# Distance-Based Triple Reordering for SPARQL Query Optimization

**Marios Meimaris** and George Papastefanatos
*Athena Research Center*

{m.meimaris, gpapas}@imis.athena-innovation.gr



ATHENA
Research & Innovation
Information Technologies

# Preliminaries

- RDF (Resource Description Framework)
  - Abstract Data model for Linked Data
  - Based on *Triples*: Subject-Predicate-Object
  - RDF datasets are *Directed Labelled Graphs*
- SPARQL
  - Query Language for RDF
  - Expressive
  - Complex

# Triple Reordering

- SPARQL Basic Graph Pattern Optimization
  - Reordering triple patterns is a significant part of low-level SPARQL query optimization.
  - The aim is to find fastest query execution plan
    - Minimization of execution time
    - Minimization of intermediate results

# Triple Reordering

```
SELECT ?X ?Y
WHERE {
      ?X rdf:type ub:Student .
      ?Y rdf:type ub:Department .
      ?X ub:memberOf ?Y .
      ?Y ub:subOrganization Of ub:University0 .
      ?X ub:email ?Z .
}
```

# Triple Reordering

```
SELECT ?X ?Y
WHERE {
t1      ?X rdf:type ub:Student .
t2      ?Y rdf:type ub:Department .
t3      ?X ub:memberOf ?Y .
t4      ?Y ub:subOrganization Of ub:University0 .
t5      ?X ub:email ?Z .
}
```

# Triple Reordering

```
SELECT ?X ?Y
WHERE {
t1      ?X rdf:type ub:Student .
t2      ?Y rdf:type ub:Department .
t3      ?X ub:memberOf ?Y .
t4      ?Y ub:subOrganization Of ub:University0 .
t5      ?X ub:email ?Z .
}
```

- Assumption: 99k nodes with rdf:type ub:Student
- How does the order of execution affect total runtime?
  - E.g. $\{t_1, t_5, t_3, t_2, t_4\}$ vs. $\{t_2, t_4, t_3, t_1, t_5\}$

# Triple Reordering

```
SELECT ?X ?Y
WHERE {
t1      ?X rdf:type ub:Student .
t2      ?Y rdf:type ub:Department .
t3      ?X ub:memberOf ?Y .
t4      ?Y ub:subOrganization Of ub:University0 .
t5      ?X ub:email ?Z .
}
```

- Assumption: 99k nodes with rdf:type ub:Student
- How does the order of execution affect total runtime?
  - E.g. $\{t_1, t_5, t_3, t_2, t_4\}$ vs. $\{t_2, t_4, t_3, t_1, t_5\}$
- Do we need all Students, or just the ones in a suborganization of University0?

# Triple Reordering

- Finding the best order under a given cost model is a factorial problem
  - O(n!) where n = the # of triple patterns
  - All permutations of n must be considered
  - Even then, optimality is relevant to the cost model

# Related Work

- Selectivity Estimation
- Cost-based cardinality estimation
- Dynamic programming
- Heuristics

# Challenges

- Reduce the size of the search space to quadratic
- Spend less time in pre-processing
- Aim for overall acceptable performance (not universally optimal)
- Require less statistics

# Proposed Solution

- Distance-based Triple Reordering:
  1. Spread the triple patterns in a multi-dimensional space, reflect cardinalities in coordinates
  2. Derive distance matrix
  3. Rank pairs of triple patterns based on lowest distances
  4. Derive final plan

# Proposed Solution

- Distance-based Triple Reordering:
  1. Spread the triple patterns in a multi-dimensional space, reflect cardinalities in coordinates
  2. Derive distance matrix
  3. Rank pairs of triple patterns based on lowest distances
  4. Derive final plan

# Proposed Solution

- Each tp becomes an *m-dimensional vector* in the $\mathbf{Q_m}$ matrix
- Given a triple t=<s,p,o>, the value of $i^{th}$ dimension is given by:

$$f(t, m_i) = \begin{cases} card(t), & m_i \in (s, o) \\ 0, & else \end{cases}$$

where $card(t)$ is given by the following rules:

- $card(t) = card(p)$, *if p is bound, and s,o are unbound*
- $card(t) = \max(1, \frac{card(p)}{|S|})$, *if p, o is bound, and s is unbound*
- $card(t) = 1$, *if s and p are bound*
- $card(t) = |\mathsf{D}|$, *in all other cases*



ATHENA
Research & Innovation
Information Technologies

# Proposed Solution

Example instantiation of query representation space ($Q_m$ matrix)

| | ?X | Student | ?Y | Dpt | Univ0 | ?Z |
|---|---|---|---|---|---|---|
| $t_1$ | 99k | 99k | 0 | 0 | 0 | 0 |
| $t_2$ | 0 | 0 | 189 | 189 | 0 | 0 |
| $t_3$ | 106k | 0 | 106k | 0 | 0 | 0 |
| $t_4$ | 0 | 0 | 239 | 0 | 239 | 0 |
| $t_5$ | 106k | 0 | 0 | 0 | 0 | 106k |

# Proposed Solution

- Distance-based Triple Reordering:
  1. Represent the triple patterns in a multi-dimensional space, reflect cardinalities in coordinates
  2. Derive distance matrix
  3. Rank pairs of triple patterns based on lowest distances
  4. Derive final plan

# Distance matrix

- Given a matrix $Q_m$, we can compute a distance matrix in a standard way
  - Size of distance matrix is|T|x|T|
  - Captures all pair-wise distances of triple patterns
  - Plug in your distance function!

# Proposed Solution

- Distance-based Triple Reordering:
  1. Represent the triple patterns in a multi-dimensional space, reflect cardinalities in coordinates
  2. Derive distance matrix
  3. Rank pairs of triple patterns based on lowest distances
  4. Derive final plan

# Proposed Solution

- The distances are sorted in ascending order
- Append each pair ($t_a$, $t_b$) from the sorted order in the plan queue
  - If non of $t_a$ and $t_b$ are in a sub-plan, create a new sub-plan and appen them
  - If $t_a$ or $t_b$ are already in a sub-plan, append $t_b$ or $t_a$ resp. in the sub-plan
  - if both are in sub-plans, skip

# Proposed Solution

- Distance-based Triple Reordering:
  1. Represent the triple patterns in a multi-dimensional space, reflect cardinalities in coordinates
  2. Derive distance matrix
  3. Rank pairs of triple patterns based on lowest distances
  4. Derive final plan

# Proposed Solution

- Check for join variables between sub-plans
- If joins exist, push up joined variables in the top of their resp. sub-plan
- For *left-deep trees*, execute final sub-plans in order
- For *bushy trees*, execute sub-plans in parallel

# Evaluation

- LUBM dataset (~15 million triples)
- 14 original queries
- 15 new queries of different patterns
  - Star
  - Chain
  - Star-chain
  - Cyclic
- Compared against *PFJ, ONS, ANT, Jena Weighted, Jena Fixed, Virtuoso* planners

# Evaluation

- % of plans that are best, for all queries

| Jena Weighted | Jena Fixed | ONS | PFJ | ANT | Virtuoso | Distance-Based |
|---|---|---|---|---|---|---|
| 59% | 48% | 28% | 59% | 31% | 66% | **90%** |

Conclusion: Distance-based reordering yielded the best plans for all query patterns except cyclic

Thank you ☺

Questions?



ATHENA
Research & Innovation
Information Technologies