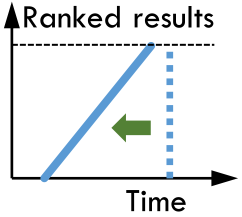
Beyond Equi-joins: Ranking, Enumeration and Factorization

Nikolaos Tziavelis, Wolfgang Gatterbauer, Mirek Riedewald

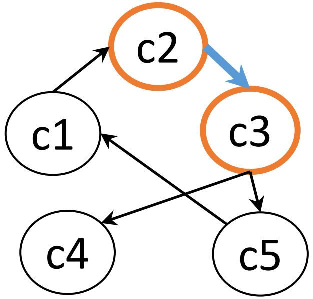
Northeastern University, Boston

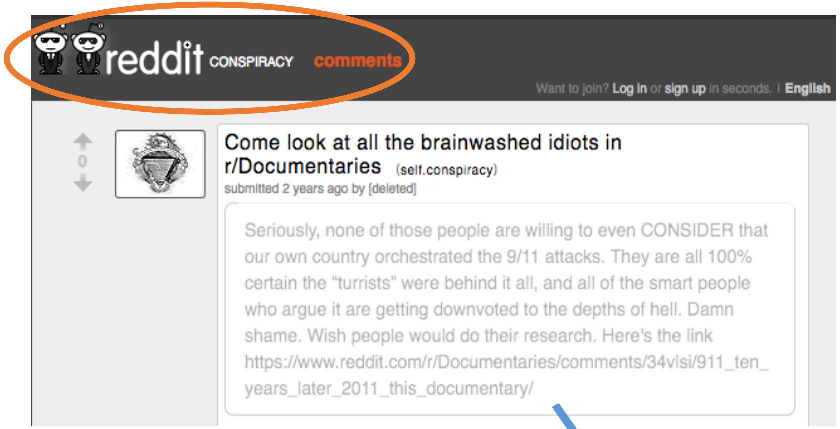


Website: https://northeastern-datalab.github.io/anyk/
Data Lab: https://db.khoury.northeastern.edu

Motivating Example

Reddit Network

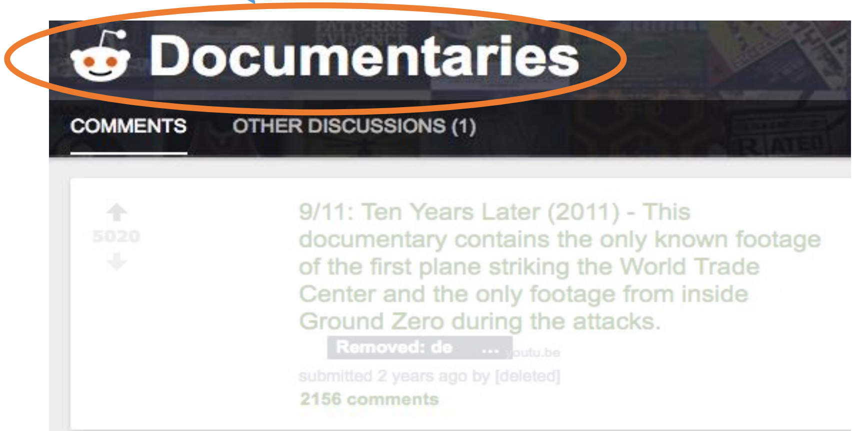



reddit CONSPIRACY comments
Want to join? Log in or sign up in seconds. | English
0

Come look at all the brainwashed idiots in r/Documentaries (self.conspiracy)

submitted 2 years ago by [deleted]

Seriously, none of those people are willing to even CONSIDER that our own country orchestrated the 9/11 attacks. They are all 100% certain the "turrists" were behind it all, and all of the smart people who argue it are getting downvoted to the depths of hell. Damn shame. Wish people would do their research. Here's the link https://www.reddit.com/r/Documentaries/comments/34vlsi/911_ten_years_later_2011_this_documentary/



9/11: Ten Years Later (2011) - This documentary contains the only known footage of the first plane striking the World Trade Center and the only footage from inside Ground Zero during the attacks.

submitted 2 years ago by [deleted]

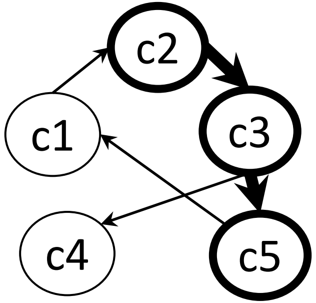
2156 comments

- Timestamp
- Sentiment measure
- Readability score

[K+ 18] Kumar, Hamilton, Leskovec, Jurafsky. Community Interaction and Conflict on the Web. WWW'18. https://doi.org/10.1145/3178876.3186141

Motivating Example

Reddit Network



(Timestamp1, Sentiment1)

$\downarrow$ $<$ $\downarrow$ $\quad$ $\downarrow$ $>$ $\downarrow$

(Timestamp2, Sentiment2)

Join in SQL:

```
select *, R1.Readability + R2.Readability as weight
from Reddit R1, Reddit R2
where  R2.Source = R1.Target
        AND R2.Timestamp > R1.Timestamp
        AND R2.Sentiment < R1.Sentiment
order by weight desc
limit 1000
```

Equality

Inequalities

Ranking

Q: - length-2 paths

- timestamps in increasing order
- sentiment in decreasing order
- top results by sum of readability

Naïve plan:

1. Compute all $O(n^2)$ join results
2. Sort them

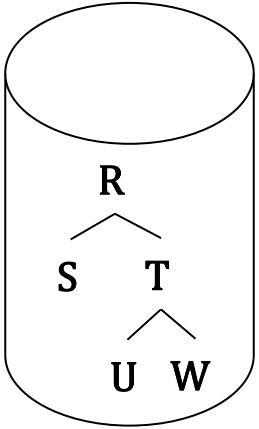
Our approach: $\boldsymbol{O(n+k)}$ for $\boldsymbol{k}$ results
(ignoring log factors)

Ranked Enumeration for Join Queries

Generalizes top-k

DB

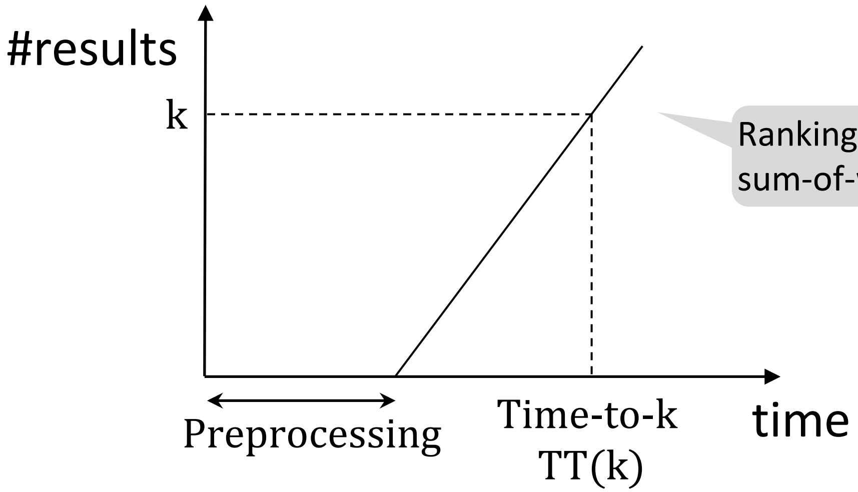
n tuples



Join

ℓ relations

Ranked Enumeration ("Any-k")



Ranking function: sum-of-weights

Assumptions

- Data complexity (ℓ, #attributes constant)
- Indexes have to be built on-the-fly
- In-memory computation

Lower bound: $\text{TT}(k) = O(n + k)$

Equi-joins [T+20]: $\text{TT}(k) = O(n + k \log k)$ (full acyclic)

This work: $\text{TT}(k) = O(n \,\text{polylog}\, n + k \log k)$

for full acyclic queries, and

DNFs of equality/inequality predicates

[T+20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB'20
https://doi.org/10.14778/3397230.3397250

Overview of our Approach

1. Preprocessing Phase

Factorized Representation

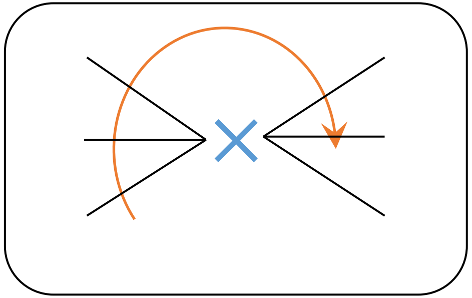
Compact representation of all query results

Size of full output: $O(n^{\ell})$

↓

Size of representation: $O(n \text{ polylog } n)$

n = #tuples
ℓ = #relations



2. Enumeration Phase

Any-k Algorithms

Traverse the representation, **prioritizing** solutions according to ranking

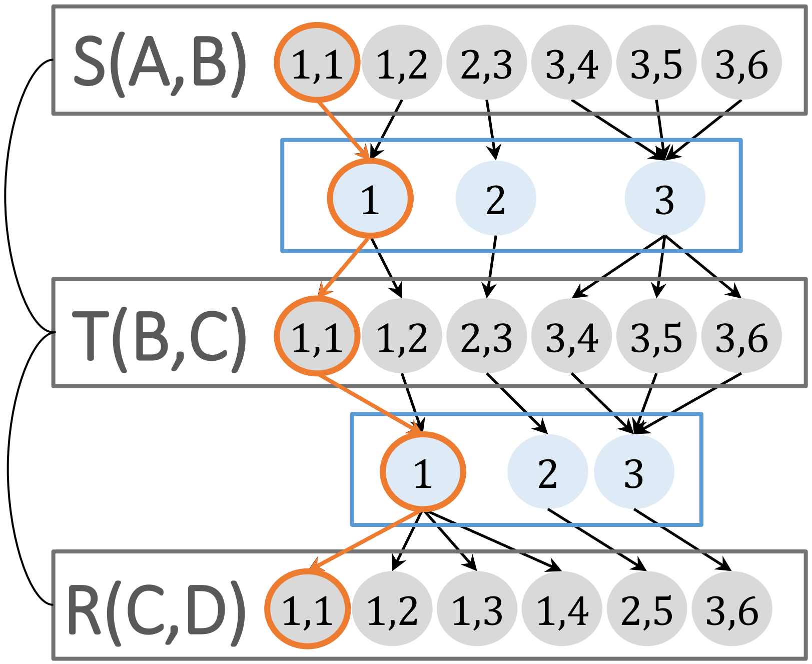
Same framework as equi-join case

Outline

- Ranked Enumeration Problem
- Equi-join Case [Prior Work]
- Handling Theta-joins
- Experiments
- Conclusion

Factorized Representation for (Acyclic) Equi-joins

Enumeration Graph



relation layers (tuples)

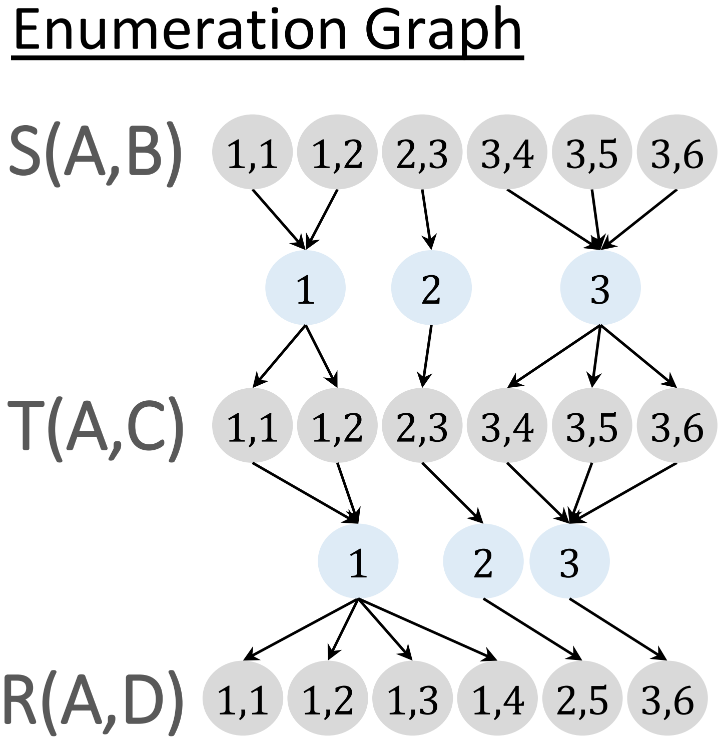
connection layers (attribute values)

path = join result

$O(n)$ time and space (via hashing)
but encodes all join results

[T+20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB'20
https://doi.org/10.14778/3397230.3397250

Enumeration Phase for (Acyclic) Equi-joins



Top-down

- k-shortest paths (or “k-lightest subtrees”)
- Traversal with Priority Queues

Bottom-up

- Dynamic Programming
- Propagate minimum weight up

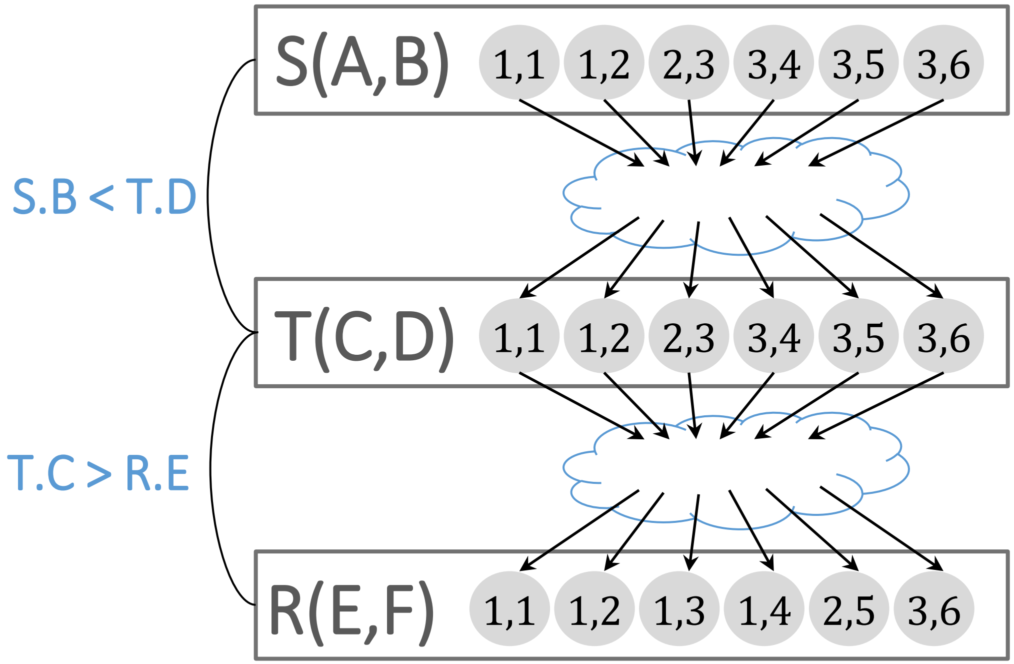
[T+20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB’20
https://doi.org/10.14778/3397230.3397250

Outline

- Ranked Enumeration Problem
- Equi-join Case [Prior Work]
- Handling Theta-joins
- Experiments
- Conclusion

Factorized Representation for (Acyclic) Theta-Joins

Enumeration Graph

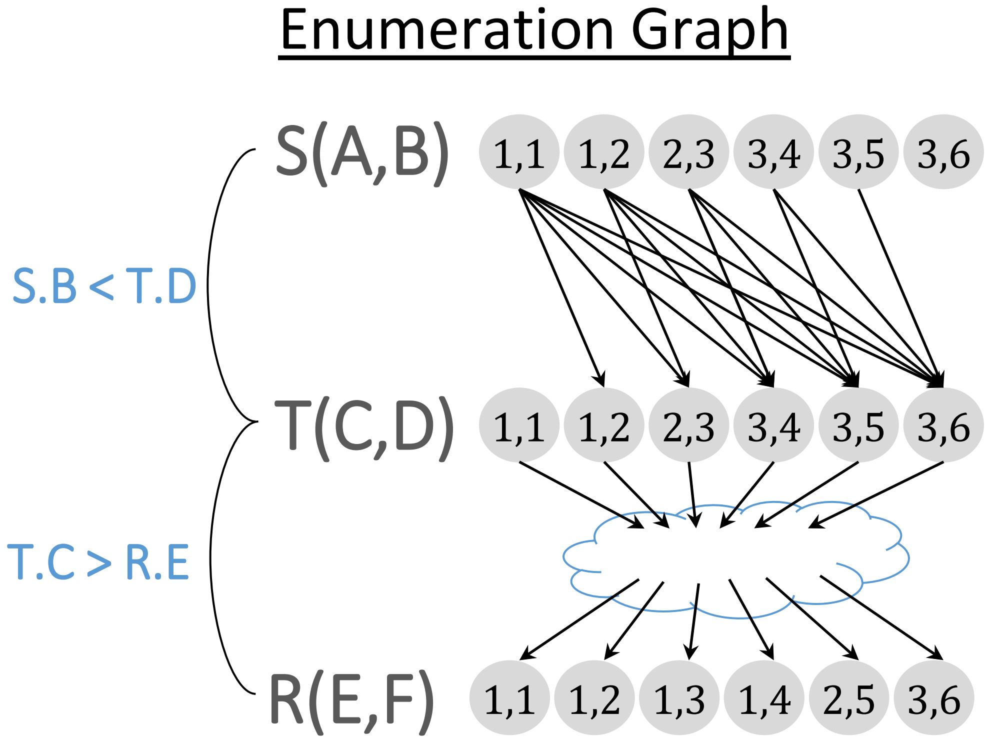


relation layers (tuples)

Tuple-Level Factorization Graph (TLFG)

- DAG between 2 relation layers
- Path from S tuple to T tuple
  ⇔
  Valid join pair
- Ranked enumeration for any TLFG
  - Size affects preprocessing time
  - Depth (longest path) affects delay

Direct TLFGs



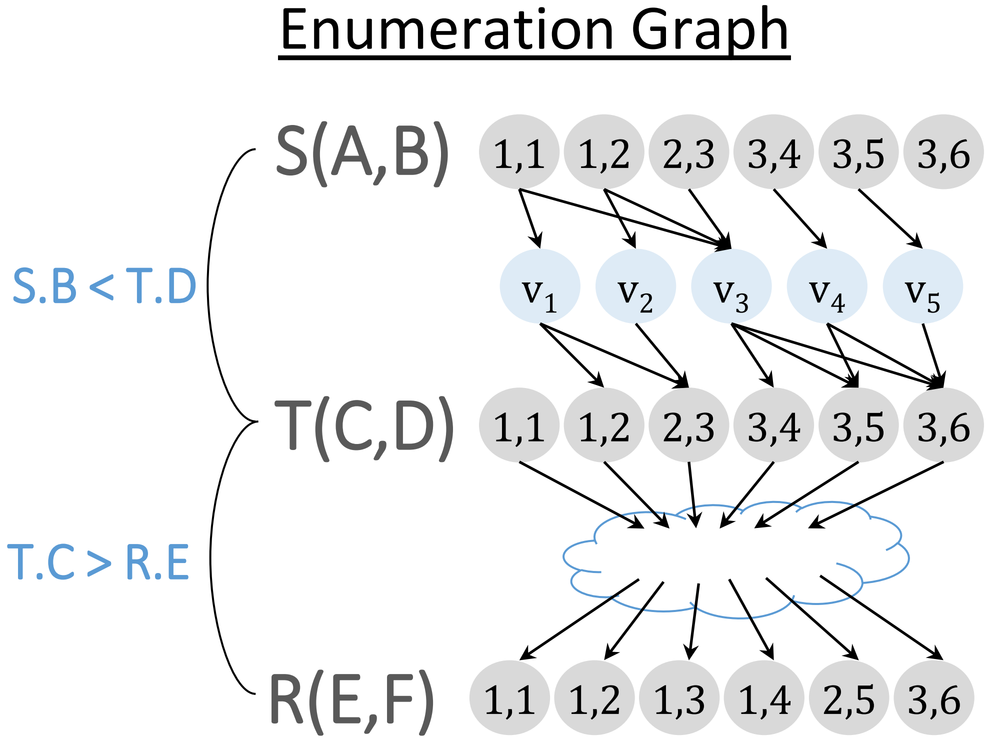
Direct TLFG

- $O(n^2)$ edges
- Depth = 1
- Works for any join condition

$$\mathrm{TT}(k) = O(n^2 + k \log k)$$

☹

Binary Partitioning for Inequality Predicate



Binary Partitioning Method

- $O(n \log n)$ size
- Depth = 2
- For 1 inequality predicate

$$\mathrm{TT}(k) = O(n \log n + k \log k)$$ ☺

$\bowtie_<$ → $\bowtie_=$

$O(n)$-size relations

$O(n \log n)$-size relations

Generalization and Extensions

- Band predicates ($|S.A - T.B| < \varepsilon$)
- Non-equality predicates ($S.A \neq T.B$)
- Conjunctions/Disjunctions of predicates

- Cyclic joins
  - Higher complexity

- Memory consumption analysis
- Optimizations for improved memory consumption

Outline

- Ranked Enumeration Problem
- Equi-join Case [Prior Work]
- Handling Theta-joins
- Experiments
- Conclusion

Experimental Setup

METHOD	DETAILS
Factorized	• Our method
QuadEqui	• Direct TLFG (materializes auxiliary relations of size $O(n^2)$ to reduce theta-join to equi-join) • Uses ranked enumeration for equi-joins • Time measured after materialization
Batch	• Time to rank all results with a Priority Queue • Time for join **not** measured
PSQL	• Prebuilt indexes • Limit clause
System X	• Commercial DBMS • In-memory optimized

Thus, lower bound on any implementation!

DBMSs

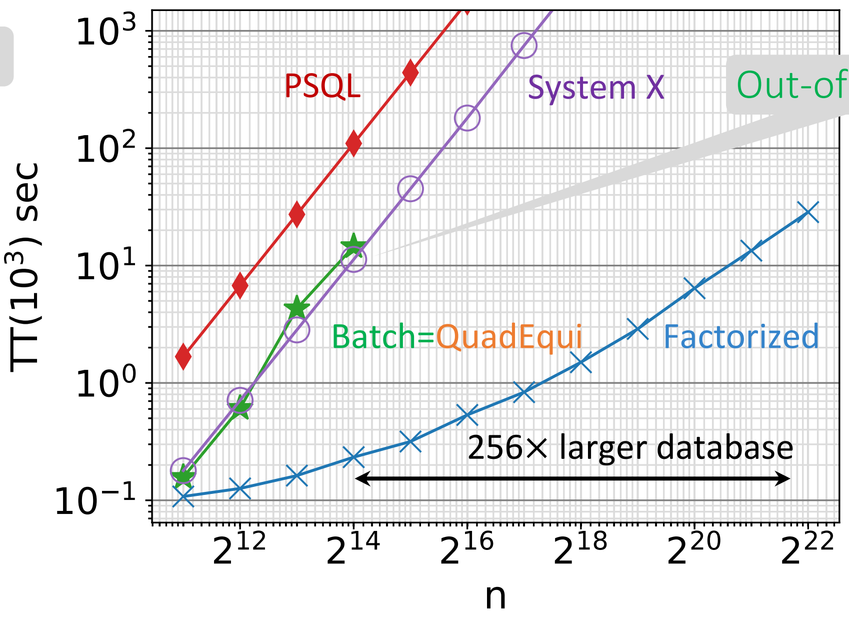
Exp1: Synthetic Data

$S_i(A_i, A_{i+1}, W)$

- Tuples values drawn randomly from integer domain
- Binary join with one inequality predicate

```
select *, S1.W + S2.W as weight
from S1, S2
where S1.A2 < S2.A3
order by weight asc
```

Top-1000



Out-of-Memory

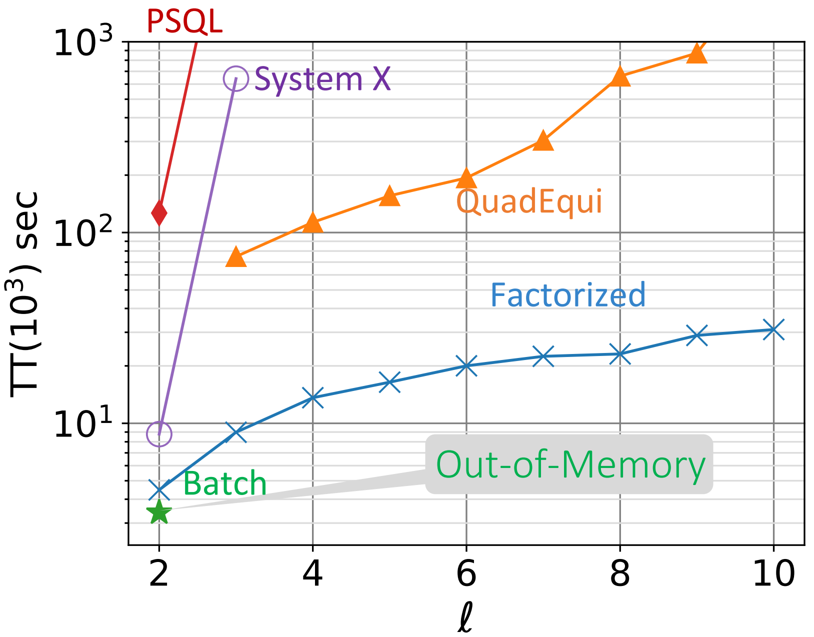
Other methods face memory problems as n increases.

n = relation size

Exp2: Paths on Reddit

Q: - length-ℓ paths ~286k edges

- timestamps in increasing order
- sentiment in decreasing order
- top results by sum of readability



```
select *
from Reddit R1, Reddit R2
where  R2.Source = R1.Target
           AND R2.Timestamp > R1.Timestamp
           AND R2.Sentiment < R1.Sentiment
order by weight desc
```

Our method is robust to different query sizes and complicated join conditions.

ℓ = #relations

Outline

- Ranked Enumeration Problem
- Equi-join Case [Prior Work]
- Handling Theta-joins
- Experiments
- Conclusion

Conclusion

- DBMSs typically struggle with complex join predicates like inequalities.

- The top join results (e.g. top-1000) can be retrieved in time comparable to sorting the input

Even for $O(n^\ell)$ join results!

- For (full) acyclic queries with DNFs of equalities and inequalities:

$$\mathrm{TT}(k) = O(n \operatorname{polylog} n + k \log k)$$

Thank you!

Website: https://northeastern-datalab.github.io/anyk/
Code available online!