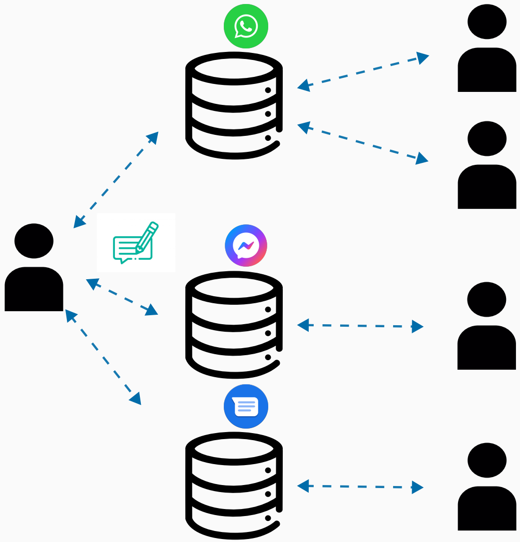
Revisiting Link Prioritization for Efficient Traversal in Structured Decentralized Environments

Ruben Eschauzier, Ruben Taelman, Ruben Verborgh

October 27, 2025

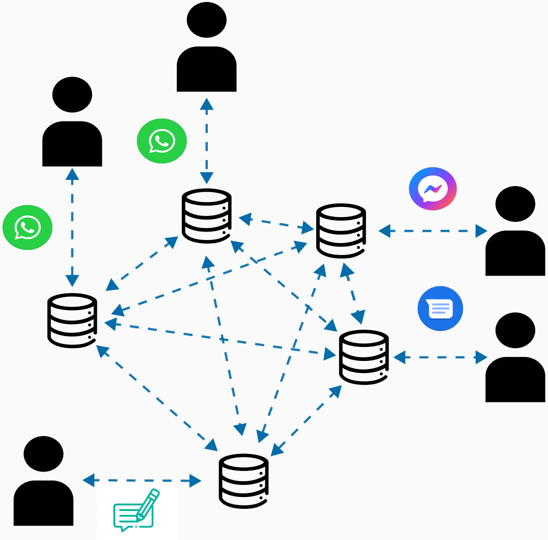
Department of Electronics and Information Systems, Ghent University Imec

The Need for Decentralized Personal Data Storage



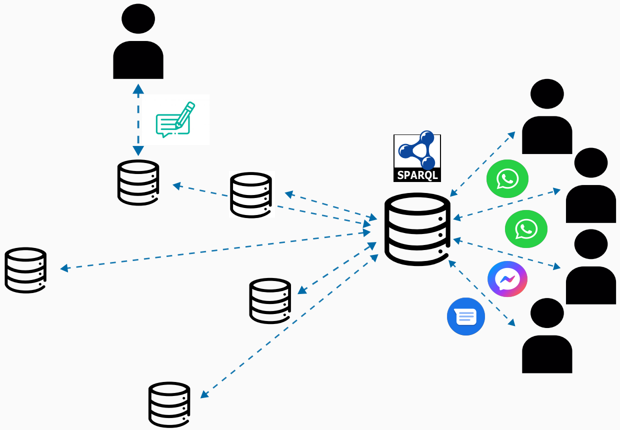
- Each application has its own data
- Stifles innovation
- Causes vendor lock-in

The Need for Decentralized Personal Data Storage



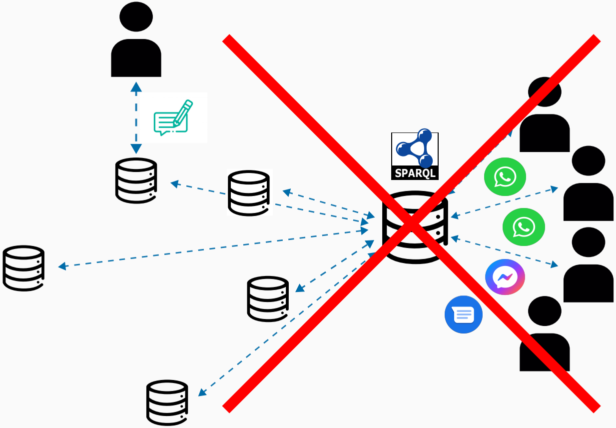
- Each application uses common data storage
- Easy to switch vendors
- Promotes innovation

The Problem with Centrally Aggregating and Querying



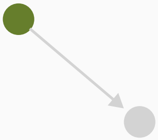
- Why not aggregate data and query it?
- Difficult in case of personal data due to privacy concerns

The Problem with Centrally Aggregating and Querying



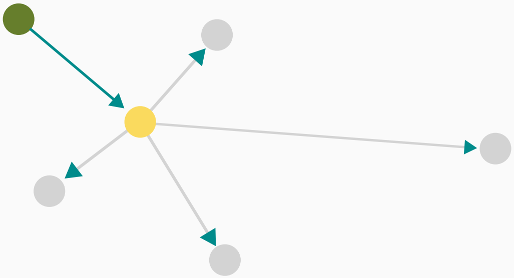
- Why not aggregate data and query it?
- Difficult in case of personal data due to privacy concerns

Link Traversal: Seed Document



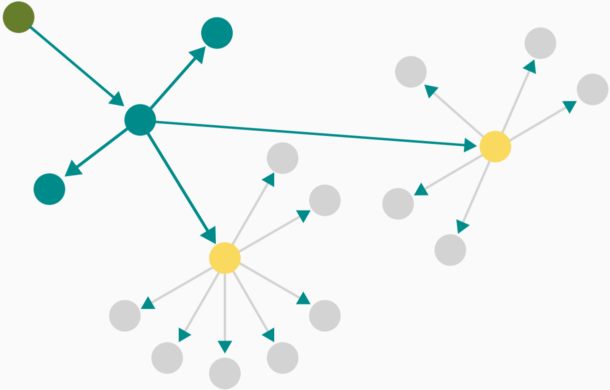
- Link Traversal starts from seed documents (URIs)
- These are provided by the user or in the query.

Link Traversal: Traversal



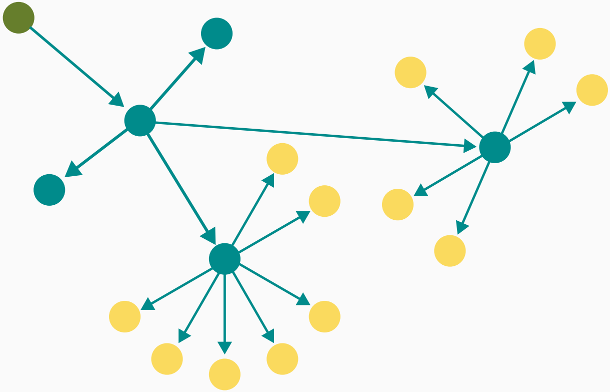
- New URIs are extracted from the seed document
- URIs are extracted in accordance with reachability criterions

Link Traversal: Traversal



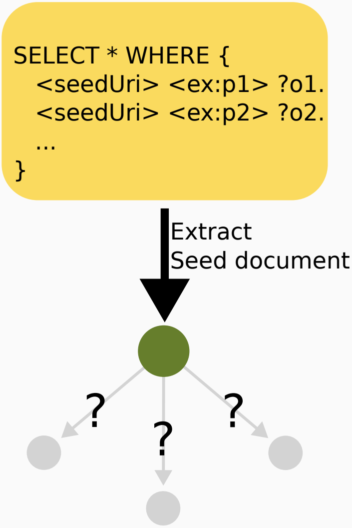
- New URIs are dereferenced and the process is repeated

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

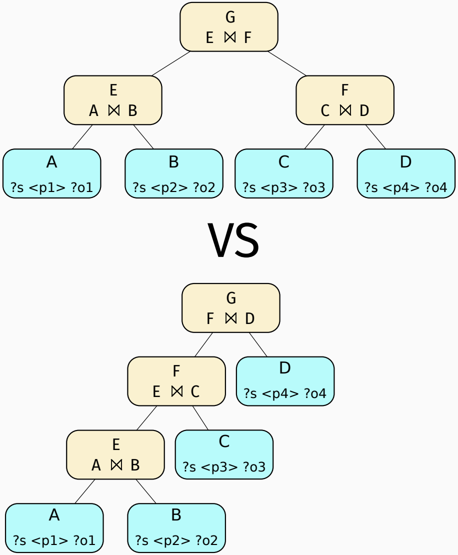


The query (partly) determines:

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

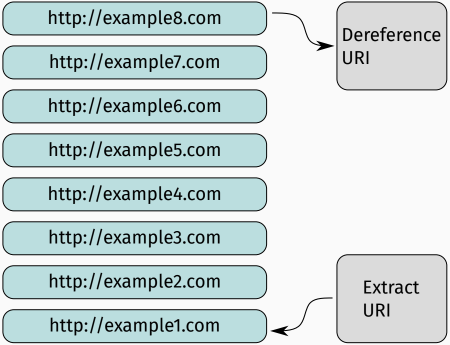
Result: limited prior knowledge for query optimization

Query Optimization for Link Traversal: Traditional Query Planning



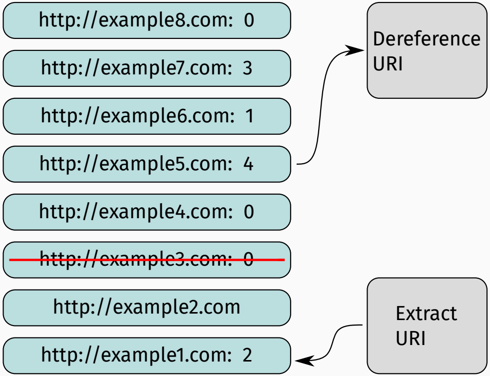
- Query optimization for link traversal involves traditional (zero knowledge) query planning
- Assume an optimize-then-execute

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: Introduction

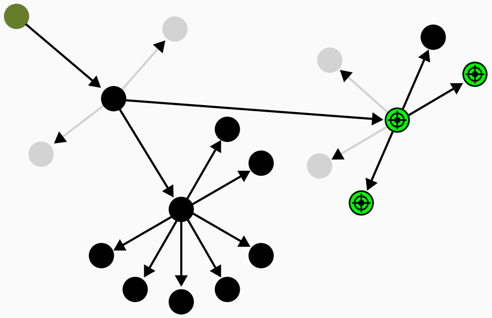


Figure 1: Actual traversal

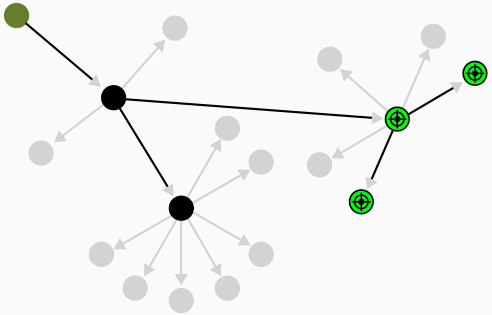


Figure 2: Optimal traversal: minimal cost path

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

R^3 metric: Definition

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: Steiner Trees

- 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

Investigated Structured 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

Prioritization algorithms (Hartig and Özsu 2016)

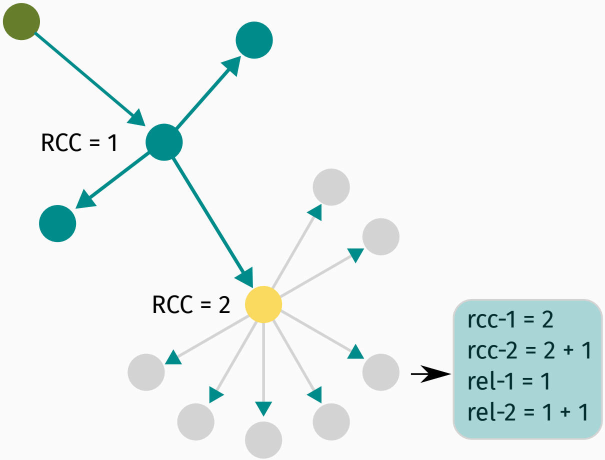
Non-adaptive

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

Graph-based

- In-degree, PageRank score

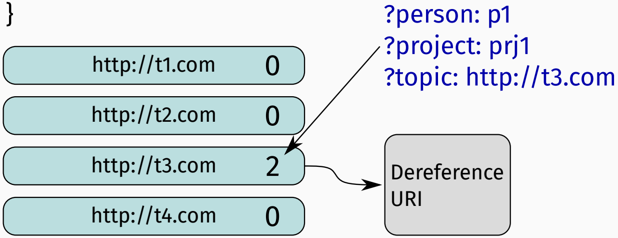
Prioritization algorithms: Result Contribution-Based (RCC)



- 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*

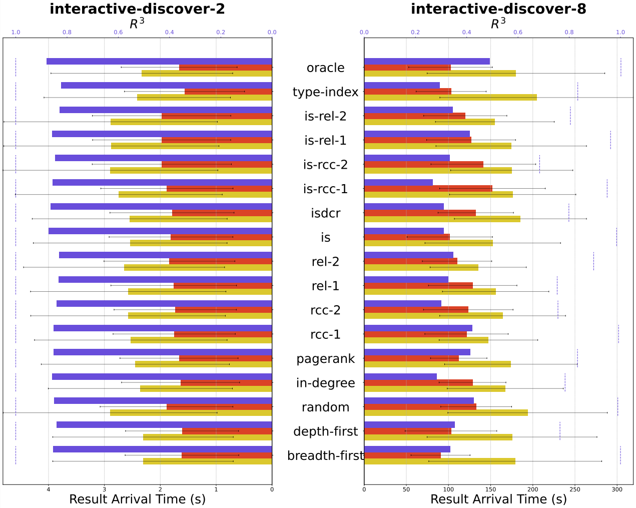
TypeIndex

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

Oracle

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

Prioritization Has Limited Impact on Result Arrival Time



- Breadth-first outperforms most algorithms
- Oracle has significantly better R^3, but not execution time

Prioritization Has Limited Impact on Result Arrival Time

	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*.