Welcome!
Information in Dynamic Web Pages
Information in Dynamic Web Pages
Support for such an incremental visualization has not received much attention in existing work on querying the Web of Data.
Let's rethink our optimization criteria for Web querying!

A case for **response time focused** query processing

Olaf Hartig
Dept. of Computer and Information Science, Linköping University, Sweden

@olafhartig
Terminology

- **Web querying**: queries directly over Web data sources
  - querying a federation of SPARQL endpoints
  - querying Linked Data on the Web (interface: URI lookups)
  - querying other types of Linked Data Fragment interfaces
  - etc.
Terminology

- **Web querying**: queries directly over Web data sources
  - querying a federation of SPARQL endpoints
  - querying Linked Data on the Web (interface: URI lookups)
  - querying other types of Linked Data Fragment interfaces
  - etc.

- **Query execution time (QET)**: time from issuing a query until the query execution process has been completed

- **Response time (RT)**: time from issuing a query until a specific portion of the query result has been produced
  - may be measured in terms of a *specific number* of result elements (i.e., solution mappings in the context of SPARQL)
  - or in terms of a *specific percentage* of result elements
Agenda

- Aiming to minimize QET is different from aiming to minimize RT
  - Evidence 1
  - Evidence 2

- Some of our work on RT-focused query processing
  - An attempt to optimize the response times of traversal-based query execution
  - An attempt to make the core fragment of SPARQL suitable for the task
Minimizing QET ≠ Minimizing RT

Evidence 1

Based on: Maribel Acosta, Maria-Esther Vidal, and York Sure-Vetter: Diefficiency Metrics: Measuring the Continuous Efficiency of Query Processing Approaches. ISWC 2017.
Executing a Query via a TPF Interface

Different client-side strategies to execute a given query over a dataset that can be accessed via a Triple Pattern Fragment (TPF) interface.

- nLDE Not Adaptive
- nLDE Selective
- nLDE Random
Minimizing QET ≠ Minimizing RT

Evidence 2

Linked Data Query Processing

• Focus: querying Linked Data live on the Web by relying only on the Linked Data principles
  - look up URIs to access original data \textit{at runtime}
Linked Data Query Processing

- Focus: querying Linked Data live on the Web by relying only on the Linked Data principles
  - look up URIs to access original data at runtime
- Queries
  - typically expressed using SPARQL (in practice, BGPs only)
  - reachability-based query semantics; i.e., scope of evaluation is virtual union of all data in a well-defined reachable subweb
Linked Data Query Processing

• Focus: querying Linked Data live on the Web by relying only on the Linked Data principles
  – look up URIs to access original data at runtime

• Queries
  – typically expressed using SPARQL (in practice, BGPs only)
  – reachability-based query semantics; i.e., scope of evaluation is virtual union of all data in a well-defined reachable subweb

• Traversal-based query execution
  – intertwines local result construction with a recursive traversal of (specific) data links
  – natural support of reachability-based query semantics (discovers reachable subweb at runtime)
Concrete Implementation Approach

Data Retrieval Operator

Triple Pattern Operator

Triple Pattern Operator

Triple pattern \((?v1, \text{knows}, ?v2)\)

Dispatcher
Data Retrieval Operator

GET http://example.org/...

RDF triple (Bob, knows, Alice)

Triple pattern (?v1, knows, ?v2)
Triple Pattern Operator

Intermediate Solution
Timestamp: 1
Bindings: ?v1 → Bob, ?v2 → Alice
Flags: [・|√|・|・]

RDF triple
( Bob, knows, Alice )

Triple pattern
( ?v1, knows, ?v2 )
Intermediate Solution
Timestamp: 1
Bindings: \(?v1 \rightarrow Alice, \ ?v2 \rightarrow Bob\)
Flags: \([\cdot|\checkmark|\cdot|\cdot]\)
Triple Pattern Operator cont’d
Properties

- Supports any reachability-based query semantics
- Highly adaptive
  - Routing of intermediate solutions
  - Inspired by “Eddies” (Anvur & Hellerstein, SIGMOD 2000)
Hypothesis

Query execution time (QET) and response time (RT) can be reduced by applying a suitable routing policy.
Test with Different Routing Policies

- Data retrieval operator simply appends to its lookup queue
- Web simulation environment, diverse test queries (here one of them)
Test with Different Routing Policies

- Data retrieval operator simply appends to its lookup queue
- Web simulation environment, diverse test queries (here one of them)
- Each bar represents geometric mean of 5 independent executions

Response time for last reported solution, relative to overall QET
Response time for first reported solution, relative to overall QET
Test with Different Routing Policies

- Data retrieval operator simply appends to its lookup queue
- Web simulation environment, diverse test queries (here one of them)
- Each bar represents geometric mean of 5 independent executions

Response time for last reported solution, relative to overall QET
Response time for first reported solution, relative to overall QET

… is essentially the same for all executions of the query
Test with Different Routing Policies

- Data retrieval operator simply appends to its lookup queue
- Web simulation environment, diverse test queries (here one of them)
- Each bar represents geometric mean of 5 independent executions

Response time for last reported solution, relative to overall QET

Response time for first reported solution, relative to overall QET

Routing policy has no impact!
Hypothesis

No!

Why?

Query execution time (QET) and response time (RT) cannot be reduced by applying a suitable routing policy.
Another Test: Impact of Data Retrieval?

5 queries of the FedBench benchmark suite, executed over *real* Linked Data on the WWW
Another Test: Impact of Data Retrieval?

Different number of lookup threads used by the data retrieval operator

- 10 threads
- 20 threads

Data retrieval op. equipped with a cache

- Cache populated by a first execution
- Times measured for a 2nd, cache-only execution (i.e., data retrieval deactivated)

5 queries of the FedBench benchmark suite, executed over real Linked Data on the WWW
Data Retrieval Dominates!

Different number of lookup threads used by the data retrieval operator

Data retrieval op. equipped with a cache
- Cache populated by a first execution
- Times measured for a 2nd, cache-only execution (i.e., data retrieval deactivated)

5 queries of the FedBench benchmark suite, executed over real Linked Data on the WWW

avg. query exec. time (seconds)

[Bar chart showing the average query execution time for 5 queries (Query 1, Query 4, Query 5, Query 9, Query 10) with 10 and 20 threads, and a cache.]
Data Retrieval Dominates!

Different number of lookup threads used by the data retrieval operator

Data retrieval op. equipped with a cache
- Cache populated by a first execution
- Times measured for a 2nd, cache-only execution (i.e., data retrieval deactivated)

5 queries of the FedBench benchmark suite, executed over real Linked Data on the WWW

Approaches to optimize QET \textit{will fail} to be effective
Hypothesis

Response time (RT) can be reduced by choosing a “good” strategy of prioritizing URI lookups.
Test: Prioritizing Lookups Randomly

- Every URI to be looked up is queued with a randomly selected priority
Test: Prioritizing Lookups Randomly

- Every URI to be looked up is queued with a randomly selected priority
- 5 independent runs of FedBench query LD10 over real Linked Data on the WWW; like before, QET essentially the same in all 5 runs

ca. 25% of QET

ca. 59% of QET

ca. 58%
Hypothesis

Response time (RT) can be reduced by choosing a “good” strategy of prioritizing URI lookups.

√
Hypothesis

What is

Response time (RT) can be reduced by choosing a “good” strategy of prioritizing URI lookups.
Our Work on RT-focused Query Processing (1/2)

An attempt to optimize the response time of traversal-based query executions

Research Question

What is
Response time (RT) can be reduced by choosing a “good” strategy of prioritizing URI lookups.

?
Taxonomy of 14 Approaches

- non-adaptive
  - breadth-first (baseline)
  - depth-first
  - random
  - (oracle)

- purely graph-based
  - PageRank
  - indegree

- adaptive
  - local processing agnostic
  - local processing aware

- solution aware graph-based
  - rcc1
  - rcc2
  - rel1
  - rel2

- hybrid graph-based
  - ISrcc1
  - ISrcc2
  - ISrel1
  - ISrel2

- intermediate solution driven
  - IS
Experiment Setup

• Controlled environment to simulate arbitrary test Webs

• 14 different test Webs, each of them generated by distributing a base dataset over a set of documents
  – distribution controlled by two probabilities, $\Phi_1$ and $\Phi_2$
  – by varying $\Phi_1$ and $\Phi_2$ systematically, the link graphs of the resulting test Webs are structured differently

• 6 test queries, can be executed over each test Web
  – for each test Web, the 6 query-specific reachable subwebs are sufficiently diverse

• Detailed analysis of this test setup: Hartig & Özsu, WWW 2014
Experiment Setup

- Controlled environment to simulate arbitrary test Webs

- **14 different test Webs**, each of them generated by distributing a base dataset over a set of documents
  - distribution controlled by two probabilities, $\Phi_1$ and $\Phi_2$
  - by varying $\Phi_1$ and $\Phi_2$ systematically, the link graphs of the resulting test Webs are structured differently

- **6 test queries**, can be executed over each test Web
  - for each test Web, the 6 query-specific reachable subwebs are sufficiently diverse

- Detailed analysis of this test setup: Hartig & Özsu, WWW 2014

84 test cases
Taxonomy of Approaches

- **non-adaptive**
  - breadth-first (baseline)
  - depth-first
  - random
  - (oracle)

- **purely graph-based**
  - PageRank
  - indegree

- **adaptive**

  - local processing agnostic
    - solution aware graph-based
      - rcc1
      - rcc2
      - rel1
      - rel2
  - local processing aware
    - intermediate solution driven
      - hybrid graph-based
        - ISrcc1
        - ISrcc2
        - ISrel1
        - ISrel2

@olafhartig
Simple, Non-Adaptive Approaches

- For each URI to be looked up, choose fixed priority when the URI is added to the lookup queue

1) priority(uri) = 1
   - breadth-first; used as our baseline

2) priority(uri) = priority(lookup that discovered uri) + 1
   - depth-first

3) priority(uri) = random number in interval [1,10]
Results

- Controlled environment to simulate arbitrary test Webs

- **14 different test Webs**, each of them generated by distributing a base dataset over a set of documents
  - distribution controlled by two probabilities, $\Phi_1$ and $\Phi_2$
  - by varying $\Phi_1$ and $\Phi_2$ systematically, the link graphs of the resulting test Webs are structured differently

- **6 test queries**, can be executed over each test Web
  - for each test Web, the 6 query-specific reachable subwebs are sufficiently diverse

- Detailed analysis of this test setup: Hartig & Özsu, WWW 2014

84 test cases
### Results

<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first</th>
<th></th>
<th>time to 50%</th>
<th></th>
<th>time to 100%</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
</tr>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
<td>17.8 %</td>
<td>53.6 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
<td>8.2 %</td>
<td>59.4 %</td>
<td>8.7 %</td>
</tr>
</tbody>
</table>

- Controlled environment to simulate arbitrary test Webs
- 14 different test Webs, each of them generated by distributing a base dataset over a set of documents
  - distribution controlled by two probabilities, $\Phi_1$ and $\Phi_2$
  - by varying $\Phi_1$ and $\Phi_2$ systematically, the link graphs of the resulting test Webs are structured differently
- 6 test queries, can be executed over each test Web
  - for each test Web, the 6 query-specific reachable subwebs are sufficiently diverse
- Detailed analysis of this test setup: Hartig & Özsu, WWW 2014

Percentage of cases in which the approaches are 10% worse/better than the baseline (BFS)

84 test cases
## Results

<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first worse</th>
<th>better</th>
<th>time to 50% worse</th>
<th>better</th>
<th>time to 100% worse</th>
<th>better</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
<td>17.8 %</td>
<td>53.6 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
<td>8.2 %</td>
<td>59.4 %</td>
<td>8.7 %</td>
</tr>
</tbody>
</table>

Percentage of cases in which the approaches are 10% worse/better than the baseline (BFS).

Unsuitable!
Taxonomy of Approaches

- **non-adaptive**
  - breadth-first (baseline)
  - depth-first
  - random
  - (oracle)

- **purely graph-based**
  - PageRank
  - indegree

- **adaptive**
  - local processing agnostic
  - local processing aware

  - **solution aware graph-based**
    - rcc1
    - rcc2
    - rel1
    - rel2

  - **intermediate solution driven**
    - ISrc1
    - ISrc2
    - ISrel1
    - ISrel2

@olafhartig
Graph-Based Approaches

• Construct a **model of the link graph** discovered during a query execution
  - Extend the model incrementally (whenever another document or a link between documents is discovered)

• Apply a **vertex scoring function** to the model
  - e.g., PageRank or in-degree

• Use scores of vertexes as priorities for respective URIs in lookup queue

• Adjust priorities after every augmentation of the model
<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first</th>
<th></th>
<th>time to 50%</th>
<th></th>
<th>time to 100%</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
</tr>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
<td>17.8 %</td>
<td>53.6 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
<td>8.2 %</td>
<td>59.4 %</td>
<td>8.7 %</td>
</tr>
<tr>
<td>indegree</td>
<td>21.7 %</td>
<td>21.7 %</td>
<td>65.8 %</td>
<td>4.1 %</td>
<td>50.7 %</td>
<td>5.8 %</td>
</tr>
</tbody>
</table>

Also unsuitable!
Taxonomy of Approaches

non-adaptive
- breadth-first (baseline)
- depth-first
- random
- (oracle)

purely graph-based
- PageRank
- indegree

adaptive

local processing agnostic

solution aware graph-based
- rcc1
- rcc2
- rel1
- rel2

local processing aware

intermediate solution driven
- IS

hybrid graph-based
- ISrcc1
- ISrcc2
- ISrel1
- ISrel2
Local-Processing Awareness

- Take information about the result construction process into account
Solution-Based Vertex Scoring

• Use a vertex-scoring function that is based on result contribution counter (RCC) of each doc

Data Retrieval Component

Result Construction Component

Solution

Bindings: \{ ?x \rightarrow u, ?y \rightarrow v \}
Contributing docs: \(d_2, d_6\)
Solution-Based Vertex Scoring

- Use a vertex-scoring function that is based on result contribution counter (RCC) of each doc

- \( rcc_X(\text{doc}) \) = sum of RCCs of all documents in the \( X \)-step in-neighborhood of \( \text{doc} \)
  - \( rcc1, rcc2 \)

- \( rel_X(\text{doc}) \) = number of documents with RCC > 0 in the \( X \)-step in-neighborhood of \( \text{doc} \)
  - \( rel1, rel2 \)
## Results

Percentage of cases in which the approaches are 10% worse/better than the baseline (BFS)

<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first</th>
<th></th>
<th>time to 50%</th>
<th></th>
<th>time to 100%</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
<td>worse</td>
<td>better</td>
</tr>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
<td>17.8 %</td>
<td>53.6 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
<td>8.2 %</td>
<td>59.4 %</td>
<td>8.7 %</td>
</tr>
<tr>
<td>indegree</td>
<td>21.7 %</td>
<td>21.7 %</td>
<td>65.8 %</td>
<td>4.1 %</td>
<td>50.7 %</td>
<td>5.8 %</td>
</tr>
<tr>
<td>rcc1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>4.1 %</td>
<td>1.4 %</td>
<td>7.2 %</td>
<td>24.6 %</td>
</tr>
<tr>
<td>rcc2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>2.7 %</td>
<td>2.7 %</td>
<td>4.1 %</td>
<td>20.3 %</td>
</tr>
<tr>
<td>rel1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>5.5 %</td>
<td>1.4 %</td>
<td>11.6 %</td>
<td>29.0 %</td>
</tr>
<tr>
<td>rel2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>11.0 %</td>
<td>0.0 %</td>
<td>2.9 %</td>
<td>26.1 %</td>
</tr>
</tbody>
</table>

No effect!
### Results

<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first</th>
<th>time to 50%</th>
<th>time to 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>worse</td>
<td>better</td>
<td>worse</td>
</tr>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
</tr>
<tr>
<td>indegree</td>
<td>21.7 %</td>
<td>21.7 %</td>
<td>65.8 %</td>
</tr>
<tr>
<td>rcc1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>4.1 %</td>
</tr>
<tr>
<td>rcc2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>2.7 %</td>
</tr>
<tr>
<td>rel1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>5.5 %</td>
</tr>
<tr>
<td>rel2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>11.0 %</td>
</tr>
<tr>
<td>intsol</td>
<td>7.2 %</td>
<td>31.9 %</td>
<td>15.1 %</td>
</tr>
<tr>
<td>isrcc1</td>
<td>2.9 %</td>
<td>30.4 %</td>
<td>5.5 %</td>
</tr>
<tr>
<td>isrcc2</td>
<td>5.8 %</td>
<td>33.3 %</td>
<td>5.5 %</td>
</tr>
<tr>
<td>isrel1</td>
<td>0.0 %</td>
<td>33.3 %</td>
<td>2.7 %</td>
</tr>
<tr>
<td>isrel2</td>
<td>2.9 %</td>
<td>31.9 %</td>
<td>4.1 %</td>
</tr>
</tbody>
</table>

**Most suitable (among the tested approaches)**

- DFS: time to 50% better, time to 100% worse
- indegree: time to 50% worse, time to 100% better
- intsol: time to 50% worse, time to 100% better
- isrcc2: time to 50% worse, time to 100% better

Percentage of cases in which the approaches are 10% worse/better than the baseline (BFS)
Taxonomy of Approaches

- **non-adaptive**
  - breadth-first (baseline)
  - depth-first
  - random
  - (oracle)

- **purely graph-based**
  - PageRank
  - indegree

- **adaptive**
  - local processing agnostic
  - solution aware graph-based
    - rcc1
    - rcc2
    - rel1
    - rel2
  - intermediate solution driven
    - ISrc1
    - ISrc2
    - ISrel1
    - ISrel2

- **local processing aware**
Oracle Approach

- Gold standard for experiments
- Intuition: the more a document contributes to the query result, the earlier it should be retrieved

\[
\text{priority}(uri) = \text{result contribution counter of the document that will be retrieved by looking up } uri
\]

where: \(\text{rcc}(doc) = \text{number of solutions whose computation requires some triple from } doc\)

- Oracle cannot be used in practice

Such information is available only after executing a query completely
## Results

<table>
<thead>
<tr>
<th>approach</th>
<th>time to a first worse</th>
<th>better</th>
<th>time to 50% worse</th>
<th>better</th>
<th>time to 100% worse</th>
<th>better</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>23.2 %</td>
<td>26.1 %</td>
<td>58.9 %</td>
<td>17.8 %</td>
<td>53.6 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>random</td>
<td>13.0 %</td>
<td>27.5 %</td>
<td>58.9 %</td>
<td>8.2 %</td>
<td>59.4 %</td>
<td>8.7 %</td>
</tr>
<tr>
<td>indegree</td>
<td>21.7 %</td>
<td>21.7 %</td>
<td>65.8 %</td>
<td>4.1 %</td>
<td>50.7 %</td>
<td>5.8 %</td>
</tr>
<tr>
<td>rcc1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>4.1 %</td>
<td>1.4 %</td>
<td>7.2 %</td>
<td>24.6 %</td>
</tr>
<tr>
<td>rcc2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>2.7 %</td>
<td>2.7 %</td>
<td>4.1 %</td>
<td>20.3 %</td>
</tr>
<tr>
<td>rel1</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>5.5 %</td>
<td>1.4 %</td>
<td>11.6 %</td>
<td>29.0 %</td>
</tr>
<tr>
<td>rel2</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>11.0 %</td>
<td>0.0 %</td>
<td>2.9 %</td>
<td>26.1 %</td>
</tr>
<tr>
<td>intsol</td>
<td>7.2 %</td>
<td>31.9 %</td>
<td>15.1 %</td>
<td>27.4 %</td>
<td>26.1 %</td>
<td>10.1 %</td>
</tr>
<tr>
<td>isrcc1</td>
<td>2.9 %</td>
<td>30.4 %</td>
<td>5.5 %</td>
<td>26.0 %</td>
<td>14.5 %</td>
<td>18.8 %</td>
</tr>
<tr>
<td>isrcc2</td>
<td>5.8 %</td>
<td>33.3 %</td>
<td>5.5 %</td>
<td>24.7 %</td>
<td>13.0 %</td>
<td>26.1 %</td>
</tr>
<tr>
<td>isrel1</td>
<td>0.0 %</td>
<td>33.3 %</td>
<td>2.7 %</td>
<td>24.7 %</td>
<td>15.9 %</td>
<td>26.1 %</td>
</tr>
<tr>
<td>isrel2</td>
<td>2.9 %</td>
<td>31.9 %</td>
<td>4.1 %</td>
<td>23.3 %</td>
<td>11.6 %</td>
<td>26.1 %</td>
</tr>
<tr>
<td>oracle</td>
<td>0.0 %</td>
<td>35.3 %</td>
<td>0.0 %</td>
<td>41.2 %</td>
<td>0.0 %</td>
<td>64.7 %</td>
</tr>
</tbody>
</table>

A lot of room for further improvement!
Our Work on RT-focused Query Processing (2/2)

An attempt to make the core fragment of SPARQL suitable for the task

Our Work on RT-focused Query Processing (2/2)

An attempt to make the core fragment of SPARQL suitable for the task

...by making it monotonic

Motivating Example

- Query:
  ```sql
  PREFIX ex: <http://example.org/>
  SELECT ?post ?text ?img WHERE {
    ?post ex:hasText ?text
    OPTIONAL { ?post ex:hasImage ?img }
  }
  ```

- Data:
  ```plaintext
  ex:post1 ex:hasText "Good ..."
  ex:post2 ex:hasText "I can ..."
  ex:post1 ex:hasImage ex:sun.png
  ```
Motivating Example

- Query:  
  ```sparql
  PREFIX ex: <http://example.org/>
  SELECT ?post ?text ?img WHERE {
    ?post ex:hasText ?text
    OPTIONAL { ?post ex:hasImage ?img }
  }
  ```

- Data:  
  ```
  ex:post1 ex:hasText "Good …"
  ex:post2 ex:hasText "I can …"
  ex:post1 ex:hasImage ex:sun.png
  ```

- Intermediate query result contains the following elements:
  \[
  \mu_1 = \{ \text{?post} \rightarrow \text{ex:post1}, \text{?text} \rightarrow \text{"Good …"} \}
  \]
Motivating Example

• Query:  
  
  ```
  PREFIX ex: <http://example.org/>
  SELECT ?post ?text ?img WHERE {
    ?post ex:hasText ?text
  }
  OPTIONAL { ?post ex:hasImage ?img }
  ```

• Data:  
  
  ```
  ex:post1 ex:hasText "Good …"
  ex:post2 ex:hasText "I can …"
  ex:post1 ex:hasImage ex:sun.png
  ```

• Intermediate query result contains the following elements:  
  
  \[
  \mu_1 = \{ \text{post} \rightarrow \text{ex:post1}, \text{text} \rightarrow "Good …" \}
  \]
  \[
  \mu_2 = \{ \text{post} \rightarrow \text{ex:post1}, \text{text} \rightarrow "I can …" \}
  \]
Motivating Example

- **Query:**
  
  ```query
  PREFIX ex: <http://example.org/>
  SELECT ?post ?text ?img WHERE {
    ?post ex:hasText ?text
    OPTIONAL { ?post ex:hasImage ?img }
  }
  ```

- **Data:**

  (discovered incrementally)

  - ex:post1 ex:hasText "Good …"
  - ex:post2 ex:hasText "I can …"
  - ex:post1 ex:hasImage ex:sun.png

- **Intermediate query result contains the following elements:**

  - \(\mu_1 = \{ \text{?post} \rightarrow \text{ex:post1, ?text} \rightarrow \text{"Good …"} \}\)
  - \(\mu_2 = \{ \text{?post} \rightarrow \text{ex:post1, ?text} \rightarrow \text{"I can …"} \}\)
  - \(\mu_3 = \{ \text{?post} \rightarrow \text{ex:post1, ?text} \rightarrow \text{"Good …"}, \text{?img} \rightarrow \text{ex:sub.png} \}\)
What’s the Issue?

- Example query is not monotonic (as the example illustrates)

- Definition: A query $Q$ is monotonic if for every pair $(\text{database 1}, \text{database 2})$ of possible databases, it holds that:

$$Q(\text{database 1}) \subseteq Q(\text{database 2})$$

- For every monotonic query, each element of any intermediate result can be output as soon as it has been computed

- For non-monotonic queries that’s not possible
  - some elements of the result can be output only after having consulted all relevant parts of the queried data
  - remember, in Web querying we access the relevant parts only incrementally, with network latencies
What’s the Issue? cont’d

• Good news: the AND-UNION-FILTER fragment of SPARQL is monotonic
  – see Arenas and Perez, PODS 2011

• Bad news: for the AND-UNION-FILTER-OPT fragment, monotonicity is undecidable
  – i.e., queries with OPTIONAL may be non-monotonic
  – see Hartig, PhD Thesis 2014

• Reminder of the formal semantics of OPT:
  If \( P \) is \( (P_1 \text{ OPT } P_2) \), then \( [P]_G = ([P_1]_G \boxtimes [P_2]_G) \cup ([P_1]_G \setminus [P_2]_G) \).
Our Proposal: The OPT$^+$ Operator

- Similar in spirit to OPT, but without causing non-monotonicity

- Definition:
  If $P$ is $(P_1 \text{ OPT}^+ P_2)$ then $[P]_G = ([P_1]_G \Join [P_2]_G) \cup [P_1]_G$.

- Properties:
  - $(P_1 \text{ OPT}^+ P_2) \equiv ((P_1 \text{ AND } P_2) \text{ UNION } P_1)$
  - Every query with OPT$^+$ can be rewritten into an equivalent one without OPT$^+$
  - If we replace OPT by OPT$^+$, complexity of evaluation drops from PSPACE-complete to NP-complete
  - Result of a query with OPT is a subset of the result of the corresponding query with OPT$^+$

- Reminder of the formal semantics of OPT:
  If $P$ is $(P_1 \text{ OPT } P_2)$, then $[P]_G = ([P_1]_G \Join [P_2]_G) \cup ([P_1]_G \setminus [P_2]_G)$.
Research Question 1

• How significant is the increase of the size of query results in practice when using the OPT$^+$ operator instead of OPT?

• Method:
  − for query logs of 4 public SPARQL endpoints, extract a set of randomly selected queries with OPTIONAL
  − use only the WHERE clause, combined with SELECT *
  − rewrite each such query into an OPT$^+$-equivalent version by using: \((P_1 \text{ OPT}^+ P_2) \equiv ((P_1 \text{ AND } P_2) \text{ UNION } P_1)\)
  − execute both versions over the corresponding dataset (either using the original SPARQL endpoint or a local triplestore with the dataset loaded)
## Results

<table>
<thead>
<tr>
<th>log</th>
<th>% of pairs with same result size</th>
<th>at most 2x increase</th>
<th>at most 10x increase</th>
<th>at most 100x increase</th>
<th>at most 1000x increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>All pairs of test queries (for which there were no errors)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBP\textsubscript{3.5.1}</td>
<td>65.74%</td>
<td>71.23%</td>
<td>80.86%</td>
<td>91.46%</td>
<td>99.57%</td>
</tr>
<tr>
<td>WD\textsubscript{all}</td>
<td>67.19%</td>
<td>98.90%</td>
<td>99.88%</td>
<td>99.89%</td>
<td>99.90%</td>
</tr>
<tr>
<td>WD\textsubscript{org}</td>
<td>98.73%</td>
<td>99.73%</td>
<td>99.95%</td>
<td>99.95%</td>
<td>99.95%</td>
</tr>
<tr>
<td>LGD</td>
<td>98.46%</td>
<td>99.88%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

- **Same result for a large fraction of queries**
- **Non-negligible number of case with substantial increase in result size**
## Results

<table>
<thead>
<tr>
<th></th>
<th>log</th>
<th>% of pairs with same result size</th>
<th>at most 2x increase</th>
<th>at most 10x increase</th>
<th>at most 100x increase</th>
<th>at most 1000x increase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All pairs of test queries (for which there were no errors)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBP₃.₅.₁</td>
<td>65.74%</td>
<td>71.23%</td>
<td>80.86%</td>
<td>91.46%</td>
<td>99.57%</td>
<td></td>
</tr>
<tr>
<td>WD_all</td>
<td>67.19%</td>
<td>98.90%</td>
<td>99.88%</td>
<td>99.89%</td>
<td>99.90%</td>
<td></td>
</tr>
<tr>
<td>WD_org</td>
<td>98.73%</td>
<td>99.73%</td>
<td>99.95%</td>
<td>99.95%</td>
<td>99.95%</td>
<td></td>
</tr>
<tr>
<td>LGD</td>
<td>98.46%</td>
<td>99.88%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td></td>
</tr>
</tbody>
</table>

| Only the pairs whose OPTIONAL query did not contain sequences of OPTIONALs |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| DBP₃.₅.₁ | 93.09% | 100.00% | 100.00% | 100.00% | 100.00% |                        |
| WD_all | 66.04% | 99.99% | 100.00% | 100.00% | 100.00% |                        |
| WD_org | 99.08% | 99.99% | 100.00% | 100.00% | 100.00% |                        |
| LGD | 98.57% | 100.00% | 100.00% | 100.00% | 100.00% |                        |

| Only the pairs whose OPTIONAL query did contain sequences of OPTIONALs |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| DBP₃.₅.₁ | 65.41% | 70.88% | 80.63% | 91.36% | 99.56% |                        |
| WD_all | 75.12% | 91.35% | 99.06% | 99.17% | 99.17% |                        |
| WD_org | 97.61% | 98.88% | 99.81% | 99.81% | 99.81% |                        |
| LGD | 92.69% | 93.67% | 99.95% | 100.00% | 100.00% |                        |
Research Question 2

- How common are queries with sequences of OPTIONAL?

Method
- Analyzed SPARQL endpoint query logs of 10 datasets

Results (all logs combined):
- 47% of 2.9M distinct queries with OPTIONAL contain more than one OPTIONAL
- almost all of these 47% contain at least one sequence of OPTIONAL (only 3,032 do not)
- 99.9% of these contain one sequence
- rest contains exactly two separate sequences

⇒ sequences of OPTIONAL are quite common
Research Question 3

• How suitable is the OPT\(^+\) operator in terms of its potential for query executions that achieve reduced response times?
  
  – Is rewriting OPT\(^+\) queries into OPT\(^+\)-equivalent versions already sufficient? (i.e., no specific algorithm for OPT\(^+\))
  
  – Does OPT\(^+\) enable a query engine to employ a specific algorithm designed to return solutions as early as possible?
  
  – Does this algorithm allow the engine to return first mappings earlier than for the corresponding query with OPTIONAL?

• Method
  
  – extend existing SPARQL engine (Jena) with OPT\(^+\) algorithms
  
  – add config option to execute OPTIONAL queries using any of these algorithms instead of its standard algorithm for OPT
  
  – execute versions of OPTIONAL queries from query log over an HDT back-end loaded with the corresponding dataset
Jena’s Algorithm for OPT

- Variation of a nested loops join (NLJ)

**Input:** \((P_L \text{ OPT } P_R)\)

\(I_L := \text{iterator over result of } P_L\)

**for each** \(\mu\) from \(I_L\) **do**

\(P_R' := \mu[P_R]\)

\(I_R' := \text{iterator over result of } P_R'\)

**if** \(I_R'\) has solution mappings **then**

**for each** \(\mu'\) from \(I_R'\) **do**

\(\mu'' := \mu \cup \mu'\)

output \(\mu''\)

**else**

output \(\mu\)
NLJ$^+$ Algorithm for OPT$^+$

- Adaptation of the NLJ-based algorithm for OPT

**Input:** $(P_L \ OPT \ P_R)$

$I_L :=$ iterator over result of $P_L$

**for each** $\mu$ from $I_L$ **do**

$P_R' := \mu[P_R]$ 

$I_R' :=$ iterator over result of $P_R'$

**if** $I_R'$ has solution mappings **then**

output $\mu$

**for each** $\mu'$ from $I_R'$ **do**

$\mu'' := \mu \cup \mu'$

output $\mu''$

**else**

output $\mu$
mNLJ$^+$ Algorithm for OPT$^+$

- Another adaptation; first consumes the LHS input completely

**Input:** ($P_L$ OPT $P_R$)

$I_L :=$ iterator over result of $P_L$

$M_L :=$ empty list of solution mappings

**for each** $\mu$ from $I_L$ do

- output $\mu$, and append $\mu$ to $M_L$

**for each** $\mu$ in $M_L$ do

- $P_R' := \mu[P_R]$

- $I_R' :=$ iterator over result of $P_R'$

- **if** $I_R'$ has solution mappings **then**

  **for each** $\mu'$ from $I_R'$ do

  - $\mu'' := \mu \cup \mu'$

  - output $\mu''$

- **else**

  output $\mu$
Test Queries

• 60K distinct OPTIONAL queries selected randomly from the DBpedia 3.5.1 query log
  - this log is comparably diverse in terms of
    i) how OPTIONAL is used and
    ii) result-size increase when OPT+ is used

• 20.8K of them have a non-empty result and can be used
Comparison of all OPT+ Approaches

Using the OPT+-equivalent versions is unsuitable
Comparison of all OPT$^+$ Approaches (cont’d)
mNLJ+ vs NLJ+

The graph shows the average differences between RTX% values (in ms) for the cases in which the approach was better. The line for OPT+ (mNLJ+) is represented by blue diamonds, while the line for OPT+ (NLJ+) is represented by red triangles. The graph indicates that there is no clear winner between the two approaches. The red line for OPT+ (NLJ+) shows much fewer cases than the blue line for OPT+ (mNLJ+).
OPT vs NLJ$^+$

(only 41 query for which the OPT version has a result size $\geq 100$)
OPT vs NLJ$^+$

no significant advantage in using OPT$^+$

avg. differences between RT1stX values (in ms) for the cases in which the approach was better
Summary
Take Away

• Minimizing QET ≠ Minimizing RT
  − Approaches to minimize QET for traversal-based query execution will fail to be effective (not so for TPF, etc)
  − QET ≠ RT100%

• Response-time focused query processing (i.e., returning result elements early) has received too little attention
  TODO: The approaches in this presentation should be understood as a beginning, not a final answer

• Approaches to prioritize data retrieval can reduce response times of traversal-based query execution
  TODO: Certainly, there are other, more effective approaches
  TODO: Ideas may be adapted to federated query processing

• Language feature have to be chosen with care