RelationalAI

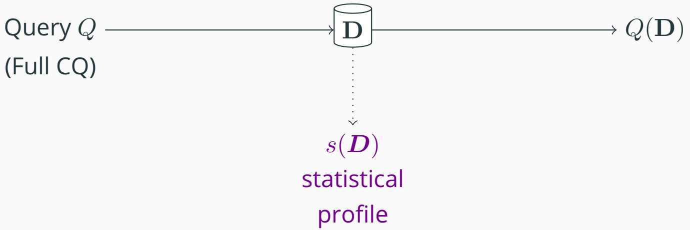
Logic & Algorithms in DB & AI – 2023

Worst-Case Optimal Joins

Hung Q. Ngo

Outline

Worst-Case Optimal Join (WCOJ) Algorithm



Definition

A "worst-case optimal" join algorithm is an algorithm computing $Q(\boldsymbol{D})$ in time

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|\right)$$

$\tilde{O}$ hides log and query-dependent factors

Hierarchy of Join Algorithms

JAAT	Join at a time
VAAT	Variable at a time
IAAT	Inequality at a time

The Algorithm is in the Pudding

Proof $\Longrightarrow$ Algorithm!

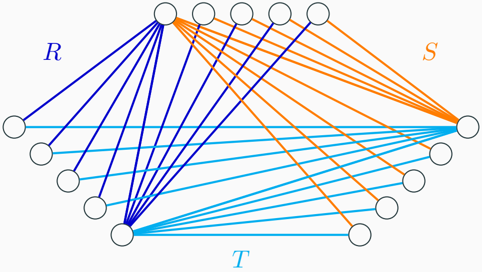
Hierarchy of Join Algorithms

$$
\begin{aligned}
\text{(not achievable ?)} \quad & \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')| \\
\text{(not achievable ?)} \quad & \leq \text{entropic-bound}(Q, s) \\
\text{(IAAT)} \quad & \leq \text{polymatroid-bound}(Q, s) && \text{PANDA} \\
\text{(VAAT)} \quad & \leq \text{chain-bound}(Q, s, \sigma) && \text{NPRR, LFTJ, GJ} \\
\text{(VAAT)} \quad & \leq \text{agm-bound}(Q, s) && \text{NPRR, LFTJ, GJ} \\
\text{(JAAT)} \quad & \leq \text{integral-edge-cover}(Q, s) && \text{Binary Plans}
\end{aligned}
$$

Runtime = $\tilde{O}(2^{\text{bound}})$

Outline

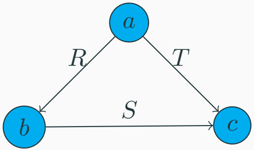
Example: JAAT Query Plans Are Sub-Optimal

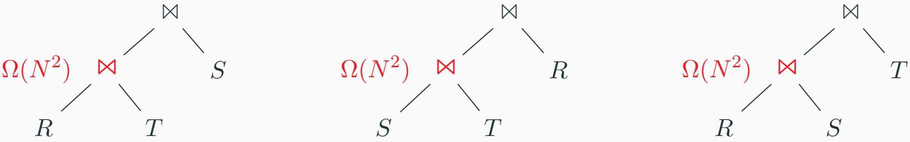


$$|R| = |S| = |T| = 2N - 1$$

$$Q_\Delta(a,b,c) = R(a,b) \wedge S(b,c) \wedge T(a,c)$$

$$\sup_{\mathbf{D}' \models \mathsf{DC}(\boldsymbol{D})} |Q_\Delta(\mathbf{D}')| = O(N^{1.5})$$





Outline

Triangle Query

$$Q_\triangle(A,B,C) \leftarrow R(A,B), S(B,C), T(A,C)$$

AGM-bound for Q: (E.g. num triangles in a graph $\leq |E|^{3/2}$)

$$|Q_\triangle| \leq |R|^{\lambda_R} \cdot |S|^{\lambda_S} \cdot |T|^{\lambda_T}$$

whenever $\boldsymbol{\lambda} = (\lambda_R, \lambda_S, \lambda_T)$ is a fractional edge cover for the triangle:

$$\lambda_R + \lambda_S \geq 1$$
$$\lambda_R + \lambda_T \geq 1 \qquad \boldsymbol{\lambda} \geq \mathbf{0}$$
$$\lambda_S + \lambda_T \geq 1$$

Pick $\boldsymbol{\lambda}$ to minimize the bound.

Triangle Query: AGM Bound from Hölder Inequality

Consider a "section" of this query on a given value $a \in \text{Dom}(A)$:

$$Q_\triangle(a, B, C) \leftarrow R(a, B), S(B, C), T(a, C)$$

Need (b, c) in the intersection $S \cap (\sigma_{A=a}R \times \sigma_{A=a}T)$, thus

$$\begin{aligned} |\sigma_{A=a}Q_\triangle| &\leq \min\{|S|, |\sigma_{A=a}R| \cdot |\sigma_{A=a}T|\} \\ &\leq |S|^{\lambda_S} \cdot (|\sigma_{A=a}R| \cdot |\sigma_{A=a}T|)^{1-\lambda_S} \\ &\leq |S|^{\lambda_S} \cdot |\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \end{aligned}$$

Triangle Query: AGM Bound Based on Hölder Inequality

Iterate over all possible values of a:

$$
\begin{aligned}
|Q_\triangle| &= \sum_a |\sigma_{A=a} Q_\triangle| \\
&\le \sum_a |S|^{\lambda_S} \cdot |\sigma_{A=a} R|^{\lambda_R} \cdot |\sigma_{A=a} T|^{\lambda_T} \\
&= |S|^{\lambda_S} \cdot \sum_a |\sigma_{A=a} R|^{\lambda_R} \cdot |\sigma_{A=a} T|^{\lambda_T} \\
\text{(Hölder)} &\le |S|^{\lambda_S} \cdot \Big(\sum_a |\sigma_{A=a} R|\Big)^{\lambda_R} \cdot \Big(\sum_a |\sigma_{A=a} T|\Big)^{\lambda_T} \\
&= |S|^{\lambda_S} \cdot |R|^{\lambda_R} \cdot |T|^{\lambda_T}
\end{aligned}
$$

$$Q_\triangle(A,B,C) \leftarrow R(A,B), S(B,C), T(A,C)$$

Algorithm 1: based on Hölder's inequality proof

for $a \in \pi_A R \cap \pi_A T$ **do**
  for $b \in \pi_B \sigma_{A=a} R \cap \pi_B S$ **do**
    for $c \in \pi_C \sigma_{B=b} S \cap \pi_C \sigma_{A=a} T$ **do**
      Report (a,b,c);

In English

- For each $a \in \pi_A R \cap \pi_A T$, enumerate $(a,b,c) \in \sigma_{A=a} Q_\triangle$

Triangle Query: VAAT is in the Pudding

Computing this "section" of the query on a given value $a \in \text{Dom}(A)$

$$Q_\triangle(a, B, C) \leftarrow R(a, B), S(B, C), T(a, C)$$

is to compute the intersection $S \cap (\sigma_{A=a}R \times \sigma_{A=a}T)$, which can be done in time

$$\tilde{O}\left(\min\{|S|, |\sigma_{A=a}R| \cdot |\sigma_{A=a}T|\}\right) \leq \tilde{O}\left(|\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \cdot |S|^{\lambda_S}\right)$$

Overall, the algorithm runs in time (Modulo $\tilde{O}(N)$ pre-processing)

$$\tilde{O}\left(\sum_a |\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \cdot |S|^{\lambda_S}\right) = \tilde{O}\left(|S|^{\lambda_S} \cdot |R|^{\lambda_R} \cdot |T|^{\lambda_T}\right)$$

Full Conjunctive Query

$$Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}} R_S(S)$$

Query hypergraph $\mathcal{H} = (V, \mathcal{E})$

AGM-bound for Q:

Assuming only cardinality constraints $(\emptyset, S, |R_S|)$

$$|Q| \leq \prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}$$

whenever $\boldsymbol{\lambda} = (\lambda_S)_{S \in \mathcal{E}}$ is a fractional edge cover for $\mathcal{H}$:

$$\forall v \in V \ : \ \sum_{S \in \mathcal{E}, v \in S} \lambda_S \geq 1 \qquad \boldsymbol{\lambda} \geq \mathbf{0}.$$

Pick $\boldsymbol{\lambda}$ to minimize the bound.

Full Conjunctive Query: AGM Bound from Hölder Inequality

Consider a "section" of this query on a given value $a \in \text{Dom}(A)$:

$$\sigma_{A=a} Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}, A \notin S} R_S(S) \wedge \bigwedge_{S \in \mathcal{E}, A \in S} \sigma_{A=a} R_S(S)$$

Now iterate over all possible values of a:

$$|Q| = \sum_a |\sigma_{A=a} Q_\triangle| \leq \sum_a \prod_{S \in \mathcal{E}, A \notin S} |R_S|^{\lambda_S} \cdot \prod_{S \in \mathcal{E}, A \in S} |\sigma_{A=a} R_S|^{\lambda_S}$$

$$\text{(Hölder's inequality)} \leq \prod_{S \in \mathcal{E}, A \notin S} |R_S|^{\lambda_S} \cdot \prod_{S \in \mathcal{E}, A \in S} \left(\sum_a |\sigma_{A=a} R_S| \right)^{\lambda_S}$$

$$= \prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}.$$

Full Conjunctive Query: VAAT Algorithm

[NPRR 12] [Veldhuizen 14]

$$Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}} R_S(S)$$

Algorithm 2: based on Hölder's inequality proof

for $a \in \bigcap_{S \in \mathcal{E}, A \in S} \pi_A R_S$ **do**

  Recursively solve the query section $\sigma_{A=a} Q$ (on variables $V - \{A\}$);

  Report $\{a\} \times \sigma_{A=a} Q$

Runtime $\tilde{O}\left(\sum_{S \in \mathcal{E}} |R_S|\right) + \tilde{O}\left(\prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}\right)$.

Proof: straightforward.

Full Conjunctive Query: Chain-Bound, VAAT Algorithm

For degree constraints *beyond* cardinality constraints

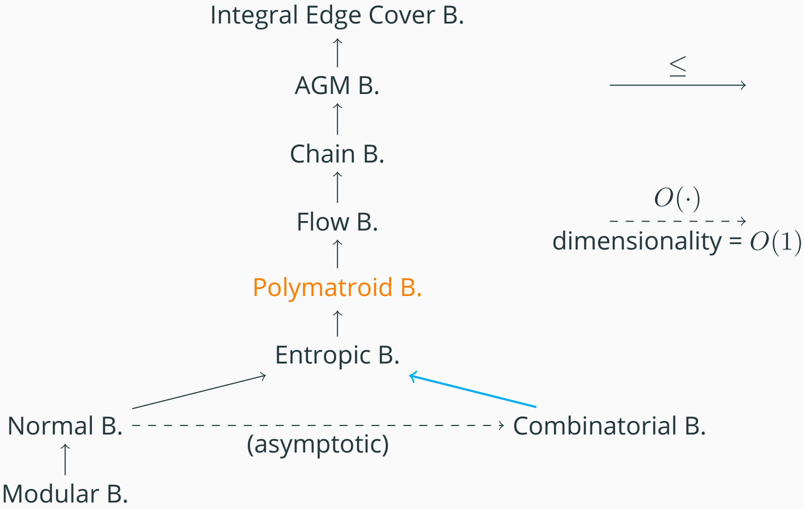
- The AGM bound does not apply.
- We use the chain-bound instead.

Find a variable ordering σ

- Arbitrary degree constraint set DC
- Runtime predicted by $\text{chain-bound}(\text{DC}, \sigma)$ Tight for acyclic DC
- VAAT algorithm meeting the chain-bound; similar analysis

Outline

Recall: Bound Hierarchy



Recall: The Entropic and Polymatroid Bounds

Γ_n is the set of n-dim polymatroids. $h(Y|X) = h(Y) - h(X), X \subseteq Y$.

Theorem

If $s(\boldsymbol{D})$ contains only degree constraints, then

$$\log \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')| \leq \sup_{h \in \overline{\Gamma}^*_n \cap DC} h(V) \leq \max_{h \in \Gamma_n \cap DC} h(V)$$

where DC is the set of linear constraints of the form

$$h(Y|X) \leq \log N$$

for each degree constraint (X, Y, N).

More explicitly,

$$
\begin{array}{llll}
\max & h(V) & & \text{dual vars} \\
\text{s.t.} & h(Y) - h(X) \leq \log N, & (X, Y, N) \in \text{DC} & \delta_{Y|X} \\
& h(I \cup J|J) - h(I|I \cap J) \leq 0, & I \perp J & \sigma_{I,J} \\
& h(X) - h(Y) \leq 0, & \emptyset \neq X \subset Y \subseteq V & \mu_{Y|X} \\
& h(Z) \geq 0, & \emptyset \neq Z \subseteq V. &
\end{array}
$$

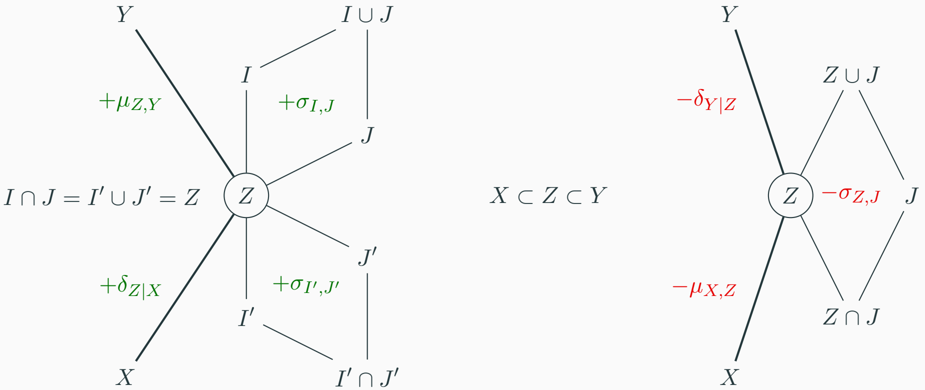
$I \perp J$ means $I \not\subseteq J$ and $J \not\subseteq I$.

$$
\begin{aligned}
\min \quad & \sum_{(X,Y,N)\in \mathsf{DC}} \log N \cdot \delta_{Y|X} \\
\text{s.t.} \quad & \mathsf{excess}(V) \geq 1, \\
& \mathsf{excess}(Z) \geq 0, \qquad \emptyset \neq Z \subseteq V. \\
& (\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu}) \geq \mathbf{0}.
\end{aligned}
$$

where, for any $\emptyset \neq Z \in 2^V$, the quantity $\mathsf{excess}(Z)$ is defined by

$$
\begin{aligned}
\mathsf{excess}(Z) := & \sum_{X:(X,Z)\in \mathsf{DC}} \delta_{Z|X} - \sum_{Y:(Z,Y)\in \mathsf{DC}} \delta_{Y|Z} + \sum_{\substack{I \perp J \\ I \cap J = Z}} \sigma_{I,J} \\
& + \sum_{\substack{I \perp J \\ I \cup J = Z}} \sigma_{I,J} - \sum_{J:J \perp Z} \sigma_{Z,J} - \sum_{X:X \subset Z} \mu_{X,Z} + \sum_{Y:Z \subset Y} \mu_{Z,Y}.
\end{aligned}
$$

Contributions of coefficients to $\text{excess}(Z)$



Definition

Given $\boldsymbol{\delta} \geq 0$, the following is a Shannon-flow inequality if it holds for all $h \in \Gamma_n$:

$$h(V) \leq \sum_{(X,Y,N) \in \mathsf{DC}} \delta_{Y|X} \cdot (h(Y) - h(X))$$

- $\boldsymbol{\delta}$ defines a Shannon-flow inequality iff $\exists(\boldsymbol{\sigma}, \boldsymbol{\mu})$ s.t. $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ is dual-feasible.
- If DC contains only cardinality constraints $(X = \emptyset, Y, N)$, then $\boldsymbol{\delta}$ defines a Shannon-flow inequality iff it is a fractional edge cover of the query hypergraph.

Shearer's Lemma!

Example: Shannon-Flow Inequality for Triangle Query

$$h(A,B,C) \leq \frac{1}{2}(h(A,B)+h(B,C)+h(A,C))$$

A step-by-step proof: [Radhakrishnan 2003]

$$\begin{aligned}
& h(A,B)+h(A,C)+h(B,C) \\
\text{(decomposition)} &= h(A)+h(B|A)+h(B,C)+h(A,C) \\
\text{(sub-modularity)} &\geq (h(A|B,C)+h(B,C))+(h(B|A)+h(A,C)) \\
\text{(composition)} &= h(A,B,C)+(h(B|A)+h(A,C)) \\
\text{(sub-modularity)} &\geq h(A,B,C)+(h(B|A,C)+h(A,C)) \\
\text{(composition)} &= h(A,B,C)+h(A,B,C)
\end{aligned}$$

Example: Another Shannon-Flow Inequality

$$h(ABCD) \leq \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

$$\begin{aligned}
& h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(decomposition)} &= h(AB) + h(B) + h(C|B) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(B) + h(C|B) + h(CD|B) + h(D|AC) + h(A|BD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BCD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|AB) + h(D|AC) + h(ABCD) \\
\text{(composition)} &= h(ABC) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(ABC) + h(D|ABC) + h(ABCD) \\
\text{(composition)} &= h(ABCD) + h(ABCD).
\end{aligned}$$

From LP-duality, there exists $\boldsymbol{\delta} \geq \mathbf{0}$ s.t.

$$\text{polymatroid-bound} := \max\{h(V) \mid h \in C \cap \Gamma_n\} = \sum_{(X,Y,N) \in \text{DC}} \delta_{Y|X} \log N,$$

and for these $\boldsymbol{\delta}$, from Farkas's lemma we have

$$h(V) \leq \sum_{(X,Y,N) \in \text{DC}} \delta_{Y|X} \cdot h(Y|X), \quad \forall h \in \Gamma_n$$

Proof Sequence for a Shannon-Flow Inequality

$$h(V) \leq \sum_{(X,Y,N)\in \text{DC}} \delta_{Y|X} \cdot h(Y|X)$$

A proof sequence is a conversion from RHS to LHS using a sequence of steps of the form

(In)equality	Steps ($X \subseteq Y$)
$h(X) + h(Y\|X) = h(Y)$	$h(X) + h(Y\|X) \to h(Y)$
$h(Y) = h(X) + h(Y\|X)$	$h(Y) \to h(X) + h(Y\|X)$
$h(Y) \geq h(X)$	$h(Y) \to h(X)$
$h(Y\|X) \geq h(Y \cup Z\|X \cup Z)$	$h(Y\|X) \to h(Y \cup Z\|X \cup Z)$

Existence of Proof Sequence

Lemma (ANS 2017)

There is a proof sequence for every Shannon-flow inequality. (The length is at most doubly exponential in $|V|$).

The Shannon-flow inequality is a linear combination of dual constraints; the proof sequence is more stringent than that.

Outline

Overview

JAAT Algorithm

VAAT Algorithm

Shannon-Flow Inequalities

IAAT Algorithm

Open Problems – Research Directions

One Inequality At A Time (IAAT)

There is an algorithm (called PANDA) that converts a proof sequence $\rightarrow$ an efficient algorithm to answer the original query

Steps ($X \subseteq Y$)	Relational Operator
$h(X) + h(Y\|X) \rightarrow h(Y)$	(join)
$h(Y) \rightarrow h(X) + h(Y\|X)$	(data partition)
$h(Y) \rightarrow h(X)$	(projection)
$h(Y\|X) \rightarrow h(Y \cup Z\|X \cup Z)$	(NOP)

Theorem

PANDA solves any conjunctive query Q in time $\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\textit{polymatroid bound}})$

Example: PANDA for Triangle Query

$$Q(A,B,C) \leftarrow R(A,B), S(B,C), T(A,C)$$

$$R^{\text{heavy}}(A,B) = \{(a,b) \ : \ |\sigma_{A=a}R| > \sqrt{N}\}$$

$$R^{\text{light}}(A,B) = \{(a,b) \ : \ |\sigma_{A=a}R| \leq \sqrt{N}\}$$

Algorithm is in the pudding!

$$\begin{aligned}
h(A,B) + h(A,C) + h(B,C) & \quad R(A,B), S(B,C), T(A,C) \\
= h(A) + h(B|A) + h(B,C) + h(A,C) & \quad R^{\text{heavy}}(A,B), R^{\text{light}}(A,B), S(B,C), T(A,C) \\
\geq (h(A|B,C) + h(B,C)) + (h(B|A) + h(A,C)) & \quad R^{\text{heavy}}(A,B), R^{\text{light}}(A,B), S(B,C), T(A,C) \\
= h(A,B,C) + (h(B|A) + h(A,C)) & \quad I^{\text{heavy}}(A,B,C), R^{\text{light}}(A,B), T(A,C) \\
\geq h(A,B,C) + (h(B|A,C) + h(A,C)) & \quad I^{\text{heavy}}(A,B,C), R^{\text{light}}(A,B), T(A,C) \\
= h(A,B,C) + h(A,B,C) & \quad I^{\text{heavy}}(A,B,C), I^{\text{light}}(A,B,C).
\end{aligned}$$

Example: PANDA for Triangle Query

The real query plan:

$$
\begin{aligned}
& R(A, B) \wedge S(B, C) \wedge T(A, C) \\
& = (R^{\text{heavy}}(A, B) \vee R^{\text{light}}(A, B)) \wedge S(B, C) \wedge T(A, C) \\
& = (R^{\text{heavy}}(A, B) \wedge S(B, C) \wedge T(A, C)) \vee (R^{\text{light}}(A, B) \wedge S(B, C) \wedge T(A, C)) \\
& = (R^{\text{heavy}}(A, B) \wedge S(B, C) \wedge T(A, C)) \vee (R^{\text{light}}(A, B) \wedge S(B, C) \wedge T(A, C)) \\
& = I^{\text{heavy}}(A, B, C) \wedge T(A, C) \vee I^{\text{light}}(A, B, C) \wedge S(B, C).
\end{aligned}
$$

- Note that $|I^{\text{heavy}}(A, B, C)| \leq N^{3/2}$ and $|I^{\text{light}}(A, B, C)| \leq N^{3/2}$.
- Overall runtime is $\tilde{O}(N^{3/2})$.

Example: PANDA for a More Interesting Example [ANS 17]

$$Q(A,B,C,D) \leftarrow R(A,B) \wedge S(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A$$

From the Shannon-flow inequality:

$$h(ABCD) \leq \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

we know

$$\log_2 |Q| \leq \frac{1}{2}[\log_2 |R| + \log_2 |S| + \log_2 |T| + 0 + 0]$$

or

$$|Q| \leq \sqrt{|R||S||T|}$$

Example: Another Shannon-Flow Inequality

$$h(ABCD) \leq \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

$$\begin{aligned}
& h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(decomposition)} &= h(AB) + h(B) + h(C|B) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(B) + h(C|B) + h(CD|B) + h(D|AC) + h(A|BD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BCD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|AB) + h(D|AC) + h(ABCD) \\
\text{(composition)} &= h(ABC) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(ABC) + h(D|ABC) + h(ABCD) \\
\text{(composition)} &= h(ABCD) + h(ABCD).
\end{aligned}$$

Example : PANDA for a More Interesting Example

$$
\begin{aligned}
& R(A,B) \wedge S(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&= R(A,B) \wedge S^{\text{heavy}}(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&\vee R(A,B) \wedge S^{\text{light}}(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&= R(A,B) \wedge S^{\text{heavy}}(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&\vee R(A,B) \wedge S^{\text{light}}(B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&= R(A,B) \wedge I^{\text{heavy}}(B,C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&\vee I^{\text{light}}(A,B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&= R(A,B) \wedge I^{\text{heavy}}(B,C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&\vee I^{\text{light}}(A,B,C) \wedge T(C,D) \wedge \text{hash}(A,C) = D \wedge \text{hash}(B,D) = A \\
&= R(A,B) \wedge J^{\text{heavy}}(A,B,C,D) \wedge \text{hash}(A,C) = D \\
&\vee J^{\text{light}}(A,B,C,D) \wedge T(C,D) \wedge \text{hash}(B,D) = A.
\end{aligned}
$$

Example : PANDA for a More Interesting Example

Main question

How to define S^{heavy} and S^{light} so that runtime is $\tilde{O}(2^{h^*(A,B,C,D)})$

$$S^{\text{heavy}}(B,C) = \{(b,c) \ : \ |\sigma_{C=c}S| > 2^{h^*(B,C)-h^*(C)}\}$$
$$S^{\text{light}}(B,C) = \{(b,c) \ : \ |\sigma_{C=c}S| \leq 2^{h^*(B,C)-h^*(C)}\}$$

Assuming h^* and $(\boldsymbol{\delta}^*, \boldsymbol{\sigma}^*, \boldsymbol{\mu}^*)$ are primal-dual optimal: ($\delta^*_{CD|\emptyset} > 0$ and $\delta^*_{AB|\emptyset} > 0$)

$$|S^{\text{light}}(B,C) \wedge T(C,D)| \leq 2^{h^*(B,C)-h^*(C)} \cdot 2^{h^*(C,D)} = 2^{h^*(B,C,D)} \leq 2^{h^*(A,B,C,D)}$$
$$|S^{\text{heavy}}(B,C) \wedge R(A,B)| \leq 2^{h^*(C)} \cdot 2^{h^*(A,B)} = 2^{h^*(A,B,C)} \leq 2^{h^*(A,B,C,D)}.$$

$=$ holds because SFI holds with $=$ for h^*.

The Actual PANDA Algorithm (Needs Another Talk)

More complicated because:

- Couldn't prove that every heavy / light copy reaches $h(V)$ eventually.
- Couldn't prove that in the proof sequence we won't ever compose terms which were decomposed in an earlier step

Main ideas to push through:

- A decomposition $h(A, B) \to h(A) + h(B|A)$ corresponds to partitioning R into logarithmically many "uniform" parts.
- Essentially, each each part, both the heavy condition and the light condition are satisfied.
- Induct on logarithmically many subproblems, including constructing a new proof sequence for each of them

The Actual PANDA Algorithm

PANDA runs in Time

$$\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\text{polymatroid bound for } Q}) = \tilde{O}(N + \text{poly}(\log N) \cdot \sup_{\boldsymbol{D'} \models s(\boldsymbol{D})} |Q(\boldsymbol{D'})|)$$

Outline

Open Problems

- Bound the minimum proof sequence length
  - Educated conjecture: polynomial($|DC|$)
- Remove polylog factor from runtime of PANDA
- Is there a more natural algorithm?
- Exploiting conditional independence

Main References

Radakrishnan 03 J. Radhakrishnan, Entropy and counting, in Computational mathematics, modelling and algorithms (J. C. Misra, editor), Narosa, 2003, 146–168.

GM 06 Martin Grohe, Dániel Marx: Constraint solving via fractional edge covers. SODA 2006: 289-298

AGM 08 Albert Atserias, Martin Grohe, Dániel Marx: Size Bounds and Query Plans for Relational Joins. FOCS 2008: 739-748

NPRR 12 Hung Q. Ngo, Ely Porat, Christopher Ré, Atri Rudra: Worst-case optimal join algorithms. PODS 2012: 37-48

Veldhuizen 14 Todd L. Veldhuizen: Triejoin: A Simple, Worst-Case Optimal Join Algorithm. ICDT 2014: 96-106

ANS 16 Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu: Computing Join Queries with Functional Dependencies. PODS 2016: 327-342

ANS 17 Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu: What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another? PODS 2017: 429-444

N 18 Hung Q. Ngo: Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. PODS 2018: 111-124

Suciu 23 Dan Suciu: Applications of Information Inequalities to Database Theory Problems. LICS 2023: 1-30

Many Thanks!