Descubrimiento de Patrones Secuenciales con Factores de Cantidad

Karim Guevara Puente de la Vega
Universidad Católica de Santa María
Universidad Nacional de San Agustín
kguevara72@gmail.com

Cesar Beltrán Castañón
Pontificia Universidad Católica
cbeltran@gmail.com

Resumen: *La minería de datos es un área de investigación que ha sido motivada por la necesidad que desde hace años atrás surge en las empresas acerca de la forma como éstas toman las decisiones y sobre qué información se soportan en estas decisiones. Es así que, la Minería de patrones secuenciales, surge a causa de la necesidad de obtener patrones que se repiten en varias transacciones en una base de datos transaccional, los cuales se relacionan con el tiempo u otro tipo de secuencia. En este artículo se presenta la propuesta de una nueva técnica para el descubrimiento de patrones secuencial a partir de una base de datos de secuencias, en donde los patrones no solo brinden información sobre la forma como éstos se relacionan en el tiempo, sino también, que en el propio proceso de minería se incluya los factores de cantidad asociados a cada uno de los ítems que forman parte de una secuencia, y como resultado de este proceso se pueda obtener información relativa a cómo se relacionan estos ítems respecto a las cantidades asociadas.*

Palabras clave: descubrimiento de patrones secuenciales, minería de patrones secuenciales, patrones frecuentes, minería de datos mezclados.

Abstract: *Data mining is a research area that has been motivated by the need arises for years back in business about how they make decisions about what information and support in these decisions. Thus, Mining sequential patterns, it arises because of the need for repeating patterns in multiple transactions in a transactional database, which is related to time or other sequence. The proposal of a new technique for the discovery of sequential patterns is presented from a database of sequences, where patterns not only provide information on how these are related in time, but in this also, that in the process of mining the factors of quantity associated with each of the items that are part of a sequence is included, and as a result of this process is to obtain information on how these items relate regarding the associated quantities.*

Keywords: discovery of sequential patterns, sequential patterns mining, frequent patterns mining mixed data.

1. Introducción

La Minería de Datos es el proceso de extraer conocimiento no explícito de bases de datos. Su objetivo es descubrir situaciones anómalas e interesantes, tendencias, patrones y secuencias en los datos [5]. La minería de patrones secuenciales es el proceso por el cual se obtienen las relaciones entre ocurrencias secuenciales, para encontrar si existe algún orden específico en el que ocurren estos eventos. En relación con esta área de estudio hay muchas investigaciones realizadas, en todas ellas se hace uso de la restricción de soporte mínimo. Algunas incluyen otras restricciones como por ejemplo el intervalo de tiempo en el que se requiere que sucedan los eventos, también el uso de taxonomías definidas por el usuario, y el hecho de permitir que los ítems de una secuencia no necesariamente deben de haber ocurrido en una sola transacción, sino podrían estar en dos o más, siempre y cuando sus tiempos de cada una de estas transacciones esté dentro de alguna pequeña ventana de tiempo determinada por el usuario.

Además, los algoritmos de minería de patrones secuenciales previos tratan los patrones secuenciales de una manera uniforme, a pesar de que estos patrones individualmente en una secuencia pueden tener diferencias importantes como por ejemplo la cantidad asociada a cada ítem que conforman cada patrón.

Por lo anterior, en el presente paper se presenta una propuesta en la que se pretende explotar estas relaciones intrínsecas de los patrones secuenciales, en este caso específico la relación respecto a la cantidad de cada uno de los items. La inclusión de este aspecto en la minería de patrones secuenciales, podrá permitirnos obtener un conjunto de patrones secuenciales que no sólo sean frecuentes sino que también nos permita conocer la forma como estas cantidades asociadas a cada ítem que está incluido en un patrón secuencial frecuente se relaciona. Por lo que la inclusión de la restricción de cantidad dentro del proceso de extracción de los patrones secuenciales frecuentes nos podría proporcionar información mucho más significativa.

El artículo está organizado de la siguiente manera: la Sección 2 trata sobre los trabajos previos. La Sección 3 da una descripción del problema. La Sección 4 introduce la técnica propuesta. La sección 5 muestra los experimentos y resultados. Las conclusiones y trabajos futuros se muestran en la sección 6 y, finalmente, las referencias en la sección 7.

2. Trabajos Previos

Las técnicas de descubrimiento de reglas de asociación son básicamente booleanas debido a que se descartan las cantidades de los ítems comprados y sólo se presta atención a si algo fue comprado o no. Una excepción notable es el trabajo de Agrawal en [2], donde se referencia el problema de la minería de reglas de asociación cuantitativas.

Otra área de estudio importante es la minería de patrones secuenciales [6] que consiste en la extracción de patrones que se repiten en varias transacciones en una base de datos transaccional, los cuales se relacionan con el tiempo u otro tipo de secuencia.

El problema de la minería de patrones secuenciales fue introducido por Agrawal [7] poniendo como ejemplo las rentas típicas que realizan los clientes en una tienda de alquiler de videos. Los clientes suelen alquilar "Star Wars", luego "Empire Strikes Back" y después "Return of the Jedi". Todas estas rentas no necesariamente tendrían que haber sido hechas de forma consecutiva, es decir, podría haber clientes que alquilaron cualquier otro video en medio de la secuencia anterior, por lo que estas secuencias de transacciones también encajarían en el mismo patrón.

Las investigaciones sobre minería de patrones secuenciales se basan en eventos ocurridos ordenadamente en el tiempo. Este problema fue introducido por Agrawal y Srikant [7], basándose en el estudio de secuencias de compras de clientes.

"Dado un conjunto de secuencias donde cada una de ellas consiste en una lista de eventos, y cada uno de los eventos consiste en una lista de items dado un nivel de soporte mínimo (min_sup). La tarea de extracción de patrones secuenciales encuentra las subsecuencias frecuentes; es decir, cada subsecuencia cuya frecuencia de ocurrencia en el conjunto de secuencia es no menor a min_sup."

La mayoría de los algoritmos implementados para la extracción de secuencias frecuentes utilizan tres tipos diferentes de enfoques de acuerdo con la forma de evaluar el soporte de los patrones secuenciales candidatos:

El primer grupo de algoritmos se basan en la propiedad Apriori, introducida por Agrawal and Srikant en la minería de reglas de asociación [1]. Esta propiedad plantea que cualquier subpatrón de un patrón frecuente también es frecuente, permitiendo podar las secuencias candidatas durante el proceso de generación de candidatos. Basados en esta heurística se propusieron algoritmos como el AprioriAll y el AprioriSome [7]. La diferencia sustancial entre estos dos algoritmos es que el AprioriAll genera las candidatas a partir de todas las secuencias grandes encontradas pero que podrían no ser maximales. Sin embargo, AprioriSome solo realiza el conteo de aquellas secuencias grandes pero que son maximales, reduciendo de este modo el espacio de búsqueda de los patrones.

En trabajos posteriores los mismos autores proponen el algoritmo *GSP (Generalization Sequential Patterns),* basado también en la técnica Apriori y superando a los anteriores en 20 magnitudes de tiempo. Hasta este momento, los algoritmos que habían sido propuestos para minería de patrones secuenciales se enfocaban en la obtención de patrones tomando sólo en consideración el soporte mínimo dado por el usuario. Pero estos patrones podían encajar en transacciones que se habían dado en intervalos de tiempo muy lejano, lo que no era conveniente para los propósitos de la minería. Así que, en este trabajo se propone la idea de que además del soporte mínimo, el usuario podía estar en la posibilidad de especificar su interés de obtener patrones que encajen en transacciones que se hayan dado en determinados periodos de tiempo, y esto lo realizan a partir de la inclusión de restricciones del máximo y mínimo de la distancia, del tamaño de la ventana en el cual se dan las secuencias y las relaciones de herencia – taxonomías, por las cuales encuentran las relaciones cruzadas a través de una jerarquía.

En estos algoritmos basados en el principio de Apriori, el mayor esfuerzo se centró en desarrollar estructuras específicas que permitieran representar los patrones secuenciales candidatos y de esta manera realizar las operaciones de conteo del soporte con mayor rapidez.

El segundo grupo lo forman los algoritmos que tratan de reducir el tamaño del conjunto de datos explorados, por medio de la ejecución de tareas de proyección de la base datos inicial y la obtención de patrones sin involucrar un proceso de generación de candidatos. Utilizando esta técnica y bajo el enfoque divide y vencerás, Pei y Han proponen el algoritmo FreeSpan (*Frecuent Pattern-Project Sequential Pattern mining*) [8] y el PrefixSpan (*Prefix-projected Sequential Pattern mining*) [9]. En estos algoritmos, la base de datos de secuencias es proyectada recursivamente en un conjunto de pequeñas bases de datos a partir de las cuales los fragmentos de las sub secuencias van creciendo basado en el conjunto actual de secuencias frecuentes, en donde extraen los patrones.

Han [8] muestra que FreeSpan extrae el conjunto completo de patrones y es más eficiente y considerablemente más rápido que el algoritmo GSP. Sin embargo, una subsecuencia puede ser generada por las combinaciones de sub cadenas en una secuencia, mientras que la proyección en FreeSpan debe seguir la secuencia en la base de datos inicial sin reducir la longitud. Además, es muy costoso el hecho de que el crecimiento de una sub secuencia sea explorada en cualquier punto de la división dentro de una secuencia candidata. Como una alternativa a este problema, Pei [9] propone PrefixSpan. La idea general es examinar solamente los prefijos de las sub secuencias y proyectar solamente sus correspondientes sub secuencias postfijas dentro de bases de datos proyectadas. En cada una de estas bases de datos proyectadas, los patrones secuenciales expandidos por la exploración solamente de los patrones locales frecuentes. PrefixSpan extrae el conjunto completo de patrones y su eficiencia y ejecución son considerablemente mejor tanto de GSP y FreeSpan.

El tercer grupo está formado por algoritmos que mantienen en memoria solamente la información necesaria para la evaluación del soporte. Estos algoritmos se basan en las llamadas listas de ocurrencia que contienen la descripción de la ubicación donde ocurren los patrones en la base de datos. Bajo este enfoque, Zaki en el año 1998 propone el algoritmo SPADE (*Sequential Pattern Discovery using Equivalence classes*) [10] donde introduce la técnica de transformación de la base de datos a formato vertical, además que a diferencia que los algoritmos basados en Apriori, no realiza múltiples pasadas sobre la base de datos, y puede extraer todas las secuencias frecuentes en solo tres pasadas. Esto se debe a la incorporación de nuevas técnicas y conceptos como son la lista de identificadores (*id-list*) con formato vertical que está asociada a las secuencias. En estas listas por medio de uniones temporales se puede generar las secuencias frecuentes. Utiliza también el enfoque basado en retícula

para descomponer el espacio de búsqueda en clases pequeñas, que pueden ser procesadas independientemente en la memoria principal. Utiliza también tanto la búsqueda en amplitud como en profundidad para encontrar las secuencias frecuentes dentro de cada clase.

Además de las técnicas mencionadas antes, en [11] se propone el primer algoritmo en implementar la idea del indexado de memoria Memisp (*MEMory Indexing for Sequential Pattern mining*). La idea central de Memisp es utilizar la memoria tanto para las secuencias de datos como para los índices en el proceso de minería y aplicar una estrategia de búsqueda e indexación para encontrar todas las secuencias frecuentes a partir de una secuencia de datos en la memoria, secuencias que fueron leídas de la base de datos en un primer recorrido. Solamente requiere un recorrido sobre la base de datos, a lo sumo dos para bases de datos demasiado grandes. Además evita la generación de candidatos y la proyección de base de datos, pero presenta como desventaja una alta utilización de CPU y de memoria.

3. Descripción del Problema

Una secuencia *s*, denotada por $<e_1e_2... e_n>$, es un conjunto ordenado de *n* elementos, donde cada elemento e_i es un conjunto de objetos (*itemset*). Un *itemset*, que se denota por $(x_1[c_1], x_2[c_2],..., x_q[c_q])$, es un conjunto no vacío de elementos *q*, donde cada elemento x_j es un ítem y está representado por un literal, y c_1 es la cantidad asociada al ítem x_1 que está representado por un número entre corchetes. Sin pérdida de generalidad, los objetos de un elemento se supone que se encuentran en orden lexicográfico por el literal. El tamaño de la secuencia *s*, denotado por |*s*|, es el número total de objetos de todos los elementos de *s*. Una secuencia *s* es una *k-sequence*, si $|s|=k$.

Por ejemplo, *<(a[5])(c[2])(a[1])>*, *<(a[2],c[4])(a[3])>* y *<(b[2])(a[2],e[3])>* son todas *3-sequences* . Una secuencia $s = <e_1e_2... e_n>$ es una subsecuencia de otra secuencia de $s'=<e_1'e_2'... e_m'>$ si existen $1 \le i_1<i_2 <\dots < i_n \le m$ tal que $e_1 \subseteq e_{i1}'$, $e_2 \subseteq e_{i2}'$,..., y $e_n \subseteq e_{in}'$. La secuencia s' contiene la secuencia *s* si *s* es una sub-secuencia de *s'*.

Asi mismo, *<(b,c)(c)(a,c,e)>* contiene *<(b)(a,e)>* en donde las cantidades pueden ser diferentes.

El soporte (*sup*) de un patrón secuencial X es definido como el porcentaje sobre la fracción de registros que contiene X del total de número de registros en la base de datos. El contador por cada ítem es incrementado en uno cada vez que el ítem es encontrado en diferentes transacciones en la base de datos durante el proceso de escaneado. Esto quiere decir que el contador de soporte no tiene en cuenta la *cantidad del ítem*. Por ejemplo, en una transacción un cliente compra tres botellas de cerveza, pero solamente se incrementa el número del contador de soporte {cerveza} por uno; en otras palabras si una transacción contiene un ítem, entonces, el contador de soporte de ese ítem sólo se incrementa en uno.

Cada secuencia en la base de datos se conoce como una secuencia de datos. El soporte de la secuencia *s*, es denotado como *s.sup*, y representa el número de secuencias de datos que contienen a *s* dividido por el número total de secuencias de datos que hay en la base de datos. *Minsup* es el umbral de soporte mínimo especificado por el usuario. Una secuencia *s* es una secuencia frecuente, también llamado un patrón secuencial, si $s.sup \ge minsup$.

Entonces dados el valor del *minsup* y la base de datos de secuencias, el problema de la minería de patrones secuenciales es descubrir el conjunto de todos los patrones secuenciales cuyos si $s.sup \ge minsup$.

Definición: dado un patrón ρ y un ítem frecuente *x* en la base de datos de secuencia, ρ' es un:

- **Patrón Tipo-1:** si puede formarse agregando al itemset *x* como un nuevo elemento de ρ.
- **Patrón Tipo-2:** si puede ser formado por la extensión de los últimos elementos de ρ con *x*.

El ítem *x* es denominado ***stem*** del patrón secuencial ρ', y ρ es el patrón prefijo de ρ'.

Sea, entonces, la siguiente base de datos de secuencias de la figura 1, la cual incluye cantidades para los items y que tiene 6 secuencias de datos.

Secuencias
C1 = <(a[1],d[2]) (b[3],c[4]) (a[3],e[2])>
C2 = <(d[2],g[1]) (c[5],f[3]) (b[2],d[1])>
C3 = <(a[5],c[3]) (d[2]) (f[2]) (b[3])>
C4 = <(a[4],b[2],c[3],d[1]) (a[3]) (b[4])>
C5 = <(b[3],c[2],d[1]) (a[3],c[2],e[2]) (a[4])>
C6 = <(b[4],c[3]) (c[2]) (a[1],c[2],e[3])>

Figura 1. Base de datos de secuencias

Considere la secuencia *C6*, la cual consta de tres elementos, el primero tiene los objetos *b* y *c*, el segundo tiene el objeto *c*, y el tercero tiene los objetos *a, c*, y *e*. Por tanto, el soporte de *<(b)(a)>* es 4/6 ya que todas las secuencias de datos a excepción de *C2* y *C3* contienen a *<(b)(a)>*. La secuencia *<(a,d)(a)>* es una subsecuencia tanto de C1 y C4; y por tanto, *<(a,d)(a)>.sup = 2/6.*

Dado el patrón *<(a)>* y el ítem frecuente *b*, se obtiene el patrón tipo-1 *<(a)(b)>* agregando *(b)* a *<(a)>,* y el patrón de tipo-2 *<(a,b)>* por la extensión de *<(a)>* con *b*.

Así mismo, *<(a)>* es el *prefijo del patrón (P_pat),* y *b* es el *stem* de ambos: *<(a)(b)>* y *<(a,b)>*.

Observe que la secuencia *null,* denotada por *<>,* es el *P_pat* de cualquier *1-secuencia* frecuente. Por lo tanto, una *k-secuencia* frecuente es como un patrón tipo-1 o tipo-2 de una *(k-1)-secuencia* frecuente.

Considere la secuencia *C6*, la cual consta de tres elementos, el primero tiene los objetos *b* y *c*, el segundo tiene el objeto *c*, y el tercero tiene los objetos *a, c*, y *e*. Por tanto, el soporte de *<(b)(a)>* es 4/6 ya que todas las secuencias de datos a excepción de *C2* y *C3* contienen a *<(b)(a)>*. La secuencia *<(a,d)(a)>* es una subsecuencia tanto de C1 y C4; y por tanto, *<(a,d)(a)>.sup = 2/6.*

4. Propuesta del algoritmo para el descubrimiento de patrones secuenciales con factores de cantidad – MSP-QF

El algoritmo de Minería de patrones secuenciales con factores de cantidad, surge de la necesidad de descubrir, a partir de una base de datos de secuencias, el conjunto de patrones secuenciales que incluyan las cantidades asociadas a cada uno de los ítems que son parte de las transacciones y de las secuencias de la base de datos, ya que al tener esta información adicional se puede conocer con mayor precisión no solamente cuál es la relación que existe entre los diversos ítems que participan en las transacciones de una secuencia, sino también como es la relación respecto a la cantidad de estos ítems.

El algoritmo MSP-QF se basa en la idea del uso de prefijos, y la creación de índices a partir de a base de datos de secuencias o de otros índices, en donde de forma recursiva realiza la búsqueda de los patrones frecuentes. Como resultado de la exploración de un determinado índice, menos y más cortas secuencias de datos necesitan ser procesadas, mientras que los patrones que se van encontrando se hacen más largos.

Además, si la base de datos de secuencia es muy grande se utilizan técnicas de particionamiento de forma que el algoritmo es aplicado a cada una de las particiones como si se tratará de una base de datos de menor tamaño.

4.1. Procedimiento del algoritmo MSP-QF

A continuación, se describen los pasos del algoritmo propuesto.

Paso 1: Particionamiento y escaneo de la base de datos de secuencias: En esta fase, dependiendo del tamaño de la base de datos, se procede a particionar la base de datos (DB) y a escanear cada una de las particiones de forma independiente. Para cada partición, se construyen las secuencias y se almacenan en la estructura *DBSeq*. A la vez, se indexa los ítems en donde se almacena el soporte de cada uno de ellos, el mismo que es hallado durante el escaneo de la DB.

Paso 2: Del índice de ítems, se filtran aquellos que son frecuentes, es decir, cuyo soporte sea mayor o igual al m*inSup* determinado por el usuario. Todos estos ítems vienen a constituir secuencias de |s|=1, por tanto forman el conjunto de *1-secuencia*. Para todas estas secuencias frecuentes se procede a discretiza las cantidades asociadas a cada ítem, y se guardan en el conjunto de patrones frecuentes.

Paso 3: Para cada uno de los patrones frecuentes *P*, hallados en el paso 2 o como resultado del paso 4, se construye el índice *P_idx*, con entradas *(ptr_ds, pos)*, donde *ptr_ds* hace referencia a una secuencia de la DB en donde aparece el patrón *P*, y *pos* a su vez es también un par (*posItemSet, posItem*). *posItemSet* es la posición del *itemset* de la secuencia y *posItem* la posición del ítem en el itemSet de la secuencia donde aparece el patrón. Los valores de *pos* permiten que las siguientes exploraciones solo se realicen a partir de estas posiciones en una secuencia determinada y se ejecuta el paso 4.

Paso 4: A partir de *P* y el índice correspondiente *P_idx*, se hallan los stems de tipo-1 o tipo-2, considerando solo aquellos ítems de las secuencias referidas en *P_idx* y los valores de *pos*. Conforme se hallan los stems se determinan sus soportes, y además se agrega a la lista de cantidades del ítem perteneciente al stem la cantidad referida en ítem de la secuencia de la DB. La información de los stems y sus cantidades se almacena en otro índice de stems.

Se repite este paso, hasta que ya no se encuentren más stems.

Paso 5: Cuando ya no hay más stems que hallar, se filtran del índice de stems del patrón *P* todos aquellos que sean frecuentes. Para todos los stems (secuencias) frecuentes se discretizan las cantidades que fueron asociadas a cada ítem y se almacenan en el conjunto de patrones frecuentes, luego de verificar que el patrón frecuente encontrado recientemente ya no haya sido agregado antes a este conjunto como resultado de haber aplicado el algoritmo a una partición de la base de datos que fue procesada con anterioridad. En caso exista, se aplica la intersección del conjunto de cantidades asociadas a la secuencia que esta almacenada como patrón frecuente y el conjunto de cantidades del patrón frecuente recientemente hallado; de lo contrario simplemente es agregado al conjunto.

Luego se procede a realizar los 3, 4 y 5 de forma recursiva con cada uno de los stems frecuentes.

Función *Discretizar:* Esta función se encarga de discretizar el conjunto de cantidades asociadas a un ítem, a un intervalo de valores calculados a partir de la media aritmética y la desviación estándar del conjunto de valores almacenados para ese ítem. Por ejemplo, la cantidad discretizada para el ítem <*(a)*> en todas las secuencias de la base de datos de la figura 1 cuyos valores son: 1,5,4,3,1; sería el intervalo formado por la media ± desviación estándar: [2.8±1.6]

4.2. Especificación del algoritmo MSP-QF

A continuación se observa la especificación del algoritmo propuesto MSP-QF.

Algoritmo *MSP-QF*

Entrada:

DB = base de datos de secuencias

minsup = soporte minimo

particion = número de secuencias incluidas en cada una de las particiones

Salida: el conjunto de todos los patrones secuenciales con factores de cantidad.

Procedimiento:

1. Particionar la *DB*

2. Cada partición de la *DB* escanerla en la memoria principal y:

   (i) construir las secuencias y almacenarlas en la estructura *DBSeq*.

   (ii) indexar los ítems.

   (iii) determinar el soporte de cada item.

   (iv) asociar las cantidades de cada ítem de una secuencia a la lista de cantidades del ítem en el índice.

3. Encontrar el conjunto de ítems frecuentes

4. Para cada ítem frecuente *x*,

(i) formar el patrón secuencial ρ = <(*x*)>

(ii) llamar a *Discretizar*(ρ) para discretizar el conjunto de cantidades asociadas a cada ítem de ρ.

(iii) almacenar ρ en el conjunto de patrones frecuentes.

(iv) llamar *Indexar*(*x*, <>, *DBSeq*) para construir el índice ρ-*idx*.

(v) llamar *Minería*(ρ, ρ-*idx*) para obtener patrones a partir del índice ρ-*idx*..

Subrutina *Indexar* (*x*, ρ, *set_Seq*)

Parametros:

x = un stem de tipo-1 o tipo-2;

ρ = el patrón prefijo (ρ-*pat*);

set_Seq = el conjunto de secuencias de datos

/* Si *set_Seq* es un índice, entonces cada secuencia de datos en el índice esta referenciada por el elemento ***ptr_ds***¸ que se encuentra en la entrada formada por **(ptr_ds, pos**) del índice */

Salida: índice ρ'-*idx*, donde ρ' representa el patrón formado por el stem *x* y el patrón prefijo ρ-*pat*.

Procedimiento:

1. Para cada secuencia de datos *ds* del *set_Seq*,

(i) Si *set_Seq* = *DBSeq* la *pos_inicial* = *0*, sino *pos_inicial* = *pos*.

(ii) Buscar el *stem* en cada secuencia *ds* a partir de la position (*pos_inicial* + 1),

1. Si el stem *x* se encuentra en la posición ***pos*** en *ds*, entonces insertar un par (***ptr_ds***, ***pos***) en el índice ρ'-*idx,* donde ***ptr_ds*** referencia a *ds*.
2. Si el stems *x* es igual al ítem *x'* de la secuencia *ds,* se agrega la cantidad *q* asociada al ítem *x',* a la lista de cantidades vinculadas a *x*.

2. Retornar el índice ρ'-*idx*.

Subrutina *Mineria*(ρ,ρ-*idx*)

Parametros:

ρ = una patrón;

ρ-*idx* = un indice.

Procedimiento:

1. Para cada secuencia de datos *ds* referenciada por el elemento *ptr_ds* de una entrada (*ptr_ds*, *pos*) en ρ-*idx*,

(i) Empezando en la posición (*pos+1*) hasta |*ds*| en *ds*, determinar los stems potenciales e incrementar el soporte de cada stem potencial en uno.

2. Filtrar aquellos stems que tengan un soporte suficientemente grande.

3. Para cada stem *x* del punto anterior,

(i) formar el patrón secuencial ρ' con patrón prefijo ρ-*pat y* el stem *x*.

(ii) llamar a *Discretizar*(ρ') para discretizar las cantidades asociadas a los ítems de ρ'.

(iii) llamar *Indexar*(*x*, ρ, ρ-*idx*) para construir el índice el índce ρ'-*idx*.

(iv) llamar *Mineria*(ρ', ρ'-*idx*) para obtener patrones a partir del índice ρ'-*idx*

Subrutina *Discretizar*(ρ)

Parametros:

ρ = una patrón que es una secuencia;

Salida: la media aritmética y la desviación estándar de las cantidades asociadas a cada ítem del patrón ρ.

Procedimiento:

1. Para cada Itemset $\gamma \in \rho$

a) Para cada ítem x $\in \gamma$

(i) Calcular la media aritmética y la desviación estándar del conjunto de cantidades asociadas al ítem *x*

(ii) Almacenar la media aritmética y la desviación estándar en ρ.

5. Experimentos y Resultados

Los experimentos realizados para probar la técnica propuesta, se ejecutaron en dos escenarios diferentes, los cuales describimos a continuación.

5.1. Escenario 1: Datos Reales

La técnica fue aplicada en el análisis de la canasta de mercado de un Supermercado Belga [3]. Estas pruebas consistieron en obtener el conjunto de patrones secuenciales frecuentes a partir de la base de datos obtenida a lo largo de tres periodos no consecutivos. El primer período va desde mediados de diciembre de 1999 hasta mediados de enero del 2000. El segundo período va desde inicios del 2000 hasta inicios de Junio del mismo año. El tercer período va desde fines de agosto del 2000 hasta fines de noviembre del 2000. Esta base de datos consta de 88163 transacciones, 3000 ítems únicos a aproximadamente 5133 clientes.

El objetivo de las pruebas realizadas en este escenario es descubrir los patrones de consumo en el tiempo de los clientes del supermercado, además de obtener la cantidad de cada uno de los ítems que serán adquiridos por los mismos, producto de la obtención de patrones secuenciales frecuentes que incluyan factores de cantidad, lo que nos permitirá tener información más precisa y significativa en cuanto a la cantidad de cada uno de los ítems.

Se efectuaron siete pruebas con soportes mínimos de 10%, 2%, 1,5%, 1,0%, 0,75%, 0,50% y 0,25%, cuyos resultaron se observan en la figura 2. Estos resultados fueron comparados con los resultados de la técnica Memisp.

MinSup (%)	MEMISP		MSP-QF	
	T. Ejec. (seg.)	No. Patrones	T. Ejec. (seg.)	No. Patrones
10.00	4	50	5	50
2.00	12	824	15	824
1.50	16	1371	19	1371
1.00	22	2773	27	2773
0.75	28	4582	35	4582
0.50	39	9286	50	9286
0.25	72	30831	89	30831

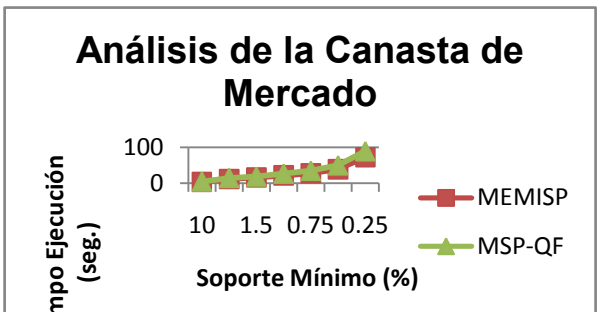

Figura 3. Resultados de las pruebas para el escenario 1.

En la prueba con minSup=2%, se obtuvieron 824 patrones secuenciales con factores de cantidad, algunos de los cuales son:

Aceituna[1.06±0.51]$

Aceituna[1.03±0.5]$ Pimenton[0.37±0.23]$

Aceituna[1.01±0.5]$ Porro[0.57±0.27]$

Apio[0.56±0.25]$ Lechuga[0.53±0.24], Lentejas[0.54±0.23]$

Apio[0.56±0.26]$ Lechuga[0.55±0.24], Pimenton[0.34±0.17]$

Lechuga[0.61±0.25], Lentejas[0.56±0.26]$ Porro[0.59±0.24], Pimenton[0.33±0.15]$

Porro[0.54±0.27], Lentejas[0.62±0.25]$ Lentejas[0.58±0.25]$ Pimenton[0.35±0.17]$

Del subconjunto anterior de patrones secuenciales podemos precisar lo siguiente respecto a las compras hechas por los clientes:

- Los clientes adquieren solamente aceituna en una cantidad de 1.06±0.51.
- Los clientes que hayan adquirido una primera vez solo aceituna, regresan una siguiente vez por pimiento o por porro, con cantidades de 0.37±0.23 y 0.57±0.27 respectivamente. Los que compran después pimiento, compraron antes aceituna en una cantidad igual a 1.03±0.5, mientras que los que adquieren porro lo hicieron con una cantidad igual a 1.01±0.5.
- Los que compran lechuga a la vez compran lentejas en cantidades iguales a 0.61±0.25 y 0.56±0.26 respectivamente. Posteriormente, estos mismos clientes compran porro y pimentón con cantidades iguales a 0.59±0.24 y 0.33±0.15.
- Los que compran porro, en la misma transacción también compran lentejas. Posteriormente vuelven a comprar solamente lentejas, y una siguiente vez compran solo pimentón, en las cantidades que se indican en el patrón

Efecto de los resultados de las pruebas en el escenario 1 sobre los objetivos de minería de datos

Respecto al primer objetivo mencionado al inicio de esta sección, se puede apreciar en el siguiente patrón secuencial frecuente obtenido durante el proceso de minería con un soporte mínimo de 2%, que se ha determinado las cantidades de cada uno de los ítems que es parte del patrón, lo que representa el consumo de cada uno de los ítems por parte de los clientes en las diferentes compras efectuadas.

Porro[0.55±0.27],Lentejas[0.59±0.27]$
Lechuga[0.57±0.25],Porro[0.55±0.26],Pimenton[0.36±0.19]$

El segundo objetivo, fue alcanzado en el sentido que la información proporcionada por los patrones es más precisa y significativa, puesto que se consideran el intervalo de valores para las cantidades de cada item, y de esta forma se obtienen patrones de comportamiento de los consumos de los clientes, respecto no solo a cuales son los productos que suelen comprar juntos sino también en que cantidades los adquieren. Por ejemplo, en el patrón secuencial con factores de cantidad mostrado anteriormente indica que existe un patrón de comportamiento de los clientes respecto a que compran primeramente *porro* con *lentejas*, en cantidades que se encuentran en el rango de valores de [0.55±0.27] y [0.59±0.27] respectivamente, luego en una siguiente compra el mismo cliente, adquiere *lechuga, porro* y *pimentón* juntos, en cantidades aproximadas de [0.57±0.25] para la *lechuga*, [0.55±0.26] para el *porro* y [0.36±0.19] para el *pimentón*.

5.2. Escenario 2: Datos Sintéticos

Para probar la técnica propuesta en este segundo escenario se generaron varias bases de datos (datasets) de forma sintética por medio de la herramienta Synthetic Data Generator [12].

En este escenario, se realizaron pruebas tanto de eficacia, eficiencia y escalabilidad de la técnica propuesta.

- La técnica es eficaz si permite descubrir la misma cantidad de patrones que otras técnicas propuestas y probadas.
- La técnica es eficiente, si los tiempos de ejecución para diferentes soportes mínimos son comparables a técnicas existentes.
- La técnica es escalable si, con el crecimiento de las base de datos, la eficiencia es proporcional a este crecimiento.

El proceso que se siguió para generación sintética de los dataset, es el que describe en [7], y bajo los parámetros citados en [11] los cuales se observan en la figura 2.

Parámetro	Descripción
|DB|	Número de secuencias de datos en la base de datos
|C|	Tamaño promedio (número de transacciones) por cliente
|T|	Tamaño promedio (número de items) por transacción
|S|	Tamaño promedio del patrón secuencial
|I|	Tamaño promedio del itemset frecuente
N_I	Número de itemsets frecuentes
N_S	Número de patrones secuenciales frecuentes
N	Número de posibles ítems
$corr_S$	Nivel de correlación (secuencia), distribución exponencial
$crup_S$	Nivel de corrupción (secuencia), distribución normal
$Corr_I$	Nivel de correlación (itemset), distribución exponencial
$Crup_I$	Nivel de corrupción (itemset), distribución normal

Figura 2. Parámetros para la generación de los dataSet

Se efectuaron un grupo de pruebas para verificar la eficacia y eficiencia del algoritmo. Para esto se generó dataset con los siguientes parámetros: *NI = 25000, NS = 5000, N = 10000, |S| = 4, |I| = 1.25, $corr_S$ = 0.25, $crup_S$ = 0.75, $corr_I$ = 0.25* y *$crup_I$ = 0.75*.

Los resultados de estas pruebas fueron comparadas con los resultados obtenidos en [11] para los algoritmos PrefixSpan-1, PrefixSpan-2 y Memisp.

Pruebas de eficiencia

Se ejecutó un primer subconjunto de pruebas para |C| = 10 y una base de datos de 200000 secuencias, en las cuales se modificó el valor para el soporte mínimo, cuyos resultados se muestran en la figura 4.

minSup	Tiempo Ejecución (seg.)			
(%)	PrefixSpan-1	PrefixSpan-2	MEMISP	MSP-QF
0.25	1491	16	460	346
0.50	812	15	210	148
0.75	265	14	98	64
1.00	176	13	54	31
1.50	85	12	32	9
2.00	30	12	15	5

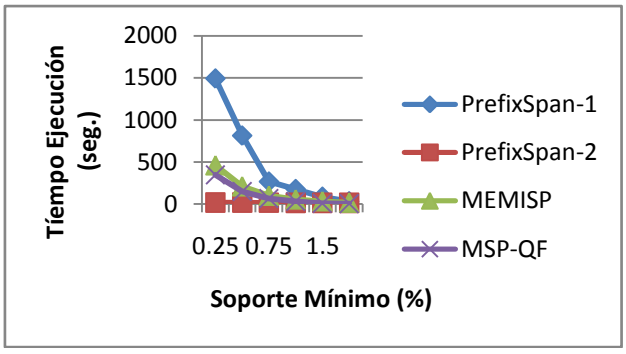

Figura 4. Resultados de las pruebas para |C|=10, |DB|=200K

El segundo subgrupo de pruebas fueron realizadas con un dataset cuyos valores de los parámetros |C| y |T| fueron de 20 y 5 respectivamente. El nuevo valor de |T| hace que el número de ítems de las transacciones aumente, lo que representa que la base de datos también es más grande y densa respecto a la cantidad de patrones secuenciales frecuentes que se vayan a obtener. Los resultados de estas pruebas son las que se observan en la figura 5.

minSup	Tiempo Ejecución (seg.)			
(%)	PrefixSpan-1	PrefixSpan-2	MEMISP	MSP-QF
0.25	7620	1113	2950	3944
0.50	5098	854	1678	2380
0.75	3237	595	1009	1571
1.00	2115	382	821	1018
1.50	981	218	572	582
2.00	349	137	394	341

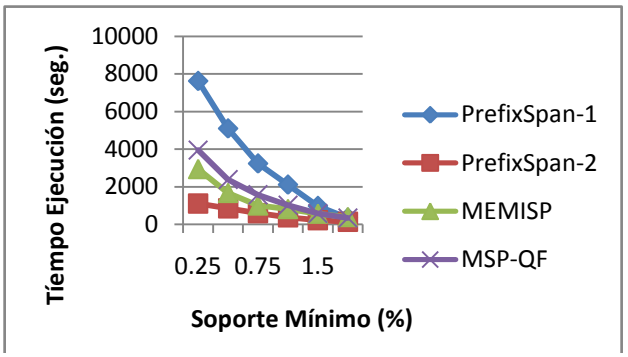

Figura 5. Resultados de las pruebas para |C|=20, |T|=5, |DB|=200K

Un último subconjunto de pruebas de eficiencia se realizó con los mismos parámetros del subconjunto anterior a excepción de |T| que se aumentó a 7.5, cuyos resultados se muestran en la figura 6.

minSup	Tiempo Ejecución (seg.)			
(%)	PrefixSpan-1	PrefixSpan-2	MEMISP	MSP-QF
0.50	16790	19000	5912	4385
0.75	5976	2838	2840	2468
1.00	4090	1899	1985	1707
1.50	1192	1175	1179	1008
2.00	886	883	885	596

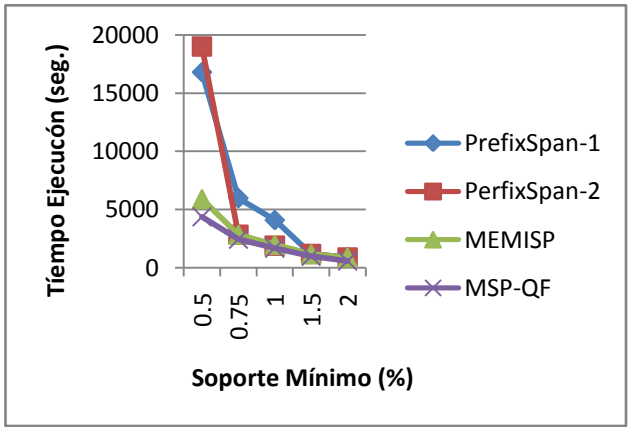

Figura 6. Resultados de las pruebas para |C|=20, |T|=7.5, |DB|=200K

Pruebas de eficacia

Las pruebas de eficacia fueron ejecutadas con los mismos dataset con los que se trabajó en las pruebas de eficiencia, es decir se realizaron en total 92 pruebas de las cuales, 4 no resultaron con las mismas cantidad de patrones secuenciales obtenidos por los algoritmos PrefixSpan-1 y PrefixSpan-2, para dataSet con valores de |C|=20 y |T| igual al 2.5, 5 y 7.5. Estas 4 pruebas, representan el 4% del total.

Pruebas de escalabilidad

Para las pruebas de escalabilidad se utilizaron data datasets generados sintéticamente, con los mismos valores de los parámetros del primer subconjunto de pruebas de eficiencia y con soporte mínimo igual a 0.75%. Para medir la escalabilidad del algoritmo estos dataset fueron generados para |DB| = 1000K, 2000K, 3000K, y así

sucesivamente hasta 10000K, es decir, un millón de secuencias.

La figura 7, se observa los resultados a estas pruebas.

|DB|	Mínimo Soporte (0.75%) / tiempo ejecución (seg.)			
	PrefixSpan-1	PrefixSpan-2	MEMISP	MSP-QF
1000 KB	1.0	1.0	1.0	1.1
2000 KB	3.8	7.7	3.2	3.2
3000 KB	5.0	22.5	3.9	3.9
4000 KB	6.3	36.7	5.1	4.8
5000 KB	7.8		6.5	5.6
6000 KB	9.2		8.2	6.5
7000 KB	11.8		9.2	7.8
8000 KB	13.1		10.8	9.2
9000 KB	16.2		11.9	10.6
10000 KB	17.3		13.1	12.2

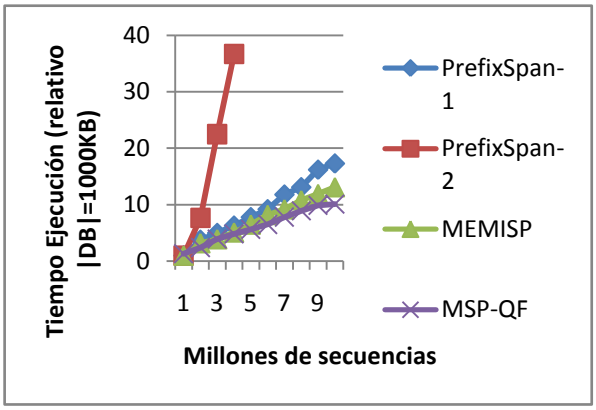

Figura 7. Resultados de las pruebas de escalabilidad

6. Conclusiones y trabajos futuros

Se ha propuesto un algoritmo para el descubrimiento de patrones secuenciales que permite, como parte del proceso de minería, inferir a partir de las cantidades asociadas a cada uno de los items de las transacciones que componen una secuencia, los factores de cantidad vinculados a los patrones secuenciales frecuentes obtenidos como resultado del proceso de minería.

La técnica propuesta ha sido diseñada de forma tal que utiliza un conjunto de índices compactos en los cuales se centra la búsqueda de los patrones secuenciales a partir de patrones frecuentes que ya han sido hallados con anterioridad y que representan los prefijos de los patrones por encontrar. Es por ello, que el tamaño de los índices va disminuyendo conforme el proceso de minería avanza.

Además, se ha logrado que la información proporcionada por los patrones frecuentes con factores de cantidad, sea mucho más precisa, puesto que no solo nos ofrece información sobre la manera de cómo es la relación temporal de los items en las diferentes transacciones, sino también, cual es la relación de las cantidades solicitadas de unos items respecto a otros, lo cual enriquece la semántica brindada por el conjunto de patrones secuenciales.

Por último, de los resultados obtenidos en la sección 5 podemos concluir afirmando que la técnica propuesta cumple con los objetivos del proceso de minería; es eficaz, es eficiente y es escalable debido a que tiene un comportamiento lineal conforme la base de datos de secuencias crece.

Como trabajos futuros se pretende adecuar la técnica propuesta de forma que el proceso de inferencia de los patrones secuenciales frecuentes con factores de cantidad pueda realizarse de manera paralela y utilizando la capacidad de multiprocesamiento de los computadores actuales, y así verificar si los tiempos de ejecución mejoran. Así mismo, extender la técnica propuesta de forma tal que se pueda incluir dentro de su diseño, la gestión de restricciones de tiempo, es decir, que el proceso de minería de patrones secuenciales nos permita descubrir patrones que ocurran en ventanas de tiempo especificadas por el usuario, y personalizar la minería de patrones secuenciales con factores de cantidad realizada por el algoritmo propuesto, de forma que, se pueda realizar este proceso considerando que los patrones obtenidos incluyan cierto tipo(s) de ítem(s).

Referencias bibliográficas

[1]. Agrawal, R. y Srikant, R. (1994). "*Fast algorithms for mining association rules". In Proceeding 20th International Conference Very Large Data Bases, VLDB*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann.

[2]. Agrawal, R. y Srikant, R. (1996). "*Mining Quantitative Association Rules in Large Relational Tables". In Proceeding of the 1996 ACM SIGMOD Conference, Montreal, Québec, Canada*.

[3]. Brijs T., Swinnen G., Vanhoof K., and Wets G. (1999). "*The use of association rules for product assortment decisions: a Case Study". In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, San Diego (USA).

[4]. Molina, L., (2001). "*Torturando los Datos Hasta que Confiesen*". Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Cataluña. Barcelona, España.

[5]. Guevara D, Jaramillo M y Landacay K., *"Inteligencia de Negocios: Minería de patrones secuenciales"*.

[6]. Agrawal, R. y Srikant, R., (1995) "*Minning Pattern Sequential"*.

[7]. Han,J., Pei,J., Mortazavi-Asl, B., Chen, Q., Dayal, U., y Hsu, M-C., (1996) "*Freespan: Frequent pattern-projected sequential pattern mining"*.

[8]. Pei J., (2001) "*PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth"*.

[9]. Zaki M. J., (1998) *Efficient enumeration of frequent sequences*.

[10]. Lin M., Lee S., (2005). *"Fast Discovery of Sequential Patterns through Memory Indexing and Database Partitioning"*. Journal of Information Science and Engineering.

[11]. *Market-Basket Synthetic Data Generator*

[12]. http://synthdatagen.codeplex.com/, ultimo acceso mayo 01, 2014.