Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries

Nikolaos Tziavelis\textsuperscript{1}, Deepak Ajwani\textsuperscript{2}, Wolfgang Gatterbauer\textsuperscript{1}, Mirek Riedewald\textsuperscript{1}, Xiaofeng Yang\textsuperscript{3}

\textsuperscript{1}Northeastern University, Boston, \textsuperscript{2}University College Dublin, \textsuperscript{3}VMWare

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

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 4.0 International License. See https://creativecommons.org/licenses/by-nc-sa/4.0/ for details
### Ranked Enumeration Example

#### SQL Query

```sql
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
```

#### Result Enumeration

1. **Rank-1**
   
   \((1, 0, 2, 3, \mathbf{18})\)

2. **Rank-2**
   
   \((2, 0, 2, 3, \mathbf{19})\)

3. **Rank-3**
   
   \((3, 1, 2, 3, \mathbf{22})\)

...
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:
- $\text{TTF} = \text{Time-to-First} = \text{TT}(1)$
- $\text{Delay}(k) = \text{Time between Rank-}k\text{ and Rank-}(k+1)$
- $\text{TTL} = \text{Time-to-Last} = \text{TT}(|\text{out}|)$
Conceptual Roadmap

Join Problems

Optimization

Ranked Enumeration

Paths/Serial

Top-1 Path Queries

DP

Any-k DP

Tropical semiring (min, +)

Cyclic/General

Top-1 Conjunctive Queries

Union of Tree-DP (UT-DP)

Any-k UT-DP

Any-k UT-DP over selective dioids
Main Result

• For Acyclic Queries:
  – $\text{TTF} = O(n)$
  – $\text{Delay}(k) = O(\log k)$
  – 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

$n$: database size
Query = fixed size
Top-1: Dynamic Programming

Bottom-up

Nodes: tuples
Edges: joining pairs
Labels: tuple weights
Bottom-up values: min total weight
Paths: join results
Any-k DP: k-shortest paths

Best result = Shortest Path (18)

2\textsuperscript{nd} Best Result = 2\textsuperscript{nd} Shortest Path (19)
Any-k DP Algorithms: 2 non-dominated families

**Anyk-Part**

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

**Variants**
- Eager
- All [Yang+ WWW’18]
- Lazy [Chang+ VLDB’15]
- Take2

Wins when k is small.

**Anyk-Rec**

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

Wins when k is large.

**TTF** = \(O(ln)\)

\[\text{Delay(k)} = O(\log n \cdot l)\]

Reusing computation may pay off – can be even *faster than sorting*!

For Cartesian product with \(n^\ell\) results:

- Anyk-Rec TTL: \(O(n^\ell (\log n + \ell))\)
- Batch-Sorting/Anyk-Part: \(O(n^\ell \log n \cdot \ell)\)

Lowest delay given linear-time pre-processing!
Generalizations

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

- **Ranking Function besides minimum sum of weights? (\text{min, +})**
  - (\text{min, max}): min traffic congestion
  - (\text{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