Query Optimization for Inference-Based Graph Databases

Sonia Horchidan

Supervised by Paris Carbone
KTH Royal Institute of Technology, Stockholm, Sweden

Abstract
Knowledge Graphs are commonly characterized by two challenges: massive scale and sparsity. The former leads to slow response times for complex queries with random data accesses, especially when they require deep graph traversals. The latter, which is caused by missing connections and characteristics in graphs modeling real information, implies that any analysis based solely on explicitly stored data is bound to yield incomplete results. This work aims to develop a novel graph database architecture that leverages the power of Graph Machine Learning to equip graph queries with prediction capabilities while offering approximate but timely results to complex queries. We discuss challenges, design decisions, and research avenues required in materializing this prototype alongside the outline of the actively-pursued research plan.

Keywords
Graph Databases, Graph Representation Learning, Query Optimization, Uncertainty Estimation

1. Introduction

For years, the data management community has been investigating systems that can store, process, query, and analyze Knowledge Graphs (KGs) to enable critical decision-making or accelerate discoveries. One of the prevalent challenges refers to *query complexity*. Due to the requirement for random accesses, above-linear algorithms are difficult to execute, especially on natural graphs, which tend to converge to skewed power-law degree distributions [1]. However, the need for fast complex computations is evident in the industry [2, 3]. Yet, answering complex queries requiring deep traversals of heterogeneous data or employing computationally expensive algorithms lacks support in common data management tools [4, 2].

As current systems attempt to address issues like distribution skew or evolution, most architectures neglect one of the major difficulties of KGs: *data incompleteness*. Interconnected data modeling natural domains are often missing connections and properties [1]. Modern graph management tools such as graph databases, graph processing systems, or graph streaming frameworks can only consult or compute properties of data that is explicitly stored but are usually unable to draw any conclusion that is not explicitly stated [5, 4]. Furthermore, systems that natively support reasoning capabilities, such as RDF stores, usually offer deductive inference based on existing data and rely heavily on ontology definitions, which can grow complex and resource-intensive to maintain, especially for massive, intricate dynamic graph datasets. Such systems cannot be used to harness the potential of incomplete data, where observations might enclose latent patterns and valuable correlations [6].

At the other end of the spectrum, Graph Machine Learning and, particularly, Graph Representation Learning show tremendous potential in tackling incompleteness through inductive reasoning. Modern methods construct node, edge, or even subgraph Euclidean representations, in the form of embeddings that encode both intrinsic information and topological information through message-passing-like mechanisms. The embeddings have been used in literature successfully for downstream tasks, showcasing notable inference abilities. Moreover, Graph ML can accurately approximate complex graph tasks and alleviate computational costs, with recent works hinting at equivalence with dynamic programming [7]. However, learned methods are currently not integrated into production-level graph data management systems for reasons amongst which we highlight a general mistrust in black-box models.

This work targets the design of a graph database system equipped with inference capabilities to tackle data incompleteness and query complexity. To increase the trustworthiness of the predicted outcomes, we investigate the feasibility of user-defined uncertainty bounds for complex graph queries. We aim to tackle these challenges through the following contributions.

1. *We design a hybrid query optimizer that considers operators with dual execution strategies: one operating using raw data accesses and one operating as ML inferences directly in learned latent spaces.*
2. *We consider methods for error estimation in ML-powered query plans, which can grant end-users control over the inferred outcomes.*
3. *We explore techniques for continuous training and inference to strike a balance between ensuring low query latency and offering predicted query results.*

VLDB 2023 PhD Workshop, co-located with the 49th International Conference on Very Large Data Bases (VLDB 2023), August 28, 2023, Vancouver, Canada
sfhor@kth.se (S. Horchidan)
© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)
CEUR-WS.org/Vol-3452/paper9.pdf

2. Research Goal

The proposed architecture moves query computation from raw data accesses and expensive pointer-chasing for traversals to querying learned latent spaces. Specifically, we leverage the power of graph representation learning to approximate complex graph queries by using latent representations (i.e., embeddings). In this attempted design, certain database operators have two operating modes, which we define as follows: (1) *DB* operators, which refer to traditional database operations executed on explicitly stored data, and (2) *ML* operators, where raw data accesses are replaced with Machine Learning inference calls. The latter does not require graph data accesses to finalize a computation, as it is carried out as matrix multiplications using a given query input and a trained model.

To exemplify this concept, we briefly describe Query2Box [8], a graph ML method that solves multi-hop queries of arbitrary length and type. The results are approximated by traversing a learned latent space using projections and intersections of hyper-rectangles. This seminal work achieves noteworthy accuracy results, proving its capabilities for approximation and completing the results sets with meaningful inferred results. Furthermore, attesting to its acceleration benefits, Query2Box's theoretical response time grows linearly in the number of hops. In contrast, multi-hop traversals of heterogeneous graphs with raw data accesses are known to have exponential time complexity [8]. Query2Box can serve as an *ML* operator in our envisioned database to approximate traversals.

However, deep learning models' decision-making process is opaque to the user, who can only observe the input and output, thus causing a lack of credibility in the predictions obtained. For this reason, despite the impressive advances and breakthroughs of ML methods, their adoption remains limited. For a graph database, particularly, offering predicted query results with no control over the desired error or uncertainty bound is counter-productive. This research aims to tackle the trustworthiness challenge by investigating methods to allow for statistically sound uncertainty bounds per query. To illustrate, the following two-hop Cypher query, which identifies the item bought by customers related to a customer named Anna, restricts the tolerated error at 5%, thus allowing for ML predictions to be employed:

```
MATCH (:Customer {name: 'Anna'})-[:RELATED_TO]->
      (:Customer)-[:BOUGHT]->(r:Item)
RETURN r
WITH MAXIMUM UNCERTAINTY 0.05;
```

The user-defined uncertainty bounds can potentially assist the database query optimizer in identifying the best physical plan. For example, to satisfy the error threshold, the system can decide to execute the first hop using an *ML* operator backed, for instance, by Query2Box, while choosing to traverse the explicitly stored data for the second hop. Therefore, a novel query optimization process is paramount to bringing this vision to fruition.

Lastly, we highlight the potential prospect of the proposed design for graph processing on GPUs. *ML* operators can naturally benefit from modern hardware acceleration techniques since the computation is reduced to tensor operations. In contrast, conventional graph queries suffer from random data accesses, which are not trivial to accelerate [9].

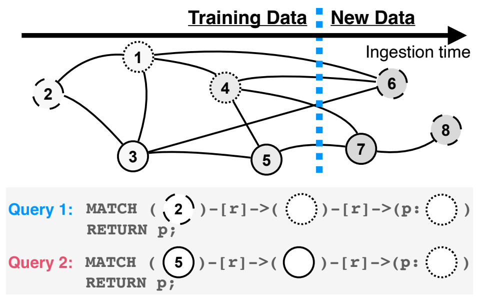
3. Challenges

With this work, we argue that executing time-consuming queries on incomplete Knowledge Graphs is counter-productive since the processing will, nonetheless, yield incomplete results sets. Instead, we relax the requirement for exact answers and discuss potential research directions to equip the capabilities of current database management systems with predictive powers.

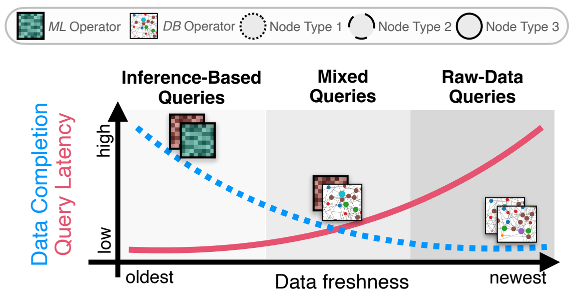
3.1. Hybrid Query Optimizer

When employing both *DB* and *ML* operators, conventional cost-based query optimizers are insufficient since they are solely based on cardinality estimation and execution time, whereas the envisioned architecture also features uncertainty constraints. To materialize the proposed predictive graph database, where the predictions are guided and controlled by uncertainty bounds, we will explore a novel optimization process that complements traditional query optimizers in database management systems. The query optimization can potentially be framed as a multi-objective cost function that minimizes IO while ensuring the uncertainty threshold is met. We will refer to the two cost models as performance-based and uncertainty-based, respectively.

Performance-based Optimization. The performance-based cost model estimates the IO costs on a given physical plan. It assigns a score for each valid physical query plan based on the expected processing cost and cardinality estimations. In the context of ML-enabled graph databases, a performance-based cost function operates using conventional database methods for cost estimation for *DB* operators. Estimating the cost of *ML* operators should follow a different approach: since the results of one *ML* operator are obtained, regardless of query, using the same input type and model as a predefined number of matrix multiplications, the cost can always be accurately computed. The cardinality estimates are, however, noteworthy. Traditional databases consider cardinality estimation to decide on the order of operators in the re-


Training Data
New Data
Ingestion time
1
2
3
4
5
6
7
8
Query 1: MATCH (2)-[r]->( )-[r]->(p:) RETURN p;
Query 2: MATCH (5)-[r]->( )-[r]->(p:) RETURN p;

(a) Examples of queries for a heterogeneous, dynamic graph, where nodes 1 to 5 were used to train the model of an *ML* operator.



(b) Relation between inference capabilities and query latency with respect to the ingestion timestamp of the queried data. Adapted from [4].

Figure 1: Latency-completeness trade-off introduced by *ML* operators in graph database querying.

sulting physical plan. Predictive databases require an additional layer of complexity: choosing between operator execution modes (i.e., *DB* or *ML*) will potentially result in different cardinalities for the result set since ML operators include plausible outcomes.

Bounding the Query Error. Similar to prior work [10], we extend graph query languages with support for user-defined error threshold, as shown in Section 2. Hence, the database user can choose if the system can maximize inferred results, obtain a mix of grounded and predicted answers, or restrict the inference capabilities, depending on application requirements. To the best of our knowledge, no general-purpose method currently exists to quantify the error of ML inference for embedding-based methods.

1. Estimating Uncertainty. Promising directions for black-box models, such as Conformal Prediction and Venn Predictors, rely on calibration sets to provide sound uncertainty bounds at the expense of producing multiple predictions or probabilities of a label, guaranteeing that the ground truth is among the outputs [11]. Such light instrumentation is desirable for the query optimization process; quantifying the uncertainty should not dominate the query latency time. This work investigates whether and how such methods can be utilized for hybrid query plans.

2. Optimization Based on Uncertainty. Given that an oracle exists to ensure accurate uncertainty estimate for one *ML* operator or query sub-plan, we also explore how to obtain a precise estimate for a whole physical plan that uses a mix *ML* and traditional *DB* operators. Intuitively, the uncertainty estimation can be regarded as error probability. Assuming that the error of *DB* operators is 0, estimating a physical plan's uncertainty can perhaps be computed as a chain of prior probabilities.

3.2. Continuous Training and Inference

Knowledge graphs are highly dynamic, both in terms of newly ingested data, but also in terms of schema changes [4, 5]. Relying on ML models for querying capabilities cannot overlook the overhead introduced by training and evaluating the deployed models. Whether training happens as a background or periodic mechanism triggered by concept drift monitors, querying a DBMS should always consult the freshest data with low latency. Newly ingested data may not have been used to train the current models backing the *ML* operators. However, due to the non-euclidean nature of graphs, our system could continue to offer predicted outcomes even for new data points up to a certain extent and with a caveat.

We exemplify this trade-off using the heterogeneous graph depicted in Figure 1a. We assume a transductive link prediction model that assists the traversal *ML* operator was trained using nodes 1 to 5. In contrast, nodes 6, 7, and 8 were ingested after training. Query 1 traverses a two-hop neighborhood (i.e., nodes 2, 1, 4) in the graph that was covered during training; therefore, the link prediction model can cater to this query. However, query 2 reaches nodes and connections that the model has not seen (e.g., node 7). Here, the system can offer predictions only for a sub-plan of the query while being forced to follow explicit links for the rest.

Figure 1b showcases the trade-off between completeness and latency. Querying new data cannot use inference-based methods, denoted as *ML* operators, until the models are updated and, therefore, need to rely on expensive traversals (i.e., *DB* operators) that result in incomplete results. Transductive graph ML models are especially affected by this compromise, as they need to build embeddings for all the entities at training time. On the other hand, inductive approaches such as GraphSage can mitigate the gap by building embeddings on-the-fly for new data but still suffer from concept drift that requires re-training [12]. This work will review low-latency training paradigms and investigate the trade-off introduced by combining *DB* and *ML* operators.

4. Related Work

Closely related to this PhD project, a recent report introduced the definition of Neural Graph Databases, which tackle the incompleteness assumption of large KGs through ML inference powered by Graph Representation Learning [13]. Our work builds on this concept by considering error-guided query optimization techniques. Furthermore, vector databases such as Milvus [14] are designed and optimized to maintain domain-agnostic vector representations of complex data. They offer fast access to the embeddings and efficient similarity searches in learned latent space. Our design requires generating and maintaining embeddings, potentially benefiting from vector databases for fast retrieval and distance-based queries in latent spaces. Moreover, our work is closely intertwined with approximate query processing (AQP) [10] Recent research enabled by learned models shows potential in overcoming the performance of sample-based AQP and achieves fast, accurate approximations using a small memory footprint [15]. Similar to AQP methods, this work aims to approximate graph queries while providing mechanisms to estimate the approximation error. Finally, our goals are akin to systems that offer logical reasoning capabilities over knowledge graphs, such as Vadalog [16]. However, this work features reasoning capabilities through ML models with bounded state size instead of computationally expensive rule-based approaches.

5. Conclusion

This paper described research directions and challenges of empowering graph databases with inference capabilities. We presented a potential query optimization strategy guided by user-defined error bounds and delved into the trade-offs to achieve highly accurate and complete results while minimizing query latency. Lastly, we discussed the proposed PhD project in the context of existing literature. As a next step, we plan to develop a prototype of the hybrid query optimizer and integrate it into open-source graph database efforts.

Acknowledgments

This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the internet topology, in: SIGCOMM, ACM, 1999, pp. 251–262.

[2] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, M. T. Özsu, The ubiquity of large graphs and surprising challenges of graph processing: extended survey, VLDB J. 29 (2020) 595–618.

[3] R. Mavlyutov, C. Curino, B. Asipov, P. Cudre-Mauroux, Dependency-driven analytics: A compass for uncharted data oceans., in: CIDR, 2017.

[4] S. Horchidan, P. Carbone, ORB: Empowering graph queries through inference, 2023. First International Workshop on Data Management for Knowledge Graphs.

[5] M. Lissandrini, D. Mottin, K. Hose, T. B. Pedersen, Knowledge graph exploration systems: are we lost?, in: CIDR, www.cidrdb.org, 2022.

[6] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37.

[7] A. J. Dudzik, P. Veličković, Graph neural networks are dynamic programmers, Advances in Neural Information Processing Systems 35 (2022) 20635–20647.

[8] H. Ren, W. Hu, J. Leskovec, Query2box: Reasoning over knowledge graphs in vector space using box embeddings, in: ICLR, OpenReview.net, 2020.

[9] X. Shi, Z. Zheng, Y. Zhou, H. Jin, L. He, B. Liu, Q. Hua, Graph processing on gpus: A survey, ACM Comput. Surv. 50 (2018) 81:1–81:35.

[10] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, I. Stoica, Blinkdb: queries with bounded errors and bounded response times on very large data, in: EuroSys, ACM, 2013, pp. 29–42.

[11] V. Vovk, A. Gammerman, G. Shafer, Algorithmic learning in a random world, Springer Science & Business Media, 2005.

[12] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017, pp. 1024–1034.

[13] H. Ren, M. Galkin, M. Cochez, Z. Zhu, J. Leskovec, Neural graph reasoning: Complex logical query answering meets graph databases, arXiv preprint arXiv:2303.14617 (2023).

[14] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Y. Cai, Z. Li, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, C. Xie, Milvus: A purpose-built vector data management system, in: SIGMOD Conference, ACM, 2021, pp. 2614–2627.

[15] Q. Ma, A. M. Shanghooshabad, M. Almasi, M. Kurmanji, P. Triantafillou, Learned approximate query processing: Make it light, accurate and fast, in: CIDR, www.cidrdb.org, 2021.

[16] L. Bellomarini, E. Sallinger, G. Gottlob, The Vadalog system: Datalog-based reasoning for knowledge graphs, Proc. VLDB Endow. 11 (2018) 975–987.