Efficient and Scalable Aggregate Computation on Temporal Graphs

Vincent Le Claire
supervised by Prof. Dr. Peter Fischer
University of Augsburg
Augsburg, Germany
vincent.leclaire@informatik.uni-augsburg.de

ABSTRACT

Many applications such as social networks generate big volumes of graph data that has additional temporal information or changes rapidly, which has lead to a significant amount of research on temporal graphs. Existing and ongoing work on temporal graphs has focused on path problems and graph databases in general. Aggregations, which are very common for relational data, are just as insightful for temporal graph data, but need to be computed efficiently and scalable. To aggreagte efficiently, useful operations on graphs need to be selected, the composition of aggregation functions needs to be investigated, and the distribution of the calculation must be studied. After tackling the research question with the simplifying assumption of static communities (which are highly coupled nodes), then the parallelization of the problem is investigated, and finally, the simplifying assumption is removed for the general case of dynamic communities. As aggregations play an important role in (temporal) relational data, this direction of research might establish them in temporal graphs just as well.

1. INTRODUCTION

Much data in the real world can naturally be expressed as a graph. For example, in a social network, graph nodes can represent people, and graph edges then represent that two people are friends in this social network. As another example, consider a traffic network that consists of streets that are interconnected by crossways. Many of these graphs are not static but change constantly. Social networks, as they are intended for interaction, are obviously dynamic. Likewise, traffic networks are not static as temporary road works or other means of blocked roads are sources of change. For temporal graphs (aka dynamic graphs), the wide range of use cases results in a plethora of different definitions and algorithms, as no common view or formalization has – so far – been achieved. One important aspect of (not only temporal) graph data is the aggregation of many values to a smaller set of meaningful metrics, as often seen in, e.g., business intelligence.

In the context of aggregates in temporal graphs, consider the following example: In a social network, users can be grouped into several communities that have stronger bounds inside than to the outside. A query for this use case might be: "For the most connected member of each community, how much did the number of friends change over the progress of every calendar week of last year?" That query has several complementary aspects: (a) It accesses the graph in substructures of varying amount (nodes, neighbours, communities), which resembles (but exceeds) the grouping of relational aggregations. (b) It has a temporal aspect of varying granularity (first, the number of friends is calculated for each week; second, only the data of the last year is examined). (c) It uses different aggregation functions (here: `sum`, `argmax`). (d) It re-uses an aggregation (the sum of the friends has to be calculated for every node because it is needed to find the most connected member in a first step).

1.1 Motivation

The core of this research proposal is to investigate aggregates on temporal graphs rather than arbitrary temporal graph algorithms. There are several reasons for this focus:

- There are many use cases of aggregations, among them, for example, weekly analytics of air traffic connections, or averaging the number of contacts between people during a global outbreak of a disease. Furthermore, many use cases of relational temporal aggregates are applicable for graphs, too. The importance of aggregates on temporal graph data is also recognized by their use in benchmarks: The LDBC Social Network Benchmark uses a temporal graph for its data, and its business intelligence workload sports several temporal graph data aggregations.

- For several temporal graph problems, there exist efficient, often parallel and/or incremental algorithms. Yet, these algorithms tend to make specific assumptions on the data model and the workload; when the uses cases are more general or differ slightly in their assumptions, the scalability suffers. Considering the more limited scope, temporal aggregates over graphs are likely to provide more room for optimization and less sensitivity to workload and data model changes – we will outline this reasoning later in this paper.

Proceedings of the VLDB 2020 PhD Workshop, August 31st, 2020. Tokyo, Japan. Copyright (C) 2020 for this paper by its authors. Copying permitted for private and academic purposes.

CEUR-WS.org/Vol-2652/paper11.pdf

- Despite all the points outlined above, aggregates over temporal graphs have not been a major research direction so far. This is in stark contrast to classical graph problems such as shortest path or adapting existing workloads on temporal data using snapshots.

2. PREVIOUS WORK

2.1 Use Cases of Temporal Graphs

One of the first examples of using temporal graphs is an article by Cervoni et al. [4]. The authors use temporal graphs for the calculation of temporal constraints.

Use cases of temporal graph metrics have been mentioned by Tang et al. [24]. Nicosia et al. [21] describe some metrics and differences to their static graph counterparts.

Holme and Saramäki [11] review many use cases for temporal networks and describe the different modeling of them.

There are examples of the usage of temporal graphs in medicine: Wainer and Sandri [26] use them for medicial diagnostics. Liu et al. [18] model medical events of a single patient and their relationship using temporal graphs.

For visualizing scientific publications and their interrelations, Erten et al. [8] use temporal graphs.

The wide range of uses of temporal graphs (along with our motivating example) reassures us that the addition of temporal information to graphs is sensible and not an academic niche.

2.2 Temporal Databases and Aggregates

The research on temporal aggregation dates back to the 1990s, co-occuring to the seminal work on temporal databases. One of the first publications regarding temporal aggregation (on relational databases) is by Snodgrass et al. [23], which is a direct result of their introduction of TSQL2. TSQL2 added temporal semantics on top of the SQL standard, which has become mostly obsolete by the inclusion of temporal semantics into SQL:2011. Following, Kline and Snodgrass further evaluate the calcuation of temporal aggregates in [16]. Zhang et al. [27] appended ranges to the temporal aggregates.

Böhlen et al. [2] focus on multi-dimensional temporal aggregates. They distinguish between two (or more) time dimensions: one or more application times, which describe at which time a tuple is valid, and a system time, which defines the time where the database system is aware of the tuple. Cheng [5] describes the aggregation of null-time intervals.

In 2013, Kaufmann et al. [15] introduce the Timeline Index, which is a main-memory structure that efficiently supports temporal aggregations on system time and other temporal operations in a relational database. Ideas of the Timeline Index will play an important role in our first step of research, as we will show in Section 4. Other work mentioned in this section lays the foundation to understand what temporal aggregations are and what expressive power they have.

2.3 Processing Systems and Databases

For our purposes, managing large-scale evolving graphs relates to both graph databases and processing systems:

Fernandes and Bernardino [9] compare several graph database systems. The most popular among them is Neo4j [1]. TGraph [12] is an ACID-compliant extension of Neo4j for temporal range queries. TGraph assumes that the graph structure changes rarely, while attributes of nodes and edges change over time. A custom data structure called DPS (Dynamic Property Storage) handles the dynamic attributes of nodes and edges, while the graph itself and the static attributes of the nodes and edges are kept in the existing Neo4j format.

Google's Pregel [20] is an early example of a graph processing system, introduced in 2010. It is a distributed system for big graphs providing a very fine-grained concurrency model; like MapReduce [7], it relies on synchronization rounds (BSP). GraphLab [19] positions itself in the machine learning domain, it thus investigates relaxed coordination models. Kineograph [6] is a distributed graph system for dynamic (that means, frequently changing) graphs. It stores its graph as snapshots for different points in time. Similarly, GraphTau [13] is distributed and stores snapshots; it builds upon Apache Spark. GraphChi [17] focuses on disk-based evolving graphs. A recent general-purposes system to analyze huge temporal graphs is Gradoop [14].

Generally speaking, graph databases tend to be limited in scalability for analytical temporal workloads, while the broad support for temporal graph operations in processing system makes it hard to devise optimization specific to aggregations.

2.4 Incremental Aggregate Maintenance

Efficient computation of graph aggregates may draw heavily from existing work on incremental aggregate computation, notably from (a) incremental view maintenance and (b) incremental computation of streaming aggregates.

Materialized views are well established in relational databases; Halevy [10] gives an extensive overview of how they are used to answer queries. Materialized views contain the result of a query, which can be, for example, an aggregation. When the underlying data changes, the materialized view is updated. Many approaches exist to perform this in an incremental manner. There is an additional interesting line of work that is helpful for our problem setting: Answering queries with views and thus view containment could provide valuable insights on stacking distinct aggregates – which is clearly harder than, for example, a cube operator with the same aggregation function.

Tangwongsan et al. [25] provide a comprehensive solution on sliding-window aggregations. For in-order arrival/expiry of the data they achieve $O(1)$, while for out-of-order expiry they still achieve $O(1)$ in the best case (which is in-order) and up to $O(\log n)$ in total out-of-order execution. The latter model matches well with arbitrary lifetime intervals of temporal data.

3. PROBLEM DEFINITION

The problem we are thus trying to tackle in our work is to efficiently compute (combinations of) aggregate values over (possibly changing) substructures of highly dynamic, huge graphs.

The implications of this problem statement can further be broken down along the following dimensions:

1. **Graph properties to aggregate**: Compared to relational or streaming/ordered models, graphs allow for an additional range of (possibly very costly and hard-to-optimize) operations such as reachability or shortest paths. These may serve as an input to the aggregate metrics (such as betweenness centrality), yet the effort

to compute them may outstrip the cost of the aggregations.

It is therefore an important tradeoff to consider how rich the support for such operations needs to be. Our current take is to allow limited neighborhood access but not arbitrary reach/iteration.

2. **Hierarchical composition of distinct aggregate functions**: While a significant body of works exists on refining/combining values over the same aggregation function (like drill-down or roll-up data cubes), using several distinct aggregation functions "on top" of each other (e.g. maximum on top of sum in the example) is not as well studied for effective evaluation.

3. **Several dimensions of varying granularity**

   (a) **Time**: As already observed, temporal aggregates may cover varying degrees of time (points, disjoint/overlapping intervals). In aggregation hierarchies, the granularities may necessarily have a containment relationship (e.g. weeks and months).

   (b) **Graph structure**: While relational data is typically "grouped" over an attribute or combinations of attributes (which often have an obvious containment relationship), graphs provide a wider range of options, covering individual nodes, neighborhoods, communities, connected components or the entire graph. Determining these "groups" may itself be a complex and expensive operation. Furthermore, on temporal data the membership of sets of lower-level groups may change over time, such as nodes changing their community, making partial computation harder to maintain.

4. **Distributed computation**: The expected problem size makes single-thread, main-memory approaches as well as full recomputation on change not very appealing. While these aspects are fairly well-understood for relational streaming and ongoing work exist for static graphs, research for partitioning and distributed state management of dynamic graphs is still in its early stages.

To narrow the problem definition, we assume a changing graph whose nodes and edges may have additional properties assinged that may also change over time, similar to the approach Huang et al. have for TGraph [12]. To give more expressive power, we assume an interval time model instead of only allowing points in time for queries. At this time, we do not assume a graph with directed nor undirected edges.

4. RESEARCH PLAN

An interesting foundation for this type of research is the Timeline Index approach of Kaufmann et al. [15]. It expresses the (relational) temporal data as a changelog of activated/deactived data points. Validity information snapshots reduce the space to be processed and allow for archival or distribution. Due to these design choices and also due to the optimization for main memory, it provides an efficient underpinning for incremental aggregations.

Temporal graphs can clearly also be expressed as a (set) of changelog(s) of nodes, edges and properties. Dealing with the underlying structure of a graphs is one of the most interesting challenges when adapting Timeline, as it affects the scope of aggregations (e.g. over nodes or communities) as well as the means of partitioning the graph for parallel execution. To make this challenge more tractable, we first keep the aggregation scopes static and drop this requirement in our last step. Additionally, we consider also materializing (partial) aggregate values to speed up aggregate computation over time, graph structure and aggregate function composition. In turn, this also introduces an additional partitioning problem.

4.1 Interfacing with Graph Computations

As a first step, we are investigating how to best express the usage of graph properties required for the metric computation. The goal is to define a suitable subset of graph operations that can be computed in an incremental manner over temporal data and can therefore "drive" an efficient temporal aggregation process. While some operations are obviously both useful for metrics and easy to derive (such property values on a single node or edge) and others are useful, but extremely hard to compute incrementally (general temporal paths), the space in between is not well-charted. We plan to investigate a broad range of metrics from use cases in order to further understand the requirements. The results will allow use to also adapt the design of the graph data storage and programming interface.

4.2 Composition of Aggregates

A second, but orthogonal problem is the combination of multiple aggregations into a hierarchy. While a wide range of single-level aggregations may be supported for incremental computation by applying ideas of, e.g., Tangwongsan et al. [25] (maybe with extensions for update sets), re-using partial results over a DAG of distinct aggregation functions clearly raises its own set of challenges. We plan to investigate expression or view containment approaches to consider both dependencies among aggregation functions and the varying granularities of time and graph structure, possibly deriving "core" aggregation parts in lower levels that can be shared for multiple aggregates or combined among these dimensions. Furthermore, deciding on when and what to materialize will be further area of investigation.

4.3 Parallelization

In order to achieve a significant amount of scalability, the computations need to be parallelized.

We see two main directions: (a) Partitioning (social) graph data in order to both maximize parallel computation and minimize communication is an ongoing challenge in the research community, e.g., due to skewed distribution of the data. We expect that some parts of the changelog, the snapshots, and the partial aggregates may just cover a small subset of highly active and highly connected nodes, while other parts may cover larger sets of lower activity. We also may have to investigate possible "cuts" within graph structures such as communities or even nodes, so partial aggregates can be computed with more parallelism.

(b) In contrast to general graph computations, many aggregate functions do not require strict consistency rules in order to produce correct results, similar to what CRDTs [22] can achieve in eventual consistency environements. The

temporal validity information provided with the data elements facilitates the reconciliation of different "episodes" and may further be used to derive the synchronization boundaries as only changes in the data require actual coordination.

4.4 Dynamic Communities

In our last step, we drop the assumption of static graph structures, so that the graph structures we aggregate over also change over time; hence, nodes and edges may migrate between substructures. This has two major consequences: (a) Determining communities is by itself an expensive operation, even on non-temporal graphs, typically scaling much faster than the number of edges. We already begun investigating some light-weight methods on the basis of Brandes et al. [3] that promise to allow for incremental community detection and evolution. (b) Sharing partial changelogs, snapshots or aggregates becomes more challenging when their membership or association to bigger structures changes. We plan to investigate methods for more fine-grained partitioning or association-strength partitioning strategies to minimize the number of large-scale recomputation.

5. CONCLUSIONS

Aggregations have important applications for relational data, and they are used frequently. Likewise, there are many use cases for aggregations of temporal graph data. There is a broad foundation of graph databases and research of other important temporal graph problems. Also, there are similarities in the challenges of computation of streaming data and the maintenance of materialized views, for example.

The research problem is four-fold: Graphs have different (sometimes very costly) operations; it is necessary to find a set of operations that is both efficient and expressive. Secondly, because some aggregations can be composed into others, it is wise to study how this can be done efficiently. Thirdly, the time dimension and the graph structure give granularity. Lastly, the problem should be solved distributive.

6. REFERENCES

[1] Neo4j. https://neo4j.com.

[2] M. Böhlen et al. Multi-dimensional aggregation for temporal data. In *EDBT*, pages 257–275, 2006.

[3] U. Brandes et al. On modularity clustering. *TKDE 2007*, 20(2):172–188, 2007.

[4] R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks. In *AIPS*, pages 13–18, 1994.

[5] K. Cheng. On computing temporal aggregates over null time intervals. In D. Benslimane et al., editors, *DEXA*, pages 67–79, Cham, 2017. Springer.

[6] R. Cheng et al. Kineograph: taking the pulse of a fast-changing and connected world. In *EuroSys 2012*, pages 85–98, 2012.

[7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *CACM*, 51(1):107–113, 2008.

[8] C. Erten et al. Exploring the computing literature using temporal graph visualization. In *Visualization and Data Analysis 2004*, volume 5295, pages 45–56. International Society for Optics and Photonics, 2004.

[9] D. Fernandes and J. Bernardino. Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In *DATA*, pages 373–380, 2018.

[10] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[11] P. Holme and J. Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[12] H. Huang et al. Tgraph: A temporal graph data management system. In *CIKM*, pages 2469–2472. ACM, 2016.

[13] A. P. Iyer et al. Time-evolving graph processing at scale. In *GRADES 2016*, pages 1–6, 2016.

[14] M. Junghanns et al. Gradoop: Scalable graph data management and analytics with hadoop. *arXiv preprint arXiv:1506.00548*, 2015.

[15] M. Kaufmann et al. Timeline index: a unified data structure for processing queries on temporal data in sap hana. In *SIGMOD*, pages 1173–1184. ACM, 2013.

[16] N. Kline and R. T. Snodgrass. Computing temporal aggregates. In *ICDE*, pages 222–231. IEEE, 1995.

[17] A. Kyrola, G. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a PC. In *OSDI 2012*, pages 31–46, 2012.

[18] C. Liu, F. Wang, J. Hu, and H. Xiong. Temporal phenotyping from longitudinal electronic health records: A graph based framework. In *ACM SIGKDD 2015*, pages 705–714. ACM, 2015.

[19] Y. Low et al. Distributed graphlab: A framework for machine learning in the cloud. *PVLDB*, 5(8):716–727, 2012.

[20] G. Malewicz et al. Pregel: a system for large-scale graph processing. In *SIGMOD*, pages 135–146, 2010.

[21] V. Nicosia et al. Graph metrics for temporal networks. In *Temporal networks*, pages 15–40. Springer, 2013.

[22] M. Shapiro et al. Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems*, pages 386–400. Springer, 2011.

[23] R. T. Snodgrass, S. Gomez, and L. E. McKenzie. Aggregates in the temporal query language tquel. *IEEE Transactions on Knowledge and Data Engineering*, 5(5):826–842, 1993.

[24] J. Tang et al. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*, pages 135–159. Springer, 2013.

[25] K. Tangwongsan, M. Hirzel, and S. Schneider. Optimal and general out-of-order sliding-window aggregation. *Proceedings of the VLDB Endowment*, 12(10):1167–1180, 2019.

[26] J. Wainer and S. Sandri. Fuzzy temporal/categorical information in diagnosis. *Journal of Intelligent Information Systems*, 13(1-2):9–26, 1999.

[27] D. Zhang et al. Efficient computation of temporal aggregates with range predicates. In *PODS 2001*, pages 237–245, 2001.