UNIVERSIDAD DEL BÍO-BÍO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

Algoritmo De Minería De Datos, Búsqueda De Patrones Periódicos En Función Del Tiempo En Base De Datos Espacio-Temporales

MEMORIA PARA OPTAR AL TÍTULO DE

INGENIERO CIVIL EN INFORMÁTICA

Autores

Guillermo Rafael Fuentes Flores

Danilo Andrés Rosas Arévalo

Profesor Guía

Dr. Claudio Gutiérrez Soto

RESUMEN

En los últimos años las Bases de Datos Espacio-Temporales han alcanzado un gran interés por parte de la comunidad científica, esto debido a la creciente demanda de servicios de móviles basadas en la localización, al progreso de los sistemas de posicionamiento global (GPS) así como también a los sistemas de Información Geográficos (SIG). Más aún, la creciente generación de información (del orden de los terabytes, como por ejemplo en los centros astronómicos) en diversos ámbitos de la ciencia y aplicaciones convencionales, han dado lugar a un nuevo concepto llamado Big Data Analytics. Big Data Analytics (note que los fundamentos de Minería de Datos son comunes para esta nueva área, por lo que la diferencia radica en la cantidad de información) agrupa un conjunto de metodologías y desarrollos matemáticos para almacenar, analizar y cruzar información con el objeto de encontrar patrones de comportamiento ocultos en los datos. Un patrón es una característica o rasgo constante y recurrente que ayuda a identificar un fenómeno o problema y sirve como indicador o modelo para predecir su comportamiento futuro. Así uno de los principales objetivos de este proyecto de título, es analizar y diseñar e implementar un algoritmo eficiente para detectar patrones secuenciales. Un patrón secuencial es una secuencia ordenada de eventos en el tiempo. Con el objeto de corroborar la eficiencia de nuestro algoritmo hemos implementado dos algoritmos ampliamente utilizados en la literatura, el algoritmo Apriori y el algoritmo Max-Subpattern. Para verificar la correctitud y eficiencia de nuestro algoritmo denominado Minus-F1 hemos utilizado datos sintéticos. Adicionalmente, hemos verificado la eficiencia (el tiempo que toma el algoritmo en encontrar patrones) de los tres algoritmos utilizando una porción de una base de datos real llamada GeoLife. Para lograr esto los datos de GeoLife han sido preprocesados y los eventos espacio-temporales han sido codificados como secuencias de texto (eventos en el espacio-tiempo), lo cual permite un mejor procesamiento. Los algoritmos fueron implementados en Java y los resultados empíricos finales muestran que nuestro algoritmo supera ampliamente a Apriori y Max-Subpttern en cuanto al ahorro de procesamiento, por lo que esta propuesta resulta ser la más eficiente.

Finalmente, una importante contribución de este proyecto de título, es que presentamos el orden en función del tiempo de los tres algoritmos, ya que en la literatura actual no es posible encontrar dicho orden para los algoritmos Apriori y Max-Subpattern. El orden del algoritmo Apriori es $O\left(m^{\frac{n}{2}} * n\right)$, el orden del Max-

Subpattern es $O\left(\sqrt[4]{\left(\frac{m*n}{2}\right)^{n^2}} * n\right)$ y el orden de nuestro algoritmo Minus–F_1 es $O\left(\frac{n^3}{\frac{n^2}{4}}\right) \approx O(n)$.

Los resultados muestran a Minus-F_1 como el más eficiente respecto al tiempo que toma en evaluar patrones periódicos sobre una misma secuencia de datos, comparados con Apriori y M-SP, con diferencias de tiempo mayores al 100% entre Minus-F_1 y M-SP y mayores al 500% entre Minus-F_1 y Apriori.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el último tiempo se ha producido un desarrollo importante de las Bases De Datos Espacio-Temporales, dada la necesidad de analizar la evolución temporal del comportamiento espacial de gran cantidad de datos que son de interés para diversos estudios según el área de investigación, por ejemplo, en biología marina, el análisis espacio temporal de un cardumen de peces permite determinar el movimiento de estos, en que temporada realizan migración o inclusive determinar si debe vetarse su pesca. En otra área, como la medicina, puede utilizarse el análisis espacio temporal sobre un cultivo de bacterias, determinar su nivel de evolución/mutación y/o grado de propagación de una epidemia [1]. Por otro lado, en un área tan distinta como la astronomía podría utilizar este tipo de análisis para determinar el tipo de trayectoria de un grupo de asteroides o la velocidad de desplazamiento de las estrellas.

Todas estas necesidades de información han provocado un importante avance en herramientas para el procesamiento y estudio de las Bases De Datos Espacio-Temporales, con la finalidad de permitir procesar los datos en relación al espacio-tiempo de los objetos o sucesos, detectando comportamientos reiterativos, aunque novedosos y útiles, los cuales son conocidos técnicamente bajo el término de “patrones”.

Este proyecto de título tiene como objetivo el análisis de algoritmos de Minería de Datos ampliamente utilizadas para la búsqueda de patrones: Apriori y Max-Subpattern, los cuales son un punto de partida para proponer un nuevo algoritmo llamado Minus-F_1, el cual reduce la cantidad de veces que debe ser analizado un conjunto de datos mediante la reducción progresiva del conjunto de subpatrones de la forma F_1 para determinar la existencia de patrones periódicos.

Los algoritmos antes señalados son implementados, testeados y puestos a prueba bajo las mismas condiciones, mismos datos a testear para los mismos períodos y mismo soporte, para determinar empíricamente cual es el que otorga los resultados más acertados y/o mejores tiempos de ejecución.

El presente trabajo se estructura en cinco capítulos: El Capítulo 1 comprende la justificación y los objetivos del presente proyecto, el Capítulo 2 provee una descripción teórica de los tópicos claves de esta investigación, las cuales son Base de Datos Espacio-Temporales, Minería de Datos, patrones y algoritmos de búsqueda de patrones. Por otro lado, en el Capítulo 3 se presentan los algoritmos propuestos: Busca-Zonas (para el pre-procesamiento de los archivos de la Base de Datos GeoLife) y Minus-F_1 (minería de patrones periódicos). Adicionalmente se describe un ejemplo práctico, un caso de prueba el cual es resuelto utilizando los algoritmos de búsqueda de patrones que proponen los tres algoritmos previamente mencionados. En el Capítulo 4 se exponen los resultados de la implementación de

los algoritmos, utilizando bases de datos de pruebas para diversos casos, haciendo uso de conjuntos de datos sintéticos y conjuntos de datos reales[1] exponiendo el comportamiento de los algoritmos para dichos casos de prueba y sus resultados en tiempo de ejecución. Las conclusiones de este trabajo son presentadas en el Capítulo 5.

[1] Los datos reales utilizados en este trabajo fueron extraídos de un conjunto de datos de trayectoria GPS del proyecto Geolife de Microsoft [L5].

CAPÍTULO 1.
OBJETIVOS

1. Objetivos del Proyecto

1.1. Objetivo General

Diseñar un nuevo algoritmo de minería de datos para realizar búsquedas de patrones periódicos en función del tiempo sobre Bases de Datos Espacio-Temporales, el cual debe detectar patrones completos y/o parciales. Para esto se ha utilizado una Base de Datos que posee un conjunto de trayectorias de personas. Así, dado un conjunto de coordenadas para un individuo, detectar si este sigue rutas constantes o similar en distintos instantes de tiempo.

1.2. Objetivos Específicos

- Estudiar e implementar dos algoritmos conocidos en el contexto de minería de datos, Apriori y Max Subpattern, para la detección de patrones periódicos en secuencias de datos espacio-temporales.
- Analizar, diseñar e implementar un nuevo algoritmo de minería de datos, para la detección de patrones periódicos.
- Generar un conjunto de datos sintéticos con el objeto de generar un banco de pruebas que permita verificar la correctitud del algoritmo propuesto, así como simular diferentes escenarios experimentales.
- Buscar una Base de Datos espacio-temporal, en la cual puedan aplicar los algoritmos de búsqueda de patrones periódicos.
- Realizar un análisis empírico del comportamiento (eficiencia en tiempo del algoritmo y exactitud) de los algoritmos implementados, considerando diferentes escenarios experimentales, usando tanto una muestra de la Base de Datos real, como los datos sintéticos.
- Analizar el orden de los algoritmos implementados.

1.3. Aportes

Se destaca como principal aporte la construcción de Minus-F_1, un nuevo algoritmo de minería de datos para la búsqueda de patrones periódicos en Bases de Datos Espacio-Temporales, específicamente sobre un conjunto de coordenadas dispuestas como trayectorias, el cual utiliza como base la construcción del conjunto de patrones F_1 de Apriori, para luego evaluar la existencia de patrones periódicos completos y patrones periódicos parciales.

1.4. Alcances y Límites

Tanto los valores de latitud, longitud y tiempo son datos reales, extraídos desde la trayectoria GPS de un dataset[2] [L6]. Este dataset en particular es producto de un proyecto de Investigación en Asia de la compañía Microsoft conocido como "GeoLife" [L5], proyecto en el cual se recopiló la información de 182 usuarios en un período de alrededor de 5 años (desde abril del 2007 hasta el mes de agosto del 2012). Por otro lado, se evaluaron dos algoritmos existentes en la literatura, más uno propio, el cual fue propuesto para detectar patrones periódicos completos y parciales, además de perfectos e imperfectos. El uso de datos sintéticos para la realización de pruebas permitió analizar la correctitud de los algoritmos en casos controlados e idóneos.

En el presente proyecto, el límite radicó en el tipo de datos a utilizar y, por tanto, las técnicas de minería de datos para encontrar un conjunto determinado de patrones, lo que se redujo a dos tipos de patrones:

- Patrones periódicos parciales y completos.
- Patrones periódicos perfectos e imperfectos.

Ambos tipos de patrones fueron obtenidos tanto de la Base de Datos sintéticos, como de GeoLife.

[2] El *DataSet* es una representación de datos residente en memoria que proporciona una modelo de programación relacional coherente independientemente del origen de datos que contiene. El *DataSet* contiene en sí, un conjunto de datos que han sido volcados desde el proveedor de datos.
Un *DataSet* contiene colecciones de *DataTables* y *DataRelations*.

CAPÍTULO 2.
CONCEPTOS PREVIOS

1. Motivación

El uso de bases de Bases de Datos Espacio-Temporales (BDET) tiene diversas aplicaciones. Por ejemplo, los Sistemas de Información Geográfica (SIG), así como el uso masificado del GPS, han dado lugar a que muchas aplicaciones hagan uso de datos espaciales y temporales, por lo que no es difícil encontrarnos con su utilización en áreas como el transporte, medio ambiente, biología y penal.

Las BDET permiten representar la naturaleza dinámica de los objetos del mundo real. Los objetos espaciales con los cuales se componen estas Bases de Datos tienden a cambiar de posición y/o forma en el tiempo. El gran número de aplicaciones que existe en la actualidad de la mano de los SIG, y el rápido flujo de información en relación a la evolución tanto de los sistemas computacionales como de la telecomunicación, han ayudado a la masificación en el uso de las BDET.

2. Conceptos Preliminares

Dada la línea de estudio que plantea el presente proyecto, es necesario comprender los siguientes conceptos

- Base de Datos Espacio Temporales
  - Trayectorias
- Descubrimiento del Conocimiento en Base de Datos
  - Minería de Datos
- Patrones

2.1. Bases de Datos Espacio-Temporales

La funcionalidad y versatilidad que proporciona tanto BDs Espaciales como BDs Temporales han permitido establecer los cimientos para el desarrollo de las Bases De Datos Espacio-Temporales (BDET). El uso de BDET en Sistemas de Información Geográfica (SIG) permite no sólo tener la información espacial de diversos objetos de estudio (por ejemplo la geoposición de un vehículo, de una persona o de un conjunto de animales) en un momento determinado, sino que, agregando características temporales, tener esta información para diversos instantes de tiempo, ya sea para intervalos de tiempo definidos o para aplicaciones en tiempo real, lo cual permite observar la evolución del objeto de estudio, por ejemplo: dirección y distancia recorrida, variación de volumen, velocidad de movimiento, entre otros.

La dimensión temporal de los datos espaciales ha sido objeto de discusión en la literatura durante mucho tiempo. Si bien existen numerosas soluciones de Sistema de Gestión de Bases de Datos (SGBD) sólo para la dimensión espacial, no se observa la misma situación para los datos espacio-temporales [2], aunque en la actualidad existen variadas extensiones para que los SGBD tengan soporte para datos de tipo espacio-temporal, por ejemplo: OSTM para Oracle [3] o PostGIS-T [2] por mencionar algunos.

Los requerimientos para modelar una BDET son:

- Representar los objetos con su respectiva posición en el espacio y su existencia en el tiempo.
- Capturar los cambios de posición en el espacio en el transcurso del tiempo.
- Capturar tanto cambios continuos como discretos.
- Definir atributos espaciales y organizarlos en el espacio.

- Capturar los cambios en los atributos espaciales en el transcurso del tiempo.
- Conectar los atributos espaciales con los objetos.
- Representar las relaciones espaciales entre los objetos en el transcurso del tiempo.
- Representar las relaciones entre los atributos espaciales en el transcurso del tiempo.

En [4] Gutiérrez postula que las aplicaciones basadas en BDET pueden ser clasificadas en tres grupos:

1. Aplicaciones que tratan con cambios continuos de los objetos.
2. Aplicaciones que incluyen objetos ubicados en el espacio y que pueden cambiar su posición por medio de la modificación de su forma geométrica o atributos espaciales.
3. Aplicaciones que integran los dos comportamientos anteriores.

2.1.1. Trayectorias

Una trayectoria espacial es la rastro generado por un objeto en movimiento en espacios geográficos, usualmente representada como una serie de puntos ordenados cronológicamente, por ejemplo: $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, donde cada punto de la forma p_i consiste en un conjunto de coordenadas geoespaciales (x, y) y una marca de tiempo (t), es decir $p_i = (x_i, y_i, t_i)$ [5]. El orden cronológico observado en el conjunto de puntos coordenados que conforma una trayectoria permite tratar a estos como un conjunto de datos espacio-temporales.

2.2. Descubrimiento del Conocimiento en Base de Datos

La Extracción de conocimiento está principalmente relacionado con el proceso de descubrimiento conocido como Descubrimiento del Conocimiento en Base de Datos (KDD por sus siglas en ingles de Knowledge Discovery in Databases), se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información [6]. Este proceso no es automático, es un proceso iterativo que exhaustivamente explora volúmenes muy grandes de datos para determinar relaciones. Es un proceso que extrae información de calidad que puede usarse para obtener conclusiones basadas en relaciones o modelos dentro de los datos.

Una de las fases primordiales del proceso KDD es el de Minería de Dato y es a causa de esto que se suele utilizar este término para definir al proceso de KDD [7]. El objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender los datos o para expresar situaciones pasadas. Para ellos es necesario tomar una serie de decisiones antes de empezar el proceso:

- Determinar qué tipo de tarea de minería es el más apropiado.
- Elegir el tipo de modelo.
- Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando.

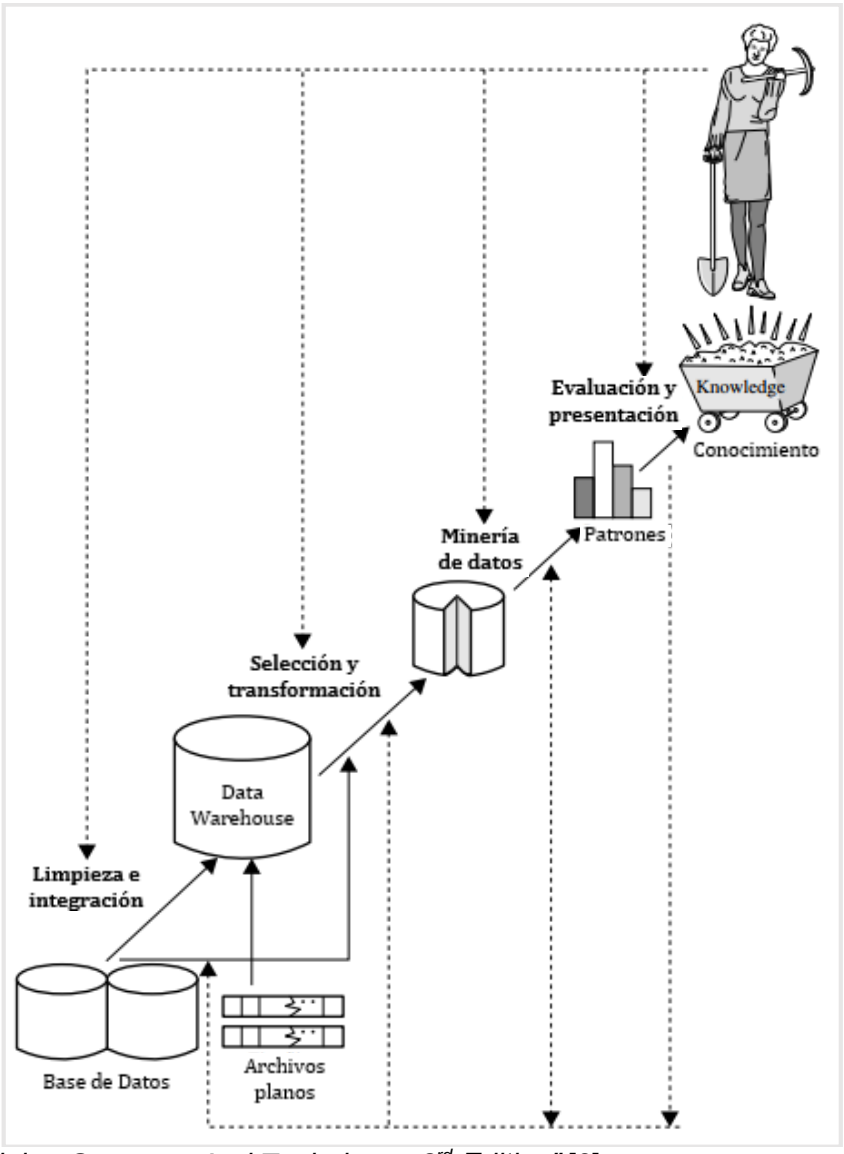
Uno de los actores más importantes dentro de KDD es el usuario, ya que es él quien determina el dominio de la aplicación, o sea, decide cómo y qué datos se utilizarán en el proceso. Por lo tanto, los pasos en el proceso global del KDD no están claramente diferenciados por ser un proceso iterativo e interactivo con el usuario experto. Las interacciones entre las decisiones tomadas en diferentes pasos, así como los parámetros de los métodos utilizados y la forma de representar el problema suelen ser extremadamente complejos [L3].

Una serie de etapas, que pueden ser vistas en Figura 1, muestran cómo se desarrolla el proceso KDD para la obtención del conocimiento [6].

1. **Limpieza de datos**, para eliminar el ruido y datos inconsistentes.
2. **Integración de datos**, donde múltiples secuencias de datos pueden combinarse[3]
3. **Selección de datos**, cuando sea relevante para la tarea de análisis se recupera de la Base de Datos.
4. **Transformación de datos**, donde los datos son transformados y consolidados en formas apropiadas para la minería realizando operaciones de resumen o de agregación.
5. **Minería de datos**, proceso esencial en el que se aplican métodos inteligentes para extraer patrones de datos.

[3] Una tendencia popular en la industria de la información es realizar la limpieza de datos y la integración de datos como un paso de preprocesamiento, donde los datos resultantes se almacenan en el datawarehouse.

Figura 1: Proceso de Descubrimiento de Conocimiento en Base de Datos (KDD)



Fuente: Data Mining Concepts And Techniques 3rd Edition" [6]

6. **Evaluación de patrones**, para identificar los patrones verdaderamente interesantes que representan el conocimiento basado en medidas de interés.
7. **Presentación del conocimiento**, donde la visualización y las técnicas de representación del conocimiento se utilizan para presentar el conocimiento minado a los usuarios.[4]

[4] A veces la transformación y consolidación de datos se realizan antes del proceso de selección de datos, particularmente en el caso del almacenamiento de datos. La reducción de datos también se puede realizar para obtener una representación más pequeña de los datos originales sin sacrificar su integridad.

2.2.1. Minería de Datos

La Minería de Datos (MD) es definida como un conjunto de técnicas orientadas al descubrimiento de información novedosa y potencialmente útil contenida en grandes conjuntos de datos, por ejemplo, conjuntos de datos obtenidos de BDET [8]. Se trata de analizar comportamientos, patrones, tendencias, asociaciones y otras características del conocimiento inmerso en los datos [9]. En [10] se describe como como un proceso no trivial de extracción de información implícita, previamente desconocida y potencialmente útil a partir de los datos.

La MD es utilizada, por ejemplo, en aplicaciones de control de procesos productivos, como herramienta de ayuda en la planificación y la decisión en marketing y finanzas entre otros. Así mismo, la minería de datos es fundamental en la investigación científica y técnica, como herramienta de análisis y descubrimiento de conocimiento a partir de observación de datos o resultados de experimentos [11]. Dada estas razones, la tecnología asociada a la Minería de Datos es considerada relevante en procesos de toma de decisiones, esto aboga a que provee valores implícitos importantes o por lo menos interesantes de obtener, desde extensos centros de recopilación de datos históricos. Estos datos pueden considerarse críticos, ya que pueden generar nueva información a partir de repositorios que han estado guardados hace mucho tiempo y que a la larga puede significar el éxito o fracaso de alguna empresa en un rubro determinado.

2.2.1.1. Tareas de la Minería de Datos

Dado el tipo de datos a procesar y la información que se desea obtener, existe varios métodos y algoritmos aplicables a la Minería de Datos, las tareas de Minería de Datos están agrupados en dos modelos [6]:

- **Modelos Descriptivos (aprendizaje supervisado)**: Las tareas descriptivas de minería caracterizan las propiedades de los datos en un conjunto de datos de destino.
  Ejemplo:
  Clasificación:
  - Perfil de un cliente de alto riesgo para préstamos bancarios. Como se observa en Figura 2, con los datos obtenidos de “Aprendizaje” se genera un modelo que luego es aplicado sobre “Testing”, donde “Fraude” es la variable a predecir. [L1]

- **Modelos Predictivos (aprendizaje no supervisado)**: Las tareas de minería predictiva realizan la inducción en los datos actuales con el fin de hacer predicciones.

Ejemplo:

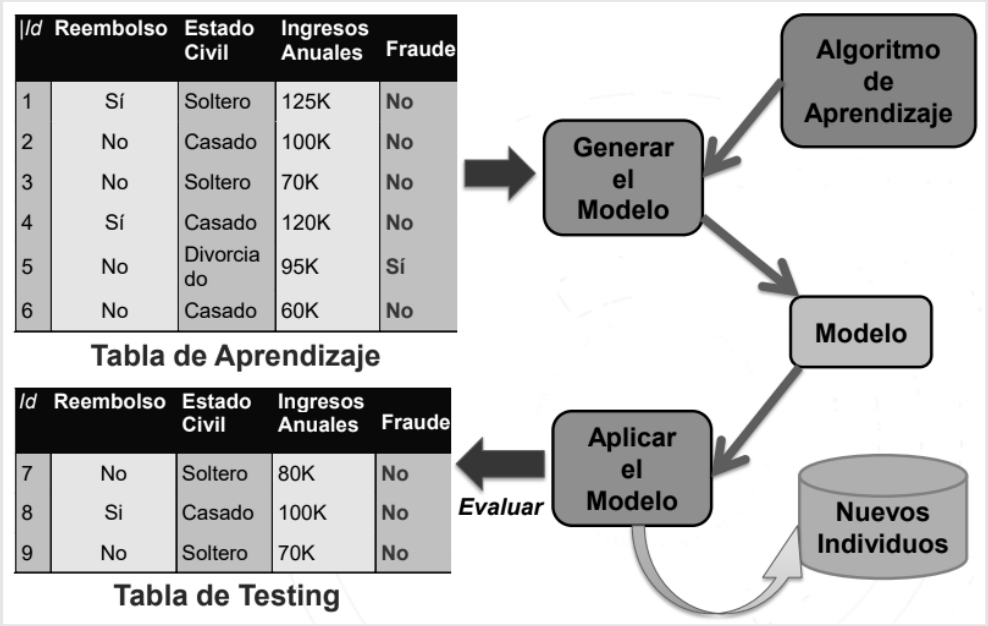
Reglas de Asociación:

- Los compradores de pañales también suelen comprar leche.

Clustering:

- Segmentación de clientes de un supermercado:
  - Clientes ocasionales que gastan mucho.
  - Clientes habituales con presupuesto limitado.
  - Clientes ocasionales con presupuesto limitado.

Figura 2: Modelo descriptivo basado en clasificación, para la predicción de fraudes



Fuente: "Modelos predictivos y descriptivos en minería de datos" [L1]

2.2.1.2. Minería de Datos en Bases de Datos Espacio-Temporales

En [12], Xiaobai Yao afirma que las técnicas de Minería de Datos pueden ser "especializadas" y "temporalizadas" y en [13] se describen cinco tipos principales de reglas espacio-temporales, que evolucionaron de sus contrapartes estáticas, estas reglas son:

- **Asociaciones Espacio-Temporales:** Al igual que la descripción de Agrawal [14] Las reglas de asociación son de la forma $X \rightarrow Y$, donde la ocurrencia de X es acompañada de la ocurrencia de Y en un $c\%$ de los casos, mientras que X e Y ocurren juntas en una transacción en un $s\%$ de los casos $(c\%, s\%)$. Se extienden estas reglas usando predicados espaciales y temporales.

- **Generalización Espacio-Temporal:** Este proceso tiene implícito el concepto de jerarquía, el cual se usa para agregar datos, de forma que se creen reglas más robustas, a expensas de la especificidad. Existen dos tipos de generalización: dominada por datos espaciales, en los cuales estos poseen mayor jerarquía, generalizando los datos de atributos por región; y la generalización dominada por datos no espaciales, en la cual los datos no espaciales ocupan una mayor jerarquía.
- **Clustering Espacio-Temporal:** Es similar al clustering de datos estáticos, pero en este caso lo que se considera para formar los clúster, son sus características espacio-temporales o sus cualidades en una región espacio-temporal.
- **Reglas evolutivas:** Estas reglas tienen un contexto espacio-temporal explícito, y describen la forma en que los objetos espaciales cambian a través del tiempo. Esto implica la adopción de predicados que sean “usables” y” entendibles” como, por ejemplo, sigue, coincide o muta.
- **Meta Reglas**: Estas reglas se originan cuando, en vez de conjuntos de datos, se analizan conjuntos de reglas en busca de tendencias o comportamientos similares.

2.3. Patrones

Como mencionamos en el Capítulo 1, uno de los objetivos de esta tesis es evaluar el desempeño de algoritmos de Minería de Datos para la búsqueda de patrones en BDET. Así, un patrón es una característica o rasgo consistente y recurrente que ayuda en la identificación de un fenómeno o problema y sirve como un indicador o modelo para la predicción de futuros comportamientos [L2]. Estos patrones son producto del proceso no trivial de extracción de información implícita, previamente desconocida y potencialmente útil a partir de datos que pueden resultar ambiguos.

Deben considerarse las siguientes medidas de interés en la obtención de patrones [6]:

- Sencillos (Útiles, interpretables y comprensibles).
- Certeros (Que sean verdaderos en general).
- Útiles (Pueden ser utilizados en decisión).
- Novedosos (Desconocidos hasta el descubrimiento).

2.3.1. Patrones Periódicos

Un patrón periódico es aquel patrón que se repite de forma regular dentro de una secuencia de datos, dicha secuencia puede ser definida como una lista ordenada de elementos de cualquier dominio, por ejemplo, una secuencia es la obtenida por el conjunto de coordenadas que define una trayectoria, ya sea de la migración de animales, o el movimiento en la ciudad de una persona. El análisis de una secuencia de datos puede llevar a encontrar conjuntos de datos contenidos dentro de esta secuencia, es decir, subsecuencias. Los patrones cuyas apariciones dentro de la secuencia de datos supera las expectativas mínimas de apariciones, es decir, tienen un gran soporte, se llaman Patrones Secuenciales Frecuentes [15]. Por otro lado, el análisis de períodos es comúnmente realizado a través de datos de series temporales, estas consisten en secuencias de valores o eventos típicamente medidos en intervalos de tiempo iguales. De manera general existen dos tipos de periodicidad:

- **Periodicidad Fija (o Conocida).**
- **Periodicidad impredecible.**

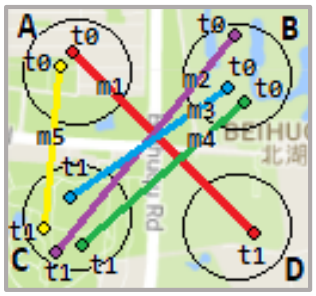
En [15] se exponen diferentes criterios para clasificar patrones periódicos, así como algoritmos capaces de identificarlos según cada criterio. Los ejemplos de las descripciones mostrados a continuación contienen secuencias de datos mostrada en forma de cadena de texto[5], cada una de estas cadenas representa un conjunto de eventos o secuencia de datos, donde cada letra es un evento y estos están organizados en forma secuencial en el tiempo.

2.3.1.1. Patrones Periódicos Completos

Un patrón periódico es considerado completo (o patrón de período completo) cuando todos los elementos del patrón presentan un comportamiento periódico. Por ejemplo, consideremos la secuencia $\{A\}\{D\}\{B\}\{C\}\{B\}\{C\}\{B\}\{C\}\{A\}\{C\}$, en esta secuencia de datos el conjunto $\{B\}\{C\}$ es un patrón periódico completo de período 2. La secuencia antes presentada puede ser una representación del siguiente ejemplo: El mapa segmentado en zonas $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$, en Figura 3, muestra la posición y trayectoria de un vehículo para dos instantes de tiempo definidos T_0 y T_1 en diferentes días. En Tabla 1 se muestra de forma organizada estos mismos datos.

[5] El uso de cadena de texto para ejemplificar secuencias de datos espaciales o espacio-temporales es tomado directamente de los ejemplos observados en [15] y [17] entre otros, las Figuras incluidas en los ejemplos pretenden facilitar interpretación de éstos datos.

Figura 3: Mapa de trayectoria de un móvil para dos instantes de tiempo diarios en cinco días



Fuente: Elaboración propia, basado en ejemplos de [15]

Tabla 1: Datos de trayectoria de un móvil para dos instantes de tiempo diarios en cinco días

Vehículo/Días		$Tiempo$	
		t_0	t_1
	Día 1	{A}	{D}
	Día 2	{B}	{C}
	Día 3	{B}	{C}
	Día 4	{B}	{C}
	Día 5	{A}	{C}

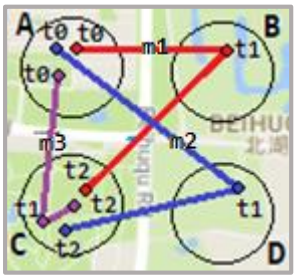
Fuente: Elaboración propia, basado en ejemplos de [15]

La periodicidad de los datos está dada por los instantes de tiempo en que se recogen los datos, para este caso el período es 2 (dados por los instantes T_0 y T_1). Los eventos ocurridos en los días 2, 3 y 4 presentan un comportamiento similar, moviéndose desde la Zona B a la Zona C, estos movimientos pueden ser considerados como patrones periódicos completos.

2.3.1.2. Patrones Periódicos Parciales:

Estos patrones se caracterizan porque, a diferencia de los patrones periódicos completos, alguno de sus elementos no presenta un comportamiento periódico. La Figura 4 sirve como ejemplo representativo de este tipo de patrones, así como los datos organizados presentes en Tabla 2.

Figura 4: Mapa de trayectoria de un móvil para tres instantes de tiempo diarios en tres días



Fuente: Elaboración propia, basado en ejemplos de [15]

Tabla 2: Datos de trayectoria de un móvil para tres instantes de tiempo diarios en tres días

Vehículo/Días		$Tiempo$		
		t_0	t_1	t_2
	Día 1	{A}	{B}	{C}
	Día 2	{A}	{D}	{C}
	Día 3	{A}	{C}	{C}

Fuente: Elaboración Propia, basado en ejemplos de [15]

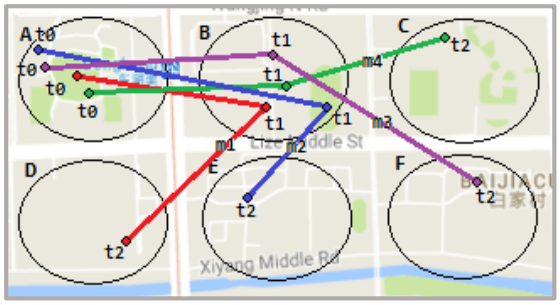
Por ejemplo, la secuencia $\{A\}\{B\}\{C\}\{A\}\{D\}\{C\}\{A\}\{C\}\{C\}$ tiene como patrón periódico el conjunto $\{A\}\{*\}\{C\}$ de período 3, para el cual el segundo elemento de este patrón no presenta un comportamiento periódico[6].

Observe que las trayectorias encontradas inician (T_0) y terminan (T_2) en una misma zona (A y C respectivamente), lo que describe un comportamiento periódico en esos instantes, mientras que su posición intermedia (T_1) no presenta este periodicidad, lo que rompe la existencia de un patrón periódico completo pero presenta la existencia de un patrón periódico parcial.

2.3.1.3. Patrones Periódicos Perfectos

Un patrón periódico perfecto es aquel que es encontrado en cada segmento periódico de una secuencia de datos, desde el primer segmento hasta finalizada la secuencia. Estos patrones consideran un parámetro denominado soporte[7], el cual se ve afectado según las apariciones del patrón en la secuencia. Si un patrón no es encontrado en todos los segmentos de la muestra, este no es considerado perfecto. Por ejemplo, la secuencia de datos: $\{A\}\{B\}\{D\}\{A\}\{B\}\{E\}\{A\}\{B\}\{F\}\{A\}\{B\}\{C\}$, presenta el segmento $\{A\}\{B\}\{*\}$ como un patrón periódico perfecto con período 3, puesto que la secuencia cuenta con 4 segmentos de período 3 y el patrón es encontrado en cada segmento, cumpliendo con un nivel de soporte de 4 sobre 4, es decir, aparece un 100% de las veces.

Figura 5: Mapa de trayectoria de un móvil para tres instantes de tiempo diarios en cuatro días



Fuente: Elaboración propia, basado en ejemplos de [15]

Tabla 3: Datos de trayectoria de un móvil para tres instantes de tiempo diarios en cuatro días

Vehículo/Días		Tiempo		
		t_0	t_1	t
	Día 1	{A}	{B}	{D}
	Día 2	{A}	{B}	{E}
	Día 3	{A}	{B}	{F}
	Día 4	{A}	{B}	{C}

Fuente: Elaboración propia, basado en ejemplos de [15]

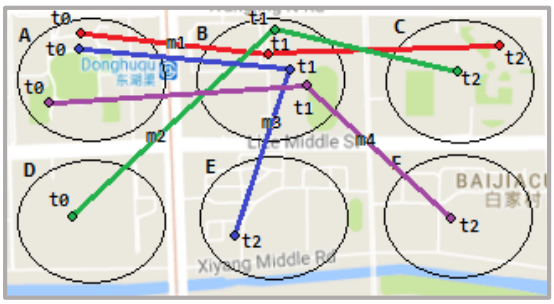
[6] Un $\{*\}$ dentro del conjunto que compone un patrón representa un elemento aleatorio el cual puede ser diferente para cada segmento en la secuencia de datos.

[7] El parámetro "soporte" determina la importancia de un patrón dentro de una secuencia de datos. Valores despreciables de soporte para un patrón puede llevar a que esté carezca de importancia para una secuencia de datos.

2.3.1.4. Patrones Periódicos Imperfectos

Si un patrón periódico no es encontrado en cada segmento periódico de una secuencia de datos, estamos frente a un patrón periódico imperfecto. Si bien el soporte de estos patrones es inferior al 100%, esta debe ser suficientemente representativa como para que el patrón sea considerado importante.

Figura 6: Mapa de trayectoria de un móvil para tres instantes de tiempo diarios en cuatro días



Fuente: Elaboración propia, basado en ejemplos de [15]

Tabla 4: Datos de trayectoria de un móvil para tres instantes de tiempo diarios en cuatro días

Vehículo/Días	Tiempo		
	t_0	t_1	t_2
Día 1	{A}	{B}	{C}
Día 2	{D}	{B}	{E}
Día 3	{A}	{B}	{E}
Día 4	{A}	{B}	{F}

Fuente: Elaboración propia, basado en ejemplos de [15]

Por ejemplo, la secuencia de datos $\{A\}\{B\}\{C\}\{D\}\{B\}\{C\}\{A\}\{B\}\{E\}\{A\}\{B\}\{F\}$, la cual está representada en la Figura 6 y Tabla 4, es posible identificar variados patrones periódicos con período 3, por ejemplo: $\{A\}\{B\}\{*\}$ y $\{*\}\{B\}\{C\}$, por mencionar algunos, los cuales tienen valores de soporte: 3/4 (75%), 2/4 (50%) respectivamente.

2.3.1.5. Patrones Periódicos Síncronos

Cuando un patrón tiene apariciones de forma periódica dentro de una secuencia de datos y este no se ve afectado por desalineaciones o distorsión estamos frente a un patrón periódico síncrono. Por ejemplo, la siguiente secuencia de datos: $\{A\}\{B\}\{C\}\{A\}\{D\}\{C\}\{A\}\{C\}\{C\}\{D\}\{E\}\{F\}$, contiene el patrón $\{A\}\{*\}\{C\}$, el cual es considerado síncrono pues se ha repetido de forma consecutiva durante tres períodos, sin encontrar segmentos de períodos diferentes entre cada segmento periódico de la secuencia.

Los ejemplos presentados en los casos anteriores (Figura 3, Figura 4, Figura 5, Figura 6) también puedes ser descritos como ejemplos de patrones síncronos, pues su aparición la secuencia de datos está descrita en períodos fijos.

2.3.1.6. Patrones Periódicos Asíncronos

Tabla 5: Datos de trayectoria de un móvil en cinco días

Días	Tiempo		
	t_0	t_1	t_2
Día 1	{A}	{B}	{C}
Día 2	{A}	{C}	{C}
Día 3	{C}	{B}	-
Día 4	{A}	{B}	{C}
Día 5	{A}	{D}	{C}

Fuente: Elaboración propia, basado en ejemplos de [15]

Es posible encontrar perturbaciones entre las repeticiones de los patrones en una secuencia de datos. Las perturbaciones pueden ocurrir a causa de inserción eventos[8] de ruido aleatorio, o por la pérdida de la ocurrencia de algún evento, rompiendo la sincronía de los períodos. Bajo estas situaciones estamos frente a patrones periódicos asíncronos. Por ejemplo, el patrón: $\{A\}\{*\}\{C\}$ encontrado en la secuencia $\{A\}\{B\}\{C\}\{A\}\{C\}\{C\}\{C\}\{B\}\{A\}\{B\}\{C\}\{A\}\{D\}\{C\}$, representada en Tabla 5, es considerado periódico, aunque asíncrono para la secuencia. Este patrón tiene cuatro apariciones sobre la secuencia, pero existe perturbación de la secuencia entre la segunda y tercera ocurrencia del patrón.

2.3.2. Patrones con Periodicidad de Símbolos, Secuencias, Segmentos y Densos:

Los patrones periódicos pueden ser clasificados de acuerdo con periodicidad de símbolos, secuencias y segmentos, así como por su densidad.

2.3.2.1. Patrones con Periodicidad de Símbolo

En una secuencia se dice que estamos frente a una periodicidad de símbolo (o patrón singular), cuando al menos un símbolo se repite periódicamente. Por ejemplo, considerando la secuencia: {x}{y}{y}{x}{u}{i}{x}{t}{r}{x}{k}{l}, si se observa, es posible determinar que {x} tiene periodicidad tres, a su vez que es un patrón periódico perfecto de la forma {x}{*}{*} para este ejemplo. Esta descripción hace referencia a los patrones de la forma

[8] Se considera un evento a cada elemento existente en una secuencia de datos.

2.3.2.2. Patrones con Periodicidad de Secuencia

(patrones parciales) Si más de un símbolo se repite en una secuencia con el mismo comportamiento periódico, estamos frente a periodicidad de secuencia. **Ejemplo:** El siguiente patrón: {x}{y}{*} es un patrón periódico parcial en la secuencia de datos {x}{y}{z}{y}{z}{z}{x}{y}{y}{x}{y}{x}.

2.3.2.3. Patrones con Periodicidad de Segmento

(patrones de completo) Si una secuencia puede representarse en gran parte como una repetición de un patrón o segmento, entonces ese tipo de periodicidad se denomina periodicidad de segmento o ciclo completo.
Ejemplo: el segmento {x}{y}{z} es un patrón con periodicidad de ciclo completo en la secuencia {x}{y}{z}{x}{y}{z}{x}{x}{x}{x}{y}{z}, donde {x}{y}{z} tiene periodicidad 3 y un valor de soporte de 3/4 (o 75%).

2.3.2.4. Patrones Periódicos Densos:

En el recorrido de una secuencia de datos es posible encontrar segmentos interesantes, aunque estos están disponibles solo en una pequeña porción de la secuencia. Ante estas circunstancias estamos frente a segmentos densos. Considere el siguiente ejemplo: El ciudadano Kane transita a diario desde su casa hasta la empresa donde él trabaja. Su recorrido durante los últimos tres meses es el siguiente: (sea este recorrido la secuencia de datos para este ejemplo)

- Mes 1: Casa → Sector A → Sector D → Empresa.
- Mes 2: Casa → Sector B → Sector K → Empresa.
- Mes 3: Casa → Sector C → Sector F → Empresa.

La secuencia no presenta comportamiento periódico en su totalidad, pero si en determinados rangos de esta, donde pueden encontrarse patrones periódicos que están condensados.

2.4. Búsqueda de Patrones: Técnicas de Minería de Datos y Algoritmos

De manera general podemos decir que existen dos formas de identificar patrones, la primera es a través de algoritmos y las segunda están basadas en métodos matemáticos como las transformadas de Fourier. Sin embargo, las

técnicas basadas en transformadas de Fourier suelen ser costosas en tiempo de ejecución y en algunos casos no proveen soluciones correctas, ya que resulta difuso determinar el inicio y el término de un período.

Así en este proyecto de título, nosotros estamos interesados en la eficiencia de procesamiento de los algoritmos. Por consiguiente, analizaremos dos algoritmos ampliamente conocidos y utilizados en el ámbito de la Minería de Datos, estos son Apriori y Max-Subpattern. Ambos algoritmos fueron creados para evaluar reglas de asociación dadas dos métricas, soporte y confianza. Estas métricas son proveídas por los analistas, y las reglas de asociación proveen un porcentaje de aparición de una regla B, dada una regla A, ya que encontrar una correlación matemática resulta compleja en tiempo y costo en términos de computo.

2.4.1. Reglas de Asociación

La tarea de reglas de asociación, propuestas por Agrawal [14], permite obtener patrones que tienen la representación de reglas y muestran conjuntos de elementos que co-ocurren de manera frecuente en un conjunto de transacciones. Basados en la implicancia que pueden ocurrir entre conjuntos de eventos se utilizan umbrales de soporte y confianza para establecer identificar patrones significativos.

En [14] se define un conjunto de ítems $I = \{i_1, i_2, i_3, \dots, i_n\}$ conformado por atributos binarios y $D = \{t_1, t_2, t_3, \dots, t_m\}$, conjunto de transacciones almacenadas en una Base de Datos sobre la cual se descubrirán las reglas. Cada transacción (Tr) en D es un subconjunto de I. Se define una regla como una implicación de la forma $X \Rightarrow Y$, donde X (antecedente) e Y (consecuente) son subconjuntos de I y la intersección de X con Y es vacía. Si X es un conjunto de ítems (itemset) con k elementos, se le denomina k-itemset. El soporte de un k-itemset X se define como la cantidad de transacciones en D que verifican a X respecto al total de transacciones en D:

$$Sup_D(X) = \frac{|V_D(X)|}{|D|}, \qquad V_D(X) = \{Tr = \{i_{Tr_1}, i_{Tr_3}, \dots, i_{Tr_k}\} \in D \;/\; X \subseteq Tr\}$$

Si el soporte de un k-itemset supera el umbral del soporte mínimo, este es denominado frecuente. A su vez se define confianza de la regla como la proporción de transacciones en D que verifican X y también verifican Y. La confianza de la regla se encuentra dado por el soporte de la unión de los conjuntos de datos X e Y de la misma:

$$Conf\ (X \rightarrow Y) = \frac{Sup\ (X \cup Y)}{Sup\ (X)}, \qquad Sup_D(X \rightarrow Y) = Sup_D(X \cup Y)$$

Para resolver el descubrimiento de reglas de asociación la tarea divide el problema que ha de resolver en dos:

- Descubrimiento de todos los conjunto de ítems frecuentes en D, es decir aquellos que superen el umbral de soporte mínimo.
- En base a los conjuntos de datos frecuentes descubiertos en la etapa previa, armar las reglas de asociación y presentar al usuario solamente aquellas que superen el umbral mínimo de confianza.

Para ejemplificar lo antes descrito considere la información de Tabla 6, donde ID representa el número de transacción, y en los registros disponibles para las otras columnas, 1 representa que el ítem está disponible en la transacción, mientras que 0 indica la ausencia de este. Los ítems: $\{Leche, Pan, Margarina, Gaseosa\}$ componen el conjunto I.

Un ejemplo de regla de asociación puede ser: $\{Leche, Pan\} \Rightarrow \{Margarina\}$, interpretado como: si el cliente compró leche y pan, también compró mantequilla, donde $X = \{Leche, Pan\}$ e $Y = \{Margarina\}$

Tabla 6: BD con 4 ítems y 5 transacciones

ID	Leche	Pan	Margarina	Gaseosa
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Fuente: Ejemplo de la canasta de compra en "Reglas de Asociación" [L7]

Para el caso del ejemplo, el soporte de X debe considerar la cantidad de transacciones, D, en relación a las veces que ocurre X en D.

$$Sup(X) = Sup(Leche, Pan) = \frac{2}{5} = 0.4$$

La confianza de la regla es calculada como sigue:

$$Conf(X \rightarrow Y) = Conf(\{leche, pan\} \Rightarrow \{margarina\}) =$$

$$\frac{Sup(\{Leche, Pan\} \cup \{Margarina\})}{Sup(\{Leche, Pan\})} = \frac{0.2}{0.4} = 0.5$$

Lo que indica que, de las reglas que contienen como antecedente $\{Leche, Pan\}$ el 50% también contiene $\{Margarina\}$, lo que hace que la regla $\{Leche, Pan\} \Rightarrow \{Margarina\}$ sea cierta en el 50% de los casos. Este ejemplo muestra a gran escala la forma en que operan las reglas de asociación.

Las bases de datos reales cuentan con miles o millones de registros que requieren ser evaluados, es por eso que las reglas de asociación requieren de la ayuda de tecnologías que permitan disminuir el costo de tiempo de estos procesos. Uno de los Algoritmos más utilizados en minería de datos para la búsqueda de conjuntos de datos frecuentes, que logra resolver el problema antes planteado, es Apriori.

2.4.2. Apriori

Propuestos por Agrawal y Srikant [16] en 1994, Apriori es un algoritmo utilizado en minería de datos, inicialmente desarrollado para ofrecer soluciones sobre bases de datos transaccionales, el cual tiene como objetivo generar conjuntos de ítems frecuentes. Dichos conjuntos son generados a partir de sub-conjuntos encontrados previamente en la Base de Datos con una tasa de aparición suficiente.

Figura 7: Algoritmo Apriori

Apriori (definición de Agrawal, R. Stikant, R)	
Entrada:	Conjunto de transacciones D, soporte mínimo
Salida:	Conjunto de transacciones frecuentes L_1 y L_k , desde k =2 hasta que L_{k-1} esté vacío.
1)	$L_1 = \{conjunto\ de\ items\ de\ largo\ 1\}$:
2)	$\boldsymbol{para}\ (k = 2;\ L_{k-1} \neq \emptyset; k++)$
3)	$C_k = apriori - gen(L_{k-1}); //nuevos\ candidatos$
4)	$\forall\ transactions\ t \in D$
5)	$\boldsymbol{C_t} = subset(C_k, t); //candidatos\ contenidos\ en\ t$
6)	$\forall\ candidatos\ c \in C_t\ \boldsymbol{do}$
7)	$c.count++;$
8)	$\boldsymbol{fin}$
9)	$L_k = \{c \in C_k \mid c.count \geq minsup\}$
10)	$\boldsymbol{fin}$
11)	$Respuesta = \bigcup_k L_k;$

Fuente: "Fast Algorithms For Mining Association Rules" [16]

Apriori responde al siguiente principio: “Si un conjunto de ítems es frecuente, también los son sus subconjuntos”, el cual está sostenido por la propiedad de anti-monotonía: Sean X e Y conjuntos de datos, $\forall X, Y : (X \subseteq Y) \Rightarrow Sup(X) \geq Sup(Y)$ [L4].

Este algoritmo realiza el descubrimiento de conjuntos de datos en D realizando k_{max} pasadas sobre la Base de Datos, donde k_{max} es el tamaño máximo de k-itemset descubierto por el algoritmo. En Figura 7 se encuentra el algoritmo Apriori definido en [16].

Ejemplo: Haciendo uso del ejemplo de la canasta de compras se explica el funcionamiento del algoritmo. En esta ocasión los ítems estarán representados con letras y agrupados en conjuntos, donde cada conjunto es una transacción en la Base de Datos. Este ejemplo sólo considera la existencia de los ítems en cada transacción, sin considerar la cantidad de un ítem sobre la misma, tampoco considera el orden de los ítems en la transacción o el orden de las transacciones.

Sea el conjunto de transacciones D, compuesto de nueve transacciones:

$$D = \{\{a, b, c, d\}, \{a, b, d\}, \{a, e\}, \{a, b, c\}, \{b, c, d\}, \{b, c\}, \{c, d\}, \{b, d\}\{a, e\}\}$$

Basados en el pseudocódigo de Apriori, en primera instancia (Figura 7, Línea 1) se separan los elementos individuales de la Base de Datos, generando sub-conjuntos de tamaño uno (1-itemset) luego cada 1-itemset es revisado en las diferentes transacciones, aumentando su soporte en 1 por cada transacción en el que esté presente, con un máximo de 9 apariciones para este ejemplo.

$$Candidatos\ L_1: \quad \{\{a: 5/9\}, \{b: 6/9\}, \{c: 5/9\}, \{d: 4/9\}, \{e: 2/9\}\}$$

El conjunto L_1 estará compuesto por los candidatos L_1 frecuentes, es decir, que cumplan con el soporte mínimo (soporte de 3 sobre 9 para en este ejemplo) y por tanto, bajo este soporte mínimo tenemos que ‘e’ no cumple el requisito y es descartado, lo que deja el conjunto frecuente L_1 como sigue:

$$L_1: \quad \{\{a: 5/9\}, \{b: 6/9\}, \{c: 5/9\}, \{d: 4/9\}\}$$

Iniciada la iteración (Figura 7, línea 2 a 10) comienza la búsqueda de candidatos L_2. Ahora, agrupando en pares los ítems frecuentes L_1 se obtienen el conjunto de ítems candidatos de tamaño 2 (Figura 7, línea 3), nuevamente es necesario obtener el soporte de estos (conteo de candidatos: Figura 7, línea 4 a 8) y luego sesgar en base al soporte mínimo los pares frecuentes (línea 9):

$$Candidatos\ L_2: \quad \{\{a,b:3\},\{a,c:2\},\{a.d:2\},\{b,c:4\},\{b,d:4\},\{c,d:3\}\}$$
$$L_2: \quad \{\{a,b:3\},\{b,c:4\},\{b,d:4\},\{c,d:3\}\}$$

Luego de identificar los itemset L_2 frecuentes e infrecuentes, se genera el conjunto de candidatos L_3, lo que significa una nueva iteración (Figura 7, línea 2 a 10), descartando cualquier itemset L_2 que contenga algún candidato L_2 infrecuente de la fase anterior.

$$Candidatos\ L_3: \quad \{\{a.b.c:No\},\{a.b.d:No\},\{a,c,d:No\},\{b,c,d:2\}\}$$
$$L_3: \quad \emptyset$$

Al no existir conjuntos L_3 frecuentes, ya que ningún itemset L_3 candidato cumple con el soporte mínimo; el algoritmo concluye habiendo determinado los conjuntos frecuentes de esta Base de Datos (Figura 7, Línea 11). El ejemplo en esta instancia es finalizado.

En el Capítulo 3, sección 2.1, se describe una versión del algoritmo Apriori orientada a la búsqueda de patrones en BDET.

2.4.3. Max-Subpattern

El algoritmo Max-Subpattern (M-SP) es descrito por primera vez en [17] y tiene por finalidad disminuir el número de conjuntos candidatos para determinar patrones periódicos en Bases de Datos en comparación al algoritmo Apriori.

M-SP utiliza la búsqueda de conjuntos F_1 de 1-patrones f_1 frecuentes (similar a la búsqueda del conjunto L_1 de Apriori, pero agregando la dimensión tiempo) pues considera una forma eficaz de reducción de conjuntos candidatos en primera instancia, por tanto, M-SP se centra en reducir los conjuntos de candidatos luego de encontrado F_1.

Este algoritmo considera las nociones de max-pattern (patrón máximo) y hit-pattern (patrón acierto), definidas en [17]. Un patrón máximo candidato frecuente**,** $Cmax$, es el patrón máximo que puede ser generado de la unión de todos los segmentos en conjunto F_1. Por ejemplo: el siguiente conjunto F_1 genera el patrón candidato $Cmax$: $abcd*$.

$$F_1: \quad \{a\ ****,*\ b\ ***,**\ c\ **,***\ d\ *\}$$
$$Cmax: \quad abcd*$$

El siguiente ejemplo permite observar que $Cmax$ permite la disyunción de más de un elemento no-$*$. Por ejemplo:

$$F_1: \{a ****,* b_1 ***,* b_2 ***,** c **,*** d *\}$$
$$Cmax: a\{b_1, b_2\}cd *.$$

El largo de un patrón es definido como L-largo, este largo está dado por la cantidad de eventos no-* existentes en un segmento o patrón, por ejemplo, el L-largo de un segmento existente en F_1 es 1, mientras que el L-largo de $Cmax$ es $|Cmax|$. Un Hit (acierto) es un subpatrón de $Cmax$, obtenido de la intersección de $Cmax$ con un segmento periódico S_i de una secuencia de datos S. Por ejemplo:

$$Cmax: a\{b_1, b_2\}cd *.$$
$$S_i: a\{b_1, b_2\}c_1\{d_1, d_2\}e$$
$$Hit: a\{b_1, b_2\} ***$$

El Hit-Set H (conjunto de aciertos), de S es el conjunto de todos los subpatrones aciertos de $Cmax$ en S. El algoritmo M-SP Hit-Set presentado en Figura 8 busca todos los patrones periódicos parciales para un período dado p en una secuencia de datos S, considerando $Cmax$ y patrón-hit para un umbral de soporte determinado.

Figura 8: Algoritmo Max-Subpattern

Max-Subpattern Hit Set	
Entrada:	Conjunto de segmentos periódicos S, soporte mínimo
Salida:	Conjunto de patrones periódicos
1)	Escanear una serie S para encontrar F_1.
2)	Formar $Cmax$ desde F_1.
3)	$\forall S_i \in S$ y $Cmax \rightarrow S_i \cap Cmax =$ hit-subpatrón $h\text{-}sp$.
4)	Si L-largo de $h\text{-}sp \geq 2$
5)	Si $h\text{-}sp$ existe en conjunto de aciertos H, h.contador+1.
6)	Si $h\text{-}sp$ no existe en H, insertar $h\text{-}sp$ en H, hit.contador=1.
7)	Con H, Generar Árbol Max-Subpattern (Figura 9)
8)	Después del análisis, obtener los patrones frecuentes del conjunto de aciertos.

Fuente: "Efficient Mining of Partial Periodic Patterns in Time Series Database" [17]

Los patrones frecuentes que se obtienes del conjunto de aciertos son encontrados utilizando la estructura de datos Árbol M-SP.

2.4.3.1. Árbol Max-Subpattern

La estructura del Árbol M-SP está diseñada para facilitar el registro del número de aciertos de cada M-SP y la derivación del conjunto de patrones frecuentes. Su diseño ejemplificado en la Figura 10 es el siguiente:

El Árbol M-SP toma $Cmax$ como el nodo raíz, donde cada subpatrón de $Cmax$ con un elemento no-* menos es un nodo secundario directo de la raíz. El árbol se expande recursivamente de acuerdo a las siguientes reglas:

- Un nodo w, si contiene más de dos elementos no-*, puede tener un conjunto de nodos hijos, donde cada uno de los nodos hijos es un subpatrón de w con una letra no-* menos.
- Un nodo con dos eventos no-* no tendrá hijos, pues los 1-patrones ya se encuentran presente en F_1.
- No se crean nodos ni descendientes de nodos que no son hit en S.
- Cada nodo tiene:
  - Campo contador de hits.
  - Vinculo al nodo padre (nulo para el nodo raíz).
  - Conjunto de vínculos a los nodos hijos. (cada vinculo a hijo señala a un hijo asociado con un elemento no-* menos. Puede ser nulo si no hay hits en hijos)

El algoritmo de inserción de elementos utilizado por el Árbol M-SP es presentado en Figura 9, según las instrucciones encontradas en [17],

Figura 9: Algoritmo de Árbol Max-Subpattern

Árbol Max-Subpattern	
Entrada:	Conjunto Hit-set, Cmax, soporte mínimo.
Salida:	Árbol Máx-Subpattern poblado.
1)	$Cmax$ es raíz del árbol.
2)	$\forall$ $h\text{-}sp \in H, h\text{-}sp = w$, recorrer árbol en búsqueda de nodo correspondiente a w, seguir ruta en relación a los eventos no-* faltantes
3)	Si w es encontrado en el árbol, nodo.contador+1.
4)	Si w no es encontrado en el árbol.
5)	Insertar w como nodo nuevo con contador=1
6)	Incluir nodos antecesores de w con contador=0 hasta conectar con el árbol.

Fuente: "Efficient Mining of Partial Periodic Patterns in Time Series Database" [17]

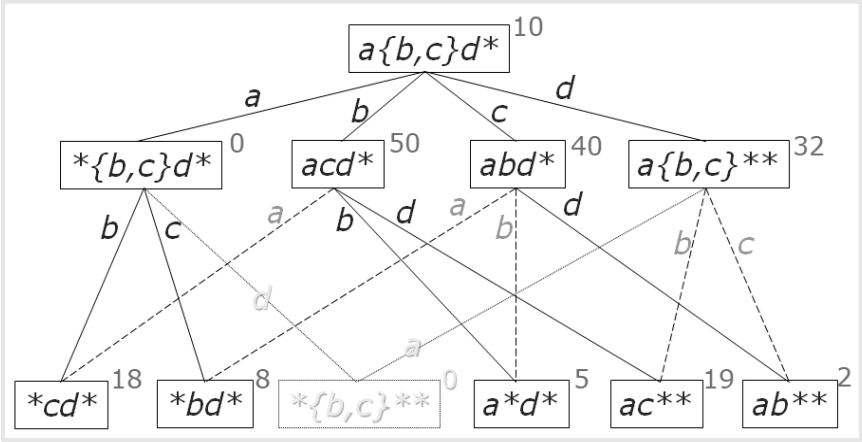
Sea w un subpatrón hit, siguiendo el método de inserción y tomando como ejemplo:

$$Cmax: \quad a\{b,c\}d*$$
$$w: \quad *cd*$$

La inserción de w en el árbol, asumiendo que este se encuentra vacío, requiere de la creación del nodo que contiene a w, asignándole un contador igual a 1, y la creación de los nodos antecesores con contador igual a cero hasta llegar al nodo raíz del árbol:

$$\underbrace{[a\{b,c\}d*]^{(0)}}_{Cmax} - \underbrace{[*\{b,c\}d*]^{(0)}}_{nodo\ intermedio} - \underbrace{[*cd*]^{(1)}}_{w}$$

Figura 10: Árbol Max-Subpattern para almacenamiento del conjunto de hits



Fuente: "Mining Time-Series Databases" [L8]

En Figura 10 se señala con una línea sólida entre nodos a una relación directa, obtenida de la inserción de elementos en el árbol, mientras que la línea punteada muestra una relación indirecta, la cual presenta una vía alternativa entre nodos y muestra posibles ancestros. Una vez insertados todos los elementos del conjunto de hits en el árbol, este puede ser recorrido y es posible calcular la frecuencia de conteo de cada nodo, la cual se obtiene al sumar los contadores de un nodo con todos sus posibles ancestros.

El conteo de frecuencia para el subpatrón $\{*\ cd\ *\}^{(18)}$ suma su contador con el contador de todos sus ancestros, tanto directos como indirectos, en este caso: $\{*\ \{b,c\}d\ *\}^{(0)}, \{acd\ *\}^{(50)}\ y\ \{a\{b,c\}d\ *\}^{(10)}$, dando una frecuencia de conteo para $\{*\ cd\ *\}$ de 78, mientras que el conteo de frecuencia para el subpatrón $\{ac\ **\}^{(19)}$ es de 111, considerando a sus ancestros: $\{acd\ *\}^{(50)}, \{a\{b,c\}\ **\}^{(32)}\ y\ \{a\{b,c\}d\ *\}^{(10)}$.

En el Capítulo 3, sección 3.2, se describe una versión del algoritmo M-SP orientada a la búsqueda de patrones en BDET.

CAPÍTULO 3.
PROPUESTA DE SOLUCIÓN

1. Introducción

El estudio realizado, análisis de distintos documentos y publicaciones relacionadas con las búsquedas de patrones en BDET, permitió centrar esta investigación en la implementación de algoritmos de minería de datos para la búsqueda de patrones, así como en la elaboración e implementación de un algoritmo propio. Se utilizó para ello unos conjuntos de datos sintéticos a modo de testeo para probar la correctitud de las implementaciones pera luego realizar pruebas con conjuntos de datos extraídos de fuentes reales.

Como se menciona anteriormente, los algoritmos seleccionados para este análisis son Apriori [16] y M-SP [17], mientras que el algoritmo propuesto para este proyecto es Minus-F_1.

El lenguaje de desarrollo elegido para el desarrollo de las soluciones ha sido Java, esto debido a que Java es un lenguaje de programación orientado a objetos, multiplataforma, robusto, confiable y altamente cotizado en el mercado, entre otras características.

2. Algoritmos Implementados

2.1. Algoritmo Apriori

En [17] se expone una versión del método que describe al algoritmo Apriori, adaptada para su uso sobre secuencias de datos obtenidas desde BDET. En la Figura 11 se presenta el algoritmo:

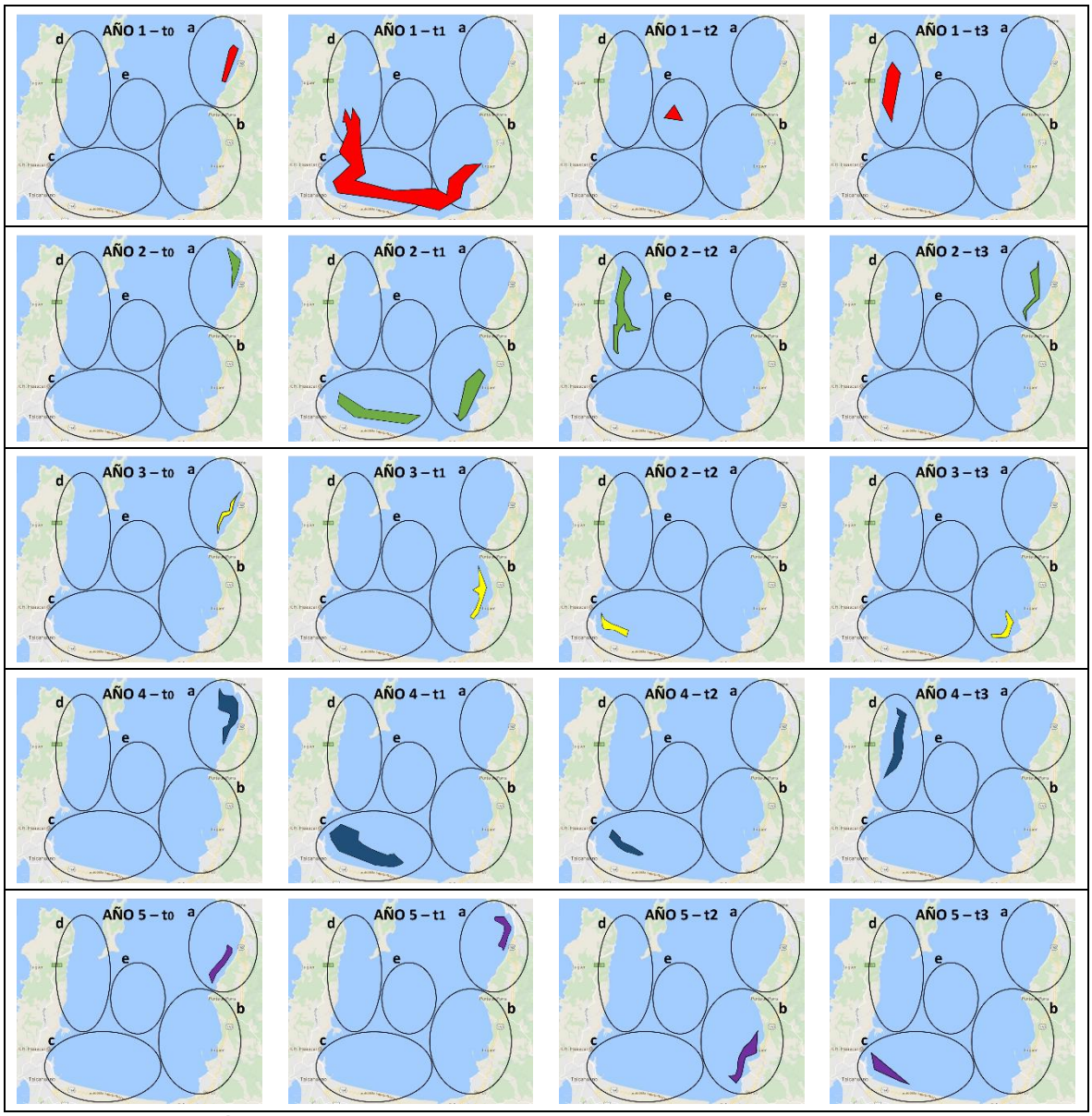
Figura 11: Algoritmo Apriori para Secuencias de Datos Espacio-Temporales

Apriori
Entrada: Secuencia de datos {S}, soporte mínimo % {s_m}, período máximo {p} **Salida:** Patrones periódicos para períodos desde 2 hasta p de la forma F_1 hasta F_j, $j \leq p$
1) Buscar eventos {m} en S, descartar repeticiones; 2) **Ciclo** Para los períodos desde el i=2 hasta i=p 3) Segmentar S, segmentos {s} de tamaño i; 4) Segmentos totales {s_t} = |S|/i, conteo mínimo {c_m} = s_t * s_m; 5) **Inicio** Generar F_1 6) Generar conjunto F_1 de segmentos f_1 de período i utilizando m; 7) Buscar cada segmento de F_1 $\{f_1\}$ en cada s de S, contar coincidencias; 8) Descartar f_1 cuyas coincidencias sean menores a c_m; 9) Mostrar F_1 (patrones f_1 en F_1); 10) **Fin 5)** 11) **Ciclo** Para j=2 hasta j=i o si F_{j-1} contiene menos de 2 f_{j-1} 12) **Inicio** Generar F_j 13) Para todos los f_1 y f_{j-1}, la unión de un f_1 y un f_{j-1} genera un f_j en F_j; 14) Buscar cada f_j en cada s de S, contar coincidencias; 15) Descartar f_j cuyas coincidencias sean menores a c_m; 16) Mostrar F_j (patrones f_j en F_j); 17) **Fin 12)** 18) **Fin 11)** 19) **Fin 2)**

Fuente: Elaboración propia basado en descripción del método de Apriori en [17]

Ejemplo: En Figura 12 se exhiben las zonas afectadas por el crecimiento irregular en la población de medusas Fragata Portuguesa, observadas en tramos trimestrales durante 5 años en diferentes zonas de las costas penquistas. Independiente del tamaño o la extensión de los eventos observados del fenómeno, en Tabla 7 se registraron la zonas afectada por este fenómeno para las temporadas en que se tomaron las muestras.

Figura 12: Mapa de crecimiento irregular de la población de Fragata Portuguesa en las costas penquistas.



Fuente: Elaboración propia

Este ejemplo describe un caso particular donde se evalúan los datos en un período conocido, por lo tanto, para este caso la iteración comprendida entre las líneas 2 y 18 del algoritmo Apriori (Figura 11) se realizará sólo una vez. Los datos antes mencionados pueden ser descritos en una secuencia de datos como una cadena, la cual representaría la información de este ejemplo:

$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e\ d}_{t_2\ t_3}}_{Año\ 1}\ \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a\ d}_{t_2\ t_3}}_{Año\ 2}\ \underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3}\ \underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4}\ \underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

Revisando los datos y aplicando una periodicidad anual (Figura 11, línea 3 y 4) tenemos:

$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e}_{t_2}\underbrace{d}_{t_3}}_{Año\ 1} - \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a}_{t_2}\underbrace{d}_{t_3}}_{Año\ 2} - \underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3} - \underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4} - \underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

Tabla 7: Registro de zonas con crecimiento irregular en la población de Fragata Portuguesa en las costas penquistas

Presencia del fenómeno por zonas en años	Tiempo (trimestres)			
	t_0 (ene-mar)	t_1 (abr-jun)	t_2 (jul-sep)	t_3 (oct-dic)
Año 1	a	b, c, d	e	d
Año 2	a	b, c	a	d
Año 3	a	b	c	b
Año 4	a	c	c	d
Año 5	a	a	b	c

Fuente: Elaboración propia

Apriori realiza el siguiente procedimiento para encontrar los patrones y sub-patrones frecuentes en esta serie espacio-temporal, la que a diferencia del ejemplo descrito en el Capítulo 2. Sección 2.4.2. Figura 7 debe considerar la dimensión de tiempo, por lo cual, interesa en qué posición temporal se observa cada evento.

En primera instancia se debe encontrar el conjunto de patrones candidatos F_1, compuesto por cada evento existente en la Base de Datos, considerando la periodicidad de esta, calcular su soporte (a razón de: $ocurrencia/periodos$) y luego conformar el conjunto F_1 con los eventos que cumplen un soporte mínimo (proceso comprendido en Figura 11 entre las líneas 5 a 10). Se usará un soporte mínimo de 3/5 para este ejemplo, lo que denota un soporte superior al 50%, este soporte estará dado por razón de tres coincidencias de cinco casos posibles.

$$\begin{aligned} eventos\ (zonas)\colon &\quad a, b, c, d \\ Candidatos\ F_1\colon &\quad \{a\ ***\colon 5/5,\ *\ a\ **\colon 1/5,\ **\ a\ *\colon 1/5,\ ***\ a\colon 0/5,\ b\ ***\colon 0/5, \\ &\quad *\ b\ **\colon 3/5,\ **\ b\ *\colon 1/5,\ ***\ b\colon 1/5,\ c\ ***\colon 0/5,\ *\ c\ **\colon 4/5, \\ &\quad **\ c\ *\colon 2/5,\ ***\ c\colon 1/5\ ,\ d\ ***\colon 0/5,\ *\ d\ **\colon 1/5,\ **\ d\ *\colon 0/5, \\ &\quad ***\ d\colon 3/5\} \\ F_1\colon &\quad \{a\ ***\colon 5/5,\ *\ b\ **\colon 3/5,\ *\ c\ **\colon 4/5,\ ***\ d\colon 3/5\} \end{aligned}$$

Luego se procede a generar el conjunto de candidato F_2 conformado por la unión de dos segmentos existentes en F_1 y posteriormente, dado el soporte mínimo, generar F_2 (iteración comprendida en Figura 11 entre líneas 11 a 18).

$$Candidatos\ F_2: \{ab **: 3/5, ac **: 3/5, a ** d: 3/5, *\{b,c\} **: 2/5, * b * d: 2/5, * c * d: 3/5\}$$
$$F_2: \{ab **: 3/5,\ ac **: 3/5, a ** d: 3/5, * c * d: 3/5\}$$

El conjunto de candidatos F_3 se genera uniendo dos segmentos en F_2, descartando todos los segmentos F_3 que contengan segmentos F_2 con soporte insuficiente (nueva iteración, Figura 11, líneas 11 a 18)

$$Candidatos\ F_3: \{a\{b,c\} **: No, ab * d: No, ac * d: 3/5\}$$
$$F_3: \{ac * d: 3/5\}$$

Como la cantidad de segmentos existentes en el conjunto F_3 es insuficiente para generar un conjunto de candidatos F_4, entonces el algoritmo concluye. El segmento existente en F_3 cumple con el soporte mínimo y se señala como máximo patrón que puede ser encontrado sobre esta secuencia. Los segmentos encontrados en F_2, al cumplir con el soporte mínimo también son considerados patrones parciales de la secuencia de datos.

Los patrones encontrados: $\{ab **\}$, $\{ac **\}$, $\{a ** d\}$, $\{* c * d\}$, $\{ac * d\}$ muestran las zonas que resultaron más afectadas por el fenómeno durante los 5 años estudiados y predicen el posible patrón de movilización de las medusas para años posteriores. Según el patrón encontrado $\{ac * d\}$ se puede indicar que las medusas son encontradas principalmente en el sector “a” en el primer trimestre del año, para luego presentar mayor actividad en el sector “c” en el segundo trimestre y un comportamiento indeterminado en el tercer trimestre, para concluir con sobrepoblación en el sector “d” durante el último trimestre del año.

2.2. Algoritmo Max-Subpattern

La versión descrita en [17] de los algoritmos M-SP Hit-Set (Figura 8) y Árbol M-SP (Figura 9) está diseñada para su uso sobre secuencias de datos espaciales y adaptable a su uso sobre secuencias espacio-temporales. Basado en estos, en Figura 13 se presenta el algoritmo M-SP para secuencias de datos espacio-temporales. Se reutiliza el ejemplo propuesto en la sección anterior (capítulo 3, sección 2.1) para mostrar el funcionamiento del algoritmo.

Ejemplo: Utilizando los datos obtenidos de Figura 12 y Tabla 7, considere la siguiente cadena como una secuencia de datos espacio-temporales con periodicidad 4.

$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e}_{t_2}\underbrace{d}_{t_3}}_{Año\ 1}\underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a}_{t_2}\underbrace{d}_{t_3}}_{Año\ 2}\underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3}\underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4}\underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

Según la periodicidad señalada previamente, la cantidad de segmentos periódicos existentes en esta serie son 5, como se señala en la Figura 13, líneas 3 y 4 del algoritmo M-SP:

$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e}_{t_2}\underbrace{d}_{t_3}}_{Año\ 1} - \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a}_{t_2}\underbrace{d}_{t_3}}_{Año\ 2} - \underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3} - \underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4} - \underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

En primera instancia, al igual que con Apriori, se debe encontrar el conjunto de candidatos F_1 compuesto por cada evento existente en la Base de Datos considerando la periodicidad de esta, calcular su soporte (a razón de: ocurrencia/períodos) y luego conformar el conjunto F_1, con los eventos que cumplen un soporte mínimo (Figura 13, líneas 5 a 10). Se usará un soporte de 3/5 para este ejemplo. Como este conjunto ya fue encontrado previamente, tenemos que:

$$F_1\colon \quad \{a\ ***\colon 5/5,\ *b\ **\colon 3/5,\ *c\ **\colon 4/5,\ ***d\colon 3/5\}$$

Una vez encontrado el conjunto F_1 se procede a conformar $Cmax$ y posteriormente encontrar el conjunto de subpatrones Hits H, descartando todas los segmentos de la forma F_1, es decir, segmentos periódicos con un solo elemento no-$*$. Las comparaciones son realizadas entre $Cmax$ y cada segmento de la secuencia S, como se indica en la Figura 13, líneas 12 a 15.

$$\begin{aligned} Cmax\colon &\quad a\{b,c\}*d \\ candidatos\ H\colon &\quad \{a\{b,c\}*d,\ a\{b,c\}*d,\ \text{ab}**,\ \text{ac}*\text{d},\ \text{a}***\} \\ H\colon &\quad \{a\{b,c\}*d\colon 2,\ \text{ab}**\colon 1,\ \text{ac}*\text{d}\colon 1\} \end{aligned}$$

El Árbol M-SP que se obtiene del conjunto de aciertos es presentado en Figura 14 y su construcción es descrita en la Figura 13 , líneas 17 a 25 del algoritmo. Se observa que, por ejemplo, la inserción del segmento hit $\{ab**\}$ en el árbol requiere de la creación del segmento hit $\{ab*d\}$, el cual es el nodo que logra conectar a $\{ab**\}$ con el nodo raíz del árbol $\{a\{b,c\}*d\}$ como sugieren las líneas 19 a 22 del algoritmo en la Figura 13.

Figura 13: Algoritmo Max-Subpattern para secuencias de datos espacio-temporales

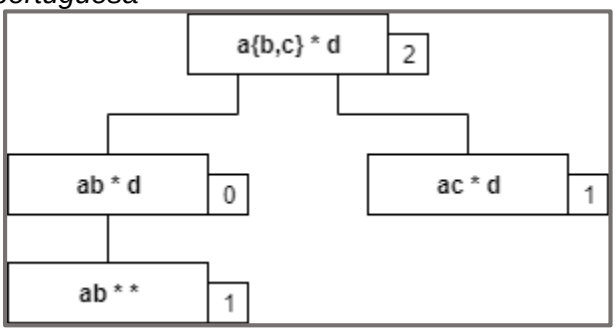
Max-Subpattern

Entrada: Secuencia de datos {S}, soporte mínimo % {s_m}, período máximo {p}
Salida: Patrones periódicos para períodos desde 2 hasta p de la forma F_1, Cmax y patrones periódicos completos y parciales (desde F_2 a F_j, $j \leq p$, sin catalogar)

```
1)  Buscar eventos {m} en S, descartar repeticiones;
2)  Para los períodos desde el i=2 hasta i=p
3)    Segmentar S, segmentos {s} de tamaño i;
4)    Segmentos totales {s_t} = |S|/i, conteo mínimo {c_m} = s_t * s_m;
5)    Inicio Generar F1
6)      Generar conjunto F1 de segmentos f1 de período i utilizando m;
7)      Buscar cada segmento de F1 {f1} en cada s de S, contar coincidencias;
8)      Descartar f1 cuyas coincidencias sean menores a c_m;
9)      Mostrar F1 (patrones f1 en F1);
10)   Fin 5)
11)   Si F1 contiene más de un f1
12)     Inicio Generar Cmax y Hits
13)       Generar el patrón máximo Cmax con la unión de todos los f1 en F1;
14)       Intersectar cada s en S con Cmax para obtener hit, descartar vacíos y tipo f1;
15)     Fin 12)
16)     Si existen 2 o más hits
17)       Inicio Generar árbol Max-Subpattern (raíz = Cmax);
18)         Inserción de hit en árbol, recorrer árbol;
19)         Si hit no existe en árbol
20)           Generar nodos desde Cmax hasta hit, nodo hijo = nodo padre -1 evento;
21)           Contador nodo hit = 1, contador nodos intermedios nuevos = 0;
22)         Fin 19)
23)         Si hit existe en árbol
24)           Contador nodo hit +1;
25)       Fin 17)
26)       Inicio Recorrer árbol M-SP y contar frecuencias
27)         Para todos los nodos hit en el árbol, Si contador de hit en nodo > 0
28)           Sumar al contador del hit el contador de nodos ancestros;
29)           Si contador final de hit ≥ c_m
30)             hit es un Max-Subpattern;
31)         Fin 27
32)         Mostrar Max-Subpatterns (hit finales con contador ≥ c_m);
33)       Fin 26)
34)     Fin 16)
35)   Fin 11)
36) Fin 2)
```

Fuente: Elaboración propia basado en descripción del método de Max-Subpattern en [17]

Figura 14: Árbol Max-Subpattern para ejemplo de zonas de crecimiento irregular de población de fragata portuguesa



Fuente: Elaboración propia

Realizando el conteo de frecuencias correspondiente, según indica en la Figura 13, líneas 26 a 33, se obtiene que las frecuencias de cada nodo quedan como sigue:

$$Frecuencia\ Hit(3): \quad \{a\{b,c\} * d: 2,\ \ ab **: 3,\ \ ac * d: 3\}$$

donde los nodos $\{ab **\}$, y $\{ac * d\}$ tienen como nodo ancestro al nodo raíz $\{a\{b,c\} * d\}$ y por tanto ambas suman el contador de este nodo a sus respectivos contadores, quedando para ambos casos con contador igual a tres, con lo cual cumplen con el soporte mínimo y por lo tanto son patrones encontrados periódicos por este algoritmo.

Apriori estima que también son patrones los segmentos: $* c * d$; $a * * d$; $ac * *$, los cuales son descartados por M-SP, Esto sucede ya que estos patrones (del conjunto F_2) son encontrados en los mismos segmentos en que aparece: $ac * d$ del conjunto F_3.

2.3. Algoritmo Minus-F_1[9]

El algoritmo Minus-F_1 desarrollado para efectos de este proyecto es presentado en Figura 15. Este algoritmo se presenta como alternativa a los algoritmos Apriori y M-SP en la búsqueda de patrones periódicos completos y parciales en BDET. Minus-F_1 tiene por objetivo reducir la cantidad de patrones y sub-patrones candidatos que se generan con los otros dos algoritmos mencionados.

Minus-F_1 utiliza el cálculo de frecuencia de los 1-patrones f_1 del conjunto F_1, obtenidos de los segmentos periódicos s observados en una secuencia de datos S (Figura 15, Líneas 2 a 7). Este algoritmo utiliza el conjunto de candidatos F_1, lo cual

[9] Algoritmo Minus-F1 es propuesto e implementado por el equipo de trabajo responsable de esta investigación (Fuentes, G., Rosas,D.)

incluye el uso de segmentos f_1 que tienen soporte menor al soporte mínimo establecido. El algoritmo realiza de forma secuencial, por cada segmento s de S la comprobación de los segmentos f_1 en F_1 que componen a s (Figura 15, Líneas 8 a 21), si cada f_1 que compone a s cumple con el soporte mínimo, s es almacenado como un patrón completo candidato p_c en conjunto P_C con contador igual a 1, mientras que si alguno de f_1 que compone a s no cumple con el soporte mínimo, este s es almacenado como un no patrón con contador n_p en conjunto N_P 1 (Figura 15, líneas 13 a 18). Por cada s procesado el contador de los f_1 en F_1 que componen a s se reduce en 1 (Figura 15, línea 19).

Previo al análisis de cada s es necesario verificar si este se encuentra en P_C o en N_P. Si este ya se encuentra en alguno de los conjuntos, no se realiza la comprobación de soporte y se aumenta el contador del segmento en el conjunto correspondiente (Figura 15, líneas 11 y 12), mientras que si el segmento no está en alguno de los conjuntos mencionados, se procesa de la forma descrita inicialmente.

El análisis de segmento concluye usa vez procesado el último segmento s de la secuencia de datos.

Una vez concluido el análisis de cada s en S se revisa el conjunto P_C (Figura 15, líneas 22 a 27) y son descartados todos los p_c en P_C que no hayan cumplido con el soporte mínimo, siendo enviados a N_P. Los patrones completos candidatos que igualen o superen el soporte mínimo pasan a ser parte del conjunto de Patrones Periódicos Completos.

Para encontrar los patrones parciales es necesario realizar la intersección de cada p_c en P_C con cada compañero de su conjunto, así como también intersectar cada n_p en N_P con cada compañero de su conjunto, además de intersectar cada p_c en P_C con cada n_p en N_P, el resultado de cada intersección generará un candidato a patrón parcial p_p que será almacenado en el conjunto P_P, descartando p_p que ya hayan sido incorporados en P_P y p_p de la forma F_1 (Figura 15, líneas 28 a 38). Una vez realizada las intersecciones correspondientes se actualiza el contador de cada p_p en base a los segmentos los conjuntos P_C y N_P y se descartan todos los p_p con contador menor al soporte mínimo.

Figura 15: Algoritmo Minus-F1 para Secuencias de Datos Espacio-Temporales

Minus-F_1

Entrada: Secuencia de datos {S}, soporte mínimo % {s_m}, período máximo {p}
Salida: Patrones periódicos para períodos desde 2 hasta p de la forma F_1, Cmax y patrones periódicos completos y parciales (desde F_2 a F_i, $j \leq p$, sin catalogar)

```
1)  Para los períodos desde el i=2 hasta i=p
2)    Segmentar S, segmentos {s} de tamaño i;
3)    Segmentos totales {s_t} = |S|/i, conteo mínimo {c_m} = s_t * s_m;
4)    Inicio Generar conjunto F1
5)      Generar cada segmento f1 de F1 en cada s de S, contar coincidencias;
6)      Mostrar F1 (patrones f1 en F1)
7)    Fin 4)
8)    Si al menos 2 segmento f1 de F1 tienen contador ≥ c_m
9)      Inicio Detectar candidatos a patrón completo {P_C} y no patrones{N_P}
10)       Para cada s de S
11)         Si s existe en P_C o N_P
12)           aumentar contador correspondiente (p_c en P_C o n_p en N_P);
13)         Si s no existe en P_C o N_P
14)           Si todos los f1 que componen a s cumplen c_m
15)             s pasa a P_C con contador 1;
16)           Si algún f1 que componen a s no cumple c_m
17)             s pasa a N_P con contador 1;
18)         Fin 13)
19)         Disminuir contador de f1 que componen a s;
20)       Fin 10)
21)     Fin 9)
22)     Inicio Detectar patrones completos
23)       Si P_C no está vacío
24)         Revisar contador de cada p_c en P_C, mover a N_P los p_c con contador < c_m;
25)         Mostrar P_C (patrones p_c en P_C);
26)       Fin 23)
27)     Fin 22)
28)     Inicio Detectar candidatos a patrón parcial {P_P}
29)       Si N_P no está vacío
30)         Si N_P contiene más de un n_p
31)           Intersectar cada n_p en N_P con sus pares para obtener p_p, Descartar
      intersecciones vacías, de forma f1 o p_p ya existentes en P_P;
32)         Si P_C no está vacío
33)           Intersectar cada n_p en N_P con cada p_c en P_C para obtener p_p, Descartar
      intersecciones vacías, de forma f1 o p_p ya existentes en P_P;
34)         Contador de cada p_p en P_P es igual a la suma de los contadores de los p_c en
      P_C y n_p en N_P tal que la intersección de p_p con ellos dé como resultado p_p;
35)         Descartar p_p en P_P con contador < c_m;
36)         Mostrar P_P (patrones p_p en P_P);
37)       Fin 29)
38)     Fin 28)
39)   Fin 8)
40) Fin 1)
```

Fuente: Elaboración propia

Ejemplo: Utilizando los datos obtenidos de Figura 12 y Tabla 7 de secciones anteriores (capítulo 3, sección 2.1): Considere la siguiente cadena como una secuencia de datos espacio-temporales con periodicidad 4.

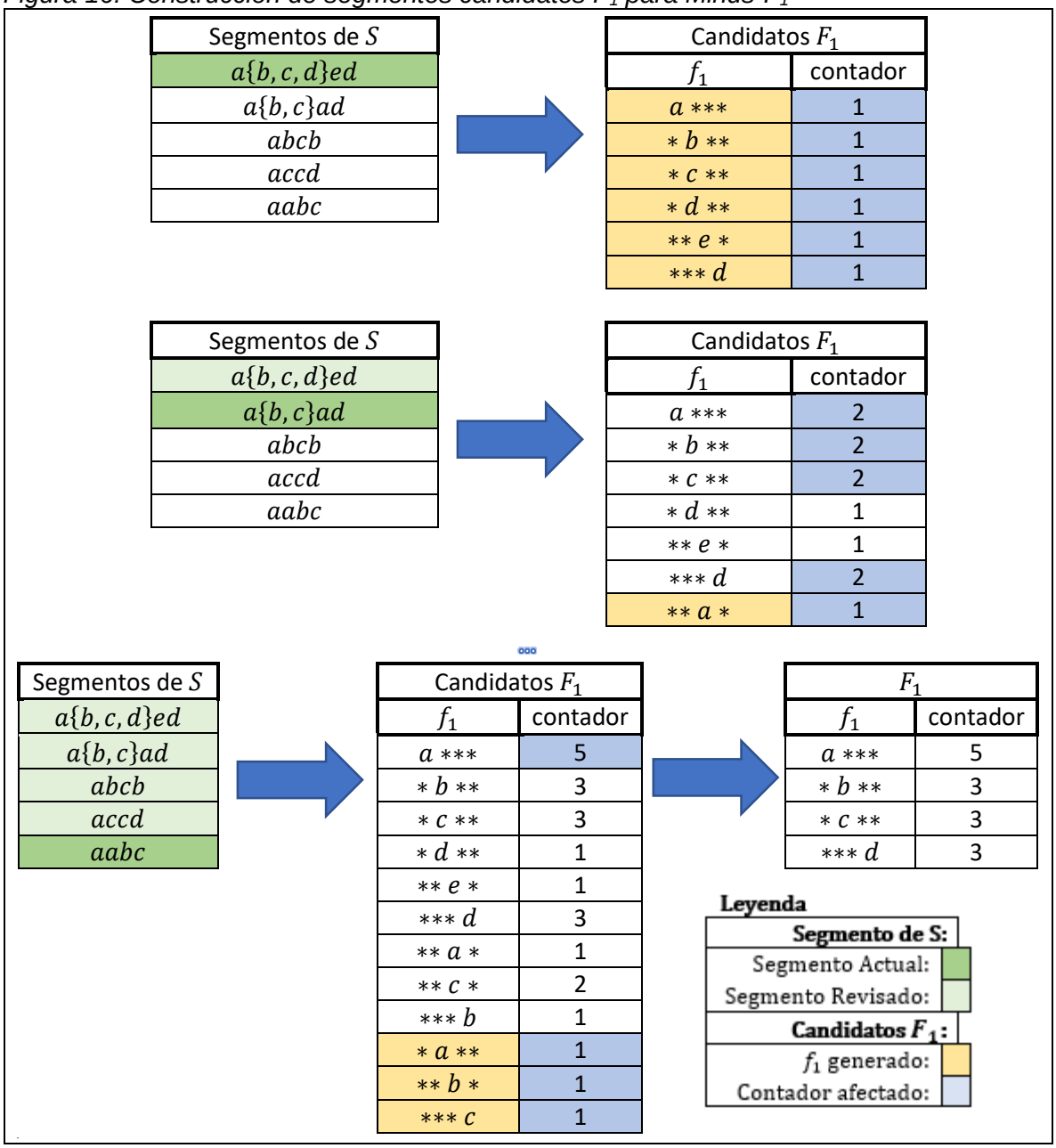
$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e}_{t_2}\underbrace{d}_{t_3}}_{Año\ 1}\underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a}_{t_2}\underbrace{d}_{t_3}}_{Año\ 2}\underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3}\underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4}\underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

Según la periodicidad señalada previamente, la cantidad de segmentos periódicos existentes en esta serie son 5:

$$S = \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c,d\}}_{t_1}\underbrace{e}_{t_2}\underbrace{d}_{t_3}}_{Año\ 1} - \underbrace{\underbrace{a}_{t_0}\underbrace{\{b,c\}}_{t_1}\underbrace{a}_{t_2}\underbrace{d}_{t_3}}_{Año\ 2} - \underbrace{\underbrace{a}_{t_0}\underbrace{b}_{t_1}\underbrace{c}_{t_2}\underbrace{b}_{t_3}}_{Año\ 3} - \underbrace{\underbrace{a}_{t_0}\underbrace{c}_{t_1}\underbrace{c}_{t_2}\underbrace{d}_{t_3}}_{Año\ 4} - \underbrace{\underbrace{a}_{t_0}\underbrace{a}_{t_1}\underbrace{b}_{t_2}\underbrace{c}_{t_3}}_{Año\ 5}$$

Al igual que con los algoritmos evaluados anteriormente se debe encontrar el conjunto de f_1 candidatos F_1. El proceso comprendido en Figura 15, líneas 3 a 6 del algoritmo, se desarrolla como ilustra la Figura 16. Posteriormente, como se expone en la Figura 17, se realizan los procedimientos comprendidos en Figura 15, líneas 9 a 21 del algoritmo.

Figura 16: Construcción de segmentos candidatos F_1 para Minus-F_1



Fuente: Elaboración propia

Figura 17: Proceso de detección de candidatos de patrón completo y no patrones

Segmentos de S
$a\{b,c,d\}ed$
$a\{b,c\}ad$
$abcb$
$accd$
$aabc$

F_1		≥ soporte	segmento cont-1
f_1	contador		
a ***	5	si	4
* b **	3	si	2
* c **	3	si	2
* d **	1	no	0
** e *	1	-	0
*** d	3	-	2
** a *	1		
** c *	2		
*** b	1		
* a **	1		
** b *	1		
*** c	1		

no patrón	Contador
$a\{b,c,d\}ed$	1

candidato	Contador
∅	

Leyenda

Segmento de S:	
Segmento actual:	
Segmento revisado:	

Segmentos de S
$a\{b,c,d\}ed$
$a\{b,c\}ad$
$abcb$
$accd$
$aabc$

F_1		≥ soporte	segmento cont-1
f_1	contador		
a ***	4	si	3
* b **	2	no	1
* c **	2	-	1
* d **	0		
** e *	0		
*** d	2	-	1
** a *	1	-	0
** c *	2		
*** b	1		
* a **	1		
** b *	1		
*** c	1		

no patrón	Contador
$a\{b,c,d\}ed$	1
$a\{b,c\}ad$	1

candidato	Contador
∅	

Leyenda

Candidatos F_1:	
Revisado aprobado	
Revisado rechazado	
No revisado	
Contador f_1-1	

Segmentos de S
$a\{b,c,d\}ed$
$a\{b,c\}ad$
$abcb$
$accd$
$aabc$

F_1		≥ soporte	segmento cont-1
f_1	contador		
a ***	1	no	0
* b **	0		
* c **	0		
* d **	0		
** e *	0		
*** d	0		
** a *	0		
** c *	0		
*** b	0		
* a **	1	-	0
** b *	1	-	0
*** c	1	-	0

no patrón	Contador
$a\{b,c,d\}ed$	1
$a\{b,c\}ad$	1
$abcb$	1
$accd$	1
$aabc$	1

candidato	Contador
∅	

Leyenda

Candidatos y no patrones:	
Segmento generado:	
Contador afectado:	

Fuente: Elaboración propia

Una vez verificados todos los segmentos se observa que P_C se encuentra vacío, esto quiere decir que ningún s de S representa a un patrón completo y por tanto, para este caso no se accede a las líneas 22 a 27 del algoritmo (Figura 15).

Se realizan las acciones indicadas en Figura 15, línea 31 para identificar p_p candidatos, luego se realizan las acciones comprendidas en Figura 15, línea 34 a 36 para establecer la aparición de cada p_p dentro de cada s en S y verificar si cumplen con el soporte mínimo, como se ilustra en la Figura 18.

Figura 18: Detección de patrones parciales con Minus-F1

no patrón	Contador
$a\{b,c,d\}ed$	1
$a\{b,c\}ad$	1
$abcb$	1
$accd$	1
$aabc$	1

parciales	Contador
$a\{b,c\}*d$	2
$ab**$	3
$ac*d$	3
$a*c*$	2

Patrón p	Contador
$ab**$	3
$ac*d$	3

Fuente: Elaboración propia

Lo observado en el último conjunto (Figura 18) muestra los patrones parciales que cumplen con el soporte mínimo y son considerados patrones periódicos en este ejemplo.

Al igual que M-SP (Capítulo 3, sección 2.2.), Minus-F1 omite los patrones candidatos que son un subconjunto de un patrón de mayor tamaño si los segmentos de ese subconjunto tienen período igual al mayor patrón, al contrario de Apriori que los encuentra todos.

3. Coordenadas y trayectorias

Para este trabajo se ha optado por estudiar el comportamiento de movimiento en relación a su posición en un instante de tiempo de usuarios, es decir, la geolocalización de personas. Basados en los registros obtenidos desde el Proyecto Geolife [L5], Base de Datos de coordenadas GPS, se observa el comportamiento de movimiento de usuarios para determinar la existencia de patrones periódicos. Los datos que contiene los archivos de Geolife son representaciones de trayectorias de usuarios en un determinado lugar, basados en latitud y longitud en determinado momento del día de una determinada fecha, con variaciones de 5 segundos entre registros en la mayoría de casos.

Cada archivo en el proyecto Geolife, en formato ".plt" presenta la trayectoria de un usuario en un día en particular, durante un período de tiempo que esté dentro de ese mismo día. Un fragmento de los datos contenidos en los archivos de Geolife es lo que se presenta en Tabla 8:

Tabla 8: Ejemplo de archivo .plt de Geolife

1	2	3	4	5	6	7
39.118683	117.242273	0	41	39335.3239699074	2007-09-10	07:46:31
39.118685	117.242265	0	39	39335.3239814815	2007-09-10	07:46:32
39.11869	117.242258	0	38	39335.3239930556	2007-09-10	07:46:33

Fuente: Elaboración propia basado en [L5]

La guía de usuario (v1.3) del Proyecto Geolife describe cada uno de los elementos como sigue:

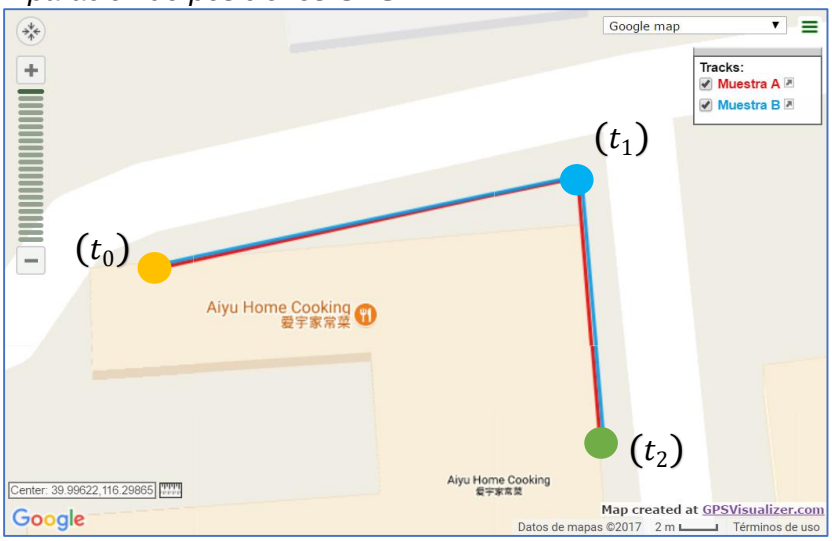
- Columna 1: Latitud, en grados decimales
- Columna 2: Longitud, en grados decimales
- Columna 3: Un valor "0" para todo el dataset (no se describe su uso[10])
- Columna 4: Altitud, en pies
- Columna 5: Días transcurridos desde el 30/12/1899
- Columna 6: Fecha del registro de la muestra en formato año-mes-día
- Columna 7: Hora en que fue obtenida de la muestra

El procedimiento de búsqueda de patrones para este dataset se desarrolla considerando la posición del usuario, dada por su latitud y longitud, en relación a la

[10] Puesto a prueba la utilidad del valor "0" correspondiente al campo 3, haciendo uso de la aplicación GPSvisualizer [L6] con un archivo de trayectoria en formato ".plt", ante la presencia del valor 0, la imagen que se muestra con la aplicación es la trayectoria descrita por los puntos coordenados presentes en el archivo, mientras que, ante ausencia de éste valor 0, la imagen resultante son solo las coordenadas presentes en el archivo, sin una conexión.

fecha de cada registro, aunque el nivel de destalle que presentan los datos de latitud, longitud y que ambos son presentados en grados, con hasta seis decimales, es probable que la búsqueda de patrones sobre estos datos produzca resultados desalentadores. La información que entrega un GPS, sobre la localización de un usuario móvil en un determinado instante de tiempo puede tener variaciones mínimas, con un rango de error válido de posición de algunos metros. Dos (o más muestras) que visualmente están en la misma posición o tienen el mismo recorrido podrían registrarse en posiciones GPS diferentes a causa del rango de error. La Figura 19 muestran dos trayectos a simple vista iguales, pero con coordenadas diferentes.

Figura 19: Comparación de posiciones GPS



Fuente: Elaboración propia, utilizando aplicación GPSvisualizer

Tabla 9: Coordenadas de segmentos A y B

	A	B
t_0	39.996225,116.298500	39.996226,116.298499
t_1	39.996265,116.298742	39.996266,116.298741
t_2	39.996150,116.298755	39.996151,116.298756

Fuente: Elaboración propia

Las coordenadas de los segmentos en la Figura 19 están indicados en Tabla 9 y presentan una diferencia en latitud y longitud mínima. Dado lo anterior, si se realiza una comparación de igualdad entre las trayectorias que exponen las columnas A y B de la Tabla 9 los resultados indican desigualdad, aunque es evidente a la vista que estas dos muestras son similares y por tanto es posible utilizar métodos que den este trato a los datos, por ejemplo algún algoritmo

presentado en [18] o [19][11]. La búsqueda de patrones en el proyecto Geolife tienen variaciones similares a las presentes en la Figura 19, si se utilizan métodos de búsqueda de patrones directamente sobre estos datos las coincidencias encontradas serán mínimas o nulas si estas no son tratadas como posiciones similares. Es probable que la agrupación de los registros GPS en conjuntos o regiones aumente la posibilidad de encontrar coincidencias sobre este dataset. En el presente proyecto se propone la creación de un método sencillo de transformación de datos que permita representar los datos de la forma expuesta en Tabla 12.

3.1. Algoritmo de Depuración de Secuencia de Datos Busca-Zonas[12]

Basados en los pasos del proceso de KDD, es necesario realizar un pre-procesamiento y transformación de datos antes de pasar al proceso de búsqueda y extracción de patrones. Como se muestra en la Figura 20, se propone la transformación de datos utilizando el siguiente algoritmo de transformación de coordenadas, denominado Busca-Zonas. El algoritmo realiza una categorización de las posiciones coordenadas en base a la subdivisión del área en que se posiciona un conjunto de coordenadas.

El método que describe al algoritmo presentado en la Figura 20 es el siguiente:

Dado un conjunto de coordenadas, con valores de latitud y longitud en grados decimales, generar una estructura que permita asociar coordenadas en zonas (segmentos). Cada segmento representa una sección delimitada en la región al que pertenece el conjunto de coordenadas.

- Leer coordenadas.
- Identificar latitud y longitud máxima y mínima.
- Delimitar región (puede incluir un margen de error de 0,0002 grados o inferior para cada valor).
- Obtener diferencias entre máximos y mínimos tanto de latitud como de longitud (Δ).
- Dividir latitud y longitud en segmentos (valor por determinar), parta desde los valores mínimos de latitud y longitud.

[11] Los artículos [18] y [19] exponen múltiples formas de evaluar patrones de trayectorias y similitudes de estas, ya sean técnicas de clustering o funciones de distancia entre otros. Si bien son técnicas muy interesantes de investigar, no hacen referencia a métodos basados en reglas de asociación, que es el punto fuerte de éste trabajo.

[12] Algoritmo Busca-Zonas es propuesto e implementado por el equipo de trabajo responsable de esta investigación (Fuentes, G., Rosas,D.)

- Asignar índice a cada cuadrante obtenido.

Secuencialmente asociar cada coordenada del archivo en la zona correspondiente, obtener el índice de la zona e insertando este índice en la secuencia de datos S.

Figura 20: Algoritmo Busca-Zonas

Busca-Zonas

Entrada: Lista de coordenadas de la forma (latitud, longitud ,tiempo) donde cada terna es un evento, segmentos de latitud {seglat}y segmentos de longitud {seglon}
Salida: Lista de zonas de coordenadas ordenadas, agrupada en Secuencia de datos {S}

```
1)  Identificar latitud y longitud máxima y mínima (lat_min, lat_max, lon_min, lon_max);
2)  Añadir margen de error  pequeño para valores anteriores;
3)  Inicio Generar área de movimiento de trayectoria
4)    Inicio Obtener Deltas y medida de segmentos
5)      delta_lat = (lat_max – lat_min);
6)      mSegLat = delta_lat/seglat
7)      delta_lon (lon_max - lon_min);
8)      mSegLon = delta_lon/seglon;
9)    Fin 4)
10)   Inicio Delimitar zonas y asignar índice
11)     Para j=0 hasta j < seglat
12)       Para k=0 hasta k < seglon definir
13)         Índice del zona[j,k];
14)         Latitud mínima del segmento ( minLat[j,k] = lat_min + mSegLat * j );
15)         Latitud máxima del segmento ( minLat[j,k] = lat_min + mSegLat * (j+1) );
16)         Longitud mínima del segmento ( minLat[j,k] = lat_min + mSegLat * k )
17)         Longitud máxima del segmento ( minLat[j,k] = lat_min + mSegLat * (k+1) );
18)       Fin 12)
19)     Fin 11)
20)   Fin 10)
21)   Inicio Identificar coordenadas en zonas
22)     Para cada coordenada recorrer cada zona
23)       Si coordenada está dentro de los límites de una zona determinada
24)         Guardar índice de zona de coordenada en secuencia de datos {S};
25)         Pasar a coordenada siguiente a consultar;
26)       Fin 23)
27)       Sino, si coordenada no está dentro de los límites de una zona determianda
28)         Pasar a la zona siguiente a consultar;
29)     Fin 22)
30)     Resultado: Secuencia de datos {S}
31)   Fin 21)
32) Fin 3)
```

Fuente: Elaboración propia

Ejemplo del algoritmo Busca Zonas:

Sea la lista de coordenadas mostrada en la Tabla 10, que representa la trayectoria de un vehículo particular, donde el primer valor representa la latitud,

seguido de la longitud y el tiempo en que se tomó el registro de la posición, representado gráficamente en la Figura 21.

Tabla 10: Lista de coordenadas, ejemplo Busca-Zonas

Latitud	Longitud	Tiempo
39.991312	116.320588	1
39.991298	116.320080	2
39.991298	116.319505	3
39.991268	116.319062	4
39.991252	116.318799	5
39.991264	116.318720	6
39.991233	116.318483	7
39.991239	116.318145	8
39.991197	116.317721	9
39.991180	116.317253	10

Fuente: Elaboración propia

Figura 21: Trayectoria basado en Tabla de coordenadas, ejemplo algoritmo Busca-Zonas

Fuente: Elaboración propia, utilizando aplicación GPSvisualizer [L6]

Tabla 11: Máximos, mínimos y diferencias en latitud y longitud. A: datos en bruto, B: datos con margen de error aplicado

A	Latitud	Longitud
Máximo	39.991312	116.320588
Mínimo	39.99118	116.317253
Δ	0.000132	0.003335

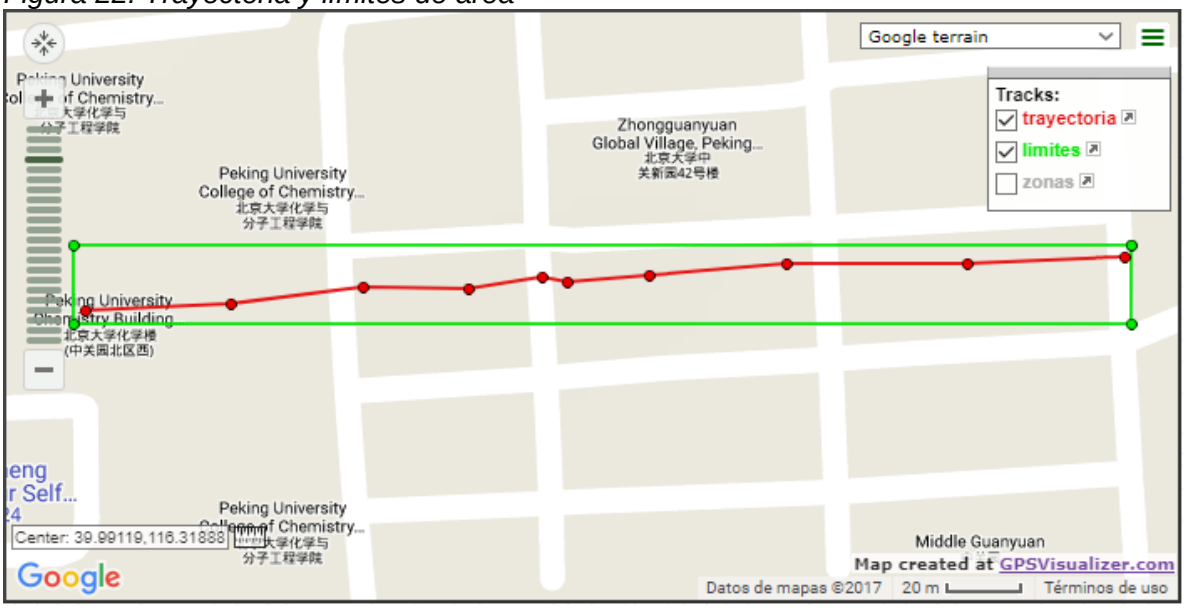
B	Latitud	Longitud
Máximo	39.99134	116.32061
Mínimo	39.99115	116.31722
Δ	0.00019	0.00339

Fuente: Elaboración propia

El algoritmo inicia detectando la latitud y longitud máxima y mínima presentes en la Tabla 10, valores que delimitan el área en donde se mueve la muestra. Se recomienda añadir un margen de error pequeño (sumar a máximo y restar a mínimo) para evitar que los puntos queden en los límites del área, luego se calcula la diferencia entre latitud y longitud máxima y mínima.

El caso presentado en la Tabla 11-A muestra los valores mínimos y máximos obtenidos directamente de la lista de coordenadas, el caso presentado en la Tabla 11-B añade a los valores en la Tabla 11-A un margen de error (0.00003°). En la Figura 22 se observa la trayectoria dentro del área definida en la Tabla 11-B.

Figura 22: Trayectoria y límites de área



Fuente: Elaboración propia, usando aplicación GPSvisualizer [L6]

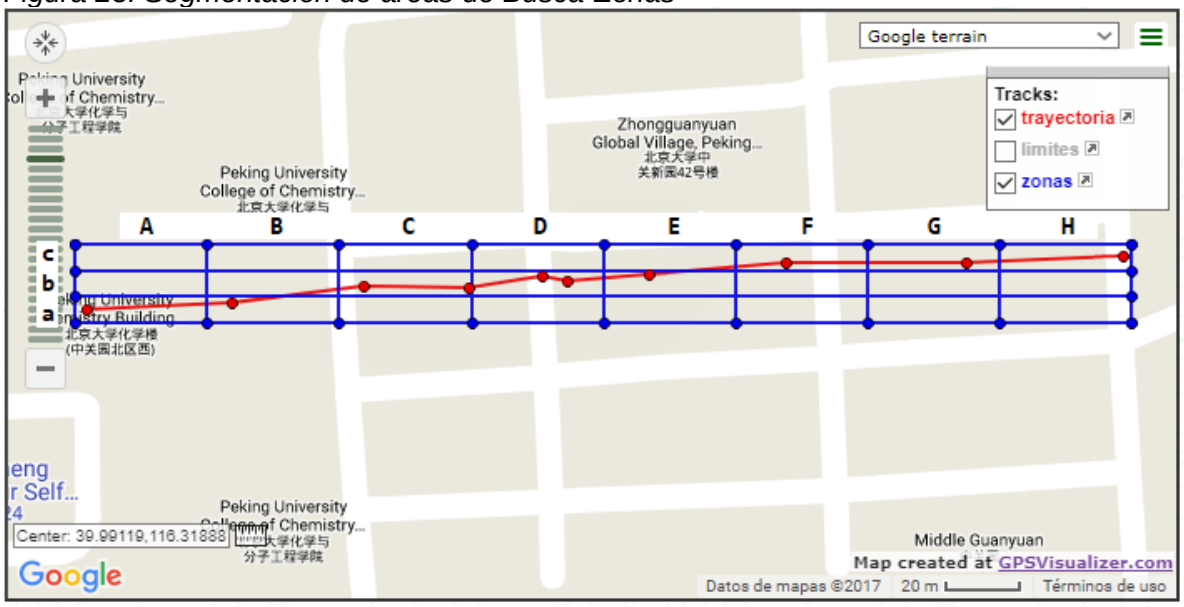
Luego de delimitada el área a la cual corresponden los puntos coordenados, se debe seccionar el área en segmentos de igual tamaño, esto a criterio a criterio del usuario. De forma arbitraria para este ejemplo se subdividirá la latitud en 3 partes y la longitud en 8 partes. La cantidad de segmentos estará dada por el resultado del producto de ambas partes.

Se divide el Δ de latitud y el Δ longitud por las partes indicadas para determinar el intervalo tanto en latitud como en longitud de cada segmento.

$$\frac{\Delta \text{Lat}}{3} = 0.00019/3 = \ 0.0000633; \ \frac{\Delta \text{Lon}}{8} = 0.00339/8 = \ 0.000424$$

Se inicia la división del área desde los puntos mínimos de latitud y longitud que se indican en Tabla 11-B y se utilizan los intervalos calculados previamente. Una vez establecidas las divisiones del área tratada se le asigna un índice a cada zona según latitud y longitud, como se indica en la Figura 23.

Figura 23: Segmentación de áreas de Busca-Zonas



Fuente: Elaboración propia, usando aplicación GPSvisualizer [L6]

Cada coordenada es asociada en la zona previamente establecida, una vez identificada la coordenada en la estructura se extrae el identificador de la zona, siendo insertado en una lista. La lista contendrá las zonas en donde estaban ubicadas las coordenadas en el mismo orden temporal. El resultado del ejemplo se muestra en la Tabla 12.

Tabla 12: Representación de coordenadas como secuencia de datos

Coordenada (lat, lon, tiempo)	Sector Latitud	Sector Longitud	Cuadrante
39.991312, 116.320588, 1	c	H	cH
39.991298, 116.320080, 2	c	G	cG
39.991298, 116.319505, 3	c	F	cF
39.991268, 116.319062, 4	b	E	bE
39.991252, 116.318799, 5	b	D	bD
39.991264, 116.318720, 6	b	D	bD
39.991233, 116.318483, 7	b	C	bC
39.991239, 116.318145, 8	b	C	bC
39.991197, 116.317721, 9	a	B	aB
39.991180, 116.317253, 10	a	A	aA
Secuencia: cH,cG,cF,bE,bD,bD,bC,bC,aB,aA			

Fuente: Elaboración propia

Los caracteres alfabéticos en minúscula y mayúscula que representan los sectores de los segmentos son sólo representativos y pueden variar para otros casos, lo importante es que cada segmento quedará representado con dos caracteres. Las coordenadas son representadas entonces por una lista alfanumérica la cual facilita la interpretación y comparación de datos para el presente caso de estudios, esta lista será la representación del conjunto de datos y denominada serie espacio-temporal. Note que el área está dada por las coordenadas de mayor y menor valor numérico y la división de zonas se realizará en base a las necesidades del usuario.

En resumen: el algoritmo Busca Zonas desarrollado recibe un archivo con coordenadas de posicionamiento GPS en un rango de tiempo determinado y entrega como resultado un archivo, llamado “cadena.txt”. El contenido de cadena.txt es la representación de la secuencia de datos espacio-temporal que será luego interpretado y analizado por los algoritmos Apriori, M-SP y Minus-F_1, adaptados para una correcta implementación sobre este tipo de datos.

En el Anexo de este informe se presenta un ejemplo que utiliza la nomenclatura de los datos como de la forma que es entregada por el algoritmo busca-zonas para luego ser evaluada por los algoritmos Apriori, M-SP y Minus-F_1.

4. Orden de Algoritmos Implementados

La evaluación de los algoritmos presentada a continuación expone la cantidad de patrones candidatos que genera cada uno de los algoritmos bajo condiciones similares en base a esta evaluación se obtiene el orden de cada algoritmo. Las condiciones comunes para los diferentes casos son las siguientes.

Sea n el tamaño de la secuencia de datos a evaluar, m la cantidad de eventos observados en la secuencia $(m \leq n)$, p el período para el cual se evalúa la secuencia y define el el tamaño de los segmentos en la secuencia $(p \leq n/2)$, f en referencia al conjunto f-patrones de candidatos con f eventos no-$*$ $(f \leq p)$ y un soporte igual a 0, lo que indica la no existencia de filtro a la hora de generar conjuntos de patrones candidatos para los diferentes algoritmos.

4.1. Apriori

La fórmula que describe la cantidad total de patrones candidatos que Apriori es capaz de generar para un período y grupo de f-patrones determinado está dado por la ecuación:

$$\boldsymbol{m^f} * \boldsymbol{C_{p,f}} = \boldsymbol{m^f} * \binom{\boldsymbol{p}}{\boldsymbol{f}} = m^f * \frac{p!}{(p-f)! * f!}$$

Donde $\boldsymbol{m^f}$ es la cantidad total de formas de tamaño f que se pueden generar con los m eventos y $\boldsymbol{C_{p,f}}$ las formas de agrupar f eventos $\boldsymbol{no}$-$*$ en un segmento de tamaño p. Por ejemplo, si tenemos 3 eventos: $A, B, C,$ las posibles combinaciones con 3 eventos que podemos hacer con ellos y las posibles formas de agrupar esos 3 eventos (formando segmentos de tipo f_3) en un segmento de tamaño 5 son presentadas en la Tabla 13 y descritas por la siguiente ecuación:

$$\boldsymbol{m^f} * \boldsymbol{C_{p,f}} = \boldsymbol{3^3} * \binom{\boldsymbol{5}}{\boldsymbol{3}} = 27 * \frac{5!}{(5-3)! * 3!} = 27 * \frac{5 * 4 * \cancel{3!}}{2! * \cancel{3!}} = 27 * 10 = 270$$

Donde cada combinación existente en la Tabla 13, sección A será representada en todas las formas disponibles en Tabla 13, sección B.

Tabla 13: Generación de candidatos de Apriori

A) $\boldsymbol{m^f = 27}$			**B)** $\boldsymbol{C_{p,f} = 10}$
AAA	BAA	CAA	$XXX--$
AAB	BAB	CAB	$XX-X-$
AAC	BAC	CAC	$XX--X$
ABA	BBA	CBA	$X-XX-$
ABB	BBB	CBB	$X-X-X$
ABC	BBC	CBC	$X--XX$
ACA	BCA	CCA	$-XXX-$
ACB	BCB	CCB	$-XX-X$
ACC	BCC	CCC	$-X-XX$
			$--XXX$

Fuente: Elaboración propia

Apriori genera todos los conjuntos f-pattern, desde $f = 1$ hasta $f = p$ en busca de patrones candidatos. El algoritmo implementado realiza una búsqueda de patrones sobre diferentes períodos, desde $p = 2$ hasta $p = n/2$, de esta forma si buscamos todos los segmentos que conforman el conjunto de 1-patrones (F_1) la ecuación resultante es:

$$\sum_{p=2}^{n/2} m * p$$

Mientras que la cantidad de segmentos candidatos que se generan para cada período y que conforman cada conjunto f-pattern está dada por la ecuación:

$$\sum_{\boldsymbol{p=2}}^{\boldsymbol{n/2}} \sum_{\boldsymbol{F=1}}^{\boldsymbol{p}} \boldsymbol{m^F} * \frac{\boldsymbol{m^F} * \boldsymbol{p}!}{(\boldsymbol{p}-\boldsymbol{f})! * \boldsymbol{f}!}$$

Por ejemplo, si $n = 6$ tenemos:

$$\sum_{p=2}^{n/2}\sum_{f=1}^{p}\frac{m^f * p!}{(p-f)! * f!}$$

$$= \overbrace{\overbrace{\frac{m^1 * 2!}{(2-1)! * 1!}}^{f_1} + \overbrace{\frac{m^2 * 2!}{(2-2)! * 2!}}^{f_2}}^{p_2}$$

$$+ \overbrace{\overbrace{\frac{m^1 * 3!}{(3-1)! * 1!}}^{f_1} + \overbrace{\frac{m^2 * 3!}{(3-2)! * 2!}}^{f_2} + \overbrace{\frac{m^3 * 3!}{(3-3)! * 3!}}^{f_3}}^{p_3}$$

$$= \frac{m^{\cancel{1}} * 2!}{\cancel{(2-1)!} * \cancel{1!}} + \frac{m^2 * \cancel{2!}}{\cancel{(2-2)!} * \cancel{2!}} + \frac{m^{\cancel{1}} * 3!}{(3-1)! * \cancel{1!}} + \frac{m^2 * 3!}{\cancel{(3-2)!} * 2!} + \frac{m^3 * \cancel{3!}}{\cancel{(3-3)!} * \cancel{3!}}$$

$$= 2m + m^2 + 3m + 3m^2 + m^3$$

$$= 5m + 4m^2 + m^3$$

Del ejemplo, el polinomio de mayor exponente es una cota superior para el cálculo de candidatos de patrones periódicos generados por Apriori, donde el exponente resulta ser el mayor valor que puede tomar p dentro de una secuencia de datos, $p = n/2$ donde $n = 6$, por lo tanto el máximo valor p para este caso es $p = 3$.

Con lo anterior podemos señalar que obtener todos los candidatos a patrón con Apriori es un procedimiento de orden:

$$\boldsymbol{O\left(m^{\frac{n}{2}}\right)}$$

Se debe tener en consideración el orden que supone recorrer la secuencia de datos, por lo cual el orden final del algoritmo es:

$$\boldsymbol{O\left(m^{\frac{n}{2}} * n\right)}$$

4.2. Max-Subpattern

El algoritmo M-SP genera tantos arboles como períodos evalúe en una secuencia de datos, exceptuando el periodo 1 (periodo de un solo evento). Si dicha secuencia es de tamaño n, la cantidad máxima de periodos p a evaluar es $n/2$, por lo tanto M-SP llega a generar $(n/2) - 1$ árboles. Cada uno de estos árboles puede tener una altura máxima de $|Cmax| - 2$.

La cantidad máxima de nodos para un árbol están dados por la siguiente ecuación:

$$\sum_{i=0}^{Cmax-1} (m * p)^i = \frac{(m * p)^{Cmax} - 1}{(m * p) - 1}$$

m: número de eventos

Todos los nodos de todos los árboles posibles, desde $p = 2$ hasta $p = n/2$, donde $Cmax$ puede alcanzar el tamaño $n/2$, está dado por la siguiente ecuación:

$$\sum_{j=2}^{n/2} \sum_{i=0}^{Cmax-1} (m * j)^i = \sum_{j=2}^{n/2} \frac{(m * j)^{Cmax} - 1}{(m * j) - 1} \leq \sum_{j=2}^{n/2} (m * j)^{Cmax} - 1 \quad \mathbf{(1)}$$

$$\mathbf{(1)} \quad \leq \quad \frac{\left[\left(m * \frac{n}{2}\right)^{\frac{n}{2}}\right]^{\frac{n}{2}+1} - 1}{\left(m * \frac{n}{2}\right)^{\frac{n}{2}} - 1} \quad \leq \quad \frac{\left(m * \frac{n}{2}\right)^{\frac{n^2}{4}+\frac{n}{2}} - 1}{\left(m * \frac{n}{2}\right)^{\frac{n}{2}} - 1} \qquad \mathbf{(2)}$$

Del numerador de $(\mathbf{2})$ tenemos que:

$$\left(m * \frac{n}{2}\right)^{\frac{n^2}{4}+\frac{n}{2}} - 1 \leq \left(m * \frac{n}{2}\right)^{\frac{n^2}{4}}$$

El resultado obtenido señala la cantidad de nodos del árbol M-SP que es capaz de llegar a generar este algoritmo. Dado esto podemos señalar que el orden de algoritmo para la generación de candidatos es:

$$O\left(\sqrt[4]{\left(\frac{m * n}{2}\right)^{n^2}}\right)$$

Se debe tener en consideración el orden que supone recorrer la secuencia de datos, por lo cual el orden final del es:

$$O\left(\sqrt[4]{\left(\frac{m*n}{2}\right)^{n^2}}*n\right)$$

4.3. Minus-F1

El algoritmo Minus-F_1 recorre la secuencia de datos una vez por cada periodo evaluado, lo que toma $O(n)$, luego para los períodos comprendidos entre 2 hasta $n/2$ se verifica la existencia de segmentos candidato a patrón. Bajo estas condiciones se tiene que el segmento más grande alcanza un tamaño igual $n/2$, lo mismo para descontar del contador la frecuencia, suponemos $n/2$
Así:

para $(p = 2 \rightarrow\ n/2)$

para $(i = 1 \rightarrow n/p)$, con n/p segmentos en la secuencia, lo que es equivalente a recorrer toda la secuencia de entrada.

{ si es candidato

(n/p) $O(\frac{n}{2}*\frac{n}{p}*\frac{n}{p})$

else

(n/p)

}

$\therefore\ O\left(\frac{n^3}{p^2}\right), donde\ p = n/2$

$\therefore\ O\left(\frac{n^3}{\frac{n^2}{4}}\right) \approx O(n)$

en el mejor de los casos y cuando el soporte tiende a 0

CAPÍTULO 4:
PRUEBAS

1. Introducción

Este capítulo muestra los resultados obtenidos de las pruebas realizadas utilizando la implementación en JAVA de los algoritmos Apriori, M-SP y Minum-F1. Cada caso de prueba realizado considera la utilización de un mismo conjunto de datos para los tres algoritmos, del cual se extrae el tiempo de ejecución de cada uno y es comparado para los distintos casos.

Las pruebas de se realizarán en las bajo las siguientes condiciones:

- Las pruebas son realizadas aplicando una función de tiempo en milisegundo la cual empieza a interactuar una vez se valide el período y soporte de los datos con que se trabajará.
- Se toma el tiempo de cada prueba 5 veces, de las cuales se obtendrá un promedio, este promedio será el representativo para tablas y gráficas.
- Las pruebas son realizadas en un equipo portátil con las siguientes características:
  - Procesador: Intel core i5 5200U, doble núcleo con velocidad de 2.2 GHz
  - Ram: 8Gb DDR3 de 1600 MHz
  - Disco duro: 500Gb 5400rpm
  - Sistema operativo: Windows 10 Home, 64bits
- Netbeans será la herramienta utilizada para la ejecución de los programas.

Uno de los problemas en la búsqueda de patrones periódicos es el procesamiento de datos, puesto que se desconoce si esta tiene o no un período definido. Para tales casos es posible realizar un proceso de búsqueda de patrones que permita examinar los datos utilizando diferentes períodos temporales.

Las versiones desarrolladas de Apriori, M-SP y Minus-F1 para este proyecto cuentan con soporte para múltiples períodos, es decir, estas versiones harán búsqueda de patrones en secuencias de datos con período desconocido, partiendo del período dos hasta un período máximo definido (menor que la mitad de la secuencia de datos), es decir, para una secuencia de datos con diez eventos buscará patrones desde período dos hasta período cinco, mientras que para secuencias con cien eventos la búsqueda de patrones partirá en el período dos y concluirá en el período indicado, menor o igual a cincuenta.

Cuando se trabaja con valores de soporte iguales o inferiores al 50% es posible que, para algunos períodos, el algoritmo detecte algunos segmentos de la secuencia de datos como patrones periódicos. La razón a este fenómeno es, por ejemplo, si una secuencia de datos es dividida solo en dos segmentos, la sola existencia de uno de los segmentos asegura su aparición en un 50% del total de la secuencia de datos.

La secuencia de datos seleccionada para el siguiente ejemplo no contiene ningún patrón. Cuando se intenta encontrar patrones de período 3 para esta muestra, esta dividirá la totalidad de sus datos en segmentos de periódicos de tamaño 3, es decir:

$$aAbBcCdDeEfFgGhHiIjJ \rightarrow \{aAbBcC - dDeEfF - gGhHiI - \not{j}\not{J}\}$$

Los segmentos son generados desde el inicio de la secuencia de datos. Los datos encontrados al final de la secuencia que no logren cumplir con el período seleccionado son descartados, pero solo para el período en estudio.

$$\{aAbBcC - dDeEfF - gGhHiI\}$$

Si secuencias de datos con estas características son estudiados bajo un soporte pequeño, por ejemplo 30%, contendrá segmentos que solo aparecen una vez, pero de igual forma serán considerados periódicos. En este ejemplo, el segmento "$aAbBcC$" tiene un 33% de aparición en la secuencia de datos, por tanto, bajo el requisito de soporte mínimo del 30% antes estipulado, este segmento representa a un patrón periódico, aparezca o no otra vez en la secuencia de datos. Para evitar situaciones como esta es necesario agregar una condición adicional, dada por la frecuencia de aparición de cada segmento del conjunto F_1, para todos los algoritmos, y para Fn también en el caso de Apriori.

2. Pruebas en Datos Sintéticos

Realizar pruebas sobre datos sintéticos permite la evaluación de casos con información conocida por el usuario, lo que permite establecer casos de pruebas de eventos controlados.

Utilizando secuencias de datos de diferente tamaño, se evaluó el funcionamiento de cada algoritmo implementado bajo las siguientes condiciones, donde cada conjunto de pruebas se realizó sobre un soporte mínimo en particular.

Los resultados de estas pruebas, expuestos en las tablas a continuación, son el tiempo de respuesta total de cada programa expresado en milisegundos para los diferentes casos evaluados aquí presentados:

Tamaño de secuencia de datos: 500, 750, 1000
Períodos evaluados: 2 a 10, 2 a 15, 2 a 25, 2 a 50, 2 a 100
Soporte mínimo evaluados: 25%, 50%, 75%

2.1. Secuencia de Datos sin Patrones.

- **Soporte mínimo: 25%**

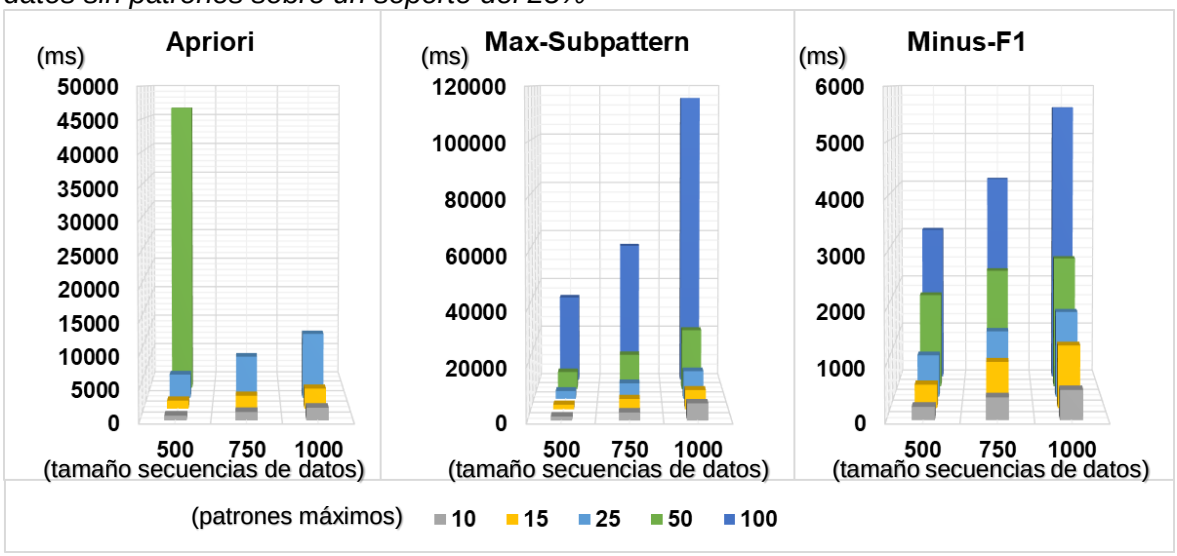
Tabla 14: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte mínimo del 25%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	721	1456	242	1275	2950	415	1915	6253	551
	15	1261	1742	441	2032	3915	871	3150	7263	1180
	25	3767	3059	810	6682	6043	1277	10357	10735	1653
	50	46381	6801	1836		13716	2326		23523	2571
	100		33927	3044		55066	4090		114979	5552

Fuente: Elaboración propia

Los resultados mostrados en Tabla 14 representan el tiempo de ejecución de cada programa, medido en milisegundos, para los diferentes casos planteados. Los resultados muestran que, ante la ausencia de patrones periódicos, M-SP y Minus-F1 son los únicos capaces de completar el proceso para todos los casos de prueba, siendo M-SP quien más tiempo demora en completar el procesado de datos. El proceso que realiza Apriori en la búsqueda de patrones de hasta período 50 o 100 es interrumpido a 30 minutos de su ejecución por desborde de la memoria del computador donde este se ejecuta. La Figura 24 expone de forma gráfica lo presentado por la Tabla 14, mientras que en Figura 25 se compara el desempeño de los tres programas bajo las mismas condiciones.

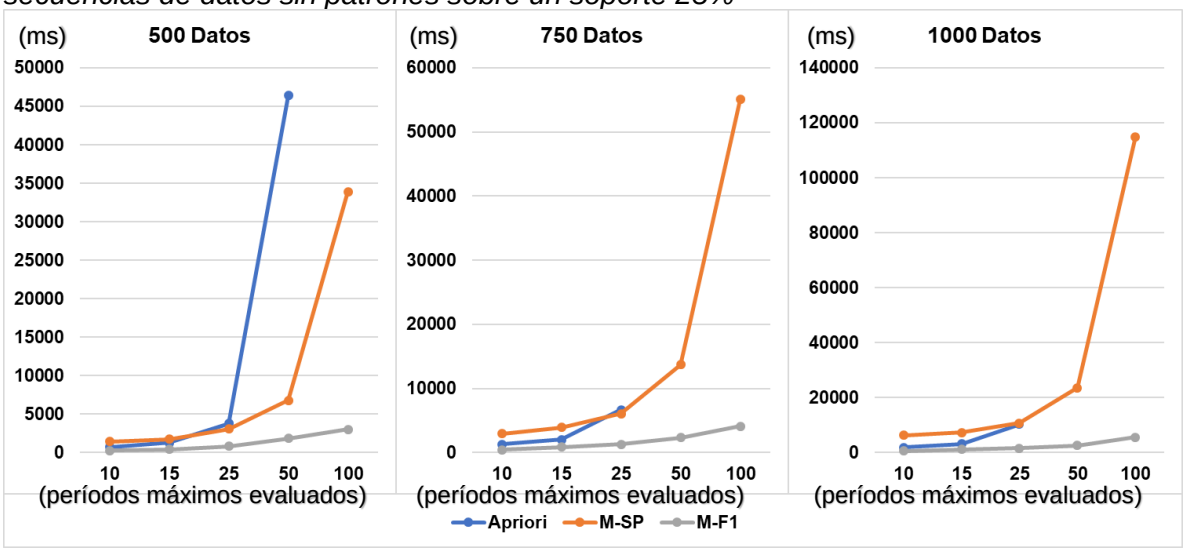
Figura 24: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte del 25%



Fuente: Elaboración propia

Como puede apreciarse en ambos cuadros gráficos, el algoritmo desarrollado para efectos de esta investigación, Minus-F1, es el que muestra un mejor desempeño en la búsqueda de patrones frente a una secuencia de datos que no contiene patrones periódicos.

Figura 25: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos sin patrones sobre un soporte 25%



Fuente: Elaboración propia

- **Soporte mínimo: 50%**

Los resultados obtenidos en las pruebas realizadas por los tres algoritmos sobre una secuencia de datos sin patrones periódicos, bajo un soporte mínimo del 50%, son encontrados en Tabla 15.

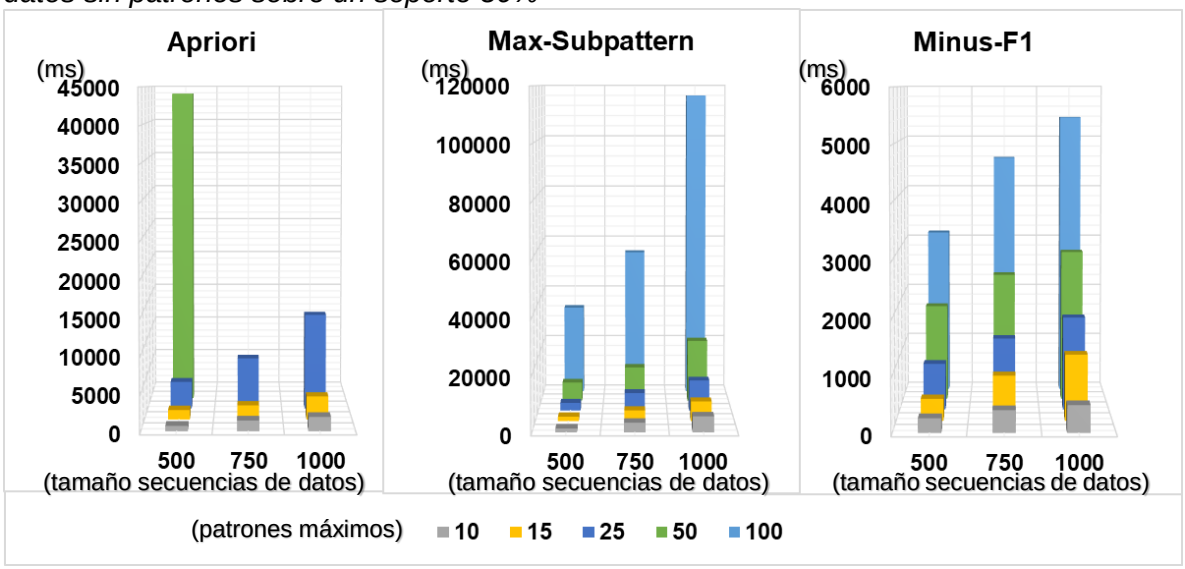
Tabla 15: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte mínimo del 50%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	714	1375	259	1422	3389	402	1899	5663	491
	15	1292	1604	399	1929	3875	821	3163	7130	1201
	25	3756	2850	857	7079	6527	1332	13214	11250	1723
	50	43966	6967	1795		12758	2401		22982	2840
	100		33072	3121		54881	4597		116034	5391

Fuente: Elaboración propia

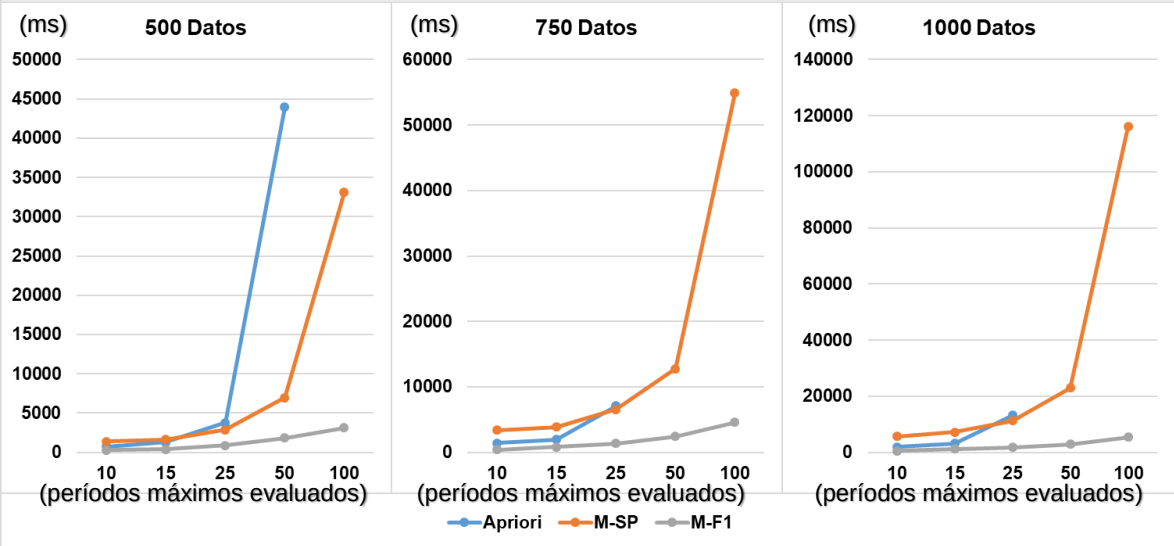
Como puede apreciarse en la Figura 26 y Figura 27, comparados con los resultados observados en la Figura 24 y Figura 25, los resultados obtenidos para una secuencia de datos sin patrones periódicos en da como resultado tiempos de ejecución similares para cada técnica, comparada con la misma técnica de la muestra anterior.

Figura 26: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte 50%



Fuente: Elaboración propia

Figura 27: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos sin patrones sobre un soporte 50%



Fuente: Elaboración propia

- **Soporte mínimo: 75%**

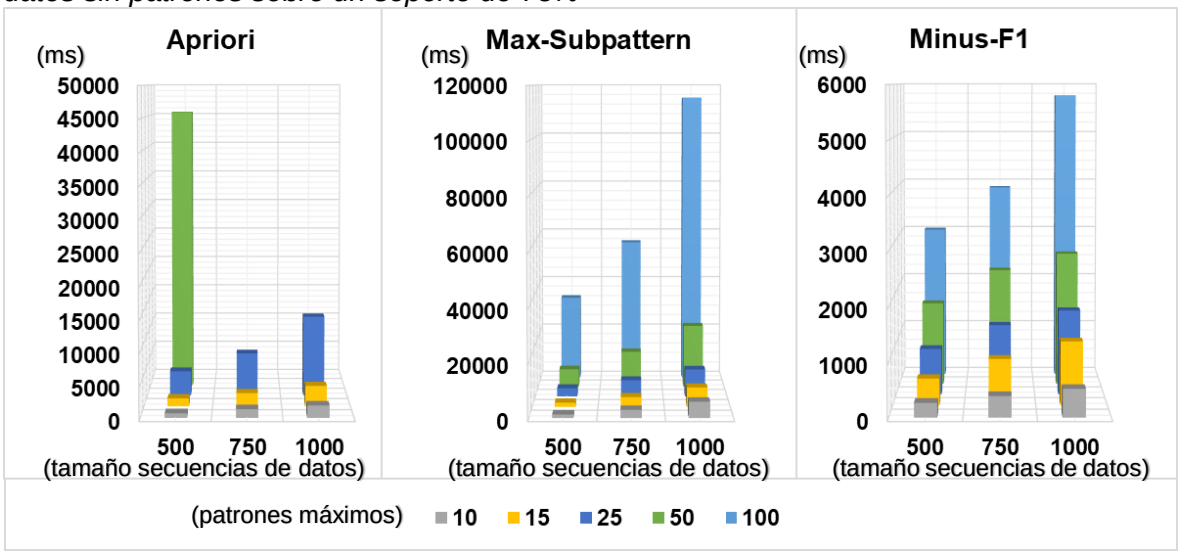
Los resultados obtenidos en las pruebas realizadas por los tres algoritmos sobre una secuencia de datos sin patrones periódicos, bajo un soporte mínimo del 75%, corroboran lo señalado en las dos muestras antes mostradas. Los resultados obtenidos son similares a los obtenidos por cada algoritmo bajo los soportes de 25% y 50%. En la Tabla 16, la Figura 28 y Figura 29 muestran el detalle de la ejecución de las pruebas realizadas.

Tabla 16: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte mínimo del 75%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	708	1321	271	1321	3006	397	1899	6012	528
	15	1251	1614	512	2070	3710	881	3224	7339	1213
	25	3931	3215	888	6856	6348	1350	12817	10521	1636
	50	45539	7002	1625		14023	2293		24385	2623
	100		32969	3005		55828	3875		114747	5762

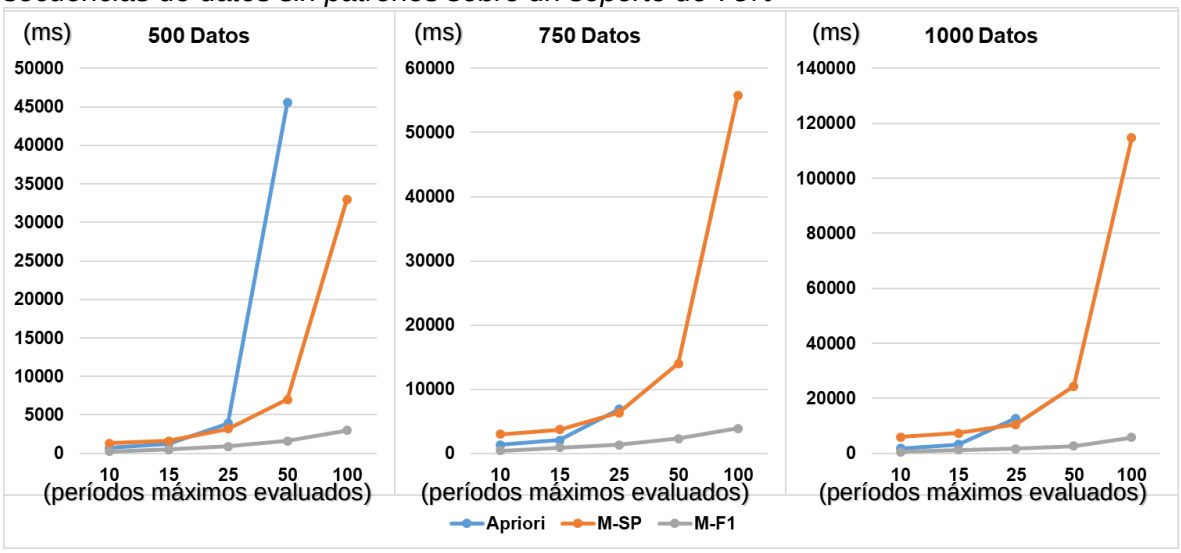
Fuente: Elaboración propias

Figura 28: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos sin patrones sobre un soporte de 75%



Fuente: Elaboración propia

Figura 29: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos sin patrones sobre un soporte de 75%



Fuente: Elaboración propia

Una secuencia de datos que no contiene patrones periódicos (ni completos y parciales) es indiferente a las exigencias de soporte mínimo que se establezca para buscar patrones sobre ella, ya que ninguno de sus segmentos cumplirá con dicho soporte para cualquier caso evaluado.

2.2. Secuencia de Datos con Patrones Periódicos

Los datos utilizados para la realización de las siguientes pruebas contienen un segmento periódico de tamaño 15, dispuesto en toda la extensión de la muestra. Este segmento inserto representa un patrón completo, visible en la evaluación de patrones de período 15. Es posible encontrar este segmento periódico y patrón completo si se evalúan períodos que sean múltiplos de 15, por ejemplo 30, 45, 60, 75 y 90

- **Soporte mínimo: 25%**

Los resultados mostrados en Tabla 17, obtenidos de las pruebas, muestran que Apriori tiene el peor tiempo de ejecución en la búsqueda de patrones en comparación con M-SP. Adicionalmente, no fue posible evaluar Apriori para todos los casos de prueba, puesto que generó desborde de memoria o tomó un tiempo para la prueba superior a 3 horas en dichos casos (sobre 10800000 ms).

La Figura 30 muestra que M-SP por su parte logra concluir con la ejecución del caso con tiempos que rondan los 50 segundo, mientras que Minus-F1 logra su objetivo en el caso con mayor cantidad de datos en un tiempo inferior a los 5 segundo. El alto tiempo que demora M-SP en ejecutarse en algunos casos responde a la cantidad de patrones que este encuentra para los diferentes períodos evaluados.

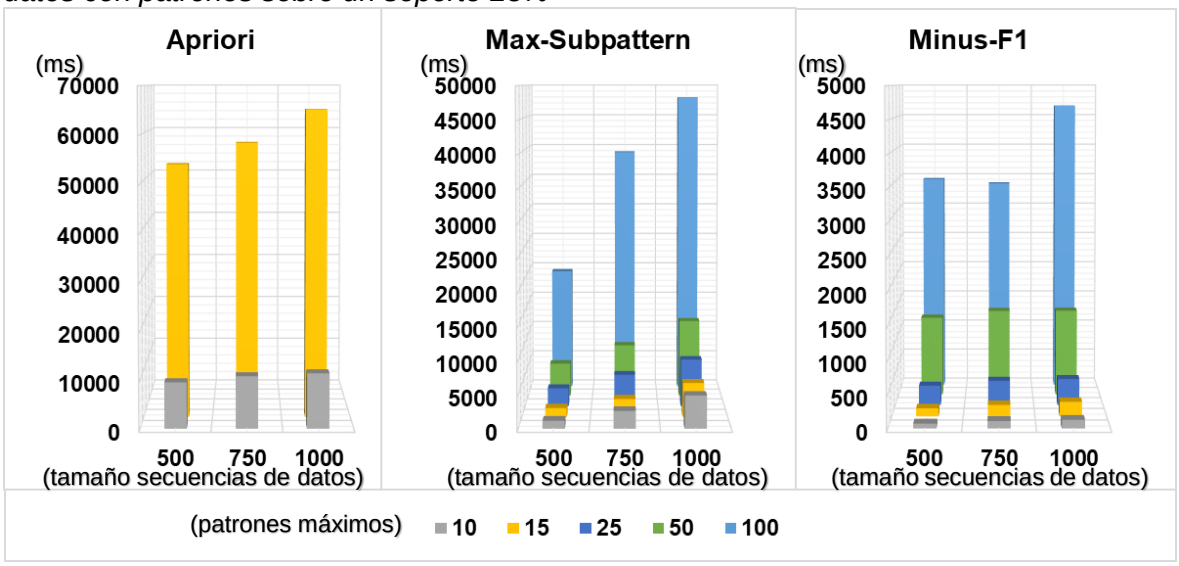
Tabla 17: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte mínimo del 25%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	9582	1180	73	10837	2638	111	11438	4913	129
	15	53589	1375	124	58067	2763	176	65032	5202	223
	25		2816	307		4940	385		7336	412
	50		5246	1253		8226	1364		12174	1368
	100		19241	3443		38983	3371		47977	4657

Fuente: Elaboración propia

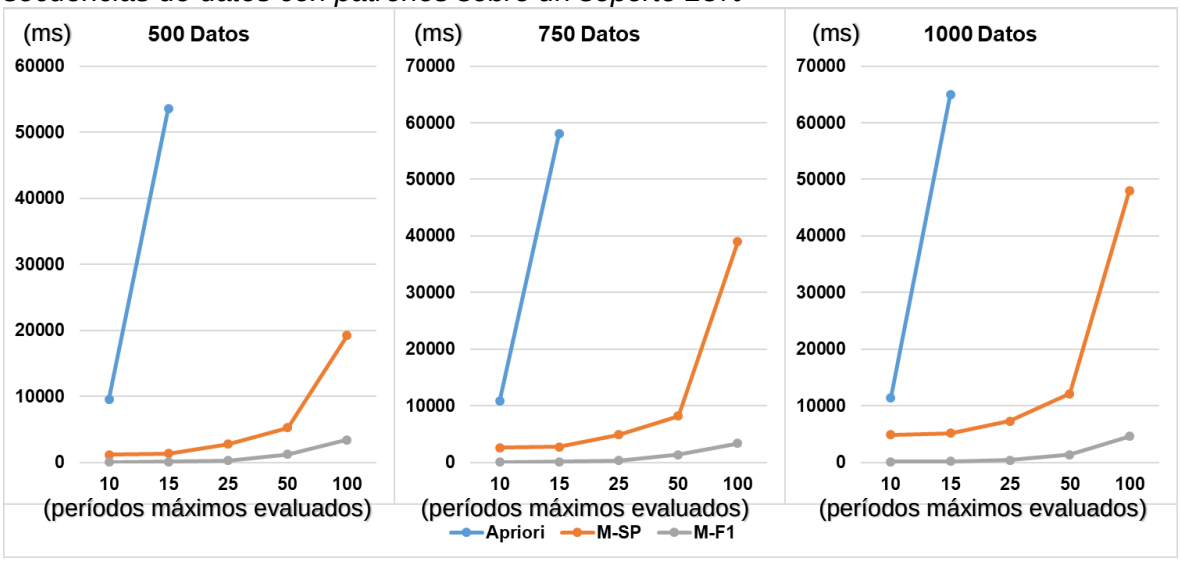
La Figura 31 expone la diferencia de tiempo de cada algoritmo para cada caso evaluado. Si bien en los casos con pocos datos el comportamiento de todos los algoritmos es similar, una vez aumenta el conjunto de datos a evaluar, aumenta la brecha de tiempo entre cada algoritmo.

Figura 30: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte 25%



Fuente: Elaboración propia

Figura 31: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos con patrones sobre un soporte 25%



Fuente: Elaboración propia

- **Soporte mínimo: 50%**

Tabla 18: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte mínimo del 50%

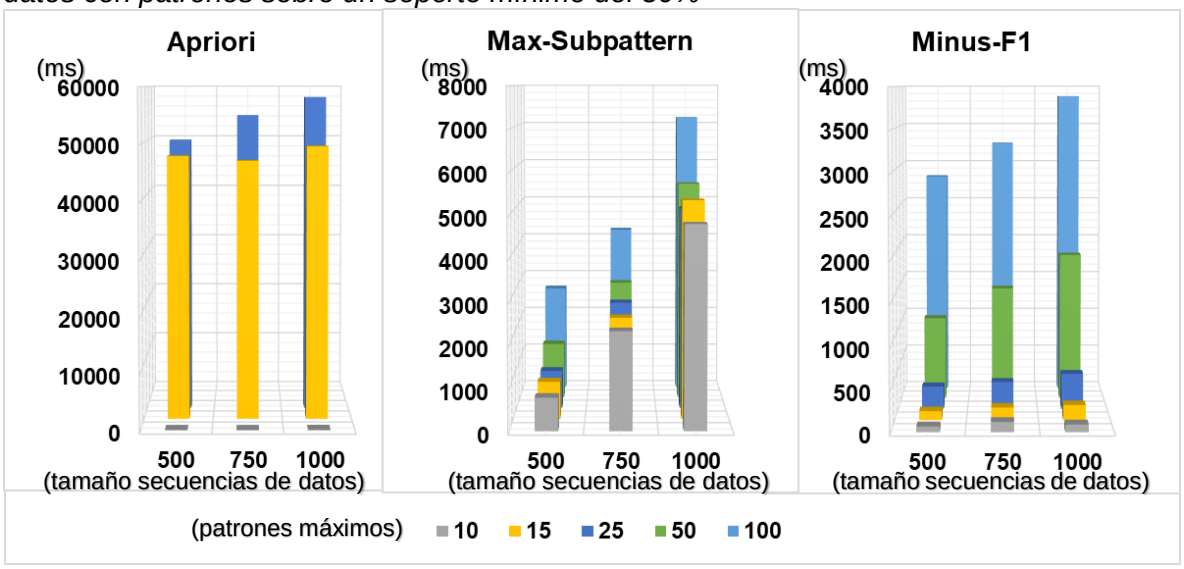
Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	141	789	68	140	2344	121	172	4831	91
	15	47566	905	109	46735	2456	152	49340	5278	183
	25	50048	922	278	54626	2633	335	58032	4987	438
	50		1377	1034		2959	1420		5500	1850
	100		2641	2815		4197	3251		7157	3868

Fuente: Elaboración propia

Los resultados obtenidos para este caso (Tabla 18) con un soporte mayor al anterior (50% en este caso) muestran que Apriori pudo ser evaluado al menos hasta encontrar patrones de tamaño 25. Esto responde a que, mientras mayor es el soporte exigido para la búsqueda de patrones, menor es la cantidad de patrones candidatos que genera y evalúa el algoritmo. Aún así Apriori no pudo completar las pruebas de búsqueda de patrones más grandes. M-SP redujo su tiempo de ejecución de forma considerable en este nuevo caso, logrando casi llegar a los tiempos de ejecución de Minus-F1. Este último tuvo tiempos de ejecución similares que el del caso evaluado con soporte 25%

La
Figura 32 y Figura 33 muestran los resultados por algoritmo y comparando los algoritmos en base al tamaño de la secuencia de datos respectivamente.

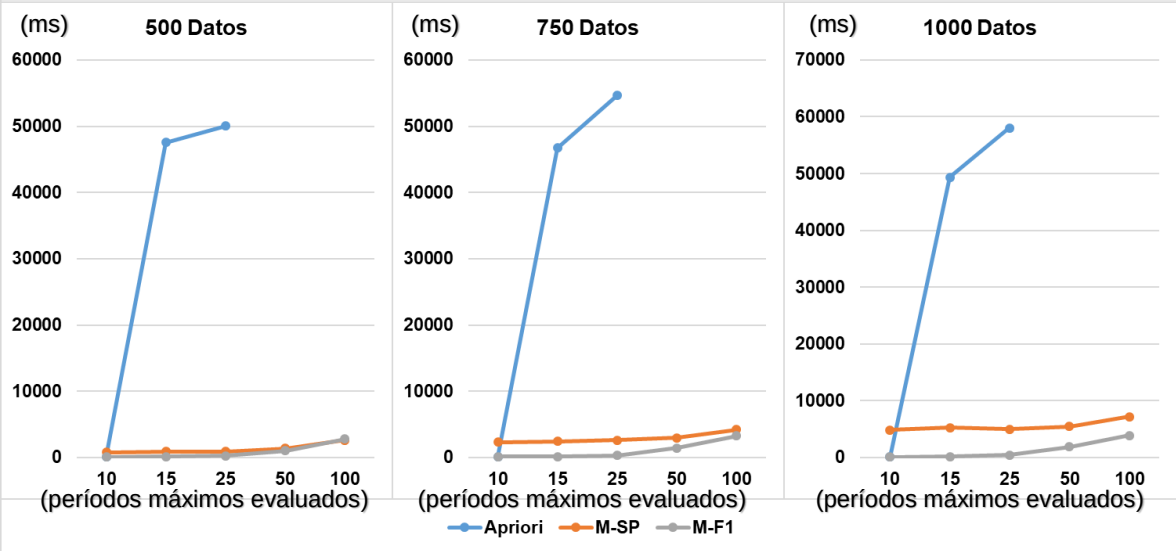
Figura 32: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte mínimo del 50%



Fuente: Elaboración propia

Gracias a la Figura 33 es posible notar la diferencia existente entre Apriori con el resto de algoritmos. Esta diferencia ocurre a causa de la gran cantidad de candidatos que debe generar Apriori. Observe que cuando el período a evaluar es muy grande, Apriori no logra completar la tarea.

Figura 33: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos sin patrones sobre un soporte 50%



Fuente: Elaboración propia

- **Soporte mínimo: 75%**

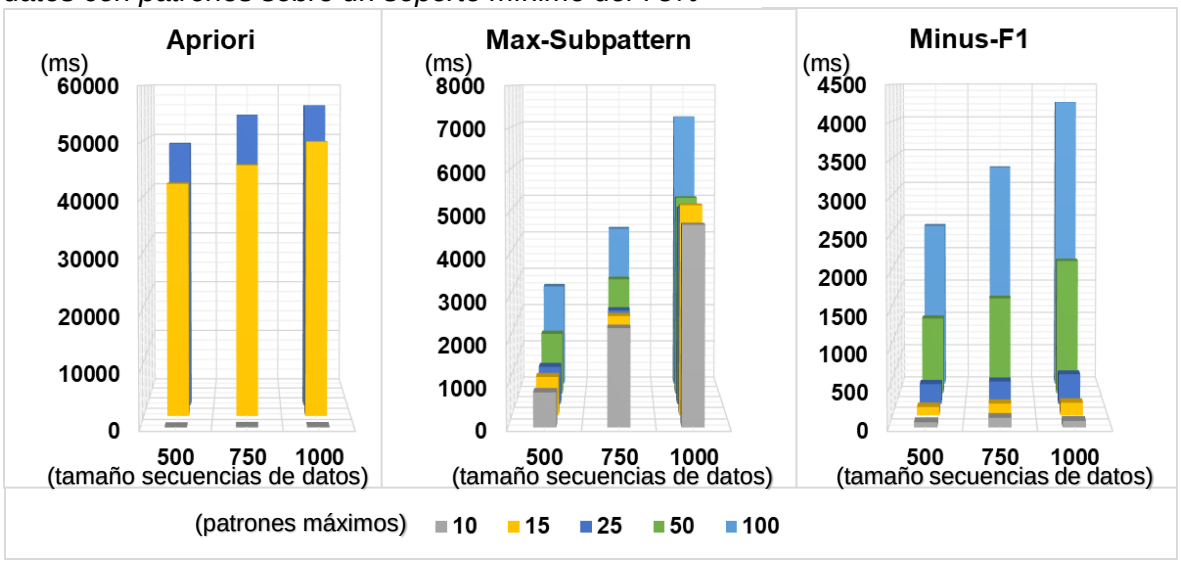
Tabla 19: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte mínimo del 75%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	125	833	71	203	2351	129	172	4774	89
	15	42322	944	115	45657	2434	161	49907	5122	174
	25	49173	947	285	54470	2376	321	56250	4941	423
	50		1562	1101		2987	1395		5100	1942
	100		2624	2385		4175	3259		7158	4230

Fuente: Elaboración propia

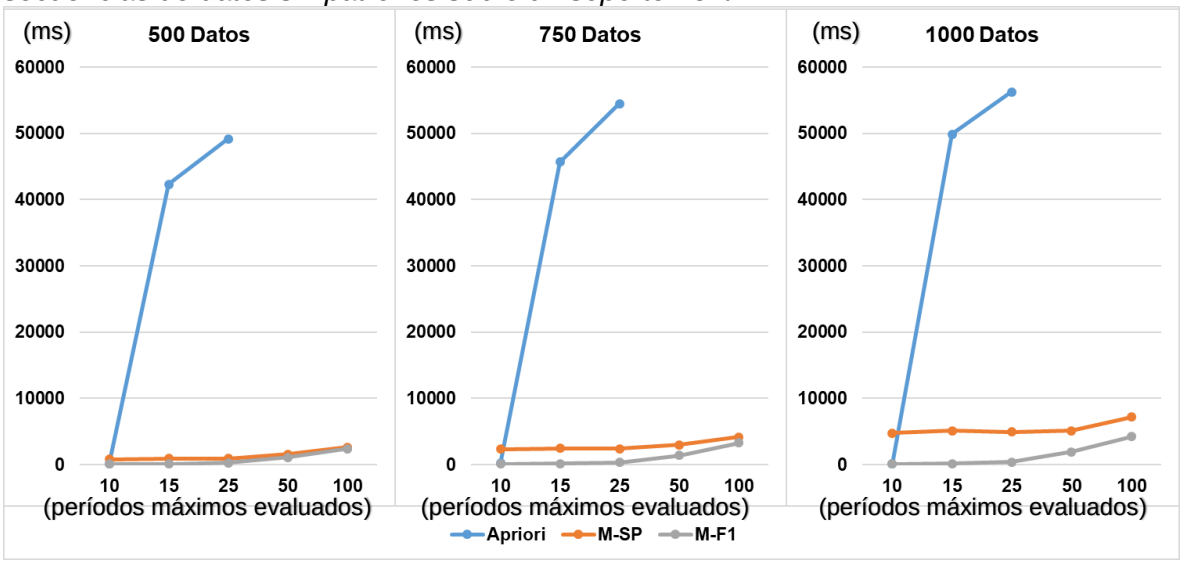
Es posible apreciar, tanto en la Tabla 19 como en la Figura 34 y Figura 35, que el comportamiento de estos algoritmos tiene variaciones mínimas respecto a su evaluación con soporte de 75% frente al soporte de 50%, esto debido a que sobre el 50% la cantidad de patrones a encontrar se ve notablemente reducida.

Figura 34: Tiempo de ejecución (ms) de cada algoritmo implementado en secuencias de datos con patrones sobre un soporte mínimo del 75%



Fuente de datos: Elaboración propia

Figura 35: Comparativo de tiempo de ejecución (ms) de algoritmos implementados en secuencias de datos sin patrones sobre un soporte 75%



Fuente: Elaboración propia

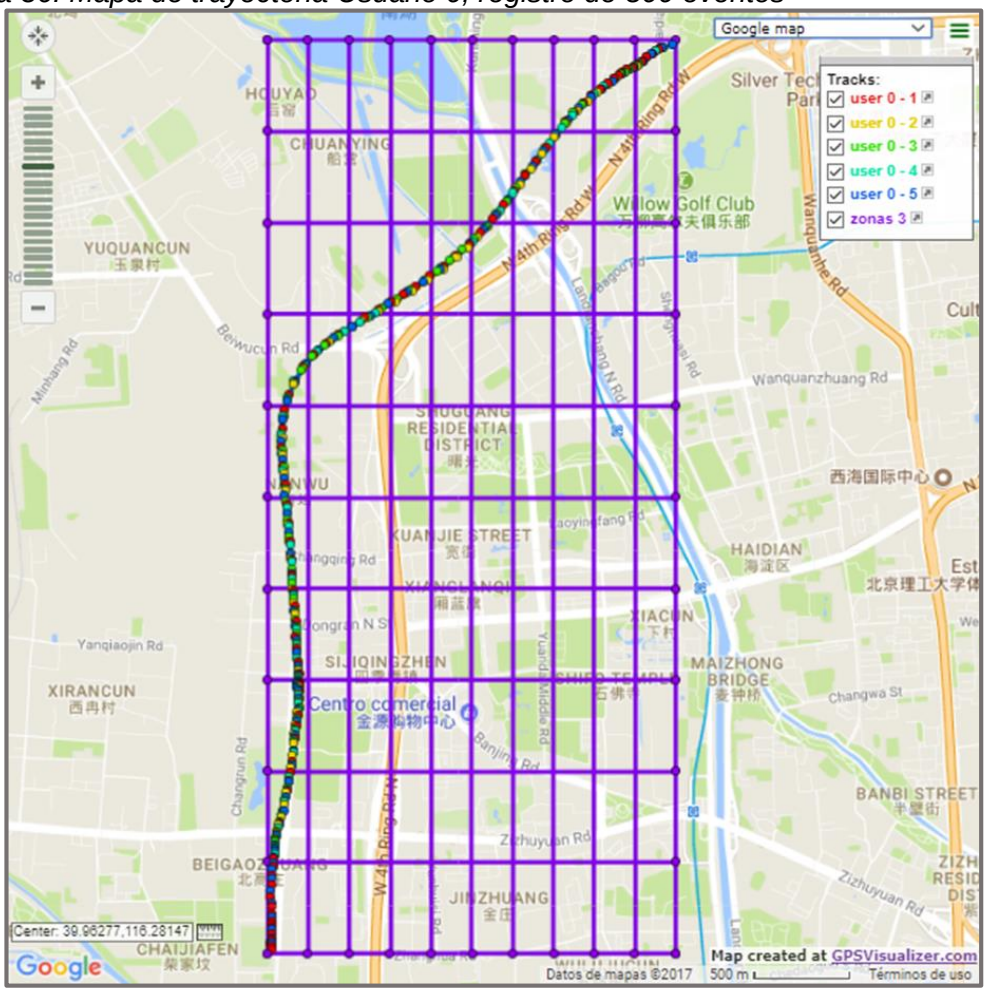
3. Pruebas en Datos Obtenidos de Geolife

El conjunto de datos utilizados para realización de las siguientes pruebas ha sido obtenido de una muestra disponible del dataset Geolife [L5], Los datos obtenidos fueron revisados previamente para asegurar que, al menos de manera visual, es posible que exista algún tipo de patrón periódico en los datos.

Los datos a consultar han sido extraídos de las muestras de 3 personas del proyecto Geolife y posteriormente procesados por la implementación en Java del algoritmo Busca-Zonas, para generar los conjuntos de datos que son capaces de procesar las implementaciones de Apriori, M-SP y Minus-F1.

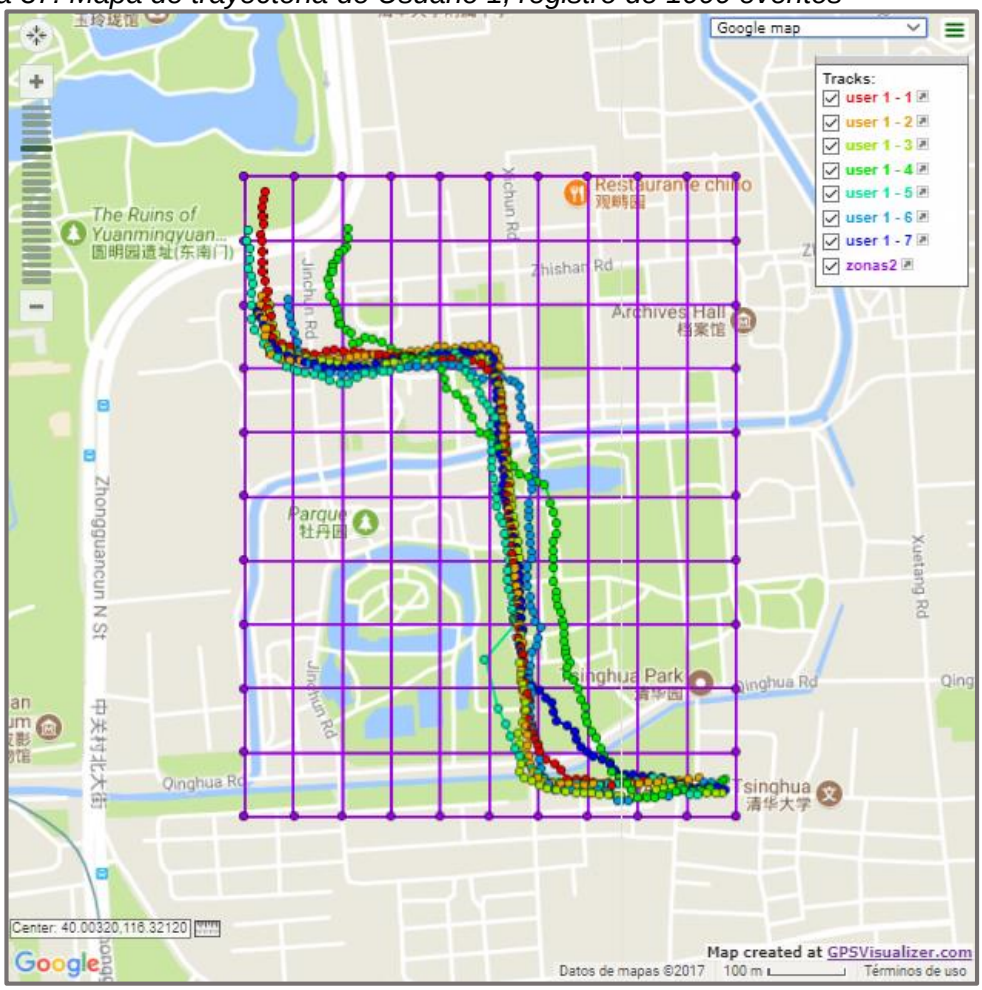
La secuencia de datos que contiene 500 eventos ha sido construida utilizando marcas temporales obtenidas de Usuario 0 (Figura 36); la secuencia de datos de datos con 750 eventos, utilizando marcas obtenidas de Usuario 2 (Figura 38) y para la secuencia de datos con 1000 eventos, marcas de Usuario 1 (Figura 37).

Figura 36: Mapa de trayectoria Usuario 0, registro de 500 eventos



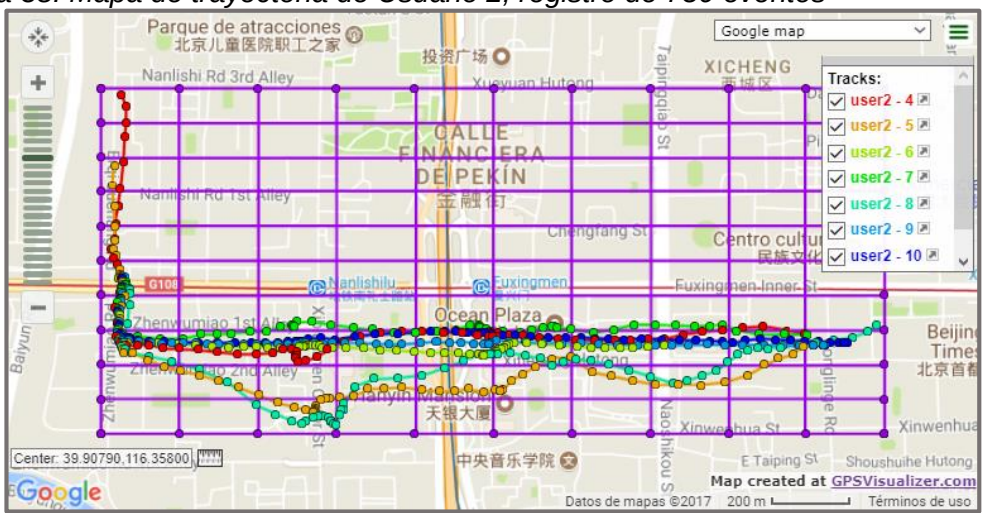
Fuente: Elaboración propia, utilizando secuencia de datos (archivos.plt) en aplicación GPSvisualizer [L6]

Figura 37: Mapa de trayectoria de Usuario 1, registro de 1000 eventos



Fuente: Elaboración propia, utilizando secuencia de datos (archivos.plt) en aplicación GPSvisualizer [L6]

Figura 38: Mapa de trayectoria de Usuario 2, registro de 750 eventos



Fuente: Elaboración propia, utilizando secuencia de datos (archivos.plt) en aplicación GPSvisualizer [L6]

- **Soporte mínimo: 25%**

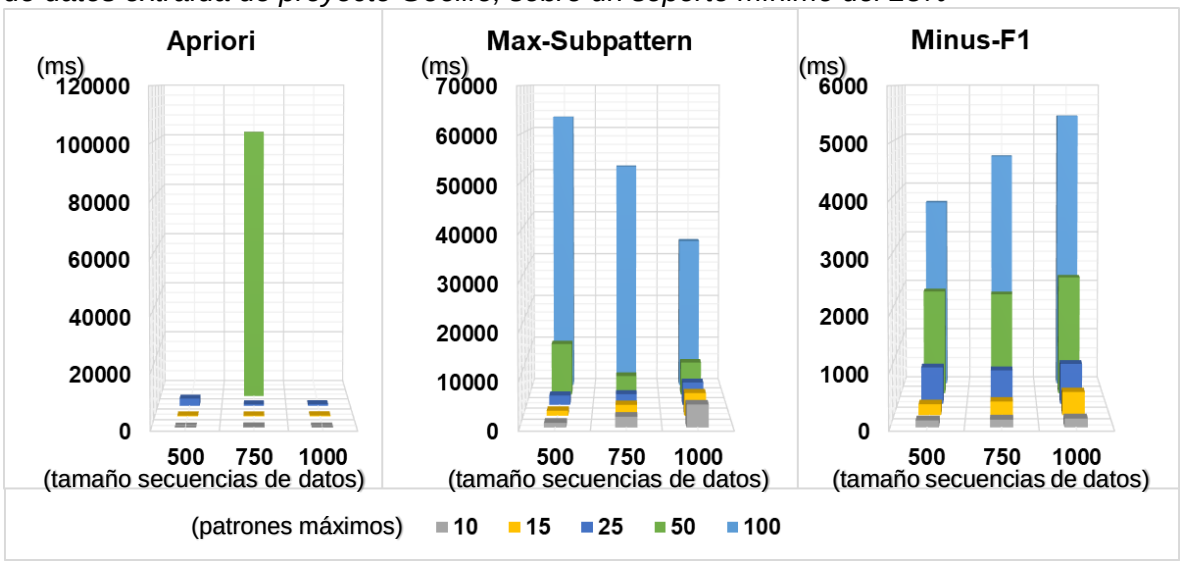
Apriori no es capaz de completar la búsqueda de patrones en las secuencias con 500 y 1000 eventos para los periodos mayores a 25, mientras que para la secuencia de datos que contiene 750 eventos, no puede completar la búsqueda en periodos mayores a 50, aunque el tiempo que le lleva realizar esta tarea es muy elevado. Los resultados pueden ser observados en Tabla 20, así como en la Figura 39 y Figura 40.

Tabla 20: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 25%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	193	968	123	267	2250	137	203	4797	157
	15	288	1094	202	298	2340	251	328	4870	428
	25	2761	2075	694	596	2358	638	626	4959	761
	50		11559	2001	102030	4249	1944		7325	2275
	100		62649	3659		51181	4582		33841	5387

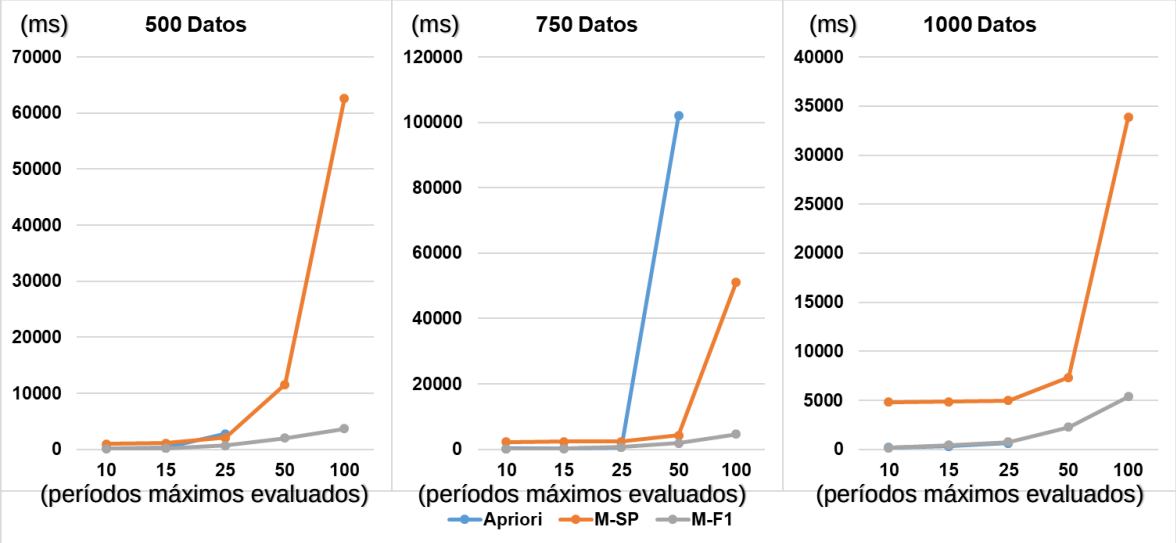
Fuente: Elaboración propia

Figura 39: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 25%



Fuente: Elaboración propia

Figura 40: Comparativo de tiempo de ejecución (ms) de algoritmos implementados, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte 25%



Fuente: Elaboración propia

Los tiempos de respuesta de M-SP nuevamente son mayores a los de Minus-F1, encontrando patrones periódicos para períodos mayores a 50 en las secuencias de datos con 500 a 1000 eventos.

- **Soporte mínimo: 50%**

Ante el aumento del soporte mínimo requerido para considerar a un patrón como interesante, disminuye la cantidad de patrones que los algoritmos detectan, esto permite a Apriori consultar por patrones periódicos con éxito en secuencias de datos con 500 a 1000 eventos hasta el periodo 50, y con la secuencia de datos con 750 eventos logra revisar la existencia de patrones hasta el periodo 100, aunque con tiempos de respuestas mayores al de los algoritmos MS-P y Minus-F_1, como se muestra en la Tabla 21.

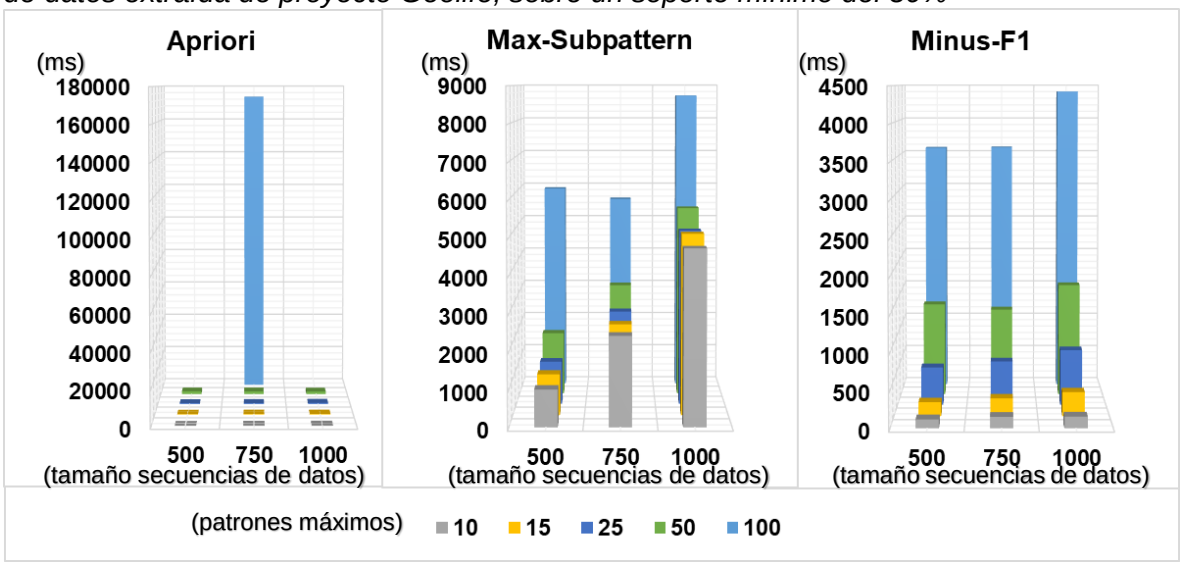
Tabla 21: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 50%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	172	1009	119	218	2437	152	266	4750	155
	15	288	1130	192	312	2506	244	344	4985	331
	25	453	1203	532	500	2633	616	616	4927	778
	50	1671	1772	1320	1612	3182	1242	1532	5438	1602
	100		5906	3568	173640	5592	3574		8688	4405

Fuente: Elaboración propia

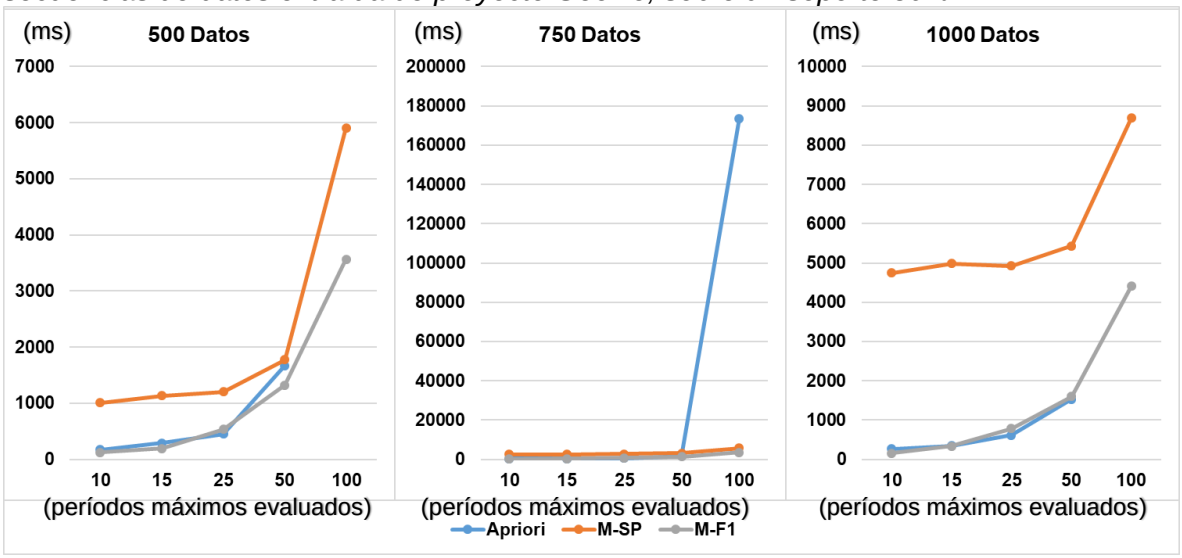
Los tiempos de ejecución máximos de M-SP disminuyeron, a causa de la menor cantidad de patrones encontrados bajo este soporte mínimo. Minus-F1 no mostró una diferencia significativa en sus tiempos de ejecución, los cuales se mantienen bajos. Los resultados en tiempo de ejecución pueden ser observados en Tabla 21, así como en la Figura 41 y Figura 42.

Figura 41: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 50%



Fuente: Elaboración propia

Figura 42: Comparativo de tiempo de ejecución (ms) de algoritmos implementados, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte 50%



Fuente: Elaboración propia

- **Soporte mínimo: 75%**

Bajo este valor de soporte los patrones interesantes detectados se reducen. Como se muestra en la Tabla 22, Apriori es capaz de revisar la existencia de patrones en todas las secuencias de datos y para todos los periodos evaluados. En este caso particular, si bien su tiempo de ejecución es alto, no es tan alto como en los casos con soporte del 25% y 50%.

Si bien Apriori tiene tiempos de ejecución bajos en la mayoría de casos respecto a M-SP, este último es más eficientes en los casos donde el período a evaluar es mayor, pudiendo buscar patrones en la secuencia con 1000 eventos en la mitad de tiempo que demora Apriori.

Los resultados de M-SP en este caso intentan acercarse en tiempo a los de Minus-F1, pero M-SP sigue demorando más.

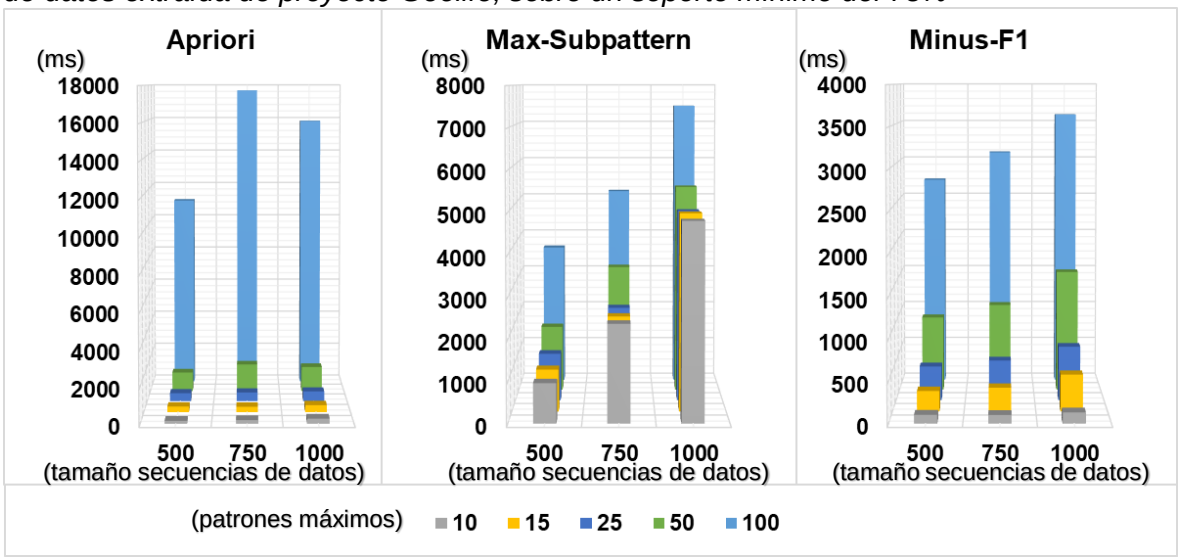
Tabla 22: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 75%

Cantidad de Eventos		500			750			1000		
Técnica		Apriori	M-SP	M-F1	Apriori	M-SP	M-F1	Apriori	M-SP	M-F1
Periodo Máximo	10	162	971	105	203	2387	103	282	4828	134
	15	310	1040	254	298	2353	294	391	4890	457
	25	465	1192	431	508	2369	508	569	4833	685
	50	1096	1666	953	1579	3243	1118	1428	5349	1553
	100	11037	3613	2716	17675	5141	3085	15827	7438	3590

Fuente: Elaboración propia

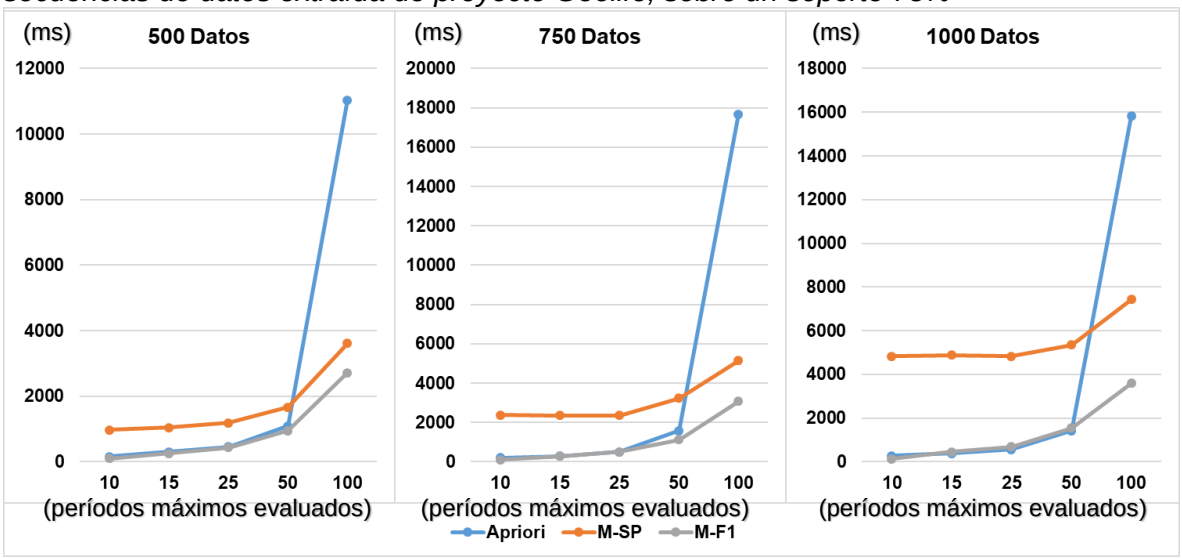
La Tabla 22, así como la Figura 43 y Figura 44 muestran el comportamiento de los tres algoritmos estudiados para este último caso de prueba. Nuevamente es posible apreciar que el algoritmo Minus-F1 sigue siendo el que menos tarda en la búsqueda y detección de patrones.

Figura 43: Tiempo de ejecución (ms) de cada algoritmo implementado, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte mínimo del 75%



Fuente: Elaboración propia

Figura 44: Comparativo de tiempo de ejecución (ms) de algoritmos implementados, usando secuencias de datos extraída de proyecto Geolife, sobre un soporte 75%



Fuente: Elaboración propia

4. Conclusión de las Pruebas

Los algoritmos Apriori, M-SP y Minus-F1, fueron implementados y comparados entre sí con la finalidad de determinar el más rápido en tiempo de ejecución al momento de realizar búsqueda de patrones sobre una secuencia de datos, dado un período y soporte determinado. En base a los resultados obtenidos por estos algoritmos en las pruebas realizas se concluye que los resultados arrojados por los distintos algoritmos varían de acuerdo a las condiciones establecidas y al tipo de secuencia de datos a ser procesada. Las pruebas se realizaron con dos conjuntos de secuencias de datos optimizadas para un resultado concreto: sin patrones y con patrones, y tres conjuntos de secuencias de datos extraídas del dataset Geolife.

Para todos los casos Minus-F1 demostró ser más eficiente en tiempo de ejecución que Apriori y M-SP, adicionalmente, los tiempos obtenidos por Minus-F1 presentaron variaciones mínimas respecto a los distintos tipos de datos evaluados (datos sin patrones, datos con patrones completos, secuencia de datos real).

Para los casos evaluados sobre las secuencias de datos que no contenían patrones, el algoritmo Apriori mostró tiempos de ejecución elevados para la mayoría de casos en que el periodo a evaluar fuese mayor a 15, por ejemplo, para una secuencia de datos con 500 elementos y con un rango de período de 2 hasta 15, los tiempos de respuestas fueron similares a los obtenidos por M-SP. Sin embargo, cuando se buscaron patrones hasta el periodo 100, Apriori no logró realizar completamente la tarea de búsqueda de patrones a causa de un desborde de memoria, mientras que M-SP logró completar la búsqueda.

En síntesis, los tres algoritmos se comportan de manera muy similar hasta los 500 datos y buscando patrones de tamaño 15 en la búsqueda de patrones en una secuencia de datos sin patrones y bajo todos los soportes mínimos evaluados. Si se requiere buscar en regiones con una mayor cantidad de datos, el algoritmo M-SP logra obtener los resultados esperados a expensas de un incremento alto en sus tiempos de ejecución.

Para el caso de análisis sobre patrones periódicos, Apriori es quien tiene el peor desempeño de los tres, sobre todo si el patrón a buscar es encontrado en un período grande y es un patrón completo, a causa de la cantidad de patrones candidatos que este genera bajo un valor de soporte pequeño, como 25 o 50%. Por el contrario, “Max Subpattern” entregas mejores resultados y de forma más rápida, ya que no presenta problema alguno en encontrar patrones periódicos, aunque sus tiempos de ejecución son más altos comparados con los de la ejecución de Minus-F1

CAPÍTULO 5:
CONCLUSIONES

Respecto a los objetivos planteados en este proyecto:

Estudiar e implementar dos algoritmos conocidos en el contexto de minería de datos, Apriori y Max Subpattern, para la detección de patrones periódicos en secuencias de datos espacio-temporales.

Como se muestra en el Capítulo 2, Sección 2.4.2. y 2.4.3., son presentados en base a sus definiciones originales los algoritmos Apriori y Max-Subpattern, mientras que en el Capítulo 3, Sección 2.1. y 2.2. son especificados sus pseudocódigos, útiles en la implementación de estos para la búsqueda de patrones sobre secuencias de datos espacio-temporales.

Analizar, diseñar e implementar un nuevo algoritmo de minería de datos, para la detección de patrones periódicos.

Como se muestra en el Capítulo 3, Sección 2.3. este objetivo se llevó a cabo de manera exitosa. Se realizó la implementación del Minus-F_1 y posteriormente fue testeado, como se muestra en el Capítulo 4.

Generar un conjunto de datos sintéticos con el objeto de generar un banco de pruebas que permita verificar la correctitud del algoritmo propuesto, así como simular diferentes escenarios experimentales.

Este objetivo es comprobable en el Capítulo 4, Sección 2, en el cual se exponen los resultados de los algoritmos en base al tiempo que demora en ejecutarse sobre secuencias de datos sintéticas, las cuales representas escenarios experimentales controlados.

Buscar una Base de Datos espacio-temporal, en la cual puedan aplicar los algoritmos de búsqueda de patrones periódicos.

Como se indica en el Capítulo 3, Sección 3.1. Los valores con los que trabajamos este proyecto de título y que fueron utilizados por los algoritmos desarrollados, resultaron ser datos espacio temporales reales, mientras que la utilización de estos conjuntos de datos es señalada además en el Capítulo 4, Sección 3.

Tanto los valores de latitud, longitud y tiempo, fueron extraídos desde la trayectoria GPS de un dataset. Este dataset en particular, es producto de un proyecto de Investigación en Asia de la compañía Microsoft conocido como “GeoLife”. Este proyecto recopiló la información de 182 usuarios en un período de alrededor de 5 años (Desde abril del 2007 hasta el mes de agosto del 2012).

Realizar un análisis empírico del comportamiento (eficiencia en tiempo del algoritmo y exactitud) de los algoritmos implementados, considerando diferentes escenarios experimentales, usando tanto una muestra de la Base de Datos real, como los datos sintéticos.

En toda la extensión del Capítulo 4 de este informe se presentó el análisis del comportamiento de los algoritmos implementados en base a la eficiencia en tiempo. Como se muestra en los resultados del Capítulo 4, Sección 2, Se utilizaron conjuntos de datos sintéticos para probar cada algoritmo en casos conocidos, por ejemplo, la presencia o ausencia total de patrones en una secuencia de datos. De la mismo forma, en el Capítulo 4, Sección 3, se presentan los resultados de la evaluación de conjuntos extraídos de Geolife, donde se mostró el resultado de cada algoritmo sobre esas secuencias.

Analizar la complejidad temporal de los algoritmos.

En el Capítulo 3, Sección 4. se muestra la complejidad temporal de cada algoritmo en base a su peor caso. Una vez estudiada la implementación realizada de cada algoritmo se llegó a la conclusión que los tres algoritmos tienen comportamiento similar, relacionado a la búsqueda inicial en la secuencia de datos del conjunto de 1-patrones F_1, donde su orden es $O(n)$.

Apriori, si bien sencillo de implementar, es un algoritmo costoso pues evaluado el orden de este se determinó que en el peor de los casos es superior a $O(n * m^{n/2})$, con un crecimiento catastrófico cuando la secuencia de datos de tamaño n a evaluar es muy grande. M-SP no se queda atrás pues su orden también puede llegar a crecer de manera desbordada: $O\left(\sqrt[4]{\left(\frac{m*n}{2}\right)^{n^2}} * n\right)$. El algoritmo desarrollado para este proyecto, Minus-F1, demostró un comportamiento moderado donde, en uno de sus peores casos presenta un comportamiento prácticamente lineal: $O(n)$.

BIBLIOGRAFÍA

[1] M. Guasch y M. Piergallini, «Bases de Datos Espacio-Temporales aplicadas al análisis y seguimiento de focos epidémicos,» de *XX Congreso Argentino de Ciencias de la Computación*, Buenos Aires, 2014.

[2] R. Simoes, G. Ribeiro, K. Reis, L. Vinhas y G. Camara, «PostGIS-T: towards a spatiotemporal PostgreSQL database extension,» de *GeoInfo*, Campos do Jordão, Sao Paulo, 2016.

[3] J. Peiquan y S. Pengfei, «OSTM: A Spatiotemporal Extension to Oracle,» *Fourth International Conference on Networked Computing and Advanced Information Management,* vol. 2, nº 1, pp. 575- 580, 2008.

[4] G. Gutierrez, «Métodos de Acceso y Procesamiento de Consulta Espacio-Temporales,» *Tesis de Doctorado en Ciencias de la Computación, Universidad de Chile, Santiago, Chile,* 2007.

[5] Y. Zheng, «Trajectory Data Mining: An Overview,» *ACM Trans. On Intelligent Systems and Technology,* vol. 6, nº 3, Mayo 2015.

[6] J. Han, M. Kamber y J. Pei, «Chapter 1 Introduction,» de *Data Mining Concepts and Techniques 3rd Edition*, Elsevier Inc, 2012, pp. 1-35.

[7] S. Vallejos, «Minería de Datos,» *Proyecto de Adscripción, Universidad Nacional del Nordeste, Corrientes, Argentina,* 2006.

[8] S. Shashi, J. Zhe, Y. A. Reem, E. Emre, T. Xun, M. G. Venkata y Z. Xun, «Spatiotemporal Data Mining: A computational Perspective,» *ISPRS International Journal of Geo-Information,* vol. 4, pp. 2306-2338, 2015.

[9] M. Perez, «Introducción,» de *Minería De Datos A Través De Ejemplos*, Madrid, RC Libros, 2014.

[10] W. Frawley, G. Piatetsky-Shapiro y C. Matheus, «Knowledge Discovery in Databases: An Overview,» *IA Magazine,* vol. 13, nº 3, pp. 57-70, 1992.

[11] R. Herrera, «Bibliomining: minería de datos y descubrimiento de conocimiento en bases de datos aplicados al ámbito bibliotecario,» *Articulo en prensa, Universidad Carlos III de Madrid,* p. 29, 2006.

[12] Y. Xiaobai, «Research Issues in Spatio-temporal Data Mining,» de *University Consortium for Geographic Information Science workshop on Geospatial Visualization and Knowledge Discovery*, Lansdowne, Virginia, 2003.

[13] J. L. B. Roddick, «Paradigms for Spatial and Spatio-Temporal Data Mining,» 2001.

[14] R. Agrawal, T. Imielinski y A. Swami, «Mining Association Rules between Sets if Items in Large Databases,» *SIGMOD '93 Proceedings of the 1993 ACM SIGMOD international conference on Management of data,* vol. 22, nº 2, pp. 207-216, 1993.

[15] G. Sirisha, M. Shashi y G. P. Raju, «Periodic Pattern Mining Algorithms and Applications,» *Global Journal of Computer Science and Technology Software & Data Engineering,* vol. 13, nº 13-C, 2013.

[16] R. Agrawal y R. Stikant, «Fast Algorithms for Mining Association Rules,» *Proceedings of the 20th VLDB Conference,* pp. 487-499, 1994.

[17] H. Jiawei, D. Guozhu y Y. Yiwen, «Efficient Mining of Partial Periodic Patterns in Time Series Database,» de *15th International Conference on Data Engineering*, Sydney, NSW, Australia, 1999.

[18] H. Wang, H. Su, K. Zheng, S. Sadiq y X. Zhou, «An Effectiveness Study on Trajectory Similitarity Measures,» *Proceedings of the Twenty-Fourth Australasian Database Conference,* vol. 137, pp. 12-22, 2013.

[19] F. Giannotti, M. Nanni, D. Pedreschi y F. Pinelli, «Trajectory Pattern Mining,» *KDD '07 Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 330-339, 2007.

LINKOGRAFÍA

[L1] "Modelos Predictivos y Descriptivos en Minería de Datos" [ppt], url. <https://es.slideshare.net/lalopg/mtodos-predictivos-y-descriptivos-minera-de-datos». 2015, consulta: febrero 2017

[L2] "Definición de Patrón (Pattern)" [diccionario web], url. <http://www.businessdictionary.com/definition/pattern.html>. consulta: julio 2017

[L3] Iribarra, F., "Descubrimiento del Conocimiento (KDD)" [blog], url. <http://mineriadatos1.blogspot.cl/2013/06/descubrimiento-del-conocimiento-kdd-el.html>, 2013. consulta: abril 2017

[L4] Berzal, F., "Reglas de Asociación" [presentación], url. <http://elvex.ugr.es/decsai/intelligent/slides/dm/D2%20Association.pdf>, 2013. consulta. diciembre 2016.

[L5] Zheng, Y. "GeoLife: Building Social Networks Using Human Location History" [articulo], url. <https://www.microsoft.com/en-us/research/project/geolife-building-social-networks-using-human-location-history>, 2009. consulta. noviembre 2016.

[L6] "GPS Visualizer" [aplicación online], url. <http://www.gpsvisualizer.com>, consulta. enero 2017

[L7] "Reglas de Asociación" [enciclopedia web], url. <https://es.wikipedia.org/wiki/Reglas_de_asociacion>, consulta. Mayo 2017.

[L8] Elfeky, M., "Mining Time-Series Databases" [ppt], url activa, link: <http://slideplayer.com/slide/4973499>, consulta. Noviembre 2016.