Streaming Graph Processing & Analytics

M. Tamer Özsu

University of Waterloo
David R. Cheriton School of Computer Science
https://cs.uwaterloo.ca/~tozsu
Graphs have become ubiquitous

Graph Databases Go Mainstream

Kurt Cagle, Former Contributor
COGNITIVE WORLD, Contributor Group ID
AI, Futurist, Technologist, Information Architect, Blogger

Graph technology is becoming mainstream, with knowledge bases leading the way. 

Every decade seems to have its database. During the 1990s, the relational database became the principal data environment, its ease of use and tabular arrangement making it a natural for the growing needs to power the data web. While relational databases remained strong, the 2000s saw the emergence of XML databases, and
Graphs have become ubiquitous

Understanding the maturing role of graph databases in the enterprise

Graph databases are making their way into enterprises and revealing the value of relationships in data sets.

According to figures from MarketsandMarkets Research Private Ltd., the graph database market is expected to reach $2.4 billion in annual revenue by 2023, growing at a 24% annual rate.

Graph databases are becoming the next big thing in data and analytics technology. According to Gartner, the application of graph processing and graph database management systems will grow at 100% annually through 2022 to continuously accelerate data preparation and enable more complex and adaptive data science.

Driving this growth is the belief that relationships between data should
Graphs have become ubiquitous

Understanding maturing role of databases in the enterprise

Graph databases are making their way in the enterprise. According to Gartner, the graph database management system is becoming more complex and adaptive data science.

Driving this growth is the belief that...

Graph databases are one of the 10 biggest data and analytics trends of 2019, according to Gartner. In fact, the advisory firm predicted that the category will experience a growth of 100% annually through 2022.

Gartner praises graph databases
Graphs have become ubiquitous

According to figures from MarketsandMarkets Research, it is expected that the $2.4 billion global graph database market
will grow at a CAGR of more than 20% through 2022. Graph databases are becoming a major technology. According to Gartner, the
industry analysts, graph databases are one of the 10 biggest data and analytics trends of 2019. In fact, the advisory firm predicted that
the category will experience a growth of 100% annually through 2022.

Driving this growth is the belief that real-world data is increasingly complex and adaptive. The rise of graphs is being driven
by the need to link and make sense of vast amounts of information. As a result, graph databases can be used to
understand the maturing role of databases in the enterprise.

Graph databases are becoming mainstream, but how can this technology improve your data management?

Why do experts say graph databases are headed for mainstream use?

Share in

Graph technology is becoming mainstream, with knowledge bases

Every decade seems to have its database. During the 1980s, becoming
the principal data environment, its ease of use made it natural for the growing needs to process
the databases. The 2000s saw the emergence of

Forbes

Debs 2020
Graphs – When Relationships are Important

Recent COVID-19 pandemic

- Model how people interact and influence each other, and how ideas and behaviours travel along social pathways

Graphs – When Relationships are Important

Recent COVID-19 pandemic

- Model how people interact and influence each other, and how ideas and behaviours travel along social pathways
- Epidemic search
  - Self assessment by checking connections
  - \{Place, flight, train, license plate\} → \{known cases\}
  - \{Source loc, Target loc\} → \{“edges” that connect them, flights, trains, vehicle license plates\}

Graphs – When Relationships are Important

Recent COVID-19 pandemic

- Model how people interact and influence each other, and how ideas and behaviours travel along social pathways
- Epidemic search
- Complex COVID-19 pathways
  - Looking at propagation in social networks
    [Kempe et al., 2003]
  - Linear threshold model
  - Independent cascade model

Graphs – When Relationships are Important

Recent COVID-19 pandemic

- Model how people interact and influence each other, and how ideas and behaviours travel along social pathways
- Epidemic search
- Complex COVID-19 pathways
- Contact tracing
  - Figuring out exactly how 5 people became infected in Tianjin
  - Vertices: people and places they traveled
  - Edges: people-people contact or travel
  - Paths: how infections link to known cases

Graphs – When Relationships are Important

Recent COVID-19 pandemic

- Model how people interact and influence each other, and how ideas and behaviours travel along social pathways
- Epidemic search
- Complex COVID-19 pathways
- Contact tracing
- Covid knowledge graph


https://covidgraph.org
Modern graphs are different and diverse

Internet
Social networks
Trade volumes & connections
Biological networks
Linked data
Road network
The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, M. Tamer Özsu
David R. Cheriton School of Computer Science
University of Waterloo
{s.sahu, amine.mhedhbi, semih.salihoglu, jimmy.lin, tamer.ozsu}@uwaterloo.ca

ABSTRACT

Graph processing is becoming increasingly prevalent across many application domains. In spite of this prevalence, there is little research about how graphs are actually used in practice. We conducted an online survey of 89 users, a review of the mailing lists, source repositories, and white papers of a large suite of graph software products, and in-person interviews with 6 users and 2 developers of these products. Our online survey aimed at understanding: (i) the types of graphs users have; (ii) the graph computations users run; (iii) which software do users use to perform their computations; and (iv) the major challenges users face when processing their graphs. We describe the participants’ responses to our questions highlighting common patterns and challenges. We further reviewed user feedback in the mailing lists, bug reports, and feature requests in the source repositories of a large suite of software products for processing graphs. Through our review we answer some new questions that were raised by participants’ responses and data we obtained revealing surprising facts about graph processing in practice. In particular, real-world graphs represent a very diverse range of entities and are often very large, scalability and visualization are undeniably the most pressing challenges faced by participants. We hope these findings can guide future research.

PVLDB Reference Format:

1. INTRODUCTION

Graph data representing connected entities and their relationships appear in many application domains, most naturally in social networks, the web, the semantic web, road maps, communication networks, bioinformatics, and finance, just to name a few examples. There has been a noticeable increase in the prevalence of work on graph processing both in research and as evidenced by the surge in the number of different commercial and research software for managing and processing graphs. Examples include graph database systems [3, 8, 14, 35, 48, 53], RDF engines [38, 64, 67], linear algebra software [6, 64], visualization software [13, 16], query languages [28].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or a fee. Articles from this volume were invited to present their results at The 44th International Conference on Very Large Data Bases, August 2018, Rio de Janeiro, Brazil. Proceedings of the VLDB Endowment, Vol. 11, No. 4 Copyright 2017 VLDB Endowment 2150-8097/17/12... $ 10.00. DOI: https://doi.org/10.14778/3164135.3164161

52-55, and distributed graph processing systems [17, 21, 27]. In the academic literature, a large number of publications that study numerous topics related to graph processing regularly appear across a wide spectrum of research venues. Despite their prevalent use, there is little research on how graph data is actually used in practice and the major challenges facing users of graph data, both in industry and research. In April 2017, we conducted an online survey across 89 users of 22 different software products, with the goal of answering 4 high-level questions:

(i) What types of graph data do users have?
(ii) What computations do users run on their graphs?
(iii) Which software do users use to perform their computations?
(iv) What are the major challenges users face when processing their graphs?

Our major findings are as follows:

• Variety: Graphs in practice represent a very wide variety of entities, many of which are not naturally thought of as vertices and edges. Most surprisingly, traditional enterprise data comprised of products, orders, and transactions, which are typically seen as the perfect fit for relational systems, appear to be a very common form of data represented in participants’ graphs.

• Ubiquity of Very Large Graphs: Many graphs in practice are very large, often containing over a billion edges. These large graphs represent a very wide range of entities and belong to organizations at all scales from very small enterprises to very large users. This is the first time we have heard a description of the type of graphs that are used in practice for a large number of machine learning and unsupervised learning applications. It is also the first time we have heard a description of the type of graphs that are used in practice for a large number of machine learning and unsupervised learning applications. It is also the first time we have heard a description of the type of graphs that are used in practice for the purpose of unsupervised learning applications.

• Challenge of Scalability: Scalability is unequivocally the most pressing challenge faced by participants. The ability to process very large graphs efficiently seems to be the biggest limitation of existing software.

• Visualization: Visualization is a very popular and central task in participants’ graph processing pipelines. After scalability, participants indicated visualizations as their second most pressing challenge, tied with challenges in graph query languages.

• Prevalence of RDF/OLM: RDF systems still play an important role in managing and processing graphs.

Our survey also highlights other interesting facts, such as the prevalence of machine learning on graph data, e.g., for clustering vertices, predicting links, and finding influential vertices. We further reviewed user feedback in the mailing lists, bug reports, and feature requests in the source code repositories of 22 software products between January and September of 2017 with two goals: (i) to answer several new questions that the participants’ responses raised; and (ii) to identify more specific challenges in different classes of graph technologies than the ones we could identify.

2. OBJECTIONS

Graph data representing connected entities and their relationships appear in many application domains, most naturally in social networks, the web, the semantic web, road maps, communication networks, bioinformatics, and finance, just to name a few examples. There has been a noticeable increase in the prevalence of work on graph processing both in research and as evidenced by the surge in the number of different commercial and research software for managing and processing graphs. Examples include graph database systems [13, 20, 26, 49, 65, 73, 90], RDF engines [52, 96], linear algebra software [17, 63], visualization software [25, 29], query languages [41, 72, 78], and distributed graph processing systems [30, 34, 48]. In the academic literature, a large number of publications that study numerous topics related to graph processing regularly appear across a wide spectrum of research venues.

Despite their prevalent use, there is little research on how graph data are actually used in practice and the major challenges facing users of graph data, both in industry and in research. In April 2017, we conducted an online survey across 89 users of 22 different software products, with the goal of answering 4 high-level questions:

(i) What types of graph data do users have?
(ii) What computations do users run on their graphs?
(iii) Which software do users use to perform their computations?
(iv) What are the major challenges users face when processing their graphs?
1 Introduction

Graph data representing connected entities and their relationships appear in many application domains, most naturally in social networks, the web, the semantic web, road maps, communication networks, biology, and finance, just to name a few examples. There has been a noticeable increase in the prevalence of work on graph processing both in research and in practice, evidenced by the surge in the number of different commercial and research software for managing and processing graphs. Examples include graph database systems [3, 8, 14, 35, 48, 53], RDF engines [36, 64, 67], linear algebra software [6, 46]; visualization software [13, 16], query languages [28, 80]. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Despite their prevalence, there is little research on how graph data are actually used in practice and the major challenges facing users of graph data, both in industry and in research. In April 2017, we conducted an online survey across 89 users of 22 different software products between January and September of 2017 with the goal of answering four high-level questions:

(i) What kind of graph data do users have?
(ii) What computations do users run on their graphs?
(iii) Which software do users use to perform their computations?
(iv) What are the major challenges users face when processing their graphs?
Graph Usage Study

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, M. Tamer Özsu
David R. Chetton School of Computer Science
University of Waterloo
{s3sahu,amine.mhedhbi,semih.salihoglu,jimmylin,tamer.ozsu}@uwaterloo.ca

Objectives

1. What kind of graph data, computations, software, and major challenges real users have in practice?
2. What types of graph data, computations, software, and major challenges researchers target in publications?

Major Findings

1. Graphs are everywhere!
2. Graphs are very large!
3. ML on graphs is very popular (> 85% of respondents have ML workloads)!
4. Scalability is the most pressing challenge (followed by visualization & query languages)!
5. Relational DBMSs still play an important role!
One particular type – streaming graphs

<table>
<thead>
<tr>
<th>Streaming aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Unbounded data ⇒ non-blocking algorithms &amp; operators (one-pass)</td>
</tr>
<tr>
<td>▶ Usually at high speed ⇒ real-time constraints</td>
</tr>
</tbody>
</table>
### Streaming aspects
- Unbounded data $\Rightarrow$ non-blocking algorithms & operators (one-pass)
- Usually at high speed $\Rightarrow$ real-time constraints

### Graph aspects
- (Typically) edges streaming
- Graph “emerges”

---

One particular type – streaming graphs
### Streaming aspects

- Unbounded data $\Rightarrow$ non-blocking algorithms & operators (one-pass)
- Usually at high speed $\Rightarrow$ real-time constraints

### Graph aspects

- (Typically) edges streaming
- Graph “emerges”

### Use case

**Alibaba**

- 500M active users, 2B catalog items
- 320K transactions/second (at peak)
- Need to process PB data in real-time in hours
Streaming Data Processing
Streaming Graph Processing
S-graffito Project
Concluding Remarks
Streaming Data Processing
Stream Systems

Inputs

One or more sources generate data continuously, in real time, and in fixed order (by timestamp)

- Sensor networks – weather monitoring, road traffic monitoring
- Web data – financial trading, news/sports tickers
- Scientific data – experiments in particle physics
- Transaction logs – point-of-sale purchases
- Network traffic analysis – IP packet headers
# Stream Systems

## Inputs

One or more sources generate data continuously, in real time, and in fixed order (by timestamp)

- Sensor networks – weather monitoring, road traffic monitoring
- Web data – financial trading, news/sports tickers
- Scientific data – experiments in particle physics
- Transaction logs – point-of-sale purchases
- Network traffic analysis – IP packet headers

## Outputs

Want to collect and process data in real-time; up-to-date answers generated continuously or periodically

- Environment monitoring
- Location monitoring
- Correlations across stock prices
- Denial-of-service attack detection
Traditional DBMS:

- Transient query
- Persistent data
- One-time result
DBMS versus DSS

Traditional DBMS:

- Transient query
- Persistent data
- One-time result

Data Stream System (DSS):

- Transient data
- Persistent queries
- Continuous results
DBMS versus DSS

Traditional DBMS:
- Transient query
- Persistent data
- One-time result

Data Stream System (DSS):
- Transient data
- Persistent queries
- Continuous results

Other differences of DSS:
- Push-based (data-driven)
- Persistent queries
- Unbounded stream; query execution as data arrives at the system – one look
- System conditions may not be stable – arrival rates fluctuate, workload may change

One-time result
Continuous results
Old vs New

- **Older systems: Data Stream Management Systems (DSMS)** [Golab and Özsu, 2010]
  - Provide the functionalities of a typical DBMS
  - Examples: STREAM, Gigascope, TelegraphCQ, Aurora, Borealis
  - Mostly single machine systems
  - From early 2000s to late 2000s

- **Newer systems: Data Stream Processing Systems (DSPS)**
  - May not have full DBMS functionality
  - Examples: Apache Storm, Heron, Spark Streaming, Flink, MillWheel, TimeStream
  - Almost all are scale-out
  - From mid-2010s
DSMS System Architecture

- Input Monitor
- Working Storage
- Summary Storage
- Static Storage
- Query Repository
- Query Processor
- Output Buffer

- Updates to Static Data
- User Queries
Stream Data Model

Append-only sequence of timestamped items that arrive in some order.

\(\langle\text{timestamp}, \text{payload}\rangle\)

What is the payload?
- Relational tuple
- Revision tuple
- Graph edge
- Sequence of events (as in publish/subscribe systems)
- Sequence of sets (or bags) of elements with each set storing elements that have arrived during the same unit of time
Streaming Graph Processing
Streaming Graphs

Time

$\tau_0 \tau_1 \tau_2 \tau_3 \tau_4 \tau_5 \tau_6 \tau_7 \tau_8 \tau_9 \tau_{10} \tau_{11} \tau_{12} \tau_{13}$

DEBS 2020
Streaming Graphs

A

B

\[ t_0 \quad t_1 \quad t_2 \quad t_3 \quad t_4 \quad t_5 \quad t_6 \quad t_7 \quad t_8 \quad t_9 \quad t_{10} \quad t_{11} \quad t_{12} \quad t_{13} \]

\( t_1 \)
Streaming Graphs

\[ t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{11} \rightarrow t_{12} \rightarrow t_{13} \]

At time \( t_1 \), there is a change in the graph.
Streaming Graphs
Streaming Graphs
Streaming Graphs

![Diagram of streaming graphs showing nodes and edges at different time points]

Time:
- $t_0$, $t_1$, $t_2$, $t_3$, $t_4$, $t_5$, $t_6$, $t_7$, $t_8$, $t_9$, $t_{10}$, $t_{11}$, $t_{12}$, $t_{13}$

Nodes:
- A, B, C, D

Edges:
- At $t_1$, edge from B to C
- At $t_4$, edge from B to C
- At $t_5$, edge from B to C, edge from A to D

DEBS 2020
Streaming Graphs

A  B
  ↓   ↓
  A   B

C  D
  ↓   ↓
  C   D

E  F
  ↓   ↓
  E   F

G  H
  ↓   ↓
  G   H

I  J
  ↓   ↓
  I   J

K  L
  ↓   ↓
  K   L

M  N
  ↓   ↓
  M   N

O  P
  ↓   ↓
  O   P

Q  R
  ↓   ↓
  Q   R

S  T
  ↓   ↓
  S   T

U  V
  ↓   ↓
  U   V

W  X
  ↓   ↓
  W   X

Y  Z
  ↓   ↓
  Y   Z

DEBS 2020
Streaming Graphs

A diagram illustrating the evolution of a graph over time, with edges and nodes labeled accordingly.
Streaming Graphs

DEBS 2020
Streaming Graphs

- Combines two difficult problems: streaming + graphs
- Unbounded $\Rightarrow$ don’t see entire graph
- Streaming rates can be very high

DEBS 2020
Streaming Graph Computation Models

- **Continuous**
  - Process each edge as it comes $\Rightarrow$ for simple transactional operations
  - Requires linear space $\Rightarrow$ unrealistic
    - Many graph problems are not solvable [McGregor, 2014]
  - Semi-streaming model $\Rightarrow$ sublinear space [Feigenbaum et al., 2005]
    - Sufficient to store vertices but not edges (typically $|V| \ll |E|$) $\Rightarrow$ dynamic graph model
    - Approximation for many graph algorithms exist
Streaming Graph Computation Models

- **Continuous**
  - Process each edge as it comes ⇒ for simple transactional operations
  - Requires linear space ⇒ unrealistic
    - Many graph problems are not solvable [McGregor, 2014]
  - Semi-streaming model ⇒ sublinear space [Feigenbaum et al., 2005]
    - Sufficient to store vertices but not edges (typically $|V| \ll |E|$) ⇒ dynamic graph model
    - Approximation for many graph algorithms exist

- **Windowed**
  - Use windows to batch edges
  - For more complex queries
Continuous Computation

Query: Vertices reachable from vertex A

<table>
<thead>
<tr>
<th>Time</th>
<th>Incoming edge</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$\langle A, B \rangle$</td>
<td>${ B }$</td>
</tr>
<tr>
<td>$t_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_4$</td>
<td>$\langle B, C \rangle$</td>
<td>${ B, C }$</td>
</tr>
<tr>
<td>$t_5$</td>
<td>$\langle A, D \rangle, \langle D, C \rangle$</td>
<td>${ B, C, D }$</td>
</tr>
<tr>
<td>$t_6$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_7$</td>
<td>$\langle C, F \rangle, \langle D, F \rangle$</td>
<td>${ B, C, D, F }$</td>
</tr>
<tr>
<td>$t_8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_9$</td>
<td>$\langle D, E \rangle, \langle A, E \rangle, \langle B, E \rangle, \langle E, F \rangle$</td>
<td>${ B, C, D, F, E }$</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Windowed Computation

**Query:** Vertices reachable from vertex A

![Graphs showing vertex A and its reachable vertices at various times](image)

(Window size=5)

<table>
<thead>
<tr>
<th>Time</th>
<th>Incoming edge</th>
<th>Expired edges</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$\langle A, B \rangle$</td>
<td></td>
<td>${B}$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$\langle B, C \rangle$</td>
<td></td>
<td>${B, C}$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$\langle A, D \rangle, \langle D, C \rangle$</td>
<td></td>
<td>${B, C, D}$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>$\langle A, B \rangle$</td>
<td></td>
<td>${B, C, D}$</td>
</tr>
<tr>
<td>$t_5$</td>
<td>$\langle C, F \rangle, \langle D, F \rangle$</td>
<td>$\langle A, B \rangle$</td>
<td>${C, D, F}$</td>
</tr>
<tr>
<td>$t_6$</td>
<td>$\langle D, E \rangle, \langle A, E \rangle, \langle B, E \rangle, \langle E, F \rangle$</td>
<td></td>
<td>${C, D, F, E}$</td>
</tr>
</tbody>
</table>
Querying Graph Streams

- **Graph query functionalities**
  - Subgraph matching queries & reachability (path) queries
  - Doing these in the streaming context
  - This is querying beyond simple transactional operations on an incoming edge
    - Edge represents a user purchasing an item \(\rightarrow\) do some operation
    - Edge represents events in news \(\rightarrow\) send an alert

- **Subgraph pattern matching under stream of updates**
  - Windowed join processing
  - Graphflow [Kankanamge et al., 2017], TurboFlux [Kim et al., 2018]
  - These are not designed to deal with unboundedness of the data graph

- **Path queries under stream of updates**
Analytics on Graph Streams

- Many use cases
  - Recommender systems
  - Fraud detection [Qiu et al., 2018]
  - ...

- Existing relevant work
  - Snapshot-based systems
    - Aspen [Dhulipala et al., 2019], STINGER [Ediger et al., 2012]
    - Consistent graph views across updates
  - Snapshot + Incremental Computations
    - Kineograph [Cheng et al., 2012], GraPu [Sheng et al., 2018], GraphIn [Sengupta et al., 2016]
    - GraphBolt [Mariappan and Vora, 2019]
    - Identify and re-process subgraphs that are effected by updates
  - Designed to handle high velocity updates
  - Cannot handle unbounded streams
    - Similar to dynamic graph processing solutions
S-graffito Project
S-Graffito project

Processing of transactional (OLTP) and analytical (OLAP) queries on high streaming rate, very large graphs.
S-Graffito project

Processing of transactional (OLTP) and analytical (OLAP) queries on high streaming rate, very large graphs.
A property graph is an attributed graph $G = (V, E, \Sigma, \psi, \phi, K, \mathcal{P})$ where $V$ is a set of vertices, $E$ is a set of edges, $\psi : E \rightarrow V \times V$ is a function that maps each edge to an ordered pair of vertices, $\Sigma$ is a set of labels and $\phi$ is a labelling function, $\phi : (V \cup E) \rightarrow \Sigma$, $K$ is a set of property keys, $\mathcal{P}$ is a set of values, and $\nu : (V \cup E) \times K \rightarrow \mathcal{P}$ is a partial function assigning values for properties to objects.
Arrivals are Streaming Graph Tuples

A streaming graph tuple (sgt) is a streaming tuple where is a pair \((\tau, p)\) where \(\tau\) is the event (application) timestamp of the tuple assigned by the data source, \(p\) defines the payload of the tuple that indicates an edge \(e \in E\) or a vertex \(v \in V\) of the property graph \(G\), and an operation \(op \in \{\text{insert, delete, update}\}\) that defines the type of the tuple.
Time-based Window & Snapshot

<table>
<thead>
<tr>
<th>τ</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>(A, B), insert</td>
</tr>
<tr>
<td>τ₄</td>
<td>(B, C), insert</td>
</tr>
<tr>
<td>τ₅</td>
<td>(A, D), insert</td>
</tr>
<tr>
<td>τ₅</td>
<td>(D, C), insert</td>
</tr>
<tr>
<td>τ₇</td>
<td>(C, F), insert</td>
</tr>
<tr>
<td>τ₇</td>
<td>(D, F), insert</td>
</tr>
<tr>
<td>τ₉</td>
<td>(B, E), insert</td>
</tr>
<tr>
<td>τ₉</td>
<td>(E, E), insert</td>
</tr>
<tr>
<td>τ₉</td>
<td>(E, F), insert</td>
</tr>
<tr>
<td>τ₁₂</td>
<td>(E, F), delete</td>
</tr>
</tbody>
</table>

\[ W(τ₅ − τ₁₀) \]

Time-based Window

A *time-based window* \( W \) over a streaming graph \( S \) is a time interval \((W^b, W^e]\) where \( W^b \) and \( W^e \) are the beginning and end times of window \( W \) and \( W^e − W^b = |W| \). The window contents \( W(c) \) is the multiset of sgts where the timestamp \( τ_i \) of each record \( t_i \) is in the window interval, i.e., \( W(c) = \{ t_i \mid W^b < τ_i ≤ W^e \} \). When it is clear from context, \( W \) is used interchangeably to refer to window interval or its contents.
### Time-based Window & Snapshot

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>((A, B), \text{insert})</td>
</tr>
<tr>
<td>( \tau_4 )</td>
<td>((B, C), \text{insert})</td>
</tr>
<tr>
<td>( \tau_5 )</td>
<td>((A, D), \text{insert})</td>
</tr>
<tr>
<td>( \tau_5 )</td>
<td>((D, C), \text{insert})</td>
</tr>
<tr>
<td>( \tau_7 )</td>
<td>((C, F), \text{insert})</td>
</tr>
<tr>
<td>( \tau_7 )</td>
<td>((D, F), \text{insert})</td>
</tr>
<tr>
<td>( \tau_9 )</td>
<td>((B, E), \text{insert})</td>
</tr>
<tr>
<td>( \tau_9 )</td>
<td>((E, E), \text{insert})</td>
</tr>
<tr>
<td>( \tau_9 )</td>
<td>((E, F), \text{insert})</td>
</tr>
<tr>
<td>( \tau_{12} )</td>
<td>((E, F), \text{delete})</td>
</tr>
</tbody>
</table>

### Time-based Window

A *time-based window* \( W \) over a streaming graph \( S \) is a time interval \((W^b, W^e]\) where \( W^b \) and \( W^e \) are the beginning and end times of window \( W \) and \( W^e - W^b = |W| \). The window contents \( W(c) \) is the multiset of sgts where the timestamp \( \tau_i \) of each record \( t_i \) is in the window interval, i.e., \( W(c) = \{ t_i \mid W^b < \tau_i \leq W^e \} \). When it is clear from context, \( W \) is used interchangeably to refer to window interval or its contents.

### Streaming Graph Snapshot

A *streaming graph snapshot* \( G_{W, \tau} \) is the graph formed by the tuples in the time-based window \( W \) at time \( \tau \).
S-graffito Project

Streaming Graph Querying

Anil Pacaci
Streaming Graph Querying Objectives

Persistent query processing over streaming graphs

1. Path navigation queries
   - Non-blocking operators for path queries
   - Regular path queries (RPQ)
     - Regular expressions that are matched against directed, labelled paths

2. A query subsystem for persistent graph queries over streaming graphs
   - Combining graph patterns & path navigation
   - Treating paths as first-class citizens

3. Querying streaming graphs with data
   - Attribute-based predicates for property graphs
**Persistent RPQ Evaluation**

- **Design space for persistent RPQ algorithms**

<table>
<thead>
<tr>
<th>Path semantics</th>
<th>Result semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Append-only</td>
<td>Append-only</td>
</tr>
<tr>
<td></td>
<td>Explicit delete</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Append-only</td>
<td>Explicit delete</td>
</tr>
</tbody>
</table>

- **Path semantics**
  - Simple paths (no repeating vertex): navigation on road networks
  - Arbitrary paths: reachability on communication networks
  - Result semantics & stream types
    - Append-only streams with fast insertions
    - Support for explicit deletions

\[ Q_1 = (\text{follows} \cdot \text{mentions}) + P_1 (\text{follows} \cdot \text{mentions}) + xyuzw \]

- **DEBS 2020**
Persistent RPQ Evaluation

- Design space for persistent RPQ algorithms

<table>
<thead>
<tr>
<th>Path semantics</th>
<th>Result semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Append-only</td>
<td>Append-only</td>
</tr>
<tr>
<td></td>
<td>Explicit delete</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Append-only</td>
<td>Explicit delete</td>
</tr>
</tbody>
</table>

- Path semantics used in practice
  - Simple paths (no repeating vertex): navigation on road networks

\[ Q_1 = (\text{follows} \cdot \text{mentions})^+ \]
Persistent RPQ Evaluation

- Design space for persistent RPQ algorithms

<table>
<thead>
<tr>
<th>Path semantics</th>
<th>Result semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Append-only</td>
<td>Append-only</td>
</tr>
<tr>
<td>Explicit delete</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>Explicit delete</td>
</tr>
</tbody>
</table>

- Path semantics used in practice
  - Simple paths (*no repeating vertex*): navigation on road networks

\[
Q_1 = (\text{follows} \cdot \text{mentions})^+
\]

```
follows  mentions  mentions  mentions
\downarrow  \downarrow  \downarrow  \downarrow
x          y          u          v
```

\[
Q_1 = (\text{follows} \cdot \text{mentions})^+ 
\]

```
follows  mentions  mentions  mentions
\downarrow  \downarrow  \downarrow  \downarrow
x          y          u          v
```

\[
(follows \cdot mentions)^+ 
\]

```
P_1 \rightarrow P_2 
```

DEBS 2020
Persistent RPQ Evaluation

- Design space for persistent RPQ algorithms

<table>
<thead>
<tr>
<th>Path semantics</th>
<th>Result semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Append-only</td>
<td>Append-only</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Append-only</td>
<td>Explicit delete</td>
</tr>
<tr>
<td>Explicit delete</td>
<td>Append-only</td>
</tr>
</tbody>
</table>

- Path semantics used in practice
  - Simple paths (no repeating vertex): navigation on road networks
  - Arbitrary paths: reachability on communication networks

$$Q_1 = (\text{follows} \cdot \text{mentions})^+$$

Simple paths

Arbitrary paths
Persistent RPQ Evaluation

- Design space for persistent RPQ algorithms

<table>
<thead>
<tr>
<th>Path semantics</th>
<th>Result semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Simple</td>
</tr>
<tr>
<td>Append-only</td>
<td>Append-only</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Append-only</td>
<td>Explicit delete</td>
</tr>
<tr>
<td></td>
<td>Explicit delete</td>
</tr>
</tbody>
</table>

- Path semantics used in practice
  - Simple paths (no repeating vertex): navigation on road networks
  - Arbitrary paths: reachability on communication networks

- Result semantics & stream types
  - Append-only streams with fast insertions
  - Support for explicit deletions
Beyond Path Navigation

Combining subgraph matching & path navigation
Combining subgraph matching & path navigation
Beyond Path Navigation

Combining subgraph matching & path navigation

Unions of Conjunctive RPQs (UCRPQ)
- SPARQL v1.1, Cypher9 (limited form), Oracle PGQL [van Rest et al., 2016]
Beyond Path Navigation

Combining subgraph matching & path navigation

Unions of Conjunctive RPQs (UCRPQ)
- SPARQL v1.1, Cypher9 (limited form), Oracle PGQL [van Rest et al., 2016]
- No algebraic closure

Recursion over edges
Beyond Path Navigation

Combining subgraph matching & path navigation

\[
\begin{align*}
  u_1 & \rightarrow c_1 & \text{worksAt} & \rightarrow u_2 \\
  (\text{follows} \cdot \text{mentions})^+ & \rightarrow u_3 & \rightarrow \cdots & \rightarrow u_n \rightarrow c_n & \rightarrow \text{worksAt} & \rightarrow u_{n+1}
\end{align*}
\]

Recursion over a graph pattern

- Unions of Conjunctive RPQs (UCRPQ)
  - SPARQL v1.1, Cypher9 (limited form), Oracle PGQL [van Rest et al., 2016]

- No algebraic closure
Beyond Path Navigation

Combining subgraph matching & path navigation

Unions of Conjunctive RPQs (UCRPQ)
- SPARQL v1.1, Cypher9 (limited form), Oracle PGQL [van Rest et al., 2016]
- No algebraic closure

Regular Queries (RQ) [Reutter et al., 2017]
- A subset of Datalog with algebraic closure
- Computationally well-behaved
- The basis of G-CORE [Angles et al., 2018]
Beyond Path Navigation

Combining subgraph matching & path navigation

Our work

- An algebra for RQ on streaming graphs
- Concrete implementation of this algebra

- No algebraic closure

The basis of G-CORE [Angles et al., 2018]
Paths as First-class Citizens

So far we focused on *existence* of a path, i.e., reachability
Paths as First-class Citizens

So far we focused on existence of a path, i.e., reachability

Where Alice ∈ p = \{u_1, \cdots, u_{n+1}\}
Paths as First-class Citizens

So far we focused on \textit{existence} of a path, i.e., reachability

\begin{itemize}
  \item Ability to store, return and compare paths
\end{itemize}

where Alice $\in p = \{u_1, \ldots, u_{n+1}\}$
 Paths as First-class Citizens

So far we focused on *existence* of a path, i.e., reachability

\[ u_1, u_2, \ldots, u_n, u_{n+1} \]

where Alice ∈ \( p = \{ u_1, \cdots, u_{n+1} \} \)

- Ability to store, return and compare paths
- Enumerate all paths
  - High complexity, FPT for certain classes [Martens and Trautner, 2019]

Ability to store, return and compare paths
Enumerate all paths
High complexity, FPT for certain classes [Martens and Trautner, 2019]
So far we focused on *existence* of a path, i.e., reachability

where Alice ∈ \( p = \{u_1, \cdots, u_{n+1}\} \)

- Ability to store, return and compare paths
- Enumerate all paths
  - High complexity, FPT for certain classes [Martens and Trautner, 2019]
- Structural restrictions on path operations
  - Length predicates [Barceló et al., 2012]
  - Closed semi-ring aggregates [Cruz and Norvell, 1989]
Paths as First-class Citizens

So far we focused on existence of a path, i.e., reachability

\[ u_1 \xrightarrow{\text{worksAt}} c_1 \xrightarrow{\text{worksAt}} u_2 \xrightarrow{(\text{follows} \cdot \text{mentions})^+} u_n \xrightarrow{\text{worksAt}} c_n \xrightarrow{\text{worksAt}} u_{n+1} \]

where \( Alice \in \{ u_1, \ldots, u_n \} \)

Our work

▶ Data model and query language that treats paths as first-class citizens
▶ Streaming, sliding-window algorithms for common path operations

- Structural restrictions on path operations
  - Length predicates [Barceló et al., 2012]
  - Closed semi-ring aggregates [Cruz and Norvell, 1989]
Querying Graphs with Data

Real-world graphs have data, so as queries
Querying Graphs with Data

Real-world graphs have data, so as queries

\[
\begin{align*}
\text{worksAt} & \quad \text{u} \quad \text{worksAt} \\
\text{(follows \cdot mentions)}^+ & \\
\text{u}_1 & \quad \text{u}_2 & \quad \text{u}_n & \quad \text{u}_{n+1} \\
\text{city} = \text{u}_{n+1}.\text{city}
\end{align*}
\]
Querying Graphs with Data

Real-world graphs have data, so as queries

\[ u_1 \text{worksAt} c_1 \text{worksAt} u_2 \text{(follows \cdot mentions)}^+ \xrightarrow{} u_n \text{worksAt} c_n \text{worksAt} u_{n+1} \text{(follows \cdot mentions)}^+ \]

\[ u_1.\text{city} = u_{n+1}.\text{city} \]

- Support for attribute-based predicates on property graphs
- Regular Property Graph Queries (RPGQ) [Bonifati et al., 2018]
  - RQ on property graphs
- Non-trivial query planning [Mulder et al., 2020]
  - Structure-based vs structure&attribute-based planning
  - Up to 30× performance differences
Querying Graphs with Data

Real-world graphs have data, so as queries

\[ u_1 \xrightarrow{\text{worksAt}} c_1 \xrightarrow{\text{worksAt}} u_2 \]
\[ u_n \xrightarrow{\text{worksAt}} c_n \xrightarrow{\text{worksAt}} u_{n+1} \]

(follows \cdot mentions)⁺

Our work

- Support for property graphs & attribute-based predicates
- Non-blocking implementation of RPGQ for streaming graphs
- Non-trivial query planning [Mulder et al., 2020]
  - Structure-based vs structure&attribute-based planning
  - Up to 30× performance differences
S-graffito Project

Streaming Graph Analytics

Aida Sheshbolouki
Streaming Graph Analytics Objectives

Building a generic analytics engine based on window semantics and vertex embeddings

1. Exploratory analysis of real-world streaming graphs
2. Representation learning over streaming graphs
3. Prediction-based analytics over streaming graphs
Exploratory Analysis of Real-world Streaming Graphs

Identifying streaming graph patterns

The emergence patterns of edges ⇒ attachment rules
The emergence patterns of key subgraphs ⇒ subgraph densification power laws

Modeling streaming graphs

Synthetic graph model that preserves realistic patterns
For pinpointing the performance of processing algorithms

Time $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{11} \rightarrow t_{12} \rightarrow t_{13} \rightarrow \ldots$

Merging components

A giant growing component
Robust against random edge removals
Not robust against targeted removals
Robust against any edge removal
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
1 Identifying streaming graph patterns
   - The emergence patterns of edges ⇒ attachment rules
     - “Rich-get-richer” conjecture
Identifying streaming graph patterns

- The emergence patterns of edges \( \Rightarrow \) attachment rules
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
   - The emergence patterns of edges $\Rightarrow$ attachment rules
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
   - The emergence patterns of edges ⇒ attachment rules
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
   - The emergence patterns of edges ⇒ attachment rules
   - The emergence patterns of key subgraphs ⇒ subgraph densification power laws
     - The number of 2,2-bicliques (butterflies) follows a power law function of the number of the number of edges
     - Bursty butterfly densification – Butterflies emerge in a bursty fashion due to the existing hubs contribution
   - sGrapp: Streaming Graph Approximation Framework for Butterfly Counting
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
   - The emergence patterns of edges \( \Rightarrow \) attachment rules
   - The emergence patterns of key subgraphs \( \Rightarrow \) subgraph densification power laws
   - The connectivity and robustness of the graph snapshots

Merging components

A giant growing component

Robust against random edge removals
Not robust against targeted removals

Robust against any edge removal
Exploratory Analysis of Real-world Streaming Graphs

1. Identifying streaming graph patterns
   - The emergence patterns of edges ⇒ attachment rules
   - The emergence patterns of key subgraphs ⇒ subgraph densification power laws
   - The connectivity and robustness of the graph snapshots

2. Modeling streaming graphs
   - Synthetic graph model that preserves realistic patterns
   - For pinpointing the performance of processing algorithms
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
2. Addressing structural evolutions
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
2. Addressing structural evolutions
3. Addressing streaming property graphs
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
2. Addressing structural evolutions
3. Addressing streaming property graphs
4. Addressing data sparsity
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
2. Addressing structural evolutions
3. Addressing streaming property graphs
4. Addressing data sparsity
5. Model optimizations
   - Heterogeneous embedding
   - Dynamic graph convolutions
   - Parameter training
Representation Learning over Streaming Graphs

Main issue: trade-off between effectiveness and efficiency

1. Unbounded stream management and processing
2. Addressing structural evolutions
3. Addressing streaming property graphs
4. Addressing data sparsity
5. Model optimizations
   - Heterogeneous embedding
   - Dynamic graph convolutions
   - Parameter training

Outcome

An embedding model based on window semantics to incrementally learn the graph evolutions and update the vertex embeddings.
Prediction-based Analytics over Streaming Graphs

1. Efficient windowed analytics
2. Window semantics
3. Graph versatility
4. Accurate predictions
Concluding Remarks
Some Take-home Messages

- Streaming graphs are real and occur in real-life applications
- We have not paid nearly sufficient attention to streaming graph challenges
- Streaming $\neq$ dynamic
  - ... most “streaming” papers are not streaming
- Unboundedness in streams raises real challenges
- Most graph problems are unbounded under edge insert/delete
- The entire field is pretty much open...
  - ... this area is tough and you are not likely to write as many papers
Thank you!

Aida Sheshbolouki
Anil Pacaci
Angela Bonifati
References I


References III


