

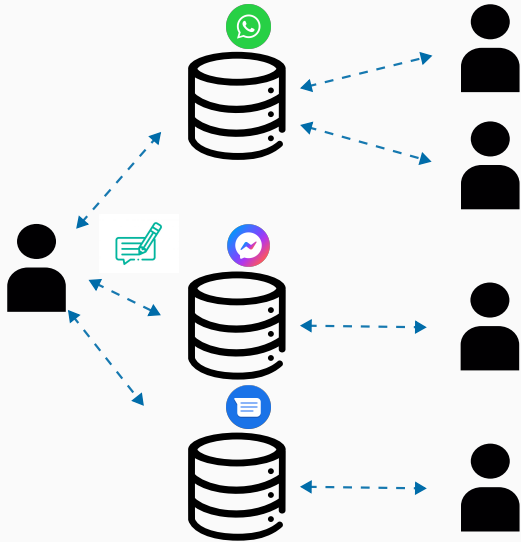
Revisiting Link Prioritization for Efficient Traversal in Structured Decentralized Environments

Ruben Eschauzier, Ruben Taelman, Ruben Verborgh

October 30, 2025

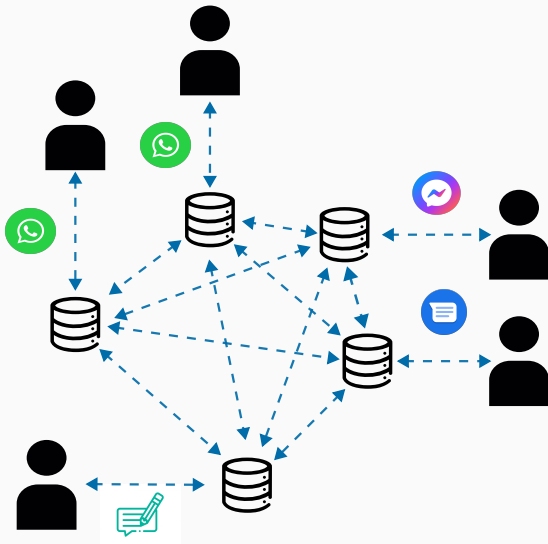
Department of Electronics and Information Systems, Ghent University Imec

The Motivation for Decentralized Personal Data Storage



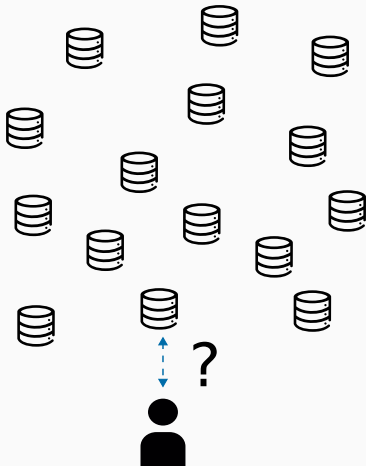
- Each application has its own data
- Stifles innovation
- Causes vendor lock-in

The Motivation for Decentralized Personal Data Storage



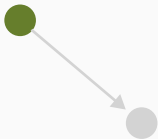
- Each application accesses a common decentralized data environment
- Easy to switch vendors
- Promotes innovation

Querying Structured Decentralized Environments



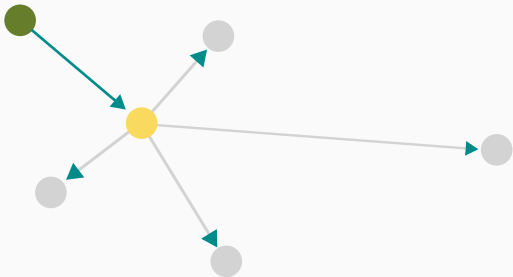
- How do we discover these sources?
- How do we ensure fine-grained access control?

Link Traversal: Seed Document



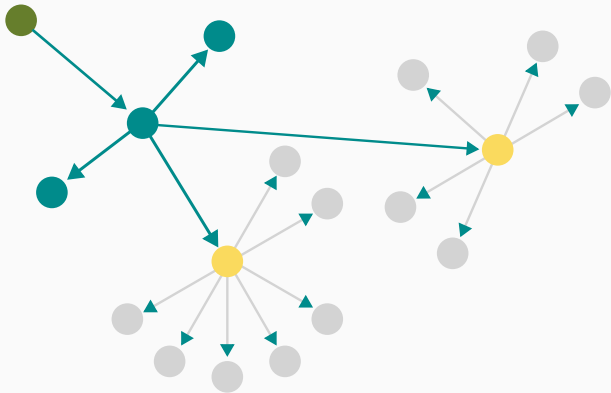
- Link Traversal starts from seed documents (URIs)
- These are provided by the user or specified in the query

Link Traversal: Traversal



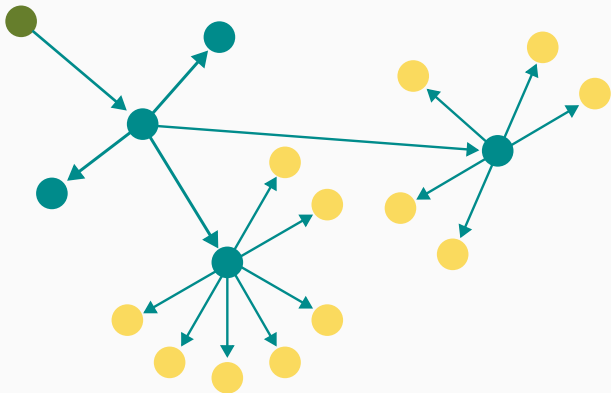
- New URIs are extracted from the seed document
- URIs are extracted in accordance with reachability criteria
- Uses the follow-your-nose principle

Link Traversal: Traversal



- New URIs are dereferenced and the process is repeated

Link Traversal: Termination

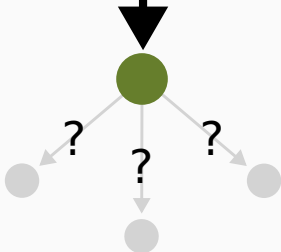


- This continues until all links are dereferenced
- Continuously produces results
- Can enforce fine-grained (document-level) access control

Query Optimization for Link Traversal

```
SELECT * WHERE {  
  <seedUri> <ex:p1> ?o1.  
  <seedUri> <ex:p2> ?o2.  
  ...  
}
```

Extract
Seed document

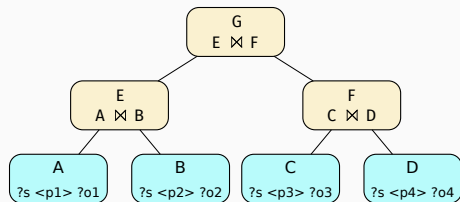


The query (partly) determines:

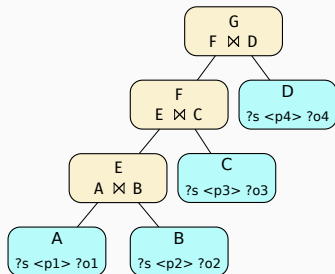
- The queried data
- The topology of the queried data
- The query-relevant documents

Result: Limited prior knowledge for query optimization

Query Optimization for Link Traversal: Traditional Query Planning

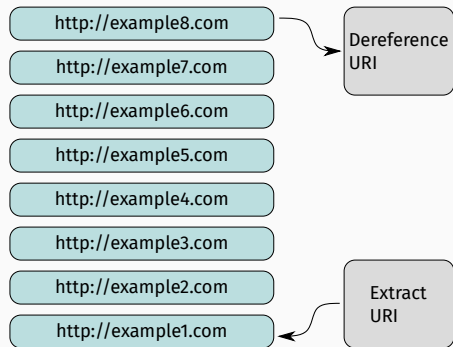


VS



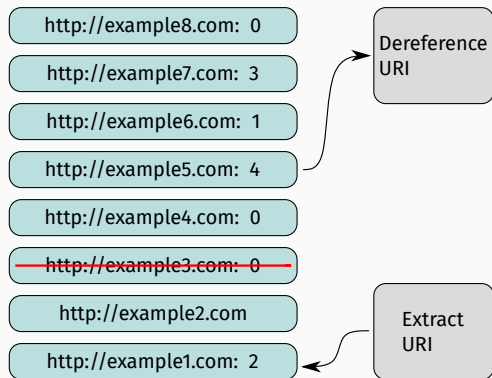
- Assume an optimize-then-execute paradigm
- Query optimization for link traversal involves traditional (zero knowledge) query planning

Query Optimization for Link Traversal: Traversal optimization



- URIs are put into a link queue
- Link traversal uses a FIFO queue by default

Query Optimization for Link Traversal: Traversal optimization



Optimizations:

- Prioritize query-relevant URIs
- Prune irrelevant URIs
- Pruning may cause results to be missed if no prior knowledge is available

Problem Statement

Evaluate the performance of existing link prioritization algorithms in structured decentralized environments

Problem Statement

- These algorithms are implemented in an engine that is no longer maintained
- The baseline performance depends on design choices orthogonal to the prioritization algorithm
- The baseline should isolate and measure the marginal effect of the prioritization strategy

R^3 Metric: Introduction

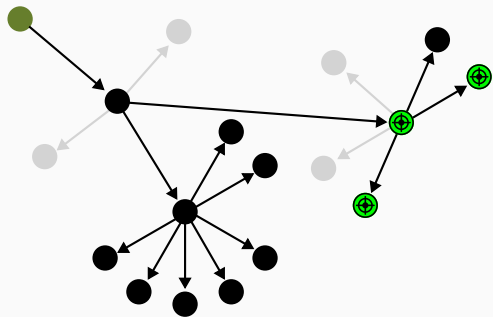


Figure 1: Actual traversal

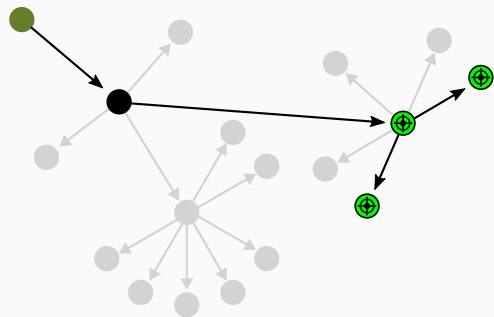


Figure 2: Optimal traversal: minimal cost path

$$R^3 = \frac{4}{13} = 0.31$$

Definition

$$R^3 = \frac{C(X)}{C(O_{\mathcal{T}})}$$

- X : the **optimal traversal order** (minimal-cost path)
- $O_{\mathcal{T}}$: traversal order produced by the link prioritization algorithm
- $C(\cdot)$: The total cost of all arcs in the traversal path

Interpretation

- $R^3 = 1$ perfect match with the optimal traversal
- $R^3 < 1$ deviation from optimal performance
- **Higher values indicate better prioritization quality.**

- Directed graph steiner tree over traversed directed graph $G = (V, A)$ with root r
- Query-relevant documents D_T serve as terminals $T \subseteq V$
- Find minimum cost sub-graph $X = (V', A')$ starting at root r and spanning all vertexes T .
- With cost: $C(X) = \sum_{a \in A'} c(a)$, with $c(a)$ the cost of an edge in the topology

Simulated Solid Environment: SolidBench (Taelman and Verborgh 2023)

- Data is stored in vaults simulating social media application

Data attributes

- Generates 158,233 RDF files
- Across 1,531 data vaults
- containing a total of 3,556,159 triples.

Analysed queries

- Use discover queries

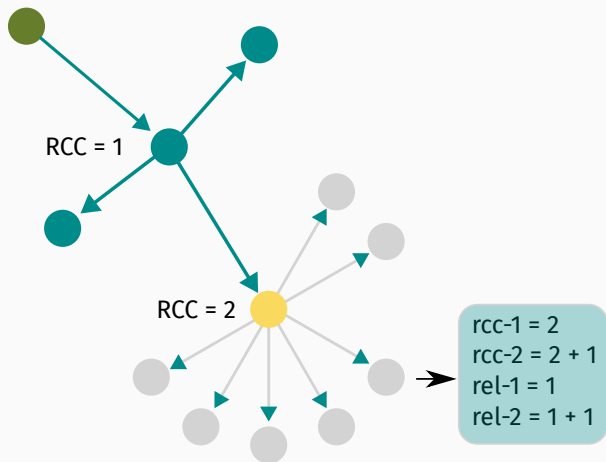
Non-adaptive

- Breadth-first (default), depth-first, random prioritization

Graph-based

- In-degree, PageRank score

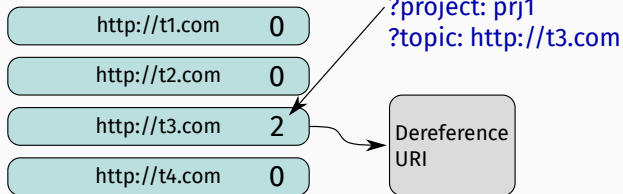
Prioritization algorithms: Result Contribution-Based (RCC)



- Nodes adaptively scored according to the number of results they contribute to
- Called *rcc-1*, *rcc-2*, *rel-1*, *rel-2*.

Prioritization algorithms: Intermediate Results-based

```
SELECT ?person ?project ?topic
WHERE {
  ?person ex:worksFor ?company .
  ?person ex:worksOn ?project .
  ?project ex:hasTopic ?topic .
}
```



- Nodes adaptively scored according to their contribution to intermediate results
- *IS* with initial priorities of 0, while *ISdcr* sets priority to the priority of the parent node - 1

Prioritization algorithms

Hybrid

- Multiply intermediate result and RCC-based scoring functions
- *is-rcc1*, *is-rcc2*, *is-rel1*, *is-rel2*

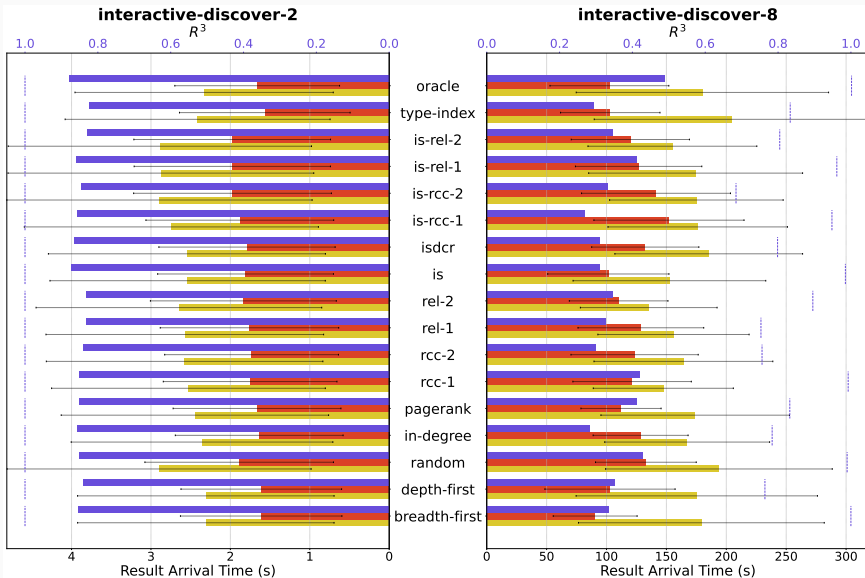
TypeIndex

- The typeIndexe point to location for resource of specific type
- Prioritize TypeIndex first

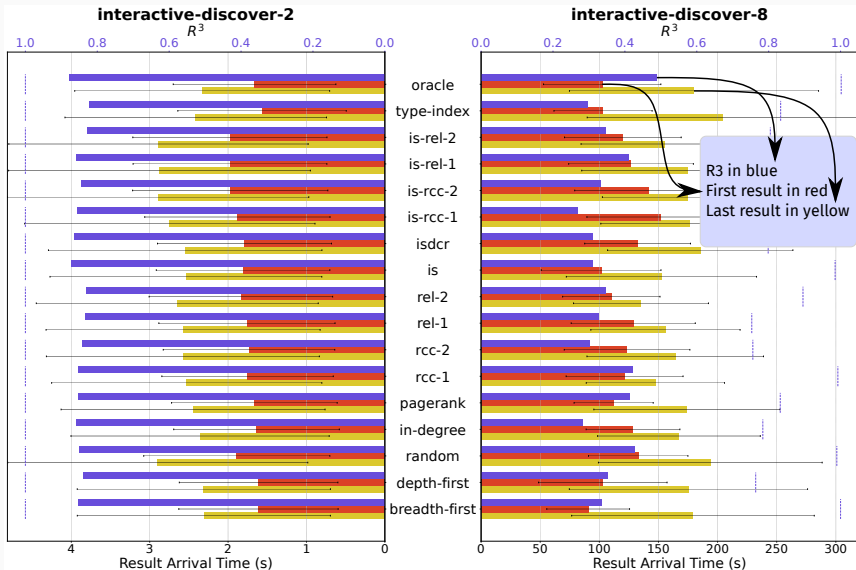
Oracle

- Compute RCC in hindsight
- Scores are propagated through the shortest path
- Serves as optimal performance oracle

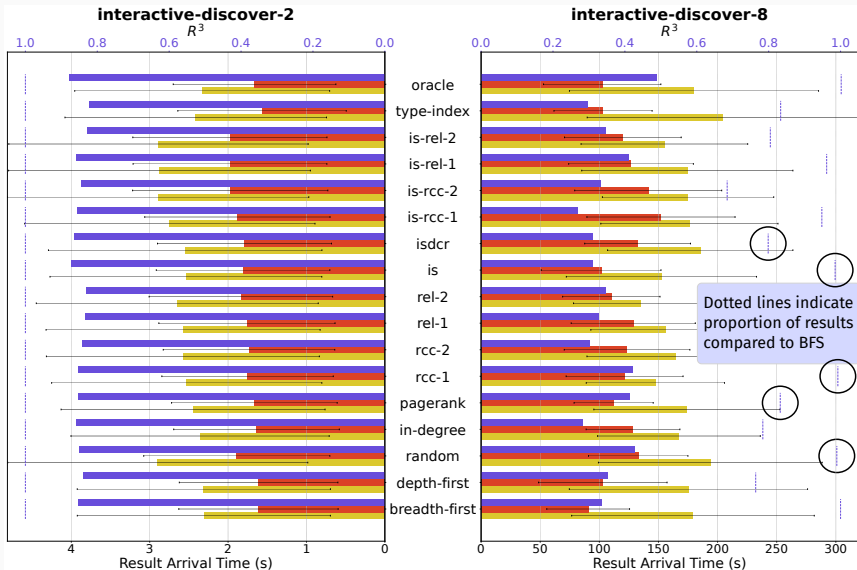
Prioritization Has Limited Impact on Result Arrival Time



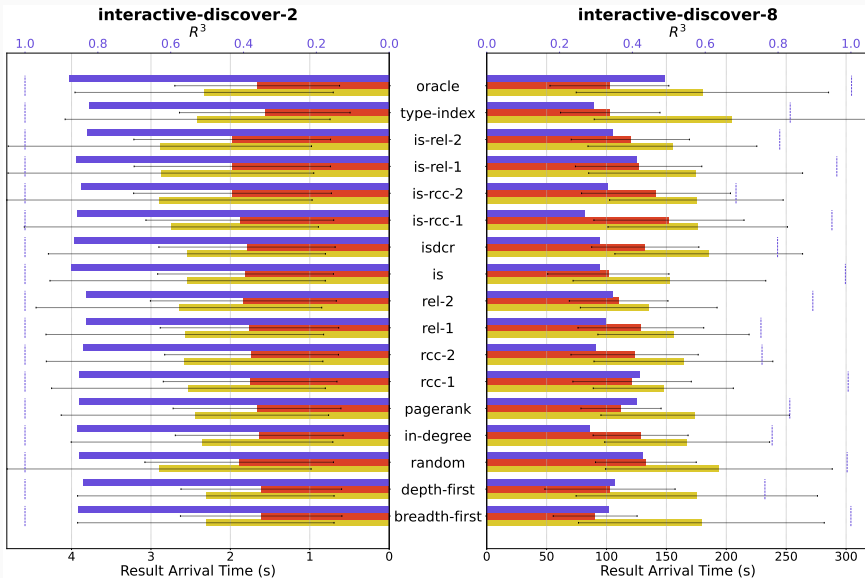
Prioritization Has Limited Impact on Result Arrival Time



Prioritization Has Limited Impact on Result Arrival Time



Prioritization Has Limited Impact on Result Arrival Time



Prioritization Has Limited Impact on Result Arrival Time





	<i>1st</i>		<i>Cmpl</i>		<i>R</i> ³	
	better	worse	better	worse	better	worse
depth-first	<u>15.7</u>	<u>17.1</u>	<u>14.3</u>	17.1	7.5	12.5
random	4.3	68.6	5.7	74.3	<u>15.0</u>	15.0
is-rcc-1	4.3	57.1	5.7	60.0	7.5	<u>5.0</u>
type-index	<u>15.7</u>	18.6	10.0	<u>15.7</u>	12.5	20.0
oracle	<u>18.6</u>	<u>11.4</u>	<u>18.6</u>	<u>12.9</u>	<u>25.0</u>	<u>2.5</u>

Table 1: Percentage of queries for which an algorithm performs at least 10% better or worse than the breadth-first baseline.

- No algorithm improves performance, and the oracle only marginally improves performance

Main takeaway: Link prioritization has limited performance benefit.

- Limited performance improvement from prioritizing links.
- Focus instead on:
 - **Query planning** Hanski et al. (2025)
 - **Traversal pruning** Tam et al. (2024)

-  Hanski, J. et al. (2025). **“Link Traversal over Decentralised Environments using Restart-Based Query Planning”**. In: *International Conference on Web Engineering*.
-  Hartig, O. and M. T. Özsu (2016). **“Walking without a map: Ranking-based traversal for querying linked data”**. In: *International Semantic Web Conference*. Springer, pp. 305–324.
-  Taelman, R. and R. Verborgh (2023). **“Link traversal query processing over decentralized environments with structural assumptions”**. In: *International Semantic Web Conference*. Springer, pp. 3–22.
-  Tam, B.-E. et al. (2024). **“Opportunities for Shape-based Optimization of Link Traversal Queries”**. In: *arXiv preprint arXiv:2407.00998*.

Supplemental Slides: Short Query

```
PREFIX snvoc: <www.ldbc.eu/ldbc_socialnet/1.0/vocabulary/>
SELECT DISTINCT ?creator ?messageContent WHERE {
  <pods/227/profile/card#me> snvoc:likes _:g_0.
  _:g_0 (snvoc:hasPost|snvoc:hasComment) ?message.
  ?message snvoc:hasCreator ?creator.
  ?otherMessage snvoc:hasCreator ?creator;
    snvoc:content ?messageContent.
}
LIMIT 10
```

Supplemental Slides: Short Query

```
PREFIX snvoc: <www.ldbc.eu/ldbc_socialnet/1.0/vocabulary/>
SELECT ?messageCreationDate ?messageContent WHERE {
  <pods/150/comments/Mexico> snvoc:id ?messageId;
    snvoc:creationDate ?messageCreationDate;
    (snvoc:content|snvoc:imageFile) ?messageContent.
}
```