Ag

AvantGraph

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

1. A general ***cardinality estimation*** framework
2. Factorization matters in ***large graphs***
3. ***Temporal-clique*** subgraph query processing

LDBC TUC 2021

Nikolay Yakovets

Ag
Avant **Graph**

Os
Open Source

Mm
Main Memory

Ra
Recursive Analytics

Vc
Vectorized Compiled

Wco
Worst-case Optimal

Fz
Factorized

Tm
Temporal

Re
Reachability

Td
Topo+data

not your grandma's graph engine!

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

1 A General Cardinality Estimation Framework

Cardinality estimation for planning

- Subgraph-matching queries on property graphs
- Comprehensive overview of cardinality estimation techniques from **basic principles** (probability theory)

A General Cardinality Estimation Framework for Subgraph Matching in Property Graphs

Wilco van Leeuwen, George Fletcher, Nikolay Yakovets

Abstract—Many techniques have been developed for the cardinality estimation problem in data management systems. In this document, we introduce a framework for cardinality estimation of query patterns over property graph databases, which makes it possible to analyze, compare and combine different cardinality estimation approaches. This framework consists of three phases: obtaining a set of estimates for some subqueries, extending this set and finally combining the set into a single cardinality estimate for the query. We show that (parts of) many of the existing cardinality estimation approaches can be used as techniques in one of the phases from our framework. The three phases are loosely coupled, this makes it possible to combine (parts of) current cardinality estimation approaches. We create a graph version of the Join Order Benchmark to perform experiments with different combinations of techniques. The results show that query patterns *without* property constraints can be accurately estimated using synopses for small patterns. Accurate estimation of query patterns *with* property constraints require new estimation techniques to be developed that capture correlations between the property constraints and the topology in graph databases.

Index Terms—Cardinality estimation, selectivity estimation, query optimization, graph databases, property graph data model.

1 Introduction

Cardinality estimation can be defined as the task of estimating the number of results returned by a given query over a given database instance. This is a fundamental data management problem, important across many practical scenarios, e.g., in query planning, approximate query evaluation, and query execution progress estimation. The basic evaluation criteria for cardinality estimation techniques are: estimation accuracy, estimation time, memory cost, and preparation time. Different applications might value these dimensions differently, leading to trade-offs, e.g., between estimation accuracy and memory footprint. It has been shown that poor query plans are regularly produced by the query optimizers of current database management systems and that cardinality estimation errors are the main reason

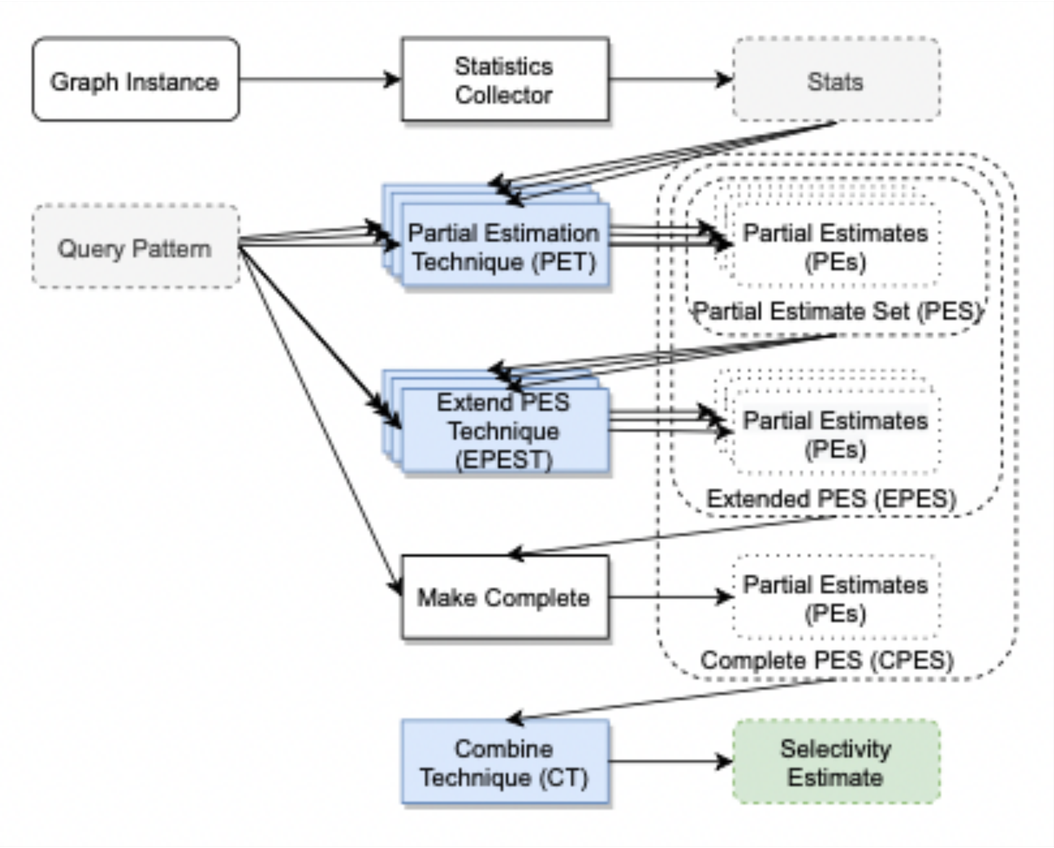
who became a member after 1990-12-08). Query Patterns are fundamental in both theory and practice, forming the backbone of virtually all queries expressed in contemporary graph database query languages [3], [4], [5]. Cardinality estimation for query patterns over property graph databases gives rise to new challenges due to the schemaless design of graph databases and the correlation between the data (property values and labels) and the topology (connectivity) of the graph. The focus of our survey allows us to shed deep light on an important, timely, highly challenging, and broadly applicable subclass of the cardinality estimation problem. Further, many of the concepts of our survey generalize to other typical settings, for example conjunctive queries on relational databases.

1 A General Cardinality Estimation Framework

Query Pattern Constraints						
Query Pattern Schema Constraints						Property-value Constraints
Labeled Topological Constraints					Property Constraints	
Topological Constraints				Data Constraints		
`vertex(`i`)`	`edge(`i`)`	`src(`v`,`e`)`	`trg(`v`,`e`)`	`hasLabel(`i`,`l`)`	`hasPropKey(`i`,`k`)`	`hasProp(`i`,`k`,`θ`,`v`)`

Overarching framework for cardinality estimation of topology+data predicates

- Statistics collection
- Partial estimation
- Extending partial estimates (assumptions)
- Completing partial estimates
- Combining complete partial estimates to produce the final estimate



1

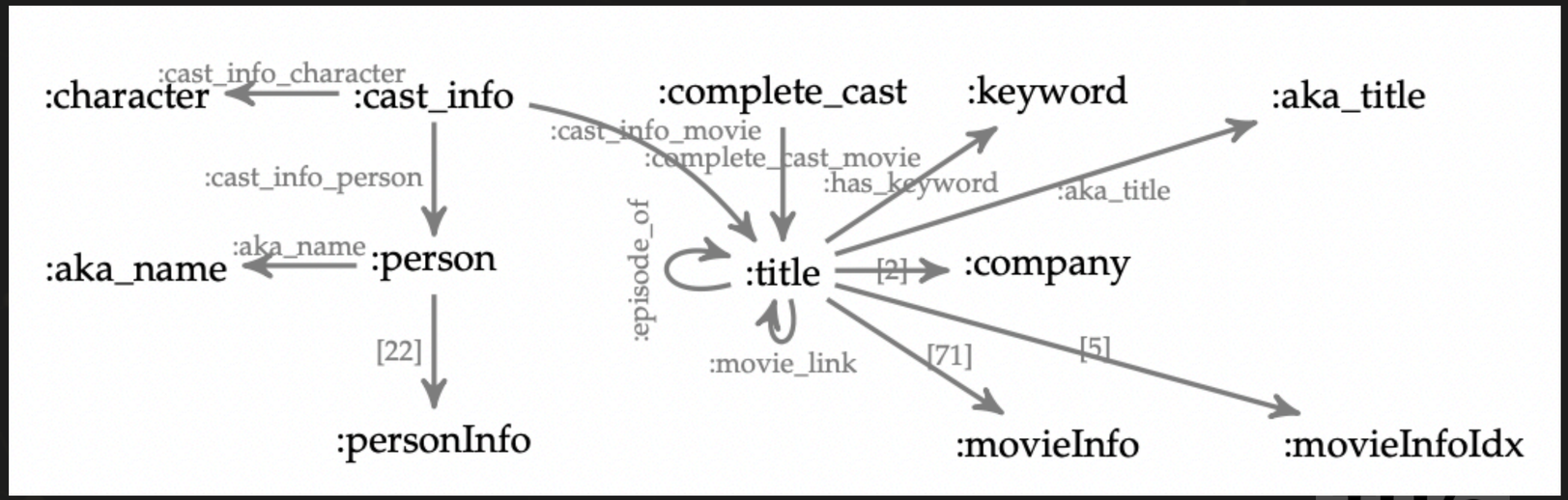
A General Cardinality Estimation Framework

Extensive benchmarking of SOTA

- Graph Join-Order Benchmark (G-JOB)
- Compare/contrast/combine of SOTA techniques to produce superior results
- Identify weaknesses and largest error contributors

Pre-print out now on arXiv!

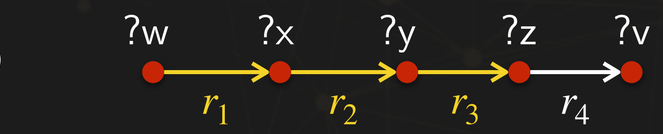
G-JOB



TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

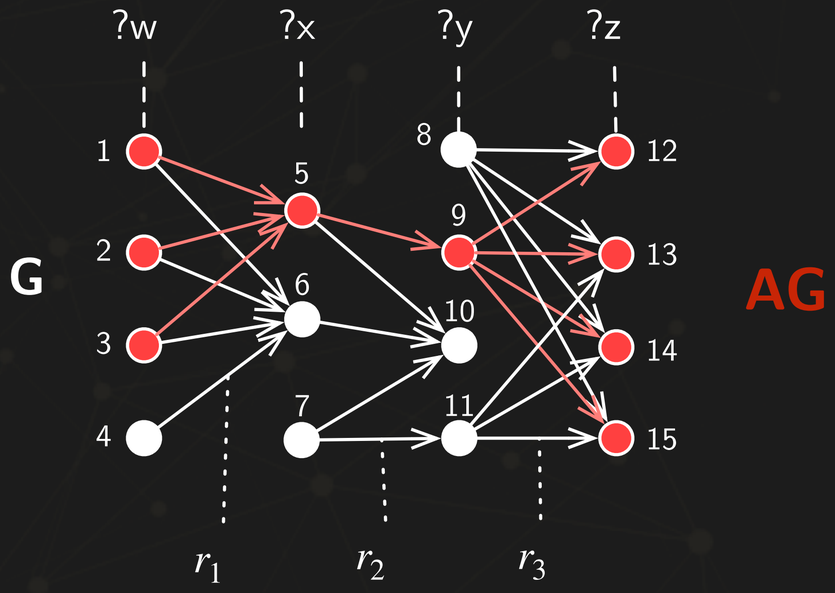
2

AnswerGraph: approach



IR

{1,5,9,13}	{1,5,9,14}	{1,5,9,15}
{2,5,9,13}	{2,5,9,14}	{2,5,9,15}
{3,5,9,13}	{3,5,9,14}	{3,5,9,15}
{1,5,9,12}	{2,5,9,12}	{3,5,9,12}



AG

Factorization: One way of reducing the size of intermediate results is to apply the concept of *factorization*

- the common unique node-pair patterns
- we call these factorized node-pairs, **Answer Graph (AG)**

Defactorization: To find all final result tuples (i.e., embeddings), all we need to do is to *defactorize* this answer graph

EDBT'21

Answer Graph: Factorization Matters in Large Graphs

Zahid Abul-Basher
University of Toronto
Toronto, Canada
zahid@mie.utoronto.ca

Nikolay Yakovets
Eindhoven University of Technology
Eindhoven, The Netherlands
hush@tue.nl

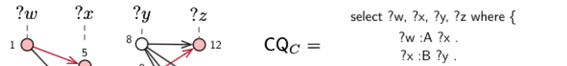
Parke Godfrey
York University
Toronto, Canada
godfrey@yorku.ca

Stanley Clark
Eindhoven University of Technology
Eindhoven, The Netherlands
s.clark@tue.nl

Mark Chignell
University of Toronto
Toronto, Canada
chignell@mie.utoronto.ca

ABSTRACT

Our *answer-graph method* to evaluate SPARQL *conjunctive queries*

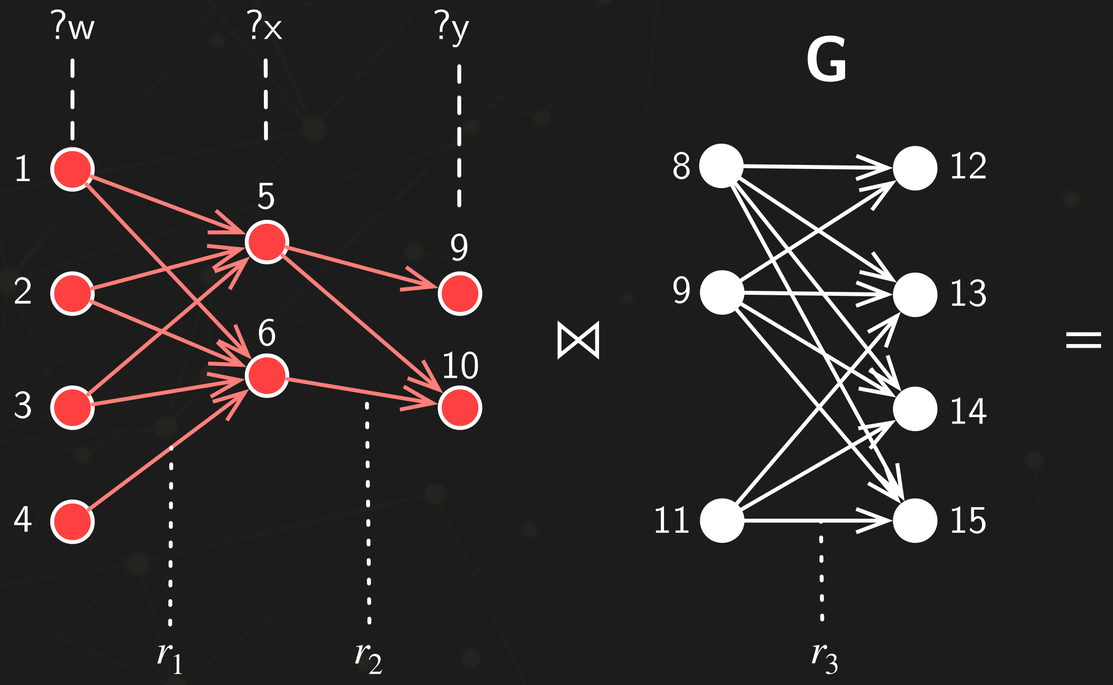


2

AnswerGraph: evaluation model

Answer graph generation

- Edge extension: to fetch data edges from the graph
- Node burn-back: to ensure the generated AG is minimal (in size)



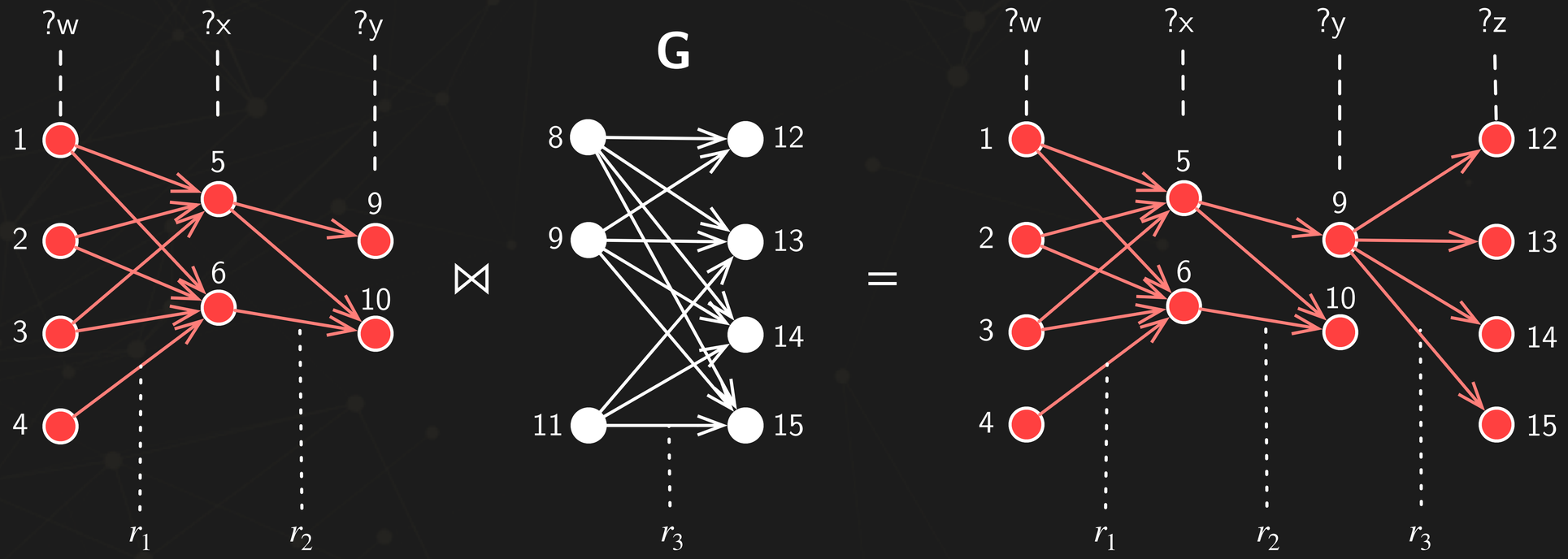
edge extension

2

AnswerGraph: evaluation model

Answer graph generation

- Edge extension: to fetch data edges from the graph
- Node burn-back: to ensure the generated AG is minimal (in size)



edge extension

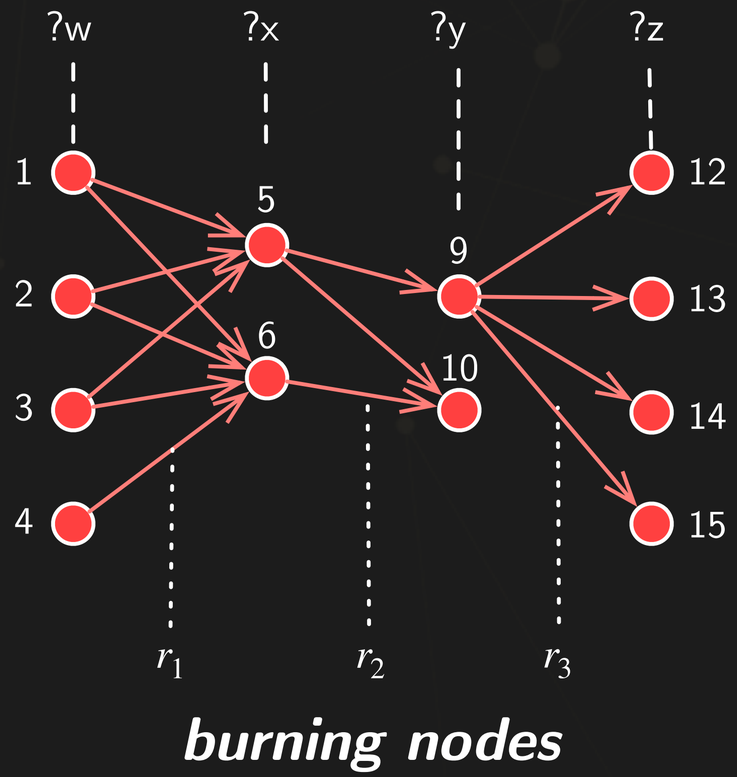
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

2

AnswerGraph: evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges



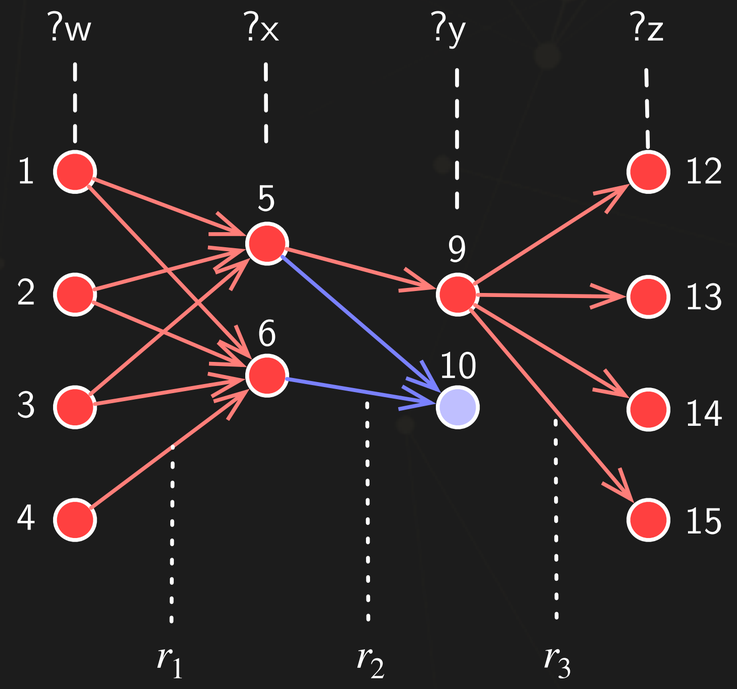
EINDHOVEN UNIVERSITY OF TECHNOLOGY

2

AnswerGraph: evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges



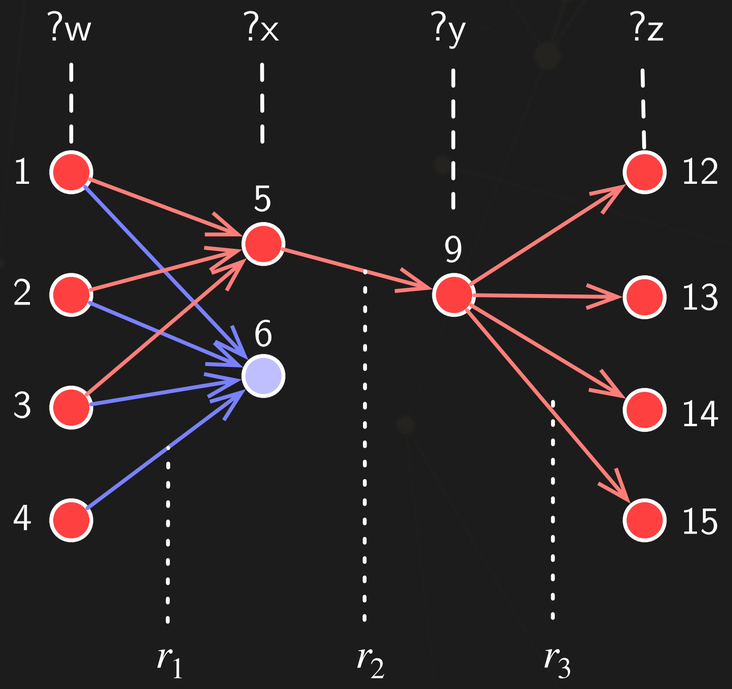
burning nodes - step 1

2

AnswerGraph: evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges



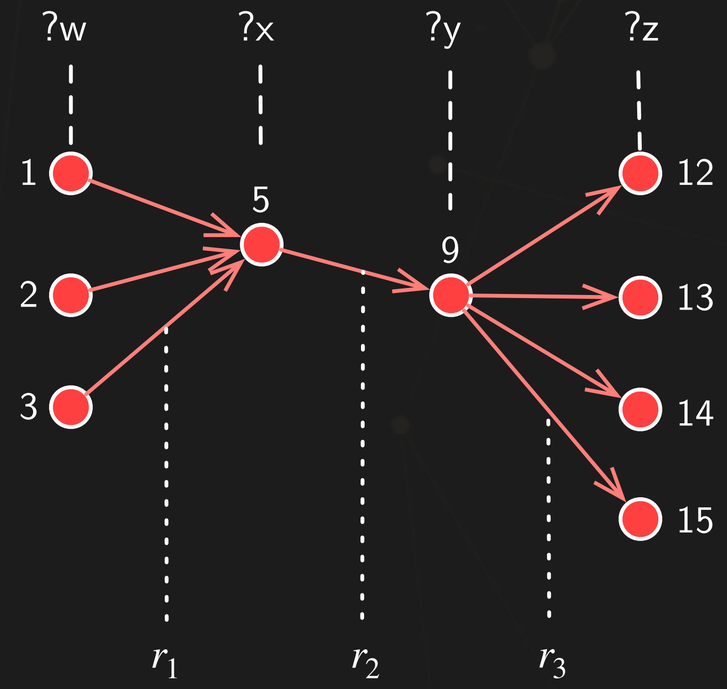
burning nodes - step 2

2

AnswerGraph: evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges

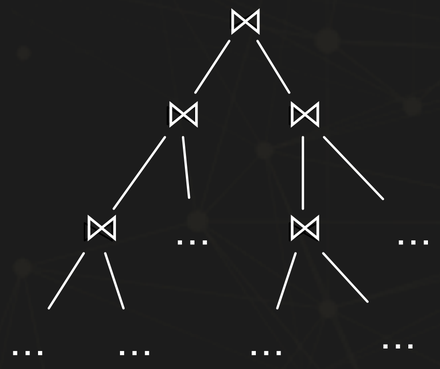


done burning nodes

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

2

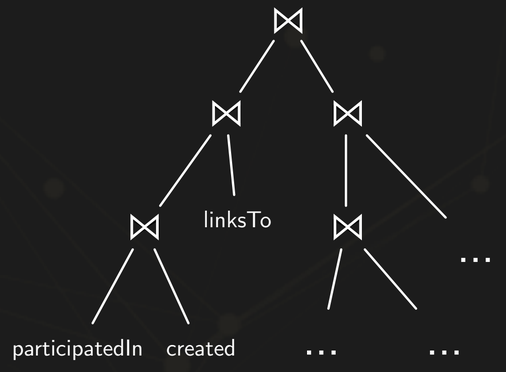
WireFrame: the framework



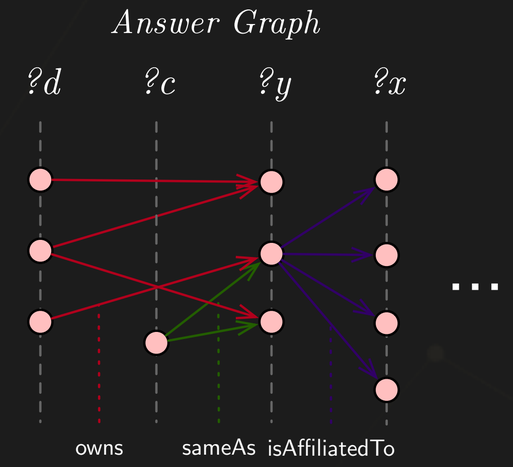
Query plan

?d	?c	?y	?a	?x	?z	?m	?b	?f	?e
60	13	2	10	4	11	22	23	24	14
60	13	2	10	6	11	22	23	24	14
60	13	2	10	15	11	22	23	24	14
..	..	..	..	..	..	..	..	..	..

Embeddings

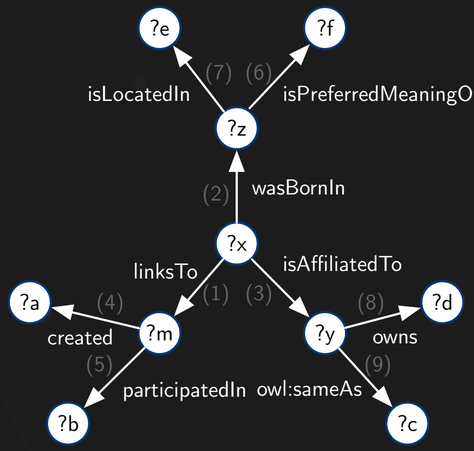


Embedding plan

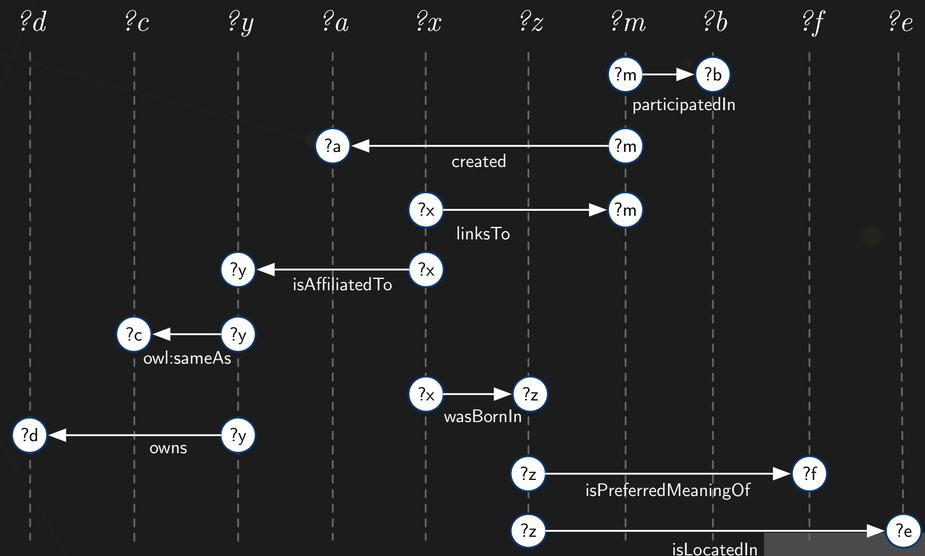



```
select distinct ?x, ?m, ?y, ?z, ?a
?b, ?c, ?d, ?e, ?f
where {
?x linksTo ?m .
?x isAffiliatedTo ?y .
?x wasBornIn ?z .
?m participatedIn ?b .
?m created ?a .
?y owl:sameAs ?c .
?y owns ?d .
?z isLocatedIn ?e .
?z isPreferredMeaningOf ?f . }
```

Conjunctive query



Query graph



Answer graph plan

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

2

AnswerGraph: Ideal answer graph

We call an answer graph ideal if it contains only those edges which participate in at least one final embedding.

Theorem: Node burn-back results in an ideal AG for **acyclic** graph CQs with number of cascaded semi-joins bounded in $O(|Q|)$.

Pf.:

- node burn-back results in an ordered sequence of semi-joins which contain (in correct order) a semi-join ordering produced by the GYO algorithm
- this corresponds to a bounded full reducer semi-join program which guarantees no dangling tuples in the AG for acyclic CQs

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

2

AnswerGraph: Ideal answer graph

We call an answer graph ideal if it contains only those edges which participate in at least one final embedding.

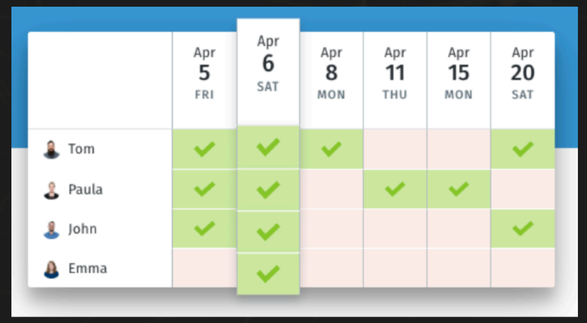
We can handle queries with higher treewidth graphs, but this requires more materialization to produce the ideal AG or using fix-point cascade

- Ultimately, this is a cost-based decision whether the extra materialization is worth the effort
- 99% of queries in practice are near acyclic

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

3

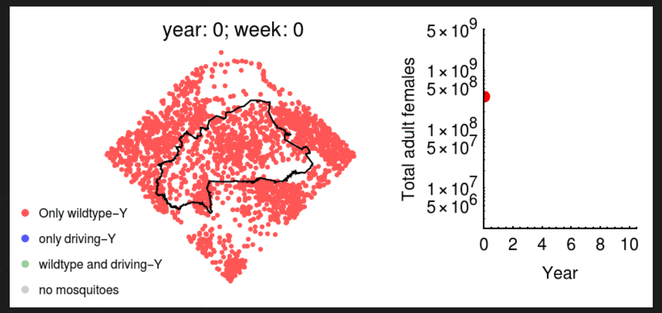
Temporal processing in AG



scheduling

collaboration

temporal network analytics



disease modelling

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

3

Leapfrog TSR-join: approach

Topological (T) + temporal (P) predicates often occur together in practical queries

- However, they are often processed in a "rigid" order: T then P, P then T
- Thus, T&P selectivities are not fully leveraged leading to slow query processing times

ICDE'21

Leveraging Temporal and Topological Selectivities in Temporal-clique Subgraph Query Processing

Kaijie Zhu[⋆,†], George Fletcher[⋆], Nikolay Yakovets[⋆]
[⋆]Eindhoven University of Technology, The Netherlands
[†]NDSC, Zhengzhou, China
{k.zhu, g.h.l.fletcher, hush}@tue.nl

***Abstract*—We study the problem of temporal-clique subgraph pattern matching. In such patterns, edges are required to jointly overlap in time within a given temporal window in addition to forming a topological sub-structure. This problem arises in many application domains, e.g., in social networks, life sciences, smart cities, telecommunications, and others. State-of-the-art subgraph matching techniques, however, are shown to be limited and inefficient in processing queries with both temporal and topological constraints. We propose an approach that takes full advantage of both topological and temporal selectivities during**

traditional "cliques"' in which nodes are *tightly interconnected in topology*. This basic problem arises in a wide range of applications beyond the transportation domain.

- In a social network where vertices represent users and edges represent the 'following' relationship, find, in the first week of August 2020, all pairs of users who at the same time followed at least three other users in common.
- For malicious network attack detection, where vertices

3

Leapfrog TSR-join: approach

Topological (T) + temporal (P) predicates often occur together in practical queries

- However, they are often processed in a "rigid" order: T then P, P then T
- Thus, T&P selectivities are not fully leveraged leading to slow query processing times

T&P approach:

- Solving **temporal-clique** subgraph queries
- Leapfrog TSR-join is a T&P version of leapfrog trie join
  - Extending tries to TSR index (trie representation)
  - Adapting LFTJ to process over the TSR index (binding production + propagation)
- Further optimizations

ICDE'21

Leveraging Temporal and Topological Selectivities in Temporal-clique Subgraph Query Processing

Kaijie Zhu[⋆,†], George Fletcher[⋆], Nikolay Yakovets[⋆]
[⋆]Eindhoven University of Technology, The Netherlands
[†]NDSC, Zhengzhou, China
{k.zhu, g.h.l.fletcher, hush}@tue.nl

***Abstract*—We study the problem of temporal-clique subgraph pattern matching. In such patterns, edges are required to jointly overlap in time within a given temporal window in addition to forming a topological sub-structure. This problem arises in many application domains, e.g., in social networks, life sciences, smart cities, telecommunications, and others. State-of-the-art subgraph matching techniques, however, are shown to be limited and inefficient in processing queries with both temporal and topological constraints. We propose an approach that takes full advantage of both topological and temporal selectivities during**

traditional "cliques"' in which nodes are *tightly interconnected in topology*. This basic problem arises in a wide range of applications beyond the transportation domain.

- In a social network where vertices represent users and edges represent the 'following' relationship, find, in the first week of August 2020, all pairs of users who at the same time followed at least three other users in common.
- For malicious network attack detection, where vertices

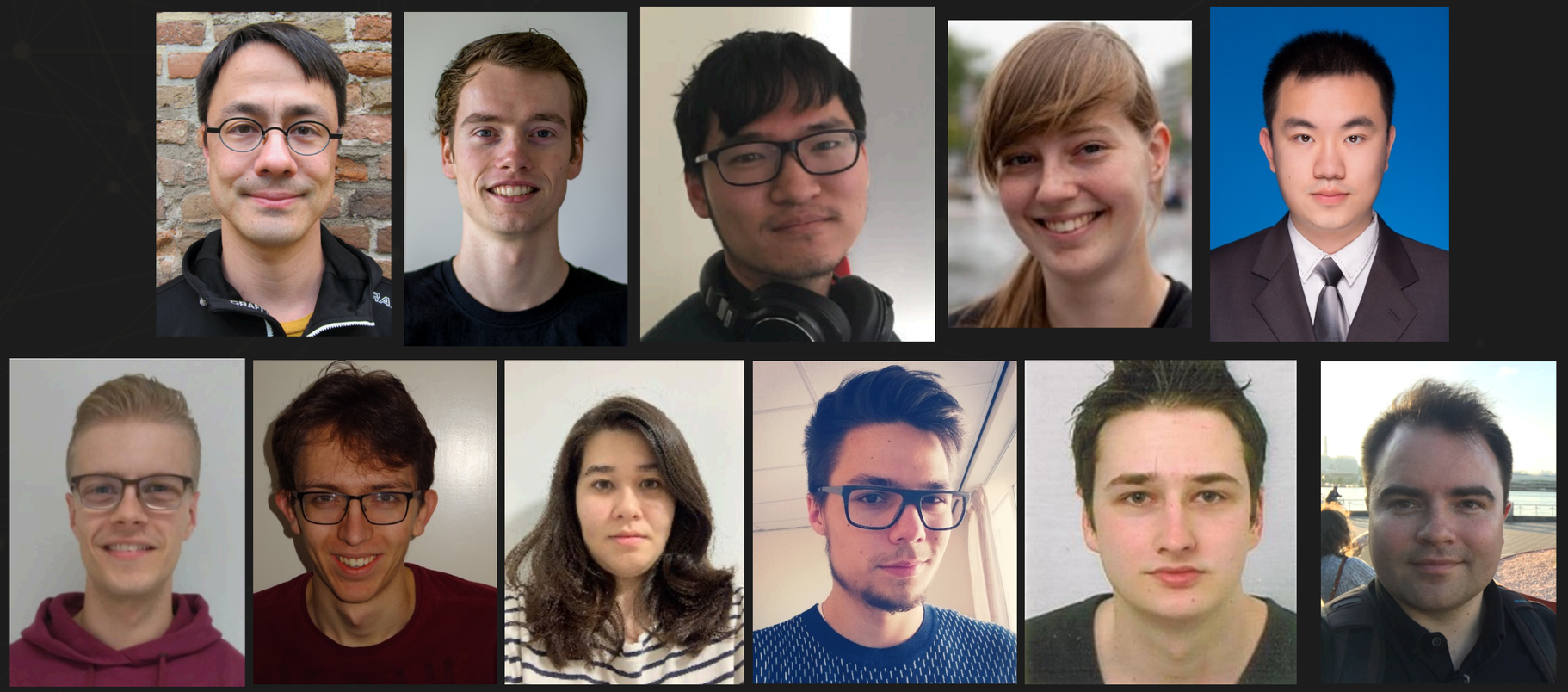
AvantGraph: the frontier of graph query processing

- Fast, near in-memory-performance processing of graph queries on external storage (Luo, TUe 2021)
- Scaling-up processing of recursive and worst-case optimal queries (van de Looij, TUe 2021)
- Lightweight indexing for worst-case optimal joins (Wortel, TUe 2021)
- Compiled and vectorized query processing (van de Wall, TUe 2020)
- Hybrid worst-case optimal join processing (de Brouwer, TUe 2020)
- ...

TU/e
EINDHOVEN
UNIVERSITY OF
TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Thank you!



...and other contributors!

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY