

# The $R^3$ Metric: Measuring Performance of Link Prioritization during Traversal-based Query Processing

Ruben Eschauzier<sup>1</sup>, Ruben Taelman<sup>1</sup> and Ruben Verborgh<sup>1</sup>

<sup>1</sup>IDLab, Department of Electronics and Information Systems, Ghent University – imec

## Abstract

The decentralization envisioned for the current centralized web requires querying approaches capable of accessing multiple small data sources while complying with legal constraints related to personal data, such as licenses and the GDPR. Link Traversal-based Query Processing (LTQP) is a querying approach designed for highly decentralized environments that satisfies these legal requirements. An important optimization avenue in LTQP is the order in which links are dereferenced, which involves prioritizing links to query-relevant documents. However, assessing and comparing the algorithmic performance of these systems is challenging due to various compounding factors during query execution. Therefore, researchers need an implementation-agnostic and deterministic metric that accurately measures the marginal effectiveness of link prioritization algorithms in LTQP engines. In this paper, we motivate the need for accurately measuring link prioritization performance, define and test such a metric, and outline the challenges and potential extensions of the proposed metric. Our findings show that the proposed metric highlights differences in link prioritization performance depending on the queried data fragmentation strategy. The proposed metric allows for evaluating link prioritization performance and enables easily assessing the effectiveness of future link prioritization algorithms.

## Keywords

Link Traversal, Link Prioritization, Query Optimization

## 1. Introduction

Currently, web user data is stored in centralized data silos, which restricts innovation and poses privacy concerns [1]. Decentralized efforts like Solid distribute user data across multiple personal data stores, which are highly personal and permissioned, requiring a querying approach that effectively handles decentralized and permissioned data.

Link Traversal-based Query Processing (LTQP) is an integrated querying approach designed to meet these requirements [2]. Starting from *seed* data sources, the LTQP query engine dynamically discovers new data sources by following hyperlinks found in previously discovered sources.

However, LTQP is currently slower compared to centralized alternatives. An optimization avenue is link prioritization, where the engine dynamically determines the order in which links should be dereferenced based on their expected relevance to the query [3, 4].

To optimize LTQP prioritization strategies, researchers need insights into the marginal performance of proposed algorithms and the theoretically optimal performance these strategies can achieve.

Previous studies have used metrics like result arrival times, total execution time, and other arrival time-based metrics [5] to assess prioritization strategies [3, 4, 6]. These measures fail to capture how close implemented prioritization approaches are to the theoretically optimal strategy. Additionally, these metrics fall short when comparing different engine implementations and environments due to unrelated differences that may skew link prioritization performance results:

- **Programming Languages:** Different languages can substantially affect engine performance [7], hindering purely time-based comparisons of algorithms.

AMW 2024: 16th Alberto Mendelzon International Workshop on Foundations of Data Management, September 30th–October 4th, 2024, Mexico City, Mexico

✉ [ruben.eschauzier@ugent.be](mailto:ruben.eschauzier@ugent.be) (R. Eschauzier); [ruben.taelman@ugent.be](mailto:ruben.taelman@ugent.be) (R. Taelman); [ruben.verborgh@ugent.be](mailto:ruben.verborgh@ugent.be) (R. Verborgh)

🌐 <https://www.rubensworks.net/> (R. Taelman); <https://ruben.verborgh.org/> (R. Verborgh)

🆔 0000-0002-6475-806X (R. Eschauzier); 0000-0001-5118-256X (R. Taelman); 0000-0002-8596-222X (R. Verborgh)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

- **Dereferencing Overhead:** Network communication times vary widely [8, 9], introducing significant noise into execution times, even for comparisons of the same engine.
- **Orthogonal Optimization Strategies:** Different query optimization [10] strategies might create differences in performance independent of prioritization strategies.

This leads to our research questions: *How can we compare the marginal algorithmic performance of prioritization algorithms in an implementation-agnostic manner* and *How can we define an Oracle baseline that represents optimal prioritization performance?*

In this paper, we introduce the metric: Relevant Retrieval Ratio ( $R^3$ ), pronounced r-cubed. This metric allows for straightforward observation of the algorithmic performance of prioritization approaches, independent of implementations, and provides a benchmark against the theoretically optimal prioritization strategy.

## 2. Existing Metrics

Existing LTQP and federated querying metrics [11] are insufficient for measuring the marginal algorithmic performance of link prioritization algorithms:

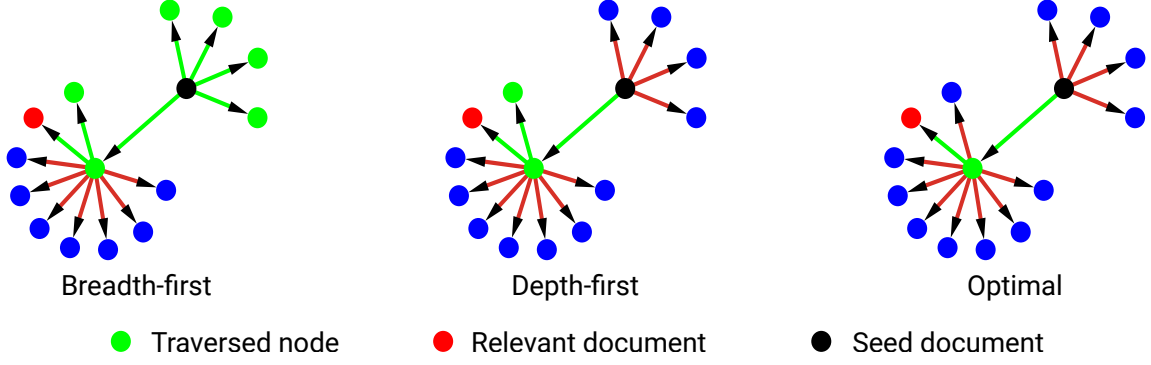
- **Query execution time:** Prioritization algorithms are unlikely to improve total query execution time [4, 3]. However, Link prioritization can improve [4, 3] arrival times of the first  $k$  results. However, the arrival times are influenced by both prioritization strategies and other unrelated engine optimizations, obscuring the prioritization algorithm’s effect.
- **Diefficiency Metrics:** Diefficiency metrics measure the continuous efficiency of query engines by reflecting the density of produced results over a given time interval [5]. This is in contrast to measuring the first  $k$  arrival times, which only measures engine performance at discrete points. However, the same limitations remain, as these metrics are based on result arrival times.
- **Number of HTTP Requests:** For federated query engines, the number of HTTP requests represents the efficiency of source selection and join planning during query execution [12, 13]. Federated engines often query SPARQL endpoints, allowing them to delegate parts of the query execution to the server [14]. In contrast, in LTQP, the engine uses HTTP requests to download documents, so the number of requests corresponds to the number of dereferenced documents. Thus, the number of HTTP requests measures the engine’s ability to prune irrelevant links. However, it does not indicate the effectiveness of *prioritization algorithms*, which reorder link dereference order, rather than pruning links.

The metric proposed in this paper complements existing metrics by measuring the marginal performance of link prioritization algorithms rather than the overall performance of the LTQP system.

## 3. Relevant Retrieval Ratio

The subweb [15] traversed during LTQP can be represented as a directed graph, where an edge from document A to document B indicates that a URI pointing to document B was first discovered in the data obtained from document A. Figure 1 shows an example of such a directed graph.

During LTQP query execution, an optimal link dereference order that finds all documents needed to answer a query exists, as shown in the right-most directed graph in Figure 1. This theoretical optimum can be considered an oracle link prioritization algorithm with perfect information on the location of query-relevant documents. The optimal path can always be achieved, as the oracle only uses links between documents that an engine could also use. An engine should strive to achieve a traversal path close to this optimal path. As such, we use the ratio between the optimal traversal path length and the actual traversal path length during query execution to evaluate the performance of link prioritization algorithms. The optimal traversal path is defined as the shortest path needed to dereference all documents required for the complete query result and is determined retrospectively.



**Figure 1:** An example subweb [15] showing the differences in traversal paths between breadth-first and depth-first methods, as well as the optimal traversal path.

Given the length of the optimal traversal path and the actual traversal path, we define the  $R^3$  metric as

$$R^3 = \frac{|T_O|}{|T_E|}, \quad |T_O|, |T_E| > 0 \quad (1)$$

where  $|T_E|$  is the length of the engine traversal path and  $|T_O|$  is the length of the optimal path. In this definition, a higher value is better, with  $R^3 = 1$  indicating optimal algorithmic link prioritization performance. Note that when  $|T_O| = 0$  or  $|T_E| = 0$ , our query has no results and the notion of optimal link prioritization does not exist. When an engine performs poorly, resulting in smaller differences in the metric's values, taking the  $\log_2$  can help visualize relative differences.

Using the  $R^3$  metric, we can compare the algorithmic efficiency of the two traversal algorithms in our example. Using breadth-first search, the path to dereference the query-relevant document contains six nodes, while the optimal traversal order contains two. Thus, the  $R^3$  metric for breadth-first traversal equals 0.33. Depth-first visits only three nodes, yielding an  $R^3$  value of 0.67. From this analysis, we can conclude that depth-first traversal outperforms breadth-first traversal on this subweb.

To find the optimal traversal path to dereference all query-relevant documents, we notice that it is equivalent to solving the directed Steiner tree problem [16] on our directed graph, with the query-relevant documents serving as terminals. As such, we can use existing Steiner tree solvers to efficiently find our optimal traversal path.

## 4. Results

Our experiment uses the Discover queries from the SolidBench benchmark [3]. To track the required information for computing the metric during query execution, we use the query engine Comunica [17]. To compute the optimal path, we will use a heuristic Steiner tree solver [18] to speed up computations. Although an exact solver would be ideal, it significantly increases computational complexity for large subwebs [19, 20].

Our experiments compare depth and breadth-first prioritization [4] using  $R^3$  and time until the final query result. We aim to validate the substantiated assumption that Discover queries do not benefit from improved link prioritization [21].

Table 1 shows the computed metrics per template instantiation and the difference in the  $\log_2$  of the metric value. We find that either method can outperform the other based on the used template. Furthermore, there is a significant difference in performance depending on the template instantiation, likely due to the different fragmentation strategies used in SolidBench [3]. Some pods store query-relevant data in a single file, while others fragment these into directories, complicating optimal traversal.

The differences in prioritization performance are not observed in time until the last result; the Pearson correlation coefficients between  $R^3$  and time until the last result are -0.042 and 0.201 for depth and breadth-first, respectively. These low correlations suggest a weak link between  $R^3$  and query execution time, possibly caused by noise in execution time.

Using the  $R^3$ , we confirm our previous paper [21] by computing a metric value instead of an engine-specific in-depth analysis of the query execution progress.

**Table 1**

The  $R^3$  for depth and breadth-first prioritization (DF, BF), separated by query template and instantiation. **D1.2** = template one, instantiation two.

Query	DF	BF	$\Delta \log_2(R^3)$	Query	DF	BF	$\Delta \log_2(R^3)$
<b>D1.0</b>	0.23	0.08	1.617	<b>D5.0</b>	0.13	0.15	-0.134
<b>D1.1</b>	0.50	0.50	0.000	<b>D5.1</b>	0.50	0.50	0.000
<b>D1.2</b>	0.50	0.50	0.000	<b>D5.2</b>	0.33	0.67	-1.000
<b>D1.3</b>	0.75	0.75	0.000	<b>D5.3</b>	0.04	0.03	0.250
<b>D1.4</b>	0.18	0.18	0.000	<b>D5.4</b>	0.18	0.16	0.196
<b>D2.0</b>	0.50	0.51	-0.031	<b>D6.0</b>	0.53	0.53	0.000
<b>D2.1</b>	0.50	0.67	-0.415	<b>D6.1</b>	0.40	0.40	0.000
<b>D2.2</b>	0.50	0.67	-0.415	<b>D6.2</b>	0.40	0.40	0.000
<b>D2.3</b>	0.90	0.90	0.000	<b>D6.3</b>	0.13	0.13	0.000
<b>D2.4</b>	0.44	0.56	-0.349	<b>D6.4</b>	0.73	0.46	0.690
<b>D3.0</b>	0.50	0.48	0.074	<b>D7.0</b>	0.05	0.05	0.000
<b>D3.1</b>	0.91	0.95	-0.058	<b>D7.1</b>	0.40	0.40	0.000
<b>D3.2</b>	0.93	0.93	-0.000	<b>D7.2</b>	0.40	0.40	0.000
<b>D3.3</b>	0.62	0.62	0.000	<b>D7.3</b>	0.15	0.15	0.000
<b>D3.4</b>	0.61	0.61	0.000	<b>D7.4</b>	0.06	0.03	0.955
<b>D4.0</b>	0.09	0.06	0.520	<b>D8.0</b>	0.45	0.64	-0.500
<b>D4.1</b>	0.44	0.44	0.000	<b>D8.1</b>	-	-	-
<b>D4.2</b>	0.50	0.44	0.170	<b>D8.2</b>	0.47	1.00	-1.100
<b>D4.3</b>	0.03	0.03	0.000	<b>D8.3</b>	-	-	-
<b>D4.4</b>	0.11	0.11	0.000	<b>D8.4</b>	0.36	0.43	-0.263

## 5. Conclusion

The current definition and implementation of the Relevant Retrieval Ratio ( $R^3$ ) have several limitations. First, due to the computational complexity of the Steiner tree problem for graphs and the lack of readily available exact solvers that work on directed graphs, our implementations use heuristics, thus leading to potentially suboptimal traversal path lengths. Second, the metric can not be computed when a query produces no results, either due to a timeout or no results existing for the query. Finally, our metric uses theoretically optimal paths. In practice, document dereference times can vary, making the theoretically optimal path potentially suboptimal. In future work, more extensive benchmarking of the metric is required to validate its effectiveness in measuring prioritization performance. Furthermore, a new metric that includes a penalty term for HTTP request time would account for real-world uncertainties in LTQP scenarios. Finally, an  $R^3$  metric for the first  $k$  results can be defined.

## 6. Acknowledgments

This research was supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10). Ruben Taelman is a postdoctoral researcher at the Research Foundation – Flanders (FWO).

## References

- [1] R. Verborgh, A data ecosystem fosters sustainable innovation, 2020.
- [2] R. Verborgh, The web's data triad, 2024. URL: <https://ruben.verborgh.org/blog/2024/05/30/the-webs-data-triad/>.
- [3] R. Taelman, R. Verborgh, Link traversal query processing over decentralized environments with structural assumptions, in: International Semantic Web Conference, Springer, 2023, pp. 3–22.
- [4] O. Hartig, M. T. Özsu, Walking without a map: Ranking-based traversal for querying linked data, in: The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15, Springer, 2016, pp. 305–324.
- [5] M. Acosta, M.-E. Vidal, Y. Sure-Vetter, Diefficiency metrics: measuring the continuous efficiency of query processing approaches, in: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part II 16, Springer, 2017, pp. 3–19.
- [6] J. Hanski, R. Taelman, R. Verborgh, Observations on bloom filters for traversal-based query execution over solid pods (2024).
- [7] M. Fourment, M. R. Gillings, A comparison of common programming languages used in bioinformatics, BMC bioinformatics 9 (2008) 1–9.
- [8] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, Tuning red for web traffic, ACM SIGCOMM Computer Communication Review 30 (2000) 139–150.
- [9] K. KOBAYASHI, T. KATAYAMA, Analysis and evaluation of packet delay variance in the internet, IEICE transactions on communications 85 (2002) 35–42.
- [10] O. Hartig, Zero-knowledge query planning for an iterator implementation of link traversal based query execution, in: Extended Semantic Web Conference, Springer, 2011, pp. 154–169.
- [11] G. Montoya, M.-E. Vidal, O. Corcho, E. Ruckhaus, C. Buil-Aranda, Benchmarking federated sparql query engines: Are existing testbeds enough?, in: The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11–15, 2012, Proceedings, Part II 11, Springer, 2012, pp. 313–324.
- [12] P. Peng, Q. Ge, L. Zou, M. T. Özsu, Z. Xu, D. Zhao, Optimizing multi-query evaluation in federated rdf systems, IEEE Transactions on Knowledge and Data Engineering 33 (2019) 1692–1707.
- [13] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, T. Tran, Fedbench: A benchmark suite for federated semantic data query processing, in: The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I 10, Springer, 2011, pp. 585–600.
- [14] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, Fedx: Optimization techniques for federated query processing on linked data, in: The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I 10, Springer, 2011, pp. 601–616.
- [15] O. Hartig, J.-C. Freytag, Foundations of traversal based query execution over linked data, in: Proceedings of the 23rd ACM conference on Hypertext and social media, 2012, pp. 43–52.
- [16] L. Zosin, S. Khuller, On directed steiner trees, in: SODA, volume 2, 2002, pp. 59–63.
- [17] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a modular sparql query engine for the web, in: The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17, Springer, 2018, pp. 239–255.
- [18] D. Watel, M.-A. Weisser, A practical greedy approximation for the directed steiner tree problem, Journal of Combinatorial Optimization 32 (2016) 1327–1370.
- [19] F. K. Hwang, D. S. Richards, Steiner tree problems, Networks 22 (1992) 55–89.
- [20] A. Lucas, Ising formulations of many np problems, Frontiers in physics 2 (2014) 74887.
- [21] R. Eschauzier, R. Taelman, R. Verborgh, How does the link queue evolve during traversal-based query processing? (2023).