Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search

Sébastien Ferré
Team SemLIS, Data and Knowledge Management, IRISA/Univ. Rennes 1

Extended Semantic Web Conference (ESWC)
6 June 2018, Heraklion, Crete, Greece

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES

UMR IRISA LIS

Overview

Overview

Query Relaxation in RDF

- Not a new topic
  - *Hurtado, Poulovassilis, Wood [2008]*
  - *Dolog et al. [2009]*
  - earlier work in deductive databases: *Gaasterland [1997]*
- What is query relaxation?
  - making a (SPARQL) query more general by relaxing constraints
    $\Rightarrow$ relaxed queries
  - e.g.: replacing a class by a super-class, a term by a variable
  - in general, conjunctive queries only
- Why relax queries ?
  - to avoid empty results or to get more results
    e.g., for users with poor knowledge of the schema
  - to find approximate answers

Research Question

Is it possible to apply query relaxation algorithms in order to do similarity search, i.e. identify the nodes of an RDF graph that are most similar to a given node n ?

- Idea
  - the query to be relaxed is the description of n
  - the approximate answers are the similar nodes of n
- Challenges
  1. node descriptions make up for (very) large queries (10s to 100s)
  2. the number of relaxed queries grows exponentially with query size
  3. many relaxation steps are necessary to find approximate answers
  4. each relaxed query may have a combinatorial number of matchings

The problem looks untractable !

Research Question

Is it possible to apply query relaxation algorithms in order to do similarity search, i.e. identify the nodes of an RDF graph that are most similar to a given node n ?

- Idea
  - the query to be relaxed is the description of n
  - the approximate answers are the similar nodes of n
- Challenges
  1. node descriptions make up for (very) large queries (10s to 100s)
  2. the number of relaxed queries grows exponentially with query size
  3. many relaxation steps are necessary to find approximate answers
  4. each relaxed query may have a combinatorial number of matchings

The problem looks untractable !

Our results in short for PARTITION+LAZYJOINS

- on 3 datasets

dataset	nb. nodes	nb. triples
MONDIAL	10k	12k triples
LUBM10	315k	1.3M triples
LUBM100	3M	13M triples

- for short queries (3-7 elements) *query relaxation*
  - one order of magnitude faster on full relaxation
  - no combinatorial explosion with nb. of relaxation steps
  - less sensitive to query complexity
  - scales linearly with total number of nodes
- for LUBM10 long queries (avg. 200 elts) *similarity search*
  - existing approaches fail for all long queries with timeout=10min
    - ⋆ they can only perform 3 relaxation steps
    - ⋆ which retrieve only 5% clusters of approximate answers
  - our approach retrieves
    - ⋆ 100% clusters (for all queries) under 10min (avg. 3min)
    - ⋆ 50% clusters in 30s
    - ⋆ 20% clusters in 2s

Our results in short for PARTITION+LAZYJOINS

- on 3 datasets

dataset	nb. nodes	nb. triples
MONDIAL	10k	12k triples
LUBM10	315k	1.3M triples
LUBM100	3M	13M triples

- for short queries (3-7 elements) *query relaxation*
  - one order of magnitude faster on full relaxation
  - no combinatorial explosion with nb. of relaxation steps
  - less sensitive to query complexity
  - scales linearly with total number of nodes
- for LUBM10 long queries (avg. 200 elts) *similarity search*
  - existing approaches fail for all long queries with timeout=10min
    - ⋆ they can only perform 3 relaxation steps
    - ⋆ which retrieve only 5% clusters of approximate answers
  - our approach retrieves
    - ⋆ 100% clusters (for all queries) under 10min (avg. 3min)
    - ⋆ 50% clusters in 30s
    - ⋆ 20% clusters in 2s

Our results in short for PARTITION+LAZYJOINS

- on 3 datasets

dataset	nb. nodes	nb. triples
MONDIAL	10k	12k triples
LUBM10	315k	1.3M triples
LUBM100	3M	13M triples

- for short queries (3-7 elements) *query relaxation*
  - one order of magnitude faster on full relaxation
  - no combinatorial explosion with nb. of relaxation steps
  - less sensitive to query complexity
  - scales linearly with total number of nodes
- for LUBM10 long queries (avg. 200 elts) *similarity search*
  - existing approaches fail for all long queries with timeout=10min
    - ★ they can only perform 3 relaxation steps
    - ★ which retrieve only 5% clusters of approximate answers
  - our approach retrieves
    - ★ 100% clusters (for all queries) under 10min (avg. 3min)
    - ★ 50% clusters in 30s
    - ★ 20% clusters in 2s

Overview

Relaxation of Query Elements

- query element = triple pattern or atomic filter expression
- common kinds of relaxations
  - class $\rightsquigarrow$ superclass: ?x a :SciFiFilm $\rightsquigarrow$?x a :Film
  - prop $\rightsquigarrow$ superprop: ?x :director :Spielberg $\rightsquigarrow$?x :creator :Spielberg
  - node $\rightsquigarrow$ variable: ?x :director :Spielberg $\rightsquigarrow$?x :director ?y

Definition: *relax*(*e*)

The set of most specific relaxations of query element *e*.

- ex: relax(?x :director :Spielberg) =
  {?x :creator :Spielberg, ?x :director ?y}

Relaxation of Queries

- query: $Q = (X \leftarrow P)$
  - pattern P: a set of query elements (conjunctive query)
  - projection X: a set of distinguished variables from P
- query normalization: to have a single copy of each RDF node
  - before: ?x ← ?x a :SciFiFilm ; :director :Hanks ; :actor :Hanks.
  - after: ?x ← ?x a :SciFiFilm ; :director ?y ; :actor ?y. FILTER (?y = :Hanks)
- relaxation = replacing an element $e \in P$ by $relax(e)$
  - when $relax(e) = \emptyset$: element removal
  - direct relaxations of above query pattern
    - ?x a :Film ; :director ?y ; :actor ?y. FILTER (?y = :Hanks)
    - ?x a :SciFiFilm ; :creator ?y ; :actor ?y. FILTER (?y = :Hanks)
    - ?x a :SciFiFilm ; :director ?y _. FILTER (?y = :Hanks)
    - ?x a :SciFiFilm ; :director ?y ; :actor ?y. _
- $RQ(Q)$ = relaxed queries of Q, in one or several relaxation steps
  - relaxation distance: number of steps to reach $Q' \in RQ(Q)$

Approximate Answers and Proper Relaxed Queries

Given an RDF graph G, and a query Q:

- $ans(Q', G)$ = answers of $Q' \in RQ(Q)$ = approximate answers of Q

proper answers

$$properAns(Q', G) = ans(Q', G) \setminus \bigcup_{Q'' \in RQ(Q) | Q'' < Q'} ans(Q'', G)$$

The approx. answers not found in more specific relaxed queries.

proper relaxed queries of Q in G

$$PRQ(Q, G) = \{Q' \in RQ(Q) \mid properAns(Q', G) \neq \emptyset\}$$

The relaxed queries with proper answers.

- every approx. answer belongs to a single PRQ
- hence, $|PRQ|$ is bounded by $|nodes(G)|^{|X|}$ (linear when $|X| = 1$), whereas $|RQ|$ is bounded by $2^{|Q|}$ (exponential)

Application to similarity search

Given an RDF graph G, and a query node n:

- query $Q(n)$ = ?x $\leftarrow$ description of n (replacing n by ?x)
  - $|Q(n)|$ large, hence $|RQ(Q(n))|$ untractable
  - $|X| = 1$, hence $|PRQ(Q(n), G)|$ linear in $|nodes(G)|$
- every other node $n' \in nodes(G)$
  has a single $Q' \in PRQ(Q(n), G)$ s.t. $n' \in properAns(Q', G)$
  - Q' characterizes what n' has in common with n
    it is a symbolic form of similarity
- several numerical measures can be derived from PRQ Q'
  - extensional distance: $|ans(Q', G)|$ (number of nodes)
  - intensional similarity: $|Q'|$ (number of query elements)
  - relaxation distance: from $Q(n)$ to Q' (number of relaxation steps)

Overview

Algorithm PARTITION: principle

Lemma

The set of PRQs defines a partition of all approx. answers because every approx. answer belongs to a single PRQ.

- We propose to incrementally partition the set of approx. answers by incrementally refining relaxed queries (reverse relaxation)
  - rather than enumerating relaxed queries Q' and evaluate $ans(Q', G)$ (algo RELAXENUM)
  - rather than enumerating approx. answers n' and compare $Q(n')$ with Q (algo NODEENUM)
- Benefits
  - partition: (much) less parts than nodes or relaxed queries
  - incremental:
    - ★ factorization of evaluations/comparisons
    - ★ any-time: interrupted ⇒ coarser partition

Algorithm PARTITION: description

Given graph G, and query $Q = X \leftarrow P$

- state = a set of clusters $C = X \leftarrow P'/E$
  - represented by a relaxed pattern P' and candidate elements E
  - that partitions the set of relaxed queries $RQ(Q)$
  - that partitions the set of approx. answers
- init: $C_0 = X \leftarrow \emptyset/P$ (all RQs, all approx. answers)
- step: splitting cluster $C = X \leftarrow P'/E$ by picking $e \in E$
  - $C_e = X \leftarrow P' + e/E - e$: approx. answers matching e
  - $C_{\overline{e}} = X \leftarrow P'/E - e + relax(e)$: approx. answers not matching e
- stop: when no cluster can be split further
  i.e., no candidate element OR no answer in cluster

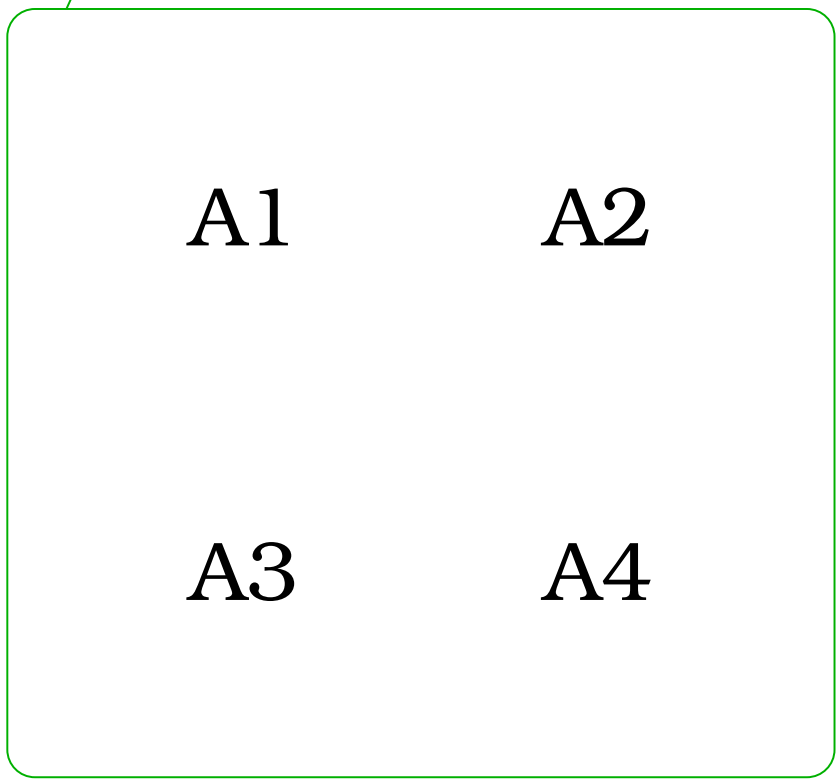
Algorithm PARTITION: illustration

$Q = ?f \leftarrow e, f, g$

/ e f g

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

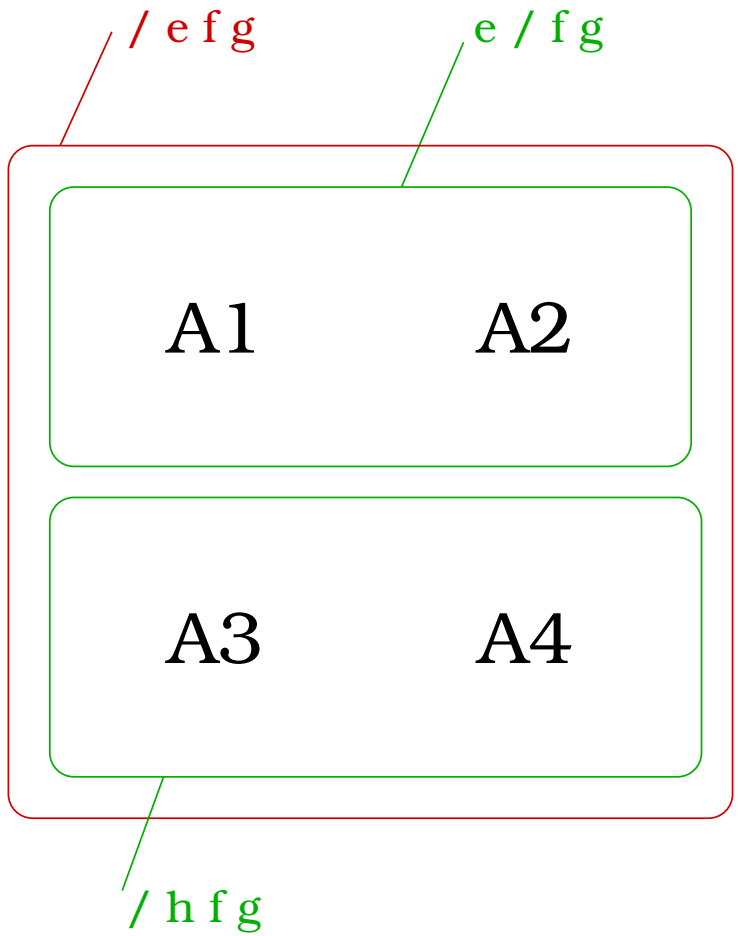
relax(e) = {h}



Algorithm PARTITION: illustration

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

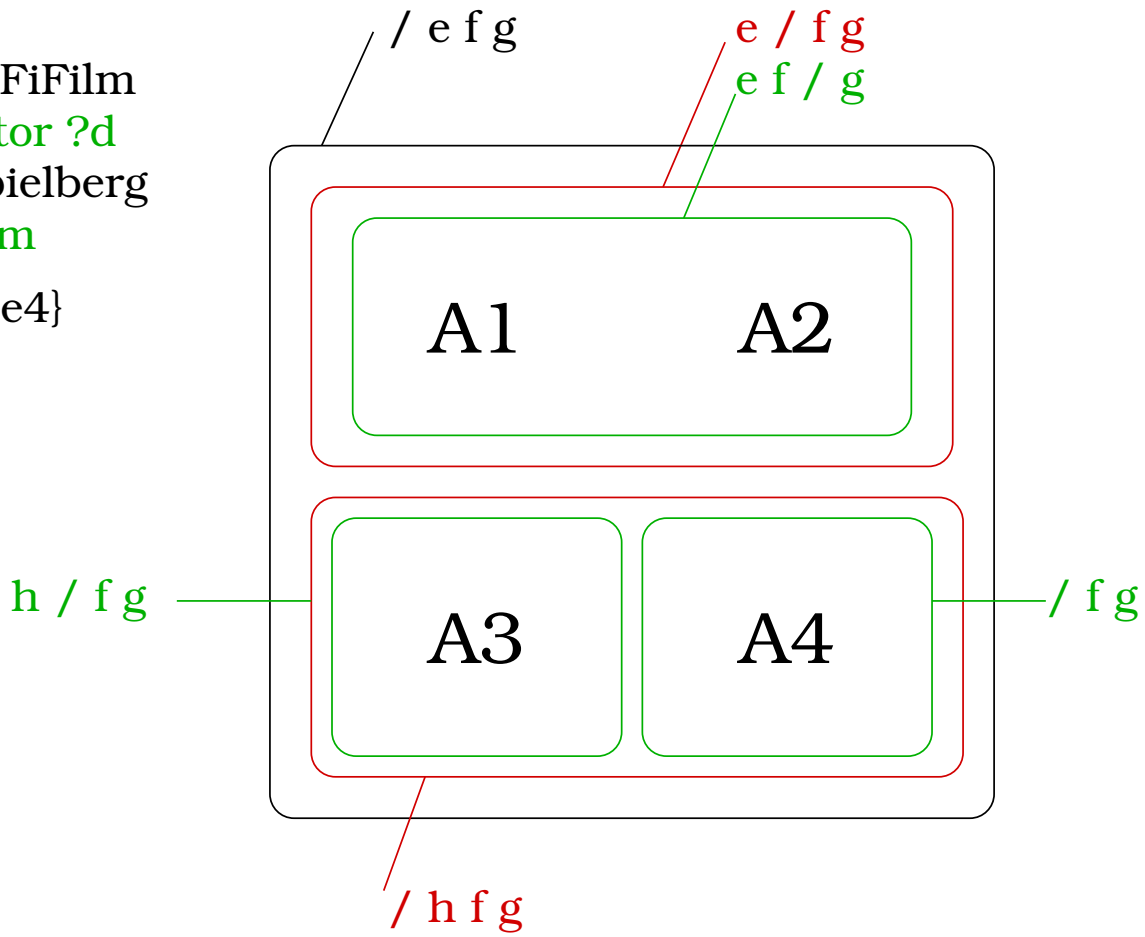
relax(e) = {h}



Algorithm PARTITION: illustration

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

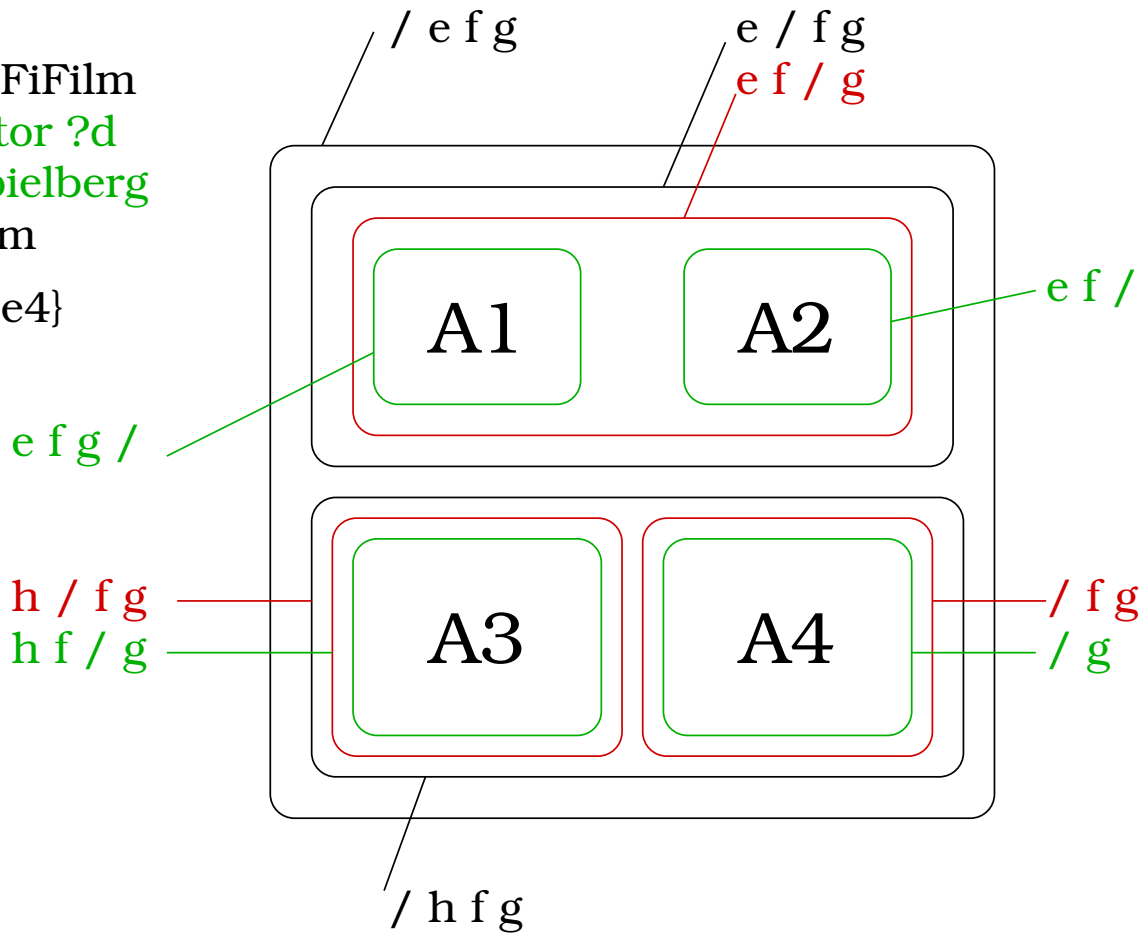
relax(e1) = {e4}



Algorithm PARTITION: illustration

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

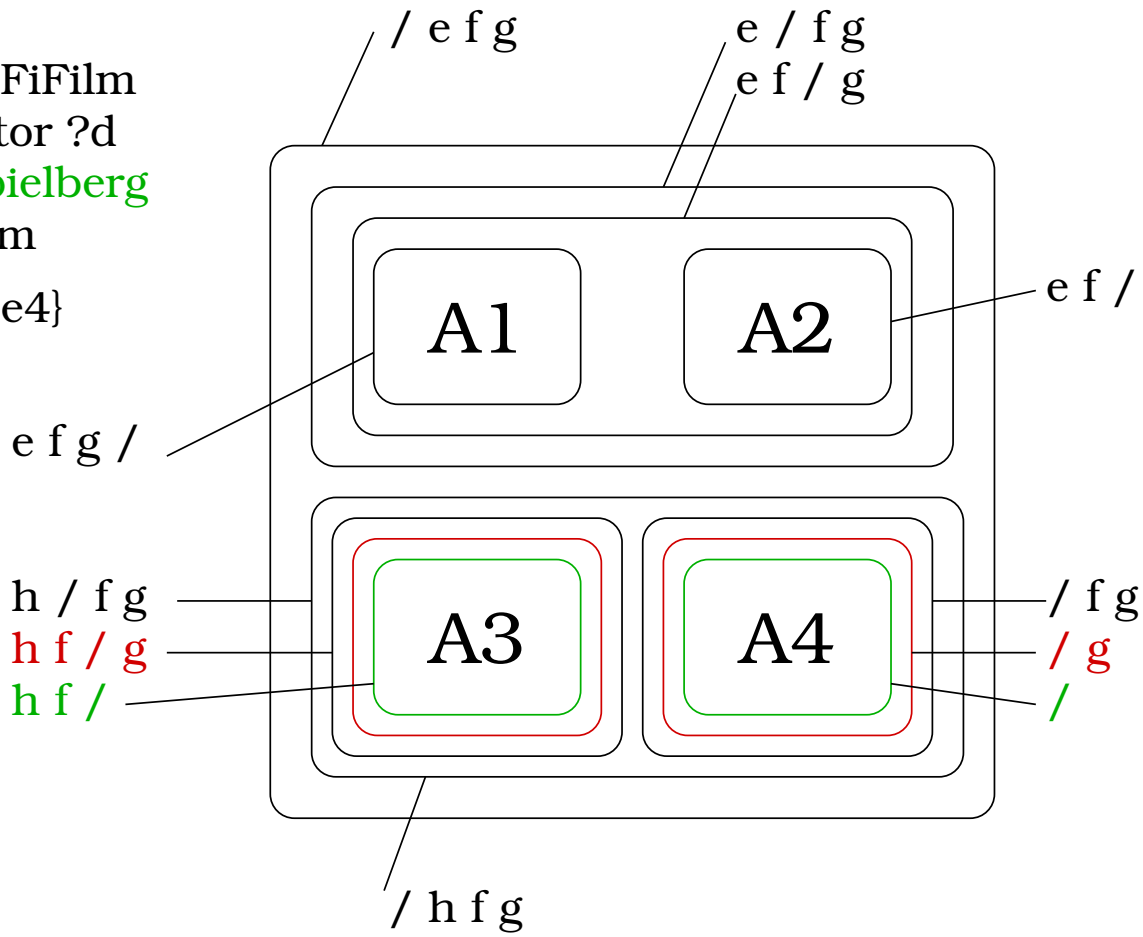
relax(e1) = {e4}



Algorithm PARTITION: illustration

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

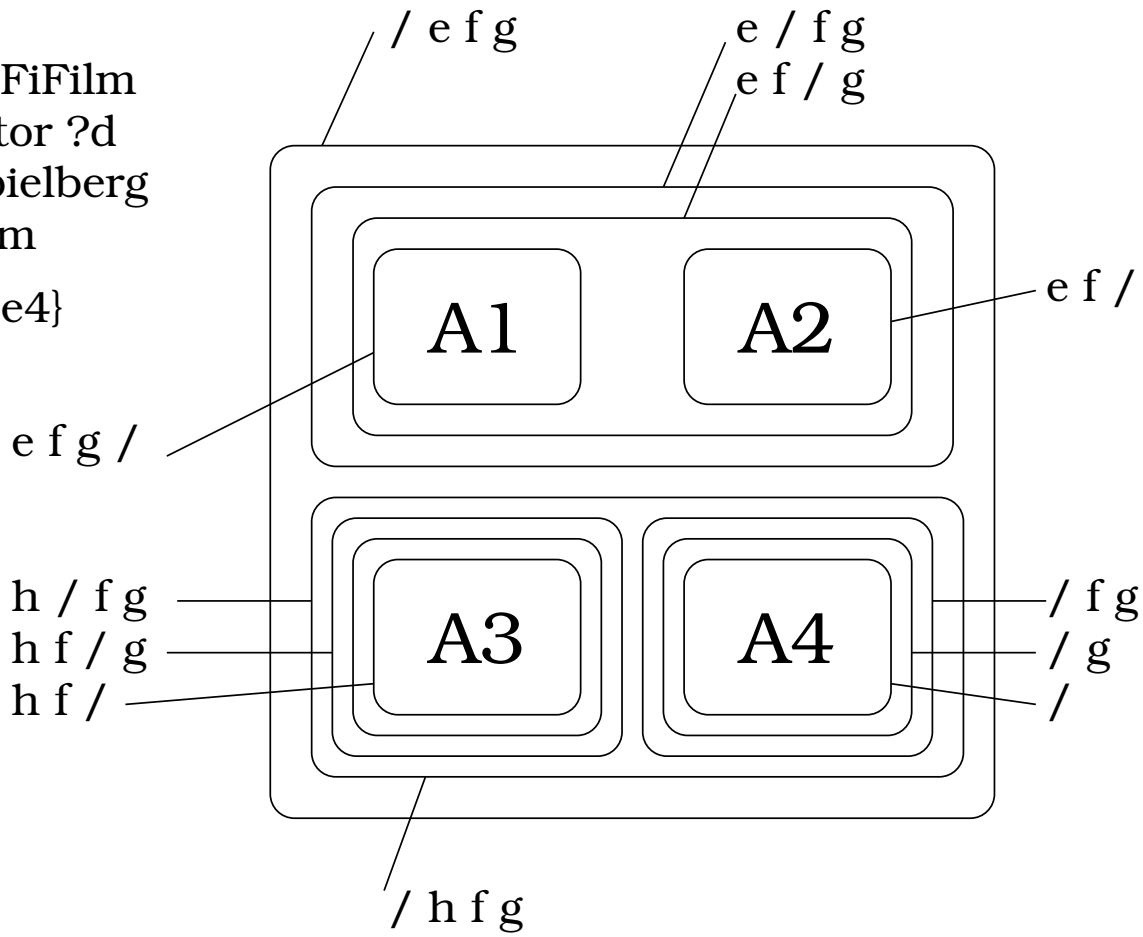
relax(e1) = {e4}



Algorithm PARTITION: illustration

e = ?f a :SciFiFilm
f = ?f :director ?d
g = ?d = :Spielberg
h = ?f a :Film

relax(e1) = {e4}



Intractability of Relaxed Query Evaluation

- Conditions
  - when a property with multiple values in the query
    - ⋆ a common situation in similarity search because the query is a node description
      e.g., film's actors, document's authors, country's neighbours, ...
  - when many values are relaxed (hence, many matchings)
- Example: on films, worst-case scenario
  - G has N films with p actors each
  - Q = ?x ← ?x a :Film. ?x :actor :A1, :A2, ..., :An .
  - Q' = ?x ← ?x a :Film. ?x :actor ?y1, ?y2, ..., ?yp .
    + weak constraints on ?y1 ... ?yp
- Complexity
  - the number of answers is bounded by N
  - BUT the number of matchings (of the relaxed pattern) is in $O(Np^p)$
  - for 1000 films with 10 actors, 10^{13} matchings !

Algorithm PARTITION+LAZYJOINS

- We only need the answers, i.e. the projection of matchings to X (?x in the example)
- We do not need to enumerate matchings

Idea

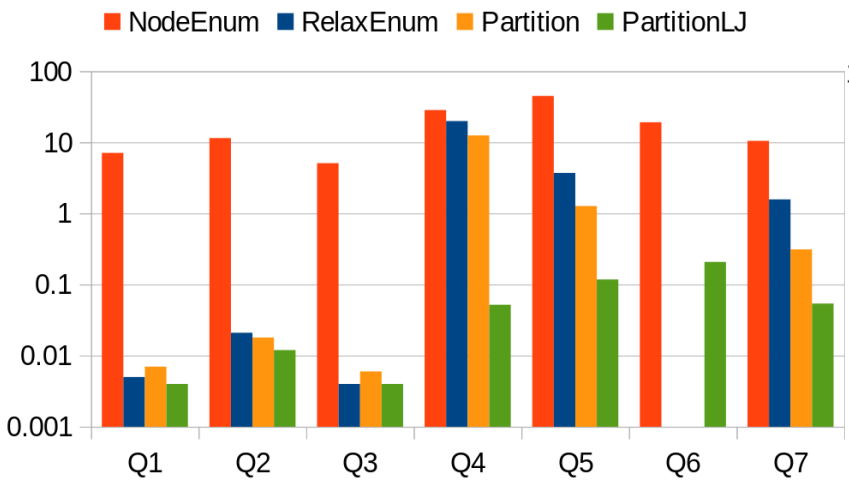
Join lazily, only as much as necessary, according to X and interactions between query elements (shared variables).

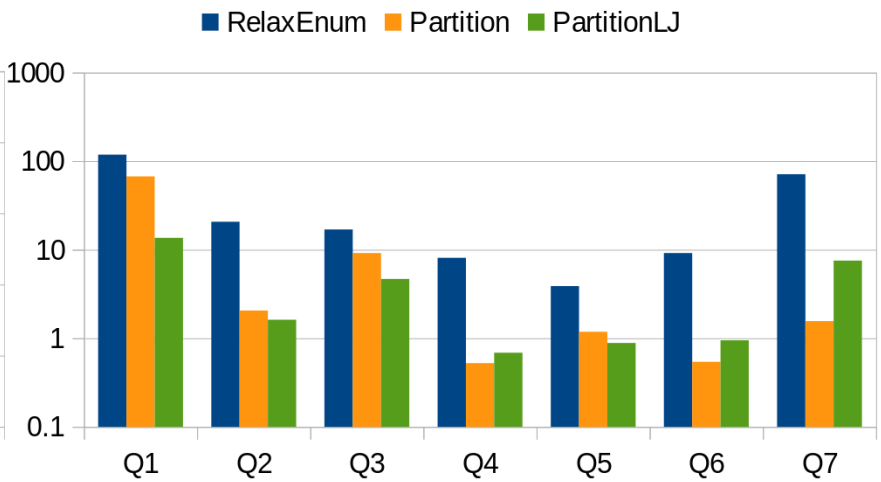
- Example: films with actors
  - one set of matchings for variable ?x
  - one set of matchings for variables ?x, ?yi (for each ?yi in ?y1...?yp)
  - no set of matchings for ?x, ?y1, ..., ?yp
- Complexity: $O(Np^2)$ matchings in total
  - from exponential to polynomial
  - for 1000 films with 10 actors, 10^5 matchings instead of 10^{13} !

Overview

Efficiency of Query Relaxation

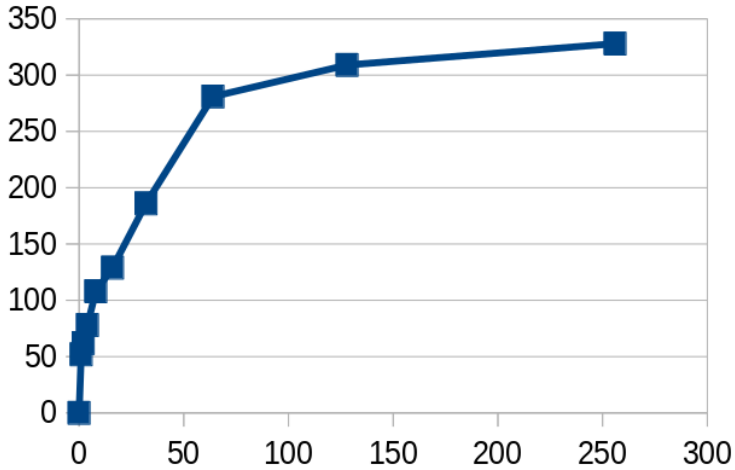
- Comparison between NODEENUM, RELAXENUM, PARTITION, and PARTITION+LAZYJOINS
- Same settings as previous work: short queries (<10 elements)
- Observations (MONDIAL and LUBM10)
  - one order of magnitude faster on full relaxation
  - no combinatorial explosion with nb. of relaxation steps
  - less sensitive to query complexity
  - scales linearly with total number of nodes (test on LUBM100)





Efficiency of Similarity Search

- Novel settings: long queries (avg. 200 elts) on LUBM10
  - existing approaches fail for all long queries with timeout=10min
    - they can only perform 3 relaxation steps
    - which retrieve only 5% PRQs of approximate answers
  - our approach retrieves
    - 100% PRQs (for all queries) under 10min (avg. 3min)
    - 50% PRQs in 30s
    - 20% PRQs in 2s



Overview

Conclusion

- query relaxation made more efficient
  - by removing two bottlenecks:
    1. enumeration of relaxed queries
    2. enumeration of matchings
- to the point it can be applied to similarity search
  - where graph patterns-as-explanation are available

Achievement

Full exploration of the relaxation space of a large query (up to 1500 elements) on a 1M-triples RDF graph (LUBM10) in a few minutes.

- anytime algorithm with early output of PRQs

Perspectives

- Improving the algorithms
  - heuristics for choosing the cluster to split, and the splitting element
  - new kinds of element relaxations
    e.g. gradual relaxation of URIs and literal values
    number $\rightsquigarrow$ interval
- Applying the algorithms to various tasks
  - classification
  - case-based reasoning
  - concept learning
  - link prediction
  - analogical reasoning/inference

The End

Questions ?

Effectiveness of Similarity Search

- How meaningful are PRQs for similarity search ?
  - Empirical evaluation by the inference of property values
- Settings
  - real data: MONDIAL, 244 countries
  - task: infer the (set of) values of 8 properties
    - ★ e.g., language, neighbor, government
    - ★ each property infered from all other properties
  - 3 nearest neighbors, majority vote
  - ranking based on intensional similarity derived from PRQs

	continent	*religion*	*wasDependentOf*	*language*	*neighbor*	*ethnicGroup*	*government*	*dependentOf*	*all*
precision	0.97	0.80	0.77	0.75	0.43	0.65	0.82	0.50	0.75
recall	0.93	0.76	0.47	0.34	0.33	0.25	0.19	0.00	0.48
F1-score	0.93	0.74	0.47	0.36	0.30	0.28	0.19	0.00	0.56
baseline F1-score	0.12	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.05

Examples on MONDIAL's entities

Clusters of nearest neighbors

- showing intensional similarity
- not showing explanation (PRQ) [see online]

Examples:

- Peru: Bolivia (28); Colombia (26); Argentina (24); Panama (22); Paraguay (22); Honduras, El Salvador, Guatemala, Nicaragua, Mexico, Costa Rica (22); ...
- Spanish: English (13); French (5); Catalan, Galician (5); Aymara, Quechua (5); Creole (5); ...
- Tahiti: Savaii, VitiLevu, Hokkaido, Hawaii, Maui (8, *volcanic Pacific islands*); Bali, Sulawesi, ... (7, *other volcanic islands*); ...