Chapter 2

Literature Review

Our proposed neighborhood queries are inspired by the usefulness of window functions in relational analytic queries [59]. A window function in SQL specifies a set of partitioning attributes A and an aggregation function f. Its evaluation first partitions the input records based on A to compute f for each partition, and each input record is then associated with the aggregate value corresponding to the partition that contains the record. Several optimization techniques [12, 5] have also been developed to evaluate complex SQL queries involving multiple window functions.

However, the semantic and evaluation of the window function are restricted by the relational model. As been analyzed previously, SQL window functions require tuples to be sorted in order to form individual windows. However such a need is hard to meet in other data domains. Therefore, optimization techniques that are developed for the relational model become inapplicable in other domains. Nevertheless, there are quite a few works that related to the neighborhood queries that we have proposed and we summarize them in this section.

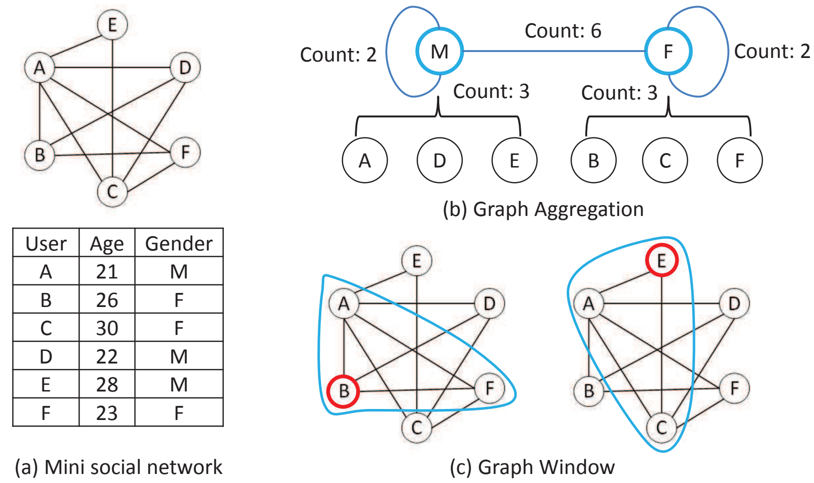
2.1 Graph Window Queries

2.1.1 Graph Aggregation

Previous graph data analytics focus on graph aggregation [61, 52, 13, 48], which are different from Graph Window Queries (GWQ). In a general model, graph aggregation comprises three steps: (1) partition graph based on attributes of vertex (and/or edges), (2) aggregate each partition to form Aggregated Nodes, and (3) link each aggregate node to form one Aggregated Graph. An illustration of the Graph Aggregation is shown in Figure 2.1 (b). In the first step, the input graph is partitioned on the"Gender" attribute of vertexes which results in two partitions. In the second step, two aggregated nodes are formed, i.e., M (stands for Male) containing nodes A, D, E and F (stands for female) containing nodes B, C, F. In the third step, the links between M and F are added, with the "count" attached on each links. Differently, Graph Window Queries perform graph analytics from the vertex-centric perspective. In GWQ, the neighborhood structure of each vertex form overlapping partitions. Then, analytics are computed over each neighborhood structure. In Figure 2.1 (c), the neighborhood structure of B and E are highlighted. Clearly, the GWQ is different from graph aggregation and they could not model each other.

2.1.2 Reachability Queries and Indexes

Classic reachability queries, which answer whether two vertexes are connected, have been studied extensively in literature. To facilitate fast query processing, many indexes are proposed [15, 16, 53, 57]. Although our graph window queries can be built on top of the reachability queries, directly using these techniques is inefficient. For example, the most related reachability query to our k-hop window query is the k-reach query [16] which test if an input pair of vertexes is within a k-hop distance. In order to compute the k-hop window query for n vertexes, there would be $\theta(n^2)$



User	Age	Gender
A	21	M
B	26	F
C	30	F
D	22	M
E	28	M
F	23	F

Figure 2.1: Illustration of Graph Aggregation and Graph Window Queries. (a) is an example social network, (b) is graph aggregation, (c) is the window of vertexes B and E.

reachability tests. This would be inefficient on graphs with over millions of vertexes.

2.1.3 Top-k Neighborhoods

In [55], the authors investigated the problem of finding the vertexes that have top-k highest aggregate values over their h-hop neighbors. This is similar to our k-hop query, while the difference is that they focus on providing pruning techniques to select the k best vertexes and our graph window query aims to compute the analytics for each vertex. Therefore, in our setting, the pruning techniques in [55] does not take effect and would be equivalent to the non-indexed approach as described in Section **??**.

2.1.4 Egocentric Networks

Egocentric networks [40, 43] have been playing an important role in network study. The egocentric networks refer to the neighborhood structure of each vertex in a graph. Although there are many works have studied on egocentric networks in structural analysis, they do not consider efficient processing of data analytics (e.g., aggregation) within each egocentric networks. Recently, Jayanta et.al. [44] proposed an EAGR system to summarize attribute information among each vertex's neighborhoods. Their technique is to build an overlay graph to leverage the shared components among vertexes' neighborhoods structures to boost query processing. Technically, EAGR runs in iterations and starts with the vertex-neighborhood mapping as the initial overlay graph. During each iteration, it sorts vertexes in an overlay graph according to their neighborhood information. Then an FP-Tree [26] is built to mining the largest shared components based on the sorted vertexes. As the algorithm iterates, the overlay graph evolves to be sparser.

The main drawback of EAGR is its high demands of resources on the overlay construction. In terms of memory cost, EAGR assume the initial vertex-neighbor mapping can be stored in memory. However, the assumption does not scale well for computing higher hop windows (such as k $\geq$ 2). For instance, a LiveJournal social network graph [1] (4.8M nodes, 69M edges) generates over 100GB mapping information for k=2 in adjacency list representation. If the neighborhood information is resided in disk, the performance of EAGR will largely reduced. In terms of computational cost, EAGR requires to sort all vertexes in a graph and build an FP-Tree in each iteration. When the graph has millions of vertexes, the indexing is largely slow down.

We tackle these drawbacks by adopting a hashing based approach that clusters each vertex according to its neighborhood similarity. During the hashing, a vertex's neighborhood information is computed on-the-fly. As compared to the sorting based approach, we do not require vertex's neighborhood to be resided in memory. In order to reduce the repetitive computation, we adopt a Dense Block heuristic to leverage the shared components among vertexes' neighborhoods. We then propose an estimation scheme that further reduces the number of neighborhood accesses. Experiments show that our schemes outperform EARG in both query processing and memory usage. Our methods are able to perform well even when EAGR algorithm

[1]Available at http://snap.stanford.edu/data/index.html, which is used [44]

fails when neighborhood information overwhelms system's memory, and our methods takes much shorter indexing time.

2.2 k-Sketch Query

Our proposed k-Sketch query is closely related to three areas: automatic news detection, frequent episode mining and top-k diversity query.

2.2.1 Automatic News Detection

An prominent usage of our k-Sketch query is to detect news themes to support *Computational Journalism*. Prior to our work, pioneer works [46, 54, 60] have studied various patterns in sequenced events to discover phenomenal news themes. Example of these news themes are presented in Table 2.1.

Sultana et al. [46] proposed the *Situational Fact* pattern. It aims to find the constraint-measure pair ($\langle C, M \rangle$) where a given event t is a skyline in dimension M among all events matching constraints C. For example, given an event in the following form: [name:Paul George, score:21, rebound:11, assist:5, block:2, steal:3, team:Pacers, opteam:Bulls, date:20130205, result:win], the constrain dimensions are (name, team, opteam, date, result) and the measure dimensions are (score, rebound, assist, block, steal). As shown in Table 2.1, the corresponding *situational fact* selects C = {team, date} and M = {score, assist, rebound} for this event as under such constraint-measure this event is a skyline. *Situational fact* is different from the *rank-aware streak* because it does not consider the consecutive event for a subject.

We etl al. [54] proposed the *One-of-the-Few* pattern to detect news themes with some rarities. *One-of-the-few* aims to select a set of measure dimensions m such that a given event is in not dominated by more than k events in M. For example, given an event in the following form: [name:Oscar Robertson, score:26710, rebound:7804, assist:9887, steal:77, block:4], the *one-of-the-few* selects the measure dimensions $M =$ {score, rebound, assist} as under these dimension, this event is not dominated by other events. Clearly the *one-of-the-few* pattern is different from k-Sketch.

Method	**Example news theme**
Situational facts [46]	Paul George had 21 points, 11 rebounds and 5 assists to become the first Pacers player with a 20/10/5 (points/rebounds/assists) game against the Bulls since Detlef Schrempf in December 1992.
One-of-the-few facts [54]	There is no player in NBA history with more points, more rebounds, and more assists than Oscar Robertson in one's career
Prominent streak [60]	Kobe scored 40+ in 9 straight games!
Rank-aware streak	Kobe scored 40+ in 9 straight games ranked 4th in NBA history!

Table 2.1: Examples of different news themes

Zhang et al.[60] proposed using prominent streak to generate interesting news themes. In [60], a *Prominent Streak* is characterized by two dimensions which are the window length and the minimum value of all events in the window. The objective is to discover the skyline (i.e., non-dominated) streaks where the dominance relationship is defined among streaks of the same subject. Our model differs from [60] in two aspects. First, we look at the global prominence (quantified by the rank) among all subjects rather than local prominence (quantified by the dominance) within one subject. Second, our model provides the best k-sketch for each subject whereas [60] returns a dominating set which could be potentially large.

2.2.2 Frequent Episode Mining

In time sequenced data mining, an episode [42, 65, 47, 35] is defined as a collection of time sequenced events which occur together within a time window. The uniqueness of an episode is determined by the contained events. The objective is to discover episodes whose occurrences exceeding a support threshold. Our sketch discovery differs from the episode mining in three major aspects. First, an episode is associated with a categorical value while our sketch is defined on numerical values. Second, the episodes are selected based on the occurrence, while in sketch, news themes are generated in a rank-aware manner. Finally, episode mining does not restrict its output size, while sketch only outputs the best k news themes. As such, episode mining techniques cannot be straightforwardly applied to sketch discovery.

2.2.3 Top-k Diversity Query

Top-k diversity queries [1, 7, 22, 14] aim to find a subset of objects to maximize a scoring function. The scoring function normally penalizes a subset if it contains similar elements. Our sketch discovery problem has two important distinctions against the top-k diversity queries. First, the inputs of the scoring function are known in advance in top-k diversity queries; whereas in our problem, the ranks of event windows are unknown. Since their calculations are expensive, we need to devise efficient methods to compute the ranks. Second, existing methods for online diversity queries [7, 22, 14] only study the update on a single result set when a new event arrives. However our online sketch maintenance incurs the problem of multiple sketch updates for each new event. Such a complex update pattern has not been studied yet and hence there is a need to develop efficient update scheme.

2.3 Movement Pattern Discovery Query

Previous works related to our trajectory pattern mining query can be grouped into three categories: *co-movement patterns*, *dynamic moving patterns* and *trajectory mining frameworks.*

2.3.1 Flock and Convoy

The difference between *flock* and *convoy* lies in the object clustering methods. In *flock*, objects are clustered based on their distances. Specifically, the objects in the same cluster need to have a pair-wised distance less than *min_dist*. This essentially requires the objects to be within a disk-region of delimiter less than *min_dist*. In contrast, *convoy* clusters objects using density-based spatial clustering [24]. Technically, *flock* utilizes a m^{th}-order Voronoi diagram [34] to detect whether a subset of object with size greater than m stays in a disk-region. *Convoy* employs a trajectory simplification [21] technique to boost pairwise distance computations in the density-based clustering. After clustering, both *flock* and *convoy* use a sequential scanning method to examine each snapshots. During the scan, object groups that appear in consecutive timestamps are detected. Meanwhile, the object groups that do not match the consecutive constraint are pruned. However, such a method faces high complexity when supporting other patterns. For instance, in *swarm*, the candidate set during the sequential scanning grows exponentially, and many candidates can only be pruned after the entire snapshots are scanned.

2.3.2 Group, Swarm and Platoon

Different from *flock* and *convoy*, all the *group*,*swarm* and *platoon* patterns have more relaxed constraints on the pattern duration. Therefore, their techniques of mining are of the same skeleton. The main idea of mining is to grow object set from an empty set in a depth-first manner. During the growth, various pruning techniques are provided to prune unnecessary branches. *Group* pattern uses a VG-graph to guide the pruning of false candidates [51]. *Swarm* designs two more pruning rules called backward pruning and forward pruning [39]. *Platoon* further leverages a prefix table structure to steer the depth-first search. As shown by Li et.al. [38], *platoon* outperforms other two methods in efficiency. However, the three patterns are not able to directly discover GCMPs. Furthermore, their pruning rules heavily rely on the depth-first search nature, which lose their efficiencies in the parallel scenario.

2.3.3 Other Related Trajectory Patterns

A closely related literature to co-movement patterns is the *dynamic moving* patterns. Instead of requiring the same set of object traveling together, *dynamic moving* patterns allow objects to temporally join or leave a group. Typical works include *moving clusters* [31], *evolving convoy* [2], *gathering* [62] etc. These works cannot model our GCMP since they enforce the global consecutiveness on the timestamps of a pattern.

2.3.4 Trajectory Mining Frameworks

Jinno et al. in [30] designed a MapReduce based algorithm to efficiently support T-pattern discovery, where a T-pattern is a set of objects visiting the same place at simliar time. Li et al. proposed a framework of processing online *evolving group* pattern [36], which focuses on supporting efficient updates of arriving objects. As these works essentially differ from co-movement pattern, their techniques cannot be directly applied to discover GCMPs.

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] Erica J Briscoe, D Scott Appling, Rudolph L Mappus IV, and Heather Hayes. Determining credibility from social network structure. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1418–1424. ACM, 2013.

[9] 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.

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

[11] Juan Miguel Campanario. Empirical study of journal impact factors obtained using the classical two-year citation window versus a five-year citation window. *Scientometrics*, 87(1):189–204, 2011.

[12] 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.

[13] 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.

[14] 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.

[15] James Cheng, Silu Huang, Huanhuan Wu, and Ada Wai-Chee Fu. Tf-label: a topological-folding labeling scheme for reachability querying in a large graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 193–204. ACM, 2013.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Lianghao Dai, Jar-der Luo, Xiaoming Fu, and Zhichao Li. Predicting offline behaviors from online features: an ego-centric dynamical network approach. In *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 17–24. ACM, 2012.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

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

[28] Clyde W Holsapple and Wenhong Luo. A citation analysis of influences on collaborative computing research. *Computer Supported Cooperative Work (CSCW)*, 12(3):351–366, 2003.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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.

[40] Huaiyu Harry Ma, Steven Gustafson, Abha Moitra, and David Bracewell. Ego-centric network sampling in viral marketing applications. In *Mining and Analyzing Social Networks*, pages 35–51. Springer, 2010.

[41] Nan Ma, Jiancheng Guan, and Yi Zhao. Bringing pagerank to the citation analysis. *Information Processing & Management*, 44(2):800–810, 2008.

[42] 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.

[43] Peter V Marsden. Egocentric and sociocentric measures of network centrality. *Social networks*, 24(4):407–422, 2002.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

[53] Hao Wei, Jeffrey Xu Yu, Can Lu, and Ruoming Jin. Reachability querying: An independent permutation labeling approach. *Proceedings of the VLDB Endowment*, 7(12):1191–1202, 2014.

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

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

[64] 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.

[65] 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.