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

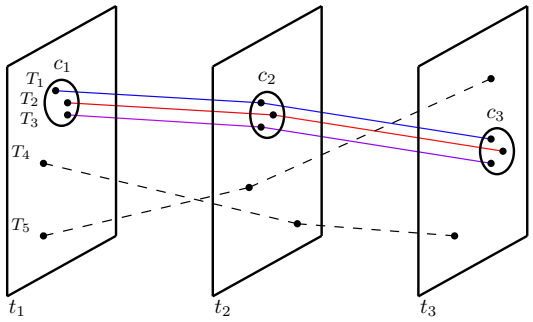


Figura 1. Ejemplo de patrones de agrupamiento.

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

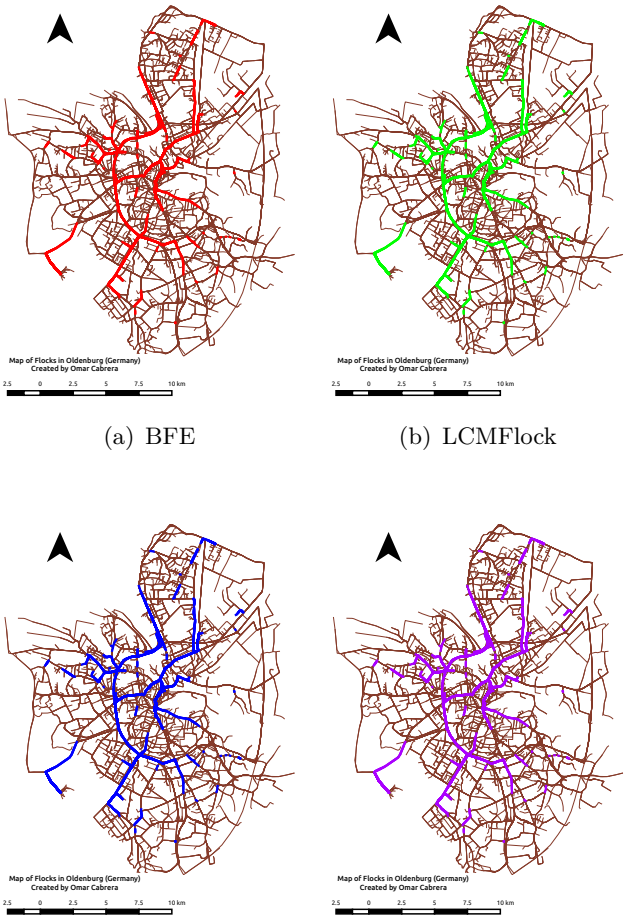
Los resultados fueron producidos usando conjuntos de datos sintéticos y reales en una máquina Dell OPTIPLEX 7010 con procesador Intel®Core™i7-3770 CPU de 3.40GHz x 8, 16 GB de RAM y 1TB 7200 RPM de Disco Duro, corriendo Debian con linux 3.2. Para todos los casos se usaron los algoritmos implementados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

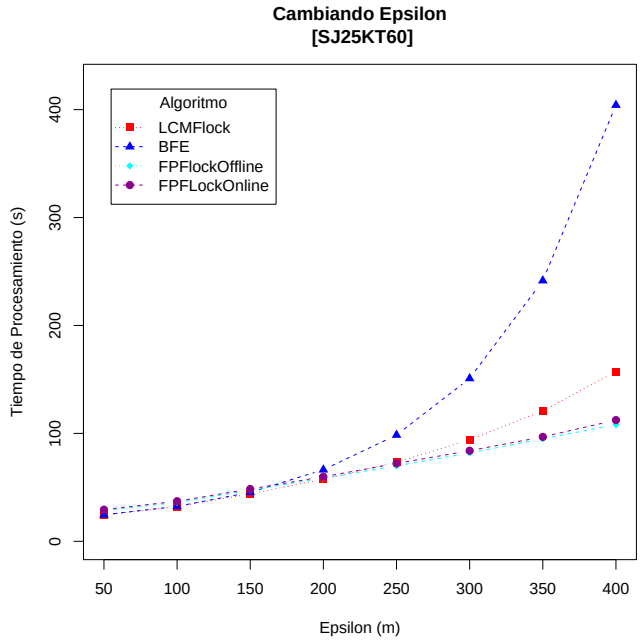
instantes de tiempo. El segundo recoge 50.000 trayectorias a partir de 2.014.346 de puntos durante 55 instantes de tiempo. La tabla III resume la información principal. Las figuras 3 y 4 muestran los tiempos de desempeño para estos dos casos de estudio, los parámetros adicionales fueron μ=5, δ=3 y μ=9,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

B. TAPAS Cologne

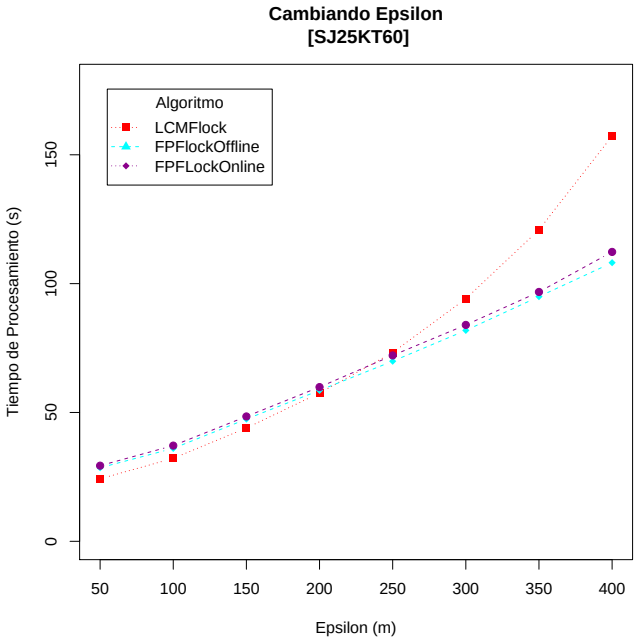
Este conjunto de datos sintético se preparó utilizando el escenario TAPAS Cologne [20] en SUMO [21], un reconocido simulador de tráfico para la movilidad urbana. El escenario de simulación TAPAS Cologne describe el tráfico dentro de la ciudad de Colonia (Alemania) durante un día entero. La principal 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

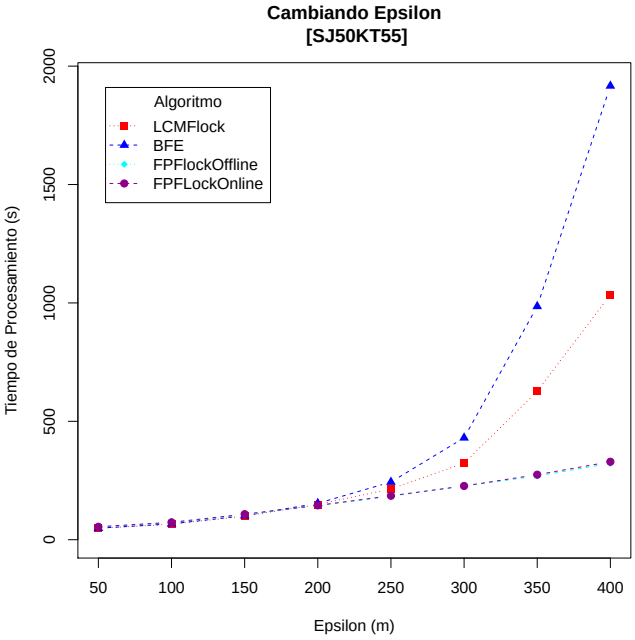


(a) BFE, LCMFlock, FPFlock

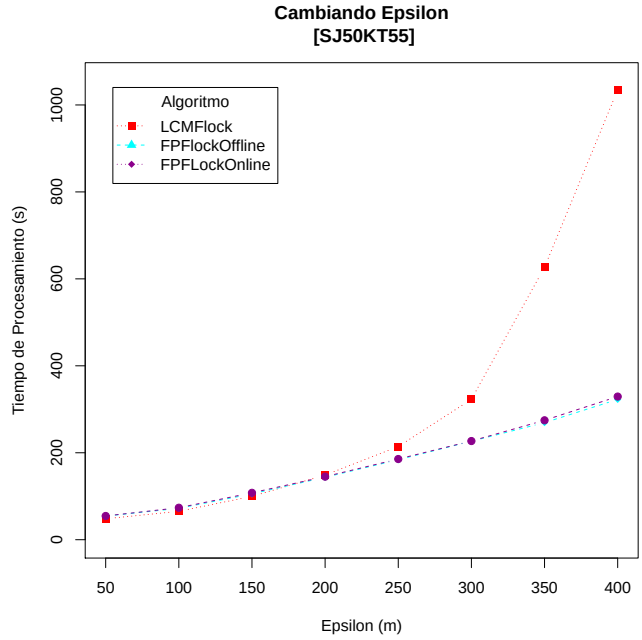


(b) LCMFlock, FPFlock

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



(a) BFE, LCMFlock, FPFlock

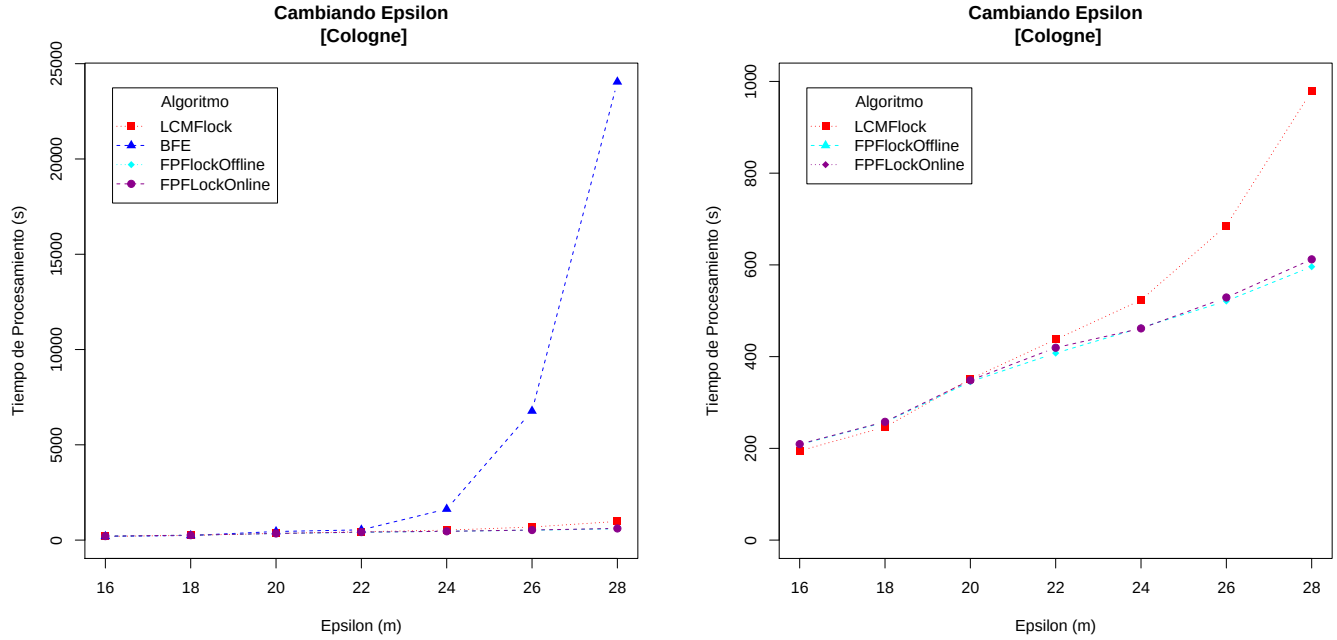


(b) LCMFlock, FPFlock

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

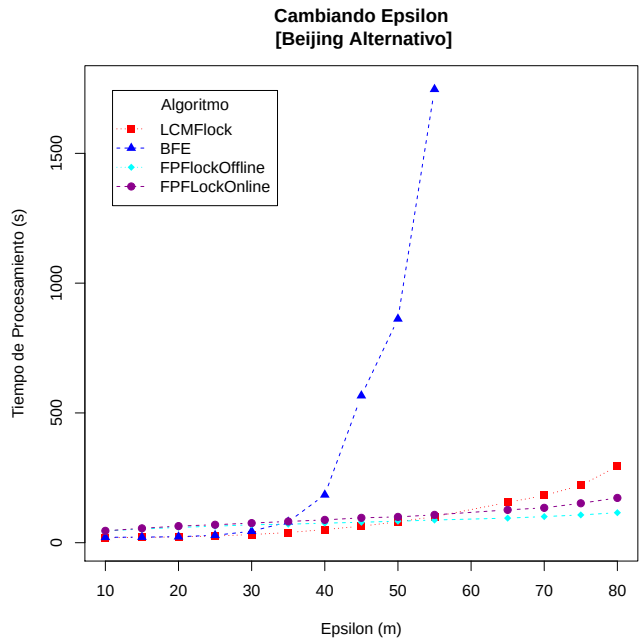
de muestra. Cada trayectoria fue interpolada por minuto (un punto por minuto) y saltos de 20 minutos o más sin señal se utilizaron para marcar una nueva trayectoria. Por último, el conjunto de datos recoge más de 1,4 millones de puntos y 21.573 trayectorias. Sin embargo, como este conjunto de datos tuvo 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

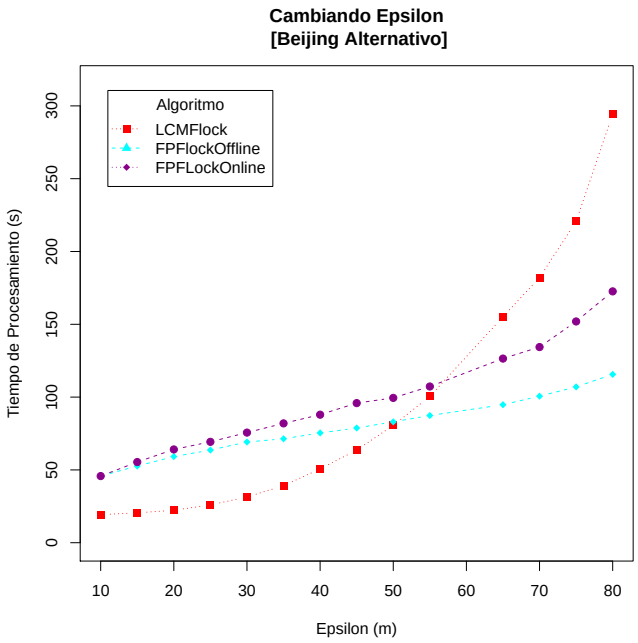
Tanto para BFE, LCMFLOCK, FPFlockOnline y FPFlockOffline el reporte de flocks se hace en una base de datos, con un identificador, tiempo de inicio, tiempo de fin y todos los puntos contenidos en dicho flock. En la tabla IV se muestra el número total de flocks reportados con un ε en particular.

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.

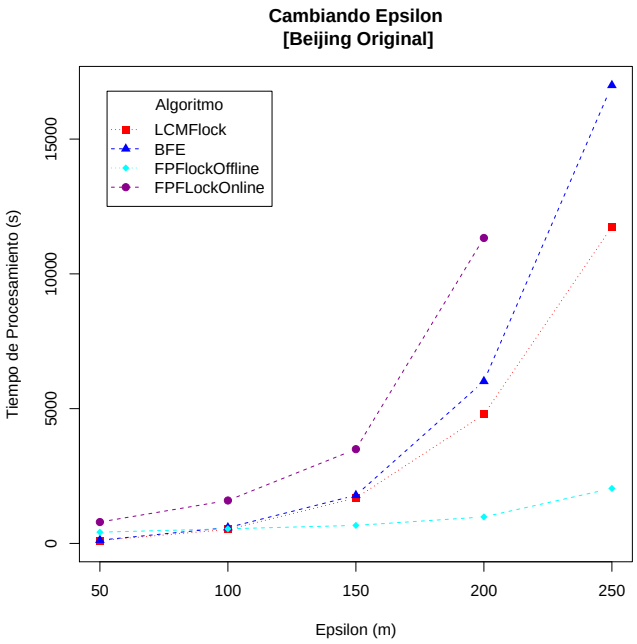


Figura 6. Caso de Prueba: Beijing Original

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.