Estimating the Number of Answers to CQs using Graph Summarisation

Giorgio Stefanoni, Egor V. Kostylev, Boris Motik

University of Oxford — Department of Computer Science — Information Systems Group

Problem Statement

- Inputs:
  - RDF graph G
  - Full conjunctive query q (without projection)
- Output: an estimate of the number of answers to q on G
- Very important problem in practice with many applications:
  1. Show the total number of answers to the user to aid browsing
  2. Query planning!

Existing Approaches

$$q = R(a, Y) \wedge S(Y, Z)$$

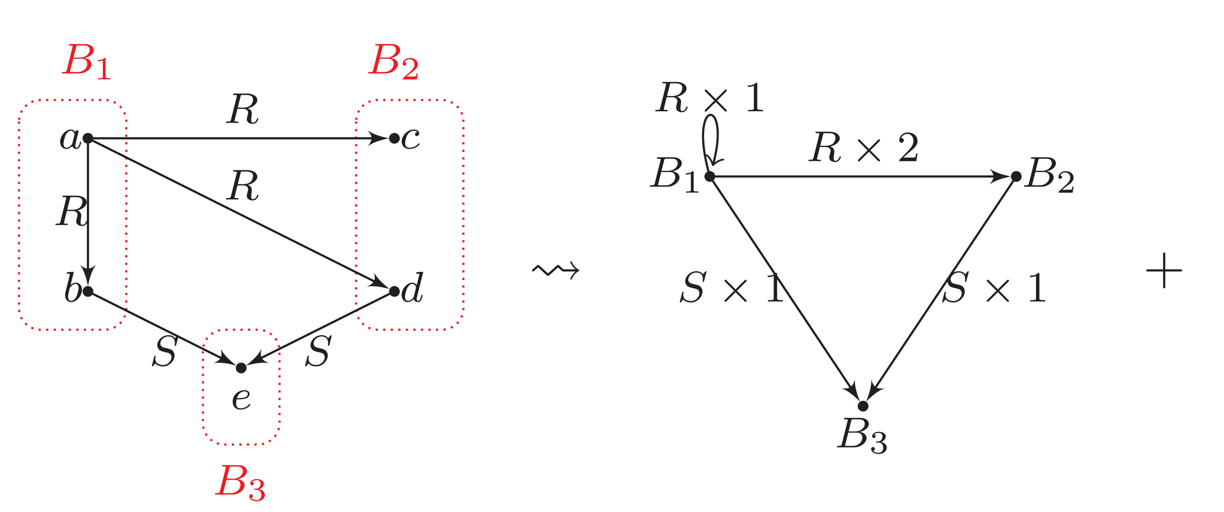
1. Gather statistics about the graph; for example:
   - Count the edges of each type: $T^R = |R|$ and $T^S = |S|$
   - Count the variability of each relation's column

   $$V^{R,2} = |\{Y \mid \exists X.\langle X, Y\rangle \in R\}| \quad V^{S,1} = |\{Y \mid \exists Z.\langle Y, Z\rangle \in S\}|$$
2. Combine statistics into an estimate; for example:
   - $estimate(q) = \frac{T^R \times T^S}{\max\{V^{R,2}, V^{S,1}\}}$

- Cardinality estimation recognised as a critical open problem in DBs
  - Estimations rely on uniformity, independence, and containment assumptions $\Rightarrow$ often fail in practice
  - Result can depend on the order in which we process the query
  - Estimations can be way off mark, particularly for 'long' queries
  - $\Rightarrow$ No formal underpinning (just a hack)!

Main Idea and Goals

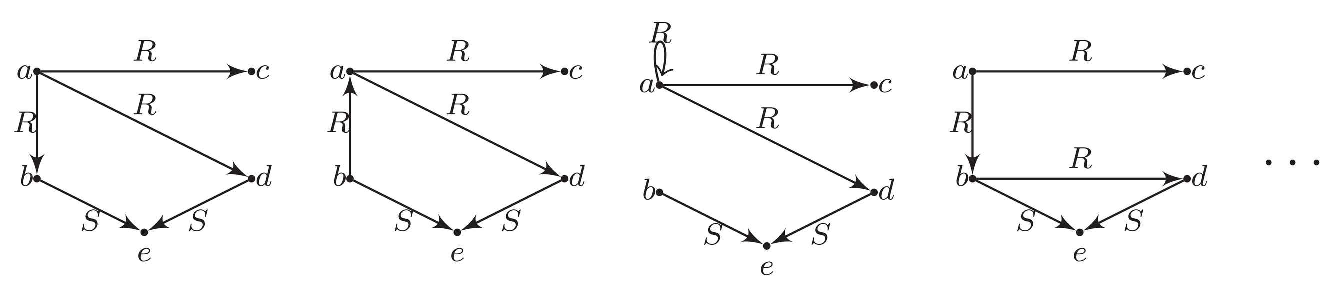
1. Use a multi-dimensional statistical representation of the graph:
   $\Rightarrow$ summarise the graph by merging nodes together



μ
$a \mapsto B_1$
$b \mapsto B_1$
$c \mapsto B_2$
$d \mapsto B_2$
$e \mapsto B_3$

Graph G — Summary $\mathcal{S}$ — Mapping

2. Semantics: summary represents all possible 'compatible' graphs



3. Estimate the expected answer over all 'compatible' graphs

$$E_q = \frac{q^{G_1} + q^{G_2} + q^{G_3} + \cdots}{\#\text{ of compatible graphs}}$$

4. Compute the standard deviation σ_q and use Chebyshev's inequality (1) to establish confidence intervals:

$$P(|X_q - E_q| \geq k \times \sigma_q) \leq \frac{1}{k^2} \qquad (1)$$

where $k > 0$ and X_q is the random variable such that $X_q(G_i) = q^{G_i}$ for each compatible graph G_i

Computing Expectation and STD Deviation

- We give a closed-form formula to compute E_q
- Computing E_q requires iterating only over the summary $\mathcal{S}$, not over the 'compatible' graphs
- For example, $E_q = \sum_{b_Y} \sum_{b_Z} \frac{R^S(\mu(a), b_Y) \times S^S(b_Y, b_Z)}{[\![\mu(a)]\!] \times [\![b_Y]\!]}$
- Computing E_q is intractable, but:
  - Intractable in query and summary size
  - Polynomial for CQs of bounded hypertreewidth without unification
  - Idea for handling unification: each group of unifiable atoms must be 'covered' by one decomposition node
- Standard deviation can be computed as $\sigma_q^2 = E_{q \wedge q'}$, where q' is the query obtained from q by replacing each variable with a fresh one.

Summarising Graphs

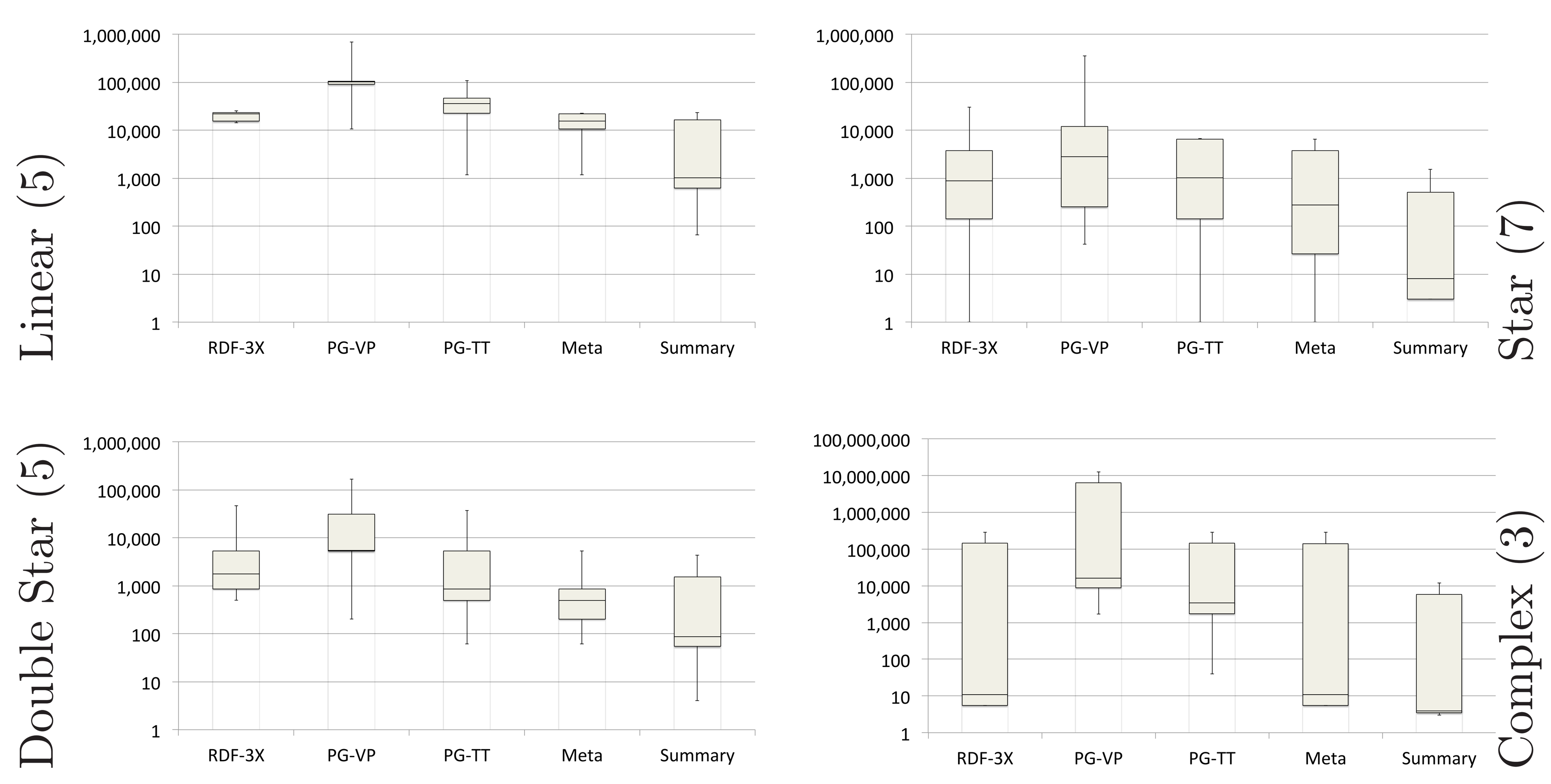
- We summarise RDF graphs based on structural similarity:
  - Assign to each vertex v a type $T(v)$ and a neighbourhood $N(v)$
  - Similarity based on Jaccard index $J(v_1, v_2) = \frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
- Basic algorithm:
  - Merge v_1 and v_2 s.t. $T(v_1) = T(v_2)$ and $J(v_1, v_2)$ is largest
  - Repeat until the number of edges falls below a threshold
- Naïve algorithm is quadratic and can only handle very small graphs.
- $\Rightarrow$ Efficient approximate solution using min-hashing

Preliminary Evaluation

Waterloo SPARQL Diversity Test Suite:

	WatDiv-100	Summary	Compression ratio
Vertices	1,053,224	539	1,954
Edges	10,903,668	9,982	1,092

Comparing estimation errors:



Comparison Systems:

- RDF-3X: an RDF system with estimator tailored to RDF workloads
- PG-VP: PostgreSQL with vertical partitioning storage
- PG-TT: PostgreSQL with triple table partitioning storage
- Meta: the best of the above three systems
- Summary: our implementation