FP-Flock: An algorithm to find frequent flock patterns in spatio-temporal databases

Omar Ernesto Cabrera Rosero[1] and Andrés Oswaldo Calderón Romero[2]

[1] Universidad de Buenos Aires
omarcabrera@udenar.edu.co
[2] Universidad de California, Riverside
acald013@ucr.edu

Abstract The widespread use of location systems such as GPS and RFID along with the massive use of mobile devices have allowed a significant increase 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 entities in motion, which are reflected in collaborations such as mobile clusters, convoy queries and flock patterns. In this paper, we provided an algorithm to find clustering patterns, traditionally known as flocks, which is based on a frequent pattern mining approach. Twoalternatives 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.

1 Introducción

Los recientes avances tecnológicos y el amplio uso de plataformas de ubicación y localización basadas en sistemas de posicionamiento global (GPS), identificación por radio frecuencia (RFID) y tecnologías disponibles 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 el desarrollo de técnicas más eficientes para procesar consultas acerca del comportamiento de objetos en movimiento, como por ejemplo, descubrir patrones de comportamiento entre las trayectorias de objetos en un período continuo de tiempo.

Recientemente, diversos estudios se han centrado en la consulta de los patrones que capturen el comportamiento de los objetos en movimiento, entre ellos se

destacan los grupos móviles [7] [10] [13], consulta de convoyes [9] [8] y patrones de agrupamiento [6] [1] [22] [18] [5]. Estos patrones descubren grupos de objetos en movimiento que tienen una "fuerte" relación espacial durante un periodo de tiempo determinado. La diferencia fundamental entre dichos patrones es la forma como se asocian en el espacio, como se combinan entre intervalos de tiempo consecutivos y la posible duración de los mismos.

Este artículo se enfoca en el descubrimiento de patrones de agrupamiento, conocidos en inglés como "flocks". La relevancia de este tipo de patrones es el análisis único que hay entre objetos en movimiento de acuerdo con las características de los objetos de estudio (animales, peatones, vehículos o fenómenos naturales), cómo interactúan entre sí y cómo se mueven juntos [12] [6].

En [22] los autores definen los patrones de agrupamiento como el problema de identificar **todos** los grupos de trayectorias que permanecen "juntas" durante un intervalo de tiempo dado y define 3 parámetros iniciales para su descubrimiento: una distancia mínima (ε), un mínimo número de objetos (μ) y un mínimo intervalo de tiempo (δ). Consideramos que los objetos en movimiento están suficientemente cerca si existe un disco con un radio dado (ε) que cubre todos los objetos que se mueven en el patrón (figura 1). Una trayectoria satisface el patrón de agrupamiento, siempre y cuando haya un mínimo número de trayectorias (μ) contenidas dentro del disco para un intervalo de tiempo especificado (δ), 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 que 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 complicado 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. El descubrimiento de patrones de agrupamiento tiene diversas aplicaciones, tales como: sistemas integrados de transporte, seguridad y monitoreo de multitudes, diseño de infraestructuras urbanas, soporte a procesos de evacuaciones y respuesta a emergencias, seguimiento a grupos de animales en entornos ecológicos, estudio e implicaciones de fenómenos naturales, entre otros. La aplicación de dichos patrones provee una alternativa diferente para solucionar problemas cotidianos mediante el análisis de la movilidad entre las entidades involucradas y cuáles son los patrones de comportamiento que existen entre ellas.

En este artículo se propone un nuevo algoritmo llamado FPFLOCK el cual se compara con los algoritmos propuestos por [22] y [16]. Se realizaron pruebas con distintos conjuntos de datos, tanto reales como sintéticos. Se escogieron úni-

camente 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 [4].

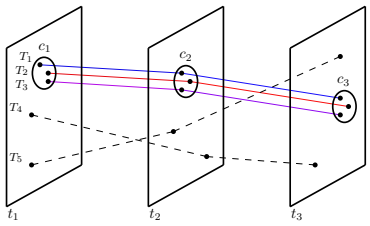


Figura 1. Ejemplo de patrones de agrupamiento.

El resto del artículo está organizado de la siguiente manera: en la sección de trabajos relacionados se describen varios estudios que se han realizado en la temática de objetos móviles como algoritmos de agrupación, convoyes y patrones de agrupamiento. En la sección 3 se describe la alternativa propuesta junto con el pseudocódigo del algoritmo y las pruebas realizadas. En la sección de experimentación computacional se compara el rendimiento de los algoritmos. Por último, se presentan las conclusiones y trabajos futuros.

2 Trabajos Relacionados

Trabajos previos de detección de patrones móviles de agrupamiento son descritos por [6] y [1]. 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 consideran 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 [6] 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.

En [22] se presenta la primera solución exacta para reportar todos los posibles patrones de agrupamiento en tiempo polinomial. También es interesante que la solución propuesta puede 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 donde el centro de cada disco puede ser ubicado. 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 y largos tiempos de respuesta. Igualmente, el algoritmo usa el parámetro δ para reportar rápidamente un patrón una vez es encontrado y lo descarta de inmediato para reducir el número de combinaciones. Esto genera un número abrumador de patrones reportados lo que dificulta de manera directa su interpretación.

[16] y [18] proponen una metodología que permite identificar patrones de agrupamiento utilizando algoritmos de minería de datos tradicionales aplicados en el descubrimiento de patrones frecuentes. La solución propuesta fue comparada con BFE y demostró un alto rendimiento con grandes conjuntos de datos sintéticos y similares resultados con conjuntos de datos reales de menor tamaño. Este algoritmo trata el conjunto de trayectorias como una base de datos transaccional al convertir cada trayectoria en una transacción definida como el conjunto de discos visitados en cada instante de tiempo por cada una de las entidades en movimiento. De esta manera, es posible aplicar cualquier algoritmo definido en el ámbito del descubrimiento de reglas de asociación y encontrar patrones frecuentes sobre el conjunto dado. En este caso, el método propuesto hace un llamado al algoritmo LCM_max, propuesto por [19], para encontrar patrones frecuentes máximos y, de esta manera, estar en la capacidad de reportar los patrones de agrupamiento más largos.

3 Alternativa propuesta

La primera parte del algoritmo propuesto por [23] encuentra el total de discos con un diámetro predefinido que contengan un mínimo número de objetos móviles establecido por el parámetro μ. El objetivo de esta primera fase es determinar aquellos objetos móviles que se encuentran relativamente cerca uno del otro de acuerdo al parámetro ε. Sin embargo, este procedimiento puede arrojar un gran número de discos repetidos o redundantes. Definimos que un disco es redundante si existe otro disco que contiene a todos los objetos encontrados por el primero y, al menos, otro objeto más. En este caso, solo el disco con el mayor número de objetos debería conservarse y se denomina un *disco máximo*. El proceso de selección de los discos máximos hace que el costo computacional sea muy alto debido a la gran cantidad de iteraciones.

En la segunda parte, el algoritmo realiza un proceso de combinación para la conformación y descubrimiento de los patrones de agrupamiento que ocurran al menos durante un intervalo de tiempo definido por el parámetro δ. Esta etapa inicia con el descubrimiento del conjunto de discos máximos en el primer instante de tiempo. Posteriormente, se encuentra el conjunto de discos máximos en el siguiente instante disponible. Cuando se han analizado dos instantes consecutivos se procede a determinar que grupos de objetos móviles han permanecido juntos durante dicho periodo al combinar los discos encontrados en el primer instante de tiempo con aquellos encontrados en el segundo. Una vez se han establecido las combinaciones se crea un conjunto de discos candidatos que contabilizan cuantos

instantes de tiempo los objetos cubiertos por los discos han permanecido juntos. A partir de aquí, el proceso se repite con el conjunto de discos candidatos y el conjunto de discos máximos que se encontrará el siguiente instante de tiempo disponible. Después de cada combinación, se analiza el conjunto de discos candidatos y se reporta aquellos que han superado el mínimo número de intervalos de tiempo establecido por el parámetro δ.

Este procedimiento puede resultar particularmente costoso dependiendo del tamaño del conjunto de discos candidatos y los conjuntos de discos máximos encontrados en cada instante de tiempo. Como un mecanismo para controlar el número de combinaciones una vez que se reporta un patrón de agrupamiento se elimina del conjunto de discos candidatos. Aunque esto disminuye el tamaño del conjunto de discos candidatos también genera dificultades a la hora de la interpretación de los resultados. Si un grupo de objetos permanece juntos por un largo periodo de tiempo, su patrón de agrupamiento será reportado de manera segmentada, en intervalos de duración de acuerdo al parámetro δ. Esto genera una explosión en el número de patrones reportados y dificulta de manera importante su comprensión.

El algoritmo propuesto por [16], usa la primera parte del algoritmo propuesto en [23] para la identificación de discos máximos. Sin embargo, trata el proceso de combinación de manera más eficiente al utilizar un enfoque basado en técnicas de identificación de patrones frecuentes ampliamente utilizados en el área del descubrimiento de reglas de asociación. Aunque el procedimiento de combinación es mucho más rápido cuenta con la desventaja de que el proceso se debe realizar 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 al proponer el algoritmo llamado FPFlock con dos variaciones: uno fuera de línea y otro en tiempo real.

3.1 FPFLockOffline

En la primera parte se identifica el conjunto de discos máximos tomando como base el algoritmo 1 propuesto por [23] al cual se le hicieron modificaciones para eliminar posibles discos repetidos y redundantes usando técnicas de minería de datos para el descubrimiento de patrones frecuentes. Puntualmente, se usó el algoritmo LCM propuesto en [20]. El pseudo-código de la primera parte del algoritmo es presentado en Algoritmo 1.

Es importante aclarar que el uso del algoritmo LCM en particular entrega importantes beneficios en el despeño computacional. En [19] se aclara que dicho algoritmo esta en capacidad de encontrar el conjunto de patrones frecuentes en tiempo lineal. Las actuales implementaciones para la remoción de discos repetidos y redundantes durante la primera fase y la combinación de los conjuntos de discos candidatos y discos máximos durante la segunda fase toman cada una $O(n^2)$ de acuerdo al tamaño de cada conjunto. En general, BFE resuelve cada caso de forma básica al combinar cada elemento de las listas involucradas y filtrando los resultados obtenidos.

El algoritmo fuera de línea, se construyó usando el pseudo-código propuesto en [18], en el cual se varía la primera parte y se utiliza el Algoritmo 1 para calcular los discos máximos. El pseudo-código del algoritmo fuera de línea es presentado en el Algoritmo 2.

3.2 FPFlockOnline

El algoritmo en tiempo real, hace modificaciones al algoritmo propuesto en [18]. Este algoritmo en su primera parte usa el Algoritmo 1, para solucionar el problema de la identificación y remoción de discos repetidos o redundantes. 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 [20] con el objetivo de reportar los patrones obtenidos hasta el momento. El pseudo-código del algoritmo en tiempo real es presentado en el Algoritmo 3.

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 [23]}
for each non-empty cell $g_{x,y} \in$ Index **do**
  $L_r \leftarrow g_{x,y}$
  $L_s \leftarrow [g_{x-1,y-1} \cdots g_{x+1,y+1}]$
  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 [20]}
end for

3.3 Validación

Para poder validar la correcta implementación de los algoritmos se siguió una metodología similar a la propuesta en [1]. Se crearon conjuntos de datos sintéticos a los cuales se les insertó aleatoriamente un número específico de trayectorias y patrones de agrupamiento. La tabla 1 relaciona los conjuntos de datos construidos. Durante la validación se evaluaron las implementaciones de BFE y LCMFlock junto con las implementaciones de las dos variaciones del algoritmo propuesto. Una copia de los conjuntos de datos y el código fuente utilizado para la validación está disponible en el repositorio del proyecto [3]. El total de patrones

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

de agrupamiento insertados en cada caso fueron correctamente descubiertos por las cuatro implementaciones.

Algoritmo 2 FPFlockOffline: Frequent pattern flock offline

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
end for
$min\_sup \leftarrow \mu$
$M \leftarrow$ call $LCM\_max(D, min\_sup)$ {call LCM Algorithm [20]}
for each $max\_pattern \in M$ **do**
  $id_0 \leftarrow max\_pattern[0]$
  $c_0 \leftarrow C[id_0]$
  $u \leftarrow c_0.points$
  $u.t_{start} \leftarrow c_0.time$
  $n \leftarrow max\_pattern.size$ {number of items in $max\_pattern$}
  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** {are disks consecutive?}
      $u \leftarrow u \cap c_i.points$
      $u.t_{end} \leftarrow c_i.time$
    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
return F

Algoritmo 3 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 [20]}
  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
return F

Para realizar una validación visual se utilizó el conjunto de datos de Oldenburg disponible en [2], el cual proporciona un conjunto de ejemplos y recursos que pueden ser utilizados en la demostración en línea o versión disponible para descarga del generador. Para empezar, se utilizó un conjunto de datos relativamente pequeño con 1000 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 del generador[4]. La simulación de datos recoge la latitud y longitud de los puntos generados durante 140 instantes de tiempo. El número total de ubicaciones almacenadas es 57016 puntos. Con este conjunto se construyeron mapas con las representaciones lineales de los patrones de agrupamiento resultantes como lo muestra la tabla 2 con las cuatro implementaciones.

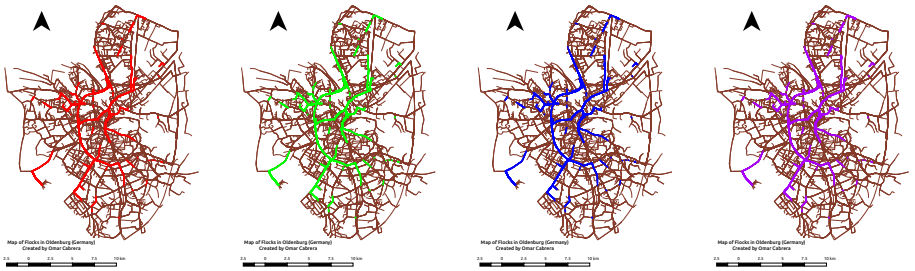
Tabla 1. Conjunto de datos validación

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

Tabla 2. Flocks generados 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

En la figura 2 se muestran los patrones de agrupamiento obtenidos con los parámetros ε=300, μ=3 y δ=3 sobre el conjunto Oldenburg. Estos mapas fueron comparados usando un como criterio la similitud estructural métrica de imagen (SSIM). Los resultados de la comparación muestran que todos los mapas son idénticos y el mismo conjunto de patrones de agrupamiento fueron reportados por los diferentes algoritmos.



(a) BFE (b) LCMFlock (c) FPFlockOnline (d) FPFlockOffline

Figura 2. Visualización Oldenburg

[4] `https://iapg.jade-hs.de/personen/brinkhoff/generator/`

4 Experimentación Computacional

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

4.1 San Joaquín

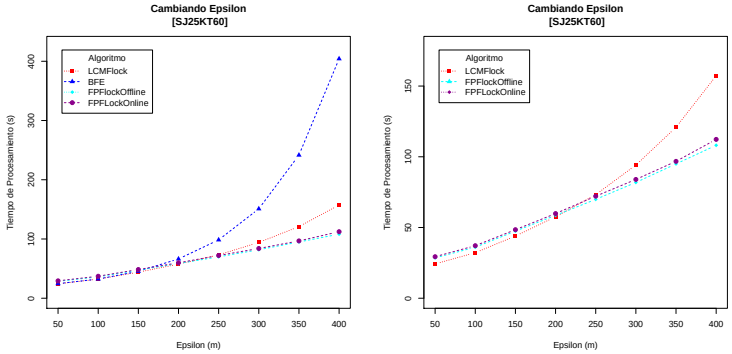
Un grupo de conjuntos de datos sintéticos fueron creados usando un modelo para la generación de objetos en movimiento, como se describe en [3]. Dos conjuntos de datos sintéticos fueron creados usando la red de San Joaquín proporcionada en el sitio web del generador [2]. El primer conjunto de datos recoge 992140 lugares simulados para 25000 objetos en movimiento durante 60 instantes de tiempo. El segundo recoge 50000 trayectorias a partir de 2014346 de puntos durante 55 instantes de tiempo. La tabla 3 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.

Tabla 3. 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

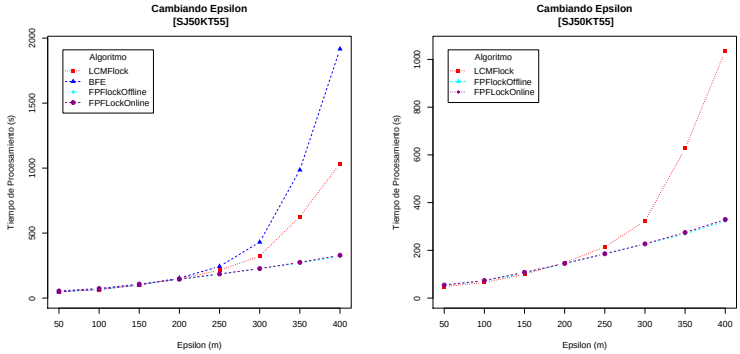
4.2 TAPAS Cologne

Este conjunto de datos sintético se preparó utilizando el escenario TAPAS Cologne [21] en SUMO [11], 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 derivan 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 [15]. El conjunto de datos original es enorme por lo que sólo está disponible al público la versión de 2 horas [17]. Debido a las restricciones de memoria, se podaron las trayectorias con duraciones más corta a 20 minutos. El conjunto de datos final recoge 88668 trayectorias y más de 3.4 millones de puntos. La tabla 3 describe los detalles sobre el conjunto de datos. La figura 5 muestra los tiempos de desempeño para este caso de estudio. Los parámetros adicionales fueron μ=10, δ=5.



(a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

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

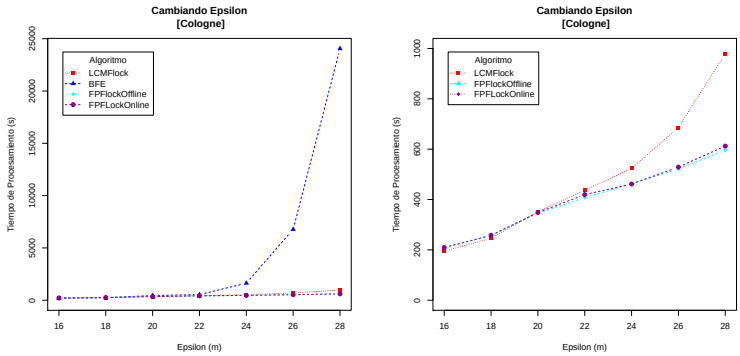


(a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

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

4.3 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 [14]. 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



(a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

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

20 minutos o más sin señal se utilizaron para marcar una nueva trayectoria. El conjunto de datos final recoge más de 1.4 millones de puntos y 21573 trayectorias.

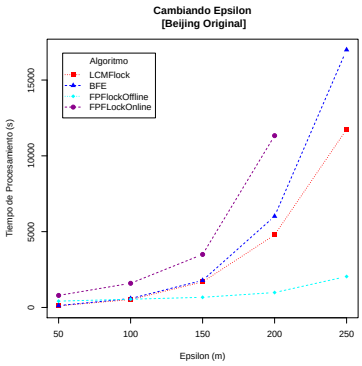
Sin embargo, como este conjunto de datos tiene una cantidad de entidades en movimiento relativamente baja (165 usuarios) en una ventana de tiempo de más de 2 años, no existieron muchas trayectorias ocurriendo al mismo tiempo. Para probar la escalabilidad de los algoritmos, se decidió crear un conjunto de datos alternativo basado en las trayectorias reales, pero modificando su fecha de inicio para que todas ellas comiencen al mismo tiempo. Una vez más, por las limitaciones de memoria, las trayectorias más cortas a 10 minutos y más largas que 3 horas se podaron. El conjunto de datos alternativo almacenó 815657 ubicaciones y 18700 trayectorias. La tabla 3 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.

4.4 Reporte de Flocks

Tanto para BFE, LCMFLOCK, FPFlockOnline y FPFlockOffline el reporte de patrones de agrupamiento se hace en una base de datos, con un identificador, tiempo de inicio, tiempo de fin y todos los puntos contenidos en dicho patrón. En la tabla 4 se muestra el número total de patrones de agrupamiento reportados con un valor de ε en particular.

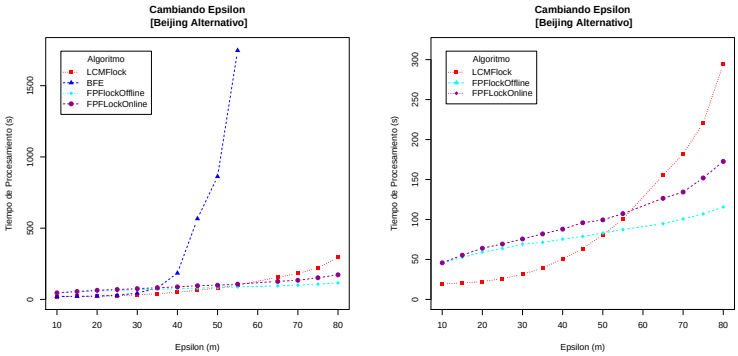
5 Conclusiones y Trabajos Futuros

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



h!

Figura 6. Caso de Prueba: Beijing Original



(a) BFE, LCMFlock, FPFlock (b) LCMFlock, FPFlock

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

Tabla 4. 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

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 pre-

sentaron los mejores resultados. Esto se configura como un aspecto clave para la búsqueda del conjunto de discos máximos 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 máximos y el descubrimiento de patrones de agrupamientos en tiempo real, dando un punto de partida para realizar implementaciones similares para su 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.

Como futuros trabajos se espera explorar una nueva implementación en paralelo que permita mejorar aún más los tiempos de respuesta y el aprovechamiento total de recursos en especial durante el descubrimiento del conjunto de discos máximos. Igualmente, una mejor presentación de los patrones obtenidos resultaría clave para el entendimiento de la información descubierta durante el análisis. Nuevas técnicas de cartografía y visualización de datos podrían ser de gran ayuda para inferir nuevo conocimiento a partir de los resultados obtenidos.

Referencias

1. Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting flock patterns. Computational Geometry **41**(3), 111 – 125 (2008), http://www.sciencedirect.com/science/article/pii/S092577210700106X
2. Birkhoff, T.: Network-based generator of moving objects. http://iapg.jade-hs.de/personen/brinkhoff/generator/ (2005), consultado Julio 2014
3. Brinkhoff, T.: A framework for generating network-based moving objects. Geoinformatica **6**(2), 153–180 (Jun 2002), http://dx.doi.org/10.1023/A:1015231126594
4. Cabrera, O., Calderón, A.: Performance analysis of flock pattern algorithms in spatio-temporal databases. In: 2014 XL Latin American Computing Conference (CLEI). pp. 1–6 (Sept 2014). https://doi.org/10.1109/CLEI.2014.6965180
5. Cao, Y., Zhu, J., Gao, F.: An algorithm for mining moving flock patterns from pedestrian trajectories. In: Web Technologies and Applications. pp. 310–321. Springer International Publishing, Cham (2016)
6. Gudmundsson, J., van Kreveld, M.: Computing longest duration flocks in trajectory data. In: Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems. pp. 35–42. GIS '06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1183471.1183479
7. Jensen, C.S., Lin, D., Ooi, B.C.: Continuous clustering of moving objects. IEEE Transactions on Knowledge and Data Engineering **19**(9), 1161–1174 (2007). https://doi.org/10.1109/TKDE.2007.1054
8. Jeung, H., Shen, H.T., Zhou, X.: Convoy queries in spatio-temporal databases. In: Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. pp. 1457–1459 (April 2008). https://doi.org/10.1109/ICDE.2008.4497588

9. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of convoys in trajectory databases. Proceedings of the VLDB Endowment **1**(1), 1068–1080 (2008)
10. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Advances in Spatial and Temporal Databases, Lecture Notes in Computer Science, vol. 3633, pp. 364–381. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/11535331_21
11. Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P.: Sumo (simulation of urban mobility). In: Proc. of the 4th middle east symposium on simulation and modelling. pp. 183–187 (2002)
12. Laube, P., van Kreveld, M., Imfeld, S.: Finding remo — detecting relative motion patterns in geospatial lifelines. In: Developments in Spatial Data Handling, pp. 201–215. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/3-540-26772-7_16
13. Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., Ma, W.Y.: Mining user similarity based on location history. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 34:1–34:10. GIS '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1463434.1463477
14. Microsoft Research Asia: Geolife gps trajectories. `http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx` (2010), consultado Julio 2014
15. MiD2002 Project: Mobility in Germany 2002. `http://daten.clearingstelle-verkehr.de/196/` (2002), consultado Julio 2014
16. Romero, A.O.C.: Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach. Master's thesis, University of Twente (2011)
17. SUMO Project: TAPAS Cologne Scenario. `http://sumo-sim.org/userdoc/Data/Scenarios/TAPASCologne.html` (2011), consultado Julio 2014
18. Turdukulov, U., Calderon Romero, A.O., Huisman, O., Retsios, V.: Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach. International Journal of Geographical Information Science **1**, 1–17 (2014). https://doi.org/10.1080/13658816.2014.889834
19. Uno, T., Kiyomi, M., Arimura, H.: Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: FIMI. vol. 126 (2004)
20. Uno, T., Kiyomi, M., Arimura, H.: 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. pp. 77–86. OSDM '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1133905.1133916
21. Varschen, C., Wagner, P.: Microscopic modeling of passenger transport demand based on time-use diaries. In: Beckmann, K.J. (ed.) Integrated micro-simulation of land use and transport development. Theory, concepts, models and practice. vol. 81, pp. 63–69 (2006)
22. Vieira, M.R., Bakalov, P., Tsotras, V.J.: 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. pp. 286–295. GIS '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1653771.1653812
23. Vieira, M., Tsotras, V.: Flock pattern queries. In: Spatio-Temporal Databases, pp. 61–83. SpringerBriefs in Computer Science, Springer International Publishing (2013). https://doi.org/10.1007/978-3-319-02408-0_4