State-of-the-Art Join Algorithms in Database (Theory) and Lower Bounds from Fine-Grained Complexity

Austen Z. Fan Paraschos Koutris Hangdong Zhao

University of Wisconsin-Madison

DB Affiliates Workshop
Oct 11, 2023

Outline

Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

$$q\left(x_1,\ldots,x_k\right) :- R_1\left(\vec{y}_1\right),\ldots,R_n\left(\vec{y}_n\right).$$

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

$$q\left(x_1, \ldots, x_k\right) : -R_1\left(\vec{y}_1\right), \ldots, R_n\left(\vec{y}_n\right).$$

Example

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

$$q\left(x_1,\ldots,x_k\right):-R_1\left(\vec{y}_1\right),\ldots,R_n\left(\vec{y}_n\right).$$

Example

Listing 3-cycles

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

$$q\left(x_1, \ldots, x_k\right) : -R_1\left(\vec{y}_1\right), \ldots, R_n\left(\vec{y}_n\right).$$

Example

Listing 3-cycles

```
SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B
JOIN Table3 t3 ON t2.C = t3.C AND t1.A = t3.A;
```

Vocabularies, I

(Select-)Project-Join = Conjunctive Query

$$q\left(x_1, \ldots, x_k\right) : -R_1\left(\vec{y}_1\right), \ldots, R_n\left(\vec{y}_n\right).$$

Example

Listing 3-cycles

```
SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B
JOIN Table3 t3 ON t2.C = t3.C AND t1.A = t3.A;
```

$\iff$

$q(x, y, z) : -R(x, y), S(y, z), T(z, x)$

Vocabularies, II

Vocabularies, II

For every CQ q, we associate a *(hyper-)graph* $\mathcal{H}_q$ to it, where the vertices are variables and the (hyper-)edges are atoms.

Vocabularies, II

For every CQ q, we associate a *(hyper-)graph* $\mathcal{H}_q$ to it, where the vertices are variables and the (hyper-)edges are atoms.

Example

Vocabularies, II

For every CQ q, we associate a *(hyper-)graph* $\mathcal{H}_q$ to it, where the vertices are variables and the (hyper-)edges are atoms.

Example

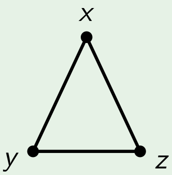
$q(x, y, z) :- R(x, y), S(y, z), T(z, x)$

Vocabularies, II

For every CQ q, we associate a *(hyper-)graph* $\mathcal{H}_q$ to it, where the vertices are variables and the (hyper-)edges are atoms.

Example

$q(x, y, z) : -R(x, y), S(y, z), T(z, x)$



Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\mathrm{In}| + |\mathrm{Out}|)$.

Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\,\mathsf{In}\,| + |\,\mathsf{Out}\,|)$.

Example

Listing 2-paths

Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\mathrm{In}| + |\mathrm{Out}|)$.

Example

Listing 2-paths

```
SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B;
```

Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\,\mathrm{In}\,| + |\,\mathrm{Out}\,|)$.

Example

Listing 2-paths

```
SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B;
```

$\Longleftrightarrow$

$q(x, y, z) : -R(x, y), S(y, z)$

Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\mathrm{In}| + |\mathrm{Out}|)$.

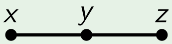
Example

Listing 2-paths

SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B;

$\Longleftrightarrow$

$q(x, y, z) :- R(x, y), S(y, z)$



Yannakakis Algorithm

Theorem (Yannakakis, 81')

If H_q is acyclic, then we can evaluate q in linear time $O(|\mathrm{In}| + |\mathrm{Out}|)$.

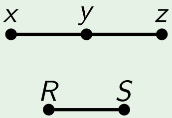
Example

Listing 2-paths

```
SELECT t1.A, t1.B, t2.C
FROM Table1 t1
JOIN Table2 t2 ON t1.B = t2.B;
```

$\Longleftrightarrow$

$q(x, y, z) : -R(x, y), S(y, z)$



Yannakakis Algorithm, Formal Version

Yannakakis Algorithm, Formal Version

Example

Yannakakis Algorithm, Formal Version

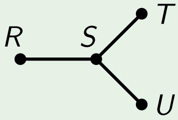
Example

$q(\vec{\mathbf{x}}) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$

Yannakakis Algorithm, Formal Version

Example

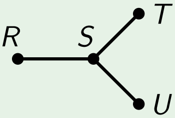
$q(\vec{\mathbf{x}}) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$



Yannakakis Algorithm, Formal Version

Example

$q(\vec{\mathbf{x}}) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$



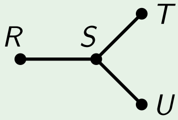
Definition (Join Tree)

A *join tree* for a CQ q is a tree $\mathcal{T}$ whose vertices are the atoms in q such that, for any pair of atoms R, S, all variables common to R and S occur on the unique path connecting R and S.

Yannakakis Algorithm, Formal Version

Example

$q(\vec{\mathbf{x}}) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$



Definition (Join Tree)

A *join tree* for a CQ q is a tree $\mathcal{T}$ whose vertices are the atoms in q such that, for any pair of atoms R, S, all variables common to R and S occur on the unique path connecting R and S.

Theorem (Yannakakis, 81')

If a CQ q has a join tree, then we can evaluate q in linear time $O(|\,\mathsf{In}\,| + |\,\mathsf{Out}\,|)$.

Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

AGM Bound

Example

$q(x, y) :- R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size?

AGM Bound

Example

$q(x, y) : -R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.

AGM Bound

Example

$q(x, y) :- R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.
$q(x, y, z) :- R(x, y), S(y, z), T(z, x)$, given $|R| = |S| = |T| = N$, what's the worst-case output size?

AGM Bound

Example

$q(x, y) :- R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.
$q(x, y, z) :- R(x, y), S(y, z), T(z, x)$, given $|R| = |S| = |T| = N$, what's the worst-case output size? $O(N^3)$?

AGM Bound

Example

$q(x, y) : -R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.
$q(x, y, z) : -R(x, y), S(y, z), T(z, x)$, given $|R| = |S| = |T| = N$, what's the worst-case output size? $O(N^3)$? No, it's $O(N^{\frac{3}{2}})$!

AGM Bound

Example

$q(x, y) :- R(x, y), S(y, z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.
$q(x, y, z) :- R(x, y), S(y, z), T(z, x)$, given $|R| = |S| = |T| = N$, what's the worst-case output size? $O(N^3)$? No, it's $O(N^{\frac{3}{2}})$!

Theorem (AGM Bound)

The worst-case output size of q is bounded by $N^{\rho^*(\mathcal{H}_q)}$, where ρ^* is the fractional edge cover.

AGM Bound

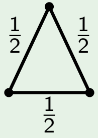
Example

$q(x,y) :- R(x,y), S(y,z)$, given $|R| = |S| = N$, what's the worst-case output size? $O(N^2)$, fine.
$q(x,y,z) :- R(x,y), S(y,z), T(z,x)$, given $|R| = |S| = |T| = N$, what's the worst-case output size? $O(N^3)$? No, it's $O(N^{\frac{3}{2}})$!

Theorem (AGM Bound)

The worst-case output size of q is bounded by $N^{\rho^*(\mathcal{H}_q)}$, where ρ^* is the fractional edge cover.

Example



Heavy-Light-Split Query Plan

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Directed 2-paths with intermediate vertices being light:

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Directed 2-paths with intermediate vertices being light: there are $N \cdot \sqrt{N}$ many and they can be found in $O(N \cdot \sqrt{N}) = O(N^{\frac{3}{2}})$ time. For each such path, check whether the endpoints are connected.

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Directed 2-paths with intermediate vertices being light: there are $N \cdot \sqrt{N}$ many and they can be found in $O(N \cdot \sqrt{N}) = O(N^{\frac{3}{2}})$ time. For each such path, check whether the endpoints are connected.

Otherwise, all vertices are heavy:

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Directed 2-paths with intermediate vertices being light: there are $N \cdot \sqrt{N}$ many and they can be found in $O(N \cdot \sqrt{N}) = O(N^{\frac{3}{2}})$ time. For each such path, check whether the endpoints are connected.

Otherwise, all vertices are heavy: but there are at most $\frac{2N}{\sqrt{N}} = O(\sqrt{N})$ many heavy vertices. Construct the $O(\sqrt{N})$-by-$O(\sqrt{N})$ matrix and use matrix multiplication to find in $O((\sqrt{N})^3) = O(N^{\frac{3}{2}})$ time.

Heavy-Light-Split Query Plan

Example (Listing Triangles)

Call a vertex *heavy* if its degree $\geq \sqrt{N}$ and *light* otherwise.

Directed 2-paths with intermediate vertices being light: there are $N \cdot \sqrt{N}$ many and they can be found in $O(N \cdot \sqrt{N}) = O(N^{\frac{3}{2}})$ time. For each such path, check whether the endpoints are connected.

Otherwise, all vertices are heavy: but there are at most $\frac{2N}{\sqrt{N}} = O(\sqrt{N})$ many heavy vertices. Construct the $O(\sqrt{N})$-by-$O(\sqrt{N})$ matrix and use matrix multiplication to find in $O((\sqrt{N})^3) = O(N^{\frac{3}{2}})$ time.

Theorem (WCOJ by Ngo, Porat, Ré & Rudra, 12')

Any full CQ q can be computed in time $O(N^{\rho^*(\mathcal{H}_q)})$.

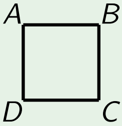
Tree Decomposition

Example

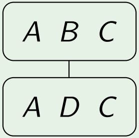
Tree Decomposition

Example

A tree decomposition for

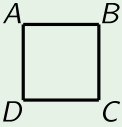


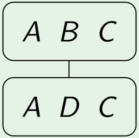
is



Tree Decomposition

Example

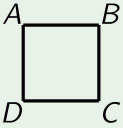
A tree decomposition for 

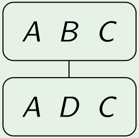
is 

Tree decomposition = Query plan

Tree Decomposition

Example

A tree decomposition for 

is 

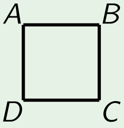
Tree decomposition = Query plan

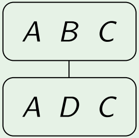
Definition (Tree Decomposition)

A *tree decomposition* of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and $\chi : V(\mathcal{T}) \to 2^{\mathcal{V}}$, such that (1) $\forall e \in \mathcal{E}$ is a subset for some $\chi(t), t \in V(\mathcal{T})$ and (2) $\forall v \in \mathcal{V}$ the set $\{t \mid v \in \chi(t)\}$ is a non-empty connected sub-tree of $\mathcal{T}$.

Tree Decomposition

Example

A tree decomposition for 

is 

Tree decomposition = Query plan

Definition (Tree Decomposition)

A *tree decomposition* of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and $\chi : V(\mathcal{T}) \to 2^{\mathcal{V}}$, such that (1) $\forall e \in \mathcal{E}$ is a subset for some $\chi(t), t \in V(\mathcal{T})$ and (2) $\forall v \in \mathcal{V}$ the set $\{t \mid v \in \chi(t)\}$ is a non-empty connected sub-tree of $\mathcal{T}$.

Just ensemble tables into bags, run WCOJ on each bag and then run Yannakakis on those bags!

Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

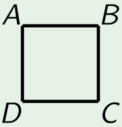
Two is better than one

Example

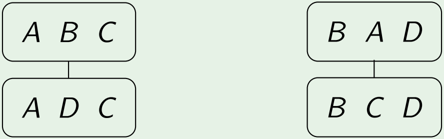
Two is better than one

Example

The 4-cycle query



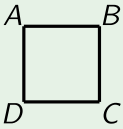
has two tree decompositions:



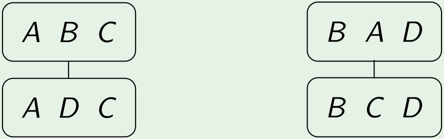
Two is better than one

Example

The 4-cycle query



has two tree decompositions:

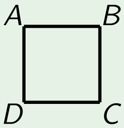


Which one to use?

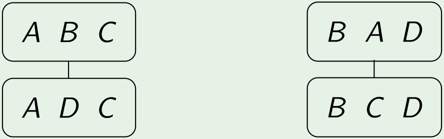
Two is better than one

Example

The 4-cycle query



has two tree decompositions:

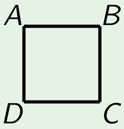


Which one to use? It depends?

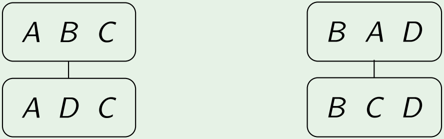
Two is better than one

Example

The 4-cycle query



has two tree decompositions:



Which one to use? It depends?

Use two decompositions at the same time!

PANDA

Example (4-Cycle)

PANDA

Example (4-Cycle)

Using one tree decomposition $\rightarrow O(N^2)$.

PANDA

Example (4-Cycle)

Using one tree decomposition $\rightarrow O(N^2)$.
Using two tree decompositions $\rightarrow O(N^{\frac{3}{2}})$!

PANDA

Example (4-Cycle)

Using one tree decomposition $\rightarrow O(N^2)$.
Using two tree decompositions $\rightarrow O(N^{\frac{3}{2}})$!
Technicality: partition the join in such a way that every output is covered by at least one query plan!

PANDA

Example (4-Cycle)

Using one tree decomposition $\rightarrow O(N^2)$.
Using two tree decompositions $\rightarrow O(N^{\frac{3}{2}})$!
Technicality: partition the join in such a way that every output is covered by at least one query plan!

Theorem (SOTA PANDA, Abo Khamis, Ngo & Suciu, 16')

Any full q can be computed in time $\tilde{O}(N^{\mathsf{subw}(q)} + |Out|)$.

PANDA

Example (4-Cycle)

Using one tree decomposition $\rightarrow O(N^2)$.
Using two tree decompositions $\rightarrow O(N^{\frac{3}{2}})$!
Technicality: partition the join in such a way that every output is covered by at least one query plan!

Theorem (SOTA PANDA, Abo Khamis, Ngo & Suciu, 16')

Any full q can be computed in time $\tilde{O}(N^{\mathrm{subw}(q)} + |Out|)$ *.*

Lemma (Marx, 10')

For any hypergraph $\mathcal{H}$, $\mathrm{subw}(\mathcal{H}) \leq \mathrm{fhtw}(\mathcal{H})$.

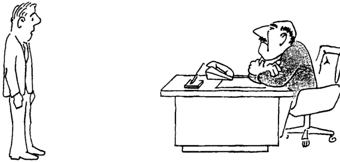
Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

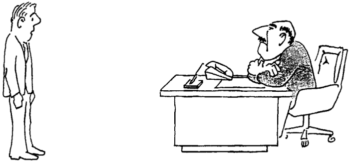
How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

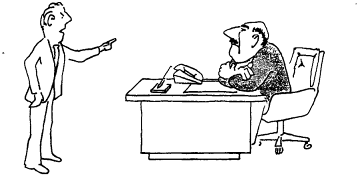
Can we do better? – Lower Bounds from Fine-Grained Complexity



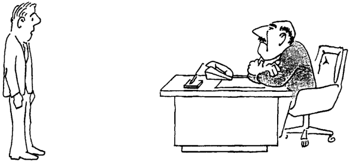
"I can't find an efficient algorithm, I guess I'm just too dumb."



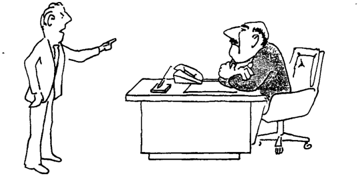
"I can't find an efficient algorithm, I guess I'm just too dumb."



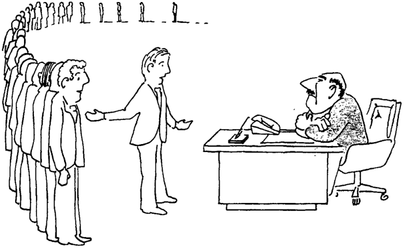
"I can't find an efficient algorithm, because no such algorithm is possible!"



"I can't find an efficient algorithm, I guess I'm just too dumb."



"I can't find an efficient algorithm, because no such algorithm is possible!"



"I can't find an efficient algorithm, but neither can all these famous people."

Lower Bounds

Lower Bounds

Theorem (Fan, Koutris & Zhao, 23')

Any q cannot be computed via a combinatorial algorithm in time $O(|I|^{\mathsf{emb}(\mathcal{H}_q)-\epsilon})$ *unless the Combinatorial k-Clique Conjecture is false.*

Lower Bounds

Theorem (Fan, Koutris & Zhao, 23')

Any q cannot be computed via a combinatorial algorithm in time $O(|I|^{\mathrm{emb}(\mathcal{H}_q)-\epsilon})$ unless the Combinatorial k-Clique Conjecture is false.

	emb	subw
Acyclic	1	1
Chordal	$=$	$=$
ℓ-cycle	$2-1/\lceil \ell/2 \rceil$	$2-1/\lceil \ell/2 \rceil$
$K_{2,\ell}$	$2-1/\ell$	$2-1/\ell$
$K_{3,3}$	2	2
A_ℓ	$(\ell-1)/2$	$(\ell-1)/2$
$\mathcal{H}_{\ell,k}$	ℓ/k	ℓ/k
Q_b	17/9	2
Q_{hb}	7/4	2

Table: Embedding power and submodular width for some query classes

Summary

Vocabularies

Acyclicity takes it all. – Yannakakis Algorithm

How is that different from a tree? – Worst-Case Optimal Join

Always have a plan B! – PANDA

Can we do better? – Lower Bounds from Fine-Grained Complexity

Thank You!

References I

Albert Atserias, Martin Grohe, and Dániel Marx, *Size bounds and query plans for relational joins*, SIAM J. Comput. **42** (2013), no. 4, 1737–1767.

Austen Z. Fan, Paraschos Koutris, and Hangdong Zhao, *The fine-grained complexity of boolean conjunctive queries and sum-product problems*, ICALP, LIPIcs, vol. 261, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 127:1–127:20.

M. R. Garey and David S. Johnson, *Computers and intractability: A guide to the theory of np-completeness*, W. H. Freeman, 1979.

References II

- Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu, *What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another?*, PODS, ACM, 2017, pp. 429–444.
- Dániel Marx, *Tractable hypergraph properties for constraint satisfaction and conjunctive queries*, J. ACM **60** (2013), no. 6, 42:1–42:51.
- Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra, *Worst-case optimal join algorithms*, J. ACM **65** (2018), no. 3, 16:1–16:40.
- Mihalis Yannakakis, *Algorithms for acyclic database schemes*, VLDB, IEEE Computer Society, 1981, pp. 82–94.