Análisis comparativo de mecanismos de minería de datos para la generación de reglas de asociación aplicables a caches de Grandes Datos

Johnny Torres[1] and Cristina L. Abad[1]

[1]Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral, Guayaquil–Ecuador
jomatorr@fiec.espol.edu.ec, cabad@fiec.espol.edu.ec

Resumen. Las minería de datos aporta una información invaluable que puede ser usada para mejorar los procesos en diversas áreas tales como la medicina, comercio, informática, entre otras. Dentro del área informática, la infraestructura de servidores web, servidores multimedia y servidores de archivos, necesita estar preparada para el manejo de grandes volúmenes de datos de manera eficiente. Uno de los retos consiste en optimizar el acceso a los datos mediante el diseño y uso de caches, de tal manera que permita minimizar el uso de los servidores y el tráfico de red. El presente trabajo evalúa algunas técnicas de minería de datos que permiten encontrar patrones de acceso a datos que pueden ser utilizados por diseñadores de algoritmos de gestión de caches para tomar decisiones de desalojo más efectivas y su consecuente mejora en el rendimiento de la caché. Se presentan los resultados de las evaluaciones de una manera sistemática, los cuales pueden ser aprovechados por otros investigadores en el diseño de caches de alto rendimiento, y los mismos muestran que se puede obtener información útil de manera eficiente para mejorar el diseño de la caché.

Palabras clave: caché, minería de datos, evaluación, eficiencia.

1 Introducción

En la era de los Grandes Datos, la eficiencia de las caches se vuelve más relevante ya que los tamaños de las mismas no crecen tan rápido como los datos que se desea cachear. De igual manera el número creciente de aplicaciones, especialmente web, pueden verse afectados por la transferencia de datos desde el servidor a los dispositivos clientes. Por esta razón, el estudio de las caches ha cobrado nuevamente importancia en la comunidad de sistemas distribuidos [1, 2].

Como solución a este problema, se busca diseñar mejores algoritmos de evicción o desalojo [2–5], que se aproximen en la medida de lo posible a un algoritmo óptimo (comúnmente denominado MIN u OPT [6, 7]) con la finalidad de aumentar la tasa de aciertos y así mejorar el rendimiento de la caché.

En el año 2005, Zhenmin Li et al. [3] propusieron el uso de técnicas de minería de datos para mejorar las predicciones de acceso a bloques de datos del disco duro y poder usar esta información para mejorar el rendimiento de una caché en sistemas

de almacenamiento de red. Sin embargo, desde ese entonces han surgido algoritmos alternativos mucho más eficientes, que resultan más adecuados en el contexto de los grandes datos.

En el presente trabajo se realiza un análisis comparativo del rendimiento de algoritmos de minería de datos que buscan patrones frecuentes y sus reglas de asociación, las cuales puedan ser utilizadas por caches para acceso a grandes datos. Nuestro análisis considera una traza (trace en inglés) de datos de la Wikipedia [8], y compara los algoritmos en cuanto al: (1) número de reglas generadas, (2) tiempo de ejecución y (3) consumo de memoria. Los resultados obtenidos muestran que se pueden obtener reglas de asociación de manera eficiente tanto en tiempo de ejecución como en consumo de la memoria, lo cual puede ser usado en combinación con las técnicas actuales en el diseño de un mejor algoritmo de manejo de caché.

Según nuestro conocimiento, este trabajo constituye el primer estudio comparativo de algoritmos de minería de datos aplicables a detectar patrones de acceso a datos en caches. Por esta razón, nuestros resultados avanzan el estado del arte al contribuir a mejorar el entendimiento de la aplicabilidad de los algoritmos de minería de patrones y reglas de asociación en el contexto del minado de patrones de acceso a datos de caches. En un futuro, esperamos utilizar nuestros resultados en el diseño de un algoritmo de desalojo más eficiente que permita mejorar la tasa de aciertos de caches de grandes datos.

El resto de este artículo se encuentra organizado de la siguiente manera. En la Sección 2 describimos las técnicas de minería de datos estudiadas. La Sección 3 presenta el problema de rendimiento de las caches y cómo este se podría mejorar mediante técnicas de minería de datos. En la Sección 4 se describen los parámetros configurables en los algoritmos estudiados y se presentan resultados de experimentos que permiten determinar el valor más apropiado de los mismos. La Sección 5 muestra los resultados de nuestra evaluación de los algoritmos estudiados en el presente trabajo. En la Sección 6 describimos los trabajos relacionados. Finalmente, en la Sección 7 presentamos las conclusiones.

2 Técnicas de minería de datos estudiadas

Las técnicas de minería de reglas de asociación entre elementos de un conjunto de datos buscan descubrir relaciones interesantes entre variables en grandes bases de datos.

Un ejemplo de minería de reglas de asociación usado por Agrawal et al. [9] muestra los patrones de compra de los clientes en los supermercados, y cómo dicha información puede ser usada por el departamento de marketing para definir nuevas promociones o la ubicación de los productos en las perchas. Similarmente, podemos obtener reglas de asociación en el acceso a páginas de sitios web como Wikipedia; un ejemplo se muestra a continuación:

$$\{ \text{Bill Clinton, George W. Bush} \} \Rightarrow \{ \text{Barak Obama} \}$$

Esta regla indica que si un usuario accede a la página web de los ex-presidentes Bill Clinton y George W. Bush, es muy probable que acceda a la página de Barak Obama. De la misma manera, se pueden encontrar reglas de asociación entre descargas de contenidos multimedia, transferencia de archivos en un sistema de almacenamiento en red, etc.

En la actualidad el ámbito de la minería de patrones y reglas de asociación se ha aplicado en áreas diversas como minería del uso de la web, detección de intrusos, detección de fraudes, producción continua, bioinformática, etc. A continuación se proporciona una breve introducción a estas técnicas.

2.1 Definición

Basado en lo propuesto por Agrawal et al. [9], el problema de reglas de asociación está definido por un conjunto de elementos $I = \{i_1, i_2, ..., i_n\}$ los cuales se agrupan en transacciones $T = \{t_1, t_2, ..., t_n\}$, donde cada transacción contiene un subconjunto de elementos de I.

Entonces, se define una regla de asociación $\{X\} \Rightarrow \{Y\}$, donde $X, Y \subseteq I$ y $X \cap Y = \emptyset$. Cada regla está compuesta de dos conjuntos disjuntos, donde I es el antecedente y Y el consecuente.

2.2 Parámetros

2.2.1 Ventana

Este parámetro define el número máximo de elementos para cada transacción.

$$ventana = len(t_i) \tag{1}$$

2.2.2 Soporte

El valor de soporte de X con respecto al conjunto de transacciones T está dado por el radio del número de transacciones que contienen el conjunto de elementos de X.

$$soporte(X) = \frac{(NumTransacciones \subseteq X)}{(NumTotalTransacciones)} \tag{2}$$

2.2.3 Confianza

El valor de confianza está definido por la proporción de transacciones que contienen $X \cup Y$ con respecto al número de transacciones que contienen X.

$$confianza(X \Rightarrow Y) = \frac{soporte(X \cup Y)}{soporte(X)} \tag{3}$$

2.2.4 Lift

Se define como el radio del soporte observado a lo esperado si X y Y fuesen independientes. En caso de tener un valor de 1 quiere decir que los subconjuntos X y Y son independientes, mientras que un valor mayor indican que están positivamente correlacionados, caso contrario negativamente correlacionados.

$$lift(X \Rightarrow Y) = \frac{soporte(X \cup Y)}{soporte(X) * soporte(Y)} \tag{4}$$

2.3 Algoritmos

En esta sección se describen algunas de las técnicas desarrolladas para descubrir reglas de asociación.

2.3.1 Apriori

Propuesto por Agrawal et al. [9], es uno de los primeros y más populares algoritmos para la minería de reglas de asociación. Este algoritmo descubre todas reglas de asociación en dos fases, usando como parámetros un valor de soporte y confianza mínimos. Difiere de algoritmos previos en la manera en que los conjuntos de elementos son considerados frecuentes y el mecanismo por el cual son generados, obteniendo así un mejor rendimiento en el orden de magnitud para un conjunto de datos grande.

2.3.2 FP Growth

Basado en una mejora del algoritmo Apriori propuesto por Han et al. [10], define una primera fase que consiste en descubrir los elementos frecuentes, y en una segunda fase en la que se generan las reglas de asociación de los elementos frecuentes encontrados, basado a parámetros de soporte y confianza mínimos. La principal diferencia con el algoritmo Apriori [9] es la implementación usada en FP Growth, la cual es más eficiente al hacer uso de un árbol de elementos frecuentes que puede ser procesado más rápidamente que la estructura de datos usada en Apriori.

2.3.3 FP Growth con medida de Lift

Usualmente las reglas de asociación se evalúan con dos medidas de relación como son soporte y confianza. Este algoritmo es una variante de FP Growth que usa una medida adicional conocida como *lift* [10]. El lift mide la correlación de los subconjuntos que forman la regla de asociación y mejora la eficiencia del algoritmo al filtrar las reglas según el grado de correlación que se defina.

2.3.4 Top-K Reglas

El algoritmo Top-K, propuesto por Fournier-Viger et al. [11], permite descubrir las mejores k reglas de asociación de un conjunto de datos. Este algoritmo es muy útil dado que evita definir manualmente el parámetro de soporte mínimo. El valor de soporte mínimo se obtiene por prueba y error, ya que es dependiente del conjunto de datos usado. Al

evitar definir manualmente el soporte mínimo se ahorra significativamente tiempo, ya que el algoritmo automatiza dicha tarea. Top-K se enfoca en obtener directamente el número *k* de reglas de asociación con el soporte más alto.

2.3.5 TNR

TNR [11] es un algoritmo aproximado para encontrar las reglas de asociación *top-k no redundantes* en un conjunto de datos. Se trata de un algoritmo aproximado porque siempre genera reglas no redundantes, pero no siempre podrían ser la top-k reglas de asociación. TNR se parametriza con un parámetro *delta*, un número positivo ≥ 0 que puede ser usado para aproximar un resultado exacto, es decir se calcula los top-k + delta reglas de asociación de los cuales se eliminan las reglas redundantes. La ventaja de este algoritmo reside en la eliminación de redundancia en las top-k reglas de asociación por consiguiente las reglas obtenidas son de mayor calidad que otros algoritmos.

2.3.6 Crecimiento de reglas (rule growth)

Este algoritmo propuesto por Fournier-Viger [12], consiste en descubrir reglas secuenciales que aparecen en bases de datos de secuencias. Una base de datos de secuencias es un conjunto de secuencias donde cada secuencia es una lista de conjuntos de datos. Un conjunto de datos es una lista de datos no ordenados. El Cuadro 1 muestra un ejemplo de una base de datos de dos secuencias, donde la secuencia S1 contiene 5 conjuntos de datos y cada conjunto diferente número de elementos. Este es un algoritmo eficiente de generación de reglas de asociación ya que considera elementos duplicados en cada secuencia para determinar los más frecuentes, a diferencia de los algoritmos anteriores que eliminan los elementos duplicados.

Cuadro 1: Ejemplo de base de datos de secuencias.

ID	Secuencia
S1	(1), (1 2 3), (1 3), (4), (3 6)
S2	(1 4), (3), (2 3), (1 5)

3 Problema

En este trabajo consideramos el problema de mejorar el rendimiento de caches de grandes datos, utilizando técnicas de minería de datos.

Sistemas como navegadores web y servidores proxy, emplean caches para almacenar respuestas previas desde los servidores, tales como páginas web e imágenes. Las caches minimizan la cantidad de información que se transmite a través de la red, dado que la información previamente almacenada en la caché usualmente puede ser usada en nuevos pedidos. Esto reduce el requerimiento de ancho de banda y procesamiento en el servidor, generando un mejor rendimiento y tiempo de respuesta para los usuarios.

Varias investigaciones [13] se han realizado para mejorar el rendimiento de caché del servidor proxy. El Cuadro 2 presenta un resumen de algunas de las políticas de reemplazo de caché más usadas.

Cuadro 2: Descripción de algunas políticas de reemplazo o desalojo de caché.

Política	**Parámetro**	**Técnica**
LFU	Número de referencias	La menos frecuentemente accesada
LRU	Tiempo de acceso	La menos recientemente accesada
GDS	Tamaño del documento S_d, Costo del documento C_d, Valor de Inflación L	El menor valor de acuerdo $P_i=C_d / S_d + L$
GDSF	Tamaño del documento S_d, Costo del documento C_d, Número de referencias recientes F_d, Correlación temporal β, Valor de Inflación L	El menor valor de acuerdo $P_i=(C_d * F_d / S_d)^{\beta} + L$

El rendimiento de las políticas de desalojo de la caché está limitado por la incapacidad de disponer de un oráculo que permita predecir futuras solicitudes de datos. A falta de dicho oráculo, los algoritmos pueden utilizar información del pasado para hacer predicciones del futuro. Con esta meta, se ha propuesto el uso de algoritmos de minería de datos para proveer de una mayor inteligencia a los algoritmos de políticas de reemplazo de caché [3].

4 Selección de parámetros

Previo al análisis comparativo de los algoritmos detallados anteriormente, realizamos una selección de los parámetros de configuración de los mismos. Para esto, probamos variando los diferentes parámetros para determinar cuál era el valor más adecuado de cada uno, dado una carga de trabajo específica: accesos a páginas, imágenes y videos de la Wikipedia [8]. Se usó como referencia el algoritmo FP Growth [10], sobre un conjunto de 10000 accesos a páginas de la Wikipedia.

Los datos de la Wikipedia fueron pre-procesados de la siguiente manera: se mapeó cada URL a un número entero único en una base de datos de transacciones o secuencias como se muestra en el Cuadro 3. Este cambio permitió utilizar implementaciones ya probadas de los algoritmos de minería de datos, disponibles en [14].

4.1 Tamaño de la ventana

El tamaño de la venta o cantidad de elementos que forma parte de una transacción t_i tiene una afectación directamente proporcional al número de reglas de asociación encontradas como lo muestra la Figura 1. Sin embargo cabe considerar que la generación

de un mayor número de reglas de asociación, dado un tamaño mayor de ventana, implica un mayor tiempo de procesamiento como lo muestra la Figura 2.

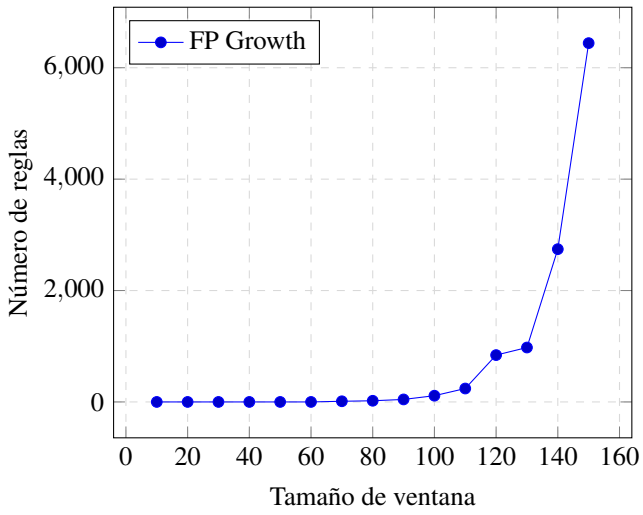

Figura 1: Número de reglas generadas en base al tamaño de la ventana.

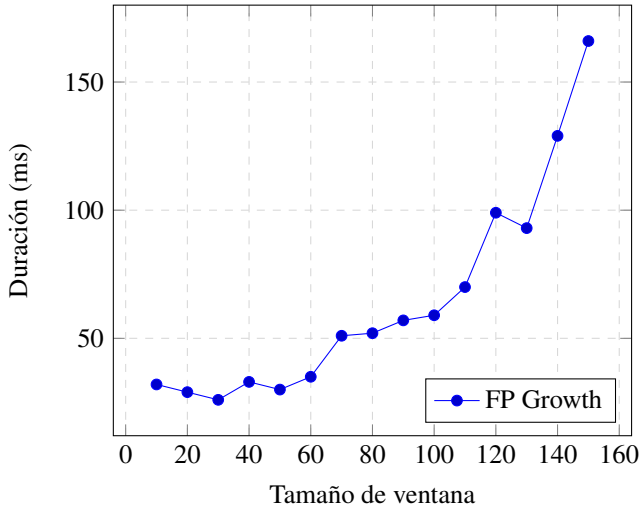

Figura 2: Rendimiento del algoritmo con respecto al tamaño de la ventana.

4.2 Soporte

Este parámetro es importante para limitar el número de reglas de asociación generadas por los algoritmos de minería de patrones frecuentes. Este valor puede definirse en un rango de 0 a 1, lo que expresa un porcentaje de soporte mínimo a usarse. La Figura 3 muestra que un valor de soporte mínimo muy bajo puede generar un número muy grande de reglas de asociación incluso para un conjunto pequeño de transacciones. En el análisis de éste y los subsiguientes parámetros se ha considerado el uso de un tamaño de ventana de 100, basado en el valor óptimo de número de reglas generadas comparado con el tiempo requerido.

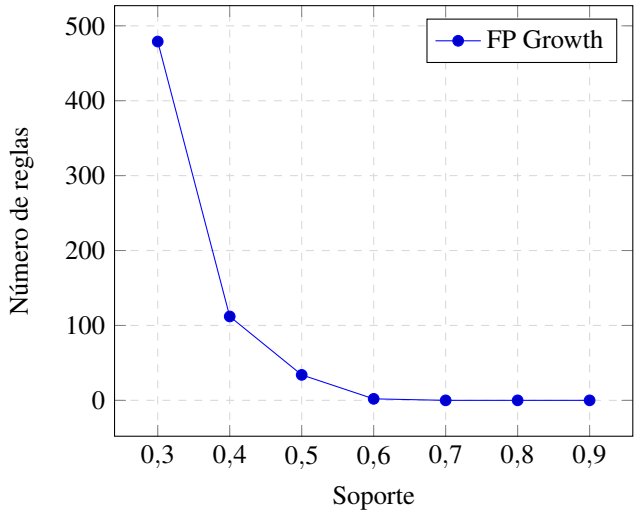

Figura 3: Número de reglas generadas con diferentes valores de soporte.

4.3 Confianza

La Figura 4 muestra que a mayor confianza el número de reglas disminuye, lo cual tiene relación al valor de soporte mínimo, ya que filtra las reglas con menor probabilidad de ocurrencia. De igual manera el valor de confianza puede ser definido en un rango de 0 a 1, indicando un porcentaje de confiabilidad mínima esperada en la generación de reglas.

4.4 Lift

Este parámetro permite establecer la correlación entre los subconjuntos en las reglas de asociación, en el que un valor mayor a 1 indica una correlación positiva. Los resultados de la Figura 5 muestran que, para el conjunto de datos usado, cuando la correlación es inversa o menor a 1 se obtiene un mayor número de reglas. Cuando las variables son

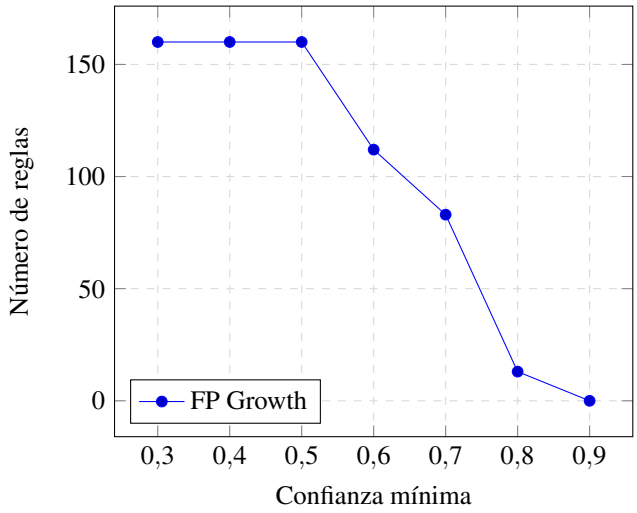

Figura 4: Número de reglas generadas con diferentes valores del parámetro de confianza.

independientes, es decir el valor de *Lift* es 1, el número de reglas disminuye. Cuando se usa correlación positiva, valor de *Lift* mayor a 1, no se generan reglas de asociación.

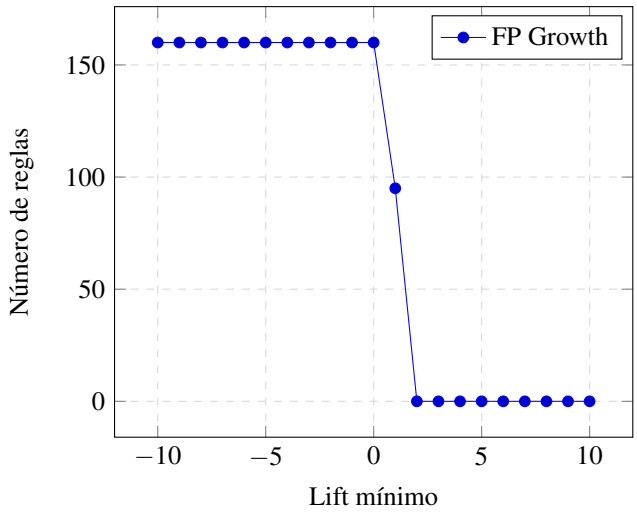

Figura 5: Número de reglas generadas para diferentes valores del parámetro lift.

4.5 Reglas encontradas

Basado en los mapeos (cuadro 3) generados durante el preprocesamiento de la traza de datos, se muestra ejemplos de reglas de asociación (cuadro 4) generadas por el algoritmo FP Growth con un soporte mínimo del 60 % y una confianza de 60 %. La primera regla indica que cada vez que un usuario solicite la URL 2, con un soporte del 60 % (aparece en 6 de las 10 transacciones) y una confiabilidad del 85 %, es probable que el usuario solicite la URL 5.

Cuadro 3: Ejemplos de mapeos usados en el preprocesamiento

Mapeo de URL a identificador único (pre-procesamiento)
2 = http://en.wikipedia.org/images/wiki-en.png
5 = http://en.wikipedia.org/w/index.php?...
44 = http://en.wikipedia.org/w/index.php?...
73 = http://meta.wikimedia.org/w/index.php?...
79 = http://upload.wikimedia.org/wikipedia/en/1/18/Monobook-bullet.png
81 = http://en.wikipedia.org/skins-1.5/monobook/user.gif

Cuadro 4: Ejemplos de reglas de asociación generadas con el algoritmo FP Growth.

Reglas generadas	**Soporte**	**Confianza**
$\{2\} \Rightarrow \{5\}$	60 %	85 %
$\{2, 79\} \Rightarrow \{5\}$	60 %	100 %
$\{5, 81\} \Rightarrow \{44\}$	60 %	100 %
$\{5, 79, 81\} \Rightarrow \{44\ 73\}$	60 %	100 %

5 Evaluación de los algoritmos

En esta sección, comparamos varios algoritmos en cuanto al número de reglas generadas, su tiempo de ejecución y su consumo de memoria. El dataset analizado es una traza de accesos a páginas, imágenes y videos de la Wikipedia [8]. Para la evaluación del rendimiento se ha considerado traza de datos de 100000 accesos a URLs de Wikipedia. Los experimentos se realizaron en un computador con un procesador 2.7 GHz Intel Core i7 ejecutando el sistema operativo OS X y 8 GB de memoria RAM libre.

Los algoritmos que se han comparado son: FP Growth [10], Rule Growth [12], Top-K [11] y TNR [11]. De los algoritmos descritos en la Sección 2, no se consideró el Apriori [9] ni el FP Growth con Lift [10] debido a que el primero ya se sabe que es menos eficiente que FP Growth [10], mientras que (para el caso del dataset analizado) la Figura 5 demuestra que el algoritmo FP Growth no es sensible al valor de Lift.

5.1 Número de reglas generadas

El número de reglas generadas tiende a disminuir mientras más grande es el tamaño del conjunto de datos, debido a que el soporte de un patrón es validado para un mayor

número de transacciones. Una excepción a esto son los algoritmos que buscan generar un número fijo de reglas (Top-K y TNR). En el caso de estos algoritmos, los configuramos para que generen 5 000 reglas; este valor lo seleccionamos en base al valor de convergencia observado para los algoritmos FP Grwoth y Rule Growth.

La Figura 6 considera agrupaciones por número de solicitudes de archivos web de Wikipedia; dichas solitudes se agrupan en transacciones de tamaño fijo de 150 solicitudes web. Cabe recalcar que el algoritmo TNR genera reglas no redundantes.

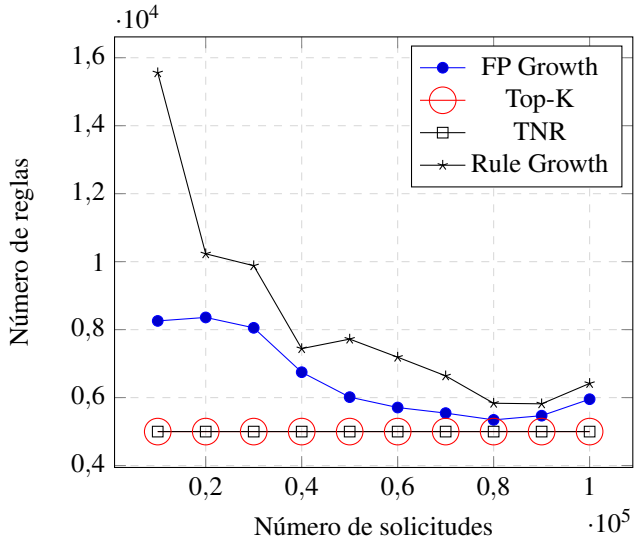

Figura 6: Evaluación del número de reglas generadas.

5.2 Tiempo de ejecución

En cuanto al tiempo de ejecución, el algoritmo FP Growth muestra un mejor rendimiento como se aprecia en la Figura 7. Esto se debe a que los otros algoritmos realizan un procesamiento adicional para obtener las reglas de asociación; especialmente TNR, que valida que no existan reglas no redundantes.

5.3 Consumo de memoria

El consumo de memoria se lo muestra la Figura 8. Mientras más grande es el tamaño del conjunto de datos mayor será el requerimiento de memoria, lo cual es un factor a considerar en el diseño de la caché. Cabe notar que Rule Growth es el algoritmo que más memoria consume debido a que el número de reglas de asociación en una base de datos secuencial es mayor que una base de datos transaccional. Este resultado nos permite concluir que Rule Growth no es un algoritmo adecuado para caches de Grandes Datos ya que no resulta escalable cuando la cantidad de transacciones es grande.

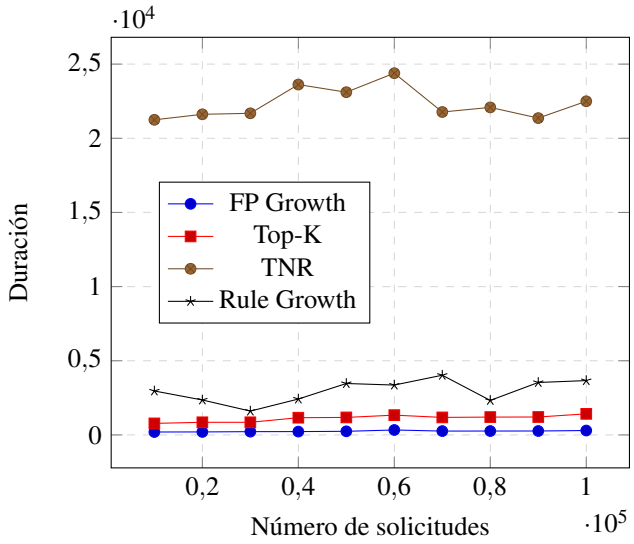

Figura 7: Evaluación del tiempo de ejecución.

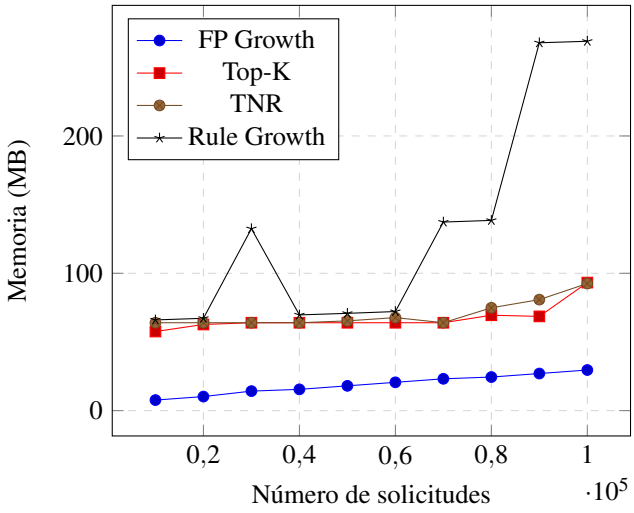

Figura 8: Evaluación del uso de memoria.

6 Trabajos relacionados

En los últimos años se han realizado investigaciones en el área de inteligencia computacional para mejorar el rendimiento de la caché. Algunas de las técnicas investigadas se basan en enfoques diferentes a la minería de patrones, tales como clasificadores Naive-

Bayes [15], redes neuronales [16], árboles de decisión rápidos [17], sistemas híbridos neuronales-difusos [18]. Dichas investigaciones proponen la adaptación del algoritmo a diferentes escenarios utilizando técnicas de machine learning, lo cual es un avance muy interesante e importante en la optimización de caches. El presente trabajo aporta con otra dimensión a los trabajos anteriormente citados, ya que permite que otros investigadores puedan considerar los algoritmos de minería de datos estudiados, para sus diseños de algoritmos de desalojo de caché más inteligentes.

7 Conclusiones

En base a los experimentos realizados, el cuadro 5 muestra un resumen comparativo de las métricas evaluadas para los algoritmos de minería de reglas de asociación, siendo FP Growth el más eficiente tanto en uso de la memoria así como en el tiempo de ejecución. La segunda columna indica si las reglas generadas son fijas o variables, mientras que la tercera columna especifica si las reglas son redundantes. Para el número de reglas, consumo de memoria, y tiempo de ejecución se ha tomado el promedio para los diferentes tamaños de ventana usados en los experimentos.

Cuadro 5: Comparación de algoritmos de minería de reglas de asociación.

Algoritmo	**Tipo**	**Red.**	**Num. Reglas**	**Memoria (MB)**	**Tiempo (Seg.)**
FP Growth	Variable	Si	6610	17.8	245.2
Top K	Fijo	Si	5000	64.3	1083.2
TNR	Fijo	No	5000	67.6	22319.0
Rule Growth	Variable	Si	8479	113.6	2895.8

El número de reglas de asociación generadas que se necesiten dependerá del tamaño de la caché, considerando otros parámetros como tamaño y costo del documento, por lo que (en esos casos) las técnicas Top-K y TNR no se ajustan adecuadamente para el diseño de una caché. Por otro lado, Rule Growth resultó no ser escalable en cuanto a su consumo de memoria, por lo que no sería recomendable para caches de Grandes Datos. En base a los resultados presentados, podemos concluir que la técnica FP Growth es la técnica que más se aproxima a una solución óptima desde el punto de vista de rendimiento, escalabilidad, y flexibilidad para el diseño de un algoritmo de manejo de caché. Sin embargo, si restringir el número de reglas es deseable, el algoritmo a seleccionar sería Top-K ya que TNR toma más tiempo en ejecutarse debido al filtrado de reglas redundantes.

En un futuro trabajo se plantea el uso de técnicas de minería de reglas de asociación en el diseño de un algoritmo eficiente para manejo de la caché, de modo que permita obtener un rendimiento mayor al que muestran las técnicas actuales. Las mejoras esperadas consisten en obtener un mayor radio de aciertos basado en un política de evicción que considere reglas de asociación encontradas en los datos.

Referencias

[1] X. Yan, J. Han, and R. Afshar, "Clospan: Mining closed sequential patterns in large datasets," in *In SDM*, 2003, pp. 166–177.

[2] P. Bangar and K. Singh, "Investigation and performance improvement of web cache recommender system," in *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on*, Feb 2015, pp. 585–589.

[3] Z. Li, Z. Chen, and Y. Zhou, "Mining block correlations to improve storage performance," *TOS*, vol. 1, no. 2, May 2005.

[4] P. J. Benadit and F. S. Francis, "Improving the performance of a proxy cache using very fast decision tree classifier," *Procedia Computer Science*, vol. 48, pp. 304 – 312, 2015, international Conference on Computer, Communication and Convergence (ICCC 2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187705091500695X

[5] P. J. Benadit, F. S. Francis, and U. Muruganantham, "Improving the performance of a proxy cache using tree augmented naive bayes classifier," *Procedia Computer Science*, vol. 46, 2015.

[6] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Syst. J.*, vol. 9, no. 2, Jun. 1970.

[7] B. V. Roy, "A short proof of optimality for the MIN cache replacement algorithm," *Information Processing Letters*, vol. 102, no. 2, 2007.

[8] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, July 2009, http://www.globule.org/publi/WWADH_comnet2009.html.

[9] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[10] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[11] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Mining top-k association rules," in *Advances in Artificial Intelligence*. Springer, 2012, pp. 61–73.

[12] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, "Rulegrowth: mining sequential rules common to several sequences by pattern-growth," in *Proceedings of the 2011 ACM symposium on applied computing*. ACM, 2011, pp. 956–961.

[13] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[14] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "Spmf: a java open-source pattern mining library," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[15] W. Ali, S. M. Shamsuddin, and A. S. Ismail, "Intelligent naive bayes-based approaches for web proxy caching," *Knowledge-Based Systems*, vol. 31, pp. 162–175, 2012.

[16] S. Romano and H. ElAarag, "A neural network proxy cache replacement strategy and its implementation in the squid proxy server," *Neural computing and Applications*, vol. 20, no. 1, pp. 59–78, 2011.

[17] M. T. Cazzolato and M. X. Ribeiro, "Classifying high-speed data streams using statistical decision trees," *Journal of Information and Data Management*, vol. 5, no. 1, p. 84, 2014.

[18] W. A. Ahmed and S. M. Shamsuddin, "Neuro-fuzzy system in partitioned client-side web cache," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 715–14 725, 2011.