Querying Graph Databases at Scale

Aidan Hogan
ahogan@dcc.uchile.cl
DCC, University of Chile
Instituto Milenio Fundamentos de los Datos
Santiago, Chile

Domagoj Vrgoč
vrdomagoj@uc.cl
Pontificia Universidad Católica de Chile
Instituto Milenio Fundamentos de los Datos
Santiago, Chile

ABSTRACT

The tutorial provides an in-depth overview of recent advances in algorithms and data structures for processing graph database queries. The focus will be on scalable algorithms that have been demonstrated to work over real world knowledge graphs. We will also present detailed performance comparisons of classical and recent algorithms. The tutorial will be divided into four sections. The first section will motivate the use of graph databases for querying knowledge graphs, and will introduce the attendees to graph data models and the query language landscape. The second section will discuss how to efficiently evaluate graph patterns, introducing the worst-case optimal join techniques and comparing them to classical join algorithms. The third section will discuss techniques for efficiently evaluating path queries and for constructing compact representations of potentially exponential sets of paths. In the final section we will introduce recent advances in compressed data structures that ease the high memory requirements of worst-case optimal join algorithms and also provide a template for evaluating path queries in a highly optimised manner.

CCS CONCEPTS

• **Information systems → Query languages for non-relational engines**; **Graph-based database models**.

KEYWORDS

graph databases; graph query languages; basic graph patterns; path queries; worst-case optimal joins; knowledge graphs

ACM Reference Format:
Aidan Hogan and Domagoj Vrgoč. 2024. Querying Graph Databases at Scale. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626246.3654695

1 OVERVIEW

The tutorial starts by providing an introduction to graph databases, which have been gaining more and more attention in recent years, both in the context of querying large-scale knowledge graphs, and in enterprise solutions. Open knowledge graphs such as Wikidata, DBpedia, LinkedGeoData, etc., provide query services that can receive millions of queries per day. However, such traffic implies significant challenges in terms of scalability, efficiency and expressivity. Similarly, enterprise graphs require complex query features in order to allow a thorough analysis of the data, requiring highly efficient algorithms for processing graph queries. Therefore, the objective of the tutorial is to give an in-depth overview of techniques used to evaluate major classes of graph database queries. Though various well-known systems like Neo4j, Blazegraph, and Neptune, have already been in use for a substantial amount of time, there remain many relevant research challenges around graph databases.

The tutorial will provide a survey of recent advances in this area divided into four sections. The first section will motivate the use of graph databases, showing live examples of querying Wikidata and other open knowledge graphs. This section will also highlight the benefits of modelling data as graphs, and introduce the audience to graph data models currently in use. We will then explore query languages defined for graph databases and explain their core features; in particular, we will discuss basic graph patterns, regular path queries, complex graph patterns, and their semantics. The second section will discuss worst-case optimal join techniques for efficiently evaluating graph patterns, which guarantee that the cost of evaluating a graph query does not exceed a certain theoretical bound, and have also been shown to enable higher levels of efficiency in practice for evaluating queries than traditional forms of joins. The third section will discuss techniques for efficiently evaluating path queries while also returning paths; such a feature is missing from many of the current graph query languages, and implies many challenges in terms of semantics and efficiency, but could be very useful in practice. We finish by explaining some recent advancements in the area of worst-case optimal algorithms, where compact data structures have been proposed to support these algorithms while alleviating their high memory requirements.

2 TOPIC AND RELEVANCE

Topic. Graph databases are being increasingly used on the Web in order to query open knowledge graphs [26], with key examples being Wikidata [28] and DBpedia [17] supporting the SPARQL 1.1 query standard [12]. According to an analysis of query logs by Malyshev et al. [20], Wikidata receives in the order of millions of queries per day, much of which come from software agents, such as Web applications that depend on the Wikidata knowledge graph. Despite these advances, such query services still exhibit performance issues that translate into timeouts and a lack of availability [26], affecting the usability of such services. More work is thus needed to develop efficient graph database engines. Furthermore, novel graph query languages – such as GQL and SQL/PGQ [8] – are emerging with

SIGMOD-Companion '24, June 9–15, 2024, Santiago, AA, Chile

ACM ISBN 979-8-4007-0422-2/24/06…$15.00
https://doi.org/10.1145/3626246.3654695

new features that satisfy practical means, such as the ability to return paths from the graph. These features have yet to see adoption in practice, as modern engines typically struggle when required to return and manipulate paths [9, 21], thus motivating the need for more research on efficient evaluation algorithms.

In this setting, we will first motivate why knowledge graphs are a promising solution for publishing data on the Web, and how graph databases enable querying these knowledge graphs. We will demonstrate real-world motivating examples of query services on the Web, including federation features that enable users to evaluate queries across multiple websites. We will discuss the challenges and opportunities that arise from such query services. We will then discuss the use of graph databases in offline scenarios and highlight the advantages that the model's flexibility brings to the process of data modelling, integration and manipulation. We continue by presenting the core features underlying all graph query languages, and show examples in concrete syntaxes such as SPARQL and Cypher/GQL. We will discuss the semantics of these features and how these languages differ. We will then turn to discussing state-of-the-art techniques for evaluating graph queries efficiently. First we will discuss state-of-the-art algorithms – based on worst-case optional joins [24] – for evaluating basic graph patterns. Second we discuss efficient algorithms for evaluating complex path queries, and in particular, queries that return paths. We will then discuss how compact data structures can be used to evaluate both worst-case optimal joins and path queries while dramatically reducing memory requirements. In all cases we will provide the abstract description of the algorithm, along with a detailed example, followed by a discussion of implementation considerations, and finishing with a detailed performance of these techniques with established baselines for graph query evaluation.

Timeliness. A number of recent developments underline the timeliness of our tutorial:

- Knowledge graphs are becoming an increasingly popular mechanism for managing diverse data at large scale, with many companies – such as Amazon, AirBnb, eBay, Google, LinkedIn, Microsoft, etc. – using knowledge graphs [15]. In this setting, graph databases offer a data management platform that enables indexing, updating, analysing and querying knowledge graphs.
- The Wikidata [28] community is actively looking for a new graph database engine – to replace Blazegraph – that can better cope with the high levels of traffic that the query service receives.[1] This concrete application indicates the importance of efficient algorithms for evaluating graph queries at scale.
- In parallel with the previous point, some new techniques are emerging to efficiently evaluate graph patterns; more specifically, worst-case optimal joins have proven to not only provide theoretical guarantees of efficiency, but to also provide notable performance benefits in practice [2, 4, 16, 23], while works on compact data structures for graphs address the issue of space [4]. Such techniques could reduce the costs of hosting query services such as Wikidata, and/or improve the quality of service provided by such endpoints. They would also pave the way towards handling larger knowledge graphs in an efficient manner, allowing more complex analysis of the underlying data.
- New graph query languages are currently being standardised, including GQL and SQL/PGQ [8]. These query languages include novel features, such as the ability to return paths, which raises open questions in terms of how to efficiently evaluate such queries on large-scale graphs like Wikidata. Furthermore, SPARQL 1.2 [14] is currently in the process of being standardised. This highlights the timeliness of discussing cutting-edge features for graph query languages, and efficient methods to evaluate them.

All of these recent developments establish the timeliness of the topic of our proposed tutorial.

Relevance. There are hundreds of public query services available on the Web based on graph databases, specifically, SPARQL engines [26]. Key amongst these engines are, as aforementioned, key knowledge graphs such as Wikidata [28] and DBpedia [17]. Our tutorial deals with the technology underlying these services, and how it can be improved in future. Furthermore, a variety of key technology companies are now managing internal knowledge graphs [15], where our tutorial again offers insights into how best to index and query such repositories of knowledge at large scale.

We foresee that our tutorial will be relevant, for example, to attendees of the conference with the following profiles:

- Practitioners involved in managing proprietary knowledge graphs with performance bottlenecks.
- Practitioners, students and other researchers interested in novel features being proposed for graph query languages.
- Practitioners interested in using public graph query services on the Web, such as Wikidata.
- Practitioners, students and other researchers more generally interested in using, constructing, managing and querying knowledge graphs.
- Students and other researchers working on problems relating to graph query languages or graph query optimisation.
- Practitioners involved in the development of graph database systems and other graph-data management platforms.

This list is intended to be illustrative rather than exhaustive.

3 CONTENTS

The tutorial is divided into four 30 minute sections as follows:

- **Part 1: Introduction to Graph Databases.** [30 min, Vrgoč] We divide the introductory material into three segments:
  a) *Motivation.* We begin by explaining what graph databases are and how they provide the backbone of knowledge graphs on the Web. For this we will examine several concrete Web Knowledge Graphs, illustrate their usefulness, and explain their defining features.
  b) *Graph data models.* We then explore the landscape of graph database formats, introducing the audience to the main models in use today. In particular, we will cover RDF [7] and property graphs [11], the two most popular formats in

[1] See https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/WDQS_backend_update/WDQS_backend_alternatives

use. We then discuss their shortcomings, in particular related to modelling Wikidata, and discuss some alternatives such as RDF* [13] and multi-layered graphs [3, 29].

c) *Graph query languages.* The standard way to access data residing in a Knowledge Graph is by querying its underlying graph database. We therefore continue with an overview of graph query languages. For this, we will first abstract the common features of all graph query languages and divide them into three categories: (i) graph patterns, which match a small graph-shaped pattern into the database; (ii) path queries, which explore how graph nodes are connected; and (iii) complex graph queries, which combine the two and extend them with additional features. We will then explain how such features are supported in concrete languages, with a specific focus on SPARQL [12], the W3C standard for querying data on the Web; and Cypher [11], the most popular property graph query language. We will also comment on the new GQL and SQL/PGQ standards [8], which are heavily influenced by Cypher.

Throughout this segment we will put a special focus on knowledge graphs that the audience can try out for themselves using only their Web browser. Concretely, we will use the Wikidata public SPARQL endpoint, and we will enable an endpoint hosting a social network dataset Pokec [18] and running on MillenniumDB [29], a property graph engine implementing GQL. While discussing graph query languages, we will try to highlight potential efficiency issues that can arise by using particular query features, or a combination thereof, and explain key research challenges in the area.

- **Part 2: Evaluation of Graph Patterns.** [30 min, Hogan] Graph patterns are a core feature common in all graph query languages. In practical terms graph patterns are equivalent to relational joins [1], but unlike relational data, graph queries usually require a large number of joins to be performed. For example, the public log of user posted Wikidata queries [20] contain patterns that consist of up to 22 joins. In this part of the tutorial we present *worst-case-optimal (wco)* join algorithms, a recent breakthrough in join query processing [24], and showcase the versatility of such algorithms in graph context [16, 25]. In particular, we will explain the inner workings of the *Leapfrog* wco-algorithm [27], and show what sort of performance gains it brings over real-world graphs. We will conclude by discussing the main challenges when implementing such algorithms and provide an in-depth overview of their performance when compared to standard join processing techniques.
- **Part 3: Evaluation of Path Queries.** [30 min, Vrgoč] Testing connectivity and retrieving paths is a key feature of graph databases which is currently not well supported in graph query engines. In this part of the tutorial we take a deep dive into state of the art algorithms for retrieving paths from knowledge graphs. In particular, we explain how the early theoretical work in the area [22] can be extended into a unifying framework for answering path queries under the diverse semantic requirements dictated by the SPARQL, GQL, and SQL/PGQ graph standards [8, 10, 12]. Following this we highlight the efficiency requirements for retrieving different types of paths [5], explain the trade-offs that one faces when handling complex restrictions for path queries, and present a series of effective approaches for handling path queries [9, 19, 21, 30]. We then discuss how to compactly represent potentially exponential sets of paths in order to be accessible to the query execution pipeline further down [21]. Throughout the presentation we will discuss different index structures that can be used to support efficient path processing, with a focus on B+trees and compact matrix representations [6] of graphs. A detailed performance comparison will be provided, showcasing the potential of future engines for handling huge sets of paths. We will also review main open problems in the area, particularly relating to novel path query features introduced in GQL and SQL/PGQ.
- **Part 4: Compressed wco-algorithms.** [30 min, Hogan] We finish by explaining some recent work [4] that addresses the main shortcoming of wco-algorithms, namely their high memory requirements. These works design compact structures that support all operations needed to run the Leapfrog algorithm, evaluating the query in this fashion. Such structures also support path queries. As in the previous sections, we provide a detailed performance comparison of such algorithms, in this case focusing on space savings they offer when compared to the default Leapfrog triejoin wco-implementation, and to classical relational techniques. This being the final section of the tutorial, we finish with an overview of main open challenges that graph databases face in the following years.

Within each section we will provide 5 minutes for Q&A and discussion, further aiming for interactive talks where audience members can ask questions, make comments, etc., during the presentations (not just at the end).

4 STYLE

The tutorial will primarily be in a lecture style, but will provide demos and interactive examples of querying the Web. In case the audience would like to try these examples for themselves, all that they will require is a browser with an internet connection. To facilitate this, in addition to lecture slides, we will provide the audience with a query cheat-sheet allowing them to try out a series of increasingly complex queries over different public endpoints, motivating and illustrating the concepts we describe in the tutorial in a hands-on manner.

5 AUDIENCE

Intended audience: Parts 1 & 2 of the tutorial are suitable for attendees that wish to learn about the technology behind modern day knowledge graphs. Attending the first two portions of the tutorial should also provide enough starting information for someone who wishes to begin using graph databases in their everyday practice. Parts 3 & 4 are aimed at more advanced users, engineers implementing data processing systems, or people interested in state of the art algorithms for evaluating graph queries.

Prerequisite knowledge: No prerequisite knowledge is needed for part 1. General understanding of algorithms, finite state automata, and computational complexity will be useful to fully understand

the concepts presented in parts 2, 3 & 4 of the tutorial, although the presentation will be self contained as much as possible using examples to illustrate concepts and techniques.

Learning outcomes: Following the tutorial an attendee should learn about different graph data models in use today, as well as query languages used to extract data from these. They should also gain an understanding of the usefulness and versatility of graph data, as well as how to model a diverse set of scenarios using graph databases. Attending parts 3 & 4 of the tutorial should provide a good overview of modern algorithms for processing graph queries, and allow developers to start deploying them in their own systems.

6 PREVIOUS EDITIONS

Parts of this tutorial were previously presented at:

- **SPIRE'22.** The 29th International Symposium on String Processing and Information Retrieval (SPIRE) was held in Concepción, Chile, with an attendance of about 50 participants. Hogan and Vrgoč were invited to give a tutorial on graph databases with the aim of introducing the topic to the SPIRE community and promoting collaboration with researchers working on text indexing and compression techniques, which were recently used to optimise graph queries at large scale. Tutorial materials are available at:
  – http://spire2022.inf.udec.cl.
- **AMW'23 Summer School.** Alberto Mendelzon Workshop (AMW) is an international workshop on data management being held annually in Latin America. The workshop includes a two day summer school aimed both at local students and international attendees. AMW'23 was held in Santiago, Chile, with an audience of about 40 students and researchers. The tutorial was of introductory level, explaining what graph databases are, how they are being used, and explaining some of the main algorithmic techniques used to process graph queries. Materials available at:
  – https://amw2023.org/program.html.

The material presented in this tutorial differs significantly in depth and scope. Specifically, compared to the SPIRE'22 tutorial, this proposal includes a more detailed synthesis of graph querying, an in-depth overview of modern algorithms for query processing over graphs, and a detailed description of implementation and performance considerations in the context of processing large graphs. Compared to the AMW'23 tutorial, this proposal is more technical in nature, and is aimed at providing general frameworks for processing large classes of graph queries. It also extends the presentation to include notions of path representations in graph query answers, compressed data structures for in-memory support of worst-case optimal algorithms, and describes performance bottlenecks in modern day graph processing.

7 ABOUT THE PRESENTERS

Aidan Hogan is an Associate Professor at the Department of Computer Science, University of Chile, and the Vice Director of the Millennium Institute for Foundational Research on Data (IMFD). His research interests relate primarily to the Semantic Web, Databases and Information Extraction; he has published over one hundred peer-reviewed works on these topics. He has been invited as a lecturer to seven summer schools and he has co-organised three summer schools. He has given previous tutorials at The Web Conference, ESWC, ISWC, SPIRE and HyperText. He is a sole author or lead author of three books; an Open Access version of the latest such book, entitled "Knowledge Graphs", is available online (https://kgbook.org). Further information is available from his homepage (http://aidanhogan.com/).

Domagoj Vrgoč is an Assistant Professor at Pontificia Universidad Católica de Chile, and an Associate Researcher at the Millennium Institute for Foundational Research on Data (IMFD). He has over a decade of research experience in the area of graph query languages and algorithms. He was invited to deliver several tutorial presentations, including AMW'23 and SPIRE'22, and he received several awards for his work, including ICDT Test-of-time award in 2023, and SIGMOD best industry paper award in 2023. He is the project lead of the open source graph engine MillenniumDB [29].

8 CONCLUSIONS

Graph databases have experienced a resurgence of interest in recent years, propelled in part by the growing adoption of knowledge graphs as a way to extract value from diverse data at large scale. This resurgence of interest has translated into the proposal of novel query languages, the development of new graph database engines, and the exploration of novel techniques to efficiently evaluate graph queries at large scale (considering both time and space). Our goal with this tutorial is to provide attendees with an overview of these recent developments, as well as insights into what the future might hold for querying graph databases at large scale.

ACKNOWLEDGEMENTS

The work was supported by Work supported by ANID – Millennium Science Initiative Program – Code ICN17_002 and ANID Fondecyt Regular projects 1221799 and 1221926.

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoč. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.*, 50(5), 2017.
[2] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoč, Mingxi Wu, and Dusan Zivkovic. Pg-schema: Schemas for property graphs. *Proc. ACM Manag. Data*, 1(2):198:1–198:25, 2023.
[3] Renzo Angles, Aidan Hogan, Ora Lassila, Carlos Rojas, Daniel Schwabe, Pedro A. Szekely, and Domagoj Vrgoc. Multilayer graphs: a unified data model for graph databases. In Vasiliki Kalavri and Semih Salihoglu, editors, *GRADES-NDA '22: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Philadelphia, Pennsylvania, USA, 12 June 2022*, pages 11:1–11:6. ACM, 2022.
[4] Diego Arroyuelo, Aidan Hogan, Gonzalo Navarro, Juan L. Reutter, Javiel Rojas-Ledesma, and Adrián Soto. Worst-case optimal graph joins in almost no space. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 102–114. ACM, 2021.
[5] Pablo Barceló Baeza. Querying graph databases. In *PODS 2013*, pages 175–188, 2013.
[6] Aydin Buluç, Jeremy T Fineman, Matteo Frigo, John R Gilbert, and Charles E Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 233–244, 2009.
[7] World Wide Web Consortium et al. Rdf 1.1 concepts and abstract syntax. 2014.

[8] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Oskar van Rest, Hannes Voigt, Domagoj Vrgoč, Mingxi Wu, and Fred Zemke. Graph pattern matching in GQL and SQL/PGQ. In *SIGMOD '22*, 2022.

[9] Benjamín Farias, Carlos Rojas, and Domagoj Vrgoc. Evaluating regular path queries in GQL and SQL/PGQ: how far can the classical algorithms take us? *CoRR*, abs/2306.02194, 2023.

[10] Nadime Francis, Amélie Gheerbrant, Paolo Guagliardo, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Liat Peterfreund, Alexandra Rogova, and Domagoj Vrgoč. A researcher's digest of GQL (invited talk). In *ICDT 2023*, 2023.

[11] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An Evolving Query Language for Property Graphs. In *SIGMOD 2018*, 2018.

[12] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. SPARQL 1.1 Query Language. W3C Recommendation, 2013.

[13] Olaf Hartig. Foundations of RDF★ and SPARQL★ (An Alternative Approach to Statement-Level Metadata in RDF). In Juan L. Reutter and Divesh Srivastava, editors, *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*, volume 1912 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[14] Olaf Hartig, Andy Seaborne, Ruben Taelman, Gregory Williams, and Thomas Pellissier Tanon. SPARQL 1.2 Query Language. W3C Working Draft, 2023.

[15] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Comput. Surv.*, 54(4):71:1–71:37, 2022.

[16] Aidan Hogan, Cristian Riveros, Carlos Rojas, and Adrián Soto. A worst-case optimal join algorithm for SPARQL. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2019.

[17] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[19] Jia Li, Wenyue Zhao, Nikos Ntarmos, Yang Cao, and Peter Buneman. Mitra: A framework for multi-instance graph traversal. *Proc. VLDB Endow.*, 16(10):2551–2564, 2023.

[20] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In *ISWC 2018*, 2018.

[21] Wim Martens, Matthias Niewerth, Tina Popp, Carlos Rojas, Stijn Vansummeren, and Domagoj Vrgoč. Representing paths in graph database pattern matching. *Proc. VLDB Endow.*, 16(7):1790–1803, 2023.

[22] Alberto O. Mendelzon and Peter T. Wood. Finding regular simple paths in graph databases. In *VLDB 1989*, pages 185–193, 1989.

[23] Amine Mhedhbi and Semih Salihoglu. Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins. *Proc. VLDB Endow.*, 12(11):1692–1704, 2019.

[24] Hung Q. Ngo. Worst-case optimal join algorithms: Techniques, results, and open problems. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 111–124. ACM, 2018.

[25] Dung T. Nguyen, Molham Aref, Martin Bravenboer, George Kollias, Hung Q. Ngo, Christopher Ré, and Atri Rudra. Join processing for graph patterns: An old dog with new tricks. *CoRR*, abs/1503.04169, 2015.

[26] Pierre-Yves Vandenbussche, Jürgen Umbrich, Luca Matteis, Aidan Hogan, and Carlos Buil Aranda. SPARQLES: Monitoring public SPARQL endpoints. *Semantic Web*, 8(6):1049–1065, 2017.

[27] Todd L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 96–106, 2014.

[28] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.

[29] Domagoj Vrgoč, Carlos Rojas, Renzo Angles, Marcelo Arenas, Diego Arroyuelo, Carlos Buil Aranda, Aidan Hogan, Gonzalo Navarro, Cristian Riveros, and Juan Romero. Millenniumdb: A persistent, open-source, graph database. *CoRR*, abs/2111.01540, 2021.

[30] Chengshuo Xu, Keval Vora, and Rajiv Gupta. Pnp: Pruning and prediction for point-to-point iterative graph analytics. In Iris Bahar, Maurice Herlihy, Emmett Witchel, and Alvin R. Lebeck, editors, *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, pages 587–600. ACM, 2019.

Received 22 December 2023; accepted 2 February 2024