Universidad Nacional del Sur

Tesis de Magister en Ciencias de la Computación

Detección de Patrones Emergentes y su formalización utilizando Minería de Datos Incremental

Walter Marcelo Grandinetti

Bahía Blanca — Argentina

2005

Universidad Nacional del Sur

Tesis de Magister en Ciencias de la Computación

Detección de Patrones Emergentes y su formalización utilizando Minería de Datos Incremental

Walter Marcelo Grandinetti

Bahía Blanca — Argentina

2005

Agradecimientos

Quisiera agradecer a mis directores Dr. Carlos Iván Chesñevar y Dr. Marcelo Alejandro Falappa, por compartir conmigo sus conocimientos, por su apoyo y en especial por su inagotable paciencia. Gracias por todo el tiempo que han podido invertir en iniciarme, guiarme y formarme en la investigación. Indudablemente, sin ellos no hubiera sido posible llegar a destino.

Por otra parte, agradezco al Profesor Guozhu Dong por su preocupación y comentarios que me han ayudado a comprender en una mayor profundidad el acercamiento propuesto por su grupo de investigación.

Agradezco al Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur por el apoyo y soporte proporcionado en todo aspecto para la realización de este trabajo.

Gracias a mis compañeros de trabajo y amigos por su cotidiana compañía y por su aporte en la formación de un excelente ambiente de trabajo.

Y un agradecimiento especial a mi familia, Raúl, Etel, Mariano y Flavia por el apoyo, la comprensión, el aliento y ayuda en todo momento.

Prefacio

Esta Tesis es presentada como parte de los requisitos para optar al grado académico de Magister en Ciencias de la Computación, de la Universidad Nacional del Sur, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Departamento de Ciencias e Ingeniería de la Computación, durante el período comprendido entre *Agosto* de 2002 y *Marzo* de 2005, bajo la dirección de los Dres. Carlos Iván Chesñevar y Marcelo Alejandro Falappa, Profesores del Departamento de Ciencias e Ingeniería de la Computación.

Lic. Walter Marcelo Grandinetti

wmg@cs.uns.edu.ar

Departamento de Ciencias e Ingeniería de la Computación

Universidad Nacional del Sur

Bahía Blanca, Marzo de 2005.

Resumen

Minería de datos, definido como el proceso de extracción de conocimiento de grandes volúmenes de datos, ha comenzado a jugar un rol importante en muchos dominios de aplicación resultando ésta especialmente útil en dominios de la medicina y biología, como también en dominios de competencia como puede ser en un entorno comercial o deportivo.

El conocimiento es representado como un conjunto de relaciones entre los datos, usualmente conocidos como *patrones*. De los cuales es importante detectar aquellos que puedan ser particularmente útiles, también conocidos como *patrones interesantes*. La minería de datos afronta dos importantes problemas: En primer lugar, establecer las posibles relaciones entre los datos que dan lugar a la generación de patrones; y en segundo lugar detectar cuándo un patrón puede ser clasificado como potencialmente interesante.

Patrones que se repiten con cierta regularidad, además de ser interesantes per se, pueden ser utilizados para clasificar nuevas instancias de valores. Con tal objetivo, se pueden utilizar conjuntos de patrones para representar clases de datos de forma que las nuevas instancias pueden ser catalogadas en distintas clases según su afinidad (cercanía) a los patrones que describen cada clase.

Análogamente, se pueden clasificar nuevas instancias de datos utilizando patrones que capturen simplemente las diferencias que existen entre las distintas clases. La ventaja de este acercamiento es que usualmente la cantidad de patrones necesarios para representar la diferencia entre dos clases es significativamente menor a utilizar patrones para representar cada clase. Estos patrones son conocidos como *patrones emergentes*, y han demostrado ser muy versatiles aún en otros dominios de aplicación como ser la detección temprana de situaciones anómalas, como pueden ser cambios climáticos, detección de intruso, etc.

El objetivo principal de esta tesis consiste en presentar una nueva forma de obtener patrones emergentes que permita mejorar las herramientas que hacen uso de dichos patrones.

Palabras clave: Minería de Datos, Patrones Emergentes, Patrones Maximales, Patrones Frecuentes, Clasificadores

Abstract

Data mining, defined as the knowledge extraction process from vast volume of data, has began to play an important role in many application domains, resulting especially useful in medicine and biology as well as in competitive domains such as commercial environments and sports activities.

The knowledge is represented as a set of relationships among data elements, usually known as *patterns*. It is important to identify from the set of patterns those that best sum up the relations among the data, called *interesting patterns*. Data mining face two major problems. First, establishing every possible relation among the data elements to produce patterns. Second, distinguishing whether a pattern can be catalogued as potentially interesting or not.

Besides being interesting, patterns occurring repeatedly may be used as classifiers of new instances. Set of patterns can be used to describe the elements present within each class, thus new instances can be sorted out according to their closeness (nearness) to the patterns describing each class.

Analogously, the classification of new instances can be done using patterns that just capture the differences between classes. The advantage of this approach is that the amount of patterns needed to describe such difference is significantly lower than the amount of patterns needed to describe each class. The patterns describing differences between classes are known as *emerging patterns* and they proved to be versatile even in other application domains such as early detection of anomalous situations (for instance, climatic changes, intruder detection, etc).

The thesis main goal consists of showing a new way of mining emerging patterns in order to allow the enhancement of the tools that make use of such patterns.

KEY WORDS: Data Mining, Emerging Patterns, Maximal Patterns, Frequent Patterns, Classifiers

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La *minería de datos* (data mining) se define como el proceso de extracción de conocimiento de grandes volúmenes de datos. Desde esta perspectiva, la minería de datos no es más que un sinónimo del término *Descubrimiento de Conocimiento en Bases de Datos* (Knowledge Discovery in Databases o abreviadamente KDD) [57]. Según otros acercamientos, el proceso de minería de datos solo es un paso más del descubrimiento de conocimiento en bases de datos. En ambas interpretaciones, el proceso de minería da como resultado un *conjunto de patrones*, que bajo la primer perspectiva será considerada como conocimiento y bajo la segunda perspectiva se necesitará de un próximo paso para determinar cuáles patrones son considerados conocimiento.

Considerando a la minería de datos en su aspecto más amplio como descubrimiento de conocimiento en bases de datos, la función de la minería de datos puede ser categorizada en dos clases: *descriptivas* y *predictivas* [37]. La tarea de minería descriptiva caracteriza las propiedades generales de los datos en la base de datos. La tarea de minería predictiva analiza los datos con objeto de construir uno o un conjunto de modelos y realizar predicciones sobre el comportamiento de nuevos conjuntos de datos. Podemos sintetizar lo anterior como sigue:

Minería Descriptiva Los datos son asociados en clases que describen los elementos que la componen en forma resumida, concisa y precisa. Tales descripciones pueden ser derivadas a través de:

1. *Caracterización de datos*: Realizando un resumen de las características o aspectos generales de la clase de datos bajo estudio (usualmente llamada *clase destino*).
2. *Discriminación de datos*: Realizando una comparación de las características generales de una clase destino de objetos de datos con las características generales de una o más clases contrastantes.

Minería Predictiva Los datos son utilizados para construir modelos que describen y distinguen clases de datos. El objetivo es poder utilizar el modelo para predecir la clase de un objeto cuya etiqueta no se conoce como también para predecir valores faltantes en un conjunto de valores como dato.

A lo largo de esta tesis utilizaremos estos dos tipos de minería de datos. Por tal motivo el próximo capítulo está destinado a proveer el conocimiento básico sobre tales tareas y las notaciones a ser utilizadas. Mostraremos a continuación las motivaciones de la búsqueda de patrones y en particular de la aplicación de los patrones emergentes.

1.1. Patrones

La búsqueda de patrones es un campo de gran interés que tiene sus orígenes en el análisis de dominios denominados "canasta de mercado" [2] pero que se ha extendido rápidamente hacia una vasta cantidad de áreas. Consideremos la siguiente aplicación real reportada en [8, 18, 19]:

Ejemplo 1.1.1

La Asociación Nacional de Basketball de Estados Unidos (NBA) se encuentra explorando aplicaciones de minería de datos que puedan ser utilizadas en conjunción con grabaciones de imágenes de partidos de basketball. El software Advanced Scout analiza los movimientos de los jugadores para ayudar a los entrenadores a organizar juegos y estrategias. Esta base de datos se encuentra disponible para todos los entrenadores de la NBA y contiene grandes cantidades de datos de cada juego, todas registradas por tiempo, mostrando información

sobre cada jugador, su rol, anotaciones, bloqueos, rebotes, etc. El entrenador por su parte puede consultar al Advanced Scout acerca de la performance de tiro, ubicación optimal, etc. Por ejemplo, un análisis de una hoja jugada-por-jugada (play-by-play) del juego entre New York Knicks y los Cleveland Cavaliers del 6 de Enero de 1995 muestra que cuando Mark Price jugaba en posición de Defensa, John Williams intentaba cuatro tiros y encestaba cada uno de ellos.

Este software no solo encontraba este patrón, sino que explicaba que este era interesante porque este difería considerablemente del porcentaje promedio de disparos del 49,3 % *de los Cavaliers durante ese juego.* ♢

Se puede observar como la minería de datos ha comenzado a jugar un rol importante en muchos dominios de aplicación resultando ésta especialmente útil en dominios de la medicina y biología para el beneficio de la humanidad, como también en dominios de competencia como puede ser en un entorno comercial o deportivo como en el ejemplo 1.1.1.

Anteriormente hemos hecho referencia a la diferencia entre patrones y conocimiento. Podemos definir al conocimiento como un subconjunto de *patrones interesantes* [37]. El software de minería de datos evalúa miles de patrones de los cuales provee a los entrenadores con solamente un conjunto de ellos, sobre los cuales aún los patrones más interesantes deben ser descubiertos por expertos del dominio. El problema surge en detectar cuándo un patrón puede ser clasificado como interesante y por lo tanto es necesario conocer la forma de los patrones interesantes y tener algún mecanismo para poder clasificarlos.

En primer lugar, existen ciertas propiedades generalmente aceptadas en relación a los patrones interesantes. Un patrón es interesante si *a*) es fácilmente entendible por humanos, *b*) es válido para nuevos datos con cierto grado de certeza, *c*) es potencialmente útil y *d*) es novedoso. Desde otro enfoque un patrón también es interesante si éste valida una hipótesis que el usuario buscaba confirmar. En segundo lugar se han propuesto diferentes medidas y/o métricas que intentan capturar tanto de forma objetiva como subjetiva cuan interesante es un patrón. Las medidas *objetivas* ayudan a detectar de forma automática

en casi cualquier dominio de aplicación a los patrones potencialmente interesantes. Estas medidas son las más comúnmente utilizadas en la literatura existente ya que son lo suficientemente genéricas. Sin embargo, la cantidad de patrones resultantes de utilizar sólo medidas objetivas es en ciertos dominios demasiado numerosa para ser analizados por el usuario. En estos casos son utilizadas como complemento a las medidas objetivas, medidas *subjetivas* que actúan como filtros. Por otro lado, las medidas subjetivas están condicionadas por las creencias del usuario sobre los datos, por lo cual estas medidas son particularmente útiles sólo si las creencias del usuario sobre el dominio son ciertas. Usualmente dicho conjunto de creencias es provisto por algún experto del dominio. Posteriormente, en la sección 2.3 veremos las medidas objetivas más utilizadas y algunas de las medidas subjetivas más generales.

1.2. Motivaciones

Usualmente se relaciona la búsqueda de patrones interesante con la búsqueda de patrones frecuentes y muchos de los esfuerzos realizados en minería de datos se focalizan en identificar eficientemente únicamente los patrones frecuentes. Una limitación de este acercamiento es que supone que es posible encontrar un umbral ρ de mínimo soporte significativo y que permite reducir el número de patrones frecuentes a un nivel manejable. Sin embargo, los patrones frecuentes no siempre son interesantes ya que algunos de ellos simplemente proveen información trivial para el usuario, por ejemplo, si se presenta el siguiente patrón frecuente "*Mark Price* $\rightarrow$ *New York Nicks*", éste es de poca utilidad ya que en la base de datos esto es un hecho, es decir, siempre que aparece en una transacción *Mark Price* también aparece *New York Nicks*. Por otro lado, este esquema de patrones frecuentes deja de lado aquellos patrones que no superan el umbral ρ, los cuales muchas veces resultan en patrones interesantes.

Ejemplo 1.2.1

Supongamos, por ejemplo, que se buscan patrones frecuentes en una tienda de ventas y que a dicha tienda algunos clientes han realizado muy pocas compras (como resultado existen

pocas transacciones y este número no supera el umbral) pero han comprado una cantidad inusual de artículos. Estas personas pueden pertenecer a alguna organización o empresa y son los responsables de realizar las compras. Sería realmente interesante para la tienda poder conocer más sobre las particularidades de estos clientes, ya sea para poder mejorar su servicio, ofrecerle otros productos, enviarles promociones especiales por cantidad, etc. Sin embargo, esta información no será detectada bajo un esquema de umbral de mínimo soporte. ◊

Este tipo de problema es conocido como "*el problema del vodka y del caviar*" [16]. Dicho problema muestra que existe un patrón confiable, el cual establece una relación entre los artículos *Beluga caviar* y *Ketel vodka*. Este patrón muestra que es muy probable que la persona que adquiera un *Ketel vodka* también lleve *Beluga caviar* (como también a la inversa en este caso particular). Sin embargo, este patrón ocurre en la base de datos con muy poca frecuencia, ya que en primer lugar no son artículos de uso diario y además son pocas las personas que tienen el poder adquisitivo necesario para comprarlas. Por otro lado, este patrón es sumamente interesante, a pesar que sea infrecuente, ya que permite reconocer el mercado, clasificar mejor la cartera de clientes o simplemente establecer mejor algunas estrategias de venta.

Tanto los patrones frecuentes como aquellos que son infrecuentes confiables[1] ayudan a detectar afinidades entre los elementos de un conjunto de datos. Por tal motivo, los patrones pueden ser utilizados para clasificación identificando para cada conjunto de datos los patrones que mejor identifican los datos asociados a dicho conjunto. La clasificación de nuevas instancias se realiza observando cuál de los conjuntos de datos modela mejor dicha instancia. El problema de este acercamiento es que la cantidad de patrones que posee cada conjunto de datos es muchas veces muy grande y como mencionáramos anteriormente éste dependerá del umbral utilizado. Si se utiliza un umbral bajo, los patrones modelan mejor el conjunto de datos pero el número de patrones es muy grande. Por otro lado, si el umbral es alto, la cantidad de patrones es baja pero los patrones no modelan con mucha exactitud

[1] En el próximo capítulo definiremos formalmente las nociones de frecuencia y confiabilidad.

el conjunto de datos y por lo tanto la clasificación se ve notablemente afectada.

Una estrategia alternativa es la utilización de *patrones emergentes* [20]. Estos patrones capturan simplemente las diferencias que existen entre los conjuntos de datos, utilizando estas diferencias como reglas para clasificar nuevas instancias. Estas diferencias no necesitan ser absolutas (es decir, no necesariamente deben aparecer en un conjunto de datos y no aparecer en el otro conjunto de datos), sino que identifican aquellas diferencias en las cuales se produce un *incremento significante* de un conjunto de datos al otro. Es importante observar que la frecuencia del patrón (diferencia) en cada conjunto de datos no es relevante sino su crecimiento de un conjunto al otro (es decir, la razón o tasa de crecimiento dada por la división entre ambas frecuencias). Por esta razón, al igual que los patrones infrecuentes interesantes, los patrones emergentes pueden no ser frecuentes en ninguno de los conjuntos de datos y ser muy interesantes. Esta característica de los patrones emergentes los hace especialmente interesantes en la búsqueda de situaciones anómalas, como por ejemplo cambios climáticos, detección de intrusos, etc. donde la frecuencia en que suceden las mismas es muy baja. Utilizando este tipo de patrones se puede comprender mejor las características principales de dichos eventos anómalos y ayudar a prevenirlos. Otra aplicación interesante de estos patrones es la detección temprana de tendencias, en las cuales su frecuencia se va incrementando con el transcurso del tiempo. Este tipo de aplicación puede ayudar a realizar algún tipo de prevención o ayudar al equipo de marketing a detectar tendencias en el mercado, entre otros ejemplos.

1.3. Contribución

En el capítulo 4 se mostrará el estado de arte en la búsqueda de *patrones emergentes* (EPs). Allí se podrá apreciar que se han propuesto varias alternativas para la búsqueda de estos patrones. Sin embargo, estas alternativas realizan simplemente la búsqueda de un subconjunto de la totalidad de EPs. Esta restricción se debe a dos motivos: *a*) Los EPs no poseen la propiedad de antimonotonía[2] la cual permite reducir significativamente el

[2]Ver propiedad 2.5.1 página 37.

espacio de búsqueda, *b*) Es inviable la enumeración y análisis de todos los potenciales EPs. En esta tesis proponemos realizar una mejor aproximación del conjunto de EPs realizando una descomposición del espacio de EPs en sectores. Estos sectores se analizarán en forma incremental permitiendo reutilizar información previamente obtenida. A continuación se enunciarán las principales contribuciones aportadas en esta tesis:

1. Resumen del estado actual del arte en la búsqueda y representación concisa de patrones frecuentes.

2. Resumen del estado actual del arte en la búsqueda de patrones emergentes.

3. Introducción de un método incremental para el descubrimiento de EPs utilizando como analogía la suma de Riemman para la aproximación del área.

4. Introducción de la noción de sectores.

5. Enunciación de propiedades de los bordes, sectores y del espacio de búsqueda.

6. Modificación del algoritmo para búsqueda de patrones maximales para efectuar una generación de patrones incremental.

7. Introducción de una nueva operación de diferencia eficiente sobre bordes. Esta nueva operación de diferencia realiza una mejora importante sobre la anterior operación, ya que la misma tiene menos restricciones para su utilización que su predecesora. Es importante mencionar que esta nueva operación de diferencia utiliza internamente la operación de diferencia previa.

8. Introducción de una mejora para ambas operaciones de diferencia utilizando la propiedad de monotonía.

9. Prueba formal de la correctitud de la nueva operación de diferencia.

10. Detección de problemas relacionados al concepto de unión de bordes.

1.4. Organización

En el capítulo 2 se presentarán las nociones fundamentales sobre el proceso de minería de datos.

En el capítulo 3 se mostrarán distintas formas de representar concisa y eficientemente el conjunto de patrones frecuentes.

En el capítulo 4 se darán los conceptos fundacionales de los patrones emergentes, así como diferentes variantes de los mismos. Dichos patrones están basados en los patrones maximales vistos en el capítulo 3.

En el capítulo 5 presentaremos nuestro aporte al problema de minería de patrones emergentes utilizando un acercamiento incremental de minería de patrones maximales y patrones emergentes.

En el capítulo 6 se proveerán comentarios finales y líneas de trabajo a seguir.

Capítulo 2

Minería de Patrones

2.1. Itemsets

De la teoría de bases de datos relacionales conocemos que las mismas están constituidas por un conjunto de relaciones $\mathcal{R}$ las cuales son descriptas a través de *esquemas*. Estos esquemas son un conjunto de atributos que tienen asociado un dominio. Una instancia de una relación está constituida por entidades o conjunto de tuplas donde cada tupla tiene un valor para cada atributo del esquema de relación. Estos valores pertenecen al dominio del atributo. Para más información sobre conceptos básicos de la teoría de bases de datos puede consultarse [70].

Antes de realizar la minería de datos es necesario realizar un preprocesamiento de los datos. Debemos recordar que la minería de datos se realiza sobre grandes volúmenes de datos y usualmente estos datos provienen de diferentes fuentes por lo que es altamente probable que existan datos inconsistentes, faltantes o ruido. Por tal motivo es necesario realizar *a*) una *limpieza de datos* (data cleaning) para remover ruido y corregir inconsistencias en los datos, *b*) una *integración de datos* (data integration) para combinar datos de múltiples orígenes en una base de datos coherente, *c*) una *transformación de datos* (data transformation) para normalizar diferentes escalas de valores utilizados, obtener atributos categóricos y sintetizar valores (como sumas totales, promedios, etc.), *d*) una *reducción de datos* (data reduction) para eliminar características innecesarias o redundantes para el proceso de minería.

Como resultado de este procesamiento, usualmente asociado a la etapa de Data Warehousing y Data Preprocessing, se obtiene un conjunto de datos que poco tiene que ver con la base de datos original y cuyo formato es apto para la tarea de minería de datos. En particular, usualmente se habrán eliminados todos los valores continuos para atributos con dominios continuos, siendo reemplazados estos por valores discretos asociados a nuevos atributos categóricos. Por ejemplo, supongamos tener dos atributos `Edad` y `Sueldo`, el atributo `Sueldo` es un atributo con dominio de valores continuo y puede ser reemplazado por un atributo categórico Ingreso cuyo dominio sean los valores {`Bajo, Medio, Alto`} o también por otro atributo categórico Ingreso cuyo dominio sean los siguientes rangos $\{0 \ldots 100, 101 \ldots 500, 501 \ldots 1000, 1001 \ldots 2000, \text{más de } 2000\}$. A su vez el atributo `Edad` aunque puede no ser un valor continuo puede ser generalizado en un nuevo atributo categórico de menos valores en su dominio como ser {`niño, joven, adulto, anciano`}.

En la búsqueda de patrones interesantes a partir de un conjunto de datos usualmente se utiliza una versión especial de base de datos, la cual es denominada *base de datos transaccional*. A continuación daremos las definiciones generales que se utilizarán a lo largo de esta tesis y mostraremos como esta definición especial de base de datos puede ser utilizada para representar cualquier otra base de datos y por lo tanto haciendo extensible todos los resultados a cualquier base de datos. Estas definiciones son usualmente compartidas por la mayoría de los trabajos analizados aunque difieren en la notación utilizada, por tal motivo utilizaremos como notación la provista en [29].

Definición 2.1.1 (Item)
Llamaremos item *a cada uno de los diferentes valores que pueden ser registrados en la base de datos transaccional.* ♦

Esta definición de item es intuitiva aunque poco formal, sin embargo permite capturar la esencia de un item en nuestro contexto. En la sección 2.6.1 se proveerá una definición formal de item dentro del contexto de los reticulados (ver definición 2.6.6 página 49).

Definición 2.1.2 (Itemset)

Sea $\mathcal{I}$ el conjunto de todos los items posibles. Llamaremos itemset *o* patrón *a cualquier conjunto $X = \{i_1, \ldots, i_k\} \subseteq \mathcal{I}$. Si el itemset contiene k items también puede ser llamado $k-$itemset o itemset de* cardinalidad *k.* ♦

Definición 2.1.3 (Transacción sobre $\mathcal{I}$)

Una transacción *sobre $\mathcal{I}$ es una tupla $T = (tid, I)$ donde tid es un identificador de transacción y I es un itemset. Diremos que dicha transacción* soporta[1] *un itemset $X \subseteq \mathcal{I}$ si $X \subseteq I$.* ♦

Definición 2.1.4 (Base de Datos Transaccional sobre $\mathcal{I}$)

Una base de datos transaccional *$\mathcal{D}$ sobre $\mathcal{I}$ es un conjunto de transacciones sobre $\mathcal{I}$.* ♦

Como se puede apreciar la definición de base de datos transaccional difiere de la definición de base de datos relacional en que las transacciones que constituyen las bases de datos no son estructuras fijas sino que la cantidad de valores que cada instancia contiene puede variar. Por lo cual la base de datos no es representable por un esquema de atributos. Sin embargo, una base de datos relacional puede ser representable en una base de datos transaccional realizando la siguiente transformación. Supongamos tener un esquema de relación $\mathcal{R} = (A_1, \ldots, A_k)$ y una instancia genérica de una tupla de la relación $\mathcal{T} = \{v_1, \ldots, v_k\}$ donde cada v_i pertenece al dominio del atributo A_i con $1 \leq i \leq k$. Trasformaremos cada tupla de relación en una nueva tupla transaccional con la siguiente forma $(tid, \{(A_1, v_1), \ldots, (A_k, v_k)\})$.

Veamos el siguiente ejemplo sobre una base de datos que contiene información sobre las personas que concurren a un restaurant se informa si dicha persona espera o no por una mesa. Este ejemplo, mencionado en [65], usualmente es utilizado para identificar las características más significativas que llevan a que una persona espere por la mesa.

Ejemplo 2.1.1

Supongamos tener una tabla con instancias de la relación Restaurant *con los siguientes*

[1] Como sinónimos a la palabra *soporta* pueden utilizarse las palabras *ocurre* o *aparece*. En forma inversa podemos decir que el itemset X *es una instancia en* T.

Ejemplo	Atributos										
	A	B	F	H	G	P	LL	R	T	E	Q
1	Si	No	No	Si	Algunos	\$\$\$	No	Si	Frances	$0-10$	Si
2	Si	No	No	Si	Lleno	\$	No	No	Chino	$30-60$	No
3	No	Si	No	No	Algunos	\$	No	No	Burger	$0-10$	Si
4	Si	No	Si	Si	Lleno	\$	No	No	Chino	$10-30$	Si
5	Si	No	Si	No	Lleno	\$\$\$	No	Si	Frances	>60	No
6	No	Si	No	Si	Algunos	\$\$	Si	Si	Italiano	$0-10$	Si
7	No	Si	No	No	Lleno	\$	Si	No	Burger	$0-10$	No
8	No	No	No	Si	Algunos	\$\$	Si	Si	Chino	$0-10$	Si
9	No	Si	Si	No	Lleno	\$	Si	No	Burger	>60	No
10	Si	Si	Si	Si	Lleno	\$\$\$	No	Si	Italiano	$10-30$	No
11	No	No	No	No	Vacío	\$	No	No	Chino	$0-10$	No
12	Si	Si	Si	Si	Lleno	\$	No	No	Burger	$30-60$	Si

Cuadro 2.1: Instancias de ejemplos para el dominio Restaurant

atributos, a) Alternativa (A) *si existe algún restaurant alternativo cercano, b*) Bar (B) *si el restaurant tiene algún área de bar confortable donde esperar, c*) Fin de Semana (F) *si el día es Viernes o Sábado, d*) Hambre (H) *si tiene hambre o no, e*) Gente (G) *cantidad de gente en el restaurant* (*Vacío, Algunos, Lleno*), *f*) Precio (P) *rango de precios del restaurant* (*\$, \$\$, \$\$\$*), *g*) Lluvia (LL) *Si lloviendo o no, h*) Reservación (R) *si tiene reservación hecha, i*) Tipo (T) *clase de restaurant* (*Frances, Italiano, Chino o Burger*), *j*) Espera Estimada (E) *el tiempo aprox. de espera* ($0-10$ *minutos,* $10-30$, $30-60$, >60). *k*) Queda (Q) *si la persona se queda en el restaurant.*

La tabla que describe una base de datos relacional se muestra en el cuadro 2.1 y como resultado de aplicar la transformación mencionada se obtiene la base de datos transaccional que se muestra en el cuadro 2.2.

Definición 2.1.5 (Cubrimiento de un Itemset)

El cubrimiento *de un itemset X en $\mathcal{D}$ consiste del conjunto de identificadores de transacciones (tidset) en $\mathcal{D}$ que soportan X:*

$$cubre(X, \mathcal{D}) = \{tid \mid (tid, I) \in \mathcal{D}, X \subseteq I\}$$ ♦

Una propiedad importante del cubrimiento de itemsets es la siguiente:

Tid	Atributos
1	(A,Si), (B,No), (F,No), (H,Si), (G,A), (P,$$$), (LL,No), (R,Si), (T,Fr), (E,0-10), (Q,Si)
2	(A,Si), (B,No), (F,No), (H,Si), (G,LL), (P,$), (LL,No), (R,No), (T,Ch), (E,30-60), (Q,No)
3	(A,No), (B,Si), (F,No), (H,No), (G,A), (P,$), (LL,No), (R,No), (T,Burg), (E,0-10), (Q,Si)
4	(A,Si), (B,No), (F,Si), (H,Si), (G,LL), (P,$), (LL,No), (R,No), (T,Ch), (E,10-30), (Q,Si)
5	(A,Si), (B,No), (F,Si), (H,No), (G,LL), (P,$$$), (LL,No), (R,Si), (T,Fr), (E,>60), (Q,No)
6	(A,No), (B,Si), (F,No), (H,Si), (G,A), (P,$$), (LL,Si), (R,Si), (T,Ital), (E,0-10), (Q,Si)
7	(A,No), (B,Si), (F,No), (H,No), (G,LL), (P,$), (LL,Si), (R,No), (T,Burg), (E,0-10), (Q,No)
8	(A,No), (B,No), (F,No), (H,Si), (G,A), (P,$$), (LL,Si), (R,Si), (T,Ch), (E,0-10), (Q,Si)
9	(A,No), (B,Si), (F,Si), (H,No), (G,LL), (P,$), (LL,Si), (R,No), (T,Burg), (E,>60), (Q,No)
10	(A,Si), (B,Si), (F,Si), (H,Si), (G,LL), (P,$$$), (LL,No), (R,Si), (T,Ital), (E,10-30), (Q,No)
11	(A,No), (B,No), (F,No), (H,No), (G,V), (P,$), (LL,No), (R,No), (T,Ch), (E,0-10), (Q,No)
12	(A,Si), (B,Si), (F,Si), (H,Si), (G,LL), (P,$), (LL,No), (R,No), (T,Burg), (E,30-60), (Q,Si)

Cuadro 2.2: Base de datos transaccional para el dominio Restaurant

Propiedad 2.1.1

Sean $X, Y \subseteq \mathcal{I}$ dos itemsets y $\mathcal{D}$ una base de datos transaccional sobre $\mathcal{I}$. Entonces

$$X \subseteq Y \Rightarrow cubre(Y) \subseteq cubre(X) \qquad \square$$

La prueba de esta propiedad es trivial ya que todas las transacciones donde Y aparece, también aparece X entonces al menos $cubre(Y) = cubre(X)$. Sin embargo, X puede aparecer en otras transacciones solo o junto a otros items no incluidos en Y entonces $cubre(Y) \subseteq cubre(X)$.

Definición 2.1.6 (Soporte de un Itemset)

El soporte *de un itemset X en $\mathcal{D}$ es el número de transacciones que cubren X en $\mathcal{D}$:*

$$soporte(X, \mathcal{D}) = \mid cubre(X, \mathcal{D}) \mid \qquad \blacklozenge$$

Definición 2.1.7 (Frecuencia de un Itemset)

La frecuencia *de un itemset X en $\mathcal{D}$ es la probabilidad de que X ocurra en la transacción $T \in \mathcal{D}$:*

$$frecuencia(X, \mathcal{D}) = P(X) = \frac{soporte(X, \mathcal{D})}{\mid \mathcal{D} \mid} \qquad \blacklozenge$$

Definición 2.1.8 (Itemset Frecuente y Umbral)

Diremos que un itemset es frecuente *si su soporte es mayor que un mínimo umbral de soporte (minimal support threshold) absoluto*[2] δ_{abs} *dado, con* $0 \leq \delta_{abs} \leq \mid \mathcal{D} \mid$. ♦

Definición 2.1.9 (Conjunto de Itemsets Frecuentes)

Sea $\mathcal{D}$ *una base de datos transaccional sobre un conjunto de items* $\mathcal{I}$ *y* δ *el mínimo umbral de soporte. El* conjunto de itemsets frecuentes *en* $\mathcal{D}$ *con respecto a* δ *es denotado por:*

$$\mathcal{F}(\mathcal{D}, \delta) = \{X \subseteq \mathcal{I} \mid soporte(X, \mathcal{D}) \geq \delta\}$$ ♦

Definición 2.1.10 (Itemset Específicos y Generales, Dong & Li, 1999)

Dados dos itemsets I_1 *e* I_2*. Diremos que* I_1 *es más general (more general) que el itemset* I_2 *si* $I_1 \subset I_2$*; también diremos que* I_2 *es más específico (more specific) que* I_1. ♦

Cuanto más específico sea un itemset, más largo es o mayor cardinalidad tiene. En forma simétrica, cuanto más general sea el itemset, menor cantidad de items posee.

Con motivo de representar en forma concisa al conjunto de itemsets, una taxonomía de itemsets a surgido como también una serie de propiedades. Veremos en más detalle estas afirmaciones en los capítulos 3 y 4. Una de las clases de itemsets más prominentes es la de itemsets maximales que se definen según la siguiente definición.

Definición 2.1.11 (Itemsets Maximales y Minimales, Dong & Li, 1999)

Dada una colección $\mathcal{C}$ *de itemsets. Un itemset* $X \in \mathcal{C}$ *se define como* maximal *en* $\mathcal{C}$ *si no existe un superconjunto propio de* X *que esté en* $\mathcal{C}$*. De forma similar, un itemset* $Y \in \mathcal{C}$ *se define como* minimal *en* $\mathcal{C}$ *si no hay ningún subconjunto propio de* Y *en* $\mathcal{C}$. ♦

A su vez una de las propiedades más utilizadas explícita o implícitamente por los algoritmos de minería de datos es la siguiente:

Propiedad 2.1.2 (Clausura de subconjuntos, Dong & Li, 1999)

Dada una colección de itemsets $\mathcal{C}$ *se dice cerrado en subconjuntos (subset-closed) si y solo*

[2]En lo sucesivo omitiremos el subíndice *abs*. Cabe aclarar que puede utilizarse un umbral relativo si se utilizara la *frecuencia* en lugar del *soporte*.

si para cualquier itemset X que pertenece a $\mathcal{C}$ ($X \in \mathcal{C}$) todos los subconjuntos también pertenecen a $\mathcal{C}$. □

Cuando sea evidente la base de datos sobre la cual estamos trabajando, se podrá omitir especificar el parámetro $\mathcal{D}$ en las funciones *cubre*, *soporte*, $frecuencia$ y en el conjunto de itemsets frecuentes $\mathcal{F}$.

2.2. Reglas de Asociación

Los itemsets frecuentes (o algún subconjunto de éstos) muestran al usuario una relación importante entre los items que constituyen dicho itemset. Sin embargo, cuanto más significativa sea la información que es mostrada al usuario, mejor será la probabilidad que esta información sea útil. Existe una forma alternativa de representar los itemsets frecuentes la cual presenta los items que conforman el itemset en un formato más amigable para el usuario, como una regla de deducción. Estas reglas son conocidas como *reglas de asociación* (Association Rules) [2, 88].

Definición 2.2.1 (Regla de Asociación)
Una regla de asociación describe una relación entre los items de un itemset y está compuesta por dos itemsets X e Y tal que $X \cap Y = \emptyset$, llamados antecedente *(o cuerpo) y* consecuente *(o cabeza) respectivamente, notada $X \rightarrow Y$.* ♦

Asociado a una regla de asociación se proveen estadísticas basadas en el soporte o frecuencia del itemset en la cual la regla está basada. Estas estadísticas proveen más información sobre el itemset permitiendo descartar gran cantidad de los patrones frecuentes para obtener aquellos que son más interesantes para el usuario. Las dos estadísticas más utilizadas con las reglas de asociación se enunciarán a continuación. En la sección 2.3 se proveen otras estadísticas que complementan las mismas y ayudan la usuario a filtrar los patrones obtenidos.

Definición 2.2.2 (Soporte y Frecuencia de una Regla de Asociación)
Sea $X \rightarrow Y$ una regla de asociación entre los itemsets X e Y que pertenecen a la base de

datos $\mathcal{D}$. El soporte y la frecuencia representan la cantidad y proporción de transacciones que contienen tanto el antecedente como el consecuente respectivamente y se definen como:

$$soporte(X \rightarrow Y, \mathcal{D}) = soporte(X \cup Y, \mathcal{D})$$
$$frecuencia(X \rightarrow Y, \mathcal{D}) = frecuencia(X \cup Y, \mathcal{D})$$ ♦

Definición 2.2.3 (Confianza de una Regla de Asociación)
Sea $X \rightarrow Y$ una regla de asociación entre los itemsets X e Y que pertenecen a la base de datos $\mathcal{D}$. La confianza (Confidence) de una regla de asociación representa la proporción de transacciones que conteniendo el antecedente también contienen el consecuente (también conocida como la probabilidad condicional que dado X ocurra Y).

$$confianza(X \rightarrow Y, \mathcal{D}) = P(Y|X) = \frac{soporte(X \cup Y, \mathcal{D})}{soporte(X, \mathcal{D})}$$

♦

Asociados con ambas estadísticas se utilizan umbrales para filtrar las reglas de asociación que no superen dichos umbrales. Se utiliza un umbral de soporte o frecuencia, el cual es innecesario si se utilizan itemsets frecuentes, como también un umbral de confianza mínima. Usualmente, las reglas se asociación se obtienen utilizando un acercamiento de dos fases:

1. Se obtienen todo los itemsets frecuentes de la base de datos utilizando el umbral de mínimo soporte.

2. Se generan a partir de los itemsets frecuentes las reglas de asociación filtrando aquellas que no superen el umbral de mínima confianza.

El ejemplo más conocido de una regla de asociación basada en un análisis de mercado, publicada en [2], muestra como se puede encontrar información potencialmente valiosa y totalmente inesperada a partir de dominios que parecían conocidos.

Ejemplo 2.2.1

Un análisis del comportamiento de consumidores en término de los productos comprados ha revelado el siguiente patrón frecuente de comportamiento:

$$compra(Pañales) \rightarrow compra(Cerveza)$$

Dicho patrón establece que cada vez que un consumidor compra pañales existe un alto grado de probabilidad que compre también cerveza. ◊

Observar que en el ejemplo anterior el itemset frecuente es {Pañales, Cerveza}. Por lo cual, el soporte de la regla de asociación es igual al soporte de itemset, como también al soporte de la regla de asociación $compra$(Cerveza) $\rightarrow$ $compra$(Pañales). Sin embargo, tal resultado no puede ser trasladado en el contexto de la confianza, es decir, la probabilidad que un consumidor compre pañales si compró cerveza es muy baja (tiene poca confianza) y por lo tanto no es una regla interesante.

Es interesante notar que el proceso de búsqueda de reglas de asociación es complementario al proceso de *recuperación de información* (information retrieval) [9]. En un proceso de recuperación de información dado una consulta se determinará el conjunto de transacciones o registros de la base de datos que corresponden con dicha consulta. En contraste, en la búsqueda de reglas de asociación se determinarán las consultas (reglas) que tengan una cantidad suficiente de transacciones o registros que la soporten.

2.3. Patrones Interesantes

En la sección 1.1 hemos mencionado algunas de las características que poseen los patrones interesantes. Por otro lado, la mera generación de itemsets o reglas de asociación frecuentes producen miles de patrones que satisfacen las restricciones de soporte y confianza. Esto hace imposible la tarea de evaluar manualmente el grado de interés sobre una determinada regla o patrón. Claramente, es deseable reducir el número de reglas de tal forma que solamente las reglas más interesantes sean identificadas. Por tal motivo, se han propuesto diversas métricas y/o medidas alternativas para evaluar cuan interesante

es una regla. Existen dos grupos de métricas:

1. *Objetivas*: Son aquellas independientes de la aplicación. Por ejemplo, las medidas de soporte y confianza.

2. *Subjetivas*: Son aquellas que relaciona el grado de interés en una regla a la información específica necesaria para un usuario en un contexto específico.

La medida objetiva de interés más conocida ha sido propuesta en [57] y establece que una regla no es interesante si la frecuencia de la regla es aproximadamente igual al producto de las frecuencias del antecedente y consecuente. Es decir, si los consecuentes y antecedentes son estadísticamente independientes entonces la regla es poco probable que sea considerada como interesante sin importar cuan frecuente o confiable sea. Formalmente,

Definición 2.3.1 (Regla Interesante, Piatetsky-Shapiro 1991)
Una regla $X \rightarrow Y$ no es interesante si

$$frecuencia(X \rightarrow Y) \approx frecuencia(X) \times frecuencia(Y)$$ ♦

Es importante notar que la definición utiliza la medida de frecuencia que es el porcentaje de transacciones que contienen el itemset con respecto al total y por lo tanto es un valor entre 0 y 1. Este resultado no es aplicable si se utiliza la medida soporte que es un valor absoluto.

Se puede extender este concepto para evaluar si un itemset es interesante y se basa en verificar si con dicho itemset se puede generar alguna regla de asociación interesante. Dicho trabajo puede encontrarse en [83], donde además se evalúan distintas formas de reglas de asociación basados en la función de dependencia estadística y la medida de *Piatetsky-Shapiro*. En particular, se define la dependencia entre los itemsets de una regla $X \rightarrow Y$ de la siguiente forma:

Definición 2.3.2 (Dependencia)
Sea $X \rightarrow Y$ una regla de asociación, y $p(Y|X)$ la probabilidad que en una transacción

dada aparezca Y *si está* X*.*

$$Dependencia(X,Y) = \frac{frecuencia(X \cup Y)}{frecuencia(X) \times frecuencia(Y)} = \frac{p(Y|X)}{frecuencia(Y)} \quad \blacklozenge$$

En esta definición se pueden identificar tres casos como resultado de la función *Dependencia*.

(a) Si $Dependencia(X,Y) = 1$ entonces X e Y son independientes.

(b) Si $Dependencia(X,Y) > 1$ entonces Y depende de X *positivamente*. En este caso la regla es interesante.

(c) Si $Dependencia(X,Y) < 1$ entonces Y depende de X *negativamente*. Este caso es investigado en [83], reescribiendo la regla $X \rightarrow Y$ como una *regla de asociación negativa* que puede ayudar a la toma de decisiones.

Otra medida objetiva popular es *elevación* (Lift) la cual es la razón (ratio) entre el soporte condicional del consecuente en las transacciones que contienen el antecedente, con el soporte del consecuente.

Definición 2.3.3 (Elevación – Lift)

$$Lift(X \rightarrow Y) = \frac{confianza(X \rightarrow Y)}{frecuencia(Y)} \quad \blacklozenge$$

Los valores de elevación superiores a 1 indican que el consecuente es más frecuente en las transacciones que contienen el antecedente que en las que no. Debe notarse que una regla de asociación con bajo soporte y alta elevación puede ser de menor interés que una regla con alto soporte y baja elevación, dado que esta última se aplica a un mayor número de individuos. La influencia captura tanto el volumen y el efecto entre los itemsets de la regla. Complementando esta medida existe otra medida objetiva llamada *influencia* (Leverage).

Definición 2.3.4 (Influencia – Leverage)

$$Leverage(X \rightarrow Y) = frecuencia(X \rightarrow Y) - frecuencia(X) \times frecuencia(Y) \quad \blacklozenge$$

Esta es la diferencia entre la frecuencia observada cuando ocurren tanto X como Y y la frecuencia que se esperaría si ambos itemsets fueran independientes.

Ejemplo 2.3.1

Supongamos que en una gran tienda de venta de múltiples artículos se realiza un proyecto de minería de datos [57]. Se busca entender la relación entre las compras de lencería con otros productos. Una simple búsqueda de reglas de asociación podría encontrar con una frecuencia y confianza extraordinariamente alta que (*art. lencería*) $\rightarrow$ (*golosinas*)*. Este patrón es interesante ya que no es esperado. Sin embargo, un análisis más detallado muestra que golosinas es un artículo que se encuentra asociado a casi todos los artículos. Es decir, que casi todos los clientes han comprado golosinas, por lo cual, esta regla no es interesante ya que el antecedente y el consecuente son independientes (según la definición 2.3.2).* $\lozenge$

Además de estas medidas para obtener el conjunto de reglas interesantes, se han propuesto numerosos trabajos [86, 89, 9, 5, 55] basados en la teoría matemática de *análisis de conceptos formales* (Formal Concept Analysis)[3] [27, 39, 73, 74, 72] que generan reglas de asociación no redundantes. Estas reglas están formadas por los *itemset cerrados frecuentes* (Frequent Closed Itemset)[4] cuentan entre sus propiedades:

1. Generación únicamente de las reglas no redundantes con mayor información.

2. Presentación al usuario de las reglas más sorprendentes que cubren a todas las otras reglas no redundantes válidas.

[3]Una breve introducción a FCA en el contexto de base de datos transaccionales se verá en la sección 3.3.1 página 97.

[4]Concepto que se analizará en la sección 3.3 página 97.

3. Extracción de un conjunto de reglas sin pérdida de información, es decir que todas las reglas de asociación con su información sobre soporte y confianza puede ser obtenida de este conjunto de reglas.

2.4. Bases de Datos

Las bases de datos poseen diferentes características y en general, se pueden establecer *dos* clases de bases de datos:

a) Bases de datos *ralas*: Tienen la particularidad que poseen pocos itemsets frecuentes. Es decir que, la cardinalidad de cada itemset frecuente es baja y por lo tanto los itemsets poseen pocos items.

b) Bases de datos *densas*: Poseen una gran cantidad de itemsets frecuentes superando ampliamente la cantidad de transacciones de la base de datos. Esto se debe a que existen itemsets frecuentes que poseen una cardinalidad grande, generando estos una cantidad exponencial[5] de itemsets frecuentes. Supongamos que el itemset X tiene una cardinalidad ℓ entonces la cantidad de itemsets frecuentes que se pueden deducir a partir de este itemset es 2^{ℓ}.

Cada una de estas clases afectan en forma diferente a la performance de los algoritmos que son utilizados para la búsqueda de patrones. En general, si un algoritmo posee una buena performance sobre una de las clases, entonces ésta performance se verá severamente afectada sobre la otra clase de base de datos. Por tal motivo, los algoritmos hacen referencia hacia que clase de base de datos está orientada su búsqueda y las comparaciones se efectúan utilizando otros algoritmos utilizados también para la misma clase. Las bases de datos ralas tienen la propiedad de que sus itemsets frecuentes tienen una baja cardinalidad (ya que de lo contrario generarían una cantidad exponencial de itemsets frecuentes), lo cual resulta beneficioso para aquellos algoritmos que realizan su búsqueda

[5]Una explicación teórica a esta afirmación puede encontrarse en la sección 2.5 y en particular si se considera la propiedad 2.5.1 de la página 37.

comenzando por los itemsets más pequeños y avanzando hacia los más grandes una vez que se han analizado todos los itemsets menores. Sin embargo, esta clase de base de datos es usualmente perjudicial para la performance de los algoritmos orientados a bases de datos densas, los cuales intentar detectar los patrones más grandes lo más pronto posible, resultando en un gran desperdicio de cómputos realizados por el algoritmo. En forma análoga, sucede lo contrario cuando se tienen bases de datos densas. Los algoritmos por niveles pasan la mayor parte del tiempo detectando patrones frecuentes, generando una cantidad exponencial de estos. Veremos en el capítulo 3 una gran variedad de algoritmos especialmente orientados hacia bases de datos densas. Por otro lado, existen algunos acercamientos que intentan resolver este problema de diferentes clases de bases de datos utilizando una estrategia que sea eficiente en forma independiente de la clase utilizada (el algoritmo `Pincer-Search` que se describe en la sección 3.2.3 (página 78) es un ejemplo de este tipo de acercamiento). Sin embargo, contrario a la motivación inicial, en lugar de obtener lo *mejor* de ambos acercamientos, su performance se ve *degradada* en ambos casos, con excepción de ciertos casos especiales (los cuales son mostrados en la descripción del algoritmo `Pincer-Search`).

Una de las consideraciones más importantes de los algoritmos de minería de datos es la representación a ser utilizada de la base de datos transaccional. Distintas representaciones pueden afectar significativamente la eficiencia de los algoritmos disminuyendo o aumentando la cantidad de accesos a almacenamiento secundario ya que esta no puede ser almacenada en memoria debido a su tamaño.

2.4.1. Representación Horizontal

La representación horizontal es la más comúnmente utilizada y consiste en representar la base de datos transaccional como un conjunto de transacciones tal como se menciona en las definiciones 2.1.3 y 2.1.4. Otra forma de representación horizontal es utilizando una matriz bidimensional de bits (bitmap) donde cada columna representa un item y cada fila una transacción. En el cuadro 2.3 se ilustra una base de datos transaccional utilizando una representación horizontal tradicional (izquierda) y una representación horizontal con

tid	Itemsets
1	BCD
2	ACD
3	ABD
4	AC
5	ABC
6	ACD
7	AB
8	ABC
9	ABC
10	ABCD

tid	A	B	C	D
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	0	1	0
5	1	1	1	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	0
9	1	1	1	0
10	1	1	1	1

Cuadro 2.3: Representación Horizonal de una Base de Datos

matriz bidimensional (derecha). Estas formas de representación horizontal son utilizadas dependiendo de la naturaleza de la base de datos. En bases de datos donde la mayoría de las transacciones son cortas, una representación tradicional ocupa relativamente menos espacio ya que simplemente tiene conocimiento de los items que constituyen la transacción. Sin embargo, en transacciones largas, un esquema de matriz puede ser más apropiado ya que por cada item simplemente se activa 1 bit (el del item) y no 8 o más bits que serían necesarios para codificar el item. Esta representación por bits (matriz) tiene la ventaja que puede ser susceptible a compresión de datos y muchas veces aún para bases de datos con transacciones cortas esta compresión mejora a la representación tradicional.

El uso principal de la base de datos durante la minería de datos consiste de calcular el soporte para un itemset (o un conjunto de ellos) dado. Para una representación horizontal esta tarea se realiza accediendo a cada transacción $\mathcal{T}$ y determinando si dicho itemset X es subconjunto de la transacción ($X \subseteq \mathcal{T}$).

Una creencia común sobre las representaciones horizontales considera que la tarea de recorrer la base de datos es la tarea que más tiempo consume en la minería de datos. Aún cuando esta tarea consume un tiempo considerable del proceso total, el consumo de tiempo más significativo se produce al momento de analizar cada transacción y no al momento de acceder la transacción. En [29] se pudo demostrar tal afirmación a través de información empírica. Esto se debe a que por cada transacción se deberá analizar si el conjunto de

itemset a calcular su soporte son subconjuntos de la transacción o si los subconjuntos de dicha transacción pertenecen al conjunto de itemset. Este análisis de subconjunto depende grandemente en la longitud de las transacciones y los algoritmos existentes para tal comparación no son muy eficientes (tal como será mostrado en próximas secciones).

2.4.2. Representación Vertical

Una forma alternativa de representar la base de datos transaccional es en forma vertical. En este caso la base de datos consiste de un conjunto de tuplas $(item, tidset)$. Donde cada item i pertenece al conjunto de items $\mathcal{I}$ y $tidset$ es el cubrimiento[6] del item i. Existen numerosos algoritmos que utilizan este tipo de representación debido a sus ventajas con respecto a la representación horizontal, entre ellos [31, 22]. Este tipo de representación también admite la representación por medio de matriz bidimensional la cual es posible comprimir y lograr superar un algoritmo horizonal optimal que tenga conocimiento completo de los itemset frecuentes que solo necesita buscar el valor de su frecuencia [85]. Entre los algoritmos que utilizan una representación vertical con matriz bidimensional comprimida podemos mencionar `Pascal` [69], `Mafia` [14] analizado en la sección 3.2.5 y `ChARM` [86] analizado en la sección 3.3.4.

En el cuadro 2.4 se ilustran la bases de datos anteriormente vista bajo una representación horizontal, ahora utilizando una representación vertical tradicional (izquierda) y una representación vertical con matriz bidimensional (derecha). Es interesante notar que son similares las representaciones bitmap tanto horizontal como vertical, sin embargo, las horizontales están organizadas por transacciones (filas) y las verticales por items (columnas).

Consideremos un itemset P y los siguientes items X e Y tal que se obtienen los siguientes itemset $PX = P \cup \{X\}$ y $PY = P \cup \{Y\}$, es decir PX y PY difieren solo en un item. El itemset P es denominado el prefijo de ambos itemsets PX y PY bajo alguna relación de orden.

El itemset PXY podrá ser generado por la intersección de los cubrimientos ($tidset$

[6] Ver definición 2.1.5 en página 12.

A	B	C	D
2	1	1	1
3	3	2	2
4	5	4	3
5	7	5	6
6	8	6	10
7	9	8	
8	10	9	
9		10	
10			

tid	A	B	C	D
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	0	1	0
5	1	1	1	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	0
9	1	1	1	0
10	1	1	1	1

Cuadro 2.4: Representación Vertical de una Base de Datos

o *tidlist*) de 2 subconjuntos inmediatos que tenga el mismo prefijo y que unidos formen PXY, es decir $PXY = PX \cup PY$. El soporte de PXY es calculado obteniendo la cardinalidad del conjunto.

En particular, el soporte de un itemset PXY puede ser calculado intersectando los *tidset* de los items que componen a PXY como se verá formalmente en el lema 2.6.2. Esto reduce significativamente la cantidad de accesos a almacenamiento secundario ya que no es necesario analizar todas las transacciones en cada caso, sino simplemente aquellas que son significativas [22]. Se puede apreciar en la figura 2.1 como la intersección de los itemsets es realizada partiendo del conjunto de *tidset* iniciales (los *tidsets* de los 1-itemset).

Es interesante notar que en el caso de bases de datos ralas, los *tidset* proporcionan una forma mucho más concisa de representar la información intermedia. Permitiendo de esta forma que mayor información pueda ser mantenida en memoria.

Esta representación es especialmente útil para ciertos algoritmos que realizan un análisis de itemset frecuentes generando los mismos en forma Depth-First[7], ya que la información necesaria para realizar el cómputo del soporte va disminuyendo a medida que se baja en profundidad (itemset más largos) con un mejor aprovechamiento de la memoria.

A pesar de estas ventajas de la representación vertical, en el caso de bases de datos densas donde existen muchos itemset frecuentes y por lo tanto la longitud de los mismos es

[7]Ver sección 3.2.1 página 64.

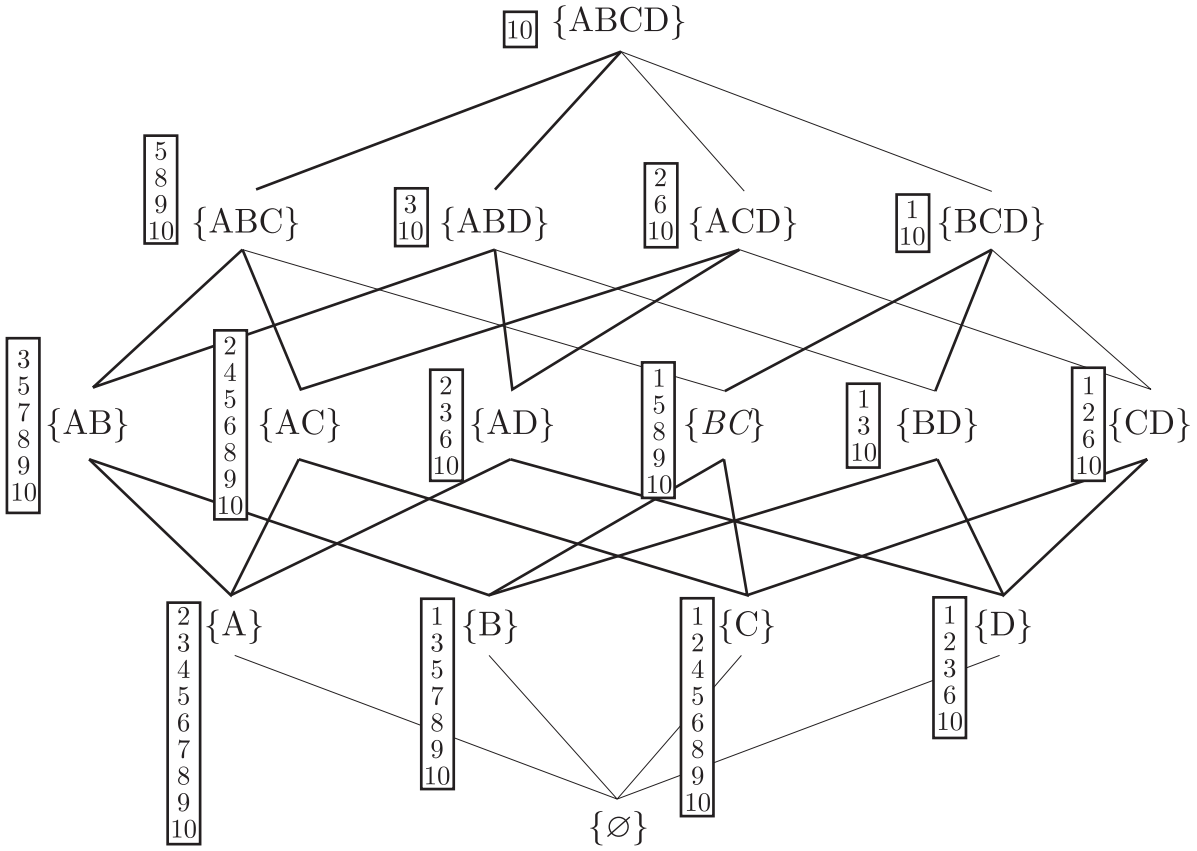


Figura 2.1: Tidsets de Itemsets

considerable, las intersecciones entre itemset requieren almacenar información intermedia en almacenamiento secundario. Por lo cual esta representación en estos dominios tiene una performance similar a la representación horizontal.

Una alternativa para estos casos de base de datos densas es utilizar compresión de bitmaps como en el caso de la representación horizontal. Esta alternativa es utilizada en [22]. Otra alternativa, recientemente propuesta en [85] y utilizada en [31][8], es utilizar el concepto de diferencias.

Diffsets

Una característica importante en las bases de datos densas es que existe una gran cantidad de itemset frecuentes, esta situación solo es posible si existen itemset frecuentes

[8]Ver sección 3.2.6 página 85.

largos. Por lo tanto, la diferencia en soporte entre un itemset frecuente de longitud k y un subconjunto inmediato de dicho itemset (longitud $k-1$) es altamente probable de ser muy poca. La idea del concepto de diferencias es propagar las transacciones en que difieren un $(k-1)$-itemset de un k-itemset, en lugar de propagar las transacciones.

Consideremos un itemset P y los siguientes items X e Y tal que se obtienen los siguientes itemset $PX = P \cup \{X\}$, $PY = P \cup \{Y\}$ y $PXY = PX \cup PY$, es decir PX y PY difieren solo en un item. Se llamará *diffset* a la diferencia entre dos *tidset* [85], notaremos como $d(PX)$ al conjunto de transacciones en la que P ocurre y no aparece X, es decir, $d(PX) = t(P) - t(X)$. Según mencionáramos anteriormente, como la diferencia de soporte es pequeña en bases de datos densas entonces este conjunto debería ser significativamente menor que el conjunto de *tidset*.

Con respecto al soporte del itemset PX, este no puede ser calculado computando la cardinalidad del conjunto $t(PX)$ ya que solamente se tiene al conjunto diferencia $d(PX)$. Sin embargo, al conocer $t(P)$ el soporte de PX puede calcularse como,

$$soporte(PX) = \mid t(P)\mid - \mid d(PX)\mid \tag{2.1}$$

En la práctica simplemente se propaga el soporte de P ($soporte(P) = \mid t(P)\mid$)

Para el caso general, para un itemset PXY solo se cuenta con $\mid t(P)\mid$, $d(PX)$ y $d(PY)$. Entonces $d(PXY) = t(PX) - t(Y) = t(PX) - t(PY)$ debe ser reexpresado de la siguiente forma,

$$\begin{aligned} d(PXY) &= t(PX) - t(PY) \\ &= t(PX) - t(PY) + t(P) - t(P) \\ &= - - (t(PX) - t(PY) - t(P) + t(P)) \\ &= -(-t(PX) + t(PY) + t(P) - t(P)) \\ &= -(t(P) - t(PX)) - t(PY) + t(P) \\ &= (t(P) - t(PY)) - (t(P) - t(PX)) \\ &= d(PY) - d(PX) \end{aligned} \tag{2.2}$$

Entonces el soporte de un itemset genérico PXY puede calcularse utilizando simplemente *diffsets* como resultado de las fórmulas 2.1 y 2.2.

$$soporte(PXY) = soporte(PX) - \mid d(PXY)\mid$$

En el inicio del proceso se utilizará la representación con *tidset* la cual puede ser cambiada a *diffset* en cualquier momento del proceso. Sin embargo es necesario que al menos 1 nodo (raíz) contenga el *tidset* sobre el cual los demás realizar la diferencia. En la figura 2.2 se muestra la generación de itemset frecuentes utilizando *diffsets*.

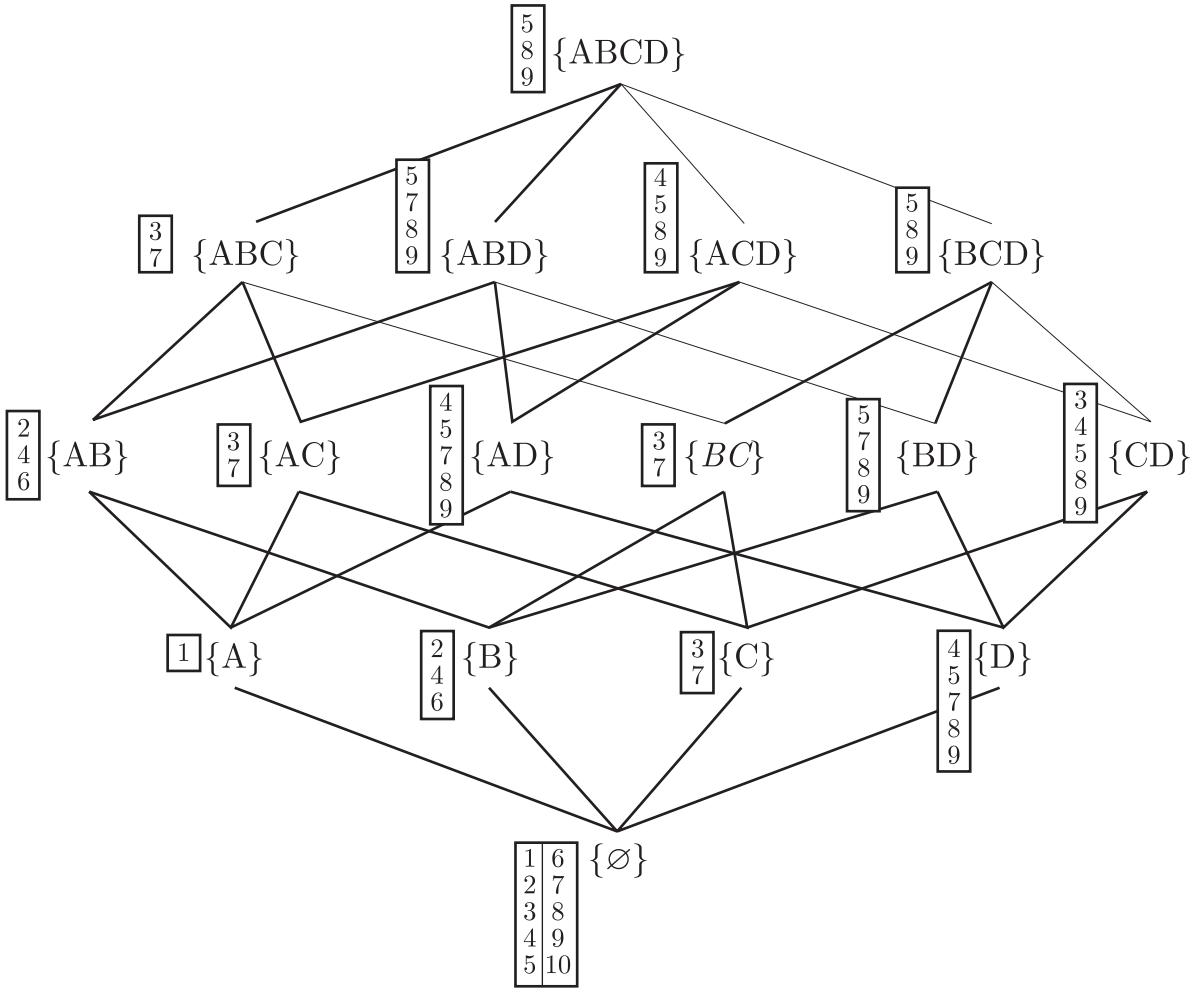


Figura 2.2: Diffsets de Itemsets

Las pruebas empíricas realizadas y reportadas en [85] confirmaron las afirmaciones

teóricas que los *diffsets* reducían significativamente el espacio necesario para el computo del soporte en base de datos relativamente densas. Sin embargo, se comprobó que sobre bases de datos ralas, el espacio necesario utilizado por los *diffsets* superaba el espacio necesario para almacenar la base de datos completa. Como resultado, sacando provecho de la facilidad para pasar de un esquema al otro, se propuso comenzar con el formato vertical más apropiado para la base de datos y en cierto punto del proceso de minería realizar el cambio al otro formato. Este punto de traspaso se define a través de una *razón de reducción* (reduction ratio) que se define para un itemset dado como la cardinalidad de su *tidset* sobre la cardinalidad de su *diffset*. Si este valor es de al menos 1 entonces es conveniente realizar la conversión a *diffset*[9]. Básicamente esta razón intenta capturar la diferencia en soporte entre un itemset y su itemset inmediato, como mencionáramos anteriormente si esta diferencia es poca la representación con *diffset* es la apropiada.

2.4.3. Representaciones Especiales

Además de la representación horizontal y vertical, se han propuesto otras alternativas las cuales pueden verse como una mezcla entre las representaciones anteriores. Esta clase de estructuras híbridas en algunos casos resulta en una representación más eficiente de la base de datos con menor espacio necesario y mejor performance en tiempo de testeo por pertenencia de elementos al conjunto de datos como es el caso de la estructura *FPTree*. Esta es similar a una estructura *trie* [3] y ha sido propuesta en [38] para la búsqueda de patrones frecuentes (FI).

La estructura *FP-Tree* es un árbol de prefijos extendido que almacena la información relevante de la base de datos en función de la frecuencia de los patrones. Consta solo de nodos de 1-itemset y estos son organizados de forma que los nodos más frecuentes se encuentran más cerca de la raíz. Con ello los items que más ocurren tienen más posibilidades de compartir un nodo. Como resultado de esta estrategia resulta una estructura altamente compacta.

[9]En [85] se puede encontrar el análisis realizado para arribar a dicha conclusión.

Definición 2.4.1 (FPTree, Han et al)

Un árbol de patrones frecuentes (Frequent Pattern Tree) (FP-Tree) consiste de:

1. *Una raíz con etiqueta "nula"*

2. *Un conjunto de subárboles de items prefijos (item prefix subtrees) (hijos de la raíz), los cuales consisten de* 3 *campos: a) Nombre del item. b) Contador de frecuencia. c) Enlace a otro nodo (Puntero).*

3. *Una tabla de items frecuentes, donde cada entrada consiste de* 2 *campos: a) Nombre del item. b) Enlace al nodo que primero aparece en el FP-Tree.* ♦

La construcción del *FPTree* se realiza efectuando dos pasadas sobre la base de datos:

1. Realizar una pasada sobre la base de datos $\mathcal{D}$, construyendo el conjunto de 1-itemset frecuentes $\mathcal{F}$ junto con sus soportes (σ). Ordenar $\mathcal{F}$ en orden descendente en función al soporte, generando una nueva lista $\mathcal{L}$.

2. Crear las estructuras *FPTree* ($\mathcal{T}$) y la tabla de cabeceras ($\mathcal{H}$).

   a) Crear la raíz de $\mathcal{T}$ y etiquetarla como "null".

   b) Para cada transacción presente en $\mathcal{D}$:

   - Seleccionar los items frecuentes y ordenarlos de acuerdo a la lista $\mathcal{L}$.
   - Sea $\mathcal{S} = \langle s_1, s_2, \ldots, s_n \rangle$ tal que $\sigma(s_i) \geq \sigma(s_j)$ y $1 \leq i \leq j \leq n$. Invocar al procedimiento $Insertar(\mathcal{S}, \mathcal{T})$ que recorrerá los elementos de $\mathcal{S}$ en orden y los insertará en el árbol $\mathcal{T}$ a partir del nodo "null" como se ilustra en el ejemplo 2.4.1.

Ejemplo 2.4.1

Supongamos tener el siguiente extracto de base de datos, donde hemos tomado solo 10 transacciones con un umbral de soporte mínimo de 3 (min_sup $= 3$*).*

TID	Items						
1	g	a	h	c	k	n	o
2	f	m	w	u	a	t	n
3	v	x	d	w	c	r	n
4	r	l	g	n	d	e	m
5	g	e	p	o	h	f	s
6	a	p	g	c	r	l	
7	m	n	a	t	w	f	
8	s	a	r	n	k	m	x
9	t	a	f	b	y		
10	g	x	k	m	d	r	n

Realizando una pasada sobre la base de datos anterior, obtenemos la siguiente lista $\mathcal{L}$ *de items ordenados en forma descendente por frecuencia.*

$\mathcal{L}$

Item	σ
n	7
a	6
g	5
m	5
r	5
f	4
c	3
d	3
k	3
t	3

$\mathcal{L}$

Item	σ
w	3
x	3
e	2
h	2
l	2
o	2
p	2
s	2
b	1
u	1

$\mathcal{L}$

Item	σ
v	1
y	1
i	0
j	0
q	0
z	0

A partir de esta lista podemos simplificar la base de datos anterior transformándola en una nueva base de datos, que contiene solo los items frecuentes de cada transacción, ordenados de acuerdo a $\mathcal{L}$*. En la práctica, este paso no es realizado, ya que el* FPTree *se va completando simplificando y ordenando cada transacción, sin generar una nueva tabla.*

TID	Items Frecuentes
1	n a g c k
2	n a m f t w
3	n r c d w x
4	n g m r d
5	g f
6	a g r c
7	n a m f t w
8	n a m r k x
9	a f t
10	n g m r d k x

A continuación se muestra la estructura FPTree *resultante de ingresar cada transacción el árbol de prefijos, junto con la tabla de enlaces de items $\mathcal{H}$. Esta tabla tiene un enlace a la transacción donde el item aparece por primera vez. A su vez, el item apuntado tiene un campo de enlace hacia otra transacción que posee el mismo item. Estos enlaces deben ser actualizados cuando una transacción es incorporada al* FPTree.[10] $\Diamond$

El algoritmo de construcción de *FPTree* realiza 2 pasadas sobre la base de datos $\mathcal{D}$. Durante la primer pasada se determinan los items frecuentes y luego durante la segunda pasada se procede a construir el *FPTree*. El costo de insertar cada transacción en el *FPTree* es del orden de la longitud de la transacción, denotado como $O(|Trans|)$.

Uno de los beneficios más importantes de esta estructura es el tamaño del *FPTree* comparado contra la base de datos $\mathcal{D}$. Frecuentemente sucederá que el *FPTree* será mucho más pequeña que la base de datos, ya que prefijos comunes de varias transacciones son fusionados, reduciendo el número de nodos necesarios.

Otra característica importante de esta estructura es que cuanto más frecuentes sean los items, más cercanos a la raíz del árbol *FPTree* se encuentran. Si los items más frecuentes fueran ubicados al final de cada patrón, estos generarían un nodo en el *FPTree* por cada patrón donde el item aparece, en cambio si los ubicáramos acorde a su frecuencia (de arriba hacia abajo), aquellos patrones con el mismo prefijo compartirían estos nodos,

[10] Los enlaces en la práctica no se generan de izquierda a derecha, sino que se generan a medida que se insertan los items en el árbol $\mathcal{T}$. Por simplicidad mostramos los enlaces de izquierda a derecha.

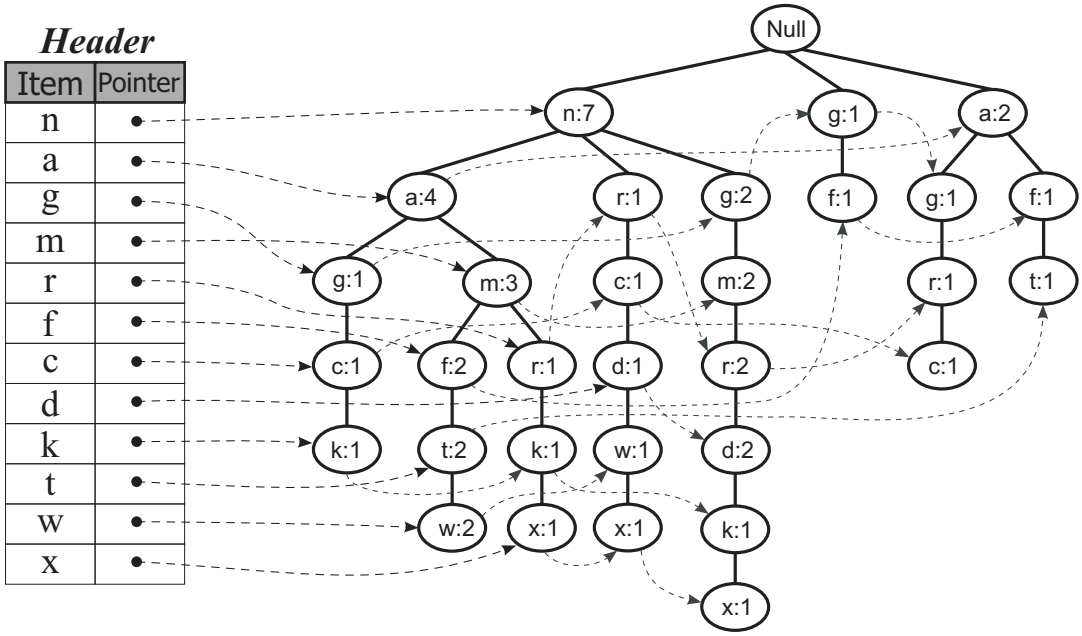


Figura 2.3: *FPTree y Tabla de Cabeceras*

reduciendo su número si estos items fueran muy frecuentes.

Además, se puede observar que un cambio en el umbral del soporte, no afecta demasiado el tamaño del *FPTree* y esto permite que los algoritmos que utilizan esta estructura escalen realmente bien, ante cambios en el umbral del soporte.

2.4.4. Repositorios de Datos

Los algoritmos que presentaremos en esta tesis son algoritmos genéricos que utilizan como entrada alguna de las representaciones de base de datos antes mencionadas. Estos algoritmos obtienen información previamente desconocida y potencialmente útil a partir de estos conjuntos de datos. Ya que existen diversos algoritmos propuestos para realizar la misma tarea, es especialmente interesante poder realizar comparaciones entre los algoritmos para determinar aquel que sea más apropiado o más eficiente. Usualmente, el estudio de algoritmos para su comparación se basa en conceptos sobre la complejidad de los mismos, independientemente de las instancias particulares con las cuales esos algoritmos son ejecutados.

Sin embargo, dentro del campo de estudio de la minería de datos, el estudio de la complejidad de los algoritmos no es un tema trivial y actualmente existen muy pocos resultados al respecto. Por tal motivo, el concepto de orden de ejecución de los algoritmos es tratado un poco informalmente en la literatura, haciendo referencia usualmente a pruebas empíricas sobre algunos conjuntos de datos recolectados lo suficientemente grande para apreciar diferencias y evaluar la aplicación de los algoritmos a dominios reales o a datos sintetizados.

Existen entonces una colección de repositorios de conjuntos de datos públicos como también una serie de programas que permiten generar un conjunto de datos con ciertas características para poder analizar la respuesta de los algoritmos a cierto tipos de datos. Las bases de datos reales usualmente son las que más peso tienen en la comparación entre los algoritmos. Los otros tipos de conjuntos de datos, también llamados sintetizados, están basados en funciones estadísticas y usualmente son utilizados para verificar el comportamiento de los algoritmos ante conjuntos de datos extremos, por ejemplo, el caso de un conjunto de datos muy denso o muy ralo.

Los siguientes son los repositorios de datos más importantes y comúnmente utilizados en la comparación de algoritmos:

- `PUMS Census`: Consiste de datos de la web sobre registro de personas y viviendas.

  `http://augustus.csscr.washington.edu/census/comp_013.html`

- `Irvine Machine Learning Database Repository`

  `http://www.ics.uci.edu/~mlearn/MLRepository.html`

  Consiste de varios conjuntos de datos:

  Splice Contiene datos recolectados sobre ADN.

  Mushroom Contiene datos describiendo características de diversas especies de hongos.

  Connect-4 Contiene una compilación de datos sobre el estado del juego Connect-4.

Chess Contiene una compilación de datos sobre estados de partidos de ajedrez.

- `Inter-University Consortium for Political and Social Research (ICPSR) - Census 2000`: Consiste de varios conjunto de datos sobre censos realizados cada 10 años con información sobre preguntas a personas sobre su vivienda, raza, edad, sexo, origen, relación familiar, ocupación, etc. Este relevamiento es sobre personas de Estados Unidos y colonias.

  `http://www.icpsr.umich.edu/CENSUS2000`

La siguiente tabla comparativa muestra las características de los conjunto de datos anteriormente mencionados. Puede observase que en dicha tabla aparecen algunos nombres de conjuntos de datos que no fueron mencionados como *pumbs, pumbs*, retail* y *splice*, dichos conjuntos de datos pertenecen a PUMS Census y ICPSR como particiones de dichos conjuntos o variaciones.[11]

Dataset	Registros	Longitud en Promedio de Registros
chess	3,196	37
connect-4	67,557	43
mushroom	8,124	23
pumsb	49,046	74
pumsb*	49,046	50
retail	213,972	31
splice	3,174	61

Los conjuntos de datos sintetizados usualmente son generados por el software provisto por el grupo de investigación Quest en IBM Almaden[12], donde se cuenta con el código para sintetizar conjuntos de datos para patrones frecuentes, patrones secuenciales frecuentes, problemas de clasificación y bases de datos densas.

Estos generadores requieren una determinada cantidad de parámetros para personalizar las características de los conjuntos de datos sintetizados. Las bases de datos sintetizadas utilizan, usualmente, como estándar el siguiente formato:

$$\mathrm{T}x\mathrm{I}y\mathrm{D}z\mathrm{K}.\mathrm{data}$$

[11] Estas variaciones generan distintos conjuntos de datos que como en el caso de *pumsb* se puede obtener un conjunto de datos denso o no (*pumbs**).

[12] `http://www.almaden.ibm.com/software/quest`

Donde x la cantidad promedio de items por transacción, y representa el tamaño promedio de los patrones y z representa la cantidad de transacciones (en miles). Por ejemplo, el conjunto de datos T5I4D100K.data contiene 100000 transacciones, el tamaño promedio de las transacciones de 5 items y el tamaño promedio de los patrones es de 4 items. Internamente, los conjuntos de datos están organizados utilizando el siguiente formato:

cid	tid	#items	itemset
⋮	⋮	⋮	⋮

Donde *cid* es un identificador de cliente, *tid* es un identificador de transacción, *#items* informa el tamaño del itemset y por último *itemset* describe los items presentes en la transacción.

2.5. Minería de Patrones Frecuentes

El problema de *minería de patrones* (pattern mining) puede ser definido de la siguiente forma:

Problema 2.5.1 (Pattern Mining)
Dado un conjunto de items $\mathcal{I}$, una base de datos transaccional $\mathcal{D}$ sobre $\mathcal{I}$ y un umbral de soporte mínimo δ, encontrar el conjunto de itemsets frecuentes $\mathcal{F}(\mathcal{D}, \delta)$. □

Este problema es un gran desafío ya que el espacio de búsqueda es exponencial sobre el número de items que aparecen en la base de datos. En particular si el número de diferentes itemsets posibles es $\mathcal{I}$, el espacio de búsqueda constará exactamente de un total de $2^{|\mathcal{I}|}$ itemsets. Obviamente, para un valor razonable de $\mathcal{I}$ para una aplicación real, la enumeración de todos los itemsets posibles es una tarea inviable. Por ejemplo para una aplicación sobre investigación genética para causas de enfermedades, es conocido que todo ser viviente está conformado de secuencias de ADN las cuales contienen 4 bloques básicos (nucleotidos), adenina (A), citosina (C), guanina (G) y timina (T). Un simple gen humano está compuesto de cientos de nucleotidos ordenados en un orden particular. Los seres humanos contienen alrededor de 100000 genes, por lo cual la cantidad de diferentes patrones diferentes a considerar es 2^{100000}, lo cual es totalmente inviable.

Por tal motivo se han propuesto varias alternativas que generan la totalidad del conjunto de itemsets frecuentes explorando un subconjunto del espacio de búsqueda. Este subconjunto de itemset es denominado *itemsets candidatos*. Formalmente,

Definición 2.5.1 (Itemset Candidato)
Dada una base de datos transaccional $\mathcal{D}$, un umbral de soporte mínimo δ y un algoritmo que computa $\mathcal{F}(\mathcal{D}, \delta)$, un itemset I es llamado candidato *si el algoritmo evalúa si I es o no frecuente.* ♦

Este subespacio de itemset candidatos es identificado utilizando algunas propiedades de los itemsets, de las cuales la más importante y más utilizada es la siguiente

Proposición 2.5.1 (Monotonicidad del Soporte)
Sean $X, Y \subseteq \mathcal{I}$ dos itemsets, $\mathcal{D}$ una base de datos transaccional sobre $\mathcal{I}$. Entonces se cumple que:

$$X \subseteq Y \Rightarrow soporte(Y) \leq soporte(X)$$

Demostración:
La prueba es inmediata de considerar que $cubre(Y) \subseteq cubre(X)$. ■

Por lo tanto obtenemos la siguiente propiedad que es común a todos los algoritmos analizados.

Propiedad 2.5.1 (Anti-monotonía)
Si un itemset es infrecuente, todos sus superconjuntos deben ser infrecuentes. Puede verse además que esta propiedad es cerrada en subconjuntos, es decir que si un itemset es frecuente todos sus subconjuntos son frecuentes. □

Esta dualidad ha sido explotada por diferentes algoritmos para recorrer el espacio de búsqueda tanto en forma top-down, bottom-up y bidireccional.

2.5.1. Apriori

El primer algoritmo eficiente para la búsqueda de itemsets frecuentes fue denominado *Apriori* y fue propuesto en [2]. Este algoritmo recorre el espacio de búsqueda por niveles generando el conjunto de itemsets candidatos C_k de longitud k antes de generar el conjunto de itemsets candidatos C_{k+1}. En primer lugar, se genera el conjunto de los 1-itemset frecuentes $\mathcal{F}_1$. A partir de este conjunto se genera el conjunto de candidatos C_2, sobre los cuales se encuentra el conjunto de los 2-itemset frecuentes $\mathcal{F}_2$. El proceso continúa hasta que ningún k-itemset frecuente puede ser encontrado.

Aún cuando la cantidad de itemset es finita, la generación completa de todos los itemset es inviable. Por dicha razón, este algoritmo saca provecho de la propiedad 2.5.1 de antimonotonía[13] de los itemsets para podar grandes cantidades de itemsets del espacio de búsqueda.

Básicamente, *Apriori* realiza iterativamente dos pasos:

1. *Paso de Unión*: Utilizando el conjunto de k-itemset frecuentes $\mathcal{F}_k$, se genera el conjunto de itemset candidatos frecuentes C_{k+1}. Esto se logra uniendo entre sí todos los pares de itemset en $\mathcal{F}_k$ que tengan un igual prefijo de longitud $k-1$. Por ejemplo, el conjunto de 3-itemsets $\{ABC, BDE, ABG, BDF, DEF\}$ generará un conjunto de candidatos $\{ABCG, BDEF\}$.

2. *Paso de Poda*: El conjunto de candidatos C_{k+1} deberá ser evaluado para verificar si contiene solamente itemset frecuentes. Realizando una pasada sobre la base de datos se descartan todos aquellos itemset de C_{k+1} que su soporte no supere el umbral mínimo. Sin embargo, para reducir este proceso se aplica la propiedad de antimonotonía, eliminando de C_{k+1} todos aquellos itemset que contengan subconjuntos no frecuentes, es decir, para cada subconjunto de longitud k de cada itemset de C_{k+1} se determina si pertenece a $\mathcal{F}_k$. Observar, en el ejemplo anterior, el itemset $\{ABCG\}$ es eliminado ya que $\{BCG\}$ no pertenece a $\mathcal{F}_3$, en cambio $\{BDEF\}$ es mantenido

[13] Algunos algoritmos denominan a esta propiedad como *propiedad Apriori* por ser este el primer algoritmo en utilizarla.

y se evalúa su soporte.

El problema principal de este algoritmo yace en la cantidad de pasadas que son necesarias realizar sobre la base de datos, la cual usualmente no es lo suficientemente pequeña como para poder ser mantenida en memoria. Por esta razón, se realizaron muchas propuestas basadas en este algoritmo y la mayoría de ellas utilizan como parte de su estrategia la propiedad de antimonotonía. Una explicación detallada de tales algoritmos se encuentra fuera de ámbito de esta tesis, por lo tanto simplemente se abordará un algoritmo mas reciente que ha de ser utilizado como base de otro algoritmo en esta tesis y que ha demostrado ser de mucho más eficiente que *Apriori* gracias al uso de una representación más compacta del conjunto de datos.

2.5.2. FPGrowth

De acuerdo a análisis realizados sobre algoritmos que utilizan *Apriori* como método para la generación de itemset frecuentes, se ha podido detectar que las operaciones más costosas de estos algoritmos son la *a*) generación de candidatos y *b*) las operaciones de comparación de patrones.

- *Generación de Candidatos*: Al utilizar bases de datos con una cantidad considerable de items, es normal encontrarse con una cantidad mayor a 10000 (10^4) items posibles. Si por ejemplo, se solicitara a algún algoritmo como *Apriori* que encuentre un itemset frecuente de longitud 100, en el mejor de los casos este algoritmo obtendría itemset candidatos solo con los items de este itemset frecuente (los cuales no se conocen, ya que el algoritmo es bottom-up y por lo tanto se conoce un superconjunto del mismo), por lo cual el total de itemsets generados es igual a $C_{100}^{100} + C_{99}^{100} + \cdots + C_{1}^{100} = \sum_{i=1}^{100} \binom{n}{i} = 2^{100} \approx 10^{30}$ candidatos. Este costo es prohibitivo para cualquier algoritmo.

- *Operaciones de comparación de patrones*: Siguiendo con el ejemplo anterior, cada vez que es generado un nuevo grupo de itemset debe ser realizada una pasada por la base de datos, comparando estos itemset contra los items de las transacciones,

con el objetivo de determinar cuál de estos itemset es frecuente. En consecuencia, además de realizar 100 pasadas por la base de datos, se está realizando una gran cantidad de comparaciones.

El algoritmo *FPGrowth*, propuesto en [38], intenta solucionar ambos problemas utilizando la estructura *FPTree* y un acercamiento "dividir y conquistar".

La principal ventaja de este método es que para generar todos los patrones frecuentes solo necesita 2 pasadas sobre la base de datos, sin importar la longitud de los itemset frecuentes. La primer pasada es utilizada para encontrar los 1-itemset frecuentes y en la segunda se construye el *FP-Tree*. Las operaciones restantes constan de explorar recursivamente el *FP-Tree*. En caso que el árbol no pueda ser alojado en memoria principal, sus primeras operaciones serán tan costosas como los accesos a la base de datos.

Como hemos visto la estructura *FP-Tree* provee una forma compacta de almacenar la información de la base de datos $\mathcal{D}$. Sin embargo, esta estructura no asegura automáticamente que el método sea eficiente, ya que aún podemos generar a partir del *FP-Tree* todo el conjunto de patrones candidatos y resolver el problema de encontrar los conjuntos frecuentes con algún método del estilo *Apriori*.

Básicamente el algoritmo propuesto se puede descomponer en dos etapas:

1. Crear el *FPTree* utilizando la base de datos $\mathcal{D}$ y un umbral de soporte mínimo.

2. Invocar al algoritmo *FP-Growth* con el árbol anterior como entrada y retornando como salida el conjunto de itemset frecuentes.

La estructura informal de *FP-Growth* puede apreciarse en el algoritmo 1. Allí se mencionan algunos términos como base de patrón condicional y crecimiento de sufijos. A continuación se mencionarán los conceptos básicos utilizados por dicho algoritmo.

Este algoritmo inicia su ejecución transformando el conjunto de datos en una estructura compacta (*FPTree*). Luego proyecta la base de datos para cada item, comenzando por el sufijo del posible itemset frecuente. Esta proyección es denominada *base de patrón condicional* (Conditional Pattern Base) (CPB). Sobre esta nueva base de datos proyectada (una por cada item) se genera un nuevo *FPTree*, esta vez llamado *FPTree Condicional*.

Algoritmo 1 Algoritmo Informal FP-Growth

1: **Si** el árbol consta de un solo camino **entonces**
2: Listar todos los itemset posibles de ese camino (conjunto de partes);
3: **sino**
4: Generar para cada nodo del árbol su *base de patrón condicional*;
5: Construir a partir de cada *base de patrón condicional* su correspondiente *FP-Tree*;
6: Aumentar el sufijo de los patrones obtenidos hasta el momento;
7: Invocar recursivamente a *FP-Growth*;
8: **fin Si**

Ejemplo 2.5.1

Considérese el ejemplo 2.4.1 de la página 30. Como se mencionara en aquel caso por cada item se generará un FPTree Condicional. *En la figura 2.4 se muestra la porción del* FPTree *original donde se encuentra w. Los prefijos de estas ramas constituyen la base de patrón condicional y su soporte es igual al soporte del itemset (w en el ejemplo) en dicha rama. A partir de esta base de patrón condicional se genera el nuevo* FPTree Condicional *para w utilizando el umbral de mínimo soporte con valor 3.* ◊

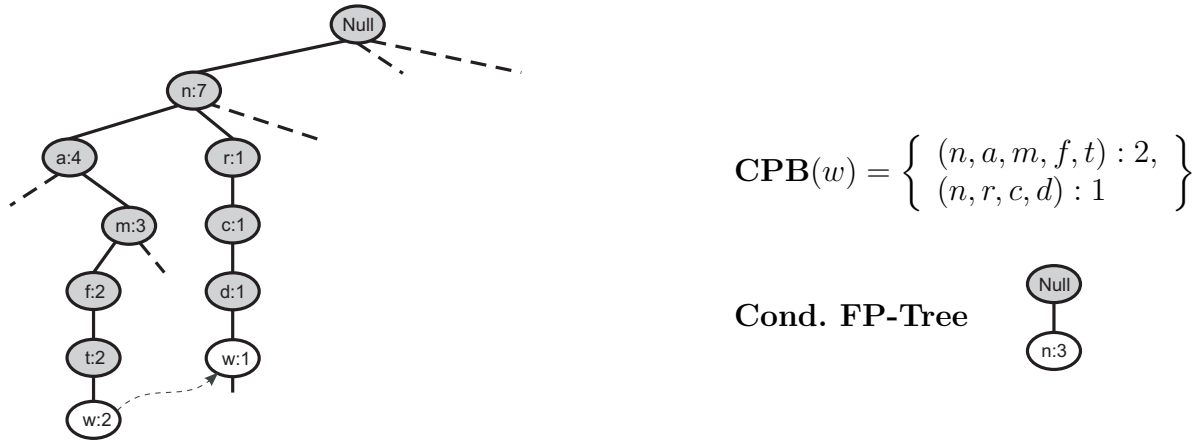


Figura 2.4: *FPTree Condicional para item w*

El algoritmo entonces procede en primer lugar a identificar los 1-itemset frecuentes en la base de datos $\mathcal{D}$ y construye su base de patrones condicionales, buscando luego los 1-itemset frecuentes en esta base de patrones condicional, sobre estos se construye otra base de patrones condicionales y se continua así sucesivamente hasta alcanzar el nodo

raíz.

El *Crecimiento del Patrón* se produce a medida que iteramos el método, ya que el primer item encontrado constituye el sufijo del itemset frecuente y los items frecuentes encontrados en la próxima iteración forman cada uno junto con el sufijo del paso anterior los nuevos sufijos de los itemset frecuentes. En cada paso recursivo, el algoritmo procede a incrementar el sufijo con los items frecuentes hasta alcanzar el nodo raíz. En el caso que un FP-Tree tenga un solo camino P, el conjunto de patrones frecuentes completo puede ser generado enumerando todas las combinaciones de los subcaminos de P con un soporte igual al mínimo entre el soporte de los items del subcamino. Una versión más completa de *FPGrowth* puede verse en el algoritmo 2.

Algoritmo 2 FP-Growth(FPTree,α): Dado un nodo α y su FPTree Condicional retorna los itemset frecuentes

```
1: Si FPTree contiene un camino único P entonces
2:    Para todo combinación β = α1α2...αk de nodos en el camino P hacer
3:       generar el patrón α ∪ β con soporte = min({αi|(1 ≤ i ≤ k)});
4:    fin Para
5: sino
6:    Para todo ai en la tabla de cabeceras de FPTree hacer
7:       generar el patrón β = α ∪ ai con β.soporte = ai.soporte;
8:       construir la base de patrones condicional de β;
9:       generar el FP-Tree condicional de β (FPTree_β);
10:      Si FPTree_β ≠ ∅ entonces
11:         invocar FP-Growth(FPTree_β, β);
12:      fin Si
13:   fin Para
14: fin Si
```

A pesar que los autores de *FPGrowth* han enunciado que este algoritmo soluciona el problema de la generación de candidatos, en [29] se demuestra que en realidad dicho resultado genera tantos candidatos como la técnica de generación de candidatos *Apriori*. Allí además es mencionado que debido a la complejidad inherente de la estructura *FPTree*, la eficiencia del algoritmo se puede apreciar realmente cuando se puede obtener una buena compresión de la base de datos.

FPGrowth*

Recientemente, se ha propuesto una mejora sobre el algoritmo básico *FP-Growth*, la cual consiste de utilizar una estructura de datos adicional junto con los *FP-Tree* para mejorar la eficiencia de este algoritmo. Esta estrategia ha sido propuesta en [32] y se basa en la siguiente observación. El trabajo principal del algoritmo *FP-Growth* es recorrer los *FP-Trees* construyendo nuevos *FP-Trees condicionales*. Por evidencia empírica se determinó que el 80 % del tiempo era utilizado para recorrer los *FP-Trees*. Por lo tanto, el objetivo de esta estrategia es reducir el tiempo necesario en recorrer estos *FP-Trees*.

El algoritmo *FP-Growth* realiza su tarea en primer lugar recorriendo el *FPTree Condicional* T_X[14] para cada item i que pertenece a la tabla de cabeceras de T_X buscando aquellos items que son frecuentes en la base de patrón condicional de $X \cup \{i\}$. Con estos items se construye la tabla de cabeceras condicional para el *FPTree Condicional* $T_{X\cup\{i\}}$. En una segunda pasada sobre T_X para cada item i se construye efectivamente $T_{X\cup\{i\}}$.

Ejemplo 2.5.2

Continuando con el ejemplo 2.4.1, para el nodo $i = r$ y $X = null$, es decir, tomando el FPTree *inicial T la base de patrón condicional para r y el* FPTree condicional T_r *se resume en la figura 2.5. En primer lugar, se genera la base de patrón condicional para r determinando los items que son frecuentes en dicha la base de patrón condicional, es decir, aquellos items j que junto con r son frecuentes en T. De la base de patrón condicional para r podemos ver que n, m y g son items frecuentes*[15] *y el único item no frecuente es a. Por último se realiza la segunda pasada sobre T generando el* FPTree condicional T_r.

La estrategia consiste en omitir la primer pasada sobre T_X para determinar cuáles son los items frecuentes en la base de patrón condicional de X. Esto se logra construyendo un arreglo A_X durante la construcción de T_X (es decir durante lo que hubiese sido la segunda pasada anterior). Dicho arreglo contiene el soporte de todos los 2-itemset presentes en T_X.

[14] En la primer iteración X corresponde con el nodo "null".
[15] Recordar que utilizábamos un umbral de mínimo soporte de 3.

$$\mathbf{CPB}(r) = \left\{ \begin{array}{l} (n, a, m) : 1, \\ (n) : 1, \\ (n, g, m) : 2, \\ (a, g) : 1 \end{array} \right\} \qquad \textbf{Cond. FP-Tree}$$

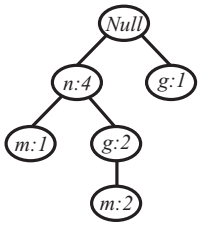


Figura 2.5: *FPTree Condicional para item r*

Ejemplo 2.5.3

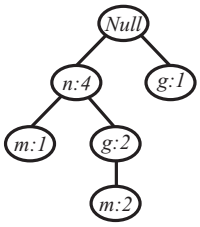
En el ejemplo anterior, cuando $X = null$ se realiza una pasada sobre la base de datos para construir la tabla de cabecera con los items frecuentes. Esta pasada solo es necesario en el primer caso. Se realiza una segunda pasada para construir el FPTree *T (T_X con $X = null$) y junto con esta pasada se construye el siguiente arreglo A (A_X con $X = null$) conteniendo todos los 2-itemset entre los items frecuentes.*

n												
a	4											
g	3	2										
m	5	3	2									
r	3	2	3	3								
f	2	4	1	2	0							
c	2	2	2	0	2	0						
d	3	0	2	2	3	0	1					
k	3	2	2	2	2	0	1	1				
t	2	3	0	2	0	3	0	0	0			
w	3	2	0	2	1	2	1	1	0	2		
x	3	1	1	2	3	0	1	2	1	0	1	
$\mathbf{A}_\emptyset$	n	a	g	m	r	f	c	d	k	t	w	x

Con el uso de esta estructura auxiliar se puede evitar realizar la primer pasada para cada item frecuente en $T_\emptyset$ ya que cada fila del arreglo $A_\emptyset$ contiene la información sobre los soporte del item junto con cada uno de los items que se encuentran arriba de este en alguna rama. Para el caso del item r, la fila de r en $A_\emptyset$ contiene la misma información que se obtendría de la base de patrón condicional.

r	3	2	3	3
$\mathbf{A}_\emptyset$	n	a	g	m

De esta forma se puede evitar la primer pasada para generar la tabla de cabeceras para r, ya que la misma puede ser generada a partir de la información de $A_\emptyset$ realizando una sola pasada sobre T para generar T_r. Es necesario notar que durante esta construcción de T_r se generará también el arreglo A_r, el cuál permitirá evitar la primer pasada sobre T_r en búsqueda de items frecuentes.



n			
g	2		
m	3	2	
$\mathbf{A}_r$	n	g	m

$\Diamond$

Con el uso de esta estrategia se mejora la eficiencia del algoritmo *FPGrowth* para los casos en los que el algoritmo funcionaba deficientemente, esto es cuando la base de datos era poco densa. Sin embargo, esta estrategia no es conveniente cuando la base de datos es muy densa. En estos casos se utiliza el algoritmo *FPGrowth* original.

2.6. Representación del Espacio de Búsqueda

Como hemos mencionado en la sección 2.5 el espacio de búsqueda asociado al problema de la minería de patrones frecuentes es exponencial en el número de items posibles ($2^{|I|}$ donde I es que conjunto de todos los items posibles). Sin embargo, existen varias representaciones para este espacio de búsqueda que son más apropiadas que la simple enumeración de todos los subconjuntos posibles. Estas representaciones permiten recorrer el espacio en forma ordenada y pudiendo aplicar heurísticas para permitir reducciones del mismo. Nos centraremos en las dos representaciones más utilizadas, *a*) Reticulados de Conjuntos y *b*) Árbol de Enumeración de Conjuntos.

Es importante mencionar que en este contexto tanto los reticulados de conjuntos como los árboles de enumeración de conjuntos no son estructuras de datos (como un hash, árbol trie, etc) sino que son utilizadas para ilustrar cómo los conjuntos de items son completamente enumerados en un problema de búsqueda.

2.6.1. Reticulados de Conjuntos

Los patrones están compuestos de items los cuales probablemente han sufrido algún cambio desde su estado original (por ejemplo, el proceso de discretización). Estos items son unidades indivisibles y a partir de ellos los patrones pueden ser comparados. Por tal motivo podemos establecer una relación entre patrones. Para una explicación más detallada puede consultarse [87, 88] o textos sobre matemática discreta [64].

Definición 2.6.1 (Orden Parcial, Zaki, 1998)
Sea P un conjunto. Un orden parcial *sobre P es una relación binaria $\preceq$, tal que para todo $X, Y, Z \in P$, la relación es:*

reflexiva $X \preceq X$.

antisimétrica $X \preceq Y$ *y* $Y \preceq X$ *implica* $X = Y$.

transitiva $X \preceq Y$ *y* $Y \preceq Z$ *implica* $X \preceq Z$.

El conjunto P con la relación $\preceq$ es llamado conjunto parcialmente ordenado (partially ordered set) o simplemente POSet. ♦

Sobre este poset se puede establecer otra relación.

Definición 2.6.2
Dado un poset $(P, \preceq)$. Se define la relación $\prec$ sobre P por:

$$X \prec Y \text{ si y solo si } Y \preceq Y \text{ y } X \neq Y$$

La relación anterior satisface las siguientes propiedades para todo $X, Y, Z \in P$:

antireflexiva $X \not\prec X$.

quasi-orden $X \prec Y$ *y* $Y \prec Z$ *implica* $X \prec Z$. ♦

Definición 2.6.3 (Cubre)

Dado un poset $(P, \preceq)$ *y* $X, Y \in P$ *diremos que* Y cubre a X *(*X es cubierto por Y*) si* $X \prec Y$ *y no existe* Z *tal que* $X \prec Z \prec Y$[16]. ♦

Podemos entonces establecer una relación de orden parcial entre el conjunto de todos los itemsets posibles.

Proposición 2.6.1

Sea S *el conjunto de items posibles y sea* $\mathcal{P}(S)$ *el powerset*[17] *(conjunto de partes) de* S*. La relación binaria* $\subseteq$ *establece un orden parcial sobre* $\mathcal{P}(S)$. ♦

La demostración de tal proposición es trivial y puede ser encontrada en la bibliografía sugerida.

Este poset $\mathcal{P}(S)$ sobre la relación $\subseteq$ $(\mathcal{P}(S), \subseteq)$ puede ser gráficamente visualizado utilizando un *diagrama Hasse* tal como se ejemplifica en la figura 2.6 utilizando un conjunto $S = \{A, B, C, D\}$. Se puede apreciar que la relación $\prec$ sobre el poset $(\mathcal{P}, \subseteq)$ es análoga a la definición **??** de subconjunto propio y la definición de cubrimiento es análoga a la definición **??** de subconjunto (superconjunto) inmediato.

Propiedad 2.6.1 (Subposet)

Un subconjunto L *de un poset* P *hereda el orden parcial sobre* P *y por lo tanto es un poset. Este subconjunto* L *es llamado* subposet *de* P. □

Definición 2.6.4

Sea P *un poset y sea* L *un subposet de* P*. Un elemento* $x \in P$ *es denominado límite superior (upper bound) para* L *en* P *si* $y \preceq x$ *para todo* $y \in L$*. Si* x *es un límite superior para* L *en* P *tal que* $x \preceq z$ *para cada límite superior* z *para* L *en* P *entonces* x *es denominado el menor límite superior (least upper bound) de* L *en* P*, notado* $x = lub(L) = \bigvee L$*. En forma similar, un elemento* $x \in P$ *es denominado límite inferior (lower bound) para* L *en* P *si* $x \preceq y$ *para todo* $y \in L$*. Si* x *es un límite inferior para* L *en* P *tal que*

[16] También puede ser expresado como $X \preceq Z \prec Y$ implica que $Z = X$.
[17] Ver def. **??** pág. **??**.

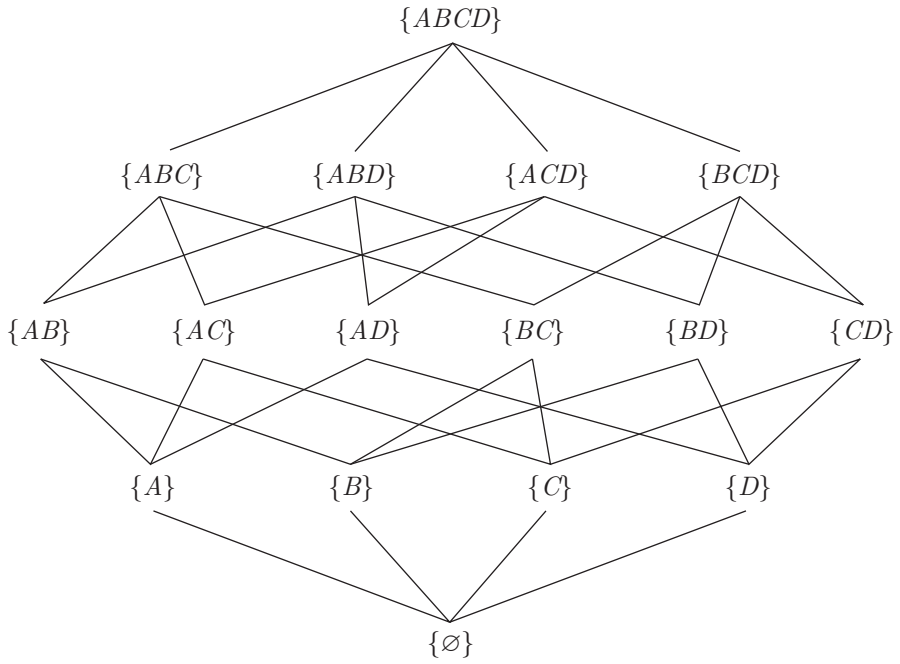


Figura 2.6: Reticulado para $\mathcal{P}(S)$ con $S = \{A, B, C, D\}$

$z \preceq x$ para cada límite inferior z para L en P entonces x es denominado el mayor límite inferior (greatest lower bound) de L en P, notado $x = glb(L) = \bigwedge L$. ♦

Propiedad 2.6.2

Dado un subposet S de P si existe $lub(S)$ o $glb(S)$ éstos son únicos. □

La demostración de la propiedad anterior es trivial ya que todo poset debe respetar la propiedad antisimétrica.

Definición 2.6.5 (Reticulado)

Un poset $(P, \preceq)$ es denominado reticulado *si para todo $x, y \in P$ existen el menor límite superior $lub(\{x, y\})$ y el mayor límite inferior $glb(\{x, y\})$.*[18] ♦

Propiedad 2.6.3 (Top y Bottom)

Sea L un reticulado y sean $X, Y \in L$. Si $Z \preceq X$ para todo $Z \in L$ entonces X es denominado límite superior universal (universal upper bound) notado como $\top$. En forma

[18] En lugar de utilizar las funciones *lub* y *glb* se puede utilizar una notación infija, por ej. $lub(\{x, y, z\}) = \bigvee \{x, y, z\} = x \vee y \vee z$, en forma similar $glb(\{x, y, z\}) = \bigwedge \{x, y, z\} = x \wedge y \wedge z$.

similar, si $Y \preceq Z$ *para todo* $Z \in L$ *entonces* Y *es denominado límite inferior universal (universal lower bound) notado como* $\perp$. □

Definición 2.6.6 (Átomo)

Si un reticulado L *tiene un limite inferior universal* $\perp$ *entonces los elementos que cubren a* $\perp$ *(*$\perp \prec I$*) son llamados* átomos. *El conjunto de átomos de* L *es notado* $\mathcal{A}(L)$. ♦

En [87] se hace notar que para el conjunto S, el poset $(\mathcal{P}(S), \subseteq)$ es un reticulado con límites inferiores y superiores universales ($\perp = \emptyset$ y $\top = S$). El conjunto de átomos $\mathcal{A}(\mathcal{P}(S))$ corresponde con el conjunto de items $\mathcal{I}$. Donde las funciones *lub* y *glb* están dadas por la unión y la intersección respectivamente.

$$lub(\{X, Y\}) = X \wedge Y = X \cup Y \qquad glb(\{X, Y\}) = X \vee Y = X \cap Y$$

Ejemplo 2.6.1

Consideremos el ejemplo de la figura 2.6. Dado el reticulado ($\mathcal{P}(\{A, B, C, D\}), \subseteq$*). Entonces,*

$$\begin{aligned}
\{A\} \vee \{C\} &= \{A, C\} \\
\{A, B\} \vee \{A, C\} &= \{A, B, C\} \\
\{A\} \wedge \{C\} &= \{\emptyset\} \\
\{A, C, D\} \wedge \{B, D\} &= \{D\} \\
\bigvee \{\{B, C\}, \{A\}, \{B, D\}\} &= lub(\{\{B, C\}, \{A\}, \{B, D\}\}) = \{A, B, C, D\} \\
\bigwedge \{\{A, B, C\}, \{B, C, D\}, \{A, B, D\}\} &= glb(\{A, B, C\}, \{B, C, D\}, \{A, B, D\}) = \{B\} \ \Diamond
\end{aligned}$$

Hasta aquí hemos analizado esta estructura con fines de representar el espacio de búsqueda para el problema de minería de itemsets frecuentes. A continuación proporcionaremos algunas definiciones que serán de utilidad en otros capítulos[19] donde se procederá a realizar algunas pruebas formales.

[19] A pesar de ser utilizadas en otro capítulo, creemos que este es el marco conveniente para su enunciación.

Definición 2.6.7 (Reticulado Booleano)

Un reticulado L es denominado reticulado booleano (Boolean Lattice) si

1. *L es* distributivo*, es decir, para todo $X, Y, Z \in L$, $X \wedge (Y \vee Z) = (X \wedge Y) \vee (X \wedge Z)$ y $X \vee (Y \wedge Z) = (X \vee Y) \wedge (X \vee Z)$*

2. *L tiene elementos $\top$ y $\bot$.*

3. *Cada elemento $X \in L$ tiene complemento, notado $\overline{X}$ o X^C, es decir $X \vee \overline{X} = \top$ y $X \wedge \overline{X} = \bot$.* ♦

Notemos que el reticulado $(\mathcal{P}(S), \subseteq)$ es un reticulado booleano. Recordar que los operadores $\wedge$ y $\vee$ en el contexto de este reticulado del conjunto de partes (powerset) de S corresponden con unión ($\cup$) e intersección ($\cap$) respectivamente.

Estos reticulados respetan las siguientes propiedades:

1. Es distributivo. Véase el cuadro **??** incisos $3a$ y $3b$ sobre leyes[20] del algebra de conjuntos.

2. Tiene límites universales superior e inferior ($\top$ y $\bot$). $\top = S$ y $\bot = \{\emptyset\}$. El elemento $\top$ en la teoría de conjuntos usualmente es notado como U el conjunto universal.

3. Todo elemento tiene complemento. Véase el cuadro **??** incisos $7a$ y $7b$.

Como ha sido mencionado anteriormente el conjunto de átomos de un reticulado corresponde al conjunto de items $\mathcal{I}$ para el caso del reticulado booleano del conjunto de partes de $\mathcal{I}$ ($\mathcal{I} = \mathcal{A}(\mathcal{P}(\mathcal{I}))$). En la sección 2.4 se mostró que existen diferentes representaciones del dataset, en particular, se observó que el dataset puede ser representado verticalmente asociando un *Tid-List* a cada itemset X, notado $\mathcal{L}(X)$, y que se podían realizar operaciones de unión e intersección de estos *Tid-List* para obtener los respectivos *Tid-List* de otros itemsets. A continuación enunciaremos un conjunto de lemas que formalizan las nociones vistas utilizando reticulados booleanos [87].

[20] Las demostraciones de tales leyes pueden encontrarse en la mayoría de los textos básicos sobre teoría de conjuntos.

Lema 2.6.1 (Zaki, 1998)

Para un reticulado booleano finito L, con $X \in L$, $X = \bigvee \{Y \in \mathcal{A}(L) | Y \preceq X\}$. □

Este lema establece que todo elemento del reticulado booleano puede ser obtenido a partir simplemente del conjunto de átomos o items.

Lema 2.6.2 (Zaki, 1998)

Para cualquier $X \in \mathcal{P}(\mathcal{I})$ sea $J = \{Y \in \mathcal{A}(\mathcal{P}(\mathcal{I})) | Y \preceq X\}$. Entonces $X = \bigcup_{Y \in J} Y$ y soporte$(X) = |\bigcap_{Y \in J} \mathcal{L}(Y)|$. □

Este lema en su primer parte es un caso particular del lema anterior instanciado para el reticulado booleano de partes de conjuntos. Por otro lado, en su segunda parte establece que el *tid-list* de cualquier itemset puede ser obtenido simplemente utilizando los *tid-list* de los items. Este resultado es importante ya que si el conjunto de items no es muy grande se podría tener toda la información necesaria en memoria sin realizar accesos a la base de datos original.

Ejemplo 2.6.2

Podemos ver que para el reticulado $(\mathcal{P}(\{A, B, C, D\}), \subseteq)$ y para el conjunto de items $\mathcal{I} = \{A, B, C, D\}$, los siguientes elementos pueden ser obtenidos simplemente utilizando el conjunto $\mathcal{I}$:

$$\begin{aligned}
\{A, B\} &= \{A\} \vee \{B\} = \{A\} \cup \{B\} \\
\{B, C\} &= \{B\} \vee \{C\} = \{B\} \cup \{C\} \\
\{A, B, C\} &= \{A, B\} \vee \{A, C\} \vee \{B, C\} = \{A\} \vee \{B\} \vee \{C\} = \{A\} \cup \{B\} \cup \{C\} \\
\{A, B, C, D\} &= \{A\} \vee \{B\} \vee \{C\} \vee \{D\} = \{A\} \cup \{B\} \cup \{C\} \cup \{D\}
\end{aligned}$$

*Asociaremos el siguiente conjunto $\mathcal{L}$ de identificadores de transacciones (*tid-list*) a cada*

uno de los elementos de $\mathcal{I}$*,*

$$\mathcal{L}(\{A\}) = \{1,3,4,5\} \qquad soporte(\{A\}) = |\mathcal{L}(\{A\})| = 4$$
$$\mathcal{L}(\{B\}) = \{1,2,3,4,5\} \qquad soporte(\{B\}) = |\mathcal{L}(\{B\})| = 5$$
$$\mathcal{L}(\{C\}) = \{2,4,5,6\} \qquad soporte(\{C\}) = |\mathcal{L}(\{C\})| = 4$$
$$\mathcal{L}(\{D\}) = \{1,3,5,6\} \qquad soporte(\{D\}) = |\mathcal{L}(\{D\})| = 4$$

Los tid-list *para cada uno de los ejemplos utilizados anteriormente pueden ser calculados de la siguiente forma:*

$$\mathcal{L}(\{A,B\}) = \mathcal{L}(\{A\}) \cap \mathcal{L}(\{B\}) = \{1,3,4,5\} \qquad soporte = 4$$
$$\mathcal{L}(\{B,C\}) = \mathcal{L}(\{B\}) \cap \mathcal{L}(\{C\}) = \{2,4,5\} \qquad soporte = 3$$
$$\mathcal{L}(\{A,B,C\}) = \mathcal{L}(\{A\}) \cap \mathcal{L}(\{B\}) \cap \mathcal{L}(\{C\}) = \{4,5\} \qquad soporte = 2$$
$$\mathcal{L}(\{A,B,C,D\}) = \mathcal{L}(\{A\}) \cap \mathcal{L}(\{B\}) \cap \mathcal{L}(\{C\}) \cap \mathcal{L}(\{D\}) = \{5\} \qquad soporte = 1 \quad \Diamond$$

A su vez como se ha mostrado para el caso de reticulados en general, cualquier itemset puede ser obtenido realizando operaciones de unión de al menos 2 de sus subconjuntos propios y su soporte puede obtenerse de forma análoga.

2.6.2. Árbol de Enumeración de Conjuntos

Otra forma de representar el espacio de búsqueda es mediante un árbol de subconjuntos, el cual consta de la enumeración ordenada de todos los subconjuntos del conjunto universal y de allí su nombre de *árbol de enumeración de conjuntos* (Set Enumeration Tree o SETree). Dicha representación ha sido propuesta en [66] y ha sido eficientemente aplicada para el aprendizaje inductivo [67] y en la generación de patrones maximales [6, 31, 14, 90].

Veamos entonces la siguiente definición en donde se define una vista parcial del árbol. Esta definición será la base para la construcción del SETree.

Definición 2.6.8 (Vista Parcial de un SETree)
Sea ind *una función que mapea cada elemento de un conjunto de atributos* $\mathcal{A}$ *con los*

naturales correspondiendo cada elemento con un valor distinto (ind : $\mathcal{A} \rightarrow \mathbb{N}$*)*[21]*. Se define una vista parcial de S como*

$$Vista(S) = \{A \in \mathcal{A} \mid \forall A' \in S,\ ind(A) > max(ind(A'))\} \qquad \blacklozenge$$

Consideremos el siguiente ejemplo para ilustrar estos conceptos.

Ejemplo 2.6.3

Sea el conjunto $\mathcal{A} = \{A, B, C, D\}$ *y se ind una función que mapea cada atributo con su posición en el alfabeto. Entonces* $ind(\{\emptyset\}) = 0$*,* $ind(A) = 1$*,* $ind(B) = 2$*,* $ind(C) = 3$ *y* $ind(D) = 4$*. Cada una de las vistas estará constituida de la siguiente forma:*

S	$max(ind(A'))$	$Vista(S)$
$\{\emptyset\}$	0	$\{A, B, C, D\}$
$\{A\}$	1	$\{B, C, D\}$
$\{B\}$	2	$\{C, D\}$
$\{C\}$	3	$\{D\}$
$\{D\}$	4	$\emptyset$
$\{A, B\}$	2	$\{C, D\}$
$\{A, C\}$	3	$\{D\}$
$\{A, D\}$	4	$\emptyset$
$\{B, C\}$	3	$\{D\}$
$\{B, D\}$	4	$\emptyset$
$\{C, D\}$	4	$\emptyset$
$\{A, B, C\}$	3	$\{D\}$
$\{A, B, D\}$	4	$\emptyset$
$\{A, C, D\}$	4	$\emptyset$
$\{B, C, D\}$	4	$\emptyset$
$\{A, B, C, D\}$	4	$\emptyset$

$\lozenge$

Un árbol de enumeración de conjuntos se define de la siguiente forma, [66].

Definición 2.6.9 (SETree)

Sea A un atributo de $\mathcal{A}$*. Un SETree se define recursivamente como:*

1. *Su raíz es el nodo etiquetado por el conjunto vacío.*

[21] Con ello se logra un orden entre los elementos del conjunto de atributos.

2. *Recursivamente, sea S una etiqueta de nodo. Entonces sus hijos se etiquetan de la siguiente forma:*

$$\{S \cup A \mid A \in Vista(S)\}$$ ♦

Ejemplo 2.6.4

Utilizando el ejemplo anterior (ej. 2.6.3) podemos determinar el conjunto de nodos del SETree resultante. En la figura 2.7 podemos visualizar el SETree.

S	$Vista(S)$	*Nodos Hijos*
$\{\emptyset\}$	$\{A, B, C, D\}$	$\{\{A\}, \{B\}, \{C\}, \{D\}\}$
$\{A\}$	$\{B, C, D\}$	$\{\{A, B\}, \{A, C\}, \{A, D\}\}$
$\{B\}$	$\{C, D\}$	$\{\{B, C\}, \{B, D\}\}$
$\{C\}$	$\{D\}$	$\{\{C, D\}\}$
$\{D\}$	$\emptyset$	$-$
$\{A, B\}$	$\{C, D\}$	$\{\{A, B, C\}, \{A, B, D\}\}$
$\{A, C\}$	$\{D\}$	$\{\{A, C, D\}\}$
$\{A, D\}$	$\emptyset$	$-$
$\{B, C\}$	$\{D\}$	$\{\{B, C, D\}\}$
$\{B, D\}$	$\emptyset$	$-$
$\{C, D\}$	$\emptyset$	$-$
$\{A, B, C\}$	$\{D\}$	$\{\{A, B, C, D\}\}$
$\{A, B, D\}$	$\emptyset$	$-$
$\{A, C, D\}$	$\emptyset$	$-$
$\{B, C, D\}$	$\emptyset$	$-$
$\{A, B, C, D\}$	$\emptyset$	$-$

◊

La ventaja en la utilización de esta estructura es que la misma permite realizar podas del espacio de búsqueda utilizando alguna heurística según el dominio.

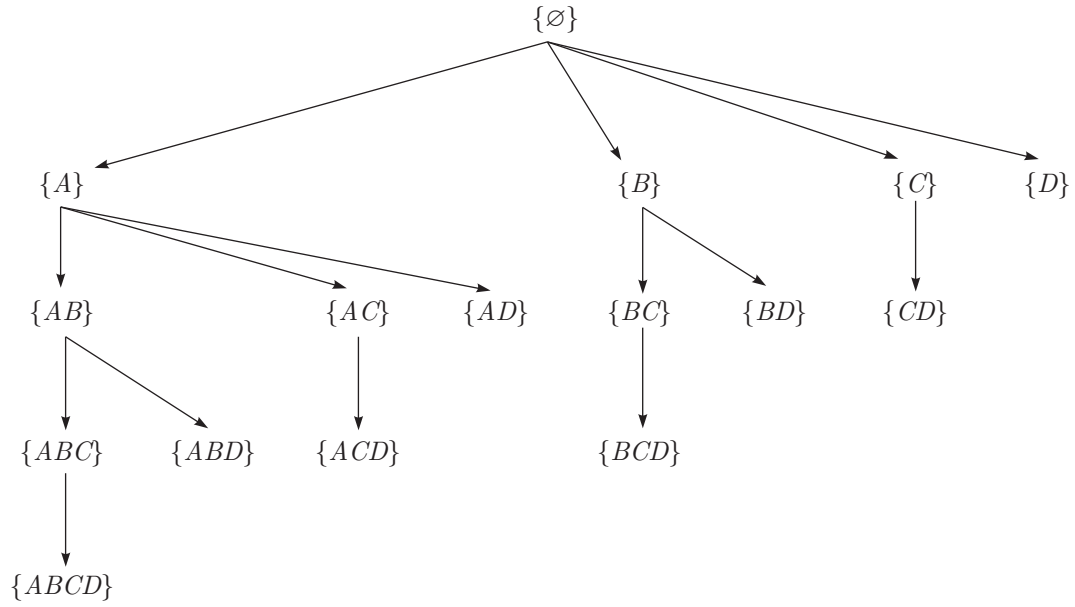


Figura 2.7: Árbol de Enumeración de Conjuntos con cuatro atributos

Capítulo 3

Representación Concisa de Patrones Frecuentes

Los algoritmos inspirados en `Apriori` tienen una buen desempeño sobre conjuntos de datos ralos[1] donde los patrones frecuentes son cortos. Esto se debe a que realizan una exploración *primero a lo ancho* (Breadth-First) del espacio de búsqueda y si los patrones frecuentes son largos puede llegar a necesitar recorrer todo el espacio de búsqueda. Existen muchas bases de datos que poseen estas características (como por ejemplo bases de datos con información biológica de los campos de análisis de ADN y proteínas, conjuntos de datos para problemas de clasificación [6], conjuntos de datos de censos y telecomunicaciones [85]). Debido a esta exploración primero a lo ancho los patrones se van expandiendo de a un item por vez. En consecuencia, si suponemos que el I es el itemset frecuente más largo entonces la cantidad de itemsets candidatos (efectivamente evaluados) es $2^{|I|}$. Por lo tanto si este itemset I es lo suficientemente largo se podría llegar a necesitar recorrer todo el espacio de búsqueda como sucede en varias bases de datos densas.

Por otro lado existen algoritmos más eficientes en la búsqueda de patrones frecuentes como FP-Growth [38] el cual utiliza una representación comprimida de dataset tal como se observa en la sección 2.4.3 o también los algoritmos que utilizan la representación de datos con Diffset [85]. Estas estructuras han demostrado ser muy eficientes en la búsqueda de patrones largos o en el manejo de bases de datos densas. Sin embargo, existe un problema que no puede ser evitado por ningún algoritmo que como resultado realice la

[1]Para más información sobre este tipo de base de datos consultar la sección 2.4 en la página 21.

enumeración de todos los patrones frecuentes. Si la base de datos es densa la enumeración de los patrones frecuentes es tan inviable como el proceso realizado por los algoritmos inspirados en `Apriori`. Por tal motivo, se han propuesto diversas alternativas para evitar esta enumeración de itemsets frecuentes. En este capítulo analizamos los acercamientos más importantes para evitar esta enumeración. En primer lugar analizaremos los denominados *patrones maximales* los cuales permiten expresar en forma concisa todo el espacio de patrones frecuentes con cierta perdida de información. En segundo lugar analizaremos la utilización de *patrones cerrados* que mejoran a los patrones maximales en el hecho que no se pierde información, pese a que en tal caso la cardinalidad de este conjunto puede llegar a ser mucho mayor (aunque siempre menor o igual a la cantidad de patrones frecuentes). Por último analizaremos acercamientos alternativos los cuales pueden ser combinados con acercamientos anteriores para mejorar su eficiencia o simplemente ser utilizados sobre el espacio de patrones frecuentes.

3.1. Definiciones

A continuación se proponen algunas definiciones sobre las clases de patrones especiales analizados en este capítulo. Podrá apreciarse que la definición de itemset frecuente maximal es un caso particular de la definición 2.1.11 (página 14).

Definición 3.1.1 (Itemset Frecuente Maximal)
Un itemset frecuente es denominado maximal *si ningún superconjunto propio*[2] *es frecuente. El conjunto de itemsets frecuentes maximales (Maximal Frequent Itemsets) es notado como* MFI. ♦

Definición 3.1.2 (Itemset Frecuente Cerrado)
Un itemset frecuente es denominado cerrado *si ningún superconjunto propio tiene el mismo soporte. El conjunto de itemsets frecuentes cerrados (Frequent Closed Itemsets) es notado como* FCI. ♦

[2]Ver definición **??** en la página **??**.

Estas clases especiales de patrones permiten reducir significativamente la cantidad de información que se muestra al usuario y además permiten que los algoritmos puedan sacar provecho de sus propiedades para reducir el espacio de búsqueda o al menos reducir el espacio y tiempo necesario para completar la tarea. En [90] se muestra la relación entre estos conjuntos de patrones con el conjunto de patrones frecuentes notado FI.

Propiedad 3.1.1

Los itemset frecuentes cerrados (FCI) son una clase especial de (FI). Todo itemset frecuente maximal (MFI) es un itemset frecuente cerrado ya que ningún superconjunto propio tiene el mismo soporte. Por lo tanto, se cumple la siguiente relación:

$$MFI \subseteq FCI \subseteq FI$$ □

Como ha sido previamente mencionado la búsqueda de FI en ciertos dominios es una tarea inviable. Pareciera que a la luz de la propiedad anterior las clases especiales de itemsets no proporcionan ninguna ayuda ya que en el peor caso los tres conjuntos son iguales ($MFI = FCI = FI$). Sin embargo, aún cuando este caso es posible es muy improbable y en la gran mayoría de los casos tanto los FCI como los MFI son enormemente más pequeños. Aún puede suceder suceder que el conjunto de FCI sea exponencialmente grande como se menciona en [90]; en tal caso el conjunto de MFI es la mejor opción.

Analicemos intuitivamente el "peor" caso, cuando los tres conjuntos de itemsets son iguales. Si el conjunto de MFI es igual al conjunto de FI, quiere decir que ninguno de los itemsets del conjunto de MFI tiene un subconjunto propio que sea frecuente (a excepción del conjunto vacío) ya que si este fuera el caso tal conjunto estaría en FI y la hipótesis de igualdad sería falsa. Si consideramos la propiedad 2.5.1 de antimonotonía, sabemos que todo subconjunto propio de cada itemset perteneciente a MFI es frecuente. Entonces la única alternativa es que todos los itemsets que aparecen en MFI sean itemsets de longitud 1 y por lo tanto el conjunto de MFI es del orden de la cantidad de items $\mathcal{I}$. Entonces, a partir de este análisis podemos deducir que si la cantidad de elementos de MFI es aproximadamente igual a la cantidad de elementos de FI, ambos conjuntos contienen

pocos itemsets frecuentes y la longitud de los itemsets presentes es corta. Por otro lado, si existen muchos FI el crecimiento de los mismos es exponencial ($2^{|\mathcal{I}|}$) y los MFI es la forma más concisa de representar todo el conjunto de FI.

3.2. Patrones Maximales

De las técnicas que permiten representar de forma concisa el conjunto de patrones frecuentes, la búsqueda de patrones frecuentes maximales es la más importante. Esto se debe a que el conjunto de MFI es la forma más concisa, hasta el momento conocida, de representar *todo* el espacio de FI. Como hemos visto en las secciones anteriores, el conjunto de FI es clausurado en subconjuntos según el soporte (Definición 2.1.2 y Propiedad 2.5.1) y por lo tanto es representable por sus itemsets más específicos (Definición 2.1.10) es decir aquellos itemsets maximales. En la figura 3.1 puede verse gráficamente como utilizando el reticulado de la sección 2.6.1, los patrones maximales frecuentes ($\{ABD\}$ y $\{BC\}$ con un círculo más obscuro) cubren a todos los patrones frecuentes (círculo claro). Por lo tanto, a partir de los itemsets frecuentes maximales se pueden obtener todos los itemsets frecuentes. Más formalmente:

Propiedad 3.2.1 (Bayardo, 1998)

Todo itemset frecuente es un subconjunto de un itemset maximal frecuente. □

Por lo tanto, la búsqueda de patrones frecuentes puede simplificarse a buscar patrones frecuentes maximales. Podemos ver que en el caso de datasets densos simplemente bastaría con realizar una búsqueda de los MFI y mostrar los mismos al usuario. De esta forma, evitamos tener que enumerar una cantidad exponencial[3] de itemsets en memoria o al mostrárselos al usuario. A su vez, en el caso de que fuera necesario y viable la enumeración de todos los patrones frecuentes, esto se podría realizar realizando una nueva pasada sobre la base de datos obteniendo el soporte para todos aquellos subconjuntos de los MFI tal como se menciona en [87].

[3]Con respecto a la cantidad de items posibles.

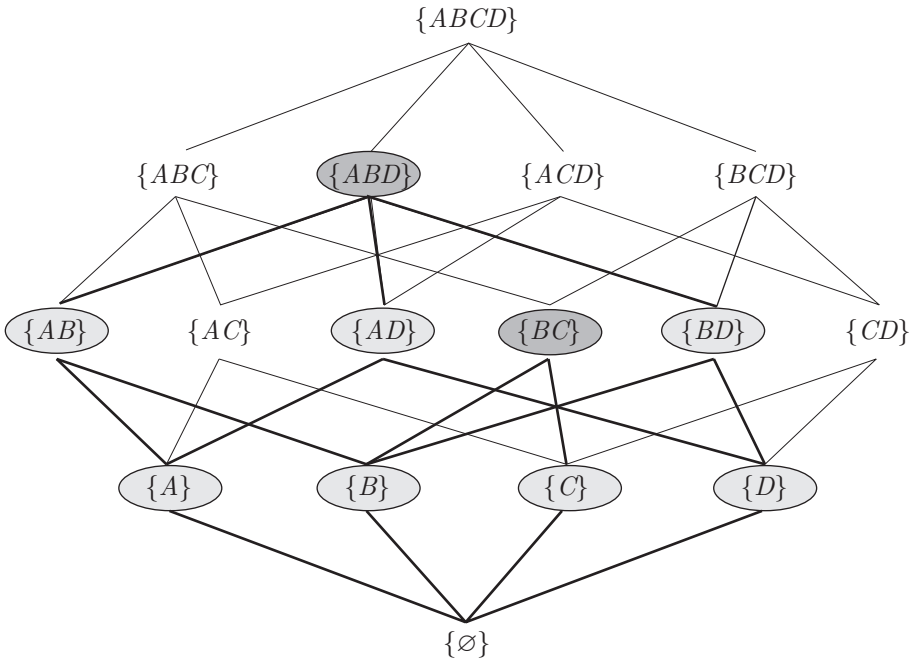


Figura 3.1: Reticulado Booleano de $\mathcal{P}(\{A, B, C, D\})$ con $MFI = \{\{A, B, D\}, \{B, C\}\}$

Existe otra propiedad de los MFI denominada *no monotónica*, reportada en [50], la cual no es respetada por el conjunto de patrones frecuentes.

Propiedad 3.2.2 (Monotonía de los Itemsets Frecuentes)
Para una base de datos dada, tanto el número de candidatos como el número de itemsets frecuentes se incrementa a medida que el umbral de mínimo soporte se decrementa. □

Sin embargo los patrones frecuentes maximales se comportan de una forma no monotónica, esto quiere decir que la cantidad de patrones maximales puede verse incrementada a medida que disminuye el umbral de mínimo soporte como también puede suceder que disminuya la cantidad de patrones maximales. Por ejemplo, cuando se utiliza un soporte mínimo de 9, el conjunto de patrones frecuentes maximales puede ser $\{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. Sin embargo, si se reduce el umbral a 6, el MFI puede ser $\{\{1, 2, 3\}\}$, como también si se incrementa a 12 el conjunto de MFI puede ser $\{\{1\}, \{2, 3\}\}$.

Esta característica de los patrones maximales puede ser muy útil en el proceso de búsqueda de MFI. Sin embargo, si no se toman las precauciones necesarias esta ventaja

puede pasar desapercibida, por ejemplo para el caso de los algoritmos que realizan una búsqueda de abajo hacia arriba (bottom-up) y primero a lo ancho (breadth-first) ya que tendrán que seguir analizando el conjunto de todos los itemsets candidatos.[4]

Con respecto a esta propiedad, en [6] se reporta que se realizaron pruebas empíricas sobre los conjuntos estándar de datos (ver sección 2.4.4) y como resultado se obtuvo que disminuyendo el umbral de soporte mínimo la cantidad de patrones maximales encontrados se incrementaba. Estos resultados se contraponen a la propiedad anteriormente mencionada de *no monotonía*. Sin embargo, en dicho trabajo no se proveen datos suficientes sobre dichas pruebas y los resultados teóricos mencionan que bajo un cierto umbral la cantidad de MFI debe disminuir hasta llegar al conjunto universal según la representación de reticulados. Por lo tanto, creemos que estas pruebas no han sido lo suficientemente completas ya que como se menciona en dicho trabajo, al reducir el umbral el tiempo requerido para buscar MFI crecía exponencialmente y el tamaño del dataset junto con este tiempo limitaban la capacidad de disminuir el umbral a los valores necesarios para observar esta disminución en dichas bases de datos.

Como se verá en la sección 5.5 esta propiedad puede ser explotada aún en otros dominios. Cabe hacer referencia que al momento de su aplicación como sucede en dicha sección no teníamos conocimiento de la existencia de este trabajo enunciando esta propiedad.

A continuación veremos las técnicas más frecuentemente utilizadas para la búsqueda de MFI y analizaremos los algoritmos más importantes. Sin embargo, hemos dejado fuera de nuestro análisis algunos de los primeros acercamientos como `MaxEclat` y `MaxClique`, [87], ya que los mismos están orientados para plataformas de minería distribuida o paralela.

3.2.1. Técnicas Utilizadas

Resumiremos a continuación las técnicas utilizadas por los algoritmos que representan el estado actual de arte en la minería de patrones frecuentes maximales. Estas técnicas no son utilizadas por todos los métodos propuestos, ya que cada una de ellas ha sido motivo de una mejora a las técnicas anteriormente utilizadas. Por lo tanto, cuanto más reciente

[4]Ver definición 2.5.1 página 37.

sea el algoritmo, más probable es que utilice muchas de estas técnicas.

SETree Etiquetado

La representación más común del espacio de búsqueda utilizada en la búsqueda de patrones maximales frecuentes son los árboles de enumeración de conjuntos (*SETree*). Estos árboles facilitan la tarea de poda del espacio de búsqueda. No obstante, se utilizará una versión etiquetada de los *SETree* donde cada nodo poseerá más información que la provista inicialmente.

Definición 3.2.1 (Grupo Candidato, Bayardo, 1998)
Un grupo candidato (Candidate Group) g consiste de dos itemsets. El primero llamado cabeza (Head) y notado $h(g)$ y el segundo itemset llamado cola (Tail) y notado $t(g)$. Notaremos al grupo g como $h(g)\,\{t(g)\}$. ⧫

Todo nodo en un *SETree* para búsqueda de patrones será representado por un grupo candidato donde su cabeza representa el itemset enumerado por el nodo y su cola es un conjunto ordenado que contiene todos los items que no están en la cabeza y que potencialmente pudieran aparecer en cualquier subnodo.

Ejemplo 3.2.1
Veamos nuevamente la figura 2.7. Aquí cada nodo del árbol representa un itemset. Si observamos el nodo $\{A\}$ dicho itemset tiene $h(g) = \{A\}$ y $t(g) = \{BCD\}$, el nodo $\{\emptyset\}$ tiene $h(g) = \{\emptyset\}$ y $t(g) = \{ABCD\}$ y el nodo $\{CD\}$ tiene $h(g) = \{CD\}$ y $t(g) = \{\emptyset\}$. En la figura 3.2 se puede observar el árbol de enumeración de conjuntos completamente etiquetado. ◊

Con esta información adicional en cada nodo se podrían conocer todos los posibles subnodos que pertenecen al espacio de búsqueda de ese nodo.

Definición 3.2.2 (Zou et al.)
Para un nodo $N = X : Y$ donde X es la cabeza de un grupo candidato g e Y su cola, el conjunto de todos los itemsets obtenidos por la concatenación de X con los itemsets de

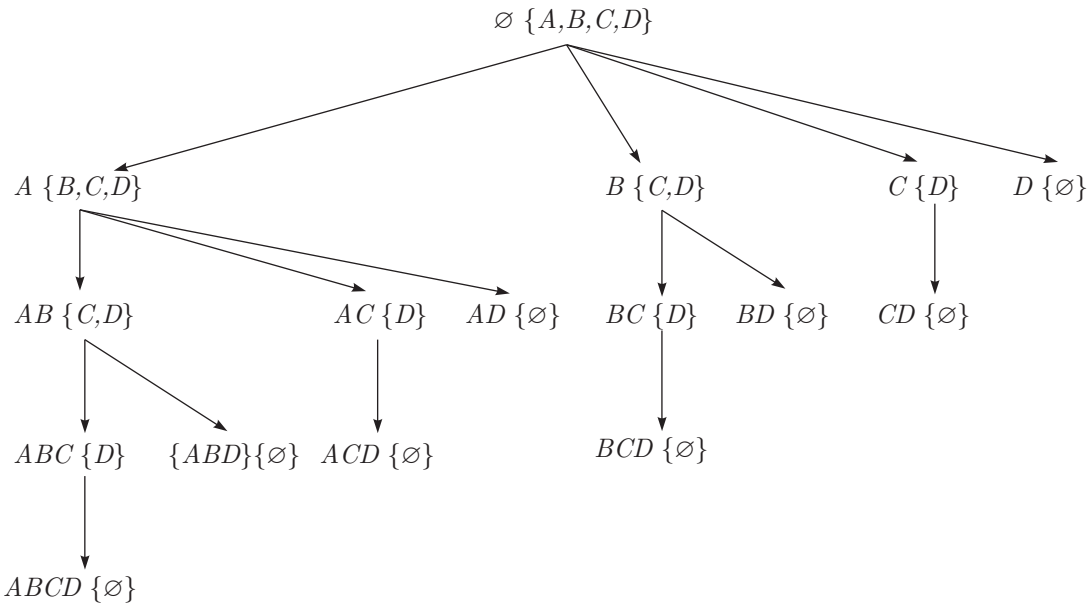


Figura 3.2: SETree Etiquetado

$\mathcal{P}(Y)$ *es llamado* espacio de búsqueda del nodo N *y notado como* $\{h(g) : t(g)\}$. *Esto es*

$$\{X : Y\} = \{X \cup V \mid V \in \mathcal{P}(Y)\} \qquad \blacklozenge$$

Ejemplo 3.2.2

El espacio de búsqueda del nodo $\{AB : CD\}$ *incluye cuatro itemsets* $\{AB\}$, $\{ABC\}$, $\{ABD\}$, $\{ABCD\}$. *Ya que* $Y = \{CD\}$ *entonces el conjunto de partes de* Y *es* $\mathcal{P}(Y) = \{\emptyset, C, D, CD\}$. $\lozenge$

En [90] puede encontrarse información más detallada sobre el proceso de etiquetado de *SETree*. Allí se mencionan varios teoremas que formalizan las nociones de particionamiento y podas sobre espacios de búsqueda en *SETree Etiquetados*.

Estrategias de Búsqueda

Hemos visto en la sección previa que los itemsets pueden ser ubicados de forma ordenada en al menos dos representaciones grafos (utilizando *Reticulados*) y árboles (utilizando *SETree* o *FPTree*). Dentro de estas representaciones se encuentran todos los itemsets de

los cuales estamos interesados en obtener *a*) Itemsets Frecuentes (FI), *b*) Itemsets Cerrados Frecuentes (FCI) y *c*) Itemsets Maximales (MFI) entre otros. Podemos entonces definir el problema de encontrar todos los itemsets interesantes como un problema de búsqueda donde:

- el *estado inicial* puede ser cualquier itemset, aunque usualmente es el itemset vacío o el itemset universal (el itemset compuesto por todos los items posibles).
- la función *sucesor* la cual permite incorporar un item del dominio a un itemset. Alternativamente, estará disponible la función *predecesor* la cual quitará un item al itemset.
- el *verificación de meta* consistirá en determinar si se han determinado todo el conjunto de itemset frecuentes, maximales o cerrados.

Lo anterior genera un espacio de búsqueda en el que las funciones sucesor y predecesor pueden ser utilizadas para recorrer el mismo. De acuerdo a como son utilizadas estas funciones se puede establecer distintas estrategias de búsqueda. En [65] se provee información detallada sobre las distintas formas de recorrer estos espacios. En general, existen dos alternativas:

1. *Búsqueda Ciega*: En este tipo de estrategia no se utiliza ninguna información que pueda ayudar a reducir el costo de alcanzar la meta.
2. *Búsqueda Heurística*: Esta estrategia saca provecho de información conocida sobre el dominio para reducir el costo de alcanzar la meta.

Dentro de la alternativa de búsqueda ciega existen varias estrategias posibles y entre las más importantes podemos mencionar:

a) *Breadth-First Search (BFS)*: Este tipo de búsqueda inicial su tarea en el nodo más bajo del espacio de búsqueda (nodo raíz). Luego todos los nodos inmediatos al nodo raíz son expandidos por la función sucesor. En general, todos los nodos del nivel l son expandidos antes que cualquier nodo del nivel $l+1$. Usualmente este tipo de búsqueda

es útil cuando las metas se encuentran cerca del nodo inicial y el factor de ramificación es pequeño (en este caso la cantidad de items).

b) *Depth-First Search (DFS)*: En este tipo de búsqueda se expande siempre el nodo que se encuentra en el nivel más bajo. Solo cuando se llega en un nodo que no se pueda expandir se procede entonces a analizar otros nodos. Esta estrategia mejora el espacio necesario utilizado para almacenar itemsets candidatos y alcanza los itemsets más largos más rápidamente.

c) *Bidirectional Search*: Esta búsqueda es iniciada por dos puntas, usualmente el nodo inicial y un nodo meta y se expanden ambos nodos hasta encontrar el camino que conecta a ambos. Dentro del contexto de búsqueda de itemset frecuentes el nodo inicial es representado por el conjunto vacío y el nodo final es representado por el conjunto universal.

La búsqueda heurística es utilizada en el contexto de búsqueda de patrones maximales (MFI)y cerrados (FCI) para evitar generar numerosos itemsets candidatos. Estas heurísticas son implementadas usualmente como una estrategia de poda utilizando como base alguna de las alternativas de búsqueda ciega anteriormente mencionadas.

Frecuencia de Itemsets

Existen tres formas para determinar si un itemset es frecuente:

1. Realizar una pasada sobre la base de datos determinando su soporte.
2. Evaluar si dicho itemset es subconjunto de algún itemset frecuente conocido.
3. Determinar un límite inferior del soporte del itemset. Si dicho límite inferior es mayor al umbral del mínimo soporte, entonces el itemset es frecuente.

Solamente la primer forma nos provee con información exacta acerca del soporte del itemset, mientras que las otras dos formas nos dan una aproximación de su soporte que nos permite determinar si es frecuente o no. En la segunda, se utiliza la propiedad de

antimonotonía[5] para llegar la conclusión que el itemset es frecuente. En la última se utiliza información recolectada durante la recorrido del espacio de búsqueda para establecer dicha conclusión. Estas tres formas permiten reducir el espacio de búsqueda realizando podas sobre el mismo como se podrá apreciar en la sección de estrategias de poda.

A continuación analizaremos esta última forma que establece un límite inferior al soporte de un itemset. La noción de límite inferior sido propuesta en [6] y luego extendida en [15]. La aproximación de un itemset se puede realizar utilizando los soportes de los itemsets subconjuntos a dicho itemset estableciendo así un *límite inferior de soporte* (Support Lower-Bound). Consideremos la siguiente definición:

Definición 3.2.3 (Función Drop, Bayardo, 1998)
La función drop *retorna el número de transacciones que son quitadas (dropped) del conjunto de soporte de un itemset cuando éste es extendido con un item*[6]*. Sea I un itemset, i un item tal que $i \notin I$. Entonces se define:*

$$drop(I, i) = soporte(I) - soporte(I \cup \{i\})$$ ♦

Esta función será utilizada para establecer el límite inferior de soporte para un itemset dado. Sin embargo, para poder realizar las demostraciones formales correspondientes utilizaremos también la siguiente definición estableciendo además la relación entre ambas definiciones.

Definición 3.2.4 (Función tid-drop)
La función tid-drop *o* tdrop *retorna el conjunto de transacciones que son quitadas (dropped) del conjunto de soporte de un itemset cuando éste es extendido con un item. Para un itemset I dado, sea i un item tal que $i \notin I$. Se define entonces* tdrop *como:*

$$tdrop(I, i) = cubre(I) - cubre(I \cup \{i\})$$
$$drop(I, i) = |tdrop(I, i)|$$ ♦

[5]Ver propiedad 2.5.1 en la página 37.
[6]Utilizaremos la notación de funciones sobre itemsets con la base de datos implícita.

Es importante observar en este caso que $cubre(I \cup \{i\}) \subseteq cubre(I)$ por la propiedad 2.1.1. Utilizando la definición 3.2.3 puede probarse la siguiente propiedad operando algebraicamente.

Propiedad 3.2.3

Sea I un itemset, i un item tal que $i \notin I$, entonces

$$soporte(I \cup \{i\}) = soporte(I) - drop(I, i). \qquad \square$$

Esta propiedad permite obtener el soporte de un itemset basado en un subconjunto inmediato y la función *drop*. Sin embargo, esta función requiere conocer el valor del soporte que queremos calcular[7]. Por lo tanto, el objetivo es aproximar esta función y así obtener el límite inferior de dicho soporte. A tal efecto presentaremos dos lemas que permitirán aproximar el valor de la función *drop* utilizando el valor de la función *drop* de un subconjunto de I permitiendo así calcular dicho soporte.

Lema 3.2.1

Sean X e Y itemsets tal $X \subset Y$, i un item tal que $i \notin Y$ (Por lo tanto $i \notin X$). Entonces:

$$cubre(X \cup \{i\}) \ \cap \ tdrop(Y, i) = \emptyset$$

Demostración:

Supongamos por contradicción que la hipótesis es falsa, es decir

$$cubre(X \cup \{i\}) \ \cap \ tdrop(Y, i) \neq \emptyset$$

Entonces podemos afirmar que existe al menos un elemento en la intersección que produce que el resultado no sea el conjunto vacío. Es decir $\exists t \in cubre(X \cup \{i\}) \cap tdrop(Y, i)$ y $t \neq \emptyset$.

$$\therefore t \in cubre(X \cup \{i\}) \Rightarrow t \ contiene \ a \ i \ \wedge$$
$$t \in tdrop(Y, i) \Rightarrow t \ no \ contiene \ a \ i$$

[7] Obsérvese que para calcular la función $drop(I, i)$ requiere conocer el valor de $cubre(I \cup \{i\})$.

Llegamos a una situación contradictoria donde la transacción t por un lado tiene que contener a i y por otro no. Esta situación provino de suponer que existía tal transacción, por lo tanto, no puede existir dicho t y la intersección es vacía. ■

Este lema puede ser verificado gráficamente observando los conjuntos representados en la figura 3.3. En figura 3.3(a) se muestran los conjuntos de itemsets X e Y y al item i. En la figura 3.3(b) son representados los conjuntos de transacciones que contienen a X, Y y al item i. Observar que en la figura (a) el item i se encuentra fuera de ambos conjuntos ya que i es un item que no pertenece a ninguno de los dos. Sin embargo, las transacciones que incluyen al item i son un subconjunto de ambos para el caso de $Y \cup \{i\}$ y $X \cup \{i\}$ aunque esto no significa que todas las transacciones que contienen al item i sean un subconjunto de las transacciones que contienen al itemset X.

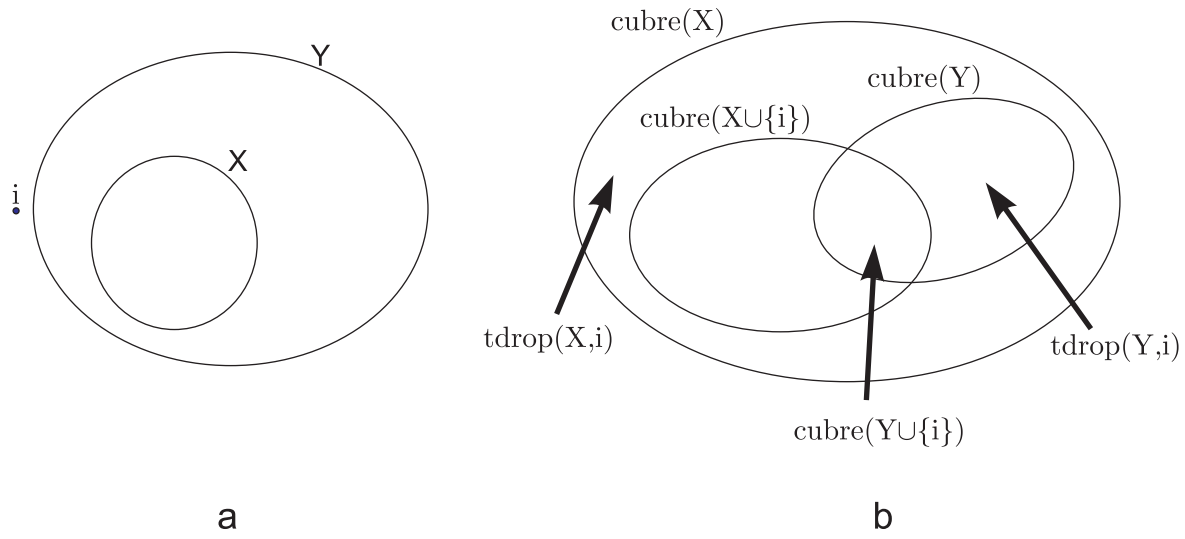


Figura 3.3: Conjuntos de Itemset y Conjuntos de Transacciones

Lema 3.2.2

Sean X e Y itemsets tal $X \subset Y$, i un item tal que $i \notin Y$. Entonces:

$$drop(X, i) \geq drop(Y, i)$$

Demostración:

$$tdrop(X,i) = cubre(X) - cubre(X \cup \{i\}) \quad [\ \textit{Por def. 3.2.4}\ ] \quad (3.1)$$

$$tdrop(Y,i) = cubre(Y) - cubre(Y \cup \{i\}) \quad [\ \textit{Por def. 3.2.4}\ ] \quad (3.2)$$

$$cubre(Y) \subseteq cubre(X) \quad [\ \textit{Por } X \subset Y \textit{ y prop. 2.1.1}\ ] \quad (3.3)$$

$$cubre(X \cup \{i\}) \cap tdrop(Y,i) = \emptyset \quad [\ \textit{Por lema 3.2.1}\ ] \quad (3.4)$$

$$tdrop(Y,i) \subseteq cubre(Y) \subseteq cubre(X) \quad [\ \textit{Por (3.2) y (3.3)}\ ] \quad (3.5)$$

$$tdrop(Y,i) \subseteq cubre(X) - cubre(X \cup \{i\}) \quad [\ \textit{Por (3.4) y (3.5)}\ ]$$

$$tdrop(Y,i) \subseteq tdrop(X,i) \quad [\ \textit{Por (3.1)}\ ]$$

$$|tdrop(Y,i)| \leq |tdrop(X,i)|$$

$$\therefore\ drop(Y,i) \leq drop(X,i) \qquad \blacksquare$$

Finalmente podemos obtener el teorema que permite determinar el soporte a partir del soporte de los subconjuntos del itemset.

Teorema 3.2.1 (Limite Inferior del Soporte, Bayardo, 1998)
Sean X e Y itemsets tal $X \subset Y$, i un item tal que $i \notin Y$. Entonces:

$$soporte(Y \cup \{i\}) \geq soporte(Y) - drop(X,i)$$

Demostración:

$$soporte(Y \cup \{i\}) = soporte(Y) - drop(Y,i) \quad [\ \textit{Por prop. 3.2.3}\ ]$$

$$\geq soporte(Y) - drop(X,i) \quad [\ \textit{Por lema 3.2.2}\ ] \qquad \blacksquare$$

Este teorema permite obtener el soporte de cualquier k-itemset si se conoce el soporte de al menos dos $(k-1)$-itemsets que sean subconjuntos del mismo. Este teorema puede

ser generalizado para calcular el soporte de cualquier itemset conociendo el soporte de los subconjunto más pequeños disponibles.

Teorema 3.2.2 (Límite Inferior de Soporte Generalizado, Bayardo, 1998) *Sean I y T dos itemsets disjuntos y $I_s \subset I$. Entonces:*

$$soporte(I \cup T) \geq soporte(I) - \sum_{i \in T} drop(I_s, i)$$

□

Es importante mencionar que las demostraciones presentadas en esta sección son una formalización de los bosquejos realizados en [6].

Estrategias de Poda

La característica más importante de los algoritmos de minería de datos es su capacidad para realizar podas sobre el espacio de búsqueda, ya sea que éste fuera representado como un reticulado, un árbol, un grafo, etc. Sin esta poda, la búsqueda de patrones frecuentes (y cualquiera de sus variantes) se reduciría a recorrer todo el espacio de búsqueda enumerando todo itemset posible, lo cual (como fuera mencionado en la sección 2.5) es una tarea inviable para dominios reales.

Por tal motivo, las estrategias de poda son el concepto clave para la efectividad de un buen algoritmo de minería de datos. En particular, estaremos interesados en analizar las técnicas utilizadas para la búsqueda de patrones maximales frecuentes.

Superconjunto Inmediato No Frecuente Como se mencionara en la propiedad 2.5.1 de anti-monotonía, si un itemset no es frecuente en un conjunto de datos entonces ningún superconjunto de este itemset puede ser frecuente. Esta técnica es utilizada por la mayoría (sino todos) de los algoritmos de minería de datos y el primero en utilizarla fue el algoritmo `Apriori`.

Utilizando la representación *SETree etiquetado*, para un nodo dado con grupo candidato g se considerará la frecuencia de cada itemset en $\{h(g) \cup \{i\} \mid i \in t(g)\}$. Si $h(g) \cup \{i\}$ resulta infrecuente entonces el espacio de búsqueda para el nodo con cabeza $h(g) \cup \{i\}$ contiene itemsets infrecuentes y puede ser podado.

Superconjunto Frecuente Esta estrategia es utilizada en *SETree etiquetados* y se basa en la siguiente observación. Para un nodo n del árbol con un grupo candidato g, el itemset más largo (itemset maximal) posible contenido en el espacio de búsqueda del nodo n es $h(g) \cup t(g)$. Si $h(g) \cup t(g)$ es frecuente, entonces no es necesario explorar ningún subnodo de n y estos pueden ser podados. En principio, esta estrategia ha sido propuesta en [6] y utilizada por la mayoría de los algoritmos más recientes.

Es importante mencionar que esta estrategia requiere realizar nuevamente un conteo para determinar la frecuencia del itemset $h(g) \cup t(g)$.

Superconjunto Maximal Frecuente Esta estrategia es similar a la estrategia anterior, consistiendo en verificar si el itemset maximal $h(g) \cup t(g)$ es frecuente. No obstante, la forma de determinar si el itemset es frecuente o no es la principal diferencia.

En esta estrategia un itemset x puede ser declarado frecuente si existe algún superconjunto de x que sea frecuente. Vemos aquí la importancia del uso de *SETree* ya que permite que superconjuntos de patrones puedan ser descubiertos y evaluados antes que los mismos patrones.

Bajo esta estrategia si un superconjunto de $h(g) \cup t(g)$ es encontrado en el conjunto de MFI parciales entonces este nodo $h(g) \cup t(g)$ puede ser podado junto con su espacio de búsqueda (subárbol).

Equivalencia con Padre Consideremos la siguiente situación: Sea un nodo $X : aY$ (es decir el grupo candidato consiste de la cabeza X y la cola aY donde X e Y son conjuntos de items, a es un item y aY representa los conjuntos de items con a como prefijo) y $soporte(X) = soporte(Xa)$. Entonces cada transacción en la que se encuentra X también se encuentra a y por consiguiente pueden ser podados todos aquellos nodos que no tengan al item a junto con X ya que solamente estaremos interesados en los itemsets maximales. Es decir, se podan todos los hermanos del nodo $Xa : Y$. Esta estrategia fue propuesta en [14] bajo el nombre de *Poda por Equivalencia con Padre* (Parent Equivalence Pruning) (PEP).

Límite Inferior de Soporte Esta estrategia se basa en utilizar información obtenida durante la travesía del espacio de búsqueda tal como fuera explicado en la sección 3.2.1 para evitar calcular el soporte de un itemset y realizar podas de subárboles en función de esta información. Esta estrategia ha sido propuesta en [6]. Allí se observa que si los subnodos de un grupo candidato g son generados mientras se recorre la cola de items en orden, un subnodo g_2 generado luego de g_1 tendrá la propiedad que $(h(g_2) \cup t(g_2)) \subset (h(g_1) \cup t(g_1))$. Esto significa que si $h(g_1) \cup t(g_1)$ es frecuente entonces todo subnodo generado luego de g_1 no será maximal y por lo tanto puede ser podado. Para la evaluación del soporte de g_1 se utiliza el teorema generalizado de límite inferior de soporte.

Reordenamiento Dinámico de Items

De las estrategias anteriormente mencionadas, se puede apreciar que la eficiencia de las dos estrategias de poda con respecto a superconjuntos (Superconjunto Frecuente y Superconjunto Maximal Frecuente) dependen de la forma del árbol, es decir de cada grupo candidato. Ambas estrategias se basan en que $h(g) \cup t(g)$ sean frecuentes, en cuyo caso realizan la poda del subárbol. Una buena heurística sería maximizar la probabilidad de que $h(g) \cup t(g)$ sean frecuentes y así reducir el espacio de búsqueda. Con tal motivo, en [6] se propone una heurística en la que se fuerza a que los items más frecuentes aparezcan en la mayor cantidad de patrones candidatos del espacio de búsqueda y los más infrecuentes que aparezcan dentro de los subárboles podados.

Para ello se propone ordenar los items según su soporte en forma incremental. Dicha heurística es conocida como *reordenamiento dinámico* (Dynamic Reordering) ya que el reordenamiento se produce para cada candidato generado a medida que se recorre el espacio de búsqueda. Si observamos nuevamente la figura 3.2 se puede ver que el item $\{D\}$ aparece en la cabeza o cola de cada nodo, a su vez el item $\{C\}$ aparece con un poco menos de frecuencia y así llegando al item $\{A\}$ el cual aparece con menor frecuencia en todo el árbol. Si se ordenan los items en orden creciente de soporte, los items más frecuentes aparecerían últimos y por consiguiente en la mayor cantidad de nodos. En el

siguiente ejemplo se muestra la ventaja del uso del reordenamiento de items junto con algunas de las estrategias de poda anteriormente mencionadas.

Ejemplo 3.2.3

Supongamos tener la siguiente base de datos de transacciones y observemos los soportes[8] *de los itemsets presentes.*

tid	*Itemsets*
1	BCD
2	ACD
3	ABD
4	AC
5	ABC
6	ACD
7	AB
8	ABC
9	ABC
10	$ABCD$

$soporte(A) = 9$; $soporte(ABC) = 4$;
$soporte(B) = 7$; $soporte(ABD) = 2$;
$soporte(C) = 8$; $soporte(ACD) = 3$;
$soporte(D) = 5$; $soporte(BCD) = 2$;
$soporte(AB) = 6$; $soporte(ABCD) = 1$.
$soporte(AC) = 7$;
$soporte(AD) = 4$;
$soporte(BC) = 5$;
$soporte(BD) = 3$;
$soporte(CD) = 4$;

En el primer nivel, para el grupo candidato $g = (\emptyset : \{A, B, C, D\})$ *ordenamos* $t(g)$ *en forma creciente por soporte* $g' = (\emptyset : \{D, B, C, A\})$. *Se generan cuatro grupos candidatos* $g_1 = (D : \{B, C, A\})$, $g_2 = (B : \{C, A\})$, $g_3 = (C : \{A\})$ *y* $g_4 = (A : \{\emptyset\})$. *Para cada uno de estos grupos candidatos se realiza el ordenamiento ahora considerando los soportes de los* 2*-itemsets, es decir, para* g_1 *se ordena* $t(g_1)$ *en función del soporte de los itemsets* $\{DB\}$, $\{DC\}$ *y* $\{DA\}$.

Si utilizamos un umbral de mínimo soporte de 4, *podemos ver en la figura 3.4 los itemsets que son podados. En dicha figura se utiliza una representación del espacio de búsqueda utilizando un* SETree *sin realizar reordenamiento dinámico. Todos aquellos items por debajo de la línea transversal son podados y los itemsets con un círculo son los itemsets maximales. Es necesario mencionar que aún cuando los itemsets han sido podados éstos han sido itemsets candidatos y por lo tanto han sido visitados.*

En la figura 3.5 se aplica el reordenamiento dinámico para cada grupo candidato. Se puede ver que la cantidad de itemsets candidatos (visitados) es menor con esta distribución del árbol de búsqueda ya que los itemsets $\{DBC\}$, $\{DBCA\}$ *y* $\{DBA\}$ *nunca son*

[8]Estos soportes son obtenidos por los algoritmos a medida que se recorre el espacio de búsqueda.

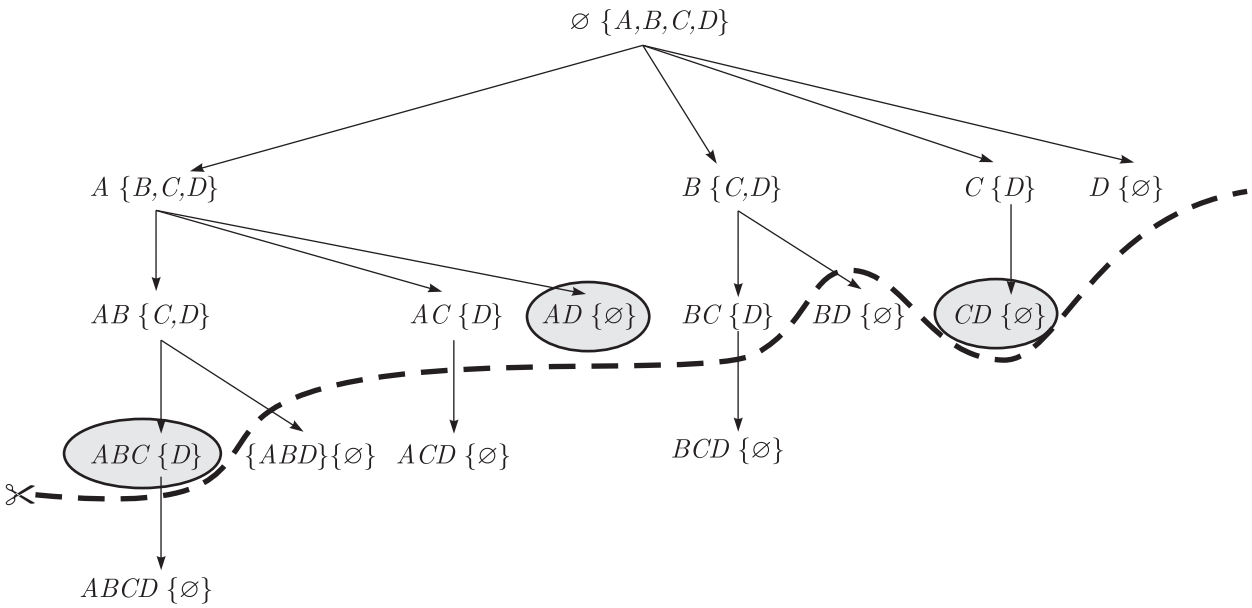


Figura 3.4: SETree con ordenamiento lexicográfico

visitados. Por otro lado, se realizan dos podas:

1. *Al evaluar el grupo candidato* $B : \{C, A\}$ *se observa que el soporte de* $h(g) \cup t(g) = \{B\} \cup \{C, A\}$ *es frecuente y por lo tanto todo el subárbol es podado. Como el itemset* $h(g) \cup t(g) = \{B, C, A\}$ *no es subconjunto de ningún* MFI *encontrado entonces este es un nuevo* MFI*. Aquí se utiliza la poda por superconjunto frecuente.*

2. *Al considerar el grupo candidato* $C : \{A\}$ *se observa que* $h(g) \cup t(g) = \{C\} \cup \{A\} = \{C, A\}$ *es un subconjunto de un* MFI *(*$\{C, A\} \subset \{B, C, A\}$*) entonces todo el subárbol es podado.* $\Diamond$

Focalización Progresiva

Uno de los problemas principales al momento de realizar las podas por superconjunto es la necesidad de realizar testeos entre conjuntos. En particular, los mejores algoritmos para testear en forma dinámica si un conjunto es un subconjunto es $O(\sqrt{|MFI|}log|MFI|)$, [31], por lo cual esta tarea no es algo trivial.

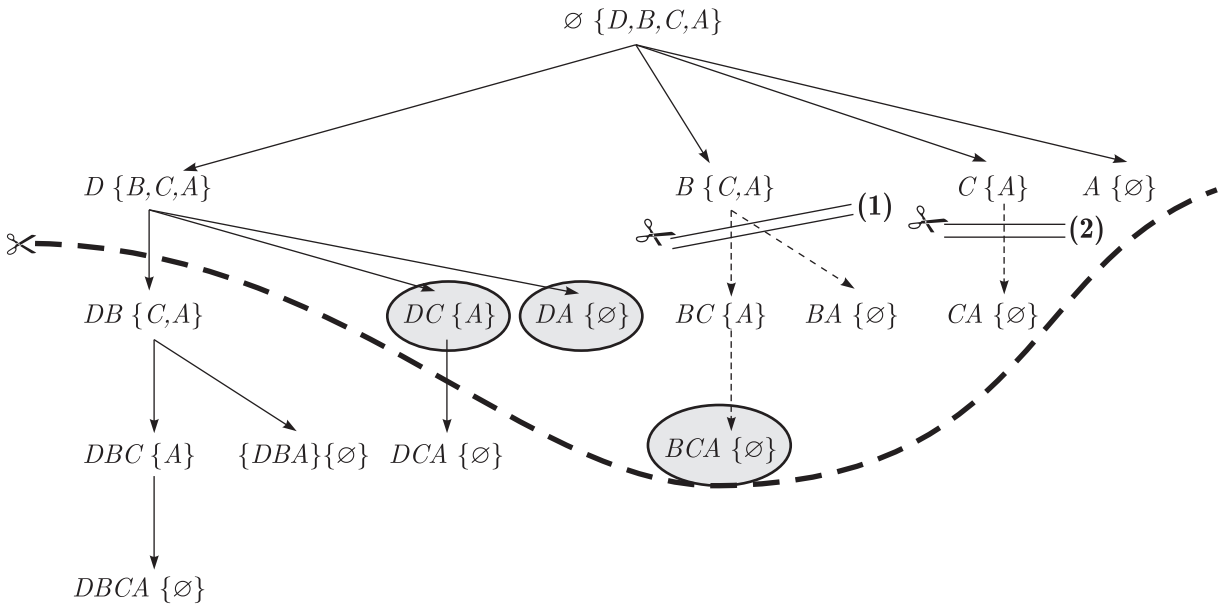


Figura 3.5: SETree con reordenamiento dinámico y poda

La estrategia de *focalización progresiva* (progressive focusing), propuesta en [31], se basa en sacar provecho de la forma en que se recorre el *SETree* para reducir la cantidad de testeos de subconjuntos necesarios. Se realiza entonces la siguiente observación, para el espacio de búsqueda de un nodo $n = h(g) : t(g)$ con grupo candidato g, solamente aquellos itemsets maximales que contengan $h(g)$ pueden ser superconjuntos de los nodos del espacio de búsqueda n. De esta forma, para cada nodo se puede reducir el número de patrones maximales a ser testeados por subconjuntos, ya que los nodos dentro del espacio de búsqueda de n recibirán una lista de *itemsets frecuentes maximales locales* (Local Maximal Frequent Itemset) ($LMFI$) en lugar de la lista completa de MFI.

Cabe mencionar que esta estrategia es válida únicamente para aquellos algoritmos que recorren primero en profundidad (Depth-First) el espacio de búsqueda (o bien algunas de sus variantes) ya que el conjunto original de MFI se puede recuperar al realizar el backtracking. Por el contrario en búsquedas de forma breadth-first se debería llevar un registro de todos los $LMFI$ lo cual no es viable.

3.2.2. MaxMiner

Este algoritmo, propuesto en [6], fue el primer algoritmo exitoso en la práctica para la búsqueda de patrones maximales frecuentes. Fue uno de los primeros algoritmos en introducir estrategias de poda diferentes a la tradicional propiedad de anti-monotonía y el primero en su clase en utilizar la representación de *SETree*. A pesar de ser uno de los primeros algoritmos todavía es considerado entre los más eficientes. Su mecanismo de búsqueda se basa en el algoritmo `Apriori` (Ver sección 2.5.1 página 38) realizando la expansión de todos los itemset frecuentes del nivel i antes de realizar la expansión de los itemsets candidatos del nivel $i + 1$. Al igual que la mayoría de estos algoritmos, `MaxMiner` utiliza un formato de base de datos horizontal. Sin embargo, este algoritmo difiere de otros algoritmos basados en `Apriori` en la utilización de la estrategia de poda por superconjunto maximal frecuente, la cual permite buscar rápidamente por adelantado los itemset frecuentes más largos y por lo tanto abandonando la travesía del espacio de búsqueda en forma bottom-up estricta.

Otra estrategia utilizada y a la vez propuesta por `MaxMiner` es la reordenación dinámica del conjunto cola $t(g)$ para cada nodo del espacio de búsqueda. Esta estrategia ayuda a optimizar las estrategias de poda reorganizando el *SETree*. En [6] se reporta que según sus pruebas empíricas sobre los repositorios de datos[9] la utilización de dicha estrategia resulta en al menos un orden de magnitud menor que bajo el ordenamiento lexicográfico standard, por lo cual cuanto menor era el soporte, mayor era la diferencia en performance entre ordenamientos.

Además de las estrategias mencionadas, `MaxMiner` propone una nueva estrategia de poda llamada poda por límite inferior de soporte. Esta estrategia permite podar subárboles sin necesidad de calcular efectivamente el soporte de un itemset a partir de la base de datos sino utilizando la información sobre soportes previamente obtenida.

En resumen, `MaxMiner` utiliza una búsqueda Breadth-First sobre un espacio de búsqueda representado por un *SETree*. Realiza una búsqueda por adelantado de patrones max-

[9] Ver sección 2.4.4 página 33.

imales permitiendo una reducción sustancial del espacio. Realizando la poda de espacio de búsqueda utilizando las estrategias de poda por superconjunto inmediato no frecuente, por superconjunto maximal frecuente y por límite inferior de soporte. Introduce y utiliza la estrategia de reordenamiento dinámico de items reduciendo aún más el espacio y obteniendo como resultado una mejor performance.

3.2.3. Pincer-Search

Este algoritmo ha sido uno de los primeros algoritmos para la búsqueda de patrones maximales frecuentes y fue propuesto en [50]. Este algoritmo utiliza como representación del espacio de búsqueda un reticulado de conjuntos utilizando para recorrer el mismo una búsqueda bidireccional. `Pincer-Search` está inspirado en la noción de espacio de versiones [53, 54] utilizando las nociones de instancias positivas y negativas de entrenamiento como los conceptos de generalización y especialización de instancias.

El algoritmo realiza una búsqueda bottom-up similar a `Apriori` a la vez que realiza una búsqueda top-down a través del reticulado. El objetivo de realizar estas búsquedas concurrentemente es utilizar la información obtenida por cada una de estas búsquedas para optimizar ambas búsquedas reduciendo la cantidad de itemsets candidatos y por lo tanto mejorando la performance.

Durante la búsqueda bottom-up, `Pincer-Search` recolecta información sobre los patrones frecuentes podando todos los patrones infrecuentes. A pesar que esta fase es similar a `Apriori`, `Pincer-Search` utiliza la información sobre los patrones infrecuentes, la cual es descartada por `Apriori`, para actualizar el conjunto de candidatos utilizada por la búsqueda top-down.

Por el otro lado, durante el proceso de búsqueda top-down se recorre el espacio de patrones infrecuentes hasta encontrar un itemset frecuente. Cuando esto sucede, este itemset es guardado como un MFI y son actualizadas ambas búsquedas. En particular, para la búsqueda bottom-up se eliminan todos aquellos subconjuntos de itemset maximal encontrado.

Estas podas realizadas sobre los conjuntos candidatos de ambas búsquedas constituyen

simplemente la aplicación de la propiedad 2.5.1 de anti-monotonía.

Este algoritmo lleva en todo momento una lista de patrones frecuentes utilizada en la búsqueda bottom-up y una lista de patrones maximales infrecuentes (llamada *Conjunto de Candidatos Frecuentes Maximales* (Maximal Frequent Candidate Set) ($MFCS$) en la cual los patrones maximales frecuentes están incluidos para la búsqueda top-down. La ventaja de este método es que cuando los MFI son muy largos o muy cortos, éstos se encuentran eficientemente. Sin embargo, si los patrones maximales se encontrarán ubicados sobre la mitad del reticulado, podemos observar que el mismo se podría comportar peor que una simple búsqueda `Apriori`.

Veamos como se comporta este algoritmo sobre una pasada normal k. En esta pasada los patrones de tamaño k son clasificados en frecuentes e infrecuentes en la búsqueda bottom-up. Si algún elemento X en $MFCS$ (de la búsqueda top-down) es clasificado como frecuente entonces todos sus subconjuntos deben ser frecuentes. Entonces, todos los subconjuntos de X de cardinalidad k pueden ser eliminados del conjunto de candidatos considerados por la búsqueda bottom-up. Esta eliminación realiza una poda importante de itemsets frecuentes no maximales. Cuanto mayor sea la diferencia entre la cardinalidad del itemset X frecuente y el valor k, mayor será la cantidad de itemsets podados y mejor será la performance del algoritmo.

En forma similar, cuando un itemset infrecuente es encontrado durante la búsqueda bottom-up, el conjunto $MFCS$ será actualizado ya que ningún subconjunto de algún itemset de $MFCS$ debe contener un itemset infrecuente. Debido a esto, la búsqueda top-down puede avanzar varios niveles por vez a diferencia de la búsqueda bottom-up la cual avanza de a 1 nivel por vez. Consideremos por ejemplo, en la primer pasada durante la búsqueda bottom-up se detectan m 1-itemsets (items) infrecuentes. Esta información es utilizada para actualizar los itemsets en $MFCS$, el cual contiene un solo elemento de cardinalidad n (donde n es la cantidad de items). En este momento, se genera un nuevo conjunto $MFCS$ con un solo elemento de cardinalidad $n-m$ que resulta de eliminar todos los items infrecuentes a itemset anterior. Por lo tanto, se avanzó m niveles en una sola pasada. Las actualizaciones sobre el conjunto $MFCS$ sin embargo no ocurren siempre

como el ejemplo mencionado, ya que se debe garantizar que todo elemento frecuente sea considerado. Por lo tanto, salvo en casos como en el ejemplo anterior, generalmente la actualización del conjunto $MFCS$ procederá de la siguiente manera:

Consideremos un itemset Y que ha sido clasificado como infrecuente y asumamos que es un subconjunto de algún itemset X en $MFCS$. Para actualizar $MFCS$, se reemplazará X por $|Y|$ itemsets, cada uno obtenido quitando un item de Y al itemset X. De esta forma ninguno de los itemsets generados contendrá a Y como subconjunto y como los subconjuntos de Y son frecuentes (por la estrategia Apriori) entonces los itemsets generados contienen únicamente subconjuntos frecuentes.

Ejemplo 3.2.4

Consideremos tener el conjunto de items $\{a, b, c, d, e, f, g, h\}$. *Inicialmente el conjunto de itemsets frecuentes es* $\{\emptyset\}$ *y el conjunto de* $MFCS = \{\{a, b, c, d, e, f, g, h\}\}$. *Supongamos que durante una pasada se detecta que los items* g *y* h *no son frecuentes. Por lo tanto, se genera un nuevo conjunto* $MFCS_1 = \{\{a, b, c, d, e, f\}\}$. *Supongamos ahora que durante la segunda pasada se detecta que los itemsets* $\{a, e\}$ *y* $\{c, d\}$ *no son frecuentes. En tal caso, se actualiza el conjunto* $MFCS_1$ *considerando primero el itemset infrecuente* $\{a, e\}$ *generando un* $MFCS_2 = \{\{b, c, d, e, f\}, \{a, b, c, d, f\}\}$ *resultante de eliminar cada item de* $\{a, e\}$ *del conjunto* $MFCS_1$. *Se realiza el mismo procedimiento para* $\{c, d\}$ *sobre* $MFCS_2$ *generando un nuevo* $MFCS_3 = \{\{b, c, e, f\}, \{b, d, e, f\}, \{a, b, d, f\}, \{a, b, c, f\}\}$ *descomponiendo cada uno de los itemsets de* $MFCS_2$ *que contienen el itemset infrecuentes* $\{c, d\}$. *Puede suceder que sea necesario eliminar de* $MFCS$ *itemsets que no sean maximales, es decir que luego de la descomposición queden en* $MFCS$ *itemsets repetidos o algún itemset subconjunto de otro.* $\Diamond$

En resumen, `Pincer-Search` utiliza un reticulado de conjuntos para representar el espacio de búsqueda y lo recorre en forma bidireccional. Al igual que el algoritmo `Apriori`, `Pincer-Search` utiliza una representación horizontal de la base de datos. Realiza una poda de patrones infrecuentes y en forma complementaria utilizando la misma propiedad, realiza la poda subconjuntos de patrones frecuentes. Este algoritmo es muy eficiente cuando los

itemsets maximales frecuentes son cortos o muy largos, sin embargo su performance se ve afectada cuando los patrones maximales se ubican en la mitad del reticulado.

3.2.4. Depth-Project

En los algoritmos que utilizan la estrategia Apriori, como `MaxMiner` y `Pincer-Search`, la generación de los patrones de longitud $k+1$ siempre es precedida por la generación de todos los patrones candidatos de longitud k (salvo cuando se generan patrones por adelantado para realizar podas). `Depth-Project` pertenece a una serie de algoritmos que divergen a este esquema ya que recorren el espacio de búsqueda en forma Depth-First. Este algoritmo ha sido propuesto en [1] y es uno de los primeros en utilizar esta estrategia.

Este algoritmo utiliza una versión ligeramente modificada del *SETree* para representar el espacio de búsqueda, el cual es denominado *árbol lexicográfico* (Lexicographic Tree). El mismo difiere de *SETree* en que en este último cada nodo es un conjunto cuando en el árbol lexicográfico cada nodo es una cadena de caracteres representando un itemset frecuente.

Al igual que los *SETree Etiquetados*, los árboles lexicográficos utilizan información adicional sobre cada nodo del árbol. En particular, se mantiene información sobre las extensiones del nodo, una proyección de la base de datos sobre ese nodo y alguna información adicional secundaria.

Las extensiones de un nodo se definen como aquellos items que permiten extender un nodo frecuente de longitud k en otro nodo de longitud $k+1$ también frecuente. Por lo tanto, podemos ver este conjunto de extensiones como un subconjunto de la cola del nodo. Este conjunto es denotado como $E(P)$ para un nodo P dado. Por otro lado, sea Q un ancestro inmediato de P, el conjunto de posibles extensiones de P, notada $F(P)$, se define como el subconjunto de $E(Q)$ tal que sus elementos ocurren lexicográficamente después que P. Lo anterior puede formalizarse como sigue:

Sea δ el mínimo soporte, $\mathcal{I}$ el conjunto de items y P, Q nodos del árbol lexicográfico

$(Q \subset P \subseteq \mathcal{I})$[10].

$$E(P) = \{i \in \mathcal{P}(\mathcal{I}) \mid i >_L max(P),\ soporte(P \cup i) \geq \delta\}$$
$$F(P) = \{i \in E(Q) \mid i >_L max(P)\}$$

Ejemplo 3.2.5

Si observamos nuevamente la figura 3.4 para el nodo $P = AB$ entonces $E(P) = \{C\}$. Nótese que D no aparece ya que ABD no es frecuente. El nodo Q ancestro inmediato de P es el nodo $Q = \{A\}$, con lo cual $E(Q) = \{A, B, D\}$. Entonces $F(P) = \{C, D\}$. $\Diamond$

Por otro lado, se realizan proyecciones de la base de datos sobre los nodos del árbol lexicográfico. Estas proyecciones corresponden con las transacciones de la base de datos relevantes para el cómputo del soporte para dicho nodo. A tal efecto, para cada transacción T de la base de datos, la transacción proyectada se define como:

$$T(P) = \begin{cases} \{\emptyset\} & \text{Si } P \not\subseteq T \\ T \cap E(P) & \text{Si } P \subseteq T \end{cases}$$

El conjunto de transacciones proyectadas sobre P, notado $\mathcal{T}(P)$, corresponde con aquellas transacciones cuya proyección sea distinto del conjunto vacío.

Es interesante observar, que este conjunto de transacciones contiene toda la información necesaria para el calculo del soporte del nodo que está siendo evaluado y también para todos los nodos que aparecen en el subárbol que tiene como nodo raíz a este nodo. Además la cantidad de transacciones del conjunto es mucho menor que el conjunto original y la cantidad de items por transacción también, lo que reduce significativamente la cantidad de evaluaciones necesarias a realizar para determinar si una transacción contribuye o no al soporte y en forma general a la determinación del soporte. Sin embargo, la tarea de proyección de la base de datos no es trivial ni en tiempo ni en espacio requerido por lo tanto se debe evaluar el costo-beneficio de dicha proyección.

Hemos mencionado que el espacio se recorre en forma Depth-First, lo que permite reutilizar información de niveles previos manteniendo un mínimo de información de los

[10]El nodo Q es un ancestro del nodo P en el árbol lexicográfico.

mismos. En general, se podrá mantener información adicional en aquellos nodos desde la raíz hasta el nodo que está siendo evaluado a diferencia de las estrategias Breadth-First que requieren mantener información en todo el nivel anterior y parte del nivel actual. En forma general, para un k-itemset sobre el cual la base de datos se proyecta, una transacción T será proyectada si el k-itemset aparece en T. Si T no es proyectada, entonces no contiene al k-itemset y por lo tanto a ningún superconjunto de dicho itemset. Por lo tanto, la base de datos proyectada del nodo Q puede ser utilizada para proyectar la base de datos sobre P realizando una menor cantidad de proyecciones que si se hubiese utilizado la base de datos original.

En [1] se propone una mejora sobre este proceso de proyección. Allí se observa que para ciertos nodos no es necesario realizar la proyección de la base de datos ya que la misma difiere muy poco de la base de datos proyectada para su nodo ancestro inmediato (y probablemente también difiere muy poco de otros ancestros). Por tal motivo, si la base de datos no es proyectada para ese nodo se agrega como parte del nodo un puntero al nodo ancestro con la proyección y un vector de bits con las transacciones que son significativas para este nodo de la proyección del ancestro. En dicho trabajo se proponen algunas heurísticas para determinar cuando realizar una proyección de la base de datos sobre un nodo o no. Es necesario mencionar que es necesario almacenar cada proyección que se realiza a lo largo de toda la rama desde la raíz al nodo que está siendo evaluado. Como consecuencia, la cantidad de proyecciones afecta en tiempo y espacio la eficiencia del algoritmo.

Para representar la base de datos, `Depth-Project` utiliza una representación horizontal utilizando cadenas de bits, donde cada bit representa un item. Si este bit está activo entonces el bit aparece en la transacción. Esta representación es mejorada con la utilización de la proyección, ya que en ésta se mantienen solo aquellos items que ocurren lexicográficamente después que un determinado itemset y por lo tanto la cantidad de items en una transacción disminuye, dejando solamente el conjunto $E(P)$ de los items más significativos. Este algoritmo utiliza la estrategia de poda de superconjunto inmediato no frecuente y poda de superconjunto frecuente propuesta en [6], junto con la estrategia

de reordenamiento dinámico.

En resumen, este algoritmo representa el espacio de búsqueda con una variante de *SETree* y recorre el mismo en forma Depth-First. Esta forma de recorrido, junto con la utilización de reordenamiento dinámico permite encontrar los patrones maximales frecuentes en forma más rápida que las estrategias Breadth-First. Realiza las mismas podas propuestas por `MaxMiner`. Utiliza una representación horizontal de la base de datos, realizando proyecciones de la misma sobre los nodos para un cómputo más eficiente del soporte de cada nodo. Estas proyecciones son realizadas según una heurística de selección de proyección lo que permite optimizar el espacio ocupado por estas proyecciones y el tiempo requerido.

3.2.5. Mafia

Otro de los algoritmos de búsqueda de patrones maximales es `Mafia`, propuesto en [14]. Al igual que `MaxMiner`, `Mafia` utiliza un árbol de enumeración de conjuntos (*SETree*) para representar el espacio de búsqueda. Sin embargo, `Mafia` recorre este espacio en forma Depth-First obteniendo de esta forma más información sobre los itemsets maximales y sobre los patrones frecuentes para poder realizar podas más inteligentes.

`Mafia` utiliza cuatro estrategias de poda: *a*) Superconjunto Inmediato No Frecuente, *b*) Superconjunto Frecuente (llamada FHUT[11]), *c*) Superconjunto Maximal Frecuente (llamada FHUTMFI) y *d*) Equivalencia con Padre (PEP). De las últimas tres estrategias, en [14] se reporta que la estrategia PEP es la que mayor impacto tiene sobre el espacio podado. Es interesante notar que esta estrategia no es utilizada por ningún otro algoritmo de búsqueda de patrones maximales, aunque si es ampliamente utilizada por los algoritmos para la búsqueda de patrones cerrados.

Este algoritmo utiliza la técnica de reordenamiento dinámico de items en la cola de los nodos. Como también la estrategia de focalización progresiva sacando provecho del recorrido en profundidad. La principal característica que distingue a este algoritmo es la utilización de una representación de la base de datos en forma vertical utilizando

[11]HUT significa Head Union Tail - Cabeza unido Cola.

bitmaps[12]. Esta estructura es optimizada de acuerdo a las características de la base de datos, ya que se realiza una compresión adaptativa dependiendo si la base de datos es rala o densa.

3.2.6. GenMax

`GenMax` es considerado como uno de los mejores métodos actuales para la minería de MFI. Ha sido propuesto en [31] y utilizado como algoritmo base para la generación de los patrones emergentes en esta tesis.

Este algoritmo utiliza para representar el espacio de búsqueda un árbol de enumeración de conjuntos (*SETree*) donde cada nodo es representado por un grupo candidato[13] y para recorrer el espacio utiliza la estrategia Depth-First junto con técnicas de poda de nodos y subárboles. Más específicamente, el algoritmo inicia su proceso con un conjunto I vacío, el cual es extendido de a un item por vez. La longitud de I corresponde con la profundidad del nodo en el *SETree* en que se encuentra. La posibilidad de poda se evalúa para cada nodo por adelantado. En forma genérica, sea una solución parcial I de longitud l, $I_{l-1} = \{i_0, i_1, \ldots, i_{l-1}\}$. Entonces el *conjunto de combinaciones* C_l corresponde a los posibles valores para el próximo item i_l, es decir aquellos valores que evitaron las técnicas de poda. Si P_l el conjunto de todos los potenciales valores para i_l, entonces este conjunto corresponde con la cola del grupo candidato y $C_l \subseteq P_l$. Si $y \in P_l - C_l$, entonces los nodos en el subárbol que tiene como nodo raíz a $I_l = \{i_0, \ldots, i_{l-1}, y\}$ no será considerado por el algoritmo.

Ejemplo 3.2.6 (Zaki, 2001)

Consideremos el espacio de búsqueda que se muestra en la figura 3.6. El espacio que deba ser mantenido por ***GenMax*** *es considerablemente más pequeño que el espacio total. Supongamos la siguiente tabla como base de datos transaccional:*

[12]Ver sección 2.4.2 página 24.
[13]Ver definición 3.2.1 en página 63.

tid	*Items*
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

Tamaño Itemset	FI*(mínimo soporte* $= 3$*)*
1	A, C, D, T, W
2	$AC, AT, AW, CD, CT, CW, DW, TW$
3	$ACT, ACW, ATW, CTW,$ **CDW**
4	**ACTW**

Comencemos con $I_0 = \emptyset$ *y* $C_1 = \{A, C, D, T, W\}$*. En el nivel* 1*, cada item en* C_1 *se agrega a* I_0*. Por ejemplo,* A *es utilizado para obtener* $I_1 = \{A\}$*. El conjunto de posibilidades para* A*,* $P_1 = \{C, D, T, W\}$ *consiste de todos los items que lexicográficamente siguen a* A*. Como solo* AC, AT *y* AW *son frecuentes entonces* $C_2 = \{C, T, W\}$*. De esta forma el subárbol para el nodo* AD *ha sido podado.*

Veamos que en la figura se establece una línea transversal, la que corresponde a la separación entre patrones frecuentes e infrecuentes. Todos los patrones frecuentes se encuentran por encima de dicha línea. Los patrones que están identificados con un círculo corresponden a los patrones maximales. Podemos apreciar que bajo esta estructura de $SETree$ *ya no se aprecia tan gráficamente como estos* MFI *cubren a todos los* FI*.* ◊

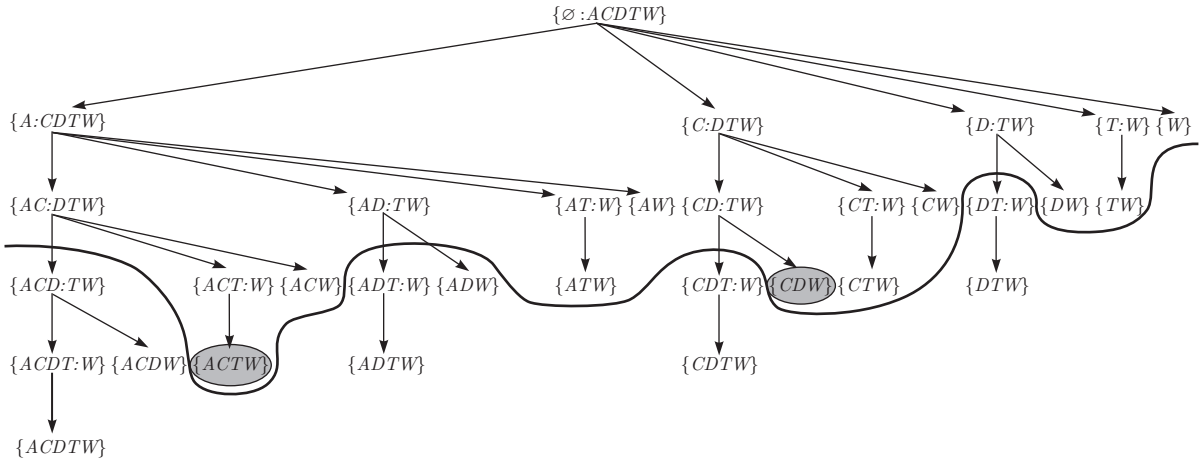


Figura 3.6: SETree con umbral de soporte mínimo de 3

El algoritmo 3 es la esencia del algoritmo `GenMax`. Se puede ver que se requieren de 1 a 2 verificaciones de superconjuntos. En la línea 4 se procede a verificar si el potencial

candidato está contenido dentro de alguno de los MFI ya encontrados, en cuyo caso todo ese subárbol es podado. Luego, en la línea 9, en el caso que I_{l+1} sea un nodo hoja (es decir sin posibles extensiones) éste puede ser un subconjunto para alguno de los MFI ya encontrados. Sin embargo, en [31] se menciona la forma en que estas dos comprobaciones pueden ser reducidas a solo una manteniendo una variable interna. Este agregado constituye una mejora importante en el caso de que la cantidad de MFI sea grande, ya que la complejidad de estas comprobaciones es en el peor caso $O(|MFI|)$[14].

Algoritmo 3 LMFI-backtrack($+I_l, +C_l, -LMFI_l, +l$): Retorna los MFI locales.

1: **Para todo** $x \in C_l$ **hacer**
2: $I_{l+1} = I \cup \{x\}$;
3: $P_{l+1} = \{y | y \in C_l, y > x\}$; {'>' se define según heurística de reordenamiento.}
4: **Si** $I_{l+1} \cup P_{l+1}$ tiene un superconjunto en $LMFI_l$ **entonces**
5: **retornar**; [El subárbol es podado]
6: **fin Si**
7: $LMFI_{l+1} = \emptyset$;
8: $C_{l+1} = \{y | y \in P_{l+1},\ I_{l+1} \cup y$ es frecuente$\}$;
9: **Si** C_{l+1} es vacío **entonces**
10: **Si** I_{l+1} no tiene superconjunto en $LMFI_l$ **entonces**
11: $LMFI_l = LMFI_l \cup I_{l+1}$;
12: **fin Si**
13: **sino**
14: $LMFI_{l+1} = \{M \in LMFI_l | x \in M\}$;
15: LMFI-backtrack($I_{l+1}, C_{l+1}, LMFI_{l+1}, l+1$);
16: **fin Si**
17: $LMFI_l = LMFI_l \cup LMFI_{l+1}$;
18: **fin Para**

Ejemplo 3.2.7

Supongamos utilizar el ejemplo 3.2.6 de la figura 3.6 sin utilizar la heurística de reordenamiento dinámico (es decir que el SETree generado es igual a la figura). Si consideramos el estado cuando $I_1 = \{A\}$, $C_1 = \{C, D, T, W\}$ *y* $x = T$ *(es decir cuando ya se consideró el caso de* $x = C$ *y* $x = D$*) entonces* $I_2 = \{A, T\}$ *y* $P_2 = \{W\}$. *Hasta el momento el conjunto* $LMFI_2 = \{\{A, C, T, W\}\}$, *por lo cual* $I_2 \cup P_2 \subset LMFI_2$ *entonces este subárbol es podado.* ◊

[14] Este límite puede mejorarse como se menciona en [31] utilizando algoritmos con tiempos amortizados.

Además de las estrategias de poda, `GenMax` reorganiza el espacio de búsqueda en forma de optimizar las podas. Para ello utiliza la estrategia de reorganización dinámica (sección 3.2.1) y propone una nueva estrategia para organizar los items la cual es combinada con la anterior. De esta forma, `GenMax` saca mejor provecho de las podas mejorando su performance. Este ordenamiento es aplicado luego de computar el conjunto de 2-itemset frecuentes y se basa en el siguiente lema:

Lema 3.2.3 (Zaki, 1998)
Sea $IF(x) = \{y | y \in \mathcal{F}_1, xy \text{ no es frecuente}\}$[15] *el conjunto que representa los 2-itemsets infrecuentes que contienen al item* $x \in \mathcal{F}_1$ *y sea* $M(x)$ *el patrón maximal más largo que contiene a* x*. Entonces* $|M(x)| \leq |\mathcal{F}_1| - |IF(x)|$. □

Se puede ver que si $|IF(x)|$ es grande entonces el conjunto de combinaciones o cola $t(g)$ es pequeño. Por lo tanto, los items se ordenan en forma decreciente de $|IF(x)|$ tal que los items como mayor probabilidad de generar itemsets infrecuentes sean generados primero podando de esa forma mayor cantidad de subárboles.

El algoritmo 4 presenta el programa principal de `GenMax`. Este algoritmo puede ser utilizado para cualquier formato de datos. Sin embargo, en [31] se presentan algunas optimizaciones para cuando `GenMax` hace uso de una base de datos vertical. Se utilizan tanto *tidsets* como *diffsets* (ver sección 2.4.2) para reducir tanto la cantidad de espacio necesario como también la cantidad de intersecciones entre conjuntos. El algoritmo inicia en el nivel inicial generando todos los *tidsets* para cada item frecuente. A partir de este nivel se evalúa la utilización de *diffsets* tal como fuera explicado en la sección 2.4.2 para lo cual en caso de ser conveniente su utilización se modifica la línea 7 del algoritmo 3 realizándose el cómputo del soporte a partir del *diffset* de y', notado $d(y')$ en lugar de computar el *tidset* de y.

En resumen, el algoritmo `GenMax` realiza una búsqueda Depth-First sobre un *SETree* que representa el espacio de búsqueda. `GenMax` propone la utilización de la estrategia de focalización progresiva (sección 3.2.1) la cual reporta una sustancial reducción en la

[15] $\mathcal{F}_1$ representa el conjunto de los 1-itemsets frecuentes.

Algoritmo 4 GenMax

1: Computar F_1 y F_2;
2: Computar $IF(x)$ para cada item $x \in F_1$;
3: Ordenar F_1; {En forma decremental por $IF(x)$ e incremental por $soporte(x)$}
4: $MFI = \emptyset$;
5: $LMFI-$backtrack$(\emptyset, F_1, MFI, 0)$;
6: **retornar** MFI;

cantidad de comprobaciones por subconjuntos. En este algoritmo también se utiliza la técnica de reordenamiento dinámico para reducir el espacio de búsqueda generado. Junto con una representación vertical de la base de datos, `GenMax` también utiliza propagación de diffsets para reducir la cantidad de operaciones necesarias para el cómputo del soporte en el caso de bases de datos densas. Por último, `GenMax` utiliza como estrategias de podas *a*) la poda por superconjunto inmediato no frecuente (eliminando todo superconjunto de un nodo no frecuente), y *b*) la poda por superconjunto maximal frecuente (evaluando si existe algún MFI que subsume dicho nodo). Es interesante mencionar que no es utilizada la estrategia de poda de superconjunto frecuente la cual ya había sido propuesta como parte del algoritmo `MaxMiner` y `Mafia` posteriormente.

3.2.7. SmartMiner

Este algoritmo, propuesto en [90], aumenta la cantidad de información que es pasada entre nodos del espacio de búsqueda, para poder mejorar la heurística de reordenamiento y producir espacios de búsqueda más pequeños. Además, este algoritmo no requiere comprobación de superconjuntos maximales, tal como es el caso de `GenMax` y `Mafia`.

`SmartMiner` utiliza *SETree Etiquetados* para representar el espacio de búsqueda. Sin embargo, esta representación es modificada para llevar más información. Además del nodo $X : Y$, `SmartMiner` agrega información sobre los patrones frecuentes ya descubiertos relevantes al nodo que está siendo evaluado. Esta información guía el proceso de búsqueda, el cual se realiza en forma Depth-First, produciendo árboles más pequeños.

Las principales diferencias con otros algoritmos que recorren el espacio en forma Depth-First son las siguientes:

1. Se posterga la creación de un nodo hasta que todos los nodos precedentes hayan sido visitados. Por otro lado, otras estrategias Depth-First crean nodos por cada item que se encuentra en la cola del nodo en orden creciente por soporte.

2. Se considera información extra para mejorar la heurística del reordenamiento dinámico. En el reordenamiento dinámico el item con menor soporte es seleccionado para ser explorado en primer lugar, ya que el espacio necesario a explorar es probablemente pequeño y pudiendo podar varias ramas del espacio de búsqueda. En `SmartMiner`, por su parte, cada nodo contiene información local y global sobre los nodos frecuentes: si un item está contenido en un itemset frecuente del longitud k entonces existen 2^k itemsets frecuentes no maximales que pueden ser podados. Esta observación es utilizada para ordenar la cola según la heurística de reordenamiento dinámico y en orden creciente del mayor valor k a partir del cual un item está contenido.

3. Se elimina el tiempo necesario para la comprobación por superconjuntos frecuentes, requerido por varias estrategias de poda.

Sin embargo, como se podrá ver se requiere tiempo extra para generar esta información la cual en cierta forma esconde el proceso de comprobación por superconjuntos. Con respecto a la heurística de reordenamiento aumentada, ésta exhibe varias similitudes con la heurística utilizada por `GenMax`, aunque este algoritmo requiere la comprobación por superconjuntos.

La información necesaria para cada nodo se define como:

Definición 3.2.5 (Información de Cola, Zou et al.)
Sea M un conjunto de itemsets frecuentes conocidos y $N = X : Y$ un nodo. La información de cola (Tail Information) de M a N, notada $TInf(N/M)$, se define como:

$$TInf(N/M) = \{Y \cap Z \mid \forall Z \in M,\ X \subseteq Z\}$$ ♦

Ejemplo 3.2.8

Supongamos que el conjunto de patrones frecuentes conocidos es $\{ABCD, ABE, ACE\}$ *y el nodo que está siendo evaluado es* $E : \{BCD\}$*. Entonces:*

$$TInf(E : \{BCD\} \,/\, \{ABCD, ABE, ACE\}) = \{B, C\}$$

Lo anterior significa que $\{EB\}$ *y* $\{EC\}$ *son frecuentes. Vemos entonces como se realiza una clase de evaluación de superconjunto aunque a nivel local. Esta información será utilizada tanto para la poda como para el reordenamiento de los items de la cola.* ◊

Un aspecto a ser considerado aún es que este algoritmo utiliza una representación vertical de la base de datos utilizando un formato de bitsets la cual permite reducir el espacio necesario pero afectando la performance en comparación con bitmaps (`Mafia`) y diffsets (`GenMax`). Por otro lado, este algoritmo realiza una proyección de la base de datos para cada nodo y de esta forma los soportes son calculados más eficientemente. Sin embargo, es necesario considerar el tiempo requerido para realizar esa proyección.

En resumen, este algoritmo utiliza una representación *SETree* del espacio de búsqueda. Incorpora información local y global de itemsets frecuentes en cada nodo para mejorar las heurísticas y reducir el espacio de búsqueda. Además de su heurística, utiliza reordenamiento dinámico y realiza podas de itemset no frecuente, superconjunto frecuente y equivalencia con padre (PEP). Utiliza una representación vertical realizando proyecciones de la base de datos sobre los nodos. No requiere testeo de superconjuntos aunque es realizado en menor escala localmente.

3.2.8. FPmax*

Este algoritmo, propuesto en [32], es uno de los algoritmos más recientes y el cual ha superado en diversas comparaciones empíricas a la mayoría de los algoritmos de actual estado del arte en búsqueda de patrones maximales frecuentes. `FPmax*` hace uso de la estructura *FPTree*[16] y utiliza como base el algoritmo *FPGrowth**[17] el cual introduce

[16] Ver sección 2.4.3 página 29.
[17] Ver sección 2.5.2 página 39.

la técnica de arreglos para mejorar la eficiencia en el recorrido de los *FPTrees*. Además de estas estructuras, `FPmax*` introduce una estructura global para registrar los patrones maximales frecuentes (MFI) que constituirán la salida del algoritmo. Es importante mencionar que este algoritmo realiza una búsqueda Depth-First sobre el espacio de búsqueda de patrones frecuentes, donde en cada paso se extiende el itemset candidato con un nuevo item. Debido a esta forma de recorrido, el conjunto de patrones maximales frecuentes (MFI) se genera a medida que se recorre la estructura, agregando nuevos MFI y quitando MFI obsoletos. A tal efecto, se utiliza una estructura *FPTree* también para almacenar el conjunto de MFI.

Este algoritmo utiliza la mayoría de las estrategias de podas y heurísticas antes mencionadas. Por un lado, la tabla de cabeceras es procesada en forma creciente según el soporte de los items, de la misma forma que lo realiza la estrategia de reordenamiento dinámico. Con esto se logra como efecto que los MFI sean encontrados antes que muchos de sus subconjuntos. Por otro lado, antes de extender un itemset con un item candidato se evalúa si este itemset junto con todos los items frecuentes en la cola del nodo son un subconjunto de algún MFI, estrategia conocida como poda de superconjunto maximal frecuente. Por último, la estrategia de poda de superconjunto frecuente también puede ser observada en este algoritmo ya que si al extender el itemset con un item, éste genera un árbol condicional con una sola rama, entonces se realiza la poda. Es importante observar que el conteo por soporte del itemset unido la cola de items frecuentes también aquí se realiza, aunque se efectúa sobre una proyección de la base de datos sobre el nodo (*FPTree Condicional*) la cual es significativamente más pequeña. En este último caso, se puede verificar que si se llegó hasta la instancia de generar un árbol con una única rama entonces este árbol junto con el itemset al cual el árbol está asociado constituyen un MFI.

`FPmax*` utiliza además la estrategia de focalización progresiva con objeto de reducir la cantidad de comparaciones por subconjunto. Para ello, se asocia con cada *FPTree Condicional* un *FPTree* de MFI local, es decir se proyectan para cada nodo solo los MFI que pueden ser utilizados para realizar comparaciones por subconjunto. Como se podrá apreciar, la estructura de *MFI-FPTree* es dinámica ya que se va actualizando a

medida que se descubren nuevos itemsets frecuentes a diferencia del *FPTree* de la base de datos que es estático.

Es interesante notar que este algoritmo se comporta en forma similar a `GenMax` aunque sobre un estructura de datos que comprime la base de datos y mejora su recorrido. Por tal razón, este algoritmo obtiene una mejor performance en la mayoría de los casos analizados que cualquier otro algoritmo antes visto. Esto quedará de manifiesto en la sección 3.2.9 donde se muestran las comparaciones entre los diferentes algoritmos.

En el algoritmo 5 se muestra la estructura general del algoritmo `FPmax*` el cual mantiene la base del algoritmo *FPGrowth** agregando las estrategias de poda y heurísticas propias de los algoritmos para búsqueda de MFI. Entre las líneas 6 y 9 se utiliza la estructura de arreglo para optimizar el recorrido del *FPTree*. En la línea 10 se ordena en forma decreciente los items para la cabecera de items (no confundir este paso con la estrategia de reordenamiento dinámico la cual no es apreciada en el algoritmo). En la línea 11, se evalúa la poda de superconjunto maximal frecuente. En la línea 13 se utiliza la estrategia de focalización progresiva. Por último, luego del paso recursivo, en la línea 1 y 2 se produce eventualmente la poda de superconjunto frecuente y la generación de patrones maximales locales los cuales son añadidos la estructura global en la línea 15 cuando se produce el backtracking.

En resumen, este algoritmo utiliza para representar la base de datos una estructura compacta denominada *FPTree* la cual se recorre en forma Depth-First generando implícitamente un espacio de búsqueda similar a un *SETree*. Se realizan podas por superconjuntos maximales frecuentes y por superconjuntos frecuentes. También son utilizadas las estrategias de reordenamiento dinámico y focalización progresiva junto con otras optimizaciones secundarias.

3.2.9. Resumen y Comparaciones

Resumiremos el estado actual de arte en la búsqueda de patrones maximales frecuentes. A tal efecto se mostrarán diversas tablas con las características de cada algoritmo mencionado. Es importante mencionar que la presente lista de algoritmos no pretende ser

Algoritmo 5 FPmax*(T,M): Recibe un FPTree T, y un MFI-FPTree M para T, retornando M actualizado

```
1: Si T contiene solo un camino P entonces
2:   Insertar P en M;
3: sino
4:   Para todo i en T.Cabecera hacer
5:     Y = T.base ∪ {i}; {T.Base es el itemset considerado}
6:     Si T.array no es VACIO entonces
7:       Cola = { Itemset frecuentes para i en T.array };
8:     sino
9:       Cola = { Itemset frecuentes en la base de patrones condicional de i};
10:    fin Si
11:    Ordenar Cola en orden decreciente según el soporte;
12:    Si Y ∪ Cola no es subconjunto de ningún elemento de M entonces
13:      Construir el FPTree Condicional de Y (T_Y) y el arreglo A_Y;
14:      Inicializar el MFI-FPTree Condicional de Y (M_Y);
15:      Invocar FPmax*(T_Y, M_Y);
16:      Combinar M_Y con M;
17:    fin Si
18:  fin Para
19: fin Si
```

una lista exhaustiva de los algoritmos existentes sino intenta capturar el estado actual mostrando los algoritmos más ampliamente difundidos y en lo posible más recientes.

DataSet	Horizontal		Vertical			
	Standard	Compr.	TidSet	TidSet Compr.	DiffSet	FPTree
MaxMiner	✓					
Pincer-Search	✓					
Depth-Project		✓				
Mafia				✓		
GenMax			✓		✓	
SmartMiner				✓		
FPmax*						✓

Búsqueda	Espacio		Método		
	Reticulados	SETree	Depth-First	Breadth-First	Bidireccional
MaxMiner	✓			✓	
Pincer-Search	✓				✓
Depth-Project		✓	✓		
Mafia		✓	✓		
GenMax		✓	✓		
SmartMiner		✓	✓		
FPmax*		✓	✓		

En la siguiente tabla resumiremos las estrategias utilizadas por los algoritmos. Por motivos de claridad en la tabla utilizaremos las siguientes siglas: Poda Superconjunto Inmediato No Frecuente (Apriori), Poda Superconjunto Frecuente (FHUT), Poda Superconjunto Maximal Frecuente (FHUTMFI), Poda Equivalencia con Padre (PEP), Poda Límite Inferior de Soporte (LUB), Estrategia de Reordenamiento Dinámico (DR), Estrategia de Focalización Progresiva (LMFI).

	Podas					DR	LMFI
	Apriori	FHUT	FHUTMFI	PEP	LUB		
MaxMiner	✓	✓			✓	✓	
Pincer-Search	✓						
Depth-Project	✓	✓				✓	
Mafia	✓	✓	✓	✓		✓	
GenMax	✓	✓	✓			✓	✓
SmartMiner	✓	✓	✓	✓		✓	✓
FPmax*	✓	✓	✓			✓	✓

A partir del análisis de estos algoritmos podemos realizar las siguientes observaciones:

- Según experimentos empíricos reflejados en resultados que han sido presentados en diferentes conferencias, los algoritmos Mafia y GenMax resultan ser los algoritmos más eficientes. Por su parte, SmartMiner reporta según sus experimentos una mejor performance que Mafia y GenMax. Sin embargo, la implementación de estos algoritmos estuvo a cargo de los mismos desarrolladores de SmartMiner a partir de los trabajos teóricos presentados, por lo cual estos resultados requieren una verificación externa que corrobore los mismos. Por otro lado, pruebas empíricas fehacientes reportan al recientemente propuesto algoritmo FPmax* como el mejor algoritmo de

búsqueda de MFI superando a `GenMax` y `Mafia` entre otros para diferentes conjuntos de datos (tanto ralos como densos) [30].

- Las estrategias de poda en general funcionan mejor en búsquedas depth-first, ya que permiten obtener un buen número de MFI generando una pequeña cantidad de itemsets frecuentes (FI). Esto permite que muchos itemsets sean podados utilizando las estrategias de poda por subconjunto maximal frecuente.

- Los algoritmos `Pincer-Search`, `Mafia` y `Depth-Project` generan un superconjunto de los patrones maximales frecuentes, y como etapa final se requiere realizar una última poda para eliminar los mismos. Por su parte, `GenMax`, `SmartMiner` y `FPmax*` solo mantienen los MFI [31].

A continuación presentamos los últimos resultados obtenidos en los experimentos realizados en FIMI 2003 con respecto a MFI utilizando una tabla donde se mostrará el desempeño de diferentes algoritmos de búsqueda de MFI sobre distintos conjuntos de datos, identificando los 2 algoritmos más eficientes para cada dataset según si el soporte utilizado fue un valor alto o bajo. [18]

Base de Datos	Soporte Alto		Soporte Bajo	
	1^{ero}	2do	1^{ero}	2do
accidents	genmax	fpmax*	fpmax*	mafia/genmax
bms1	fpmax*	lcm	lcm	fpmax*
bms2	afopt	fpmax*	afopt	fpmax*
bmspos	fpmax*	genmax	fpmax*	afopt
chess	fpmax*	afopt	mafia	fpmax*
connect	fpmax*	afopt	fpmax*	afopt
kosarak	fpmax*	genmax	afopt	fpmax*
mushroom	fpmax*	mafia	fpmax*	mafia
pumsb	genmax	fpmax*	fpmax*	afopt
pumsb*	fpmax*	mafia	mafia	fpmax*
retail	afopt	lcm	afopt	lcm
T10I5N1KP5KC0.25D200K	fpmax*	afopt	fpmax*	afopt
T20I10N1KP5KC0.25D200K	apriori_borgelt	genmax	fpmax*	afopt
T30I15N1KP5KC0.25D200K	genmax	fpmax*	apriori_borgelt	fpmax*

[18] Para más información dirigirse a `http://fimi.cs.helsinki.fi/experiments/fimi03/maximal/` donde se encuentran los gráficos con los tiempos de ejecución para cada conjuntos de datos.

3.3. Patrones Cerrados

Como hemos mencionado en la sección 3.1 los itemsets frecuentes cerrados (FCI) son un superconjunto de los itemsets frecuentes maximales (MFI). Por lo tanto, la cardinalidad del conjunto de MFI es menor o igual[19] a la cardinalidad del conjunto de FCI. Como resultado, el conjunto de MFI resulta la forma más concisa de representar todo el conjunto de itemsets frecuentes (FI). Sin embargo, aún cuando con los MFI es posible conocer todos los FI, no es posible conocer su soporte, y a tal efecto es necesario enumerar el conjunto de FI y obtener el soporte realizando una nueva pasada sobre la base de datos.

Los FCI tienen la ventaja de representar en forma concisa los FI y además, sin necesidad de generar los FI, conocer el soporte de cada FI. Esta característica hace que los FCI a pesar de no ser tan concisos como los MFI para representar FI, sean una alternativa interesante ya que tienen incorporado la información sobre el soporte.

La noción de conjuntos cerrados tiene sus orígenes en el concepto matemático de Análisis de Conceptos Formales. En este marco se intenta enumerar todos los conjuntos cerrados. Por lo tanto, este concepto ha sido extendido para permitir considerar solamente los conjuntos cerrados frecuentes.

Al igual que con el conjunto de MFI, los FCI pueden ser obtenidos utilizando los algoritmos que son propuestos para generar todo el conjunto de FI y luego filtrar aquellos itemsets que no sean cerrados (o maximales en el caso de MFI). Sin embargo, este acercamiento es ineficiente ya que no saca provecho de las características propias de los MFI y FCI que permiten reducir el espacio de búsqueda. Por lo tanto, los algoritmos que presentaremos intentan aplicar estrategias y heurísticas inherentes de los FCI para reducir este espacio.

3.3.1. Análisis de Conceptos Formales

El *análisis de conceptos formales* (Formal Concept Analysis) (FCA) [27] es una técnica ampliamente difundida desde su origen en 1982. Esta técnica ha sido aplicada en diversas

[19] En la mayoría de los casos es estrictamente menor.

áreas del conocimiento como psicología, ciencias sociales, ingeniería civil, antropología, medicina, biología, ciencias lingüísticas, matemáticas, ingeniería del software y ha sido clasificada como la columna vertebral del descubrimiento de conocimiento conceptual [17, 75]. Esta técnica esta basada en la teoría de reticulados y busca formalizar matemáticamente la definición filosófica de *concepto*. Para ello los datos son organizados en un reticulado sobre la cual se pueden establecer implicaciones, dependencias, regularidades, excepciones y determinar la estructura inherente de los mismos. Una presentación completa y rigurosa sobre los fundamentos matemáticos del análisis de conceptos formales se encuentra fuera del alcance de esta tesis y puede encontrarse en [27]. A continuación enunciaremos simplemente las definiciones más importantes que se utilizarán para la búsqueda de itemsets cerrados.

Desde un el punto de vista filosófico un *concepto* es una unidad de pensamiento constituida por dos partes, la *extensión* y la *intensión*[20]. La *extensión* cubre todos los objetos que pertenecen al concepto y la *intensión* comprende todos los atributos válidos para todos los objetos [7][21]. En una idea más intuitiva, la *intensión* de un concepto puede ser pensado como la colecciones de asociaciones que el concepto crea en la mente de la persona que lo usa o escucha, mientras que la *extensión* se refiere a los referentes que el concepto describe.

Ejemplo 3.3.1

Consideremos el concepto de "auto" [81]. Los objetos asociados a este concepto tiene los siguientes atributos: tiene ruedas, tiene motor, tiene asientos, sirve para transporte, etc. A su vez todos los objetos que tienen estos atributos son llamados "autos", por ejemplo, toyota, mercedes, nissan, etc. ♢

Ejemplo 3.3.2

En otro contexto, la noción de concepto podría tener una cierta analogía a la definición de clase en POO. Donde cada clase está determinada por sus atributos y existen instancias

[20]Utilizaremos los términos utilizados en la bibliografía ya que no existe una traducción del término *intensión* por no ser palabra válida del idioma inglés.

[21]Se puede consultar también la siguiente dirección web `http://en.wikipedia.org/wiki/Intension`.

de esa clase denominados objetos. ◊

Se pueden establecer varias relaciones:

1. *Relaciones entre Conceptos*: Establece una jerarquía de conceptos estableciendo una relación "subconcepto-superconcepto".

2. *Relaciones entre Atributos*: Implicaciones entre atributos, de igual forma que las capturadas por las reglas de asociación.

3. *Relación de incidencia*: Relaciones entre objetos y atributos ("Un objeto tiene un atributo").

La idea propuesta en [27] combina los objetos, atributos y relación de incidencia en una definición matemática de un contexto formal. Es decir, dado un conjunto de objetos y un conjunto de atributos y una relación que incide de un conjunto a otro se define un concepto. Estas relaciones entre los distintos conjuntos de objetos y los conjuntos de atributos está representado por un *contexto formal* (formal context).

Definición 3.3.1 (Contexto Formal, Wille et al., 1999)
Un contexto formal (Formal Context) es un tupla (G, M, I)*, donde* G *y* M *son conjuntos e* $I \subseteq G \times M$ *es la relación de* incidencia*. Los elementos de* G *son llamados objetos y los elementos de* M *son llamados atributos. Para un* $g \in G$ *y* $m \in M$ *arbitrarios,* gIm *denota que* g *está relacionado con* m*, es decir,* $(g, m) \in I$*.* ♦

Los símbolos G y M provienen de la traducción al idioma alemán de los términos *objetos* ("**G**egenstände") y *atributos* ("**M**erkmale"). Los *contextos formales* son representados por *tablas de incidencia* (cross table) donde las filas representan los distintos objetos y las columnas representan los atributos. Cada relación entre los objetos y los atributos se marcará en la tabla usualmente con una cruz.

Ejemplo 3.3.3
Consideremos el siguiente contexto formal de seres vivos el cual es un extracto del ejemplo presentado en [81]. El conjunto G *de objetos esta dado por* $G = \{$*Paloma, Gallina, Pato,*

Objetos	2 Patas	4 Patas	Plumas	Pelo	Vuela	Corre	Nada	Caza	Melena
Paloma	×		×		×				
Gallina	×		×			×			
Pato	×		×		×		×		
Ganso	×		×		×		×		
Halcón	×		×		×			×	
Águila	×		×		×			×	
Zorro		×		×		×		×	
Perro		×		×		×			
Leon		×		×		×		×	×

Cuadro 3.1: Contexto Formal de Seres Vivos

Ganso, Halcón, Águila, Zorro, Perro, León} y el conjunto de atributos M está constituido por M ={*2 Patas, 4 Patas, Plumas, Pelo, Vuela, Corre, Nada, Caza, Melena*}. *En el cuadro 3.1 se puede apreciar la representación del contexto formal.*

La noción central de FCA es una dualidad llamada *conexión de Galois* (Galois Connection) [58]. Esta dualidad es observada en las relaciones entre objetos y atributos. Una conexión de Galois implica que si existe una relación entre un conjunto (en este caso de objetos) y otro conjunto (atributos), si uno de los conjuntos se agranda el otro disminuye.

Definición 3.3.2 (Conexión de Galois, Wille et al., 1999)
Sea (G, M, I) *un contexto formal con* $X \subseteq G$ *e* $Y \subseteq M$. *Los mapeos,*

$$s : \mathcal{P}(G) \mapsto \mathcal{P}(M) \qquad s(X) = \{m \in M \mid (\forall g \in X)\ gIm\}$$
$$t : \mathcal{P}(M) \mapsto \mathcal{P}(G) \qquad t(Y) = \{g \in G \mid (\forall m \in Y)\ gIm\}$$

definen una conexión de Galois entre $\mathcal{P}(G)$ *y* $\mathcal{P}(M)$, *el conjunto de partes de* G *y* M *respectivamente.* ♦

El conjunto $s(X)$ es el conjunto de atributos que son comunes a todos los objetos X y $t(Y)$ es el conjunto de objetos que son comunes a todos los atributos Y. En [88, 55] se mencionan las siguientes propiedades de las conexiones de Galois.[22]

[22]Estas propiedades son análogas a la proposición 2.5.1 vista en la página 37.

Propiedad 3.3.1 (Propiedades de Conexiones de Galois)
Sea (G, M, I) un contexto formal, $X, X_1, X_2 \subseteq G$; $Y, Y_1, Y_2 \subseteq M$ y s, t funciones entre $\mathcal{P}(G)$ y $\mathcal{P}(M)$ definiendo conexiones de Galois.

1. $X_1 \subseteq X_2 \rightarrow s(X_2) \subseteq s(X_1)$.
2. $Y_1 \subseteq Y_2 \rightarrow t(Y_2) \subseteq t(Y_1)$.
3. $Y \subseteq s(X) \Leftrightarrow X \subseteq t(Y)$. □

Ejemplo 3.3.4
Consideremos las siguientes aplicaciones de s y t sobre el ejemplo del cuadro 3.1.

$$s(\{Paloma\}) = \{2\ Patas, Plumas, Vuela\}$$
$$s(\{Gallina\}) = \{2\ Patas, Plumas, Corre\}$$
$$s(\{Paloma, Gallina\}) = \{2\ Patas, Plumas\} \quad [\ Prop.\ 1\ ]$$
$$t(\{2\ Patas, Plumas\}) = \Big\{Paloma, Gallina, Pato, Ganso, Halcón, Águila\Big\} \quad [\ Prop.\ 3\ ]$$
$$t(\{4\ Patas, Pelo, Corre, Caza\}) = \{Zorro, León\}$$
$$t(\{4\ Patas, Pelo, Corre\}) = \{Zorro, Perro, León\} \quad [\ Prop.\ 2\ ]$$

◊

Definición 3.3.3 (Operador de Clausura)
Sea S un conjunto y c una función $c : \mathcal{P}(S) \mapsto \mathcal{P}(S)$. Diremos que c es un operador de clausura (closure operator) sobre S si para todo $X, Y \subseteq S$ se satisfacen las siguientes propiedades:

1. *Extensión: $X \subseteq c(X)$.*
2. *Monotonicidad: Si $X \subseteq Y$ entonces $c(X) \subseteq c(Y)$.*
3. *Idempotencia: $c(c(X)) = c(X)$.* ⧫

Definición 3.3.4 (Conjunto Clausurado)

Sea c un operador de clausura sobre S. Un subconjunto X de S es llamado clausurado *o cerrado (closed) si $c(X) = X$.* ♦

Se puede demostrar que la composición de las funciones s y t es un operador de clausura, ya sea tomando $c = s \circ t$, o $c = t \circ s$. Estamos entonces en condiciones de dar una definición formal de concepto.

Definición 3.3.5 (Concepto, Wille et al., 1999)

Un concepto (Concept) de un contexto formal (G, M, I) es un par (X, Y) donde $X \subseteq G$, $Y \subseteq M$, $s(X) = Y$ y $t(Y) = X$. El conjunto X es llamado extensión y el conjunto Y es llamado intensión. También se podrá notar un concepto como $X \times Y$. ♦

Es importante destacar que, por definición, tanto X como Y son conjuntos cerrados, es decir, $X = t(Y) = t(s(X)) = t \circ s(X) = c(X)$ y de igual forma $Y = c(Y)$. Esta propiedad de clausura de las conexiones de Galois permite obtener un conjunto clausurado comenzando con cualquier conjunto tanto de objetos como de atributos. Por lo tanto un concepto formal puede ser encontrado tomando un subconjunto de objetos, encontrando todos sus atributos y determinando todos los objetos que tengan dichos atributos.

Ejemplo 3.3.5

Tomemos por ejemplo el objeto Paloma y calculemos la función $t(s(Paloma))$,

$$\begin{aligned} t(s(Paloma)) &= t(\{2\ Patas,\ Plumas,\ Vuela\}) \\ &= \{Paloma,\ Gallina,\ Pato,\ Ganso,\ Halcón,\ Águila\} \end{aligned}$$

Este conjunto $\{Paloma,\ Gallina,\ Pato,\ Ganso,\ Halcón,\ Águila\}$ es un conjunto cerrado constituido por todos los objetos que tienen los atributos compartidos con el objeto Paloma ($\{2\ Patas,\ Plumas,\ Vuela\}$).

En forma similar si tomamos un conjunto de atributos como $\{Pelo,\ Corre\}$ y le apli-

camos la función t,

$$s(t(\{Pelo,\ Corre\})) = s(\{Zorro,\ Perro,\ León\})$$
$$= \{4\ Patas,\ Pelo,\ Corre\}$$

Que es el conjunto cerrado que representa todos los atributos compartidos por los objetos que tenían el conjunto de atributos $\{Pelo,\ Corre\}$. ◊

En forma genérica un concepto formal puede ser generado tanto por atributos como por objetos.

Definición 3.3.6 (Concepto de Atributos y de Objetos)

Sea $m \in M$ *y* $g \in G$. *Se llamará concepto de atributo (attribute concept) al concepto formal* $\alpha(m) = (t(m), c(m)) = t(m) \times c(m)$ *y se llamará concepto de objeto (object concept) al concepto formal* $\beta(g) = (c(g), s(g)) = c(g) \times s(g)$. ⧫

Definición 3.3.7 (Subconcepto Formal)

Un concepto formal (X_1, Y_1) *es un subconcepto formal (formal subconcept) de* (X_2, Y_2), *notado* $(X_1, Y_1) \leq (X_2, Y_2)$ *si y solo si* $X_1 \subseteq X_2$ *(y por consecuencia de las propiedades* $Y_2 \subseteq Y_1$*). Diremos además que* (X_2, Y_2) *es un superconcepto formal.* ⧫

Ejemplo 3.3.6

Continuando con el ejemplo del cuadro 3.1 podemos generar 12 *conceptos. Para ello calculemos los correspondientes conceptos de objetos y atributos.*[23]

C	*Objeto*	*s(Objeto)*	*c = t(s(Objeto))*
1	$\emptyset$	M	$\emptyset$
2	*Paloma*	{2 *Patas, Plumas, Vuela*}	{*Paloma, Pato, Ganso, Halcón, Águila*}
3	*Gallina*	{2 *Patas, Plumas, Corre*}	{*Gallina*}
4	*Pato*	{2 *Patas, Plumas, Vuela, Nada*}	{*Pato, Ganso*}
5	*Halcón*	{2 *Patas, Plumas, Vuela, Caza*}	{*Halcón, Águila*}
6	*Zorro*	{4 *Patas, Pelo, Corre, Caza*}	{*Zorro, León*}
7	*Perro*	{4 *Patas, Pelo, Corre*}	{*Zorro, Perro, León*}
8	*León*	{4 *Patas, Pelo, Corre, Caza, Melena*}	{*León*}

[23]Solo se describen aquellos conceptos que generan un nuevo concepto.

C	*Atrib.*	t(*Atrib.*)	$c = s(t($*Atrib.*$))$
9	$\emptyset$	G	$\emptyset$
10	*Caza*	{*Halcón, Águila, Zorro, León*}	{*Caza*}
11	*Corre*	{*Gallina, Zorro, Perro, León*}	{*Corre*}
12	*Plumas*	{*Paloma, Gallina, Pato, Ganso, Halcón, Águila*}	{2 *Patas, Plumas*}

$\Diamond$

Notaremos $\mathcal{B}(G, M, I)$ (del alemán "Begriffe")[24] al conjunto de conceptos formales de (G, M, I). Este conjunto de conceptos genera un reticulado denominado *reticulado de Galois* (Galois Lattice) (también llamado reticulado de conceptos) formado por la relación de orden $\leq$. En resumen, un reticulado de conceptos consiste del conjunto de conceptos de un contexto formal y la relación de subconcepto entre los conceptos.

Por otro lado, las operaciones $\vee$ (join) y $\wedge$ (meet) se definen como:

$$join : (X_1 \times Y_1) \vee (X_2 \times Y_2) = t(s(X_1 \cup X_2)) \times (Y_1 \cap Y_2)$$
$$meet : (X_1 \times Y_1) \wedge (X_2 \times Y_2) = (X_1 \cap X_2) \times s(t(Y_1 \cup Y_2))$$

En el reticulado, cada concepto formal se representa como un nodo y si los conceptos c_1 y c_2 se encuentran en relación $c_1 \leq c_2$ entonces de dibujará una línea entre ambos. La forma tradicional de etiquetar un reticulado de conceptos consiste en ubicar los objetos formales (*extensión*) ligeramente por *debajo* del nodo y los atributos formales (*intensión*) ligeramente por *arriba* del nodo.

Otra forma de etiquetar cada concepto es utilizar un esquema de etiquetado minimal o reducido en donde cada objeto y atributo solo aparece una vez en el reticulado. Con este esquema, una etiqueta de objeto g será dibujada en el concepto de objeto $\beta(g)$ y una etiqueta de atributo m será dibujada en el concepto de atributo $\alpha(m)$.

La *extensión* de un concepto formal bajo el esquema reducido está dado por todas las etiquetas de objeto que pueden ser alcanzadas por caminos que descienden desde el concepto formal. Este conjunto de conceptos a lo largo de los caminos que desciende se conoce como *order ideal*. Por otro lado, la *intensión* de un concepto se puede obtener considerando todas las etiquetas de atributos que se alcanzan por caminos que suben desde el concepto formal. Este conjunto es conocido como *order filter*.

[24] Cuya traducción al castellano significa "Conceptos".

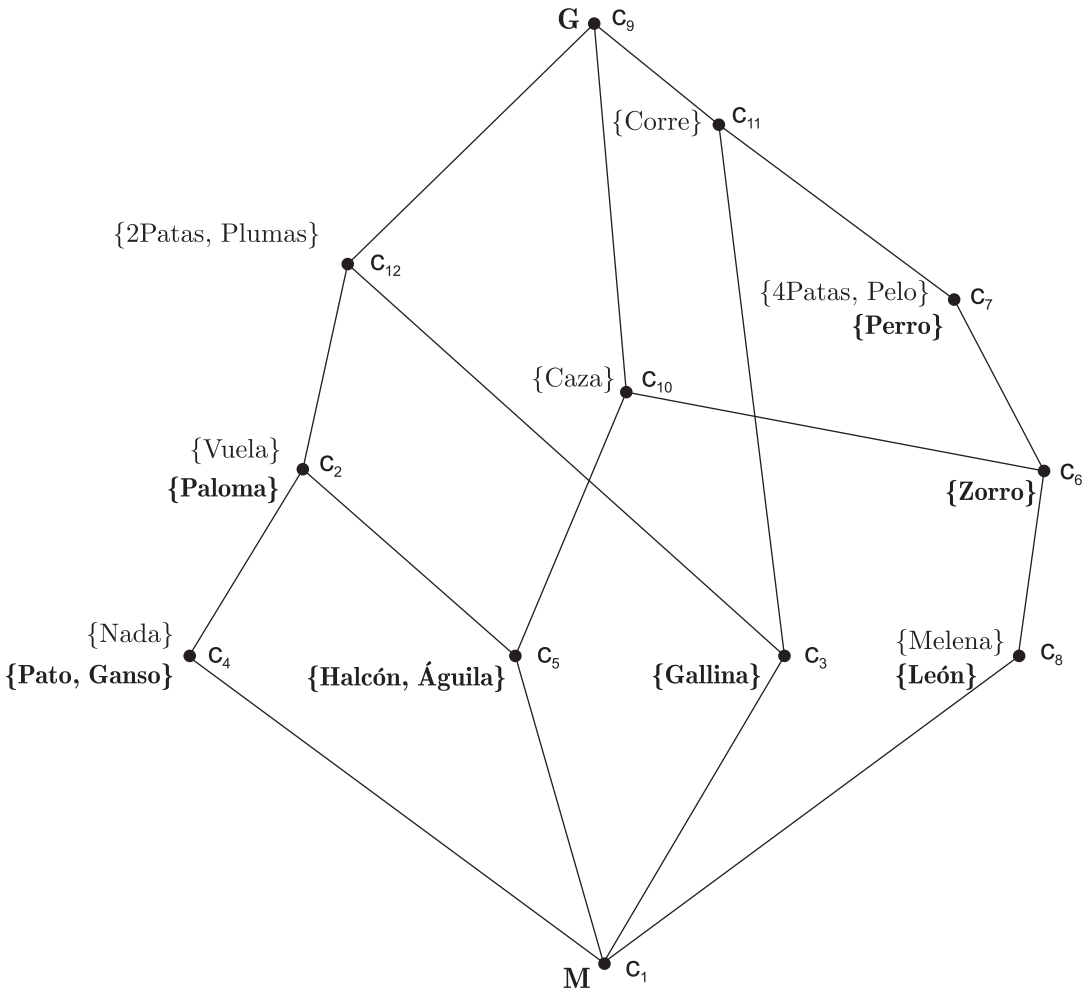


Figura 3.7: Reticulado de Galois para el Contexto de Seres Vivos

Ejemplo 3.3.7

En la figura 3.7 se aprecia el reticulado de Galois con un etiquetado reducido asociado al cuadro 3.1 y los conceptos identificados en el ejemplo 3.3.6. Podemos ver entonces que para el concepto C_5 *el nodo se etiqueta con una etiqueta de objeto con valor* {*Halcón, Águila*}*. Desde allí, subiendo podemos identificar su* intensión *o conjunto de atributos* {*Vuela, Plumas,* 2 *Patas, Caza*}*. De la misma forma para el concepto* C_{11} *el nodo se etiqueta con una etiqueta de atributo con valor* {*Corre*} *y descendiendo se obtiene el conjunto de* extensión *o conjunto de objetos* {*Perro, Zorro, León, Gallina*}*. Por otro lado, el nodo etiquetado tanto con etiqueta de atributo* {*Vuela*} *como con etiqueta de*

objeto $\{Paloma\}$ *representa el concepto* C_2 *y desde allí puede identificarse los atributos* $\{Vuela, Plumas, 2\ Patas\}$ *y los objetos* $\{Paloma, Halcón, Águila, Pato, Ganso\}$. *Además, si quisiéramos conocer todos los objetos que tienen atributos* $\{Caza\}$ *(*C_{10}*) y* $\{Pelo, 4\ Patas\}$ *(*C_7*), buscando las ramas en común hacia abajo encontramos a* $\{Zorro, León\}$. *En forma similar, si quisiéramos obtener atributos comunes entre* $\{Gallina\}$ *(*C_3*) y* $\{León\}$ *(*C_8*) encontramos hacia arriba que comparten el atributo* $\{Corre\}$ *(*C_{11}*).* ◊

Para más información sobre Análisis de Conceptos Formales (FCA) se recomienda visitar las siguientes direcciones electrónicas:

- `http://www.fcahome.org.uk/`
- `http://www.aifb.uni-karlsruhe.de/WBS/gst/FBA03.shtml`
- `http://www.mathematik.tu-darmstadt.de/ags/ag1/fag1/`

3.3.2. Itemsets Cerrados Frecuentes

El análisis de conceptos formales (FCA) ha sido utilizado para generar conjuntos cerrados con aplicaciones en múltiples áreas. Existen, por lo tanto, varios algoritmos propuestos para realizar dicha tarea. Estos algoritmos además de obtener conjuntos cerrados, también permiten generar reglas de asociación entre los objetos, entre los atributos y entre objetos y atributos, como también generar reglas de implicación. Una característica importante a destacar de estas reglas es que las reglas de asociación generadas no son redundantes logrando así mostrar solamente las reglas más interesantes al usuario. Sin embargo, estos algoritmos se han desarrollado utilizando pequeñas bases de datos donde la cantidad de itemsets no era una limitación y no afectaba el desempeño del algoritmo.

Como veremos a continuación, los conjuntos cerrados no están limitados a bases de datos pequeñas sino que el concepto de conjuntos cerrados puede aplicarse al contexto de minería de datos como también al marco matemático de FCA que ayuda a generar dichos conjuntos. A diferencia de FCA donde se enumeran todos los conjuntos cerrados, en el contexto de minería de datos se generarán solamente aquellos conjuntos cerrados que sean frecuentes y por lo tanto se agrega el umbral de mínimo soporte.

tid	*Itemsets*
1	CD
2	ACD
3	AB
4	AC
5	ABC
6	CD
7	AB
8	ABC
9	ABC
10	$ABCD$

δ	1	2	3	4	5	6	7	8	9	10
A		×	×	×	×		×	×	×	×
B			×		×		×	×	×	×
C	×	×		×	×	×		×	×	×
D	×	×				×				×

Cuadro 3.2: Ejemplo de Contexto de Minería de Datos

Veamos como utilizar el marco matemático de FCA en el contexto de minería de datos. Consideremos un reticulado booleano $(\mathcal{P}(\mathcal{I}), \subseteq)$ como fuera definido en la sección 2.6.1, donde $\mathcal{I}$ representa el conjunto de items y $\mathcal{P}$ consiste del conjunto de partes de $\mathcal{I}$. Sea $\mathcal{T}$ el conjunto de identificadores de transacciones ($Tidset$) y δ una relación binaria tal que $\delta \subseteq \mathcal{I} \times \mathcal{T}$ donde cada tupla (x, y) que pertenece a δ denota el hecho que el item $x \in \mathcal{I}$ está relacionado con la transacción $y \in \mathcal{T}$.

Definición 3.3.8 (Contexto de Minería de Datos)
Un contexto de minería de datos es un tupla $\mathcal{D} = (\mathcal{I}, \mathcal{T}, \delta)$ donde $\mathcal{I}$ y $\mathcal{T}$ son conjuntos finitos de itemset y tidset respectivamente. ⧫

Ejemplo 3.3.8
En el cuadro 3.2 se muestra la base de datos origen junto con la tabla de incidencia que representa el contexto de minería de datos. ◊

Definición 3.3.9 (Funciones de Mapeo, Zaki, 1998)
Sea $(\mathcal{I}, \mathcal{T}, \delta)$ un contexto de minería de datos, $X \subseteq \mathcal{I}$ e $Y \subseteq \mathcal{T}$. Los siguientes mapeos

definen conexiones de Galois entre $\mathcal{P}(\mathcal{I})$ *y* $\mathcal{P}(\mathcal{T})$.

$$t : \mathcal{I} \mapsto \mathcal{T} \qquad t(X) = \{y \in \mathcal{T} \mid (\forall x \in X)\ x\delta y\}$$
$$i : \mathcal{T} \mapsto \mathcal{I} \qquad i(Y) = \{x \in \mathcal{I} \mid (\forall y \in Y)\ x\delta y\}$$ ♦

Intuitivamente, la función de mapeo $t(X)$ representa el conjunto de todas las transacciones (*tidset*) que contienen el itemset X y la función de mapeo $i(Y)$ representa el conjunto de items (o el *itemset*) que está contenido en todas las transacciones en Y. Es importante distinguir que la función de mapeo s en el contexto de FCA corresponde con la función de mapeo t, ya que realizan un mapeo de los objetos a los atributos y en el caso de la minería de datos los objetos están representados por itemset y los atributos por tidset. Por otro lado la función t de FCA corresponde con la función i en el contexto de minería de datos. Esta aclaración es importante, ya que se produce una colisión de nombres de función por lo cual en adelante utilizaremos la notación de minería de datos por defecto (a no ser que explícitamente se mencione lo contrario).

Lema 3.3.1 (Operador de Clausura, Zaki, 1998)
Sea $X \subseteq \mathcal{I}$ *e* $Y \subseteq \mathcal{T}$. *Sea* $c_{it}(X)$ *y* $c_{ti}(Y)$ *la composición de las funciones de mapeo* i *y* t *tal que* $c_{it}(X) = i \circ t(X) = i(t(X))$ *y* $c_{ti}(Y) = t \circ i(Y) = i(t(X))$. *Entonces* $c_{it} : \mathcal{P}(\mathcal{I}) \mapsto \mathcal{P}(\mathcal{I})$ *y* $c_{ti} : \mathcal{P}(\mathcal{T}) \mapsto \mathcal{P}(\mathcal{T})$ *son operadores de clausura sobre itemsets y tidsets respectivamente.*□

Definición 3.3.10 (Itemset y Tidset Cerrado, Zaki, 1998)
Llamaremos itemset cerrado *al itemset* $X \subseteq \mathcal{I}$ *tal que* $X = c_{it}(X)$. *En forma dual, llamaremos* tidset cerrado *al tidset* $Y \subseteq \mathcal{T}$ *tal que* $Y = c_{ti}(Y)$. ♦

Definición 3.3.11 (Concepto en el contexto de minería de datos, Zaki, 1998)
Llamaremos concepto *de un contexto* $(\mathcal{I}, \mathcal{T}, \delta)$ *al par* $X \times Y$ *donde* X *es un itemset cerrado e* Y *es un tidset cerrado.* ♦

Definición 3.3.12 (Concepto de item y de transacción, Zaki, 1998)
Llamaremos concepto de item (item concept) al concepto $C(x) = c_{it}(x) \times t(x)$ *y concepto de transacción (tid concept) al concepto* $C(y) = i(y) \times c_{ti}(y)$. ♦

Estas definiciones de concepto de item y de transacción son análogas a las definiciones de concepto de objetos y de atributos.

Ejemplo 3.3.9

Consideremos el ejemplo 3.3.8 y calculemos el tidset cerrado para la transacción 5,

$$c_{ti}(\{5\}) = t(i(\{5\})) = t(\{A, B, C\}) = \{5, 8, 9, 10\}$$

Por lo tanto, el tidset cerrado es $\{5, 8, 9, 10\}$ *y su correspondiente itemset cerrado es* $\{A, B, C\}$*. Entonces el par* $\{A, B, C\} \times \{5, 8, 9, 10\}$ *forma un concepto.* $\Diamond$

El conjunto de conceptos forma un reticulado con las siguientes operaciones $\vee$ (join) y $\wedge$ (meet) análogas a las definidas para FCA:

$$join : (X_1 \times Y_1) \vee (X_2 \times Y_2) = c_{it}(X_1 \cup X_2) \times (Y_1 \cap Y_2)$$
$$meet : (X_1 \times Y_1) \wedge (X_2 \times Y_2) = (X_1 \cap X_2) \times c_{ti}(Y_1 \cup Y_2)$$

Ejemplo 3.3.10

Podemos generar entonces el reticulado de Galois asociado a la base de datos del ejemplo 3.3.8. A continuación se detallan los conceptos de item y conceptos de transacciones y en la figura 3.8 se muestra dicho reticulado.

	Item	$t(Item)$	$c_{it} = i(t(Item))$	*Concepto*	*Etiq.*
C_1	$\emptyset$	$\mathcal{T}$	$\emptyset$	$\emptyset \times \mathcal{T}$	$\emptyset$
C_2	A	$\{2, 3, 4, 5, 7, 8, 9, 10\}$	$\{A\}$	$\{A\} \times \{2, 3, 4, 5, 7, 8, 9, 10\}$	$\{A\}$
C_3	B	$\{3, 5, 7, 8, 9, 10\}$	$\{A, B\}$	$\{A, B\} \times \{3, 5, 7, 8, 9, 10\}$	$\{3, 7\}$
C_4	C	$\{1, 2, 4, 5, 6, 8, 9, 10\}$	$\{C\}$	$\{C\} \times \{1, 2, 4, 5, 6, 8, 9, 10\}$	$\{C\}$
C_5	D	$\{1, 2, 6, 10\}$	$\{C, D\}$	$\{C, D\} \times \{1, 2, 6, 10\}$	$\{1, 6\}$

	Tid	$i(Tid)$	$c_{ti} = t(i(Tid))$	*Concepto*	*Etiq.*
C_6	2	$\{A, C, D\}$	$\{2, 10\}$	$\{A, C, D\} \times \{2, 10\}$	$\{2\}$
C_7	4	$\{A, C\}$	$\{2, 4, 5, 8, 9, 10\}$	$\{A, C\} \times \{2, 4, 5, 8, 9, 10\}$	$\{4\}$
C_8	5	$\{A, B, C\}$	$\{5, 8, 9, 10\}$	$\{A, B, C\} \times \{5, 8, 9, 10\}$	$\{5, 8, 9\}$
C_9	10	$\mathcal{I}$	$\{10\}$	$\mathcal{I} \times \{10\}$	$\{10\}$

$\Diamond$

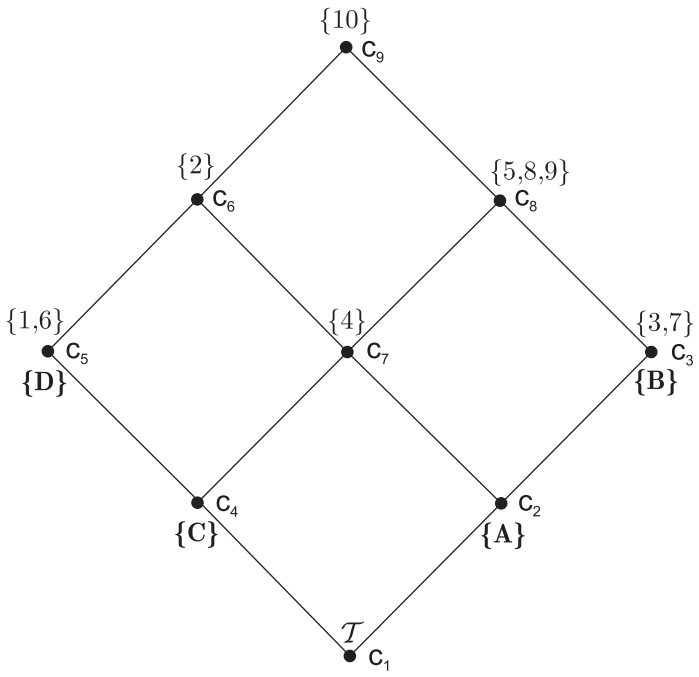


Figura 3.8: Reticulado de Galois para la base de datos del ejemplo 3.3.8

En particular, estaremos interesados en aquellos itemset cerrados cuyo soporte supere el umbral de mínimo soporte, es decir los itemset cerrados frecuentes (FCI). Por tal motivo es necesario conocer el soporte de los itemsets cerrados.

Definición 3.3.13 (Soporte de Itemset Cerrado, Zaki, 1998)
El soporte de un itemset cerrado X o concepto $X \times Y$ es igual a la cardinalidad del tidset cerrado $Y = t(X)$, es decir, soporte$(X) = |Y| = |t(X)|$. ♦

Como ha sido mencionado, una de las particularidades que los itemsets cerrados frecuentes es que a partir de ellos se pueden generar todos los itemsets frecuentes y además conocer el soporte de cualquier itemset sin necesidad de realizar una nueva pasada por la base de datos.

En primer lugar, todo itemset puede ser generado utilizando la operación *glb* (*join* o $\vee$) entre itemsets cerrados. En particular para generar los itemsets frecuentes bastará utilizar los itemsets frecuentes cerrados (FCI).

Ejemplo 3.3.11

Supongamos un umbral de mínimo soporte de 3*. Utilizando el ejemplo anterior podemos observar que los únicos conceptos o itemsets cerrados que no son frecuentes son* C_6 *y* C_9*. Supongamos querer evaluar si el itemset* $\{B, D\}$ *es frecuente, entonces utilizamos los conceptos* $c_{it}(B) = \{A, B\}$ *y* $c_{it}(D) = \{C, D\}$*.*

$$(\{A, B\} \times \{3, 5, 7, 8, 9, 10\}) \vee (\{C, D\} \times \{1, 2, 6, 10\}) = c_{it}(\{A, B, C, D\}) \times (\{10\}) = C_9$$

Por lo tanto el itemset $\{B, D\}$ *es infrecuente. Si consideramos el itemset* $\{B, C\}$ *utilizando los conceptos* $c_{it}(B) = \{A, B\}$ *y* $c_{it}(C) = \{C\}$*.*

$$(\{A, B\}\times\{3, 5, 7, 8, 9, 10\})\vee(\{C\}\times\{1, 2, 4, 5, 6, 8, 9, 10\}) = c_{it}(\{A, B, C\})\times(\{5, 8, 9\}) = C_8$$

Por lo tanto el itemset $\{B, C\}$ *es frecuente.* ◊

Lema 3.3.2 (Soporte)

El soporte de un itemset X *es igual al soporte de su clausura, es decir,* $soporte(X) = soporte(c_{it}(X))$*.* □

Ejemplo 3.3.12

Para los anteriores itemsets, $soporte(\{B, D\}) = soporte(\{A, B, C, D\}) = |\{10\}| = 1$ *(infrecuente) y* $soporte(\{B, C\}) = soporte(\{A, B, C\}) = |\{5, 8, 9\}| = 3$ *(frecuente).* ◊

3.3.3. A-Close

El primer algoritmo eficiente para la búsqueda de patrones cerrados es `A-Close` y fue propuesto en [55]. Allí se propone el uso de patrones cerrados para la enumeración concisa de los patrones frecuentes y para la generación de reglas de asociación no redundantes y dicho trabajo es continuado en [5].

Para la generación de los itemsets cerrados, `A-Close` está basado en las propiedades de los reticulados de itemset cerrados. Este algoritmo consta de 2 etapas bien definidas:

1. Se determina un *conjunto de generadores.*

2. Se aplica el operador de clausura c_{it} sobre el conjunto de generadores a fin de obtener el conjunto de itemsets cerrados.

En la primer etapa se determina el conjunto de itemsets frecuentes más pequeños que generan por aplicación del operador de clausura c_{it} un itemset cerrado. Durante esta etapa se realiza una búsqueda bottom-up del reticulado del conjunto de partes de items por niveles similar a la estrategia utilizada por `Apriori`, podando *a*) aquellos itemsets candidatos infrecuentes y *b*) aquellos itemsets candidatos frecuentes que no pueden ser generadores.

Durante la segunda etapa se procede a computar la clausura de todos los generadores encontrados en la primer etapa. Por tal motivo, para cada itemset generador X se procede a determinar todas las transacciones donde X ocurre y luego se obtiene el conjunto de items comunes a todas esas transacciones, es decir, $c_{it}(X) = \bigcap_{X \subseteq i(T)} i(T)$ donde T es un identificador de transacción (*tid*). Debido a que esta operación es costosa, en [55] se propone una optimización que reduce el número de clausuras a ser realizadas.

Para obtener el conjunto de generadores, `A-Close` utiliza el concepto de *generadores minimales* de itemsets cerrados. Este conjunto permite que se realicen la menor cantidad de operaciones de clausura posibles para obtener el conjunto de itemsets cerrados.

Definición 3.3.14 (Generador Minimal, Bastide et al. 2000)
Sea X un itemset cerrado. Un itemset Y es generador *de X si y solo si a) $Y \subseteq X$ y b) $soporte(Y) = soporte(X)$. El itemset Y es un* generador propio *si $Y \subset X$ (por definición Y no puede ser cerrado). Sea $G(X)$ el conjunto de generadores de X. El itemset Y es un* generador minimal *X si $Y \in G(X)$ y no existe un subconjunto de Y en $G(X)$.*♦

Con el fin de reducir la cantidad de generadores y obtener un conjunto minimal de estos, además de la poda de generadores infrecuentes, se utiliza el siguiente lema.

Lema 3.3.3 (Bastide et al. 2000)
Sea X_1 un itemset y X_2 un subconjunto de X_1 donde $soporte(X_1) = soporte(X_2)$ entonces $c_{it}(X_1) = c_{it}(X_2)$ y $\forall X_3 \subseteq \mathcal{I}, c_{it}(X_1 \cup X_3) = c_{it}(X_2 \cup X_3)$. □

Este lema provee los fundamentos para la poda de itemsets frecuentes que no pueden ser generadores, es decir, aquellos itemsets (X_2) para los cuales dentro del conjunto de generadores ya existe otro itemset (X_1) que genera la misma clausura y para completar se puede observar que este itemset X_2 es redundante ya que unido a cualquier otro itemset (X_3) generaría el mismo conjunto que X_1 unido a ese mismo itemset.

Además de estas dos podas, se realiza una tercer poda debido a la utilización del algoritmo `Apriori`, la cual consiste en eliminar aquellos itemsets del nivel $i+1$ que tengan algún subconjunto inmediato que no pertenezca al nivel i. Este análisis debe realizarse ya que por el método de combinación de itemsets utilizado por `Apriori`, éste puede generar itemset candidatos de subconjuntos podados. Por ejemplo, supongamos que en el nivel 2 se encuentran los generadores AB y AC pero el itemset BC fue podado (ya sea por infrecuente o redundante) entonces puede suceder que al combinar AB y AC se genere el itemset ABC para el nivel 3 el cual es claramente infrecuente o redundante. Esta verificación se presenta en [55] como otro tipo de poda del espacio de búsqueda que mejora la eficiencia del algoritmo `A-Close`. Sin embargo, a nuestro entender esta eliminación no constituye un nuevo tipo de poda sino simplemente se intenta mantener consistentes las podas ya realizadas.

Como último paso se requiere eliminar del conjunto de generadores aquellos itemset minimales que generen la misma clausura ya que estos pueden no haber sido totalmente eliminados por los mecanismos de poda. Supongamos por ejemplo que $G(ABCD) = \{AB, ABC, CD, BCD\}$. En tal caso, el conjunto de generadores mínimos está constituido por el conjunto $\{AB, CD\}$ sin embargo ambos elementos generan la misma clausura y por lo tanto uno de ellos es redundante.

Ejemplo 3.3.13

Consideremos nuevamente la base de datos utilizada en el ejemplo 3.3.8 (pág. 107). A continuación se muestra nuevamente dicha base de datos junto con los soportes de cada potencial itemset. Supongamos tener un umbral de mínimo soporte de 2.

tid	*Itemsets*
1	CD
2	ACD
3	AB
4	AC
5	ABC
6	CD
7	AB
8	ABC
9	ABC
10	$ABCD$

Soportes

$A = 8$, $AB = 6$, $ABC = 4$, $ABCD = 1$
$B = 6$, $AC = 6$, $ABD = 1$
$C = 8$, $AD = 2$, $ACD = 2$
$D = 4$, $BC = 4$, $BCD = 1$
$BD = 1$,
$CD = 4$,

Como primer paso para obtener los generadores, se construye un conjunto G_1 *con los* 1*-itemset candidatos y se eliminan aquellos itemsets no frecuentes; en nuestro ejemplo no existe ningún itemset en* G_1 *en estas condiciones.*

G_1

A	8
B	6
C	8
D	4

$\xrightarrow{Produce}$

G_2

AB	6
AC	6
AD	2
BC	4
BD	1
CD	4

$\xrightarrow{Poda}$

G_2

AC	6
AD	2
BC	4

$\xrightarrow{Produce}$

G_3

ABC	4
ACD	2

$\xrightarrow{Poda}$

G_3

$\emptyset$

A continuación se produce G_2 *a partir de* G_1 *realizando la combinación entre itemsets y calculando el soporte para cada itemset generado. Luego se realiza la poda sobre* G_2*, donde se elimina el itemset* BD *por ser infrecuente y los itemsets* AB *y* CD *por el lema 3.3.3 ya que* $soporte(AB) = soporte(B)$ *y* $soporte(CD) = soporte(D)$*.*

Nuevamente, se genera a partir de G_2 *el nuevo conjunto de generadores candidatos* G_3 *y se procede a podar en este caso ambos itemsets, ya que existen subconjuntos de los mismos que no pertenecen a* G_2 *y por lo tanto mal generados. En particular para el itemset* ACD *podemos ver que en* G_2 *no existe el itemset* CD *y en forma similar el itemset* AB *para* ABC*.*

El conjunto de generadores minimales está compuesto por la unión de G_1, G_2 *y* G_3*. A los itemsets generadores entonces se le aplica el operador de clausura obteniendo los*

itemsets cerrados. Puede compararse dicho resultado con el obtenido en el ejemplo 3.3.8.

G

Generador	*Clausura*	*Soporte*
A	$\{A\}$	8
B	$\{AB\}$	6
C	$\{C\}$	8
D	$\{CD\}$	4
AC	$\{AC\}$	6
AD	$\{ACD\}$	2
BC	$\{ABC\}$	4

$\Diamond$

En resumen, este algoritmo provee un nuevo acercamiento para la representación concisa de patrones frecuentes utilizando el concepto matemático de FCA y realizando una analogía entre los conceptos y los patrones cerrados. Este concepto matemático ha sido utilizado en forma posterior por varios algoritmos constituyéndose en el marco teórico de la búsqueda de patrones cerrados. Por otro lado, este algoritmo utiliza un acercamiento `Apriori` al problema de generación de patrones cerrados por lo cual se generan muchos itemsets irrelevantes como puede haber quedado de manifiesto en el ejemplo. También se saca provecho de las características inherentes de los patrones cerrados al realizar podas de patrones irrelevantes. Dicha poda constituye una mejora importante en la performance del algoritmo con respecto a la utilización simplemente del algoritmo `Apriori` o similar. Sin embargo, a partir de información empírica se observó que en conjuntos de datos con un alto soporte y poca cantidad de patrones frecuentes `Apriori` supera en performance al algoritmo `A-Close` [86].

3.3.4. Charm

El algoritmo `ChARM` propuesto en [86] realiza una búsqueda en el espacio de itemsets en forma Depth-First utilizando como representación un SETree visto en la sección 2.6.2. Este algoritmo, al igual que `A-Close`, utiliza el concepto matemático de FCA para manipular los itemsets y realiza podas para reducir el espacio de búsqueda. `ChARM` permite explorar tanto el espacio de itemsets como el espacio de tidsets, permitiendo así seleccionar, de acuerdo a las características particulares de la base de datos, la forma más eficiente de

recorrer el mismo.

Utiliza para representar la base de datos una representación vertical, la cual facilita la representación de los conceptos o patrones cerrados. Debido a esto los soportes y conceptos se calculan realizando uniones de itemsets e intersecciones de tidsets. Al igual que `A-Close`, el algoritmo `ChARM` realiza dos tipos de poda *a*) de itemsets infrecuentes, *b*) de itemsets no cerrados. Sin embargo, `ChARM` extiende la poda de itemsets no cerrados con respecto a la poda realizada en `A-Close`. Este nuevo tipo de poda permite reducir significativamente el espacio de búsqueda y está basada en cuatro propiedades.

En primer lugar, recordemos que en un SETree los nodos hijos a un nodo dado se generan combinando dicho nodo con sus nodos hermanos mayores (los que se encuentran a la derecha del mismo). Supongamos que existe un orden total entre los itemsets definido por la relación $\leq$, es decir, para cualquier par de itemsets X_1 y X_2, $X_1 \leq X_2$ si y solo si X_1 se encuentra antes que X_2 según la función de orden total. Supongamos que X_2 es uno de los hermanos mayores de un nodo X_1 entonces $X_1 \leq X_2$. Sean las funciones de mapeo i y t según la definición 3.3.9 representado el conjunto de itemsets que existen en común en un conjunto de transacciones (tidset) y el conjunto de transacciones donde ocurre dicho itemset respectivamente. Sean X_1 y X_2 dos nodos hermanos tal que $X_1 \leq X_2$ entonces se establecen las siguientes propiedades al momento de explorar X_1[25]:

1. Si $t(X_1) = t(X_2)$ entonces $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_1) = t(X_2)$. Se poda el nodo X_2 y se reemplaza toda ocurrencia en el SETree del itemset X_1 por $X_1 \cup X_2$ ya que la clausura de X_1 y X_2 mapean al mismo itemset.

2. Si $t(X_1) \subset t(X_2)$ entonces $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_1) \neq t(X_2)$. Se reemplaza toda ocurrencia del itemset X_1 por $X_1 \cup X_2$. En este caso, se sabe que en todas las transacciones donde aparece X_1 también aparece X_2. Por tal motivo, se puede reemplazar X_1 por $X_1 \cup X_2$ ya que la clausura de X_1 genera el itemset X_2 entre otros. Sin embargo, el nodo X_2 no puede ser eliminado ya que generará una clausura diferente (ya que al menos existe una transacción donde ocurre X_2 y no ocurre X_1).

[25]Nótese que en este momento el nodo X_2 aún no ha sido explorado aunque si generado.

3. Si $t(X_1) \supset t(X_2)$ entonces $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_2) \neq t(X_1)$. En forma simétrica al caso anterior, se reemplaza toda ocurrencia del itemset X_2 por el itemset $X_1 \cup X_2$ y se mantiene el nodo X_1.

4. Si $t(X_1) \neq t(X_2)$ entonces $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) \neq t(X_1) = t(X_2)$. En este caso, los itemsets son incomparables y por lo tanto ambas ramas llevan a diferentes clausuras. No se realiza poda alguna.

Nótese que aún cuando las propiedades 2 y 3 parecen similares, en la propiedad 2 el nodo X_1 es reemplazado por un nuevo nodo $X_1 \cup X_2$ junto con toda ocurrencia del itemset X_1 en el SETree. Sin embargo, en la propiedad 3 genera un nuevo nodo de la combinación de X_1 y X_2 y luego reemplaza todas las ocurrencias del itemset X_2. Veamos el siguiente ejemplo que hemos utilizado para ilustrar el algoritmo `A-Close`.

Ejemplo 3.3.14

Sea la base de datos definida en el ejemplo 3.3.8 de la página 107. Supongamos tener un umbral de mínimo soporte de 2.

tid	*Itemsets*
1	CD
2	ACD
3	AB
4	AC
5	ABC
6	CD
7	AB
8	ABC
9	ABC
10	$ABCD$

Soportes

$A = 8,$	$AB = 6,$	$ABC = 4,$	$ABCD = 1$
$B = 6,$	$AC = 6,$	$ABD = 1$	
$C = 8,$	$AD = 2,$	$ACD = 2$	
$D = 4,$	$BC = 4,$	$BCD = 1$	
	$BD = 1,$		
	$CD = 4,$		

*Como primer paso se genera el conjunto de items (*1*-itemset) frecuentes, utilizaremos para representar el espacio de búsqueda un SETree con los itemsets ordenados en forma lexicográfica. Para cada uno de ellos, se realiza la combinación con sus nodos hermanos mayores para generar sus nodos hijos. Como* `ChARM` *realiza una exploración Depth-First, se analiza en primer lugar el nodo* $\{A\}$ *generando todo el subárbol correspondiente a ese nodo antes de comenzar a explorar el nodo* $\{B\}$ *y así sucesivamente.*

Obsérvese en la figura 3.9 que $t(\{A\}) \supset t(\{B\})$ *lo cual según la propiedad* 3 *nos dice que siempre que* $\{B\}$ *ocurre también ocurre* $\{A\}$*; esto nos lleva a generar el nodo* $\{A, B\}$ *y podar el nodo* $\{B\}$ *ya que está cubierto por este nuevo nodo generado. Para el caso de la combinación de los nodos* $\{A\}$ *y* $\{C\}$ *o* $\{A\}$ *y* $\{D\}$*, observamos que ambos pares de nodos son incomparables y por lo tanto se generan nuevos nodos y se mantienen los nodos* $\{C\}$ *y* $\{D\}$*. Como resultado de dichas combinaciones se observa luego que el nodo* $\{A, B, D\}$ *se descarta por ser infrecuente y el nodo* $\{A, D\}$ *se poda al realizar la exploración del nodo* $\{A, C\}$ *ya que* $t(\{A, C\}) \supset t(\{A, D\})$*. De forma similar, el nodo* $\{D\}$ *se poda al combinarlo con el nodo* $\{C\}$ $(t(C) \supset t(D))$*.*

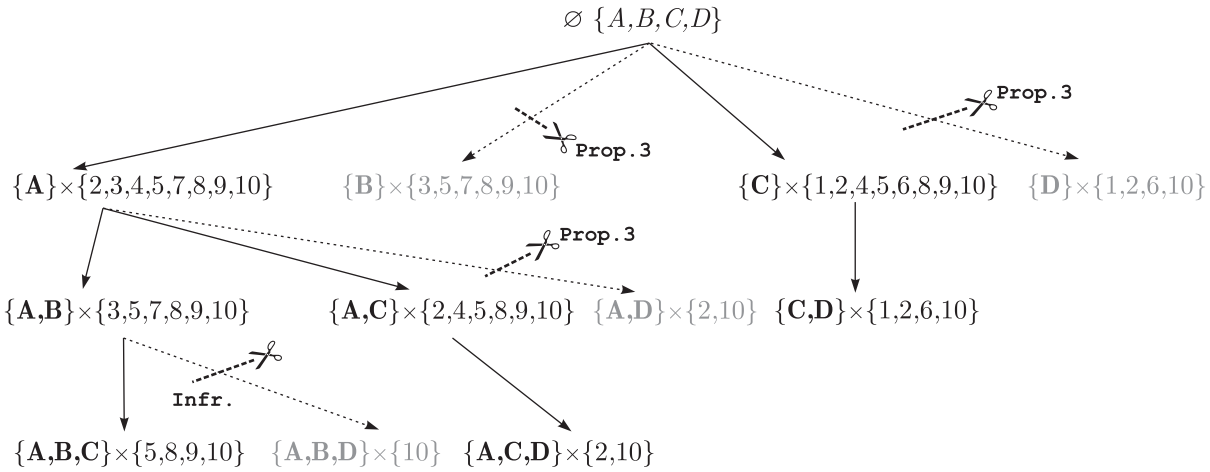


Figura 3.9: SETree generado por `ChARM` utilizando un orden lexicográfico.

El conjunto de nodos que posee el SETree (descartando los nodos candidatos podados) constituyen el conjunto de itemsets cerrados o conceptos frecuentes. Entonces este conjunto está constituido por $\{A, C, AB, AC, CD, ABC, ACD\}$*. Puede compararse dicho resultado con el obtenido por el algoritmo* `A-Close`*.* ◊

Como ha sido mencionado en la sección 3.2.1, la forma en que son ordenados los itemsets afecta la performance de las estrategias de poda. Por tal motivo, `ChARM` utiliza la técnica de reordenamiento dinámico, ordenando los itemsets según su soporte en forma creciente. De esta forma se logra que ocurran en mayor proporción podas según la

propiedad 2 en lugar de la propiedad 3 reduciendo aún más el espacio de búsqueda tal como ha sido mencionado.

Ejemplo 3.3.15

En la figura 3.10 puede verse el SETree resultante de aplicar un reordenamiento dinámico sobre el ejemplo anterior. Nótese que el espacio de búsqueda se ha reducido de 12 *nodos a* 9 *nodos. Además las combinaciones entre los pares de nodos* $(\{A\},\{C\})$ *y* $(\{C\},\{D\})$ *se transformaron en podas según la propiedad* 2 *reduciendo así el espacio de búsqueda al no generar nuevos nodos.* ◊

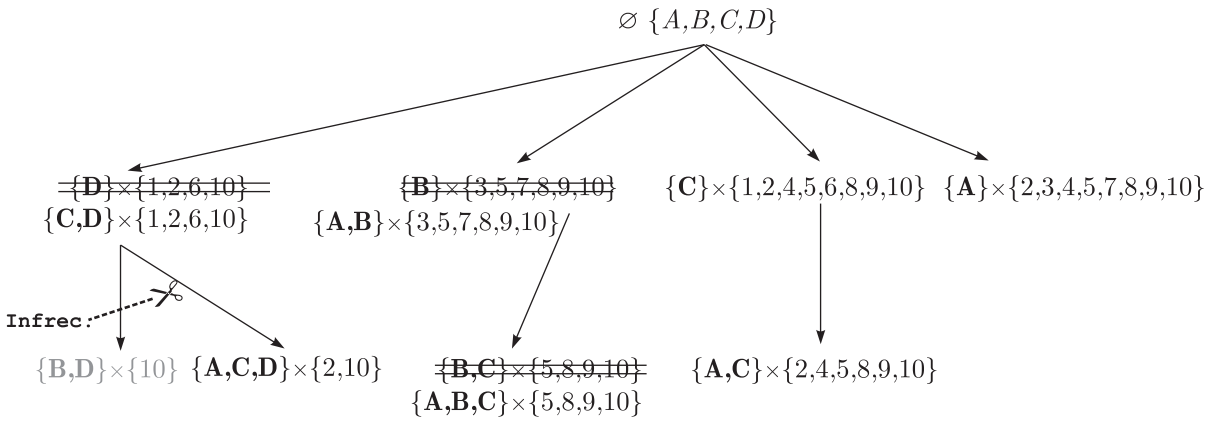


Figura 3.10: SETree generado por `ChARM` utilizando reordenamiento dinámico.

En resumen, el algoritmo `ChARM` utiliza una representación de la base de datos vertical sobre la cual realiza una exploración Depth-First. Utilizando algunas heurísticas de los itemsets cerrados logra realizar tres tipos de poda de itemset cerrados junto con la poda de itemsets infrecuentes. Utiliza la técnica de reordenamiento dinámico de itemsets lo que permite reducir aún más el espacio de búsqueda. Además, este algoritmo recorre en forma simultánea tanto el espacio de itemsets como el espacio de tidset, lo cual permite saltar varios niveles en la representación SETree del espacio de búsqueda.

3.3.5. Closet+

El algoritmo `Closet+`, propuesto en [80], incorpora nuevas estrategias de poda y capacidad para realizar nuevos tipos de recorridos al algoritmo `Closet` [56]. Este algoritmo incorpora tanto recorridos de arriba hacia abajo (Top-down) como el tradicional recorrido de abajo hacia arriba (Bottom-up) sobre la estructura `FPTree`. Con estas mejoras, `Closet+` se compara en performance con el algoritmo `ChARM` mencionado anteriormente.

`Closet+` utiliza la estructura FPTree[26] para representar el espacio de búsqueda y la base de datos, realizando proyecciones de la base de datos para construir la base de patrones condicionales en forma similar al algoritmo `FP-Growth`[27]. Sin embargo, a diferencia de `FP-Growth`, este algoritmo realiza podas que pueden eliminar algunos itemsets frecuentes. Dichos itemsets corresponden a itemsets que no pueden resultar en itemsets cerrados y por lo tanto se evita su procesamiento. Existen entonces tres tipos de poda que podemos identificar en `Closet+`, en primer lugar la poda de itemsets no frecuentes, en segundo lugar la poda de itemsets que son subconjuntos de algún itemset cerrado previamente encontrado y por último, sacando provecho de la estructura `FPTree`, se podan aquellos items que tengan el mismo soporte en al menos un par de tablas de cabecera generadas para diferentes niveles.

Los itemsets cerrados son mantenidos en una estructura `FPTree` global sobre la cual se realizan las comparaciones por subconjuntos cerrados. Para lograr una mayor eficiencia en la búsqueda de subconjuntos en esta estructura, `Closet+` incorpora dos estructuras Hash organizadas en dos niveles. La primer tabla hash utiliza como llave el último item del itemset a buscar y como resultado nos retorna un puntero a la tabla hash asociada a dicho item, esta segunda tabla hash se indexa por soporte y en el caso que múltiples itemsets terminen con el mismo item y el mismo soporte en dicha tabla se mantiene una lista enlazada. A partir de esta segunda tabla se puede acceder a la estructura `FPTree` de itemsets cerrados para verificar si existe alguno de estos FCI que sea superconjunto del itemset que se está analizando.

[26] Ver sección 2.4.3 en página 29.
[27] Ver sección 2.5.2 en página 39.

Como hemos mencionado en la sección 2.4.3, la estructura `FPTree` es una forma muy compacta de representar la base de datos llegando en algunos casos a representar la base de datos completa en memoria. Sin embargo, esta estructura es solamente eficiente si la base de datos es relativamente densa, ya que en caso contrario el `FPTree` junto con la tabla de cabeceras puede necesitar más espacio que la base de datos misma. Por tal motivo, `Closet+` incorpora dos tipos de recorridos de la estructura `FPTree`; uno de ellos es el tradicional recorrido bottom-up en el cual se generan las distintas bases de patrones condicionales y a partir de ellos se arman los `FPTree` condicionales. Este recorrido de abajo hacia arriba se utiliza en el caso de bases de datos densas logrando buenos resultados (según se reporta en [80] superando a `ChARM` en tiempo de ejecución y espacio utilizado). Por otro lado, para el caso de bases de datos ralas, se realiza un recorrido top-down del `FPTree` sin realizar proyecciones físicas de la base de datos. En su lugar se utiliza simplemente una estructura auxiliar realizando pseudo-proyecciones. De esta manera, se evita el costo asociado a la generación de los `FPTree` condicionales que en el caso de base de datos ralas usualmente *no* constituyen una mejora en espacio.

En resumen, `Closet+` realiza un recorrido depth-first utilizando una estructura `FPTree` para representar tanto el espacio de búsqueda como la base de datos. Para compensar su pobre performance sobre bases de datos ralas, como sucede en `Closet`, este algoritmo incorpora un esquema híbrido de recorridos, utilizando recorridos tanto top-down como bottom-up según corresponda. Al igual que `A-Close` y `ChARM`, este algoritmo realiza tanto podas de itemsets infrecuentes como podas de itemsets no cerrados.

3.3.6. FPclose

Este algoritmo ha sido propuesto en [32] junto con el algoritmo `FPmax*` para búsqueda de MFI. Ambos algoritmos reportan los mejores tiempos de ejecución sobre diferentes tipos de bases de datos. `FPclose` es similar a `FPmax*` en la estrategia utilizada, incorporando la estructura de arreglos para reducir la cantidad de recorridos necesarios sobre el `FPTree`. Al igual que el algoritmo `Closet+`, `FPclose` utiliza la estructura `FPTree` para representar el espacio de búsqueda y la base de datos. Sin embargo, `FPclose` no mantiene

una estructura `FPTree` global de itemsets cerrados, sino que utiliza la estrategia de focalización progresiva[28] generando múltiples `FPTree` condicionales de itemsets cerrados y así logrando reducir el número de comparaciones por subconjuntos.

Este algoritmo también reduce el espacio de búsqueda realizando podas de itemsets infrecuentes y podas de itemsets no cerrados, es decir, de aquellos itemsets que son un subconjunto de algún itemsets cerrado ya detectado. A diferencia de `Closet+`, este algoritmo realiza únicamente una exploración top-down en forma depth-first. Sin embargo, al utilizar el reordenamiento dinámico (al igual que `FPmax*`) y la focalización progresiva, `FPclose` obtiene una performance muy superior a `Closet+`.

En resumen, este algoritmo representa una de las mejores opciones para la búsqueda de FCI permitiendo administrar eficientemente la memoria con una estructura `FPTree` e incorporando técnicas de reordenamiento dinámico de items y estrategias de poda propuestas en diversos algoritmos.

3.3.7. Conclusiones

Hemos analizado en esta sección los fundamentos de los itemsets cerrados identificando las características propias de los mismos las cuales han sido explotadas por los diferentes algoritmos presentados para una búsqueda más eficiente de los mismos. En particular, hemos analizado los principales algoritmos del estado actual del arte propuestos para la búsqueda de FCI. Sin embargo, hemos dejado fuera de consideración algunos otros acercamientos al mismo problema por una cuestión de mantener acotado el espacio dedicado a dichos algoritmos (por ejemplo, otro algoritmo también propuesto es `Mafia`, algoritmo que hemos analizado en detalle en la sección 3.2.5 y el cual activando una de sus opciones permite computar los patrones cerrados).

3.4. Patrones Minimales No Frecuentes

Hasta el momento hemos visto que el conjunto de los itemsets maximales (MFI) es la forma más concisa de representar el conjunto de itemsets frecuentes (FI). Estos itemsets

[28] Ver sección 3.2.1 en página 75.

maximales, según la definición 2.1.10, son los patrones frecuentes más específicos de la base de datos. Existe otra representación alternativa, que tiene tanto poder de brevedad y está formada por los itemsets no frecuentes minimales o más generales. Tanto el conjunto de itemsets frecuentes más específicos (maximales) como el conjunto de itemsets no frecuentes más generales (minimales) forman un borde superior (en el caso de los itemsets maximales) e inferior (en el caso de los itemsets no frecuentes minimales) donde todos los itemsets por debajo (arriba) del borde superior (inferior) respetan la misma propiedad que el borde donde están contenidos. Es decir, los itemset cubiertos por un itemset maximal son frecuentes y todo itemset que cubre a un itemset minimal es no frecuente. Formalmente se definen dichos bordes de la siguiente forma:

Definición 3.4.1 (Bordes Positivos y Negativos, Toivonen)
Sea $\mathcal{P}$ un conjunto de patrones, $\preceq$ un orden parcial sobre $\mathcal{P}$ y $\mathcal{S} \subseteq \mathcal{P}$. Sea $\mathcal{S}$ cerrado por debajo bajo la relación $\preceq$, es decir, si $\varphi \in \mathcal{S}$ y $\gamma \preceq \varphi$ entonces $\gamma \in \mathcal{S}$. El borde $\mathcal{B}d(\mathcal{S})$ consiste de aquellos patrones φ tal que todos los patrones más generales que φ están en $\mathcal{S}$ y ningún patrón más específico que φ está en $\mathcal{S}$:

$$\mathcal{B}d(\mathcal{S}) = \{\varphi \in \mathcal{P} |\ \forall \gamma \in \mathcal{P} \text{ tal que } \gamma \prec \varphi \text{ entonces } \gamma \in \mathcal{S} \text{ y} \\ \forall \theta \in \mathcal{P} \text{ tal que } \varphi \prec \theta \text{ entonces } \theta \notin \mathcal{S}\}$$

Llamaremos borde positivo (positive border) $\mathcal{B}d^+(\mathcal{S})$ al conjunto de patrones $\varphi \in \mathcal{B}d(\mathcal{S})$ que pertenecen al conjunto $\mathcal{S}$:

$$\mathcal{B}d^+(\mathcal{S}) = \{\varphi \in \mathcal{P} |\ \forall \theta \in \mathcal{P} \text{ tal que } \varphi \prec \theta \text{ entonces } \theta \notin \mathcal{S}\}$$

y borde negativo (negative border) $\mathcal{B}d^-(\mathcal{S})$ al conjunto de patrones $\varphi \in \mathcal{B}d(\mathcal{S})$ que no pertenecen al conjunto $\mathcal{S}$:

$$\mathcal{B}d^-(\mathcal{S}) = \{\varphi \in \mathcal{P} \backslash \mathcal{S} |\ \forall \gamma \in \mathcal{P} \text{ tal que } \gamma \prec \varphi \text{ entonces } \gamma \in \mathcal{S}\}$$ ♦

El conjunto de patrones frecuentes puede ser representado en forma concisa tanto por el conjunto de patrones maximales como por el conjunto de patrones minimales no frecuentes, es decir, por el borde positivo como por el negativo.

Ejemplo 3.4.1

Consideremos nuevamente la figura 3.1 (página 61) donde se pueden apreciar el conjunto de patrones frecuentes y los patrones maximales o borde positivo. Por otro lado, en la figura 3.11 se muestran los patrones que pertenecen al borde negativo, el cuál está constituido por el siguiente conjunto de itemsets no frecuentes $\{\{A,C\},\{C,D\}\}$*. A partir de dicho borde negativo, pueden ser obtenidos todos los patrones no frecuentes ya que estos son superconjuntos de los elementos del borde negativo.* $\Diamond$

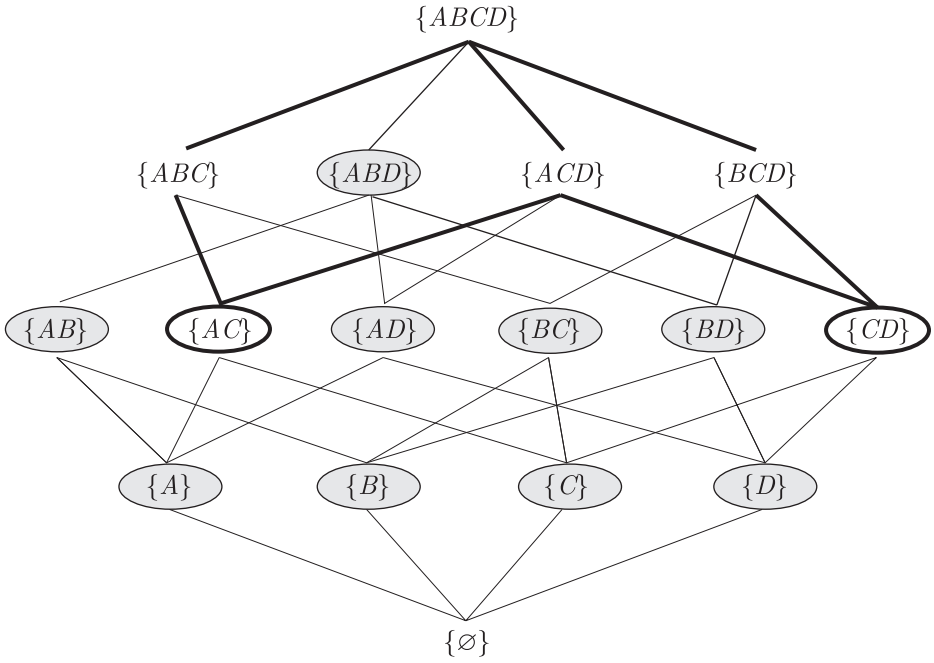


Figura 3.11: Reticulado Booleano de $\mathcal{P}(\{A,B,C,D\})$ con $\mathcal{B}d^{-}(\mathcal{S}) = \{\{A,C\},\{C,D\}\}$

Lo interesante de este borde negativo es que este conjunto usualmente es más reducido que el conjunto de patrones maximales. Sin embargo, obtener el borde negativo no es una tarea tan simple como sucede en su correspondiente positivo. Para computar el borde negativo se utiliza el borde positivo y se realiza un complemento del mismo. A partir de estos complementos se puede generar un hipergrafo para realizar una recorrido minimal y obtener así el borde negativo, este acercamiento es propuesto en [76].

Definición 3.4.2 (Hipergrafo Simple)

Sea R un conjunto. Una colección H de subconjuntos de R es un hipergrafo simple *sobre R, si ningún elemento de H es vacío y si $X, Y \in H$ y $X \subseteq Y$ implica $X = Y$. Los elementos de H son llamados arcos (o hiperarcos) del hipergrafo y los elementos de R son los vértices del hipergrafo.* ♦

Definición 3.4.3 (Recorrido)

Sea H un hipergrafo simple sobre un conjunto R. Se denomina recorrido (traversal) T de H a un subconjunto de R que intersecta todos los arcos de H. Es decir, T es una recorrido *si y solo si $T \cap X \neq \emptyset$ para todo $X \in H$. Una travesía T de H es una* recorrido minimal *si no existe $T' \subset T$ tal que T' sea una recorrido.* ♦

Estas nociones matemáticas de hipergrafos están íntimamente relacionadas a la clase de patrones frecuentes [51, 23, 52]. Sea R el conjunto de vértices y sea el conjunto de arcos del hipergrafo simple H el conjunto complemento de los itemsets del borde positivo, es decir, para cada X perteneciente al borde positivo existe un arco en H dado por $R \backslash X$. Consideremos un itemset $Y \subseteq R$:

a) Si existe un arco $R \backslash X$ tal que $Y \cap (R \backslash X) = \emptyset$ entonces $Y \subseteq X$ y por lo tanto Y es frecuente.

b) Si no existe un arco $R \backslash X$ tal que $Y \cap (R \backslash X) = \emptyset$ entonces Y no puede ser frecuente, ya que de serlo debería ser un subconjunto de algún elemento del borde positivo.

Entonces, Y no es frecuente si y solo si Y es una recorrido de H. Por lo tanto, el conjunto de travesías minimales es el conjunto de itemset minimales no frecuentes, es decir, el borde negativo.

3.5. Otras Representaciones

Además de estos acercamientos de representaciones concisas para los patrones frecuentes, existen otras alternativas no tan conocidas pero sin embargo importantes en

su aporte. Estas representaciones tienen un fuerte marco teórico y usualmente no son aplicadas debido a que aún no se cuenta con un mecanismo eficiente que la soporte.

3.5.1. Conjuntos Libres

El *conjunto de itemsets libres* (free-set), propuesto en [11, 12], representan el conjunto de itemsets que están formados por items con poca relación entre ellos, es decir, que ocurren pocas veces simultáneamente en la misma transacción. Este conjunto de itemsets libres es interesante ya que puede ser utilizado para aproximar el soporte de cualquier itemset, en particular, de los itemsets frecuentes. A partir de éstos itemsets libres se puede hallar una cota inferior y superior para el soporte de cualquier itemsets, con la particularidad que dicha cota es lo suficientemente cercana al valor real como para poder estimar con un mínimo grado de incertidumbre el error cometido por dicha estimación. Los *free-set* son definidos a partir de la concepto de reglas de asociación fuerte según un cierto δ.

Definición 3.5.1 (Regla δ-Fuerte, Boulicaut et al.)
Una regla δ-fuerte sobre una base de datos $\mathcal{D}$ se define como una regla de asociación $X \rightarrow Y$ tal que $soporte(X) - soporte(X \cup Y) \leq \delta$, es decir, la regla no es satisfecha en a lo sumo δ transacciones. ⧫

Definición 3.5.2 (δ-free-set, Boulicaut et al.)
Sea una base de datos $\mathcal{D}$ sobre el conjunto de items $\mathcal{I}$. El itemset $X \subseteq \mathcal{I}$ es un δ-free-set si y solo si no existe una regla δ-fuerte basada en X en $\mathcal{D}$. El conjunto de todos los δ-free-sets es notado $Free(\mathcal{D}, \delta)$. ⧫

Una particularidad de este conjunto de *free-sets* es que si catalogamos la pertenencia a este conjunto como una propiedad, entonces dicha propiedad es antimonótona. Es decir todo itemset subconjunto de un itemset libre también debe ser libre.

Teorema 3.5.1 (Boulicaut et al.)
Sea X un itemset. Para todo $Y \subseteq X$ si $X \in Free(\mathcal{D}, \delta)$ entonces $Y \in Free(\mathcal{D}, \delta)$. □

Esta propiedad permite que los free-set puedan ser generados utilizando una estrategia similar a `Apriori`, comenzando por los itemsets libres más pequeños y expandiendo solamente estos. También permite realizar podas sobre el espacio de búsqueda si se utiliza alguna otra estrategia.

Como mencionáramos éstos itemsets libres permiten aproximar el soporte de cualquier itemset, como se formaliza en el siguiente lema.

Lema 3.5.1

Sea una base de datos $\mathcal{D}$ sobre el conjunto de items $\mathcal{I}$. El itemset $X \subseteq \mathcal{I}$ y $\delta \in [0, |\mathcal{D}|]$ entonces existe un $Y \subseteq X$ tal que $Y \in Free(\mathcal{D}, \delta)$ y

$$soporte(Y) \geq soporte(X) \geq soporte(Y) - \delta|X| \qquad \square$$

Sin embargo, el conjunto de itemsets libres que pueden ser utilizado es usualmente intratable, por lo cual se propone utilizar los itemsets libres frecuentes, es decir

$$FreqFree(\mathcal{D}, \sigma, \delta) = Freq(\mathcal{D}, \sigma) \cap Free(\mathcal{D}, \delta)$$

que son representados en forma concisa utilizando el borde negativo[29] de los itemsets libres frecuentes.

Es interesante notar que este conjunto de *itemsets libres* esta relacionado con el concepto de *patrones cerrados*, ya que como se menciona en [11, 13], los patrones cerrados están íntimamente relacionados a los 0-free-sets, es decir a los free-sets con un $\delta = 0$. Por lo tanto, los free-sets son un superconjunto de los itemsets cerrados, sin embargo esto no implica que la cardinalidad de este conjunto es mayor, ya que como en el caso de los patrones maximales, estos pueden ser expresados muchas veces en forma más concisa que los patrones cerrados. Por otro lado, los itemsets libres tiene la particularidad de que pueden ser aplicados eficientemente en dominios que los cuales los patrones cerrados no son muy eficientes, como por ejemplo en conjuntos de datos con ruido. En este tipo de conjunto la utilización de itemsets cerrados puede llevar a generar muchos conceptos y en particular para cada ruido existente. Sin embargo, el uso de free-sets con un δ distinto de

[29] ver sección 3.4 página 122.

cero puede resultar en importantes reducciones del espacio de búsqueda, como se reporta en [11, 12].

Existe una extensión de los free-sets que son denominados *Conjuntos libres bajo disyunción* (Disjunction-free sets) los cuales son propuestos en [41]. Un itemset I es llamado *libre bajo disyunción* si no existen dos items $i_1, i_2 \in I$ tal que $soporte(I) = soporte(I - i_1) + soporte(I - i_2) - soporte(I - i_1, i_2)$. Nótese que los itemset libres son el caso especial cuando $i_1 = i_2$.

3.5.2. Itemsets No Derivables

Los *itemsets no derivables* (non-derivable itemsets o NDI) son una extensión del concepto de conjuntos libres bajo disyunción. Este conjunto de itemsets no derivables, propuesto en [15], permite representar en forma concisa el conjunto de itemset frecuentes utilizando solamente los itemsets que no pueden ser obtenidos por la utilización de ningún otro itemset. Por otro lado, si el soporte de un itemset puede ser calculado exactamente utilizando la información del soporte de otros itemsets entonces estos itemsets son denominados *derivables* (DI).

Mostraremos entonces como se puede derivar exactamente el soporte de un itemset a partir del soporte de un conjunto de itemsets (usualmente subconjuntos del mismo). Estas nociones se basan en la aproximación de soportes utilizando límites ya vista en la sección 3.2.1.

Supongamos un itemset $I \subseteq \mathcal{I}$ del cual desconocemos su soporte. En primer lugar, proyectaremos la base de datos utilizando solamente los items que aparecen en I generando una nueva base de datos con estos items.

Definición 3.5.3 (I-Proyección, Goethals et al., 1996)
Sea $I \subseteq \mathcal{I}$ un itemset.

- *La proyección de I sobre una transacción T, notada $\pi_I T$, se define como $\pi_I T = \{i \mid \ i \in T \cap I\}$.*
- *La proyección de I en una base de datos $\mathcal{D}$, notada $\pi_I \mathcal{D}$, consiste de todas las*

transacciones proyectadas de I. ♦

La cantidad de transacciones sobre la proyección de I en la base de datos $\mathcal{D}$ que contienen exactamente al itemset I se denomina la I-fracción.

Definición 3.5.4 (I-Fracción, Goethals et al., 1996)
Sea $\mathcal{I}$ el conjunto de items posibles y I, J itemsets tal que $I \subseteq J \subseteq \mathcal{I}$. La I-fracción de $\pi_J\mathcal{D}$, notada $f_I^J(\mathcal{D})$, es igual a la cantidad de transacciones en $\pi_J\mathcal{D}$ que consisten exactamente del conjunto I. ♦

Ejemplo 3.5.1
Consideremos la siguiente base de datos transaccional, un itemset $J = \{A, B, C\}$ del cual desconocemos su soporte y sea un itemset $I = \{A, C\} \subseteq J$. Realicemos la proyección de J sobre la base de datos.

tid	*Itemsets*
1	CD
2	ACD
3	AB
4	AC
5	ABC
6	CD
7	AB
8	ABC
9	ABC
10	$ABCD$

$\xrightarrow{\pi_J\mathcal{D}}$

tid	*Itemsets*
1	C
2	AC
3	AB
4	AC
5	ABC
6	C
7	AB
8	ABC
9	ABC
10	ABC

La I-fracción de $\pi_J\mathcal{D}$ está dada por la función $f_{\{A,C\}}^{\{A,B,C\}}(\mathcal{D}) = 2$, ya que solamente las transacciones 2 y 4 en la base de datos proyectada contienen exactamente al itemset $\{A, C\}$. ◊

Recordemos que el *cubrimiento* de un itemset (definición 2.1.5) consiste de todas las transacciones donde éste ocurre. Por otro lado, el *soporte* de un itemset (definición 2.1.6) se calcula obteniendo la cardinalidad del conjunto de transacciones que cubren al itemset, es decir, $soporte(I, \mathcal{D}) = \mid cubre(I, \mathcal{D}) \mid$.[30]

[30] Utilizaremos la notación reducida de cubre y soporte especificando la base de datos en forma implícita.

Supongamos $I, J \subseteq \mathcal{I}$, donde $\mathcal{I}$ es un conjunto de items y $J = I \cup \{A_1, A_2, \ldots, A_n\}$. Notar que $cubre(J) = \bigcap_{i=1}^{n} cubre(I \cup \{A_i\})$ y $\mid \bigcup_{i=1}^{n} cubre(I \cup \{A_i\}) \mid = \mid cubre(I) \mid - f_I^J$. Utilizando el principio de inclusión y exclusión de conjuntos[31] y la relación entre cubrimiento y soporte se obtienen las siguientes equivalencias:

$$\begin{aligned} \mid cubre(I) \mid - f_I^J = & \sum_{1 \le i \le n} \mid cubre(I \cup \{A_i\}) \mid - \sum_{1 \le i < j \le n} \mid cubre(I \cup \{A_i, A_j\}) \mid + \cdots \\ & + (-1)^n \sum_{1 \le i \le n} \mid cubre(J - \{A_i\}) \mid + (-1)^{n+1} \mid cubre(J) \mid \end{aligned}$$

$$\begin{aligned} (-1)^n soporte(J) - f_I^J = & \sum_{1 \le i \le n} soporte(I \cup \{A_i\}) - \sum_{1 \le i < j \le n} \mid soporte(I \cup \{A_i, A_j\}) \mid + \cdots \\ & + (-1)^n \sum_{1 \le i \le n} soporte(J - \{A_i\}) - soporte(I) \end{aligned}$$

Obsérvese que el $(-1)^{n+1} \mid cubre(J) \mid$ pasa al otro término con el mismo signo ($(-1)^n soporte(J)$) porque se decrementa el exponente. La parte derecha de esta última igualdad es notada como $\sigma(I, J)$ [15]. Este resultado es utilizado en el siguiente teorema, junto con la observación que f_I^J siempre es positivo, para establecer límites superiores e inferiores de soporte para un itemset J dado.

Teorema 3.5.2 (Goethals et al., 1996)

Para todos los itemsets $I, J \subseteq \mathcal{I}$, $\sigma(I, J)$ es un límite inferior (superior) del soporte de J si $\mid J - I \mid$ es par (impar). La diferencia $\mid soporte(J) - \sigma(I, J) \mid$ es igual a f_I^J. □

El valor de $\mid J - I \mid$ es igual a n, el cual representa la cantidad de elementos que ocurren en el itemset J pero no aparecen en el itemset I. Utilizando este teorema se puede aproximar o inclusive calcular exactamente el soporte de cualquier itemset derivable a partir del soporte de sus subconjuntos.

Ejemplo 3.5.2

Consideremos nuevamente la base de datos anterior sobre la cual hemos calculado el

[31] Ver definición **??** en página **??**.

soporte de todos los itemsets subconjuntos a $J = \{A, B, C\}$.

tid	*Itemsets*
1	CD
2	ACD
3	AB
4	AC
5	ABC
6	CD
7	AB
8	ABC
9	ABC
10	$ABCD$

Soportes

$s(A) = 9$
$s(B) = 6$
$s(C) = 8$
$s(AB) = 6$
$s(AC) = 6$
$s(BC) = 4$

I	n	*Límite*	$\sigma(I, J)$
A	2	*inf.*	$s(AB) + s(AC) - s(A) = 12 - 9 = 3$
B	2	*inf.*	$s(AB) + s(BC) - s(B) = 10 - 6 = \mathbf{4}$
C	2	*inf.*	$s(AC) + s(BC) - s(C) = 10 - 8 = 2$
AB	3	*sup.*	$s(AB) = 6$
AC	3	*sup.*	$s(AC) = 6$
BC	3	*sup.*	$s(BC) = \mathbf{4}$

Por lo tanto, $\sigma(\{B\}, J) \leq soporte(J) \leq \sigma(\{B, C\}, J)$ *que son los límites más restringidos. Por consiguiente,* $4 \leq soporte(J) \leq 4$ *y* $soporte(J) = 4$. $\Diamond$

De esta forma se puede calcular el soporte de un itemset sin realizar ninguna pasada sobre la base de datos, sino simplemente conociendo los valores de los itemsets subconjuntos al mismo. Este conjunto de itemsets constituyen los itemsets derivables. Por otro lado, aquellos itemsets que no pueden ser derivados porque no se cuenta con suficiente información, son los itemsets no derivables. En [15] se utiliza este conjunto de itemsets no derivables (NDI) para representar en forma concisa el conjunto de patrones frecuentes. Allí también se demuestra que los itemsets derivables se obtienen a partir de los NDI y por lo tanto no necesitan ser enumerados en la representación.

3.6. Conclusiones

En este capítulo hemos analizado distintas alternativas para representar en forma concisa el conjunto de patrones frecuentes. La principal ventaja de estas representaciones es que permiten recuperar en forma completa el conjunto de patrones frecuentes. Como ha sido mencionado, en muchas aplicaciones la enumeración de los patrones frecuentes es una tarea inviable ya que la cantidad de los mismos crecen en una forma exponencial con respecto a la cantidad de items. Por otro lado, el crecimiento de la cardinalidad de las representaciones analizadas en este capítulo, a diferencia de los patrones frecuentes, no

están vinculadas a la cantidad de items sino a la cantidad de patrones frecuentes. Por tal motivo, en algunas de las representaciones se podía observar que cuantos más patrones frecuentes había, más concisa resultaba la representación. En particular, hemos analizado en detalle dos tipos de representaciones concisas de patrones frecuentes. En primer lugar, la representación utilizando patrones maximales. En segundo lugar, la representación utilizando patrones cerrados. En un tercer lugar, hemos visto una representación complementaria a los patrones maximales la cual consiste de representar el borde negativo. Por último hemos analizado, algunas representaciones alternativas que están basadas en eliminar redundancia.

Los patrones maximales, junto con la representación por borde negativo, constituyen la forma más compacta de representar los patrones frecuentes. Sin embargo, esta ventaja es opacada por el hecho que solamente se tiene un límite inferior de los soportes de los patrones frecuentes, es decir que, para obtener el conjunto de patrones frecuentes con sus soportes es necesario realizar una nueva pasada sobre el conjunto de datos. Por otro lado, la representación utilizando patrones cerrados tiene la ventaja que para todo patrón frecuente se puede obtener su soporte sin necesidad de realizar una pasada sobre el conjunto de datos. Sin embargo, el costo de esta información resulta en una representación menos concisa que los patrones maximales.

Capítulo 4

Patrones Emergentes

En este capítulo introduciremos la noción de patrones emergentes (EPs). Los patrones emergentes intentan capturar los aspectos más significativos de cada clase de datos. Esta propiedad los hace muy interesantes para aplicaciones como clasificación. Sin embargo, los EPs no se limitan a dichas aplicaciones solamente sino que también han sido aplicados exitosamente en la detección de tendencias en bases de datos temporales. Actualmente están siendo aplicados experimentalmente en la detección de intrusos, ya que como se verá en este capítulo, los EPs ayudan a detectar patrones significativos que posean un soporte bajo o medio los cuales no son detectados por otras herramientas de minería de datos.

La investigación sobre patrones emergentes tiene su origen en 1998 [20]. Diferentes ramas de investigación han surgido como resultado de dicho trabajo, en particular se han utilizado EPs para *a*) la construcción de clasificadores, *b*) el descubrimiento de *patrones especializados* (niche patterns) y *c*) para el análisis de datos biológicos como secuencias de ADN y expresiones genéticas, entre otros. Este capítulo está basado en dichos trabajos e intenta capturar el estado actual de los patrones emergentes.

4.1. Introducción

Dentro del ámbito de la minería de datos la tarea fundamental de toda herramienta es obtener patrones potencialmente útiles, novedosos y válidos en función de los datos.

Dependiendo de dónde fueron recolectados los datos (dominio) y cuál es el propósito

de su utilización, no es infrecuente que los datos se encuentren divididos en conjuntos. Por ejemplo, una empresa de venta al público compila la información relacionada a las compras efectuadas por sus clientes en forma anual. A partir de dicha información se podría identificar tendencias de compra de los clientes realizando una comparación entre los diferentes conjuntos de datos.

En otros dominios existen dos categorías de datos *a*) Datos pre-clasificados (datos de entrenamiento) y *b*) Datos sin clasificar. Los datos clasificados se encuentran divididos en *clases* y a partir de las características de los elementos que componen cada clase se podrá determinar a qué clase corresponderá una instancia perteneciente a los datos sin clasificar. Por ejemplo, consideremos una escuela de entrenamiento canino en la cual se han registrado las características de los canes como su raza, visión, altura, peso, orden de nacimiento, etc. junto con la información si pasó o no un cierto test de entrenamiento. Una nueva instancia corresponderá con un nuevo prospecto para la escuela, la cual podrá evaluar si aceptar o no al can en función de la clasificación.

Los patrones emergentes son una herramienta muy versátil, la cual puede ser aplicada para múltiples propósitos y, en particular, han demostrado ser muy eficientes tanto en clasificación como en el descubrimiento de tendencias superando en exactitud, eficiencia y completitud a las herramientas existentes en este momento.

En forma general, los patrones emergentes (EPs) son aquellos itemsets cuyo soporte se incrementa significativamente de un conjunto de datos $\mathcal{D}_1$ a otro $\mathcal{D}_2$. Este incremento se calcula como la relación (ratio) entre los soportes de un itemset en cada conjunto de datos.

Un ejemplo de EP es el siguiente hecho [43] "*La incidencia de cáncer de pulmón entre fumadores es* 14 *veces mayor que entre los no fumadores*". Aquí se pueden apreciar que existen dos clases de instancias "*fumadores*" y "*no fumadores*" y el patrón emergente de la clase en cuestión "*tiene cáncer de pulmón*".

4.2. Motivaciones

Aún cuando existen muchas herramientas de probada eficacia y eficiencia para clasificación y detección de tendencias, existen varias razones que motivan la utilización de EPs como ha sido remarcado en [43].

1. Aún cuando las reglas de clasificación inducidas por otros métodos puedan convertirse en EPs, estas son solo un subconjunto de los EPs, por lo que importantes EPs quedan sin ser detectados afectando la exactitud del método.

2. Una de las principales ventajas de la búsqueda de EPs es que se pueden obtener patrones significativos que poseen un soporte relativamente bajo. En la sección de ejemplos se podrá apreciar la utilidad de dichos patrones, ya que en la mayor parte de la literatura se intenta capturar aquellos patrones que poseen un alto grado de frecuencia. Sin embargo, en ciertas ocasiones, los patrones más útiles son aquellos con un bajo soporte. Una situación característica se presenta cuando se intenta detectar comportamiento anómalo, por ejemplo en la detección de intrusos, detección de fraudes, detección temprana de eventos o cambios ambientales.

   Los patrones emergentes - a diferencia de otros métodos - intentan capturar aquellos patrones con un bajo soporte que son realmente significativos según el dominio. Otros métodos no son tan eficientes ya que simplemente enumeran todos los patrones que superan cierto umbral, para lo cual existen una enorme cantidad de patrones que hacen inviable la tarea del método y aún cuando esta tarea tenga éxito tales patrones carecen de significado para el experto o son demasiados para poder considerarlos.

3. Otras técnicas, como por ejemplo los estudios estadísticos tradicionales, solamente se basan en unas pocas variables por vez. Los patrones emergentes por su parte, capturan la interacción entre *múltiples* items. Esta relaciones difíciles de encontrar por otros métodos pueden suministrar nueva información. En [43] se puede observar un caso real extraído de un conjunto de datos de tumor de colon que contiene dos clases, 22 muestras de tejidos normales y 40 muestras de tejidos clasificados con

tumor. Para dicho conjunto de datos han sido recolectados 2000 atributos. Como resultado de la tarea de minería se ha podido observar que varios EPs de muchos items tienen un soporte mayor que EPs de pocos items. Por ejemplo, uno de los EPs de dos items tenía un soporte del $40,91\,\%$ mientras que un EP de ocho items tenía un soporte del $77,27\,\%$. Además como se mencionará en próximas secciones, los EPs pierden la propiedad de antimonotonía[1] y como consecuencia se pueden encontrar EPs aún cuando sus subconjuntos no lo sean.

4.3. Conceptos Fundacionales

Como ha sido mencionado el concepto de patrones emergentes ha sido propuesto por [43]. Las siguientes definiciones aparecen en dicho trabajo y como también en [42, 46, 20, 49, 48, 45, 47, 21, 4, 44]. Asumamos que existen dos conjuntos de datos sobre los cuales obtendremos los patrones emergentes, los cuales notaremos como $\mathcal{D}_1$ y $\mathcal{D}_2$. Recordemos que el soporte de un itemset X sobre un conjunto de datos $\mathcal{D}$, notado como $supp_{\mathcal{D}}(X)$, es calculado como la cantidad de transacciones en $\mathcal{D}$ que contienen a X, notado $count_{\mathcal{D}}(X)$, sobre la cantidad de transacciones que contiene $\mathcal{D}$, notado $\mid \mathcal{D} \mid$. Si no existen ambigüedades escribiremos simplemente $supp_i(X)$ en lugar de $supp_{\mathcal{D}_i}(X)$.

Definición 4.3.1 (Tasa de Crecimiento, Dong & Li, 1999)
La tasa de crecimiento (growth rate) de un conjunto de datos $\mathcal{D}_1$ a otro $\mathcal{D}_2$ de un itemset X se define por la siguiente función:

$$GrowthRate_{\mathcal{D}_1 \to \mathcal{D}_2}(X) = \begin{cases} 0 & Si\ supp_1(X) = 0\ y\ supp_2(X) = 0 \\ \infty & Si\ supp_1(X) = 0\ y\ supp_2(X) \neq 0 \\ \frac{supp_2(X)}{supp_1(X)} & en\ caso\ contrario \end{cases}$$ ♦

Definición 4.3.2 (Patrón Emergente, Dong & Li, 1999)
Sea ρ un umbral (threshold) para la tasa de crecimiento tal que $\rho > 1$. Un itemset X es llamado $\rho-$patrón emergente (emerging pattern) $\rho-EP$ o simplemente EP de $\mathcal{D}_1$ a $\mathcal{D}_2$ si la tasa de crecimiento de X es mayor o igual al umbral ρ ($GrowthRate_{\mathcal{D}_1 \to \mathcal{D}_2}(X) \geq \rho$).♦

[1] Ver propiedad 2.5.1 en página 37.

En dominios temporales (es decir cuando los datos se encuentran ordenados por tiempo) el conjunto de datos $\mathcal{D}_1$ es llamado *soporte* (background) y $\mathcal{D}_2$ es el conjunto de datos *destino* (target). En otros dominios donde los datos están relacionados por clases (por ejemplo la clase venenoso y la clase comestible de la base de datos de hongos [43]), el conjunto de datos $\mathcal{D}_1$ es llamada clase *negativa* (negative) y $\mathcal{D}_2$ es llamada clase *positiva* (positive).

Hemos visto que además de analizar a los patrones a partir de su soporte, usualmente también son analizados según su confianza[2]. Los patrones emergentes también pueden ser analizados según dicho parámetro, el cual se encuentra directamente relacionado con la tasa de crecimiento como se puede observar en las siguientes igualdades:

$$\begin{aligned} Confianza(X) =& \frac{supp_2(X)}{supp_2(X) + supp_1(X)} = \frac{\frac{supp_2(X)}{supp_1(X)}}{\frac{supp_2(X)+supp_1(X)}{supp_1(X)}} = \frac{\frac{supp_2(X)}{supp_1(X)}}{\frac{supp_2(X)}{supp_1(X)} + \frac{supp_1(X)}{supp_1(X)}} \\ =& \frac{GrowthRate_{\mathcal{D}_1 \to \mathcal{D}_2}(X)}{GrowthRate_{\mathcal{D}_1 \to \mathcal{D}_2}(X) + 1} \end{aligned}$$

4.4. Ejemplos y Aplicaciones

A continuación presentaremos una serie de ejemplos típicos previamente reportados en [43, 42]. Algunos de estos ejemplos fueron obtenidos sobre bases de datos reales[3], otros sobre bases de datos sintetizadas y otros simplemente tienen como objetivo demostrar la potencial aplicación de los patrones emergentes por lo que no están basados en pruebas reales. En dichos trabajos se presentan aún más ejemplos los cuales no han sido incluidos por razones de espacio y por ser demasiado específicos usualmente relacionados a expresiones genéticas para la detección de cancer y leucemia.

Ejemplo 4.4.1

Utilizando el conjunto de datos proporcionado por UCI Machine Learning Repository, se han realizado búsquedas de EPs utilizando un umbral de 2,5 *[42, pág. 38]. Entre los millones de EPs encontrados, se ha podido observar que aquellos patrones con una gran tasa*

[2]Ver definición 2.2.3 página 16.

[3]Ver información sobre repositorios en la sección 2.4.4 página 33.

	Soporte en Clase		
EP	Venenoso	Comestible	Tasa de Crec.
X	0 %	63.9 %	∞
Y	81.4 %	3.8 %	21.4

$\Diamond$

X = {(Olor = ninguno), (Tamaño de Lámina = Ancha), (Nro. de Anillos = Uno)}
Y = {(Moretones = no), (Espaciado de Lámina = Angosta), (Color del Velo = Blanco)}

Cuadro 4.1: Dos EPs característicos del conjunto de datos de Hongos

de crecimiento de una clase a otra, tienen características evidentes que permiten diferenciar hongos venenosos de aquellos comestibles. En el cuadro 4.4.1 se pueden apreciar dos de los EPs típicos, uno de cada clase.

Es interesante resaltar que por ejemplo algunos subconjuntos de X, { Tamaño de Lámina = Ancha } y { Nro. de Anillos = Uno } no calificaron como EPs. Esta característica tiene consecuencias muy fuertes sobre la forma en que los EPs pueden ser obtenidos ya que los métodos tradicionales basan principalmente su eficiencia en la explotación de dicha propiedad.

Hemos mencionado previamente que los patrones emergentes pueden ser utilizados para detectar *tendencias*. A grandes rasgos, las tendencias son variaciones constantes entre conjuntos de datos. Usualmente las tendencias más interesantes son aquellas que tienen una tasa de crecimiento considerable pero cuyo soporte en cada clase es bajo. Podemos observar que dichos patrones son difícil de identificar por métodos tradicionales ya que existen una enorme cantidad de patrones con un soporte bajo y para cada uno habría que comparar su soporte en cada conjunto de datos. El siguiente ejemplo muestra la importancia de dichos patrones.

Ejemplo 4.4.2

En un artículo de Dayton Daily News el 6/10/2002 *apareció el siguiente párrafo "Los bajos aranceles y altos standards atraen a estudiantes de Estados Unidos a Canadá" identificando una tendencia emergente de estudiantes estadounidenses que se inscriben para cursar en universidades canadienses. En dicho artículo se hace mención que existe un*

total de aproximadamente 5000 *alumnos estadounidenses inscriptos y que se ha producido un incremento del* 85 % *durante los tres últimos años.*

*Claramente se puede apreciar que el soporte de los alumnos estadounidenses inscriptos (*5000*) con respecto al universo de estudiantes inscriptos en universidades canadienses (*N*) es bajo (*$5000/N$*).* ◊

En el siguiente ejemplo quedará de manifiesto como EPs con un soporte bajo puede proveer nueva información sobre dominios que se creían "bien" comprendidos. Usualmente, este tipo de situación sucede porque los patrones que se conocen sobre las aplicaciones poseen pocas variables mientras que los EPs tienen la capacidad de detectar patrones más largos y significativos.

Ejemplo 4.4.3

Supongamos que hubieron 1000 *compras del siguiente patrón* { *Computadora, Modem, Software Educativo* } *en* 1985 *sobre un total de* 20 *millones de transacciones y en* 1986 *hubieron* 2100 *compras del mismo patrón sobre un total de* 21 *millones de transacciones. Puede observarse que la tasa de crecimiento del año* 85 *al* 86 *ha sido del* 100 %(2)*, ya que el soporte de tal patrón en* 1985 *es del* 5 % *y en* 1986 *es del* 10 %*.*

Tal patrón es una oportunidad interesante para realizar alguna inversión o publicidad, aún cuando puede apreciarse que sobre el total de ventas su frecuencia es muy baja. ◊

Por último consideremos un ejemplo donde se combinan los conceptos hasta aquí mencionados. En este ejemplo se podrá apreciar la utilidad de patrones con bajo soporte en la clasificación sobre un dominio real como es la asistencia en decisiones médicas.

Ejemplo 4.4.4

Consideremos una aplicación acerca de pacientes con alguna enfermedad difícil de curar, donde uno de los conjuntos de datos contiene el registro de los pacientes que han sido tratados exitosamente y otro conjunto de datos de aquellos que no. Un patrón emergente hipotético muy útil podría ser $\{S_1, S_2, T_1, T_2, T_3\}$ *con una tasa de crecimiento de* 9 *del conjunto de datos de no curados a curados. Dicho de otra manera, de entre todos los*

pacientes con síntomas S_1 y S_2 y que han recibido como tratamiento T_1, T_2 y T_3, el número de pacientes curados es 9 veces mayor al número de pacientes no curados. Esto podría sugerir que una cierta combinación de tratamientos debería ser aplicada cuando una cierta combinación de síntomas ocurre. Aún cuando el EP pudiera tener un soporte bajo, por ejemplo un 1 %, dicho patrón constituye nuevo conocimiento al campo de la medicina, el cual podrá ser muy útil si no existe otra mejor sugerencia. Estos patrones podrían aún contradecir conocimiento previo en referencia a cierto tratamiento sobre cada síntoma. Un conjunto seleccionado de EPs pueden ser una gran guía al momento de decidir que tratamiento deberá ser administrado. ◊

4.5. Representación Concisa de Patrones Emergentes

Como ha quedado evidenciado en los ejemplos previos el conjunto de EPs es potencialmente grande. El cálculo de dichos patrones por medio de la enumeración exhaustiva es una tarea que lo convierte en inviable aún para los dominios más pequeños y utilizando umbrales grandes. Sin embargo, como se demostrará posteriormente estos patrones poseen una propiedad que permite representarlos en forma concisa. Explotando esta propiedad el espacio de EPs puede ser representado simplemente utilizando el conjunto de los elementos más específicos y el conjunto de los elementos más generales. Esta propiedad ha sido propuesta en [34] y luego adaptada en [43] para espacios de EPs. Más información puede encontrarse en [34] donde podrá observarse su aplicación en Version Spaces y ATMS.

4.5.1. Conjuntos Cerrados y Anticadenas

Definiremos a continuación la noción de conjuntos cerrados por debajo y por arriba utilizando el operador relacional $\preceq$ definido como $\subseteq$. Una explicación más detallada de dichos conceptos puede encontrarse en [34].

Definición 4.5.1 (Conjuntos Cerrados, Gunter et al., 1997)
Un conjunto $\mathcal{S} \subseteq \mathcal{P}$ se dice cerrado por debajo (downward closed) si $x \in \mathcal{S}$ y $y \preceq x$ implica

que $y \in \mathcal{S}$. Esto será también notado como:

$$\downarrow \mathcal{S} = \{x \in \mathcal{P} \mid x \preceq y \text{ para algún } y \in \mathcal{S}\}.$$

En forma dual un conjunto $\mathcal{S} \subseteq \mathcal{P}$ se dice cerrado por arriba (upward closed) si

$$\mathcal{S} = \uparrow \mathcal{S} = \{x \in \mathcal{P} \mid y \preceq x \text{ para algún } y \in \mathcal{S}\}. \quad \blacklozenge$$

Ejemplo 4.5.1

Supongamos tener el conjunto de atributos $\mathcal{A} = \{ABCD\}$[4] y sea $\mathcal{P} = PowerSet(\mathcal{A})$. Utilizaremos el símbolo $\preceq$ como operador relacional según la definición 2.6.1. El reticulado resultante para dicho poset se puede observar en la figura 4.1.

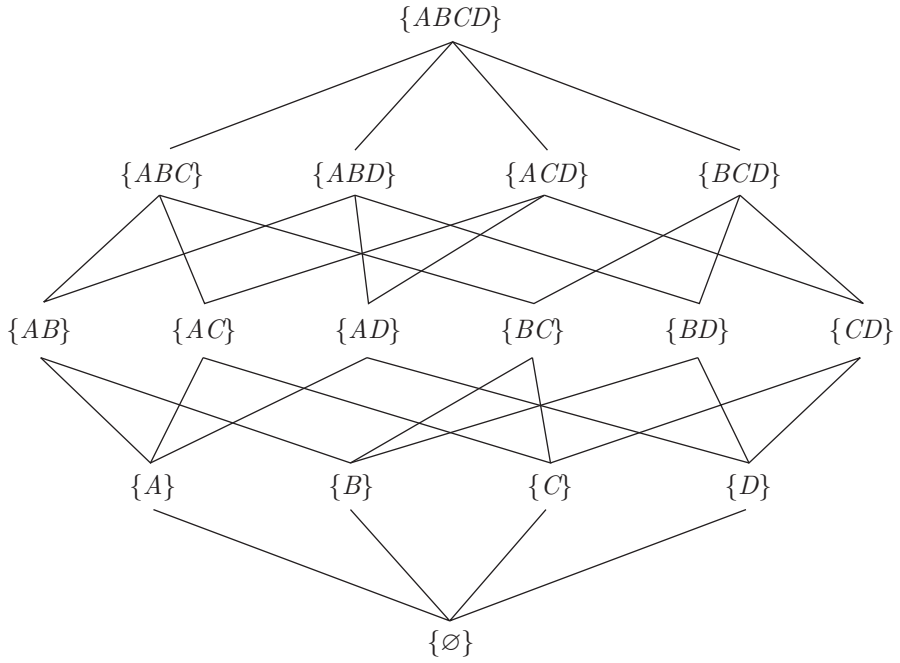


Figura 4.1: Poset para el conjunto de partes de $\{A, B, C, D\}$.

Supongamos un conjunto $\mathcal{S}_1 \subset \mathcal{P}$ tal que $\mathcal{S}_1 = \{\{\emptyset\}, \{A\}, \{B\}, \{D\}, \{BD\}\}$. Entonces

$$\downarrow \mathcal{S}_1 = \mathcal{S}_1 = \{\{\emptyset\}, \{A\}, \{B\}, \{D\}, \{BD\}\}$$

[4]Si no hay lugar a ambigüedad preferiremos esta notación abreviada, en lugar de $\{A, B, C, D\}$.

Por lo tanto $\mathcal{S}_1$ es cerrado por debajo, sin embargo no es cerrado por arriba[5] *ya que*

$$\uparrow \mathcal{S}_1 = \mathcal{P} \neq \mathcal{S}_1$$

Supongamos otro conjunto $\mathcal{S}_2 \subset \mathcal{P}$ tal que $\mathcal{S}_2$ contiene todos los elementos que contienen el atributo A o ambos atributos C y D,

$$\mathcal{S}_2 = \left\{ \begin{array}{c} \{ABCD\}, \{ABC\}, \{ACD\}, \{ABD\}, \\ \{BCD\}, \{AB\}, \{AC\}, \{AD\}, \{CD\}, \{A\} \end{array} \right\}$$

Fácilmente puede verificarse que $\mathcal{S}_2$ es cerrado por arriba o que $\mathcal{S}_2 = \uparrow \mathcal{S}_2$, sin embargo, podemos ver que no es cerrado por debajo. En efecto, se tiene

$$\downarrow \mathcal{S}_2 = \{\{\emptyset\}, \{A\}, \{B\}, \{C\}, \{D\}, \{BC\}, \{BD\}\} \cup \mathcal{S}_2 \neq \mathcal{S}_2 \qquad \Diamond$$

Es interesante tener conjuntos que sean cerrados por arriba (abajo) ya que solo debemos mantener los elementos más pequeños (grandes) para representar todo el conjunto.

Ejemplo 4.5.2

Tomemos los conjuntos $\mathcal{S}_1$ y $\mathcal{S}_2$ utilizados en el ejemplo 4.5.1. Sea $\mathcal{S}_1'$ el conjunto de elementos maximales de $\mathcal{S}_1$ y sea $\mathcal{S}_2'$ el conjunto de elementos minimales de $\mathcal{S}_2$ tal que $\mathcal{S}_1' = \{\{A\}, \{BD\}\}$ y $\mathcal{S}_2' = \{\{A\}, \{CD\}\}$. El conjunto $\mathcal{S}_1'$ puede verse como la frontera (boundary) superior de $\mathcal{S}_1$ y $\mathcal{S}_2'$ como la frontera inferior de $\mathcal{S}_2$. En la figura 4.2 pueden apreciarse dichos conjuntos. $\Diamond$

Definición 4.5.2 (Conjuntos Maximales y Minimales, Gunter et al., 1997)

Sea $x \in \mathcal{S}$ tal que $\mathcal{S} \subseteq \mathcal{P}$. Diremos que x es maximal[6] *(minimal) en $\mathcal{S}$ si para todo $y \in \mathcal{S}$ tal que $x \preceq y$ ($y \preceq x$) implica que $x = y$. Denotaremos como $max(\mathcal{S})$ y $min(\mathcal{S})$ al conjunto de elementos maximales y minimales respectivamente.* $\blacklozenge$

[5] Es importante notar que dado que $\emptyset \subseteq \mathcal{S}_1$ entonces $\uparrow \mathcal{S}_1$ genera todo el conjunto $\mathcal{P}$.

[6] En la sección 2.1.11 ya habíamos definido la noción de itemset minimal y maximal la cual es muy similar. Sin embargo, se desea mostrar como se definen dichos conjuntos de itemsets en el contexto de los reticulados de subconjuntos.

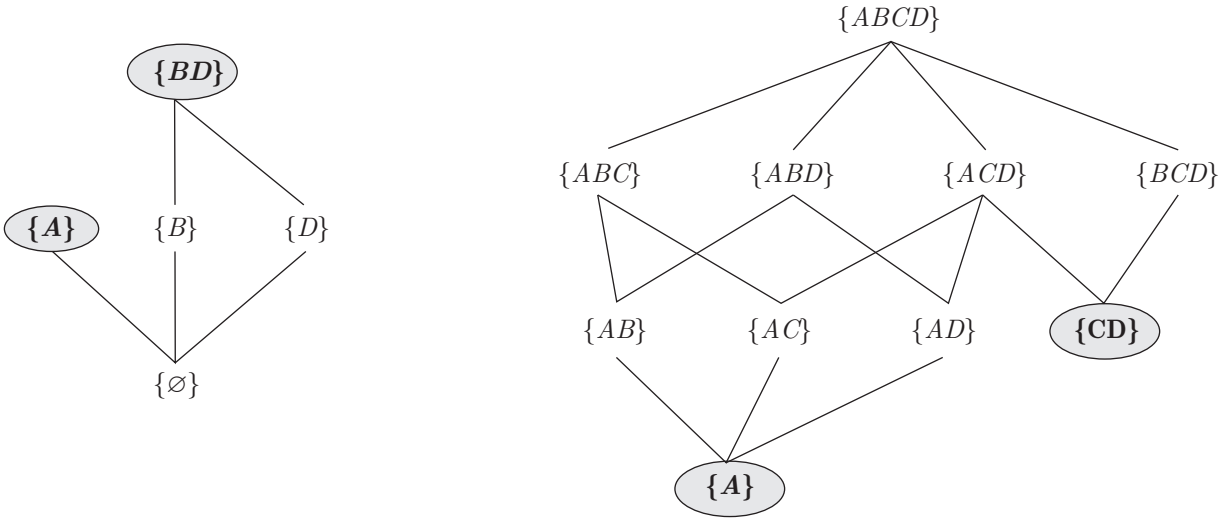


Figura 4.2: Elementos minimales y maximales del poset del Powerset($\{A, B, C, D\}$)

Como hemos visto en el ejemplo estos conjuntos maximales y minimales permitirán representar en forma concisa ya sea un conjunto cerrado por arriba o por debajo como se observa en el siguiente lema.

Lema 4.5.1 (Gunter et al., 1997)

Sea $\mathcal{P}$ un poset y supongamos que $\mathcal{S}'$ es un subconjunto finito[7] *de $\mathcal{P}$. Entonces*

1. $\downarrow \mathcal{S}' = \downarrow max(\mathcal{S}')$
2. $\uparrow \mathcal{S}' = \uparrow min(\mathcal{S}')$ □

Una lectura superficial de este lema nos dice que utilizar a un conjunto como límite o frontera superior (inferior) tiene el mismo efecto que utilizar el conjunto de sus elementos maximales (minimales) como límite o frontera superior (inferior).

Para terminar esta sección introduciremos una propiedad que se utiliza para determinar que subconjuntos de un poset pueden ser utilizados como fronteras para sus subconjuntos cerrados por arriba o por debajo. Nótese la similitud con la definición de conjuntos maximales y minimales.

[7]Consideramos solo conjuntos finitos, ya que sino no se puede asegurar la existencia de elementos maximales y minimales.

Definición 4.5.3 (Anticadenas, Gunter et al., 1997)
Sea $< \mathcal{P}, \preceq >$ un poset y $\mathcal{S} \subseteq \mathcal{P}$. Diremos que $\mathcal{S}$ es una anticadena (anti-chain) si solo contiene elementos no comparables distintos (si $x, y \in \mathcal{S}$ y $x \preceq y$ entonces $x = y$). ♦

El concepto de anticadenas es de gran importancia ya que ha permitido definir operaciones entre anticadenas que representan espacios cerrados y obtener nuevos espacios cerrados. Entre esas operaciones se puede encontrar diferencia, unión e intersección homogénea y heterogénea.

4.5.2. Espacios Convexos

Es de especial interés una clase de subconjunto de poset llamado *espacio convexo*. De la misma forma que los conjuntos cerrados permitían expresar en forma concisa un gran número de patrones, los espacios convexos generalizan aún más este concepto.

Definición 4.5.4 (Espacio Convexo, Gunter et al., 1997)
Sea $< \mathcal{P}, \preceq >$ un poset. Un subconjunto $\mathcal{C} \subseteq \mathcal{P}$ es llamado espacio convexo (convex space) si para cada $x, y, z \in \mathcal{P}$, las condiciones $x \preceq y \preceq z$ y $x, z \in \mathcal{C}$ implican que $y \in \mathcal{C}$. ♦

A partir de la definición podemos observar que los espacios convexos pueden ser representados utilizando la frontera inferior (x) y la superior (z). Para que el espacio convexo pueda ser representado es necesario que los conjuntos cerrados por arriba y por debajo sean anticadenas. Representaremos entonces un espacio convexo como la intersección de un conjunto cerrado por arriba con el conjunto cerrado por debajo.

Definición 4.5.5 (Gunter et al., 1997)
Sea $< \mathcal{P}, \preceq >$ un poset y supongamos $\mathcal{U}, \mathcal{L} \subseteq \mathcal{P}$. Definiremos el espacio convexo $\mathcal{B}$ como,

$$\mathcal{B}(\mathcal{U}, \mathcal{L}) = \{y \in \mathcal{P} \mid x \preceq y \preceq z \text{ para algún } x \in \mathcal{L},\ z \in \mathcal{U}\}$$ ♦

Lema 4.5.2 (Gunter et al., 1997)
Sea $< \mathcal{P}, \preceq >$ un poset y supongamos $\mathcal{C} \subseteq \mathcal{P}$. Entonces $\mathcal{C}$ es un espacio convexo si y solo si $\mathcal{C} = \mathcal{B}(min(\mathcal{C}), max(\mathcal{C}))$. □

Observamos entonces que un conjunto convexo puede ser representado por un par de anticadenas utilizando la operación de intersección. Sin embargo, existen otras formas de definir espacios convexos como podemos apreciar en el siguiente lema.

Lema 4.5.3 (Gunter et al., 1997)
Sea $< \mathcal{P}, \preceq >$ *un poset y supongamos* $\mathcal{U}_1, \mathcal{U}_2$ *son conjuntos cerrados por arriba y* $\mathcal{L}_1, \mathcal{L}_2$ *son conjuntos cerrados por debajo de* $\mathcal{P}$*. Cada uno de los siguientes subconjuntos de* $\mathcal{P}$ *son espacios convexos:*

$$\mathcal{U}_1 \cap \mathcal{L}_1 \qquad \mathcal{U}_1 - \mathcal{L}_1 \qquad \mathcal{L}_1 - \mathcal{U}_1 \qquad \mathcal{L}_1 - \mathcal{L}_2 \qquad \mathcal{U}_1 - \mathcal{U}_2 \qquad \square$$

4.5.3. Bordes

Como hemos visto los espacios convexos pueden ser utilizados para representar en forma concisa una clase especial de poset. En [43, 42] se utiliza este concepto para representar el conjunto de patrones emergentes. Los conceptos utilizados en dichos trabajos son los mismos que hemos presentado hasta aquí, si bien se utilizarán otros términos para referirse a los mismos conceptos. En particular, nos referiremos a los espacios convexos como *intervalos cerrados* y se definirá la noción de borde en forma similar a la definición 4.5.5. Aún cuando las definiciones propuestas en [34] y en [20] son similares, dichos trabajos fueron desarrollados en forma independiente y para dominios diferentes. Por un lado, en [34] se provee un conjunto completo de operaciones algebraicas sobre bordes. Por otro lado, en [20] solamente se utiliza la operación de diferencia entre bordes, la cual difiere sustancialmente de la operación de diferencia propuesta en [34].

Definición de Bordes

Con objeto de tener una definición más apropiada de espacios convexos para el dominio de patrones, en el trabajo de (Dong & Li, 1999) se reescribe tal definición utilizando el operador de conjuntos $\subseteq$ en lugar del operador genérico $\preceq$.

Definición 4.5.6 (Gunter et al., 1997; Dong & Li, 1999)

Una colección $\mathcal{C}$ de conjuntos es un espacio convexo (convex space) si para todo X, Y y Z tal que $X \subseteq Y \subseteq Z$ y $X, Z \in \mathcal{C}$ implica que $Y \in \mathcal{C}$. ♦

Definición 4.5.7 (Bordes, Dong & Li, 1999)

Un par ordenado $\langle \mathcal{L}, \mathcal{R} \rangle$ es llamado borde (border) si a) tanto $\mathcal{L}$ como $\mathcal{R}$ son colecciones de conjuntos anticadenas [8] *y b) cada elemento de $\mathcal{L}$ es un subconjunto de algún elemento de $\mathcal{R}$ y cada elemento de $\mathcal{R}$ es un superconjunto de algún elemento de $\mathcal{L}$.*

La colección de conjuntos representada por el borde $\langle \mathcal{L}, \mathcal{R} \rangle$ consiste de todos aquellos itemsets que son superconjuntos para algún elemento de $\mathcal{L}$ y subconjunto de algún elemento de $\mathcal{R}$. Esta colección es representada como sigue:

$$[\mathcal{L}, \mathcal{R}] = \{Y \mid \exists X \in \mathcal{L}, \exists Z \in \mathcal{R} \text{ tal que } X \subseteq Y \subseteq Z\}$$

La colección $\mathcal{L}$ es llamado frontera izquierda (left bound) del borde y $\mathcal{R}$ es llamada frontera derecha (right bound). ♦

Ejemplo 4.5.3

Se puede observar que la colección representada por el borde $\langle \{a, bc\}, \{abc, bcd\} \rangle$, ilustrada en la figura 4.3, es el conjunto $\{a, ab, ac, bc, abc, bcd\}$ donde $\mathcal{L}$ consiste de los elementos más generales del espacio y $\mathcal{R}$ consiste de los elementos más específicos. Nótese que aún cuando los siguientes elementos $\{b, c, d, bd, cd, acd, abd, abcd\}$ son subconjuntos del borde derecho o super conjuntos del borde izquierdo, dichos elementos no pertenecen al conjunto de elementos representado por el borde. ◊

Ejemplo 4.5.4

La colección representada por el borde $\langle \mathcal{L}, \mathcal{R} \rangle = \langle \{1, 3\}, \{123, 134\} \rangle$ es el conjunto

$$[\mathcal{L}, \mathcal{R}] = \{\{1\}, \{3\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$$ ◊

[8] Ver definición 4.5.3 en página 144.

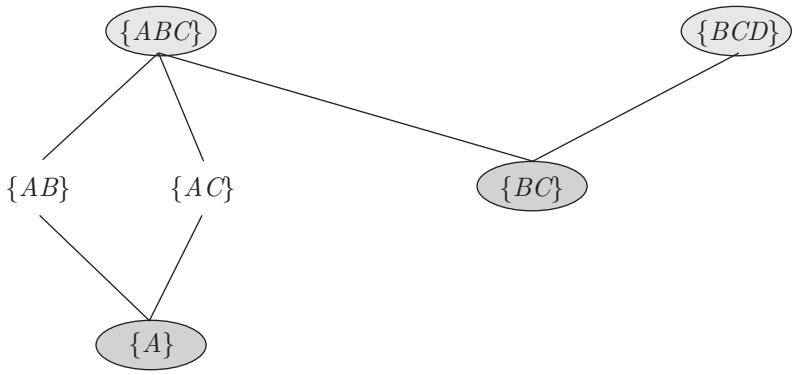


Figura 4.3: Reticulado del borde $\langle\{a, bc\}, \{abc, bcd\}\rangle$.

Ejemplo 4.5.5

La colección representada por el borde $\langle\mathcal{L}_1, \mathcal{R}_1\rangle = \langle\{\emptyset\}, \{\emptyset\}\rangle$ *consiste* $[\mathcal{L}_1, \mathcal{R}_1] = \{\emptyset\}$. *La colección representada por* $\langle\mathcal{L}_2, \mathcal{R}_2\rangle = \langle\{\emptyset\}, ab, bcd\rangle$ *es un conjunto cerrado por debajo y representa el conjunto*

$$[\mathcal{L}_2, \mathcal{R}_2] = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{ab\}, \{bc\}, \{bd\}, \{cd\}, \{bcd\}\}$$

El borde $\langle\mathcal{L}_3, \mathcal{R}_3\rangle = \langle\{\emptyset\}, U\rangle$ *es un conjunto cerrado por arriba y por debajo y representa el conjunto* $[\mathcal{L}_3, \mathcal{R}_3] = Powerset(U)$. $\Diamond$

En general se puede observar que aún cuando la colección a representar sea extremadamente grande, su descripción utilizando bordes es pequeña. Por ejemplo, la colección de elementos representada por $\langle\{a\}, \{abcdefghijkl\}\rangle$ comprende 2^{11} elementos.

Cabe notar la diferencia entre los dos tipos de notaciones $\langle\mathcal{L}, \mathcal{R}\rangle$ y $[\mathcal{L}, \mathcal{R}]$. Un borde $\langle\mathcal{L}, \mathcal{R}\rangle$ es un objeto sintáctico formado por dos límites $\mathcal{L}$ y $\mathcal{R}$ y su semántica es $[\mathcal{L}, \mathcal{R}]$ que consiste del intervalo de conjuntos limitados por $\mathcal{L}$ por debajo y por $\mathcal{R}$ por arriba. Las dos condiciones de la definición aseguran que los bordes son minimales en tamaño.

La siguiente proposición establece que existe una correspondencia uno a uno entre los bordes y las colecciones de intervalos cerrados, asegurando además la existencia de un borde único.

Proposición 4.5.1 (Dong & Li, 1999)

Cada colección de intervalos cerrados de conjuntos $\mathcal{S}$ *tiene un borde único* $\langle\mathcal{L}, \mathcal{R}\rangle$, *donde*

$\mathcal{L}$ es la colección de conjuntos minimales en $\mathcal{S}$ y $\mathcal{R}$ es la colección de conjuntos maximales en $\mathcal{S}$. ♦

En general se preferirá el uso de bordes con conjuntos unitarios tanto para el borde izquierdo como para el borde derecho. Es fácil ver que todo borde puede ser descompuesto en bordes constituidos por conjuntos unitarios.

Operación de Diferencia entre Bordes

La operación principal a ser utilizada en el cálculo de patrones emergentes es la diferencia. Dicha operación se realiza entre dos bordes con características particulares y retorna un nuevo borde representando al intervalo cerrado obtenido de la diferencia.

Definición 4.5.8 (Diferencia de Bordes, Dong & Li, 1999)
Dado un par ordenado de bordes $\langle \{\emptyset\}, \mathcal{R}_1 \rangle$ y $\langle \{\emptyset\}, \mathcal{R}_2 \rangle$, se define una diferencia de bordes (border differential) $\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ como el borde $\langle \mathcal{L}, \mathcal{R} \rangle$ tal que $[\mathcal{L}, \mathcal{R}] = [\{\emptyset\}, \mathcal{R}_1] - [\{\emptyset\}, \mathcal{R}_2]$. ♦

Como se puede observar los bordes utilizados como entrada tienen características especiales ya que ambos requieren que su frontera izquierda sea el conjunto $\emptyset$. Esta restricción es equivalente a pedir que ambos conjuntos sean cerrados por debajo según la definición 4.5.1. La razón de esta restricción es que la operación diferencia no puede garantizar que el conjunto resultante sea un conjunto convexo para el caso de bordes en general. Sin embargo, como parte de la contribución de esta tesis veremos en la sección 5.5 una extensión de la operación de diferencia que permite relajar esta restricción y aún así obtener un conjunto convexo.

Ejemplo 4.5.6
Sea $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle = \langle \{\emptyset\}, \{abcd\} \rangle$ y $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle = \langle \{\emptyset\}, \{bc, bd, cd\} \rangle$. Tales bordes representan

los siguientes conjuntos convexos:

$$[\{\emptyset\}, \mathcal{R}_1] = \{\emptyset, a, b, c, d, ab, ac, ad, bc, bd, cd, abc, abd, acd, bcd, abcd\}$$
$$[\{\emptyset\}, \mathcal{R}_2] = \{\emptyset, b, c, d, bc, bd, cd\}$$
$$[\{\emptyset\}, \mathcal{R}_1] - [\{\emptyset\}, \mathcal{R}_2] = \{a, ab, ac, ad, abc, abd, acd, bcd, abcd\}$$

Puede observarse el resultado de esta diferencia en el reticulado de la figura 4.4. Los itemsets minimales de la diferencia son $\mathcal{L} = \{a, bcd\}$ *y los maximales son* $\mathcal{R} = \{abcd\}$*. Por lo tanto el borde resultante de la diferencia es* $\langle\{\emptyset\}, \mathcal{R}_1\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle = \langle\{a, bcd\}, \{abcd\}\rangle$.◊

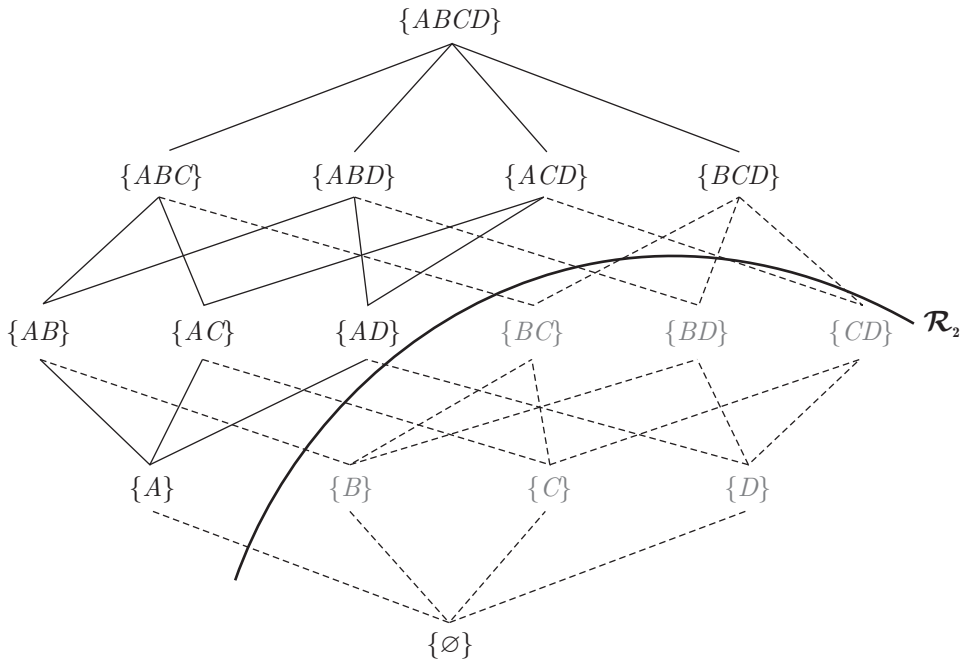


Figura 4.4: Reticulado resultante de la diferencia de bordes.

Veamos cómo podríamos llegar a una situación en donde el espacio no sea convexo si no se respetaran las restricciones impuestas sobre los bordes.

Ejemplo 4.5.7

Supongamos tener los bordes $\langle\mathcal{L}_1, \mathcal{R}_1\rangle = \langle\{1\}, \{123, 134\}\rangle$ *y* $\langle\mathcal{L}_2, \mathcal{R}_2\rangle = \langle\{3\}, \{134\}\rangle$.

Dichos bordes representan los siguientes conjuntos convexos:

$$[\mathcal{L}_1, \mathcal{R}_1] = \{1, 12, 13, 14, 123, 134\}$$
$$[\mathcal{L}_2, \mathcal{R}_2] = \{3, 13, 34, 134\}$$

Sin embargo la diferencia de estos conjuntos producirá otro conjunto no convexo.

$$[\mathcal{L}_1, \mathcal{R}_1] - [\mathcal{L}_2, \mathcal{R}_2] = \{1, 12, 123\}$$

Observar el conjunto candidato de elementos minimales es $\mathcal{L} = \{1\}$ y el conjunto candidatos de elementos maximales es $\mathcal{R} = \{123\}$. Por lo tanto el borde tentativo para representar la diferencia sería $\langle \mathcal{L}, \mathcal{R} \rangle = \langle \{1\}, \{123\} \rangle$. Sin embargo el itemset $\{13\}$ no pertenece a la diferencia y por lo tanto existe un itemset Y tal que no respeta la definición 4.5.6 de bordes. ◊

Las siguientes proposiciones tienen por objeto demostrar que la diferencia de bordes genera un nuevo espacio convexo. Junto con las proposiciones incluiremos sus respectivas demostraciones por ser de utilidad en la generalización de la operación de diferencia propuesta como aporte de este tesis en la sección 5.5. En primer lugar se demostrará que una operación de diferencia aún más restringida respeta tal propiedad. Luego se generalizará dicho resultado para las restricciones previamente enunciadas. Demostraciones análogas para otras operaciones como unión e intersección pueden encontrarse en [34].

Proposición 4.5.2 (Diferencia de bordes bien definida (1), Dong & Li, 1999) *La diferencia de bordes $\langle \{\emptyset\}, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ está bien definida y su resultado es o bien $\langle \{\emptyset\}, \{\emptyset\} \rangle$ o su frontera derecha es $\{U\}$. Mas aún, este será $\langle \{\emptyset\}, \{\emptyset\} \rangle$ cuando algún itemset de $\mathcal{R}_2$ contenga U.*

Demostración:
Sea $D = [\{\emptyset\}, \{U\}] - [\{\emptyset\}, \mathcal{R}_2]$. Pueden surgir dos casos:

1. *$D = \emptyset$. Entonces D es convexo y la diferencia es $\langle \{\emptyset\}, \{\emptyset\} \rangle$.*
2. *$D \neq \emptyset$. Sea $\mathcal{R}_2 = \{S_1, \ldots, S_k\}$. Se deben probar dos puntos.*

- *D es convexo. Sea X e Y en D y Z un itemset tal que $X \subseteq Z \subseteq Y$. Como Y está en D entonces $Y \subseteq U$. Entonces $Z \subseteq U$. Como X está en D, X no es subconjunto de ningún S_i. Como $X \subseteq Z$ de aquí surge que Z no es subconjunto de ningún S_i. Combinando este resultado con $Z \subseteq U$ se puede ver que Z está en D. Por lo tanto D es convexo.*

- *U el único itemset maximal en D. Como $D \neq \emptyset$, es evidente que U no es subconjunto de ningún S_i entonces U está en D. Como todos los itemsets de D son subconjuntos de U, entonces U es el único itemset maximal de D.* ■

La siguiente proposición muestra que la operación de diferencia de bordes está bien definida para las situaciones en que las fronteras izquierdas sean el conjunto vacío y sin imponer restricciones sobre las fronteras derechas. Tal resultado es obtenido realizando una reducción a la proposición anterior donde la operación de diferencia se define con ambas fronteras izquierdas con conjuntos vacíos y la frontera derecha $\mathcal{R}_1$ consta de un conjunto unitario.

Proposición 4.5.3 (Diferencia de bordes bien definida (2), Dong & Li, 1999) *La diferencia de bordes $\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ está bien definida ya que se puede reescribir $\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ como operaciones de diferencia más simples $\langle \{\emptyset\}, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ de la siguiente forma: Sea $\mathcal{R}_1 = \{U_1, \ldots, U_m\}$ y sea $\langle \mathcal{L}'_i, \mathcal{R}'_i \rangle = \langle \{\emptyset\}, \{U_i\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$. Entonces $\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle = \langle \mathcal{L}, \mathcal{R} \rangle$ donde $\mathcal{L} = \cup_{i=1}^{m} \mathcal{L}'_i$ y $\mathcal{R} = \cup_{i=1}^{m} \mathcal{R}'_i$.* ♦

Consideremos el siguiente ejemplo donde se puede apreciar la descomposición realizada sobre el primer borde según la proposición 4.5.3.

Ejemplo 4.5.8

Sean $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle = \langle \{\emptyset\}, \{abc, acd\} \rangle$ y $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle = \langle \{\emptyset\}, \{ac, bcd\} \rangle$ un par de bordes. Se pretende realizar la diferencia $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle - \langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ tal como se observa en la figura 4.5. Aplicando la proposición anterior se descompone la diferencia en tantas operaciones como

sea la cardinalidad del conjunto $\mathcal{R}_1$*. En particular,*

$$\langle \mathcal{L}'_1, \mathcal{R}'_1 \rangle = \langle \{\emptyset\}, \{abc\} \rangle - \langle \{\emptyset\}, \{ac, bcd\} \rangle = \{ab, abc\} = \langle \{ab\}, \{abc\} \rangle$$
$$\langle \mathcal{L}'_2, \mathcal{R}'_2 \rangle = \langle \{\emptyset\}, \{acd\} \rangle - \langle \{\emptyset\}, \{ac, bcd\} \rangle = \{ad, acd\} = \langle \{ad\}, \{acd\} \rangle$$
$$\langle \mathcal{L}, \mathcal{R} \rangle = \langle \cup_{i=1}^{2} \mathcal{L}_i, \cup_{i=1}^{2} \mathcal{R}_i \rangle = \langle \{ab, ad\}, \{abc, acd\} \rangle \qquad \Diamond$$

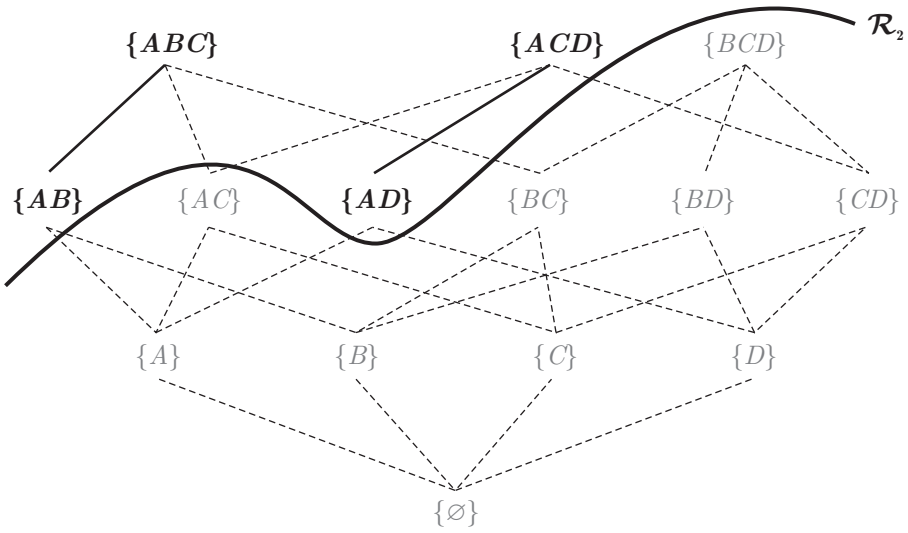


Figura 4.5: Diferencia entre bordes.

En adelante se asumirá (sin pérdida de generalidad) que las operaciones de diferencia son de la forma $\langle \{\emptyset\}, U \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$ basados en la proposición 4.5.3. Además se asumirá que ningún itemset de $\mathcal{R}_2$ contiene al conjunto $\{U\}$ ya que su resultado es trivialmente vacío.

Operación de Diferencia Naïve

Como se puede apreciar en los ejemplos 4.5.6 y 4.5.7 el cálculo de la diferencia entre bordes se puede realizar expandiendo cada borde y realizando la diferencia de conjuntos entre ambos. Como paso final se procede a buscar aquellos itemsets maximales y minimales del conjunto resultado. Es interesante notar que gracias a los resultados obtenidos por las proposiciones 4.5.2 y 4.5.3 sabemos que la frontera derecha para cada diferencia parcial es

el conjunto $\{U_i\}$[9] y la frontera derecha para la diferencia es la unión de estos resultados, y por lo tanto podemos reducir el problema a simplemente buscar los itemsets minimales del conjunto resultado.

Aún sacando provecho de esta heurística, la diferencia de bordes utilizando enumeración de candidatos es ineficiente y muchas veces inviable. Recordemos que los bordes representan en forma concisa un gran número de patrones y cuanto más conciso es el borde mayor es el número de patrones que representa. Si se procede a expandir cada uno de los bordes se generará una cantidad exponencial de patrones para cada borde, lo cual limita la utilidad del método.

Operación de Diferencia sin Expansión de Bordes

Se han propuesto alternativas que no requieren la enumeración de elementos de la expansión de los bordes para realizar la diferencia. Todas estas alternativas buscan simplemente obtener la frontera izquierda ya que como ha sido mencionado la frontera derecha es obtenida trivialmente. Una de esas alternativas es propuesta en [51] donde se utiliza un algoritmo de generación de candidatos por niveles. Otra de las alternativas es la propuesta en [43, 42, 20] donde se genera un conjunto de candidatos basado en las fronteras derechas de ambos bordes y además se propone una estrategia incremental para reducir aún más la cardinalidad de este conjunto. Esta última estrategia es la utilizada para la generación de patrones emergentes y por nuestra diferencia extendida (sección 5.5). Por tal motivo analizaremos esta propuesta en más detalle.

La propuesta de Dong et al. está basada en tres ideas principales:

1. Generar solamente aquellos itemsets que tienen una alta probabilidad de pertenecer a la frontera izquierda.

2. Evitar enumeración completa de los elementos del conjunto candidato.

3. Evitar generar itemsets no minimales.

[9] Se asumía que ningún itemset de $\mathcal{R}_2$ contenía al conjunto $\{U\}$.

En primer lugar se definirá el conjunto $\mathcal{S}$ al cual nos referiremos como el conjunto que contiene los elementos potenciales pertenecientes a la diferencia. Este conjunto está compuesto por el producto cartesiano entre conjuntos derivados de las fronteras derechas de ambos bordes. La siguiente proposición es la base fundamental del algoritmo de diferencia.

Proposición 4.5.4 (Dong & Li, 1999)
Sea $\mathcal{S} = \{\{s_1, \ldots, s_k\} \mid s_i \in U - S_i, 1 \leq i \leq k\}$ *entonces*

$$min([\{\emptyset\}, \{U\}] - [\{\emptyset\}, \{S_1, \ldots, S_k\}]) = min(S) \qquad \blacklozenge$$

En la proposición anterior la función *min* se define en forma análoga a la definición 4.5.2 donde $\preceq$ representa $\subseteq$. La demostración a dicha proposición puede encontrarse en [43, pág. 11]. Es interesante notar que al aplicar la función *min* sobre la diferencia de bordes se obtiene $\mathcal{L}$ y como consecuencia de la proposición anterior dicha diferencia no es necesaria que sea realizada sino simplemente realizar el cálculo de $\mathcal{S}$ y sobre este conjunto aplicar la función *min*. Como se puede apreciar el conjunto $\mathcal{S}$ depende de la cardinalidad de $\mathcal{R}_2$ y de la cardinalidad de cada uno de los elementos de $\mathcal{R}_2$, el cual será notablemente más pequeño que la enumeración de los elementos de los bordes.

Consideremos el siguiente ejemplo para observar cómo se aplica esta proposición.

Ejemplo 4.5.9
Sean los bordes $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle = \langle \{\emptyset\}, \{abcd\} \rangle$ *y* $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle = \langle \{\emptyset\}, \{a, bd, cd\} \rangle$. *Tales bordes representan los siguientes conjuntos convexos utilizando la estrategia naïve para el cálculo de la diferencia:*

$$\begin{aligned}
[\{\emptyset\}, \mathcal{R}_1] &= \{\emptyset, a, b, c, d, ab, ac, ad, bc, bd, cd, abc, abd, acd, bcd, abcd\} \\
[\{\emptyset\}, \mathcal{R}_2] &= \{\emptyset, a, b, c, d, bd, cd\} \\
[\{\emptyset\}, \mathcal{R}_1] - [\{\emptyset\}, \mathcal{R}_2] &= \{ab, ac, ad, bc, abc, abd, acd, bcd, abcd\} \\
\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle &= \langle \{ab, ac, ad, bc\}, \{abcd\} \rangle
\end{aligned}$$

Sea U *el conjunto* $\{abcd\}$ *y* $S_1 = \{a\}$, $S_2 = \{bd\}$ *y* $S_3 = \{cd\}$. *Entonces* $U - S_1 = \{bcd\}$,

$U - S_2 = \{ac\}$ y $U - S_3 = \{ab\}$. El conjunto $\mathcal{S}$ resultante es:

$$\mathcal{S} = \{baa, bab, bca, bcb, cac, cab, cba, cbb, daa, dab, dca, dcb\}$$

Observar que al realizar el producto cartesiano entre los $U - S_i$ se obtiene un multibag como el de arriba. Sin embargo, la definición de $\mathcal{S}$ dice que es una colección (conjunto) de conjuntos. Por tal motivo eliminaremos los elementos repetidos de cada itemset y luego eliminaremos los itemsets repetidos ordenando los elementos en cada itemset para identificar mejor aquellos elementos repetidos.

$$\begin{aligned}\mathcal{S} &= \{ab, ab, abc, bc, ac, abc, abc, bc, ad, abd, acd, bcd\} \\ &= \{ab, abc, bc, ac, ad, abd, acd, bcd\}\end{aligned}$$

Utilizando la proposición 4.5.4 se obtiene que:

$$\begin{aligned}min([\{\emptyset\}, \mathcal{R}_1] - [\{\emptyset\}, \mathcal{R}_2]) = min(\mathcal{S}) &= min(\{ab, abc, bc, ac, ad, abd, acd, bcd\}) \\ &= \{ab, bc, ac, ad\}\end{aligned}$$

Por lo tanto $\langle \{\emptyset\}, \mathcal{R}_1 \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle = \langle \{ab, ac, ad, bc\}, \{abcd\} \rangle$. ◊

Aún cuando esta forma de computar de diferencia es varios ordenes de magnitud menor que realizar la expansión de los bordes, la misma puede ser mejorada utilizando una expansión incremental, evitando de esta forma expandir el conjunto $\mathcal{S}$ en su totalidad.

La segunda idea entonces consiste en generar itemsets parciales del conjunto $\mathcal{S}$ y proceder a minimizar el conjunto antes de realizar el próximo incremento. En [43] se propone el siguiente fragmento de código para esta expansión incremental:

1: $\mathcal{L} = \{\{x\} \mid x \in U - S_1\}$;
2: **Para** i = 2 hasta k **hacer**
3: $\quad \mathcal{L} = \min(\{X \cup \{x\} \mid X \in \mathcal{L}, x \in U - S_i\}$;
4: **fin Para**

Este proceso reduce notablemente la cantidad de itemsets en el conjunto $\mathcal{S}$ durante los resultados parciales. Utilizaremos el ejemplo anterior para mostrar la mejora producida por esta idea.

Ejemplo 4.5.10

Sean los bordes $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle = \langle \{\emptyset\}, \{abcd\} \rangle$ *y* $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle = \langle \{\emptyset\}, \{a, bd, cd\} \rangle$. *Sea* $U = \{abcd\}$, $S_1 = \{a\}$, $S_2 = \{bd\}$ *y* $S_3 = \{cd\}$. *Entonces* $U - S_1 = \{bcd\}$, $U - S_2 = \{ac\}$ *y* $U - S_3 = \{ab\}$.

Inicializaremos $\mathcal{L} = \{b, c, d\}$[10] *utilizando* $U - S_1$.

Utilizando los elementos de $U - S_2 = \{ac\}$ *obtenemos un nuevo* $\mathcal{L}$*:*

$$\mathcal{L} = min(\{ab, ac, ad, bc, c, cd\}) = \{ab, c, ad\}$$

Utilizando los elementos de $U - S_3 = \{ab\}$ *obtenemos el* $\mathcal{L}$ *final:*

$$\mathcal{L} = min(\{ab, ac, bc, ad, abd\}) = \{ab, ac, bc, ad\} \qquad \Diamond$$

Notar que la cardinalidad del conjunto $\mathcal{L}$ en el ejemplo nunca superó los 6 itemsets cuando en el ejemplo anterior la cardinalidad del conjunto llegó a ser de 12 itemsets con elementos repetidos y 8 itemsets sin repetidos. Esta reducción es mucho más evidente en ejemplos con un valor de k moderado (tal como $k = 300$) donde expandir totalmente el conjunto $\mathcal{S}$ consistiría de al menos 2^{300} itemset si cada cada $U - S_i$ consistiera de al menos 2 itemsets.

Como último agregado a este proceso de diferencia entre bordes se procederá a evitar generar itemsets no minimales. Consideramos nuevamente el ejemplo 4.5.10:

Ejemplo 4.5.11

Sea el conjunto inicial $\mathcal{L} = \{b, c, d\}$ *y* $U - S_2 = \{ac\}$. *En el proceso de generar el nuevo conjunto* $\mathcal{L}$ *podemos observar que uno de los itemsets del conjunto* $\mathcal{L}$ *es un subconjunto de* $U - S_2$ ($c \subseteq ac$). *Por lo tanto cuando se utilice dicho itemset en la generación del*

[10] Según nuestra convención de notación el conjunto $\{b, c, d\}$ representa el conjunto $\{\{b\}, \{c\}, \{d\}\}$ y el conjunto $\{bcd\}$ representa el conjunto $\{b, c, d\}$.

nuevo $\mathcal{L}$ se generará el siguiente conjunto $\{ac, c\}$. Pero al realizar la minimización solo quedará el itemset $\{c\}$. $\Diamond$

La idea se basa en eliminar la generación de tales candidatos realizando esa comprobación de subconjunto. En caso afirmativo *a*) se transferirá el itemset del viejo $\mathcal{L}$ al nuevo y *b*) se quitará el elemento del conjunto $U - S_i$ para evitar la generación de dichos candidatos.

Además de estas notables mejoras, en [43] se considera una optimización para el proceso de minimización. En el algoritmo 6 se puede observar el pseudocódigo de la operación de diferencia, la cual utiliza los conceptos hasta aquí mencionados. La estrategia para evitar la generación de itemsets no minimales puede observarse en las líneas 6–8. La expansión y minimización de cada $T_i = U - S_i$ se lleva a cabo en las líneas 9–12. La expansión incremental y minimización se lleva a cabo en las líneas 4–14. El caso especial en que algún itemset del segundo borde cubra al conjunto unitario de la frontera derecha del primer borde es considerado en las líneas 2–3.

4.6. Cálculo de Patrones Emergentes

Hemos visto en secciones anteriores que los intervalos cerrados (espacios convexos) pueden ser representados en forma concisa utilizando bordes. En particular los conjuntos cerrados por debajo y por arriba son espacios convexos en los que solo es necesario especificar uno de los bordes. También ha sido analizada la operación de diferencia sobre bordes utilizando diferentes alternativas. Veremos entonces en esta sección como estos conceptos son utilizados para calcular y representar patrones emergentes.

En primer lugar formalizaremos el problema del descubrimiento de patrones emergentes de un conjuntos de datos $\mathcal{D}_1$ a un conjunto $\mathcal{D}_2$.

Definición 4.6.1 (Minería de Patrones Emergentes, Dong & Li, 1999)
El problema de descubrir patrones emergentes (EPs) para un umbral ρ de tasa de crec-

Algoritmo 6 BorderDiff($\langle \{\emptyset\}, U\rangle$,$\langle \{\emptyset\}, S_1, \ldots, S_k\rangle$): $\langle \{\emptyset\}, U\rangle - \langle \{\emptyset\}, S_1, \ldots, S_k\rangle$

1: $T_i \leftarrow U - S_i$ para $1 \leq i \leq k$;
2: **Si** al menos un T_i es vacío **entonces**
3: **retornar** $\langle \{\emptyset\}, \{\emptyset\}\rangle$
4: **fin Si**
5: $\mathcal{L} \leftarrow \{\{x\} \mid x \in T_i\}$;
6: **Para** $i \leftarrow 2$ hasta k **hacer**
7: $Nuevo\mathcal{L} \leftarrow \{X \in \mathcal{L} \mid X \cap T_i \neq \emptyset\}$;
8: $\mathcal{L} \leftarrow \mathcal{L} - Nuevo\mathcal{L}$;
9: $T_i \leftarrow T_i - \{x \mid \{x\} \in Nuevo\mathcal{L}\}$;
10: **Para todo** $X \in \mathcal{L}$ **hacer**
11: **Para todo** $x \in T_i$ **hacer**
12: **Si** $X \cup \{x\}$ no es superconjunto de ningún $Z \in Nuevo\mathcal{L}$ **entonces**
13: Insertar $X \cup \{x\}$ en $Nuevo\mathcal{L}$;
14: **fin Si**
15: **fin Para**
16: **fin Para**
17: $\mathcal{L} \leftarrow Nuevo\mathcal{L}$;
18: **fin Para**
19: **retornar**$\langle \mathcal{L}, \{U\}\rangle$;

imiento dado consiste en encontrar todos los ρ-EPs. ♦

Podemos visualizar este problema en un espacio de 2 dimensiones cuyos ejes corresponden a los soportes en ambos conjuntos de datos tal como se aprecia en la figura 4.6 [20]. Se identificará un punto (σ_1, σ_2) en este plano como todos los itemsets X cuyo $supp_1(X) = \sigma_1$ y $supp_2(X) = \sigma_2$. Dado un umbral ρ se puede establecer una recta en el plano, donde todos los ρ-EPs de $\mathcal{D}_1$ a $\mathcal{D}_2$ se encuentran ubicados en (la región enmarcada por) el triángulo ΔACE.

4.6.1. Acercamiento Naïve

El primer acercamiento para descubrir patrones emergentes de un conjunto de datos $\mathcal{D}_1$ a otro $\mathcal{D}_2$ consiste de dos pasos [42]. En primer lugar se procede a calcular el soporte en ambos conjuntos para todos los itemsets posibles. Luego se realiza una evaluación para cada itemset si su tasa de crecimiento (growth-rate) es mayor o igual a un umbral ρ dado. Como resultado se retorna la enumeración de todos los itemsets que pasen satisfactoria-

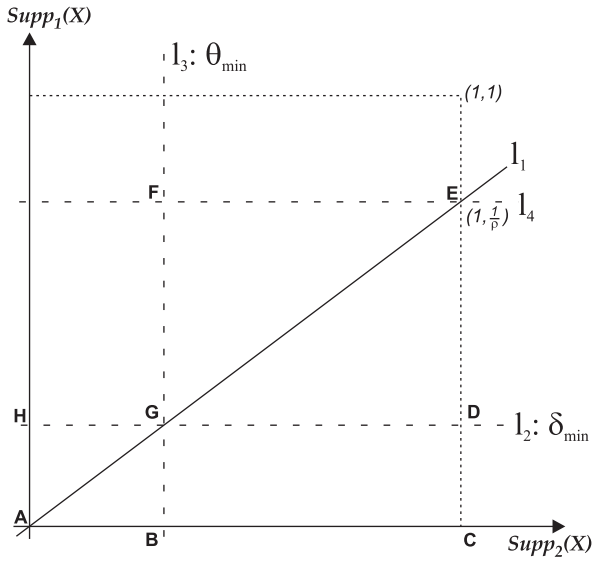


Figura 4.6: Plano de soportes 2D

mente la evaluación.

Claramente esta enumeración de patrones durante todo el proceso de minería hace inviable al método ya que el número total de itemsets crecerá exponencialmente en relación al número de atributos y la cardinalidad de cada atributo (cantidad de valores que puede tomar en dominios discretos) que contenga el conjunto de datos. Por ejemplo supongamos una relación descrita por 3 atributos donde cada atributo es binario (2 valores); entonces el número total de itemsets posibles de la relación es 26. Tal valor es calculado teniendo en cuenta el reticulado que genera. Existen $\binom{3}{1}2^1$ itemsets unitarios (cada atributo por la cantidad de valores posibles que puede tomar), $\binom{3}{2}2^2$ itemsets de 2 items (2-itemset) y $\binom{3}{3}2^3$ 3-itemsets. Si se considerara una aplicación real como la base de datos de hongos, evaluada en [43, 42, 20], se observaría que contiene 22 atributos categóricos con un promedio de 6 valores posibles cada uno, por lo cual el número total de itemsets posibles de 3 items es aproximadamente $\binom{22}{3} * 6^3 = 332640$. Claramente es inviable sino imposible enumerar todos los itemsets de una aplicación real.

Otra estrategia consiste en buscar todos aquellos patrones cuyo soporte supere cierto

umbral en el conjunto de datos destino[11] y luego solo para estos patrones calcular su soporte en el conjunto de datos de soporte identificando aquellos que tengan una tasa de crecimiento superior al umbral.

Aún cuando esta estrategia constituye una mejora con respecto al primer acercamiento la cantidad de itemsets potenciales en el primer paso sigue haciendo inviable el método. Más aún, este acercamiento aún cuando retorna una colección de ρ-EPs no es completo al no considerar aquellos itemsets que no superaron el umbral del primer paso pero cuya tasa de crecimiento es superior al ρ. Para solucionar dicho problema es necesario reducir el umbral del primer paso, generando aún más patrones y así perdiendo la eficacia de la heurística.

4.6.2. Descomposición del Problema

En lugar de enumerar los itemsets potenciales y evaluar su tasa de crecimiento, la estrategia propuesta en [43, 42, 20] consiste en dividir el problema en tres subproblemas más pequeños y utilizar los conceptos de bordes y espacios convexos para evitar la enumeración y la evaluación individual de itemsets.

Para tal efecto utilizaremos nuevamente el plano de soportes al cual le añadiremos tres rectas imaginarias, las cuales nos delimitarán tres sectores definiendo tres subproblemas (fig. 4.7). Llamaremos a estas rectas l_2, l_3 y l_4 donde l_2 y l_3 son de especial interés ya que en la intersección de ambas rectas (las cuales son perpendiculares) se ubicará el punto que determinará la calidad y cantidad de los patrones encontrados. En [20] es mencionado que este punto debería estar lo más cercano al origen posible. Este concepto será retomado en la sección 4.7.

1. Encontrar los EPs del rectángulo $BCDG$: Se puede observar en la figura 4.7 que el conjunto de EPs que está en este rectángulo consiste de los EPs cuyo soporte en $\mathcal{D}_2$ es mayor o igual a la recta $l_3 : supp_2(X) = \theta_{min}$ y soporte en $\mathcal{D}_1$ es menor a la recta $l_2 : supp_1(X) = \delta_{min}$.

[11] Ver definiciones de conjunto de datos destino y soporte en la página 136.

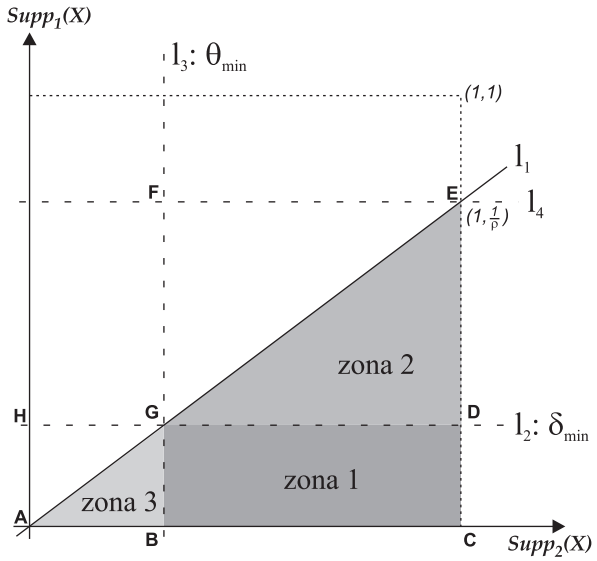


Figura 4.7: Subproblemas en el Plano de Soporte

La estrategia seguida en [43, 42, 20] se basa en buscar patrones emergentes en este rectángulo. La ventaja principal de esta estrategia es que se evita el manejo de una cantidad exponencial de candidatos por la utilización de bordes logrando una mayor eficiencia en tiempo y espacio aún en la búsqueda de EPs con mucha cantidad de items. La utilización de bordes facilita la tarea al usuario ya que la cantidad de patrones a analizar se decrementa drásticamente.

Para que esta estrategia sea viable es necesario probar que el espacio de EPs que se encuentra en el rectángulo $BCDG$ es convexo y además es necesario un mecanismo para manipular bordes.

2. Encontrar los EPs del triángulo ΔGDE: El conjunto de EPs que se encuentran en este triángulo puede obtenerse de un conjunto de itemsets candidatos que se encuentra en el rectángulo $BCDG$ que consta de aquellos itemsets cuyo soporte en $\mathcal{D}_2$ es mayor o igual a la recta $l_3 : supp_2(X) = \theta_{min}$, soporte en $\mathcal{D}_1$ es mayor o igual a la recta $l_2 : supp_1(X) = \delta_{min}$.

3. Encontrar los EPs del triángulo ΔABG: En este triángulo se encuentran aquellos

EPs cuyo soporte no supera ninguno de los umbrales propuestos. Como se menciona en [20] aquí se encuentran muchos itemsets ya que como es sabido en el caso promedio de los conjuntos de datos existen muchos itemsets de bajo soporte.

Cabe mencionar que estos itemsets de bajo soporte son muy útiles, ya que pueden proveer nueva información sobre algún concepto ya conocido, marcar en forma *temprana* alguna tendencia o detectar algún comportamiento *anómalo*. Ejemplos de estos casos pueden verse en la sección. A partir de estos ejemplos podemos ver que probablemente muchos de los EPs más significativos o predictivos se encuentren en este área.

En el trabajo de Dong y Li [20, 43] las áreas triangulares no son utilizadas para la tarea de minería de EPs, si bien dichos autores proponen dos alternativas que utilizan estrategias recursivas que intentan solucionar estos problemas. En este capítulo solo evaluaremos la estrategia utilizada para obtener los EPs en el rectángulo $BCDG$ mientras que las demás alternativas serán evaluadas en la sección 5.1.

4.6.3. Generación de Bordes

Como se verá en próximas secciones para el cálculo del rectángulo $BCDG$ se utilizarán operaciones sobre bordes. En particular necesitaremos dos bordes, uno de los cuales representará los itemsets cuyo soporte en $\mathcal{D}_1$ sea mayor o igual a l_2 al que llamaremos $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$ y otro que representará los itemsets cuyo soporte en $\mathcal{D}_2$ sea mayor o igual a l_3 al que llamaremos $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ (figura 4.8). Como fuera mencionado dichos conjuntos son convexos y por lo tanto representables mediante bordes.

En [43, 42, 20] se propone la utilización del algoritmo Max-Miner[12] para la generación de estos bordes. A tal efecto se utilizan los umbrales δ_{min} y θ_{min} de las rectas l_2 y l_3, para el cálculo de patrones maximales en los conjuntos de datos $\mathcal{D}_1$ y $\mathcal{D}_2$ respectivamente.

Notar que los patrones maximales obtenidos son los patrones más específicos de cada conjunto de datos, por lo que son el conjunto ideal para formar parte de la frontera derecha

[12] Ver sección 3.2.2 página 77.

de cada borde. Además por la propiedad de antimonotonía (Propiedad 2.5.1) sabemos que todos los subconjuntos de estos patrones tienen un soporte mayor o igual a éstos y por lo tanto la frontera izquierda de cada borde estará constituida por el conjunto $\emptyset$.

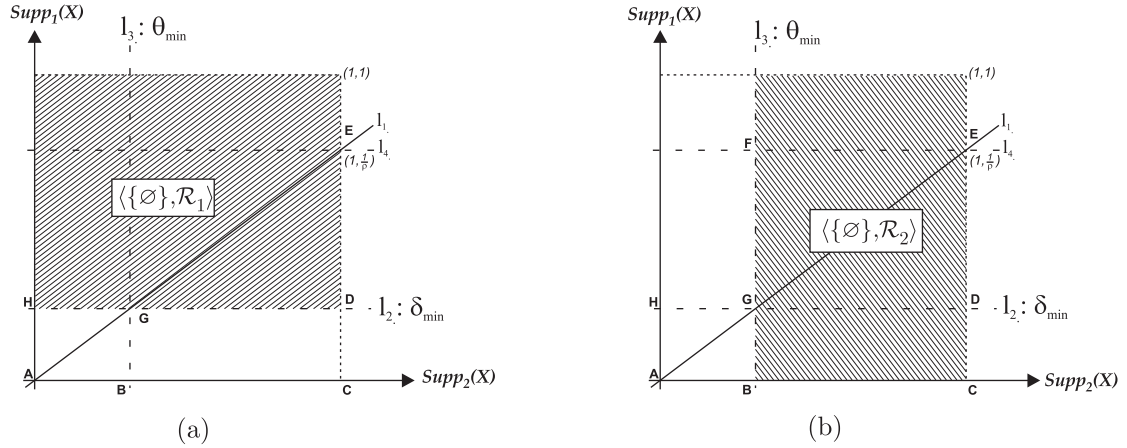


Figura 4.8: Bordes en Plano de Soportes

4.6.4. Utilización de Bordes

En la sección 4.6.3 ha sido mostrado que el espacio de itemsets que se encuentran entre las rectas $l_2 : supp_1(X) = \delta_{min}$ y $supp_1(X) = 1$ en el conjunto de datos $\mathcal{D}_1$ puede ser representado mediante un borde al cual llamaremos $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle = \langle \{\emptyset\}, \mathcal{R}_1 \rangle$. De similar manera se representa el espacio de itemsets que se encuentran entre las rectas $l_3 : supp_2(X) = \theta_{min}$ y $supp_2(X) = 1$ en el conjunto de datos $\mathcal{D}_2$ mediante un borde al cual llamaremos $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle = \langle \{\emptyset\}, \mathcal{R}_2 \rangle$. En la figura 4.8 se muestran las áreas cubiertas por dichos espacios de itemsets.

Como podemos apreciar, tenemos definidos dos bordes sobre el espacio de itemsets, uno de los cuales $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$ cubre el rectángulo $BCDG$. El otro borde $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ por su parte cubre exactamente todo el resto del borde $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$. Por lo tanto, si realizamos la operación de diferencia $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle - \langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ tendríamos exactamente el conjunto de patrones emergentes del rectángulo $BCDG$ (figura 4.9).

El punto principal es verificar si dicha operación puede ser realizada ya que como

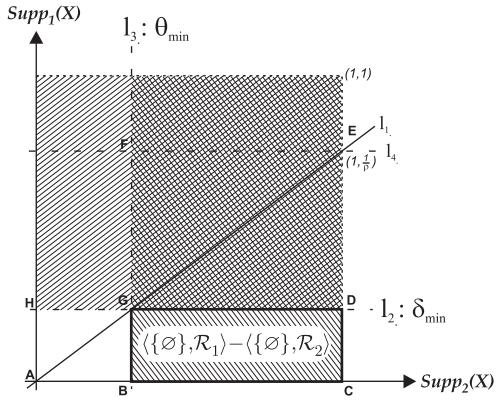


Figura 4.9: Diferencia de Bordes

hemos visto en el ejemplo 4.5.7 la operación de diferencia sobre bordes no puede ser aplicada a cualquier clase de bordes. Para ello se deben verificar las condiciones según la definición 4.5.8. Podemos ver entonces que ambos bordes respetan la restricción de tener una frontera izquierda $\mathcal{L} = \{\emptyset\}$. Además la proposición 4.5.3 garantiza que el borde $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$ puede descomponerse para generar bordes de fronteras derechas con conjuntos unitarios.

En [43, 42, 20] se utiliza esta idea para obtener los EPs del rectángulo $BCDG$. Para ello se utiliza un algoritmo llamado MBD-LLBORDER el cual básicamente invoca múltiples veces al algoritmo BORDERDIFF[13]. Cada llamada corresponde la descomposición de la frontera derecha del primer borde.

4.7. Jumping Emerging Patterns

Hasta aquí hemos visto como lo patrones emergentes pueden ser utilizados eficientemente para detectar tendencias, realizar tareas de clasificación y buscar características anormales en un conjunto de datos como una de las principales diferencias con respecto a otros acercamientos. También se ha analizado la posibilidad de realizar un manejo más eficiente de estos patrones emergentes estableciendo un marco de trabajo donde los patrones pueden ser analizados sin necesidad de una enumeración completa.

[13]Ver algoritmo 6 página 158.

Sin embargo, como se ha podido notar a través de los ejemplos mencionados en las secciones anteriores, la cantidad de patrones emergentes es usualmente grande. En un esfuerzo por reducir la cantidad de patrones manteniendo la significancia de los patrones obtenidos (en el caso de los clasificadores esto significará que la exactitud en la clasificación se mantenga) se han propuestos diversos tipos especiales de patrones emergentes. Los distintos tipos de EPs han sido clasificados de acuerdo a las propiedades especiales que presentan siendo cada uno de ellos un subconjunto de los EPs vistos en las secciones precedentes. Por lo tanto, todos los conceptos vistos hasta el momento seguirán teniendo validez en estas secciones. En esta sección veremos el más prominente de estos tipos especiales de EPs denominado *jumping emerging patterns* o JEPs y en la sección 4.8 mencionaremos los restantes tipos especiales de EPs.

4.7.1. Espacio de JEPs

Dentro de los patrones emergentes pueden detectarse una clase especial de EPs que consta de aquellos patrones que tienen un valor infinito como tasa de crecimiento (ver def. 4.3.1). Tales patrones son aquellos que aparecen en el conjunto destino pero no así en el o los conjunto/s de soporte. Es decir, el soporte de estos patrones es del 0 % en el conjunto de datos de soporte y su soporte en el conjunto destino es distinto de 0 % (aparecen al menos una vez).

Definición 4.7.1 (Jumping Emerging Pattern, Dong & Li, 1999)
Un patrón emergente abrupto (jumping emerging pattern) (JEP) de un conjunto de datos $\mathcal{D}_1$ a un conjunto de datos $\mathcal{D}_2$ es definido como un patrón emergente de $\mathcal{D}_1$ a $\mathcal{D}_2$ con una tasa de crecimiento de ∞. ♦

Usualmente en la literatura referente a JEPs [43, 42, 47, 45, 48, 46] se utilizan los conjuntos de datos $\mathcal{D}_p$ y $\mathcal{D}_n$, para referirse al conjunto de datos de instancias positivas y negativas respectivamente. Por una cuestión de simplificación en la notación evitaremos mencionar que los JEPs surgen del conjunto de datos $\mathcal{D}_1$ al conjunto de datos $\mathcal{D}_2$ o en problemas relacionados a clasificación del conjunto de datos $\mathcal{D}_n$ a $\mathcal{D}_p$, tal como se menciona

habitualmente. En aquellos casos en que se prefiera un conjunto de datos distinto se hará una mención explícita del mismo.

Definición 4.7.2 (Espacio de JEPs, Li et al.)
Dado un conjunto de datos $\mathcal{D}_p$ de instancias positivas y un conjunto de datos $\mathcal{D}_n$ de instancias negativas, el espacio de JEPs (JEP space) con respecto a $\mathcal{D}_p$ y $\mathcal{D}_n$ es definido como el conjunto de JEP de $\mathcal{D}_n$ a $\mathcal{D}_p$. ♦

El espacio de JEPs puede ser visualizado en el plano de soportes que utilizamos para ubicar a los patrones emergentes (ver figura 4.10). Si quisiéramos ubicar un JEP X con un punto $(supp_2(X), supp_1(X))$ en el plano, sabemos que X tiene un soporte distinto de cero en $\mathcal{D}_2$ y que el soporte de X en $\mathcal{D}_1$ es igual a cero por lo tanto el itemset X es un JEP que yace en el eje horizontal del plano de soportes $(supp_2(X), supp_1(X))$. En particular, sabemos que todos los JEP estarán representados por la recta del eje horizontal del plano de soportes.

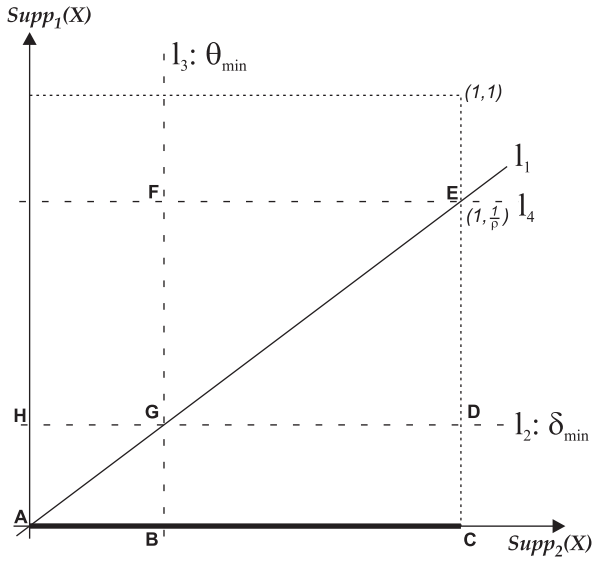


Figura 4.10: Plano de soportes para JEPs.

($\mathcal{D}_n$) Instancias Negativas

Instancia	Items
1	$\{b, c, e\}$
2	$\{b, e, f\}$

($\mathcal{D}_p$) Instancias Positivas

Instancia	Items
1	$\{a, b, c\}$
2	$\{a, c, d\}$

$$JEPs = \left\{ \begin{array}{c} \{abc\}, \{ab\}, \{ac\}, \{a\} \\ \{acd\}, \{ad\}, \{cd\}, \{d\} \end{array} \right\}$$

Cuadro 4.2: Conjunto de Datos Positivos, Negativos y JEPs

Puede observarse además que la colección de JEPs tampoco es *cerrado en subconjuntos*[14]. Por ejemplo si consideramos la tabla 4.2 vemos que el itemset $\{abc\}$ es un JEP (entre otros) mientras que $\{b\}$, $\{c\}$ y $\{bc\}$ no son JEPs.

Sin embargo, el espacio de JEPs es al igual que el espacio de EPs un espacio convexo, con lo cual se pueden utilizar los resultados y algoritmos vistos para EPs. En particular, podremos representar el conjunto de JEPs con un par de conjuntos $\mathcal{L}$ y $\mathcal{R}$ que constituirán los bordes del espacio de JEPs. Es interesante notar que no todo subconjunto de EPs resulta en un espacio convexo como ya ha sido mostrado en el ejemplo 4.5.7 donde podemos suponer que cada borde representa un conjunto de EPs y la diferencia es un subconjunto del primer conjunto de EPs. En consecuencia es necesario demostrar la afirmación que el espacio de JEPs es convexo. Incluiremos dicha demostración además del teorema por su simplicidad y porque permite entender y trabajar un poco más en detalle con las definiciones. Esto nos será de mucha utilidad como se verá en el próximo capítulo.

Teorema 4.7.1 (Li et al.)

Dado un conjunto $\mathcal{D}_p$ de instancias positivas y un conjunto $\mathcal{D}_n$ de instancias negativas, el espacio de JEPs con respecto a $\mathcal{D}_p$ y $\mathcal{D}_n$ es un espacio convexo.[15]

[14] Ver definición 2.1.2 página 14.

[15] Ver definiciones 4.5.4 y 4.5.6 correspondiendo a (Gunter et al., 1997) y (Dong & Li, 1999) respectivamente.

Demostración:

Supongamos que X y Z son itemsets que satisfacen a) $X \subseteq Z$ b) X y Z son JEPs. Entonces para cualquier itemset Y que satisface $X \subseteq Y \subseteq Z$, Y es también un JEP. Esto es porque se satisface que:

- *X no aparece en $\mathcal{D}_n$. Entonces todos sus supersets, los cuales son más específicos*[16] *que X, no pueden aparecer en $\mathcal{D}_n$. Por lo tanto, Y no aparece en $\mathcal{D}_n$.*
- *El soporte de itemset Z en $\mathcal{D}_p$ no es cero. Entonces, todos los subconjuntos de Z, los cuales son más generales que Z, tienen un soporte en $\mathcal{D}_p$ distinto de cero.*

En consecuencia, el espacio de JEPs es convexo. ■

4.7.2. Espacios Horizontales

Para la tarea de descubrir el espacio de JEPs se podrían utilizar los mismos métodos utilizados para obtener el espacio de EPs. Como mencionáramos oportunamente, el espacio de JEPs puede identificarse en el plano de soporte como la recta que pasa por el eje horizontal. Si llevamos el punto G de la figura hacia el origen tendríamos un rectángulo $BCDG$ que cubriría exactamente el conjunto de JEPs. Sin embargo, se puede sacar provecho de las particularidades de los JEPs para esta tarea. En [42, 45, 47] se define con tal efecto un nuevo espacio convexo denominado espacio horizontal.

Definición 4.7.3 (Espacio Horizontal, Li et al.)

Dado un conjunto de datos $\mathcal{D}$ todos los itemsets X con soporte en $\mathcal{D}$ distinto de cero ($supp_{\mathcal{D}}(X) \neq 0$) conforman un espacio convexo denominado espacio horizontal (horizontal space). ♦

El objetivo de los espacios horizontales es describir utilizando bordes el conjunto de todos los itemsets que aparecen en el conjunto de datos. Es de especial importancia notar que este espacio se define con respecto a un solo conjunto de datos, a diferencia de los

[16] Ver definición 2.1.10 página 14.

espacios emergentes que se definen con respecto a dos o más conjuntos de datos, aunque usualmente solamente se utiliza un par de conjuntos de datos $\mathcal{D}_p$ y $\mathcal{D}_n$.

Como la idea es describir todos los itemsets distintos de cero, si existe algún itemset con tal propiedad entonces cualquier subconjunto de ese itemset tiene soporte distinto de cero, en consecuencia el itemset más general es vacío y por lo tanto frontera izquierda $\mathcal{L}$ del borde y el conjunto de los itemsets más específicos constituyen la frontera derecha $\mathcal{R}$. Observar que este espacio convexo es un conjunto cerrado por debajo.[17]

Presentaremos entonces el algoritmo 7 utilizado en [42] para obtener el espacio horizontal para un conjunto de datos $\mathcal{D}$ dado. Este algoritmo busca los patrones maximales del conjunto de datos, pero a diferencia de los métodos presentados en la sección 3.2 este algoritmo no utiliza un umbral. El algoritmo retorna como resultado un conjunto de patrones maximales que representan exactamente a todos los itemsets que aparecen en $\mathcal{D}$ y no cubre ningún itemset que no aparezca en $\mathcal{D}$.

Algoritmo 7 HorizonMiner(Conjunto de Datos $\mathcal{D}$): $\langle \{\emptyset\}, \mathcal{R} \rangle$

1: $\mathcal{R} = T_1$; {El conjunto de datos $\mathcal{D}$ está ordenado de $T_1, \ldots, T_n$.}
2: **Para** $i = 2$ hasta n **hacer**
3: **Si** T_i no es subconjunto de ningún itemset en $\mathcal{R}$ **entonces**
4: Agregar T_i a $\mathcal{R}$;
5: Quitar todos los T de $\mathcal{R}$ tal que $T \subset T_i$;
6: **fin Si**
7: **fin Para**
8: **retornar**$\langle \{\emptyset\}, \mathcal{R} \rangle$;

Podemos entonces ahora describir el espacio de JEPs utilizando el concepto de espacio horizontal. Intuitivamente el espacio de JEP puede verse como los itemsets que aparecen en $\mathcal{D}_p$ y no aparecen en $\mathcal{D}_n$. Dicho de otra forma, son los itemset que aparecen en $\mathcal{D}_p$ pero no aparecen en ambos conjuntos de datos. Si utilizamos operaciones de conjuntos, el conjunto a obtener es $\mathcal{D}_p - (\mathcal{D}_p \cap \mathcal{D}_n)$. Sin embargo, claramente puede observarse que se obtendría el mismo resultado simplemente restando a $\mathcal{D}_p$ el conjunto $\mathcal{D}_n$, por lo cual el conjunto resultante estaría dado por la operación $\mathcal{D}_p - \mathcal{D}_n$. Formalmente,

[17] Ver definición 4.5.1 página 140.

Proposición 4.7.1 (Li et al.)

Sea conjunto $\mathcal{D}_p$ de instancias positivas y $\mathcal{D}_n$ un conjunto de instancias negativas, el espacio de JEP con respecto a $\mathcal{D}_p$ y $\mathcal{D}_n$ está constituido por:

$$[\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]$$

donde $[\{\emptyset\}, \mathcal{R}_p]$ es el espacio horizontal de $\mathcal{D}_p$ y $[\{\emptyset\}, \mathcal{R}_n]$ es el espacio horizontal de $\mathcal{D}_n$. ♦

Utilizando el algoritmo 7 conjuntamente con el algoritmo 6 para realizar la diferencia de bordes se puede computar en forma eficiente el espacio de JEPs. En [42] se presenta el algoritmo JEPPRODUCER donde se realizan los siguientes pasos

1. El cómputo de ambos espacios horizontales.
2. La descomposición del espacio horizontal de $\mathcal{D}_p$ según la proposición 4.5.3
3. Para cada uno de los bordes resultantes se realiza la diferencia con el espacio horizontal $\mathcal{D}_n$.
4. La unión de los bordes resultantes.

Es interesante notar que la unión de dos espacios de JEPs resulta en un nuevo espacio convexo de JEPs, realizando simplemente la unión de las fronteras de ambos bordes. Esto permite retornar como resultado un único borde representado el espacio buscado. Esta propiedad no es respetada por los EPs generales donde en muchas oportunidades el resultado está compuesto por un conjunto de bordes como veremos en el siguiente capítulo.

4.7.3. Mantenimiento Incremental del Espacio de JEPs

Ciertamente, los algoritmos propuestos para realizar la búsqueda de JEPs mejoran en varios ordenes de magnitud a los acercamientos naïve. Sin embargo, el cómputo de dichos bordes no es una tarea trivial. El cálculo del espacio horizontal de cualquier conjunto de

datos es extremadamente costoso[18] y por lo tanto debería ser evitado siempre y cuando sea posible. Usualmente los datos van siendo actualizados y dependiendo del dominio estas actualizaciones ocurrirán con mayor o menor frecuencia;existen por ejemplo, dominios muy dinámicos como los encontrados en e-commerce y dominios basados en la web los cuales son constantemente actualizados con nueva información.

Un acercamiento naïve sería realizar el cómputo desde cero cada vez que es solicitada información sobre tendencias, patrones de clasificación o anomalías. Este método puede ser viable si los bordes con los que se cuenta han sido computados sobre datos no existentes en el presente instante. Esto puede suceder como mencionáramos en entornos muy dinámicos donde, por ejemplo, los datos de una semana atrás no existen más. Sin embargo, en muchos dominios las actualizaciones por eliminación son esporádicas y la mayor parte de las actualizaciones son constituidas por inserciones de nuevas instancias. Por ejemplo, en una base de datos de ventas de mercaderías, se registran todas las ventas realizando una inserción por cada venta nueva, rara vez se realiza una eliminación de una venta.

Un acercamiento más inteligente sería utilizar el conocimiento previo (los bordes ya calculados) y por cada actualización sobre los conjuntos de datos realizar un mantenimiento de los bordes para reflejar dichos cambios. Este acercamiento es conocido como *mantenimiento incremental* y ha sido utilizado exitosamente en diversas áreas de minería de datos. En [42, 47] se proponen un conjunto de algoritmos destinados a manejar dichas actualizaciones sobre los conjuntos de datos. Como ha sido mencionado y se verá en detalle en la sección 4.10.2 los espacios de JEPs generalizan a los espacios de versiones por lo que los algoritmos propuestos tienen mucha similitud con algunos de los algoritmos propuestos para manejar actualizaciones en forma incremental sobre espacios de versiones propuestos en [54, 40, 68, 71].

En primer lugar veremos dos operaciones sobre bordes además de la operación de diferencia ya vista. Estas operaciones son similares a las propuestas en [34, 40] para espacios de versiones y ATMS.

[18] Un costo similar al cálculo de patrones maximales.

Operaciones de Union y Diferencia sobre Bordes

Las operaciones de unión y diferencia aquí mencionadas y propuestas en [42, 47] no pueden ser utilizadas sobre bordes generales. Estos bordes deben respetar ciertas restricciones antes de poder aplicar dichas operaciones. Operaciones sobre bordes más generales pueden encontrarse en [34, 40].

Union Dados dos conjuntos $\mathcal{D}_{p_1}$ y $\mathcal{D}_{p_2}$ de instancias positivas y un conjunto $\mathcal{D}_n$ de instancias negativas. Sean los bordes $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$ del espacio de JEPs de $\mathcal{D}_{p_1}$ con respecto a $\mathcal{D}_n$ y $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ del espacio de JEPs de $\mathcal{D}_{p_2}$ con respecto a $\mathcal{D}_n$. La unión de ambos bordes genera otro borde $\langle \mathcal{L}_3, \mathcal{R}_3 \rangle$ representando $[\mathcal{L}_1, \mathcal{R}_1] \cup [\mathcal{L}_2, \mathcal{R}_2]$ donde $\mathcal{R}_3$ es el conjunto de elementos más específicos en $\mathcal{R}_1 \cup \mathcal{R}_2$ y $\mathcal{L}_3$ es $\mathcal{L}_1 \cup \mathcal{L}_2$.

Observar que la restricción aquí es que ambos bordes tienen el conjunto de instancias negativas en común.

Intersección Dados un conjunto $\mathcal{D}_p$ de instancias positivas y dos conjuntos $\mathcal{D}_{n_1}$ y $\mathcal{D}_{n_2}$ de instancias negativas. Sean los bordes $\langle \mathcal{L}_1, \mathcal{R}_1 \rangle$ del espacio de JEPs de $\mathcal{D}_p$ con respecto a $\mathcal{D}_{n_1}$ y $\langle \mathcal{L}_2, \mathcal{R}_2 \rangle$ del espacio de JEPs de $\mathcal{D}_p$ con respecto a $\mathcal{D}_{n_2}$. La intersección de ambos bordes genera otro borde $\langle \mathcal{L}_3, \mathcal{R}_3 \rangle$ representando $[\mathcal{L}_1, \mathcal{R}_1] \cap [\mathcal{L}_2, \mathcal{R}_2]$ donde, $\mathcal{R}_3$ es $\mathcal{R}_1 \cap \mathcal{R}_2$ y $\mathcal{L}_3$ es el conjunto de elementos más generales de $\{A \cup B \mid A \in \mathcal{L}_1 \text{ y } B \in \mathcal{L}_2\}$[19].

En esta operación la restricción que opera sobre los bordes es que ambos bordes poseen el mismo conjunto de instancias positivas.

Inserción de nuevas Instancias

La inserción de nuevas instancias puede ocurrir tanto en $\mathcal{D}_p$ como en $\mathcal{D}_n$. Para notar este nuevo conjunto de instancias a insertar se utilizará el símbolo Δ, utilizando Δ_p y Δ_n para notar los conjuntos de nuevas instancias a insertar en el conjunto de instancias positivas $\mathcal{D}_p$ y en el conjunto de instancias negativas $\mathcal{D}_n$ respectivamente. Con $\mathcal{D}_p$ y $\mathcal{D}_n$ nos seguiremos refiriendo a los conjuntos de instancias antes de ser actualizados. A su vez,

[19] Es necesario eliminar de $\mathcal{L}_3$ aquellos elementos que no contengan a ningún elemento de $\mathcal{R}_3$.

representaremos el espacio horizontal de cada conjunto de datos por los bordes $\langle \{\emptyset\}, \mathcal{R}_p \rangle$ para el conjunto $\mathcal{D}_p$ de instancias positivas, $\langle \{\emptyset\}, \mathcal{R}_n \rangle$ para el conjunto $\mathcal{D}_n$ de instancias negativas, $\langle \{\emptyset\}, \mathcal{R}_p^{\Delta} \rangle$ para el conjunto de Δ_p y $\langle \{\emptyset\}, \mathcal{R}_n^{\Delta} \rangle$ para el conjunto de Δ_n.

Consideraremos la actualización del conjunto de instancias positivas y negativas por separado ya que cualquier actualización en ambos conjuntos puede descomponerse en una secuencia de actualizaciones a un conjunto y luego al otro.

Nuevas instancias positivas El nuevo espacio de JEPs con respecto a $(\mathcal{D}_p \cup \Delta_p)$ y $\mathcal{D}_n$ es representado por la unión de los conjuntos $\mathcal{D}_p$ y Δ_p realizando luego la diferencia con el conjunto $\mathcal{D}_n$,

$$\left([\{\emptyset\}, \mathcal{R}_p] \cup \left[\{\emptyset\}, \mathcal{R}_p^{\Delta}\right]\right) - [\{\emptyset\}, \mathcal{R}_n] = ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup \left(\left[\{\emptyset\}, \mathcal{R}_p^{\Delta}\right] - [\{\emptyset\}, \mathcal{R}_n]\right)$$

donde el primer término es el espacio de JEPs representado por el borde ya conocido (conocimiento previo) y el segundo término es el espacio con respecto a las nuevas instancias positivas Δ_p según el conjunto $\mathcal{D}_n$. Por consiguiente, la inserción de nuevas instancias se reducirá a calcular el borde de JEPs con respecto a Δ_p y $\mathcal{D}_n$ y luego realizar la unión con el borde del espacio de JEPs anterior. Cabe notar que el conjunto Δ_p usualmente es considerablemente más pequeño que el conjunto $\mathcal{D}_p$.

Nuevas instancias negativas El nuevo espacio de JEPs con respecto a $\mathcal{D}_p$ y $(\mathcal{D}_n \cup \Delta_n)$ en forma similar al caso anterior es representado por la unión de los conjuntos $\mathcal{D}_n$ y Δ_n realizando luego la diferencia del conjunto $\mathcal{D}_p$ con dicha unión.

$$[\{\emptyset\}, \mathcal{R}_p] - \left([\{\emptyset\}, \mathcal{R}_n] \cup \left[\{\emptyset\}, \mathcal{R}_n^{\Delta}\right]\right) = ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cap \left([\{\emptyset\}, \mathcal{R}_p] - \left[\{\emptyset\}, \mathcal{R}_n^{\Delta}\right]\right)$$

Nuevamente el primer término es el conocimiento previo y el segundo término es el espacio de JEPs con respecto solamente a las nuevas instancias negativas y $\mathcal{D}_p$.

Se puede apreciar en ambos casos que el trabajo a realizar es notablemente inferior que el cómputo desde cero de todo el espacio de JEPs. Simplemente se debe calcular un espacio más pequeño utilizando las nuevas instancias y realizar luego alguna operación con el borde anterior.

Eliminación de Instancias

Utilizaremos nuevamente la notación Δ_p y Δ_n para referirnos a las actualizaciones sobre los conjuntos $\mathcal{D}_p$ y $\mathcal{D}_n$ respectivamente. Sin embargo, Δ_p y Δ_n representarán las instancias a ser eliminadas de $\mathcal{D}_p$ y $\mathcal{D}_n$. Nuevamente consideraremos dos casos:

Eliminación de instancias positivas Este es el caso más simple ya que la actualización del espacio de JEPs $\langle \mathcal{L}, \mathcal{R} \rangle$ puede realizarse computando el nuevo espacio horizontal de $(\mathcal{D}_p - \Delta_p)$ generando un nuevo borde $\langle \{\emptyset\}, \mathcal{R}'_p \rangle$ y luego quitando de $\mathcal{L}$ todos aquellos itemsets que no sean cubiertos por algún itemset de $\mathcal{R}'_p$. El borde $\langle \mathcal{L}, \mathcal{R}'_p \rangle$ representará el espacio de JEPs con respecto a $(\mathcal{D}_p - \Delta_p)$ y $\mathcal{D}_n$.

Eliminación de instancias negativas Esta eliminación de instancias tiene como resultado que sea incrementado el tamaño del nuevo espacio de JEPs (no necesariamente la cardinalidad de los bordes). Para mostrar el mantenimiento necesario en forma simple realizaremos algunos renombres. Denotaremos $A = [\{\emptyset\}, \mathcal{R}_p]$ el espacio horizontal de $\mathcal{D}_p$ de instancias positivas, $B = [\{\emptyset\}, \mathcal{R}'_n]$ el espacio horizontal de $(\mathcal{D}_n - \Delta_n)$ y $C = [\{\emptyset\}, \mathcal{R}_{\Delta_n}]$ el espacio horizontal de Δ_n.

$$
\begin{aligned}
A - B &= A - B - (A \cap B) \\
&= A - B - (A \cap B) - C \cup (A \cap C) \\
&= (A - (B \cup C)) \cup (A \cap (C - B)) \\
&= ([\{\emptyset\}, \mathcal{R}_p] - ([\{\emptyset\}, \mathcal{R}'_n] \cup [\{\emptyset\}, \mathcal{R}_{\Delta_n}])) \cup (A \cap (C - B)) \\
&= ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup (A \cap (C - B))
\end{aligned}
$$

El primer término corresponde con el espacio de JEPs previo a la actualización y resta por calcular el espacio de JEPs resultante de $C - B$. Luego se procederá a

realizar las operaciones de unión e intersección correspondiente y se ajustará el borde izquierdo de la misma forma que en el caso de la eliminación de instancias positivas.

Es válido observar que la eliminación de instancias positivas poco tiene de actualización del conocimiento previo, ya que se realiza el cálculo del nuevo borde generando nuevamente el espacio horizontal de $\mathcal{D}_p$ (actualizado). Sin embargo, se realiza una actualización sobre el borde izquierdo evitando generar el espacio horizontal de instancias negativas y la correspondiente diferencia.

Otras operaciones

En [42, 47] también se proponen actualizaciones de los bordes para cuando se producen inserciones o eliminaciones de items a los conjuntos de datos. Este tipo de modificación requiere actualizar los bordes del espacio de JEPs previo ya que algunos patrones del borde podrían contener a dichos items y ya no son válidos o nuevos patrones deberían ser incorporados a los bordes. Afortunadamente estas actualizaciones también pueden ser realizadas en forma incremental. Por lo tanto, si consideramos estas modificaciones junto con las modificaciones propuestas anteriormente, se puede observar que todas las operaciones posibles a realizar sobre el conjunto de datos tanto $\mathcal{D}_p$ como $\mathcal{D}_n$ pueden ser realizadas en forma incremental mejorando así notablemente la eficiencia del método propuesto. Las operaciones propuestas además respetan las restricciones impuestas en las operaciones de bordes de unión, intersección y diferencia y por lo tanto aplicables.

4.8. Otros Patrones Emergentes

4.8.1. Patrones Emergentes Fuertes

Una de las propiedades más importantes en el área de minería de datos es la propiedad de anti-monotonía [2]. Como hemos mencionado tanto para EPs como para JEPs (pág. 166) estos espacios de patrones no exhiben la propiedad de clausura en subconjuntos[20]. Sin

[20]Ver definición 2.1.2 página 14

embargo, existe una clase especial de conjunto de patrones emergentes que respeta dicha propiedad, los cuales son llamados *patrones emergentes fuertes*.

Definición 4.8.1 (Dong & Li, 1999)
Para un umbral ρ dado para la tasa de crecimiento, un patrón emergente es llamado patrón emergente fuerte (strong emerging pattern) si todos sus subconjuntos no vacíos también son patrones emergentes. ♦

Una de las ventajas de utilizar este tipo de patrones es que son representables simplemente utilizando un borde. Por otro lado, la dificultad para detectar estos patrones es la misma que para buscar patrones emergentes (EP) y es probable que detectando únicamente patrones emergentes fuertes se dejarán de lado patrones significativos. Sin embargo, se pretende con dicha definición encontrar alguna propiedad que pueda ayudar a mejorar el estado actual de los mecanismos de minería para patrones emergentes.

4.8.2. Patrones Emergentes Extendidos

Otra clase especial de patrones emergentes la constituyen los patrones emergentes extendidos. A pesar de su nombre, estos patrones no son una extensión (superclase) de los patrones emergentes, sino por el contrario una subclase (subconjunto) de los mismos. Este nombre se debe a que estos patrones son una extensión de los JEPs que como mencionáramos en ciertos trabajos son llamados patrones emergentes [42].

Definición 4.8.2 (Patrones Emergentes Extendidos, Li et al.)
Dado un conjunto $\mathcal{D}_p$ de instancias positivas, un conjunto $\mathcal{D}_n$ de instancias negativas y dos números enteros δ_p y δ_n ($\delta_p > \delta_n \geq 0$). Un espacio de EP extendido (extended EP space) consiste de aquellos itemsets que ocurren al menos δ_p *instancias positivas y* a lo sumo δ_n *instancias negativas.* ♦

Como se puede observar estos patrones corresponden con los EP cuyo umbral de tasa de crecimiento es establecida por $\frac{\delta_p}{\delta_n}$. A su vez, extienden los JEPs ya que si $\delta_p = 1$ y $\delta_n = 0$ se genera el espacio de los JEPs. La búsqueda de estos patrones es realizada de la misma

forma que para los patrones emergentes, generando los respectivos bordes y realizando la diferencia entre los mismos. Como hemos visto en la sección 4.7.3, el mantenimiento del espacio de JEPs es muy simple. Esto no ocurre con los patrones emergentes, como tampoco con los patrones emergentes extendidos. En [42] se mencionan algunas consideraciones a tener en cuenta para el mantenimiento de estos espacios junto con las dificultades que podrían aparecer.

4.8.3. Patrones Emergentes Fronterizos, de Planicie y de Sombra

Se han propuesto varias clases de patrones emergentes los cuales según sus características pueden ser utilizados más eficientemente para una u otra tarea. A su vez, también se ha realizado una subclasificación dentro de los JEPs. En esta sección consideraremos una subclase especial de JEPs propuesta en [48] que intenta proveer más información sobre los JEPs obtenidos.

La primer subclase de JEP intenta capturar aquellos JEPs que mejor capturan la diferencia entre ambos conjuntos de datos $\mathcal{D}_p$ y $\mathcal{D}_n$. Por la propiedad de anti-monotonía sabemos que todo subconjunto de un itemset tiene que tener un soporte mayor o igual en el conjunto de datos al soporte del itemset.

Definición 4.8.3 (Boundary EP, Li et al.)
Un patrón emergente fronterizo (Boundary EP) es un JEP cuyos subconjuntos propios no son JEPs. ♦

Estos EP fronterizos tienen la particularidad de que cuando un item es quitado del patrón, dejan de ser JEP. Esto es posible únicamente si dejan de tener un soporte vacío en el conjunto de $\mathcal{D}_n$. Por lo tanto, estos patrones forman la frontera entre los JEP y los que no lo son. Además todos los otros JEPs tienen un soporte inferior en consecuencia estos EP fronterizos capturar mejor la diferencia entre los conjuntos de datos.

Otro subclase de JEPs que está relacionado a los EP fronterizos son los *EP de planicie*. Estos patrones son calificados de tan útiles como los patrones emergentes fronterizos en

la determinación de las diferencias más significativas entre conjuntos de datos.

Definición 4.8.4 (Plateau EP, Li et al.)
Dado un patrón emergente fronterizo, todos sus superconjuntos que tienen el mismo soporte son llamados sus EPs de planicie (Plateau EPs). ♦

Además el conjunto de EPs de planicie forman un espacio que también es convexo y por lo tanto representable con bordes.

Definición 4.8.5 (Espacio de Plateau EP, Li et al.)
Todos los EPs de planicie de todos los EPs fronterizos con la misma frecuencia son llamados un espacio de EPs de planicie (Plateau Space)(P-Space). ♦

Por último, hemos notado que todo subconjunto de un patrón emergente fronterizo no es JEP por lo tanto su soporte en $\mathcal{D}_n$ es distinto de cero. Estos itemsets aún cuando no son JEPs son útiles para medir el grado de interés de un patrón fronterizo.

Definición 4.8.6 (Patrones de Sombra, Li et al.)
Todos los subconjuntos inmediatos de un EP fronterizo son llamados patrones de sombra (Shadow Patterns). ♦

Según [48] si la tasa de crecimiento de los patrones de sombra se acerca a $+\infty$ entonces es razonable la existencia del EP fronterizo, en cambio si la tasa de crecimiento es pequeña entonces el EP fronterizo era no esperado y por lo tanto poco interesante.

4.9. Utilización de Hipergrafos

Los hipergrafos han sido utilizado en muchas aplicaciones en ciencias de la computación. Entre ellas podemos mencionar diagnóstico minimal y circumscripción proposicional [62], aprendizaje de fórmulas booleanas [10] y teoría de conmutación booleana [24]. En minería de datos, la teoría de hipergrafos está relacionada a la búsqueda de patrones

maximales[21] y a la búsqueda de itemsets no frecuentes minimales[22] [33].

Hemos visto que los patrones emergentes por su parte se relacionan con los patrones maximales. Necesitan de éstos para la generación de los bordes. Los patrones emergentes también se relacionan con los patrones no frecuentes minimales, ya que el objetivo de los EPs encontrar patrones minimales contrastantes entre los conjuntos de datos. Debido a esta relación, existe una relación natural entre los patrones emergentes (en particular de los JEPs) y los hipergrafos la cual es utilizada en [4].

Un hipergrafo se define como un par (V, E), donde V es el conjunto de vértices y E es el conjunto de arcos, donde cada arco es un subconjunto de V. Es importante mencionar que a diferencia de los grafos en general donde cada arco es un par de vértices, los arcos en los hipergrafos (también llamados hiperarcos) pueden conectar más de un vértice[23]. Un *recorrido* (traversal) de un hipergrafo es cualquier conjunto de vértices que contiene al menos un elemento de cada arco. Un *recorrido minimal* (minimal traversal) es un recorrido para la cual ninguno de sus subconjuntos propios es también un recorrido.

Veamos el siguiente ejemplo, extraído de [4][24], que muestra como pueden ser utilizados los hipergrafos para la búsqueda de JEPs.

Ejemplo 4.9.1

Sean los conjuntos de datos $\mathcal{D}_p = \{\{bdgj\}, \{cegj\}, \{cfhj\}, \{afhj\}\}$ *de instancias positivas y* $\mathcal{D}_n = \{\{adgj\}, \{adgi\}, \{cfhi\}, \{aegj\}\}$ *de instancias negativas. Los JEPs minimales de* $\mathcal{D}_p$ *con respecto a* $\mathcal{D}_n$ *son* $\{\{b\}, \{ce\}, \{cg\}, \{cj\}, \{fj\}, \{hj\}, \{af\}, \{ah\}\}$. *Descomponiendo el problema podemos identificar los JEPs en cada conjunto de* $\mathcal{D}_p$, *del conjunto* $\{bdgj\}$ *con respecto a* $\mathcal{D}_n$ *el JEP* $\{b\}$; *de* $\{cegj\}$ *con respecto a* $\mathcal{D}_n$ *los JEPs* $\{\{ce\}, \{cg\}, \{cj\}\}$; *de* $\{cfhj\}$ *con respecto a* $\mathcal{D}_n$ *los JEPs* $\{\{cj\}, \{fj\}, \{hj\}\}$ *y de* $\{afhj\}$ *con respecto a* $\mathcal{D}_n$ *los JEPs* $\{\{af\}, \{ah\}, \{fj\}, \{hj\}\}$.

Para cada conjunto en $\mathcal{D}_p$ *se generará un hipergrafo donde los vértices del mismo*

[21] Ver sección 3.2 página 60.

[22] Ver sección 3.4 página 122.

[23] Observese que también un hiperarco puede estar constituido por el conjunto vacío $\emptyset$ o también por un conjunto de un solo vértice (este último caso es representable mediante un par (v_i, v_i)).

[24] El ejemplo ha sido modificado levemente en su notación para continuar con la nomenclatura hasta aquí utilizada.

corresponderán a los elementos de dicho conjunto y los arcos se definirán con respecto a $\mathcal{D}_n$*. Por cada conjunto de* $\mathcal{D}_n$ *se definirá un hiperarco que constará de sustraer al conjunto elegido de* $\mathcal{D}_p$ *los elementos presentes en el conjunto de* $\mathcal{D}_n$*. Así, por ejemplo, para* $\{cfhj\} \in \mathcal{D}_p$ *se forma un hipergrafo con* $V = \{c, f, h, j\}$[25] *y* 4 *hiperarcos, hiperarco* 1 $\{cfhj\} - \{adgj\} = \{cfh\}$*, hiperarco* 2 $\{cfhj\} - \{adgi\} = \{cfhj\}$*, hiperarco* 3 $\{cfhj\} - \{cfhi\} = \{j\}$*, hiperarco* 4 $\{cfhj\} - \{aegj\} = \{cfh\}$*. Las* 3 *travesías minimales de este hipergrafo son* $\{\{cj\}, \{fj\}, \{hj\}\}$ *que corresponden con la anteriormente mencionada. Observese que como uno de los hiperarcos posee solo un elemento, el cual debe estar en todos los recorridos y por lo tanto en todos los JEPs con respecto a ese conjunto de* $\mathcal{D}_p$*.*$\Diamond$

Esta conexión de los patrones emergentes con los hipergrafos a pesar de ser muy interesante sigue teniendo los problemas de los acercamientos previos ya que aún continua abierto el problema de establecer un método eficiente para encontrar travesías minimales en un hipergrafo. Sin embargo, esta conexión es interesante ya que cualquier aporte en ambos acercamientos producirá beneficios múltiples.

4.10. Comparaciones con otros acercamientos

4.10.1. Reglas discriminantes vs EPs

Muchos otros acercamientos intentan capturar cambios y diferencias entre conjuntos de datos. En particular uno de los primeros métodos puede encontrarse en [36], donde se define una clase particular de reglas, denominadas reglas discriminantes. Las reglas discriminantes están orientadas a capturar los conceptos que diferencian a un clase destino de una clase contrastante. Para ello se utiliza junto con las reglas una medida cuantitativa, denominada peso discriminante, que da la frecuencia de una regla en la clase destino con respecto a la clase contrastante. A su vez, en [36], se compara estas reglas con las técnicas de clasificación de machine learning basados en árboles de decisión ID3 [59, 60] y C4.5 [61].

Puede apreciarse la similitud con los EPs, ya que una regla no es más que un patrón con cierta información. Sin embargo, existe una diferencia fundamental entre estos métodos.

[25] Utilizando la notación estándar de conjuntos.

En el método propuesto por [36] las reglas o patrones encontrados tienen un soporte alto en los conjuntos de datos donde son descubiertos. Por otro lado, los patrones emergentes no se basan en el soporte sino en la relación entre los soportes de ambas clases, por lo cual un patrón con bajo soporte en ambas clases pero con una relación superior al umbral sería identificado por el método de patrones emergentes pero no sería detectado como una regla discriminante. Hemos visto que estos patrones de bajo soporte son fundamentales en muchos dominios de aplicación como una detección temprana de una tendencia o en la búsqueda de anomalías. Por lo pronto, podemos decir que este acercamiento de patrones emergentes extiende la propuesta de patrones discriminantes, generando clasificadores más exactos.

4.10.2. Version Spaces vs JEP

El espacio de JEPs tiene cierta similitud con el concepto de *espacio de versiones* (version spaces) [53]. Sin embargo como se nota en [42] existen diferencias importantes entre ambos métodos. La diferencia entre ambos conceptos puede verse contrastando las definiciones de espacio de JEPs y espacio de versiones. A tal efecto, se proporciona esta última a continuación,

Definición 4.10.1 (Espacio de Versiones, Li et al.)
Dados un conjunto $\mathcal{D}_p$ de instancias positivas y un conjunto $\mathcal{D}_n$ de instancias negativas, el espacio de versiones (version space) con respecto a $\mathcal{D}_p$ y $\mathcal{D}_n$ es el conjunto de todos los itemsets cuyo soporte en $\mathcal{D}_p$ es $100\,\%$ y cuyo soporte en $\mathcal{D}_n$ es $0\,\%$. ♦

Puede observarse que la diferencia entre ambos es la restricción del espacio de versiones de considerar solamente aquellas instancias positivas que tengan un soporte del $100\,\%$ cuando en el espacio de JEPs son considerados todos aquellas instancias que tengan un soporte distinto de $0\,\%$. Consecuentemente, podríamos decir que el espacio de JEP es una generalización del concepto de espacios de versiones. A su vez podemos ver que el espacio de versiones está contenido dentro del espacio de JEPs como este último está contenido dentro del espacio de EPs.

A pesar de estas diferencias, existen muchas similitudes entre ambos conceptos. A nivel de estructura, ambos métodos representan sus elementos (instancias en el caso del espacio de JEPs e hipótesis en el caso de espacio de versiones) dentro de un reticulado. Este establece un orden parcial entre los elementos a partir de un operador relacional que permite determinar la posición de un elemento dentro del reticulado. Esta relación de orden en ambos métodos está dada por los conceptos de generalidad y especificidad, permitiendo así diferenciar los elementos más específicos y los elementos más generales del conjunto de instancias o hipótesis.

Otra característica común en ambos métodos es que el espacio de elementos representado es un espacio convexo[26] y por lo tanto es representable en forma concisa a través de un par de conjuntos fronterizos que corresponden a los elementos más específicos y a los elementos más generales. Además en ambos métodos se realizan operaciones sobre estos conjuntos fronterizos en lugar de aplicar las modificaciones sobre cada elemento representado.

Es importante notar también se han propuesto para ambos métodos algoritmos que realizan el mantenimiento de estos conjuntos convexos de forma incremental como puede apreciarse en la sección 4.7.3. Sin embargo es necesario resaltar un punto de principal importancia. En muchas de las aplicaciones reales se ha detectado que el espacio de versiones resulta en un conjunto vacío cuando los espacios de JEPs contienen varias características discriminantes. Por ejemplo, en [42] se reporta que en la mayoría de las bases de datos de *UCI Machine Learning Repository* los espacios de versiones corresponden a un conjunto vacío. En particular, en la base de datos de hongos, con clases venenosos y comestibles, los espacios de JEPs de la clase venenoso a comestible está representado con un borde de 1606 itemsets como frontera izquierda y 4208 itemsets como frontera derecha. En la base de datos tic-tac-toe, el espacio de JEPs es representado por un borde con 1592 itemsets en la frontera izquierda y 626 itemsets en la frontera derecha. En ambos casos el espacio de versiones resultó vacío.

[26] Ver definiciones 4.5.4 (página 144) y 4.5.6 (página 145).

4.10.3. Disjuntive Version Spaces vs JEP

El espacio de JEPs está intimamente relacionado al concepto de *espacio de versiones disyuntivo* (disjunctive version spaces o DiVS) [68]. Este método extiende el concepto de espacio de versiones agregando el operador de disyunción (OR). Básicamente, dado un conjunto $\mathcal{D}_p$ de instancias positivas y un conjunto $\mathcal{D}_n$ de instancias negativas, un espacio de versiones disyuntivo es la disyunción de los espacios de versiones $H(Ex)$ para cada instancia $Ex \in \mathcal{D}_p$. Donde $H(Ex)$ es la conjunción de los conjuntos de hipótesis que cubren Ex y discriminan Ce (instancia negativa), notada $D(Ex, Ce)$, para $Ce \in \mathcal{D}_n$. El DiVS es representado por

$$DiVS = \bigvee_{Ex \in \mathcal{D}_p} \left(\bigwedge_{Ce \in \mathcal{D}_n} D(Ex, Ce) \right)$$

Al igual que el espacio de JEPs, los DiVS son generalizaciones de los espacios de versiones. De hecho, en [42] se argumenta que los DiVS describen precisamente el espacio de JEPs. Sin embargo, a pesar de describir el mismo espacio la forma en que realizan las operaciones sobre tal espacio es de fundamental importancia y marca la diferencia entre ambos métodos.

En particular, a pesar que ambos métodos utilizan una representación concisa de los elementos del espacio, los DiVS no realizan un mantenimiento incremental del espacio. Por consiguiente, es ineficiente con respecto a los métodos aplicados en los espacios de JEPs. Además, la representación utilizada por los DiVS, a pesar de ser concisa, frecuentemente es poco entendible. En [42] se muestra una comparación entre ambas representaciones, pudiendo apreciar allí la poca legibilidad de este método con respecto a los bordes del espacio de JEPs. Por último, aún cuando ambos métodos han sido aplicados a problemas de clasificación, en [42, 43] se reporta una mayor exactitud en los clasificadores basados en JEPs con respecto a los basados en DiVS.

4.10.4. Otros tipos de bordes vs JEP

El concepto de bordes no es un concepto nuevo, ya que como mencionáramos ha sido utilizado en machine learning desde espacios de versiones [53, 54, 40, 68, 71] entre otros.

Más recientemente, en [51] se ha utilizado este concepto para descubrir reglas interesantes utilizando bordes positivos y negativos. Sin embargo, la diferencia principal con estos métodos es que los bordes utilizados respetan la propiedad de clausura de subconjuntos[27] cuando los bordes utilizados por los espacios de JEPs respetan la propiedad de intervalo cerrado o espacio convexo. Como ventaja de los métodos que utilizan conjuntos que respetan la clausura de subconjuntos podemos mencionar que basta utilizar un solo borde para representar todo el espacio. Sin embargo, en [51] se propone utilizar bordes negativos y positivos así que por cada uno se necesitará un borde. Por otro lado, los conjuntos convexos son una generalización de los anteriores por lo que si un conjunto es clausurado en subconjunto esto implica que es un espacio convexo[28] por lo cual los espacios de JEPs pueden ser aplicados a un mayor número de dominios.

También como hemos mencionado en secciones previas los bordes han sido utilizado por [34] para dominios como ATMS y espacios de versiones. Aquí también son utilizados conjuntos clausurados en subconjuntos pero también son utilizados conjuntos convexos. El dominio donde son aplicados estos métodos y el conjunto de operaciones constituyen la principal diferencia entre ambos acercamientos. En [34] se describen más operaciones que las descritas para espacios de JEPs, si bien la operación de diferencia de [42, 43] es completamente diferente.

4.11. Conclusiones

Los patrones emergentes con alto soporte y con muy pocos atributos son utilizados en forma diaria. A menudo, estos son utilizados en reportes diarios de noticias, por ejemplo, existen muchas noticias que reportan hechos como que el número de hogares con acceso a internet a crecido un 90 %. También, estos son obtenidos para proveer información con valor científico, como por ejemplo el patrón emergente que establece que la incidencia de cáncer de pulmón entre fumadores es mucho más alta que entre los no fumadores.

En este capítulo se ha introducido la noción de patrones emergentes y ha sido mostra-

[27] Ver definición 4.5.1 página 140.
[28] No se verifica la situación inversa, es decir, si $A \rightarrow B$ no necesariamente $B \rightarrow A$.

do una forma eficiente de calcular los mismos sacando provecho de las propiedades de los espacios convexos para representarlos en forma concisa, evitando de esta forma enumerar el conjunto de EPs. Además se provee una operación para manipular estos conjuntos sin necesidad de la enumeración, permitiendo de esta forma que el cómputo de los patrones emergentes sea viable aún con umbrales de mínimo soporte muy bajos. Sin embargo, la principal desventaja del método propuesto es que éste simplemente obtiene un subconjunto del conjunto de EPs dejando de lado gran cantidad de EPs potencialmente valiosos.

A partir de los patrones emergentes han surgido varias clases de patrones emergentes, de los cuales la clase más prominente son los JEPs. Estos JEPs están caracterizados por poseer una tasa de crecimiento muy abrupta (∞) y ha sido utilizados especialmente en tareas de clasificación, donde han demostrado superar en exactitud de predicción a clasificadores como ID3 y C4.5. Se ha mostrado, además, como esta clase especial de patrones emergentes pueden ser eficientemente actualizados en forma incremental, aprovechando de esta forma el trabajo previamente realizado. Luego, hemos mencionado brevemente las distintas clases de patrones emergentes propuestos. Por último, hemos analizado las posibles relaciones entre la propuesta de patrones emergentes con otros acercamientos similares

Capítulo 5

Minería Incremental de Patrones Emergentes

Existen muchas clases de patrones interesantes que pueden ser extraídos a partir de los datos. Una clase particularmente útil son los llamados *patrones emergentes* (emerging patterns o EPs). Los EPs son itemsets cuyo soporte crece significativamente de un conjunto de datos a otro. Ellos son especialmente útiles para indicar cambios y diferencias entre conjuntos de datos, como también para capturar tendencias emergentes cuando éstos son aplicados a bases de datos temporales.

En [20] se menciona que EPs con soporte bajo o medio pueden dar una mejor percepción y orientación a expertos aún sobre dominios que se creían bien entendidos. Como un ejemplo de una tendencia descubierta recientemente que apareció como un artículo establecía "*Una matriculación baja (low tuition) y los altos estándares atraen a estudiantes estadounidenses a estudiar en Canadá* " (*Dayton Daily News*, 10/6/2002). Esta tendencia emergente de estudiantes estadounidenses trasladándose a estudiar a universidades canadienses corresponde un incremento del 85 % durante 3 años sobre un total de 5000 inscripciones. Esta tendencia interesante es un patrón emergente con bajo soporte ($\frac{5000}{N}$, donde N es el total de estudiantes inscriptos en Canadá) pero con una gran tasa de crecimiento (1,85).

En nuestra opinión, los métodos utilizados para la búsqueda de esta clase de información acerca de los datos ha sido aplicada exitosamente solo sobre un espacio de búsqueda restringido. Estos métodos son muy eficientes para la búsqueda de subclases de EPs o

incluso para la búsqueda de un subconjunto de EPs pero no son tan eficientes cuando el objetivo es obtener el conjunto de todos los EPs o al menos una buena aproximación. En la próxima sección veremos que algunas propuestas que intentan solucionar este problema utilizando los acercamientos existentes. Nuestro objetivo es desarrollar un nuevo método que aproxime el conjunto de EPs pero que saque provecho de información previamente computada, requiriendo de esta forma menos accesos a la base de datos, utilizando un acercamiento incremental.

5.1. Acercamientos Propuestos

En la sección 4.6.2 se mencionó la forma se puede descomponer el espacio de EPs para realizar una búsqueda eficiente de un subconjunto de EPs. El espacio de EPs se descompone entonces en tres zonas como se puede ver en la figura 4.7 de las cuales solo se realiza la tarea de minería solo sobre la zona 1. Allí se propone una extensión para buscar EPs en la zona 2 (ΔGDE). Esta zona consiste de los EPs que superan el umbral de mínimo soporte en ambos conjuntos de datos. Sin embargo, se evita realizar la búsqueda sobre la zona 3 (ΔABG) porque éste consiste de un gran número de patrones y no se ha encontrado un algoritmo eficiente para resolver este desafío. Se realiza, además, otra propuesta para intentar cubrir aún más el espacio de EPs y aproximar mejor dicho espacio. Tal acercamiento busca reducir el espacio ocupado por las zonas 2 y 3 realizando múltiples búsquedas de la zona 1. Ambos acercamientos utilizan como base la operación de diferencia de bordes tal como fuera mencionada en la sección 4.6.3.

5.1.1. Búsquedas Recursivas de EPs sobre ΔGDE

La primer propuesta se basa en realizar la búsqueda de EPs sobre el rectángulo $BCDG$ (zona 1) que se encuentra enmarcado en el triángulo ΔACE. Esta búsqueda, como ha sido mencionado, descompone el espacio en tres sectores. Uno de estos sectores es el triángulo ΔGDE, el cual puede verse como un nuevo espacio de EPs sobre el cual se puede realizar la búsqueda. Por tal motivo, la propuesta consiste en aplicar recursivamente el

algoritmo de búsqueda de EPs sobre esta zona, como si este triángulo ΔGDE fuera el triángulo ΔACE. Este acercamiento puede verse gráficamente en la figura 5.1.

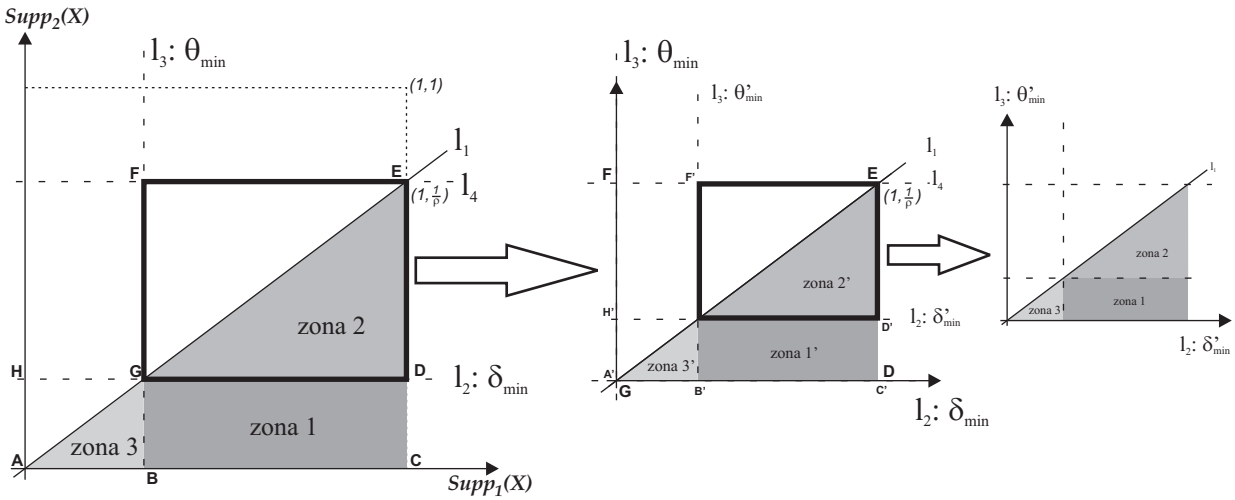


Figura 5.1: Búsquedas Recursivas de EPs

Este acercamiento puede ser utilizado para aproximar todo el triángulo ΔACE utilizando un punto G cercano al origen. Este método, sin embargo, sufre de dos desventajas:

1. *No toma en cuenta la información existente de búsquedas previas.* Recordemos que para buscar patrones en el rectángulo $BCDG$ se realiza la diferencia de dos bordes que están representados por los conjuntos maximales de ambos conjuntos de datos. Es importante notar que al incrementar el soporte de mínimo umbral en ambos conjuntos de datos estamos restringiendo los patrones que pueden estar comprendidos en ambos bordes. Por tal motivo, sería interesante utilizar esta información sobre los patrones maximales existentes y restringir este conjunto en lugar de comenzar sin ninguna información al respecto. Esta reutilización de información, como hemos visto en la sección donde se aplican algunas estrategias de poda, reduce significativamente el espacio de búsqueda mejorando en muchos ordenes de magnitud la eficiencia del algoritmo. Además al aplicar repetitivamente esta estrategia, en cada etapa recursiva el cómputo de bordes se reduciría con respecto a la etapa anterior.

2. *No es claro como se realiza la proyección de los conjuntos de datos, ni como esta*

proyección puede ser obtenida eficientemente. Nótese que para la búsqueda recursiva es importante utilizar conjuntos de datos más pequeños, es decir una proyección de los conjuntos de datos los cuales contienen los itemsets que superan θ_{min} en $\mathcal{D}_1$ y δ_{min} en $\mathcal{D}_2$. Sin embargo, la propuesta no especifica la utilización de un conjunto de datos proyectado, por lo tanto, no queda claro si ésta se realizará y en caso afirmativo, de que forma, o se utilizará ambos conjuntos de datos completos en cada paso recursivo.

5.1.2. Búsquedas Múltiples de EPs sobre ΔACE

La segunda propuesta consiste en realizar múltiples búsquedas del rectángulo $BCDG$ utilizando diferentes umbrales de tasas de crecimiento ρ, sacando provecho a la relación $\delta_{min} \times \rho = \theta_{min}$. Por ejemplo, supongamos que $\delta_{min} = \sigma$ y $\theta_{min} = \sigma \times \rho$ entonces se puede formar un rectángulo $BCDG$ en el plano 2-D utilizando los siguientes puntos $B = (\sigma \times \rho, 0)$, $C = (1, 0)$, $D = (1, \sigma)$, $G = (\sigma \times \rho, \sigma)$. Como ρ es un número fijo y $\sigma \in [0, 1]$ se puede tomar una secuencia de valores para σ de forma tal que se pueda aproximar el conjunto de EPs. Esta estrategia se ilustra en la figura 5.2.

Esta alternativa tiene la ventaja, con respecto a la propuesta anterior, que no necesita realizar una proyección del conjunto de datos, ya que simplemente ser realiza una modificación de los umbrales de mínimo soporte en cada conjunto de datos. Sin embargo, esta propuesta también tiene sus deficiencias:

1. *No considera información existente de búsquedas previas*. En forma análoga a la propuesta anterior, no considerar cómputos previamente realizados tiene por consecuencia que dichos cómputos deberán ser realizados en forma repetitiva para cada área considerada. Por tal motivo la performance del algoritmo de búsqueda se ve severamente degradada.

2. *La intersección entre los conjuntos de EPs de cada paso no es vacía*. Como se puede observar en la figura 5.2, cada rectángulo $BCDG$ que es obtenido se superpone con los rectángulos de pasos previos. Como consecuencia, el conjunto de EPs resultante

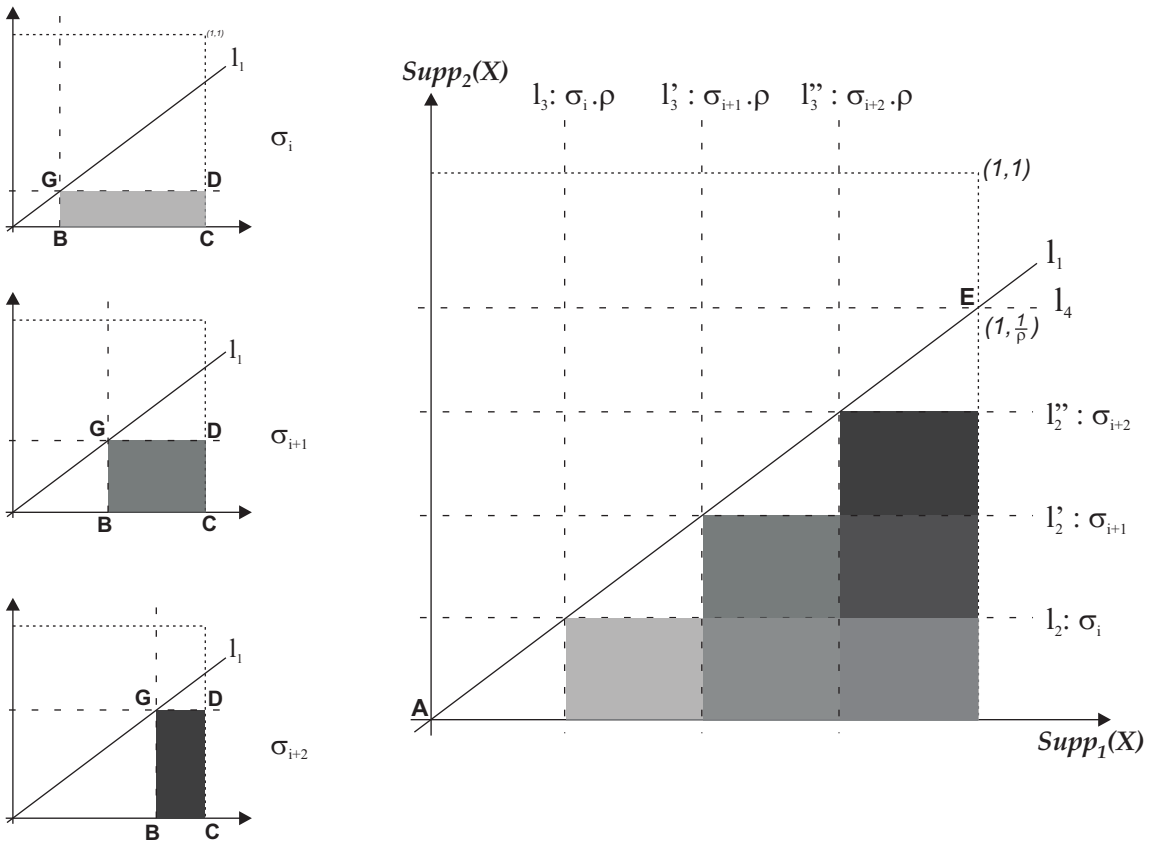


Figura 5.2: Múltiples búsquedas de EPs

estará compuesto por la unión de todos los rectángulos resultantes previa eliminación de EPs duplicados.

5.2. Acercamiento Incremental

Nuestra propuesta apunta a realizar una búsqueda del conjunto de EPs realizando una aproximación al mismo de igual forma que los acercamientos anteriormente mencionados. Sin embargo a diferencia de éstos, nuestra propuesta tiene como objetivo principal utilizar toda la información existente que pueda ser de ayuda para evitar cómputos innecesarios. Además proponemos particionar el espacio de EPs en subespacios de forma tal que cada uno de ellos consista de una cantidad tratable de itemsets.

La propuesta se basa en la observación que el espacio de EPs puede ser pensado como

la enumeración de los itemsets que conforman el área que se encuentra por debajo de la recta $l_1 : supp_2(X) = \rho \times supp_1(X)$ (triángulo ΔACE). Nuestra propuesta consiste en dividir el triángulo ΔACE en k rectángulos iguales de longitud $\lambda \in [0, 1]$, buscando EPs en cada uno de estos rectángulos disjuntos. Nuestra propuesta será análoga al acercamiento denominado *suma de Riemann* aproxima el área debajo de una función continua en un plano 2-D calculando la sumatoria de las áreas de múltiples rectángulos adyacentes. Otra ventaja de esta estrategia es que los rectángulos definidos son disjuntos a diferencia de la propuesta mencionada en la sección 5.1.1.

Con motivo de aprovechar mejor la información obtenida proponemos generar los rectángulos comenzando desde $supp_1(X) = 1$ hasta $supp_1(X) = 0$. Sin embargo, es interesante y posible explorar también la generación de rectángulos en forma creciente aunque de esta forma la reutilización de información y aplicación de heurísticas se ve reducida. A pesar de ello, esta generación creciente de rectángulos es una mejor alternativa que las propuestas anteriormente ya que existe una reutilización de información y un mejoramiento en la performance. A estos rectángulos los llamaremos *sectores* y estarán numerados desde 1 a k tal como se aprecia en la figura 5.3. La idea es generar la colección de patrones maximales (necesaria para la formación de bordes sobre los cuales se realizará la diferencia) del sector $i-1$ a partir de los patrones maximales del sector i para cualquier $1 \leq i \leq k$. De esta forma los patrones maximales previamente calculados podrán ser utilizados en su totalidad ya que al disminuir el soporte estos aún continúan siendo frecuentes aunque posiblemente no maximales. En la figura 5.3, el punto G tiene coordenadas $(1-\delta, \frac{1-\delta}{\rho})$ y las rectas que determinan los soportes mínimos para cada conjunto de datos están dadas por las siguientes ecuaciones:

$$\begin{aligned} supp_1(X) = \theta^i_{min} &= (i-1)\lambda \\ supp_2(X) = \delta^i_{min} &= \frac{1-(k-i+1)\lambda}{\rho} \end{aligned}$$

Analizaremos a continuación como se realizará la generación de EPs en cada sector. Una vez que hemos generado los EPs pertenecientes a cada sector podemos asegurar que éstos no se encuentran duplicados por lo tanto la unión de todos estos sectores conforma

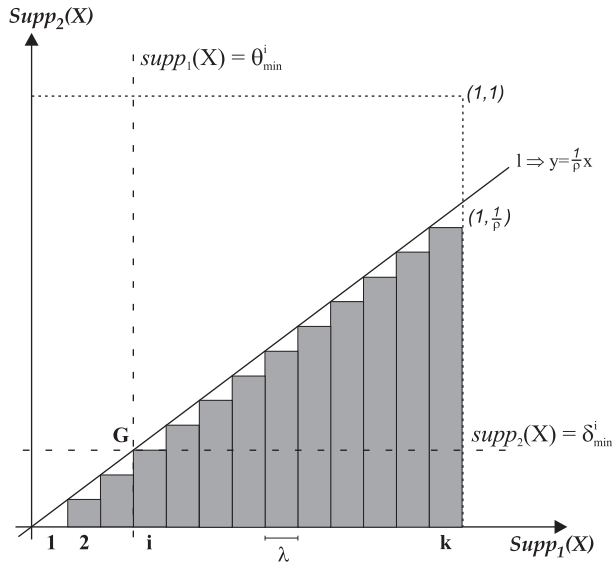


Figura 5.3: Descomposición del triángulo ΔACE en *sectores*

una aproximación al conjunto completo de EPs. Este paso de unión de sectores será analizado en la sección 5.6.

5.2.1. Búsqueda de EPs en el sector k

En primer lugar, se realizará la búsqueda de patrones emergentes pertenecientes al sector k sobre el cual no se posee información alguna. Por tal motivo estos EPs será extraídos utilizando el algoritmo mencionado en la sección 4.6.3. Podemos identificar dos pasos importantes de este algoritmo:

1. *Búsqueda de patrones maximales*. Estos patrones formarán parte de los bordes de ambos conjuntos de datos.

2. *Diferencia de bordes*. Se genera el conjunto de EPs realizando la diferencia entre los bordes computados en el paso anterior.

Esta información sobre los patrones maximales y EPs (representados en forma concisa por un borde) será reutilizada para la búsqueda de los patrones maximales del sector $k-1$. De esta forma, la generación de los bordes para el sector $k-1$ consistirá simplemente de continuar la búsqueda de patrones maximales utilizando un soporte mínimo más bajo.

Es importante mencionar que este paso, aún cuando comienza sin información alguna, tiene la ventaja que se utiliza un soporte bastante alto y por lo tanto la cantidad de patrones que cumplirán dichas restricciones son pocos y por la propiedad de anti-monotonía también sabemos que dichos patrones serán relativamente cortos y por lo tanto podrán ser ubicados fácilmente por cualquiera de los algoritmos mencionados en la sección 3.2.

5.2.2. Búsqueda incremental de EPs

En forma general, la tarea de minería para cualquier sector $i \in [1, k-1]$ se realizará utilizando los patrones maximales del sector $i+1$. A diferencia del sector k que realiza la tarea de minería exclusivamente a partir de los conjuntos de datos, la minería del sector $i \in [1, k-1]$ utilizará la información previa para mejorar la eficiencia de la minería tanto en espacio como en tiempo. Como ya ha sido mencionado, los algoritmos de minería de EPs necesitan de los patrones maximales de ambos conjuntos de datos para realizar su tarea. Estos patrones maximales pueden ser recalculados a partir de los conjuntos de datos actualizando los umbrales, o ajustando simplemente los patrones maximales obtenidos previamente a los nuevos umbrales, siendo esta última la idea principal de nuestro acercamiento.

Consideremos los pasos necesarios para obtener el conjuntos de EPs para un sector arbitrario $i-1$ con $i \in [1, k]$. Podemos suponer que se posee información sobre los patrones maximales y EPs obtenidos en el sector i, ya que si consideramos como caso base que $i = k$ entonces estaríamos considerando el sector $k-1$ y se contaría con información del sector k, el cual hubiera sido procesado utilizando el algoritmo estándar. La minería de EPs del sector $i-1$, que se ilustra en la figura 5.4, procederá de la siguiente forma:

1. Calcular los patrones maximales con soportes $supp_1(X) = \theta_{min}^{i-1}$ y $supp_2(X) = \delta_{min}^{i-1}$ en forma incremental.

2. Generar el borde B_2 a partir de los patrones maximales de $\mathcal{D}_2$ obtenidos en el paso anterior.

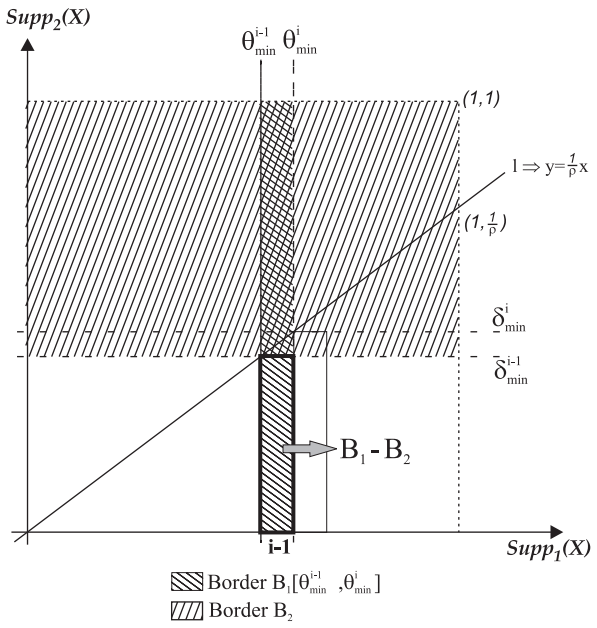


Figura 5.4: Búsqueda de EPs incremental

3. Generar el borde $B_1[\theta^{i-1}_{min}, \theta^i_{min}]$ a partir de $\mathcal{D}_1$ donde cada itemset $X \in B_1$ satisface que $supp_1(X) \in [\theta^{i-1}_{min}, \theta^i_{min}]$. Es decir que, el borde B_1 será generado utilizando los patrones maximales obtenidos para el sector i como frontera derecha y el nuevo conjunto de patrones maximales conformando la frontera izquierda. Formalmente, $B_1[\theta^{i-1}_{min}, \theta^i_{min}] = \langle \mathrm{MFI}_{\theta^i_{min}}, \mathrm{MFI}_{\theta^{i-1}_{min}} \rangle$.

4. Por último, se aplica la operación de diferencia sobre los bordes obtenidos, es decir, $B_1[\theta^{i-1}_{min}, \theta^i_{min}] - B_2$.

Estos pasos que conforman la columna vertebral de nuestro acercamiento, poseen algunas dificultades para su implementación. En primer lugar, analizaremos la generación de patrones maximales en forma incremental en sección 5.4. En segundo lugar, la generación de bordes, requerida en los pasos 2 y 3, se realizará como ha sido explicada oportunamente en la sección 4.6.3. Por último, la operación de diferencia del último paso requiere la utilización de un borde (B_1) cuya frontera izquierda es distinta de $\emptyset$, en consecuencia la operación de diferencia tal como fuera descrita en la sección 4.5.3 no es aplicable. Por tal motivo, en la sección 5.5 extenderemos la operación de diferencia para un caso más

general, es decir, cuando el borde B_1 es genérico.

La idea de utilizar información existente sobre patrones maximales para mejorar la performance de los algoritmos cuando se modifican los umbrales de mínimo soporte o de mínima confianza, ha sido explorada exitosamente en [77, 78, 79]. Estos acercamientos proponen mantener actualizados los patrones maximales en entornos de bases de datos evolutivas [63, 26, 82, 84, 35, 28] utilizando técnicas de minería incrementales, interactivas y de procesamiento paralelo. Su principal objetivo es reducir el tiempo y espacio requerido para actualizar el conjunto de patrones frecuentes cuando se produce una incorporación de nuevos datos o si se modifican los parámetros definidos por el usuario (por ejemplo, confianza y soporte) en forma arbitraria.

Sin embargo, en dichos trabajos se explora simplemente la generación de nuevos patrones maximales. Nuestro acercamiento difiere sustancialmente de dicho trabajo ya que no estamos interesados en obtener simplemente patrones maximales sino obtener los mismos para generar el par de bordes que nos permita obtener el conjunto de EPs en forma incremental. Por otro lado, en nuestro acercamiento no es necesario considerar la modificación de cualquier parámetro del algoritmo de minería de patrones maximales, ya que simplemente consideraremos la modificación de umbral de mínimo soporte y simplemente se disminuirá gradualmente el mismo. Esto permite mejorar aún más el algoritmo de búsqueda de patrones maximales para un caso más específico. Sin embargo, se debe notar que los acercamientos más generales aún pueden ser utilizados.

5.3. Sectores y Bordes: Propiedades

Daremos a continuación una serie de definiciones y teoremas con el objeto de proveer un marco de trabajo para la extensión de la operación de diferencia. En primer lugar, mostraremos como se pueden computar los límites que definen cada sector en función del parámetro λ que determina el ancho de cada sector.

Definición 5.3.1 (Fronteras)

Cada sector i está delimitado por cuatro umbrales o fronteras*:*

1. *Inferior: Definido por la línea* $Supp_2(X) = 0$.

2. *Superior: Definido por la línea* $Supp_2(X) = \delta^i_{min}$.

3. *Izquierdo: Definido por la línea* $Supp_1(X) = \theta^i_{min}$.

4. *Derecho: Definido por la línea* $Supp_1(X) = \theta^i_{min} + \lambda$.

donde $\delta^i_{min}, \theta^i_{min} \in \mathbb{N} - \{0\}$ *y* λ *es un parámetro definido por el usuario. Estos umbrales establecen que todos los EPs tendrán un soporte en el conjunto de datos* $\mathcal{D}_1$ *mayor o igual al umbral izquierdo y menor al umbral derecho. De igual forma, los EPs tendrán un soporte en el conjunto de datos* $\mathcal{D}_2$ *mayor o igual al umbral inferior y menor o igual al umbral superior.* ♦

Propiedad 5.3.1 (Fronteras Izquierda y Superior)
Las fronteras izquierda y superior para cualquier sector $i-1$ *pueden ser derivadas a partir de las fronteras del sector* i *para todo* $i \in [1, k]$. *Sean* δ^i_{min}, $\theta^i_{min} \in \mathbb{N} - \{0\}$ *y* $\lambda > 0$ *entonces,*

$$\delta^{i-1}_{min} = \delta^i_{min} - \lambda$$
$$\theta^{i-1}_{min} = \theta^i_{min} - \lambda$$ □

La demostración de dicha propiedad es trivial si se considera que todos los sectores utilizan el mismo λ determinando el ancho del sector.

En segundo lugar, introduciremos un nuevo operador que tiene por objeto distinguir entre la pertenencia a un conjunto genérico y la pertenencia a un borde[1] y hacer más claras las demostraciones.

Definición 5.3.2 (Pertenencia a un Borde)
Diremos que un itemset Y pertenece *a un borde* $B = \langle \mathcal{L}, \mathcal{R} \rangle$, *notado* $Y \sqsubseteq B$, *si existen itemsets* X *y* Z *tal que* $X \in \mathcal{L}$, $Z \in \mathcal{R}$ *y* $X \subseteq Y \subseteq Z$. ♦

[1]Nótese que los bordes también son conjuntos.

Una propiedad interesante si se exploran los EPs en el plano 2-D desde atrás hacia adelante por el eje $Supp_2(X)$ (es decir, desde $Supp_2(X) = 1$ a $Supp_2(X) = 0$) es que los soportes, con respecto al conjunto $\mathcal{D}_2$, de los EPs que pertenecen al sector $i-1$ son menores a los soportes, con respecto al mismo conjunto $\mathcal{D}_2$, de los EP que pertenecen al sector i. Las siguientes demostraciones formalizan dicha observación.

Lema 5.3.1

Sea B_i un borde que describe los EPs pertenecientes al sector i. Si $X \sqsubseteq B_i$ (X pertenece a B_i) entonces $supp_1(X) \leq \delta^i_{min}$ and $supp_2(X) \geq \theta^i_{min}$. □

Lema 5.3.2

Si $Supp_1(X) \leq \delta^i_{min}$ y $Supp_2(X) \geq \theta^i_{min}$ entonces $X \sqsubseteq B_j$ para algún j tal que $i \leq j \leq k$. □

Teorema 5.3.1 (Monotonicidad de los Bordes)

Si $X \subset Z$, $X \sqsubseteq B_i$ y $Z \sqsubseteq B_j$ implica $i \geq j$. □

Demostración:

Asumamos que $i < j$:

$$supp_2(X) \overset{(Prop.\ 2.5.1)}{\geq} supp_2(Z) \overset{(Teo.\ 5.3.1)}{\geq} \theta^j_{min} \tag{5.1}$$

$$supp_1(X) \overset{(Lem.\ 5.3.1)}{\leq} \delta^i_{min} \overset{(Prop.\ 5.3.1)}{\leq} \delta^j_{min} \quad [\ Hip.\ i < j] \tag{5.2}$$

$$\overset{(Lem.\ 5.3.2)}{\Rightarrow} X \sqsubseteq B_\ell \text{ para algún } i \overset{(Hip.)}{<} j \leq \ell \leq k \quad [\ Por\ (5.1)\ y\ (5.2)\ ]$$

Llegamos a una contradicción ya que $X \sqsubseteq B_i$. Por lo tanto, $i \geq j$. ■

Es importante observar que como consecuencia del teorema previo si la minería se realizar desde el sector k hacia el sector 1 para cualquier itemset presente en el sector i sus subconjuntos propios deben haber sido descubiertos, es decir, dichos subconjuntos pertenecerán a un sector mayor o igual a i o *no serán* patrones emergentes (EPs).[2] Formalmente:

[2]Recordar que los patrones emergentes no respetan la propiedad de anti-monotonía.

Corolario 5.3.1

Sean $B_i = \langle \mathcal{L}_i, \mathcal{R}_i \rangle$ y $B_j = \langle \mathcal{L}_j, \mathcal{R}_j \rangle$ dos bordes tal que $i \leq j$. Si $Y \sqsubseteq B_i$ y $X \in \mathcal{L}_j$ entonces $Y \not\subset X$. □

Demostración:

Supongamos que $Y \subset X$. Ya que $i \leq j$ entonces:

1. $i = j$,

$$Y \sqsubseteq B_i \wedge X \in \mathcal{L}_j \ \wedge \ Y \subset X \qquad [\ \textit{Hipótesis}\ ]$$

$$Y \sqsubseteq B_j \qquad [\ i = j\ ] \qquad (5.3)$$

$$(\exists X_j \in \mathcal{L}_j)\ (\exists Z_j \in \mathcal{R}_j)\ X_j \subseteq Y \subseteq Z_j \qquad [\ \textit{Def. 5.3.2 y (5.3)}\ ] \qquad (5.4)$$

$$X_j \subseteq Y \subset X \qquad [\ \textit{Por (5.4)}\ ] \qquad (5.5)$$

$$X \notin \mathcal{L}_j \qquad [\ \textit{Por (5.5)}\ X_j \subset X\ ]$$

Contradicción.

2. $i < j$,

$$Y \sqsubseteq B_i \wedge X \in \mathcal{L}_j \ \wedge \ Y \subset X \qquad [\ \textit{Hipótesis}\ ] \qquad (5.6)$$

$$X \sqsubseteq B_k, j \leq k \qquad [\ \textit{Por (5.6) y Def. 5.3.2}\ ] \qquad (5.7)$$

$$i \geq k \geq j \qquad [\ \textit{Por (5.7) y Teo. 5.3.1}\ ]$$

Contradicción. ■

Por último, el siguiente teorema tiene una implicación importante en nuestro acercamiento permitiéndonos podar los bordes antes de realizar la operación de diferencia[3]. Este teorema se basa en determinar la región del plano de soportes 2-D donde se ubicarán el conjunto de itemsets que son subconjuntos propios de un itemset dado.

Teorema 5.3.2

Sea el par (θ, δ) las coordenadas en el plano de soportes para un itemset I dado, tal que $supp_1(I) = \theta$ y $supp_2(I) = \delta$. Cualquier itemset subconjunto de I debe ubicarse en la región

[3]Esta poda puede observarse en la sección 5.5.3.

delimitada por las siguientes líneas: la inferior $supp_2(X) = \delta$*, la izquierda* $supp_1(X) = \theta$*, la derecha* $supp_1(X) = 1$ *y la superior* $supp_2(X) = 1$*.* □

Demostración:

Sean X *e* Y *un par de itemsets tal que* $X \subseteq Y$*:*

$$supp_1(Y) = \theta \quad \wedge \quad supp_2(Y) = \delta \qquad [\ \textit{Hipótesis}\ ] \qquad (5.8)$$

$$supp_1(X) \geq supp_1(Y) \quad \wedge \quad supp_2(X) \geq supp_2(Y) \qquad [\ \textit{Prop. 2.5.1}\ ] \qquad (5.9)$$

$$supp_1(X) \geq \theta \quad \wedge \quad supp_2(X) \geq \delta \qquad [\ \textit{Por (5.8) y (5.9)}\ ] \qquad \blacksquare$$

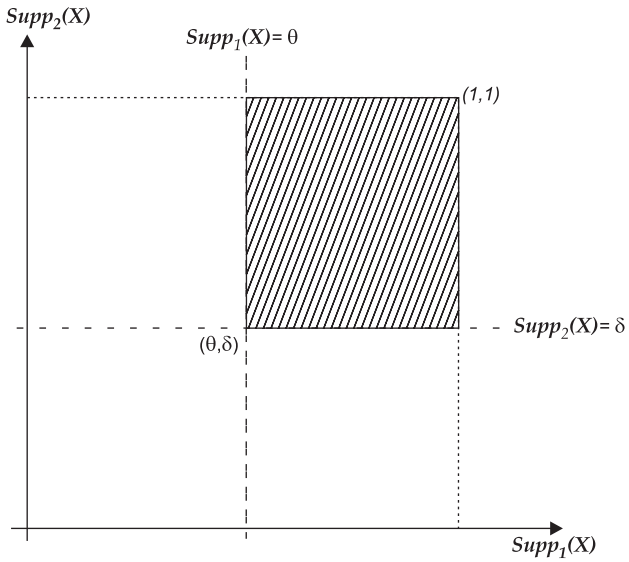


Figura 5.5: Propiedad de antimonotonía en el plano de soportes

En la figura 5.5 puede observase gráficamente la región donde los itemsets subconjuntos a un itemset dado deben encontrarse.

5.4. Minería Incremental de Patrones Maximales

El problema de minería de patrones maximales ha sido explorado en profundidad en la sección 3.2. Como hemos mencionado, los distintos métodos varían en performance según las características del conjunto de datos utilizado. Debido a esto, como los distintos

métodos poseen una performance comparable, utilizaremos `GenMax` por su simplicidad y su adaptabilidad a nuestro marco de trabajo.

Los patrones maximales (MFIs) de cada sector se obtendrán realizando una expansión de los MFIs del sector previo. Obsérvese que el umbral de mínimo soporte para cada conjunto de datos se disminuye[4] en forma progresiva a medida que se avanza hacia el origen. Por lo tanto, los patrones maximales obtenidos para sectores previos seguirán siendo útiles aunque algunos de ellos perderán su propiedad de maximal por ser subconjunto de algún nuevo patrón maximal. Durante la búsqueda de nuevos patrones maximales realizaremos, en general, dos clases de podas:

1. Podas de itemsets infrecuentes.

2. Podas de itemsets subsumidos.

Obsérvese que, como el soporte es reducido progresivamente, con el transcurso del tiempo los itemsets infrecuentes serán frecuentes. En contraste, los itemsets, frecuentes o no, subsumidos por otro itemset no serán de importancia ya que son subconjunto de algún MFI y por lo tanto cualquier subconjunto generado por éstos también podrá ser generado por el MFI. La idea básica de nuestro acercamiento para la búsqueda incremental de MFIs consiste en mantener una lista de aquellos itemsets que fueron podados por ser infrecuentes, los cuales corresponden al borde negativo como fuera mencionado en la sección 3.4. Mantendremos esta lista ordenada en forma decreciente por soporte. De esta forma, para realizar la expansión de los MFIs, recorreremos dicha lista hasta encontrar un itemset infrecuente utilizando el nuevo soporte, entonces todos los itemsets hasta el itemset infrecuente serán utilizados en conjunción con los anteriores MFIs para generar el nuevo conjunto de MFIs.

Este conjunto de MFIs con itemsets del borde negativo y el nuevo soporte conformarán la entrada para el algoritmo `GenMax`, el cuál continuará su ejecución desde el punto en que había terminado anteriormente. Esto permite evitar tener que comenzar de cero la generación de patrones maximales bajo un nuevo soporte.

[4]En inglés se utiliza el término *drill-down*.

5.5. Operación de Diferencia Extendida

En la sección 5.2.2 se observó que se realizará una operación de diferencia para cada sector i. En primer lugar, se obtienen los itemsets del conjunto de datos $\mathcal{D}_1$ cuyo soportes se encuentran en la región $[\theta^{i-1}_{min}, \theta^{i}_{min}]$ y los itemsets del conjunto de datos $\mathcal{D}_2$ cuyo soporte supere δ^{i}_{min}. Luego, se aplicará la operación de diferencia entre ambos conjuntos, restando el conjunto de itemsets de $\mathcal{D}_2$ al conjunto de itemsets de $\mathcal{D}_1$. Esta operación dará como resultado el borde ilustrado en la figura 5.4. Sin embargo, es importante notar que la operación de diferencia[5] está solamente definida para bordes cerrados por debajo[6]. Por tal motivo, necesitamos una operación de diferencia que permita la utilización de al menos un borde sin restricciones, el cual describa los itemsets del conjunto de datos $\mathcal{D}_1$. Para lograr dicho objetivo, realizaremos una extensión de la operación de diferencia, definida en la sección 4.5.3, aprovechando su eficiencia. Probaremos entonces la viabilidad de dicha extensión a través de las siguientes proposiciones, las cuales extienden las proposiciones 4.5.2 y 4.5.3.

Proposición 5.5.1 (Diferencia de bordes extendida bien definida (1))
La diferencia de bordes $\langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ *está siempre* bien definida*, resultando en el borde* $\langle \{\emptyset\}, \{\emptyset\}\rangle$ *cuando algún itemset en* $\mathcal{R}_2$ *contiene al itemset* U *o en un borde cuya frontera derecha es* $\{U\}$. ♦

Proposición 5.5.2 (Diferencia de bordes extendida bien definida (2))
La diferencia de bordes $\langle \mathcal{L}_1, \mathcal{R}_1\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ *está siempre* bien definida. ♦

Para la demostración de ambas proposiciones se pueden utilizar las mismas demostraciones provistas para las proposiciones 4.5.2 y 4.5.3. Nuestra operación de diferencia se basa en la suposición que cualquier operación de diferencia de un borde general con respecto a un borde cerrado por debajo puede ser transformada en una o más operaciones de diferencia donde ambos bordes sean cerrados por debajo. Para demostrar formalmente

[5] Ver definición 4.5.8 página 148.
[6] Ver definición 4.5.1 página 140.

dicha afirmación realizaremos una descomposición del problema en proposiciones, las cuales se encuentran resumidas en los siguientes pasos:

1. Transformar el borde general B_1 en un conjunto de bordes con único conjunto tanto para la frontera izquierda como para la frontera derecha.

2. Podar del primer borde, aquellos itemsets que no pertenecen al segundo borde, es decir, aquellos itemsets para los cuales no existe un itemset en el segundo borde que sea superconjunto. Estos itemsets podados formarán parte del borde resultante sin necesidad de participar en la diferencia.

3. Realizar dos tipos podas sobre la frontera derecha del segundo borde B_2.

   a) Filtrar aquellos itemsets que no podan ningún itemset en B_1, es decir, si existe algún itemset en B_2 que no es superconjunto de ningún itemset de B_1 entonces dicho itemset no es necesario que se considere en la diferencia.

   b) Filtrar los itemset que no sean útiles para la diferencia utilizando el teorema 5.3.2 que hacer referencia a la propiedad de antimonotonía en el contexto del plano de soportes.

4. Transformar los bordes resultantes de dichos filtrados en bordes cerrados por debajo.

5. Realizar la diferencia de bordes.

6. Realizar una transformación inversa para obtener el resultado.

El primer paso permite descomponer el problema en varios subproblemas. El segundo y tercer paso nos permite reducir el espacio de trabajo necesario para la operación de diferencia. El cuarto paso, consiste en realizar una transformación de los bordes a un dominio donde se pueda aplicar la operación de diferencia. En el quinto paso, se realiza la diferencia en este dominio. Por último, se transforma el borde resultante al dominio original. Analizaremos cada uno de estos pasos en las siguientes secciones.

5.5.1. Descomposición del primer borde

El objetivo de este primer paso es transformar el borde general (el primer borde) en un borde lo más cercano posible a ser cerrado por debajo, es decir, la frontera izquierda consistirá únicamente del conjunto vacío. Utilizaremos el primer borde para generar un conjunto de bordes tales que la unión de los itemsets generados por estos bordes sea igual al borde original. Esta sintetización de bordes se logra generando un nuevo borde por cada itemset que aparezca en la frontera izquierda del borde original. Asumiremos, como en [20] , que la frontera derecha del borde general (B_1) consta simplemente de un itemset que llamaremos U. Esta suposición es garantizada por la proposición 4.5.3 [7]. La descomposición se realizará de la siguiente forma:

Definición 5.5.1 (Descomposición de B_1)
Sea $B_1 = \langle\{x_1, x_2, \ldots, x_n\}, \{U\}\rangle$. *El borde* B_1 *se* descompone *en el siguiente conjunto de bordes:*

$$\{\langle\{x_i\}, \{U\}\rangle \mid x_i \in \mathcal{L}_1\}$$ ♦

Mostraremos como se puede aplicar esta descomposición para simplificar la operación de diferencia utilizando el siguiente ejemplo:

Ejemplo 5.5.1
Sea $B_1 = \langle\{\{1\}, \{2,3\}, \{3,4\}\}, \{\{1,2,3,4\}\}\rangle$ *y* $B_2 = \langle\{\emptyset\}, \{\{2,3\}, \{4\}\}\rangle$. *Al realizar la diferencia* $B_1 - B_2$ *la misma resulta en un espacio convexo, denotado por el borde* $\langle\{\{1\}, \{3,4\}\}, \{\{1,2,3,4\}\}\rangle$. *Esta operación de diferencia se ilustra en la figura 5.6.*

Luego de aplicar la descomposición de la frontera izquierda al borde B_1, *se puede realizar la operación de diferencia utilizando el mismo borde* B_2 *sobre los bordes resultantes de la descomposición. Dicha situación se ilustra en la figura 5.7.*

Al realizar la unión de los bordes resultantes de la operación de diferencia, se observa en la figura 5.8 que el conjunto de itemsets final es igual que el que hubiera resultado si no se hubiese realizado la descomposición. Es importante aclarar que la unión de estos

[7] Ver página 151.

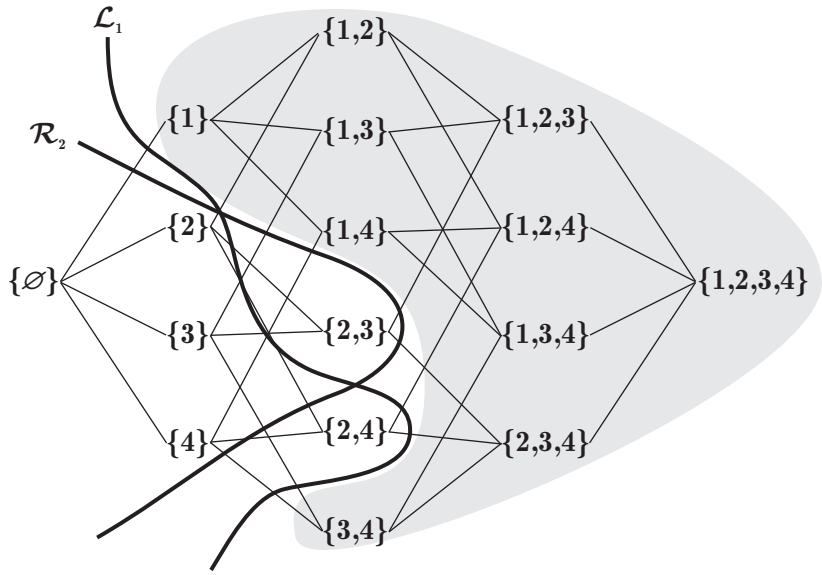


Figura 5.6: Operación de diferencia entre B_1 y B_2.

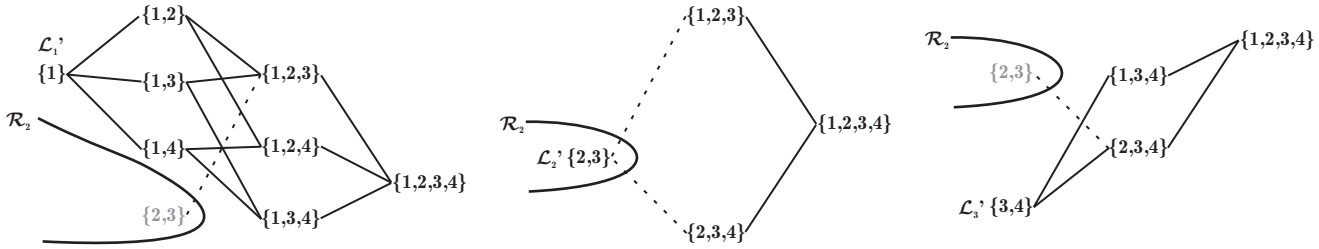


Figura 5.7: Operación de diferencia entre B_1 y B_2 luego de la descomposición.

resultados parciales consiste simplemente en unir las fronteras izquierdas ya que la frontera derecha para todos los bordes es la misma ($\{U\}$). *Sin embargo, debe observase que las fronteras izquierdas pueden contener itemsets no minimales como sucede con los conjuntos* $\{1,2,3\}$ *y* $\{1,3,4\}$. *Por tal motivo, antes de realizar la unión de las fronteras izquierdas es necesario eliminar los itemsets no minimales.* ◊

La siguiente proposición formaliza el hecho que la diferencia entre cada uno de los bordes pertenecientes a la descomposición del borde B_1 con respecto a B_2 produce el mismo resultado que la diferencia de borde general B_1 con respecto al borde cerrado por

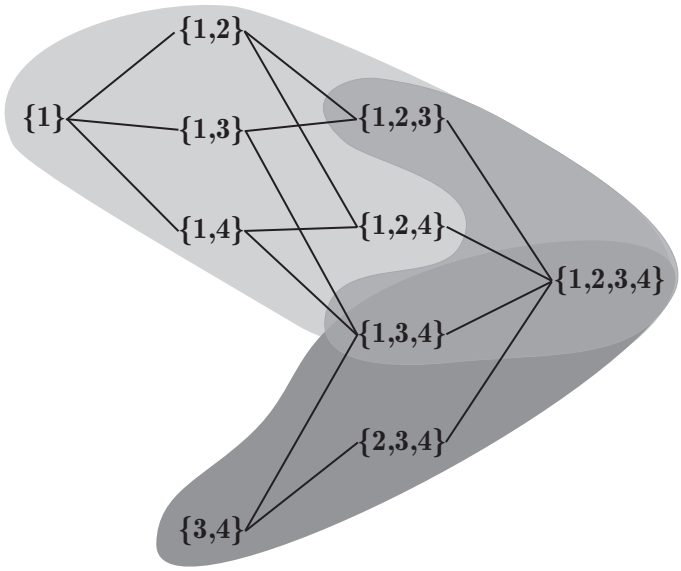


Figura 5.8: Resultado de la unión de diferencias parciales.

debajo B_2.

Proposición 5.5.3

Sea $\mathcal{L}_1 = \{x_1, x_2, \ldots, x_n\}$ *y* $\mathcal{R}_2 = \{z_1, z_2, \ldots, z_m\}$. *Si* $\langle \mathcal{L}'_i, \{U\}\rangle = \langle \{x_i\}, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ *para todo* $x_i \in \mathcal{L}_1$ *con* $1 \leq i \leq n$ *entonces:*

$$\langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle = \langle \cup_{i=1}^{n} \mathcal{L}'_i, \{U\}\rangle \qquad \blacklozenge$$

Demostración:

Probaremos dicha igualdad por doble inclusión:

($\sqsubseteq$) *Si* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ *entonces* $Y \sqsubseteq \langle \cup_{i=1}^{n} \mathcal{L}'_i, \{U\}\rangle$

$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ [*Hipótesis*]

$(\exists X_i \in \mathcal{L}_1)\ X_i \subseteq Y$ [$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle$] (5.10)

$(\forall Z \in \mathcal{R}_2)\ Y \not\subseteq \mathcal{R}_2$ [$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2\rangle$] (5.11)

$Y \sqsubseteq \langle \{X_i\}, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle = \langle \mathcal{L}'_i, \{U\}\rangle$ [*Por* (5.10) *and* (5.11)]

$Y \sqsubseteq \langle \cup_{i=1}^{n} \mathcal{L}'_i, \{U\}\rangle$

($\sqsupseteq$) *Si* $Y \sqsubseteq \langle \cup_{i=1}^{n} \mathcal{L}'_i, \{U\} \rangle$ *entonces* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$

$$(\exists X_i \in \cup_{i=1}^{n} \mathcal{L}'_i)\ X_i \subseteq Y \subseteq U \tag{5.12}$$

$$(\exists i)\ X_i \in \mathcal{L}'_i \qquad [\ X_i \in \cup_{i=1}^{n} \mathcal{L}'_i\ ]$$

$$X_i \sqsubseteq \langle \mathcal{L}'_i, \{U\} \rangle \qquad [\ \textit{Def. de borde}\ ]$$

$$X_i \sqsubseteq \langle \{X_i\}, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle \tag{5.13}$$

$$X_i \in \mathcal{L}_1 \qquad [\ \textit{Def.}\ \mathcal{L}'\ ] \tag{5.14}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle \qquad [\ \textit{Por}\ (5.12)\ y\ (5.14)\ ] \tag{5.15}$$

$$X_i \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle \qquad [\ \textit{Por}\ (5.13)\ ] \tag{5.16}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle \qquad [\ \textit{Por}\ (5.16)\ y\ X_i \subseteq Y\ ] \tag{5.17}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle \qquad [\ \textit{Por}\ (5.15)\ y\ (5.17)\ ] \quad \blacksquare$$

5.5.2. Poda del primer borde

Como segundo paso, procederemos a simplificar la operación de diferencia utilizando una versión reducida del primer borde. El objetivo será detectar en la frontera izquierda del primer borde aquellos itemsets que sean subconjunto de algún itemset que pertenezca a la frontera derecha del segundo borde, es decir, detectaremos de la frontera izquierda del primer borde aquellos itemsets que no formarán parte del borde resultante de la diferencia ya que son podados por el segundo borde. Generaremos un nuevo primer borde solamente con dichos itemsets como frontera izquierda, dejando los otros itemsets para incluirlos luego de la diferencia. Estos últimos itemsets serán incluidos luego ya que no son podados por ningún itemset del segundo borde. Veamos con un ejemplo la aplicación de esta poda.

Ejemplo 5.5.2

Sea $B_1 = \langle \{\{1\}, \{2,3\}, \{3,4\}\}, \{\{1,2,3,4\}\} \rangle$ *y* $B_2 = \langle \{\emptyset\}, \{\{2,3\}, \{4\}\} \rangle$. *La diferencia* $B_1 - B_2$ *ha sido ilustrada en la figura 5.6. Obsérvese que para* $\{1\}$ *y* $\{3,4\}$ *no existe ningún superconjunto en la frontera derecha de* B_2 *entonces generamos un nuevo borde* $B'_1 = \langle \{\{2,3\}\}, \{\{1,2,3,4\}\} \rangle$. *La diferencia* $B'_1 - B_2$, *ilustrada en la figura 5.7, producirá el borde* $B'_1 - B_2 = \langle \{\{1,2,3\}, \{2,3,4\}\}, \{\{1,2,3,4\}\} \rangle$. *Luego de re-*

alizar dicha diferencia se procederá a incluir los itemsets dejados afuera, generando una nueva frontera izquierda para el borde. Como resultado, se obtendrá el siguiente borde $\langle\{\{1\},\{3,4\}\},\{\{1,2,3,4\}\}\rangle$. ◊

Es interesante mencionar que esta poda produce mejores resultados cuando es aplicada antes o durante la descomposición del primer borde. Mostremos formalmente que si efectuamos la diferencia entre los bordes $B_1 = \langle\mathcal{L}_1, \mathcal{R}_1\rangle$ y $B_2 = \langle\mathcal{L}_2, \mathcal{R}_2\rangle$, quitando de $\mathcal{L}_1$ los itemsets que no poseen ningún superconjunto en $\mathcal{R}_2$, el resultado es igual a realizar la diferencia $B_1 - B_2$.

Proposición 5.5.4

Sea $\mathcal{L}_1 = \{x_1, x_2, \ldots, x_n\}$ *y* $\mathcal{R}_2 = \{z_1, z_2, \ldots, z_m\}$. *Sea* $\mathcal{L}'_1 = \{x_i \in \mathcal{L}_1 \mid (\exists z_j \in \mathcal{R}_2)\ x_i \subseteq z_j\}$. *El conjunto de los elementos dejados fuera de la diferencia (remaining set) será* $\mathcal{L}''_1 = \mathcal{L}_1 - \mathcal{L}'_1$. *Si* $\langle\mathcal{L},\mathcal{R}\rangle = \langle\mathcal{L}'_1,\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$ *entonces:*

$$\langle\mathcal{L}_1,\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle = \langle\mathcal{L}''_1 \cup \mathcal{L},\{U\}\rangle \qquad \blacklozenge$$

Demostración:

Probaremos dicha igualdad por doble inclusión:

($\sqsubseteq$) *Si* $Y \sqsubseteq \langle\mathcal{L}_1,\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$ *entonces* $Y \sqsubseteq \langle\mathcal{L}''_1 \cup \mathcal{L},\{U\}\rangle$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle \tag{5.18}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle \tag{5.19}$$

$$(\exists X \in \mathcal{L}_1)\ X \subseteq Y \subseteq U \qquad [\ Por\ (5.18)\ ]$$

$$i)\ X \in \mathcal{L}_1'' \qquad [\ \mathcal{L}_1 = \mathcal{L}_1' \cup \mathcal{L}_1''\ ]$$

$$Y \sqsubseteq \langle \mathcal{L}_1'', \{U\} \rangle \tag{5.20}$$

$$Y \sqsubseteq \langle \mathcal{L}_1'' \cup \mathcal{L}, \{U\} \rangle \qquad [\ Por\ (5.19)\ y\ (5.20)\ ]$$

$$ii)\ X \in \mathcal{L}_1' \qquad [\ \mathcal{L}_1 = \mathcal{L}_1' \cup \mathcal{L}_1''\ ]$$

$$Y \sqsubseteq \langle \mathcal{L}_1', \{U\} \rangle \tag{5.21}$$

$$Y \sqsubseteq \langle \mathcal{L}, \{U\} \rangle \qquad [\ Por\ (5.19)\ y\ (5.21)\ ] \tag{5.22}$$

$$Y \sqsubseteq \langle \mathcal{L}_1'' \cup \mathcal{L}, \{U\} \rangle \qquad [\ Por\ (5.22)\ ]$$

($\sqsupseteq$) *Si* $Y \sqsubseteq \langle \mathcal{L}_1'' \cup \mathcal{L}, \{U\} \rangle$ *entonces* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$.

1. $(\exists X \in \mathcal{L}_1'')\ X \subseteq Y \subseteq U$

$$\mathcal{L}_1'' \subseteq \mathcal{L}_1 \qquad [\ \mathcal{L}_1'' = \mathcal{L}_1 - \mathcal{L}_1'\ ] \tag{5.23}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle \qquad [\ X \in \mathcal{L}_1\ ]$$

$$X \notin \mathcal{L}_1' \qquad [\ Por\ (5.23)\ y\ X \in \mathcal{L}_1''\ ]$$

$$X \not\subseteq Z \quad \forall Z \in \mathcal{R}_2 \qquad [\ Def.\ \mathcal{L}_1'\ ]$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle$$

2. $(\exists X \in \mathcal{L})\ X \subseteq Y \subseteq U$

$$X \sqsubseteq \langle \mathcal{L}'_1, \{U\} \rangle$$

$$X \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle \tag{5.24}$$

$$Y \sqsubseteq \langle \mathcal{L}'_1, \{U\} \rangle$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle \qquad [\ \mathcal{L}'_1 \subseteq \mathcal{L}_1\ ] \tag{5.25}$$

$$(\forall Z \in \mathcal{R}_2)\ X \not\subseteq Z \qquad [\ Por\ (5.24)\ ] \tag{5.26}$$

$$(\forall Z \in \mathcal{R}_2\ \ \exists X_i \in X)\ X_i \notin Z \qquad [\ Por\ (5.26)\ ]$$

$$(\forall X_i \in X)\ X_i \in Y \qquad [\ X \subseteq Y\ ] \tag{5.27}$$

$$(\forall Z \in \mathcal{R}_2)\ Y \not\subseteq Z \qquad [\ Por\ (5.27)\ ] \tag{5.28}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2 \rangle \qquad [\ Por\ (5.28)\ ] \tag{5.29}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\} \rangle - \langle \{\emptyset\}, \mathcal{R}_2 \rangle \qquad [\ Por\ (5.25)\ y\ (5.29)\ ] \quad \blacksquare$$

5.5.3. Poda del segundo borde

Como último paso para simplificar la diferencia, realizaremos una poda sobre el conjunto de itemsets que constituyen la frontera derecha del segundo borde. De esta frontera serán podados todos aquellos itemsets que no sean esenciales para la diferencia. Para realizar esta detección proponemos dos podas o filtros que analizaremos a continuación.

Filtrado por superconjunto

La primer poda es análoga a la poda realizada sobre el primer borde. La misma consiste en quitar de la frontera derecha de B_2 todos aquellos itemsets que no sean superconjuntos de algún itemset de B_1, es decir, aquellos itemsets que no podan ningún itemset de B_1 cuando es realizada la diferencia. En particular, solamente necesitamos explorar la frontera izquierda de B_1.

Ejemplo 5.5.3

Se puede observar en la figura 5.6 (página 205) que el itemset $\{4\}$ *perteneciente a la frontera derecha de* B_2 *no aporta nada para la diferencia y por tal motivo puede ser*

filtrado. De igual forma, en la figura 5.7 (página 205) el itemset $\{2,3\}$ *no es esencial en la diferencia con el borde* $\langle\{\{1\},\{1,2,3,4\}\rangle$ *como tampoco con el borde* $\langle\{\{3,4\},\{1,2,3,4\}\rangle$.$\Diamond$

Demostraremos que si se eliminan de la frontera derecha de B_2 los itemsets que no son superconjuntos de ningún itemset de B_1, el resultado de realizar la diferencia $B_1 - B_2'$, donde B_2' consiste de B_2 sin estos itemsets es igual que la diferencia original $B_1 - B_2$.

Proposición 5.5.5

Sean las fronteras $\mathcal{L}_1 = \{x_1, x_2, \ldots, x_n\}$ *y* $\mathcal{R}_2 = \{z_1, z_2, \ldots, z_m\}$. *Sea* $\mathcal{R}_2' = \{z_i \in \mathcal{R}_2 \mid (\exists x_j \in \mathcal{L}_1)\ x_j \subseteq z_i, 1 \leq i \leq m, 1 \leq j \leq n\}$ *entonces:*

$$\langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle = \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2'\rangle \qquad \blacklozenge$$

Demostración:

Probaremos dicha igualdad por doble inclusión:

($\sqsubseteq$) $Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$ *entonces* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2'\rangle$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle \tag{5.30}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2\rangle \tag{5.31}$$

$$\mathcal{R}_2' \subseteq \mathcal{R}_2 \qquad [\ \textit{Def. } \mathcal{R}_2'\ ] \tag{5.32}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2'\rangle \qquad [\ \textit{Por (5.31) y (5.32)}\ ] \tag{5.33}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2'\rangle \qquad [\ \textit{Por (5.30) y (5.33)}\ ]$$

($\sqsupseteq$) *Si* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}'_2\rangle$ *entonces* $Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle \tag{5.34}$$

$$(\exists X \in \mathcal{L}_1)\ X \subseteq Y \subseteq U \qquad [\ Por\ (5.34)\ ] \tag{5.35}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}'_2\rangle \tag{5.36}$$

$$(\forall Z' \in \mathcal{R}'_2)\ Y \not\subseteq Z' \qquad [\ Por\ (5.36)\ y\ \emptyset \subseteq Y\ ] \tag{5.37}$$

$$(\forall Z \in \mathcal{R}_2)(\nexists X' \in \mathcal{L}_1)\ X' \subseteq Y \subseteq Z \qquad [\ Por\ (5.37)\ y\ Def.\ \mathcal{R}'_2\ ] \tag{5.38}$$

$$(\forall Z \in \mathcal{R}_2)\ X \not\subseteq Z \qquad [\ Por\ (5.35)\ y\ (5.38)\ ] \tag{5.39}$$

$$(\forall Z \in \mathcal{R}_2)\ Y \not\subseteq Z \qquad [\ Por\ (5.35)\ y\ (5.39)\ ] \tag{5.40}$$

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R}_2\rangle \qquad [\ Por\ (5.40)\ ] \tag{5.41}$$

$$Y \sqsubseteq \langle \mathcal{L}_1, \{U\}\rangle - \langle \{\emptyset\}, \mathcal{R}_2\rangle \qquad [\ Por\ (5.34)\ y\ (5.41)\ ] \quad \blacksquare$$

Filtrado por propiedad de anti-monotonía

La segunda poda que se podría realizar sobre la frontera derecha del segundo borde consiste eliminar de dicho borde aquellos itemsets que no participan de la operación de diferencia. Por tal motivo, si se observa nuevamente la figura 5.4 se podría establecer que los itemsets importantes del borde B_2 son aquellos que se intersectan con los itemsets de B_1. Sin embargo, obtener esta intersección es tan costosa como realizar la diferencia original. Sin embargo, se puede utilizar el teorema 5.3.2 para eliminar *la mayoría* de los itemsets que se encuentran a la derecha en el plano de soportes del sector que actualmente está siendo explorado, tal como se muestra en la figura 5.9.

Para eliminar la mayoría de los itemsets de dicha región se procede a recorrer la frontera derecha del segundo borde eliminando aquellos itemsets que se encuentren a la derecha de la región de intersección. Por el teorema 5.3.2 sabemos que todos los subconjuntos de dichos itemsets caerán en la región derecha a la intersección y por lo tanto no son útiles para la diferencia. El problema consiste en detectar los itemsets de la frontera derecha de B_2 que se encuentran a la derecha de la intersección. Proponemos utilizar alguna de las siguientes alternativas *a*) calcular para cada itemset de la frontera derecha de B_2 ($\mathcal{R}_2$) su

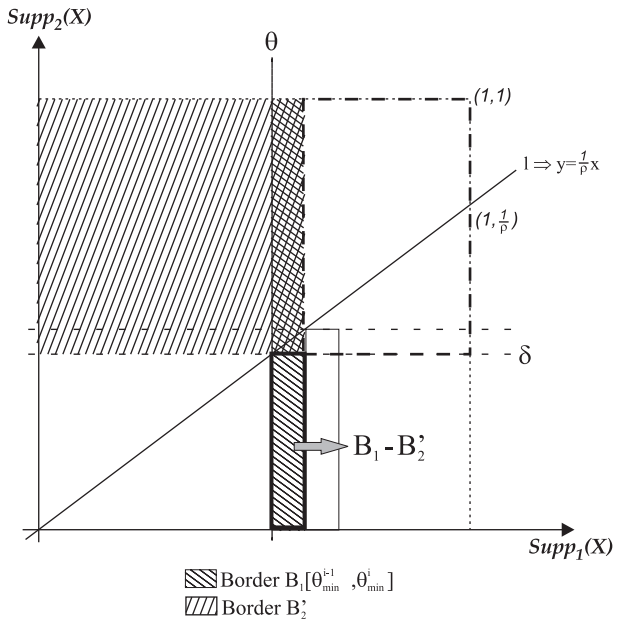


Figura 5.9: Diferencia de bordes utilizando poda por monotonía.

soporte en el conjunto de datos $\mathcal{D}_1$ y de esa forma eliminar aquellos que tengan un soporte mayor a θ^i_{min}, o *b*) utilizar el borde derecho de B_1 ($\mathcal{R}_1$) para detectar los itemsets de $\mathcal{R}_2$ que sean superconjuntos de algún itemset de $\mathcal{R}_1$, esta forma no es tan efectiva como la anterior pero requiere un tiempo de procesamiento mucho menor.

Debe notarse que no todos los itemsets de dicha región son podados, ya que se realiza la poda utilizando los itemsets la frontera derecha de B_2 que se encuentran en dicha región. De esta forma se asegura que los itemsets que son eliminados pertenecen a la región en cuestión. Sin embargo, existen itemsets pertenecientes a esta región que son generados por otros itemsets de $\mathcal{R}_2$ los cuales no se encuentran a la derecha de la intersección. Supongamos tener un itemset de $\mathcal{R}_2$ cuyo soporte en $\mathcal{D}_1$ es menor a δ^i_{min}, entonces por el teorema 5.3.2 sus itemsets subconjuntos pueden ubicarse en el plano de soportes a la derecha y por arriba de la ubicación de dicho itemset en el plano. En particular, podrían ubicarse en la región que hemos intentado podar. Por lo tanto, no todos los itemsets de esta región son podados por este mecanismo.

Aún cuando este filtrado es interesante existen conflictos con la *poda por superconjun-*

tos ya que dicha poda subsume a la poda por anti-monotonía, es decir, si algún itemset de $\mathcal{R}_2$ se encontraba a la derecha de la región intersección entonces dicho itemset no podaba ningún itemset de B_1 y por lo tanto éste era quitado en la poda por superconjuntos. Sin embargo, hemos incluido este resultado porque el testeo por superconjuntos es una operación relativamente costosa. En dicho caso, de no utilizarse la poda por superconjuntos, se puede mejorar la operación de diferencia utilizando la poda por anti-monotonía.

5.5.4. Transformación de bordes

Por último procederemos a realizar la transformación de los bordes con motivo de poder aplicar la operación de diferencia sobre los mismos. Para realizar dicha transformación realizaremos los siguientes supuestos acerca de los bordes:

1. En el borde B_1 las fronteras izquierda y derecha constan de conjuntos unitarios, es decir, que dichas fronteras constan de un solo itemset.

2. El borde B_2 es cerrado por debajo, es decir, su frontera izquierda consiste del conjunto vacío. Cada itemset de la frontera derecha de B_2 es esencial y por lo tanto para cada itemset en dicha frontera existe al menos un itemset en la frontera izquierda de B_1 que es subconjunto de éste.

Estos requisitos permiten realizar una transformación más sencilla de los bordes y realizar demostraciones más simples. Por otro lado, estos supuestos no constituyen restricciones para la aplicación de la operación de diferencia, ya que tomando cualquier borde general y aplicando la descomposición de bordes[8] se puede obtener una colección de bordes con las características mencionadas en la primer suposición. Para la segunda suposición, recordemos que al igual que la operación de diferencia tradicional, nuestra operación de diferencia también utiliza un borde cerrado por debajo, sin embargo utilizaremos además la poda por superconjuntos para que el segundo supuesto también sea válido.

[8] Ver definición 5.5.1 página 204.

El objetivo de este paso es realizar una operación de diferencia sobre bordes más generales utilizando como base la operación de diferencia sobre bordes cerrados por debajo. Para ello, transformaremos los bordes a un dominio equivalente donde pueda ser realizada la operación de diferencia y luego utilizando una operación de transformación inversa obtendremos el resultado de la diferencia como si se hubiera realizado una diferencia sin restricciones. Intuitivamente, la transformación consistirá de utilizar el itemset de la frontera izquierda del primer borde para transformar ambos bordes. El procedimiento consiste en restar dicho itemset a cada itemset de las fronteras de ambos bordes. En particular, se realizará dicha diferencia solamente a las fronteras derechas de ambos bordes, ya que la frontera izquierda del primer borde es el mismo itemset y por lo tanto el resultado es una frontera con el conjunto vacío (cerrada por debajo) y la frontera izquierda del segundo borde como mencionáramos era el conjunto vacío.

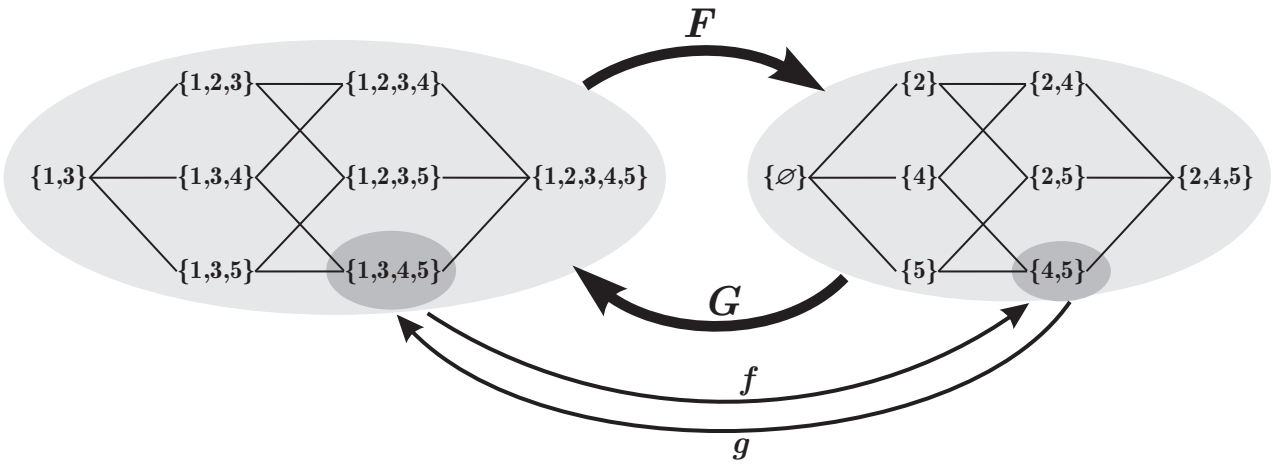


Figura 5.10: Funciones de transformación de itemsets y bordes.

En la figura 5.10 se puede apreciar como se realiza la transformación de un borde genérico a un borde cerrado por debajo y viceversa. Obsérvese que en la figura se utilizan dos clases de funciones, las funciones con letra mayúscula son aplicadas sobre bordes y las funciones con letra minúscula son aplicadas sobre itemsets. Luego de realizar dicha transformación, se aplica la diferencia de bordes tal como se propone en el algoritmo 6 y

al borde resultante de dicha operación se le aplica la función inversa a la transformación anteriormente efectuada, es decir, los itemsets de ambas fronteras serán unidos con el itemset restado anteriormente.

Daremos a continuación una serie de demostraciones que formalizan y muestran la correctitud de estas nociones intuitivas. En primer lugar, definiremos formalmente las funciones de transformación demostrando que cada una de estas funciones tiene una función inversa. En general, utilizaremos el conjunto U para representar el conjunto de todos los items posibles.

Definición 5.5.2 (Funciones f y g)
Sean f y g funciones con dominio en $\mathcal{P}(U) \times \mathcal{P}(U)$ y codominio en $\mathcal{P}(U)$ entonces dichas funciones se definen como $f(X, S) = X - S$ y $g(X, S) = X \cup S$. ♦

Utilizaremos estas funciones f y g sobre itemsets para definir un par de funciones F y G para la transformación entre reticulados de itemsets. La función F aplicará la función f a cada itemset del reticulado y en forma análoga la función G aplicará la función g. Definiremos estas funciones F y G en el contexto de una representación de reticulados utilizando bordes y por tal motivo la aplicación de f y g solo se realizará sobre las fronteras[9].

Definición 5.5.3 (Funciones F y G)
Sea $\mathcal{K}$ la colección de todos los posibles reticulados convexos. Las funciones F y G, con dominio $\mathcal{K} \times \mathcal{P}(U)$ y codominio $\mathcal{K}$, se definen de la siguiente forma:

$$F(\langle \mathcal{L}, \mathcal{R} \rangle, S) = \langle \{f(x_i, S) \mid x_i \in \mathcal{L}, 1 \leq i \leq n\}, \{f(z_j, S) \mid z_j \in \mathcal{R}, 1 \leq j \leq m\} \rangle$$
$$G(\langle \mathcal{L}, \mathcal{R} \rangle, S) = \langle \{g(x_i, S) \mid x_i \in \mathcal{L}, 1 \leq i \leq n\}, \{g(z_j, S) \mid z_j \in \mathcal{R}, 1 \leq j \leq m\} \rangle$$ ♦

Obsérvese que la función g es la función inversa de f y que la función G es la función inversa a F. Utilizaremos para dicha demostración muchas de las leyes provistas para la teoría de conjuntos[10], las cuales se encuentran resumidas en el cuadro **??**. En el caso

[9] Es importante observar que como se mencionara anteriormente en la práctica solo es necesario operar sobre la frontera derecha.

[10] Recordar que los itemsets con conjuntos de items.

de obtener el conjunto complemento sobre algún itemset se asumirá implícitamente que dicha complementación se realiza con respecto a un conjunto U de items.

Propiedad 5.5.1

Sean X y S itemsets tal que $S \subseteq X \subseteq U$ entonces $X = g(f(X,S),S)$. □

Demostración:

$$\begin{aligned}
g(f(X,S),S) &= g(X-S,S) = (X-S)\cup Y && [\ \textit{Def. } f \textit{ y } g\ ]\\
&= (X\cap S^c)\cup S && [\ \textit{Prop. } A-B = A\cap B^c\ ]\\
&= (X\cup S)\cap (S^c\cup S) && [\ \textit{(3a) Ley Distributiva}\ ]\\
&= (X\cup S)\cap U && [\ \textit{(7a)}\ A\cup A^c = U\ ]\\
&= X\cup S && [\ \textit{(5b) Identidad: } A\cap U = A\ ]\\
&= X && [\ Y\subseteq X\ ]\ \blacksquare
\end{aligned}$$

Propiedad 5.5.2

Sea S un itemset y $\langle \mathcal{L}, \{U\}\rangle$ un borde tal que $(\forall X \in \mathcal{L})\ S \subseteq X$ entonces

$$\langle \mathcal{L}, \{U\}\rangle = G(F(\langle \mathcal{L}, \{U\}\rangle, S), S)$$ □

Demostración:

$$\begin{aligned}
\langle \mathcal{L}, \{U\}\rangle &= \langle \{g(f(X,S),S) \mid X\in\mathcal{L}\}, \{g(f(U,S),S)\}\rangle && [\ \textit{Propiedad 5.5.1 y } S\subseteq X\ ]\\
&= G(\langle \{f(X,S) \mid X\in\mathcal{L}\}, \{f(U,S)\}\rangle, S) && [\ \textit{Definición de } G\ ]\\
&= G(F(\langle \{X \mid X\in\mathcal{L}\}, \{U\}\rangle, S), S) && [\ \textit{Definición de } F\ ]\\
&= G(F(\langle \mathcal{L}, \{U\}\rangle, S), S) && [\ \textit{Definición de } \mathcal{L}\ ]\ \blacksquare
\end{aligned}$$

A continuación, mostraremos que la operación de diferencia cuando el primer borde es genérico[11] es igual a realizar una transformación F de los bordes, realizar la diferencia tradicional y aplicar al resultado una transformación inversa G, es decir:

$$\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle = \underbrace{G(\underbrace{F(\langle\{X\},\{U\}\rangle,X)}-\underbrace{F(\langle\{\emptyset\},\mathcal{R}_2\rangle,X)},X)}$$

En el siguiente ejemplo se puede apreciar gráficamente como se realizar la diferencia extendida de bordes.

Ejemplo 5.5.4

Sea el borde $B_1 = \langle\{2,4\},\{1,2,3,4,5\}\rangle$ *y el borde* $B_2 = \langle\{\emptyset\},\{1,2,4,5\}\rangle$ *y el conjunto de items* $U = \{1,2,3,4,5\}$. *En la figura 5.11 se ilustra la aplicación de la función* F *sobre el borde* B_1 *y en la figura 5.12 la aplicación de la función* F *sobre el borde* B_2. *Si aplicáramos la definición de diferencia sobre itemsets entonces el borde resultante debería ser* $\langle\{2,3,4\},\{1,2,3,4,5\}\rangle$.

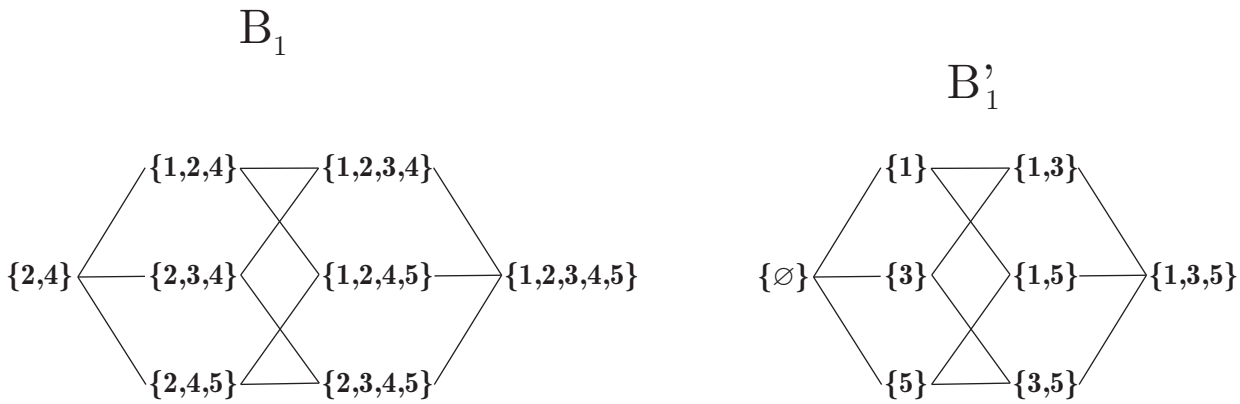


Figura 5.11: Transformación del borde B_1 por la aplicación de la función F.

Los bordes resultantes de la transformación aplicando la función F *con el itemset* $\{2,4\}$ *son* $B'_1 = \langle\{\emptyset\},\{1,3,5\}\rangle$ *y* $B'_2 = \langle\{\emptyset\},\{1,5\}\rangle$. *Si aplicamos la operación de diferencia sobre dichos bordes obtenemos como resultado un borde* $B'_3 = \langle\{3\},\{1,3,5\}\rangle$, *que se ilustra en la figura 5.13.*

[11]Es decir, no es cerrado por debajo pero respetando las condiciones establecidas al comienzo de la sección.

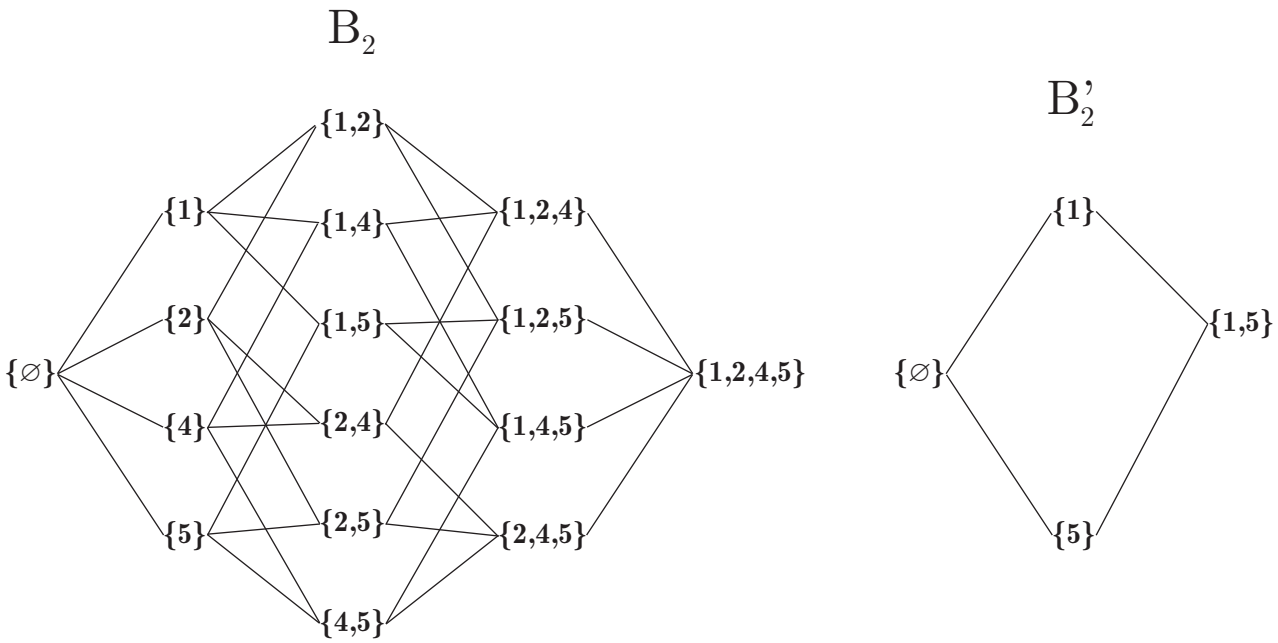


Figura 5.12: Transformación del borde B_2 por la aplicación de la función F.

Por último, aplicamos la función G con el itemset $\{2,4\}$ sobre dicho borde generando el borde $\langle\{2,3,4\},\{1,2,3,4,5\}\rangle$, el cual es igual al que se debería haber obtenido. Dicho borde se ilustra en la figura 5.13. ◊

Este resultado se demuestra formalmente en la proposición 5.5.7; sin embargo, por motivos de claridad y para facilitar la comprensión de la demostración, la misma ha sido dividida en varias proposiciones y propiedades, las cuales enunciaremos a continuación.

Propiedad 5.5.3

Sean S, Y e Y' itemsets tal que $S \subseteq Y \subseteq U$ y $S \cap Y' = \emptyset$.

$$Y = g(Y', S) \Leftrightarrow Y' = f(Y, S)$$ □

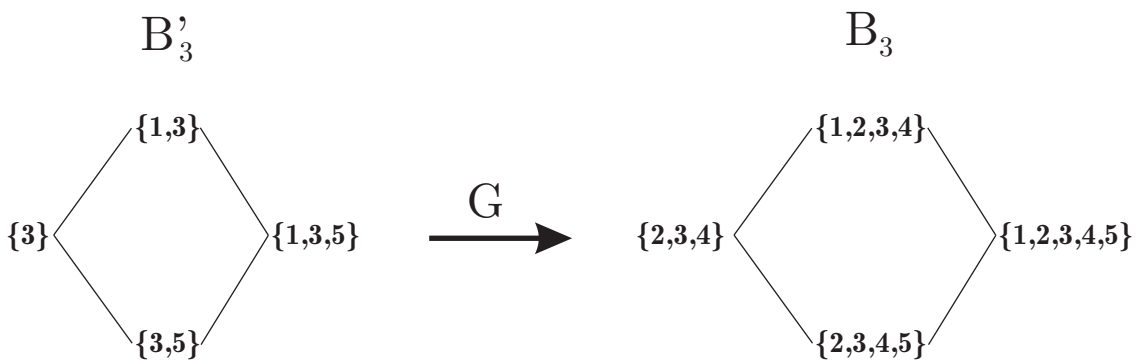


Figura 5.13: Transformación del borde B'_3 por la aplicación de la función G.

Demostración:

$$
\begin{aligned}
Y &= g(Y', S) \\
Y &= Y' \cup S && [\ \textit{Definición de } g\ ] \\
Y \cap S^c &= (Y' \cup S) \cap S^c \\
Y - S &= (Y' \cap S^c) \cup (S \cap S^c) && [\ \textit{Ley Distributiva y } A - B = A \cap B^c\ ] \\
Y - S &= (Y' \cap S^c) \cup \emptyset && [\ A \cap A^c = \emptyset\ ] \\
Y - S &= Y' \cap S^c = Y' - S && [\ \textit{Ley Identidad y } A - B = A \cap B^c\ ] \\
Y - S &= Y' && [\ S \cap Y' = \emptyset\ ] \\
f(Y, S) &= Y' && [\ \textit{Definición de } f\ ]
\end{aligned}
$$

■

La siguiente propiedad establece la correspondencia entre los itemsets de los reticulados resultante de aplicar las transformaciones.

Propiedad 5.5.4

Sea S un itemset y $\langle \mathcal{L}, \mathcal{R} \rangle$ un borde tal que $(\forall X \in \mathcal{L})\ S \subseteq X$ entonces

$$Y' \sqsubseteq F(\langle \mathcal{L}, \mathcal{R} \rangle, S) \ \Leftrightarrow\ Y \sqsubseteq \langle \mathcal{L}, \mathcal{R} \rangle \textit{ donde } Y = g(Y', S)$$ □

Demostración:

$\Rightarrow$ $(\exists X \in \mathcal{L}, \exists Z \in \mathcal{R})$

$$\begin{array}{ccccccc} f(X,S) & \subseteq & Y' & \subseteq & f(Z,S) & [\ \textit{Definición de } F\ ] \\ g(f(X,S),S) & \subseteq & g(Y',S) & \subseteq & g(f(Z,S),S) & [\ S \subseteq X\ ] \\ X & \subseteq & g(Y',S) & \subseteq & Z & [\ \textit{Propiedad 5.5.1}\ ] \\ X & \subseteq & Y & \subseteq & Z & \\ & & Y & \sqsubseteq & \langle \mathcal{L}, \mathcal{R} \rangle & [\ \textit{Definición de Borde}\ ] \end{array}$$

$\Leftarrow$ *Si se utiliza la demostración previa en orden inverso, se puede probar que existe* Y' *tal que* $f(X,S) \subseteq Y' \subseteq f(Z,S)$ *por lo tanto* $Y' \sqsubseteq F(\langle \mathcal{L}, \mathcal{R} \rangle, S)$. ■

La siguiente propiedad establece que si un itemset no pertenece a un borde entonces al aplicar la función de transformación el itemset resultante no puede pertenecer al borde transformado, es decir que, al aplicar la función de transformación f no pueden aparecer itemsets en el borde transformado que al aplicar luego la función g no hayan pertenecido al borde original.

Propiedad 5.5.5

Sea S *un itemset y* $\langle \{\emptyset\}, \mathcal{R} \rangle$ *un borde tal que* $(\forall Z \in \mathcal{R})\ S \subseteq Z \subseteq U$ *entonces*

$$Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R} \rangle \Leftrightarrow f(Y,S) \not\sqsubseteq F(\langle \{\emptyset\}, \mathcal{R} \rangle, S) \qquad \square$$

Demostración:

$\Rightarrow$ *Si* $Y \not\sqsubseteq \langle \{\emptyset\}, \mathcal{R} \rangle$ *entonces* $\emptyset \not\subseteq Y$ *o* $(\forall Z \in \mathcal{R})\ Y \not\subseteq Z$ *según la definición de borde*[12]. *Como* $\emptyset \subseteq Y$ *entonces* $(\forall Z \in \mathcal{R})\ Y \not\subseteq Z$.

$$F(\langle \{\emptyset\}, \mathcal{R} \rangle, S) = \langle \{\emptyset\}, \{Z - S \mid \forall Z \in \mathcal{R}\} \rangle \quad [\ \textit{Definición de } F \textit{ y } f\ ] \quad (5.42)$$

$$\exists y_i \in Y \textit{ tal que } (\forall Z \in \mathcal{R})\ y_i \notin Z \quad [\ \forall Z \in \mathcal{R}\ (Y \not\subseteq Z)] \quad (5.43)$$

$$y_i \notin S \quad [\ S \subseteq Z\ y\ (5.43)\ ] \quad (5.44)$$

$$y_i \in f(Y,S) \quad [\ \textit{Def. } f,\ (5.43)\ y\ (5.44)\ ] \quad (5.45)$$

$$f(Y,S) \not\sqsubseteq F(\langle \{\emptyset\}, \mathcal{R} \rangle, S) \quad [\ (5.42),\ (5.43)\ y\ (5.45)\ ]$$

[12] Ver definición 4.5.8 página 148.

$\Leftarrow$ *Si* $f(Y,S) \not\sqsubseteq F(\langle\{\emptyset\},\mathcal{R}\rangle,S) = \langle\{\emptyset\},\{Z-S \mid \forall Z\in\mathcal{R}\}\rangle$ *entonces* $\emptyset \not\subseteq f(Y,S)$ *o* $(\forall Z\in\mathcal{R})\ f(Y,S) \not\subseteq Z-S$ *según la definición de borde. Como* $\emptyset \subseteq f(Y,S)$ *entonces* $(\forall Z\in\mathcal{R})\ f(Y,S) \not\subseteq Z$.

$$\exists y'_i \in f(Y,S) \text{ tal que } (\forall Z\in\mathcal{R})\ y'_i \notin Z-S \quad [\ (\forall Z\in\mathcal{R})\ Y \not\subseteq Z\ ] \tag{5.46}$$

$$y'_i \in Y \quad [\ \text{Def. } f \text{ y } (5.46)\ ] \tag{5.47}$$

$$y'_i \notin S \quad [\ \text{Def. } f \text{ y } (5.46)\ ] \tag{5.48}$$

$$y'_i \notin Z \quad (\forall Z\in\mathcal{R}) \quad [\ (5.46) \text{ y } (5.48)\ ] \tag{5.49}$$

$$Y \not\sqsubseteq \langle\{\emptyset\},\mathcal{R}\rangle \quad [\ (5.47) \text{ y } (5.49)\ ] \quad \blacksquare$$

La siguiente propiedad nos permite asegurar que si la $\mathcal{R}_2$ no contiene ningún itemset igual al itemset de $\mathcal{R}_1$ entonces $\mathcal{R}_1$ será la frontera derecha del borde resultante de la diferencia y la frontera izquierda de dicho borde solamente estará constituida por itemsets superconjuntos a los itemsets de $\mathcal{L}_1$. Es importante mencionar que, en el caso de que exista algún itemset en $\mathcal{R}_2$ que sea superconjunto al itemset de $\mathcal{R}_1$ entonces la diferencia produce el borde vacío.

Propiedad 5.5.6

Sean $\langle\{X\},\{U\}\rangle$ *y* $\langle\{\emptyset\},\mathcal{R}_2\rangle$ *dos bordes tales que* $(\forall Z\in\mathcal{R}_2)\ X\subseteq Z\subset U$ *entonces* $\langle\mathcal{L},\{U\}\rangle = \langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$ *donde* $(\forall x_i\in\mathcal{L})\ X\subseteq x_i$. □

Demostración:

1. El itemset $\{U\}$ es la frontera derecha de $\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$.

   Dem: Como $\{U\}$ *es la frontera derecha de* $\langle\{X\},\{U\}\rangle$ *y* $(\forall Z\in\mathcal{R}_2)\ X\subseteq Z\subset U$ *existe un* $Y \sqsubseteq \langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$ *entonces* $Y\subseteq U$. *Esto se debe a que si* $Z\subset U$ *entonces existe al menos un* Y *en la diferencia* ($Y=U$). *Por lo tanto,* U *es el itemset superconjunto a todo itemset en la diferencia y en consecuencia es frontera derecha.*

2. Si $\mathcal{L}=\{x_1,x_2,\ldots,x_k\}$ es la frontera izquierda de la diferencia $\langle X,U\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle$ entonces $(\forall x_i\in\mathcal{L})\ X\subseteq x_i$.

Dem: Como $(\forall x_i \in \mathcal{L})\ x_i \sqsubseteq \langle X, U\rangle$ *y* $x_i \not\sqsubseteq \langle\{\emptyset\}, \{\mathcal{R}_2\}\rangle$ *entonces* $X \subseteq x_i \subseteq U$. ■

La siguiente propiedad establece el hecho que, si se realiza una operación de diferencia extendida y al resultado se le aplica una transformación utilizando la función F, el resultado sería igual a aplicar en primer lugar la trasformación utilizando la función F sobre cada uno de los bordes y luego realizar una operación de diferencia tradicional sobre los bordes resultantes. Formalmente,

Proposición 5.5.6

Sean los bordes $\langle\{X\}, \{U\}\rangle$ *y* $\langle\{\emptyset\}, \mathcal{R}_2\rangle$ *tal que* $(\forall Z \in \mathcal{R}_2)\ X \subseteq Z$ *y* $Z \subset U$.

$$F(\langle\{X\}, \{U\}\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle, X) = F(\langle\{X\}, \{U\}\rangle, X) - F(\langle\{\emptyset\}, \mathcal{R}_2\rangle, X) \qquad \blacklozenge$$

Demostración:

Demostraremos dicha igualdad por doble inclusión:

$(\subseteq)\ F(\langle\{X\}, \{U\}\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle, X) \subseteq F(\langle\{X\}, \{U\}\rangle, X) - F(\langle\{\emptyset\}, \mathcal{R}_2\rangle, X)$

Sea $Y' \sqsubseteq F(\langle\{X\}, \{U\}\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle, X)$.

$$\begin{aligned}
&(\forall Z \in \mathcal{R}_2)\ X \subseteq Z \subset U && [\ \textit{Hipótesis.}\ ]\\
&Y = g(Y', X) \sqsubseteq \langle\{X\}, \{U\}\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle && [\ \textit{Propiedad}\ (5.5.4)\ ]\\
&Y' = f(Y, X) && [\ \textit{Hip. y Propiedad}\ (5.5.3)\ ]\\
&Y \sqsubseteq \langle\{X\}, \{U\}\rangle && [\ \textit{Propiedad}\ (5.5.6)\ ]\\
&Y' = f(Y, X) \sqsubseteq F(\langle\{X\}, \{U\}\rangle, X) && [\ \textit{Hip. y Propiedad}\ (5.5.4)\ ]\\
&Y \not\sqsubseteq \langle\{\emptyset\}, \mathcal{R}_2\rangle && [\ \textit{Def. Diferencia}\ ]\\
&Y' = f(Y, X) \not\sqsubseteq F(\langle\{\emptyset\}, \mathcal{R}_2\rangle, X) && [\ \textit{Hip. y Propiedad}\ (5.5.5)\ ]
\end{aligned}$$

$$\therefore\ Y' \sqsubseteq F(\langle\{X\}, \{U\}\rangle, X) - F(\langle\{\emptyset\}, \mathcal{R}_2\rangle, X)$$

$(\supseteq)\ F(\langle\{X\}, \{U\}\rangle - \langle\{\emptyset\}, \mathcal{R}_2\rangle, X) \supseteq F(\langle\{X\}, \{U\}\rangle, X) - F(\langle\{\emptyset\}, \mathcal{R}_2\rangle, X)$

Let $Y' \sqsubseteq F(\langle\{X\},\{U\}\rangle, X) - F(\langle\{\emptyset\},\mathcal{R}_2\rangle, X)$.

$$
\begin{aligned}
&(\forall Z \in \mathcal{R}_2)\ X \subseteq Z \subset U && [\ \textit{Hipótesis}\ ]\\
&f(Y,X) = Y' \sqsubseteq F(\langle\{X\},\{U\}\rangle, X) && [\ \textit{Definición de } F\ ]\\
&Y \sqsubseteq \langle\{X\},\{U\}\rangle && [\ \textit{Hip. y Propiedad } (5.5.4)\ ]\\
&Y \not\sqsubseteq \langle\{\emptyset\},\{\mathcal{R}_2\}\rangle && [\ \textit{Propiedad } (5.5.5)\ ]\\
&Y \sqsubseteq \langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\{\mathcal{R}_2\}\rangle && [\ \textit{Def. Diferencia}\ ]\\
&Y' \sqsubseteq F(\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\{\mathcal{R}_2\}\rangle, X) && [\ \textit{Hip. y Propiedad } (5.5.4)\ ]
\end{aligned}
$$

■

Por último, la siguiente proposición nos provee de la equivalencia entre la diferencia extendida y la aplicación de transformaciones realizando la diferencia tradicional.

Proposición 5.5.7

La operación de diferencia del borde $\langle\{X\},\{U\}\rangle$ *con respecto al borde* $\langle\{\emptyset\},\mathcal{R}_2\rangle$ *tal que* $(\forall Z \in \mathcal{R}_2)\ X \subseteq Z \subset U$ *verifica la siguiente igualdad:*

$$\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle = G(\ F(\langle\{X\},\{U\}\rangle, X) - F(\langle\{\emptyset\},\mathcal{R}_2\rangle, X)\ , X) \qquad \blacklozenge$$

Demostración:

$$
\begin{aligned}
\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle &= G(F(\langle\{X\},\{U\}\rangle - \langle\{\emptyset\},\mathcal{R}_2\rangle, X), X) && [\ \textit{Propiedad } (5.5.2)\ ]\\
&= G(F(\langle\{X\},\{U\}\rangle, X) - F(\langle\{\emptyset\},\mathcal{R}_2\rangle, X), X) && [\ \textit{Proposición } (5.5.6)\ ]
\end{aligned}
$$

■

5.5.5. Algoritmo de Diferencia Extendida de Bordes

Las proposiciones mencionadas en las secciones previas son utilizadas para simplificar la operación de diferencia y transformar los bordes en entradas válidas para el algoritmo de diferencia tradicional. En la figura 5.14 se ilustran los pasos necesarios para realizar la operación de diferencia. Es importante observar, que las funciones F y G son aplicadas

sobre bordes que en su mayoría consisten de fronteras con un único itemset y por tal motivo dichas funciones pueden ser computadas eficientemente. A continuación mostraremos el algoritmo que realiza la operación de diferencia extendida, el cual está basado en el algoritmo 6 y las proposiciones antes mencionadas.

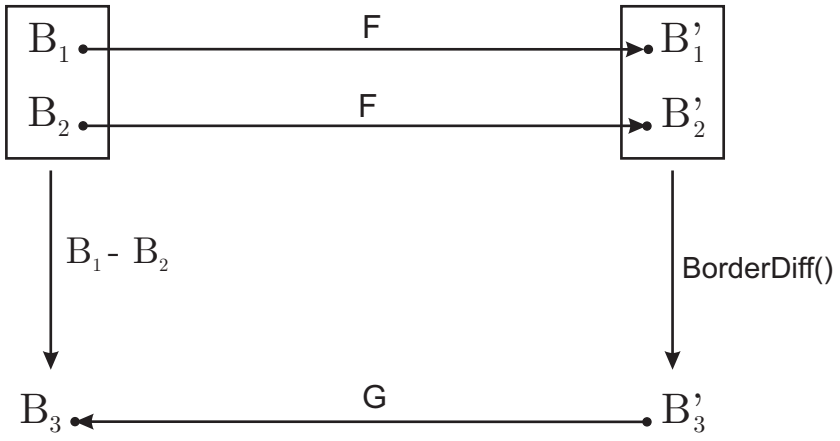


Figura 5.14: Transformaciones necesarias para la operación de diferencia extendida.

Algoritmo 8 EXTENDED BORDER-DIFFERENTIAL($\langle \mathcal{L}_1, \{U\}\rangle$, $\langle \{\emptyset\}, \mathcal{R}_2\rangle$)

1: $\mathcal{L}'_1 = \{x_i \in \mathcal{L}_1 \mid (\exists z_j \in \mathcal{R}_2)\ x_i \subseteq z_j\}$;
2: $\mathcal{L}''_1 = \mathcal{L}_1 - \mathcal{L}'_1$; {Proposición 5.5.4}
3: $NewL = \mathcal{L}''_1$;
4: **Para todo** $x_i \in \mathcal{L}'_1$ **hacer** {Proposición 5.5.3}
5: $\quad \mathcal{R}'_2 = \{z_j \in \mathcal{R}_2 \mid x_i \subset z_j\}$; {Proposición 5.5.5}
6: $\quad B :=$ BorderDiff($\mathbf{F}(\langle\{x_i\}, \{U\}\rangle, x_i), \mathbf{F}(\langle\{\emptyset\}, \mathcal{R}'_2\rangle, x_i)$);
7: $\quad \langle \mathcal{L}, \mathcal{R}\rangle := \mathbf{G}(B, x_i)$; {Proposición 5.5.7}
8: $\quad NewL := NewL \cup \mathcal{L}$; {Proposiciones 5.5.3 y 5.5.4}
9: **fin Para**
10: **retornar** $\langle NewL, \{U\}\rangle$;

5.6. Unión de sectores

Hemos visto como se puede explorar el plano de soportes para obtener los EPs pertenecientes a cada sector. También ha sido mencionado que los sectores son conjuntos disjuntos de EPs y en consecuencia el conjunto resultante de EPs está dado por la unión de todos

Algoritmo 9 function F($\langle \mathcal{L}, \mathcal{R} \rangle$, X)

1: $NewL = \emptyset$;
2: $NewR = \emptyset$;
3: **Para todo** $x_i \in \mathcal{L}$ **hacer**
4: $\quad NewL = NewL \cup (x_i - X)$;
5: **fin Para**
6: **Para todo** $z_j \in \mathcal{R}$ **hacer**
7: $\quad NewR = NewR \cup (z_j - X)$;
8: **fin Para**
9: **return** $\langle NewL, NewR \rangle$

Algoritmo 10 function G($\langle \mathcal{L}, \mathcal{R} \rangle$, X)

1: $NewL = \emptyset$;
2: $NewR = \emptyset$;
3: **Para todo** $x_i \in \mathcal{L}$ **hacer**
4: $\quad NewL = NewL \cup (x_i \cup X)$;
5: **fin Para**
6: **Para todo** $z_j \in \mathcal{R}$ **hacer**
7: $\quad NewR = NewR \cup (z_j \cup X)$;
8: **fin Para**
9: **return** $\langle NewL, NewR \rangle$

estos sectores (conjuntos de EPs). Esta operación de unión entre conjuntos de EPs puede realizarse expandiendo todos los bordes y realizando la unión de conjuntos tradicional. El principal problema de este acercamiento es que se necesita realizar la expansión de los bordes que representan los EPs de cada sector, generando una gran cantidad de itemsets. Un acercamiento más interesante consistiría en realizar la unión de los EPs de estos sectores utilizando su representación de borde convexo y obtener una nueva representación concisa de borde convexo, sin realizar en ningún momento la expansión de bordes, es decir, simplemente operando sobre las fronteras de los bordes de cada sector. Sin embargo, hemos identificado algunas dificultades para realizar dicha unión entre bordes que analizaremos en esta sección.

La primer propuesta para realizar la unión consiste simplemente en realizar la unión de las fronteras izquierdas generando una nueva frontera izquierda y en forma análoga con las fronteras derechas. Este propuesta se basa en que los bordes son disjuntos por lo cual no existen itemsets repetidos entre fronteras. Sin embargo, la simple unión de fronteras izquierdas y derechas de bordes disjuntos puede resultar en un borde no representado en forma concisa, tal como se ilustra en el siguiente ejemplo.

Ejemplo 5.6.1

Sean $B_1 = \langle\{\{a,b\},\{b,c\}\},\{\{a,b,c\}\}\rangle$ *y* $B_2 = \langle\{\{b\}\},\{\{a,b\},\{b,c\}\}\rangle$ *bordes genéricos disjuntos. Si se realiza la unión de fronteras entre dichos bordes se obtiene la unión* $B_1 \cup B_2 = \langle\{\{b\},\{a,b\},\{b,c\}\},\{\{a,b\},\{b,c\},\{a,b,c\}\}\rangle$. *Sin embargo, el resultado deseado consistía de* $\langle\{\{b\}\},\{\{a,b,c\}\}\rangle$. *Esta situación se representa en la figura 5.15.* ◊

Esta situación puede evitarse si se evalúa que la frontera izquierda solo tenga itemsets minimales, es decir, que no exista ningún itemset en la frontera izquierda que tenga un subconjunto propio que también pertenezca a la frontera izquierda; y de la misma forma, que la frontera derecha solo contenga itemsets maximales. Sin embargo, este proceso es costoso debido a los testeos por subconjuntos necesarios para cada itemset de las fronteras.

A pesar de dicha solución a este problema, existe otro problema aún no resuelto más complejo. Como ha sido notado en [20, 43], los bordes convexos no son cerrados bajo las

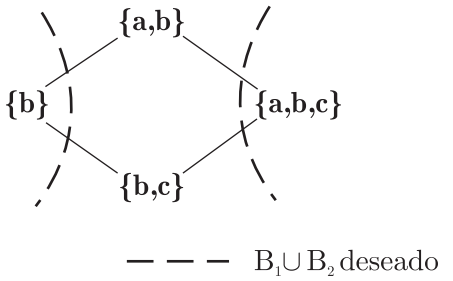


Figura 5.15: Unión de bordes

operaciones de unión e intersección. Esto significa que, al aplicar una operación de unión o intersección sobre un par de bordes convexos puede resultar en un borde que *no sea* convexo. Este hecho se ilustra en el siguiente ejemplo. Es importante aclarar que en el lema 4.5.3 se hace referencia a que la intersección de un par de bordes resulta en un espacio convexo. Sin embargo, los bordes convexos utilizados para aplicar dicho lema son cerrados por debajo, restricción que no es válida en nuestro contexto. Como hemos mencionado oportunamente, en [34] se exploran diferentes operaciones entre espacios convexos aunque no se provee una operación de unión para nuestro contexto.

Ejemplo 5.6.2

Sean $B_1 = \langle\{\{1\}\}, \{\{1,2,3\}\}\rangle$ *y* $B_2 = \langle\{\{3\}\}, \{\{1,3,4\}\}\rangle$. *Dichos bordes convexos generan los siguientes conjuntos de itemsets:*

$$B_1 = [\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}]$$
$$B_2 = [\{3\}, \{1,3\}, \{3,4\}, \{1,3,4\}]$$

Si se produce una expansión de bordes y luego se realiza la unión de dichos conjuntos se obtiene $B_1 \cup B_2 = [\{1\}, \{3\}, \{1,2\}, \{1,3\}, \{3,4\}, \{1,2,3\}, \{1,3,4\}]$.

Sin embargo, si se produce una unión de borde realizando la unión de las fronteras se produce el siguiente borde $B_1 \cup B_2 = \langle\{\{1\}, \{3\}\}, \{\{1,2,3\}, \{1,3,4\}\}\rangle$. *Sin embargo, el conjunto resultante de este borde difiere con respecto al conjunto anterior ya que los conjuntos* $\{2,3\}$ *y* $\{1,4\}$ *que no aparecen en la unión original, si aparecen en el borde generado. Esta situación se ilustra en la figura 5.16.* ◊

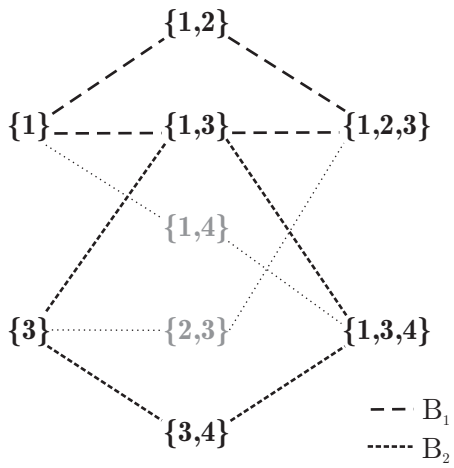


Figura 5.16: Generación errónea de itemsets en la unión de bordes (1)

Podríamos entonces encontrar una situación donde ocurran ambos problemas, como se puede apreciar en el siguiente ejemplo:

Ejemplo 5.6.3

Sean $B_1 = \langle\{\{1\}\}, \{\{1,2,3\}\}\rangle$ *y* $B_2 = \langle\{\{1,3\}\}, \{\{1,3,4\}\}\rangle$. *Realizando la unión de fronteras se observa que el itemset* $\{1,4\}$ *pertenece a la unión pero no pertenecía a ninguno de los bordes iniciales (*B_1 *y* B_2*). Por otro lado, el borde resultante de* $B_1 \cup B_2$ *no tiene fronteras minimales ya que* $B_1 \cup B_2 = \langle\{\{1\}, \{1,3\}\}, \{\{1,2,3\}, \{1,3,4\}\}\rangle$ *donde puede observarse que el itemsets* $\{1,3\}$ *es un superconjunto del itemset* $\{1\}$. *Esta situación se ilustra en la figura 5.17.* $\Diamond$

Podemos entonces formalizar estas observaciones para detectar bajo que circunstancias se produce un conflicto en la unión de bordes al unir simplemente fronteras. Diremos que se producirá un conflicto de unión cuando el siguiente conjunto sea distinto de vacío:

$$\{Y \mid (\exists X \in \mathcal{L}_i, \exists Z \in \mathcal{R}_j, X \subset Y \subset Z) \ \wedge \ Y \not\sqsubseteq \ \langle\{\emptyset\}, \mathcal{R}_i\rangle \ \wedge \ Y \not\sqsubseteq \ \langle\mathcal{L}_j, \mathcal{R}_j\rangle\}$$

Utilizando dicha evaluación realizaremos simplemente las uniones de aquellos bordes que no tengan conflictos intentando combinar aquellos bordes que quedaron fuera de esta primer combinación. Es importante mencionar que esta unión de bordes la hemos tratado

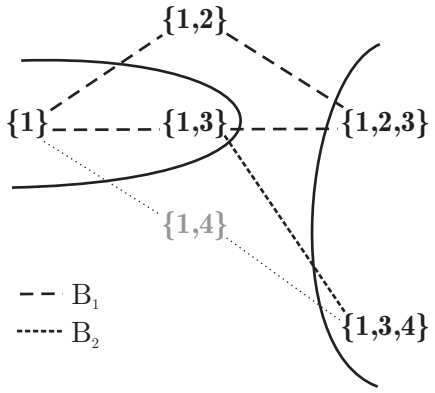


Figura 5.17: Generación errónea de itemsets en la unión de bordes (2)

como un proceso anexo a la búsqueda de EPs cuando se realiza la comparación con los otros métodos. Esto se debe a que no hemos encontrado aún un procedimiento efectivo para realizar esta unión. Por otro lado, los métodos alternativos analizados tampoco retornan un borde convexo que consiste de todos los EPs y en consecuencia este mismo problema de realizar la unión de bordes convexos se presenta en tales alternativas.

5.7. Conclusiones

En el capítulo 4 hemos mencionado que la principal desventaja del acercamiento propuesto en [43, 42, 20] es que la aproximación que se realiza sobre el conjunto de EPs no es muy exacta. Como consecuencia, patrones emergentes potencialmente significativos podrían quedar fuera del conjunto de EPs generados por este acercamiento. Por tal motivo, en este capítulo se analizan distintas alternativas para una generación más aproximada al conjunto completo de EPs. Hemos analizado dos alternativas propuestas en [43] y hemos propuesto una tercer alternativa que permite calcular el conjunto de EPs en forma incremental, sacando provecho de esta forma al trabajo previamente realizado. En consecuencia, este acercamiento logra una mayor performance que las otras alternativas. Por otro lado, los conjunto de EPs generados por nuestro acercamiento son disjuntos evitando de esta forma la generación de EPs duplicados.

Este acercamiento propuesto necesita de un enfoque teórico diferente al utilizado pre-

viamente. Por tal motivo, hemos generalizado la operación de diferencia para permitir la utilización de bordes más genéricos y hemos reutilizado la información de los sectores para realizar la minería en forma incremental. Estas propuestas están fundamentadas mediante las pruebas formales correspondientes. Por último, hemos identificado los problemas subyacentes en la operación de unión de bordes convexos, los cuales impiden proveer como resultado un solo borde convexo.

Capítulo 6

Conclusiones

Mencionaremos a continuación las principales conclusiones obtenidas junto con las posibles alternativas para continuar con este trabajo.

6.1. Contribuciones de esta tesis

A lo largo de esta tesis, en primer lugar hemos analizado el estado actual del arte en el campo de la representación concisa de patrones frecuentes y patrones emergentes. También hemos propuesto una forma para obtener un conjunto más completo de patrones emergentes lo cual posibilita la generación de clasificadores más exactos.

En particular, en los capítulos 1 y 2 se proveen las nociones fundamentales del trabajo de minería de datos (data mining). Se analiza como convertir grandes volúmenes de datos en información utilizando patrones y de esta forma proveer al usuario con información relevante, significativa, interesante y previamente desconocida. Se muestra la estructura de los patrones y como éstos pueden ser clasificados en interesantes y irrelevantes. Se analiza además alternativas para el almacenamiento de los datos como también representaciones temporales durante la minería de datos. El uso apropiado de estas alternativas influye directamente sobre la performance y la escalabilidad de los algoritmos de minería de patrones. Por último, se muestran los algoritmos más emblemáticos para la tarea de minería de patrones frecuentes. Dichos algoritmos proveen ideas y heurísticas utilizadas por la mayoría de los algoritmos de minería de patrones cualquiera sea la clase de patrón utilizada.

En el capítulo 3 se analizan distintas alternativas para representar el conjunto de patrones frecuentes en forma eficiente. Por tal motivo, se proponen distintas alternativas que reducen el espacio requerido para representar todo el conjunto de patrones frecuentes. Estas representaciones son de vital importancia en esta tesis ya que su utilización hace viable las propuestas realizadas en los subsiguientes capítulos. Se analizan por tal motivo las ventajas y desventajas de cada representación y para cada una de ellas se comparan los distintos algoritmos presentes. A pesar que se muestran la mayoría de las representaciones concisas alternativas de patrones frecuentes, se ha hecho hincapié en las dos representaciones que proveen una mayor ganancia en espacio y poder de síntesis. Debido a la gran diversidad de representaciones alternativas y de algoritmos propuestos para cada alternativa, hemos analizado los algoritmos principales propuestos para cada representación y comparado las diferentes ventajas que proveen cada uno. Por otro lado, en forma más abstracta hemos comparado los pro y contras de las diferentes representaciones alternativas. De esta forma, al comparar las representaciones y algoritmos, hemos explorado en profundidad el estado actual de la representación de patrones frecuentes.

En el capítulo 4 se analiza el enfoque utilizado para la generación de patrones emergentes y su área de aplicación. Se proveen los fundamentos que permiten generar patrones emergentes en forma eficiente utilizando una representación concisa de patrones frecuentes y realizando operaciones sobre espacios convexos. Se analizaron las distintas aplicaciones propuestas para esta clase de patrones y las distintas clases de patrones emergentes que han dado lugar a clasificadores más exactos. Se proveen, además, los conceptos básicos que permiten entender el acercamiento propuesto y de esta forma proveer un marco de trabajo para el aporte propuesto. También, se muestran otras clases de patrones emergentes que permiten proveer al usuario de información más detallada sobre el dominio analizado. Por último, se provee un resumen sobre la relación de entre los espacios convexos de patrones emergentes y los espacios de versiones propuestos por Mitchell [53].

Los acercamientos actuales utilizan un subconjunto de los patrones emergentes para detectar tendencias y realizar tareas de clasificación. A pesar de ello, tales acercamientos han demostrado ser muy efectivos. En el capítulo 5 se analizan distintas alternativas para

una generación de patrones emergentes más completa, intentando generar un conjunto más completo de patrones emergentes sin degradar la performance significativamente. Analizamos dos propuestas por otros autores y proponemos una nueva alternativa, la cual se diferencia de las demás en que permite reutilizar información previamente obtenida. Esto permite reducir la cantidad de trabajo necesario para obtener el conjunto de patrones emergentes. Para nuestra propuesta nos hemos basado en el método de aproximación del área para funciones continuas conocida como *Suma de Riemman*. Una nueva operación de diferencia sobre bordes fue necesaria para poder llevar a cabo tal propuesta. Por ello se provee en dicho capítulo las pruebas formales que demuestran la correctitud de nuestra propuesta.

6.2. Líneas de trabajo futuro

Hemos demostrado formalmente la validez del acercamiento incremental y de las estrategias de poda las cuales reducen significativamente el espacio de búsqueda. Sin embargo, creemos que es importante verificar dichos resultados en forma empírica. Por tal motivo, éste es el paso más inmediato a seguir a futuro.

Luego de realizar tal comprobación, existen varias alternativas a seguir:

- *Mejorar la exactitud de los clasificadores basados en EPs utilizando un conjunto más completo de EPs.*

  En primer lugar, se debería intentar sacar provecho a este forma de calcular un conjunto más completo de EPs para mejorar la exactitud de los actuales clasificadores basados en EPs. Esto no significa que se utilicen más instancias de entrenamientos sino instancias que permitan clasificar mejor que mediante los acercamiento previos eran muy difíciles de poder ser detectadas debido a su bajo soporte.

- *Solucionar el problema de la unión de bordes convexos.*

  En segundo lugar, se puede intentar mejorar el acercamiento propuesto representando en forma aún más concisa la solución. De esta forma, en lugar de retornar una

cantidad de bordes igual a la cantidad de sectores, unir los sectores que no posean problemas. Este paso, costoso en algunas instancias, logra reducir significativamente el espacio necesario para los bordes ya que como hemos visto, cuanto más patrones representan los bordes, menor es la cantidad de itemsets necesarios en la frontera para representarlos.

- *Aplicar la estrategia de transformación en otros dominios* (*Espacios de Versiones*).

  Por último, en la sección 4.10 hemos mencionado la relación que existe entre los patrones emergentes y, principalmente, los espacios de versiones. Por tal motivo, es interesante analizar si la operación de diferencia extendida tiene algún tipo de aplicación bajo este contexto.

Bibliografía

[1] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Knowledge Discovery and Data Mining*, pages 108–118, 2000.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD Conference*, pages 207–216. ACM Press, 1993.

[3] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1982. AHO a 82:1 P-Ex.

[4] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. Fast algorithms for mining emerging patterns. In Elomaa et al. [25], pages 39–50.

[5] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In J. Lloyd, V. Dahl, and U. Furbach, editors, *Computational Logic – CL 2000. Proc. CL '00. LNAI 1861*, pages 972–986. Springer, Heidelberg 2000, 2000.

[6] Roberto J. Bayardo. Efficiently mining long patterns from databases. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD Conference*, pages 85–93. ACM Press, 1998.

[7] Russell Bertran. *Introduction to Mathematical Philosophy*. New York: Simon and Schuster, 1971.

[8] Inderpal S. Bhandari, Edward Colet, Jennifer Parker, Zachary Pines, Rajiv Pratap, and Krishnakumar Ramanujam. Advanced scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, 1(1):121–125, 1997.

[9] Peter Bollmann-Sdorra, Alaaeldin M. Hafez, and Vijay V. Raghavan. A theoretical framework for association mining based on the boolean retrieval model. *Lecture Notes in Computer Science*, 2114:21–55, 2001.

[10] Endre Boros, Vladimir Gurvich, Leonid Khachiyan, and Kazuhisa Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In Helmut Alt and Afonso Ferreira, editors, *STACS*, volume 2285 of *Lecture Notes in Computer Science*, pages 133–141. Springer, 2002.

[11] J. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries, 2003.

[12] Jean-Francois Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 62–73, 2000.

[13] Jean-Francois Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queris by means of free-sets. In *Principles of Data Mining and Knowledge Discovery*, pages 75–85, 2000.

[14] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *ICDE*, pages 443–452, 2001.

[15] Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Computer Science*, pages 74–85. Springer-Verlag, 2002.

[16] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering*, 13(1):64–78, 2001.

[17] Richard Jeffrey Cole. *Document Retrieval using Formal Concept Analysis.* PhD thesis, School of Information Technology, Griffith University, 2001.

[18] Vladan Devedzic. Knowledge discovery and data mining in databases.

[19] Vladan Devedzic. Knowledge modeling - state of the art.

[20] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. *ACMKDD*, 1999.

[21] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. Caep: Classification by aggregating emerging patterns. In Setsuo Arikawa and Koichi Furukawa, editors, *Discovery Science*, volume 1721 of *Lecture Notes in Computer Science*, pages 30–42. Springer, 1999.

[22] Brian Dunkel and Nandit Soparkar. Data organization and access for efficient data mining. In IEEE Computer Society, editor, *In Proc. of the 15th ICDE Int. Conf. on Data Engineering*, pages 522–529, 1999.

[23] Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, 1995.

[24] Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.

[25] Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors. *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2431 of *Lecture Notes in Computer Science*. Springer, 2002.

[26] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases - an overview. *Ai Magazine*, 13:57–70, 1992.

[27] B. Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Heidelberg, Springer, 1999.

[28] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. DEMON: Mining and monitoring evolving data. *Knowledge and Data Engineering*, 13(1):50–63, 2001.

[29] Bart Goethals. *Efficient Frequent Pattern Mining.* PhD thesis, University of Limburg, Diepenbeek, Belgium, December 2002.

[30] Bart Goethals and Mohammed Javeed Zaki. Advances in frequent itemset mining implementations: Report on fimi'03, 2003.

[31] Karam Gouda and Mohammed Javeed Zaki. Efficiently mining maximal frequent itemsets. In *ICDM*, pages 163–170, 2001.

[32] Gosta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *FIMI'03 Workshop on Frequent Itemset Mining Implementations*, November 2003.

[33] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS*, pages 209–216. ACM Press, 1997.

[34] Carl Gunter, Teow-Hin Ngair, and Devika Subramanian. The common order-theoretic structure of version spaces and atmss. *Artificial Intelligence*, 95:357–407, 1997.

[35] J. Han, Y. Cai, N. Cercone, and Y. Huang. Discovery of data evolution regularities in large databases, 1994.

[36] Jiawei Han and Yongjian Fu. Exploration of the power of attribute-oriented induction in data mining. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhr Smyth, and

Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AIII Press/MIT Press, 1996.

[37] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 3 edition, August 2000.

[38] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press, 05 2000.

[39] Joachim Hereth, Gerd Stumme, Rudolf Wille, and Uta Wille. Conceptual knowledge discovery and data analysis. In *International Conference on Conceptual Structures*, pages 421–437, 2000.

[40] Haym Hirsh. Generalizing version spaces. *Machine Learning*, 17(1):5–46, 1994.

[41] Marzena Kryszkiewicz and Marcin Gajek. Concise representation of frequent patterns based on generalized disjunction-free generators. In *PAKDD*, pages 159–171, 2002.

[42] Jinyan Li. *Mining Emerging Patterns to Construct Accurate and Efficient Classifiers*. PhD thesis, The University of Melbourne, January 2001.

[43] Jinyan Li and Guozhu Dong. Mining border description of emerging patterns from dataset pairs. Technical report, Wright University, USA, 2004.

[44] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In Djamel A. Zighed, Henryk Jan Komorowski, and Jan M. Zytkow, editors, *PKDD*, volume 1910 of *Lecture Notes in Computer Science*, pages 191–200. Springer, 2000.

[45] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowl. Inf. Syst.*, 3(2):131–145, 2001.

[46] Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao, and Limsoon Wong. Deeps: A new instance-based lazy discovery and classification system. *Machine Learning*, 54(2):99–124, 2004.

[47] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In Pat Langley, editor, *ICML*, pages 551–558. Morgan Kaufmann, 2000.

[48] Jinyan Li and Limsoon Wong. Geography of differences between two classes of data. In Elomaa et al. [25], pages 325–337.

[49] Jinyan Li and Limsoon Wong. Solving the fragmentation problem of decision trees by discovering boundary emerging patterns. In *ICDM*, pages 653–656. IEEE Computer Society, 2002.

[50] Dao-I Lin and Zvi M. Kedem. Pincer search: A new algorithm for discovering the maximum frequent set. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, *EDBT*, volume 1377 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 1998.

[51] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[52] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[53] Tom M. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, 1978.

[54] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[55] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*, 1540:398–416, 1999.

[56] Jian Pei, Jiawei Han, and Runying Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[57] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W.J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, MA, 1991.

[58] Uta Priss. Introduction to and overview of formal concept analysis. to appear in Annual Review of Information Science and Technology, Vol. 40, 2004.

[59] J. R Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[60] J. R Quinlan. Decision trees and multi-valued attributes. *Machine Intelligence*, (11):305–318, 1988.

[61] J. R Quinlan. C4.5: Programs for machine learning. *Morgan Kaufmann series in machine learning*, 1993.

[62] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[63] John F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.

[64] Kenneth Ross and Charles Wright. *Discrete Mathematics*. Prentice Hall, 2 edition, 1988.

[65] Stuart Russell and Peter Norving. *Artificial Intelligence: A modern approach*. Prentice-Hall, 1995.

[66] Ron Rymon. Search through systematic set enumeration. In *Proc. Of Third International Conf. On Principles of Knowledge Representation and Reasoning*, pages 539–550, 1992.

[67] Ron Rymon. An SE-tree based characterization of the induction problem. In *International Conference on Machine Learning*, pages 268–275, 1993.

[68] Michèle Sebag. Delaying the choice of bias: A disjunctive version space approach. In *ICML*, pages 444–452, 1996.

[69] Pradeep Shenoy, Jayant R. Haritsa, S. Sundarshan, Gaurav Bhalotia, Mayank Bawa, and Devavrat Shah. Turbo-charging vertical mining of large databases. In *In Proc. of the 16th ICDE Int. Conf. on Data Engineering*, pages 22–33, 2000.

[70] Avi Silberschatz, Henry Korth, and Sudarshan. *Database System Concepts*. McGraw-Hill, 3 edition, 1998.

[71] Evgueni Smirnov and Peter J. Braspenning. Version space learning with instance-based boundary sets. In *ECAI*, pages 460–464, 1998.

[72] Gerd Stumme. Conceptual knowledge discovery with frequent concept lattices.

[73] Gerd Stumme. Formal concept analysis on its way from mathematics to computer science. In U. Priss, D. Corbett, and G. Angelova, editors, *Conceptual Structures: Integration and Interfaces*, pages 2–19. Springer, 2002. Proc. ICCS 2002, LNAI 2393, Springer, Heidelberg 2002.

[74] Gerd Stumme, Rudolf Wille, and Uta Wille. Conceptual knowledge discovery in databases using formal concept analysis methods. In *Principles of Data Mining and Knowledge Discovery*, pages 450–458, 1998.

[75] Thomas Alan Tilley. *Formal Concept Analysis applications to Requirements Engineering and Design*. PhD thesis, School of Information Technology and Electrical Engineering, University of Queensland, December 2004.

[76] Hannu Toivonen. *Discovery of frequent patterns in large data collections*. Report a-1996-5, University of Helsinki, Department of Computer Science, November 1996. ISBN 951-45-7531-8.

[77] A.A. Veloso, B. Gusmao, W. Meira, M. Carvalho, S. Parthasarathy, and M. Zaki. Efficiently mining approximate models of associations in evolving databases. In *6th European Conference on Principles of Knowledge Discovery in Databases*, August 2002.

[78] A.A. Veloso, W. Meira, M. Carvalho, S. Parthasarathy, and M. Zaki. Parallel, incremental and interactive mining for frequent itemsets in evolving databases. In *Proceedings of the 3nd SIAM International Conference on Data Mining*, San Francisco, USA, May 2003.

[79] A.A. Veloso, W. Meira, Jr., M.B. de Carvalho, B. Possas, S. Parthasarathy, and M. Javeed Zaki. Mining frequent itemsets in evolving databases. In *Proceedings of the 2nd SIAM International Conference on Data Mining*, Arlington, Virginia, USA, April 2002.

[80] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, USA, 2003.

[81] Bastian Wormuth and Peter Becker. Introduction to formal concept analysis. Slides - Tutorial session during ICFCA'04, 2004.

[82] Beat Wüthrich and Kamalakar Karlapalem. Data mining opportunities in very large object oriented databases.

[83] Xindong Wu, Chengqi Zhang, and Shichao Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems*, 22(3):381–405, July 2004.

[84] B. Wuthrich. Knowledge discovery in databases, 1995.

[85] Mohammed Zaki and Karam Gouda. Fast vertical mining using diffsets. Technical Report 01-1, Rensselaer Polytechnic Institute, 2001.

[86] Mohammed Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. *2nd SIAM International Conference on Data Mining, Arlington*, April 2002.

[87] Mohammed J. Zaki. *Scalable Data Mining for Rules*. PhD thesis, University of Rochester, July 1998.

[88] Mohammed J. Zaki and Mitsunori Ogihara. Theoretical foundations of association rules. In *In Proceedings of 3 rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, 1998.

[89] Mohammed J. Zaki and Benjarath Phoophakdee. Mirage: A framework for mining, exploring and visualizing minimal association rules.

[90] Qinghua Zou, Wesley W. Chu, and Baojing Lu. Smartminer: A depth first algorithm guided by tail information for mining maximal frequent itemsets.