Una primera aproximación al descubrimiento de subgrupos bajo el paradigma *MapReduce*

F. Pulgar-Rubio[1], C.J. Carmona[2], A.J. Rivera-Rivas[1], P. González[1], M.J. del Jesus[1]

Departamento de Informática, Universidad de Jaén [1]
{fpulgar,pglez,mjjesus}@ujaen.es
Área de Lenguajes y Sistemas Informáticos del Departamento de Ingeniería Civil, Universidad de Burgos [2]
cjcarmona@ubu.es

Resumen A día de hoy, existe un incremento exponencial de los datos que hace necesario buscar nuevas metodologías y técnicas que sean capaces de analizar gran cantidad de información. Este concepto se conoce como Big Data y se divide en cuatro dimensiones: volumen, velocidad, variedad y veracidad. En esta contribución se presenta una versión para tratamiento de datos masivos del algoritmo de descubrimiento de subgrupos, NMEEF-SD, bajo el paradigma *MapReduce*. Este nuevo enfoque permite analizar gran cantidad de información de una forma más rápida y eficiente. Además, se incluye un estudio experimental sobre la base de datos Census para mostrar las bondades del enfoque introducido.

Keywords: Descubrimiento de subgrupos, Big Data, NMEEF-SD

1. Introducción

El desarrollo que las tecnologías para la generación y transmisión de información que las tecnologías de la información ha experimentado en los últimos años ha provocado un crecimiento exponencial de los datos obtenidos en multitud de campos como física, meteorología, bioinformática, etc. De hecho, se considera que a partir de 2012 cada día se crean más de 2,5 trillones de bytes de datos[1].

El concepto Big Data [20] abarca una colección de conjuntos de datos cuyo tamaño y complejidad desafían los sistemas de gestión de bases de datos estándar y la aplicación de técnicas de extracción de conocimiento. En un principio se consideraba que el conocimiento obtenido al analizar un mayor conjunto de datos permitiría realizar análisis más precisos de los resultados. Sin embargo, los algoritmos estándar de minería de datos no son capaces de hacer frente a estos enormes conjuntos de datos [15]. De esta manera, los algoritmos deben de volver a diseñarse y adaptarse teniendo en cuenta las soluciones que están siendo utilizadas para analizar grandes volúmenes de datos [10].

[1] http://www-01.ibm.com/software/data/bigdata/

El descubrimiento de subgrupos es una técnica de minería de datos para describir problemas mediante reglas descriptivas obtenidas mediante aprendizaje supervisado [13,17]; es decir, obtener reglas que describan los datos para una variable objetivo. Hasta este momento, el descubrimiento de subgrupos no ha sido analizado con grandes conjuntos de datos utilizando las tecnologías de tratamiento masivo de datos aportadas por la comunidad.

En este trabajo, se presenta una adaptación del algoritmo NMEEF-SD [3] bajo el paradigma *MapReduce* [7] implementado mediante SPARK [19], organizándose en las siguientes secciones. En la Sección 2 se presentan las principales propiedades del paradigma *MapReduce*, en la Sección 3, se puede observar la definición y principales propiedades del descubrimiento de subgrupos, así como un resumen del algoritmo NMEEF-SD, y en la Sección 4 se puede observar el estudio experimental realizado. Para finalizar se presentan las conclusiones obtenidas en el trabajo junto con los trabajos futuros.

2. Big Data y el paradigma *MapReduce*

El gran avance producido en el campo de las tecnologías de la información ha conllevado a las organizaciones y empresas a nuevos desafíos en el análisis de grandes cantidades de información, y de ahí surge el concepto Big Data aplicado fundamentalmente a información que no puede ser procesada o analizada mediante técnicas clásicas [16]. Big Data se suele describir mediante el modelo de las 4 Vs^2 más importantes dentro de este campo:

- *Volumen* o cantidad de información que se necesita analizar.
- *Velocidad* con la que deberían ser analizados los datos y devolver resultados.
- *Variedad* de fuentes datos que se pueden analizar.
- *Veracidad* e integridad de los datos.

La propuesta más extendida para afrontar este tipo de problemas es el paradigma *MapReduce* [7] propuesto por dos desarrolladores de Google, Dean and Ghemawat, y se puede observar en la Fig. 1. Tal y como se puede ver en este imagen, el paradigma se centra en dos funciones fundamentales:

- Función Map: El nodo maestro del clúster realiza una segmentación del conjunto de datos de entrada en bloques independientes y los distribuye a los nodos de trabajo. A continuación, el nodo de trabajo procesa el problema más pequeño, y pasa la respuesta de nuevo a su nodo principal. El Map utiliza el concepto de par clave-valor como entrada y emite un conjunto intermedio de pares clave-valor como salida. Antes de ejecutar la función Reduce, el paradigma agrupa todos los valores intermedios asociados con la misma clave y los transforma para acelerar el cálculo en la función Reduce.
- Función Reduce: El nodo maestro recoge todos los mapas y los combina de alguna manera para formar el producto final. Considerando los pares clave-valor, esta función acepta la clave intermedia proporcionada y genera como resultado final el correspondiente par de clave y valor.

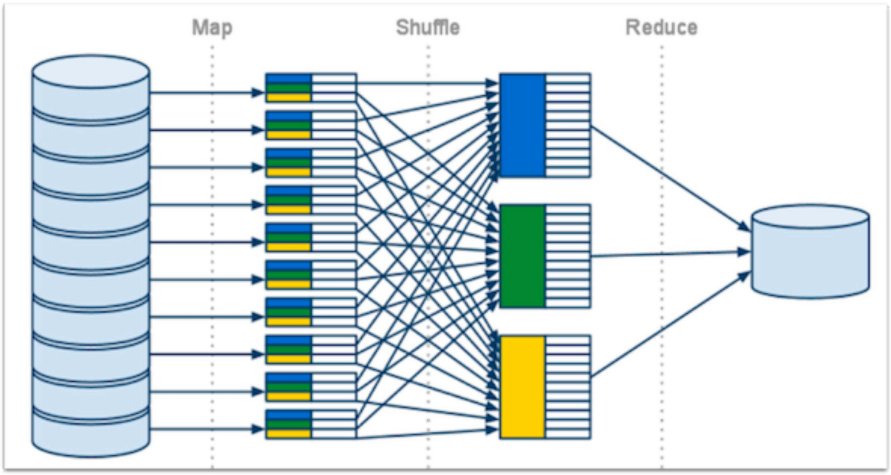

Figura 1. Representación del paradigma *MapReduce*

Este paradigma ha sido ampliamente utilizado bajo Apache Hadoop [16], un software de código abierto escrito en Java basado en un sistema de almacenamiento distribuido denominado HDFS (Hadoop Distributed File System). Sin embargo, Hadoop no es lo suficientemente eficiente con algoritmos iterativos, principalmente por la sobrecarga en el arranque de los procesos. De hecho, se han realizado muchos esfuerzos para mejorar este problema de sobrecarga. Entre las soluciones aportadas por la comunidad destaca SPARK [19] que mejora respecto a Hadoop cuando se utilizan soluciones iterativas, conservando la escalabilidad y tolerancia a fallos de *MapReduce*. El concepto fundamental de la mejora de SPARK se centra en el uso de los conjuntos de datos distribuidos resilientes (RDD). Estas estructuras permiten controlar el particionamiento de los datos con tolerancia a fallos. SPARK se desarrolló en la Universidad de Berkeley AMPLab y se utiliza para ejecutar aplicaciones a gran escala, como el filtrado de spam y la predicción del tráfico. Spark permite trabajar en Java, Python, R y Scala, pero debido a que SPARK está desarrollado fundamentalmente en Scala se recomienda la utilización de dicho lenguaje para aprovechar mejor sus características.

3. Descubrimiento de subgrupos

El descubrimiento de subgrupos es un tipo de inducción descriptiva que pretende generar modelos basados en reglas cuya finalidad es descriptiva, empleando una perspectiva predictiva para obtenerlos [12,6]. Se trata por tanto de una tarea con objetivos básicamente descriptivos que incluye características de la inducción predictiva. Este concepto se define como [18]:

[2] http://www-01.ibm.com/software/data/bigdata/

> *En descubrimiento de subgrupos, asumimos una población de individuos dada (objetos, clientes, ...) y una propiedad de estos individuos en la que estemos interesados. La tarea del descubrimiento de subgrupos es entonces descubrir los subgrupos de la población que son estadísticamente "más interesantes"; es decir, individuos que sean tan grandes como sea posible y tengan una distribución estadística los más atípica posible, con respecto a la propiedad de interés.*

Así, una regla (R), que consiste de una descripción de un subgrupo inducido, puede ser definida formalmente como [14]:

$$R : Cond \rightarrow VarObj$$

donde $VarObj$ es el valor de la variable de interés o variable objetivo para la tarea de descubrimiento de subgrupos (puede aparecer además en la bibliografía específica como $Clase$), y $Cond$ es comúnmente una conjunción de funciones (pares atributo-valor) que es capaz de describir una distribución estadística inusual con respecto a la variable objetivo.

Existen diferentes elementos a especificar en el diseño de un algoritmo de descubrimiento de subgrupos [1], donde uno de los más destacados son las medidas de calidad utilizadas para el proceso de búsqueda y/o evaluación de los algoritmos. A continuación, se detallan las medidas de calidad más utilizadas en la literatura y en este trabajo:

- *Atipicidad*: Esta medida se describe como el balance entre la cobertura de la regla y la ganancia de precisión [14]. Se puede calcular como:

$$Atip(R) = \frac{n(Cond)}{n_s}\left(\frac{n(VarObj \cdot Cond)}{n(Cond)} - \frac{n(VarObj)}{n_s}\right) \quad (1)$$

  donde n_s es el número de ejemplos, $n(Cond)$ es el número de ejemplos que satisfacen la condición de la regla, $n(VarObj \cdot Cond)$ es el número de ejemplos que satisfacen la condición y además pertenecen al valor de la variable objetivo en la regla y $n(VarObj)$ son todos los ejemplos del valor de la variable objetivo.
- *Sensibilidad*: Esta medida mide la proporción de ejemplos correctamente descritos [13]. Se puede calcular como:

$$Sens(R) = \frac{n(VarObj \cdot Cond)}{n(VarObj)} \quad (2)$$

  Esta medida de calidad se utiliza ara evaluar la calidad de los subgrupos en el espacio ROC (*Receiver Operating Characteristic*). La medida de sensibilidad combina la precisión y generalidad generada para un valor de la variable objetivo.

- *Confianza difusa*: Determina la frecuencia relativa de los ejemplos que satisfacen tanto el antecedente como el consecuente de una regla entre aquellos que satisfacen sólo el antecedente [9]. Se calcula como:

$$CnfD(R) = \frac{\sum_{E^k \in E / E^k \in VarObj} APC(E^k, R)}{\sum_{E^k \in E} APC(E^k, R)} \quad (3)$$

  donde APC es el grado de compatibilidad entre un ejemplo y el antecedente de una regla difusa.

Dentro del descubrimiento de subgrupos, el algoritmo NMEEF-SD [3] es uno de los más conocidos y se ha aplicado de forma exitosa en distintos problemas reales de bioinformática [2], e-learning [4], o minería de uso web [5]. Este algoritmo está basado en un enfoque evolutivo difuso [11] que pretende obtener subgrupos difusos generales y precisos. Para ello, el algoritmo incorpora componentes para potenciar estas características con operadores de inicialización y mutación sesgada y reinicialización basada en cobertura. El algoritmo evolutivo es multi-objetivo y permite la selección de distintas medidas de calidad como objetivos, aunque la combinación de Atipicidad (Ec. 1) y Sensibilidad (Ec. 2) obtiene los mejores resultados tal y como se puede observar en [3].

El funcionamiento de NMEEF-SD está basado en el algoritmo NSGA-II [8] incorporando como eje fundamental la reinicialización basada en cobertura. En una etapa inicial el algoritmo inicializa una población con parte de los individuos sesgados a un número máximo de variables y la otra parte de forma completamente aleatoria. A continuación mediante distintos operadores genéticos de selección, cruce multipunto y mutación sesgada se genera una nueva población que es unida a la anterior. Es en este momento cuando se aplica la ordenación mediante frentes de dominancia, de forma que el primer frente es el frente de Pareto (individuos no dominados), el segundo frente son los individuos dominados únicamente por un individuo, el tercero los dominados por dos, y así sucesivamente. Estos frentes van incorporándose de forma ordenada en la población principal de la siguiente generación hasta completarla. Si tras un número determinado de evaluaciones el frente de Pareto cubre siempre los mismos ejemplos del conjunto de datos, es decir, si el frente de Pareto no evoluciona durante un periodo del proceso evolutivo, se aplica la reinicialización basada en cobertura cuyo objetivo es crear nuevos individuos en la población que cubran ejemplos no cubiertos hasta el momento por el Pareto. Este operador aporta al algoritmo una mayor diversidad en las soluciones obtenidas.

La adaptación del algoritmo al paradigma *MapReduce* se ha llevado a cabo mediante una nueva implementación utilizando el lenguaje Scala debido a que es el lenguaje que mejor aprovecha las virtudes de Spark de cara al procesamiento en paralelo de los datos puesto que es el lenguaje principal utilizado en el desarrollo de Spark.

La parte fundamental del nuevo modelo se centra en las funciones asociadas al paradigma *MapReduce*:

- *Map*. En un primer paso, el algoritmo distribuye los datos de entrada en tantas particiones como se indiquen. Posteriormente, el algoritmo procesa cada una de esas particiones de forma independiente para, finalmente, devolver los resultados obtenidos, es decir, en esta etapa se obtiene un conjunto de reglas asociado a la partición correspondiente. Es importante destacar que para aprovechar al máximo el procesamiento en paralelo mediante SPARK se permite realizar tantas divisiones del conjunto de datos como COREs formen el sistema distribuido.
- *Reduce*. El algoritmo recoge todas las reglas obtenidas en cada map y las unifica en un único conjunto de reglas. Posteriormente, realiza un análisis de las reglas con respecto al conjunto completo de los datos y elimina aquellas que están repetidas y reduciendo de forma sustancial el conjunto inicial de reglas. Por definición, el descubrimiento de subgrupos busca conocimiento parcial en el conjunto de datos en vez de modelos completos y de ahí la decisión tomada sobre la función Reduce en esta primera aproximación.

Es importante destacar que el algoritmo permite al usuario definir, de forma previa a la ejecución, el número de particiones en las que se van a dividir los datos, es decir, el número de conjuntos de datos que serán procesados en paralelo. Tal y como se ha indicado anteriormente, el número de particiones seleccionado debe ser menor o igual que el número de COREs total del sistema distribuido para aprovechar las ventajas del procesamiento en paralelo.

4. Estudio experimental

En este trabajo se presenta un estudio preliminar para demostrar la capacidad del nuevo enfoque NMEEFSD-BigData para trabajar con grandes conjuntos de datos frente al algoritmo NMEEF-SD. Para ello, se trabaja con el conjunto de datos *Census* disponible en el repositorio UCI KDD[3]. Este conjunto de datos contiene datos demográficos y variables relacionadas con el empleo de un censo de 1994 y 1995 llevado a cabo por la *US Census Bureau*. El conjunto de datos está compuesto por 199,523 instancias y unos 40 atributos, como por ejemplo, edad, educación, sexo, capital, país de nacimiento del padre y de la madre, semanas trabajadas al año, etc.

Para ejecutar tanto la versión clásica como la de *BigData* del algoritmo NMEEF-SD, se han utilizado los parámetros indicados en la Tabla 1. Además, para la ejecución en el sistema distribuido SPARK se han considerado las siguientes propiedades y parámetros indicados en la Tabla 2.

A continuación, en la Tabla 3 se detallan los promedios de los resultados obtenidos por los algoritmos ($Alg.$), en los distintos conjuntos de datos analizados (BD) con las particiones ($Part.$) utilizadas para el paradigma *MapReduce*. Las medidas de calidad analizadas son el número de reglas ($Reglas$), el número de variables en el antecedente ($Vars$), Atipicidad ($Atip$), Sensibilidad ($Sens$) y la Confianza ($Conf$). Además, se incluye el tiempo ($T(seg.)$) que se ha tardado en

[3] https://kdd.ics.uci.edu/

Tabla 1. Parámetros de los algoritmos empleados en el estudio experimental

Algoritmo	Parámetros
NMEEFSD	Número de etiquetas lingüísticas=3, máximo número de evaluaciones=10000, tamaño de la población=51, porcentaje de cruce=60 %, porcentaje de mutación=10 %, objetivos=atipicidad y sensibilidad, mínima confianza=60 %.

Tabla 2. Propiedades y parámetros considerados en el sistema distribuido SPARK

Propiedad	Propósito	Valor
Nodos	Indica el número máximo de nodos que tiene el servidor distribuido	14
Cores	Número máximo de núcleos por cada nodo	24
Memoria	Indica la memoria máxima a utilizar por nodo	52G
Serialización	Tamaño máximo permitido al serializar tareas	100M

construir el modelo final de reglas por el algoritmo en las distintas ejecuciones medido en segundos.

Tabla 3. Resultados obtenidos en el estudio experimental

BD	*Alg*	*Part.*	*Reglas*	*Vars*	*Atip*	*Sens*	*Conf*	*T(seg.)*
Census 10 %	Clásico	-	45	4.84	0.013	0.841	0.963	713
	BigData	2	47	4.83	0.012	0.848	0.962	657
	BigData	4	104	4.75	0.012	0.829	0.964	329
	BigData	8	156	5.13	0.013	0.818	0.965	167
	BigData	16	257	5.48	0.013	0.815	0.965	91
	BigData	32	414	5.59	0.014	0.787	0.968	50
Census 100 %	Clásico	-	46	4.91	0.012	0.847	0.963	756
	BigData	2	67	4.48	0.013	0.827	0.964	744
	BigData	4	120	4.87	0.013	0.815	0.966	361
	BigData	8	158	5.09	0.013	0.819	0.965	191
	BigData	16	273	5.48	0.013	0.809	0.966	98
	BigData	32	421	5.61	0.014	0.795	0.967	55

Los resultados permiten apreciar una clara disminución del tiempo de ejecución, gracias a la aplicación del paradigma *MapReduce* en el algoritmo NMEEFSD, al incrementar el número de particiones. De hecho, se obtienen reducciones superiores al 90 % cuando se utilizan más de 30 particiones para mapear el conjunto de datos. A pesar de que la mejora en tiempo es sustancial, es necesario necesario realizar un estudio pormenorizado de las medidas de calidad más destacadas dentro del descubrimiento de subgrupos:

- El *número de reglas* aumenta de forma lineal de la misma forma que aumentan las particiones que se indican en la paralelización del algoritmo. Esto será

objeto de estudio en mayor profundidad, para intentar optimizar la función *Reduce* que en estos momentos se ha considerado, ya que un algoritmo de descubrimiento de subgrupos debe obtener pocas reglas, con pocas variables (aportar generalidad) con la mayor precisión y atipicidad posible. Por otro lado, esta solución aporta un mayor conjunto de conocimiento independiente a los expertos que quizás pueda ser interesante dependiendo la naturaleza del problema.

- El *número de variables* aumenta con el número de particiones, es decir, el crecimiento es directamente proporcional al número de particiones. Este incremento conlleva una mayor especialización de los subgrupos obtenidos, es decir, reducción de la generalidad, y puede ser debido a que se reduce drásticamente el número de instancias cuando se eleva el número de particiones.
- La *atipicidad* mantiene los mismos valores desde el algoritmo original y permanece prácticamente igual para todas las ejecuciones.
- La *sensibilidad* se reduce levemente cuando el numero de particiones se incrementa debido a la mayor especialización de los subgrupos, tal y como se ha comentado con respecto al aumento del número de variables.
- Por el contrario, este incremente en el número de variables conlleva una mejora leve en la medida de *confianza* tal y como se puede observar en los resultados obtenidos.

5. Conclusiones y trabajos futuros

En este trabajo se presenta una primera aproximación *BigData* para uno de los algoritmos más destacados dentro de la literatura del descubrimiento de subgrupos, el algoritmo NMEEF-SD. Este nuevo enfoque está basado en el paradigma *MapReduce* de forma que se realizan tantas divisiones (*mappers*) del conjunto de datos, como se indiquen en el fichero de parámetros, y posteriormete se ejecuta sobre cada partición el algoritmo evolutivo. Una vez todas las ejecuciones se han acabado, se ejecuta la función de combinación de todos los resultados en un único fichero, eliminando los subgrupos repetidos.

En esta primera aproximación, se muestran las bondades del método con respecto a la reducción en tiempos de ejecución, manteniendo calidad en los resultados obtenidos con respecto a generalidad, precisión y atipicidad. Sin embargo, es destacable ver el incremento de reglas extraídas por el algoritmo cuando el número de *mappers* también se incrementa, y la especialización de las reglas con un ligero incremento de variables. Además, se observó en estudios preliminares que la utilización de un número elevado de mapas, por ejemplo 64, conlleva un incremento muy alto de reglas para una reducción de tiempo insignificante ya que se obtuvieron más de 700 reglas en un tiempo muy cercano al obtenido con 32 mapas para ambos conjuntos de datos. Por otro lado, sería conveniente realizar un análisis de las distribuciones de datos en particiones, así como sus dependencias en distintas ejecuciones.

Como trabajos futuros es fundamental abordar el incremento en la cardinalidad de las reglas, realizando un amplio estudio para ver distintas posibilidades a

la hora de realizar las funciones *Shuffle* y *Reduce* de los algoritmos de descubrimiento de subgrupos para *BigData*. Tampoco se puede dejar de lado la necesidad de analizar el posible solapamiento entre subgrupos y la posibilidad de incluir pesos en los mismos para eliminar reglas que están por debajo de umbral.

Acknowledgments.

Este trabajo ha sido subvencionado por el Ministerio de Economía y Competitividad bajo el proyecto TIN2012-33856, Fondos FEDER.

Referencias

1. M. Atzmueller, F. Puppe, and H. P. Buscher, *Towards Knowledge-Intensive Subgroup Discovery*, Proceedings of the Lernen - Wissensentdeckung - Adaptivität - Fachgruppe Maschinelles Lernen, 2004, pp. 111–117.
2. C. J. Carmona, C. Chrysostomou, H. Seker, and M. J. del Jesus, *Fuzzy Rules for Describing Subgroups from Influenza A Virus Using a Multi-objective Evolutionary Algorithm*, Applied Soft Computing **13** (2013), no. 8, 3439–3448.
3. C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera, *NMEEF-SD: Non-dominated Multi-objective Evolutionary algorithm for Extracting Fuzzy rules in Subgroup Discovery*, IEEE Transactions on Fuzzy Systems **18** (2010), no. 5, 958–970.
4. C. J. Carmona, P. González, M. J. del Jesus, and S. Ventura, *Subgroup discovery in an e-learning usage study based on Moodle*, Proceedings of the International Conference of European Transnational Education, 2011, pp. 446–451.
5. C. J. Carmona, S. Ramírez-Gallego, F. Torres, E. Bernal, M. J. del Jesus, and S. García, *Web usage mining to improve the design of an e-commerce website: OrOliveSur.com*, Expert Systems with Applications **39** (2012), 11243–11249.
6. C.J. Carmona, P. González, M.J. del Jesus, and F. Herrera, *Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms*, WIREs Data Mining and Knowledge Discovery **4** (2014), no. 2, 87–103.
7. J. Dean and S. Ghemawat, *MapReduce: Simplified data processing on large clusters*, Communications of the ACM **1** (2008).
8. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Willey & Sons, 2001.
9. M. J. del Jesus, P. González, F. Herrera, and M. Mesonero, *Evolutionary Fuzzy Rule Induction Process for Subgroup Discovery: A case study in marketing*, IEEE Transactions on Fuzzy Systems **15** (2007), no. 4, 578–592.
10. A. Fernandez, S. Río, V. López, A. Bawakid, M.J. del Jesus, J.M. Benítez, and F. Herrera, *Big Data with Cloud Computing: An Insight on the Computing Environment, MapReduce and Programming Frameworks*, WIREs Data Mining and Knowledge Discovery **5** (2014), no. 4, 380–409.
11. F. Herrera, *Genetic fuzzy systems: taxomony, current research trends and prospects*, Evolutionary Intelligence **1** (2008), 27–46.
12. F. Herrera, C. J. Carmona, P. González, and M. J. del Jesus, *An overview on Subgroup Discovery: Foundations and Applications*, Knowledge and Information Systems **29** (2011), no. 3, 495–525.

13. W. Kloesgen, *Explora: A Multipattern and Multistrategy Discovery Assistant*, Advances in Knowledge Discovery and Data Mining, American Association for Artificial Intelligence, 1996, pp. 249–271.
14. N. Lavrac, B. Cestnik, D. Gamberger, and P. A. Flach, *Decision Support Through Subgroup Discovery: Three Case Studies and the Lessons Learned*, Machine Learning **57** (2004), no. 1-2, 115–143.
15. A. Sathi, *Big Data Analytics: Disruptive Technologies for Changing the Game*, MC Press, 2012.
16. T. White, *Hadoop, The Definitive Guide*, O'Reilly Media, Inc., 2012.
17. S. Wrobel, *An Algorithm for Multi-relational Discovery of Subgroups*, Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery, LNAI, vol. 1263, Springer, 1997, pp. 78–87.
18. ———, *Inductive logic programming for knowledge discovery in databases*, ch. Relational Data Mining, pp. 74–101, Springer, 2001.
19. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing*, Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012.
20. P. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch, and G. Lapis, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, McGraw-Hill, 2011.