Scalable Multi-Query Optimization for SPARQL

Wangchao Le[1] Anastasios Kementsietsidis[2] Songyun Duan[2] Feifei Li[1]

[1]University of Utah

[2]IBM Research

April 2, 2012

Outline

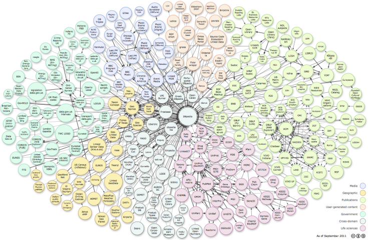
Outline

Introduction

- We are inundated with a large collection of RDF (Resource Description Framework) data.

Introduction

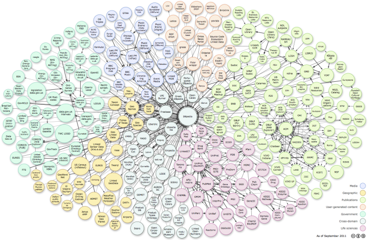
- We are inundated with a large collection of RDF (Resource Description Framework) data.
    - DBpedia, Uniprot, Freebase etc



Internally ...

```
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:dcterms="http://purl.org/dc/terms/">
        <rdf:Description rdf:about="urn:x-states:New York">
            <dcterms:alternative>NY</dcterms:alternative>
        </rdf:Description>
</rdf:RDF>
```

- We are inundated with a large collection of RDF (Resource Description Framework) data.
    - DBpedia, Uniprot, Freebase etc



Internally ...

```
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:dcterms="http://purl.org/dc/terms/">
        <rdf:Description rdf:about="urn:x-states:New York">
            <dcterms:alternative>NY</dcterms:alternative>
        </rdf:Description>
</rdf:RDF>
```

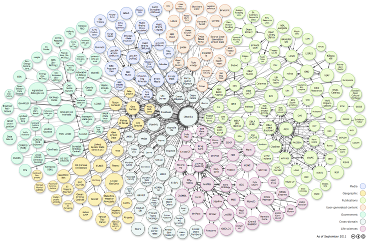
Triple format:

<http://.../New York> <http://purl.org/dc/terms/alternative> "NY" .

subject predicate object

Introduction

- We are inundated with a large collection of RDF (<u>R</u>esource <u>D</u>escription <u>F</u>ramework) data.
  - DBpedia, Uniprot, Freebase etc
  - A large graph and encode rich semantics



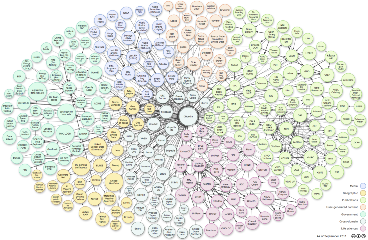
Internally ...

```
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:dcterms="http://purl.org/dc/terms/">
        <rdf:Description rdf:about="urn:x-states:New York">
            <dcterms:alternative>NY</dcterms:alternative>
        </rdf:Description>
</rdf:RDF>
```

Triple format:
<http://.../New York> <http://purl.org/dc/terms/alternative> "NY" .
subject predicate object

Introduction

- We are inundated with a large collection of RDF (Resource Description Framework) data.
  - DBpedia, Uniprot, Freebase etc
  - A large graph and encode rich semantics



Internally ...

```
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:dcterms="http://purl.org/dc/terms/">
        <rdf:Description rdf:about="urn:x-states:New York">
            <dcterms:alternative>NY</dcterms:alternative>
        </rdf:Description>
</rdf:RDF>
```

Triple format:

<http://.../New York> <http://purl.org/dc/terms/alternative> "NY" .

subject predicate object

Query language: SPARQL

Introduction

- We are inundated with a large collection of RDF (Resource Description Framework) data.
    - DBpedia, Uniprot, Freebase etc
    - A large graph and encode rich semantics
- Available engines to manage RDF data?

Introduction

- We are inundated with a large collection of RDF (Resource Description Framework) data.
    - DBpedia, Uniprot, Freebase etc
    - A large graph and encode rich semantics
- Available engines to manage RDF data?
    - RDBMS: Migrate RDF, *e.g.*, Sesame, JenaSDB etc.
    - Generic RDF stores: *e.g.*, RDF3X, JenaTDB etc.

[VP07] D.J. Abadi, et al. Scalable semantic web data management using vertical partitioning. In *VLDB*, 2007.

[HEX08] C. Weiss, et al. Hexastore: sextuple indexing for semantic web data management. In *VLDB*, 2008.

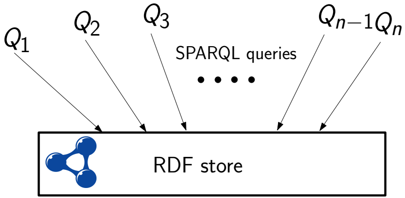
[RDF3X] T. Neumann, G. Weikum. RDF-3X: a RISC-style engine for RDF. In *VLDB*, 2008.

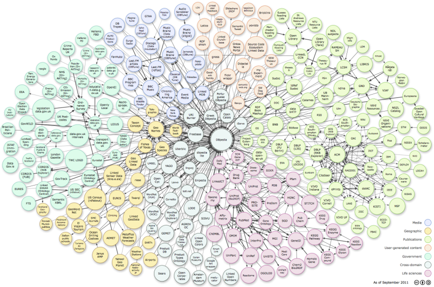
[SJP09] T. Neumann, G. Weikum. Scalable Join Processing on Very Large RDF Graphs. In *SIGMOD*, 2009.

[BM10] M. Atre, et al. Matrix "Bit" loaded: A Scalable Lightweight Join Query Processor for RDF Data. In *WWW*, 2010.

[SSQ11] J. Huang, et al. Scalable SPARQL Querying of Large RDF Graphs. In *VLDB*, 2011.

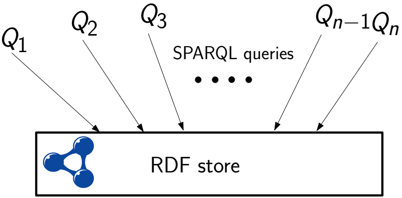
Introduction

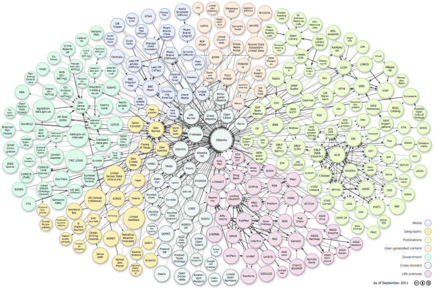




Introduction

- Observation: queries share common parts
- Multi-query optimization





Introduction

- A tempting choice: turn to MQO in relational databases [MQO88][MQO90][MQO00]
  - SPARQL$\leftrightarrow$relational algebra [EPS08][FSR07].
  - Exist quite a few relational solutions for RDF store.

[MQO90] T. Sellis, et al. On the Multiple-Query Optimization Problem. In *TKDE*, 1990.

[MQO88] T. Sellis, et al. Multiple-query optimization. In *TODS*, 1988.

[MQO00] P. Roy, et al. Efficient and extensible algorithms for multi query optimization. In *SIGMOD*, 2000.

[EPS08] R. Angles, et al. The Expressive Power of SPARQL. In *ISWC*, 2008.

[FSR07] A. Polleres, et al. From SPARQL to rules (and back). In *WWW*, 2007.

Introduction

- A tempting choice: turn to MQO in relational databases [MQO88][MQO90][MQO00]
  - SPARQL$\leftrightarrow$relational algebra [EPS08][FSR07].
  - Exist quite a few relational solutions for RDF store.
- For SPARQL and RDF, new issues arise in practice.

Introduction

- A tempting choice: turn to MQO in relational databases [MQO88][MQO90][MQO00]
    - SPARQL$\leftrightarrow$relational algebra [EPS08][FSR07].
    - Exist quite a few relational solutions for RDF store.
- For SPARQL and RDF, new issues arise in practice.
    - Convert SPARQL to SQL: not all engines use RDBMS
    - Conversion to SQL $\rightarrow$ a large number of joins

Introduction

- A tempting choice: turn to MQO in relational databases [MQO88][MQO90][MQO00]
  - SPARQL↔relational algebra [EPS08][FSR07].
  - Exist quite a few relational solutions for RDF store.
- For SPARQL and RDF, new issues arise in practice.
  - Convert SPARQL to SQL: not all engines use RDBMS
  - Conversion to SQL → a large number of joins
  - Store dependent solution

Outline

Preliminary

- We foucs on two types of queries

Preliminary

- We foucs on two types of queries

  Type 1: Q := SELECT RD WHERE GP

  Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^+$

Preliminary

- We foucs on two types of queries

Type 1: Q := SELECT RD WHERE GP

Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^{+}$

subj	pred	obj
p1	name	"Alice"
p1	zip	10001
p1	mbox	alice@home
p1	mbox	alice@work
p1	www	http://home/alice
p2	name	"Bob"
p2	zip	10001
p3	name	"Ella"
p3	zip	10001
p3	www	http://work/ella
p4	name	"Tim"
p4	zip	"11234"

(a) triple table D

```
SELECT ?name
WHERE { ?x name ?name, ?x zip 10001,

}
```

(b) Example query Q_{OPT}

name
"Alice"
"Bob"
"Ella"

Preliminary

- We foucs on two types of queries

Type 1: Q := SELECT RD WHERE GP

Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^{+}$

subj	pred	obj
p1	name	"Alice"
p1	zip	10001
p1	mbox	alice@home
p1	mbox	alice@work
p1	www	http://home/alice
p2	name	"Bob"
p2	zip	10001
p3	name	"Ella"
p3	zip	10001
p3	www	http://work/ella
p4	name	"Tim"
p4	zip	"11234"

(a) triple table D

```
SELECT ?name , ?mail, ?hpage
WHERE  { ?x name ?name, ?x zip 10001,
       OPTIONAL {?x mbox ?mail }
       OPTIONAL {?x www ?hpage }}
```

(b) Example query Q_{OPT}

name
"Alice"
"Bob"
"Ella"

Preliminary

- We foucs on two types of queries

Type 1: Q := SELECT RD WHERE GP

Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^+$

subj	pred	obj
p1	name	"Alice"
p1	zip	10001
p1	mbox	alice@home
p1	mbox	alice@work
p1	www	http://home/alice
p2	name	"Bob"
p2	zip	10001
p3	name	"Ella"
p3	zip	10001
p3	www	http://work/ella
p4	name	"Tim"
p4	zip	"11234"

(a) triple table D

```
SELECT ?name , ?mail, ?hpage
WHERE { ?x name ?name, ?x zip 10001,
        OPTIONAL {?x mbox ?mail }
        OPTIONAL {?x www ?hpage }}
```

(b) Example query Q_{OPT}

name	mail	hpage
"Alice"	alice@home	http://home/alice
"Alice"	alice@work	http://home/alice
"Bob"		
"Ella"		http://work/ella

(c) Output Q_{OPT}(D)

Preliminary

- We foucs on two types of queries

  Type 1: Q := SELECT RD WHERE GP

  Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^{+}$

- Problem statement.

Preliminary

- We foucs on two types of queries

  Type 1: Q := SELECT RD WHERE GP

  Type 2: Q_{OPT} := SELECT RD WHERE GP (OPTIONAL GP_{OPT})$^{+}$

- Problem statement.
  - Input: a set $\mathcal{Q}$ of **Type 1** queries and a data graph G

Preliminary

- We foucs on two types of queries

  Type 1: Q := SELECT RD WHERE GP

  Type 2: $\mathrm{Q}_{\mathrm{OPT}}$:= SELECT RD WHERE GP (OPTIONAL $\mathrm{GP}_{\mathrm{OPT}})^{+}$

- Problem statement.
  - Input: a set $\mathcal{Q}$ of **Type 1** queries and a data graph G
  - Output: a set of rewritten queries, $\mathcal{Q}_{\mathrm{OPT}}$ of **Type 1** and **Type 2** queries

Preliminary

- We foucs on two types of queries

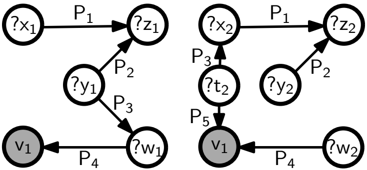
  Type 1: Q := SELECT RD WHERE GP

  Type 2: $\mathrm{Q}_{\mathrm{OPT}}$:= SELECT RD WHERE GP (OPTIONAL $\mathrm{GP}_{\mathrm{OPT}})^{+}$

- Problem statement.
  - Input: a set $\mathcal{Q}$ of **Type 1** queries and a data graph G
  - Output: a set of rewritten queries, $\mathcal{Q}_{\mathrm{OPT}}$ of **Type 1** and **Type 2** queries
  - Requirements:
    - soundness and completeness: $\mathcal{Q}_{\mathrm{OPT}}(\mathrm{G}) \equiv \mathcal{Q}\ (\mathrm{G})$.
    - cost: $\frac{\mathcal{T}_r(\mathcal{Q})+\mathcal{T}_e(\mathcal{Q}_{\mathrm{opt}})}{\mathcal{T}_e(\mathcal{Q})} \leq 1$

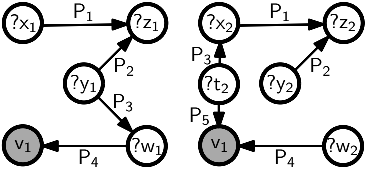
Our approach

Motivating example



(a) Query Q_1 (b) Query Q_2

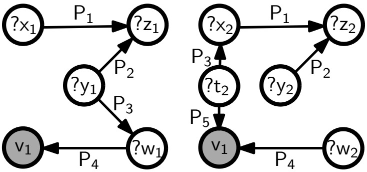
Motivating example



(a) Query Q_1 (b) Query Q_2

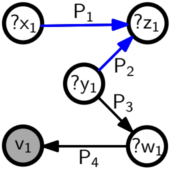
: constant : variable

Motivating example

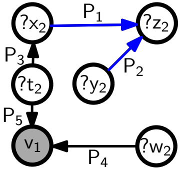


(a) Query Q_1 (b) Query Q_2

Motivating example



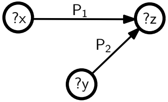
(a) Query Q_1



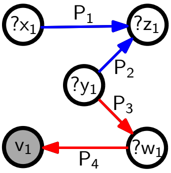
(b) Query Q_2

SELECT *
WHERE { $?x$ P_1 $?z$, $?y$ P_2 $?z$,

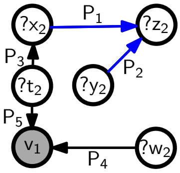
}



Motivating example



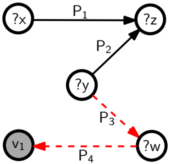
(a) Query Q_1



(b) Query Q_2

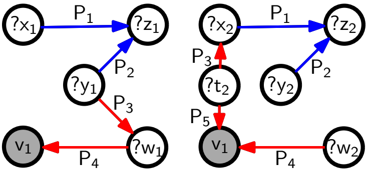
SELECT *
WHERE { $?x$ P_1 $?z$, $?y$ P_2 $?z$,
OPTIONAL {$?y$ P_3 $?w$, $?w$ P_4 v_1 }

}



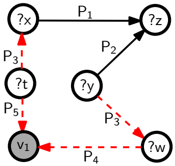
(I) Structure only Q_{OPT}

Motivating example



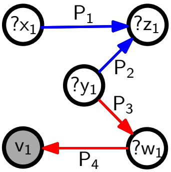
(a) Query Q_1 (b) Query Q_2

SELECT *
WHERE { $?x$ P_1 $?z$, $?y$ P_2 $?z$,
  OPTIONAL {$?y$ P_3 $?w$, $?w$ P_4 v_1 }
  OPTIONAL {$?t$ P_3 $?x$, $?t$ P_5 v_1, $?w$ P_4 v_1 }
}

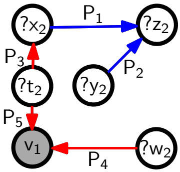


(I) Structure only Q_{OPT}

Motivating example



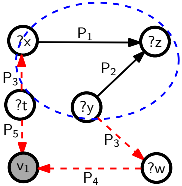
(a) Query Q_1



(b) Query Q_2

SELECT *
WHERE { $?x$ P_1 $?z$, $?y$ P_2 $?z$,
OPTIONAL {$?y$ P_3 $?w$, $?w$ P_4 v_1 }
OPTIONAL {$?t$ P_3 $?x$, $?t$ P_5 v_1, $?w$ P_4 v_1 }
}

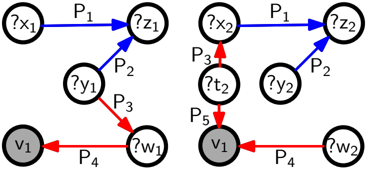
Evaluated once→potential saving



(I) Structure only Q_{OPT}

OPTIONALs are evaluated on top of the common substructures (intermediate results cached by engine).

Motivating example



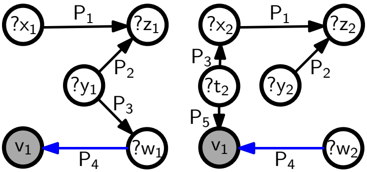
(a) Query Q_1 (b) Query Q_2

pattern p	α(p)
$?x$ P_1 $?z$	30%
$?y$ P_2 $?z$	20%
$?y$ P_3 $?w$	18%
$?w$ P_4 v_1	1%
$?t$ P_5 v_1	2%

*Max common subquery is not selective

(II) Using cost in optimization

Motivating example



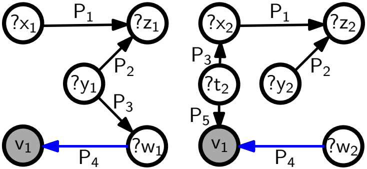
(a) Query Q_1 (b) Query Q_2

pattern p	α(p)
$?x$ P_1 $?z$	30%
$?y$ P_2 $?z$	20%
$?y$ P_3 $?w$	18%
$?w$ P_4 v_1	1%
$?t$ P_5 v_1	2%

*Max common subquery is not selective

(II) Using cost in optimization

Motivating example



(a) Query Q_1 (b) Query Q_2

SELECT *
WHERE { ?w P_4 v_1,
OPTIONAL {?x_1 P_1 ?z_1, ?y_1 P_2 ?z_1, ?y_1 P_3 ?w }
OPTIONAL {?x_2 P_1 ?z_2, ?y_2 P_2 ?z_2, ?t_2 P_3 ?x_2, ?t_2 P_5 v_1 }
}

(II) Using cost in optimization

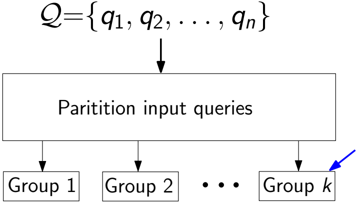
Our approach

$$\mathcal{Q}=\{q_1, q_2, \ldots, q_n\}$$

Our approach

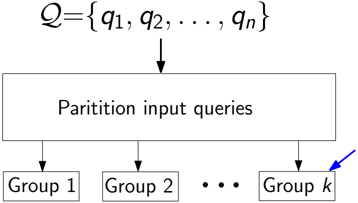
$\mathcal{Q}=\{q_1, q_2, \ldots, q_n\}$ ⟵ They often do not share one common subquery

Our approach



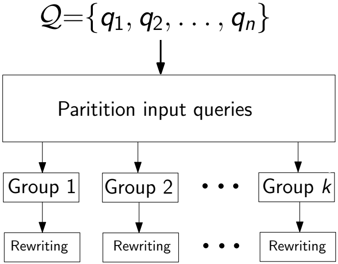
- Similar queries can be optimized together

Our approach

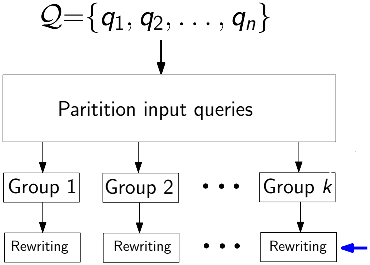


- Similar queries can be optimized together
  - finding structure similarity is expensive
  - group by predicates
  - distance: Jaccard similarity of predicate sets

Our approach

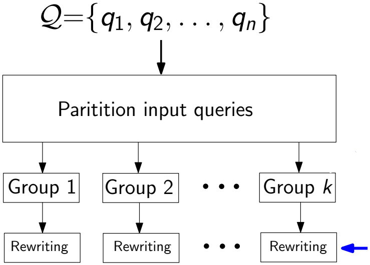


Our approach



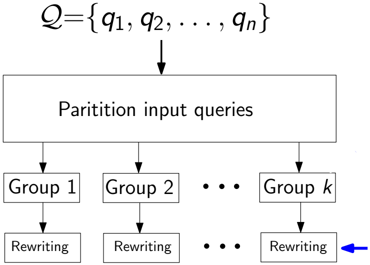
- Recursively rewrite a subset of type 1 queries (hierarchically)$\rightarrow$ a set of type 2 queries

Our approach

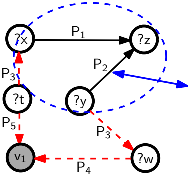


- Recursively rewrite a subset of type 1 queries (hierarchically)$\rightarrow$ a set of type 2 queries
  - finding common edge subgraphs
  - optimizations to avoid bad efficiency
  - cost: guard against bad rewritings
  - approx. by the min selectivity in common subquery

Our approach

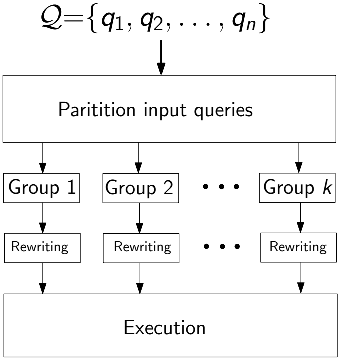


- Recursively rewrite a subset of type 1 queries (hierarchically)$\rightarrow$ a set of type 2 queries
  - finding common edge subgraphs
  - optimizations to avoid bad efficiency
  - cost: guard against bad rewritings
  - approx. by the min selectivity in common subquery

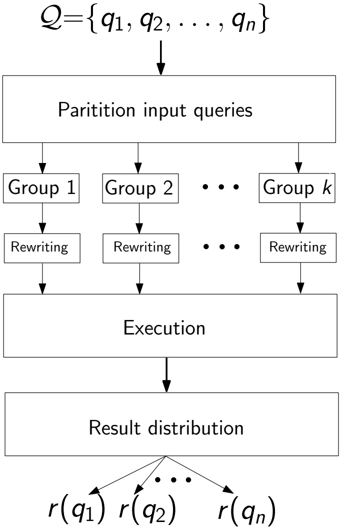


pattern p	$\alpha(p)$
?x P_1 ?z	30%
?y P_2 ?z	20%
?y P_3 ?w	18%
?w P_4 v_1	1%
?t P_5 v_1	2%

Our approach



Our approach



Our approach

- Related issues

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query→**Type 1** queries

X	Y	Z
a		
b		c
d	e	
f		g

RD of a **Type 1** query: e.g., X and Z

↑↓

columns from results of the **Type 2** rewriting

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query→**Type 1** queries

X	Y	Z
a		
b		c
d	e	
f		g

RD of a **Type 1** query: e.g., X and Z

↑↓

columns from results of the **Type 2** rewriting

  - Soundness and completeness

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query→**Type 1** queries

X	Y	Z
a		
b		c
d	e	
f		g

RD of a **Type 1** query: e.g., X and Z

↑↓

columns from results of the **Type 2** rewriting

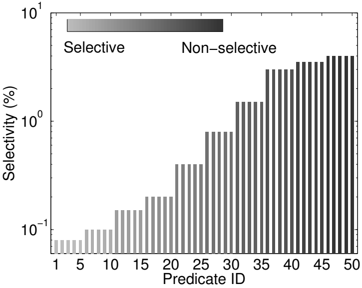
  - Soundness and completeness
  - Extensibility of the solution: more general queries
    - handle variable predicates
    - OPTIONAL queries

Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory

Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory
- Dataset
  - Extend LUBM benchmark generator:
    randomness in structure, variances of sel.



Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory
- Dataset
  - Extend LUBM benchmark generator:
    randomness in structure, variances of sel.
- RDF stores: Jena TDB 0.85 etc

Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory
- Dataset
  - Extend LUBM benchmark generator:
    randomness in structure, variances of sel.
- RDF stores: Jena TDB 0.85 etc
- Queries
  Ensure randomness in structure, e.g., star, chain and circle

Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory
- Dataset
  - Extend LUBM benchmark generator:
    randomness in structure, variances of sel.
- RDF stores: Jena TDB 0.85 etc
- Queries

Parameter	Symbol	Default	Range
Dataset size	D	4M	3M to 9M
Number of queries	$\lvert\mathcal{Q}\rvert$	100	60 to 160
Size of query (num of triple patterns)	$\lvert Q\rvert$	6	5 to 9
Number of seed queries	κ	6	5 to 10
Size of seed queries	$\lvert q_{\text{cmn}}\rvert$	$\sim \lvert Q\rvert/2$	1 to 5
Max selectivity of patterns in Q	$\alpha_{max}(Q)$	random	0.1% to 4%
Min selectivity of patterns in Q	$\alpha_{min}(Q)$	1%	0.1% to 4%

Experiments

- Implementation highlights
  - C++
  - 64-bit Linux, 2GHz Xeon(R) CPU, 4GB memory
- Dataset
  - Extend LUBM benchmark generator: randomness in structure, variances of sel.
- RDF stores: Jena TDB 0.85 etc
- Queries

Parameter	Symbol	Default	Range
Dataset size	D	4M	3M to 9M
Number of queries	$\|\mathcal{Q}\|$	100	60 to 160
Size of query (num of triple patterns)	$\|Q\|$	6	5 to 9
Number of seed queries	κ	6	5 to 10
Size of seed queries	$\|q_{\text{cmn}}\|$	$\sim \|Q\|/2$	1 to 5
Max selectivity of patterns in Q	$\alpha_{max}(Q)$	random	0.1% to 4%
Min selectivity of patterns in Q	$\alpha_{min}(Q)$	1%	0.1% to 4%

- Rewriting w/ structure: MQO-S; rewriting w/ structure and cost: MQO

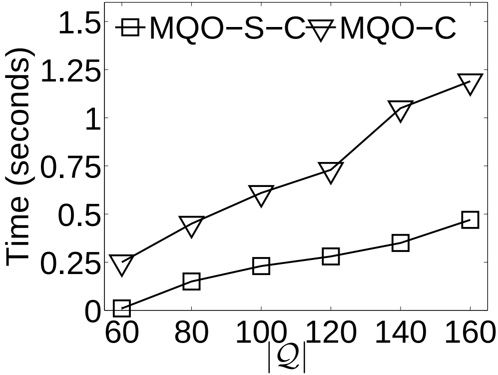
Experiments

- Time on rewriting
  MQO-S-C: structure based rewriting

  MQO-C: rewriting integrating with cost

Experiments

- Time on rewriting
  MQO-S-C: structure based rewriting
  MQO-C: rewriting integrating with cost



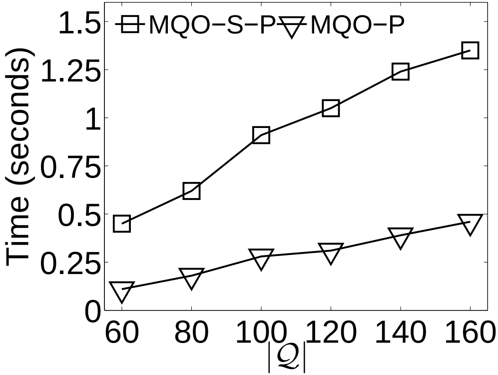
*Costly/bad rewritings are rejected
→more rounds of comparisons.

Experiments

- Time on distributing results
  MQO-S-P: parsing results from MQO-S
  MQO-P: parsing results with MQO

Experiments

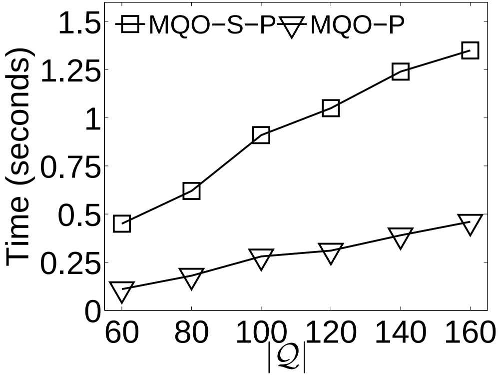
- Time on distributing results
  MQO-S-P: parsing results from MQO-S
  MQO-P: parsing results with MQO



*Non-selective common subqueries increase the set of results.

Experiments

- Time on distributing results
  MQO-S-P: parsing results from MQO-S
  MQO-P: parsing results with MQO



*Non-selective common subqueries increase the set of results.

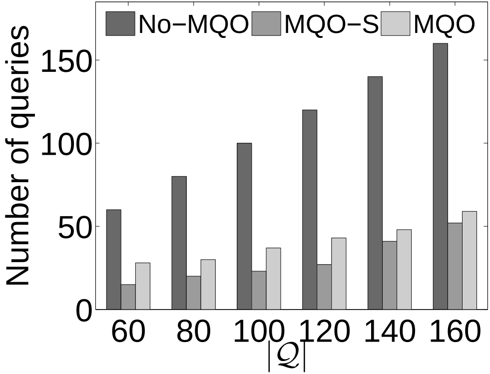
*Both rewriting and parsing are efficiently doable

Experiments

- Varying num of queries in a batch
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost

Experiments

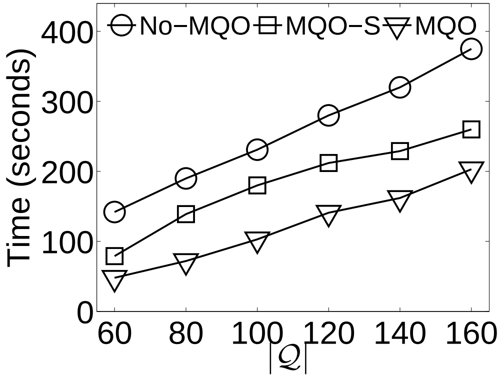
- Varying num of queries in a batch
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost



*Both reduce the num of queries to be exectued

Experiments

- Varying num of queries in a batch
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost

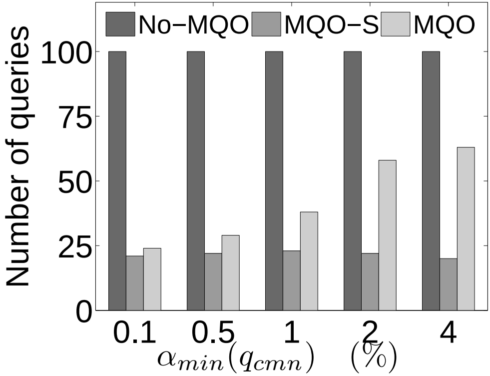


Experiments

- Varying min. selectivity in seed queries
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost

Experiments

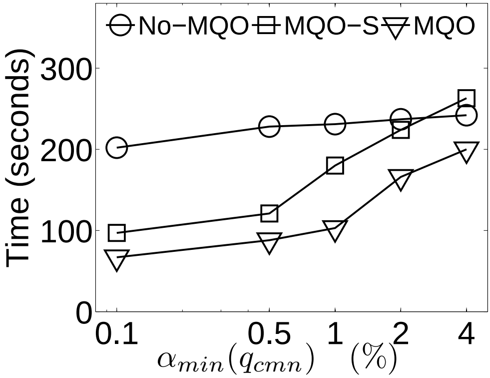
- Varying min. selectivity in seed queries
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost



*MQO: reject more bad rewritings;
MQO-S: not sensitive

Experiments

- Varying min. selectivity in seed queries
  No-MQO: no optimization
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost



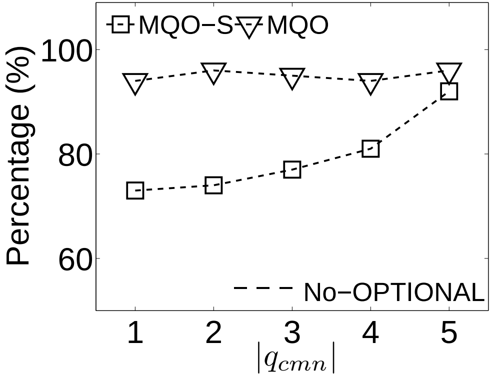
Experiments

- Varying seed size
  MQO-S: optimization based on structural rewriting

  MQO: integrating cost

  percentage$=\frac{\mathcal{T}_e(\text{common subquery})}{\mathcal{T}_e(\mathrm{Q}_{opt})} \times 100\%$

- Varying seed size
  MQO-S: optimization based on structural rewriting
  MQO: integrating cost
  percentage$=\frac{\mathcal{T}_e(\text{common subquery})}{\mathcal{T}_e(\text{Q}_{opt})} \times 100\%$



*MQO-S: up to 25% time on optional

Conclusions

- In dealing RDF data on the Web, store independency is important
- Combining SPARQL language and graph algorithms can achieve MQO, *i.e.*, by rewriting queries
- Cost must be taken in consideration during rewriting

Thank You

Q and A

Our approach

- Partition input queries

Our approach

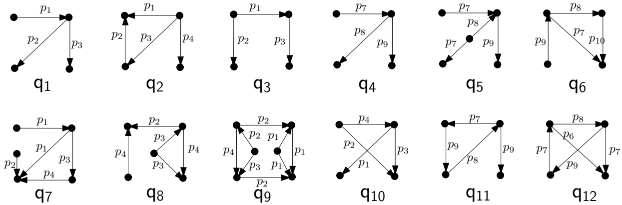
- Partition input queries
  - Object: similar queries can be optimized together in rewriting

Our approach

- Partition input queries
    - Object: similar queries can be optimized together in rewriting

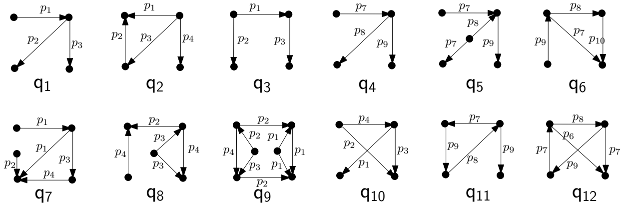
Our approach

- Partition input queries
  - Object: similar queries can be optimized together in rewriting



Our approach

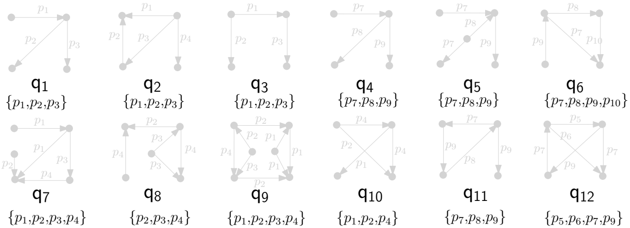
- Partition input queries
  - Object: similar queries can be optimized together in rewriting



Distance: Jaccard similarity on predicates

Our approach

- Partition input queries
  - Object: similar queries can be optimized together in rewriting

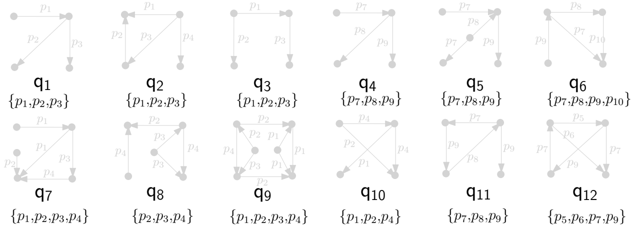


Distance: Jaccard similarity on predicates

- Represent each query as a set of predicates.
- Measure the similarity of a pair of queries by set similarity

Our approach

- Partition input queries
  - Object: similar queries can be optimized together in rewriting

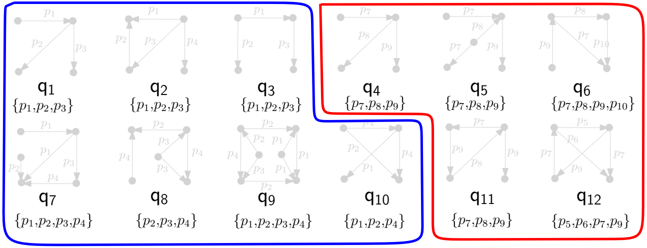


Distance: Jaccard similarity on predicates

- Represent each query as a set of predicates.
- Measure the similarity of a pair of queries by set similarity
- Grouping: k-means

Our approach

- Partition input queries
  - Object: similar queries can be optimized together in rewriting



Distance: Jaccard similarity on predicates

- Represent each query as a set of predicates.
- Measure the similarity of a pair of queries by set similarity
- Grouping: k-means

Our approach

- Hierarchical rewriting and clustering (inside a group)

Our approach

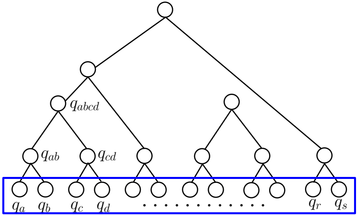
- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

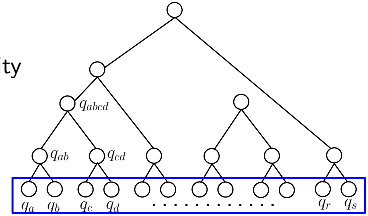


Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity

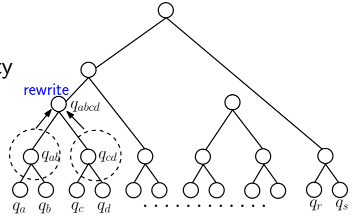


Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity

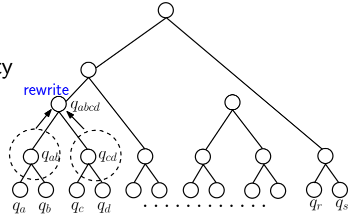


Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity



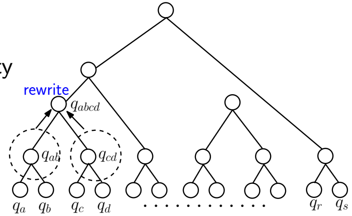
- Rewriting $\rightarrow$ finding maximal common triple patterns
- In the language of graph . . .

Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity



- Rewriting $\rightarrow$ finding maximal common triple patterns
- In the language of graph . . .

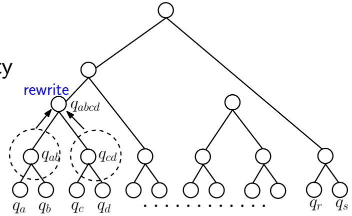
[CLQ01] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. In *Theoretical Computer Science*, 2001.

Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity



- Rewriting $\rightarrow$ finding maximal common triple patterns
- In the language of graph . . .

  [CLQ01] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. In *Theoretical Computer Science*, 2001.

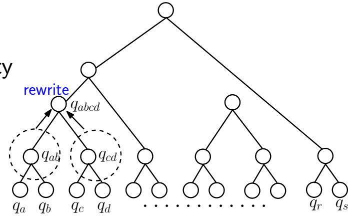
  - maximal common connected edge subgraphs

Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity



- Rewriting $\rightarrow$ finding maximal common triple patterns
- In the language of graph ...

  [CLQ01] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. In *Theoretical Computer Science*, 2001.

  - maximal common connected edge subgraphs

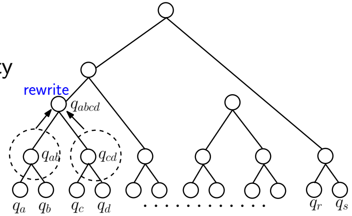
    $\rightarrow$ maximal common connected *induced* sugraphs in *linegraphs*

Our approach

- Hierarchical rewriting and clustering (inside a group)

  Rewrite pairs of queries bottom up

  Pair up queries with max Jaccard similarity



- Rewriting $\rightarrow$ finding maximal common triple patterns
- In the language of graph ...

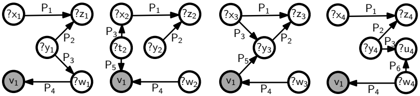
  [CLQ01] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. In *Theoretical Computer Science*, 2001.

  - maximal common connected edge subgraphs

    $\rightarrow$ maximal common connected *induced* sugraphs in *linegraphs*

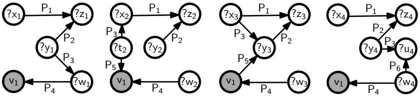
    $\rightarrow$ maximal cliques in the product graph

Our approach



(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

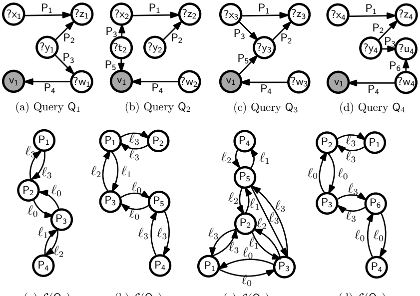
Our approach



(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

- Linegraph: invert vertices and edges

Our approach

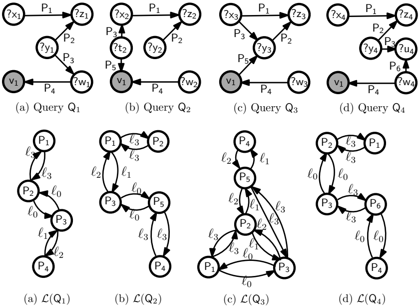


(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3

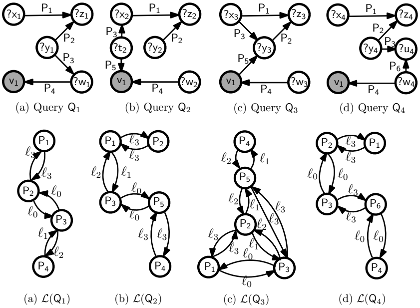
Our approach



- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3

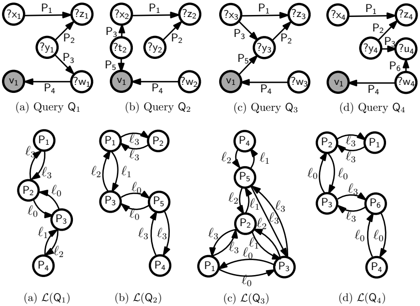
- product graph: simultaneous walk

Our approach



- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3
- product graph: simultaneous walk

Our approach

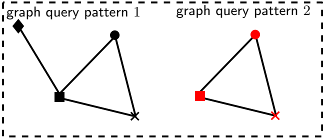


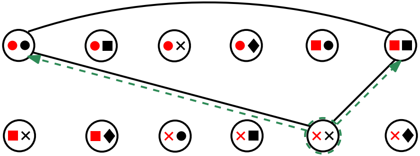
(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3

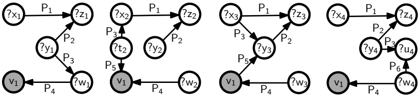
- product graph: simultaneous walk





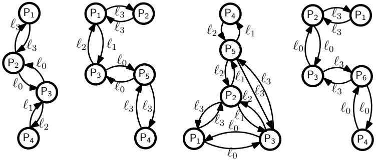
The triangle (clique) highlights the common subgraph composed by ■ × ●

Our approach



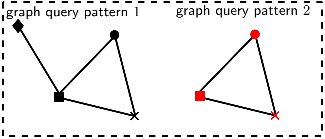
(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

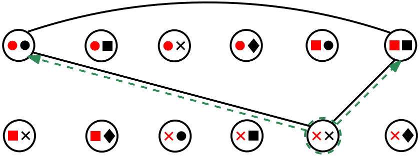
- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3



(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

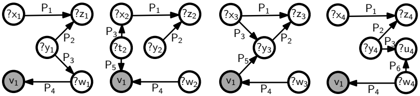
- product graph: simultaneous walk
- blowup in size, esp. > 2 queries affect clique detection



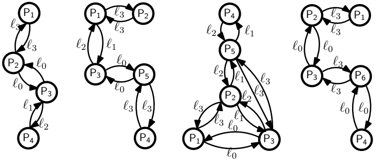


The triangle (clique) highlights the common subgraph composed by ■ × ●

Our approach



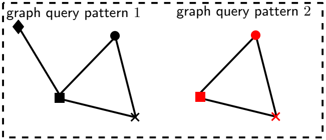
(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

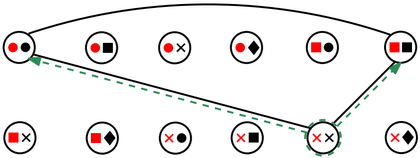


(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

- Linegraph: invert vertices and edges
- sub−sub:ℓ_0, sub−obj:ℓ_1, obj−sub:ℓ_2, obj−obj:ℓ_3

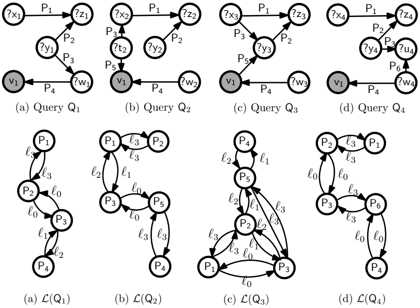
- product graph: simultaneous walk
- blowup in size, esp. > 2 queries affect clique detection
- optimize the product graph





The triangle (clique) highlights the common subgraph composed by ■ × ●

Our approach

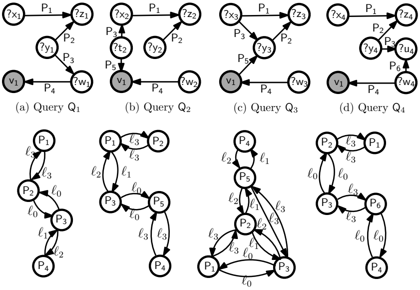


- Linegraph: invert vertices and edges
- sub—sub:ℓ_0, sub—obj:ℓ_1, obj—sub:ℓ_2, obj—obj:ℓ_3

- product graph: simultaneous walk
- blowup in size, esp. > 2 queries affect clique detection
- optimize the product graph

- prune non-common predicates
- check the constants

Our approach



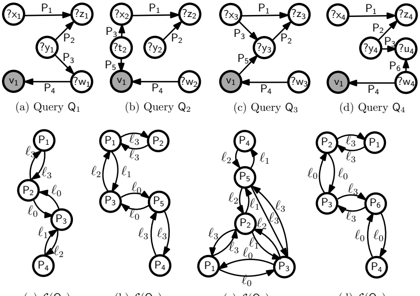
(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

- Linegraph: invert vertices and edges
- sub–sub:ℓ_0, sub–obj:ℓ_1, obj–sub:ℓ_2, obj–obj:ℓ_3

- product graph: simultaneous walk
- blowup in size, esp. > 2 queries affect clique detection
- optimize the product graph

- prune non-common predicates
- check the constants
- prune vertices with non-common neighborhoods



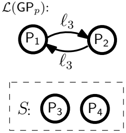
(a) Query Q_1 (b) Query Q_2 (c) Query Q_3 (d) Query Q_4

(a) $\mathcal{L}(Q_1)$ (b) $\mathcal{L}(Q_2)$ (c) $\mathcal{L}(Q_3)$ (d) $\mathcal{L}(Q_4)$

- Linegraph: invert vertices and edges
- sub—sub:ℓ_0, sub—obj:ℓ_1, obj—sub:ℓ_2, obj—obj:ℓ_3

- product graph: simultaneous walk
- blowup in size, esp. > 2 queries affect clique detection
- optimize the product graph

- prune non-common predicates
- check the constants
- prune vertices with non-common neighborhoods



Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

[CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

[CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

[CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

[CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting
  - Structure: maximize size of the common subquery in a rewriting
  - Evaluation on cost: guard against bad rewritings

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting
  - Structure: maximize size of the common subquery in a rewriting
  - Evaluation on cost: guard against bad rewritings
  - Measure: min selectivity in the common subquery for approximation

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting
  - Structure: maximize size of the common subquery in a rewriting
  - Evaluation on cost: guard against bad rewritings
  - Measure: min selectivity in the common subquery for approximation
  - Cost: discard bad rewritings, keep good ones in hierarchical rewriting

Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting
  - Structure: maximize size of the common subquery in a rewriting
  - Evaluation on cost: guard against bad rewritings
  - Measure: min selectivity in the common subquery for approximation
  - Cost: discard bad rewritings, keep good ones in hierarchical rewriting

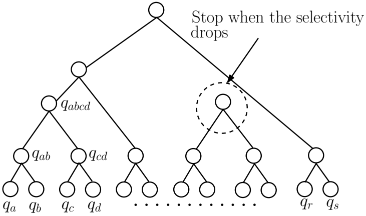
Our approach

- Find maximal cliques in the product graph [CLQ02][CLQ03]

  [CLQ02] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics*, 2002.

  [CLQ03] E. Tomita et al. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science, LNCS*, 2003.

- Integrate cost into rewriting
  - Structure: maximize size of the common subquery in a rewriting
  - Evaluation on cost: guard against bad rewritings
  - Measure: min selectivity in the common subquery for approximation
  - Cost: discard bad rewritings, keep good ones in hierarchical rewriting



Our approach

- Related issues

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query$\rightarrow$**Type 1** queries

name	mail	hpage
"Alice"	alice@home	http://home/alice
"Alice"	alice@work	http://home/alice
"Bob"		
"Ella"		http://work/ella

RD of a **Type 1** query

$\uparrow\downarrow$

columns from results of the **Type 2** rewriting

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query→**Type 1** queries

name	mail	hpage
"Alice"	alice@home	http://home/alice
"Alice"	alice@work	http://home/alice
"Bob"		
"Ella"		http://work/ella

RD of a **Type 1** query

↑↓

columns from results of the **Type 2** rewriting

  - Soundness and completeness

Our approach

- Related issues
  - Distributing results, *i.e.*, **Type 2** query→**Type 1** queries

name	mail	hpage
"Alice"	alice@home	http://home/alice
"Alice"	alice@work	http://home/alice
"Bob"		
"Ella"		http://work/ella

RD of a **Type 1** query

↑↓

columns from results of the **Type 2** rewriting

  - Soundness and completeness
  - Extensibility of the solution: more general queries
    - handle variable predicates
    - nested OPTIONALs