Towards Parallel Detection of Moving Flock Patterns in Large Spatio-temporal Datasets

Andres Calderon

May 15, 2017

1 Introduction

Nowadays, spatio-temporal data is ubiquitous. Thanks to new technologies and the proliferation of location devices (such as Internet of Things, Remote Sensing, Smart phones, GPS, RFID, etc.), the collection of huge amount of spatio-temporal data is now possible. With the appearance of this datasets also appears the need of new techniques which allow the analysis and detection of useful patterns in large spatio-temporal databases.

Applications for this kind of information are diverse and interesting, in particular if they come in the way of trajectory datasets [15, 11]. Case of studies range from transportation system management [5, 17] to Ecology [18, 20]. For instance, [29] explore the finding of complex motion patterns to discover similarities between tropical cyclone paths. Similarly, [1] use eye trajectories to understand which strategies people use during a visual search. Also, [10] track the behavior of tiger sharks in the coasts of Hawaii in order to understand their migration patters.

Recently, there has been an increasing interest in exploiting more complex movement patterns in spatio-temporal datasets. Traditional range and nearest neighbor queries do not capture the collective behavior of moving objects. Moving cluster [19], convoys [16] and flock patterns [3, 9] are new movement patterns which unveil how entities move together during a minimum time interval.

In particular, a moving flock pattern show how objects move close enough during a given period of time. A better understanding on how entities move in space is of special interest in areas such as sports [14], surveillance and security [24, 27], urban development [12, 23] and socio-economic geography [7].

Despite the fact that much more data become available, state-of-the-art techniques to mine complex movement patterns still depict low scalability and poor performance in big spatial data. The present work aims to find an initial solution to implement a parallel method to discover moving flock patterns in large spatio-temporal datasets. It

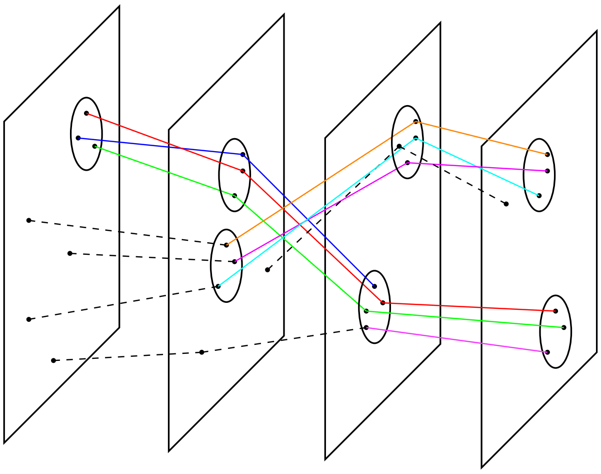

Figure 1: Moving flock pattern example.

is thought that new trends in distributed in-memory framework for spatial operations could help to speed up the detection of this kind of patterns.

The following section will state the related work in the area. Section 3 will explain the details of the implementation of the proposed solution while section 4 will present their experimental results. Finally, section 5 will discuss some conclusions and future work.

2 Related work

Recently increase use of location-aware devices (such as GPS, Smart phones and RFID tags) has allowed the collection of a vast amount of data with a spatial and temporal component linked to them. Different studies have focused in analyzing and mining this kind of collections [22, 25]. In this area, trajectory datasets have emerged as an interesting field where diverse kind of patterns can be identified [37, 31]. For instance, authors have proposed techniques to discover motion spatial patterns such as moving clusters [19], convoys [16] and flocks [3, 9]. In particular, [30] proposed BFE (Basic Flock Evaluation), a novel algorithm to find moving flock patterns in polynomial time over large spatio-temporal datasets.

A flock pattern is defined as a group of entities which move together for a defined lapse of time [3] (figure 1). Applications to this kind of patterns are rich and diverse. For example, [4] finds moving flock patterns in iceberg trajectories to understand their movement behavior and how they related to changes in ocean's currents.

The BFE algorithm presents an initial strategy in order to detect flock patterns. In that, first it finds disks with a predefined diameter (ε) where moving entities could be close enough at a given time interval. This is a costly operation due to the large number of points and intervals to be analyzed ($\mathcal{O}(2n^2)$ per time interval). The technique uses a grid-based index and a stencil (see figure 2) to speed up the process, but the complexity is still high.

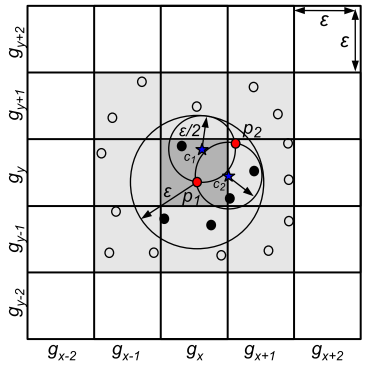

Figure 2: Grid-based index used in [30].

[4] and [29] use a frequent pattern mining approach to improve performance during the combination of disks between time intervals. Similarly, [28] introduce the use of plane sweeping along with binary signatures and inverted indexes to speedup the same process. However, the above-mentioned methods still keep the same strategy as BFE to find the disks at each interval.

[2] and [8] use depth-first algorithms to analyze the time intervals of each trajectory to report maximal duration flocks. However, these techniques are not suitable to find patterns in an on-line fashion.

Given the high complexity of the task, it should not be surprising the use of parallelism to increase performance. [**?**] use extremal and intersection sets to report maximal, longest and largest flocks on the GPU with the limitations of its memory model.

Indeed, despite the popularity of cluster computing frameworks (in particular whose supporting spatial capabilities [6, 33, 13, 32]) there are not significant advances in this area. At the best of our knowledge, this work is the first to explore in-memory distributed systems towards the detection of moving flock patterns.

3 Parallelizing flock detection

Given that the finding of disks at each time interval is one of the most costly operations towards the detection of moving flock patterns, the main goal of this work is to implement a parallel method to detect that set of disks. In order to do that, we will use the spatial operations offered by Simba [32], a distributed in-memory spatial analytics engine based on Apache Spark. This section explains the details of the algorithm implemented in Simba and how some spatial predicates introduced by it can leverage the finding of disks.

3.1 Spatial operations on Simba

Simba (Spatial In-Memory Big data Analytics) extends the Spark SQL engine to provide rich spatial operations through both SQL and the DataFrame API. Besides, it

```
SELECT
        *
FROM
        points p1
DISTANCE JOIN
        points p2
ON
        POINT(p2.x, p2.y) IN CIRCLERANGE(POINT(p1.x, p1.y), ε)
WHERE
        p1.id < p2.id
```

Figure 3: SQL statement on Simba to find points lying inside an ε distance.

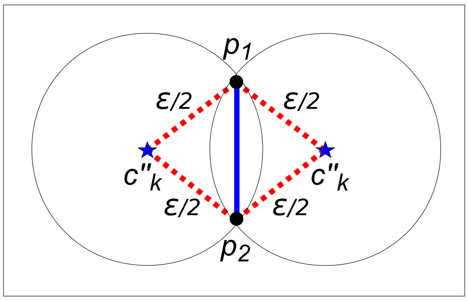

Figure 4: Disks for $\{p_1, p_2\},\ d(p_1, p_2) \leq \varepsilon$ [30].

introduces two-layer spatial indexing and cost-based optimizations to support efficient spatial queries in parallel. Simba is open source and public available at the project's webpage[1].

In particular, `DISTANCE JOIN` and `CIRCLERANGE` operators were used in order to find groups of points lying close enough each other. The algorithm uses a user-defined distance (ε) to define the diameter of the disk. Figure 3 shows the SQL statement used to find pairs of points inside an ε distance in parallel.

3.2 Finding disks for pairs of points

Once the set of pairs of points has been found, a map function computes the center of the two possible disks per each pair according to the BFE algorithm. [30] states that: "For each such pair there are exactly two disks with radius $\frac{\varepsilon}{2}$ that have those points on their circumference". Figure 4 illustrates how to find the center of those disks.

After that, the complete set of candidate disks is collected and ready to be processed by the following phases of the BFE algorithm.

The implementation of the proposed method was written in Scala 2.10.6 and tested

[1] http://www.cs.utah.edu/~dongx/simba/

in Simba/Spark 1.6.0. The current code can be accessed at the authors' repository[2].

4 Experiments

The main idea of the experiments is to assess the performance of the parallel method against a sequential version of the BFE algorithm proposed by [30]. A public available implementation written in Python can be accessed at this repository[3]. The source code was modified to stop after the finding of the disks and report the total number of computed disks. Same configuration was followed by the parallel implementation.

Next, a set of experiments evaluates the execution time of both algorithms on two real datasets. Further details of the settings and datasets are discussed below.

4.1 Beijing dataset

This dataset was extracted from the Geolife project[4] [34, 36, 35]. It collects GPS trajectories of 182 users in a period of over three years (from April 2007 to August 2012) for an overall total of 17,621 trajectories. The timestamp field was ignored and duplicate locations were removed to simulate an unique and large time interval. In total, the point dataset contains ≈18 million points.

An initial set of experiments takes relatively small samples of the data and runs the algorithms under different values of ε. Experiments were deployed in a single-node machine with a 4-core Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz processor, 8 GB of RAM running Ubuntu 16.04 LTS, Python 3.5 and Simba/Spark 1.6.0. Figure 5 show the results of these experiments.

4.2 Porto dataset

This dataset was extracted from the ECML/PKDD'15 Taxi Trajectory Prediction Challenge[5] [21, 26]. It collects a complete year (from 01/07/2013 to 30/06/2014) of trajectories for all the 442 taxis running in the city of Porto, in Portugal. After pre-processing and duplicate removal the collection had ≈17.7 million points.

This set of experiments takes data samples of 1, 2, 4, 8 and 16 million of points. Similarly, it runs the algorithms under different values of ε. This time, experiments were deployed in a 4-node academic cluster with the following setup: an 8-core Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz processor and 15.5 GB of RAM per node. The systems run Centos 6.8, Python 3.5 and Simba/Spark 1.6.0. Figure 6 show the results of these experiments.

[2] https://github.com/aocalderon/PhD/tree/master/Y2Q1/SDB/Project/Code/Scripts/pbfe2

[3] https://github.com/poldrosky/FPFlock

[4] https://www.microsoft.com/en-us/download/details.aspx?id=52367

[5] https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

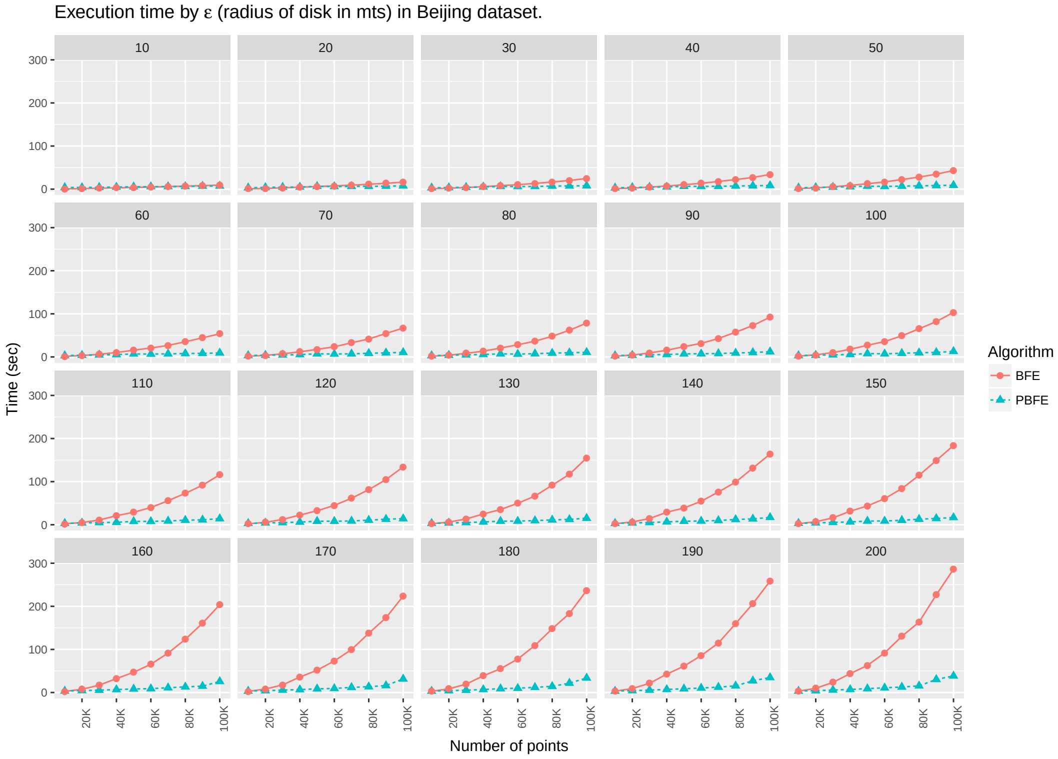

Figure 5: Execution time for Beijing dataset.

5 Conclusions and future work

An implementation of a parallel method to detect disks for the BFE algorithm has been presented. The proposed method proves to be scalable and reliable. Experiments shows that execution time improves up to 3 orders of magnitude compared to the sequential implementation of the BFE algorithm for the same step.

Parallel implementation for the remaining steps of the BFE algorithm are part of the future work. It is expected to work on parallel strategies to prune redundant and incomplete disks together with a parallel strategy to join the sets of valid disks between time intervals. In addition, data pre-processing and result visualization are still open issues.

References

[1] T. A. Amor, S. D. S. Reis, D. Campos, H. J. Herrmann, and J. S. Andrade. Persistence in eye movement during visual search. *Scientific Reports*, 6:20815, Feb. 2016. 00000.

[2] H. Arimura, T. Takagi, X. Geng, and T. Uno. Finding All Maximal Duration Flock Patterns in High-dimensional Trajectories. 00000, 2014.

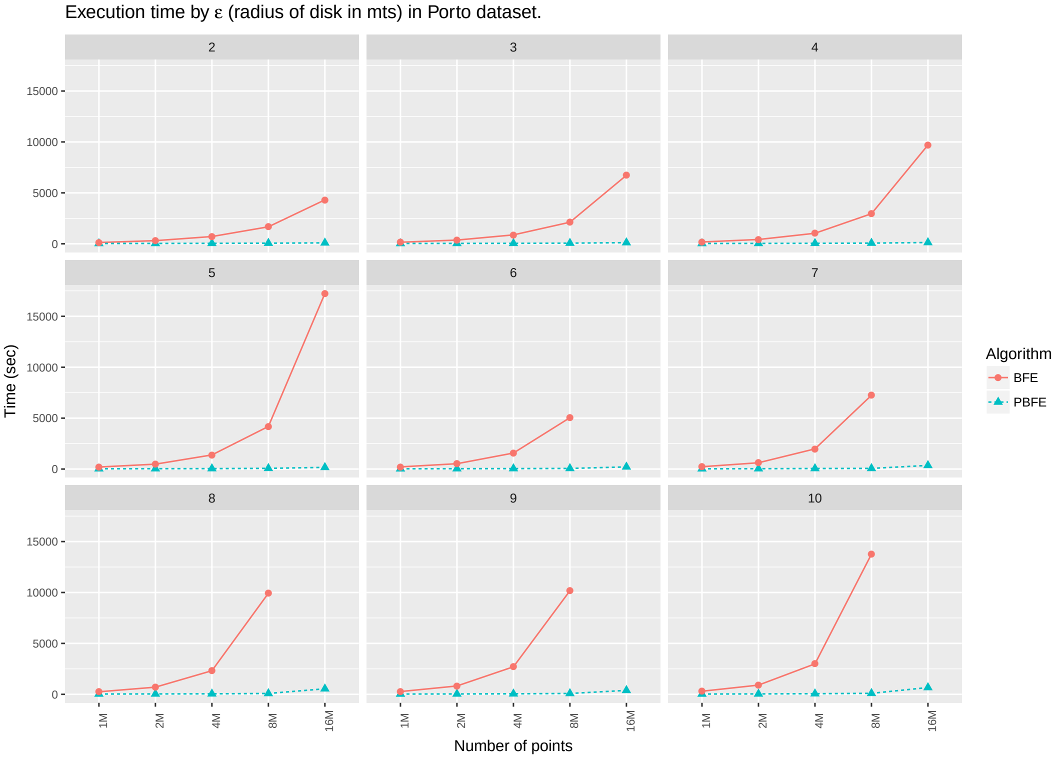

Figure 6: Execution time for Porto dataset.

[3] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, Nov. 2008. 00000.

[4] A. Calderon. Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach. Master's thesis, University of Twente, 2011. 00000.

[5] G. Di Lorenzo, M. Sbodio, F. Calabrese, M. Berlingerio, F. Pinelli, and R. Nair. AllAboard: Visual Exploration of Cellphone Mobility Data to Optimise Public Transport. *IEEE Transactions on Visualization and Computer Graphics*, 22(2):1036–1050, Feb. 2016. 00004.

[6] A. Eldawy. SpatialHadoop: Towards flexible and scalable spatial processing using mapreduce. pages 46–50. ACM Press, 2014. 00020.

[7] A. Frank, J. Raper, and J. P. Cheylan, editors. *Life and Motion of Socio-Economic Units*. CRC Press, London ; New York, 1 edition edition, Dec. 2000. 00075.

[8] X. Geng, T. Takagi, H. Arimura, and T. Uno. Enumeration of complete set of flock patterns in trajectories. pages 53–61. ACM Press, 2014.

[9] 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, GIS '06, pages 35–42, New York, NY, USA, 2006. ACM. 00222.

[10] K. N. Holland, B. M. Wetherbee, C. G. Lowe, and C. G. Meyer. Movements of tiger sharks (Galeocerdo cuvier) in coastal Hawaiian waters. *Marine Biology*, 134(4):665–673, 1999. 00000.

[11] P. Huang and B. Yuan. Mining Massive-Scale Spatiotemporal Trajectories in Parallel: A Survey. In *Trends and Applications in Knowledge Discovery and Data Mining*, volume 9441 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2015. 00001.

[12] X. Huang, Y. Zhao, C. Ma, J. Yang, X. Ye, and C. Zhang. TrajGraph: A Graph-Based Visual Analytics Approach to Studying Urban Network Centralities Using Taxi Trajectory Data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):160–169, Jan. 2016. 00000.

[13] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest. GeoMesa: A distributed architecture for spatio-temporal fusion. page 94730F, May 2015. 00000.

[14] S. Iwase and H. Saito. Tracking Soccer Player Using Multiple Views. In *MVA*, pages 102–105, 2002. 00000.

[15] H. Jeung, M. L. Yiu, and C. S. Jensen. Trajectory pattern mining. In *Computing with Spatial Trajectories*, pages 143–177. Springer, 2011. 00045.

[16] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of Convoys in Trajectory Databases. *Proc. VLDB Endow.*, 1(1):1068–1080, Aug. 2008. 00332.

[17] K. Johansson and H. Terelius. An efficiency measure for road transportation networks with application to two case studies. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 5149–5155. IEEE, 2015. 00000.

[18] A. Johnston, D. Fink, M. D. Reynolds, W. M. Hochachka, B. L. Sullivan, N. E. Bruns, E. Hallstein, M. S. Merrifield, S. Matsumoto, and S. Kelling. Abundance models improve spatial and temporal prioritization of conservation resources. *Ecological Applications*, 25(7):1749–1756, 2015. 00010.

[19] P. Kalnis, N. Mamoulis, and S. Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. In C. B. Medeiros, M. J. Egenhofer, and E. Bertino, editors, *Advances in Spatial and Temporal Databases*, Lecture Notes in Computer Science, pages 364–381. Springer Berlin Heidelberg, Aug. 2005. 00000.

[20] F. A. La Sorte, D. Fink, W. M. Hochachka, and S. Kelling. Convergence of broad-scale migration strategies in terrestrial birds. *Proceedings of the Royal Society B: Biological Sciences*, 283(1823):20152588, Jan. 2016. 00000.

[21] H. T. Lam, E. Diaz-Aviles, A. Pascale, Y. Gkoufas, and B. Chen. (Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories. *ECML/PKDD Discovery Challenge 2015*, 2015. 00001.

[22] Y. Leung. *Knowledge Discovery in Spatial Data.* Springer Science & Business Media, Mar. 2010. 00037.

[23] Y. Long and J.-C. Thill. Combining smart card data and household travel survey to analyze jobs–housing relationships in Beijing. *Computers, Environment and Urban Systems*, 53:19–35, Sept. 2015. 00000.

[24] D. Makris and T. Ellis. Path detection in video surveillance. *Image and Vision Computing*, 20(12):895–903, Oct. 2002. 00169.

[25] H. J. Miller and J. Han. *Geographic Data Mining and Knowledge Discovery.* Taylor & Francis, Inc., Bristol, PA, USA, 2001. 00000.

[26] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Predicting Taxi-Passenger Demand Using Streaming Data. *Trans. Intell. Transport. Sys.*, 14(3):1393–1402, Sept. 2013. 00048.

[27] C. Piciarelli, G. L. Foresti, and L. Snidaro. Trajectory clustering and its applications for video surveillance. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 40–45, Sept. 2005. 00000.

[28] P. S. Tanaka, M. R. Vieira, and D. S. Kaster. An Improved Base Algorithm for Online Discovery of Flock Patterns in Trajectories. *Journal of Information and Data Management*, 7(1):52, 2016. 00000.

[29] U. Turdukulov, A. Calderon, 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*, 28(10):2013–2029, Oct. 2014. 00000.

[30] 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*, GIS '09, pages 286–295, New York, NY, USA, 2009. ACM. 00000.

[31] M. R. Vieira and V. Tsotras. *Spatio-Temporal Databases: Complex Motion Pattern Queries.* Springer Science & Business Media, Oct. 2013. 00000.

[32] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo. Simba: Efficient In-Memory Spatial Analytics. pages 1071–1085. ACM Press, 2016. 00000.

[33] J. Yu, J. Wu, and M. Sarwat. A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1410–1413, May 2016. 00000.

[34] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding mobility based on GPS data. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 312–321. ACM, 2008. 00462.

[35] Y. Zheng, X. Xie, and W.-Y. Ma. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010. 00388.

[36] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, pages 791–800. ACM, 2009. 00959.

[37] Y. Zheng and X. Zhou. *Computing with Spatial Trajectories.* Springer Science & Business Media, Oct. 2011. 00000.