Chapter 2

Literature Review

2.1 Graph Window Queries

Our proposed graph window functions (GWFs) for graph databases is inspired by the usefulness of window functions in relational analytic queries [29].

A window function in SQL typically 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 [5, 3] have also been developed to evaluate complex SQL queries involving multiple window functions.

However, the semantics and evaluation of window functions are very different between relational and graph contexts. Specifically, the partitions (i.e., subgraphs) associated with GWFs are not necessarily disjoint; thus, the evaluation techniques developed for relational context [5, 3] are not applicable to GWFs.

GWFs are also different from graph aggregation [31, 25, 6, 23] in graph OLAP. In graph OLAP, information in a graph are summarized by partitioning the graph's nodes/edges (based on some attribute values) and computing aggregate values for each partition. GWFs, on the other hand, compute aggregate values for each graph node w.r.t. the subgraph associated with the node. Indeed, such differences also arise in the relational context, where different techniques are developed to evaluate OLAP and window function queries.

In [27], the authors investigated the problem of finding the vertices that have top-k highest aggregate values over their h-hop neighbors. They proposed mechanisms to prune the computation by using two properties: First, the locality between vertices is used to propagate the upper-bound of aggregation; Second, the upper-bound value of aggregates can be estimated from the distribution of attribute values. However, all these pruning techniques are not applicable in our work, as we need to compute the aggregation value for every vertex. In such a scenario, techniques in [27] degrade to the non-indexed approach as described in Section 4.

Indexing techniques have been proposed to efficiently determine whether an input pair of vertices is within a distance of k-hops (e.g. k-reach index [8]) or reachable (e.g. reachability index [28]). However, such techniques are not efficient for computing the k-hop window or topological window for a set of n vertices with a time complexity of $O(n^2)$.

[20] proposed an EAGR system, which uses the famous VNM heuristic and Frequent-Pattern Tree to find the shared component among each vertex's neighborhoods. It starts by building an overlay graph as a bipartite graph representing the vertex-neighbor mapping. Then it aims to find the bi-cliques in the overlay graph. Each bi-clique represents a set of vertices whose neighborhood aggregates can be shared. Once a bi-clique is found, it is inserted back to the overlay graph as a virtual node to remove redundant edges. EAGR find bicliques in iterations. During each iteration, it sorts each vertices in overlay graph by their neighborhood information. Then the sorted vertices are split into equal-sized chunks. For each chunk, it then builds a FP-Tree to mining the large bi-cliques. As the algorithm iterates, the overlay graph

evolves to be less dense.

The main drawback of EAGR is its demands of high memory usage on overlay construction. It requires the neighborhood information to be pre-computed, which is used in the sorting phase of each iteration. In EAGR the neighborhood information is assumed to 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. Similarly, if the neighborhood information is computed on-the-fly, EAGR needs to perform the computation in each iteration, which largely increases indexing time.

We tackle this drawback by adapting a hash based approach that clusters each vertex based on its neighborhood similarity. During the hashing, the vertex's neighborhood information is computed on-the-fly. As compared to sorting based approach, we do not require vertex's neighborhood to be pre-reside in memory. In order to reduce the repetitive computation of vertex's neighborhood, we further propose an estimation based indexing construction algorithm that only require a vertex's small hop neighborhood to be computed during clustering. As our experiments show, our proposed methods can perform well even when EAGR algorithm fails when neighborhood information overwhelms system's memory. To further reduce the neighborhood access, we adapted a Dense Block heuristic process each vertex in one pass. Experiments shows that the performance of our heuristic is comparable to EAGR's, but with much shorter indexing time.

[1] Available at http://snap.stanford.edu/data/index.html, which is used [20]

2.2 k-Sketch Query

In this section, we briefly review three closely related areas: automatic news detection, frequent episode mining and top-k diversity query.

2.2.1 Automatic News Detection

Earlier works on automatic news theme generation were focused on finding interesting themes from a single event. For example, Sultana et al. [21] proposed the *Situational Fact* pattern, which is modeled as a skyline point under certain dimensions. Wu et al. [26] proposed the *One-of-the-Few* concept to detect news themes with some rarities. Examples of candidate news themes for the above two patterns are illustrated in Table 2.1.

Method	**Example news theme**
Situational facts [21]	Ellen's tweet generates 3.3M retweets with 170,000 comments.
One-of-the-few facts [26]	Perry is one of the three candidates who received $600k
Prominent streak [30]	Kobe scored 40+ in 9 straight games!
Rank-aware theme	Kobe scored 40+ in 9 straight games ranked 4th in NBA history!

Table 2.1: Examples of different news themes

Zhang et al.[30] proposed using prominent streak to generate interesting news themes. In [30], a *Prominent Streak* is characterized by a 2D point which represents the window length and the minimum value of all events in the window. The objective is to discover the non-dominated event windows for each subject, where the dominance relationship is defined among streaks of the same subject. Our model differs from [30] 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

[30] returns a dominating set which could be potentially large.

2.2.2 Frequent Episode Mining

In time sequenced data mining, an episode [19, 33, 22, 15] 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, 4, 10, 7] 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 [4, 10, 7] 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

Related works can be grouped into three categories: *co-movement patterns*, *dynamic moving patterns* and *trajectory mining frameworks*. In this section, we distinguish the parallel GCMP mining from these concepts.

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 [11]. Technically, *flock* utilizes a m^{th}-order Voronoi diagram [14] to detect whether a subset of object with size greater than m stays in a disk-region. *Convoy* employs a trajectory simplification [9] 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 [24]. *Swarm* designs two more pruning rules called backward pruning and forward pruning [18]. *Platoon* further leverages a prefix table structure to steer the depth-first search. As shown by Li et.al. [17], *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* [13], *evolving convoy* [2], *gathering* [32] 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 [12] 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 [16], 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.