CSE599d: Advanced Query Processing

Lecture 16: Worst-Case Optimal Join

Dan Suciu

University of Washington

Announcements

- HW3 is graded
- HW4 is posted and due on Friday, March 13

Today's Agenda

- Worst-Case Optimal Join (WCOJ)
- Extensions of the AGM bound: adding keys, adding projections

Worst Case Optimal Joins

Types of Optimality

Q is fixed.
Problem: given $\boldsymbol{D}$, compute $Q(\boldsymbol{D})$.

Runtime of algorithm A: $T_A(\boldsymbol{D})$.

An algorithm A is:

Instance Optimal if $\forall \boldsymbol{D}$ and any algorithm B, $T_A(\boldsymbol{D}) = O(T_B(\boldsymbol{D}))$.
If Q is acyclic free-connex, Yannakakis' algorithm is instance optimal.

Worst-Case Optimal if $\forall \boldsymbol{D}, \exists \boldsymbol{D}'$, $|\boldsymbol{D}'| \leq |\boldsymbol{D}|$ s.t. for any B, $T_A(\boldsymbol{D}) = O(T_B(\boldsymbol{D}'))$.
If Q is full CQ, then A is WCOJ iff $T_A(\boldsymbol{D}) = O(AGM(Q))$.

Generic Join

WCOJ algorithms for full conjunctive query:

- NPRR: first WCOJ, complex [Ngo et al., 2012].
- TrieJoin: developed independently and deployed [Veldhuizen, 2014].
- Generic Join (GJ) is a much simplified [Ngo et al., 2013].
- GJ is sometimes called WCOJ; we will call it GJ.

Generic Join (GJ) Overview

Computes a full CQ in time $O(AGM(Q, \boldsymbol{D}))$.

Consists of k nested loops, one for each query variable X_i, $i = 1, k$.

Each loop i iterates over the intersection of all columns X_i.

GJ for the Triangle Query

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

Algorithm 1: Generic Join for the triangle query

for $x \in R.X \cap T.X$ **do**

GJ for the Triangle Query

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

Algorithm 2: Generic Join for the triangle query

for $x \in R.X \cap T.X$ **do**
    for $y \in R[X = x].Y \cap S.Y$ **do**

GJ for the Triangle Query

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

Algorithm 3: Generic Join for the triangle query

```
for x ∈ R.X ∩ T.X do
    for y ∈ R[X = x].Y ∩ S.Y do
        for z ∈ S[Y = y].Z ∩ T[X = x].Z do
            emit(x, y, z);
```

GJ for the Triangle Query

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

Algorithm 4: Generic Join for the triangle query

```
for x ∈ R.X ∩ T.X do
    for y ∈ R[X = x].Y ∩ S.Y do
        for z ∈ S[Y = y].Z ∩ T[X = x].Z do
            emit(x, y, z);
```

Runtime:

$$T(\boldsymbol{D}) = O(AGM(Q,\boldsymbol{D})) = \min((|R|\cdot|S|\cdot|T|)^{\frac{1}{2}}, |R|\cdot|S|, |R|\cdot|T|, |S|\cdot|T|)$$

Notations

To describe GJ in general, we need some notations:

Database instance: $\boldsymbol{D} = (R_1^D, \ldots, R_m^D)$

Residual relation: $R_j^D[X = x] := \begin{cases} \sigma_{X=x}(R_j^D) & \text{if } X \in \texttt{Vars}(R_j) \\ R_j^D & \text{otherwise} \end{cases}$

Residual database: $\boldsymbol{D}[X = x] = (R_1^D[X = x], \ldots, R_m^D[X = x])$

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 5: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 6: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

---

if $i > n$ **then** emit$(x_1, \ldots, x_n)$;

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 7: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

---

if $i > n$ **then** emit$(x_1, \ldots, x_n)$;

else

  for $x_i \in \bigcap_{j: X_i \in \mathtt{Vars}(R_j)} R_j.X_i$ **do**

    |

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 8: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

---

if $i > n$ **then** emit$(x_1, \ldots, x_n)$;
else
  for $x_i \in \bigcap_{j: X_i \in \texttt{Vars}(R_j)} R_j.X_i$ **do**
    Eval($\boldsymbol{D}_{\text{residual}}[X_i = x_i], (x_1, \ldots, x_i), i + 1$);

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 9: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

---

if $i > n$ **then** emit$(x_1, \ldots, x_n)$;
else
  for $x_i \in \bigcap_{j: X_i \in \texttt{Vars}(R_j)} R_j.X_i$ **do**
    Eval($\boldsymbol{D}_{\text{residual}}[X_i = x_i], (x_1, \ldots, x_i), i + 1$);

$GJ(\boldsymbol{D})$:= Eval($\boldsymbol{D}, (\,), 1$)

GJ for a Full Conjunctive Query

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m)$$

Order the variables arbitrarily: $X_1, X_2, \ldots, X_n$

Algorithm 10: Eval($\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i$)

---

if $i > n$ **then** emit$(x_1, \ldots, x_n)$;
else
  for $x_i \in \bigcap_{j: X_i \in \texttt{Vars}(R_j)} R_j.X_i$ **do**
    Eval($\boldsymbol{D}_{\text{residual}}[X_i = x_i], (x_1, \ldots, x_i), i+1$);

$GJ(\boldsymbol{D})$:= Eval($\boldsymbol{D}, (\,), 1$)

Runtime: $\boxed{T_{GJ}(\boldsymbol{D}) = O(AGM(Q, \boldsymbol{D}))}$

Runtime of GJ

We will prove that the runtime is $AGM(Q, \boldsymbol{D})$.

Notice that $AGM(Q, \boldsymbol{D})$ may be smaller than the largest relations!

E.g. for the triangle query, $AGM = |S| \cdot |T|$ when $|R| \gg |S| \cdot |T|$

GJ runs in time $O(|S| \cdot |T|)$: does not read entire R!

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of Eval$(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of $\text{Eval}(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

$$T_i(\boldsymbol{D}_{\text{residual}}) = T_{\text{intersect}} + \sum_{x_i} T_{i+1}(\boldsymbol{D}_{\text{residual}}[X_i = x_i])$$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of Eval$(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

$$T_i(\boldsymbol{D}_{\text{residual}}) = T_{\text{intersect}} + \sum_{x_i} T_{i+1}(\boldsymbol{D}_{\text{residual}}[X_i = x_i])$$

$$\overset{\text{induction}}{=} T_{\text{intersect}} + O\left(\sum_{x_i}\left(\prod_{j:X_i \in \texttt{Vars}(R_j)} |R_j^D[X_i = x_i]|^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)\right)$$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of $\text{Eval}(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

$$T_i(\boldsymbol{D}_{\text{residual}}) = T_{\text{intersect}} + \sum_{x_i} T_{i+1}(\boldsymbol{D}_{\text{residual}}[X_i = x_i])$$

$$\overset{\text{induction}}{=} T_{\text{intersect}} + O\left(\sum_{x_i}\left(\prod_{j:X_i \in \texttt{Vars}(R_j)} |R_j^D[X_i = x_i]|^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)\right)$$

$$\overset{\texttt{Hölder}}{\leq} T_{\text{intersect}} + O\left(\prod_{j:X_i \in \texttt{Vars}(R_j)}\left(\sum_{x_i} |R_j^D[X_i = x_i]|\right)^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)$$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of $\text{Eval}(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

$$T_i(\boldsymbol{D}_{\text{residual}}) = T_{\text{intersect}} + \sum_{x_i} T_{i+1}(\boldsymbol{D}_{\text{residual}}[X_i = x_i])$$

$$\overset{\text{induction}}{=} T_{\text{intersect}} + O\left(\sum_{x_i}\left(\prod_{j:X_i \in \texttt{Vars}(R_j)} |R_j^D[X_i = x_i]|^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)\right)$$

$$\overset{\texttt{Hölder}}{\leq} T_{\text{intersect}} + O\left(\prod_{j:X_i \in \texttt{Vars}(R_j)}\left(\sum_{x_i} |R_j^D[X_i = x_i]|\right)^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)$$

$$= T_{\text{intersect}} + O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$$

Runtime of GJ

Theorem For any fractional edge cover $\boldsymbol{w}$, $\boxed{T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)}$

Proof Let $T_i(\boldsymbol{D}_{\text{residual}})$:= runtime of Eval$(\boldsymbol{D}_{\text{residual}}, (x_1, \ldots, x_{i-1}), i)$

$$T_i(\boldsymbol{D}_{\text{residual}}) = T_{\text{intersect}} + \sum_{x_i} T_{i+1}(\boldsymbol{D}_{\text{residual}}[X_i = x_i])$$

$$\overset{\text{induction}}{=} T_{\text{intersect}} + O\left(\sum_{x_i}\left(\prod_{j:X_i \in \texttt{Vars}(R_j)} |R_j^D[X_i = x_i]|^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)\right)$$

$$\overset{\texttt{Hölder}}{\leq} T_{\text{intersect}} + O\left(\prod_{j:X_i \in \texttt{Vars}(R_j)}\left(\sum_{x_i} |R_j^D[X_i = x_i]|\right)^{w_j} \cdot \prod_{j:X_i \notin \texttt{Vars}(R_j)} |R_j^D|^{w_j}\right)$$

$$= T_{\text{intersect}} + O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right) \overset{\text{need}}{=} O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$$

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$?

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$? $O(\sum_i |S_i| \log |S_i|)$.

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$? $O(\sum_i |S_i| \log |S_i|)$.

If they are already sorted: $O(\sum_i |S_i|)$

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$? $O(\sum_i |S_i| \log |S_i|)$.

If they are already sorted: $O(\sum_i |S_i|)$

What is the AGM bound of $Q(X) = S_1(X) \wedge \cdots \wedge S_k(X)$?

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$? $O(\sum_i |S_i| \log |S_i|)$.

If they are already sorted: $O(\sum_i |S_i|)$

What is the AGM bound of $Q(X) = S_1(X) \wedge \cdots \wedge S_k(X)$? $\min_i |S_i|$

Intersection

Let $S_1, \ldots, S_k$ be sets.

What is the complexity of computing $S_1 \cap \cdots \cap S_k$? $O(\sum_i |S_i| \log |S_i|)$.

If they are already sorted: $O(\sum_i |S_i|)$

What is the AGM bound of $Q(X) = S_1(X) \wedge \cdots \wedge S_k(X)$? $\min_i |S_i|$

We need to compute the intersection $S_1 \cap \cdots \cap S_k$ in time $O(\min_i |S_i|)$.

Optimal Intersection

$Q(X) = S_1(X) \wedge \cdots \wedge S_k(X)$
$AGM(Q) = \min_j |S_j|$.

Assume the sets $S_1, \ldots, S_k$ are sorted.

Algorithm: iterate over the smallest set S_i, probe in the rest.

$\boxed{T = \tilde{O}(\min_i |S_i|)}$ where $\tilde{O}$ means "plus a log factor"

Runtime of GJ

To finishing the proof of $T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$,

we need to show $T_{\text{intersect}} = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$

Runtime of GJ

To finishing the proof of $T_{GJ}(\boldsymbol{D}) = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$,

we need to show $T_{\text{intersect}} = O\left(\prod_{j=1,m} |R_j^D|^{w_j}\right)$

$$T_{\text{intersect}} = O\left(\min_{j:X_i \in \text{Vars}(R_j)} |R_j|\right) \overset{\boldsymbol{w} \text{ covers } X_i}{=} O\left(\prod_{j:X_i \in \text{Vars}(R_j)} |R_j|^{w_j}\right) \leq O\left(\prod_{j=1,m} |R_j|^{w_j}\right)$$

Discussion

- GJ is different paradigm from query plans using binary joins: difficult to integrate in a general DBMS [Freitag et al., 2020, Wang et al., 2024].
- GJ runs in time $O(AGM)$ for ANY choice of variable order.
- In practice, the variable order does affect the runtime significantly [Wang et al., 2023].

Extensions of the AGM Bound

Extensions

- Extensions of the AGM bound:
  - Add functional dependencies and domain cardinalities.
  - Apply to projections (i.e. Group-by)
- Extension of GJ to iterate over tuples instead of values.

Domain Sizes

If $|\mathsf{Dom}(X)|$ is known, simply add it to the hypergraph.

Domain Sizes

If $|\mathsf{Dom}(X)|$ is known, simply add it to the hypergraph.

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

We know $|R|, |S|, |T|, |\mathsf{Dom}(X)|$.

Domain Sizes

If $|\mathsf{Dom}(X)|$ is known, simply add it to the hypergraph.

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

We know $|R|, |S|, |T|, |\mathsf{Dom}(X)|$.

$$Q(X,Y,Z) = A(X) \wedge R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$$

Domain Sizes

If $|\mathsf{Dom}(X)|$ is known, simply add it to the hypergraph.

$$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X).$$

We know $|R|, |S|, |T|, |\mathsf{Dom}(X)|$.

$$Q(X,Y,Z) = A(X) \wedge R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$$

Fractional edge covers: $(0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ and $(1,0,1,0)$

$$\boxed{AGM(Q) = \min((|R| \cdot |S| \cdot |T|)^{1/2}, |A| \cdot |S|)}$$

Simple Functional Dependencies

Given FDs, $|Q| \ll AGM(Q)$.

E.g. $R(X,Y) \wedge S(Y,Z)$: N^2 becomes N when $Y \rightarrow Z$.

Simple Functional Dependencies

Given FDs, $|Q| \ll AGM(Q)$.

E.g. $R(X,Y) \wedge S(Y,Z)$: N^2 becomes N when $Y \rightarrow Z$.

$\boldsymbol{U} \rightarrow \boldsymbol{V}$ is simple if $|\boldsymbol{U}| = 1$.

Simple Functional Dependencies

Given FDs, $|Q| \ll AGM(Q)$.

E.g. $R(X,Y) \wedge S(Y,Z)$: N^2 becomes N when $Y \rightarrow Z$.

$\boldsymbol{U} \rightarrow \boldsymbol{V}$ is simple if $|\boldsymbol{U}| = 1$.

Method [Khamis et al., 2016]:

- Expand Q to Q^+ by replacing each atom $R(\boldsymbol{Y})$ with $R'(\boldsymbol{Y}^+)$.
- Return $AGM(Q^+)$.
- This bound is tight. Proof: very useful exercise.

Example

$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$

Fractional edge covers: $(1,1,0), (1,0,1), (0,1,1), (1/2,1/2,1/2)$

$$\boxed{|Q| \leq \min(|R| \cdot |S|, |R| \cdot |T|, |S| \cdot |T|, \sqrt{|R| \cdot |S| \cdot |T|})}$$

Example

$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$
Fractional edge covers: $(1,1,0), (1,0,1), (0,1,1), (1/2,1/2,1/2)$

$$\boxed{|Q| \leq \min(|R| \cdot |S|, |R| \cdot |T|, |S| \cdot |T|, \sqrt{|R| \cdot |S| \cdot |T|})}$$

Assume that $S.Y$ is a key: $Y \rightarrow Z$

Example

$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$
Fractional edge covers: $(1,1,0), (1,0,1), (0,1,1), (1/2,1/2,1/2)$

$$\boxed{|Q| \leq \min(|R| \cdot |S|, |R| \cdot |T|, |S| \cdot |T|, \sqrt{|R| \cdot |S| \cdot |T|})}$$

Assume that $S.Y$ is a key: $Y \rightarrow Z$
$Q^+(X,Y,Z) = R'(X,Y,Z) \wedge S(Y,Z) \wedge T(Z,X)$

Example

$Q(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X)$
Fractional edge covers: $(1,1,0), (1,0,1), (0,1,1), (1/2,1/2,1/2)$

$$\boxed{|Q| \leq \min(|R| \cdot |S|, |R| \cdot |T|, |S| \cdot |T|, \sqrt{|R| \cdot |S| \cdot |T|})}$$

Assume that $S.Y$ is a key: $Y \rightarrow Z$
$Q^+(X,Y,Z) = R'(X,Y,Z) \wedge S(Y,Z) \wedge T(Z,X)$
Fractional edge covers: $(1,0,0), (0,1,1)$

$$\boxed{|Q| \leq \min(|R|, |S| \cdot |T|)}$$

A Generalization of GJ

Was already described in [Ngo et al., 2013]!

Instead of iterating over one variable X_i, iterate over a tuple $\boldsymbol{X}$.

- Compute query $Q|_{\boldsymbol{X}}$ where each R_j is projected on $\boldsymbol{X} \cap \text{Vars}(R_j)$.
- For each tuple $\boldsymbol{x}$, evaluate the query on $\boldsymbol{D}[\boldsymbol{X} = \boldsymbol{x}]$.

Example: $Q(X, Y, Z) = R(X, Y) \wedge S(Y, Z) \wedge T(Z, X)$:

- Compute $Q_1(X, Y) = R(X, Y) \wedge S(Y, -) \wedge T(-, X)$
- For each $(x, y) \in Q_1$, compute $Q_2(Z) = R(x, y) \wedge S(y, Z) \wedge T(Z, x)$.

A Generalization of GJ

Algorithm 11: Eval($Q_{\text{residual}}, \boldsymbol{D}_{\text{residual}}, \boldsymbol{z}$)

if $Vars(Q_{residual}) = \emptyset$ **then** emit $\boldsymbol{z}$;
else
  Let $(\boldsymbol{X}, \boldsymbol{Y})$:= arbitrary partition of $\texttt{Vars}(Q_{\text{residual}})$;
  for $\boldsymbol{x} \in Eval(Q_{residual}|_{\boldsymbol{X}}, \Pi_{\boldsymbol{X}}(\boldsymbol{D}_{residual}), ())$ **do**
    Eval($Q_{\text{residual}}|_{\boldsymbol{Y}}, \Pi_{\boldsymbol{Y}}(\boldsymbol{D}_{\text{residual}}[\boldsymbol{X} = \boldsymbol{x}]), (\boldsymbol{z}, \boldsymbol{x})$);

"Standard" GJ partitions $\{X_i, X_{i+1}, \ldots, X_n\}$ into $\{X_i\}$ and $\{X_{i+1}, \ldots, X_n\}$.

Proof of $T(\boldsymbol{D}) = O(AGM(Q, \boldsymbol{D}))$ is the same: replace Hölder with Friedgut.

Discussion

- The expansion procedure Q^+ is easy, but limited only to simple FDs: $AGM(Q^+)$ is always an upper bound on Q's output, but may not be tight.

- The Extension of GJ is quite practical, because we can iterate over tuples.

The Trie

Motivation

Before we run GJ, we must store the database in a structure that allows us to compute the intersection in time $O\left(\min_j |R_j^D|\right)$

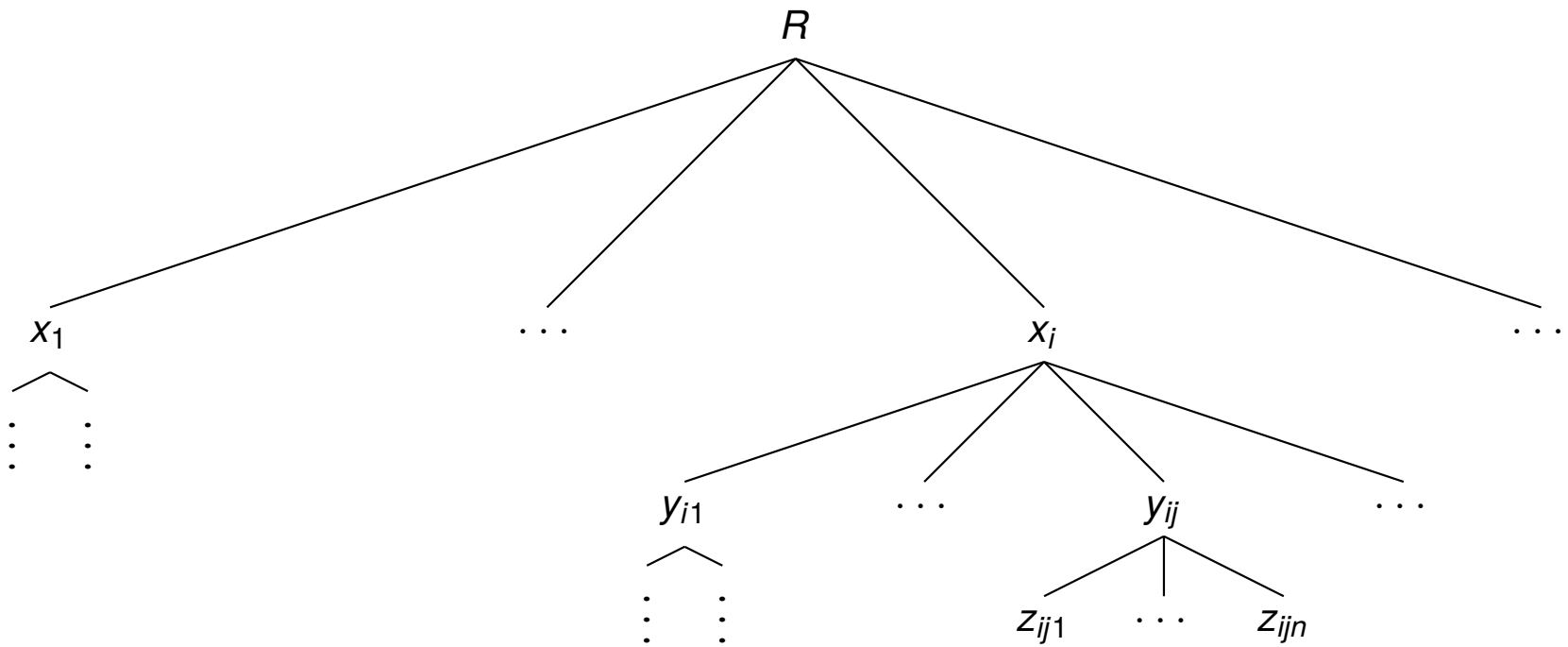
Options:

- Store each R_j^D in a Trie [Veldhuizen, 2014].
- Sort each R_j^D.
- Build a B+ tree for R_j^D.

Storing a Relation to a Trie

Fix the order of the attributes of the relation R, e.g. $R(X, Y, Z)$.

Sort $R.X$: $x_1 < x_2 < \cdots$
For each x_i, sort $R[X = x_i].Y$: $y_1 < y_2 < \cdots$
For each x_i, y_j, sort $R[X = x_i, Y = y_j].Z$: $z_1 < z_2 < \cdots$



Discussion

- Trie indices are computed offline.
- When variable order is incompatible with the query, then we must re-index. Expensive!
- Discuss in class how GJ can run in time faster than $\max(|R|, |S|, |T|)$.

The Fractional Tree Width

The Fractional Treewidth

$G = (V, E)$ a hypergraph.

Let $TD = (T, \chi)$ be a tree decomposition of Q. It's fractional width is:

$$\boxed{\text{FTW}(TD) := \max_{u \in \text{Nodes}(T)} \rho^*(\chi(u))}$$

where $\rho^*(\chi(u))$ is the fractional edge covering number of the hypergraph $(\chi(u), \{e \cap \chi(u) \mid e \in E\})$

The fractional tree width of G is:

$$\boxed{\text{FTW}(G) := \min_{TD} \text{FTW}(TD)}$$

Query Evaluation

$$Q(X_1, \ldots, X_n) = R_1(\boldsymbol{Y}_1) \wedge \cdots \wedge R_m(\boldsymbol{Y}_m) \qquad |R_1^D|, \ldots, |R_m^D| \leq N.$$

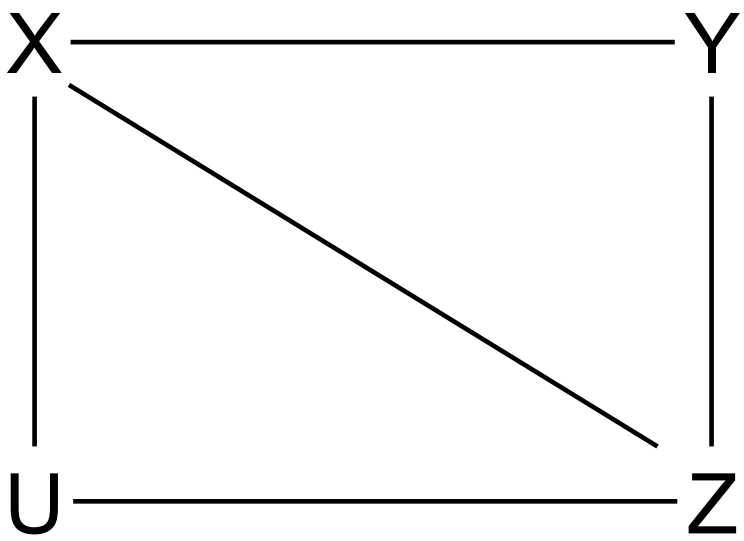
To compute $Q(\boldsymbol{D})$:

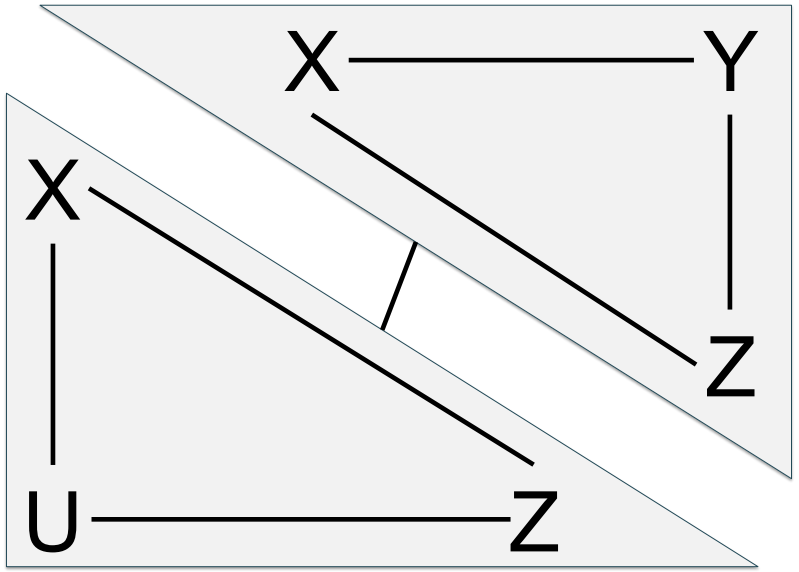
- Decompose Q into a tree (T, χ).
- Materialize each bag $\chi(u)$ using GJ.
- Run Yannakakis' algorithm on the acyclic query consisting of all bags.

Runtime: $\boxed{T(Q, \boldsymbol{D}) = O\left(N^{\text{FTW}(Q)} + \text{OUT}\right)}$

Example

$$Q(X,Y,Z,U) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,U) \wedge K(U,X) \wedge L(X,Z)$$





$\text{FTW} = 3/2$, runtime $T = O\left(N^{3/2} + \text{OUT}\right)$

Discussion

- The fractional treewidth $\max_u \rho^*(\chi(u))$ replaces the hypertreewidth $\max_u \rho(\chi(u))$, because GJ is a better algorithm than joining the atoms of an edge cover.

- Interesting distinction:
  - For HTW we use as few atoms in each bag u as possible, because $\rho(\chi(u))$ is the number atoms that cover $\chi(u)$.
  - For FTW we use ALL atoms that contain some variable of $\chi(u)$, because the more we use the smaller ρ^* becomes.

- But finding a good TD is difficult:
  Given G, checking whether $\text{FTW}(G) \leq 2$ is NP-complete.
  I am not aware of any good heuristics.


Freitag, M. J., Bandle, M., Schmidt, T., Kemper, A., and Neumann, T. (2020).
Adopting worst-case optimal joins in relational database systems.
Proc. VLDB Endow., 13(11):1891–1904.


Khamis, M. A., Ngo, H. Q., and Suciu, D. (2016).
Computing join queries with functional dependencies.
In Milo, T. and Tan, W., editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 327–342. ACM.


Ngo, H. Q., Porat, E., Ré, C., and Rudra, A. (2012).
Worst-case optimal join algorithms: [extended abstract].
In Benedikt, M., Krötzsch, M., and Lenzerini, M., editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 37–48. ACM.


Ngo, H. Q., Ré, C., and Rudra, A. (2013).
Skew strikes back: new developments in the theory of join algorithms.
SIGMOD Rec., 42(4):5–16.


Veldhuizen, T. L. (2014).
Triejoin: A simple, worst-case optimal join algorithm.
In Schweikardt, N., Christophides, V., and Leroy, V., editors, *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 96–106. OpenProceedings.org.


Wang, J., Trummer, I., Kara, A., and Olteanu, D. (2023).
ADOPT: adaptively optimizing attribute orders for worst-case optimal join algorithms via reinforcement learning.
Proc. VLDB Endow., 16(11):2805–2817.


Wang, Y. R., Willsey, M., and Suciu, D. (2024).
From binary join to free join.
SIGMOD Rec., 53(1):25–31.