FP-FLOCK: UN ALGORITMO PARA EL DESCUBRIMIENTO DE PATRONES DE AGRUPACIÓN DE OBJETOS MÓVILES EN BASES DE DATOS ESPACIO TEMPORALES

OMAR ERNESTO CABRERA ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2014

FP-FLOCK: UN ALGORITMO PARA EL DESCUBRIMIENTO DE PATRONES DE AGRUPACIÓN DE OBJETOS MÓVILES EN BASES DE DATOS ESPACIO TEMPORALES

OMAR ERNESTO CABRERA ROSERO

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR: ANDRES OSWALDO CALDERON ROMERO, MSC.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2014

NOTA DE RESPONSABILIDAD

"Las ideas y conclusiones aportadas en la tesis de grado, son responsabilidad exclusiva de sus autores".

Articulo 1º del acuerdo Nº 324 del 11 de octubre de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño

"La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor"

Artículo 13, Acuerdo N. 005 de 2010 emanado del Honorable Consejo Académico.

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado

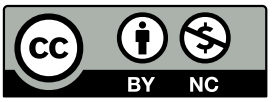
_____

Firma del jurado

_____

Firma del jurado

San Juan de Pasto, 2014



You are free:

to Share – to copy, distribute and transmit this work

to Remix – to adapt this work

Under the following conditions:

Attribution – You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work)

Noncommercial – You may not use this work for commercial purposes.

Subject to conditions outlined in the license.

Dedicado a:
Mis padres por darme su apoyo incondicional,
creer en mi y motivarme día a día.

Mi hermano y hermana por brindarme su apoyo y compañía.

Sandra Cristina Muñoz Castillo (My Elf Princess)
por haberme entregado su amor, lealtad y
darme siempre soporte, fuerza y esperanza.
"Lucas Poldrosky"

AGRADECIMIENTOS

Al Sistema de Investigaciones de la Universidad de Nariño, por haber financiado esta investigación.

A Andrés Oswaldo Calderón Romero, mi asesor y amigo, por su disposición, voluntad, compromiso y por haberme despertado el interés en el tema.

A Ricardo Timarán Pereira, por haberme abierto las puertas en el grupo de investigación GRIAS y así poder iniciar mi vida investigativa.

A todos mis amigos y amigas que han estado apoyándome de una u otra manera, en especial a Mateo Guerrero Restrepo, Maria Isabel Gómez y Alicia Guerrero.

A Jairo Guerrero Garcia, por ser mi profesor y mi amigo.

A todos los profesores, por sus enseñanzas y apoyo sincero a lo largo del pregrado.

A toda la comunidad de Software Libre por enseñarme el valor de compartir.

RESUMEN

El amplio uso de sistemas de localización como dispositivos GPS y RFID junto con el masivo uso dispositivos móviles han hecho que la disponibilidad y acceso a bases de datos espacio temporales se hayan incrementado de manera considerable durante los últimos años. Esta gran cantidad de datos ha motivado el desarrollo de técnicas más eficientes para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos móviles durante un periodo continuo de tiempo. Diversos estudios se han centrado en la consulta de patrones que capturan el comportamiento de los diversas entidades en movimiento, los cuales se reflejan en colaboraciones tales como clústers móviles, consulta de convoyes y patrones de agrupamiento. En esta investigación se da a conocer una propuesta para descubrir patrones de agrupamiento, tradicionalmente conocidas como flocks, la cual esta basada en un enfoque de patrones frecuentes. Se presenta el algoritmo FP-Flock para la detección de patrones tanto en línea como fuera de línea. Ambas alternativas fueron comparadas con dos algoritmos del mismo tipo, Basic Flock Evaluation (BFE) y LCMFlock. El desempeño y comportamiento se midió en distintos conjuntos de datos, tanto sintéticos como reales.

Palabras clave: patrones de agrupamiento, patrones frecuentes de minería, patrones de movimiento, base de datos espacio-temporal.

ABSTRACT

The widespread use of location systems such as GPS and RFID along with the massive use of mobile devices have allowed a significantly increasing in the availability and access to spatio-temporal databases in recent years. This large amount of data has motivated the development of more efficient techniques to process queries about the behavior of moving objects, like discovering behavior patterns among trajectories of moving objects over a continuous period of time. Several studies have focused on the query patterns that capture the behavior of the various entities in motion, which are reflected in collaborations such as mobile clusters, convoys query and clustering patterns. In this research, we provided an approach to find grouped patterns, traditionally known as flocks, which is based on an approach of frequent patterns. FP-Flock algorithm is presented for detecting patterns both online and offline are presented. Both alternatives were compared with two algorithms of the same type, Basic Flock Evaluation (BFE) and LCMFlock. The performance and behavior was measured in different datasets, both synthetic and real.

Keywords: flock patterns, frequent patterns mining, movement patterns, spatio-temporal databases.

CONTENIDO

ACRÓNIMOS

BFE Basic Flock Evaluation

CAD Computer Aided Design

DBMS Database Management System

FPFLock Frequent Pattern Flock

GPS Global Positioning System

LCM Linear time Closed itemset Miner

LCMFlock Linear time Closed item set Miner Flock

MOD Moving Objects Databases

RFID Radio Frequency IDentification

SUMO Simulation of Urban MObility

VLSI Very Large Scale Integration

INTRODUCCIÓN

Los recientes avances tecnológicos y el amplio uso de la localización basada en sistemas de posicionamiento global (GPS), identificación por radio frecuencia (RFID) o tecnologías en dispositivos móviles han hecho que el acceso a bases de datos espacio temporales se haya incrementado de una manera acelerada. Esta gran cantidad de información ha motivado a desarrollar técnicas más eficientes para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos en un período contínuo de tiempo.

Sin embargo, los métodos que existen para la consulta de trayectorias se centran principalmente en responder un único rango simple de predicado y consultas de vecinos más cercanos, por ejemplo: "encontrar todos los objetos en movimiento que se encontraban en la zona A a las 10 de la mañana" o "encontrar el coche que condujo cerca de la ubicación B durante el intervalo de tiempo de 10 de la mañana a 1 de la tarde". Recientemente, diversos estudios se han centrado en la consulta de los patrones para la captura del comportamiento de los objetos en movimiento reflejada en colaboraciones tales como clusters móviles [1] [2], consulta de convoyes [3] y patrones de agrupamiento [4] [5] [6] [7]. Estos patrones descubren grupos de objetos en movimiento que tienen una "fuerte" relación en el espacio durante un tiempo determinado. La diferencia entre todos esos patrones es la forma como se unen los objetos móviles entre intervalos de tiempo, su duración en el tiempo y la manera de cómo se combinan.

Esta investigación se enfoca en el descubrimiento de patrones de agrupamiento, conocidos como "flocks", entre los objetos en movimiento de acuerdo a las características de los objetos de estudio (animales, peatones, vehículos o fenómenos naturales), cómo interactúan entre sí y cómo se mueven juntos [8] [9].

La implementación de este tipo de análisis tiene diversas aplicaciones, tales como: sistemas integrados de transporte, seguridad y monitoreo, seguimiento a grupos de animales y fenómenos naturales. El analizar como se mueven los objetos en la tierra y cuáles son los patrones de comportamiento que existen entre sí puede aportar valiosa información para dar solución a diversos problemas en el mundo.

En este proyecto se propone un nuevo algoritmo llamado FP-Flock el cual se compara con los algoritmos propuestos por [6] y [10]. Se realizaron pruebas con distintos conjuntos de datos, tanto reales como sintéticos. Se escogieron únicamente estos algoritmos para la comparación debido a que este análisis está enfocado a patrones de agrupamiento (flocks) y estos son los algoritmos más representativos que existen hasta el momento en este campo. Además, las implementaciones de los algoritmos mencionados están disponibles en [11].

Planteamiento del problema

Se han propuesto algoritmos con el fin de descubrir patrones de agrupación analizando conjuntos de datos con el fin de encontrar los discos que se pueden unir entre instantes de tiempo consecutivos. El número de discos posibles en un intervalo de tiempo dado puede ser bastante grande y el costo de unir los discos entre los intervalos de tiempo puede llegar a ser muy alto. Manejar y analizar todas las combinaciones posibles tiene un impacto directo en el rendimiento de los algoritmos. [6], [5] han probado algunas heurísticas y aproximaciones encaminadas a reducir el número de discos evaluados. [10] define una metodología para extraer patrones de agrupación en conjuntos de datos de trayectoria basados en patrones frecuentes, el cual es un tema frecuentemente abordado en la minería de datos, con el objetivo de hacer frente a los inconvenientes antes mencionados.

Sin embargo, el número y la calidad de los patrones descubiertos hacen particularmente difícil realizar una correcta interpretación de los resultados. Igualmente, en [6] los resultados experimentales presentados en los anteriores trabajos muestran que todavía existe una alta complejidad computacional a la hora de efectuar la combinación de los discos. Esto repercute en altos tiempos de respuesta y un alto número de patrones resultantes que hace particularmente dificil su interpretacion. Inclusive, en [10] es necesario aplicar una etapa de postprocesamiento para descartar patrones redundantes.

En esta investigación se propone un algoritmo con el fin de encontrar el número de discos posibles en un intervalo de tiempo, mejorar la combinación de los mismos y por ende su rendimiento sin perder la calidad del patrón encontrado. El algoritmo propuesto será evaluado y comparado con los algoritmos propuestos en [6] y [10].

Objetivo general

Proponer un algoritmo para el descubrimiento de patrones de agrupación de objetos

móviles en bases de datos espacio temporales.

Objetivos Específicos

- Apropiar el conocimiento en bases de datos espacio temporales, trayectorias y patrones de agrupamiento en objetos móviles.
- Implementar y evaluar los algoritmos seleccionados con el fin de identificar los problemas asociados a su rendimiento y su posterior comparación con la propuesta realizada.
- Diseñar el algoritmo para el descubrimiento de patrones de agrupación en objetos en movimiento.
- Comparar los tiempos de respuesta y la claridad de los patrones obtenidos del algoritmo propuesto con respecto a los algoritmos existentes que fueron seleccionados.

Organización del documento

En el capítulo uno, se presenta la fundamentación teórica. En el capítulo dos, los trabajos realacionados sobre patrones de agrupamiento en objetos móviles. En el capítulo tres, se presenta la metodología utilizada en la investigación. En el capítulo cuatro, se describe el algoritmo FP-Flock con sus dos variaciones, una en línea y otra fuera de línea. En el capítulo cinco, se presenta la experimentación computacional utilizando base de datos sintéticas y reales. En el capítulo seis, se muestra la discución sobre la experimentación computacional. En el capítulo siete, se presentan las conclusiones y trabajos futuros, y por último se muestran las referencias bibliográficas y los anexos.

1 FUNDAMENTOS TEÓRICOS

1.1 BASE DE DATOS ESPACIO-TEMPORALES

Dos campos que han surgido a mediados y finales de la década de 1980 son las bases de datos espaciales y bases de datos temporales las cuales se han convertido, en cierto modo como las precursoras de las bases de datos espacio-temporales.

Base de datos espaciales

Las bases de datos espaciales tratan del eficiente almacenamiento y la recuperación de objetos en el espacio que tienen identidad y extensión bien definidas, ubicaciones, así como ciertas relaciones geométricas y / o topológica entre ellas, debido al desarrollo en campos de aplicación (GIS, diseño VLSI, CAD) que necesita tratar con grandes cantidades de entidades geométricas, geográficas o espaciales. Además de algunos estables y maduros prototipos basado en fundamentos algebraicos de tipo solido, proveedores de sistemas gestores de bases de datos comercial (DBMS) han ofrecido extensiones de sus productos, soportando tipos espaciales y operaciones (Oracle Spatial, DB2 Spatial Extender, PostgresGIS , Microsoft SQL Server, MySQL). Sin lugar a dudas, los resultados en bases de datos espaciales han impulsado varias líneas de investigación importantes en bases de datos de objetos en movimiento (MOD), como por ejemplo: consultas populares de MOD (rango, vecino mas cercano), propiedades topológicas y relaciones entre tipos espaciales. [12]

Base de datos temporales

Muchas aplicaciones de bases de datos, como por ejemplo, contabilidad, gestión de cartera, registros médicos y gestión de inventario; información que varia en el tiempo. El centro de las bases de datos temporales se diferencia por:

Tiempo válido de un hecho, es el tiempo en el cual un dato particular es recogido y se convierte en verdad hasta donde el mundo representado por la base de datos es referida, posiblemente abarque el pasado, el presente y el futuro. Sin embargo, el tiempo válido no puede ser conocido, o los registros no pueden ser relevantes

para las aplicaciones soportadas por la base de datos o en el caso que los modelos de base de datos del mundo real puedan variar entre los diferentes mundos reales.

Tiempo de transacción de un hecho, es el tiempo que un hecho dado es el actual en la base de datos. El tiempo de transacción puede asociarse con diferentes entidades de bases de datos, como, por ejemplo, objetos y valores que no son hechos, porque no pueden ser verdaderos o falsos en forma aislada. Por lo tanto, todas las entidades de la base de datos tiene un aspecto de tiempo de transacción, que tiene una duración: desde inserción (lógico) a la eliminación de una determinada entidad.

Capturar el tiempo variable utilizando los modelos de datos tradicionales y lenguajes de consulta puede ser una actividad engorrosa y, como consecuencia, se necesitan construcciones que permitirán capturar los tiempos válidos y operación de los hechos, lo que conduce a relaciones temporales. Además, en los lenguajes de consulta se necesitan extensiones sintácticas que permiten las operaciones de base de datos en modelos temporales. [12]

1.2 TRAYECTORIAS

Desde el punto de vista de los usuarios, el concepto de trayectoria se basa en la evolución de la posición de un objeto que se desplaza en el espacio durante un intervalo de tiempo dado. Por lo tanto, la trayectoria es por definición un concepto espacio-temporal. Pero mientras este en movimiento puede ser visto como una característica de algunos de los objetos que los diferencia de los objetos que no están en movimiento (por ejemplo, edificios, carreteras), el concepto que un objeto se desplace implica que su movimiento está destinado a cumplir una meta significativa que requiere desplazarse de un lugar a otro. Desplazarse para alcanzar un objetivo tiene una cantidad finita de tiempo (y cubre una cierta distancia en el espacio), por lo tanto, las trayectorias son inherentemente definidas por un intervalo de tiempo. Este intervalo de tiempo está delimitado por el instante en que el objeto comienza un desplazamiento (llamado t_{inicio}) y el instante en que el recorrido termina (t_{fin}). Identificar t_{inicio} y t_{fin} dentro del marco de tiempo del conjunto en que el objeto se mueve es una decisión de la aplicación, es decir, una especificación basado en el usuario. La siguiente definición formal define un punto de trayectoria en movimiento en una perspectiva de base de datos.

Definición 1 (Trayectoria) Una trayectoria es el registro de usuario definido de la evolución de la posición (pervivida como un punto) de un objeto que esta en movimiento en el espacio durante un intervalo de tiempo dado con el fin de conseguir un objetivo dado.

$$[t_{inicio}, t_{fin}] \rightarrow \text{Espacio}$$

La función del espacio de tiempo es definida por el usuario y no es necesariamente el proporcionado por el mecanismo de adquisición de datos, cuya forma general es una secuencia de pares (punto de muestra, tiempo). Los datos sin procesar a menudo tiene que someterse a un proceso de limpieza para corregir errores y aproximaciones en la adquisición de datos. Además, la aplicación puede estar interesada en sólo un subconjunto de los puntos limpios de la muestra, por ejemplo, omitiendo puntos adquiridos durante la noche y únicamente guardar movimiento en el día o sustitución de una secuencia de puntos irrelevantes (desde la perspectiva de la aplicación) con un punto único representante (por ejemplo, para la representación de paradas como un punto único).

Otra diferencia importante entre un objeto con trayectorias y un objeto en movimiento es que, para un objeto en movimiento, la función de espacio de tiempo que describe su posición se define sobre toda la vida útil del objeto. En su lugar, una trayectoria se da mediante la restricción de la función a un intervalo de tiempo específico, $[t_{inicio}, t_{fin}]$, incluido en el ciclo de vida del objeto. Una trayectoria es un segmento de la ruta espacio-temporal cubierta por un objeto en movimiento (Figura 1.1). Por consiguiente, durante su vida útil de un objeto puede desplazarse un número de trayectorias, una después de la otra. Por ejemplo, un pájaro tiene una trayectoria de migración en la primavera de 2006, desde África a Europa, seguida de otra trayectoria en otoño de 2006 desde Europa a África, seguida de otras trayectorias en 2007, etc.

Para la Figura 1.1, una ruta espacio-temporal para un objeto en movimiento en curso y sus muchas trayectorias definidas por una segmentación semántica de la ruta. En este ejemplo partes del camino no pertenece a ninguna trayectoria, corresponden al movimiento que es irrelevante a la aplicación. Los objetos que se desplazan no necesariamente se mueven continuamente durante una trayectoria. En consecuencia, las trayectorias pueden ser ellas mismas segmentadas semánticamente mediante la definición de una secuencia temporal de sub-intervalos de tiempo donde alternativamente la

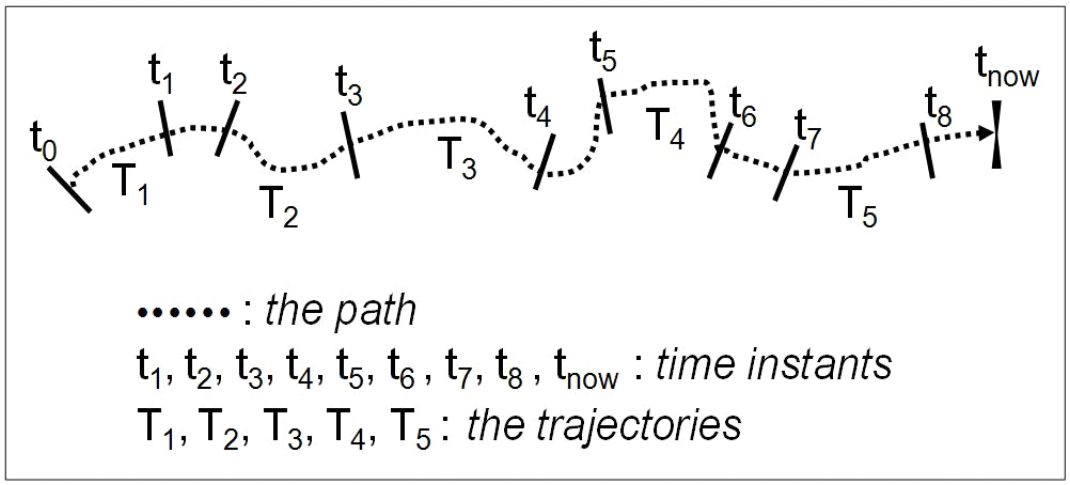


Figura 1.1: Ruta Espacio Temporal [13]).

posición del objeto cambia y permanece fija. La primera se llama movimiento y la segunda parada. Podemos ver entonces una trayectoria como una secuencia de movimientos que van de una parada a la siguiente (o como una secuencia de paradas que separan los movimientos). Por ejemplo, un ave que se ha apartado de la migración hará una parada en algún lugar durante algún tiempo para alimentarse, otra parada para descansar, y así sucesivamente hasta que se alcanza el final de su trayectoria. Vendedores en viaje de negocios se detendrán en todas las localidades donde planeaban encontrarse con un cliente.

Identificar paradas (y movimientos) dentro de una trayectoria es la responsabilidad de la aplicación. Paradas físicas (es decir, el hecho de que la posición del objeto es el mismo durante dos o más instantes consecutivos) no se tienen en cuenta para las paradas conceptuales, ya que puede ser debido a eventos que son irrelevantes para la aplicación. Por ejemplo, la parada hecha por vendedor para beber un café es irrelevante para la aplicación de seguimiento de la empresa. En su lugar, la parada de hecho para cumplir con un cliente es significativo. La aplicación podría estar interesado en contar el número de paradas por trayectoria, y, obviamente, las paradas que se cuentan son sólo las paradas importantes. Se asume que se mueve y se para completamente para cubrir la trayectoria (es decir, no hay instante en $[t_{inicio}, t_{fin}]$ que no pertenezca ni a un movimiento ni a una parada).

Se definen semánticamente paradas y movimientos de la siguiente manera:

Definición 2 (Parada) Una parada es una parte de una trayectoria, de tal manera que:

El usuario ha definido explícitamente que es parte de la trayectoria ([$t_{inicioparadax}$, $t_{finparadax}$]) para representar una parada. La extensión temporal [$t_{inicioparadax}$, $t_{finparadax}$] es un intervalo de tiempo no vacía, y el objeto que se desplaza no se mueve (por lo que la vista de la aplicación de esta trayectoria se refiere), es decir, el rango espacial de la trayectoria para el intervalo [$t_{inicioparadax}$, $t_{finparadax}$] es de un solo punto. Todas las paradas están temporalmente disjuntas, es decir, la extensión temporal de dos paradas son siempre disjuntos.

Definición 3 (Movimiento) Un movimiento es una parte de una trayectoria, de tal manera que:

La parte está delimitada por dos extremidades que representan ya sea dos consecutivas paradas, o t_{inicio} y la primera parada, o la última parada y t_{fin}, o [t_{inicio}, t_{fin}]].

La extensión temporal [$t_{iniciomovimientox}$, $t_{finmovimientox}$] es un intervalo de tiempo no vacía,y el rango espacial de la trayectoria para el intervalo [$t_{iniciomovimientox}$, $t_{finmovimientox}$] es la línea espacio-temporal (no un punto) definido por la función de trayectoria (de hecho, es la línea poligonal construida sobre los puntos de muestreo en el intervalo [$t_{iniciomovimientox}$, $t_{finmovimientox}$]).

Desde el punto de vista de diseño de base de datos, un movimiento es un punto de tiempo variable definida en el intervalo de tiempo [$t_{iniciomovimientox}$, $t_{finmovimientox}$]. [13]

1.3 IDENTIFICACIÓN DE PATRONES EN OBJETOS EN MOVIMIENTO

Debido a la creciente disponibilidad de bases de datos espaciales, se han propuesto diferentes metodologías con el fin de encontrar información significativa que permanece oculta en este tipo de datos. Tratar de comprender cómo diversas entidades se mueven en un contexto espacial, han demostrado ser útiles en temas tan diversos como el deporte [14], la geografía socioeconómica [15], la migración de los animales [16] y la seguridad y la vigilancia [17] [18].

Los primeros enfoques de recuperación de información de bases de datos espacio-temporales incluyen para esto consultas destinadas a responder gama simple de predicado o consultas de vecinos más cercanos, por ejemplo, “encontrar todos los objetos en movimiento dentro de la zona A entre las 10 a.m. a las 02:00 PM” o "¿Cuántos coches condujeron entre la plaza principal y el aeropuerto el viernes". Extensiones de consultas espaciales en paquetes comunes de software de SIG y DBMS que son capaces de ejecutar este tipo de de consultas, sin embargo estas técnicas tratan de encontrar la mejor solución a explorar cada objeto espacial a la vez de acuerdo con alguna métrica de distancia (por lo general euclidiana). Como resultados, es difícil de capturar el comportamiento colectivo y las correlaciones entre las entidades involucradas utilizando este tipo de consultas.

Recientemente, han surgido unos nuevos intereses para la consulta de captura de patrones de “grupo” o conducta “común” entre las entidades en movimiento. De particular interés es el desarrollo de enfoques para identificar grupos de objetos en movimiento cuya participación en una relación fuerte y la interacción en una región espacial definida durante una duración de tiempo determinado. Algunos ejemplos de este tipo de enfoques son movimiento de cluster [2] [1], consultas de convoy [3] y patrones de agrupamiento [4] [5] [19] [7].

Aunque diferentes interpretaciones pueden ser tomadas, un patrón de agrupamiento se refiere a un número predefinido de entidades que permanecen lo suficientemente cerca durante al menos un intervalo de tiempo dado. El desafiar a identificar este tipo de patrones de movimiento es especialmente relevante debido a la interacciones intrínsecas entre los miembros del grupo, especialmente en el contexto de los animales, peatones o vehículos. En esta investigación un marco alternativo para descubrir en movimiento acuden patrones se propone. Esta investigación esta basada en un algoritmo existente denominado BFE propuesto por [19], y otro llamado LCMFlock propuesto [7] el cual extiende el algoritmo de BFE tomando ventaja de algoritmos de patrones frecuentes de minería en el area del aprendizaje de reglas de asociación.

1.4 PATRÓN DE AGRUPAMIENTO

[6] define patrones de agrupamiento también conocidos como flocks, como el problema de identificar todos los grupos de trayectorias que permanecen “juntas” por la duración

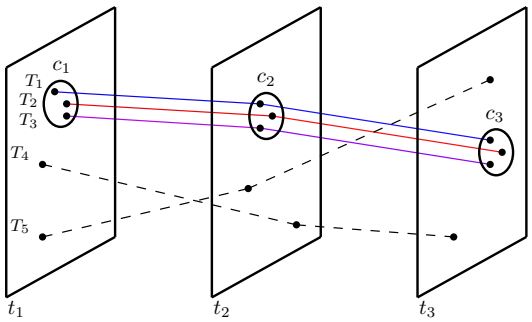


Figura 1.2: Ejemplo de patrones de agrupamiento [6].

de un intervalo de tiempo dado. Se Considera que los objetos en movimiento están suficientemente cerca si existe un disco con un radio dado que cubre todos los objetos que se mueven en el patrón (figura 1.2). Una trayectoria satisface el patrón anterior, siempre y cuando suficientes trayectorias están contenidos dentro del disco para el intervalo de tiempo especificado, es decir, la respuesta se basa no sólo en el comportamiento de una trayectoria dada, sino también en las más cercanas a ella. Uno de los enfoques para descubrir patrones móviles de agrupamiento consiste en encontrar un conjunto adecuado de discos en cada instante de tiempo y luego combinar los resultados de un instante de tiempo a otro. Como consecuencia, el rendimiento y el número de patrones encontrados depende del número de los discos y cómo éstos se combinan.

En el ejemplo de la figura 1.2, se muestra un patrón de agrupamiento el cual contiene tres trayectorias (T1, T2 y T3) que están dentro de un disco en tres instantes de tiempo consecutivos. Los discos se pueden mover libremente en el espacio bidimensional con el fin de acomodar los tres objetos en movimiento y su centro no necesariamente tiene que ser la localización de alguno de los objetos. Esto hace que el descubrimiento de patrones sea mucho más complicada porque hay un número infinito de posibles colocaciones del disco en cualquier instante de tiempo y el posible número de combinaciones puede llegar a ser muy alta y costosa.

1.5 BFE (BASIC FLOCK EVALUATION)

Este algoritmo en línea presentado por [19], al parecer es el primer trabajo que presenta una solución exacta para reportar patrones de agrupamiento (flocks) en tiempo polinomial.

Un patron de agrupamiento Flock(μ, ε, δ), esta definido de la siguiente manera:

Definición: Dado un conjunto de trayectorias τ , un mínimo número de trayectorias $\mu > 1(\mu \in N)$, una máxima distancia $\epsilon > 0$ definida sobre la función de distancia d, y una duración de tiempo mínimo $\delta > 1(\delta \in N)$. Un patron de agrupamiento Flock(μ, ϵ, δ) reporta todas las colecciones máximas de tamaño F de trayectorias donde: para cada f_k en F, el número de trayectorias en f_k es mayor o igual que $\mu(|f_k| \geq \mu)$ y existen instancias de tiempo consecutivos δ tal que para cada $t_i \in [f_k^{t_1}..f_k^{t_1+\delta}]$, hay un disco con centro $c_k^{t_i}$ y el radio $\epsilon/2$ que cubre todo los puntos en $f_k^{t_i}$. Es decir: $\forall f_k \in F, \forall t_i \in [f_k^{t_1}..f_k^{t_1+\delta}], \forall T_j \in f_k : |f_k^{t_i}| \geq \mu, d(p_j^{t_i}, c_k^{t_i}) \leq \epsilon/2$

1.6 PATRONES FRECUENTES EN BASE DE DATOS TRADICIONALES

Patrones frecuentes son conjuntos de elementos, subsecuencias, o subestructuras que aparecen en un conjunto de datos con una frecuencia no inferior a un umbral especificado por el usuario [20]. La cuestión de las modalidades interesantes desvelando en las bases de datos en diferentes contextos ha sido un tema de investigación recurrente durante los últimos 15 años. Minería de datos general ha sido ampliamente reconocido como un crítico campo por empresas de todo tipo. Como parte de los métodos de minería de datos, la tarea de aprendizaje de reglas de asociación han estudiado diferentes algoritmos de minería de patrones frecuente de identificar tendencias relevantes en los conjuntos de datos en diferentes disciplinas [21] [22] [23].

Una de las áreas en las que las técnicas de aprendizaje de reglas de asociación y el patrón frecuente algoritmos de minería se han aplicado con más frecuencia en el análisis de datos y tendencias del mercado en transacciones de clientes de grandes supermercados y tiendas [24]. Por lo general, esta técnica tiene dado el nombre del problema de la cesta de compras a pesar de que los métodos derivados de resolverlo puede ser aplicado en diferentes contextos [25].

El problema de la cesta de compras representa un intento por parte de un minorista para descubrir qué artículos de sus clientes compran con frecuencia juntos [26]. El objetivo es la comprensión de la comportamiento de un cliente típico y la identificación de los objetos de valor y relaciones entre ellos. Para este tipo de problema de la entrada es una base de datos con la información dada acerca de los artículos comprados. Cuando

un cliente paga por sus productos en el cajero, un registro con los artículos comprados se inserta en la base de datos. En una vista general, es suficiente para capturar sólo el ID de la transacción y la identificación del producto (un registro por cada artículo comprado). Se le conoce como TID: conjunto de elementos esquema. Como los registros de la base de datos por lo general se refieren a transacciones, estas bases de datos se denominan bases de datos transaccionales. El objetivo de las compras análisis de la cesta es encontrar conjuntos de elementos (conjuntos de elementos) que están "asociados" y el hecho de su asociación a menudo se llama una regla de asociación [26].

Por ejemplo, si se sabe que un alto porcentaje de clientes están comprando leche y pan al mismo tiempo, en sus visitas a un supermercado, esta relación representa una regla de asociación. Se puede utilizar para formular nuevas estrategias de marketing, promociones, introducción de nuevos productos, diseño de catálogo, cruces de marketing o planificación de espacio en las estanterías [25]. Es habitual localizar elementos asociados en diferentes pasillos y de alta rentabilidad o nuevos productos entre ellos para asegurarse de que están expuestos a más clientes [26]. [27] analizan otros casos de estudio aplicados en el comercio y el marketing, donde se exploran métodos diferentes reglas de asociación. durante los últimos años muchas mejoras y nuevas técnicas se han desarrollado y propuesto con el fin de mejorar y tomar ventaja de los beneficios del análisis de reglas de asociación.

1.7 LCM (LINEAR TIME CLOSED ITEMSET MINER)

El problema de LCM propuesto por [9] se define de la siguiente manera.

Sea I un conjunto de elementos. Sea D una base de datos transaccional de tal manera que cada registro (llamada transacción) es un conjunto de elementos. La frecuencia de un conjunto de elementos es el número de transacciones, incluyendo el conjunto de elementos. Para un número dado t (llamado soporte), un conjunto de elementos se dice que es frecuente si su frecuencia es no menos de t. Un conjunto de elementos frecuente se llama máxima si está incluido en ningún otro conjunto de elementos frecuentes, y se llama cerrada si está incluido en ningún otro conjunto de elementos de la misma frecuencia. La tarea de LCM, es enumerar (sacar, o contar) todos los conjuntos de elementos frecuentes, todos los conjuntos de elementos frecuentes máximos, o todos los conjuntos de elementos frecuentes cerrados en una base de datos transaccional para un

soporte dado.

2 TRABAJOS RELACIONADOS

Las capacidad de recolectar datos de objetos en movimiento ha ido aumentando rápidamente y el interés de consulta de patrones que describen el comportamiento colectivo también ha aumentado. [6] enumera tres grupos de patrones "colectivos" en bases de datos de objetos en movimiento: clústers móviles, consulta de convoyes y patrones de agrupamiento.

Los clústers móviles [1] [2] [28] y consultas de convoyes [3] [29], tienen en común que se basan en algoritmos de clústering, principalmente en algoritmos basados en densidad como el algoritmo DBSCAN[30].

Los clústers móviles se definen entre dos instantes de tiempo consecutivos. Los clústers se pueden unir sólo si el número de objetos comunes entre ellos están por encima del parámetro predefinido. Un clúster es reportado si no hay otro nuevo clúster que pueda ser unido a éste. Este proceso se aplica cada vez para todos los instantes de tiempo en el conjunto de datos.

Las consultas de convoyes se definen como un clúster denso de trayectorias que permanecen juntas al menos por un tiempo contínuo predefinido.

Las principales diferencias entre las dos técnicas son la forma en que se unen los grupos entre dos intervalos consecutivos de tiempo y el uso de un parámetro adicional para especificar un tiempo mínimo de duración. Aunque estos métodos están estrechamente relacionados con los patrones de agrupamiento, ninguno de ellos asume una forma predefinida.

Previos trabajos de detección de patrones de agrupamiento móviles son descritos por [4] y [5]. Ellos introducen el uso de discos con un radio predefinido para identificar grupos de trayectorias que se mueven juntos en la misma dirección, todas las trayectorias que se encuentran dentro del disco en un instante de tiempo particular se considera un patrón candidato. La principal limitación de este proceso es que hay un número infinito de posibles ubicaciones del disco en cualquier instante de tiempo. En efecto, en [4] se ha demostrado que el descubrimiento de agrupaciones fijas, donde los patrones de las

mismas entidades permanecen juntas durante todo el intervalo, es un problema NP-complejo.

[6] son los primeros en presentar una solución exacta para reportar patrones de agrupación en tiempo polinomial, y también pueden trabajar efectivamente en tiempo real. Su trabajo revela que el tiempo de solución polinomial se puede encontrar a través de la identificación de un número discreto de ubicaciones para colocar el centro del disco. Los autores proponen el algoritmo BFE (Basic Flock Evaluation) basado en el tiempo de unión y combinación de los discos. La idea principal de este algoritmo es primero encontrar el número de discos válidos en cada instante de tiempo y luego combinarlos uno a uno entre tiempos adyacentes. Adicionalmente, se proponen otros cuatro algoritmos basados en métodos heurísticos, para reducir el número total de candidatos a ser combinados y, por lo tanto, el costo global del algoritmo. Sin embargo, el pseudocódigo y los resultados experimentales muestran todavía una alta complejidad computacional, largos tiempos de respuesta, debido a que este algoritmo usa δ para partir los flocks hace que el número de patrones encontrados sea mayor y esto hace difícil su interpretación.

[10] y [7] proponen una metodología que permite identificar patrones de agrupamiento utilizando tradicionales y potentes algoritmos de minería de datos usando patrones frecuentes, el cual fue comparado con BFE demostrando un alto rendimiento con conjuntos de datos sintéticos, aunque con conjuntos de datos reales el tiempo de respuesta siguió siendo eficiente pero similar a BFE. Este algoritmo trata el conjunto de trayectorias como una base de datos transaccional al convertir cada trayectoria, que se define como un conjunto de lugares visitados, en una transacción, definida como un conjunto de ítems. De esta manera, es posible aplicar cualquier algoritmo de reglas de asociación y encontrar patrones frecuentes sobre el conjunto dado, este algoritmo hace un llamado a LCM propuesto por [31] para encontrar patrones máximos y eso permite encontrar los flocks más largos.

3 METODOLOGÍA

Para poder realizar esta investigación, primero se realizó una apropiación de conocimiento y se definió el marco conceptual sobre el cual se iba a realizar el proyecto. Primero fue importante entender el problema que se desarrollaría, identificando los patrones en objetos en movimiento, para posteriormente enfocarse en los patrones de agrupamiento conocidos como "flocks".

Se compararon dos algoritmos, BFE y LCMFlock propuestos por [19] y [7], con el fin de identificar los problemas asociados a su rendimiento. Se escogió únicamente estos dos algoritmos para la comparación debido a que este análisis se enfocaría únicamente a patrones de agrupamiento (flocks) y estos son los algoritmos más representativos que existen hasta el momento en este campo.

Para realizar esta comparación se implementaron los algoritmos BFE y LCMFLOCK basados en el pseudo-código publicado por [6] y [10] respectivamente, usando Python versión 3 debido a la facilidad y comodidad para el programador. Realizar esta implementación tenía como objetivo hacer una inspección más detallada de cada algoritmo y de las estructuras utilizadas en la implementación con el fin de encontrar sus inconvenientes y buscar una mejora. El código fuente se puede descargar desde el repositorio del proyecto [1].

A continuación se describen los detalles de la implementación de los algoritmos BFE y LCMFlock.

3.1 IMPLEMENTACIÓN DE BFE

Este algoritmo se divide en dos partes: la primera parte, encontrar los discos dado un radio (ϵ) y un número mínimo de puntos (μ) para cada instante de tiempo. La segunda parte, encontrar el número de puntos que permanecen juntos (flocks) durante un rango de tiempo (δ).

[1] Repositorio del proyecto: `https://github.com/poldrosky/FPFlock`

Para la primera parte es necesaria la utilización tanto de diccionarios de datos como estructuras kd-tree para la búsqueda del vecino más cercano, en esta implementación se usó la clase scipy.spatial.cKDTree de SciPy [2] la cual proporciona un índice dentro de un conjunto de puntos k-dimensionales que se pueden utilizar para buscar rápidamente los vecinos más cercanos de cualquier punto.

Para realizar este proceso de implementación fue de gran utilidad el uso del software QGIS, un sistema de información geográfica libre y de código abierto disponible en [32], debido a que la primera parte del algoritmo esta basada en encuentrar de los discos máximos, se utilizo QGIS para realizar pruebas unitarias a medida que se avanzaba en las etapas de depuración, como lo muestra la figura 3.1.

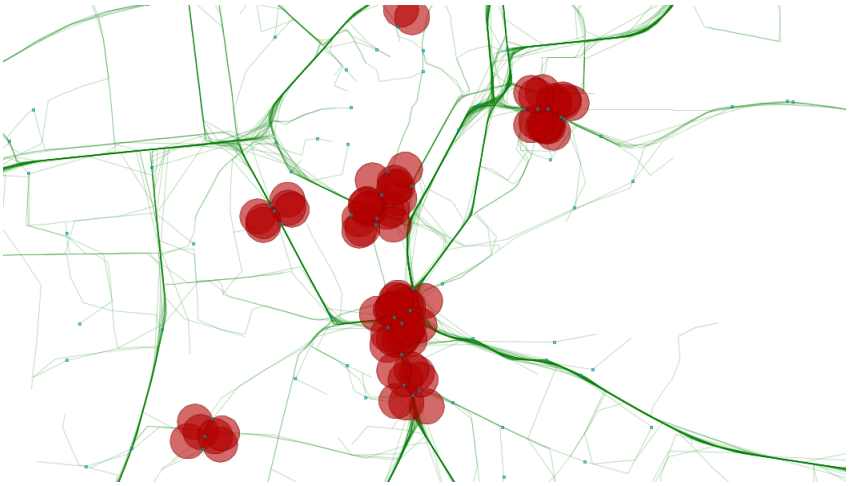
Los conjuntos de discos para cada instante de tiempo se almacenaron en diccionarios los cuales fueron combinados en la segunda parte del algoritmo. Finalmente, se almacenaron en una base de datos los flocks que cumplían los parámetros solicitados. Para cada flock se almacenó un arreglo con los objetos que pertenecen a dicho flock, y los tiempos de inicio y final para cada caso.

3.2 IMPLEMENTACIÓN DE LCMFLOCK

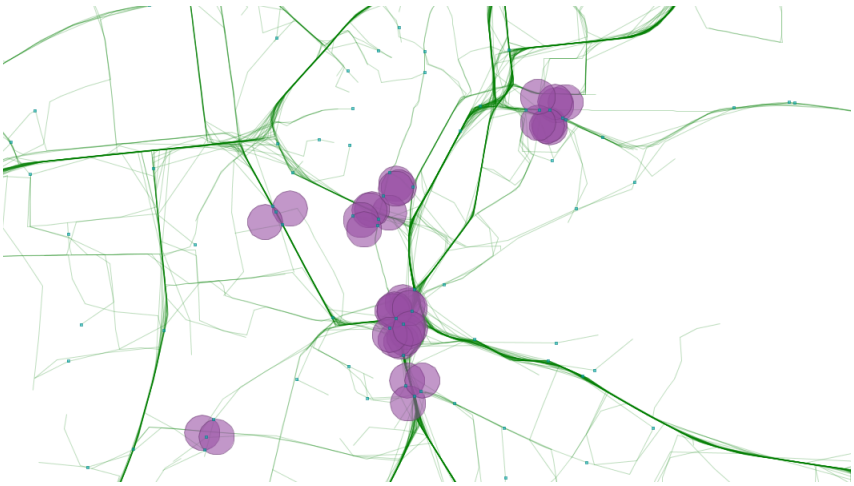
Este algoritmo usa la primera parte del algoritmo de BFE para encontrar los discos. En la segunda parte, para abordar el problema de combinatoria utiliza un enfoque de patrones frecuentes, en el cual se construyó un diccionario de datos asociando la localización de los puntos de cada trayectoria con su respectivo disco generado una versión transaccional del conjunto de datos. En la figura 3.2, se muestra como se realiza conversión de las trayectorias en transacciones. Este conjunto es pasado como parámetro, junto con el número mínimo de puntos (μ), al algoritmo LCM[31], disponible para descargar en [33].

Aquí se encuentran disponibles dos variantes del programa; LCM_max y LCM_closed los cuales recuperarán el conjunto de patrones máximo y cerrado[34]. LCMFLOCK utiliza el concepto de patrones máximos y cerrados para identificar los patrones de agrupamiento de mayor duración. De esta manera, el parámetro δ se utiliza solo para filtrar aquellos patrones que no cumplan con este parámetro [10].

[2]Scipy es un ecosistema basado en Python, software de código abierto para las matemáticas, la ciencia y la ingeniería. `http://www.scipy.org/`



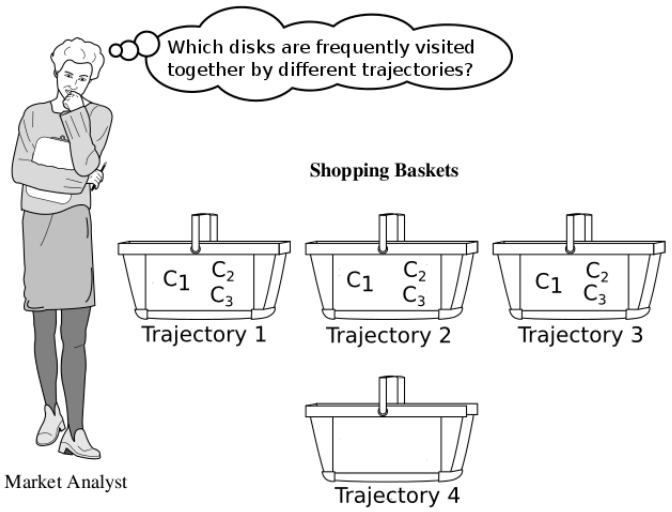
(a) Todos los discos



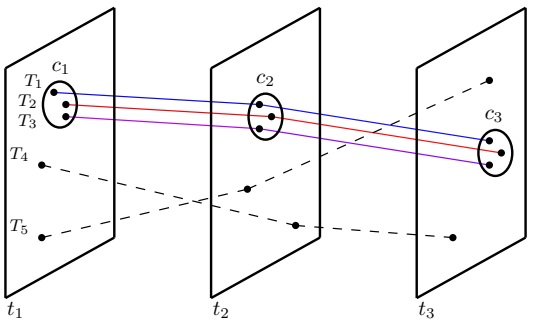
(b) Discos que cumplen μ mínimo

(c) Discos máximos

Figura 3.1: Proceso para encontrar los discos máximos [10].



(a) [10]



(b) [6]

Figura 3.2: Ejemplos de patrones frecuentes en trayectorias.

Tabla 3.1: Conjunto de datos validación

Dataset	Red	Número de Trayectorias	Número de Flocks	Instantes de tiempo
SJ5000T100t100f	Sintética	5000	100	100
SJ5000T100t200f	Sintetica	5000	200	100
SJ5000T100t300f	Sintetica	5000	300	100
SJ5000T100t400f	Sintetica	5000	400	100
SJ5000T100t500f	Sintetica	5000	500	100
SJ2500T100t500f	Sintética	2500	500	100
SJ7500T100t500f	Sintética	7500	500	100
SJ10000T100t500f	Sintética	10000	500	100
SJ12500T100t500f	Sintética	12500	500	100
SJ15000T100t500f	Sintética	15000	500	100
SJ17500T100t500f	Sintética	17500	500	100
SJ20000T100t500f	Sintética	20000	500	100

La salida de LCM es un archivo de texto, donde cada línea es un patrón que contiene un conjunto de ID's de discos separados por espacios que representan los patrones de agrupamiento (flocks). Sin embargo, se requiere de un análisis posterior para verificar que dichos discos ocurran en tiempos consecutivos. De cada patrón, se extraen los objetos que encierra cada disco así como los tiempos de inicio y final los cuales son almacenados en una base de datos.

3.3 VALIDACIÓN

Para poder validar la correcta implementación de los algoritmos se siguió una metodología similar a la propuesta en [5]. Se crearon conjuntos de datos sintéticos a los cuales se les insertó aleatoriamente un número específico de trayectorias y flocks. La tabla 3.1, relaciona los conjuntos de datos construidos y con los cuales los algoritmos fueron validados. Una copia de los conjuntos de datos y el script utilizado para la validación está disponible en el repositorio del proyecto [3]. El total de flocks insertados en cada caso fueron correctamente descubiertos por las dos implementaciones.

[3]Conjuntos de prueba: `https://github.com/poldrosky/FPFlock/tree/master/Src/Datasets/`

3.4 ANÁLISIS DE BFE y LCMFLOCK

Se realizó un análisis de desempeño entre los algoritmos de BFE y LCMFLlock, usando varios conjuntos de prueba, este análisis se puede detallar en [11]. Algunas de las conclusiones de este análisis, fueron:

- Las estructuras de tipo árbol, y en específico, los kd-tree presentaron los mejores resultados. Esto se configura como un aspecto clave para la búsqueda del conjunto final de discos para cada instante de tiempo.
- Aunque el proceso de identificación de los discos en cada instante de tiempo es compartido por ambos algoritmos, el proceso de combinación es mucho más eficiente utilizando un enfoque basado en técnicas de patrones frecuentes.
- Aunque LCMFLock presenta mejores resultados, este aún se ve afectado por el proceso de identificación de los discos. El número de discos a evaluar crece exponencialmente a medida que crece el valor ε de y la densidad de puntos dentro del conjunto.
- En BFE, el reporte de flocks es demasiado alto en comparación con LCMFlock. BFE separa los flocks de acuerdo al parámetro δ en procura de limitar el número de discos a comparar. Esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija. En LCMFlock esto se soluciona con el uso del algoritmo LCM para la detección de patrones máximos y cerrados.
- La capacidad de encontrar los flocks más largos ciertamente es una ventaja de LCMFlock sobre BFE. Sin embargo, LCMFlock, a diferencia de BFE, requiere de una ventana fija de tiempo donde será aplicado lo que le impide reportar patrones en tiempo real.

Debido a este análisis realizado el algoritmo propuesto, presentado en el siguiente capítulo se enfocó en resolver dos problemas claves: mejorar el desempeño durante la búsqueda de los discos e implementar mecanismos de detección en tiempo real.

4 ALGORITMO FP-FLOCK

En el algoritmo propuesto por [19], en la primera parte del algoritmo se encuentra el total de discos con los objetos móviles que están juntos, se eliminan discos que no cumplan μ, este proceso genera que se encuentren varios de los discos solapados, para ello, por último, se realiza una limpieza de los discos solapados haciendo iteraciones en los discos encontrados y dejando únicamente aquellos discos con mayor número de miembros denominados discos máximos, este proceso de limpieza de los discos solapados hace que el costo computacional sea muy alto debido a la cantidad de iteraciones. En la segunda parte, el algoritmo realiza un proceso de combinación para el descubrimiento de flocks en el cual si el ε aumenta el algoritmo puede llegar a colapsar. Este algoritmo separa los flocks de acuerdo al parámetro δ en procura de limitar el número de discos a comparar, esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija.

Para el algoritmo propuesto por [10], ya que usa la primera parte del algoritmo en [19], tiene el mismo problema de los discos solapados. Ya en la segunda parte, al tratar el proceso de combinación es mucho más eficiente ya que utiliza un enfoque basado en técnicas de patrones frecuentes, aunque con la desventaja que el proceso se realiza en una ventana fija de tiempo lo que le impide reportar patrones en tiempo real.

La alternativa propuesta resuelve los problemas que poseen los algoritmos anteriores utilizando patrones frecuentes, se proponen el algoritmo llamado FP-Flock con dos variaciones, uno fuera de línea y otro en tiempo real.

4.1 FP-FLOCKOFFLINE

En la primera parte se hace el cálculo de los discos máximos, para lo cual se tomó como base el algoritmo 1 propuesto por [19] al cual se le hicieron modificaciones para hacer una limpieza de los discos solapados usando patrones frecuentes de minería. Puntualmente, se usó el algoritmo LCM [9]. El pseudo-código de la primera parte del algoritmo es presentado en Algoritmo 1.

Algoritmo 1 Computing maximal disks.

Input: set of points $T[t_i]$ for timestamp t_i
Output: sets of maximal disks C

```
1: C ← ∅
2: Index.Build(T[t_i], ε) {call Algorithm 1 in [19]}
3: for each non-empty cell g_{x,y} ∈ Index do
4:       L_r ← g_{x,y}
5:       L_s ← [g_{x−1,y−1} ... g_{x+1,y+1}]
6:       if |L_s| ≥ μ then
7:             for each l_r ∈ L_r do
8:                   H ← Range(l_r, ε), |H| ≥ μ, d(l_r, l_s) ≤ ε, l_s ∈ L_s
9:                   for each l_j ∈ H do
10:                        if {l_r, l_j} not yet computed then
11:                              compute left disk {c} defined by l_r, l_j and diameter ε
12:                              D ← points ∈ c
13:                        end if
14:                  end for
15:            end for
16:      end if
17:      min_sup ← 1
18:      C ← call LCM_max(D, min_sup) {call LCM Algorithm [9]}
19: end for
```

El algoritmo fuera de línea, se construyó usando el pseudo-código propuesto en [7], pero utilizando el algoritmo descrito anteriormente.

4.2 FP-FLOCKONLINE

El algoritmo en tiempo real, hace modificaciones al algoritmo propuesto en [7], este algoritmo en su primera parte usa el Algoritmo 1, para solucionar el problema de los discos solapados. En su segunda parte, se va liberando memoria en cada transacción, teniendo en cuenta las trayectorias que en ese instante de tiempo tuvieron un corte. En este algoritmo por cada intervalo de tiempo se realiza un llamado al algoritmo LCM propuesto por [9] con el objetivo de reportar los patrones obtenidos hasta el momento.

El pseudo-código del algoritmo en tiempo real es presentado en Algoritmo 2

4.3 VALIDACIÓN

Para poder validar la correcta implementación de los algoritmos se realizó el mismo proceso de validación mencionado en el capítulo anterior, con la implementación de

Algoritmo 2 FP-FlockOnline: Frequent pattern flock online.

Input: parameters μ, ε and δ, set of points T
Output: flock patterns F

```
1: for each new time instance t_i ∈ T do
2:     C ← call Index.Disks(T[t_i]) {call Algorithm 1 in this paper}
3:     for each c_i ∈ C do
4:         P ← c_i.points
5:         for each p_i ∈ P do
6:             c_i.time ← t_i
7:             D[p_i] ← add c_i.id
8:         end for
9:         for each p_i ∈ P do
10:            if D[p_i] not was updated then
11:                delete D[p_i]
12:            end if
13:        end for
14:    end for
15:    min_sup ← μ
16:    M ← call LCM_max(D, min_sup) {call LCM Algorithm [9]}
17:    for each max_pattern ∈ M do
18:        id_0 ← max_pattern[0]
19:        c_0 ← C[id_0]
20:        u ← c_0.points
21:        u.t_start ← c_0.time
22:        n ← max_pattern.size
23:        for i = 1 to n do
24:            id_i ← max_pattern[i]
25:            c_i ← C[id_i]
26:            if c_i.time = c_{i-1}.time + 1 then
27:                u ← u ∩ c_i.points
28:                u.t_end ← c_i.time
29:            else
30:                if u.t_end − u.t_start > δ and u ∉ F then
31:                    F ← add u
32:                    u.t_start ← c_i.time
33:                end if
34:            end if
35:        end for
36:        if u.t_end − u.t_start > δ and u ∉ F then
37:            F ← add u
38:        end if
39:    end for
40: end for
```

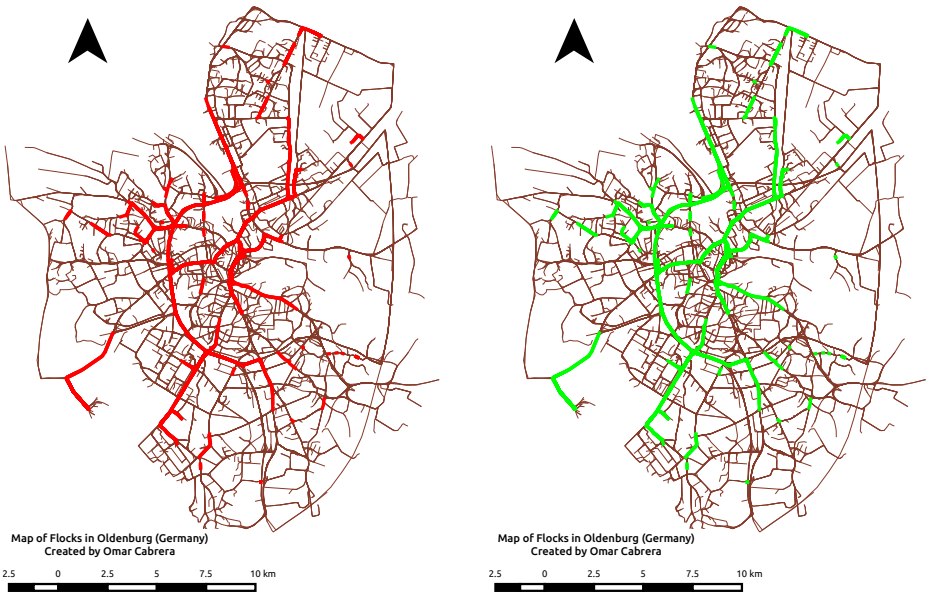
Tabla 4.1: Número de flocks generados por los algoritmos en el conjunto de Oldenburg μ=3, δ=3

$\varepsilon(m)$	BFE	LCM	FP-FlockOnline	FP-FlockOffline
50	131	27	150	27
100	639	109	663	109
150	1135	247	1226	247
200	2755	523	2683	523
250	5423	1150	6877	1150
300	11196	2365	16671	2365

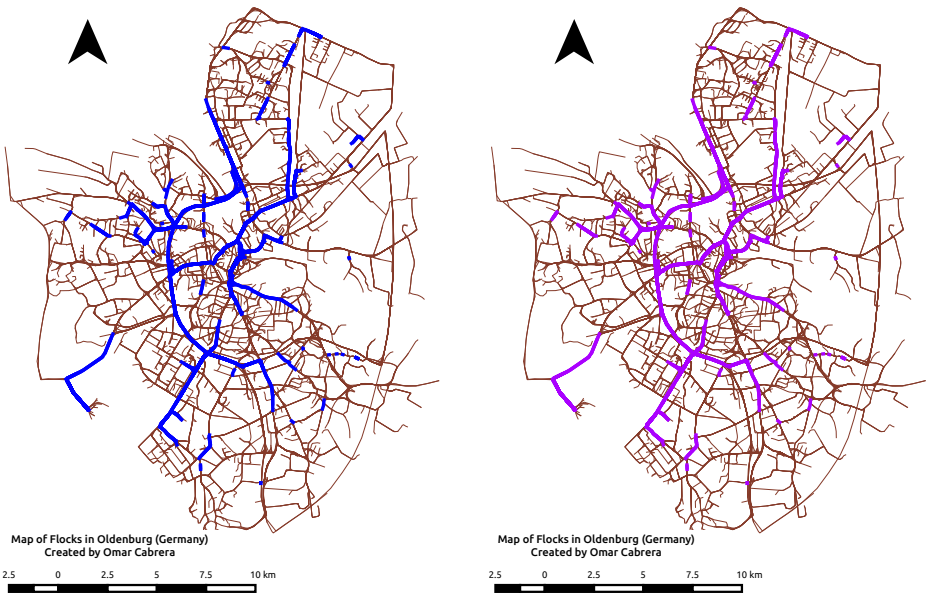
Fuente: Esta investigación.

BFE y LCMFlock, posterior a ello se realizó un proceso de validación visual.

Para realizar una validación visual se utilizó el conjunto de datos de Oldenburg disponible en [35], el cual proporciona un conjunto de ejemplos y recursos que pueden ser utilizados en la demostración en línea o versión descargable del generador. Para empezar, se utilizó un conjunto de datos relativamente pequeña posición de 1.000 objetos en movimiento al azar en la ciudad alemana de Oldenburg. Los datos de la red (bordes y nodos) están disponibles en el sitio web. La simulación de datos recoge la latitud y longitud de los puntos generados durante 140 intervalos de tiempo. El número total de ubicaciones almacenadas es 57.016 puntos. Con este conjunto se construyeron mapas con las representaciones lineales de los flocks resultantes como lo muestra la tabla 4.1 con las cuatro implementaciones. En la figura 4.1 se muestran los flocks obtenidos con los parámetros ε=300, μ=3 y δ=3 sobre el conjunto Oldenburg. Estos mapas se los comparó usando un módulo que implementa la similitud estructural métrica de imagen (SSIM) [36]. Los resultados de la comparación muestran que todos los mapas son idénticos, y por lo tanto, los mismos flocks son reportados por los diferentes algoritmos.



(a) BFE (b) LCMFlock



(c) FP-FlockOnline (d) FP-FlockOffline

Figura 4.1: Visualización Oldenburg

5 EXPERIMENTACIÓN COMPUTACIONAL

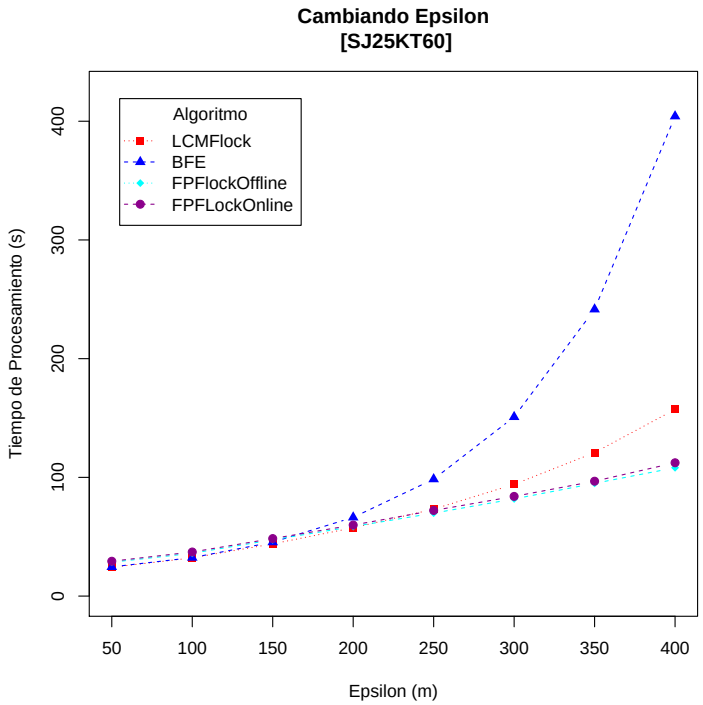
Los resultados fueron producidos usando conjuntos de datos sintéticos y reales en una máquina Dell OPTIPLEX 7010 con procesador Intel®Core™i7-3770 CPU de 3.40GHz x 8, 16 GB de RAM y 1TB 7200 RPM de Disco Duro, corriendo el Sistema Operativo Debian con linux 3.2. Para todos los casos se usaron los algoritmos implementados en Python, versión 3.

5.1 SAN JOAQUIN

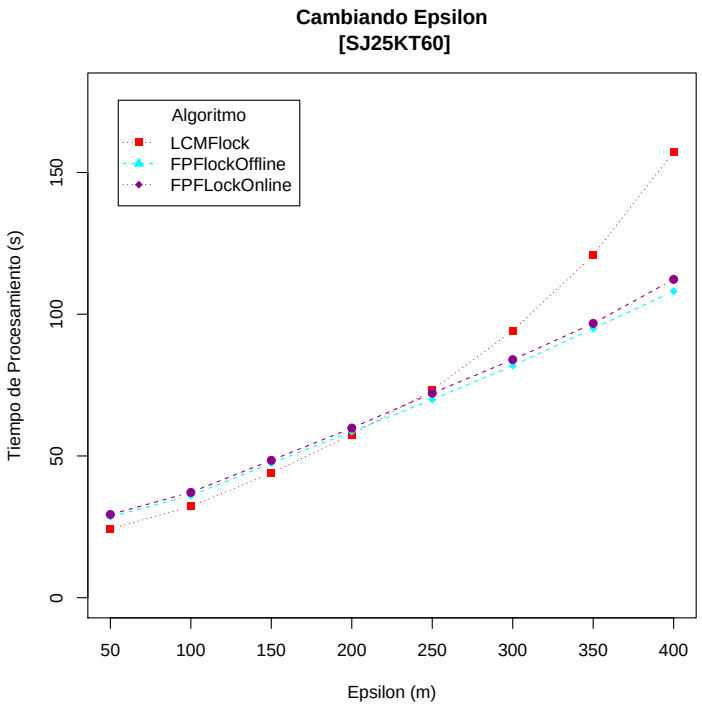
Un grupo de conjuntos de datos sintéticos fueron creados usando un modelo para la generación de objetos en movimiento, como se describe en [37]. Dos conjuntos de datos sintéticos fueron creados usando la red de San Joaquín proporcionada en el sitio web del generador [35]. El primer conjunto de datos recoge 992140 lugares simulados para 25.000 objetos en movimiento durante 60 instantes de tiempo. El segundo recoge 50.000 trayectorias a partir de 2.014.346 de puntos durante 55 instantes de tiempo. La tabla 5.1, resume la información principal. Las figuras 5.1 y 5.2 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=5, δ=3 y μ=9, δ=3, respectivamente.

Tabla 5.1: Conjunto de datos

Dataset	Red	Número de trayectorias	Número de puntos	Duración promedio de la trayectoria
SJ25KT60	San Joaquin	25000	992140	40
SJ50KT55	San Joaquin	50000	2014346	37
TAPAS Cologne	Cologne, Alemania	88668	3403463	38
Beijing_Original	Beijing, China	21573	1411846	65
Beijing_Alternativo	Beijing, China	18700	815657	43

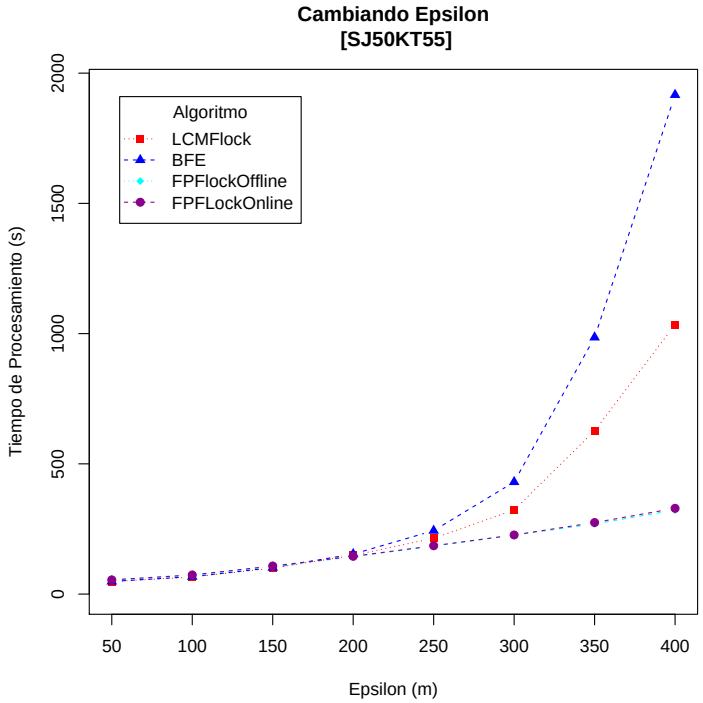


(a) BFE, LCMFlock, FPFlock

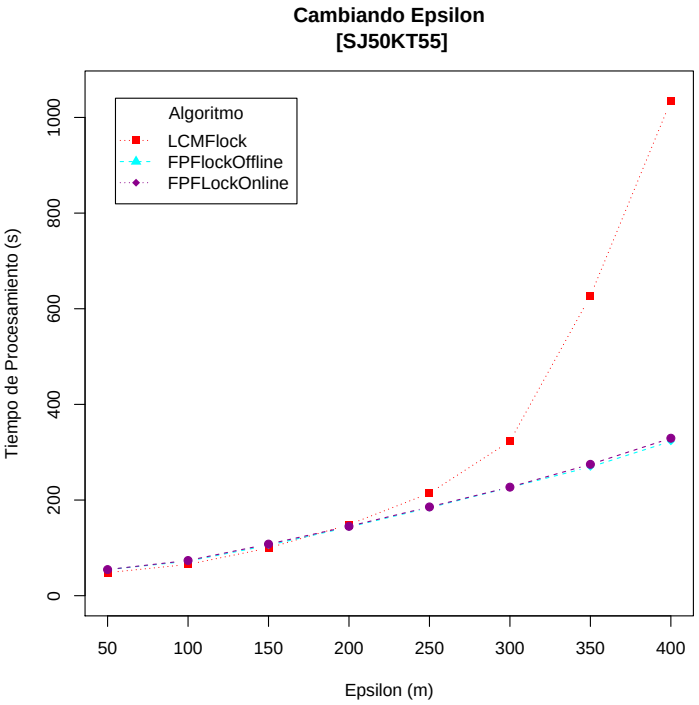


(b) LCMFlock, FPFlock

Figura 5.1: Caso de Prueba: SJ25KT60. (a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock



(a) BFE, LCMFlock, FP-Flock



(b) LCMFlock, FP-Flock

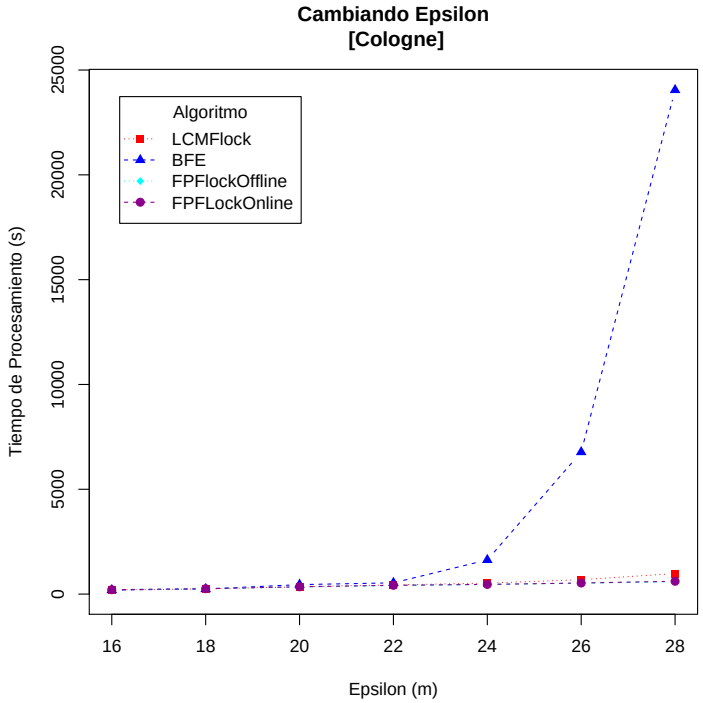
Figura 5.2: Caso de Prueba: SJ50KT55. (a) BFE, LCMFlock, FP-Flock (b) LCMFlock, FP-Flock

5.2 TAPAS COLOGNE

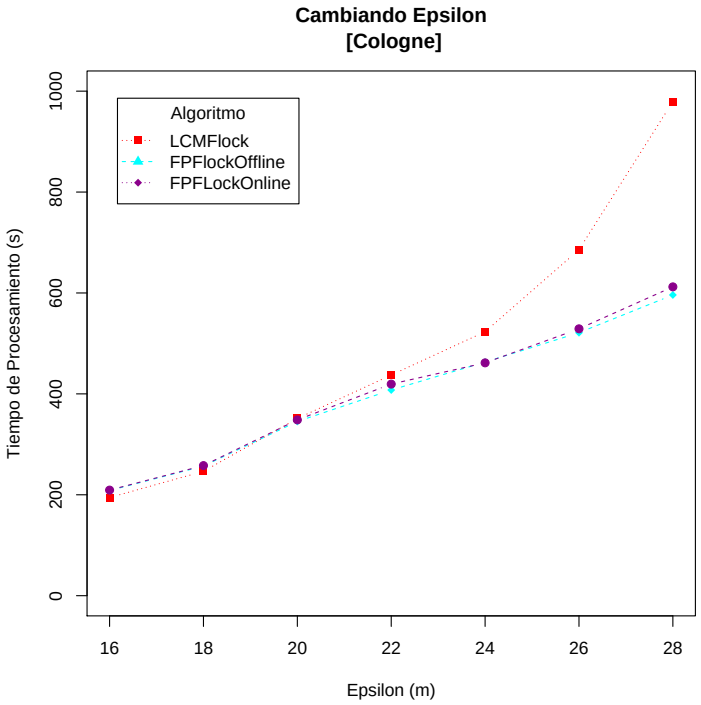
Este conjunto de datos sintético se preparó utilizando el escenario TAPAS Cologne [38] en SUMO [39], un reconocido simulador de tráfico para la movilidad urbana. El escenario de simulación TAPAS Cologne, describe el tráfico dentro de la ciudad de Colonia (Alemania) durante un día entero. La principal ventaja de este conjunto de datos es que sus trayectorias no se generan aleatoriamente. Los datos de la demanda original, se deriva de TAPAS, un sistema que calcula la tendencia de movilidad para una población con base en la información sobre los hábitos de viaje de los alemanes y en la información sobre la infraestructura de la zona en que viven [40]. El conjunto de datos original es enorme por lo que sólo está disponible al público la versión de 2 horas [41]. Debido a las restricciones de memoria, se podaron las trayectorias más cortas que 20 minutos. El último conjunto de datos recoge 88.668 trayectorias y más de 3,4 millones de puntos. La tabla 5.1, describe los detalles sobre el conjunto de datos. Las figura 5.3 muestra los tiempos de desempeño para este caso de estudio. Los parámetros adicionales fueron μ=10, δ=5.

5.3 MOVIMIENTO DE PEATONES EN BEIJING

Este conjunto de datos reales recopila información de movimiento de un grupo de personas en toda el área metropolitana de Beijing, China[42]. El conjunto de datos se recogió durante el proyecto Geolife por 165 usuarios anónimos en un período de dos años entre abril de 2007 y agosto de 2009. Las ubicaciones fueron grabadas por diferentes dispositivos GPS o teléfonos inteligentes y la mayoría de ellos presentan una frecuencia de muestreo alta. La región alrededor del quinto anillo vial en el área metropolitana de Beijing mostró la mayor concentración de trayectorias. Esto fue usado para generar un conjunto de datos de muestra. Cada trayectoria fue interpolada por minuto (un punto por minuto) y saltos de 20 minutos o más sin señal se utilizaron para marcar una nueva trayectoria. Por último, el conjunto de datos recoge más de 1,4 millones de puntos y 21.573 trayectorias. Sin embargo, como este conjunto de datos tuvo poca cantidad de entidades en movimiento (165 usuarios) en una ventana de tiempo de más de 2 años, no existieron muchas trayectorias ocurriendo al mismo tiempo. Para probar la escalabilidad de los algoritmos, se decidió crear un conjunto de datos alternativo basado en las trayectorias reales, pero forzando para que todas ellas comiencen al mismo tiempo. Una



(a) BFE, LCMFlock, FP-Flock



(b) LCMFlock, FP-Flock

Figura 5.3: Caso de Prueba: Tapas Cologne. (a) BFE, LCMFlock, FP-Flock (b) LCM-Flock, FP-Flock

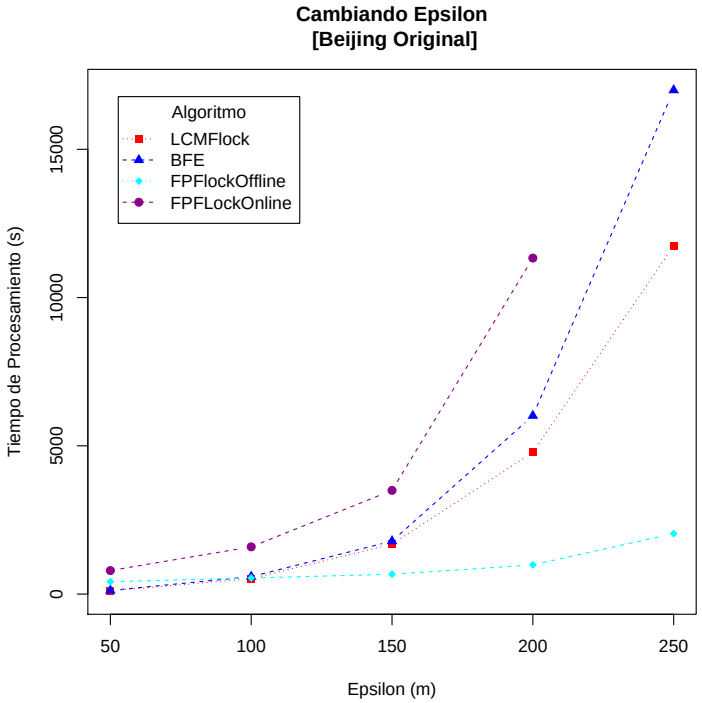
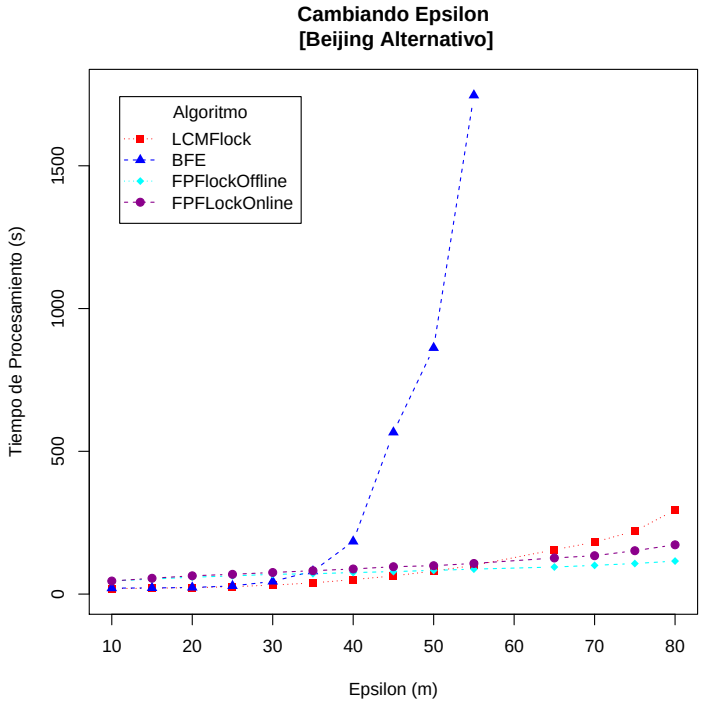


Figura 5.4: Caso de Prueba: Beijing Original

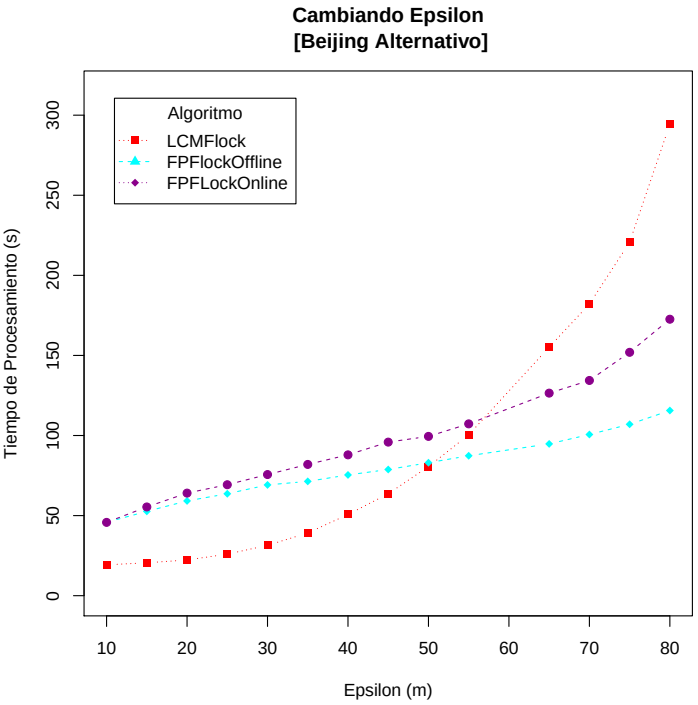
vez más, por las limitaciones de memoria, las trayectorias más cortas que 10 minutos y más largas que 3 horas se podaron. El conjunto de datos alternativo almacenó 815.657 ubicaciones y 18.700 trayectorias. La tabla 5.1 resume los detalles para ambos conjuntos de datos. Las figuras 5.4 y 5.5 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=3, δ=3 y μ=5, δ=5 respectivamente.

5.4 REPORTE DE FLOCKS

Tanto para BFE, LCMFlock, FP-FlockOnline y FP-FlockOffline el reporte de flocks se hace en una base de datos, con un identificador, tiempo de inicio, tiempo de fin y todos los puntos contenidos en dicho flock. En la tabla 5.2, se muestra el número total de flocks reportados con un ε en particular.



(a) BFE, LCMFlock, FP-Flock



(b) LCMFlock, FP-Flock

Figura 5.5: Caso de Prueba: Beijing Alternativo. (a) BFE, LCMFlock, FP-Flock (b) LCMFlock, FP-Flock

Tabla 5.2: Número de Flocks

Dataset	ε	Número de Flocks BFE	Número de Flocks LCMFlock	Número de Flocks FP-FlockOnline	Número de Flocks FP-FlockOffline
SJ25KT60	300	35805	5466	26877	5466
SJ50KT55	300	45201	6396	23638	6396
TAPAS Cologne	28	9415451	31509	145217	31509
Beijing_Original	250	16628029	4373139	-	4373139
Beijing_Alternativo	50	6110427	19233	63566	19233

6 DISCUSIÓN

Para hacer una comparación de rendimiento de los algoritmos, es importante que se los evalué de acuerdo a las características de cada algoritmo, para ello se debe comparar el algoritmo BFE propuesto por [19] con la alternativa FPFlockOnline y el algoritmo LCMFlock propuesto por [10] con la alternativa FPFlockOffline.

En el conjunto de San Joaquin, se puede mirar que el algoritmo de BFE, crece muy rápidamente a medida que crece ε, más aún en el rango entre ε 300 y 400, como lo muestra la figura 5.1 y la figura 5.2. LCMFlock en ese mismo rango de ε se comporta de igual manera pero en menor escala. La alternativa propuesta se comporta de una manera lineal. Estos resultados son comparables con los descritos en [10].

En el conjunto de Tapas Cologne, el algoritmo de BFE, luego de un ε mayor a 24 se incrementa muy rápido hasta colapsar como lo muestra en la figura 5.3(a). De igual manera, LCMFlock se incrementa luego de un ε mayor a 24 pero en menor escala. La alternativa propuesta se comporta de una manera lineal. Estos resultados son comparables con los descritos en [7].

En el conjunto de Beijing original se puede observar que la alternativa en tiempo real con ε igual a 200 colapsa, los algoritmos BFE y LCMFlock crecen rapidamente con ε mayor a 200, (BFE más rapidamente que LCMFlock), mientras que la alternativa fuera de línea tiene un buen rendimiento con respecto a los otros algoritmos. Esto hace pensar que la alternativa en tiempo real colapsa debido a que en la implementación del algoritmo de LCM, se escribe y se lee en disco repetidamente debido a que el conjunto tiene varios instantes de tiempo. Estos resultados son comparables con los descritos en [10].

En el conjunto de Beijing alternativo, el algoritmo BFE incrementa su tiempo de respuesta muy rapidamente con un ε mayor a 40 y llega a colapsar en un ε igual a 50, como lo muestra la figura 5.5(a). En la figura 5.5(b), se observa un rápido incremento en la ejecución de LCMFlock con un ε mayor a 50. Estos resultados son comparables con los descritos en [10] .

En general, para todos los conjuntos, se observa que la alternativa propuesta en la mayoría de los casos tiene un mejor desempeño sobre BFE y LCMFlock a medida que aumenta ε, esto debido al enfoque basado en técnicas de patrones frecuentes.

En BFE, el reporte de flocks es demasiado alto en comparación con LCMFlock (tabla 5.2). BFE separa los flocks de acuerdo al parámetro δ en procura de limitar el número de discos a comparar. Esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija. En LCMFlock esto se soluciona con el uso del algoritmo LCM para la detección de patrones máximos y cerrados. En FPFlock para la interpretación de flocks resultantes no requiere hacer un análisis adicional ya que en la alternativa en tiempo real los flocks más largos son reportados en cada instante de tiempo y en la alternativa fuera de linea el reporte de flocks es igual a LCMFLock.

La capacidad de encontrar los flocks más largos ciertamente es una ventaja de LCMFlock como de la alternativa fuera de línea sobre BFE y la alternativa en tiempo real. Sin embargo, los algoritmos fuera de linea, a diferencia de los que son en tiempo real, requieren de una ventana fija de tiempo donde ser aplicados. En ciertas aplicaciones, como seguridad y gestión de tráfico, ésta es una característica muy relevante. Vale la pena explorar con más profundidad mecanismos que permitan aplicar los algoritmos de detección de patrones frecuentes en tiempo real en este tipo de situaciones.

7 CONCLUSIONES Y TRABAJOS FUTUROS

Se diseñó un algoritmo llamado FP-Flock con dos variaciones: una fuera de línea y otra en tiempo real, utilizando un enfoque basado en la detección de patrones frecuentes. La propuesta ha demostrado un buen rendimiento en diferentes casos de prueba, tanto sintéticos como reales.

Durante el proceso de implementación se evaluaron diferentes estructuras de datos para optimizar las consultas espaciales. Después de las diferentes pruebas fue evidente que las estructuras de tipo árbol, y en específico, los kd-tree presentaron los mejores resultados. Esto se configura como un aspecto clave para la búsqueda del conjunto final de discos para cada instante de tiempo. Entre mejores implementaciones de este tipo de estructuras los resultados de ejecución mejorarían de manera considerable.

El enfocar esta investigación en una metodología basada en patrones frecuentes, mejoró el desempeño durante la búsqueda de los discos y el descubrimiento de flocks en tiempo real, dando un punto de partida para realizar implementaciones similares para la optimización usando este enfoque.

Sin embargo, en la alternativa propuesta se realizó un llamado al algoritmo LCM, disponible de forma binaria como una aplicación externa. Para solucionar el problema de reiterados llamados de lectura y escritura en disco, se debe implementar de manera nativa el algoritmo de LCM dentro de la solución propuesta.

Para la implementación de los algoritmos, los autores de los algoritmos no hacen ninguna sugerencia de poder realizar la implementación en paralelo, para futuros trabajos se recomienda realizar un modelo de implementación en paralelo para usar el mayor rendimiento de la máquina en la que se esté probando.

Además, se obtuvo los siguientes productos:

- Se participó con el poster “FP-Flock: Una alternativa para el descubrimiento de patrones de agrupación de objetos móviles basadas en técnicas de patrones frecuentes”, en el Congreso Andino de Computación, Informática y Educación

(CACIED), el cuál esta publicado en el libro de resúmenes del congreso con ISBN 978-958-8609-67-6.

- Se implementaron los algoritmos propuestos en Python 3, los cuales se encuentran en el repositorio de la investigación.
- Se construyó un conjunto de base de datos sintéticas a partir de un script para la validación, tanto el script como las bases de datos se encuentran en el repositorio de la investigación.
- Se construyeron las bases de datos de pruebas para ser usadas en los algoritmos implementados.
- Se participó en la Conferencia Latinoamericana en Informática (CLEI 2014) realizada en Septiembre 15 al 19, 2014, con el artículo "Performance analysis of flock pattern algorithms in spatio-temporal databases", con ISBN: 978-1-4799-6129-0.
- Se escribió un artículo con los resultados finales el cual esta siendo evaluado para su publicación en una revista.

REFERENCIAS

[1] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1161–1174, 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2007.1054

[2] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, C. Bauzer Medeiros, M. Egenhofer, and E. Bertino, Eds. Springer Berlin Heidelberg, 2005, vol. 3633, pp. 364–381. [Online]. Available: http://dx.doi.org/10.1007/11535331_21

[3] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1068–1080, 2008.

[4] J. Gudmundsson and M. van Kreveld, "Computing longest duration flocks in trajectory data," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS '06. New York, NY, USA: ACM, 2006, pp. 35–42. [Online]. Available: http://doi.acm.org/10.1145/1183471.1183479

[5] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle, "Reporting flock patterns," *Computational Geometry*, vol. 41, no. 3, pp. 111 – 125, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092577210700106X

[6] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. New York, NY, USA: ACM, 2009, pp. 286–295. [Online]. Available: http://doi.acm.org/10.1145/1653771.1653812

[7] U. Turdukulov, A. O. Calderon Romero, O. Huisman, and V. Retsios, "Visual mining of moving flock patterns in large spatio-temporal data sets

using a frequent pattern approach," *International Journal of Geographical Information Science*, vol. 1, pp. 1–17, 2014. [Online]. Available: http://dx.doi.org/10.1080/13658816.2014.889834

[8] P. Laube, M. van Kreveld, and S. Imfeld, "Finding remo — detecting relative motion patterns in geospatial lifelines," in *Developments in Spatial Data Handling*. Springer Berlin Heidelberg, 2005, pp. 201–215. [Online]. Available: http://dx.doi.org/10.1007/3-540-26772-7_16

[9] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining," in *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, ser. OSDM '05. New York, NY, USA: ACM, 2005, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/1133905.1133916

[10] A. O. C. Romero, "Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach," Master's thesis, University of Twente, 2011.

[11] O. E. Cabrera Rosero and A. O. Calderón Romero, "Performance analysis of flock pattern algorithms in spatio-temporal databases," in *CLEI2014*, 2014.

[12] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*. Springer Science and Business Media, LLC, 2011.

[13] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. De Macedo, F. Porto, and C. Vangenot, "A conceptual view on trajectories," *Data & knowledge engineering*, vol. 65, no. 1, pp. 126–146, 2008.

[14] S. Iwase and H. Saito, "Parallel tracking of all soccer players by integrating detected positions in multiple view images," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4. IEEE, 2004, pp. 751–754.

[15] A. Frank, J. Raper, and J. Cheylan, *Life and motion of socio-economic units*. CRC, 2001.

[16] H. Dettki, G. Ericsson, and L. Edenius, "Real-time moose tracking: an internet based mapping application using GPS/GSM-collars in Sweden," *Alces*, vol. 40, pp. 13–21, 2004.

[17] D. Makris and T. Ellis, "Path detection in video surveillance," *Image and Vision Computing*, vol. 20, no. 12, pp. 895–903, 2002.

[18] C. Piciarelli, G. Foresti, and L. Snidara, "Trajectory clustering and its applications for video surveillance," in *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on.* IEEE, 2006, pp. 40–45.

[19] M. Vieira and V. Tsotras, "Flock pattern queries," in *Spatio-Temporal Databases*, ser. SpringerBriefs in Computer Science. Springer International Publishing, 2013, pp. 61–83. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02408-0_4

[20] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[21] C. Creighton and S. Hanash, "Mining gene expression databases for association rules," *Bioinformatics*, vol. 19, no. 1, pp. 79–86, 2003.

[22] H. J. Miller and J. Han, *Geographic data mining and knowledge discovery.* CRC Press, 2009.

[23] C. Zhang and S. Zhang, *Association rule mining: models and algorithms.* Springer-Verlag, 2002.

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

[25] J. Han and M. Kamber, *Data Mining, Southeast Asia Edition: Concepts and Techniques.* Morgan kaufmann, 2006.

[26] D. Tsur, J. D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal, "Query flocks: A generalization of association-rule mining," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 1–12.

[27] P. Giudici, *Applied data mining: statistical methods for business and industry.* John Wiley & Sons, 2005.

[28] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining user similarity based on location history," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:10. [Online]. Available: http://doi.acm.org/10.1145/1463434.1463477

[29] H. Jeung, H. T. Shen, and X. Zhou, "Convoy queries in spatio-temporal databases," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, April 2008, pp. 1457–1459.

[30] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[31] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *FIMI*, vol. 126, 2004.

[32] QGIS Development Team, *QGIS Geographic Information System*, Open Source Geospatial Foundation, 2009. [Online]. Available: http://qgis.osgeo.org

[33] B. Goethals. (2004) Frequent itemset mining implementations repository. http://fimi.cs.helsinki.fi/. Consultado Julio 2014.

[34] J. Han and J. Pei, "Mining frequent patterns by pattern-growth: Methodology and implications," *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 14–20, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/380995.381002

[35] T. Birkhoff. (2005) Network-based generator of moving objects. http://iapg.jade-hs.de/personen/brinkhoff/generator/. Consultado Julio 2014.

[36] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, April 2004.

[37] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1015231126594

[38] C. Varschen and P. Wagner, "Microscopic modeling of passenger transport demand based on time-use diaries," in *Integrated micro-simulation of land use and transport development. Theory, concepts, models and practice*, K. J. Beckmann, Ed., vol. 81, 2006, pp. 63–69.

[39] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)," in *Proc. of the 4th middle east symposium on simulation and modelling*, 2002, pp. 183–187.

[40] MiD2002 Project. (2002) Mobility in Germany 2002. http://daten.clearingstelle-verkehr.de/196/. Consultado Julio 2014.

[41] SUMO Project. (2011) TAPAS Cologne Scenario. http://sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html.

[42] Microsoft Research Asia. (2010) Geolife gps trajectories. http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx. Consultado Julio 2014.

ANEXOS

- Poster presentado en el Congreso Andino de Computación, Informática y Educación (CACIED).
- Artículo publicado en la Conferencia Latinoamericana en Informática (CLEI 2014).
- Artículo que esta en etapa de evaluación en revista indexada "Polibits".

OMAR ERNESTO CABRERA ROSERO[1]

ANDRÉS OSWALDO CALDERÓN ROMERO [2]

Eje Temático: Sistemas inteligentes y de conocimiento
Temática: Base de datos espacio temporal

Los recientes avances tecnológicos y el amplio uso de localización por sistemas de posicionamiento global (GPS), identificación por radio frecuencia (RFID) y tecnologías en dispositivos móviles han hecho que las bases de datos espacio temporales recolectadas se hayan incrementado con un porcentaje acelerado. Esta gran cantidad de información ha motivado a desarrollar eficientes técnicas para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos en un periodo continuo de tiempo. Diversos estudios se han centrado en la consulta de los patrones para la captura del comportamiento de los objetos en movimiento reflejada en colaboraciones tales como clústeres móviles, consulta de convoyes y patrones de agrupación. Este trabajo se enfocará en el descubrimiento de patrones de agrupación, conocidos como "flocks", entre los objetos en movimiento de acuerdo a las características de los objetos de estudio (animales, peatones, vehículos o fenómenos naturales), cómo interactúan entre si y como se mueven juntos. El objetivo principal de esta investigación será definir una serie de pasos resumidos en un algoritmo usando técnicas de descubrimiento de patrones frecuentes y minería de datos ampliamente conocidas para mejorar el tiempo de obtención de los patrones. La implementación de este algoritmo tiene diversas aplicaciones tales como: sistemas integrados de transporte, seguridad y monitoreo y el seguimiento a grupos de animales y fenómenos naturales.

Palabras Clave: Base de datos espacio temporales, patrones de movimiento, patrones frecuentes, minería de datos.

1 Estudiante de Ingeniería de Sistemas, Universidad de Nariño, Pasto (Colombia). Correo electrónico: omarcabrera@udenar.edu.co.

2 Docente hora catedra, Universidad de Nariño, Pasto (Colombia). M.Sc en Geoinformática. Correo electrónico: aocalderon@udenar.edu.co.

Recent advances in technology and the widespread use of tracking global positioning systems (GPS), radio frequency identification (RFID) and mobile technologies have made the collected spatio-temporal datasets increase at an accelerated pace. This large amount of data has led to develop efficient techniques to process queries about the behavior of moving objects, like the discovering of patterns among trajectories in a continuous period of time. Several studies have focused on the query of patterns capturing the behavior of moving objects reflected in collaborations such as mobile clusters, convoy queries and flock patterns. This paper will focus on discovering moving flock patterns among moving objects according to the characteristics of the study objects (animals, pedestrians, vehicles or natural phenomena), how they interact with each other and how they move together. The main objective of this research is to define a set of steps outlined in an algorithm using frequent pattern and data mining techniques well known to improve the time performance in the discovery process. The implementation of this algorithm has various applications such as integrated transport systems, security and monitoring and tracking groups of animals and natural phenomena.

Keywords: Spatio-temporal datasets, moving flock patterns, frequent patterns, data mining.

Performance analysis of flock pattern algorithms in spatio-temporal databases

Omar Ernesto Cabrera Rosero
Universidad de Nariño
San Juan de Pasto, Colombia
Email: omarcabrera@udenar.edu.co

Andrés Oswaldo Calderón Romero
Universidad de Nariño
San Juan de Pasto, Colombia
Email: aocalderon@udenar.edu.co

***Abstract*—Recent advances in technology and the widespread use of tracking global positioning systems, such as GPS and RFID, and mobile technologies have made the access to spatio-temporal datasets increase at an accelerated pace. This large amount of data has led to develop efficient techniques to process queries about the behavior of moving objects, like the discovering of patterns among trajectories in a continuous period of time. Several studies have focused on the query of patterns capturing the behavior of moving objects reflected in collaborations such as mobile clusters, convoy queries and flock patterns. In this paper, a comparison between two algorithms for flocking, Basic Flock Evaluation (BFE) and LCMFLOCK, is presented in order to measure their performance and behavior in different datasets, both synthetic and real. This research is the first step towards proposing new algorithms in order to improve the drawbacks reported by the former methods.**

Keywords*—movement patterns, frequent patterns mining, spatio-temporal databases, flock patterns.

I. Introducción

Los recientes avances tecnológicos y el amplio uso de la localización basada en sistemas de posicionamiento global (GPS), identificación por radio frecuencia (RFID) o tecnologías en dispositivos móviles han hecho que acceso a bases de datos espacio temporales se haya incrementado de una manera acelerada. Esta gran cantidad de información ha motivado a desarrollar técnicas eficientes, para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos en un período contínuo de tiempo.

Los métodos que existen para la consulta de trayectorias se centran principalmente en responder un único rango simple de predicado y consultas de vecinos más cercanos, por ejemplo: "encontrar todos los objetos en movimiento que se encontraban en la zona A a las 10 de la mañana" o "encontrar el coche que condujo cerca de la ubicación B durante el intervalo de tiempo de 10 de la mañana a 1 de la tarde". Recientemente, diversos estudios se han centrado en la consulta de los patrones para la captura del comportamiento de los objetos en movimiento reflejada en colaboraciones tales como clusters móviles [1] [2], consulta de convoyes [3] y patrones de agrupamiento [4] [5] [6] [7]. Estos patrones descubren grupos de objetos en movimiento que tienen una "fuerte" relación en el espacio durante un tiempo determinado. La diferencia entre todos esos patrones es la forma de definir la relación entre los objetos en movimiento y su duración en el tiempo.

Este artículo se enfocará en el descubrimiento de patrones de agrupamiento, conocidos como "flocks", entre los objetos en movimiento de acuerdo a las características de los objetos de estudio (animales, peatones, vehículos o fenómenos naturales), cómo interactúan entre sí y cómo se mueven juntos [8] [9]. [6] define patrones de agrupamiento como el problema de identificar todos los grupos de trayectorias que permanecen "juntas" por la duración de un intervalo de tiempo dado. Consideramos que los objetos en movimiento están suficientemente cerca si existe un disco con un radio dado que cubre todos los objetos que se mueven en el patrón (Figura 1). Una trayectoria satisface el patrón anterior, siempre y cuando suficientes trayectorias están contenidos dentro del disco para el intervalo de tiempo especificado, es decir, la respuesta se basa no sólo en el comportamiento de una trayectoria dada, sino también en las más cercanas a ella. Uno de los enfoques para descubrir patrones móviles de agrupamiento consiste en encontrar un conjunto adecuado de discos en cada instante de tiempo y luego la fusión de los resultados de un instante de tiempo a otro. Como consecuencia, el rendimiento y el número de patrones final depende del número de los discos y cómo éstos se combinan.

En el ejemplo de la figura 1 se muestra un patrón de agrupamiento el cual contienen tres trayectorias {T1, T2, T3} que están dentro de un disco en tres instantes de tiempo consecutivos. Los discos se pueden mover libremente en el espacio bidimensional con el fin de acomodar los tres objetos en movimiento y su centro no necesariamente tiene que ser la localización de alguno de los objetos. Esto hace que el descubrimiento de patrones sea mucho más complicada porque hay un número infinito de posibles colocaciones del disco en cualquier instante de tiempo y el posible número de combinaciones puede llegar a ser muy alta y costosa.

La implementación de este tipo de análisis tiene diversas aplicaciones tales como: sistemas integrados de transporte, seguridad y monitoreo, seguimiento a grupos de animales y fenómenos naturales, encontrando así una alternativa diferente para solucionar los problemas en el mundo, analizando como se mueven los objetos en la tierra y cuáles son los patrones de comportamiento que existen

978-1-4799-6129-0/14/$31.00

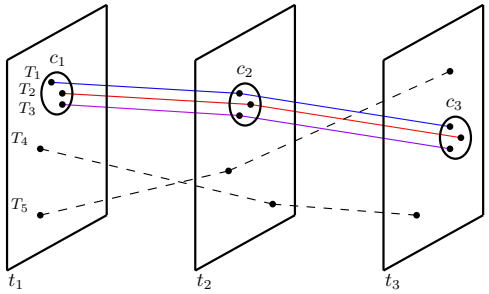


Figura 1. Ejemplo de patrones de agrupamiento.

entre sí.

En este artículo se muestra una comparación entre dos algoritmos, propuestos por [6] y [10], con el fin de identificar los problemas asociados a su rendimiento, probándolos en distintos conjuntos de datos tanto reales como sintéticos. Se escogió únicamente estos dos algoritmos para la comparación debido a que este análisis es enfocado únicamente a patrones de agrupamiento (flocks) y estos son los algoritmos más representativos que existen hasta el momento en este campo. Una posterior fase de la investigación buscará proponer un nuevo algoritmo que solucione los inconvenientes reportados y encontrados en estas dos alternativas.

El resto del articulo esta organizado de la siguiente manera. En la sección trabajos relacionados se describe varios estudios que se han realizado en objetos móviles como algoritmos de clústers, convoyes y patrones de agrupamiento. En la sección 3 describe la implementación de los algoritmos analizados y las pruebas de los mismos. En la sección experimentación computacional se presenta los conjuntos de datos y se compara el rendimiento de los algoritmos. Por último, se presentan las conclusiones y trabajos futuros.

II. Trabajos Relacionados

Las capacidad de recolectar datos de objetos en movimiento ha ido aumentando rápidamente y el interés de consulta de patrones que describen el comportamiento colectivo también ha aumentado. [6] enumera tres grupos de patrones "colectivos" en bases de datos de objetos en movimiento: clústers móviles, consulta de convoyes y patrones de agrupamiento.

Los clústers móviles [1] [2] [11] y consultas de convoyes [3] [12], tienen en común que se basan en algoritmos de clústering, principalmente en algoritmos basados en densidad como el algoritmo DBSCAN[13].

Los clústers móviles se definen entre dos instantes de tiempo consecutivos. Los clústers se pueden unir sólo si el número de objetos comunes entre ellos están por encima del parámetro predefinido. Un clúster es reportado si no hay otro nuevo clúster que pueda ser unido a éste. Este proceso se aplica cada vez para todos los instantes de tiempo en el conjunto de datos.

Las consultas de convoyes se definen como un clúster denso de trayectorias que permanecen juntas al menos por un tiempo contínuo predefinido.

Las principales diferencias entre las dos técnicas son la forma en que se unen los grupos entre dos intervalos consecutivos de tiempo y el uso de un parámetro adicional para especificar un tiempo mínimo de duración. Aunque estos métodos están estrechamente relacionados con los patrones de agrupamiento, ninguno de ellos asume una forma predefinida.

Previos trabajos de detección de patrones de agrupamiento móviles son descritos por [4] y [5]. Ellos introducen el uso de discos con un radio predefinido para identificar grupos de trayectorias que se mueven juntos en la misma dirección, todas las trayectorias que se encuentran dentro del disco en un instante de tiempo particular se considera un patrón candidato. La principal limitación de este proceso es que hay un número infinito de posibles ubicaciones del disco en cualquier instante de tiempo. En efecto, en [4] se ha demostrado que el descubrimiento de agrupaciones fijas, donde los patrones de las mismas entidades permanecen juntas durante todo el intervalo, es un problema NP-complejo.

[6] son los primeros en presentar una solución exacta para reportar patrones de agrupación en tiempo polinomial, y también pueden trabajar efectivamente en tiempo real. Su trabajo revela que el tiempo de solución polinomial se puede encontrar a través de la identificación de un número discreto de ubicaciones para colocar el centro del disco. Los autores proponen el algoritmo BFE (Basic Flock Evaluation) basado en el tiempo de unión y combinación de los discos. La idea principal de este algoritmo es primero encontrar el número de discos válidos en cada instante de tiempo y luego combinarlos uno a uno entre tiempos adyacentes. Adicionalmente se proponen otros cuatro algoritmos basados en métodos heurísticos, para reducir el número total de candidatos a ser combinados y, por lo tanto, el costo global del algoritmo. Sin embargo, el pseudocódigo y los resultados experimentales muestran todavía una alta complejidad computacional, largos tiempos de respuesta y un gran número de patrones que hace difícil su interpretación.

[10] y [7] proponen una metodología que permite identificar patrones de agrupamiento utilizando tradicionales y potentes algoritmos de minería de datos usando patrones frecuentes, el cual fue comparado con BFE demostrando un alto rendimiento con conjuntos de datos sintéticos, aunque con conjuntos de datos reales el tiempo de respuesta siguió siendo eficiente pero similar a BFE. Este algoritmo trata el conjunto de trayectorias como una base de datos transaccional al convertir cada trayectoria, que se define como un conjunto de lugares visitados, en una transacción, definida como un conjunto de ítems. De esta manera, es posible aplicar cualquier algoritmo de reglas de asociación y encontrar patrones frecuentes sobre el conjunto dado.

III. Implementación

Se implementaron los algoritmos BFE y LCMFLOCK basados en el pseudo-código publicado por [6] y [10] respec-

tivamente, usando Python versión 3 debido a la facilidad y comodidad para el programador. Realizar esta implementación tenía como objetivo poder apropiar el conocimiento y además hacer una inspección más detallada de cada algoritmo con el fin de encontrar sus inconvenientes y buscar una mejora. El código fuente se lo puede descargar desde el repositorio del proyecto [1].

A. BFE

Este algoritmo se divide en dos partes: la primera parte, encontrar los discos dado un radio (ϵ) y un número mínimo de puntos (μ) para cada instante de tiempo. La segunda parte, encontrar el número de puntos que permanecen juntos (flocks) durante un rango de tiempo (δ).

Para la primera parte es necesaria la utilización tanto de diccionarios de datos como estructuras kd-tree para la búsqueda del vecino más cercano, en esta implementación se usó la clase scipy.spatial.cKDTree de SciPy [2] la cual proporciona un índice dentro de un conjunto de puntos k-dimensionales que se pueden utilizar para buscar rápidamente los vecinos más cercanos de cualquier punto. El conjunto de discos para cada instante de tiempo se almacenaron en diccionarios los cuales fueron combinados en la segunda parte del algoritmo. Finalmente, se almacenaron en una base de datos los flocks que cumplían los parámetros solicitados. Para cada flock se almacenó un arreglo con los objetos que pertenecen a dicho flock, y los tiempos de inicio y final para cada caso.

B. LCMFLOCK

Este algoritmo usa la primera parte del algoritmo de BFE para encontrar los discos. En la segunda parte, para abordar el problema de combinatoria utiliza un enfoque de patrones frecuentes de minería, en el cual se construyó un diccionario de datos asociando la localización de los puntos de cada trayectoria con su respectivo disco para generar una versión transaccional del conjunto de datos. Este conjunto es pasado como parámetro, junto con el número mínimo de puntos (μ), al algoritmo LCM[14], disponible para descargar en [15]. Aquí se encuentran disponibles dos variantes del programa; LCM_max y LCM_closed los cuales recuperarán el conjunto de patrones máximo y cerrado[16]. LCMFLOCK utiliza el concepto de patrones máximos y cerrados para identificar los patrones de agrupamiento de mayor duración. De esta manera, el parámetro δ se utiliza solo para filtrar aquellos patrones que no cumplan con este parámetro.

La salida de LCM es un archivo de texto, donde cada línea es un patrón que contiene un conjunto de ID's de discos separados por espacios que representan los patrones de agrupamiento (flocks). Sin embargo, se requiere de un análisis posterior para verificar que dichos discos ocurran en tiempos consecutivos. De cada patrón, se extraen los objetos que encierra cada disco así como los tiempos de inicio y final los cuales son almacenados en una base de datos.

[1]Repositorio del proyecto: https://github.com/poldrosky/FPFlock

[2]Scipy es un ecosistema basado en Python, software de código abierto para las matemáticas, la ciencia y la ingeniería. http://www.scipy.org/

Tabla I. Conjunto de datos validación

Dataset	Red	Número de Trayectorias	Número de Flocks	Instantes de tiempo
SJ5000T100t100f	Sintética	5000	100	100
SJ5000T100t200f	Sintetica	5000	200	100
SJ5000T100t300f	Sintetica	5000	300	100
SJ5000T100t400f	Sintetica	5000	400	100
SJ5000T100t500f	Sintetica	5000	500	100
SJ2500T100t500f	Sintética	2500	500	100
SJ7500T100t500f	Sintética	7500	500	100
SJ10000T100t500f	Sintética	10000	500	100
SJ12500T100t500f	Sintética	12500	500	100
SJ15000T100t500f	Sintética	15000	500	100
SJ17500T100t500f	Sintética	17500	500	100
SJ20000T100t500f	Sintética	20000	500	100

C. Validación

Para poder validar la correcta implementación de los algoritmos se siguió una metodología similar a la propuesta en [5]. Se crearon conjuntos de datos sintéticos a los cuales se les insertó aleatoriamente un número específico de trayectorias y flocks. La tabla I relaciona los conjuntos de datos construidos y con los cuales los algoritmos fueron validados. Una copia de los conjuntos de datos y el script utilizado para la validación está disponible en el repositorio del proyecto [3]. El total de flocks insertados en cada caso fueron correctamente descubiertos por las dos implementaciones.

IV. Experimentación Computacional

Los resultados fueron producidos usando conjuntos de datos sintéticos y reales en una máquina Dell OPTIPLEX 7010 con procesador Intel®Core™i7-3770 CPU de 3.40GHz x 8, 16 GB de RAM y 1TB 7200 RPM de Disco Duro, corriendo Debian con linux 3.2. Para todos los casos se usaron los algoritmos implementados en Python version 3.

A. San Joaquin

Un grupo de conjuntos de datos sintéticos fueron creados usando un modelo para la generación de objetos en movimiento, como se describe en [17]. Dos conjuntos de datos sintéticos fueron creados usando la red de San Joaquín proporcionada en el sitio web del generador [18]. El primer conjunto de datos recoge 992140 lugares simulados para 25.000 objetos en movimiento durante 60 instantes de tiempo. El segundo recoge 50.000 trayectorias de 2.014.346 de puntos durante 55 instantes de tiempo. La tabla II resume la información principal. Las figuras 2 y 3 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=5, δ=3 y μ=9, δ=3 respectivamente.

B. TAPAS Cologne

Este conjunto de datos sintético se preparó utilizando el escenario TAPAS Cologne [19] en SUMO [20], un reconocido simulador de tráfico para la movilidad urbana. El

[3]Conjuntos de prueba: https://github.com/poldrosky/FPFlock/tree/master/Src/Datasets/

Tabla II. CONJUNTO DE DATOS

Dataset	Red	Número de trayectorias	Número de puntos	Duración promedio de la trayectoria
SJ25KT60	San Joaquin	25000	992140	40
SJ50KT55	San Joaquin	50000	2014346	37
TAPAS Cologne	Cologne, Alemania	88668	3403463	38
Beijing_Original	Beijing, China	21573	1411846	65
Beijing_Alternativo	Beijing, China	18700	815657	43

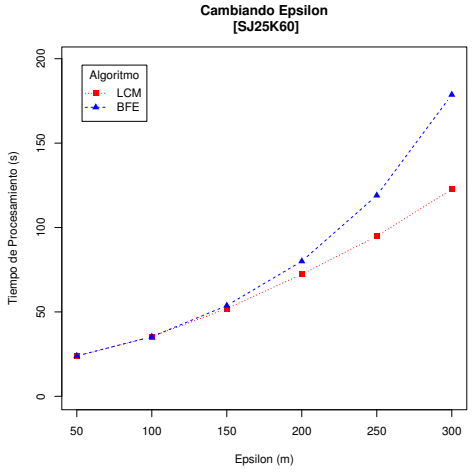


Figura 2. Caso de Prueba: SJ25K60

escenario de simulación TAPAS Cologne describe el tráfico dentro de la ciudad de Colonia (Alemania) durante un día entero. La principal ventaja de este conjunto de datos es que sus trayectorias no se generan aleatoriamente. Los datos de la demanda original, se deriva de TAPAS, un sistema que calcula la tendencia de movilidad para una población con base en la información sobre los hábitos de viaje de los alemanes y en la información sobre la infraestructura de la zona en que viven [21]. El conjunto de datos original es enorme por lo que sólo está disponible al público la versión de 2 horas [22]. Debido a las restricciones de memoria, se

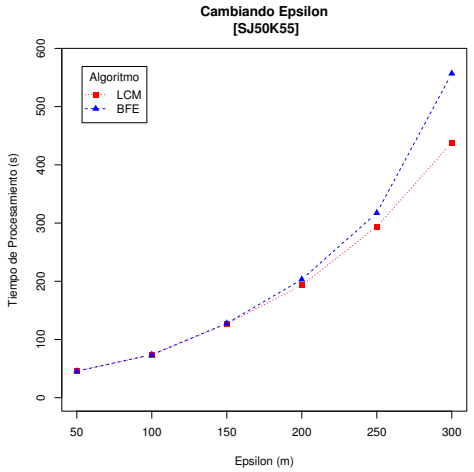


Figura 3. Caso de Prueba: SJ50K55

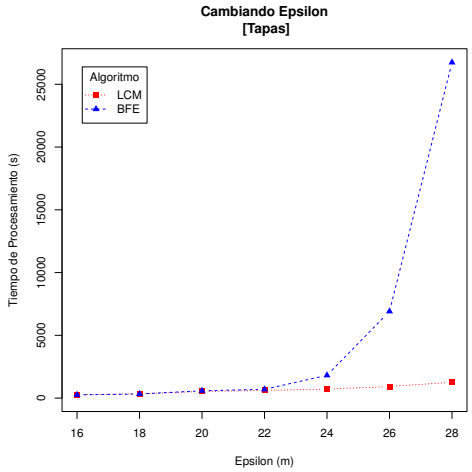


Figura 4. Caso de Prueba: Tapas Cologne

podaron las trayectorias más cortas que 20 minutos. El último conjunto de datos recoge 88.668 trayectorias y más de 3,4 millones de puntos. La tabla II describe los detalles sobre el conjunto de datos. Las figura 4 muestra los tiempos de desempeño para este caso de estudio, los parámetros adicionales fueron μ=10, δ=5.

C. Movimiento de peatones en Beijing

Este conjunto de datos reales recopila información de movimiento de un grupo de personas en toda el área metropolitana de Beijing, China[23]. El conjunto de datos se recogió durante el proyecto Geolife por 165 usuarios anónimos en un período de dos años entre abril de 2007 y agosto de 2009. Las ubicaciones fueron grabadas por diferentes dispositivos GPS o teléfonos inteligentes y la mayoría de ellos presentan una frecuencia de muestreo alta. La región alrededor del quinto anillo vial en el área metropolitana de Beijing mostró la mayor concentración de trayectorias. Esto fue usado para generar un conjunto de datos de muestra. Cada trayectoria fue interpolada por minuto (un punto por minuto) y saltos de 20 minutos o más sin señal se utilizaron para marcar una nueva trayectoria. Por último, el conjunto de datos recoge más de 1,4 millones de puntos y 21.573 trayectorias. Sin embargo, como este conjunto de datos tuvo una poca cantidad de entidades en movimiento (165 usuarios) en una ventana de tiempo de más de 2 años, no existieron muchas trayectorias ocurriendo al mismo tiempo. Para probar la escalabilidad se decidió crear un conjunto de datos alternativo basado en las trayectorias reales, pero forzando para que todas ellas comiencen al mismo tiempo. Una vez más, por las limitaciones de memoria, las trayectorias menores que 10 minutos y mayores que 3 horas se podaron. El conjunto de datos alternativo almacenó 815.657 ubicaciones y 18.700 trayectorias. La tabla II resume los detalles para ambos conjuntos de datos. Las figuras 5 y 6 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=3, δ=3 y μ=5, δ=5 respectivamente.

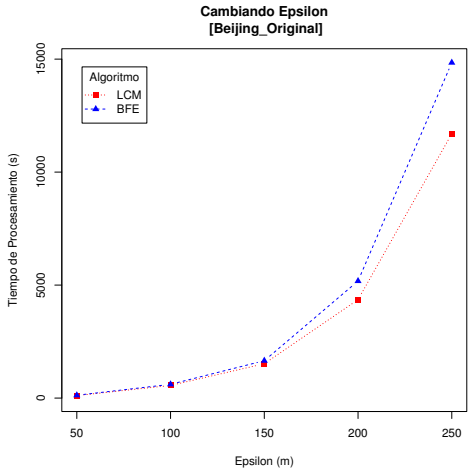


Figura 5. Caso de Prueba: Beijing Original

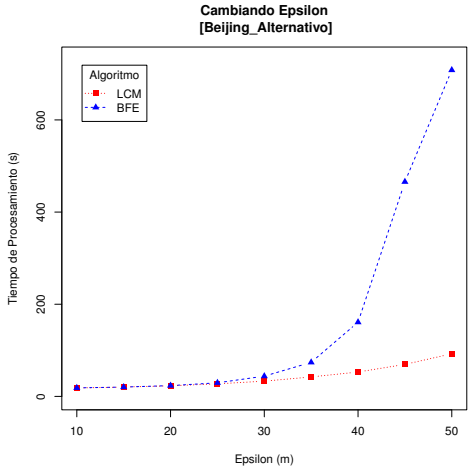


Figura 6. Caso de Prueba: Beijing Alternativo

D. Reporte de Flocks

Tanto para BFE como LCMFLOCK el reporte de flocks se hace en una base de datos, con un identificador, tiempo de inicio, tiempo de fin y todos los puntos contenidos en dicho flock. En la tabla III se muestra el número total de flocks reportados por BFE y LCMFLOCK con el ϵ más grande en cada caso de prueba.

Tabla III. Número de Flocks

Dataset	ϵ	Número de Flocks BFE	Número de Flocks LCMFlock
SJ25KT60	300	35805	5466
SJ50KT55	300	45201	6396
TAPAS Cologne	28	9415451	31509
Beijing_Original	250	16628029	4373139
Beijing_Alternativo	50	6110427	19233

V. Conclusiones y Trabajos Futuros

Durante el proceso de implementación se evaluaron diferentes estructuras de datos para optimizar las consultas espaciales. Después de las diferentes pruebas fue evidente que las estructuras de tipo árbol, y en específico, los kd-tree presentaron los mejores resultados. Esto se configura como un aspecto clave para la búsqueda del conjunto final de discos para cada instante de tiempo. Entre mejores implementaciones de este tipo de estructuras los resultados de ejecución mejorarían de manera considerable.

Con base en los resultados de los experimentos se puede observar un mejor desempeño por parte de LCMFLOCK en todos los conjuntos de datos evaluados. En particular, las diferencias fueron más notables al aumentar el valor de ϵ. Aunque el proceso de identificación de los discos en cada instante de tiempo es compartido por ambos algoritmos, el proceso de combinación es mucho más eficiente utilizando un enfoque basado en técnicas de patrones frecuentes. El proceso de combinación de discos efectuado por BFE inclusive llegó a colapsar a medida que el valor de ϵ iba creciendo.

Aunque LCMFLOCK presenta mejores resultados, este aún se ve afectado por el proceso de identificación de los discos. El número de discos a evaluar crece exponencialmente a medida que crece el valor de ϵ y la densidad de puntos dentro del conjunto. Los tiempos de respuesta en esta etapa afectan por igual a ambos algoritmos. Futuros trabajos deberían enfocarse en otras técnicas para minimizar el impacto de la búsqueda de los discos.

Para la implementación de los algoritmos, los autores de los algoritmos no hacen ninguna sugerencia de poder realizar la implementación en paralelo, para futuros trabajos se debe tratar de realizar un modelo de implementación en paralelo para usar el mayor rendimiento de la máquina en la que se esté probando.

En BFE, el reporte de flocks es demasiado alto en comparación con LCMFLOCK (tabla III). BFE separa los flocks de acuerdo al parámetro δ en procura de limitar el número de discos a comparar. Esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija. En LCMFLOCK esto se soluciona con el uso del algoritmo LCM para la detección de patrones máximos y cerrados.

La capacidad de encontrar los flocks más largos ciertamente es una ventaja de LCMFLOCK sobre BFE. Sin embargo, LCMFLOCK, a diferencia de BFE, requiere de una ventana fija de tiempo donde será aplicado lo que le impide reportar patrones en tiempo real. En ciertas aplicaciones, como seguridad y gestión de tráfico, ésta es una característica muy relevante. Vale la pena explorar con más profundidad mecanismos que permitan aplicar los algoritmos de detección de patrones frecuentes en tiempo real en este tipo de situaciones.

Los dos algoritmos evaluados han demostrado su funcionalidad y utilidad, sin embargo, la metodología basado en patrones frecuentes ha probado ser la de mejor desempeño. En el futuro, esta investigación se enfocará en dos

problemas claves de este enfoque: mejorar el desempeño durante la búsqueda de los discos e implementar mecanismos de detección en tiempo real.

Agradecimientos

Al sistema de investigación de la Universidad de Nariño (Colombia) por financiar la presente investigación en el marco de la convocatoria de tesis de grado.

Referencias

[1] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1161–1174, 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2007.1054

[2] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, C. Bauzer Medeiros, M. Egenhofer, and E. Bertino, Eds. Springer Berlin Heidelberg, 2005, vol. 3633, pp. 364–381. [Online]. Available: http://dx.doi.org/10.1007/11535331_21

[3] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1068–1080, 2008.

[4] J. Gudmundsson and M. van Kreveld, "Computing longest duration flocks in trajectory data," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS '06. New York, NY, USA: ACM, 2006, pp. 35–42. [Online]. Available: http://doi.acm.org/10.1145/1183471.1183479

[5] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle, "Reporting flock patterns," *Computational Geometry*, vol. 41, no. 3, pp. 111 – 125, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092577210700106X

[6] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. New York, NY, USA: ACM, 2009, pp. 286–295. [Online]. Available: http://doi.acm.org/10.1145/1653771.1653812

[7] U. Turdukulov, A. O. Calderon Romero, O. Huisman, and V. Retsios, "Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach," *International Journal of Geographical Information Science*, vol. 1, pp. 1–17, 0. [Online]. Available: http://dx.doi.org/10.1080/13658816.2014.889834

[8] P. Laube, M. van Kreveld, and S. Imfeld, "Finding remo — detecting relative motion patterns in geospatial lifelines," in *Developments in Spatial Data Handling*. Springer Berlin Heidelberg, 2005, pp. 201–215. [Online]. Available: http://dx.doi.org/10.1007/3-540-26772-7_16

[9] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining," in *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, ser. OSDM '05. New York, NY, USA: ACM, 2005, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/1133905.1133916

[10] A. O. C. Romero, "Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach," Master's thesis, University of Twente, 2011.

[11] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining user similarity based on location history," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:10. [Online]. Available: http://doi.acm.org/10.1145/1463434.1463477

[12] H. Jeung, H. T. Shen, and X. Zhou, "Convoy queries in spatio-temporal databases," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, April 2008, pp. 1457–1459.

[13] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[14] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *FIMI*, vol. 126, 2004.

[15] B. Goethals. (2004) Frequent itemset mining implementations repository. http://fimi.cs.helsinki.fi/. Consultado Julio 2014.

[16] J. Han and J. Pei, "Mining frequent patterns by pattern-growth: Methodology and implications," *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 14–20, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/380995.381002

[17] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1015231126594

[18] T. Birkhoff. (2005) Network-based generator of moving objects. http://iapg.jade-hs.de/personen/brinkhoff/generator/. Consultado Julio 2014.

[19] C. Varschen and P. Wagner, "Microscopic modeling of passenger transport demand based on time-use diaries," in *Integrated micro-simulation of land use and transport development. Theory, concepts, models and practice*, K. J. Beckmann, Ed., vol. 81, 2006, pp. 63–69.

[20] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)," in *Proc. of the 4th middle east symposium on simulation and modelling*, 2002, pp. 183–187.

[21] MiD2002 Project. (2002) Mobility in Germany 2002. http://daten.clearingstelle-verkehr.de/196/. Consultado Julio 2014.

[22] SUMO Project. (2011) TAPAS Cologne Scenario. http://sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html.

[23] Microsoft Research Asia. (2010) Geolife gps trajectories. http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx. Consultado Julio 2014.

FP-Flock: Un algoritmo para encontrar patrones frecuentes de agrupación en bases de datos espacio temporales

Omar Ernesto Cabrera Rosero* †, Andrés Oswaldo Calderon Romero* †.
Ricardo Timarán Pereira* † *Universidad de Nariño. †Grupo de Investigación Aplicada en Sistemas (GRIAS).

***Abstract*—El amplio uso de sistemas de localización como dispositivos GPS y RFID junto con el masivo uso dispositivos móviles han hecho que la disponibilidad y acceso a bases de datos espacio temporales se hayan incrementado de manera considerable durante los últimos años. Esta gran cantidad de datos ha motivado el desarrollo de técnicas más eficientes para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos móviles durante un periodo continuo de tiempo. Diversos estudios se han centrado en la consulta de patrones que capturan el comportamiento de los diversas entidades en movimiento, los cuales se reflejan en colaboraciones tales como clústers móviles, consulta de convoyes y patrones de agrupamiento. En este artículo se da a conocer una propuesta para descubrir patrones de agrupamiento, tradicionalmente conocidas como flocks, la cual esta basada en un enfoque de patrones frecuentes. Se presentan dos alternativas para la detección de patrones tanto en línea como off-line. Ambas alternativas fueron comparadas con dos algoritmos del mismo tipo, Basic Flock Evaluation (BFE) y LCMFlock. El desempeño y comportamiento se midió en distintos conjuntos de datos, tanto sintéticos como reales.**

Keywords*—*movement patterns, frequent patterns mining, spatio-temporal databases, flock patterns.

I. Introducción

LOS recientes avances tecnológicos y el amplio uso de la localización basada en sistemas de posicionamiento global (GPS), identificación por radio frecuencia (RFID) o tecnologías en dispositivos móviles han hecho que el acceso a bases de datos espacio temporales se haya incrementado de una manera acelerada. Esta gran cantidad de información ha motivado a desarrollar técnicas más eficientes para procesar consultas acerca del comportamiento de los objetos en movimiento, como descubrir patrones de comportamiento entre las trayectorias de objetos en un período contínuo de tiempo.

Sin embargo los métodos que existen para la consulta de trayectorias se centran principalmente en responder un único rango simple de predicado y consultas de vecinos más cercanos, por ejemplo: "encontrar todos los objetos en movimiento que se encontraban en la zona A a las 10 de la mañana" o "encontrar el coche que condujo cerca de la ubicación B durante el intervalo de tiempo de 10 de la mañana a 1 de la tarde". Recientemente, diversos estudios se han centrado en la consulta de los patrones para la captura del comportamiento de los objetos en movimiento reflejada en colaboraciones tales como clusters móviles [1] [2], consulta de convoyes [3] y patrones de agrupamiento [4] [5] [6] [7]. Estos patrones descubren grupos de objetos en movimiento que tienen una "fuerte" relación en el espacio durante un tiempo determinado. La diferencia entre todos esos patrones es la forma como se unen los objetos móviles entre intervalos de tiempo, su duración en el tiempo y la manera de cómo se combinan.

Este artículo se enfoca en el descubrimiento de patrones de agrupamiento, conocidos como "flocks", entre los objetos en movimiento de acuer-

O. Cabrera. Ingeniero de Sistemas, Universidad de Nariño (Pasto - Colombia). E-mail: omarcabrera@udenar.edu.co.

A. Calderón. Profesor hora catedra, Departamento de Sistemas, Universidad de Nariño (Pasto - Colombia), M.Sc en Geoinformática e Ingeniero de Sistemas. E-mail: aocalderon@udenar.edu.co.

R. Timarán. Profesor Asociado, Departamento de Sistemas, Universidad de Nariño (Pasto - Colombia), PhD. en Ingeniería, MSc. en Ingeniería, Esp. en Multimedia e Ingeniero de Sistemas y Computación. E-mail: ritimar@udenar.edu.co.

do a las características de los objetos de estudio (animales, peatones, vehículos o fenómenos naturales), cómo interactúan entre sí y cómo se mueven juntos [8] [9]. [6] define patrones de agrupamiento como el problema de identificar todos los grupos de trayectorias que permanecen "juntas" por la duración de un intervalo de tiempo dado. Consideramos que los objetos en movimiento están suficientemente cerca si existe un disco con un radio dado que cubre todos los objetos que se mueven en el patrón (figura 1). Una trayectoria satisface el patrón anterior, siempre y cuando suficientes trayectorias están contenidos dentro del disco para el intervalo de tiempo especificado, es decir, la respuesta se basa no sólo en el comportamiento de una trayectoria dada, sino también en las más cercanas a ella. Uno de los enfoques para descubrir patrones móviles de agrupamiento consiste en encontrar un conjunto adecuado de discos en cada instante de tiempo y luego combinar los resultados de un instante de tiempo a otro. Como consecuencia, el rendimiento y el número de patrones encontrados depende del número de los discos y cómo éstos se combinan.

En el ejemplo de la figura 1 se muestra un patrón de agrupamiento el cual contienen tres trayectorias (T1, T2 y T3) que están dentro de un disco en tres instantes de tiempo consecutivos. Los discos se pueden mover libremente en el espacio bidimensional con el fin de acomodar los tres objetos en movimiento y su centro no necesariamente tiene que ser la localización de alguno de los objetos. Esto hace que el descubrimiento de patrones sea mucho más complicada porque hay un número infinito de posibles colocaciones del disco en cualquier instante de tiempo y el posible número de combinaciones puede llegar a ser muy alta y costosa.

La implementación de este tipo de análisis tiene diversas aplicaciones tales como: sistemas integrados de transporte, seguridad y monitoreo, seguimiento a grupos de animales y fenómenos naturales. El analizar como se mueven los objetos en la tierra y cuáles son los patrones de comportamiento que existen entre sí puede aportar valiosa información para dar solución a diversos problemas en el mundo.

En este artículo se propone un nuevo algoritmo llamado FPFLOCK el cual se compara con los algoritmos propuestos por [6] y [10]. Se realizaron

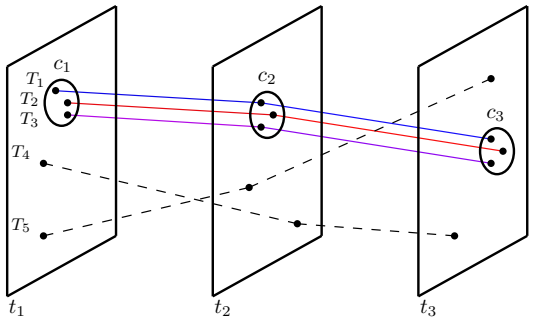


Figura 1. Ejemplo de patrones de agrupamiento.

pruebas con distintos conjuntos de datos, tanto reales como sintéticos. Se escogieron únicamente estos algoritmos para la comparación debido a que este análisis está enfocado a patrones de agrupamiento (flocks) y estos son los algoritmos más representativos que existen hasta el momento en este campo. Además las implementaciones de los algoritmos mencionados están disponibles en [11].

El resto del artículo esta organizado de la siguiente manera. En la sección trabajos relacionados se describen varios estudios que se han realizado en la temática de objetos móviles como algoritmos de clústers, convoyes y patrones de agrupamiento. En la sección 3 se describe la alternativa propuesta junto con el pseudocódigo del algoritmo y las pruebas de los mismos. En la sección experimentación computacional se presenta los conjuntos de datos y se compara el rendimiento de los algoritmos. En la siguiente sección se realiza una discusión de los resultados de la comparación de la alternativa propuesta frente a los otros algoritmos. Por último, se presentan las conclusiones y trabajos futuros.

II. Trabajos Relacionados

Las capacidad de recolectar datos de objetos en movimiento ha ido aumentando rápidamente y el interés de consulta de patrones que describen el comportamiento colectivo también ha aumentado. [6] enumera tres grupos de patrones "colectivos" en bases de datos de objetos en movimiento: clústers móviles, consulta de convoyes y patrones de agrupamiento.

Los clústers móviles [1] [2] [12] y consultas de convoyes [3] [13], tienen en común que se basan en algoritmos de clústering, principalmente en

algoritmos basados en densidad como el algoritmo DBSCAN [14].

Los clústers móviles se definen entre dos instantes de tiempo consecutivos. Los clústers se pueden unir sólo si el número de objetos comunes entre ellos están por encima del parámetro predefinido. Un clúster es reportado si no hay otro nuevo clúster que pueda ser unido a éste. Este proceso se aplica cada vez para todos los instantes de tiempo en el conjunto de datos.

Las consultas de convoyes se definen como un clúster denso de trayectorias que permanecen juntas al menos por un tiempo contínuo predefinido.

Las principales diferencias entre las dos técnicas son la forma en que se unen los grupos entre dos intervalos consecutivos de tiempo y el uso de un parámetro adicional para especificar un tiempo mínimo de duración. Aunque estos métodos están estrechamente relacionados con los patrones de agrupamiento, ninguno de ellos asume una forma predefinida.

Previos trabajos de detección de patrones de agrupamiento móviles son descritos por [4] y [5]. Ellos introducen el uso de discos con un radio predefinido para identificar grupos de trayectorias que se mueven juntos en la misma dirección, todas las trayectorias que se encuentran dentro del disco en un instante de tiempo particular se considera un patrón candidato. La principal limitación de este proceso es que hay un número infinito de posibles ubicaciones del disco en cualquier instante de tiempo. En efecto, en [4] se ha demostrado que el descubrimiento de agrupaciones fijas, donde los patrones de las mismas entidades permanecen juntas durante todo el intervalo, es un problema NP-complejo.

[6] son los primeros en presentar una solución exacta para reportar patrones de agrupación en tiempo polinomial, y también pueden trabajar efectivamente en tiempo real. Su trabajo revela que el tiempo de solución polinomial se puede encontrar a través de la identificación de un número discreto de ubicaciones para colocar el centro del disco. Los autores proponen el algoritmo BFE (Basic Flock Evaluation) basado en el tiempo de unión y combinación de los discos. La idea principal de este algoritmo es primero encontrar el número de discos válidos en cada instante de tiempo y luego combinarlos uno a uno entre tiempos adyacentes. Adicionalmente se proponen otros cuatro algoritmos basados en métodos heurísticos, para reducir el número total de candidatos a ser combinados y, por lo tanto, el costo global del algoritmo. Sin embargo, el pseudocódigo y los resultados experimentales muestran todavía una alta complejidad computacional, largos tiempos de respuesta, debido a que este algoritmo usa δ para partir los flocks hace que el número de patrones encontrados sea mayor y esto hace difícil su interpretación.

[10] y [7] proponen una metodología que permite identificar patrones de agrupamiento utilizando tradicionales y potentes algoritmos de minería de datos usando patrones frecuentes, el cual fue comparado con BFE demostrando un alto rendimiento con conjuntos de datos sintéticos, aunque con conjuntos de datos reales el tiempo de respuesta siguió siendo eficiente pero similar a BFE. Este algoritmo trata el conjunto de trayectorias como una base de datos transaccional al convertir cada trayectoria, que se define como un conjunto de lugares visitados, en una transacción, definida como un conjunto de ítems. De esta manera, es posible aplicar cualquier algoritmo de reglas de asociación y encontrar patrones frecuentes sobre el conjunto dado, este algoritmo hace un llamado a LCM propuesto por [15] para encontrar patrones máximos y eso permite encontrar los flocks más largos.

III. Alternativa propuesta

En el algoritmo propuesto por [16], en la primera parte del algoritmo se encuentran el total de discos con los objetos móviles que están juntos, se eliminan discos que no cumplan μ, este proceso genera que se encuentren varios de los discos solapados, para ello por último se realiza una limpieza de los discos solapados haciendo iteraciones en los discos encontrados y dejando únicamente aquellos discos con mayor número de miembros denominados discos máximos, este proceso de limpieza de los discos solapados hace que el costo computacional sea muy alto debido a la cantidad de iteraciones. En la segunda parte, el algoritmo realiza un proceso de combinación para el descubrimiento de flocks en el cual si el ε aumenta el algoritmo puede llegar a colapsar. Este algoritmo separa los flocks de acuerdo al parámetro δ en procura de limitar el número de

discos a comparar, esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija.

Para el algoritmo propuesto por [10], ya que usa la primera parte del algotitmo en [16], tiene el mismo problema de los discos solapados. Ya en la segunda parte, al tratar el proceso de combinación es mucho más eficiente ya que utiliza un enfoque basado en técnicas de patrones frecuentes, aunque con la desventaja que el proceso se realiza en una ventaja fija de tiempo lo que le impide reportar patrones en tiempo real.

La alternativa propuesta resuelve los problemas que poseen los algoritmos anteriores utilizando patrones frecuentes, se proponen el algoritmo llamado FPFlock con dos variaciones, uno fuera de linea y otro en tiempo real.

A. FPFLockOffline

En la primera parte se hace el cálculo de los discos máximos, para lo cual se tomó como base el algoritmo 1 propuesto por [16] al cual se le hicieron modificaciones para hacer una limpieza de los discos solapados usando patrones frecuentes de minería. Puntualmente, se uso el algoritmo LCM [9]. El pseudo-código de la primera parte del algoritmo es presentado en Algoritmo 1.

Algoritmo 1 Computing maximal disks.

Input: set of points $T[t_i]$ for timestamp t_i
Output: sets of maximal disks C

$C \leftarrow \emptyset$
Index.Build($T[t_i], \varepsilon$) {call Algorithm 1 in [16]}
for each non-empty cell $g_{x,y} \in$ Index **do**
    $L_r \leftarrow g_{x,y}$
    $L_s \leftarrow [g_{x-1,y-1} ... g_{x+1,y+1}]$
    if $|L_s| \geq \mu$ **then**
        for each $l_r \in L_r$ **do**
            $H \leftarrow Range(l_r, \varepsilon), |H| \geq \mu, d(l_r, l_s) \leq \varepsilon, l_s \in L_s$
            for each $l_j \in H$ **do**
                if $\{l_r, l_j\}$ not yet computed **then**
                    compute left disk $\{c\}$ defined by l_r, l_j and diameter ε
                    D $\leftarrow$ points $\in c$
                end if
            end for
        end for
    end if
    $min\_sup \leftarrow 1$
    C $\leftarrow$ call $LCM\_max(D, min\_sup)$ {call LCM Algorithm [9]}
end for

El algoritmo fuera de linea, se construyó usando el pseudo-código propuesto en [7], pero utilizando el algoritmo descrito anteriormente.

B. FPFlockOnline

El algoritmo en tiempo real, hace modificaciones al algoritmo propuesto en [7], este algoritmo en su primera parte usa el Algoritmo 1, para solucionar el problema de los discos solapados. En su segunda parte se va liberando memoria en cada transacción, teniendo en cuenta las trayectorias que en ese instante de tiempo tuvieron un corte. En este algoritmo por cada intervalo de tiempo se realiza un llamado al algoritmo LCM propuesto por [9] con el objetivo de reportar los patrones obtenidos hasta el momento.

El pseudo-código del algoritmo en tiempo real es presentado en Algoritmo 2

C. Validación

Para poder validar la correcta implementación de los algoritmos se siguió una metodología similar a la propuesta en [5]. Se crearon conjuntos de datos sintéticos a los cuales se les insertó aleatoriamente un número específico de trayectorias y flocks. La tabla I relaciona los conjuntos de datos construidos y con los cuales los algoritmos fueron validados. Durante la validación se evaluaron las implementación de BFE y LCMFlock junto con las implementación de las dos variaciones del algoritmo propuesto. Una copia de los conjuntos de datos y el script utilizado para la validación está disponible en el repositorio del proyecto [1]. El total de flocks insertados en cada caso fueron correctamente descubiertos por las cuatro implementaciones.

Para realizar una validación visual se utilizó el conjunto de datos de Oldenburg disponible en [17], el cual proporciona un conjunto de ejemplos y recursos que pueden ser utilizados en la demostración en línea o versión descargable del generador. Para empezar, se utilizó un conjunto de datos relativamente pequeña posición de 1.000 objetos en movimiento al azar en la ciudad alemana de Oldenburg. Los datos de la red (bordes y nodos) están disponibles en el sitio web. La

[1]Conjuntos de prueba: https://github.com/poldrosky/FPFlock/tree/master/Src/Datasets/

Tabla I. Conjunto de datos validación

Dataset	Red	Número de Trayectorias	Número de Flocks	Instantes de tiempo
SJ5000T100t100f	Sintética	5000	100	100
SJ5000T100t200f	Sintetica	5000	200	100
SJ5000T100t300f	Sintetica	5000	300	100
SJ5000T100t400f	Sintetica	5000	400	100
SJ5000T100t500f	Sintetica	5000	500	100
SJ2500T100t500f	Sintética	2500	500	100
SJ7500T100t500f	Sintética	7500	500	100
SJ10000T100t500f	Sintética	10000	500	100
SJ12500T100t500f	Sintética	12500	500	100
SJ15000T100t500f	Sintética	15000	500	100
SJ17500T100t500f	Sintética	17500	500	100
SJ20000T100t500f	Sintética	20000	500	100

Algoritmo 2 FPFlockOnline: Frequent pattern flock online.

Input: parameters μ, ε and δ, set of points T
Output: flock patterns F

for each new time instance $t_i \in T$ **do**
  $C \leftarrow$ call $Index.Disks(T[t_i])$ {call Algorithm 1 in this paper}
  for each $c_i \in C$ **do**
    $P \leftarrow c_i.points$
    for each $p_i \in P$ **do**
      $c_i.time \leftarrow t_i$
      $D[p_i] \leftarrow$ add $c_i.id$
    end for
    for each $p_i \in P$ **do**
      if $D[p_i]$ not was updated **then**
        delete $D[p_i]$
      end if
    end for
  end for
  $min\_sup \leftarrow \mu$
  $M \leftarrow$ call $LCM\_max(D, min\_sup)$ {call LCM Algorithm [9]}
  for each $max\_pattern \in M$ **do**
    $id_0 \leftarrow max\_pattern[0]$
    $c_0 \leftarrow C[id_0]$
    $u \leftarrow c_0.points$
    $u.t_{start} \leftarrow c_0.time$
    $n \leftarrow max\_pattern.size$
    for $i = 1$ **to** n **do**
      $id_i \leftarrow max\_pattern[i]$
      $c_i \leftarrow C[id_i]$
      if $c_i.time = c_{i-1}.time + 1$ **then**
        $u \leftarrow u \cap c_i.points$
        $u.t_{end} \leftarrow c_i.time$
      else
        if $u.t_{end} - u.t_{start} > \delta$ **and** $u \notin F$ **then**
          $F \leftarrow$ add u
          $u.t_{start} \leftarrow c_i.time$
        end if
      end if
    end for
    if $u.t_{end} - u.t_{start} > \delta$ **and** $u \notin F$ **then**
      $F \leftarrow$ add u
    end if
  end for
end for

simulación de datos recoge la latitud y longitud de los puntos generados durante 140 intervalos de tiempo. El número total de ubicaciones almacenadas es 57.016 puntos. Con este conjunto se construyeron mapas con las representaciones lineales de los flocks resultantes como lo muestra la tabla II con las cuatro implementaciones. En la figura 2 se muestran los flocks obtenidos con los parámetros ε=300, μ=3 y δ=3 sobre el conjunto Oldenburg. Estos mapas se los comparó usando un módulo que implementa la similitud estructural métrica de imagen (SSIM) [18]. Los resultados de la comparación muestran que todos los mapas son idénticos, y por lo tanto, los mismos flocks son reportados por los diferentes algoritmos.

IV. Experimentación Computacional

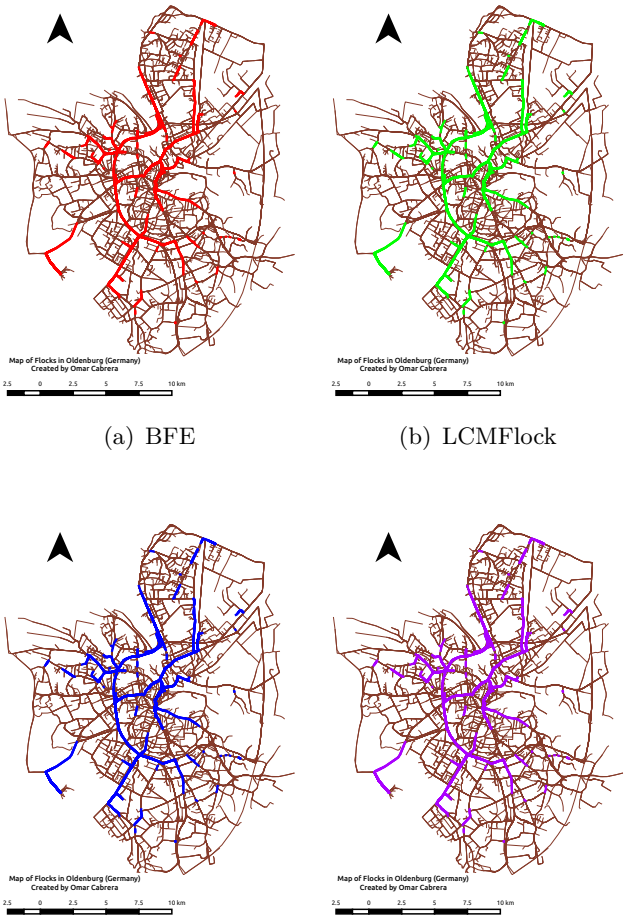
Los resultados fueron producidos usando conjuntos de datos sintéticos y reales en una máquina Dell OPTIPLEX 7010 con procesador Intel®Core™i7-3770 CPU de 3.40GHz x 8, 16 GB de RAM y 1TB 7200 RPM de Disco Duro, corriendo Debian con linux 3.2. Para todos los casos se usaron los algoritmos implementados en Python version 3.

A. San Joaquin

Un grupo de conjuntos de datos sintéticos fueron creados usando un modelo para la generación de objetos en movimiento, como se describe en [19]. Dos conjuntos de datos sintéticos fueron creados usando la red de San Joaquín proporcionada en el sitio web del generador [17]. El primer conjunto de datos recoge 992140 lugares simulados para 25.000 objetos en movimiento durante 60

Tabla II. Número de flocks generados por los algoritmos en el conjunto de Oldenburg μ=3, δ=3

$\varepsilon(m)$	BFE	LCM	FPFlockOnline	FPFlockOffline
50	131	27	150	27
100	639	109	663	109
150	1135	247	1226	247
200	2755	523	2683	523
250	5423	1150	6877	1150
300	11196	2365	16671	2365



(a) BFE (b) LCMFlock

(c) FPFlockOnline (d) FPFlockOffline

Figura 2. Visualización Oldenburg

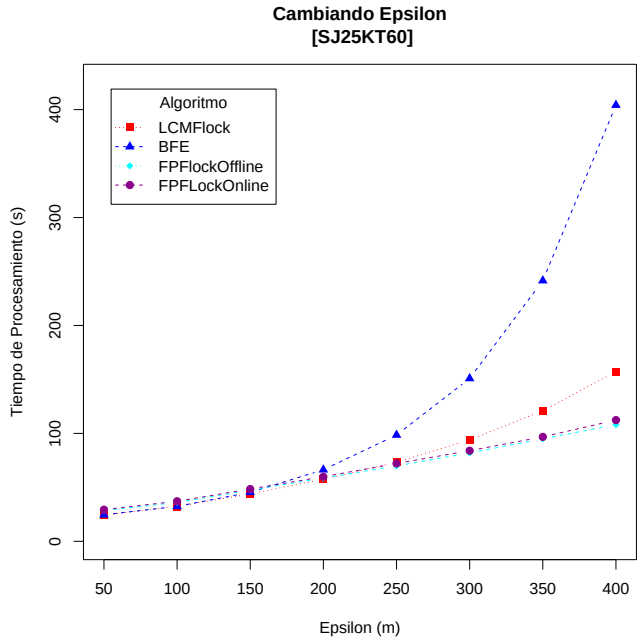
instantes de tiempo. El segundo recoge 50.000 trayectorias a partir de 2.014.346 de puntos durante 55 instantes de tiempo. La tabla III resume la información principal. Las figuras 3 y 4 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=5, δ=3 y μ=9, δ=3 respectivamente.

B. TAPAS Cologne

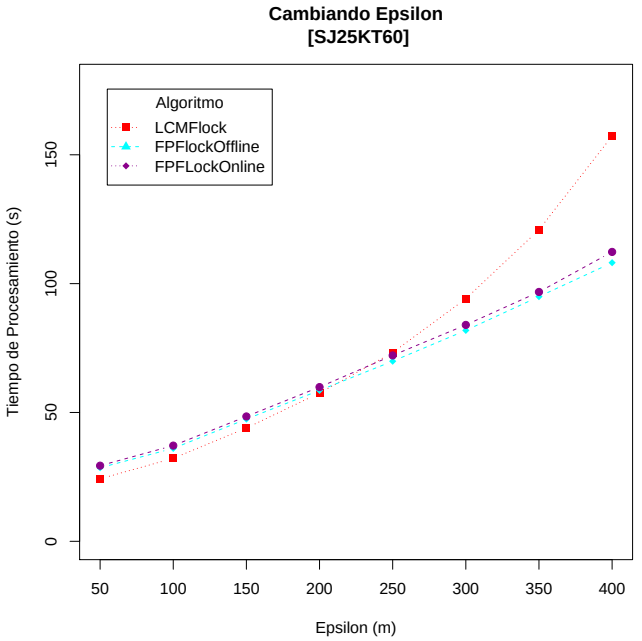
Este conjunto de datos sintético se preparó utilizando el escenario TAPAS Cologne [20] en SUMO [21], un reconocido simulador de tráfico para la movilidad urbana. El escenario de simulación TAPAS Cologne describe el tráfico dentro de la ciudad de Colonia (Alemania) durante un día entero. La principal ventaja de este conjunto de datos es que sus trayectorias no se generan aleatoriamente. Los datos de la demanda original, se deriva de TAPAS, un sistema que calcula la tendencia de movilidad para una población con base en la información sobre los hábitos de viaje de los alemanes y en la información sobre la infraestructura de la zona en que viven [22]. El conjunto de datos original es enorme por lo que sólo está disponible al público la versión de 2 horas [23]. Debido a las restricciones de memoria, se podaron las trayectorias más cortas que 20 minutos. El último conjunto de datos recoge 88.668 trayectorias y más de 3,4 millones de puntos. La tabla III describe los detalles sobre el conjunto de datos. Las figura 5 muestra los tiempos de desempeño para este caso de estudio. Los parámetros adicionales fueron μ=10, δ=5.

C. Movimiento de peatones en Beijing

Este conjunto de datos reales recopila información de movimiento de un grupo de personas en toda el área metropolitana de Beijing, China [24]. El conjunto de datos se recogió durante el proyecto Geolife por 165 usuarios anónimos en un período de dos años entre abril de 2007 y agosto de 2009. Las ubicaciones fueron grabadas por diferentes dispositivos GPS o teléfonos inteligentes y la mayoría de ellos presentan una frecuencia de muestreo alta. La región alrededor del quinto anillo vial en el área metropolitana de Beijing mostró la mayor concentración de trayectorias. Esto fue usado para generar un conjunto de datos

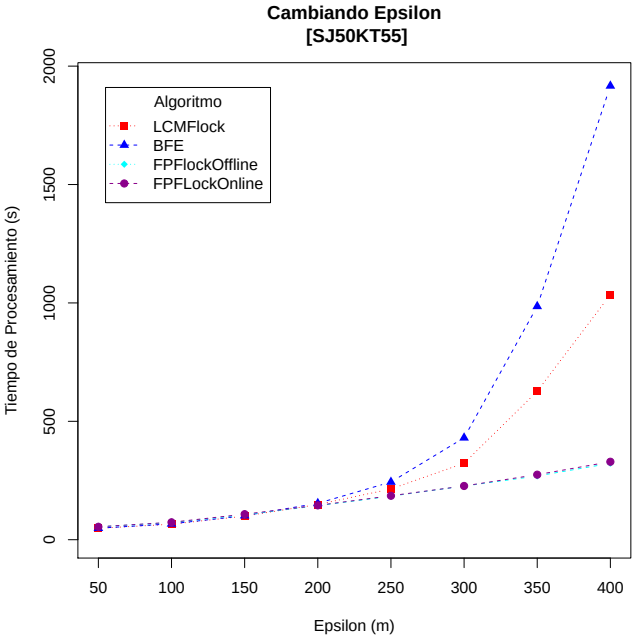


(a) BFE, LCMFlock, FPFlock

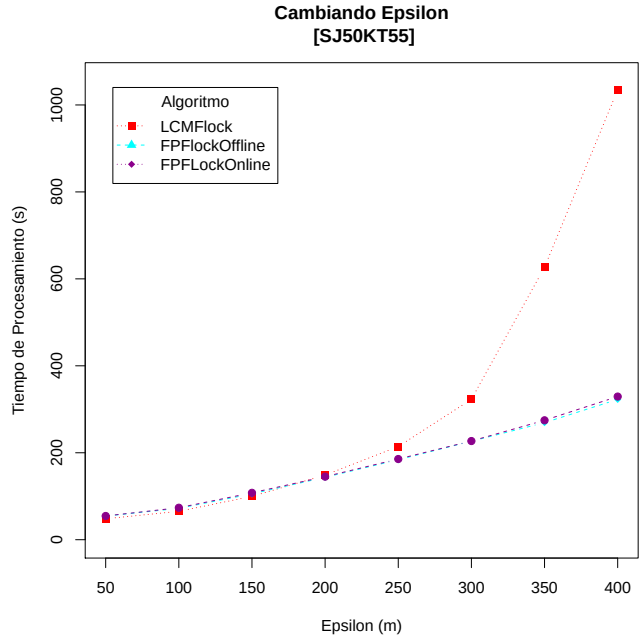


(b) LCMFlock, FPFlock

Figura 3. Caso de Prueba: SJ25KT60. (a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock



(a) BFE, LCMFlock, FPFlock

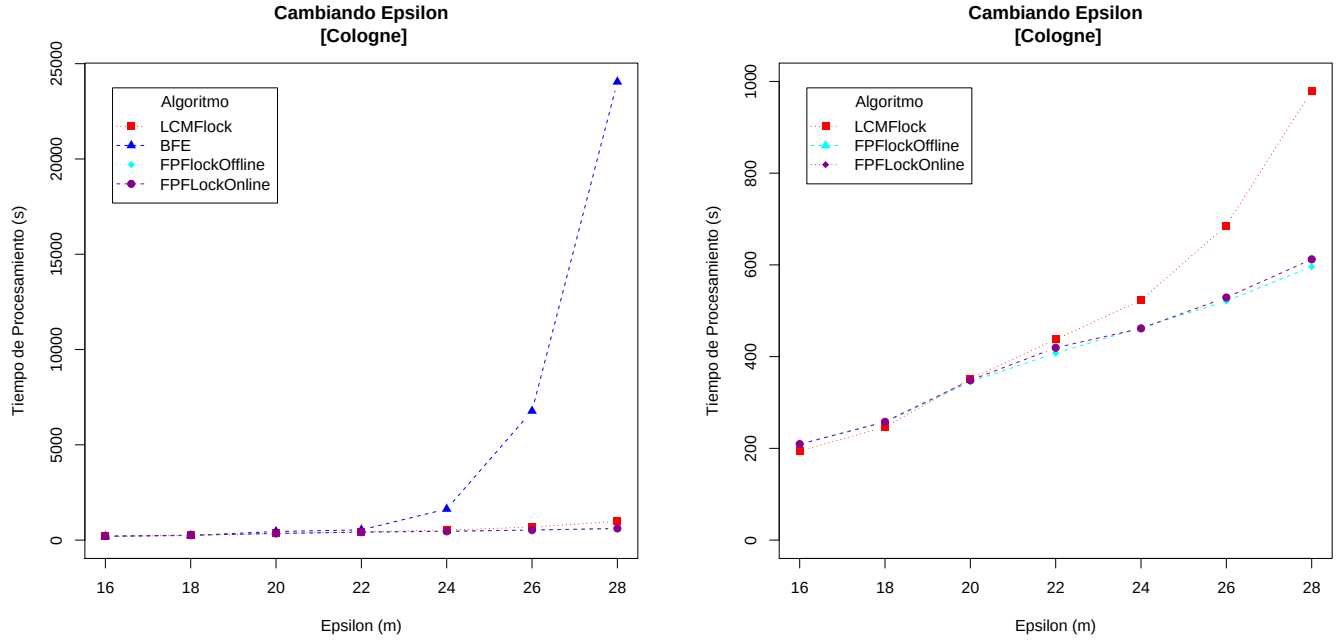


(b) LCMFlock, FPFlock

Figura 4. Caso de Prueba: SJ50KT55. (a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

Tabla III. CONJUNTO DE DATOS

Dataset	Red	Número de trayectorias	Número de puntos	Duración promedio de la trayectoria
SJ25KT60	San Joaquin	25000	992140	40
SJ50KT55	San Joaquin	50000	2014346	37
TAPAS Cologne	Cologne, Alemania	88668	3403463	38
Beijing_Original	Beijing, China	21573	1411846	65
Beijing_Alternativo	Beijing, China	18700	815657	43



(a) BFE, LCMFlock, FPFlock

(b) LCMFlock, FPFlock

Figura 5. Caso de Prueba: Tapas Cologne. (a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

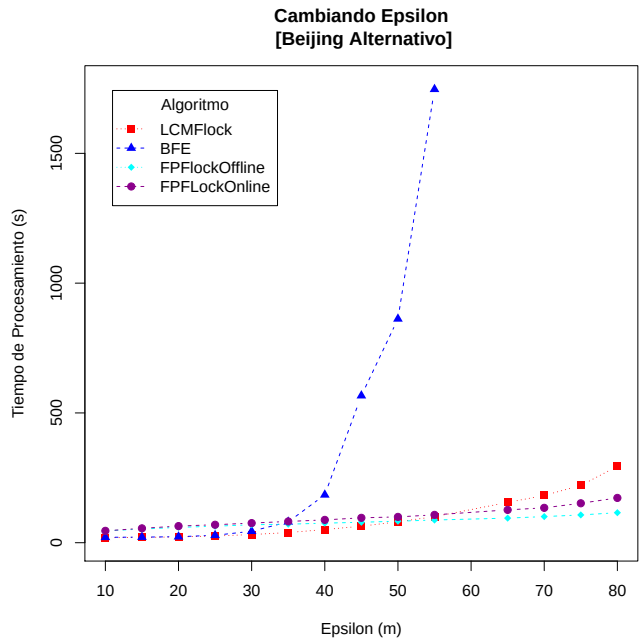
de muestra. Cada trayectoria fue interpolada por minuto (un punto por minuto) y saltos de 20 minutos o más sin señal se utilizaron para marcar una nueva trayectoria. Por último, el conjunto de datos recoge más de 1,4 millones de puntos y 21.573 trayectorias. Sin embargo, como este conjunto de datos tuvo una poca cantidad de entidades en movimiento (165 usuarios) en una ventana de tiempo de más de 2 años, no existieron muchas trayectorias ocurriendo al mismo tiempo. Para probar la escalabilidad de los algoritmos, se decidió crear un conjunto de datos alternativo basado en las trayectorias reales, pero forzando para que todas ellas comiencen al mismo tiempo. Una vez más, por las limitaciones de memoria, las trayectorias más cortas que 10 minutos y más largas que 3 horas se podaron. El conjunto de datos alternativo almacenó 815.657 ubicaciones y 18.700 trayectorias. La tabla III resume los detalles para ambos conjuntos de datos. Las figuras 6 y 7 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=3, δ=3 y μ=5, δ=5 respectivamente.

D. Reporte de Flocks

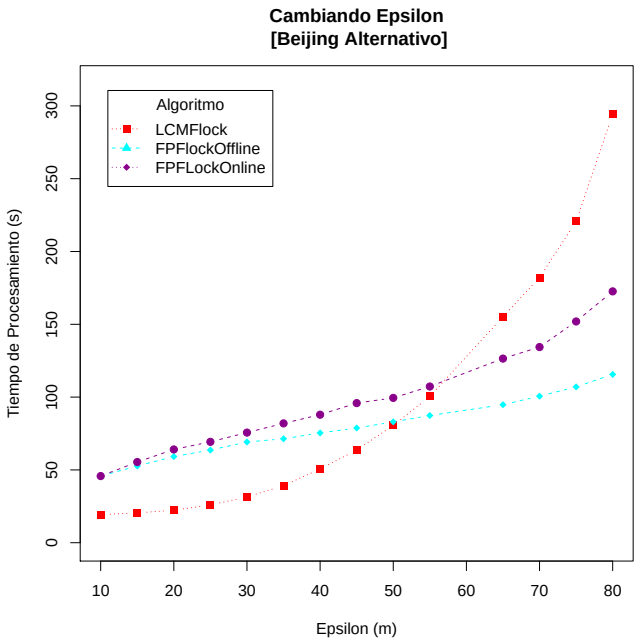
Tanto para BFE, LCMFLOCK, FPFlockOnline y FPFlockOffline el reporte de flocks se hace en una base de datos, con un identificador, tiempo de inicio, tiempo de fin y todos los puntos contenidos en dicho flock. En la tabla IV se muestra el número total de flocks reportados con un ε en particular.

V. DISCUSIÓN

Para hacer una comparación de rendimiento de los algoritmos, es importante que se los evalué de acuerdo a las características de cada algoritmo,



(a) BFE, LCMFlock, FPFlock



(b) LCMFlock, FPFlock

Figura 7. Caso de Prueba: Beijing Alternativo. (a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

Tabla IV. Número de Flocks

Dataset	ϵ	Número de Flocks BFE	Número de Flocks LCMFlock	Número de Flocks FPFlockOnline	Número de Flocks FPFlockOffline
SJ25KT60	300	35805	5466	26877	5466
SJ50KT55	300	45201	6396	23638	6396
TAPAS Cologne	28	9415451	31509	145217	31509
Beijing_Original	250	16628029	4373139	-	4373139
Beijing_Alternativo	50	6110427	19233	63566	19233

para ello se debe comparar el algoritmo BFE propuesto por [16] con la alternativa FPFlockOnline y el algoritmo LCMFlock propuesto por [10] con la alternativa FPFlockOffline.

En el conjunto de San Joaquin, se puede mirar que el algoritmo de BFE, crece muy rápidamente a medida que crece ε, más aún en el rango entre ε 300 y 400, como lo muestra la figura 3 y la figura 4. LCMFlock en ese mismo rango de ε se comporta de igual manera pero en menor escala. La alternativa propuesta se comporta de una manera lineal.

En el conjunto de Tapas Cologne, el algoritmo de BFE, luego de un ε mayor a 24 se incrementa muy rapido hasta colapsar como lo muestra en la figura 5(a). De igual manera, LCMFlock se incrementa luego de un ε mayor a 24 pero en menor escala. La alternativa propuesta se comporta de una manera lineal.

En el conjunto de Beijing original se puede observar que la alternativa en tiempo real con ε igual a 200 colapsa, los algoritmos BFE y LCMFlock crecen rapidamente con ε mayor a 200, (BFE más rapidamente que LCMFlock), mientras que la alternativa fuera de línea tiene un buen rendimiento con respecto a los otros algoritmos. Esto hace pensar que la alternativa en tiempo real colapsa debido a que en la implementación del algoritmo de LCM, se escribe y se lee en disco repetidamente debido a que el conjunto tiene varios instantes de tiempo.

En el conjunto de Beijing alternativo, el algoritmo BFE incrementa su tiempo de respuesta muy rapidamente con un ε mayor a 40 y llega a colapsar en un ε igual a 50, como lo muestra la figura 7(a). En la figura 7(b), se observa un rápido incremento en la ejecución de LCMFlock con un ε mayor a 50.

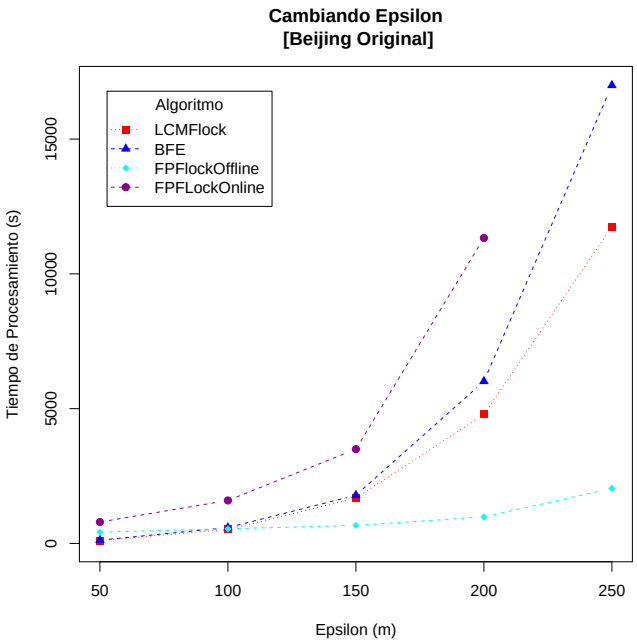


Figura 6. Caso de Prueba: Beijing Original

En general, para todos los conjuntos, se observa que la alternativa propuesta en la mayoría de los casos tiene un mejor desempeño sobre BFE y LCMFlock a medida que aumenta ε, esto debido al enfoque basado en técnicas de patrones frecuentes.

En BFE, el reporte de flocks es demasiado alto en comparación con LCMFLOCK (tabla IV). BFE separa los flocks de acuerdo al parámetro δ en procura de limitar el número de discos a comparar. Esto dificulta la interpretación de resultados ya que es necesario de un análisis adicional para encontrar los flocks más largos a partir de aquellos con una duración fija. En LCMFLOCK esto se soluciona con el uso del algoritmo LCM para la detección de patrones máximos y cerrados. En FPFlock para la interpretación de flocks resultantes no requiere hacer un análisis adicional ya que en la alternativa en tiempo real los flocks más largos son reportados en cada instante de tiempo y en la alternativa fuera de linea el reporte de flocks es igual a LCMFLock.

La capacidad de encontrar los flocks más largos ciertamente es una ventaja de LCMFLOCK como de la alternativa fuera de línea sobre BFE y la alternativa en tiempo real. Sin embargo, los algoritmos fuera de linea, a diferencia de los que son en tiempo real, requieren de una ventana fija de tiempo donde ser aplicados. En ciertas aplicaciones, como seguridad y gestión de tráfico, ésta es una característica muy relevante. Vale la pena explorar con más profundidad mecanismos que permitan aplicar los algoritmos de detección de patrones frecuentes en tiempo real en este tipo de situaciones.

VI. Conclusiones y Trabajos Futuros

Se diseñó una propuesta llamada FPFlock con dos variaciones: una fuera de línea y otra en tiempo real, utilizando un enfoque basado en la detección de patrones frecuentes. La propuesta ha demostrado un buen rendimiento en diferentes casos de prueba, tanto sintéticos como reales.

Durante el proceso de implementación se evaluaron diferentes estructuras de datos para optimizar las consultas espaciales. Después de las diferentes pruebas fue evidente que las estructuras de tipo árbol, y en específico, los kd-tree presentaron los mejores resultados. Esto se configura como un aspecto clave para la búsqueda del conjunto final de discos para cada instante de tiempo. Entre mejores implementaciones de este tipo de estructuras los resultados de ejecución mejorarían de manera considerable.

El enfocar esta investigación en una metodología basada en patrones frecuentes, mejoró el desempeño durante la búsqueda de los discos y el descubrimiento de flocks en tiempo real, dando un punto de partida para realizar implementaciones similares para la optimización usando este enfoque.

Sin embargo, en la alternativa propuesta se realizó un llamado al algoritmo LCM, disponible de forma binaria como una aplicación externa. Para solucionar el problema de reiterados llamados de lectura y escritura en disco, se debe implementar de manera nativa el algoritmo de LCM dentro de la solución propuesta.

Para la implementación de los algoritmos, los autores de los algoritmos no hacen ninguna sugerencia de poder realizar la implementación en paralelo, para futuros trabajos se recomienda realizar un modelo de implementación en paralelo para usar el mayor rendimiento de la máquina en la que se esté probando.

Agradecimientos

Al sistema de investigación de la Universidad de Nariño (Colombia) por financiar la presente investigación.

Referencias

[1] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1161–1174, 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2007.1054

[2] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, C. Bauzer Medeiros, M. Egenhofer, and E. Bertino, Eds. Springer Berlin Heidelberg, 2005, vol. 3633, pp. 364–381. [Online]. Available: http://dx.doi.org/10.1007/11535331_21

[3] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1068–1080, 2008.

[4] J. Gudmundsson and M. van Kreveld, "Computing longest duration flocks in trajectory data," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS '06. New York, NY, USA: ACM, 2006, pp. 35–42. [Online]. Available: http://doi.acm.org/10.1145/1183471.1183479

[5] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle, "Reporting flock patterns," *Computational Geometry*, vol. 41, no. 3, pp. 111 – 125, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092577210700106X

[6] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. New York, NY, USA: ACM, 2009, pp. 286–295. [Online]. Available: http://doi.acm.org/10.1145/1653771.1653812

[7] U. Turdukulov, A. O. Calderon Romero, O. Huisman, and V. Retsios, "Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach," *International Journal of Geographical Information Science*, vol. 1, pp. 1–17, 2014. [Online]. Available: http://dx.doi.org/10.1080/13658816.2014.889834

[8] P. Laube, M. van Kreveld, and S. Imfeld, "Finding remo — detecting relative motion patterns in geospatial lifelines," in *Developments in Spatial Data Handling*. Springer Berlin Heidelberg, 2005, pp. 201–215. [Online]. Available: http://dx.doi.org/10.1007/3-540-26772-7_16

[9] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining," in *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, ser. OSDM '05. New York, NY, USA: ACM, 2005, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/1133905.1133916

[10] A. O. C. Romero, "Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach," Master's thesis, University of Twente, 2011.

[11] O. E. Cabrera Rosero and A. O. Calderón Romero, "Performance analysis of flock pattern algorithms in spatio-temporal databases," in *CLEI2014*, 2014.

[12] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining user similarity based on location history," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:10. [Online]. Available: http://doi.acm.org/10.1145/1463434.1463477

[13] H. Jeung, H. T. Shen, and X. Zhou, "Convoy queries in spatio-temporal databases," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, April 2008, pp. 1457–1459.

[14] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[15] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *FIMI*, vol. 126, 2004.

[16] M. Vieira and V. Tsotras, "Flock pattern queries," in *Spatio-Temporal Databases*, ser. SpringerBriefs in Computer Science. Springer International Publishing, 2013, pp. 61–83. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02408-0_4

[17] T. Birkhoff. (2005) Network-based generator of moving objects. http://iapg.jade-hs.de/personen/brinkhoff/generator/. Consultado Julio 2014.

[18] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, April 2004.

[19] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1015231126594

[20] C. Varschen and P. Wagner, "Microscopic modeling of passenger transport demand based on time-use diaries," in *Integrated micro-simulation of land use and transport development. Theory, concepts, models and practice*, K. J. Beckmann, Ed., vol. 81, 2006, pp. 63–69.

[21] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)," in *Proc. of the 4th middle east symposium on simulation and modelling*, 2002, pp. 183–187.

[22] MiD2002 Project. (2002) Mobility in Germany 2002. http://daten.clearingstelle-verkehr.de/196/. Consultado Julio 2014.

[23] SUMO Project. (2011) TAPAS Cologne Scenario. http://sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html.

[24] Microsoft Research Asia. (2010) Geolife gps trajectories. http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx. Consultado Julio 2014.