SPARQL BGP Optimization

For native RDF graph implementations

Markus Stocker,
HP Laboratories Bristol

Manchester, 23. October 2007

About me

- Markus Stocker
- Born in Switzerland, 1979, Ascona
- Languages: De, It (En, Fr)
- MSc in Informatics (2006), UZH
- HPL Bristol (6 months): Implementation of a static optimizer for Jena ARQ
- Hobbies: Jogging, swimming, tennis, photography, hiking, music

Overview

- SPARQL and Basic Graph Pattern
- The optimization problem
- Estimate selectivities
- The ARQ optimizer
  - BGP abstraction
  - Heuristics
  - The optimization algorithm
- Evaluation

SPARQL and BGP

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X ?Y ?Z
WHERE  {
?X rdf:type ub:GraduateStudent .
?Y rdf:type ub:University .
?Z rdf:type ub:Department .
?X ub:memberOf ?Z .
?Z ub:subOrganizationOf ?Y .
?X ub:undergraduateDegreeFrom ?Y
}
```

- BGP: Set of triple patterns
- Fundamental in SPARQL as they define the access to the RDF graph

The Optimization Problem

- What is the best EP for the following BGP?

```
?x rdf:type uv:Person .
?x uv:hasSocialSecurityNumber “555-05-7880”
```

- Data of the university domain
- Staff members, professors, graduates, undergraduates: all of *type* Person
- OWL schema states that,
  - Social security number property is inverse functional, i.e. object uniquely determines subject
  - Class Person is domain of the property
- Optimization: Join order, more selective first
  - Rearrange the triple patterns
  - Drop the first pattern, provided the data is consistent
- We focus on *main memory* (native) graph implementations (RDBMS is a different story!)

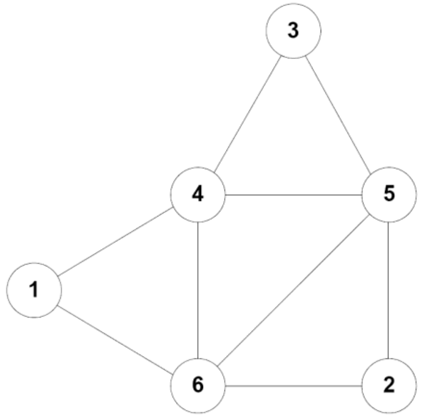
Estimate Selectivities

- Summary of statistics about RDF data
  - Total number of triples
  - Total number of resources
  - Total number of triples per predicate
  - Histogram distribution of objects
  - Result set sizes of joined patterns:
    [?x P1 ?y] and [?x P2 ?z]
- Framework for the selectivity estimation
  - Selectivity for a triple pattern
  - Selectivity for a joined triple pattern

The ARQ Optimizer I

- BGP abstraction

  1 ?X rdf:type ub:GraduateStudent .
  2 ?Y rdf:type ub:University .
  3 ?Z rdf:type ub:Department .
  4 ?X ub:memberOf ?Z .
  5 ?Z ub:subOrganizationOf ?Y .
  6 ?X ub:undergraduateDegreeFrom ?Y

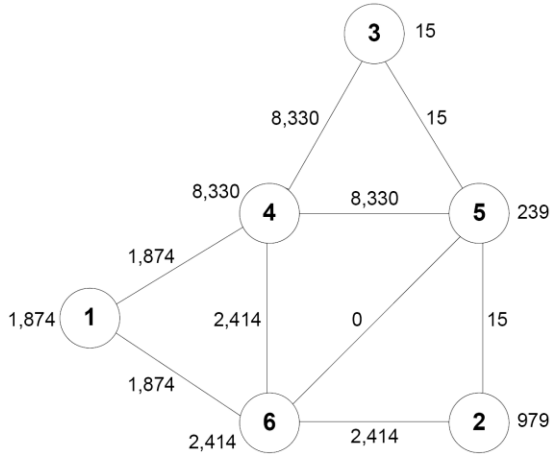


Connected graph **g**

- BGP is a set *G* of undirected connected graphs **g** = (*N*,*E*)
- The graphs **g** have different semantics to an RDF graph!
- *N* is a set of triple patterns (nodes of **g**)
- *E* is a set of joined triple patterns (edges of **g**)
- Triple patterns are joined if they share a common bound or unbound component (subject, predicate, object)

The ARQ Optimizer II

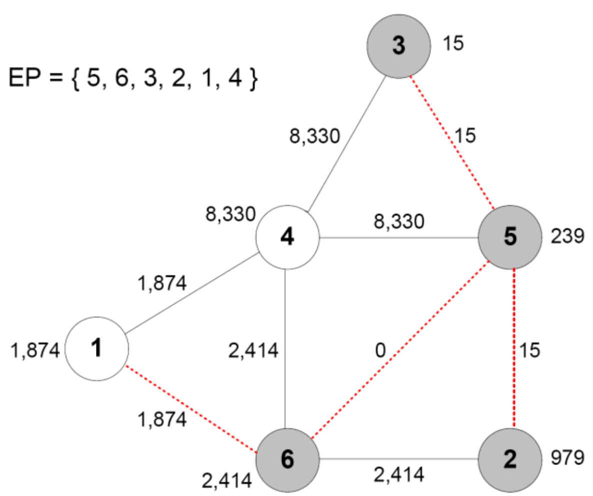
- Heuristics
  - Estimate the selectivity of (joined) triple patterns
  - Heuristics without pre-computed statistics
    - Variable counting (and variations), Jena graph statistics
  - Heuristics with pre-computed statistics
    - Build on selectivity estimation framework



Weighted connected graph.
The result set sizes of (joined) triple patterns. Heuristics compute selectivities, i.e. normalized sizes.

The ARQ Optimizer III

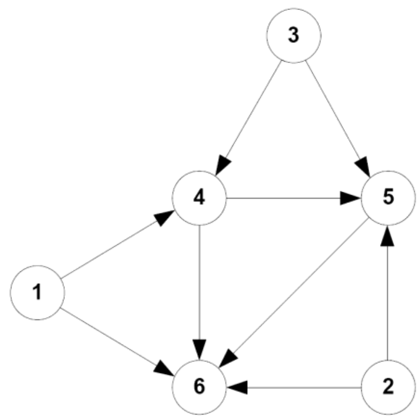
- The optimization algorithm
  - Select edge with lowest estimated selectivity, mark the nodes as visited
  - Add the nodes to EP, more selective first
  - Iteratively select the edge with
    - Lowest estimated selectivity: Minimum selectivity approach
    - Visited node: Avoid Cartesian products as intermediate result sets by selecting triple patterns (nodes) which join with previous EP



Execution Plan as DAG

Original BGP

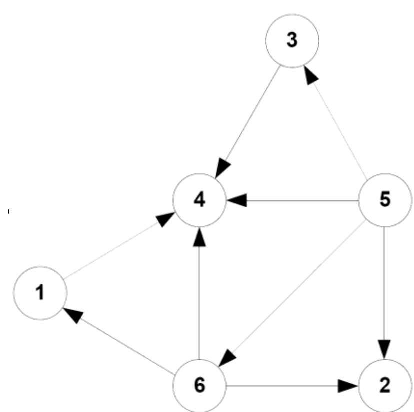
1 ?X rdf:type ub:GraduateStudent .
2 ?Y rdf:type ub:University .
3 ?Z rdf:type ub:Department .
4 ?X ub:memberOf ?Z .
5 ?Z ub:subOrganizationOf ?Y .
6 ?X ub:undergraduateDegreeFrom ?Y



Abstracted
EP as DAG for the original BGP
(3 source nodes for all adjacent arcs, 3 Cartesian products)

Optimized BGP

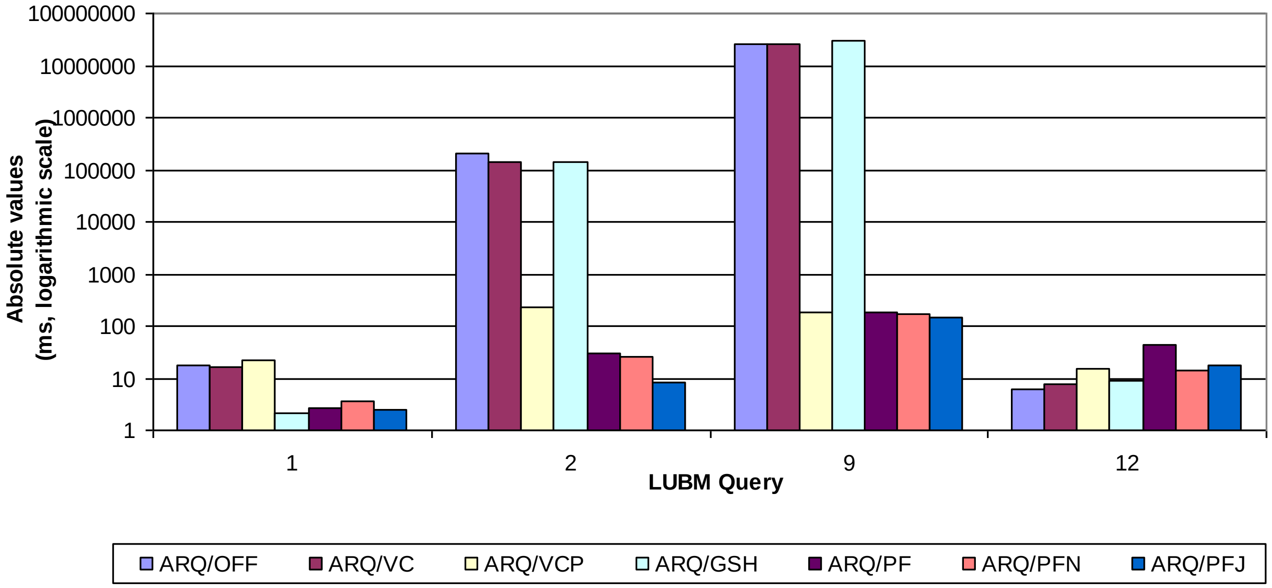
5 ?Z ub:subOrganizationOf ?Y .
6 ?X ub:undergraduateDegreeFrom ?Y .
3 ?Z rdf:type ub:Department .
2 ?Y rdf:type ub:University .
1 ?X rdf:type ub:GraduateStudent .
4 ?X ub:memberOf ?Z .



Abstracted
EP as DAG for the optimized BGP
(1 source nodes for all adjacent arcs, no Cartesian products)

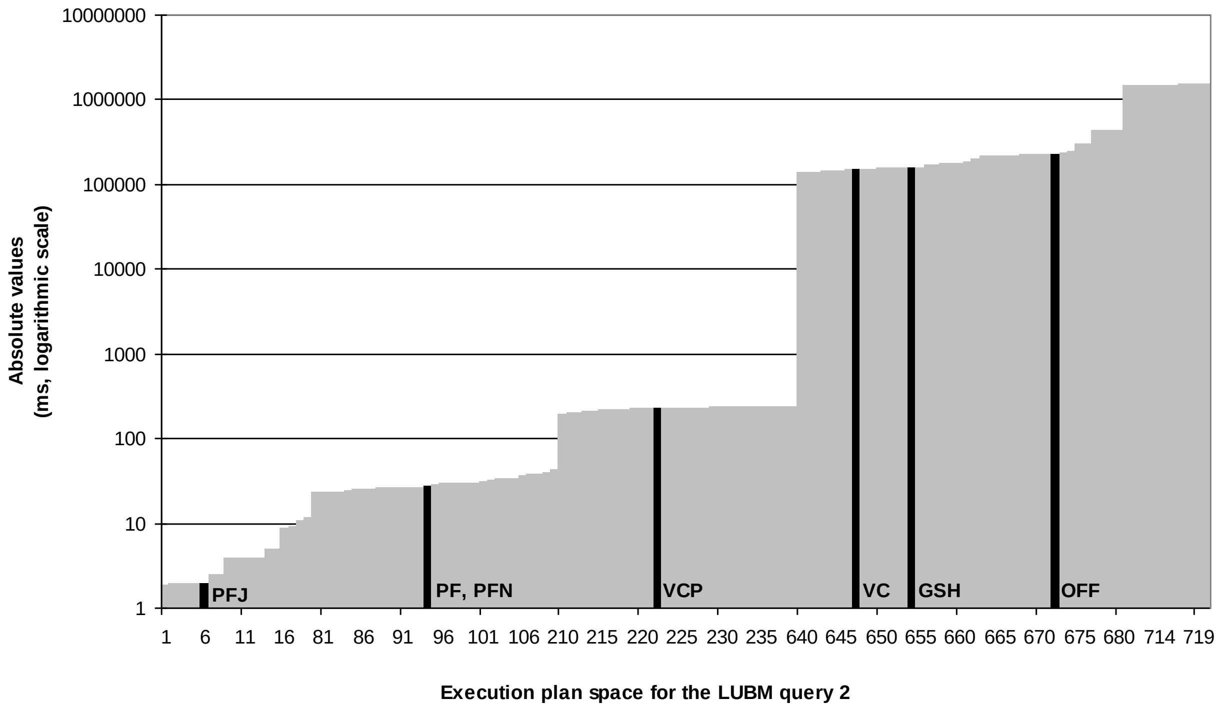
Evaluation I

- On Lehigh University Benchmark, one university and OWL-DL entailment, 156,407 triples
- AMD Opteron dual core with 8 GB main memory
- Query performance



Evaluation II

- Execution plan space for the LUBM query 2

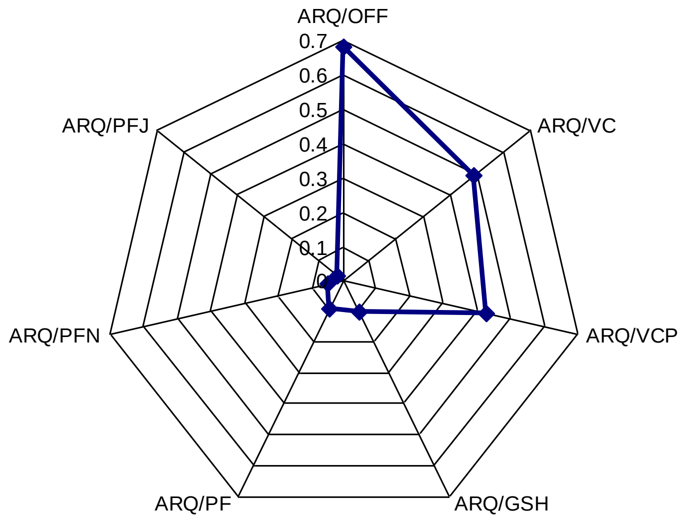


PFJ: #6
PF, PFN: #94
VCP: #222
VC: #647
GSH: #654
OFF: #672

Best: 1.87 ms
PFJ: 1.94 ms
Worst: 1,532,992.16 ms

Evaluation III

- The normalized distance from the best performing EP for each heuristic averaged over the 14 (-1) LUBM queries



OFF: 0.68
VC: 0.48
VCP: 0.42
GSH: 0.10
PF: 0.09
PFN: 0.04
PFJ: 0.02

Conclusions & Limitations

- Optimization of BGP for main memory graph implementations
- ARQ optimizer with heuristics
- Framework for the selectivity estimation of (joined) triple patterns
- Main memory has an important limitation: Scaling. Nevertheless, study of SPARQL optimization for native graph implementations is important for efficient query evaluation on the Semantic Web (will RDBMS technology survive the graph oriented Semantic Web?).

Future Work

- SPARQL syntax
  - FILTER
  - OPTIONAL
  - UNION
- Typed histograms
  - Selectivity of ?age <= 30
- Distributed in-memory graph models

Questions?

- Any questions?
- Thank you for your attention!