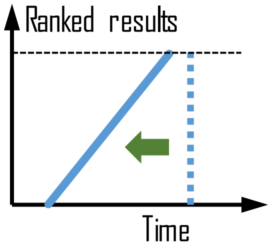
# Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries

Nikolaos Tziavelis[1], Deepak Ajwani[2], Wolfgang Gatterbauer[1], Mirek Riedewald[1], Xiaofeng Yang[3]

[1]Northeastern University, Boston, [2]University College Dublin, [3]VMWare




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

# Ranked Enumeration Example

join (R.B = S.B), join (S.C = T.C)

**R**

| A | B | w |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 0 | 2 |
| 3 | 1 | 3 |
| 4 | 3 | 4 |

**S**

| B | C | w |
|---|---|---|
| 0 | 1 | 5 |
| 0 | 2 | 7 |
| 1 | 1 | 8 |
| 1 | 2 | 9 |

**T**

| C | D | w |
|---|---|---|
| 1 | 1 | 20 |
| 1 | 2 | 40 |
| 2 | 3 | 10 |
| 2 | 4 | 30 |

Weights

```
select    A, B, C, D,
          R.w + S.w + T.w as weight
from      R, S, T
where     R.B=S.B and S.C=T.C
order by weight ASC
limit k   any-k
```

(~~limit k~~ any-k)

Enumerate results in order

Rank-1: (1, 0, 2, 3, 18) ⇨ Rank-2: (2, 0, 2, 3, 19) ⇨ Rank-3: (3, 1, 2, 3, 22) ⇨ …

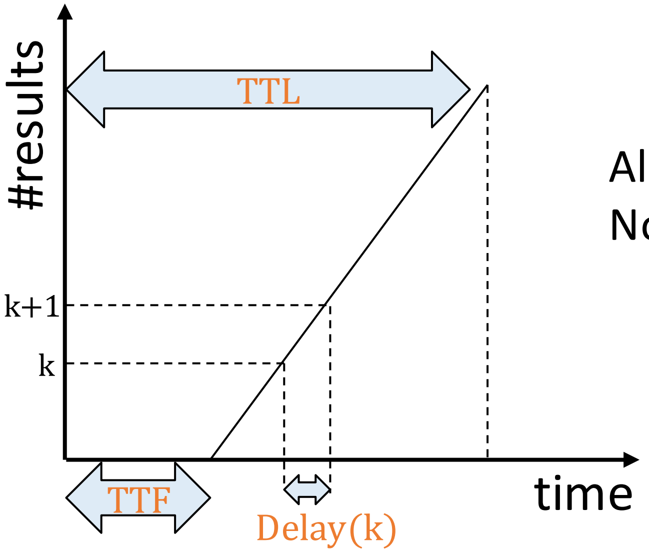
SUM of weights

# Ranked Enumeration: Problem Definition

**"Any-k"**

Anytime algorithms + Top-k for Conjunctive Queries

Most important results first (ranking function on output tuples, e.g. sum of weights)



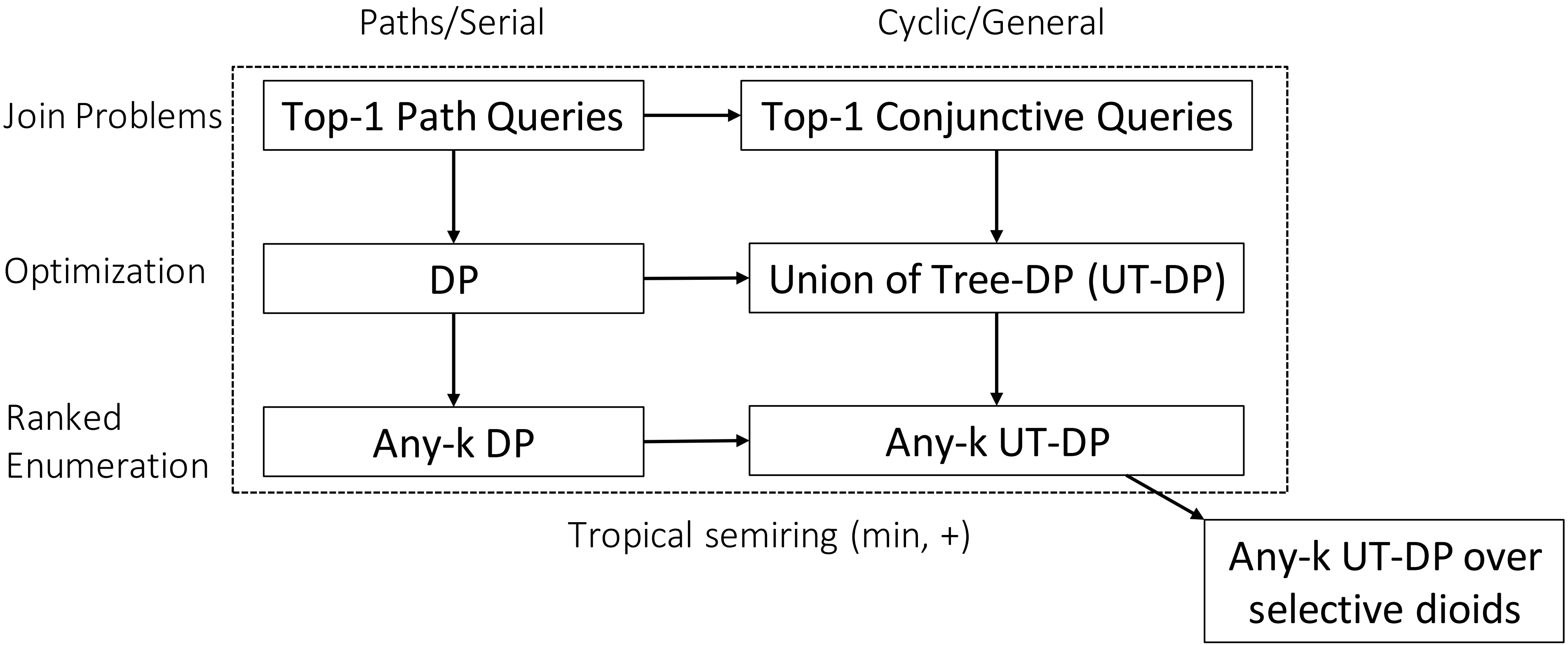

All results eventually returned

No need to set *k* in advance

RAM Cost Model:

- TTF = Time-to-First = TT(1)
- Delay(k) = Time between Rank-k and Rank-(k+1)
- TTL = Time-to-Last = TT(|out|)

# Conceptual Roadmap



# Main Result

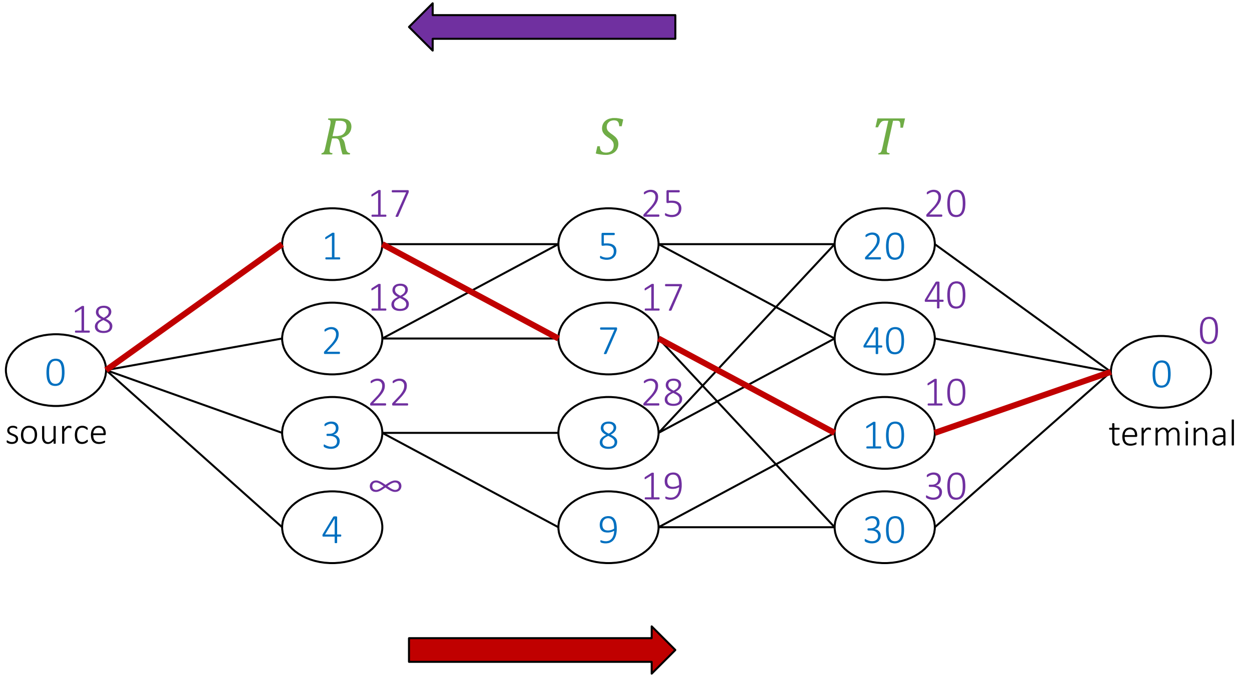
$n$: database size
Query = fixed size

- For Acyclic Queries:
  - $\text{TTF} = O(n)$
  - Delay($k$) = $O(logk)$
  - We get $k$ results (sorted) in just $O(n + k \log k)$ for any $k$!

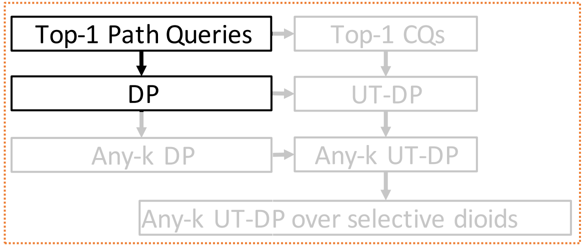
- For Cyclic Queries:
  - Higher TTF, according to best tree decomposition(s) available
  - Inherent cost of cyclicity

# Top-1: Dynamic Programming

Bottom-up




Top-down for Top-1 result




Nodes: tuples

Edges: joining pairs

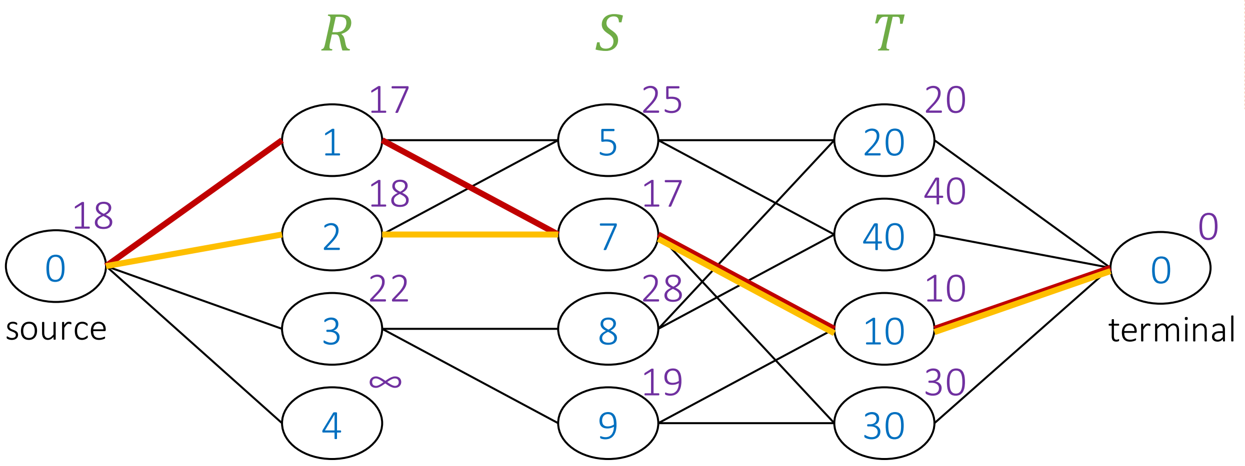
Labels: tuple weights

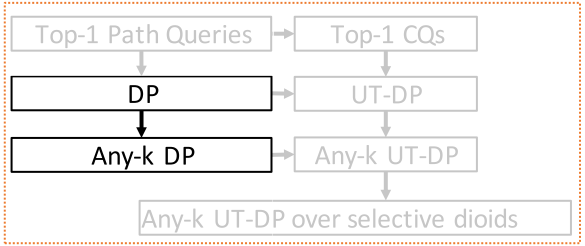
Bottom-up values: min total weight

Paths: join results

# Any-k DP: k-shortest paths

2nd Best Result = 2nd Shortest Path (19)




Best result = Shortest Path (18)

# Any-k DP Algorithms: 2 non-dominated families

## Anyk-Part

Repeatedly partitions the solution space.
Relies on [Lawler MS'76]

Wins when k is small.

$n$: database size
$l$: query size

<u>Variants</u>

- Eager
- All [Yang+ WWW'18]
- Lazy [Chang+ VLDB'15]
- **Take2** →

$$\text{TTF} = O(ln)$$
$$\text{Delay(k)} = O(logk + l)$$

Lowest delay given linear-time pre-processing!

## Anyk-Rec

Recursively computes lower-rank paths (suffixes) and reuses them.
Inspired by[Jiménez+ WEA'03]

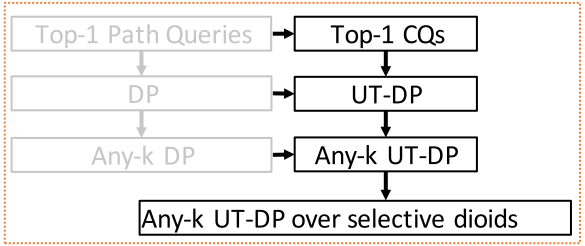
Wins when k is large.

$$\text{TTF} = O(ln)$$
$$\text{Delay(k)} = O(logn \cdot l)$$

Reusing computation may pay off –
can be even faster than sorting!
For Cartesian product with $n^\ell$ results:
Anyk-Rec TTL: $O\left(n^\ell(\log n + \ell)\right)$
Batch-Sorting/Anyk-Part: $O\left(n^\ell \log n \cdot \ell\right)$

# Generalizations

- Paths → Trees (Acyclic)

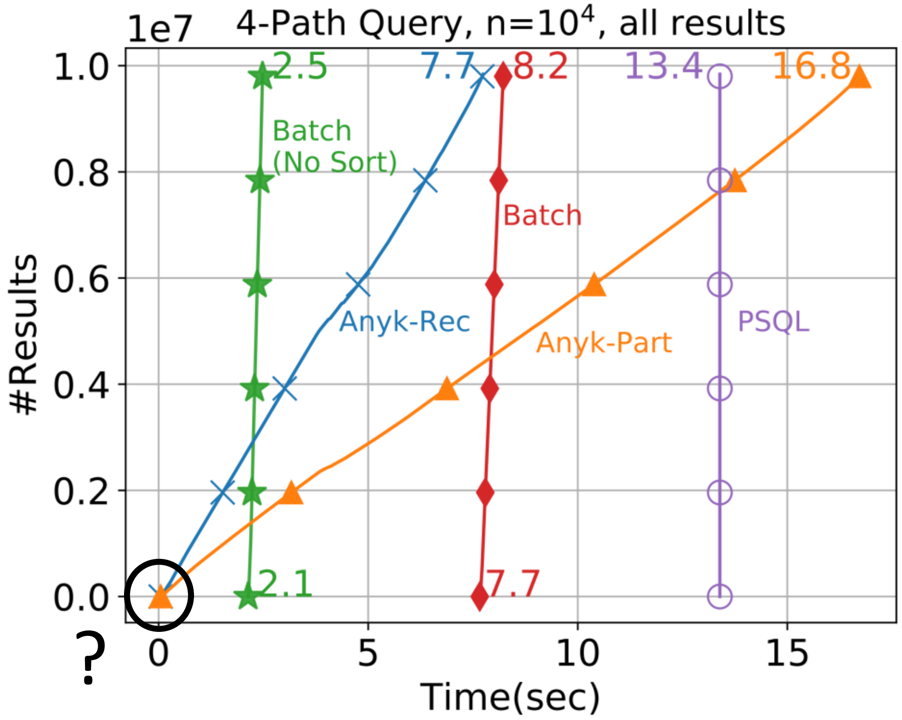


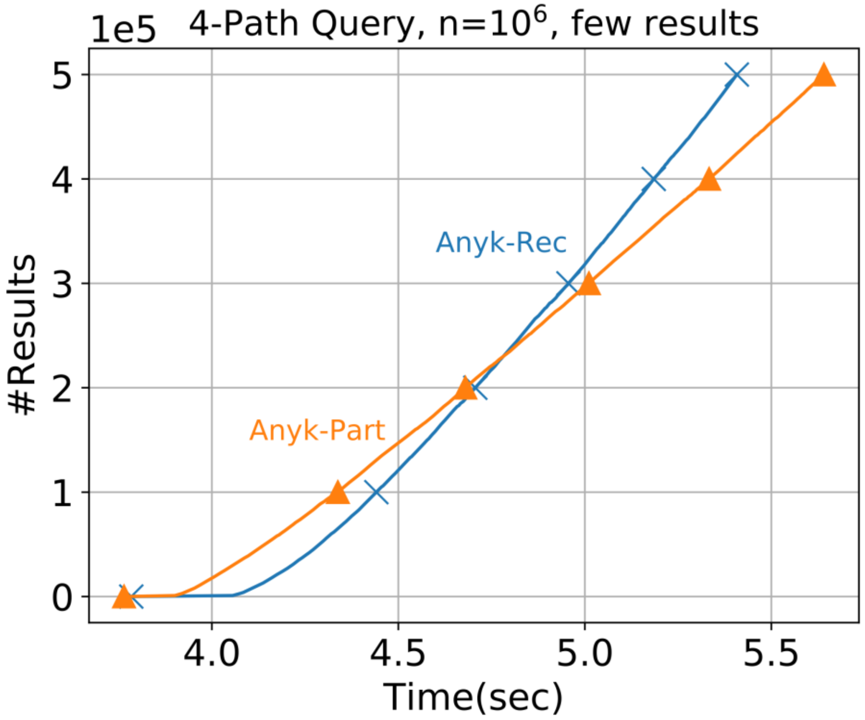

- Trees → Cycles
  - Decompose into a union of acyclic queries
  - e.g. 6-cycle $\mathrm{TTF} = O\left(n^{5/3}\right)$
    same as state-of-the-art Boolean query

- Ranking Function besides minimum sum of weights? $(\min, +)$
  - $(\min, \max)$: min traffic congestion
  - $(\max, \times)$ for non-negative reals: highest-prob. results
  - Lexicographic ordering (any, independent of join order)

Algebraic characterization as selective dioids

# Experiments




- Anyk starts much faster than Batch
- Anyk-Rec also finishes faster than Batch

- Anyk-Part is usually faster in the beginning

# Conclusions

- Ranked enumeration of arbitrary conjunctive queries
  [Yang+ ExploreDB'18]
  - Linear pre-processing (or higher for cyclic)
  - Logarithmic delay
  - Two competing algorithmic approaches


- Acknowledgements
  - National Institutes of Health (NIH) award number R01NS091421
  - National Science Foundation (NSF) award number CAREER IIS-1762268