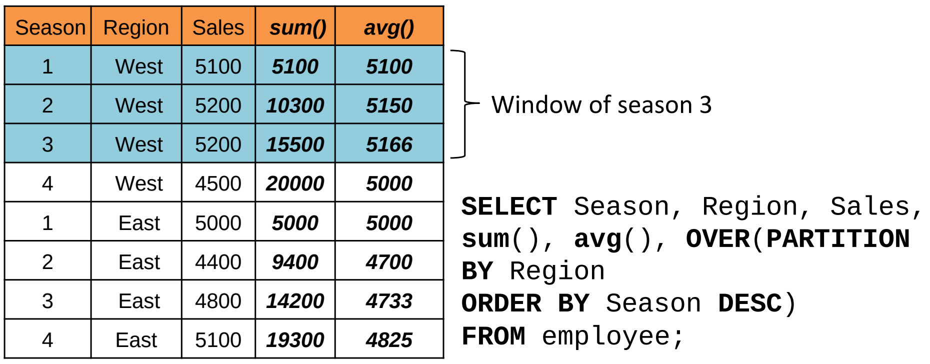
Chapter 1

Introduction

With the maturity of database technologies, nowadays applications collect data in all domains at an unprecedented scale. For example, billions of social network users and their activities are collected in the form of *graphs*; Thousand sensor reports are collected per second in the form of *time series*; Hundreds of millions of temporal-locations are collected as *trajectories*, to name just a few. Flooded by the tremendous amount of data, it is emerging to provide useful and efficient analytics for various data domains. Traditional SQL analytics which comprises of operations (such as, partition, sorting and aggregation) in the relational domain become limited in non-structured domains. In SQL context, interesting analytics such as graph traversal and pattern detection often involve complex joins which are very hard to optimize without domain knowledge. In this thesis, we explore the neighborhood data analytic, which in SQL is expressed by the window function, on different domains and demonstrate how to efficiently deploy neighborhood analytic to gain useful insights.

1.1 Neighborhood Analytic

By its self-describing name, neighborhood analytic aims to provide summaries of each object over its vicinity. In contrast to the global analytics which aggregates the entire collection of data as a whole, neighborhood analytic provides a personalized view on each object per se. Neighborhood data analytics originates from the window function defined in SQL which is illustrated in Figure 1.1.



Season	Region	Sales	***sum()***	***avg()***
1	West	5100	***5100***	***5100***
2	West	5200	***10300***	***5150***
3	West	5200	***15500***	***5166***
4	West	4500	***20000***	***5000***
1	East	5000	***5000***	***5000***
2	East	4400	***9400***	***4700***
3	East	4800	***14200***	***4733***
4	East	5100	***19300***	***4825***

Window of season 3

```
SELECT Season, Region, Sales,
sum(), avg(), OVER(PARTITION
BY Region
ORDER BY Season DESC)
FROM employee;
```

Figure 1.1: A SQL window function computing running sum and average of sales. The window of season-3 is highlighted.

As shown in the figure, the sales report contains five attributes: “Season”, “Region” and “Sales” are the facts, “`sum`()” and “`avg`()” are the analytics representing the running sum and average. A window function is represented by the `over` keyword. In this context, the window of a tuple o_i contains other tuples o_j such that o_i and o_j are in the same “region” and o_j’s “season” is prior to o_i’s. The window of season-3 for region-“West” is highlighted. Apart from this example, there are many other usages of the window function in the relational context. Being aware of the success of the window function, SQL 11 standard incorporates “`LEAD`” and “`LAG`” keywords which offer fine-grained specifications on a tuple’s window.

Despite the usefulness, there are very few works reporting the window analytics in the non-relational domain. This may dues to the usage of *sorting* in relational windows. For example, in Figure 1.1, objects need to be sorted according to “Season”, and then the window of each objects is implicitly formed. However, in non-relational context, sorting may be ambiguous and even undefined.

To generalize the window function to other domains, we propose the neighborhood analytics in a broader context. Given a set of objects (such as tuples in relational domain or vertexes in graph domain), the neighborhood analytic is a composite function ($\mathcal{F} \circ \mathcal{N}$) applied on every object. $\mathcal{N}$ is the *neighborhood function*, which contains the related objects of an object; $\mathcal{F}$ is an *analytic function*, which could be aggregate, rank, pattern matching, etc. Apparently, the relational window function is a special case of the neighborhood analytics. For example, the window function in Figure 1.1 can be represented as $\mathcal{N}(o_i) = \{o_j | o_i.season > o_j.season \wedge o_i.region = o_j.region\}$ and $\mathcal{F} = \texttt{avg}$. Since the *sorting* requirement is relaxed, our neighborhood analytics is able to enrich the semantic of relational window notations and can be applied on many other domains.

1.2 Scope of the Thesis

In this thesis, we explore the neighborhood analytics in three prevalent data domains, namely **attributed graph**, **sequence data** and **trajectory**. Since the neighborhood analytics could be very broad, this thesis only focus on the following two intuitive neighborhood functions:

Distance Neighborhood: the neighborhood is defined based on numeric distance, that is $\mathcal{N}(o_i, K) = \{o_j | \texttt{dist}(o_i, o_j) \leq K\}$, where `dist` is a distance function and K is a distance threshold.

Comparison Neighborhood: the neighborhood is defined based on the comparison of objects, that is $\mathcal{N}(o) = \{o_i | o.a_m \ \texttt{cmp} \ o_i.a_m\}$, where a_m is an attribute of object and `cmp` is a binary comparator.

There could be other types of neighborhood functions with different level of granularity. However, despite the simpleness of these two neighborhood functions, they are indeed versatile in representing many useful analytics.

1.3 Contributions

In brief, this thesis entitles a twofold contribution. First, by sewing different $\mathcal{N}$s and $\mathcal{F}$s, several novel neighborhood queries are proposed for *graph*, *sequence data* and *trajectory* domains respectively. Second, this thesis deals with the efficiency issues in deploying the corresponding analytic queries to handle data of nowadays scale. The roadmap of this thesis is shown in Figure 1.2.

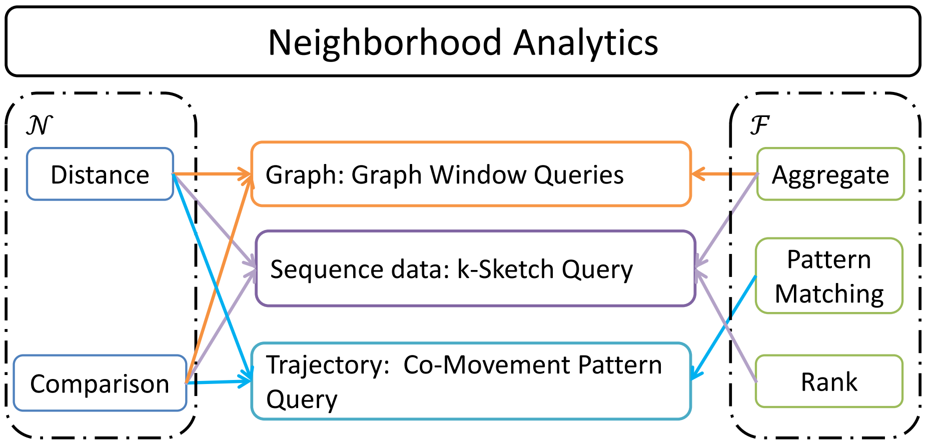


Figure 1.2: The road map of this thesis. There are three major contributions as highlighted in the center. Each contribution is a neighborhood analytic based on different $\mathcal{N}$ and $\mathcal{F}$ as indicated by arrows.

In a nutshell, we propose the neighborhood queries in three data domains. In *graph*, we identify two types of queries: *k-hop window query* is based on the *distance* neighborhood and *topological window query* is based on the *comparison* neighborhood. In *sequence data*, we propose a nested *distant* and *comparison* neighborhoods query named k-Sketch to finding striking rank-aware streaks. In *trajectory*, we analyze existing movement patterns based on *distance* and *comparison* neighborhoods and unify the existing works with a general pattern discovery query. Next, we present our contributions in detail for each of the data domains.

1.3.1 Window Queries for Graph Data

The first piece of the thesis deals with neighborhood query on graph data. Nowadays information network are typically modeled as attributed graphs where the vertexes correspond to objects and the edges capture the relationships between these objects. As vertexes embed a wealth of information (e.g., user profiles in social networks), there are emerging demands on analyzing these data to extract useful insights. We propose the concept of *window analytics* for attributed graph and identify two types of such analytics as shown in the following examples:

Example 1.3.1 (k-hop window)**.** In a social network (such as Linked-In and Facebook etc.), users are normally modeled as vertexes and connectivity relationships are modeled as edges. In the social network scenario, it is of great interest to summarize the most relevant connections to each user such as the neighbors within 2-hops. Some analytic queries such as summarizing the related connections' distribution among different companies, and computing age distribution of the related friends can be useful. In order to answer these queries, collecting data from every user's neighborhoods within 2-hop is necessary.

Example 1.3.2 (Topological window)**.** In biological networks (such as Argocyc, Ecocyc etc.), genes, enzymes and proteins are vertexes and their dependencies in a pathway are edges. Because these networks are directed and acyclic, in order to study the protein regulating process, one may be interested to find out the statistics of molecules in each protein production pathway. For each protein,we can traverse the graph to find every other molecules that are in the upstream of its pathway. Then we can group and count the number of genes and enzymes among those molecules.

The two *windows* shown in the examples are essentially neighborhood functions defined for each vertex. Specifically, let $G = (V : E : A)$ be an attributed graph, where V is the set of vertexes, E is the set of edges, and each vertex v is associated as a multidimensional points $a_v \in A$ called attributes. The *k-hop* window is a *distance* neighborhood function, i.e., $\mathcal{N}_1(v, k) = \{u | \texttt{dist}(v, u) \leq k\}$, which captures the vertexes that are k-hop nearby. The *topological* window, $\mathcal{N}_2(v) = \{u | u \in v.ancestor\}$, is a *comparison* neighborhood function that captures the ancestors of a vertex in a directly acyclic graph. The analytic function $\mathcal{F}$ is an aggregate function (sum, avg, etc.) on A.

Apart from demonstrating the useful use-cases on these two windows, we also investigate on supporting efficient window processing. We propose two different types of indexes: Dense Block Index (DBIndex) and Inheritance Index (I-Index). The DBIndex and I-Index are specially optimized to support k-hop window and topological window processing. These indexes integrate the aggregation process with partial work sharing techniques to achieve efficient computation. In addition, we develop space-and-performance efficient techniques for the index construction. Notably, DBIndex saves upto 80% of indexing time as compared to the state-of-the-art competitor and upto 10x speedup in query processing.

1.3.2 k-Sketch Query on Time Sequenced Events

The second piece of the thesis explores the neighborhood query for time sequenced events. In time sequenced events analysis, an important and revenue-generating task is to detect sensational patterns. An outstanding neighborhood based pattern is the *streak* [50]. Streak has been found useful in many sequenced applications, such as network monitoring, stock analysis and sport reporting. For example, streaks may be of the following forms:

1. [STOCK]:“Apple Inc. has an average price of USD 115.5 in the last week”

2. [SPORTS]: “Kobe has scored at least 60 points in three straight games’

Essentially, a *streak* is constructed from two concepts: an *aggregate function* (e.g., average, min) applied on a *time window* of events (e.g., seven days, three games). This makes the streak as a neighborhood query of events where the neighborhood function is defined as time windows.

However, it is challenging to directly adopt streaks in discovering sensational patterns because the strikingness of a streak is unknown. For example, *Streak 1* would not be striking if all NBA players score over 60 points. On the contrary, knowing that NBA players in average only score 20 points, then *Streak 1* is quite striking. To quantify the strikingness of a streak, we propose a *rank-aware streak* that measures the strikingness of a streak by comparing among all streaks under same condition (i.e., window length).

Technically, the rank-aware streak can be formally described using a joint neighborhood function. Let $e_s(t)$ denote the event of subject s at time t. Then the rank-aware streaks are generated using neighborhood analytics in a two-step manner:

(1) a *distance neighborhood* $\mathcal{N}_1(o_i, w) = \{o_j | o_i.t - o_j.t \leq w\}$ groups a consecutive w events for each event. Let $\overline{v}$ be the aggregate value associated with $\mathcal{N}_1$, then the output of this step is a set of *event windows* of the form $n = \langle o_i, w, t, \overline{v} \rangle$.

(2) a *comparison neighborhood* $\mathcal{N}_2(n_i) = \{n_j | n_j.w = n_i.w \wedge n_i.\overline{v} \geq n_j.\overline{v}\}$ ranks a subject's event window among all other event windows with the same window size. The result of the this step is a tuple $\langle o_i, w, t, r \rangle$, where r is the *rank*.

As the rank-aware streak contains the position of the streak among its cohort, we are able to quantitative measure the strikingness based on the rank value. For example, the rank-aware version of *Streak 1* would be "Kobe has scored at least 60 points in three straight games, which is best in the league". This clearly suggests that *Streak 1* is striking.

Another challenge in the pattern discovery is that the amount of streaks for a subject is huge. For a subject with N events, there are $O\binom{N}{2}$ streaks as well as rank-aware streaks. In real life, "Kobe" has played 1000 games which introduces near half million streaks. Such a large number poses difficulties for selecting phenomenal streaks. We observe that many streaks share almost identical information. For example, Kobe has scored 81 points in one game. This single-event streak ranks number 1 among all streaks. Subsequently, Kobe has scored an average of 47 points for 2 straight games, which is only ranked number 10. Therefore, reporting the second streak does not provide additional information for the total sketch query. Based on this observation, we propose a k-Sketch query to effectively select the k most *representative streaks*.

In this thesis, we further study the technical issues in running k-Sketch queries in both online and offline scenarios. Lastly, we conduct experiments on four real datasets, and the results demonstrate the efficiency and effectiveness of our proposed algorithms: the running time achieves up to 500x speedup as compared to baseline and the quality of the detected sketch is endorsed by the anonymous users from Amazon Mechanical Turk [1].

1.3.3 Co-Movement Pattern Query in Trajectory Data

The third piece of the thesis studies the neighborhood query on the trajectory domain. In the trajectory domain, an important mining task is to discover traveling patterns among moving objects. A *co-movement pattern* [31, 53] is a popularly studied movement pattern which refers to a group of moving objects traveling together for a certain period of time. A pattern is prominent if the group size exceeds M and the length of duration exceeds K. The group is defined based on the spatial proximity of objects, which can be seen as the spatial neighbors of the objects. Rooted from the basic movement definition and driven by different mining applications, there are several instances of co-movement patterns that have been developed with more advanced constraints.

[1] `https://requester.mturk.com`

Table 1.1 summarizes several popular co-movement patterns with different constraints with respect to clustering in spatial proximity, consecutiveness in temporal duration and computational complexity. In particular, the flock [21] and the group [42] patterns require all the objects in a group to be enclosed by a disk with radius r; whereas the convoy [23], the swarm [33] and the platoon [32] patterns resort to density-based spatial clustering. In the temporal dimension, the flock [21] and the convoy [23] require all the timestamps of each detected spatial group to be consecutive, which is referred to as global consecutiveness; whereas the swarm [33] does not impose any restrictions. The group [42] and the platoon [32] adopt a compromised approach by allowing arbitrary gaps between consecutive segments, which is called local consecutiveness. They introduce a parameter L to control the minimum length of each local consecutive segment.

Pattern	**Proximity**	**Consecutiveness**
flock [21]	disk based	global
convoy [23]	density based	global
swarm [33]	density based	-
group [42]	disk based	local
platoon [32]	density based	local

Table 1.1: Constraints of co-movement patterns.

As shown, there are various types of co-movement patterns facilitating different application needs and it is cumbersome to deploy and optimize each of the pattern discovery algorithms. Therefore, it calls for a general framework to provide versatile and efficient support of all these pattern discoveries. We observe that these co-movement patterns can be uniformly represented as a two-step neighborhood query as follows: (1) $\mathcal{N}_1$ is a *distance neighborhood* used to determine the spatial proximity of objects. For example, flock and group patterns uses the *disk-based* clustering, which is equivalent to $\mathcal{N}_1(o_i) = \{o_j | \texttt{dist}(o_i, o_j) < r\}$ for each object. Convoy, swarm and platoon patterns uses the *density-based* clustering, which is equivalent to $\mathcal{N}_1(o_i) = \{o_j | \texttt{dist}(o_j, o_k) \leq \epsilon \wedge o_k \in \mathcal{N}_1(o_i)\}$. (2) $\mathcal{N}_2$ is a comparison neighborhood used to determine the temporal constraints, i.e., $\mathcal{N}_2(o_i) = \{o_j, T | \forall t \in T, C_t(o_i) \equiv C_t(o_j)\}$, where $C_t(\cdot)$ returns the neighborhoods (i.e., N_1) of an object at time t.

Enlightened by the neighborhood unification, we propose a *General Co-Movement Pattern* (GCMP) query to capture all existing co-movement patterns. In GCMP, we treat the proximity detection (i.e., $\mathcal{N}_1$) as a black box and only focus on the pattern detection (i.e., $\mathcal{N}_2$). It is notable that the GCMP query is able to detect any of the existing co-movement patterns by adopting different analytic functions.

On the technical side, we study how to efficiently processing GCMP in a MapReduce platform to gain scalability for large-scale trajectory databases. In particular, we propose two parallel frameworks: (1) TRPM, which partitions trajectories by replicating snapshots in the temporal domain. Within each partitions, a line-sweep method is developed to find all patterns. (2) SPARE, which partitions trajectories based on object's neighborhood. Within each partitions, a variant of Apriori enumerator is applied to generate all patterns. We then show the efficiency of both our methods in the Apache Spark platform with three real trajectory datasets upto 170 million points. The results show that SPARE achieves upto 14 times efficiency as compared to TRPM, and 112 times speedup as compared to the state-of-the-art centralized schemes.

1.4 Thesis Organization

The remaining part of the thesis are organized as follows: in Chapter 2, we summarize related literature on our proposed neighborhood based queries in different data domains. In Chapter 3, we present the window function on graph data. In Chapter 4, we present the k-sketch query on sequence data. In Chapter 5, we present a pattern mining framework on trajectory data. Chapter 6 summarizes this thesis and highlights future directions.

Bibliography

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[2] Htoo Htet Aung and Kian-Lee Tan. Discovery of evolving convoys. In *International Conference on Scientific and Statistical Database Management*, pages 196–213. Springer, 2010.

[3] Baruch Awerbuch, Yossi Azar, Amos Fiat, and Tom Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 519–530. ACM, 1996.

[4] Jie Bao, Yu Zheng, David Wilkie, and Mohamed F Mokbel. A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology*, 2013.

[5] Srikanth Bellamkonda, Hua-Gang Li, Unmesh Jagtap, Yali Zhu, Vince Liang, and Thierry Cruanes. Adaptive and big data scale parallel execution in oracle. *Proceedings of the VLDB Endowment*, 6(11):1102–1113, 2013.

[6] Michael A Bender, Martín Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.

[7] Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.

[8] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.

[9] Ronald S Burt. *Structural holes: The social structure of competition.* Harvard university press, 2009.

[10] Yu Cao, Chee-Yong Chan, Jie Li, and Kian-Lee Tan. Optimization of analytic window functions. *Proceedings of the VLDB Endowment*, 5(11):1244–1255, 2012.

[11] Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han, and S Yu Philip. Graph olap: Towards online analytical processing on graphs. In *2008 Eighth IEEE International Conference on Data Mining*, pages 103–112. IEEE, 2008.

[12] Lisi Chen and Gao Cong. Diversity-aware top-k publish/subscribe for text stream. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 347–362. ACM, 2015.

[13] James Cheng, Zechao Shang, Hong Cheng, Haixun Wang, and Jeffrey Xu Yu. K-reach: who is in your small world. *Proceedings of the VLDB Endowment*, 5(11):1292–1303, 2012.

[14] Tom Choe, Alexander Skabardonis, and Pravin Varaiya. Freeway performance measurement system: operational analysis tool. *Transportation Research Record: Journal of the Transportation Research Board*, (1811):67–75, 2002.

[15] Emilio Coppa and Irene Finocchi. On data skewness, stragglers, and mapreduce progress indicators. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 139–152. ACM, 2015.

[16] Artur Czumaj, Mirosław Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1):37–46, 2007.

[17] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[18] Marina Drosou and Evaggelia Pitoura. Diverse set selection over dynamic data. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1102–1116, 2014.

[19] Katherine Edwards, Simon Griffiths, and William Sean Kennedy. Partial interval set cover–trade-offs between scalability and optimality. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 110–125. Springer, 2013.

[20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[21] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 35–42. ACM, 2006.

[22] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. Data in, fact out: automated monitoring of facts by factwatcher. *Proceedings of the VLDB Endowment*, 7(13):1557–1560, 2014.

[23] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[24] Ryota Jinno, Kazuhiro Seki, and Kuniaki Uehara. Parallel distributed trajectory pattern mining using mapreduce. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 269–273. IEEE, 2012.

[25] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *International Symposium on Spatial and Temporal Databases*, pages 364–381. Springer, 2005.

[26] Ingrid M Keseler, Julio Collado-Vides, Socorro Gama-Castro, John Ingraham, Suzanne Paley, Ian T Paulsen, Martín Peralta-Gil, and Peter D Karp. Ecocyc: a comprehensive database resource for escherichia coli. *Nucleic acids research*, 33(suppl 1):D334–D337, 2005.

[27] YongChul Kwon, Magdalena Balazinska, Bill Howe, and Jerome Rolia. Skewtune: mitigating skew in mapreduce applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 25–36. ACM, 2012.

[28] Patrick Laube, Marc van Kreveld, and Stephan Imfeld. Finding remodetecting relative motion patterns in geospatial lifelines. In *Developments in spatial data handling*, pages 201–215. Springer, 2005.

[29] Srivatsan Laxman, PS Sastry, and KP Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419. ACM, 2007.

[30] Xiaohui Li, Vaida Ceikute, Christian S Jensen, and Kian-Lee Tan. Effective online group discovery in trajectory databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2752–2766, 2013.

[31] Xiaohui Li and Tan Kian-Lee. *Managing moving objects and their trajectories.* PhD thesis, National University of Singapore, 2013.

[32] Yuxuan Li, James Bailey, and Lars Kulik. Efficient mining of platoon patterns in trajectory databases. *Data & Knowledge Engineering*, 100:167–187, 2015.

[33] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

[34] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289, 1997.

[35] Jayanta Mondal and Amol Deshpande. Eagr: Supporting continuous ego-centric aggregate queries over large dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 1335–1346. ACM, 2014.

[36] Jian Pei, Jiawei Han, Runying Mao, et al. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, pages 21–30, 2000.

[37] Afroza Sultana, Naeemul Hassan, Chengkai Li, Jun Yang, and Cong Yu. Incremental discovery of prominent situational facts. In *2014 IEEE 30th International Conference on Data Engineering*, pages 112–123. IEEE, 2014.

[38] Nikolaj Tatti and Boris Cule. Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1):34–66, 2012.

[39] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008.

[40] Virginia Vassilevska and Ali Pinar. Finding nonoverlapping dense blocks of a sparse matrix. *Lawrence Berkeley National Laboratory*, 2004.

[41] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245. ACM, 2003.

[42] Yida Wang, Ee-Peng Lim, and San-Yih Hwang. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering*, 57(3):240–282, 2006.

[43] Zhengkui Wang, Qi Fan, Huiju Wang, Kian-Lee Tan, Divyakant Agrawal, and Amr El Abbadi. Pagrol: parallel graph olap over large-scale attributed graphs. In *2014 IEEE 30th International Conference on Data Engineering*, pages 496–507. IEEE, 2014.

[44] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. On one of the few objects. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1487–1495. ACM, 2012.

[45] Xifeng Yan, Bin He, Feida Zhu, and Jiawei Han. Top-k aggregation queries over large networks. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 377–380. IEEE, 2010.

[46] Hilmi Yildirim, Vineet Chaoji, and Mohammed J Zaki. Dagger: A scalable index for reachability queries in large dynamic graphs. *arXiv preprint arXiv:1301.0977*, 2013.

[47] Jeffrey Xu Yu and Jiefeng Cheng. Graph reachability queries: A survey. In *Managing and Mining Graph Data*, pages 181–215. Springer, 2010.

[48] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[49] Fred Zemke. What. s new in sql: 2011. *ACM SIGMOD Record*, 41(1):67–73, 2012.

[50] Gensheng Zhang, Xiao Jiang, Ping Luo, Min Wang, and Chengkai Li. Discovering general prominent streaks in sequence data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):9, 2014.

[51] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph cube: on warehousing and olap multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 853–864. ACM, 2011.

[52] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. On discovery of gathering patterns from trajectories. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 242–253. IEEE, 2013.

[53] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[54] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

[55] Wenzhi Zhou, Hongyan Liu, and Hong Cheng. Mining closed episodes from event sequences efficiently. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 310–318. Springer, 2010.