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
{s3sahu,amine.mhedhbi,semih.salihoglu,jimmylin,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 aimed at understanding: (i) the types of graphs users have; (ii) the graph computations users run; (iii) the types of graph software users use; 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 were able to answer some new questions that were raised by participants' responses and identify specific challenges that users face when using different classes of graph software. The participants' responses and data we obtained revealed surprising facts about graph processing in practice. In particular, real-world graphs represent a very diverse range of entities and are often very large, and scalability and visualization are undeniably the most pressing challenges faced by participants. We hope these findings can guide future research.

PVLDB Reference Format:
Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *PVLDB*, 11(4): xxxx-yyyy, 2018.
DOI: https://doi.org/10.1145/3164135.3164139

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 [38, 64, 67], linear algebra software [6, 46], visualization software [13, 16], query languages [28, 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.

Proceedings of the VLDB Endowment, Vol. 11, No. 4

DOI: https://doi.org/10.1145/3164135.3164139

Despite their prevalence, 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 graph data?

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 ones. This refutes the sometimes heard assumption that large graphs are a problem for only a few large organizations such as Google, Facebook, and Twitter.
- *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 visualization as their second most pressing challenge, tied with challenges in graph query languages.
- *Prevalence of RDBMSes:* Relational databases 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 iden-

tify in participants' responses. For some of the questions in our online survey, we also compared the graph data, computations, and software used by the participants with those studied in academic publications. For this, we reviewed 90 papers from 6 conferences across different academic venues.

In addition to discussing the insights we gained through our study, we discuss several directions about the future of graph processing. We hope our study can inform research about real use cases and important problems in graph processing.

2. METHODOLOGY AND PARTICIPANTS

In this section, we first describe the format of our survey and then how we recruited the participants. Next we describe the demographic information of the participants, including the organizations they come from and their roles in their organizations. Then we describe our methodology of reviewing academic publications. We end this section by describing our methodology for reviewing the user feedback in the mailing lists, bug reports, and feature requests in the source code repositories of the software products.

2.1 Survey Format

The survey was in the format of an online form. All of the questions were optional and participants could skip any number of questions. There were 2 types of questions:

(i) *Multiple Choice*: There were 3 types of multiple choice questions: (a) yes-no questions; (b) questions that allowed only a single choice as a response; and (c) questions that allowed multiple choices as a response. The participants could use an *Other* option when their answers required further explanation or did not match any of the provided choices. We randomized the order of choices in questions about the computations participants run and the challenges they face.

(ii) *Short Answer*: For these questions, the participants entered their responses in a text box.

There were 34 questions grouped into five categories: (i) demographic questions; (ii) graph datasets; (iii) graph and machine learning computations; (iv) graph software; and (v) workload breakdown and major challenges.

2.2 Participant Recruitment

We prepared a list of 22 popular software products for processing graphs (see Table 1) that had public user mailing lists covering 6 types of technologies: graph database systems, RDF engines, distributed graph processing systems (DGPSes), graph libraries to run and compose graph algorithms, visualization software, and graph query languages.[1] Our goal was to be as comprehensive as possible in recruiting participants from the users of different graph technologies. However, we acknowledge that this list is incomplete and does not cover all of the graph software used in practice.

We conducted the survey in April 2017, and used 4 methods to recruit participants from the users of these 22 software products:

- *Mailing Lists*: We posted the survey to the user mailing lists of the software in our list.
- *Private Emails*: Five mailing lists: (i) Neo4j; (ii) OrientDB; (iii) ArangoDB; (iv) JanusGraph; and (v) NetworkX, allowed us to send private emails to the users. We sent private emails to 171 users who were active on these mailing lists between February and April of 2017.

[1]The linear algebra software we considered, e.g., BLAS [6] and MATLAB [46], either did not have a public mailing list or their lists were inactive.

Table 1: Software products used for recruiting participants and the number of active mailing list users from February to April 2017.

Technology	Software	# Users	
Graph Database System	ArrangoDB [3]	40	233
	Caley [8]	14	
	DGraph [14]	33	
	JanusGraph [35]	32	
	Neo4j [48]	69	
	OrientDB [53]	45	
RDF Engine	Apache Jena [38]	87	115
	Sparksee [64]	5	
	Virtuoso [67]	23	
Distributed Graph Processing Engine	Apache Flink (Gelly) [17]	24	39
	Apache Giraph [21]	8	
	Apache Spark (GraphX) [27]	7	
Query Language	Gremlin [28]	82	82
Graph Library	Graph for Scala [22]	4	97
	GraphStream [24]	8	
	Graphtool [25]	28	
	NetworKit [50]	10	
	NetworkX [51]	27	
	SNAP [62]	20	
Graph Visualization	Cytoscape [13]	93	116
	Elasticsearch (X-Pack Graph) [16]	23	
Graph Representation	Conceptual Graphs [11]	6	6

- *Slack Channels*: Two of the software products on our list, Neo4j and Cayley, had Slack channels for their users. We posted the survey to these channels.
- *Twitter*: A week after posting our survey to the mailing lists and Slack channels and sending private emails, we posted a tweet with a link to our survey to 7 of the 22 software products that had an official Twitter account. Only Neo4j retweeted our tweet.

Participants that we recruited through different methods shared the same online link and we could not tell the number of participants recruited from each method. In particular, we suspect that there were more users from graph database systems mainly because their mailing lists contained more active users, as can be seen in Table 1. Moreover, 4 of the 5 mailing lists that allowed us to send private emails and the Slack and Twitter channels belonged to graph database systems. We note that after posting the survey on Twitter, we received 12 responses.

In the end, there were 89 participants. Below, we give an overview of the organizations these participants work in and the role of the participants in their organizations.

Field of Organizations: We asked the participants which field they work in. Participants could select multiple options. Table 2 shows the 12 choices and participants' responses. In the table, "R" and "P" indicate researchers and practitioners (defined momentarily), respectively. In addition to the given choices, using the *Other* option, participants indicated 5 other fields: education, energy market, games and entertainment, investigations and audits, and grassland management. In total, participants indicated 17 different fields, demonstrating that graphs are being used in a wide variety of fields. Throughout the survey, we group the participants into 2 categories:

- *Researchers* are the 36 participants who indicated at least one of their fields as research in academia or research in an industry lab. Some of these participants further selected other choices as their fields, the most popular of which were information and technology, government, defense and space, and health care.

Table 2: The participants' fields of work.

Field	Total	R	P
Information & Technology	48	12	36
Research in Academia	31	31	0
Finance	12	2	10
Research in Industry Lab	11	11	0
Government	7	3	4
Healthcare	5	3	2
Defence & Space	4	3	1
Pharmaceutical	3	0	3
Retail & E-Commerce	3	0	3
Transportation	2	0	2
Telecommunications	1	1	0
Insurance	0	0	0
Other	5	2	3

Table 3: Size of the participants' organizations.

Size	Total	R	P
1 - 10	27	17	10
10 - 100	23	6	17
100 - 1000	14	4	10
1000 - 10000	6	4	2
>10000	15	4	11

- *Practitioners* are the remaining 53 participants who did not select research in academia or an industry lab. The top two fields of practitioners were information and technology and finance, indicated by 36 and 10 people, respectively.

In the remainder of this paper, we will explicitly indicate when the responses of the researchers and practitioners to our survey questions differ significantly. In the absence of an explicit comparison, readers can assume that both groups' responses were similar.

Size of Organizations: Table 3 shows the sizes of the organizations that the participants work in, which ranged from very small organizations with less than 10 employees to very large ones with more than 10,000 employees.

Role at Work: We asked the participants their roles in their organizations and gave them the following 4 choices: (i) researcher; (ii) engineer; (iii) manager; and (iv) data analyst. Participants could select multiple options. The top 4 roles were engineers, selected by 54, researchers, selected by 48, data analysts, selected by 18, and managers, selected by 16. The other roles participants indicated were architect, devops, and student.

2.3 Review of Academic Publications

In order to compare the graph data, computations, and software academics work on with those that our participants indicated, we surveyed papers in the proceedings of the following conferences: VLDB 2014 [34], KDD 2015 [9], ICML 2016 [4], OSDI 2016 [40], SC 2016 [69], and SOCC 2015 [20]. Our goal in choosing these conferences was to select papers from a variety of venues where researchers working on graph processing publish. Specifically, our list consists of venues in databases, data mining, machine learning, operating systems, high performance computing, and cloud computing. For each paper in these proceedings, we first selected the ones that directly studied a graph computation or were developing graph processing software. We omitted papers that were not primarily focused on graph processing, even if they used a graph algorithm as a subroutine to solve a problem. For example, we omitted a paper studying a string matching algorithm that uses a graph algorithm as a subroutine. In the end, we selected 90 papers.

For each of the 90 papers, we identified: (i) the graph datasets used in experiments; (ii) the graph and machine learning computations that appeared in the paper; and (iii) the graph software used in the paper. In our survey, we asked users questions about which graph and machine learning computations they perform. The choices we provided in these questions came from the computations we identified in these publications (see Sections 4.1 and 4.2 and Appendices A and B for details).

2.4 Review of Emails and Code Repositories

To answer some questions that participants' responses raised and to identify more specific challenges users face than the ones we identified from participants' responses, we reviewed emails in the mailing lists of the 22 software products between January and September of 2017. In addition, 20 of these 22 software products had open source code repositories. We reviewed the bug reports and feature requests (*issues* henceforth) in these repositories between January and September of 2017. We also reviewed the repositories of 2 popular graph visualization tools: Gephi [19] and Graphviz [26]. For emails and issues before January 2017, we performed a targeted keyword search to find more instances of the challenges we identified in the January-September 2017 review.

In total, we reviewed over 6000 emails and issues. The overwhelming majority of the emails and issues were routine engineering tasks, such as users asking how to write a query or developers asking for integration with another software. The number of emails and issues that were useful for identifying challenges were 311 in total. We review these challenges in Section 6.2. Table 20 in the appendix shows the exact number of emails and issues we reviewed for each product, and the number of commits in it's code repository to give readers a sense of how active these repositories are.

3. GRAPH DATASETS

In this section, we describe the properties of the graph datasets that the participants work with.

3.1 Real-World Entities Represented

We asked the participants about the real-world entities that their graphs represent. We provided them with 4 choices and the participants could select multiple of them.

(i) *Humans*: e.g., employees, customers, and their interactions.
(ii) *Non-Human Entities*: e.g., products, transactions, or web pages.
(iii) *RDF or Semantic Web*.
(iv) *Scientific*: e.g., chemical molecules or biological proteins.

For the participants who selected non-human entities, we followed up with a short-answer question asking them to describe what these are. Participants indicated 52 different kinds of non-human entities, which we group into 7 broad categories.[2] We indicate the acronyms we use in our tables for each category in parentheses:

(i) *Products* (NH-P): e.g., products, orders, and transactions.
(ii) *Business and Financial Data* (NH-B): e.g., business assets, funds, or bitcoin transfers.
(iii) *Web Data* (NH-W).
(iv) *Geographic Maps* (NH-G): e.g., roads, bicycle sharing stations, or scenic spots.
(v) *Digital Data* (NH-D): e.g., files and folders or videos and captions.

[2]Six entities that the participants mentioned did not fall under any of our 7 categories, which we list for completeness: call records, computers, cars, houses, time slots, and specialties.

Table 4: Real-world entities represented by the participants' graphs and studied in publications. Legend for non-human entities: *Products* (NH-P), *Business and Financial Data* (NH-B), *Web Data* (NH-W), *Geographic Maps* (NH-G), *Digital Data* (NH-D), *Infrastructure Networks* (NH-I), *Knowledge and Textual Data* (NH-K).

Category	Human	RDF	Scientific	Non-Human	NH-P	NH-B	NH-W	NH-G	NH-D	NH-I	NH-K
Total	45	23	15	60	13	11	4	7	5	9	11
R	18	11	9	22	1	6	2	4	1	7	6
P	27	12	6	38	12	5	2	3	4	2	5
A	54	8	11	63	2	8	30	11	0	2	3

Table 5: The sizes of the participants' graphs.

(a) Number of vertices.

Vertices	Total	R	P
<10K	22	11	11
10K - 100K	22	9	13
100K - 1M	19	7	12
1M - 10M	17	6	11
10M - 100M	20	10	10
>100M	27	10	17

(b) Number of edges.

Edges	Total	R	P
<10K	23	11	12
10K - 100K	22	9	13
100K - 1M	13	3	10
1M - 10M	9	5	4
10M - 100M	21	8	13
100M - 1B	21	8	13
>1B	20	8	12

(c) Total uncompressed bytes.

Size	Total	R	P
<100MB	23	12	11
100MB - 1GB	19	9	10
1GB - 10GB	25	9	16
10GB - 100GB	17	5	12
100GB - 1TB	20	8	12
>1 TB	17	5	12

Table 6: Sizes of organization that have graphs with >1B edges.

Size	1 - 10	10 - 100	100 - 1000	>10000
#	4	4	7	4

(vi) *Infrastructure Networks* (NH-I): e.g., oil wells and pipes or wireless sensor networks.

(vii) *Knowledge and Textual Data* (NH-K): e.g., keywords, lexicon terms, words, and definitions.

Table 4 shows the responses. In the table, the number of academic publications that use each type of graph is listed in the *A* row. We highlight 2 interesting observations:

- *Variety*: Real graphs capture a very wide variety of entities. Readers may be familiar with entities such as social connections, infrastructure networks, and geographic maps. However, many other entities in the participants' graphs may be less natural to think of as graphs. These include malware samples and their relationships, videos and captions, or scenic spots, among others. This lends credence to the cliché that graphs are everywhere.
- *Product Graphs*: Products, orders, and transactions were the most popular non-human entities represented in practitioners' graphs, indicated by 12 practitioners. This contrasts with their relative unpopularity among researchers and academics: only 1 researcher and 2 papers used these graphs. Such product-order-transaction data is traditionally the classic example of enterprise data that perfectly fits the relational data model. It is interesting that enterprises represent similar product data as graphs, possibly because they find value in analyzing connections in such data.

We also note that we expected scientific graphs to be used mainly by researchers. Surprisingly, scientific graphs are prevalent among practitioners as well.

3.2 Size

We asked the participants the number of vertices, number of edges, and total uncompressed size of their graphs. They could select multiple options. Tables 5a, 5b, and 5c show the responses. As shown in the tables, graphs of every size, from very small ones with less than 10K edges to very large ones with more than 1B edges, are prevalent across both researchers and practitioners. We make one interesting observation:

- *The Ubiquity of Very Large Graphs:* A significant number of participants work with very large graphs. Specifically, 20 participants (8 researchers and 12 practitioners) indicated using graphs with more than a billion edges. Moreover, the 20 participants with graphs with more than one billion edges are from organizations with different scales, ranging from very small to very large, as shown in Table 6. This refutes the common assumption that only very large organizations—such as Google [45], Facebook [10], and Twitter [61] that have web and social network data—have very large graphs. Finally, we note that these large graphs represent a variety of entities, including social, scientific, RDF, product, and digital data,[3] indicating that very large graphs appear in a wide range of domains.

One thing that is not clear from our survey is how much larger the participants' graphs are beyond the maximum limits we inquired about (100 million vertices, 1 billion edges, and 1 TB uncompressed data). In order to answer this question, we categorized the graph sizes mentioned in the user emails we reviewed that were beyond these sizes. Focusing on the number of edges, we found 42 users with 1-10B-edge graphs, 17 with 10B-100B-edge graphs, and 7 users processing graphs over 100B edges. Two participants also clarified through an email exchange that their graphs contained 4B and 30B edges. As in our survey results, these large graphs represented a wide range of entities, such as product-order-transaction data, or entities from agriculture and finance. Table 18 in the appendix shows the exact distribution of sizes we identified.

3.3 Other Questions on Graph Datasets

Topology: We asked the participants whether their graphs were: (i) *directed or undirected*; and (ii) *simple graphs or multigraphs*. We clarified that multigraphs are those with possibly multiple edges between two vertices, while simple graphs do not allow multiple edges between two vertices. Tables 7a and 7b show the responses.

Types of Data Stored on Vertices and Edges: We asked the participants whether they stored data on the vertices and edges of their graphs. All participants except 3 indicated that they do. We asked

[3]Some participants selected multiple graph sizes and multiple entities, so we cannot perform a direct match of which graph size corresponds to which entity. The entities we list here are taken from the participants who selected a single graph size and entity, so we can directly match the size of the graph to the entity.

Table 7: The topology and stored data types of the participants' graphs.

(a) Directed vs. Undirected

Topology	Total	R	P
Only Directed	63	23	40
Only Undirected	11	6	5
Both	15	7	8

(b) Simple vs. Multigraphs

Topology	Total	R	P
Only Simple Graphs	26	9	17
Only Multigraphs	50	20	30
Both	13	7	6

(c) Data types stored on vertices and edges.

Type	Vertices			Edges		
	Total	R	P	Total	R	P
String	79	31	48	66	24	42
Numeric	63	23	40	59	23	36
Date/Timestamp	56	19	37	49	18	31
Binary	15	8	7	8	4	4

Table 8: Frequency of changes.

Frequency	Total	R	P
Static	40	21	19
Dynamic	55	22	33
Streaming	18	9	9

the types of data they store and gave them 4 choices: (i) string; (ii) numeric; (iii) date or timestamp; and (iv) binary. Table 7c shows participants' responses. Five participants also indicated storing JSON, lists, and geographic coordinates using the *Other* option.

Dynamism: We asked the participants how frequently the vertices and edges of their graphs change, i.e., are added, deleted, or updated. We provided 3 choices with the following explanations: (i) *static*: there are no or very infrequent changes; (ii) *dynamic*: there are frequent changes, and all changes are stored permanently; and (iii) *streaming*: there are very frequent changes and the participants' software discards some of the graph after some time. Table 8 shows the responses. Surprisingly 18 participants (9 researchers and 9 practitioners) indicated having streaming graphs, which represented at least humans, products, and semantic web data.

4. COMPUTATIONS

In this section, we describe the computations that the participants perform on their graphs.

4.1 Graph Computations

Our goal in this question was to understand what types of graph queries and computations, not including machine learning computations, participants perform on their graphs. We asked a multiple choice question that contained as choices a list of queries and computations followed by a short answer question that asked for computations that may not have appeared in the first question as a choice. In the multiple choice question, instead of asking for a set of ad-hoc queries and computations, we selected a list of graph queries and computations that appeared in the publications of the 6 conferences we reviewed (recall Section 2.3), using our best judgment to categorize similar computations under the same name. We describe our detailed methodology in Appendix A.

Table 9 shows the 13 choices we provided in the multiple choice question, the responses we got, and the number of academic publications that use or study each computation. As shown in the table, all of the 13 computations are used by both researchers and practitioners. Except for two computations, the popularity of these computations is similar among participants' responses and academic publications. The exceptions are neighborhood and reachability queries, which are respectively used by 51 and 27 participants, but studied in only 3 publications. Finding connected components appears to be a very popular and fundamental graph computation—it is the most popular graph computation overall and also among practitioners. We suspect it is a common pre-processing or cleaning step, e.g., to remove singleton vertices, across many tasks.

Table 9: Graph computations performed by the participants and studied in publications.

Computation	Total	R	P	A
Finding Connected Components	55	18	37	12
Neighborhood Queries (e.g., finding 2-degree neighbors of a vertex)	51	19	32	3
Finding Short / Shortest Paths	43	18	25	17
Subgraph Matching (e.g., finding all diamond patterns, SPARQL)	33	14	19	21
Ranking & Centrality Scores (e.g., PageRank, Betweenness Centrality)	32	17	15	22
Aggregations (e.g., counting the number of triangles)	30	10	20	7
Reachability Queries (e.g.,checking if u is reachable from v)	27	7	20	3
Graph Partitioning	25	13	12	5
Node-similarity (e.g., SimRank)	18	7	11	3
Finding Frequent or Densest Subgraphs	11	7	4	2
Computing Minimum Spanning Tree	9	5	4	2
Graph Coloring	7	3	4	3
Diameter Estimation	5	2	3	2

A total of 13 participants answered our follow-up short answer question on other graph queries and computations they run. Example answers include queries to create schemas and graphs, custom bioinformatics algorithms, and finding k-cores in a weighted graph.

4.2 Machine Learning Computations

We next asked participants what kind of machine learning computations they perform on their graphs. Similar to the previous question, these questions were formulated to identify the machine learning computations that appeared in the academic publications we reviewed. We describe our detailed methodology in Appendix B. We asked the following 2 questions:

- *Which machine learning computations do you run on your graphs?* The choices were: clustering, classification, regression (linear or logistic), graphical model inference, collaborative filtering, stochastic gradient descent, and alternating least squares.
- *Which problems that are commonly solved with machine learning do you solve using graphs?* The choices were: community detection, recommendation system, link prediction, and influence maximization.[4]

Tables 10a and 10b show the responses and the number of academic publications that use or study each computation. It is clear that

[4]In the publications, link prediction referred to problems that predict a missing edge in a graph or data on an existing edge. Influence maximization referred to finding influential vertices in a graph, e.g., those that can bring more vertices to the graph. We did not provide detailed explanations about the problems to the participants.

Table 10: Machine learning computations and problems performed by the participants and studied in publications.

(a) Machine learning computations.

Computation	Total	R	P	A
Clustering	42	22	20	15
Classification	28	10	18	2
Regression (Linear / Logistic)	11	5	6	2
Graphical Model Inference	10	5	5	2
Collaborative Filtering	9	4	5	2
Stochastic Gradient Descent	4	2	2	3
Alternating Least Squares	0	0	0	2

(b) Problems solved by machine learning algorithms.

Computation	Total	R	P	A
Community Detection	31	15	16	5
Recommendation System	26	10	16	2
Link Prediction	25	10	15	2
Influence Maximization	14	5	9	2

Table 11: Graph traversals performed by the participants.

Traversal	Total	R	P
Breadth-first-search or variant	19	5	14
Depth-first-search or variant	12	4	8
Both	22	8	14
Neither	20	11	9

machine learning is used very widely in graph processing. Specifically, 61 participants indicated that they either perform a machine learning computation or solve a problem using machine learning on their graphs. Clustering is the most popular computation performed, while community detection is the most popular problem solved using machine learning. None of the participants selected alternating least squares as a computation they perform.

4.3 Other Questions on Computations

Streaming Computations: We asked the participants if they performed incremental or streaming computations on their graphs: 32 participants (16 researchers and 16 practitioners) indicated that they do. We followed up with a question asking them to describe the incremental or streaming computations that they perform. A total of 4 participants indicated computing graph or vertex-level statistics and aggregations; A total of 3 participants indicated incremental or streaming computation of the following algorithms: approximate connected components, k-core, and hill climbing. For completeness, we list the other computations participants mentioned: computing node or community properties, calculating approximate answers to simple queries, incremental materialization, incremental enhancement of the knowledge graph, and scheduling.

We note that the 22 software products in Table 1 have limited or no support for incremental and streaming computations. It would be interesting to clarify which software the participants use to perform their streaming and incremental computations.

Traversals: We asked the participants which fundamental traversals, breadth-first search or depth-first search, they use in their algorithms. Table 11 shows the responses. Participants commonly use both kinds of traversals.

5. GRAPH SOFTWARE

We next review the properties of the different graph software that the participants use.

Table 12: Software for graph queries and computations.

Software	Total	R	P	A
Graph Database System (e.g., Neo4j, OrientDB, TitanDB)	59	20	39	1
Apache Hadoop, Spark, Pig, Hive	29	11	18	2
Apache Tinkerpop (Gremlin)	23	9	14	1
Relational Database Management System (e.g., MySQL, PostgreSQL)	21	6	15	1
RDF Engine (e.g., Jena, Virtuoso)	16	8	8	1
Distributed Graph Processing Systems (e.g., Giraph, GraphX)	14	8	6	17
Linear Algebra Library / Software (e.g., MATLAB, Maple, BLAS)	8	6	2	3
In-Memory Graph Processing Library (e.g., SNAP, GraphStream)	7	5	2	2

5.1 Software Types

Software for Querying and Performing Computations: We asked the participants which types of graph software they use to query and perform computations on their graphs. The choices included 5 types of software from Table 1 as well as distributed data processing systems (DDPSes), such as Apache Hadoop and Spark, relational database management systems (RDBMSes), and linear algebra libraries and software, such as BLAS and MATLAB. Table 12 shows the exact choices and responses: 84 participants answered this question and each selected 2 or more types of software. We highlight 3 interesting observations:

- *Popularity of Graph Database Systems*: The most popular choice was graph database systems. We suspect this is partly due to their increasing popularity and partly due to the inherent bias in the participants we recruited—as explained in Section 2.2, more of them came from users of graph database systems. We did not ask the participants which specific graph database system they used.
- *Popularity of RDBMSes*: 21 participants (6 researchers and 15 practitioners) chose RDBMSes. We consider this number high given that we did not recruit participants from the mailing lists of any RDBMS. Interestingly, 16 of these 20 participants also indicated using graph database systems. From our survey, we cannot answer what the participants used RDBMSes for. It is possible that they use an RDBMS as the main transactional storage and a graph database system for graph-specific tasks such as traversals.
- *Unpopularity of DGPSes*: Only 6 practitioners indicated using a DGPS, such as Giraph, GraphX, and Gelly. This contrasts with DGPSes' popularity among academics, where they are the most popular systems, studied by 17 publications. One can consider graph database systems as RDBMSes that are specialized for graphs and DGPSes as DDPSes that are specialized for graphs. In light of this analogy, we note that there is an opposite trend in the usage of these groups of systems. While more participants indicated using graph database systems than RDBMSes, significantly more participants indicated using DDPSes than DGPSes.

Software for Non-Querying Tasks: We asked the participants which types of graph software, possibly an in-house one, they use for tasks other than querying graphs. Table 13 shows the choices and the responses. We highlight one interesting observation:

- *Importance of Visualization:* Visualization software is, by a large margin, the most popular type of software participants use

Table 13: Software used for non-querying tasks.

Software	Total	R	P	A
Graph Visualization	55	22	33	1
Build / Extract / Transform	14	8	6	0
Graph Cleaning	5	1	4	0
Synthetic Graph Generator (e.g., Graph 500's graph generator)	4	3	1	13
Specialized Debugger	2	0	2	0

Table 14: Architectures of the software used by participants.

Architecture	Total	R	P
Single Machine Serial	31	17	14
Single Machine Parallel	35	21	14
Distributed	45	17	28

among the 5 choices. This clearly shows that graph visualization is a very common and important task. As we discuss in Section 6, participants also indicated visualization as one of their most important challenges when processing graphs.

5.2 Other Questions on Software

Software Architectures: We asked the participants the architectures of the software products they use for processing graphs. The choices were single machine serial, single machine parallel, and distributed. Table 14 shows the responses. Distributed products were the most popular choice and users' selections highly correlated with the size of graphs they have. For example, 29 of the 45 participants that selected distributed architecture had graphs over 100M edges.

Data Storage in Multiple Formats: We asked the participants whether or not they store a single graph in multiple formats: 33 participants answered yes and the most popular multiple format combination was a relational database format and a graph database format. Appendix C provides the detailed responses.

6. PRACTICAL CHALLENGES

In this section, we first discuss the challenges in graph processing that the participants identified, followed by a discussion of the challenges that we identified through our review of user emails and code repositories of different types of graph technologies.

6.1 Challenges Identified from Survey

We asked the participants 2 questions about the challenges they face when processing their graphs. First, we asked them to indicate their top 3 challenges out of 10 choices we provided. Table 15 shows the choices and the participants' responses. Second, we asked them to state their biggest challenge in a short-answer question. Three major challenges stand out unequivocally from the responses:

- *Scalability*: The ability to process large graphs is the most pressing challenge participants face. Scalability was the most popular choice in the first question for both researchers and practitioners. Moreover, it was the most popular answer in the second question where 13 participants reiterated that scalability is their biggest challenge. The specific scalability challenges that the participants mentioned include inefficiencies in loading, updating, and performing computations, such as traversals, on large graphs.
- *Visualization*: Perhaps more surprisingly, graph visualization emerges as one of the top 3 graph processing challenges, as indicated by 39 participants in the first question and 1 participant in the short-answer question. This is consistent with the participants indicating visualization as the most popular non-query task they perform on their graphs, as discussed in Section 5.1.
- *Query Languages and APIs*: Query languages and APIs present another common graph processing challenge, as indicated by 39 participants in the first question and 5 participants in the short-answer question. The specific challenges mentioned in the short-answer responses include expressibility of query languages, compliance with standards, and integration of APIs with existing systems. For instance, one participant found current graph query languages to have poor support for debugging queries and another participant indicated their difficulty in finding software that complies fully with SPARQL standards.

6.2 Challenges Identified from Review

To go beyond the survey and to understand more specific challenges users face or new functionalities users want, we studied the user emails and code repositories of different classes of software. Below, we categorize the challenges we found on visualization in graph database systems, RDF engines, DGPSes, and graph libraries, separately under *Visualization*. We also list the challenges we found in graph database systems and RDF engines related to query languages separately under *Query Languages*. The exact counts of emails and issues we found for each challenge is in Table 19 in the appendix.

Graph Database Systems and RDF Engines:

- *High-Degree Vertices*: Users want the ability to process very high-degree vertices in a special way. One common request is to skip finding paths that go over such vertices either for efficient querying or because users do not find such paths interesting.
- *Hyperedges*: Hyperedges are edges between more than 2 vertices, e.g., a family relationship between three individuals. In graph database systems and RDF engines, there is no native-way to represent hyperedges. The user discussions include suggestions to simulate hyperedges, such as having a "hyperedge vertex" and linking the vertices in the hyperedge to this mock vertex.
- *Versioning and Historical Analysis*: Users want the ability to store the history of the changes made to the vertices and edges and query over the different versions of the graph. These requests are made in systems that do not support versioning and the discussions are on how to add versioning support at the application layer.
- *Schema and Constraints*: Users want the ability to define schemas over their graphs, analogous to DTD and XSD schemas for XML data [15], usually as a means to define constraints over their data. Examples include enforcing that the graph is acyclic or that some vertices always have a certain property.
- *Triggers*: Users ask for trigger-like capabilities in their graph database systems. Examples include automatically adding a particular property to vertices during insertion or creating a backup of a vertex or an edge in the filesystem during updates. We note that some systems do support limited trigger functionality, such as OrientDB's *hooks* or Neo4j's *TransactionEventHandler* API.

Graph Visualization Software:

- *Customizability:* One common challenge is to have the ability to customize the layout and design of the rendered graph, such as the shape or color of the vertices and edges.
- *Layout*: Another common challenge is drawing graphs with certain structures on the screen according to a specific layout users had in mind. The most common example is drawing *hierarchical* graphs, i.e., those in which some vertices are drawn on top of other vertices in an organizational hierarchy. Other

Table 15: The graph processing challenges selected by the participants.

Challenge	Total	R	P
Scalability (i.e., software that can process larger graphs)	45	20	25
Visualization	39	17	22
Query Languages / Programming APIs	39	18	21
Faster graph or machine learning algorithms	35	19	16
Usability (i.e., easier to deploy, configure, and use)	25	10	15
Benchmarks	22	12	10
Extract & Transform	20	6	14
More general purpose graph software (e.g., that can process offline, online, and streaming computations)	20	9	11
Graph Cleaning	17	7	10
Debugging & Testing	10	2	8

examples include the drawing of star graphs, planar graphs, or a specialized tree layout, such as a *phylogenetic tree* [43].

- *Dynamic Graph Visualization:* Several users want support for or have challenges in animating the additions, deletions, and updates in a dynamic graph that is changing over time.

Users also have challenges in rendering large graphs with thousands or even millions of vertices and edges.

Query Languages: One of the most popular discussions in user emails of graph database systems and RDF engines was writing different queries in the query language of the software. In almost every case, there was a way of satisfying the users' needs. Below we list 2 such types of queries that could be interesting to researchers.

- *Subqueries*: Many users have challenges in the expression or performance of subqueries, i.e., using a query as part of another query. The challenges vary across different systems. Some users want the ability to embed SQL as a subquery in SPARQL. Other users want the results of a subquery to be a graph that can further be queried,[5] or to use a subquery as a predicate in another query.
- *Querying across Multiple Graphs*: A common request in graph database systems and RDF engines is to construct queries that span multiple graphs, such as using the results of a traversal in one graph to start traversals in another. This is analogous to querying over multiple tables by joins in RDBMSes.[6]

Profiling and debugging slow queries and using indices correctly to speed up queries are other common topics among users.

DGPSes and Graph Libraries:

- *Off-the-Shelf Algorithms:* The most common request we found in DGPSes and graph libraries is the addition of a new algorithm that users could use off-the-shelf. All of these software products provide lower-level programming APIs using which users can compose graph algorithms. A small number of users want enhancements to these APIs as well. From our review, it appears that users of these software products find more value in directly using an already implemented algorithm than implementing the algorithms themselves.
- *Graph Generators:* All of the DGPSes and graph libraries in our list have modules to generate synthetic graphs. Our review revealed that users find these graph generators useful, e.g., for testing algorithms. A common request was the ability to generate different kinds of synthetic graphs, such as k-regular graphs or random directed power-law graphs.

[5]This feature is called *composition* and is supported in SPARQL but not in the languages of some graph database systems.

[6]This functionality is supported in RDF engines but not supported in some graph database systems.

Table 16: Time spent by the participants on different tasks.

Task	0 - 5 hours	5 - 10 hours	>10 hours
Analytics	30	18	23
Testing	40	12	20
Debugging	37	18	15
Maintenance	46	14	13
ETL	44	14	10
Cleaning	52	10	6

- *GPU Support:* Several users, both in DGPSes and graph libraries, want support for running graph algorithms on GPUs.

In every DGPS we reviewed, a common challenge is users' computations running out of cluster memory or having problems when using disk. We also note that except for Gelly, every DGPS and every graph library either have a visualization component or users have requests to add one, showing the importance of visualization across users of a range of different graph technologies.

7. WORKLOAD BREAKDOWN

We asked the participants how many hours per week they spend on 6 graph processing tasks and provided them with 3 choices: (i) less than 5 hours; (ii) 5 to 10 hours; and (iii) more than 10 hours. Table 16 shows the choices and ranks the tasks in terms of the number of participants that selected more than 10 hours first, then 5 to 10 hours, and then less than 5 hours. According to this ranking, the participants spend the most time in analytics and testing and the least time on ETL and cleaning.

8. RELATED WORK

To the best of our knowledge, our survey is the first study that has been conducted across users of a wide spectrum of graph technologies to understand graph datasets, computations, and software that is in use, and the challenges users face.

Several surveys in the literature have conducted user studies to compare the effectiveness of different techniques used to perform a particular graph processing task, primarily in visualization [7, 31] and query languages [36, 57, 58]. Additionally, several software vendors have conducted surveys of their users to understand how their software is used to process graphs. Some of these surveys are publicly available [18, 49, 63]. However, these surveys are limited to studying a specific software product.

There are also numerous surveys in the literature studying different topics related to graph processing. Examples include surveys on query languages for graph database systems and RDF engines [2, 29, 32], graph algorithms [1, 30, 39, 68], graph processing

systems [5, 44], and visualization [12, 66]. These surveys do not study how users use the technologies in practice.

9. CONCLUSION AND FUTURE WORK

Managing and processing graph data is prevalent across a wide range of fields in research and industry. We surveyed 89 users and reviewed user emails and code repositories of 22 software products. The participants' responses and our review provide useful insights into the types of graphs users have, the software and computations users use, and the major challenges users face when processing their graphs. We hope that these insights will help guide research on graph processing.

Our study also raises several interesting questions we cannot answer from our survey and review:

- *Benefits of Representing Data as Graphs*: We were surprised to see that many users represent as graphs entities that may not naturally be thought of as graphs, e.g., videos and captions, products-orders-transactions, pieces and properties of mechanical engines. What benefit do the participants find in representing such data as graphs and what benefit do they get from graph queries and analyses on such data?
- *Benefits of Visualization*: Visualization appears as the most popular non-querying task users perform on their graphs. Visualization can be used at different points in data processing pipelines, such as data exploration [54], query formulation [56], or debugging [60]. In our review of emails and code repositories, we identified use cases that were only on data exploration and producing visual charts. At what stage of their graph processing pipelines do the participants use visualization? What benefits do the users get from visualizing graphs, especially large ones?
- *Incremental and Streaming Computations*: 32 participants indicated performing incremental or streaming graph computations. However, their computations were not clear and we did not find instances of streaming or incremental graph computations in our review of emails and issues. Which software do users use to perform their incremental and streaming computations?

We are conducting a follow-up survey consisting of one-on-one interviews with the users of different graph technologies. We hope to be able to answer some of the above questions in a future publication.

We conclude with two final remarks. First, we found product-order-transaction graphs to be the most popular type of graph. Workloads that process these product data appear in popular SQL benchmarks, such as TPC-C [65], and are well studied in research on relational systems. However, existing graph benchmarks, such as LDBC [41] and Graph500 [23], do not yet provide workloads and data to process product graphs. Such benchmarks are great facilitators of research, and the development of benchmarks using product graphs and workloads would be useful in research.

Second, query languages and APIs emerged as one of the top challenges in our survey and certainly the most popular discussion topic in emails and code repositories. These challenges can be partly mitigated by a collaborative effort to standardize the query languages of different graph software that satisfy users' needs. One such successful effort is the adoption of SPARQL as a standard for querying RDF data. Similar efforts are ongoing for developing standard query languages and JDBC-like interfaces [37] for property graphs, such as the Gremlin language [59], the efforts to standardize openCypher [52], the discussions in the LDBC community [42], and a recent ISO Ad hoc group [33], exploring the features that should go into a standard graph query language.[7] There is also ongoing effort to develop a standard set of linear algebra operations for expressing graph algorithms [47].

[7]Personal communication with members of the PGQL [55] team, who are part of the ISO group.

10. ACKNOWLEDGMENTS

We are grateful to Chen Zou for helping us in using online survey tools and drafting an early version of this survey. We are also grateful to Jeremy Chen and Chathura Kankanamge for their valuable comments on the survey and help in categorizing the academic publications, user emails, and issues.

11. REFERENCES

[1] C. C. Aggarwal and H. Wang. *Graph Data Management and Mining: A Survey of Algorithms and Applications*, pages 13–68. Springer US, 2010.

[2] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc. Foundations of Modern Graph Query Languages. *CoRR*, abs/1610.06264, 2016.

[3] ArrangoDB. `https://www.arangodb.com`.

[4] M. Balcan and K. Q. Weinberger, editors. *Proceedings of the International Conference on Machine Learning*, 2016. `http://jmlr.org/proceedings/papers/v48/`.

[5] O. Batarfi, R. E. Shawi, A. G. Fayoumi, R. Nouri, S.-M.-R. Beheshti, A. Barnawi, and S. Sakr. Large Scale Graph Processing Systems: Survey and an Experimental Evaluation. *Cluster Computing*, 18(3):1189–1213, 2015.

[6] Basic Linear Algebra Subprograms. `http://www.netlib.org/blas`.

[7] S. Bridgeman and R. Tamassia. *A User Study in Similarity Measures for Graph Drawing*, pages 19–30. Springer Berlin Heidelberg, 2001.

[8] Caley Graph Database. `https://cayley.io`.

[9] L. Cao, C. Zhang, T. Joachims, G. I. Webb, D. D. Margineantu, and G. Williams, editors. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2015. `http://dl.acm.org/citation.cfm?id=2783258`.

[10] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan. One Trillion Edges: Graph Processing at Facebook-Scale. *PVLDB*, 8(12):1804–1815, 2015.

[11] Conceptual Graphs. `http://conceptualgraphs.org`.

[12] W. Cui and H. Qu. A Survey on Graph Visualization. PhD Qualifying Exam Report, Computer Science Department, Hong Kong University of Science and Technology, 2007.

[13] Cytoscape. `http://www.cytoscape.org`.

[14] DGraph. `https://dgraph.io`.

[15] DTD and XSD XML Schemas. `https://www.w3.org/standards/xml/schema`.

[16] Elasticsearch X-Pack Graph. `https://www.elastic.co/products/x-pack/graph`.

[17] Apache Flink. `https://flink.apache.org`.

[18] Apache Flink User Survey 2016. `https://github.com/dataArtisans/flink-user-survey-2016`.

[19] Gephi. `https://gephi.org`.

[20] S. Ghandeharizadeh, S. Barahmand, M. Balazinska, and M. J. Freedman, editors. *Proceedings of the Symposium on Cloud Computing*, 2015. `http://doi.org/10.1145/2806777`.

[21] Apache Giraph. `https://giraph.apache.org`.

[22] Graph for Scala. `http://www.scala-graph.org`.

[23] Graph 500 Benchmarks. `http://graph500.org`.

[24] GraphStream. `http://graphstream-project.org`.

[25] Graph-tool. https://graph-tool.skewed.de.
[26] Graphviz. https://graphviz.readthedocs.io.
[27] Apache Spark GraphX. https://spark.apache.org/graphx.
[28] Apache TinkerPop. https://tinkerpop.apache.org.
[29] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. *A Comparison of RDF Query Languages*, pages 502–517. Springer Berlin Heidelberg, 2004.
[30] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.
[31] D. Holten and J. J. van Wijk. A User Study on Visualizing Directed Edges in Graphs. In *Proceedings of International Conference on Human Factors in Computing Systems*, pages 2299–2308, 2009.
[32] F. Holzschuher and R. Peinl. Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4j. In *Proceedings of the Joint EDBT/ICDT Workshops*, pages 195–204, 2013.
[33] ISO/IEC Directives, Part 1. http://www.iso.org/sites/directives/directives.html#toc_marker-16.
[34] H. V. Jagadish and A. Zhou, editors. *PVLDB, Volume 7*, 2013-2014. http://www.vldb.org/pvldb/vol7.html.
[35] JanusGraph. http://janusgraph.org.
[36] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying Knowledge Graphs by Example Entity Tuples. *CoRR*, abs/1311.2100, 2013.
[37] JDBC. http://www.oracle.com/technetwork/java/overview-141217.html.
[38] Apache Jena. https://jena.apache.org.
[39] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology Visualization Methods: A Survey. *ACM Computing Surveys*, 39(4):10, 2007.
[40] K. Keeton and T. Roscoe, editors. *Proceedings of the Symposium on Operating Systems Design and Implementation*, 2016. https://www.usenix.org/conference/osdi16.
[41] LDBC Benchmarks. http://ldbcouncil.org/benchmarks.
[42] LDBC D6.6.4 Standardization Report. http://ldbcouncil.org/sites/default/files/LDBC_D6.6.4.pdf.
[43] I. Letunic and P. Bork. Interactive Tree Of Life: An Online Tool for Phylogenetic Tree Display and Annotation. *Bioinformatics*, 23(1):127–128, 2006.
[44] Y. Lu, J. Cheng, D. Yan, and H. Wu. Large-scale Distributed Graph Computing Systems: An Experimental Evaluation. *PVLDB*, 8(3):281–292, 2014.
[45] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A System for Large-scale Graph Processing. In *Proceedings of International Conference on Management of Data*, pages 135–146, 2010.
[46] MATLAB. https://www.mathworks.com.
[47] T. Mattson, D. A. Bader, J. W. Berry, A. Buluç, J. J. Dongarra, C. Faloutsos, J. Feo, J. R. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. E. Leiserson, A. Lumsdaine, D. A. Padua, S. Poole, S. P. Reinhardt, M. Stonebraker, S. Wallach, and A. Yoo. Standards for Graph Algorithm Primitives. In *Proceedings of High Performance Extreme Computing Conference*, pages 1–2, 2013.
[48] Neo4j. https://neo4j.com.
[49] The 2016 State of the Graph Report, https://neo4j.com/resources/2016-state-of-the-graph.
[50] NetworKit. https://networkit.iti.kit.edu.
[51] NetworkX. https://networkx.github.io.
[52] openCypher. http://www.opencypher.org.
[53] OrientDB. https://orientdb.com.
[54] M. Paradies, M. Rudolf, and W. Lehner. GraphVista: Interactive Exploration Of Large Graphs. *CoRR*, abs/1506.00394, 2015.
[55] PGQL: Property Graph Query Language. http://pgql-lang.org.
[56] R. Pienta, F. Hohman, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. Visual Graph Query Construction and Refinement. In *Proceedings of International Conference on Management of Data*, pages 1587–1590, 2017.
[57] R. Pienta, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. VISAGE: Interactive Visual Graph Querying. In *Proceedings of International Working Conference on Advanced Visual Interfaces*, pages 272–279, 2016.
[58] M. Rath, D. Akehurst, C. Borowski, and P. Mäder. Are graph query languages applicable for requirements traceability analysis? In *Proceedings of International Conference on Requirements Engineering: Foundation for Software Quality*, 2017.
[59] M. A. Rodriguez. The Gremlin Graph Traversal Machine and Language. *CoRR*, abs/1508.03843, 2015.
[60] S. Salihoglu, J. Shin, V. Khanna, B. Q. Truong, and J. Widom. Graft: A Debugging Tool For Apache Giraph. Technical report, Stanford University, 2014. http://ilpubs.stanford.edu:8090/1109/.
[61] A. Sharma, J. Jiang, P. Bommannavar, B. Larson, and J. Lin. GraphJet: Real-Time Content Recommendations at Twitter. *PVLDB*, 9(13):1281–1292, 2016.
[62] SNAP: Standford Network Analysis Project. https://snap.stanford.edu.
[63] Lightbend Apache Survey 2015. https://info.lightbend.com/COLL-20XX-Spark-Survey-Report_LP.html.
[64] Sparksee. http://www.sparsity-technologies.com.
[65] The TPC-C benchmark. http://www.tpc.org/tpcc.
[66] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing Group Structures in Graphs: A Survey. *Computer Graphics Forum*, 36(6):201–225, 2017.
[67] OpenLink Virtuoso. https://virtuoso.openlinksw.com.
[68] C. Wang and J. Tao. Graphs in Scientific Visualization: A Survey. *Computer Graphics Forum*, 36(1):263–287, 2017.
[69] J. West and C. M. Pancake, editors. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016. https://dl.acm.org/citation.cfm?id=3014904.

APPENDIX

A. CHOICES OF GRAPH COMPUTATIONS

One way to ask this question is to include a short-answer question that asks "What queries and graph computations do you perform on your graphs?" However, the terms graph queries and computations are very general and we thought this version of the question could be under-specified. We also knew that participants respond less to short-answer questions, so instead we first asked a multiple choice question followed by a short answer question for computations that may not have appeared in the first question as a choice.

In a multiple choice question, it is very challenging to provide a list of graph queries and computations from which participants can select, as there is no consensus on what constitutes a graph

Table 17: Data storage formats.

Data Storage Format	#
Graph Databases	10
Relational Databases	8
RDF Store	5
NoSQL Store (Key-value, HBase)	5
XML / JSON	4
JGF / GML / GraphML	4
CSV / Text files	3
Elasticsearch	3
Binary	2

Table 18: Graph sizes in user emails and issues.

(a) Number of vertices.

Vertices	#
100M - 1B	10
1B-10B	17
10B - 100B	1
>100B	2

(b) Number of edges.

Edges	#
1B - 10B	42
10B - 100B	17
100B - 500B	6
>500B	1

Table 19: Challenges found in user emails and issues.

Challenge	#
Graph DBs and RDF Engines	
High-degree Vertices	24
Hyperedges	18
Triggers	18
Versioning and Historical Analysis	14
Schema & Constraints	10
Visualization Software	
Layout	31
Customizability	30
Large-graph Visualization	8
Dynamic Graph Visualization	4
Query Languages	
Subqueries	7
Querying Across Multiple Graphs	6
DGPS and Graph Libraries	
Off-the-shelf Algorithms	41
Graph Generators	7
GPU Support	3

computation, let alone a reasonable taxonomy of graph computations. We decided to select a list of graph queries and computations that appeared in the publications of six conferences, as described in Section 2.3. We use the term graph computation here to refer to a query, a problem, or an algorithm.

For each of the 90 papers, we identified each graph computation, if (i) it was directly studied in the paper; or (ii) for papers describing a software, it was used to evaluate the software. We used our best judgment to categorize the computations that were variants of each other or appeared as different names under a single category. For example, we identified motif finding, subgraph finding, and subgraph matching as *subgraph matching*. When reviewing papers studying linear algebra operations, e.g., a matrix-vector multiplication, for solving a graph problem such as BFS traversal, we identified the graph problem and not the linear algebra operation as a computation.

Finally, for each identified and categorized computation, we counted the number of papers that study it and selected the ones that appeared in at least 2 papers. In the end, we provided the participants with the 13 choices that are shown in Table 9.

B. CHOICES OF MACHINE LEARNING COMPUTATIONS

Similar to graph computation, machine learning computation is a very general term. Instead of providing a list of ad-hoc computations as choices, we reviewed each machine learning computation that appeared in the 90 graph papers we had selected. Specifically, the list of machine learning computations we identified included the following: (i) *high-level classes of machine learning techniques*, such as clustering, classification, and regression; (ii) *specific algorithms and techniques*, such as stochastic gradient descent and alternating least squares that can be used as part of multiple higher-level techniques; and (iii) *problems* that are commonly solved using a machine learning technique, such as community detection, link prediction, and recommendations. We then selected the computations, i.e., high-level techniques, specific techniques, or problems, that appeared in at least 2 papers. As in the graph computations question, we used our best judgment to identify and categorize similar computations under the same name.

C. STORAGE IN MULTIPLE FORMATS

We asked the 33 participants who said that they store their data in multiple formats, which formats they use as a short-answer question. Out of the 33 participants, 25 responded. Their responses contained explicit data storage formats as well as the internal formats of different software. Table 17 shows the number of responses we received for the main formats. A relational database and a graph database format combination was the most popular combination. Other combinations varied significantly, examples of which include HBase and Hive, GraphML and CSV, and XML and triplestore.

D. OTHER TABLES

Table 18 shows the sizes of graphs we found in user emails and issues. Table 19 shows the number of emails and issues we identified for each specific challenge we discussed in Section 6.2. Table 20 shows the total number of emails and issues we reviewed for each software product from January to September of 2017. The table also shows the number of commits in the code repositories of each software product during the same period.

Table 20: The number of emails and issues we reviewed, and the code commits in the repositories of each software product.

Technology	Software	# Emails	# Issues	# Commits
Graph Database	ArrangoDB	140	466	5264
	Caley	50	57	151
	DGraph	175	558	760
	JanusGraph	225	308	411
	Neo4j	286	243	4467
	OrientDB	169	668	918
RDF Engine	Apache Jena	307	126	471
	Sparksee	8	NA	NA
	Virtuoso	72	61	179
Distributed Graph Processing Engine	Apache Flink (Gelly)	34	68	48
	Apache Giraph	19	34	23
	Apache Spark (GraphX)	23	28	11
Query Language	Gremlin	409	206	1285
Graph Library	Graph for Scala	10	12	18
	GraphStream	18	26	7
	Graphtool	121	66	172
	NetworKit	37	30	236
	NetworkX	78	148	171
	SNAP	57	17	34
Graph Visualization	Cytoscape	388	264	8
	Elasticsearch (X-Pack Graph)	50	38	NA
	Gephi	NA	147	10
	Graphviz	NA	58	277
Graph Representation	Conceptual Graphs	30	NA	NA