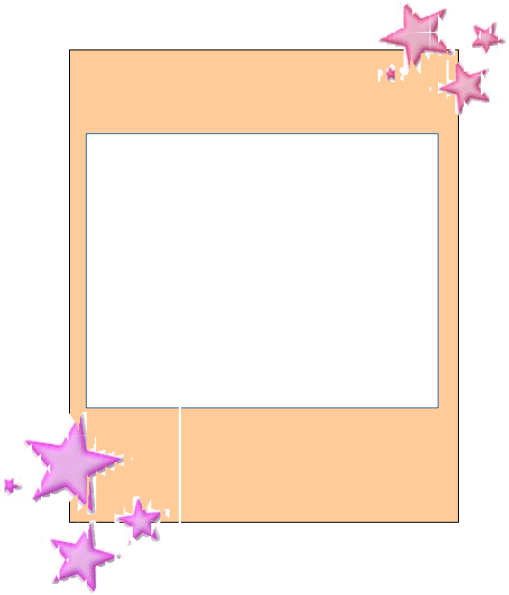
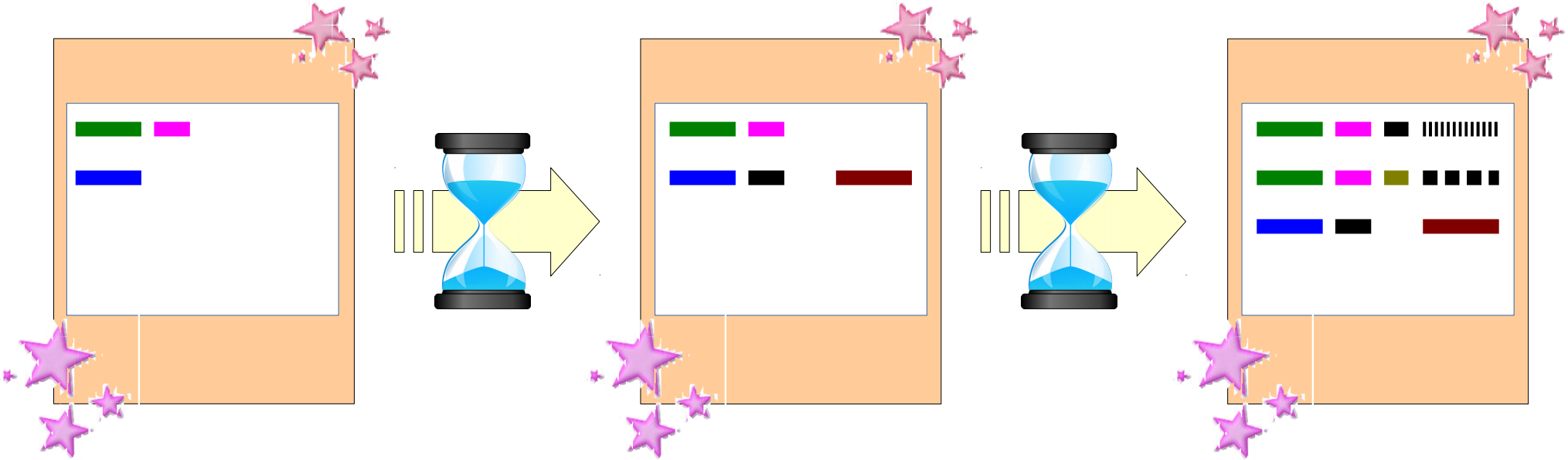
Welcome!

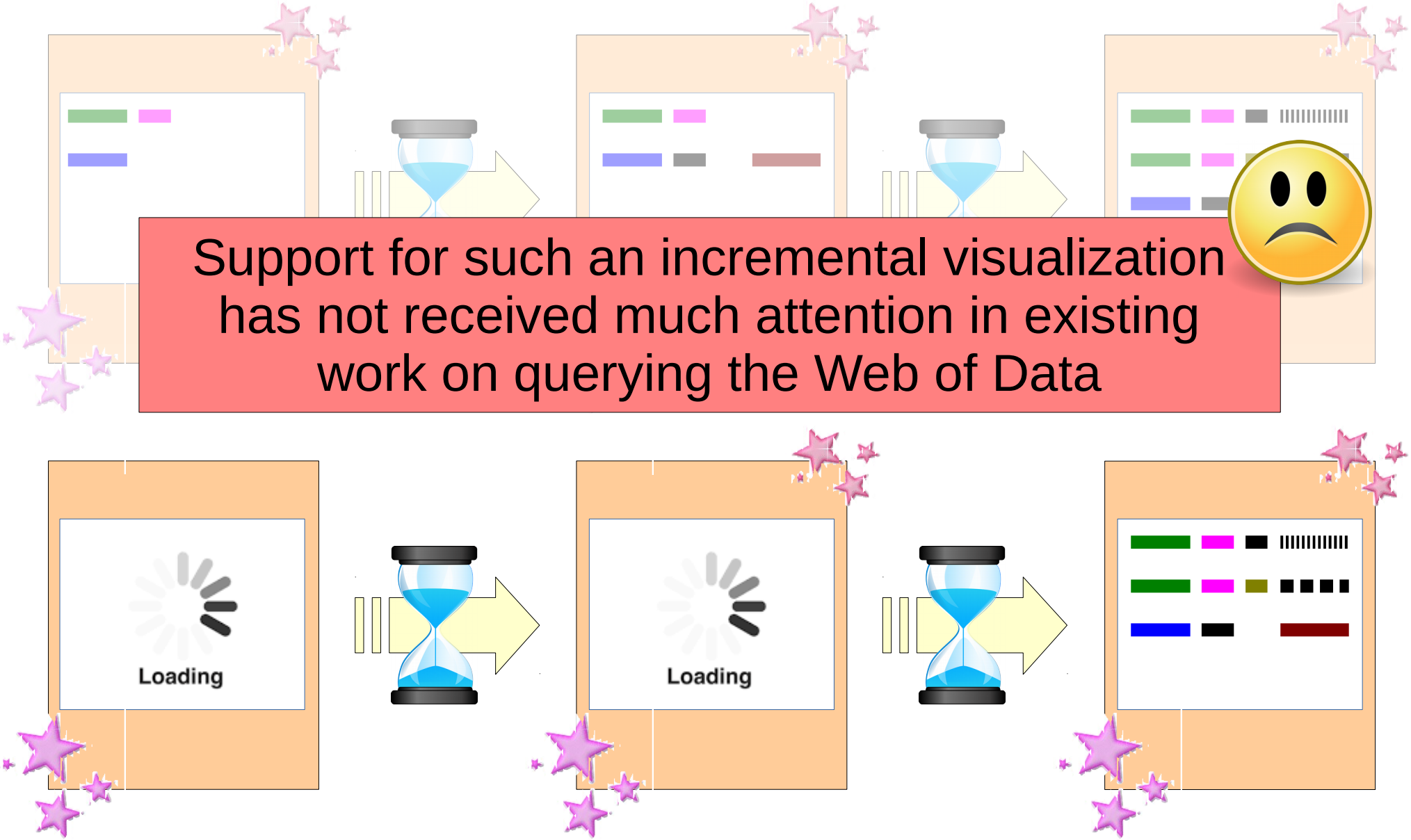
Information in Dynamic Web Pages



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

LINKÖPING UNIVERSITY

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

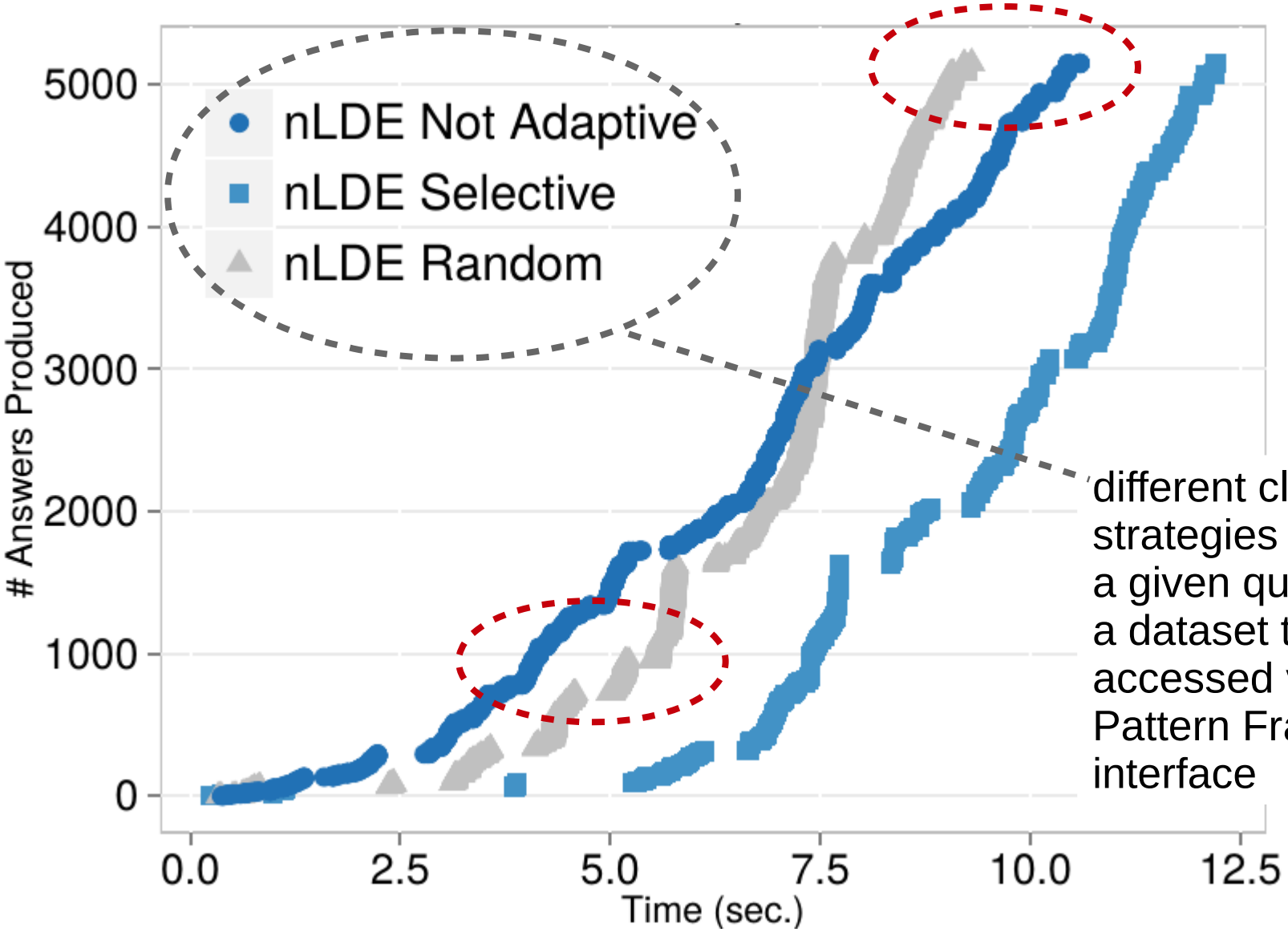
- Aiming to minimize QET is different from aiming to minimize RT
  - Evidence 1
  - Evidence 2
- Some of our work on RT-focused query processing
  - An attempt to optimize the response times of traversal-based query execution
  - An attempt to make the core fragment of SPARQL suitable for the task

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

Minimizing QET ≠ Minimizing RT

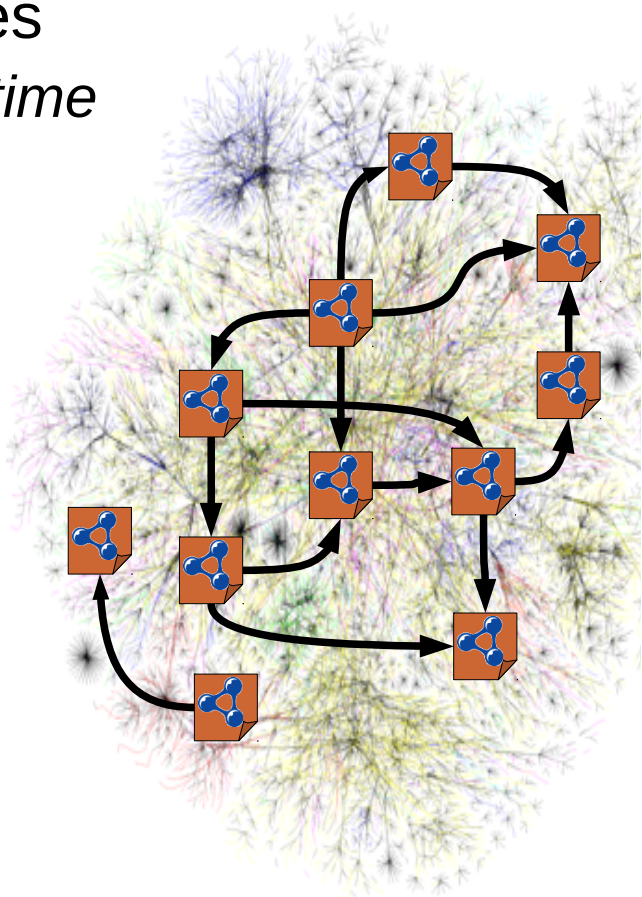
Evidence 2

Based on: Olaf Hartig and M. Tamer Özsu: *Walking without a Map: Ranking-Based Traversal for Querying Linked Data*. ISWC 2016.

Olaf Hartig and M. Tamer Özsu: *Optimizing Response Times of Traversal-Based Linked Data Queries (Extended Version). arXiv:1607.01046*

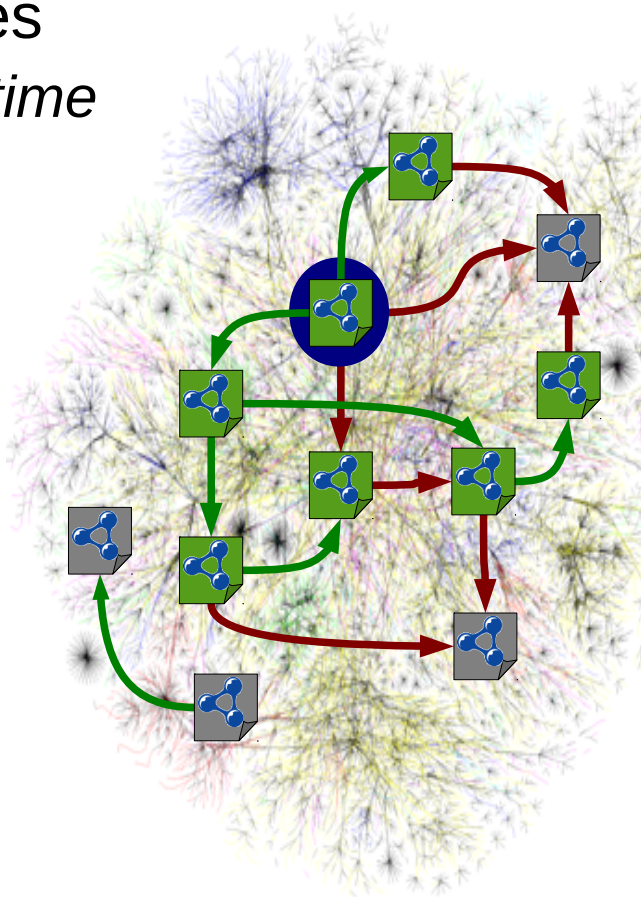
Linked Data Query Processing

- 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*



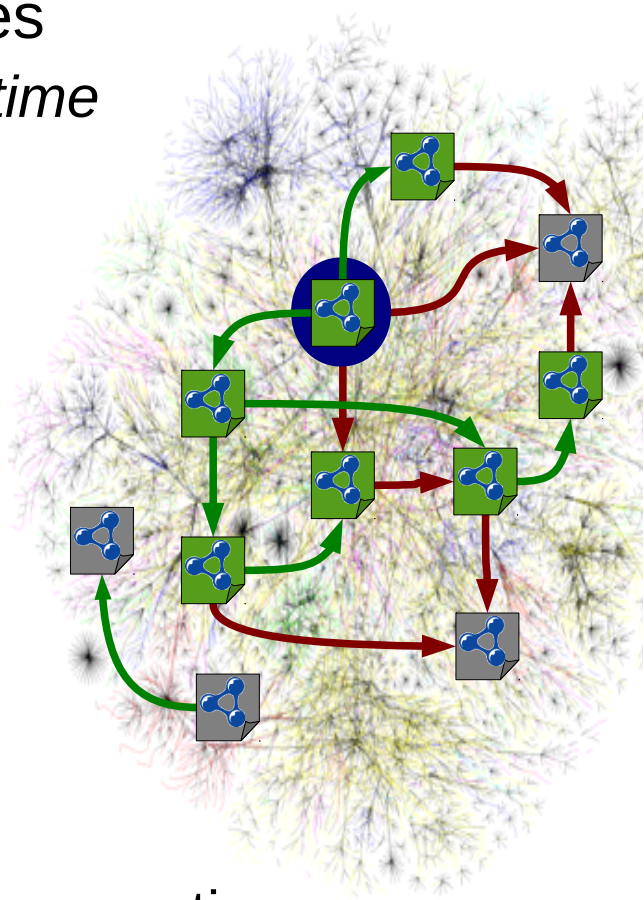
Linked Data Query Processing

- 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*

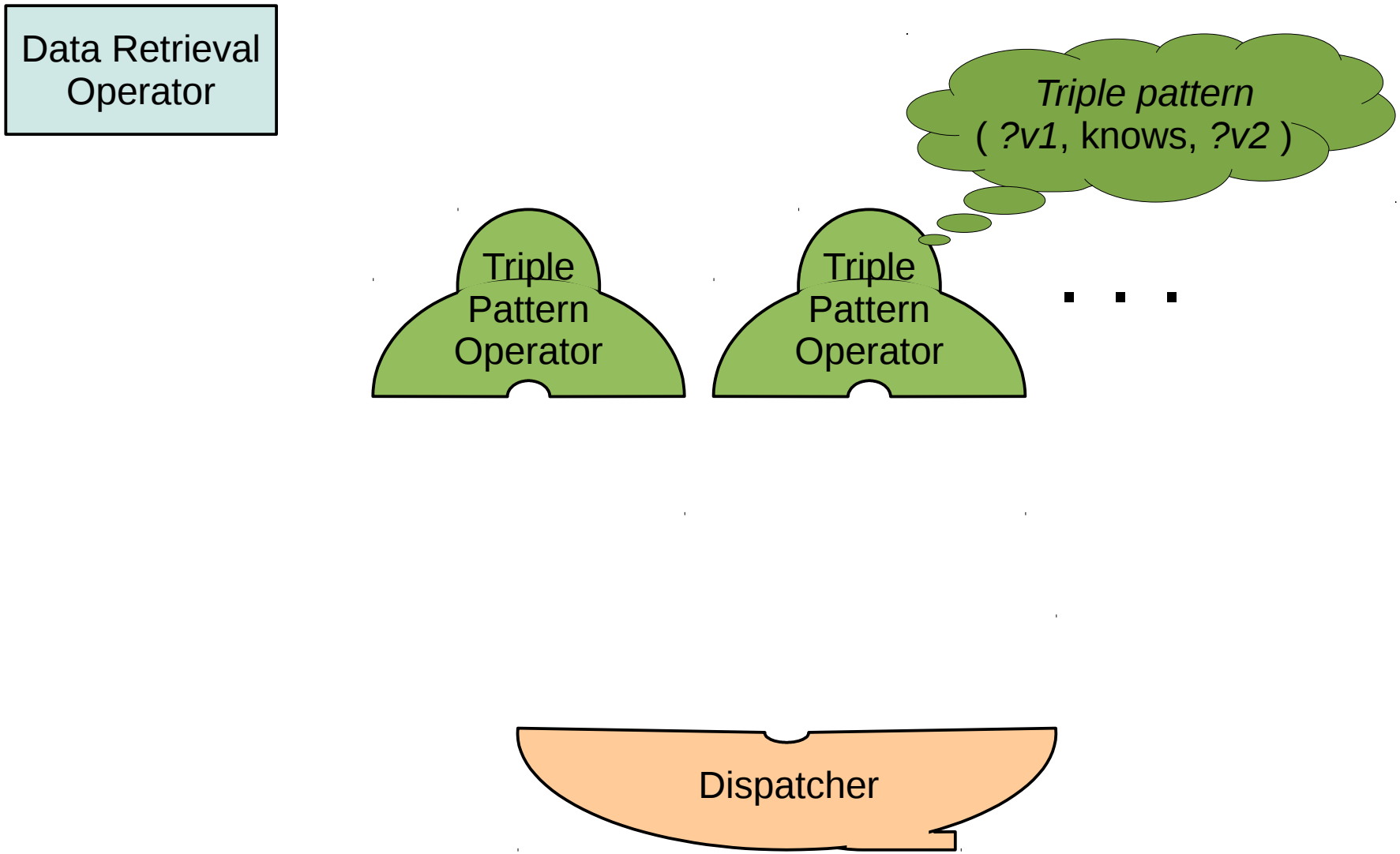


Linked Data Query Processing

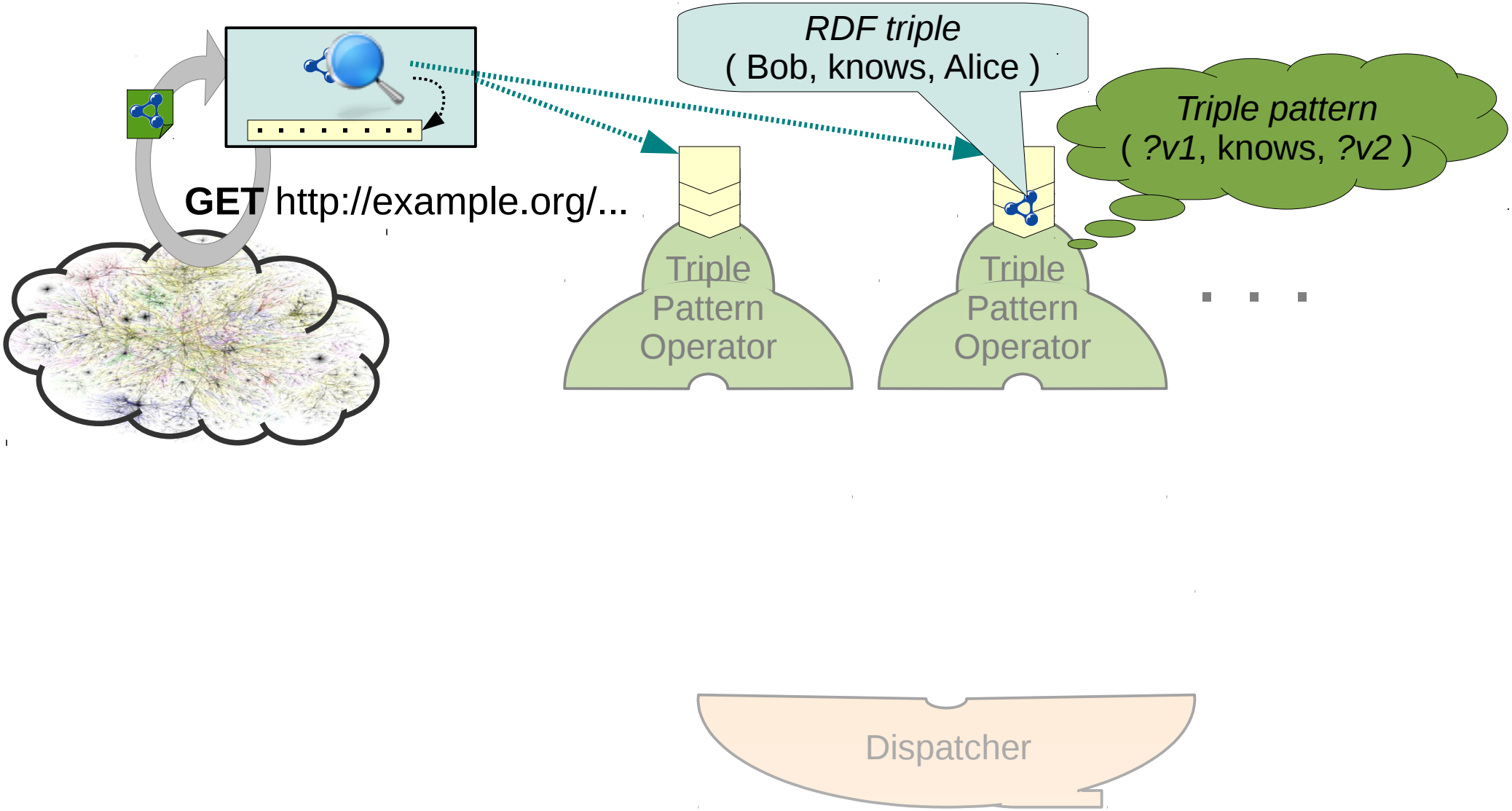
- Focus: querying Linked Data live on the Web by relying only on the Linked Data principles
  - look up URIs to access original data *at runtime*
- Queries
  - typically expressed using SPARQL (in practice, BGPs only)
  - reachability-based query semantics; i.e., scope of evaluation is virtual union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



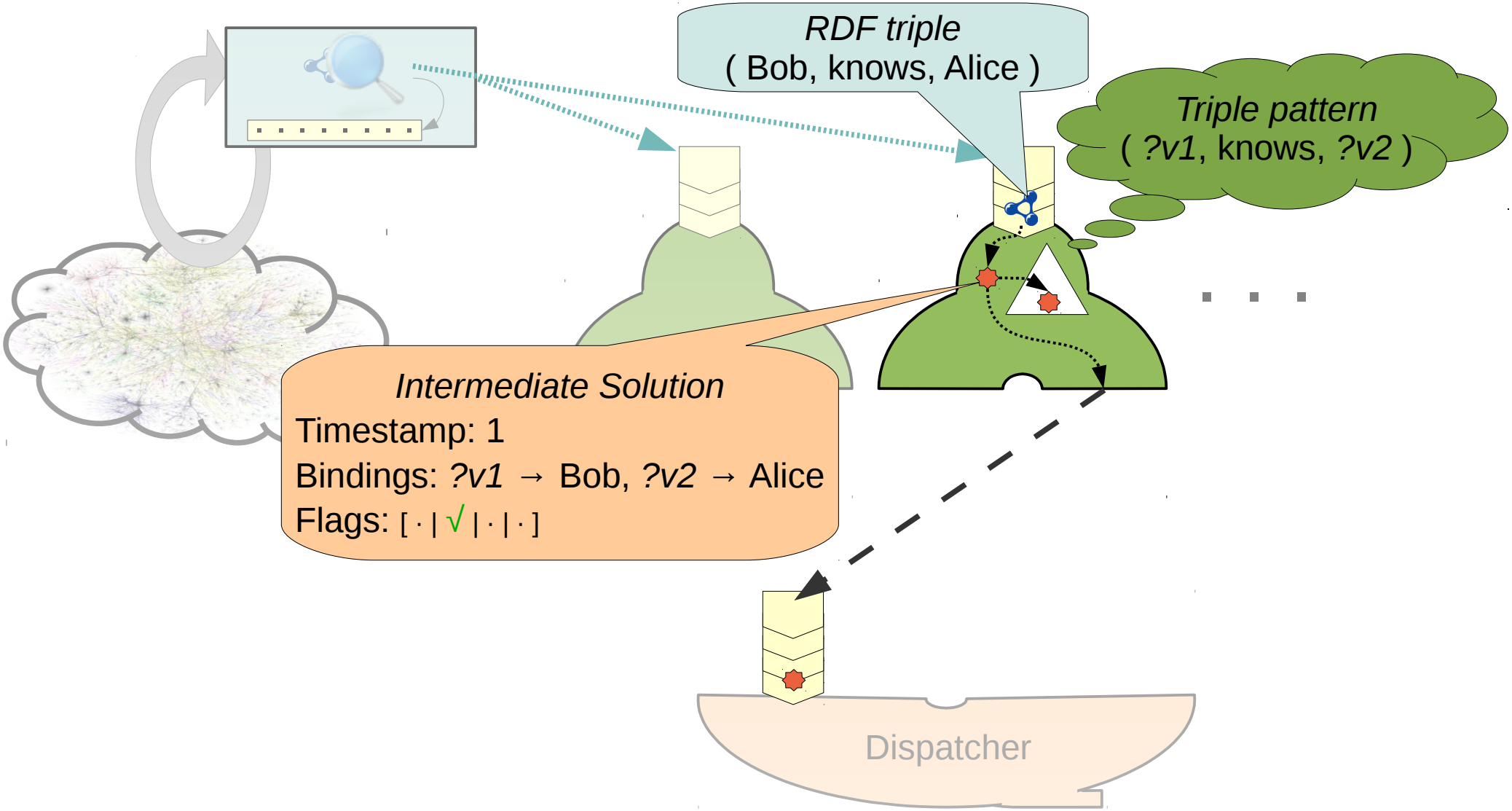
Concrete Implementation Approach



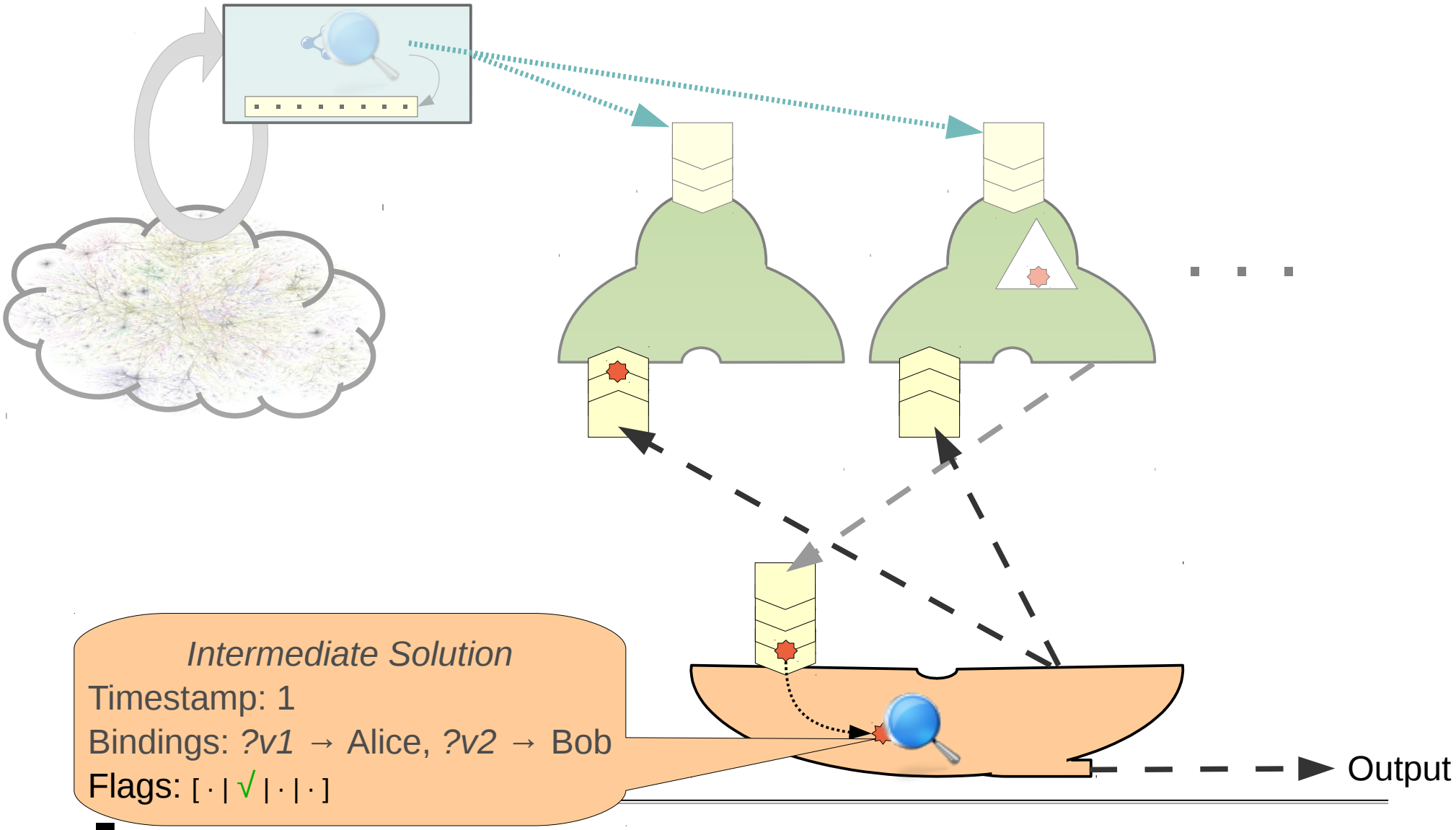
Data Retrieval Operator



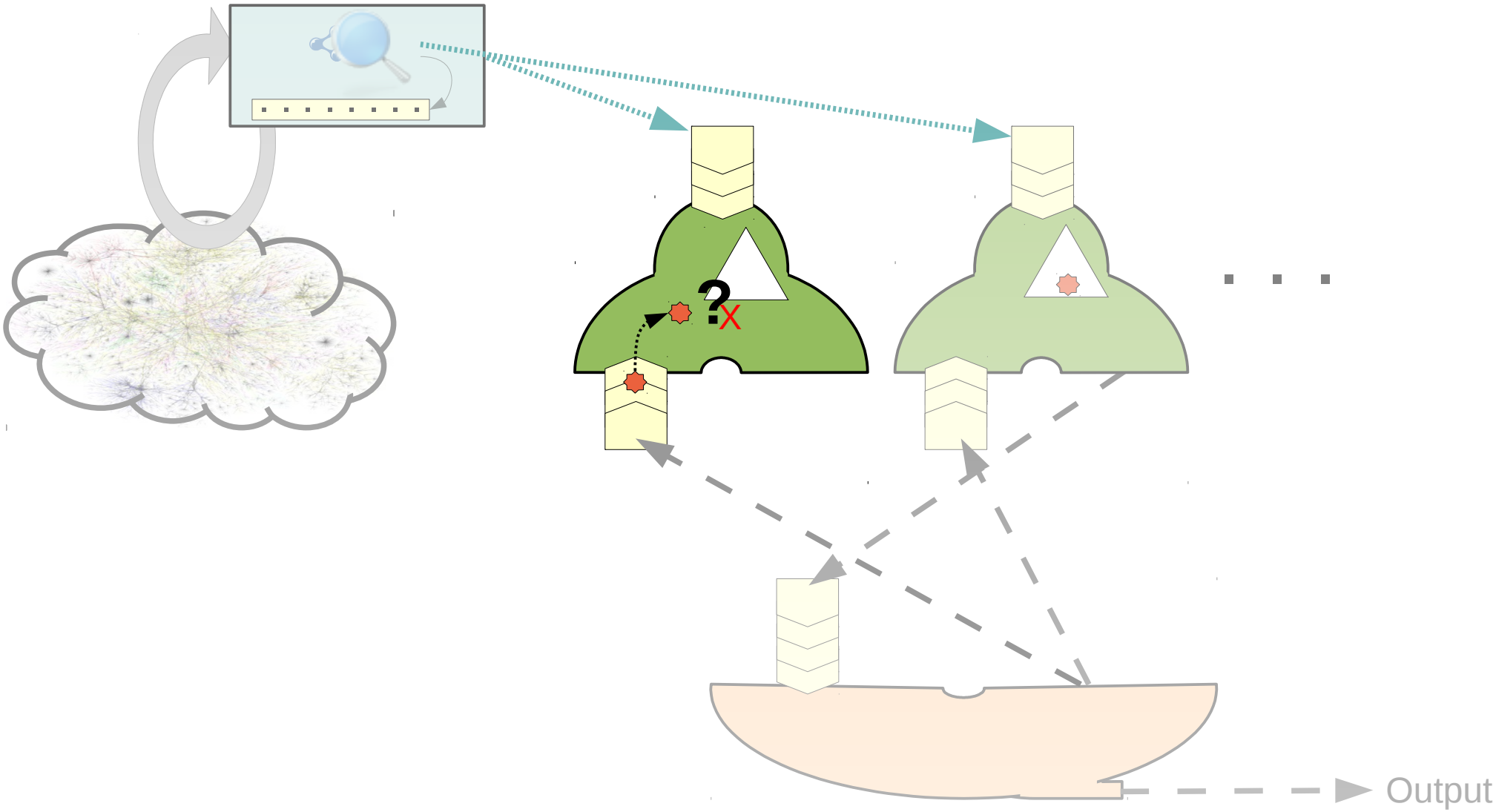
Triple Pattern Operator



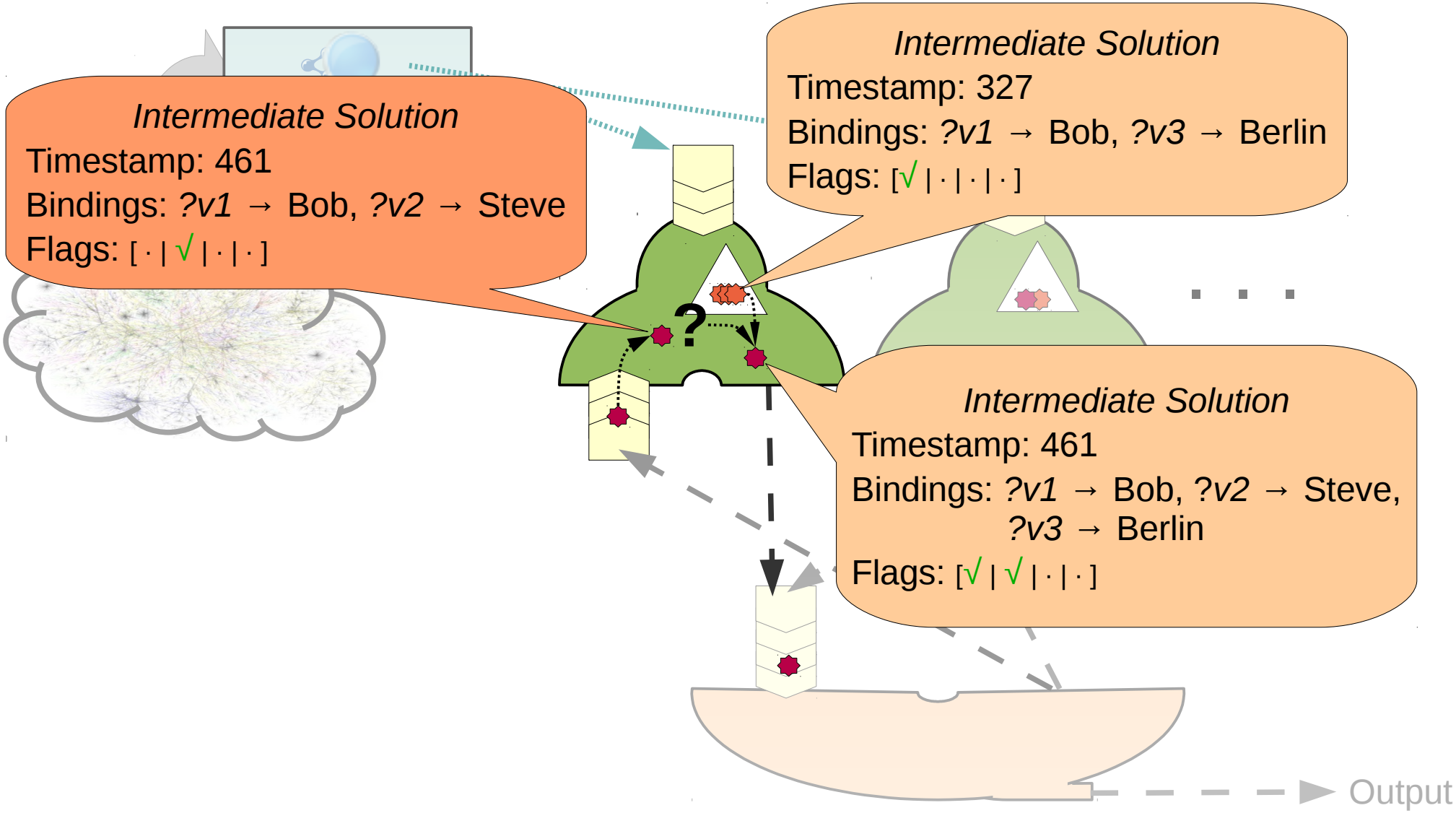
Dispatcher


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

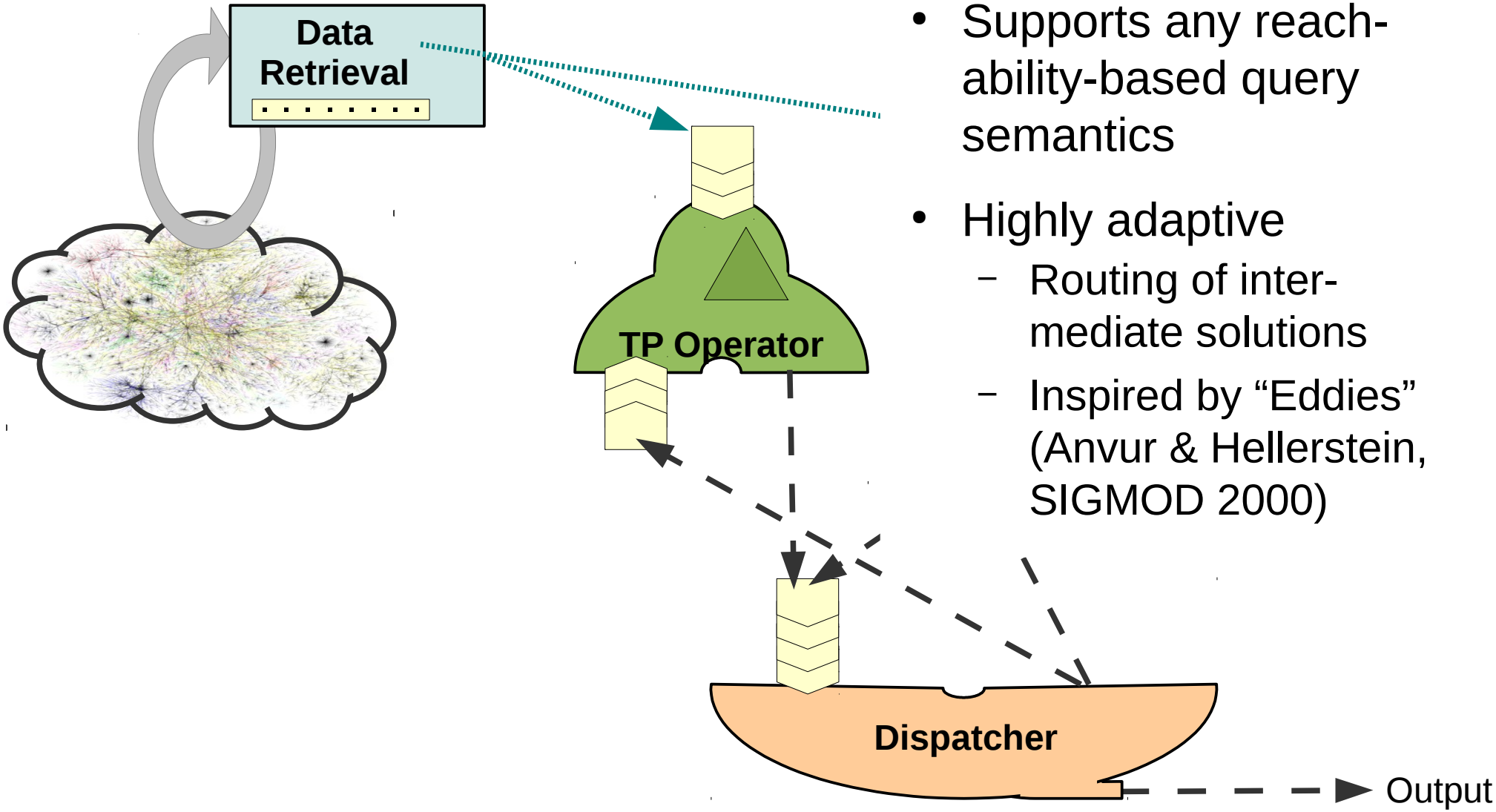
Triple Pattern Operator cont'd



Triple Pattern Operator cont'd



Properties



- Supports any reach-ability-based query semantics
- Highly adaptive
  - Routing of inter-mediate solutions
  - Inspired by “Eddies” (Anvur & Hellerstein, SIGMOD 2000)

Hypothesis

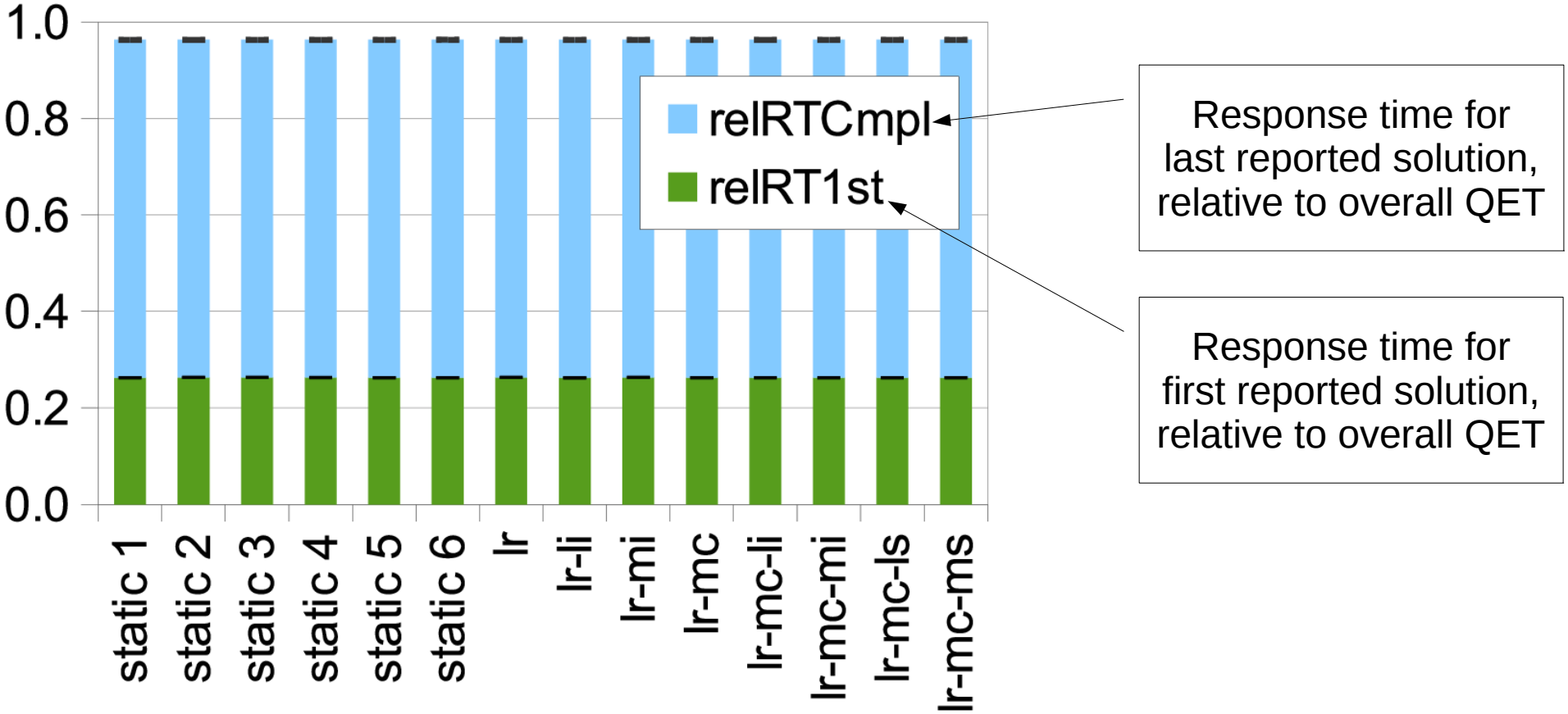
Query execution time (QET) and response time (RT)
can be reduced
by applying a suitable routing policy.

Test with Different Routing Policies

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

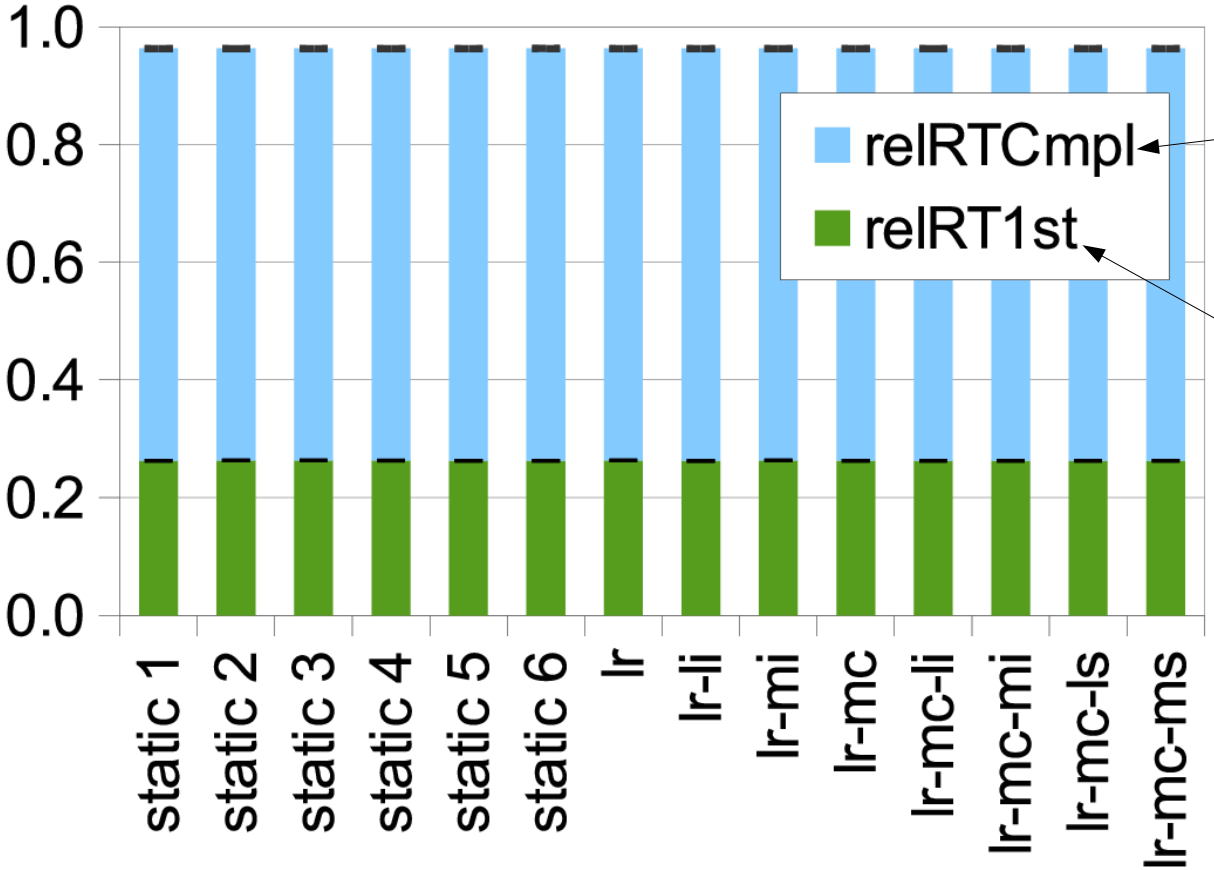
Test with Different Routing Policies

- 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



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



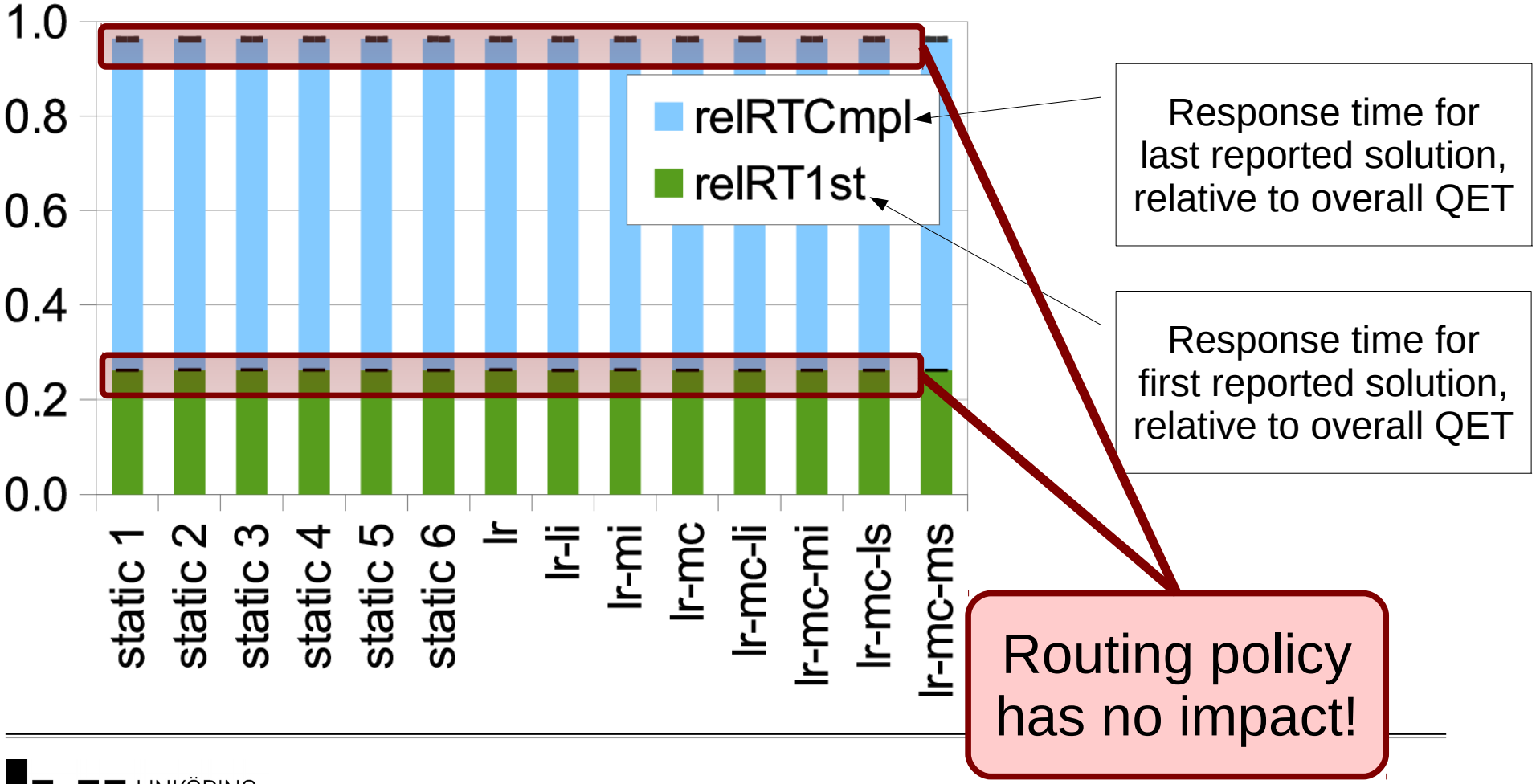
Response time for last reported solution, relative to overall QET

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



Hypothesis

~~*Query execution time (QET) and response time (RT) can be reduced by applying a suitable routing policy*~~

No!

Why?

Another Test: Impact of Data Retrieval?

Query 1 Query 4 Query 5 Query 9 Query 10

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

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

Query 1 Query 4 Query 5 Query 9 Query 10

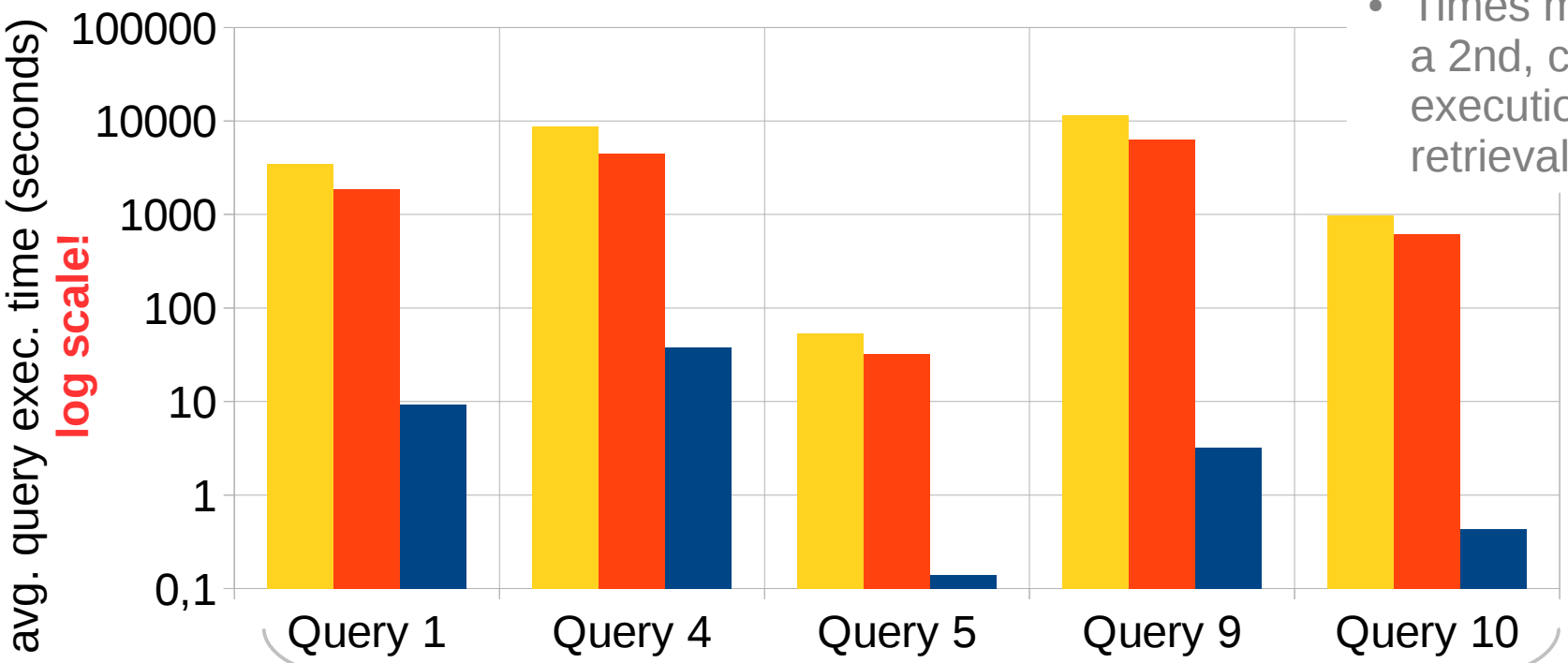
5 queries of the FedBench benchmark suite, executed over *real* Linked Data on the WWW

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)



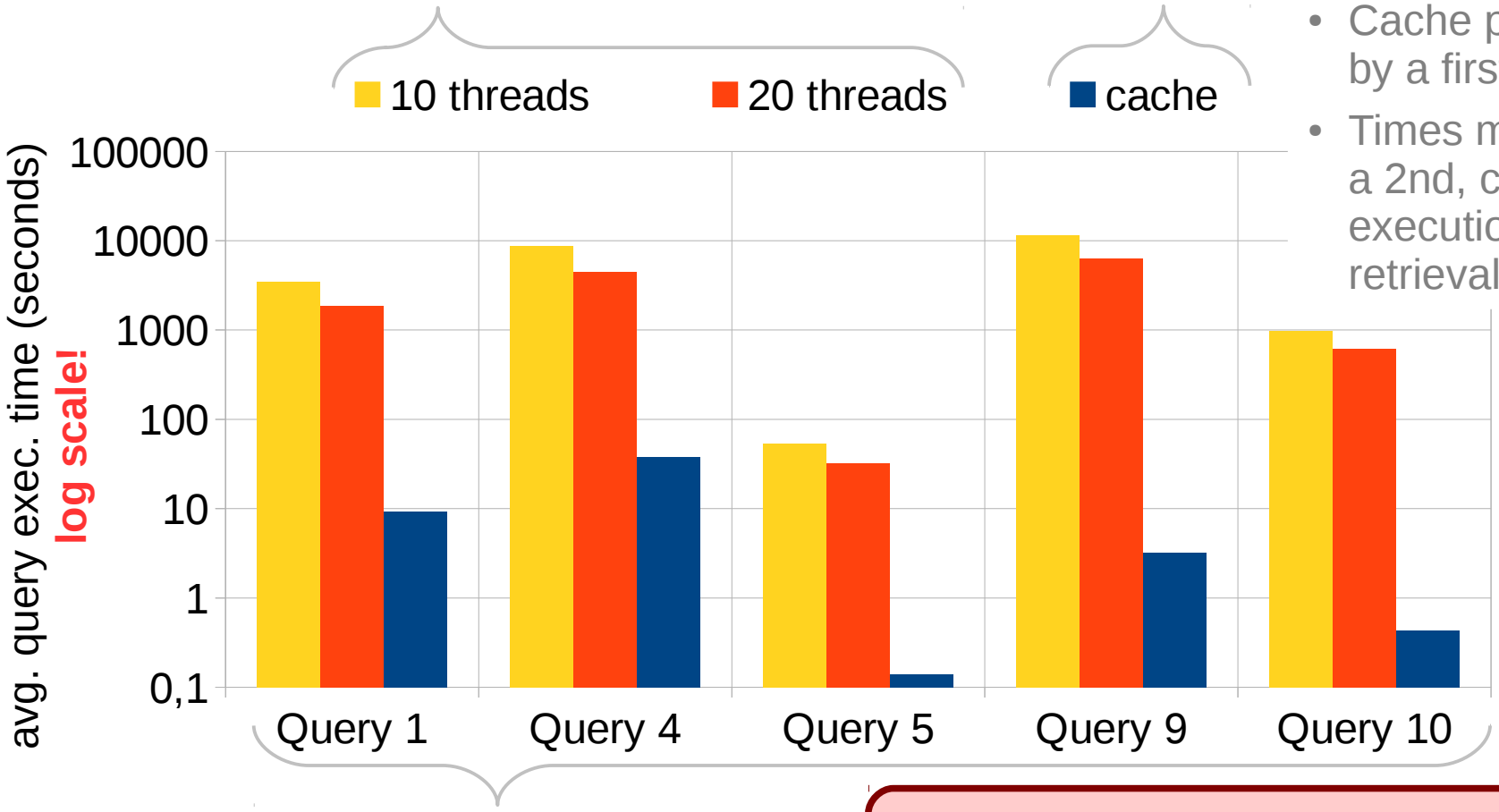
5 queries of the FedBench benchmark suite, executed over *real* Linked Data on the WWW

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)

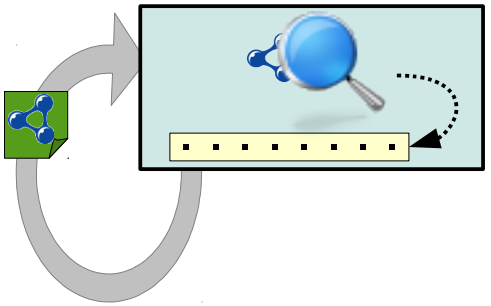


5 queries of the FedBench benchmark suite, executed over *real* Linked Data on the WWW

Approaches to optimize QET *will fail* to be effective

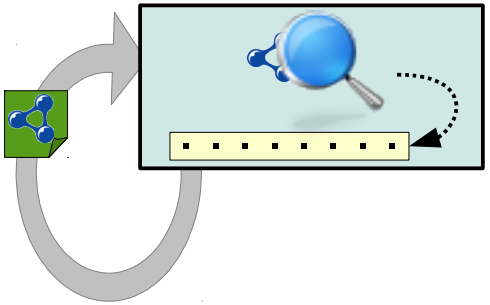
Hypothesis

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



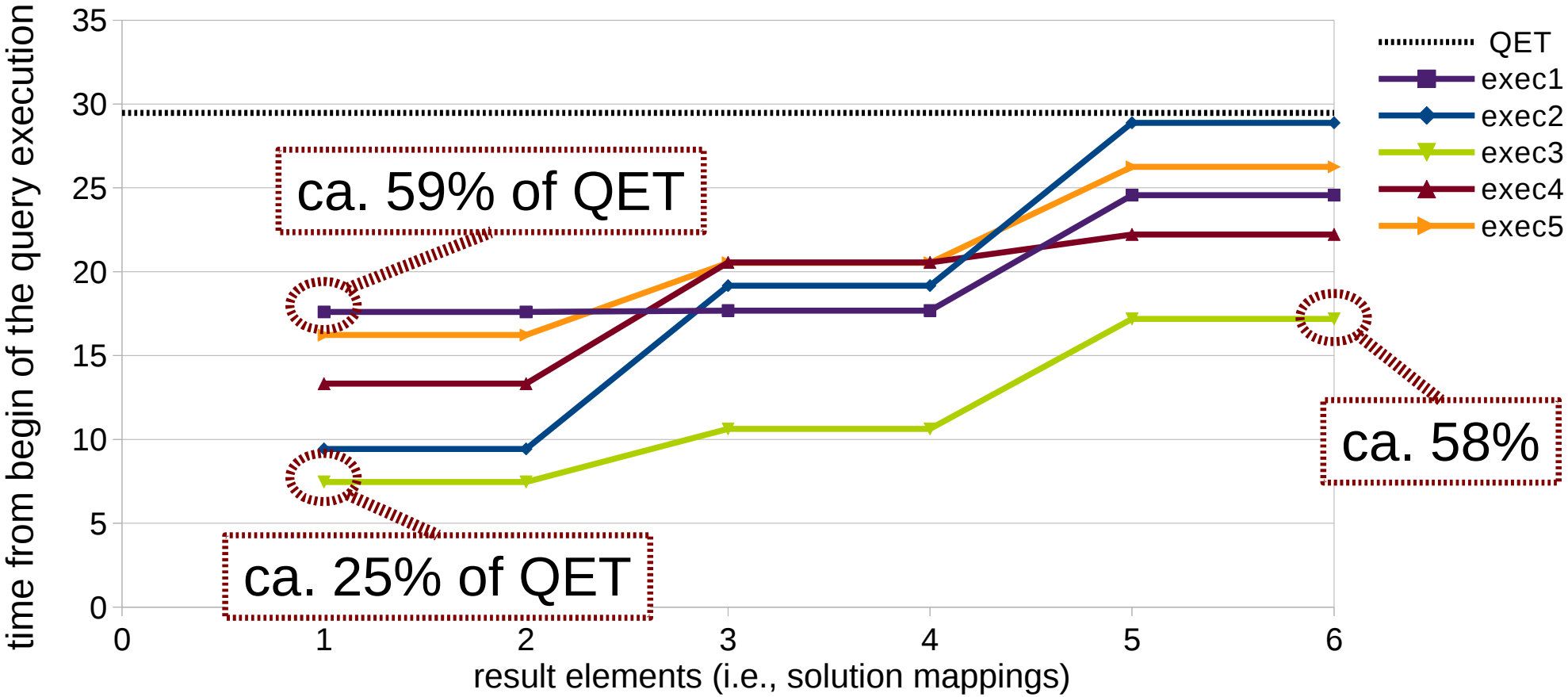
Test: Prioritizing Lookups Randomly

- 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



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

Based on: Olaf Hartig and M. Tamer Özsu: *Walking without a Map: Ranking-Based Traversal for Querying Linked Data*. ISWC 2016.

Olaf Hartig and M. Tamer Özsu: *Optimizing Response Times of Traversal-Based Linked Data Queries (Extended Version). arXiv:1607.01046*

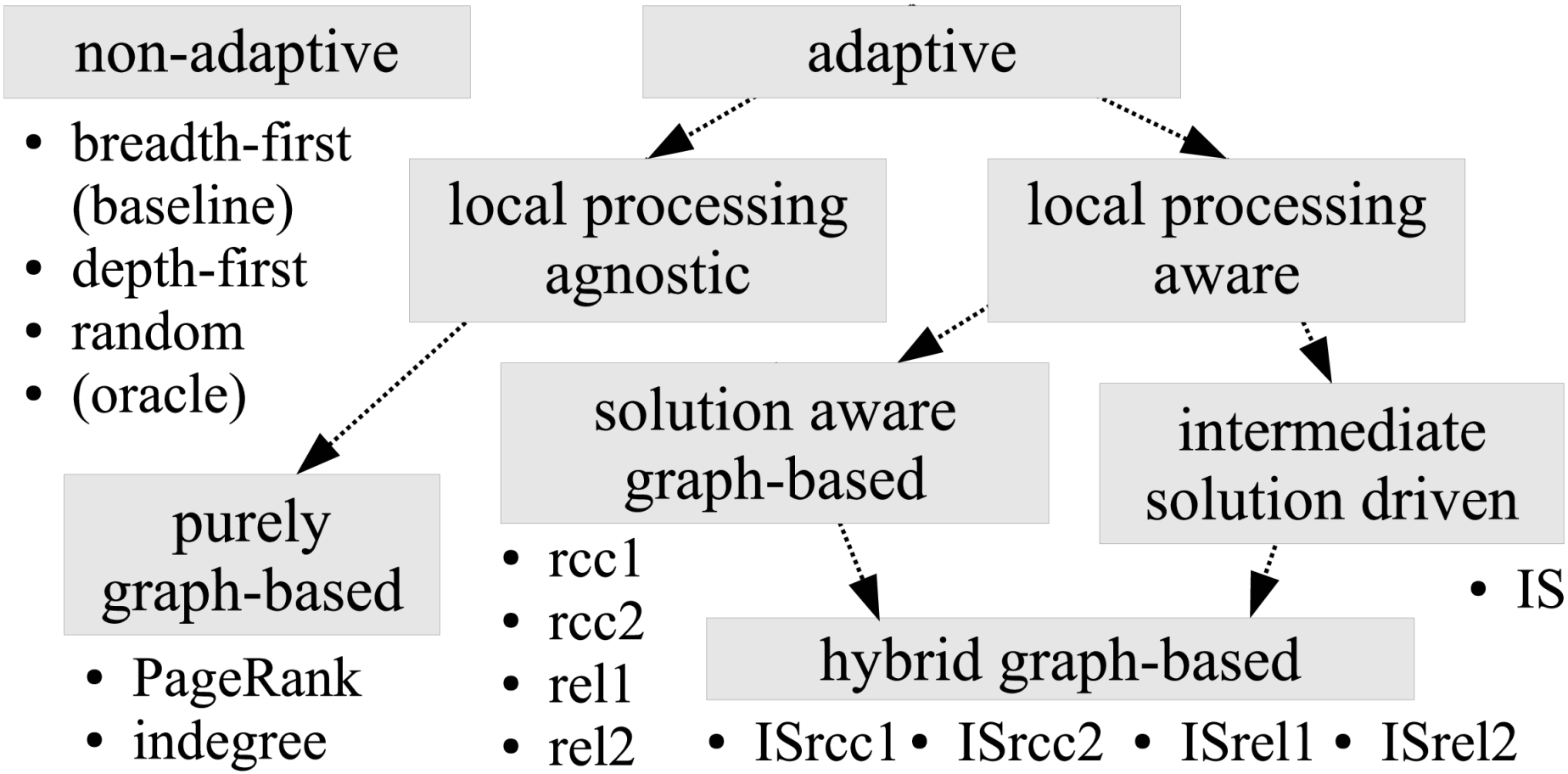
Research Question

What is

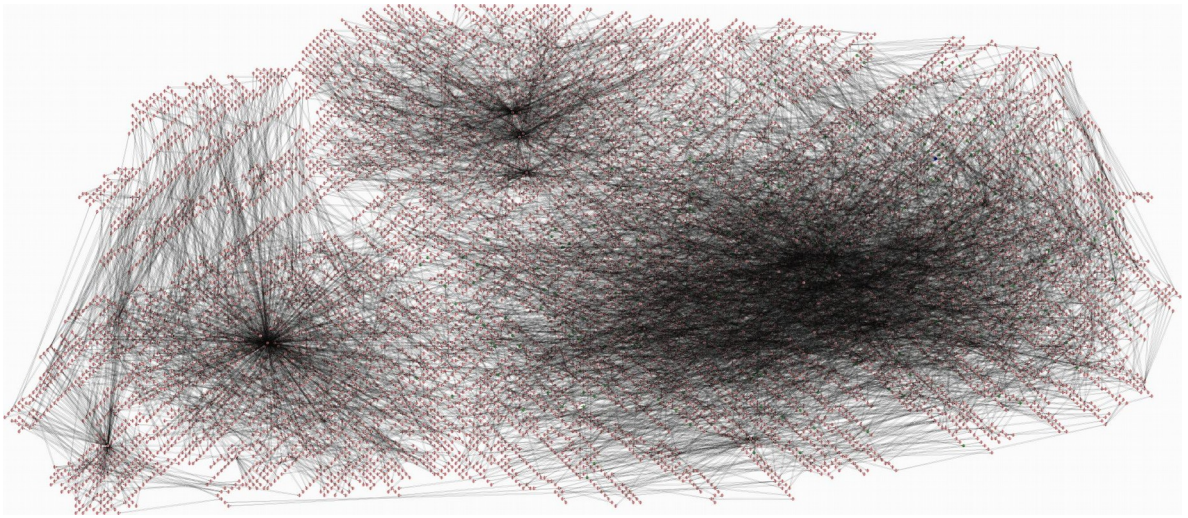
Response time (RT) can be reduced by choosing ***a "good" strategy*** *of prioritizing URI lookups.* √

?

Taxonomy of 14 Approaches

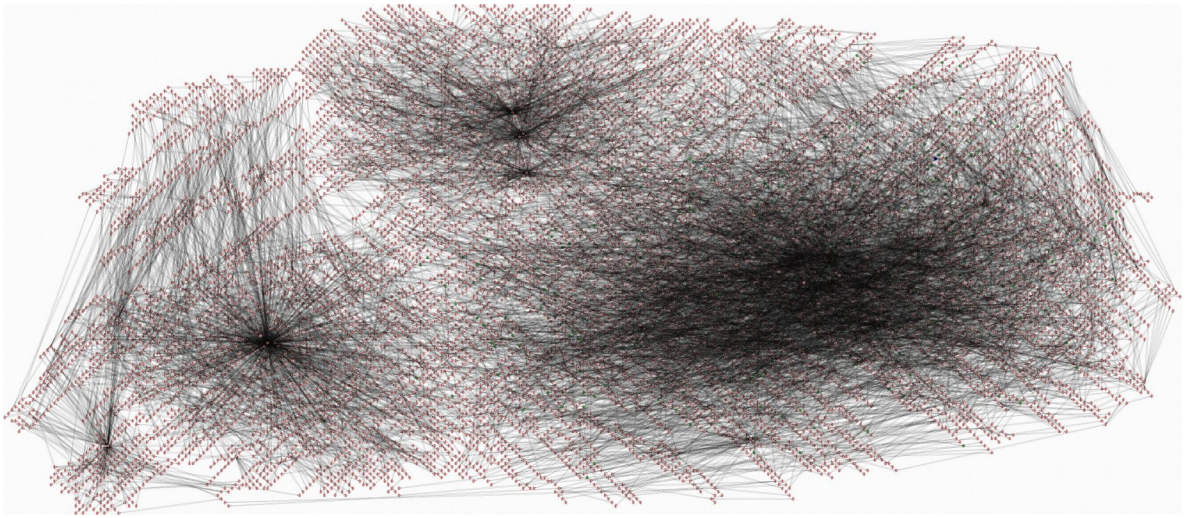


Experiment Setup



- 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, Φ_1 and Φ_2
  - by varying Φ_1 and Φ_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

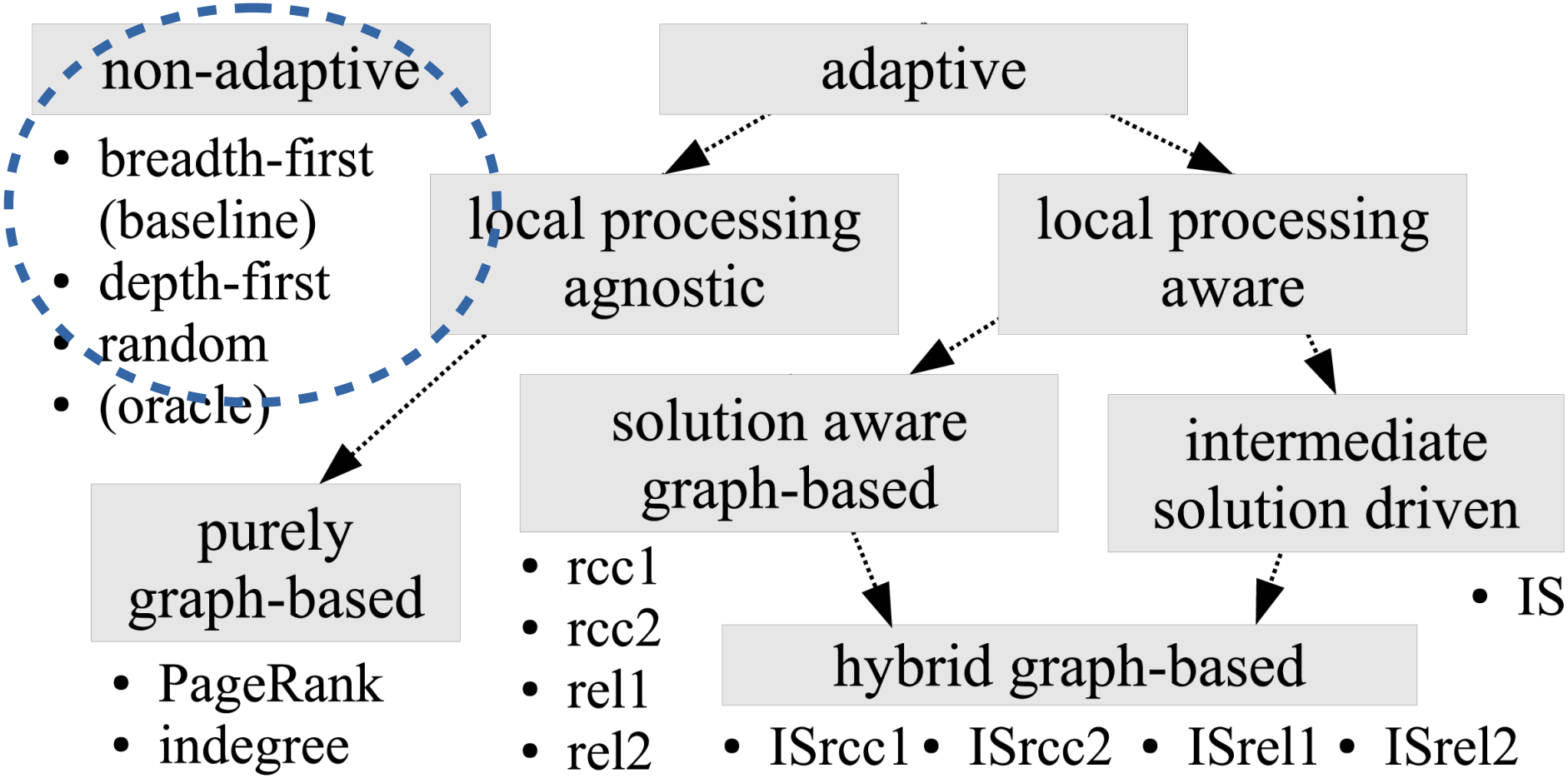
Experiment Setup



- 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, Φ_1 and Φ_2
  - by varying Φ_1 and Φ_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

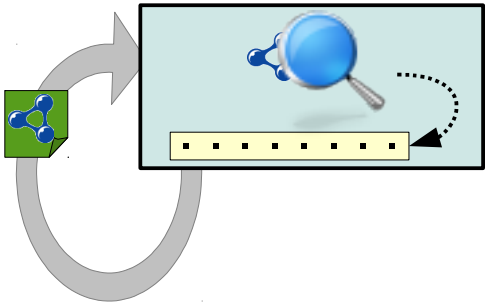
84 test cases

Taxonomy of Approaches



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, Φ_1 and Φ_2
  - by varying Φ_1 and Φ_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

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

	time to a first		time to 50%		time to 100%	
approach	worse	better	worse	better	worse	better
DFS	23.2 %	26.1 %	58.9 %	17.8 %	53.6 %	10.1 %
random	13.0 %	27.5 %	58.9 %	8.2 %	59.4 %	8.7 %

- 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, Φ_1 and Φ_2
  - by varying Φ_1 and Φ_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

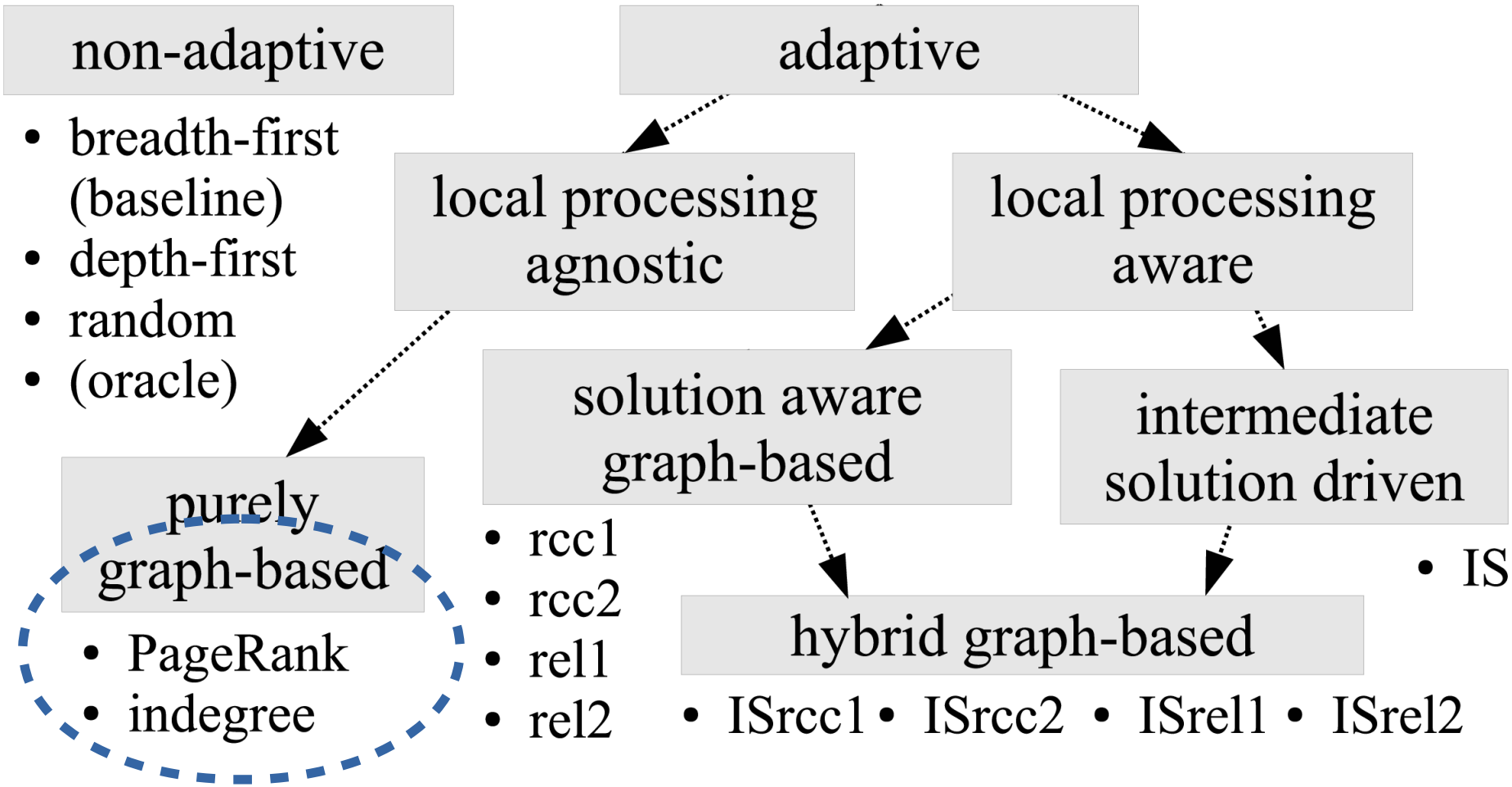
Results

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

	time to a first		time to 50%		time to 100%	
approach	worse	better	worse	better	worse	better
DFS	23.2 %	26.1 %	58.9 %	17.8 %	53.6 %	10.1 %
random	13.0 %	27.5 %	58.9 %	8.2 %	59.4 %	8.7 %

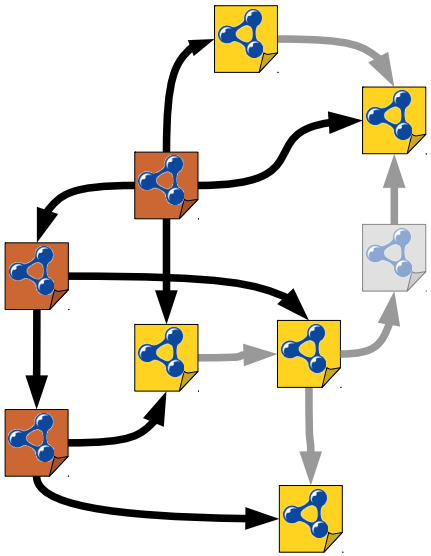
Unsuitable!

Taxonomy of Approaches



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



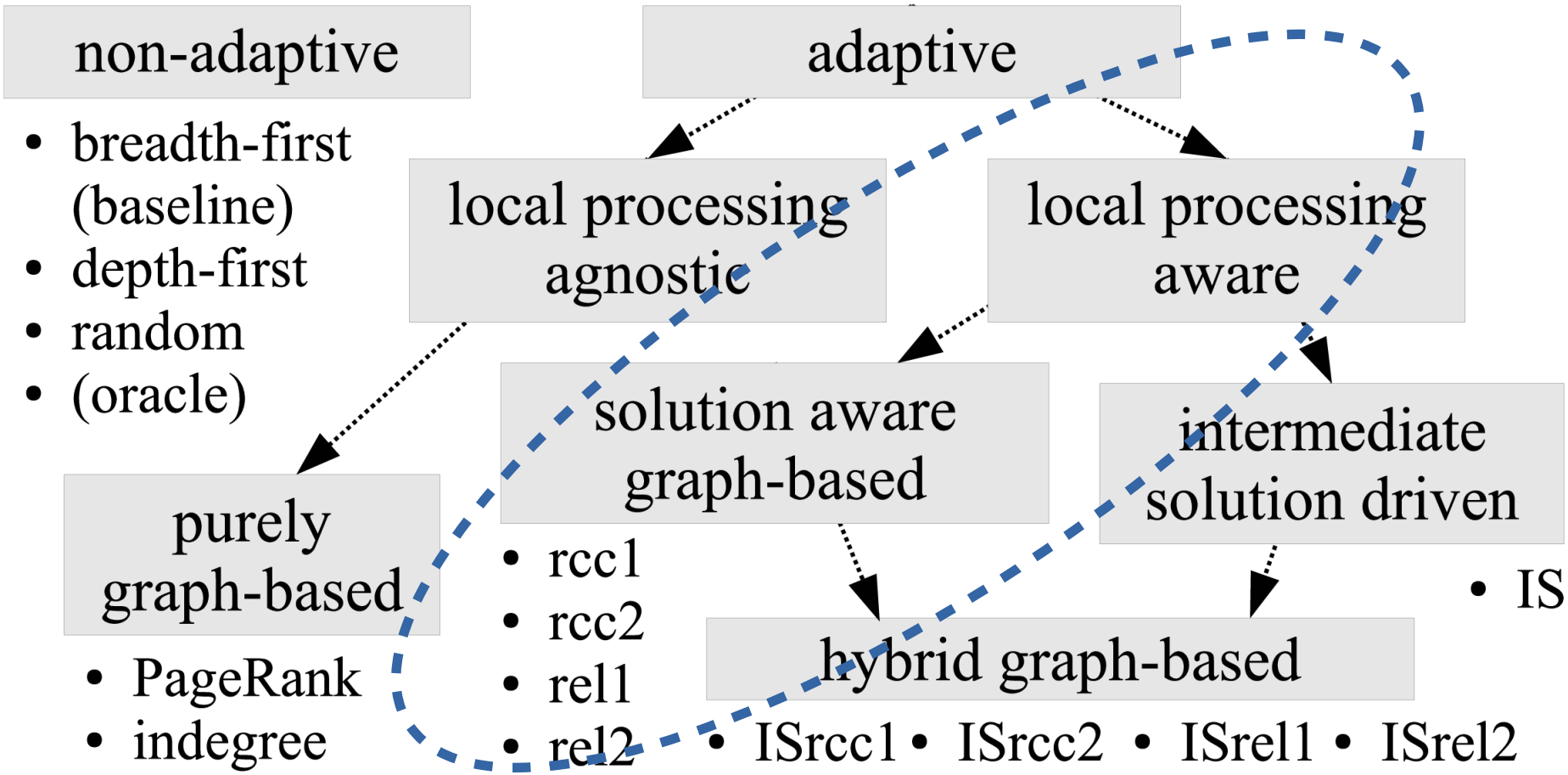
Results

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

	time to a first		time to 50%		time to 100%	
approach	worse	better	worse	better	worse	better
DFS	23.2 %	26.1 %	58.9 %	17.8 %	53.6 %	10.1 %
random	13.0 %	27.5 %	58.9 %	8.2 %	59.4 %	8.7 %
indegree	21.7 %	21.7 %	65.8 %	4.1 %	50.7 %	5.8 %

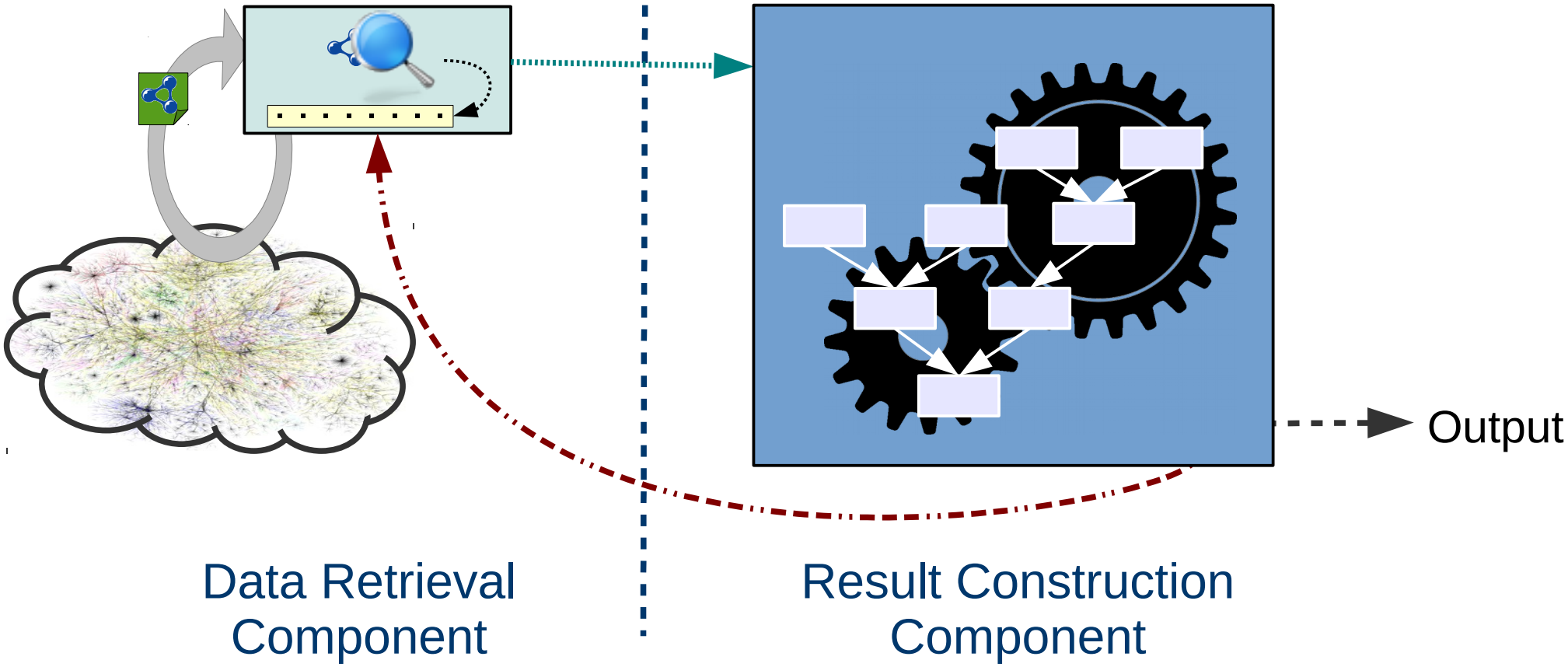
Also unsuitable!

Taxonomy of Approaches



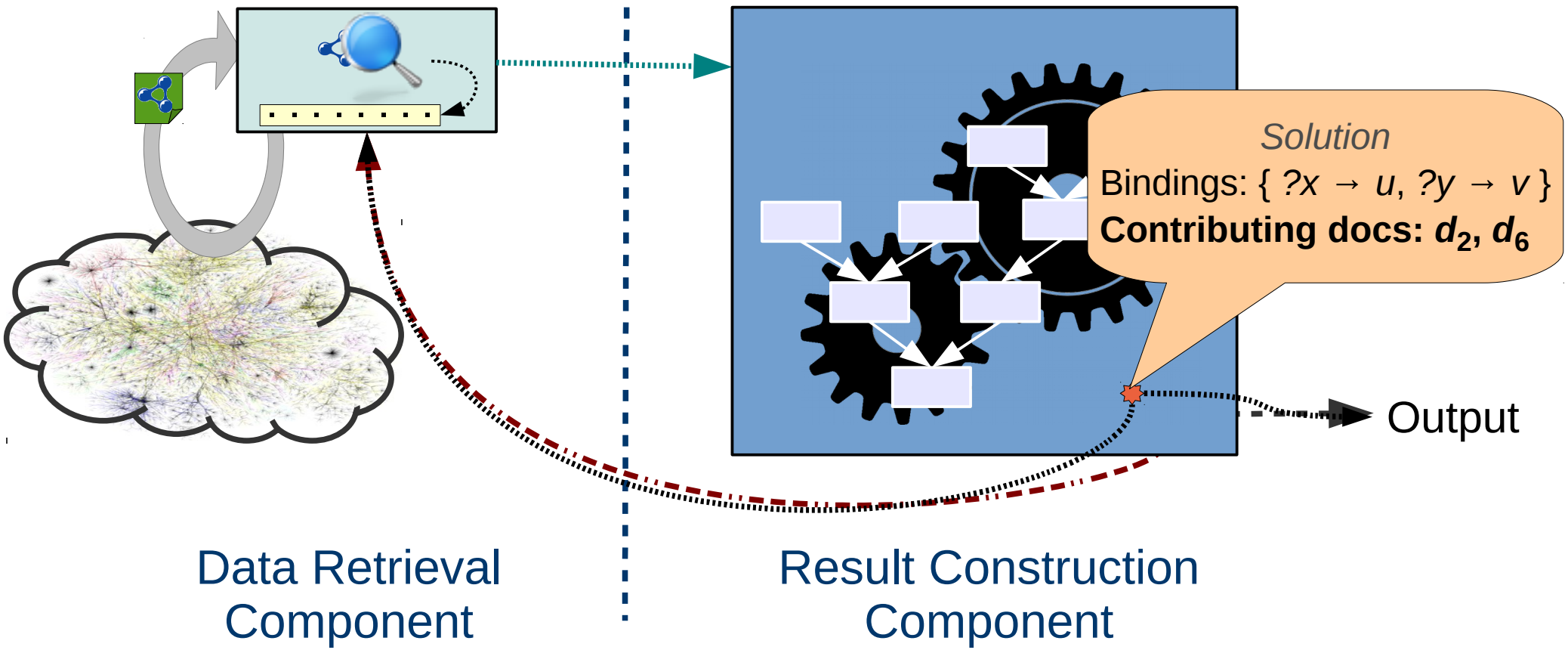
Local-Processing Awareness

- 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

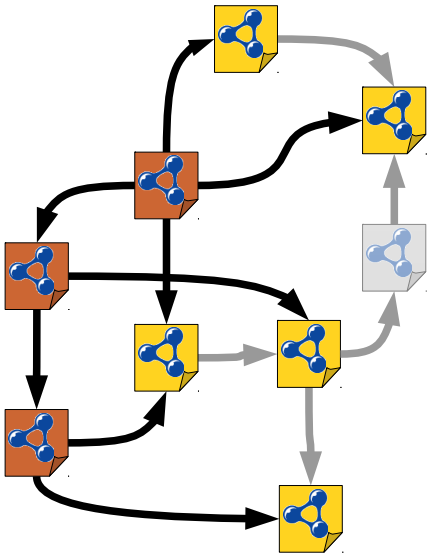


Solution-Based Vertex Scoring

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

- rcc***X***(*doc*) = sum of RCCs of all documents in the *X*-step in-neighborhood of *doc*
  - rcc1, rcc2

- rel***X***(*doc*) = number of documents with RCC > 0 in the *X*-step in-neighborhood of *doc*
  - rel1, rel2



Results

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

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

No effect!

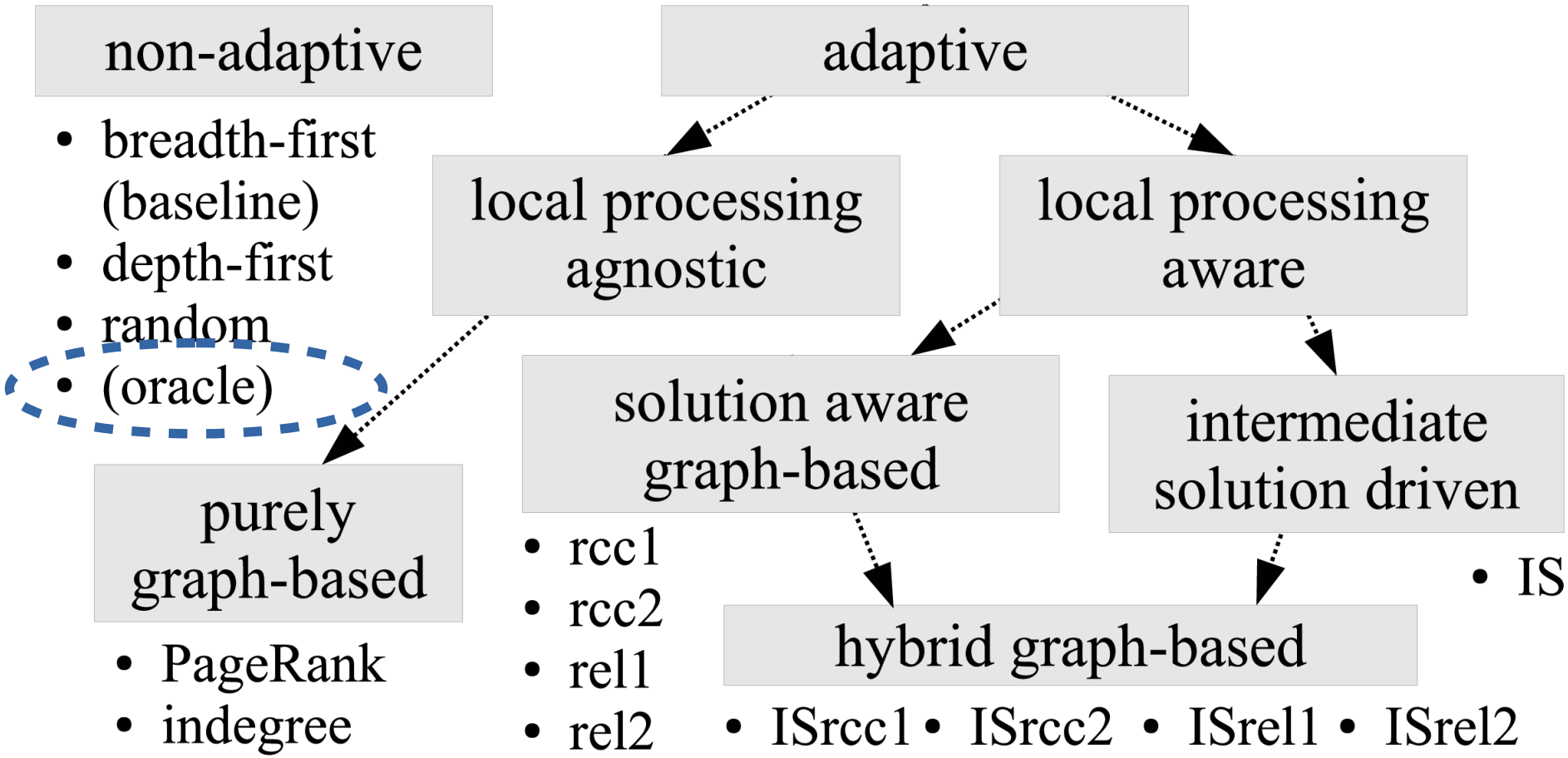
Results

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

approach	time to a first		time to 50%		time to 100%	
	worse	better	worse	better	worse	better
DFS	[illegible]	[illegible]	[illegible]	[illegible]	53.6 %	10.1 %
random	[illegible]	[illegible]	[illegible]	[illegible]	59.4 %	8.7 %
indegree	21.7 %	21.7 %	65.8 %	4.1 %	50.7 %	5.8 %
rcc1	0.0 %	0.0 %	4.1 %	1.4 %	7.2 %	24.6 %
rcc2	0.0 %	0.0 %	2.7 %	2.7 %	4.1 %	20.3 %
rel1	0.0 %	0.0 %	5.5 %	1.4 %	11.6 %	29.0 %
rel2	0.0 %	0.0 %	11.0 %	0.0 %	**2.9 %**	**26.1 %**
intsol	7.2 %	31.9 %	15.1 %	27.4 %	26.1 %	10.1 %
isrcc1	2.9 %	30.4 %	5.5 %	26.0 %	14.5 %	18.8 %
isrcc2	5.8 %	33.3 %	5.5 %	24.7 %	13.0 %	26.1 %
isrel1	**0.0 %**	**33.3 %**	**2.7 %**	**24.7 %**	15.9 %	26.1 %
isrel2	2.9 %	31.9 %	4.1 %	23.3 %	11.6 %	26.1 %

Most suitable (among the tested approaches)

Taxonomy of Approaches



Oracle Approach

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

priority(*uri*) = result contribution counter of the document that will be retrieved by looking up *uri*

where: rcc(*doc*) = number of solutions whose computation requires some triple from *doc*

Such information is available only after executing a query completely

- Oracle cannot be used in practice

Results

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

	time to a first		time to 50%		time to 100%	
approach	worse	better	worse	better	worse	better
DFS	[illegible]	[illegible]	[illegible]	[illegible]	53.6 %	10.1 %
random	[illegible]	[illegible]	[illegible]	[illegible]	59.4 %	8.7 %
indegree	[illegible]	[illegible]	[illegible]	[illegible]	50.7 %	5.8 %
rcc1	0.0 %	0.0 %	4.1 %	1.4 %	7.2 %	24.6 %
rcc2	0.0 %	0.0 %	2.7 %	2.7 %	4.1 %	20.3 %
rel1	0.0 %	0.0 %	5.5 %	1.4 %	11.6 %	29.0 %
rel2	0.0 %	0.0 %	11.0 %	0.0 %	**2.9 %**	**26.1 %**
intsol	7.2 %	31.9 %	15.1 %	27.4 %	26.1 %	10.1 %
isrcc1	2.9 %	30.4 %	5.5 %	26.0 %	14.5 %	18.8 %
isrcc2	5.8 %	33.3 %	5.5 %	24.7 %	13.0 %	26.1 %
isrel1	**0.0 %**	**33.3 %**	**2.7 %**	**24.7 %**	15.9 %	26.1 %
isrel2	2.9 %	31.9 %	4.1 %	23.3 %	11.6 %	26.1 %
oracle	0.0 %	35.3 %	0.0 %	41.2 %	0.0 %	64.7 %

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

Based on: Sijin Cheng and Olaf Hartig: *OPT+: A Monotonic Alternative to OPTIONAL in SPARQL*. J of Web Engineering 18(1-3), 2019.

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

Based on: Sijin Cheng and Olaf Hartig: *OPT+: A Monotonic Alternative to OPTIONAL in SPARQL*. J of Web Engineering 18(1-3), 2019.

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
```

Motivating Example

- 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:

μ_1 = { ?post → ex:post1, ?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: (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:

μ_1 = { ?post → ex:post1, ?text → "Good …" }

μ_2 = { ?post → ex:post1, ?text → "I can …" }

Motivating Example

- 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
```

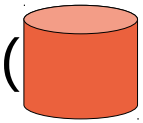
- Intermediate query result contains the following elements:

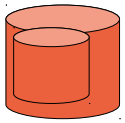
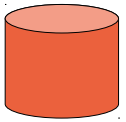
μ_1 = { ?post → ex:post1, ?text → "Good …" }

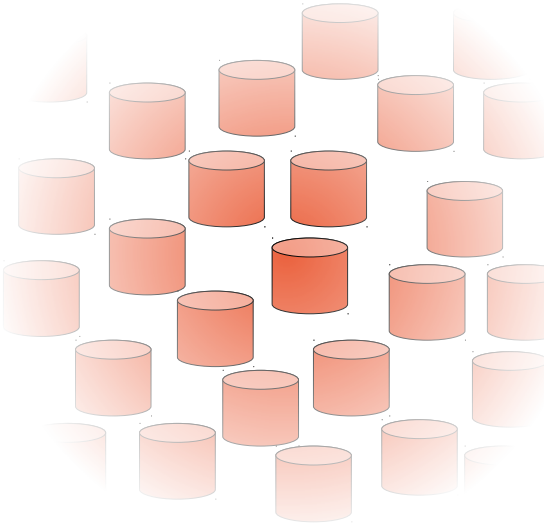
μ_2 = { ?post → ex:post1, ?text → "I can …" }

μ_3 = { ?post → ex:post1, ?text → "Good …", ?img → 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 ( , ) of possible databases, it holds that:

 $\Longrightarrow$ $Q($  $) \subseteq Q($  $)$



- 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 $[\![P]\!]_G = ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \cup ([\![P_1]\!]_G \setminus [\![P_2]\!]_G)$.

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 $[\![P]\!]_G = ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \cup [\![P_1]\!]_G$.
- Properties:
  - $(P_1 \; OPT^+ \; P_2) \equiv ((P_1 \; AND \; P_2) \; UNION \; P_1)$
  - 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 $[\![P]\!]_G = ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \cup ([\![P_1]\!]_G \setminus [\![P_2]\!]_G)$.

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: $(P_1 \, OPT^+ \, P_2) \equiv ((P_1 \, AND \, P_2) \, UNION \, P_1)$
  - execute both versions over the corresponding dataset (either using the original SPARQL endpoint or a local triplestore with the dataset loaded)

Results

log	% of pairs with same result size	at most 2x increase	at most 10x increase	at most 100x increase	at most 1000x increase
All pairs of test queries (for which there were no errors)					
$DBP_{3.5.1}$	65,74%	71,23%	80,86%	91,46%	99,57%
WD_{all}	67,19%	98,90%	99,88%	99,89%	99,90%
WD_{org}	98,73%	99,73%	99,95%	99,95%	99,95%
LGD	98,46%	99,88%	100,00%	100,00%	100,00%

Same result for a large fraction of queries

Non-negligible number of case with substantial increase in result size

Results

log	% of pairs with same result size	at most 2x increase	at most 10x increase	at most 100x increase	at most 1000x increase
All pairs of test queries (for which there were no errors)					
$DBP_{3.5.1}$	65,74%	71,23%	80,86%	91,46%	99,57%
WD_{all}	67,19%	98,90%	99,88%	99,89%	99,90%
WD_{org}	98,73%	99,73%	99,95%	99,95%	99,95%
LGD	98,46%	99,88%	100,00%	100,00%	100,00%
Only the pairs whose OPTIONAL query did *not* contain sequences of OPTIONALs					
$DBP_{3.5.1}$	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					
$DBP_{3.5.1}$	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 OPT P_R)
I_L := iterator over result of P_L
for each μ from I_L **do**
    P_R' := $\mu[P_R]$
    I_R' := iterator over result of P_R'
    if I_R' is has solution mappings **then**
        for each μ' from I_R' **do**
            μ'' := $\mu \cup \mu$'
            output μ''
    else
        output μ

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 μ from I_L **do**
    P_R' := $\mu[P_R]$
    I_R' := iterator over result of P_R'
    if I_R' is has solution mappings **then**
        output μ
        for each μ' from I_R' **do**
            μ'' := $\mu \cup \mu'$
            output μ''
    else
        output μ

mNLJ$^+$ Algorithm for OPT$^+$

- Another adaptation; first consumes the LHS input completely

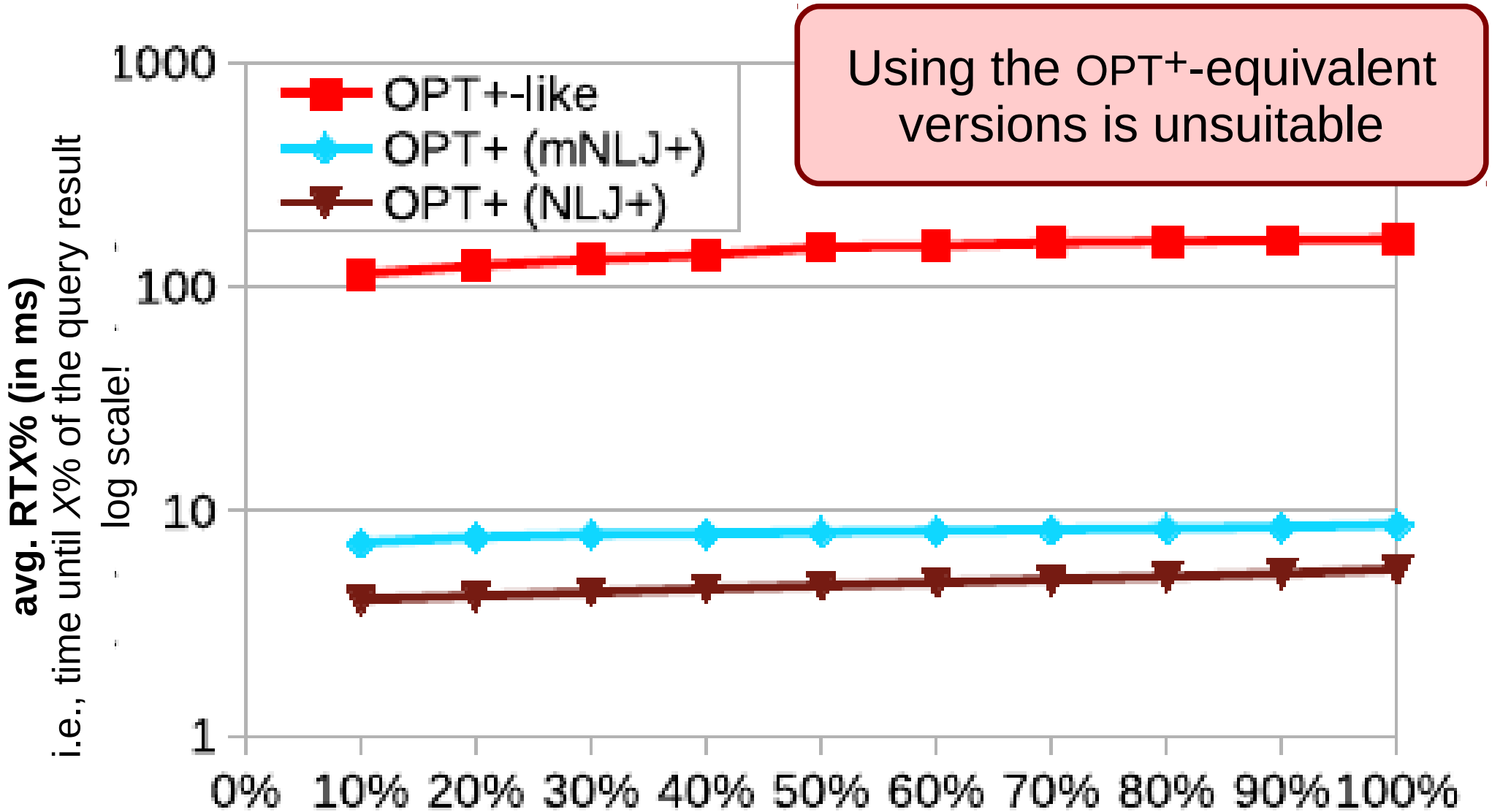
Input: (P_L OPT P_R)
I_L := iterator over result of P_L
M_L := empty list of solution mappings
for each μ from I_L **do**
    output μ, and append μ to M_L
for each μ in M_L **do**
    P_R' := $\mu[P_R]$
    I_R' := iterator over result of P_R'
    if I_R' is has solution mappings **then**
        for each μ' from I_R' **do**
            μ'' := $\mu \cup \mu$'
            output μ''
    else
        output μ

Test Queries

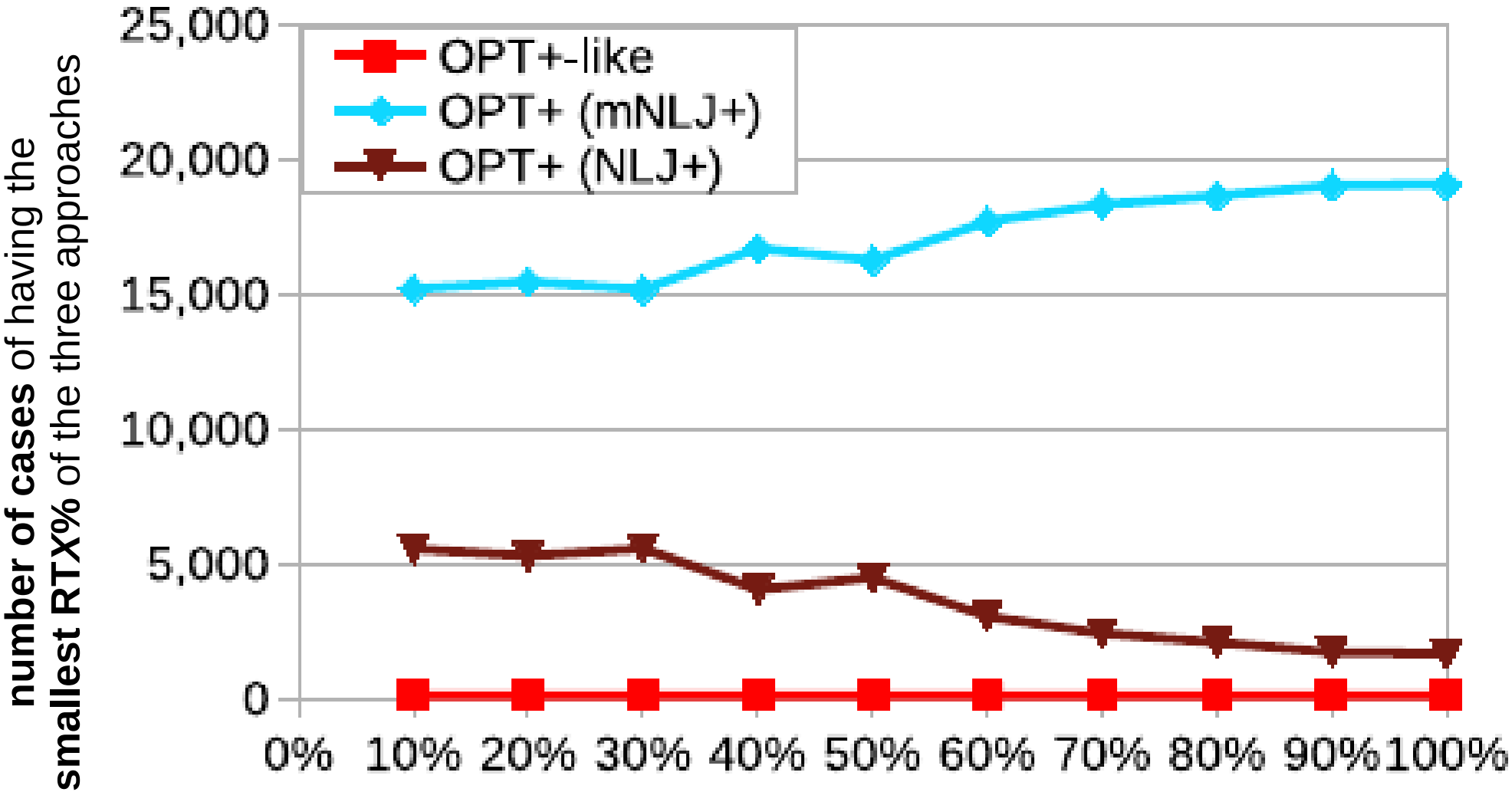
- 60K distinct OPTIONAL queries selected randomly from the DBpedia 3.5.1 query log
  - this log is comparably diverse in terms of
    i) how OPTIONAL is used and
    ii) result-size increase when OPT^+ is used

- 20.8K of them have a non-empty result and can be used

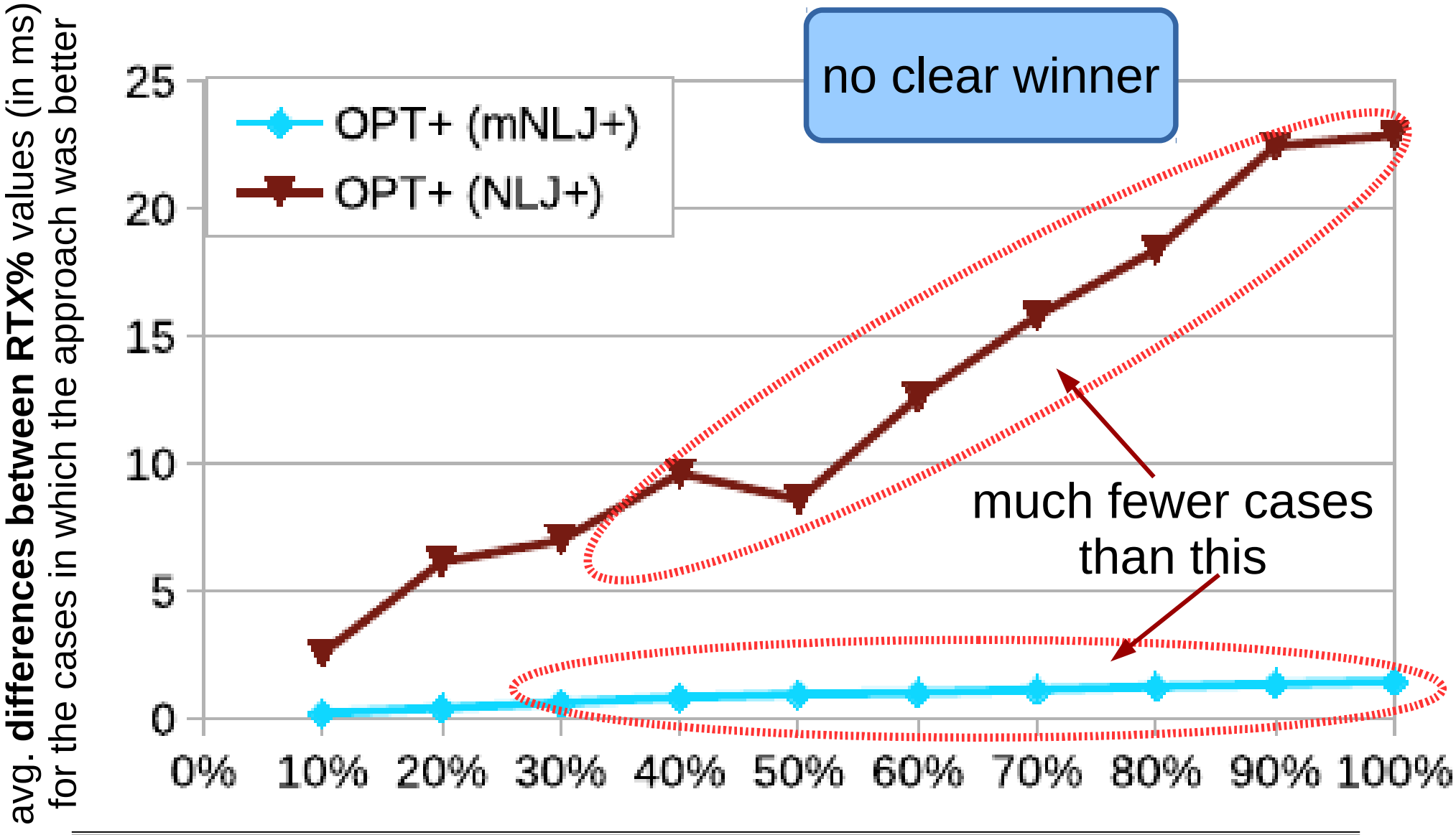
Comparison of all OPT+ Approaches



Comparison of all OPT$^+$ Approaches (cont'd)

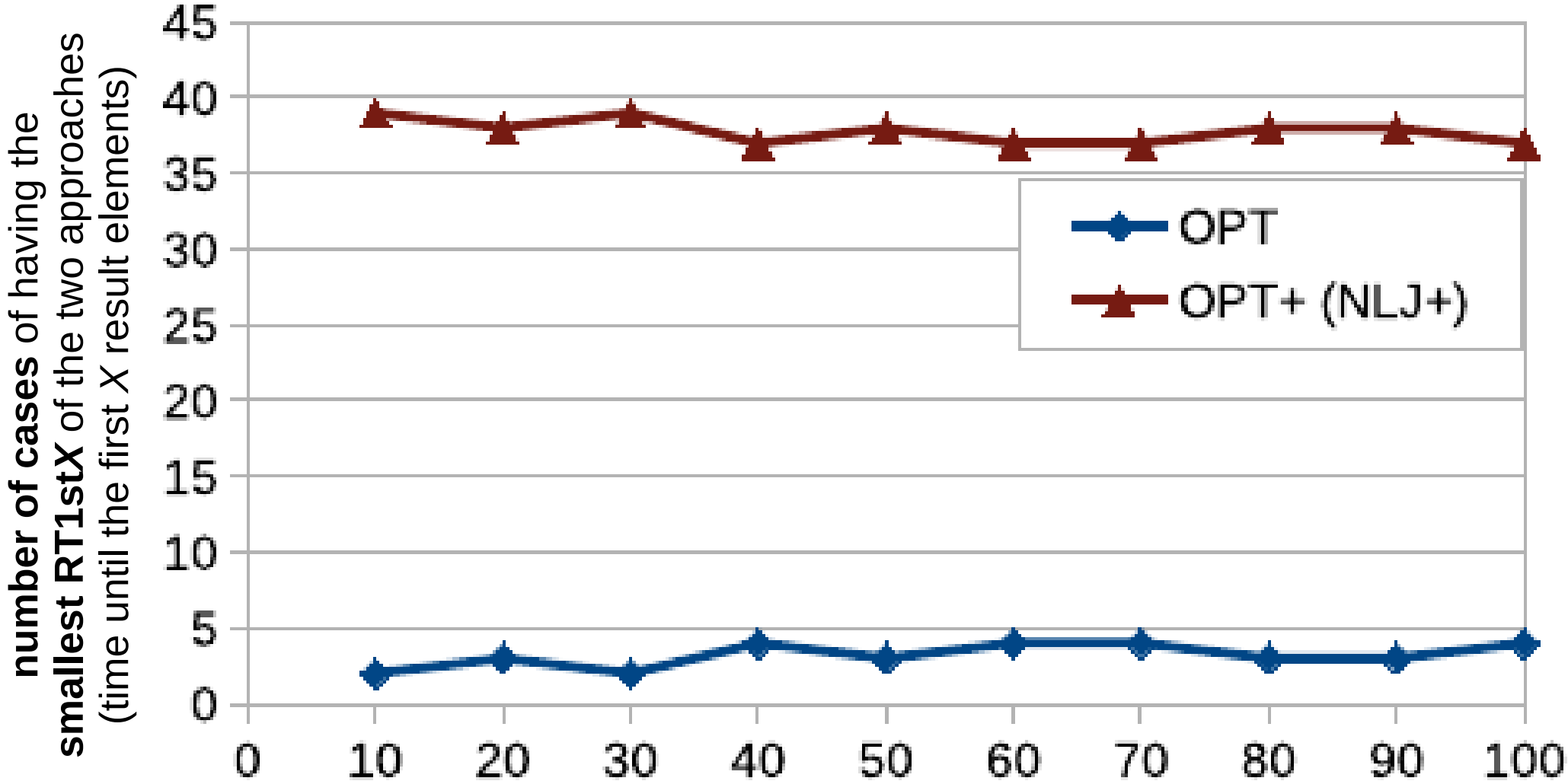


mNLJ$^+$ vs NLJ$^+$

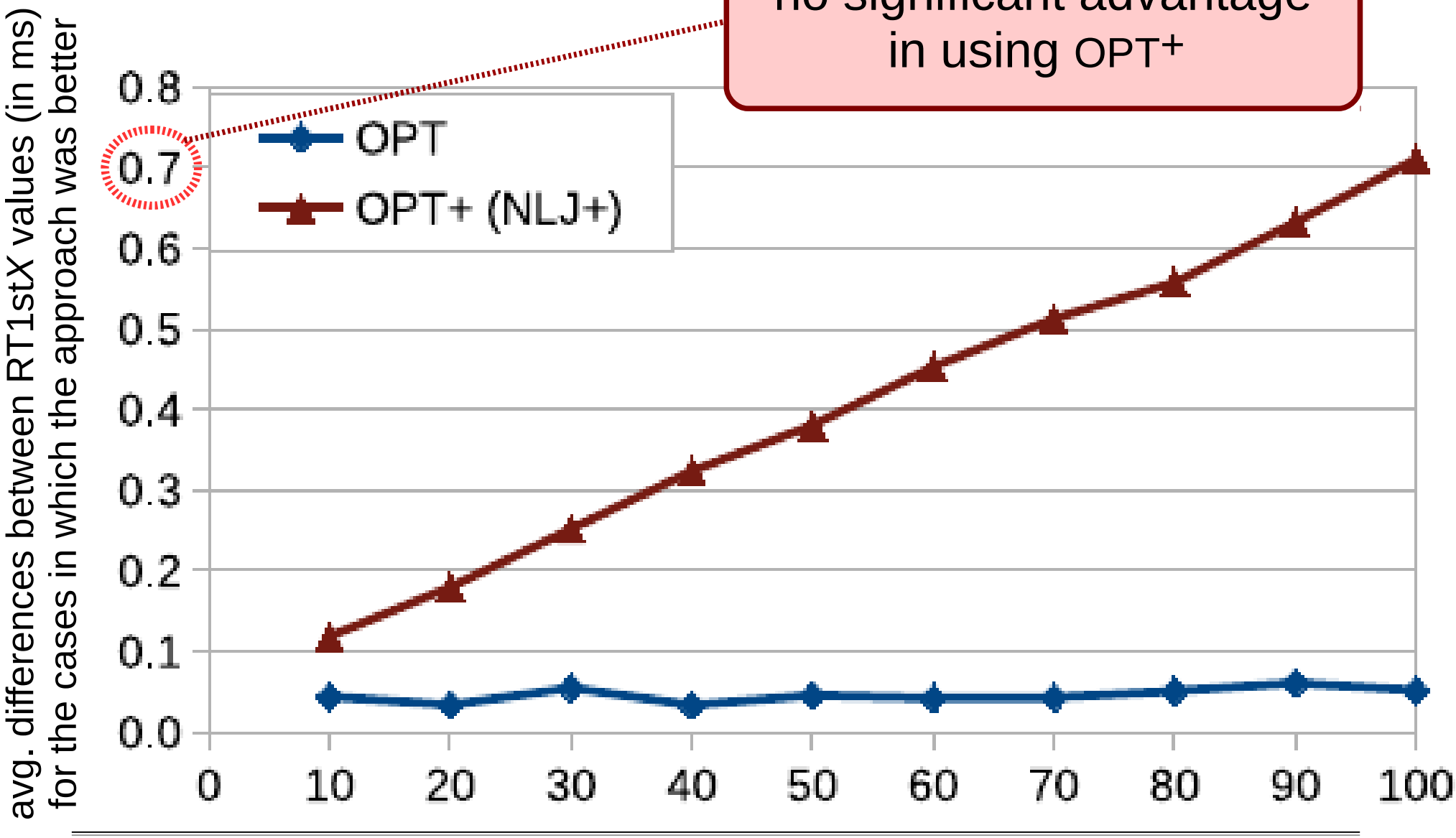

avg. **differences between RTX%** values (in ms)
for the cases in which the approach was better
25
20
15
10
5
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
OPT+ (mNLJ+)
OPT+ (NLJ+)
no clear winner
much fewer cases
than this

OPT vs NLJ$^+$

(only 41 query for which the OPT version has a result size ≥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
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
60
70
80
90
100
OPT
OPT+ (NLJ+)

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

www.liu.se

LINKÖPING
UNIVERSITY