Any-k: Ranked Enumeration for Dynamic Programming

Nikolaos Tziavelis
Based on joint work with: Wolfgang Gatterbauer, Mirek Riedewald
Northeastern University, Boston

Website: https://northeastern-datalab.github.io/anyk/
Ranked Enumeration for Combinatorial Problems

**Enumeration**

- In order of importance

**Optimization**

- 2\textsuperscript{nd} best, 3\textsuperscript{rd} best, ...?

**Ranked Enumeration**

- "Any-k"
- Anytime algorithms + Top-k

**Dynamic Programming**

(& semirings)
Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms

• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure

• Conclusion
• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms
• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure
• Conclusion
Dynamic Programming

View of DP as a Directed Acyclic Graph (DAG)

\[ f(i) = \min \{ c_{ij} + f(j), c_{ir} + f(r) \} \]
Example: Longest Increasing Subsequence
Longest Increasing Subsequence with DP

Dynamic Programming

\[
\text{for } i = n, \ldots, 2, 1:
\]

\[
f(i) = \max\{f(j) + 1 | j > i, v(j) > v(i)\}
\]

return \(\max\{f(i) | i \in [n]\}\)

Shortest path in DAG

\[
\text{for } i = n, \ldots, 2, 1, s:
\]

\[
d(i) = \min\{d(j) + c_{ij} | (i, j) \in E\}
\]

return \(-d(s)\)
Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms
• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure
• Conclusion
DP & Semirings

Previous example:

\[ f(i) = \max \{ f(j) + 1 | j \in \ldots \} \quad \text{and} \quad f(i) = \bigoplus_j \{ f(j) \otimes c_{ij} \} \]

Commutative Semiring \((W, \oplus, \otimes, 0, 1)\)

1. \((W, \oplus, 0)\) is a commutative monoid
2. \((W, \otimes, 1)\) is a commutative monoid
3. \(\otimes\) distributes over \(\oplus\):
   \( (x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z) \)
4. \(0\) annihilates \(\otimes\):
   \(0 \otimes x = 0\)

Mohri. Semiring frameworks and algorithms for shortest-distance problems. JALC'02  
Total Order & Selectivity

- Additional property for total order: Selectivity
  - $\forall xy: (x \oplus y = x) \lor (x \oplus y = y)$
  - Semiring with Selectivity = Selective Dioid

- “Natural” total order: $x \leq y$ iff $x \oplus y = x$

- Examples:
  - Tropical semiring $(\mathbb{R}^\infty, \min, +, \infty, 0)$ ✓
  - Viterbi semiring $([0,1], \max, \times, 0, 1)$ ✓
  - Boolean semiring $\{0,1\}, \lor, \land, 0, 1)$ ✓
  - Natural numbers semiring $(\mathbb{N}, +, \times, 0, 1)$ ✗
    - Can count #paths / #solutions
    - What would the 2\textsuperscript{nd} best solution be here?
Distributivity → Monotonicity

**Monotonicity in Selective Dioids:** \( x \leq y \Rightarrow x \otimes z \leq y \otimes z \)

**Proof**

\[
x \leq y \\
x \oplus y = x \\
(x \oplus y) \otimes z = x \otimes z \\
(x \otimes z) \oplus (y \otimes z) = x \otimes z \\
(x \otimes z) \leq y \otimes z
\]

Equivalent to "optimal substructure" property in DP

\[
s \quad \quad \quad m \\ x \quad \quad \\ y \\ z \\ t \\ x \leq y \otimes z
\]
Monotonicity Classes for Ranking Functions

Holistic-Monotone
[Fagin+ 03]

Subset-Monotone
[Kimelfeld+ 06]

Median

Commutative
Selective
Dioids

\[ f(1, 2, 3, 4) \leq f(5, 6, 7, 8) \]

\[ f(1, 2, 3, 4) \leq f(1, 2, 2, 8) \]

---


Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms
• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure
• Conclusion
Any-k Algorithms

- Best answer (DP) $\rightarrow$ shortest path in DAG
- $k^{th}$ best answer $\rightarrow k^{th}$ shortest path in DAG
- Best we know for subset-monotone ranking functions:

$$TT(k) = O(|G| + k(\log k + \ell))$$
Measures of Enumeration: $\text{TT}(k)$ vs Delay

- Fastest answers: $\text{TT}(k)$
- What about delay?

- We can upper bound $\text{TT}(k)$ with delay:
  - delay $\leq c \Rightarrow \text{TT}(k) \leq |\text{Prep}| + ck$
- Improving the delay of an algorithm with “good $\text{TT}(k)$” can slow it down
- Lower bounds are more general if stated for $\text{TT}(k)$
TT($k$) vs Delay Gap

Is the gap “real”?

**Answer #1** [CS23]

Incremental delay => delay at the cost of a log factor for unranked enumeration

**Answer #2**

For ranked enumeration, we can get a better algorithm for TT($k$)

The Anyk-Part+ Algorithm

\[ O(|G| + k(\log k + \ell)) \rightarrow O(|G| + k(\log N + \ell)) \]

Previously best known

Small k: \( O(|G|) \) dominates, same
Large k: Better
Faster than sorting for entire sorted output
Monotonicity Classes

Holistic-Monotone
[Fagin+ 03]

Subset-Monotone
[Kimelfeld+ 06]

Strong-Subset-Monotone
[Tziavelis+ 22]

((−∞, ∞], min, +, ∞, 0)
([−∞, ∞], min, max, ∞, −∞)

Examples
Median
([0, ∞], min, ×, ∞, 1)

Algorithms
Threshold Algorithm
AnyK-Part
AnyK-Rec

AnyK-Part+

Examples

[18]

[129] Median

Algorithm

[126] Threshold

Algorithm

AnyK-Part

AnyK-Rec

Example

Strong-Subset-Monotonicity

\[ f(X_1, Y_1) \leq f(X_1, Y_2) \land f(X_1) \leq f(X_2) \Rightarrow f(X_2, Y_1) \leq f(X_2, Y_2) \]

\((\neg \infty, \infty], \min, +, \infty, 0)\quad \checkmark\]

1) (1, 1, 1)
2) (1, 1, 2)
3) (1, 4, 1)

\[ f(2,1,1) < f(2,1,2) \]
\[ f(2,1,2) < f(2,4,1) \]

\(([0, \infty], \min, \times, \infty, 1)\quad \times\]

1) (0, 9, 9)
2) (0, 1, 1)
3) (0, 2, 2)

\[ f(1,9,9) < f(1,1,1) \]
History of Algorithms for k-Shortest Paths on a DAG

Implicit Path Enumeration

Top-1 Subroutine

Subset-Monotonicity (distributivity)

[Lawler 1972], [Murty 1968]
→ CQs [Kimelfeld, Sagiv 2006]
→ CSPs [Greco, Scarcello 2011]

[Lawler, Kalaba 1960] (Fixed k)
→ Top-k semiring in FAQ
[Obo-Khamis+ 2016]
→ d-DNNFs: [Bourhis+ 2022]

Strong-Subset-Monotonicity

Explicit Path Enumeration

deviations

Anyk-Part
→ Graph patterns
[Yang+ 2018] “All” variant
[Chang+ 2015] “Lazy” variant
→ Free-connex CQs [Tziavelis+ 2020]
“Quick” variant uses “IQS”
[Paredes, Navarro 2006]

Anyk-Rec/REA
[Jiménez, Marzal 1999]
→ Full CQs: [Deep, Koutris 2021]
→ Free-connex CQs: [Tziavelis+ 2020]

Anyk-Part+ [Tziavelis+ 2022]

+ restrict computed paths per node
+ top-down recursive calls

[Deep, Navarro 2006]

Same TT(k) for explicit

[Bellman, Kalaba 1960] (Fixed k)

→ CQs [Kimelfeld, Sagiv 2006]
→ CSPs [Greco, Scarcello 2011]

+ heapify deviation structure

[Hoffman, Pavley 1959]

top-k for every node (bottom-up)

+ restrict computed paths per node

[Bourhis+ 2022]

+ top-down recursive calls

[Deep, Koutris 2021]

+ restrict computed paths per node

[Bourhis+ 2022]
Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms

• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure

• Conclusion
Ranked Enumeration for Conjunctive Queries

\[ Q(x, y, z, u) : - R(x, y), S(y, z), T(z, u) \]

\[ w_R + w_S + w_T \]

**Increasing sum of weights**

(1, 1, 4, 1, 111) \[ \rightarrow \] (2, 1, 4, 1, 112) \[ \rightarrow \] (1, 1, 6, 4, 231) \[ \rightarrow \] ...
Path CQ $\rightarrow$ DP

\[
Q(x, y, z, u) : - R(x, y), S(y, z), T(z, u)
\]

\[w_R + w_S + w_T\]

\[
R(x, y) \quad y \quad S(y, z) \quad z \quad T(z, u)
\]

\[\begin{array}{cccc}
1, & 1 & 10 & 1, 4 \\
2, & 1 & 20 & 1, 5 \\
3, & 2 & 30 & 1, 6 \\
4, & 2 & 40 & 2, 6 \\
\end{array}\]

\[\begin{array}{cccc}
1, 4 & z=4 & 100 & 4, 1 \\
1, 5 & z=5 & 200 & 4, 2 \\
1, 6 & z=6 & 300 & 5, 3 \\
2, 6 & & 400 & 6, 4 \\
\end{array}\]

\[\text{Factorization: } O(n^2) \rightarrow O(n)\]

Also: $O(n \text{ polylog } n)$ for inequality joins

\[\text{DAG}\]

- Nodes = Tuples
- Edges = Joining pairs
- Paths = Join answers

Tziavelis, Gatterbauer, Riedewald. Beyond Equi-joins: Ranking, Enumeration and Factorization. PVLDB’21 https://doi.org/10.14778/3476249.3476306
Join Trees

• Acyclic CQs $\iff$ Join Trees

• Join Tree:
  - Atoms as nodes
  - For each variable $X$, the nodes containing $X$ are connected

$Q(x, y, z, u): \neg R(x, y), S(y, z), T(z, u), U(z, v)$

Path $\rightarrow$ Tree $\rightarrow$ Cyclic

DP

Hypertree decompositions
Tree Queries

Tree-DP
(Non-serial Dynamic Programming with “diverging branches”)

\[ R \]
\[ S \]
\[ T \]
\[ U \]
Guarantees for Full Acyclic CQs

- **Data Complexity & Subset-Monotonicity** [TAGRY20,DK21]:
  \[
  \text{TT}(k) = O(n + k \log k)
  \]

- **Combined Complexity & Subset-Monotonicity** [TAGRY20]:
  \[
  \text{TT}(k) = O(n\ell \alpha + k(\log k + \ell \alpha))
  \]

- **Combined Complexity & Strong-Subset-Monotonicity** [TGR22]:
  \[
  \text{TT}(k) = O(n\ell \alpha + k(\log(\min\{k, n^{\ell - \text{diam}(Q)+1}\}) + \ell \alpha))
  \]

$n$: #tuples
$\ell$: #atoms
$\alpha$: arity

[https://doi.org/10.14778/3397230.3397250](https://doi.org/10.14778/3397230.3397250)

[DK21] Deep, Koutris. Ranked Enumeration of Conjunctive Query Results. ICDT’21
[https://doi.org/10.4230/LIPIcs.ICDT.2021.5](https://doi.org/10.4230/LIPIcs.ICDT.2021.5)

Ranked Enumeration in Practice

https://doi.org/10.14778/3397230.3397250
Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms

• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure

• Conclusion
Lexicographic Orders

\[
Q(x, y, z, u) : - R(x, y), S(y, z), T(z, u)
\]

\[x \rightarrow y \rightarrow z \rightarrow u\]

- Lexicographic order as semiring:
  - Option 1: Map to \((\min, +)\) semiring with appropriate weights
  - Option 2: Define semiring on tuples with one position per variable and two appropriate operations (lexicographic min, union)
- Logarithmic delay for any lexicographic order
- Can we do better by taking into account the structure of the query?
Lexicographic Orders

\[ Q(x, y, z, u) : \neg R(x, y), S(y, z), T(z, u) \]

\[ x \rightarrow y \rightarrow z \rightarrow u \]

Constant-delay enumeration if lexicographic order agrees with a (reverse) \( \alpha \)-elimination order for the query. [BKOZ13, BDG07]


Ranking Function Compatible with Tree Decomposition

\[ Q(x, y, z) : - R(x, y), S(y, z) \]

\[ g(x, y) + g(y, z) \]

Not subset-monotone

Logarithmic delay for ranking functions that are **compatible with a tree decomposition** (which determines preprocessing). [DK21]
Outline

• Ranked Enumeration & Dynamic Programming
  – DP as a DAG
  – Semirings
  – Any-k Algorithms

• Ranked Enumeration for (Full) Conjunctive Queries
  – Mapping CQs to DP
  – Ranking Function & Query Structure

• Conclusion
Conclusion

• Same ranked enumeration algorithms appear in many different problems and the common link is DP and semirings
• Different monotonicity notions allow for different algorithms
• We can potentially do more if we take into account the structure of the problem
• Practical results outperforming database systems by orders of magnitude
Open Questions

• Precise characterization of tractable queries + ranking functions.
  - Lower bounds
  - Algorithms for Holistic-Monotone ranking functions (e.g., MEDIAN)?
  - Can we leverage the structure of the query more to cover more cases?
• Understand better the relationship between $TT(k)$ and delay
• Relationship of this DP framework to circuits
  - Can all these algorithms be carried over?

Thank you!

Website: https://northeastern-datalab.github.io/anyk/