Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Un nuevo algoritmo eficiente para la búsqueda de patrones parciales frecuentes

Proyecto de título presentado por Marcelo Alejandro Stöckle Márquez
de la Carrera Ingeniería Civil Informática
Dirigida por Claudio Gutiérrez-Soto

2021

Resumen

Palabras Clave — Trayectorias, min-hashing, patrones parciales frecuentes, data mining, locality-sensitive hashing.

Índice general

Índice de figuras

Capítulo 1

Introducción

Un patrón periódico es una serie específica de símbolos que aparece repetidas veces a lo largo de una serie de tiempo. Los patrones periódicos emergen de manera natural dentro de los datos secuenciales: rutinas, errores sistemáticos, repeticiones. La repetición embebida en nuestro propio genoma no aparece para introducir redundancia, sino para cumplir una función auxiliar.

La búsqueda de patrones periódicos frecuentes (BPPF) es una forma de minería de datos cuyo propósito es descubrir los patrones periódicos más prevalentes en una base de datos de series de tiempo.

De interés para la BPPF son dos formas de patrones periódicos: completos y parciales. Para clasificar un patrón periódico como completo, cada elemento del patrón debe exhibir periodicidad. Un patrón periódico parcial (PPP) es una definición menos estricta donde solo ciertos elementos del patrón exhiben periodicidad, y al resto se les puede denominar "comodines" (Gadiraju, 2013). Este estudio tratará exclusivamente de la BPPF de patrones parciales, o BPPPF.

El problema de descubrir PPPs en una base de datos de series de tiempo es altamente complejo, puesto que el número de patrones por probar aumenta combinatorialmente con respecto al número de ítems (distintos valores que la serie de tiempo puede tomar en cada posición) y al largo, o periodo, del patrón. Soluciones convencionales involucran de una forma u otra al algoritmo Apriori para reducir dramáticamente el espacio de búsqueda.

En su implementación más ingenua, el algoritmo Apriori requiere en el peor caso hacer tantas lecturas a la base de datos como el largo del patrón de interés. Debido a esto, esta implementación no es escalable a la búsqueda de patrones largos. Soluciones más modernas como *FP-growth* o *Max-subpattern*, sin embargo, emplean estructuras que optimizan la búsqueda y requieren a lo más dos lecturas de la base de datos.

Un aspecto común a estas soluciones es que todas determinan el soporte y/o la confianza de un PPP para determinar si este efectivamente es frecuente. Determinar ambos indicadores requiere una o múltiples búsquedas exhaustivas de la base de datos. Sin embargo, se puede argumentar que en la tarea de descubrir patrones interesantes existen instancias en que determinar con exactitud el soporte y la confianza no es más útil que reportar, por ejemplo, un ranking o lista ordenada de los patrones más frecuentes. Los valores exactos de soporte y confianza son útiles para determinar esta jerarquía, pero al fin y al cabo, asumiendo garantías mínimas, para un

experto la información más relevante son los patrones mismos.

La razón por la que este argumento es siquiera una consideración es que el uso de técnicas aproximadas de minería de datos abre las puertas a nuevas implementaciones más eficientes con estructuras de datos escalables.

Con esta posibilidad en mente, en este estudio se propuso explorar métodos aproximados para la BPPPF. Con pocos antecedentes, se tuvo que conducir una revisión de la literatura acerca de métodos aproximados de búsqueda e indexación en bases de datos de series de tiempo.

Un método prometedor para estos propósitos es una variante de *locality-sensitive hashing* (LSH) denominada *min-hashing* (Cohen et al., 2001). A grandes rasgos, LSH es una familia de algoritmos para la búsqueda aproximada que emplean funciones de hash con la conveniente propiedad de que la colisión entre datos en una tabla hash es más probable entre datos similares que entre datos diferentes. La métrica más común de similitud es la distancia euclideana o el largo de la línea entre dos puntos en un espacio. Existe una definición clara de la distancia euclideana entre series de tiempo, y aún otras definiciones más elaboradas que dan cuenta de la maleabilidad temporal de las series de tiempo (*dynamic time warping*), y aunque estas métricas sirven para determinar la similitud entre series de tiempo con datos en un espacio lineal, no así en el tipo de series de tiempo usualmente involucradas en la BPPPF. Estas últimas consisten usualmente de una secuencia ordenada de *ítems* denotados por un identificador, sin garantías de ordinalidad. Una métrica mucho más conveniente para este caso viene dada por el coeficiente de Jaccard. Con *min-hashing* es posible comparar segmentos de series de tiempo y descubrir segmentos similares, y esto ha sido empleado exitosamente en la tarea de determinar infracciones al derecho de autor en bases de datos multimedia (tarea popularmente conocida como *Content ID*) (Chiu et al., 2010).

La ventaja más importante que se le atribuye a los métodos de búsqueda aproximada es que estos no requieren de estructuras de datos "volatiles", es decir, estructuras cuya dimensionalidad es muy sensible al tamaño del espacio de búsqueda. Otros algoritmos como FP-growth y Max-subpattern presentan este problema donde el volumen de la estructura de árbol empleada para indexar una base de datos crece a una velocidad insostenible cuando el número de símbolos supera los cientos o miles.

Teniendo en cuenta las posibles aplicaciones de *min-hashing* y LSH, este estudio propone adaptar esta técnica para resolver la BPPPF.

1.1. Objetivo General

Implementar una solución aproximada para la búsqueda de patrones periódicos y comparar su rendimiento contra técnicas del estado del arte.

1.2. Objetivos específicos

- Identificar algoritmos de búsqueda aproximada en series de tiempo del estado del arte.
- Diseñar e implementar una solución aproximada para la búsqueda de patrones parciales periódicos frecuentes (BPPPF).

- Obtener resultados empíricos utilizando una base de datos sintética y una base de datos real.
- Compilar y presentar los resultados en la forma de un artículo científico.

1.3. Fundamento/justificacion del proyecto

Las implementación modernas de la búsqueda de patrones parciales periódicos frecuentes (BPPPF) se ven limitada por su adherencia al algoritmo Apriori y el requerimiento de determinar con exactitud el soporte y confianza para determinar si un PPP es efectivamente frecuente, como también por las limitantes en el rendimiento de las operaciones de indexación y búsqueda. Estas limitaciones restringen el rango posible de aplicaciones, como por ejemplo cualquier aplicación que requiera actualizaciones frecuentes del modelo de datos, conocidas comúnmente como aplicaciones *online*.

Este estudio propone explorar un método de búsqueda aproximada capaz de descubrir PPP candidatos con suficientes garantías (no falsos positivos, pocos falsos negativos). Logrando esto, se espera que la solución implementada supere significativamente en rendimiento a los algoritmos exactos de BPPPF.

1.4. Detalle del informe

La siguiente sección ofrece de forma resumida el trasfondo y antecedentes necesarios para la comprensión del resto de este documento (capítulo 2). Luego, se ofrece una breve reseña de las técnicas de búsqueda aproximada del estado del arte que aparecen reportadas en la literatura académica reciente (capítulo 3). El siguiente capítulo describe en detalle los métodos y experimentos desarrollados durante este estudio (capítulo 4), y a continuación se presenta un resumen de los resultados observados (capítulo 5). Finalmente, este documento concluye con la discusión e interpretación de los resultados observados, como también de las implicaciones de los métodos desarrollados en este estudio (capítulo 6).

Capítulo 2

Marco Teórico

2.1. Patrones periódicos completos y parciales

Los patrones periódicos aparecen naturalmente en las series de tiempo o secuencias ordenadas, donde uno o múltiples ítems aparecen en el mismo orden repetidas veces a lo largo de una secuencia. Se denomina **periodo** (p) al largo de esta sub-secuencia.

Soporte es la proporción de la cantidad de veces en que aparece un patrón contra el total de periodos a lo largo de la serie de tiempo (número que aparecerá como l/p).

$$Soporte(P) = \frac{NumInstancias}{l/p}$$

A modo de ejemplo, en la secuencia {ABCBCBCA}, {BC} es un patrón periódico de periodo 2, con soporte igual a 3/4. Se dice que un patrón es **frecuente** si cumple con un requerimiento mínimo de soporte.

Un patrón periódico parcial (PPP) es una definición menos estricta donde solo ciertos elementos del patrón exhiben periodicidad, y al resto se les puede denominar "comodines". Considere la secuencia {AABAACBCBAAAAAB}: {AA*} es un patrón parcial con soporte igual a 4/5. Con dos componentes definidas y un comodín, se dice que este en un patrón parcial de orden 2 (o patrón-2), periodo 3 (Gadiraju, 2013).

Las definiciones de patrón completo o parcial no excluden el uso de más de un símbolo en un elemento del patrón: a modo de ejemplo, {AA*}, {CAA}, {B(A, C)A} o {**(B, C)} son todos patrones de periodo igual a 3.

2.1.1. Subpatrón, superpatrón

Existe la relación de subpatrón o superpatrón entre dos PPPs. Sean $s = s_1, ..., s_p$ y $s' = s'_1, ..., s'_p$ dos PPPs en el mismo dominio; se dice que s' es un subpatrón de s si $s'_i \subseteq s_i$ en cada posición donde $s'_i \neq *$. Por otro lado, se dice que s es un subpatrón de s'.

Por ejemplo, $***AB$ es un subpatrón-2 del patrón-4 $A * C\{A, C\}B$.

2.2. Búsqueda de patrones parciales periódicos frecuentes

2.2.1. Método Apriori

El método Apriori (Agrawal y Srikant, 1994) para la búsqueda de patrones parciales periódicos frecuentes (BPPPF) aprovecha una propiedad similar al lema Apriori para la minería de reglas de asociación, como sigue: **Cada subpatrón de un patrón frecuente de periodo *p* es a la vez un patrón frecuente de periodo *p***.

Aprovechando este principio, el método Apriori consiste entonces de dos etapas:

(a) Dado un criterio mínimo de soporte (o confianza), encontrar F_1: el conjunto completo de patrones-1 frecuentes de periodo p.

(b) Encontrar F_i con i desde 2 a p, o hasta el punto en que el conjunto F_i sea vacío.

La búsqueda de F_i es más eficiente que la búsqueda a fuerza bruta porque si determinamos que un patrón s no pertenece a F_i, descartamos la posibilidad de encontrar cualquier superpatrón de s en $F_j, i < j$.

2.2.2. max-subpattern

Max-subpattern (Han et al., 1999) es una técnica que emplea una estructura de árbol denominada *max-subpattern tree* para resolver la BPPPF. La raiz de dicho árbol la conforma un patrón parcial hipotético que consiste de la combinación de todos los PPPs frecuentes en F_1. A este PPP hipotético se conoce cómo **max-pattern**. El fundamento teórico de esta técnica postula que cualquier PPP frecuente en una base de datos secuenciales es un subpatrón de max-pattern.

El algoritmo max-subpattern se lleva a cabo en dos etapas. En una primera etapa se escanea por completo la base de datos secuenciales para encontrar F_1, y por asociación, max-pattern. En una segunda etapa, se recorre nuevamente la base de datos buscando especificamente subpatrones de max-pattern, a las que se les llama **hit-patterns**. Hit-patterns son insertados en el árbol max-subpattern, y cada hoja conserva un contador del número de ocurrencias de un hit-pattern específico. Concluída esta etapa, el soporte de cada patrón parcial inserto en el árbol se determina sumando los contadores desde una hoja hasta la raíz.

2.3. Locality-sensitive hashing

Por definición, *Locality-sensitive hashing* (LSH) es una técnica que resuelve el problema (R, c)-vecinos cercanos (abreviado (R, c)-NN), donde la meta es reportar un punto dentro de una distancia cR de una consulta q asumiendo que existe un punto a distancia R de q en el set de datos (Datar et al., 2004).

Para un set de datos en un dominio S con una métrica de distancia D, donde $B(q, r)$ es la *bola* de radio r centrada en q, una familia de funciones hash *locality-sensitive* se define como:

Definición 2.1 *Una familia* $H = h : S \to U$ *se dice* (r_1, r_2, p_1, p_2)*-sensitive para* D *si para cada* $v, q \in S$

- *si* $v \in B(q, r_1)$ *entonces* $Pr_H[h(q) = h(v)] \geq p_1$

- *si* $v \notin B(q, r_2)$ *entonces* $Pr_H[h(q) = h(v)] \leq p_2$

En términos simples, esta definición implica que en un tabla hash hipotética indexada por $h(v)$ la probabilidad de colisión entre dos elementos es mayor que una cierta cota mínima si ambos elementos son relativamente cercanos, y es menor que una cota máxima si los elementos se encuentran relativamente aparte.

Para abordar el problema (R, c)-NN, se requiere una familia de funciones hash (R, cR, p_1, p_2)-*sensitive*, $c > 1$, donde p_1 es considerablemente mayor que p_2. La función vastamente adoptada propuesta en (Datar et al., 2004) que satisface este requisito es: dados $\mathbf{a} \in S$ y $b, r \in U$,

$$h(\mathbf{v}) = \left\lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{r} \right\rfloor$$

h en este caso define todas las particiones posibles de S de un ancho r, y se ha demostrado que según esta función de hash, $Pr_H[h(\mathbf{q}) = h(\mathbf{v})]$, la probabilidad de colisión, disminuye monotónicamente a medida que $||\mathbf{q}, \mathbf{v}||$ incrementa (Huang et al., 2015).

Un algoritmo de LSH consiste de tres etapas: primero, el pre-procesamiento, donde cada entrada de la base de datos es indexada según la función de hash. Salvo algunas excepciones, como regla no se ocupa tan solo una función de hash, sino que se obtienen múltiples funciones empleando parámetros aleatorios. En una seguna etapa, el sistema recibe una consulta que es indexada al igual que la base de datos usando las mismas funciones hash. Al comparar los hash se calculan colisiones entre la consulta y la base de datos. Si una entrada de la base de datos muestra un número suficiente de colisiones, se dice que es un candidato a vecino cercano y es apartada para post-procesamiento. Finalmente, en el post-procesamiento se ratifica que los candidatos efectivamente se traten de vecinos cercanos. Dependiendo de la implementación, en ciertas instancias este último paso puede ser innecesario.

2.3.1. Min-hashing

Min-hashing es una variante de LSH propuesta originalmente por Cohen et al. con el fin de descubrir reglas de asociación (RA) de alta confianza, en contraste a las RAs de alta soporte que son más comúnmente reportadas. Para entender la importancia de esta distinción asuma una base de datos transaccional descrita por una matriz 0/1 M, con n filas y m columnas: cada columna representa una transacción, y cada fila un ítem en el catálogo (cada columna básicamente representa un conjunto de ítems). En esta representación, una RA de alto soporte, digamos 40 %, aparecería al encontrar dos columnas que coinciden al menos a lo largo del 40 % de la columna (coincidencia en términos de la conjunción lógica).

$$Sop(c_i, c_j) = \frac{|C_i \cap C_j|}{n}$$

donde C_i es el conjunto de las filas con el valor uno a lo largo de la columna c_i.

Una RA de alta confianza, en cambio, aparecería como dos filas que coinciden al menos a lo largo del 40 % **de la unión lógica de ambas filas**.

$$S(c_i, c_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

Nótese que esta es una definición especial (para ser exactos, esta es la definición del coeficiente de Jaccard), ya que la expresión de confianza común y corriente es asimétrica para una de las columnas.

$$Conf(c_i, c_j) = \frac{|C_i \cap C_j|}{|C_i|}$$

Con esta distinción, las RAs de alta confianza consiguen representar instancias donde es posible que ninguno de un par de ítems aparezca en la base de datos con alto soporte, caso que es común en bases de datos transaccionales con un catálogo muy grande de ítems.

Siguiendo con la misma representación matricial, $S(c_i, c_j)$ sirve como una métrica de distancia para la búsqueda de similitud. (Cohen et al., 2001) propuso una manera de estimar de manera eficiente S, el coeficiente de Jaccard entre dos conjuntos, empleando una representación numérica de conjuntos que involucra la siguiente función hash:

Imagine que las filas de M son permutadas aleatoriamente k veces. Se define $h_l(c_j), l = 1, ..., k$, como la primera fila diferente de cero en la permutación l de la columna j. $h_l(c_j)$ puede ser embebido en una matriz $k \times m$ a la que llamaremos $\widehat{M}$.

A partir de esta representación se define $\widehat{S}(c_i, c_j)$, una aproximación de la similaridad entre dos conjuntos, como:

$$\widehat{S}(c_i, c_j) = \frac{|\{l|1 \leq l \leq k \wedge \widehat{M}_{li} = \widehat{M}_{lj}\}|}{k}$$

En términos más simples, $\widehat{S}(c_i, c_j)$ es la fracción de las coincidencias entre dos columnas a lo largo de las filas en $\widehat{M}$. Se puede demostrar que para k suficientemente grande, $\widehat{S}(c_i, c_j) \simeq S(c_i, c_j)$.

Un algoritmo de MinHash computa la matríz "firma" $\widehat{M}$ donde cada conjunto en la muestra es representado por una combinación de k funciones MinHash. De este modo, la búsqueda aproximada se puede llevar a cabo al comparar la firma de una consulta $\mathbf{q}$ con la matriz $\widehat{M}$, y evaluando $\widehat{S}(\mathbf{q}, c_j)$ para encontrar los conjuntos más similares.

Capítulo 3

Estado del Arte

Locality-sensitive hashing (LSH) está entre las alternativas más populares para la búsqueda aproximada, con aplicaciones prácticas dentro y fuera del campo de la computación, debido en gran parte a su velocidad y bajo costo de almacenamiento. Más recientemente, sin embargo, avances en la teoría del LSH han dado lugar a nuevas vertientes bajo una categoría denominada *data-dependent hashing* que se distingue de los ejemplos pasados de LSH en que las funciones de hash empleadas son diseñadas a partir de una exploración supervisada o no supervisada de la base de datos, de modo similar al aprendizaje de máquina. Un ejemplo popular es *deep hashing* (Ge et al., 2019) que emplea *deep networks* para aprender funciones de hash.

Esto no quiere decir que la vertiente original del LSH no ha encontrado nuevos avances. *Index-of-max* (IoM) hashing es un nuevo modelo de LSH popularizado en un artículo del 2018 (Jin et al., 2018) diseñado para la encriptación y protección de patrones biométricos.

Query-aware LSH (Huang et al., 2015) (QALSH) es una evolución prometedora de la forma original de LSH, a la que en este contexto se le denomina *query-oblivious* LSH. Un algoritmo de query-oblivious LSH obtiene particiones del espacio de búsqueda a partir de una selección aleatoria de parametros con la esperanza de conseguir la mejor partición posible del espacio consumiendo tan poco almacenamiento como sea posible. QALSH argumenta que una partición del espacio de los datos no es necesaria. Empleando una función de hash contínua, los valores hash son almacenados de manera ordenada en un árbol B+ de modo que al llegar una nueva consulta es posible encontrar una partición centrada en el hash de la consulta por medio de una búsqueda por rango.

Min-hashing (Cohen et al., 2001) es una propuesta que adapta LSH para su uso en la minería de reglas de asociación. El uso del coeficiente de Jaccard como medida de similitud ha hecho a esta una solución popular para la búsqueda de similitud en series de tiempo (Chiu et al., 2010).

Capítulo 4

Estudio del problema y solución propuesta

4.1. Contexto del problema

La búsqueda de patrones periódicos parciales frecuentes (BPPPF) comienza por el sondeo de una serie de tiempo finita $S = s_1, s_2, ..., s_L$, $s_i \in I$, donde I es un conjunto finito numerable de ítems sin ordinalidad. Se predefine un largo fijo, o periodo, para la búsqueda, de modo que un tramo de periodo P se define como $T(i) = s_i, s_{i+1}, ..., s_{i+P-1}$. Se extraen, sin sobreposición, todos los tramos posibles a lo largo de S y se intenta determinar los patrones periódicos parciales (PPPs) que aparecen con frecuencia en $\mathbf{T} = T(0), T(1P), T(2P), ..., T(\lfloor L/P \rfloor - 1)$ a partir de todas las combinaciones de patrones posibles. Válgase recalcar que el número de PPPs posibles, es decir, toda combinación posible de items (incluyendo al ítem comodín) en un patrón de periodo P corresponda a

$$(|I| + 1)^P - 1$$

A grandes rasgos, las soluciones a la BPPPF navegan el espacio de PPPs que son posibles de encontrar en un set de datos (o más bien, un espacio reducido), y corrobora que se traten de PPPs frecuentes estimando valores individuales de soporte y/o confianza.

La meta de este trabajo es, idealmente, diseñar una técnica aproximada de BPPPF basada en el *locality-sensitive hashing* (LSH), a la que nos podemos referir como BPPPF-LSH, capaz de reportar PPPs frecuentes. Para este propósito, nos referimos a trabajos como el de (Chiu et al., 2010) para comprender el ámbito de esta misión.

El primer requisito para diseñar una solución basada en el LSH, o búsqueda aproximada, es ingeniar una transformación, o función hash, capaz de insertar datos en un espacio donde una métrica de distancia refleje apropiadamente diferencias o, por el contrario, similitud entre datos. Más aún, dicha métrica de distancia debe ser relevante al problema en cuestión. Una función hash con estas propiedades dícese ser *locality-sensitive*.

En última instancia, fue esta la cuestión que definió el alcance de este trabajo: idear una

función hash locality-sensitive relevante a la búsqueda de BPPPF, verificar su validez poniéndola en práctica para la solución de problemas reales de BPPPF, y evaluar el rendimiento de la nueva solución en comparación con otras técnicas del estado del arte.

4.2. Metodología propuesta

4.2.1. Set de datos: Trayectorias GPS

A modo de implementar un problema real de BPPPF, se empleó un corpus de trayectorias GPS publicado por Microsoft Research Asia, última actualización en el 9 de agosto del 2012, bajo el título *"Project Geolife"* (Zheng et al., 2008, 2010, 2009), disponible para uso académico con descarga en la siguiente URL: `https://www.microsoft.com/en-us/download/details.aspx?id=52367&from=https%3A%2F%2Fresearch`.

Desde su última actualización, el corpus cuenta con trayectorias GPS de 182 usuarios capturadas entre abril del 2007 y agosto del 2012 a lo largo de 30 ciudades en China pero principalmente en Pekín. Cada archivo, de formato ".plt", da cuenta de la trayectoria registrada por un usuario a lo largo de un día, y distribuye siete atributos a lo largo de siete columnas.

39.966641	116.331531	0	230	39750.4093	10-29-2008	9:49:26
39.966627	116.331421	0	229	39750.4094	10-29-2008	9:49:31
39.966597	116.331301	0	227	39750.4094	10-29-2008	9:49:34

Figura 4.1: Estructura general de un documento en el formato .plt. Los documentos no incluyen un encabezado, y cada fila describe un punto en el tiempo de una determinada trayectoria GPS.

- Columna 1: Latitud, en grados decimales.
- Columna 2: Longitud, en grados decimales.
- Columna 3: Dígito "0" a lo largo de todas las filas.
- Columna 4: Altitud, en pies.
- Columna 5: Fecha, días transcurridos desde el 30/12/1899.
- Columna 6: Fecha, como cadena.
- Columna 7: Hora, como cadena.

4.2.2. Preprocesamiento

Los campos de interés para la BPPPF son la latitud y la longitud. La naturaleza continua de estos atributos no es apta para la BPPPF, sin embargo, como se ilustra en la figura 4.2, al sacrificar en parte la fidelidad es posible traducir los registros a una serie finita de símbolos discretos al condensar coordenadas exactas en cuadrantes definidos en la región relevante.

Se llevaron a cabo intentos de automatizar la definición de los límites de los cuadrantes a modo de minimizar la pérdida de información, pero ningún intento rindió mejores resultados que la observación y selección deliberada de las dimensiones generales de los cuadrantes. Los mejores

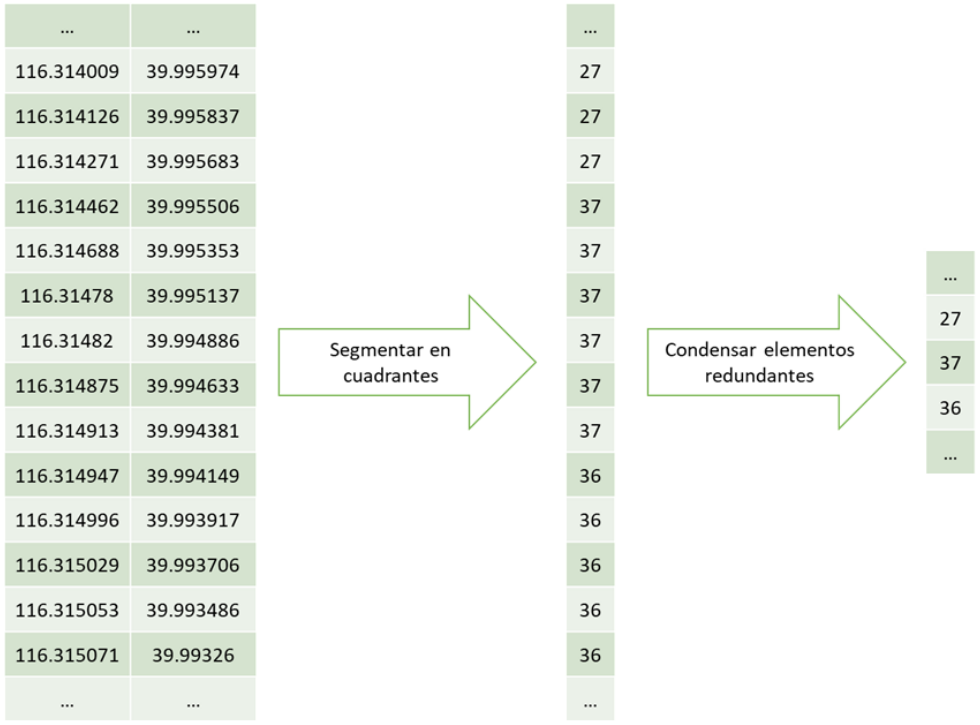

Figura 4.2: Esquema general del preprocesamiento. Los registros de trayectorias GPS, compuestos por latitud y longitud, son segmentados en un número finito de cuadrantes espaciales. A modo de reducir la redundancia en las secuencias resultantes, cada serie es reducida a una representación exclusiva de los desplazamientos de cuadrante a cuadrante.

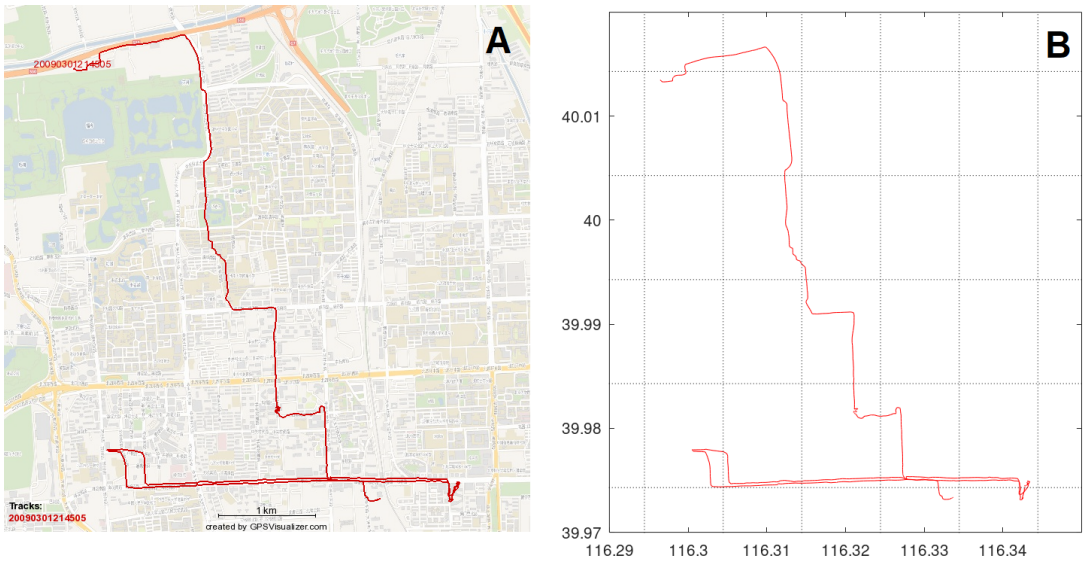

Figura 4.3: Ejemplo de una trayectoria GPS (A), y la segmentación en cuadrantes correspondiente (B).

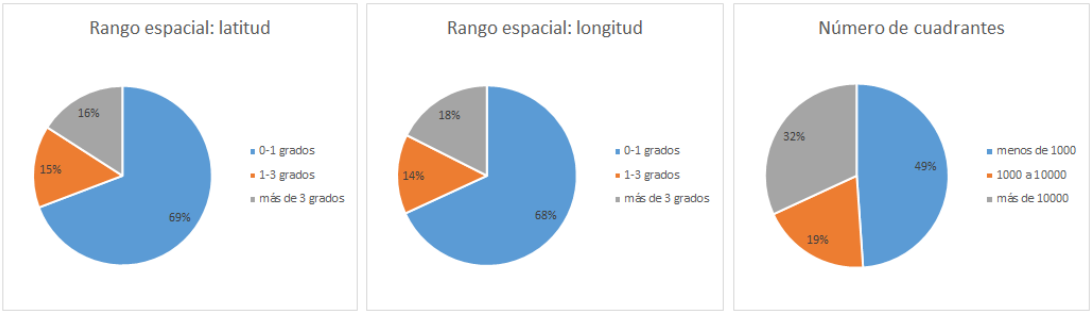

Figura 4.4: Distribución general del rango espacial en que se sitúan las trayectorias de los 182 usuarios en la muestra. La gran mayoría (82 %) se puede situar en un rango de 3.00 x 3.00 grados en unidades del sistema de coordenadas geográficas. Tras la segmentación de los espacios en cuadrantes, el 68 % de la muestra produce una partición de menos de 10000 cuadrantes.

resultados se dieron al fijar dimensiones de 0,01 x 0,01 para cada cuadrante (unidades de grados decimales en el sistema de coordenadas geográficas, aproximadamente 1,11 x 1,11 kilómetros). Las coordenadas exactas de las líneas delimitantes fueron definidas separadamente para cada uno de los 182 individuos de acuerdo a los siguientes pasos:

(a) Encontrar las coordenadas mínimas y máximas de latitud y longitud (dígase lat_{min}, lat_{max}, lon_{min} y lon_{max}).
(b) Las coordenadas de las delimitantes son

$$(lat_{min} : 0{,}01 : lat_{max}) + (lat_{max} - lat_{min}) \textbf{ rem } 0{,}01$$

$$(lon_{min} : 0{,}01 : lon_{max}) + (lon_{max} - lon_{min}) \textbf{ rem } 0{,}01$$

Luego, cada cuadrante es asignado un identificador único desde el cero al número total de cuadrantes menos uno. Nótese que la ordinalidad de estos identificadores no tiene mayor importancia.

La figura 4.3 presenta un ejemplo singular de trayectoria GPS y la partición en cuadrantes correspondiente. Válgase recalcar que esta es tan solo una de las múltiples trayectorias que forman el registro de un usuario.

Por cada usuario se lleva a cabo una solo iteración de BPPPF, es decir, todos sus registros son agregados a una misma base de datos de series de tiempo. Esto no quiere decir que los registros son concatenados para formar una sola serie de tiempo. Una vez determinado el periodo de interés para una iteración de BPPPF, cada registro es estudiado por separado, y uno por uno, tramos de la serie de tiempo del periodo indicado son seleccionados y agregados a la base de datos correspondiente a dicho usuario.

Una última consideración fue condensar la cantidad de información redundante en cada serie de tiempo. Debido a la pérdida de información como resultado de la segmentación de datos espaciales en cuadrantes, las series de datos resultantes incluyen largos segmentos donde el mismo símbolo se repite un número indefinido de veces. La información que se puede discernir a partir de estas largas secuencias monótonas habla de ya sea la duración en la que el usuario de detiene en

ciertos cuadrantes, o las diferencias en el tiempo que tarda en desplazarse. Sin embargo, para un algoritmo de BPPPF es difícil extraer información útil de este estilo, pues esta información no se refleja en los valores de soporte que reportan dichos algoritmos. Por lo tanto, se optó por emplear una representación abreviada de las trayectorias donde son de interés tan solo los movimientos de cuadrante a cuadrante (figura 4.2). Nos referiremos a estas secuencias como **series de tiempo abreviadas**.

4.2.3. Selección

Por la mayor parte, los registros GPS disponibles para cada usuario dan cuenta de trayectorias contenidas en un rango de alrededor de 10000 km^2, o 100 x 100 km. Algunos registros, sin embargo, dan cuenta de viajes de larga distancia que a) no comparten mucho en común con el resto de los registros de un usuario, b) obstaculizan el preprocesamiento al incrementar el número de cuadrantes en consideración a un número insostenible para los algoritmos de BPPPF, y, en última instancia, c) se tratan de *outliers*, y no contribuyen patrones periódicos interesantes (figura 4.5).

Se decidió buscar y rechazar estos registros outliers antes de dar comienzo a la etapa de preprocesamiento. Este proceso de selección se condujo en dos niveles: a nivel de registros y a nivel de usuarios.

Selección a nivel de registros

Previo al preprocesamiento, se buscaron registros con medias de latitud y longitud fuera de la norma. Empleando la técnica de análisis de grupos *k-means* con $k = 2$, las medias bidimensionales fueron clasificadas en dos grupos. Para determinar si el menor de los dos es efectivamente un grupo outlier, se comprobó que el centroide del grupo menor se encuentra fuera del rango completo del grupo mayor. Los registros de un grupo outlier fueron rechazados para la BPPPF.

Selección a nivel de usuarios

Posterior al preprocesamiento fueron rechazados usuarios con registros insuficientes, o demasiado cortos para garantizar si quiera satisfacer los requisitos mínimos de soporte (menos de 100 elementos acumulados a lo largo de todas las series de tiempo abreviadas).

Más aún, fueron rechazados usuarios cuya segmentación rindió un número insostenible de cuadrantes (más de 10000). La figura 4.4 da una idea de la distribución del número de cuadrantes tras la segmentación a lo largo de los 182 usuarios.

En total, fueron rechazados 102 usuarios.

4.2.4. BPPPF: Método hash

PPPhash, una función locality-sensitive para patrones parciales

Sean

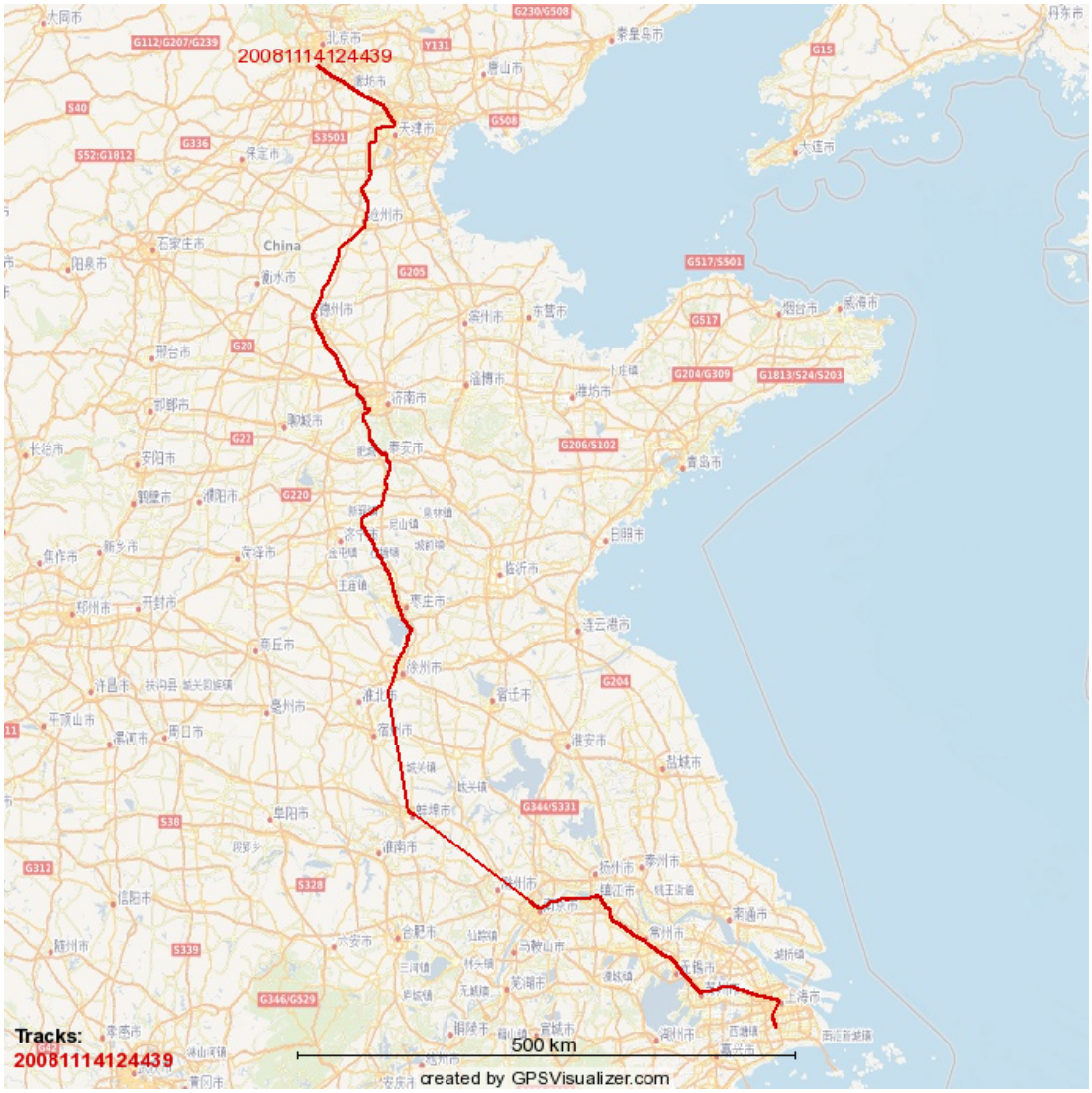

Figura 4.5: Ejemplo de trayectoria GPS rechazada por el proceso de selección. La escala de esta trayectoria es cientos de veces mayor a otras trayectorias en el registro.

$I = \{A\ B\ C\}$
$|I| = 3$
$P = 5$

Patrón	A	B	C	A	B
Representación numérica	0	1	2	0	1
h: Pares 0-1	1	5	6	1	3
h: Pares 0-2	2	3	7	0	4
h: Pares 0-3	0	4	6	1	5
h: Pares 0-4	1	3	7	2	3

$$h(T_i, T_j) = |I|T_i + T_j$$

$$T \rightarrow V$$
$$[ABCAB] \rightarrow [1\ 5\ 6\ 1\ 3\ 2\ 3\ 7\ 0\ 4]$$
$$|V| = P \times \lfloor P/2 \rfloor = 10$$

Figura 4.6: Demuestra la construcción de V a partir de un tramo arbitrario de periodo 5. Cada símbolo es representado por un número en $\{0, 1, 2\}$. Más abajo, cada fila representa el resultado de $h(T_i, T_j)$ entre distintos pares de símbolos en el tramo. La tabla muestra cada combinación de pares posible, sin embargo, las últimas dos filas son redundantes: cada valor aquí es el recíproco de un valor en las primeras dos filas: $h(T_j, T_i)$. Finalmente, V es la concatenación de todas las filas no redundantes.

- $S = s_1, s_2, ..., s_L$, $s_i \in I$ una serie de tiempo finita de largo L, donde I es un conjunto finito numerable de ítems sin ordinalidad,
- $T(i) = s_i, s_{i+1}, ..., s_{i+P-1}$ un tramo de largo P, $P << L$, y
- $\mathbf{T} = T(0), T(1P), T(2P), ..., T(\lfloor L/P \rfloor - 1)$ la colección ordenada de todos los tramos encontrados, sin sobreposición, en S.

Se define la función $h : I^2 \rightarrow H$ como

$$h(T_i, T_j) = |I|T_i + T_j$$

donde $|I|$ es la cardinalidad del conjunto I. Por conveniencia cada símbolo en I es representado por un número entero entre cero e $|I| - 1$. De este modo, sabemos inmediatamente que el rango de H es $\{0, 1, ..., (|I|^2 - 1)\}$.

Asumiendo que I no es vacío ni incluye elementos repetidos, se comprueba que la transformación h es inyectiva en $h : I^2 \rightarrow H$.

$$h(A_1, B_1) = h(A_2, B_2)$$
$$|I|A_1 + B_1 = |I|A_2 + B_2$$
$$|I|(A_1 - A_2) = (B_2 - B_1)$$

Dado que $|I| > 0$, esta igualdad solo se comprueba cuando

$$\begin{aligned} (A_1 - A_2) = 0 \wedge (B_2 - B_1) = 0 \qquad & \text{por lo tanto,} \\ h(A_1, B_1) = h(A_2, B_2) \qquad & \text{implica que} \\ A_1 = A_2 \wedge B_1 = B_2 & \end{aligned}$$

Emplearemos h para producir un vector en H que identifique únicamente todas las formas de patrones periódicos en I, completos o parciales.

Se define el vector V como

$$V_i(T) = h(T_{i \bmod P}, T_{(i+1+\lfloor i/P \rfloor) \bmod P}) \qquad i = 0, ..., (\lfloor P/2 \rfloor P)$$

La figura 4.6 demuestra en términos más simples los pasos para construir V. Básicamente, se evalúa cada par posible de ítems en un tramo T, hasta cierto punto. En el ejemplo, los pares 0-1 y 0-4, o 0-2 y 0-3 son recíprocos, y por lo tanto, redundantes. El número total de elementos no redundantes en esta relación está dado por $\lfloor P/2 \rfloor P$, y esta cifra, por su parte, determina la dimensionalidad de V. Esta acotación es significante, ya que implica que la dimensionalidad de V no depende de ningún modo del número de símbolos, o $|I|$.

Para facilitar la comprensión, llamaremos *PPPhash* al vector V. La ventaja más relevante de este método de representación de patrones es que ofrece una forma muy sencilla de comparación que es especialmente útil para la BPPPF: la conjunción lógica. Sean $A, B \in \mathbf{T}$

Patrón	I = {A B C}	0,1	1,2	2,3	3,4	4,0	0,2	1,3	2,4	3,0	4,1
T1	**ABCAB**	1	5	6	**1**	3	2	**3**	7	0	**4**
T2	**CBAAB**	7	3	0	**1**	5	6	**3**	1	2	**4**
T1 ∩ T2	***B*AB**	0	0	0	**1**	0	0	**1**	0	0	**1**

$$\sum T1 \cap T2 = \frac{3!}{2!\,(3-2)!} = 3$$

Patrón	I = {A B C D E}	0,1	1,2	2,3	3,4	4,5	5,6	6,0	0,2	1,3	2,4	3,5	4,6	5,0	6,1	0,3	1,4	2,5	3,6	4,0	5,1	6,2
T1	**ABEEBAC**	1	**9**	24	21	**5**	2	10	4	9	**21**	20	7	0	11	4	**6**	**20**	22	5	**1**	14
T2	**DBECBAB**	16	**9**	22	11	**5**	1	8	19	7	**21**	10	6	3	6	17	**6**	**20**	11	8	**1**	9
T1 ∩ T2	***BE*BA***	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	**1**	**1**	0	0	**1**	0

$$\sum T1 \cap T2 = \frac{4!}{2!\,(4-2)!} = 6$$

Figura 4.7: Conjunción lógica entre dos vectores V. El vector resultante adquiere dos propiedades útiles: 1) los valores individuales iguales a 1 denotan pares en el patrón parcial común entre T1 y T2. 2) La suma del vector resultante depende tan solo del orden del patrón parcial común. En el ejemplo de arriba, la suma es $3 = 3\mathbf{C}2$, mientra que abajo la suma es $6 = 4\mathbf{C}2$.

$$Sim_i(A,B) = \begin{cases} 1, & V_i(A) = V_i(B) \\ 0, & V_i(A) \neq V_i(B) \end{cases}$$

El vector resultante $Sim(A,B)$ denota con precisión el patrón parcial compartido entre los tramos A y B, como se ilustra en la figura 4.7. La suma de los elementos en $Sim(A,B)$ es una métrica de similitud entre patrones que es sensible al orden del patrón parcial común entre A y B, y no es sensible a diferencias específicas entre los ítems que conforman A o B.

$$\sum Sim(A,B) = \frac{o!}{2!(o-2)!} \qquad o \text{ es el orden del patrón común entre A y B}$$

Algoritmo

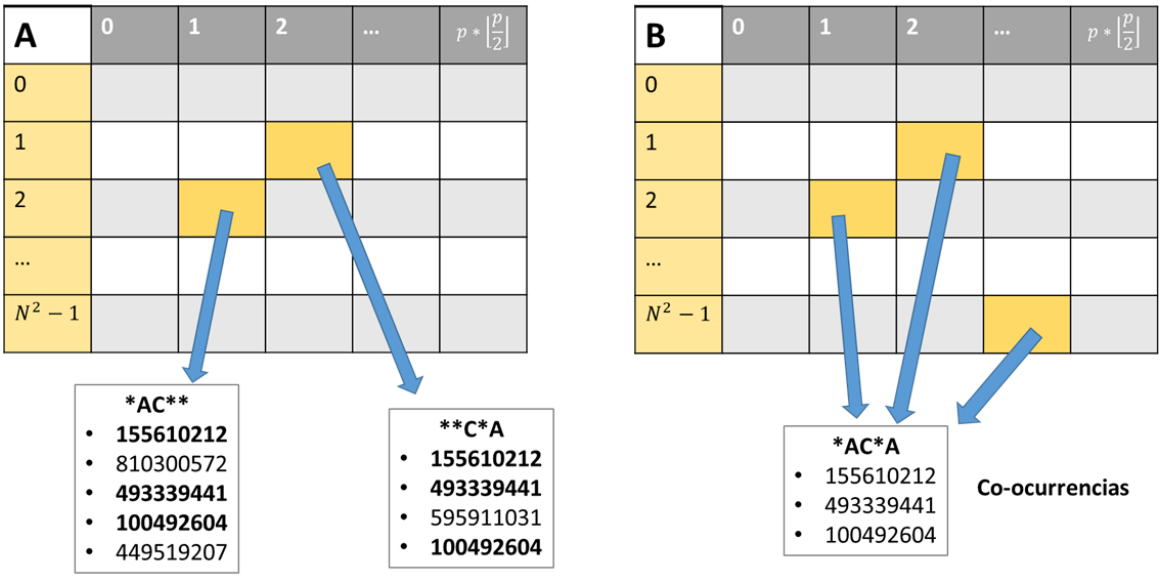

Figura 4.8: A y B representan una tabla de frecuencias hipotética. N^2-1 es el rango de $h(T_i, T_j)$, mientras que $P\lfloor P/2 \rfloor$ es la dimensionalidad de V. Cada celda en la tabla representa un patrón parcial de orden 2. Si sabemos que *AC** y **CA** representan dos patrones parciales frecuentes en la base de datos, y contamos con punteros indicando todas las ocurrencias de alguno de los dos patrones parciales, podemos inferir que el patrón *AC*A aparece en todas las co-ocurrencias entre los listados *AC** y **C*A. Una tercera línea en **B** sirve para denotar que lo mismo es cierto en relación al patrón parcial *A**A.

Con el fin de determinar la efectividad de PPPhash como herramienta para discernir y clasificar PPPs frecuentes, se diseño un algoritmo para la BPPPF que aprovecha las características de las funciones hash para, idealmente, brindar una solución eficiente.

A diferencia de otros algoritmos de BPPPF el primer paso es encontrar F_2, el conjunto de patrones parciales frecuentes de segundo orden. Este proceso es expeditado gracias a PPPhash:

(a) Primero, se escanea la base de datos de series de tiempo, tramo a tramo y sin superposición, e inmediatamente se crea una base de datos paralela donde cada tramo es representado por un vector PPPhash.
(b) En paralelo al paso anterior, se construye una tabla de frecuencias de dimensiones $|I|^2 \times (\lfloor P/2 \rfloor P)$ donde se cuenta el total de ocurrencias de cada uno de los $|I|^2$ símbolos posibles, distinguiendo en cuál índice de PPPhash aparece. Se recomienda el uso de una matriz *sparse* para este paso.
(c) Identificar F_2 al encontrar todos los índices donde la tabla de frecuencias es mayor al soporte mínimo. Cada celda en la tabla de frecuencias corresponde con un único patrón parcial de segundo orden.

Los pasos para determinar F_3 hasta F_P se basan en el siguiente principio.

Ejemplo: si T1 = *AC** y T2 = **C*A son PPPs frecuentes ($P = 5$), y se cuenta con un listado de identificadores para cada tramo donde ocurre ya sea T1 o T2, llámese dichos listados L1 y L2, entonces, si la conjunción lógica o número de coocurrencias entre L1 y L2 es mayor al requerimiento mínimo de soporte, se puede concluir que T4 = *AC*A es un PPP frecuente (figura 4.8), y más aún, L4 es la conjunción lógica de L1 y L2.

Este principio es cierto para cualquier combinación entre un PPP de orden 2 (T2) y un PPP de orden mayor o igual a 2 (T1). Sin embargo, T4 no siempre es un PPP de orden uno mayor que T1. Para garantizar esto, y así reducir significativamente el número de comparaciones por computar, conviene verificar lo siguiente:

- Determinar todos los índices diferentes de comodín en T2 (orden 2).
- Determinar todos los índices diferentes de comodín en T1 (orden mayor o igual a 2).
- Verificar que el índice menor en T2 sea igual al índice mayor en T1.
- Verificar que el índice mayor en T2 no aparece en T1.

La sugerencia de que T2 se de posterior a T1 no es arbitraria. En realidad es crucial para prevenir la redundancia en la búsqueda pareada. En el mismo ejemplo anterior, sabemos que T1 = *AC** y T2 = **C*A son PPPs frecuentes, pero esto implica a la vez que T3 = *A**A también es frecuente. Como resultado, las comparaciones de T1 y T2, T1 y T3 o T2 y T3 brindarán el mismo resultado: T4. Sin las prevenciones necesarias, esta realidad elevará exponencialmente el número de comparaciones redundantes. Sin embargo, las verificaciones listadas más arriba son suficientes para garantizar cero comparaciones redundates sin algún tipo de pérdida de información.

El siguiente paso involucra determinar los listados L para cada PPP frecuente en F_2 (llámese L_2). La tabla de frecuencias obtenida en el paso anterior nos brinda de antemano los tamaños de los listados asociados a cada PPP en F_2 y otorga la ventaja de prevenir la alocación de memoria dinámica.

A continuación, encontraremos F_3 al comparar L_2 con sí mismo, teniendo en cuenta las sugerencias listadas más arriba para prevenir comparaciones redundantes. Si el número de coocurrencias entre dos listados supera el requerimiento mínimo de soporte, el PPP de tercer orden compuesto por la conjunción de ambos patrones pasa a conformar F_3, y el listado de coocurrencias similarmente pasa a conformar L_3.

De manera similar, se consigue F_4 al comparar L_3 con L_2, F_5 al comparar L_4 con L_2, y así hasta F_P comparando L_{P-1} con L_2.

4.2.5. BPPPF: Max-subpattern

Como punto de comparación, todas las pruebas se realizaron en paralelo con una implementación en Octave del algoritmo max-subpattern para la BPPPF. Max-subpattern es implementado en tres etapas:

(a) Recorrer todos los tramos de un periodo predefinido en una base de datos secuenciales y elaborar una tabla de frecuencias de todos los posibles patrones parciales de orden 1. Inicializar el árbol max-subpattern con **Max-pattern**, el patrón parcial hipotético conformado por la combinación de todos los patrones en F_1, en la raíz.

(b) Recorrer por segunda vez la base de datos secuenciales, buscando específicamente subpatrones de max-pattern, llamados hit-patterns o candidatos. Insertar hit-subpatterns en el árbol.

(c) Determinar el soporte de los PPP candidatos sumando los contadores desde una hoja (cada hoja corresponde a un PPP) hasta la raíz. Identificar PPPs frecuentes.

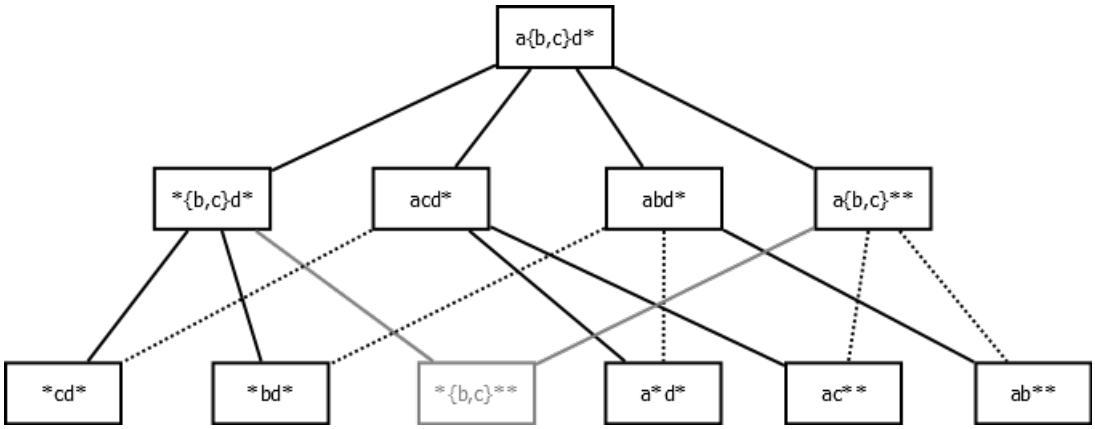

Figura 4.9: Ejemplo de un árbol max-subpattern. Cada nodo representa un PPP y mantiene un contador del número de ocurrencias de dicho PPP en la base de datos. En la raíz se encuentra el patrón max-pattern: A{B, C}D*, combinación de los patrones-1: A***, *B**, *C** y **D*. Patrones de orden uno como el aparece en gris no son insertados al árbol. Las líneas punteadas denotan la asociación entre un hijo y un ancestro realizada posterior a la creación del nodo hijo. Para fines prácticos esta distinción es irrelevante.

Cada hoja del árbol corresponde a un PPP e incluye un contador y punteros hacia hijos y ancestros (figura 4.9). Un hijo es un subpatrón del padre que resulta de la omisión de un elemento del PPP padre. El número de hijos de una hoja es a lo más equivalente al número de elementos en el PPP padre. Los ancestros de una hoja son todos los superpatrones en el árbol que resultan de la adición de un único elemento en el PPP hijo. Durante la creación de cada nodo nuevo se deben asociar el nuevo nodo con todos los ancestros correspondientes.

Una vez concluida la creación del árbol max-subpattern, es posible determinar la frecuencia de cada PPP embebido en la estructura sumando los contadores desde la hoja hasta la raíz.

Capítulo 5

Resultados

Para determinar el rendimiento y la confiabilidad de la metodología desarrollada, a la que por simplicidad nos referiremos como *PPPhash*, se condujeron pruebas empleando una base de datos real de trayectorias GPS, y se evaluaron los resultados y el rendimiento en comparación con *max-subpattern*, un algoritmo bien documentado de BPPPF.

5.0.1. Preliminares

Se condujeron en total seis pruebas independientes con una combinación de los siguientes parámetros de búsqueda:

- Periodo (P): 4, 5 o 6
- Soporte minimo: 15 o 20

Un mayor rango de opciones serie deseable, particularmente con respecto al soporte mínimo, sin embargo, tras concluir el preprocesamiento, las series de datos resultantes no son lo suficientemente largas para acomodar requerimientos muy altos de soporte, y por otro lado, requerimientos muy bajos conducían en muchos casos a estructuras de datos insostenibles. A pesar de estas limitaciones, las pruebas realizadas son suficientes para ilustrar tendencias claras en el rendimiento de ambos algoritmos. La figura 5.1 da cuenta del impacto de la elección del soporte mínimo sobre el rendimiento de ambos algoritmos.

5.0.2. BPPPF: Resultados

Una vez concluida cada prueba, se comprobó que los conjuntos F_2 hasta F_P producidos tanto por el método PPPhash o max-subpattern fueran idénticos. La figura 5.2 muestra la distribución de los volúmenes de F_1 hasta F_P a lo largo de todos los usuarios. Conjuntos vacíos fueron excluidos de esta representación.

5.0.3. BPPPF: Rendimiento

Ya sea PPPhash o max-subpattern, el rendimiento de un ciclo de búsqueda completo aparece fuertemente influido for dos factores: el número total de cuadrantes, y más directamente, el

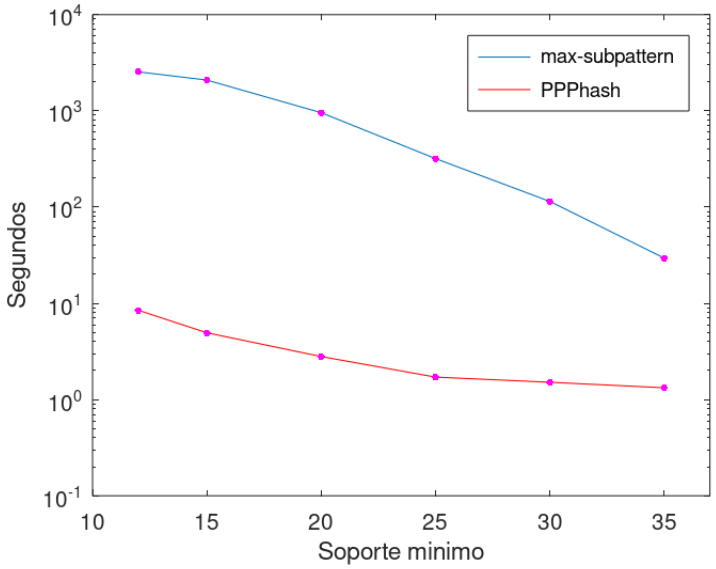

Figura 5.1: Un ejemplo de la evaluación del rendimiento para max-subpattern y PPPhash sometidos a distintos requerimientos de soporte ($P = 4$) muestra la tendencia esperada donde un requerimiento menor de soporte conlleva tiempos de ejecución mayores.

volumen de F_1. En teoría, ambos algoritmos son perfectamente aptos para resolver eficientemente la BPPPF incluso en ejemplos con un número de cuadrantes absurdamente grande. Esto se debe a que solo pasan a formar parte de las estructuras de datos aquellos registros de cuadrantes aceptados como parte de un PPP frecuente de orden uno, en el caso de max-subpattern, o un PPP frecuente de orden dos, en el caso de PPPhash.

La figura 5.3 ilustra la relación entre el tamaño de F_1 y el rendimiento de la búsqueda. Aparece a simple vista la relación logarítmica

$$\begin{aligned} logR &= c|F_1| \\ R &= 2^{c|F_1|} = c^{|F_1|} \qquad c > 1 \end{aligned}$$

donde R es el rendimiento de la búsqueda en segundos. Alternativamente,

$$\begin{aligned} logR &= c|F_2| \\ R &= 2^{c|F_2|} = c^{|F_2|} \qquad c > 1 \end{aligned}$$

Es aún más evidente en el caso particular de PPPhash (figura 5.4). Los resultados sugieren que el coeficiente c, que podemos entender como el factor determinante del rendimiento, es decisivamente menor para PPPhash que para max-subpattern. Sin embargo, no es prudente

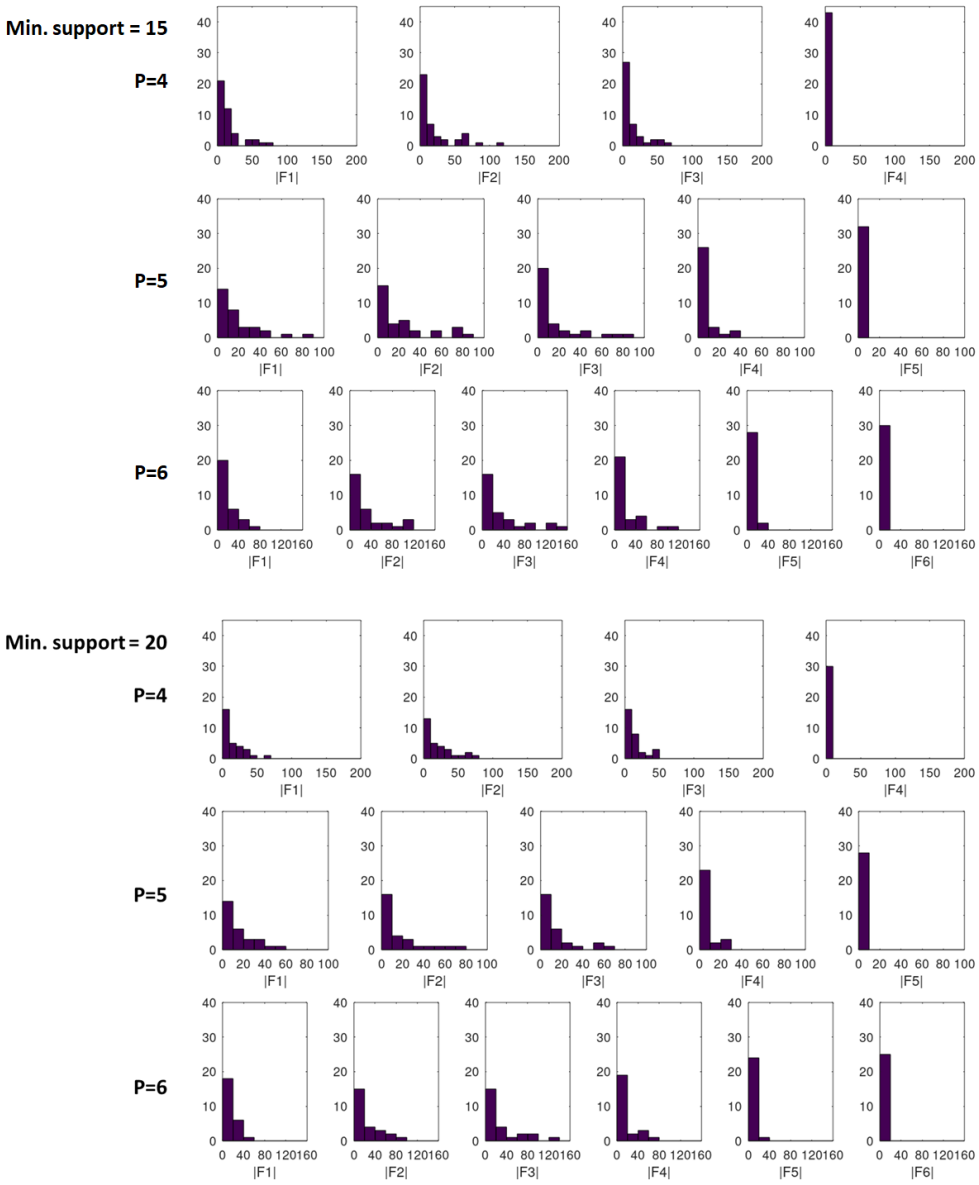

Figura 5.2: Resultados; distribución de frecuencias entre los distintos PPPs frecuentes: desde F_1 hasta F_P.

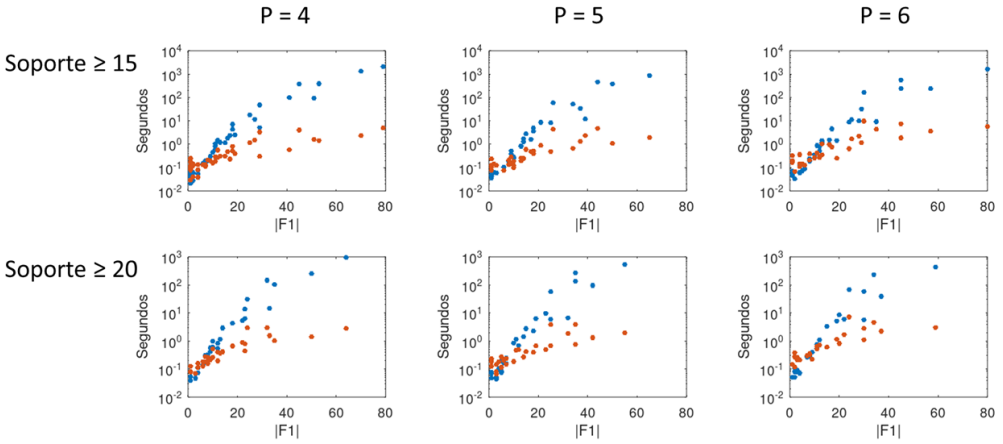

Figura 5.3: Rendimiento en segundos (escala logarítmica) versus volumen del conjunto F_1.

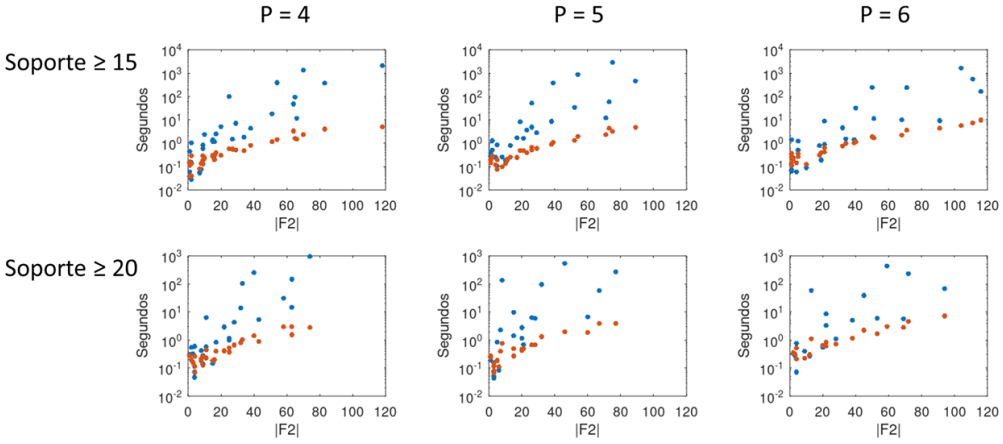

Figura 5.4: Rendimiento en segundos (escala logarítmica) versus volumen del conjunto F_2.

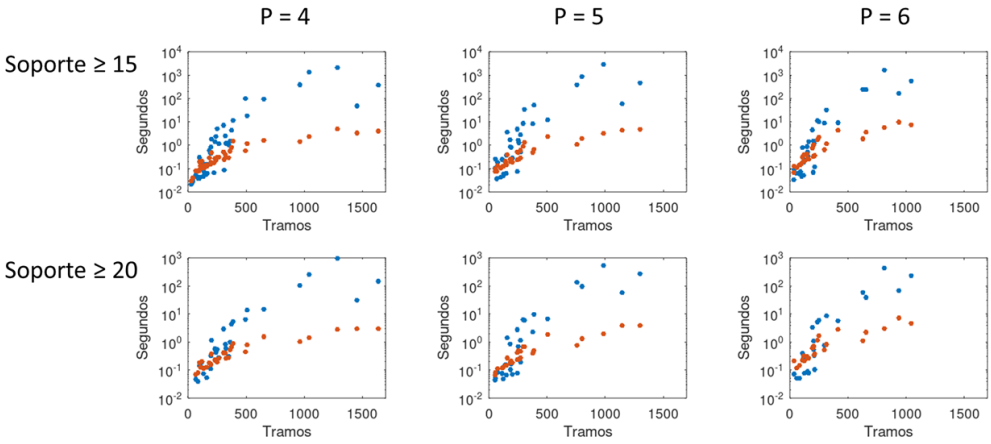

Figura 5.5: Rendimiento en segundos (escala logarítmica) versus el número de tramos examinados.

llegar a conclusiones prematuras al respecto antes de conducir más pruebas de las que permite el alcance de este trabajo.

Intentos de relacionar el rendimiento de la búsqueda con el número de tramos no ofrece evidencias conclusivas (figura 5.5), y esto en cambio da evidencia de que la navegación de las estructuras de datos involucradas en ambos algoritmos es un factor delimitante mucho más importante que el sondeo de la base de datos.

Capítulo 6

Conclusiones

Este trabajo representa un primer paso en el desarrollo de un método aproximado para la búsqueda de patrones parciales periódicos frecuentes (BPPPF). Entre los algoritmos de búsqueda aproximada más reconocidos en el presente están los que se basan en el trabajo de Datar e Indyk (Datar et al., 2004) donde se propone un framework para diseñar estos algoritmos haciendo uso de funciones hash denominadas *locality-sernsitive hash functions*, o funciones LSH. En términos generales, los algoritmos de búsqueda basados en LSH proveen una solucion aproximada al problema k-vecinos cercanos, o k-NN (en particular, se llama ANN a un k-NN aproximado), sin embargo, muchos problemas pueden ser replanteados como un problema de búsqueda de vecinos cercanos.

PPPhash, la transformación propuesta en este trabajo, inserta tramos de una base de datos secuenciales de un periodo definido (P) en un espacio lineal de dimensionalidad conocida ($P\lfloor P/2 \rfloor$), con una métrica de distancia que es sensible exclusivamente al único parámetro de interés para la BPPPF: el patrón parcial común a dos tramos independientes. O más específicamente: el orden de dicho patrón parcial.

A modo de ejemplo, asuma que T_1 = AABBA y T_2 = AACBB son tramos de una base de datos secuenciales con los símbolos $I = \{\text{A, B, C}\}$. El patrón parcial común entre T_1 y T_2 es AA*B*, un patrón parcial de orden 3. Si pretendemos buscar tramos similares, podemos usar como indicador el orden del patrón parcial común entre dos tramos. Si el patrón parcial común entre T_3 y T_4 es **CAA, esto no implica que T_3 y T_4 sean menos o más similares entre sí que T_1 y T_2. Es por esto que es crucial que la medida de similitud propuesta no sea sensible a diferencias específicas en el patrón parcial común. Vale recalcar que el orden del patrón parcial común como un indicador de similitud tiene un paralelo en ámbito de los conjuntos: el coeficiente de Jaccard.

PPPhash como método de representación de bases de datos secuenciales permite plantear la BPPPF como un problema de k-NN: sea $\mathbf{q}$ un patrón parcial arbitrario de periodo P. Considere la distancia como el recíproco del indicador de similitud descrito más arriba (sencillamente distancia = P − similitud).

- El conjunto de todos los vecinos cercanos a una distancia igual a 0 revela el soporte de $\mathbf{q}$ en la base de datos.
- El conjunto de todos los vecinos cercanos de orden menor o igual a 1 revela todos los

subpatrones de **q** de orden 1 menos que **q**.
- El conjunto de todos los vecinos cercanos de orden menor o igual a 2 revela todos los subpatrones de **q** de orden 2 menos que **q**.
- etc.

Paso a paso, este proceso revela todos los subpatrones de **q** hasta los patrones de orden 1.

Una implementación de este estilo no se probó en este estudio (y tampoco ofrece algún tipo de garantia de mejor rendimiento), pero sirve para ilustrar que es posible visualizar la BPPPF como un problema de vecinos cercanos con la ayuda de PPPhash.

El trabajo para diseñar un algoritmo aproximado para la BPPPF basado en LSH (BPPPF-LSH) va más allá del alcance de este trabajo, sin embargo, PPPhash ofrece un fundamento teórico para dicho emprendimiento.

El algoritmo propuesto que emplea PPPhash como método de representación de patrones parciales periódicos (PPPs) demuestra la confiabilidad de este método, y sorprendemente, los resultados sugieren que la técnica propuesta basada en el uso de tablas hash resuelve la BPPPF con notablemente mayor eficiencia que alternativas como max-subpattern. Sin embargo, se requerirá de más pruebas en distintos entornos para aseverar esto último con certeza.

Referencias

Rakesh Agrawal y Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. En *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pág. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994. ISBN 1558601538.

Chih-Yi Chiu, Hsin-min Wang, y Chu-Song Chen. Fast min-hashing indexing and robust spatio-temporal matching for detecting video copies. *TOMCCAP*, 6, 2010. doi:10.1145/1671962.1671966.

Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey Ullman, y Cheng Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering, IEEE Transactions on*, 13:64 – 78, 2001. doi:10.1109/69.908981.

Mayur Datar, Nicole Immorlica, Piotr Indyk, y Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. SCG '04, pág. 253–262. Association for Computing Machinery, New York, NY, USA, 2004. ISBN 1581138857. doi:10.1145/997817.997857. URL `https://doi.org/10.1145/997817.997857`.

N. Gadiraju. Periodic pattern mining-algorithms and applications. *Global Journal of Computer Science and Technology Software Data Engineering*, 13, 2013.

Lin-Wei Ge, Jun Zhang, Yi Xia, Peng Chen, Bing Wang, y Chun-Hou Zheng. Deep spatial attention hashing network for image retrieval. *Journal of Visual Communication and Image Representation*, 63:102577, 2019. ISSN 1047-3203. doi:https://doi.org/10.1016/j.jvcir.2019.102577. URL `http://www.sciencedirect.com/science/article/pii/S1047320319301920`.

Jiawei Han, Guozhu Dong, y Yiwen Yin. Efficient mining of partial periodic patterns in time series database. págs. 106–115. 1999. ISBN 0-7695-0071-4. doi:10.1109/ICDE.1999.754913.

Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, y Wilfred Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proc. VLDB Endow.*, 9(1):1–12, 2015. ISSN 2150-8097. doi:10.14778/2850469.2850470. URL `https://doi.org/10.14778/2850469.2850470`.

Z. Jin, J. Y. Hwang, Y. Lai, S. Kim, y A. B. J. Teoh. Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing. *IEEE Transactions on Information Forensics and Security*, 13(2):393–407, 2018. doi:10.1109/TIFS.2017.2753172.

Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, y Wei-Ying Ma. Understanding mobility based on gps data. En *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pág. 312–321. Association for Computing Machinery, New York, NY, USA, 2008. ISBN 9781605581361. doi:10.1145/1409635.1409677. URL `https://doi.org/10.1145/1409635.1409677`.

Yu Zheng, Xing Xie, y Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33:32–39, 2010.

Yu Zheng, Lizhu Zhang, Xing Xie, y Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. En *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pág. 791–800. Association for Computing Machinery, New York, NY, USA, 2009. ISBN 9781605584874. doi:10.1145/1526709.1526816. URL `https://doi.org/10.1145/1526709.1526816`.

Apéndice A

Propuesta de Tesis

UNIVERSIDAD DEL BIO-BIO
FACULTAD CIENCIAS EMPRESARIALES

SOLICITUD INSCRIPCIÓN ACTIVIDAD DE TITULACIÓN

1. IDENTIFICACIÓN ALUMNO(S).

NOMBRE	**:**	Marcelo Alejandro Stöckle Márquez
DIRECCIÓN	**:**	
TELEFONO	**:**	98215427
E-MAIL	**:**	marcelo.stockle1701@alumnos.ubiobio.cl
CARRERA	**:**	Ingeniería Civil Informática
DEPTO.	**:**	Sistemas de Información

NOMBRE	**:**	
DIRECCIÓN	**:**	
TELEFONO	**:**	
E-MAIL	**:**	
CARRERA	**:**	
DEPTO.	**:**	Sistemas de Información

2. TÍTULO QUE IDENTIFICARÁ LA ACTIVIDAD DE TITULACIÓN.

Un nuevo algoritmo eficiente para la búsqueda de patrones parciales frecuentes.

3. PROFESOR GUÍA.

NOMBRE	**:**	**Claudio Gutiérrez Soto**
FIRMA	**:**	**……**

4. PERSONAS, INSTITUCIONES O EMPRESAS EN QUE SE SOLICITARÁ APOYO Y ASESORÍA.

NOMBRE	**:**	
RUBRO	**:**	
FIRMA	**:**	**……**

5. NOMBRE DE LA PERSONA RESPONSABLE DE LA EMPRESA QUE SUPERVISARA AL ALUMNO.

NOMBRE	**:**	
CARGO	**:**	

6. OBJETIVOS GENERALES Y ESPECÍFICOS DE LA ACTIVIDAD DE TITULACIÓN.

Objetivos Generales:

Implementar una solución aproximada para la búsqueda de patrones periódicos y comparar su rendimiento contra algoritmos del estado del arte.

Objetivos Específicos:

- Identificar algoritmos de búsqueda aproximada en series de tiempo del estado del arte.
- Diseñar e implementar una solución aproximada para la búsqueda de patrones parciales frecuentes (BPPF).
- Obtener resultados empíricos utilizando una base de datos sintética y una base de datos real.
- Compilar y presentar los resultados en la forma de un artículo científico.

7. JUSTIFICACIÓN DEL PROYECTO PROPUESTO.

El rendimiento de las soluciones modernas para la búsqueda de patrones parciales frecuentes (BPPF) se ve limitado por su adherencia al algoritmo Apriori y el requerimiento de determinar con exactitud el soporte y confianza para determinar si un patrón periódico parcial (PPP) es efectivamente frecuente. Estas limitaciones restringen el rango posible de aplicaciones, como por ejemplo cualquier aplicación que requiera actualizaciones frecuentes del modelo de datos, comúnmente llamadas aplicaciones online.

Este estudio propone explorar un método de búsqueda aproximada capaz de descubrir PPP candidatos con suficientes garantías (no falsos positivos, pocos falsos negativos). Logrando esto, se espera que la solución implementada supere significativamente en rendimiento a los algoritmos exactos de BPP.

8. PLAN DE TRABAJO A DESARROLLAR.

N°	Actividad	Fecha inicio estimada	Duración
1	Identificar algoritmos de búsqueda aproximada en series de tiempo del estado del arte.	---	Concluido
2	Diseñar e implementar una nueva solución	14 de abr 2021	6 semanas
3	Implementar alternativas y comparar el desempeño	26 de may 2021	3 semanas
4	Recopilar y documentar resultados y conclusiones	16 de jun 2021	8 semanas

9. DESCRIPCIÓN DE LOS ASPECTOS FUNDAMENTALES DE LA METODOLOGÍA A UTILIZAR.

La búsqueda de patrones periódicos frecuentes (BPPF) es una forma de minería de datos cuyo propósito es descubrir patrones periódicos en una base de datos de series de tiempo.

El problema de descubrir patrónes periódicos parciales (PPPs) en una base de datos de series de tiempo es altamente complejo, puesto que el número de patrones por probar aumenta combinatorialmente con respecto al número de ítems (distintos valores que la serie de tiempo puede tomar en cada posición) y el largo, o periodo, del patrón. Soluciones convencionales involucran de una forma u otra al algoritmo Apriori para reducir dramáticamente el espacio de búsqueda.

En su implementación más ingenua, el algoritmo Apriori requiere en el peor caso hacer tantas lecturas a la base de datos como el largo del patrón. Debido a esto, esta implementación no es escalable a la búsqueda de patrones largos. Soluciones más modernas como *FP-growth* o *Max-subpattern*, sin embargo, emplean estructuras que optimizan la búsqueda y requieren a lo más dos lecturas de la base de datos.

Un aspecto común a estas soluciones es que todas determinan el soporte y la confianza de un PPP para determinar si este efectivamente es frecuente. Determinar ambos indicadores requiere una búsqueda exhaustiva de la base de datos, sin embargo, se puede argumentar que en la tarea de descubrir patrones interesantes existen instancias en que determinar con exactitud el soporte y la confianza no es un aspecto clave, siempre que los patrones descubiertos satisfagan algunas garantías mínimas.

10. TRABAJOS SIMILARES REALIZADOS PREVIAMENTE

Aunque es difícil encontrar estudios recientes de alto perfil que traten expresamente sobre la búsqueda de patrones periódicos frecuentes, se encuentran avances relevantes aplicados a campos como la biometría o a los sistemas de huella digital para contenido multimedia.

Unos de los antecedentes más importantes a estas aplicaciones existe en el trabajo de Edith Cohen [1] quien introdujo un método no convencional para la minería de reglas de asociación conocido hoy en día como *min-hashing*, que serviría para replantear la MPPF.

11. BIBLIOGRAFÍA A USAR.

Referencia Bibliográfica

[1] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey Ullman, y Cheng Yang. Finding interesting associations without support pruning. IEEE Transactions on Knowledge and Data Engineering, 13:64–78, 2001. doi:10.1109/69.908981

LA PRESENTE SOLICITUD DE INSCRIPCIÓN DE LA ACTIVIDAD DE TITULACIÓN SIGNIFICA UN COMPROMISO DE CUMPLIR LO ESTIPULADO EN ELLA.

.....	.....
FIRMA ALUMNO	**FIRMA ALUMNO**

FECHA PRESENTACIÓN SOLICITUD: .....

RESOLUCIÓN DIRECTOR DE ESCUELA / JEFE DE CARRERA
- **APROBADO** ☐ - **REPROBADO** ☐
OBSERVACIONES: ……..
.....
.....
.....
_____
FIRMA DIRECTOR DE ESCUELA
FECHA RESOLUCIÓN: ……