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

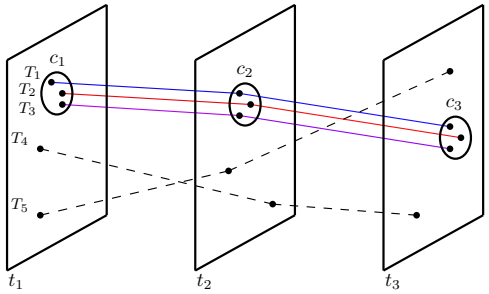

Figura 1. Ejemplo de patrones de agrupamiento.

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

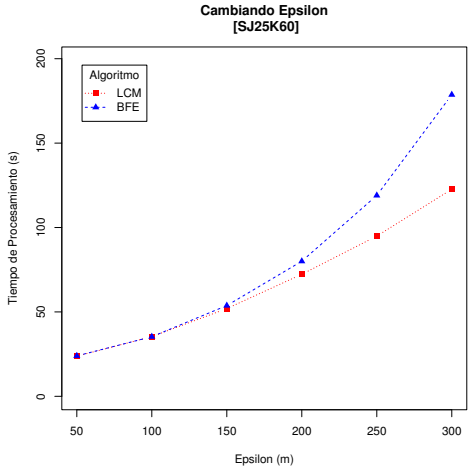

Figura 2. Caso de Prueba: SJ25K60

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

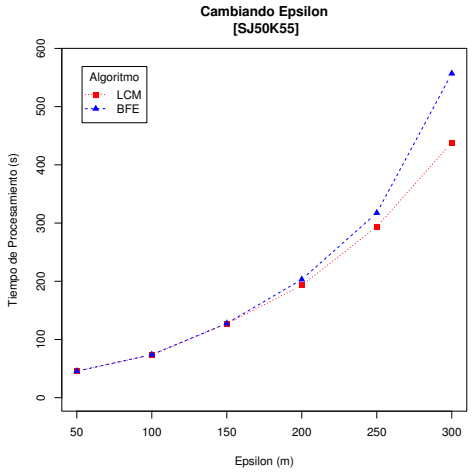

Figura 3. Caso de Prueba: SJ50K55

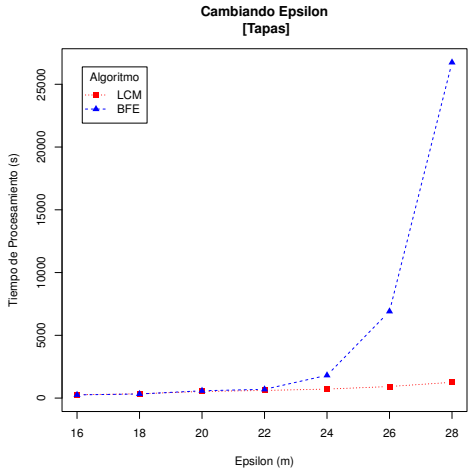

Figura 4. Caso de Prueba: Tapas Cologne

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.

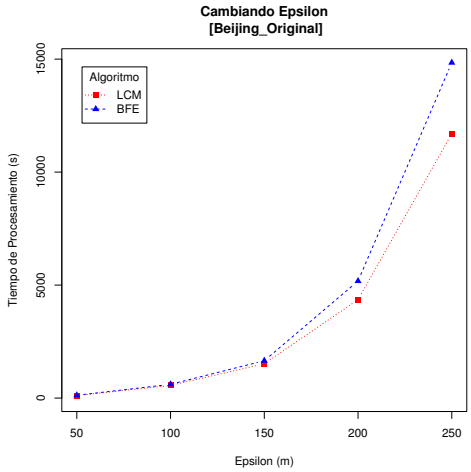

Figura 5. Caso de Prueba: Beijing Original

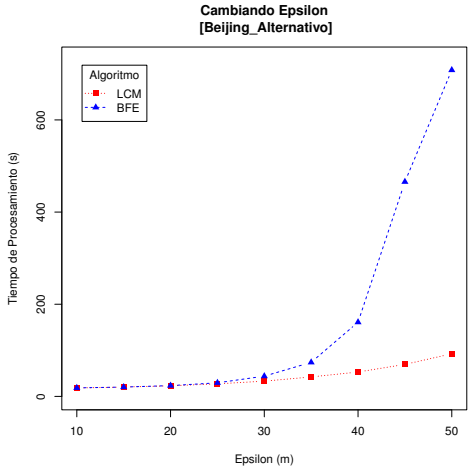

Figura 6. Caso de Prueba: Beijing Alternativo

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.