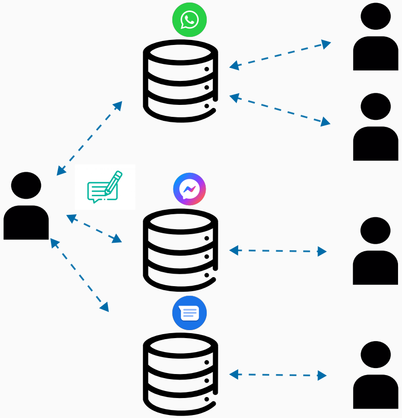
Revisiting Link Prioritization for Efficient Traversal in Structured Decentralized Environments

Ruben Eschauzier, Ruben Taelman, Ruben Verborgh

October 14, 2025

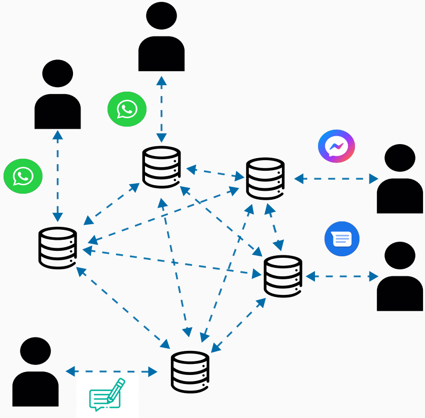
Department of Electronics and Information Systems, Ghent University Imec

The Need for Decentralized Personal Data Storage



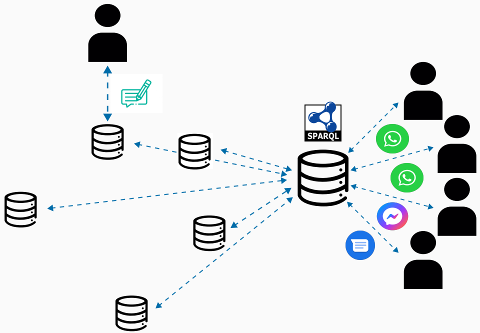
- Each application has its own data
- Stifles innovation
- Causes vendor lock-in

The Need for Decentralized Personal Data Storage



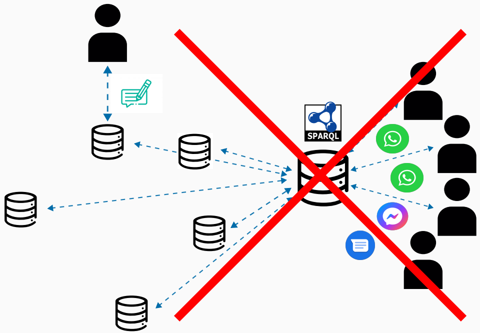
- Each application uses common data storage
- Easy to switch vendors
- Promotes innovation

The Problem with Centrally Aggregating and Querying



- Why not aggregate data and query it?
- Impossible in case of personal data due to privacy concerns

The Problem with Centrally Aggregating and Querying

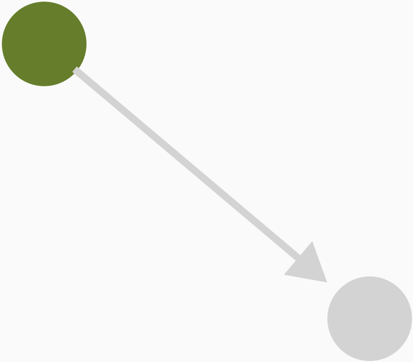


- Why not aggregate data and query it?
- Impossible in case of personal data due to privacy concerns

Link Traversal-based Query Processing

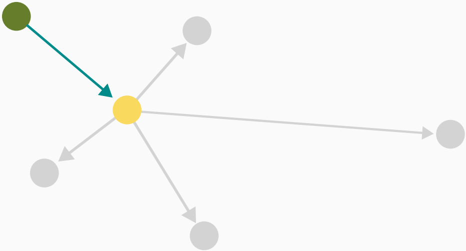
- Link Traversal iteratively dereferences data to query over
- Continuously produces results
- Can enforce fine-grained (document-level) access-control

Link Traversal: Seed Document

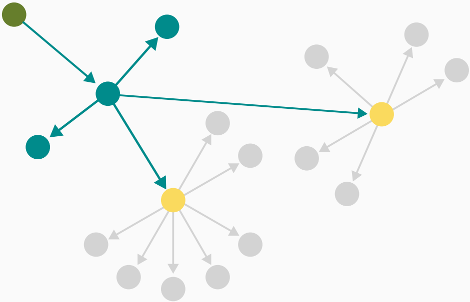


- Link Traversal starts from seed documents (URIs)
- These are provided by the user or in the query.

Link Traversal: Traversal

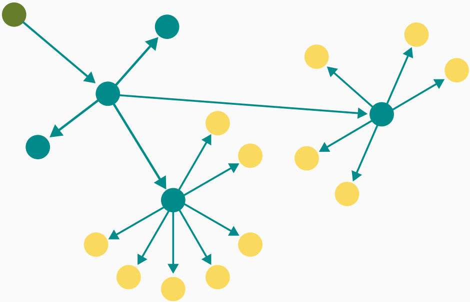


- New URIs are extracted from the seed document
- URIs are extracted in accordance with reachability criterions



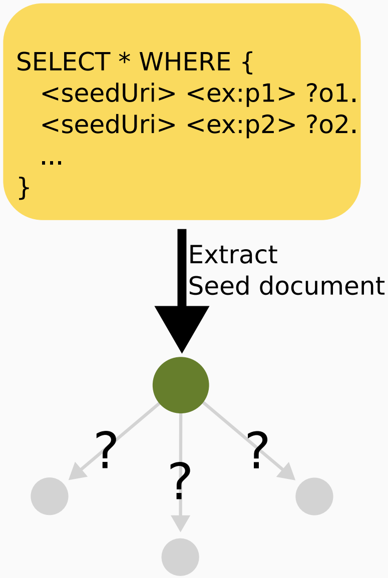
- New URIs are dereferenced and the process is repeated

Link Traversal: Termination



- This continues untill all links are dereferenced

Query Optimization for Link Traversal

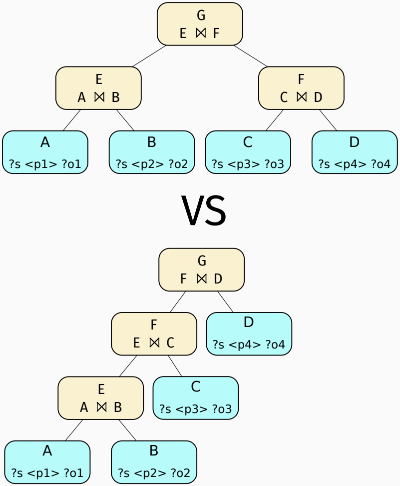


The query (partly) determines:

- 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

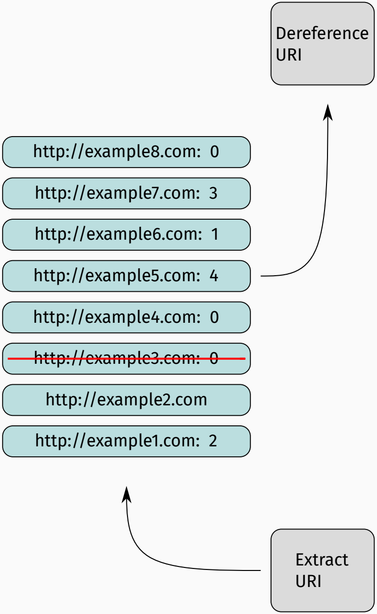


- Query optimization for link traversal involves traditional (zero knowledge) query planning

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 can lead to missing results without prior knowledge

Problem Statement

Investigate the performance of link prioritization algorithms in literature in new 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 measure marginal prioritization performance

R^3 metric: Motivation

Context
During link traversal, the engine traverses a fixed topology entirely

Goal
The goal of prioritization is to find query-relevant (where-provenance) documents as soon as possible.

Challenge
Find the optimal traversal order of a given query in hindsight to compare the traversal order taken by the engine to.

R^3 metric: Steiner trees

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

R^3 metric

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.

R^3 Metric Illustration

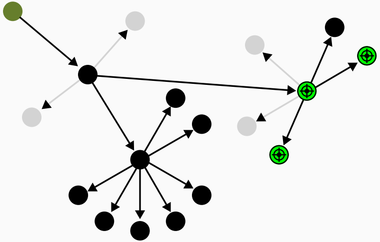


Figure 1: Actual traversal

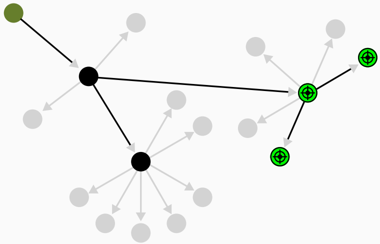


Figure 2: Optimal traversal: minimal cost path

$$R^3 = \frac{5}{13} = 0.38$$

Investigated Structered Decentralized Environment

Simulated Solid Environment: SolidBench Taelman and Verborgh 2023

- Data is stored in vaults
- Vaults use document-centric structure
- Vaults simulate a 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

Metrics Used

- Time until first result and last result
- Relative to breadth-first traversal baseline
- R^3

Prioritization algorithms (Hartig and Özsu 2016)

Non-adaptive

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

Graph-based

- In-degree, PageRank score

Result-based

- Uses result contribution count (RCC) of each node (URI)
- Priority equal to the sum / count of non-zero RCC of the 1 or 2-hop in-neighbours of a node
- Called rcc-1, rcc-2, rel-1, rel-2 respectively.

Prioritization algorithms

Intermediate-results

- Uses intermediate solutions in the engine
- Priority of URI equal to the largest intermediate result with that URI bound
- IS sets initial priorities to 0, while ISdcr sets priority to the priority of the parent node - 1

Hybrid

- Multiply intermediate and full result scoring functions
- is-rcc1, is-rcc2, is-rel1, is-rel2

TypeIndex

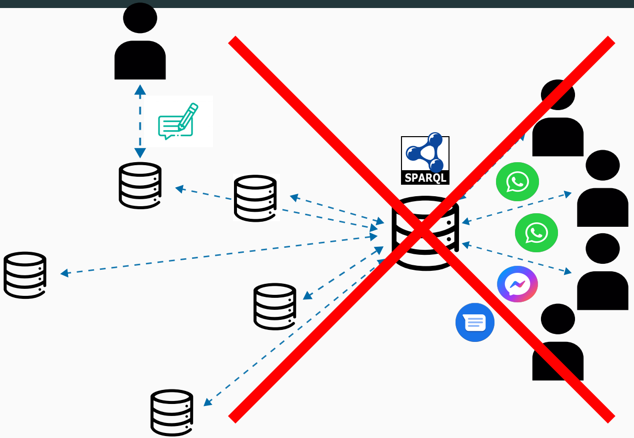
- TypeIndex points to location for resource of specific type
- Prioritize TypeIndex

Prioritization algorithms

Oracle

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

Results



- TODO: Insert image from paper here (discover 2 and 8)
- Breadth-first outperforms most algorithms
- Oracle has significantly better R^3, but not execution time

Results

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

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

Link Prioritization in Structured Decentralized Environments

- Link prioritization in Solid environment will not significantly improve query performance
- Research should instead focus on query planning Hanski et al. (2025) or 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.