones, seleccionando parámetros estándar comunes que funcionan bien en la mayoría de los casos.

Tabla 3. Tiempo de ejecución en segundos conforme aumenta el número de registros sobre el conjunto de datos higgs

$\#Reg$	$MOPNAR$	$MOPNAR-BigData$	$\#Maps$
100000	2540	1429	4
200000	5758	1565	8
400000	9965	1701	16
800000	21522	1976	32
1600000	41634	2881	64
3200000	N.D	6843	128

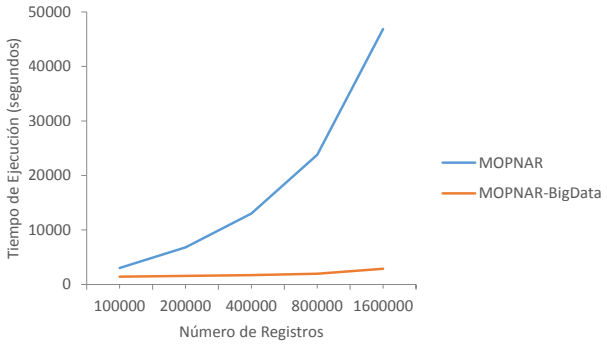

Figura 2. Relación entre el tiempo de ejecución y el número de registros para los algoritmos MOPNAR y MOPNAR-BigData en las distintas versiones del problema higgs

Para llevar a cabo el análisis de escalabilidad se han generado versiones reducidas del conjunto de datos higgs. Para el algoritmo MOPNAR-BigData se ha considerado un tamaño de partición de 25000 registros. En la Tabla 3 se muestra el número de registros de cada subconjunto de datos ("#Reg"), los tiempos de ejecución obtenidos por MOPNAR y MOPNAR-BigData, así como el número de particiones o *maps* que realiza MOPNAR-BigData para cada versión del conjunto de datos ("#Maps"). El símbolo N.D (No Disponible) es incluido para indicar que el algoritmo no fue capaz de abordar el problema asociado. La Figura 2 muestra la relación entre el tiempo de ejecución y el número de registros para ambos algoritmos.

Analizando los resultados obtenidos podemos ver como el algoritmo MOPNAR tiende a escalar exponencialmente cuando el número de registros del problema aumenta. También podemos observar que dicho algoritmo no fue capaz de completar el experimento para la versión del conjunto de datos higgs con 3200000 registros. Por otra parte, se puede comprobar como la diferencia de tiempo entre MOPNAR-BigData y MOPNAR es mayor conforme el tamaño del problema aumenta, llegando a ser la versión *MapReduce* hasta 16 veces más rápida que la versión secuencial.

Tabla 4. Resultados obtenidos por MOPNAR y MOPNAR-BigData en las distintas versiones del problema higgs

Algoritmos	*#R*	*Soporte*	*Confianza*	*Interes*	*Factor Certeza*	*Netconf*	*Atributos*	*%Registros*	*Tiempo*
				higgs (100000 registros)					
MOPNAR	**62,60**	**0,39**	0,84	**3,89**	0,74	**0,47**	**2,93**	99,31	2540
MOPNAR-BigData	49	0,32	**0,86**	1,89	**0,77**	0,38	3,00	**100**	**1429**
				higgs (200000 registros)					
MOPNAR	**159,20**	0,30	0,82	**3,21**	0,75	**0,52**	2,71	97,49	5758
MOPNAR-BigData	49	**0,46**	**0,98**	2,62	**0,93**	0,46	**2,39**	**100**	**1565**
				higgs (400000 registros)					
MOPNAR	**211,20**	0,41	0,87	**4,79**	0,80	0,50	2,64	96,12	9965
MOPNAR-BigData	46,00	**0,51**	**0,92**	2,02	**0,88**	**0,53**	**2,42**	**99,98**	**1701**
				higgs (800000 registros)					
MOPNAR	**224,60**	0,32	0,86	**6,44**	0,74	0,53	3,11	99,44	21522
MOPNAR-BigData	75	**0,43**	**0,95**	4,59	**0,91**	**0,65**	**2,58**	**100**	**1976**
				higgs (1600000 registros)					
MOPNAR	**575,80**	**0,54**	**0,92**	2,00	0,78	0,33	2,75	98,21	41634
MOPNAR-BigData	125	0,39	**0,92**	**3,55**	**0,87**	**0,66**	**2,25**	**99,98**	**2881**

Tabla 5. Resultados obtenidos por MOPNAR-BigData en los problemas poker, susy y higgs

Algoritmos	*#R*	*Soporte*	*Confianza*	*Interes*	*Factor Certeza*	*Netconf*	*Atributos*	*%Registros*	*Tiempo*
					poker				
MOPNAR-BigData	28	0,19	0,91	2,85	0,83	0,47	2,61	97,30	1421
					susy				
	303	0,38	0,97	7,16	0,96	0,82	2,53	100	12316
					higgs				
	196	0,36	0,95	4,08	0,93	0,70	2,32	99,96	34773

En cuanto al análisis de rendimiento, la Tabla 4 muestra los resultados medios para todas las reglas obtenidas por MOPNAR y MOPNAR-BigData en las versiones reducidas del conjunto de datos higgs, donde $\#R$ representa el número de reglas; *Soporte*, $Confianza$, *Interes*, $Factor\ Certeza$, $Netconf$ representan los valores medios de soporte, confianza, interés, factor de certeza y netconf, respectivamente; *Atributos* es el número medio de atributos involucrados en las reglas; $\%Registros$ es el tanto por ciento de registros cubiertos por las reglas generadas y $Tiempo$ se corresponde con el tiempo de ejecución empleado en ese problema.

Como podemos observar, MOPNAR y MOPNAR-BigData obtienen valores similares para todas las medidas de interés. Sin embargo, cuando el número de registros del problema aumenta, el método propuesto presenta mejoras en casi todas las medidas. Destacar que MOPNAR-BigData obtiene menor número de reglas que la versión secuencial, donde cada regla nos proporciona un conocimiento interesante y diferente del problema.

Por último, la Tabla 5 presenta los resultados medios para todas las reglas obtenidas por nuestra propuesta en los tres problemas *big data* analizados (la cabecera de esta tabla ha sido introducida anteriormente). Debido al tamaño de

estos problemas, en este estudio se ha considerado un tamaño de partición de 50000 registros.

A partir del análisis de los resultados podemos destacar que nuestra propuesta MOPNAR-BigData obtiene un conjunto de RACPNs con pocos atributos y valores elevados para las medidas de interés, proporcionando reglas comprensibles de muy buena calidad. Además, teniendo en cuenta que estamos abordando problemas *big data*, nuestra propuesta obtuvo un conjunto reducido de reglas y con un buen cubrimiento del conjunto de datos, facilitando así la interpretabilidad desde el punto de vista del usuario y aportándole un conocimiento sobre la totalidad del problema.

5. Conclusiones

En este trabajo se ha presentado un nuevo método iterativo para extraer RACPNs en problemas de *big data*, denominado MOPNAR-BigData. Nuestra propuesta se basa en el algoritmo MOPNAR y su diseño sigue un esquema *MapReduce*, una de las soluciones más populares para abordar problemas de *big data*.

Los resultados obtenidos muestran que nuestra propuesta obtiene RACPNs con altos valores de las medidas de interés en los tres problemas *big data* utilizados. Además, obtiene conjuntos reducidos de reglas con un alto cubrimiento de los conjuntos de datos (más del 99 % en todos los conjuntos de datos), ofreciéndole al usuario información comprensible sobre todo el problema.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad bajo los proyectos TIN2014-57251-P, TIN2011-28956-C02-02 y TIN2014-55894-C2-1-R y los proyectos regionales andaluces P11-TIC-7765 y P10-TIC-6858.

Referencias

1. Fernández, A., Río, S., López, V., Bawakid, A., del Jesus, M., Benítez, J., Herrera, F.: Big data with cloud computing: An insight on the computing environment, mapreduce and programming frameworks. WIREs Data Mining and Knowledged Discovery. **4**(5) (2014) 380–409
2. Alatas, B., Akin, E.: An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules. Soft Computing **10**(3) (2006) 230–237
3. Martín, D., Rosete, A., Alcalá-Fdez, J., Herrera, F.: QAR-CIP-NSGA-II: A new multi-objective evolutionary algorithm to mine quantitative association rules. Information Sciences **258** (2014) 1–28
4. Martínez-Ballesteros, M., Nepomuceno-Chamorro, I., Riquelme, J.: Discovering gene association networks by multi-objective evolutionary quantitative association rules. Journal of Computer and System Sciences **80**(1) (2014) 118–136

5. Martín, D., Rosete, A., Alcalá-Fdez, J., Herrera, F.: A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules. IEEE Transactions on Evolutionary Computation **18**(1) (2014) 54–69
6. Martínez-Ballesteros, M., Bacardit, J., Troncoso, A., Riquelme, J.: Enhancing the scalability of a genetic algorithm to discover quantitative association rules in large-scale datasets. Integrated Computer-Aided Engineering **22**(1) (2015) 21–39
7. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. Commun. ACM **51**(1) (2008) 107–113
8. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. (1993) 207–216
9. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. ACM SIGMOD Record **25**(2) (1996) 1–12
10. Martínez-Ballesteros, M., Martínez-Álvarez, F., Troncoso, A., Riquelme, J.: Selecting the best measures to discover quantitative association rules. Neurocomputing **126** (2014) 3–14
11. Geng, L., Hamilton, H.: Interestingness measures for data mining: A survey. ACM Computing Surveys **38**(3) (2006) 1–42
12. Ramaswamy, S., Mahajan, S., Silberschatz, A.: On the discovery of interesting patterns in association rules. In: Proceedings of the 24th International on Very Large Data Bases. (1998) 368–379
13. Shortliffe, E., Buchanan, B.: A model of inexact reasoning in medicine. Mathematical Biosciences **23** (1975) 351–379
14. Ahn, K.I., Kim., J.Y.: Efficient mining of frequent itemsets and a measure of interest for association rule mining. Journal of Information & Knowledge Management. **3**(3) (2004) 245–257
15. Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sarma, J.S., Murthy, R., Liu, H.: Data warehousing and analytics infrastructure at facebook. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. SIGMOD '10, New York, NY, USA, ACM (2010) 1013–1020
16. White, T.: Hadoop: The Definitive Guide. 1st edn. O'Reilly Media, Inc. (2009)
17. Lin, M., Lee, P., Hsueh, S.: Apriori-based frequent itemset mining algorithms on MapReduce. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. ICUIMC '12, New York, NY, USA, ACM (2012) 1–8
18. Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.: PFP: Parallel FP-Growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems. RecSys '08, New York, NY, USA, ACM (2008) 107–114
19. Wang, L., Feng, L., Zhang, J., Liao, P.: An efficient algorithm of frequent itemsets mining based on MapReduce. Journal of Information and Computational Science **11**(8) (2014) 2809–2816
20. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation **13**(2) (2009) 284–302
21. Fidelis, M., Lopes, H., Freitas, A.: Discovering comprehensible classification rules with a genetic algorithm. In: Proceedings of the 2000 Congress on Evolutionary Computation, 2000. Volume 1. (2000) 805–810
22. Bache, K., L.M.: Uci machine learning repository. `http://archive.ics.uci.edu/ml` (2015) [Online; consultado en Septiembre 2015].