RDF Query Relaxation Strategies Based On Failure Causes

Géraud FOKOU   Stéphane JEAN   Allel HADJALI
Mickaël BARON

LIAS/ENSMA and University of Poitiers, France

eswc16

13th ESWC 2016
May 29th - June 2nd | Heraklion, Crete, Greece

Motivation

- Observation
  - Between May and July 2010 10.2% queries on DBpedia are failing i.e return zero result [Muhammad15]

Motivation

- Observation
  - Between May and July 2010 10.2% queries on DBpedia are failing i.e return zero result [Muhammad15]
- Reasons: difficulties to query knowledge Base (KB)
  - Incomplete and Heterogeneous Data
  - Large size and complex schema semantics for end-users
  - Correct formulation of queries returning the intended result
  - ...

Motivation

- Observation
  - Between May and July 2010 10.2% queries on DBpedia are failing i.e return zero result [Muhammad15]
- Reasons: difficulties to query knowledge Base (KB)
  - Incomplete and Heterogeneous Data
  - Large size and complex schema semantics for end-users
  - Correct formulation of queries returning the intended result
  - ...

Need for RDF query relaxation to provide alternatives answers

Related Work

- Relaxation operators [Hurtado08, Fokou14 ...]
  - replacing class (property) by superclass (superproperty)
  - replacing constants by variables
  - removing triple patterns
  - ...

Related Work

- Relaxation operators [Hurtado08, Fokou14 ...]
  - replacing class (property) by superclass (superproperty)
  - replacing constants by variables
  - removing triple patterns
  - ...
- Relaxation process [Huang12]
  - generate many relaxed queries (using relaxation operator)
    - execute them in a ranking order
    - stop when alternative answers are obtained

Related Work

- Relaxation operators [Hurtado08, Fokou14 ...]
  - replacing class (property) by superclass (superproperty)
  - replacing constants by variables
  - removing triple patterns
  - ...
- Relaxation process [Huang12]
  - generate many relaxed queries (using relaxation operator)
    - execute them in a ranking order
    - stop when alternative answers are obtained

⇒ Problems for failing queries

- Can execute an exponential number of queries
- Query parts responsible of failure are unknown

Related Work

- Relaxation operators [Hurtado08, Fokou14 ...]
  - replacing class (property) by superclass (superproperty)
  - replacing constants by variables
  - removing triple patterns
  - ...

- Relaxation process [Huang12]
  - generate many relaxed queries (using relaxation operator)
    - execute them in a ranking order
    - stop when alternative answers are obtained

⇒ Our idea

- Find all the query parts responsible of its failure
- Leverage them for speeding up the relaxation process

Problem Statement

- Previous contribution: finding query failure causes [Fokou15]
  - Failure causes: **Minimal Failure Subquery (MFS)**
    - smallest subquery which return empty answers set
  - Two algorithms to find them: LBA and MBA algorithms
  - Expensive cost time: NP-Hard Problem

Problem Statement

- Previous contribution: finding query failure causes [Fokou15]
  - Failure causes: **Minimal Failure Subquery (MFS)**
    - smallest subquery which return empty answers set
  - Two algorithms to find them: LBA and MBA algorithms
  - Expensive cost time: NP-Hard Problem

Research questions: prune relaxed queries search space

- MFSs useful for query relaxation process?
- MFSs of the initial query and/or the relaxed queries
  - Leverage MFSs of relaxed queries?
  - MFSs of relaxed queries related to ones of the initial query?

Problem Statement

- Previous contribution: finding query failure causes [Fokou15]
  - Failure causes: **Minimal Failure Subquery (MFS)**
    - smallest subquery which return empty answers set
  - Two algorithms to find them: LBA and MBA algorithms
  - Expensive cost time: NP-Hard Problem

Research questions: prune relaxed queries search space

- MFSs useful for query relaxation process?
- MFSs of the initial query and/or the relaxed queries
  - Leverage MFSs of relaxed queries?
  - MFSs of relaxed queries related to ones of the initial query?

Goal

$\Rightarrow$ Finding the good Tradeoff:
MFS computation cost vs relaxed queries pruned thanks to MFS

Outline

- Motivation and Problem Statement
- Preliminaries
- Query Relaxation Strategies
- Experimental results
- Conclusion and Perspectives

Preliminaries: Triple pattern relaxation

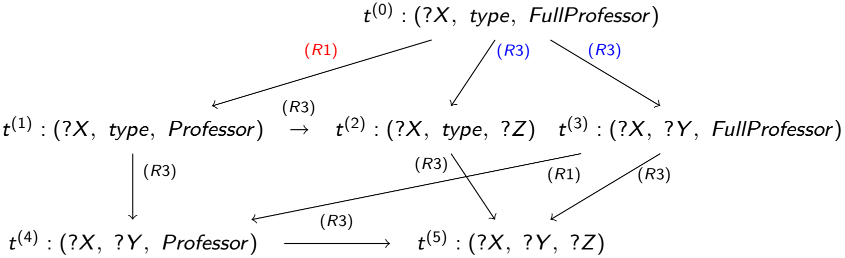
Triple pattern $t = (?X,\ type,\ FullProfessor)$

- Generate triple pattern with three rules:
  - R_1 : *class* $\Rightarrow$ *superclass* $R_1(t) = (?X,\ type,\ Professor) = t^{(1)}$
  - R_2 : *property* $\Rightarrow$ *superproperty*
  - R_3 : *constant* $\Rightarrow$ *variable* $R_3(t) = (?X,\ ?Z,\ FullProfessor) = t^{(2)}$
- Order by similarity: $sim(t^{(0)}, t^{(k)})$

Preliminaries: Triple pattern relaxation

Triple pattern $t = (?X,\ type,\ FullProfessor)$

- Generate triple pattern with three rules:
  - R_1 : *class* $\Rightarrow$ *superclass* $R_1(t) = (?X,\ type,\ Professor) = t^{(1)}$
  - R_2 : *property* $\Rightarrow$ *superproperty*
  - R_3 : *constant* $\Rightarrow$ *variable* $R_3(t) = (?X,\ ?Z,\ FullProfessor) = t^{(2)}$
- Order by similarity: $sim(t^{(0)}, t^{(k)})$



Preliminaries: Query pattern relaxation

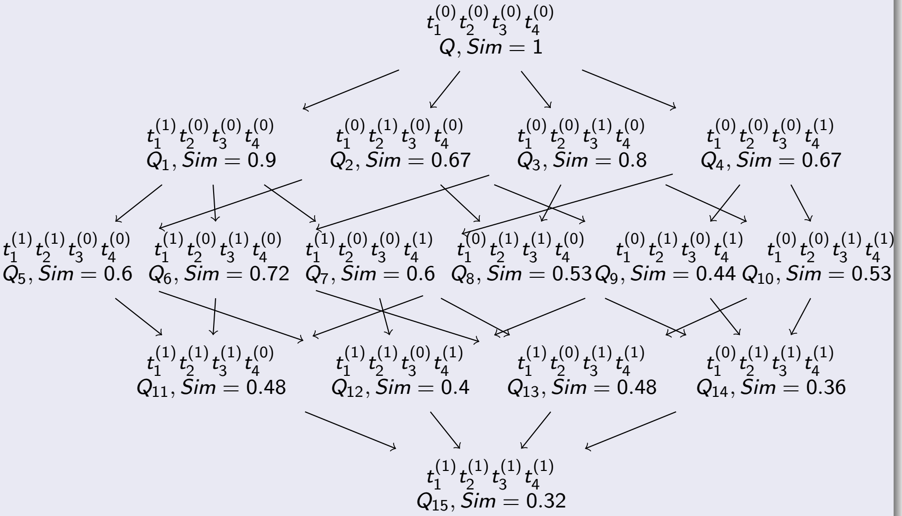
Query pattern relaxation

- Consider a query $Q = t_1^{(0)} \wedge \ldots \wedge t_n^{(0)}$, conjunction of triple pattern
- A relaxed query Q' of Q will be $Q' = t_1^{(k_1)} \ldots \wedge t_n^{(k_n)}$ such that $\exists k_i > 0$
- $Sim(Q, Q') = \prod_{i=1}^{n} Sim(t_i^{(0)}, t_i^{(k_i)})$
- An iterative relaxation of a query give its relaxation graph
- Example $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$

```
SELECT ?p ?n WHERE {
?p type Lecturer          (t1)
?p nationality ?n         (t2)
?p teacherOf SW           (t3)
?p age 46 }               (t4)
```

Preliminaries: Query pattern relaxation

Query relaxation graph



Outline

- Motivation and Problem Statement
- Preliminaries
- Query Relaxation Strategies
- Experimental results
- Conclusion and Perspectives

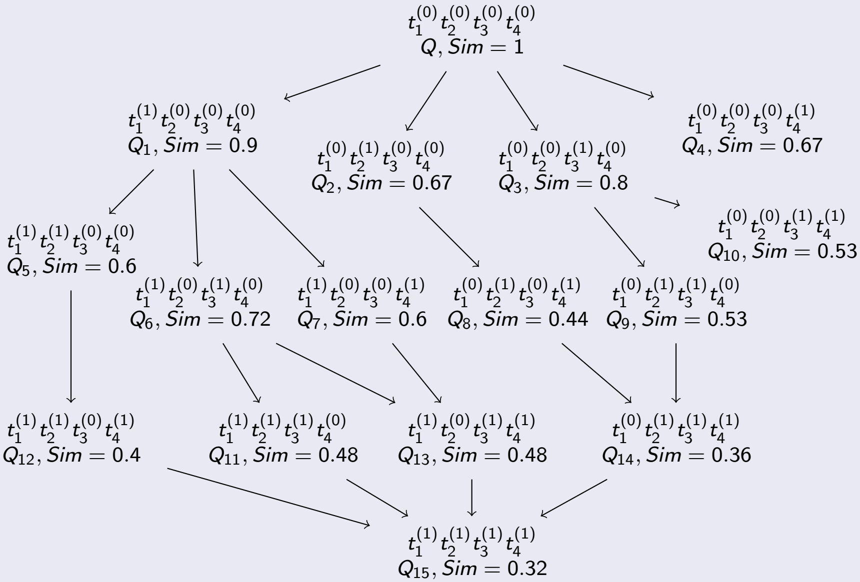
Relaxation Strategies (1): Best First Search (BFS)

⇒ BFS idea, existing work [Huang12]

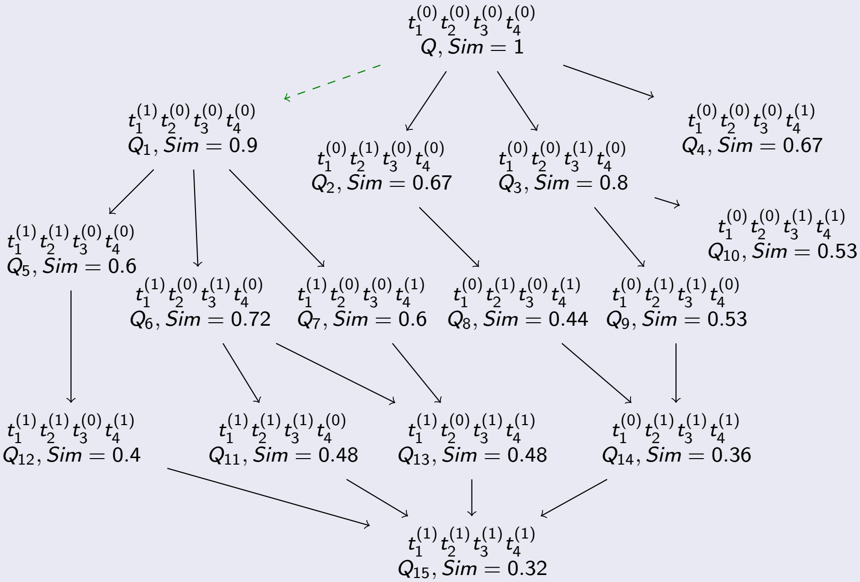
- Build the lattice of all relaxed queries
- Execute relaxed queries from the more to the least similar
- Stop when alternatives answers are found

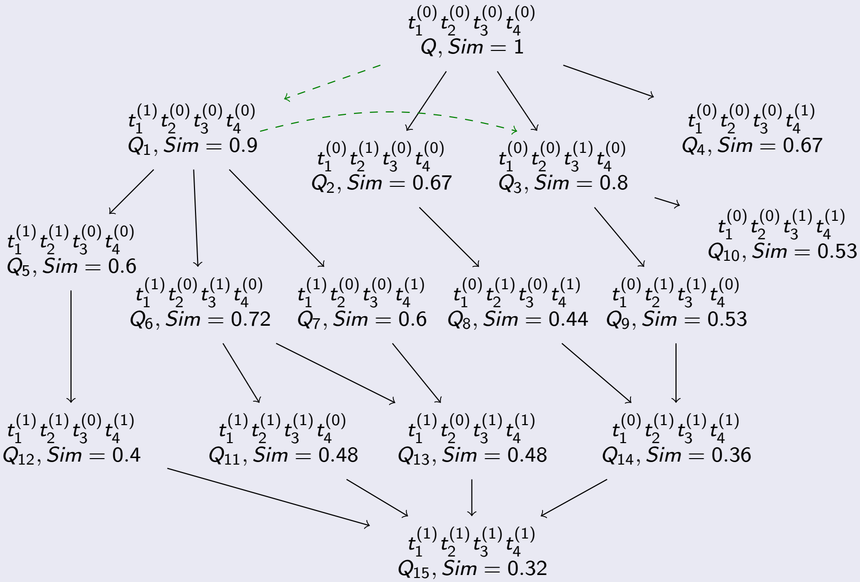
- Example $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$
- We assume that triple pattern could be relaxed at most once

Relaxation Strategies (1): Best First Search (BFS)

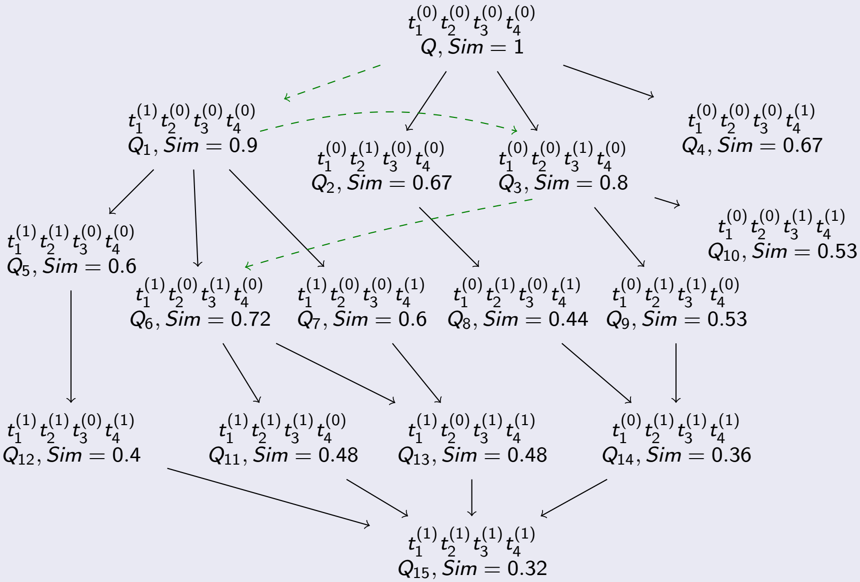


Relaxation Strategies (1): Best First Search (BFS)

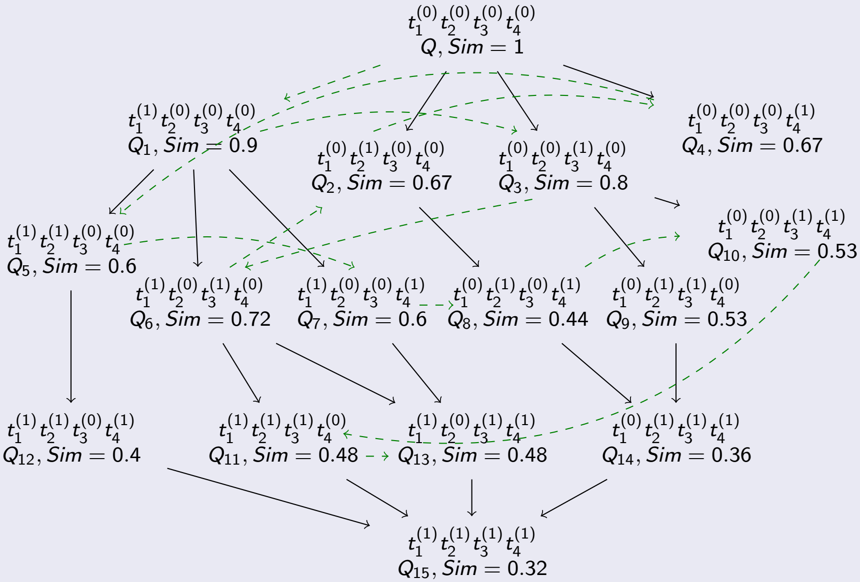



$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q, Sim = 1$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q_1, Sim = 0.9$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_2, Sim = 0.67$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_3, Sim = 0.8$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_4, Sim = 0.67$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{10}, Sim = 0.53$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_5, Sim = 0.6$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_6, Sim = 0.72$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_7, Sim = 0.6$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_8, Sim = 0.44$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_9, Sim = 0.53$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_{12}, Sim = 0.4$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_{11}, Sim = 0.48$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{13}, Sim = 0.48$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{14}, Sim = 0.36$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{15}, Sim = 0.32$

Relaxation Strategies (1): Best First Search (BFS)



Relaxation Strategies (1): Best First Search (BFS)



Relaxation Strategies (1): Best First Search (BFS)

⇒ BFS idea, existing work [Huang12]

- Build the lattice of all relaxed queries
- Execute relaxed queries from the more to the least similar
- Stop when alternatives answers are found

- Example $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$
- We assume that triple pattern could be relaxed at most once
- Execution order $Q_1, Q_3, Q_6, Q_2, Q_4, Q_5, Q_7, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$
- 16 queries executed in the worst case

Relaxation Strategies (1): Best First Search (BFS)

⇒ BFS idea, existing work [Huang12]

- Build the lattice of all relaxed queries
- Execute relaxed queries from the more to the least similar
- Stop when alternatives answers are found

- Example $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$
- We assume that triple pattern could be relaxed at most once
- Execution order
  $Q_1, Q_3, Q_6, Q_2, Q_4, Q_5, Q_7, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$
- 16 queries executed in the worst case

Weakness

⇒ Executed all relaxed queries even those they fail surely since they contain failure causes of the initial query

Relaxation Strategies (2): MFS-Based Search (MBS)

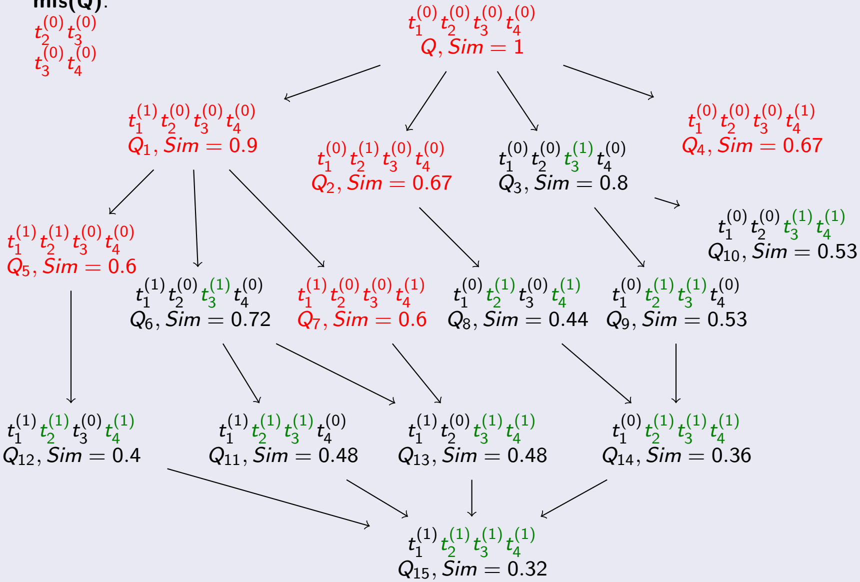
⇒ MFS-Based Search (MBS) idea

- Use MFS to prune some relaxed query
- Theoretical result: If relaxed queries still contain an MFS of the initial query, it fails ⇒ Pruned relaxed queries that contains an MFS

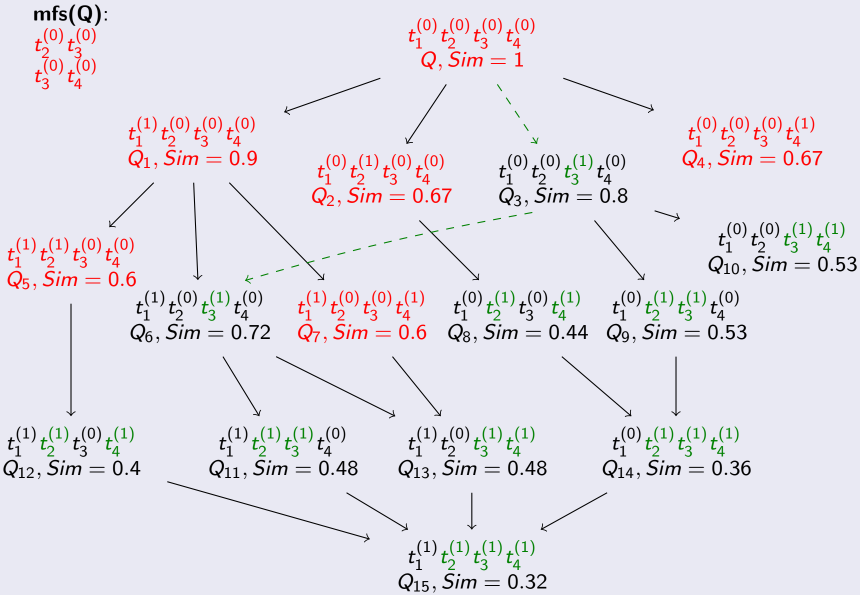
- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$

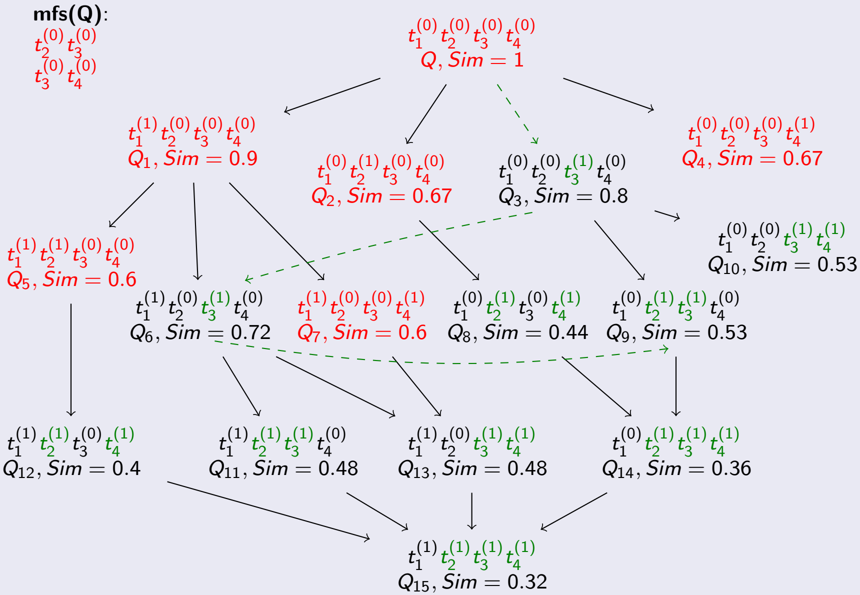
Relaxation Strategies (2): MFS-Based Search (MBS)

mfs(Q):
$t_2^{(0)} t_3^{(0)}$
$t_3^{(0)} t_4^{(0)}$



Relaxation Strategies (2): MFS-Based Search (MBS)

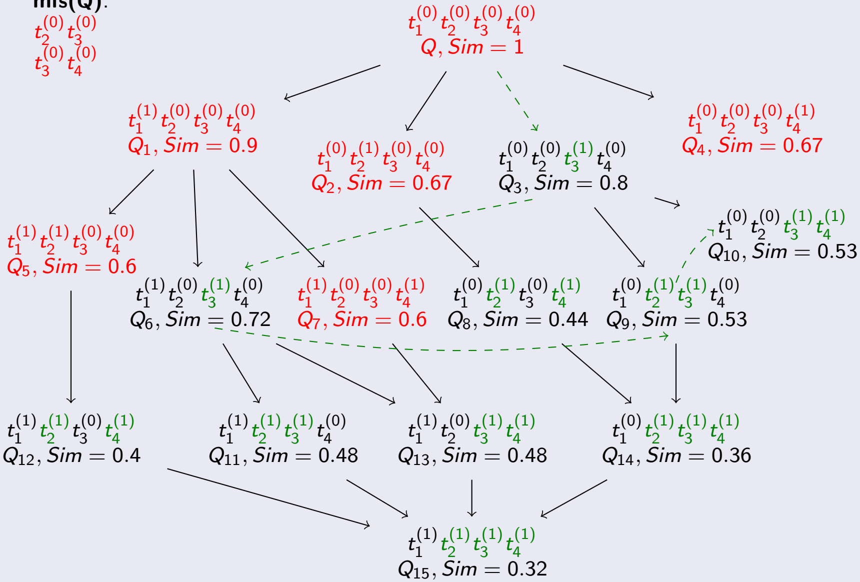



mfs(Q):
$t_2^{(0)} t_3^{(0)}$
$t_3^{(0)} t_4^{(0)}$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q, Sim = 1$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q_1, Sim = 0.9$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_2, Sim = 0.67$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_3, Sim = 0.8$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_4, Sim = 0.67$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_5, Sim = 0.6$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{10}, Sim = 0.53$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_6, Sim = 0.72$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_7, Sim = 0.6$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_8, Sim = 0.44$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_9, Sim = 0.53$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_{12}, Sim = 0.4$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_{11}, Sim = 0.48$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{13}, Sim = 0.48$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{14}, Sim = 0.36$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{15}, Sim = 0.32$

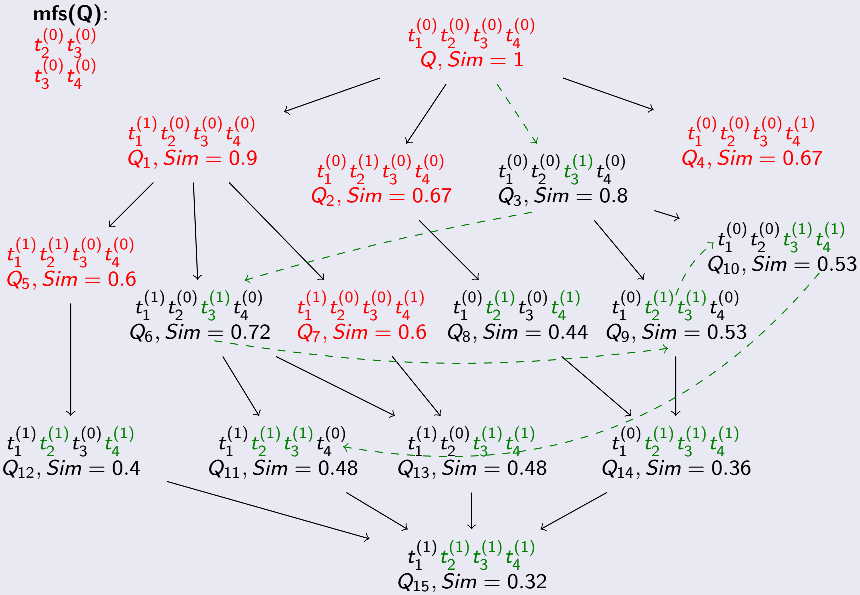
mfs(Q):

$t_2^{(0)} t_3^{(0)}$

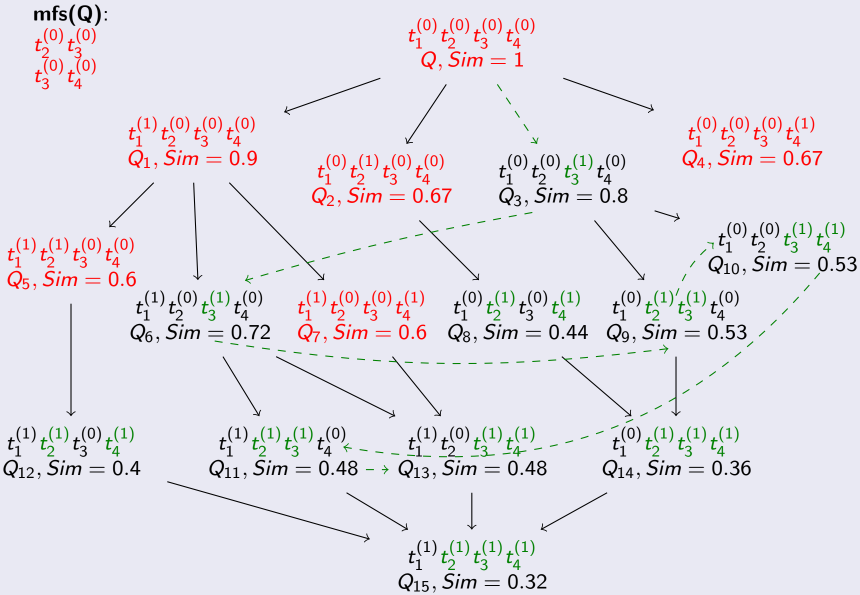
$t_3^{(0)} t_4^{(0)}$



Relaxation Strategies (2): MFS-Based Search (MBS)



Relaxation Strategies (2): MFS-Based Search (MBS)



Relaxation Strategies (2): MFS-Based Search (MBS)

⇒ MFS-Based Search (MBS) idea

- Use MFS to prune some relaxed query
- Theoretical result: If relaxed queries still contain an MFS of the initial query, it fails ⇒ Pruned relaxed queries that contains an MFS

- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Execution order $Q_3, Q_6, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$
- 10 relaxed queries executed in the worst case (31% pruned)

Relaxation Strategies (2): MFS-Based Search (MBS)

⇒ MFS-Based Search (MBS) idea

- Use MFS to prune some relaxed query
- Theoretical result: If relaxed queries still contain an MFS of the initial query, it fails ⇒ Pruned relaxed queries that contains an MFS

- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Execution order $Q_3, Q_6, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$
- 10 relaxed queries executed in the worst case (31% pruned)

Weakness

⇒ Some relaxed queries may still fail and we do not use their MFSs to prune further the search space

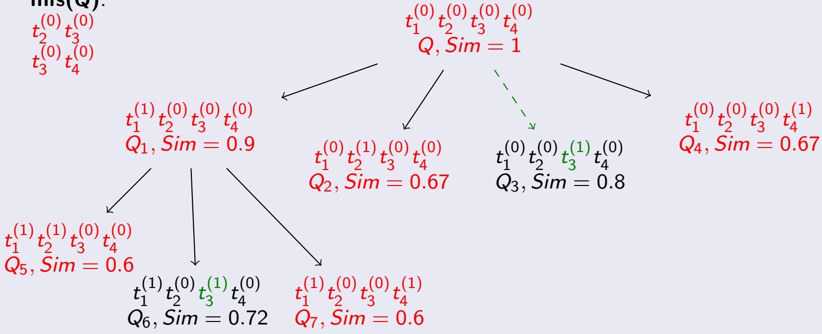
⇒ Optimized MFS Based Search (O-MFS) idea

- Deduce MFSs of a failing relaxed query from the ones of the initial queries
- Theoretical result: the MFSs of a relaxed query are not necessarily included in the ones of the initial queries. But, if an MFS of the initial query still fail in the relaxed query then it is one of its MFS

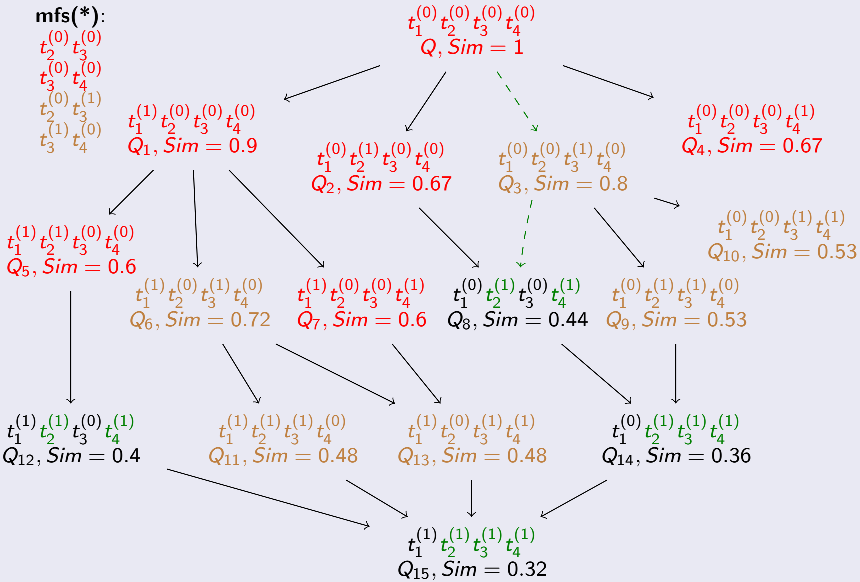
- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Failing relaxed query $Q' = t_1^{(0)} \wedge t_2^{(0)} \wedge t_3^{(1)} \wedge t_4^{(0)}$
- The two MFSs are still failing $t_2^{(0)} \wedge t_3^{(1)}$ and $t_3^{(1)} \wedge t_4^{(0)}$

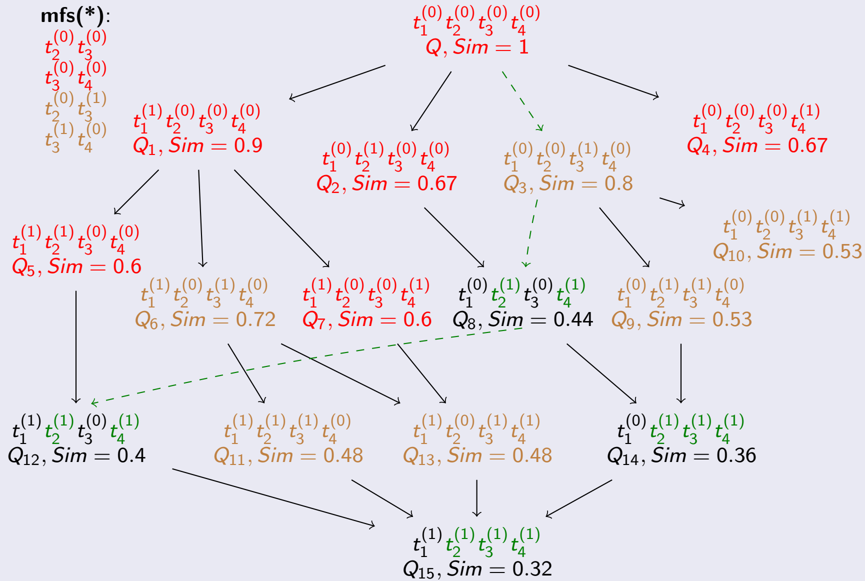
Relaxation Strategies (3): Optimized MFS Based Search

mfs(Q):
$t_2^{(0)} t_3^{(0)}$
$t_3^{(0)} t_4^{(0)}$



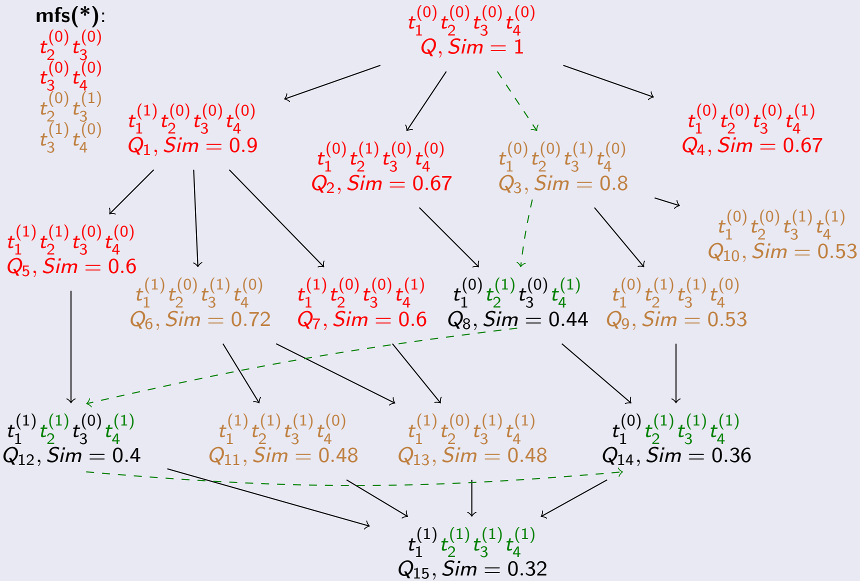
Relaxation Strategies (3): Optimized MFS Based Search

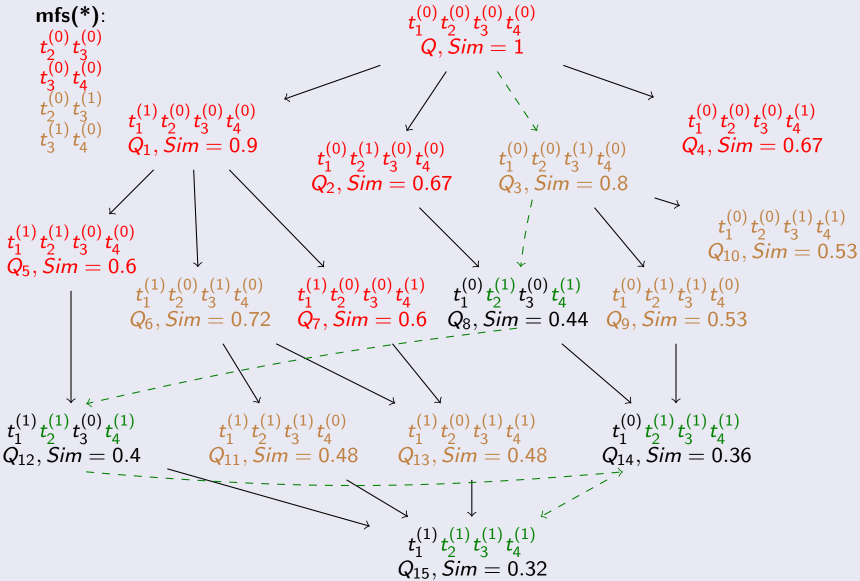




mfs(*):
$t_2^{(0)} t_3^{(0)}$
$t_3^{(0)} t_4^{(0)}$
$t_2^{(0)} t_3^{(1)}$
$t_3^{(1)} t_4^{(0)}$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q, Sim = 1$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q_1, Sim = 0.9$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_2, Sim = 0.67$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_3, Sim = 0.8$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_4, Sim = 0.67$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_5, Sim = 0.6$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{10}, Sim = 0.53$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_6, Sim = 0.72$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_7, Sim = 0.6$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_8, Sim = 0.44$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_9, Sim = 0.53$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_{12}, Sim = 0.4$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_{11}, Sim = 0.48$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{13}, Sim = 0.48$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{14}, Sim = 0.36$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{15}, Sim = 0.32$

Relaxation Strategies (3): Optimized MFS Based Search




mfs(*):
$t_2^{(0)} t_3^{(0)}$
$t_3^{(0)} t_4^{(0)}$
$t_2^{(0)} t_3^{(1)}$
$t_3^{(1)} t_4^{(0)}$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q, Sim = 1$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(0)}$
$Q_1, Sim = 0.9$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_2, Sim = 0.67$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_3, Sim = 0.8$
$t_1^{(0)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_4, Sim = 0.67$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(0)}$
$Q_5, Sim = 0.6$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(0)}$
$Q_6, Sim = 0.72$
$t_1^{(1)} t_2^{(0)} t_3^{(0)} t_4^{(1)}$
$Q_7, Sim = 0.6$
$t_1^{(0)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_8, Sim = 0.44$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_9, Sim = 0.53$
$t_1^{(0)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{10}, Sim = 0.53$
$t_1^{(1)} t_2^{(1)} t_3^{(0)} t_4^{(1)}$
$Q_{12}, Sim = 0.4$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(0)}$
$Q_{11}, Sim = 0.48$
$t_1^{(1)} t_2^{(0)} t_3^{(1)} t_4^{(1)}$
$Q_{13}, Sim = 0.48$
$t_1^{(0)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{14}, Sim = 0.36$
$t_1^{(1)} t_2^{(1)} t_3^{(1)} t_4^{(1)}$
$Q_{15}, Sim = 0.32$

Relaxation Strategies (3): Optimized MFS Based Search

⇒ Optimized MFS Based Search (O-MFS) idea

- Deduce MFSs of a failing relaxed query from the ones of the initial queries
- Theoretical result: the MFSs of a relaxed query are not necessarily included in the ones of the initial queries. But, if an MFS of the initial query still fail in the relaxed query then it is one of its MFS

- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Failing relaxed query $Q' = t_1^{(0)} \wedge t_2^{(0)} \wedge t_3^{(1)} \wedge t_4^{(0)}$
- The two MFSs are still failing $t_2^{(0)} \wedge t_3^{(1)}$ and $t_3^{(1)} \wedge t_4^{(0)}$
- Execution order $Q_3, Q_9, Q_{12}, Q_{14}, Q_{15}$
- Five relaxed queries executed in the worst case (68% pruned)

Relaxation Strategies (3): Optimized MFS Based Search

⇒ Optimized MFS Based Search (O-MFS) idea

- Deduce MFSs of a failing relaxed query from the ones of the initial queries
- Theoretical result: the MFSs of a relaxed query are not necessarily included in the ones of the initial queries. But, if an MFS of the initial query still fail in the relaxed query then it is one of its MFS

- Example: Q has two MFS, $Q^{*} = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Failing relaxed query $Q' = t_1^{(0)} \wedge t_2^{(0)} \wedge t_3^{(1)} \wedge t_4^{(0)}$
- The two MFSs are still failing $t_2^{(0)} \wedge t_3^{(1)}$ and $t_3^{(1)} \wedge t_4^{(0)}$
- Execution order $Q_3, Q_9, Q_{12}, Q_{14}, Q_{15}$
- Five relaxed queries executed in the worst case (68% pruned)

Weakness

⇒ All the MFSs of the relaxed queries are not necessarily found

Relaxation Strategies (4): Full MFS Based Search

⇒ Full MFS Based Search (F-MFS) idea

- Find new MFS in relaxed queries which failed
- New MFS contains at least one repaired MFS
- Using an adaptation of LBA algorithm for finding new MFS

- Example: Q has two MFS, $Q^* = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Failing relaxed query $Q' = t_1^{(0)} \wedge t_2^{(1)} \wedge t_3^{(0)} \wedge t_4^{(1)}$
- No MFSs still fail in Q'
- Failing subquery part with a repaired MFS $t_1^{(0)} \wedge t_3^{(0)} \wedge t_4^{(1)}$

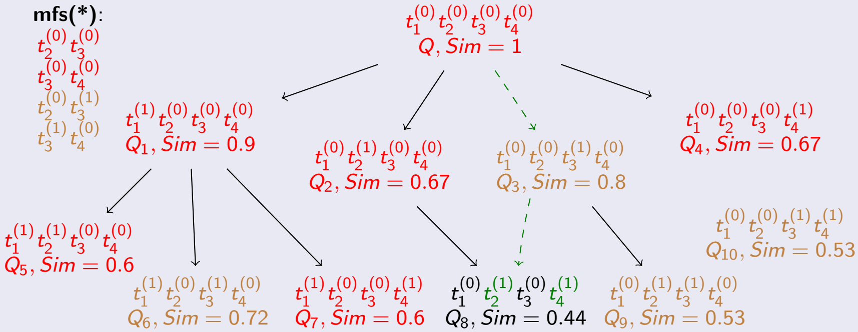
mfs(*):

$t_2^{(0)} t_3^{(0)}$

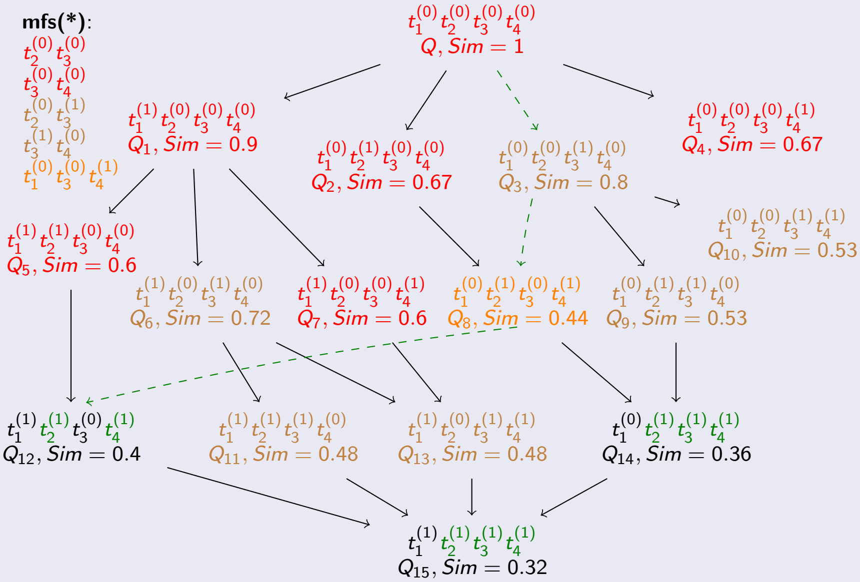
$t_3^{(0)} t_4^{(0)}$

$t_2^{(0)} t_3^{(1)}$

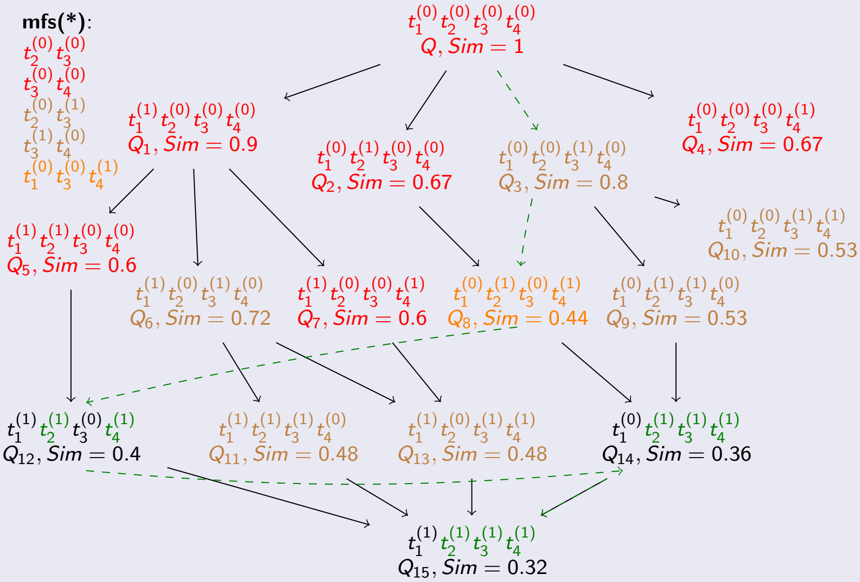
$t_3^{(1)} t_4^{(0)}$



Relaxation Strategies (4): Full MFS Based Search



Relaxation Strategies (4): Full MFS Based Search



Relaxation Strategies (4): Full MFS Based Search

⇒ Full MFS Based Search (F-MFS) idea

- Find new MFS in relaxed queries which failed
- New MFS contains at least one repaired MFS
- Using an adaptation of LBA algorithm for finding new MFS

- Example: Q has two MFS, $Q^{*} = t_2^{(0)} \wedge t_3^{(0)}$, $Q^{**} = t_3^{(0)} \wedge t_4^{(0)}$
- Failing relaxed query $Q' = t_1^{(0)} \wedge t_2^{(1)} \wedge t_3^{(0)} \wedge t_4^{(1)}$
- No MFSs still fail in Q'
- Failing subquery part with a repaired MFS $t_1^{(0)} \wedge t_3^{(0)} \wedge t_4^{(1)}$
- Execution order $Q_3, Q_9, Q_{12}, Q_{14}, Q_{15}$ not change
- Five relaxed queries executed in the worst case (68% pruned)
- High cost time due to LBA computation

Outline

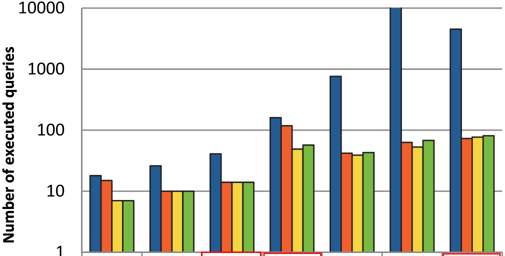
- **Motivation and Problem Statement**
- **Preliminaries**
- **Query Relaxation Strategies**
- **Experimental results**
- **Conclusion and Perspectives**

Experimental Setup

- Java implementation of MBS, O-MBS and F-MBS
  `http://www.lias-lab.fr/forge/projects/qars`
- Triplestores: Jena TDB and Virtuoso
- Dataset: LUBM100 and LUBM1000
- Queries: 7 failing queries ranging 1 to 15 triple patterns (TP)
- Hardware: Intel XEON CPU E5-2630 4Ghz and 32GB RAM
- Comparison to Best-First Search algorithm (BFS)[Huang12]
- Experiments
  - Scalability comparison of different approaches
  - Impact of triplestore: Jena TDB vs Virtuoso
  - Impact of the size of datasets

Experiment observations

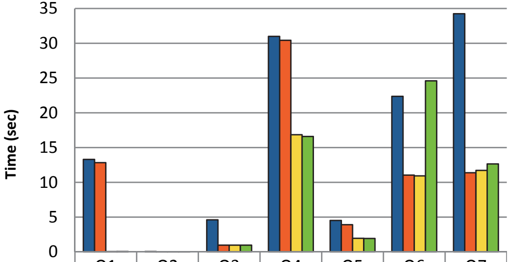
- Our algorithms outperformed BFS
- MBS, O-MBS and F-MBS performance comparison
  - Efficiency order for Q7: MBS, O-MBS and F-MBS
  - Efficiency order for Q4: O-MBS, F-MBS, MBS
  - For Q3 the three has the same number of executed queries



	Q1	Q2	Q3	Q4	Q5	Q6	Q7
BFS	18	26	41	160	761	13809	4520
MBS	15	10	14	118	42	63	73
O-MBS	7	10	14	49	39	53	77
F-MBS	7	10	14	57	43	68	81

Experiment observations

- Our algorithms outperformed BFS
- MBS, O-MBS and F-MBS performance comparison
- Cost time evaluation confirm the previous results



	Q1	Q2	Q3	Q4	Q5	Q6	Q7
BFS	13,3	0,06	4,61	30,99	4,52	22,37	34,25
MBS	12,83	0,01	0,97	30,43	3,91	11,04	11,38
O-MBS	0,05	0,01	0,97	16,85	1,95	10,93	11,72
F-MBS	0,06	0,01	0,97	16,60	1,93	24,60	12,65

Results

Experiments results

- MFS based process outperformed BFS
- MBS, O-MBS and F-MBS: Depend to the query and its MFS
- Deal with MFS computation time vs relaxed queries pruned
- In most of the cases, we observe that O-MBS is a good deal

Results

Experiments results

- MFS based process outperformed BFS
- MBS, O-MBS and F-MBS: Depend to the query and its MFS
- Deal with MFS computation time vs relaxed queries pruned
- In most of the cases, we observe that O-MBS is a good deal

- The results are the same in Virtuoso and Jena TDB
- The triple store does not change the factor between process
- The size of the dataset does not change the factor

Outline

- **Motivation and Problem Statement**
- **Preliminaries**
- **Query Relaxation Strategies**
- **Experimental results**
- **Conclusion and Perspectives**

Conclusion and Perspectives

- Contribution: 3 strategies to improve relaxation
  - MBS: pruned relaxed queries based on MFS of initial query
  - O-MBS: MBS + search if relaxed MFSs are still MFSs in relaxed queries. Pruned the search space with these new MFSs
  - F-MBS: O-MBS + if relaxed queries fail then compute their MFSs
- Experimental results
  - Our Strategies outperform the state-of-the-art algorithm
  - O-MBS is the best tradeoff
- Perspectives
  - Extend our approach to support other SPARQL queries
  - Relevance of our approach with other relaxation operators
  - Combination of this process with another cooperative techniques for handling unsatisfactory answer problem