Instance-optimal Database Joins

Mahmoud Abo Khamis
Relational AI

Logic and Algorithms in Database Theory and AI Boot Camp
Simons Institute

Aug 21, 2023
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation
    - Reuse Prebuilt DSs for many queries (amortization)
Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size $N$ is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation
    - Reuse Prebuilt DSs for many queries (amortization)
    - Sublinear time is possible
Beyond Worst-case Analysis: Some Models

- Parameterized Complexity
- Adaptive Analysis
- Instance Optimality
- Average-case
- ...


Instance Optimality: Goal

- Given an input instance $I$ to some problem $P$
Instance Optimality: Goal

- Given an input instance $I$ to some problem $P$
  - Find a lower bound $f(I)$ on the runtime of any algorithm $A$ on $I$
Instance Optimality: Goal

- Given an input instance $I$ to some problem $P$
  - Find a lower bound $f(I)$ on the runtime of any algorithm $A$ on $I$
- Design an algorithm $A^*$ whose runtime is $O(m \cdot f(I))$ for every $I$
Instance Optimality: Goal

- Given an input instance $I$ to some problem $P$
  - Find a lower bound $f(I)$ on the runtime of any algorithm $A$ on $I$
- Design an algorithm $A^*$ whose runtime is $O(m \cdot f(I))$ for every $I$
  - $m$ is the optimality ratio
Instance Optimality: General Approach

- Every algorithm must produce a proof $C$ of output correctness (certificate)
Instance Optimality: General Approach

- Every algorithm must produce a proof $\mathcal{C}$ of output correctness (certificate)
- The minimum certificate size $|\mathcal{C}|$ is a lower bound on the runtime
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - $C \leftarrow \emptyset$ (The certificate)
  - While $C$ does not yet prove the output
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - $C \leftarrow \emptyset$ (The certificate)
  - While $C$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - \( C \leftarrow \emptyset \) (The certificate)
  - While \( C \) does not yet prove the output
    - \( Q \leftarrow \) Some query to the input
    - \( C \leftarrow C \cup Q \)
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - $C \leftarrow \emptyset$ (The certificate)
  - While $C$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $C \leftarrow C \cup Q$
    - Show that every certificate $C'$ contains $\geq 1/m$ of $Q$
Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS’03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $\mathcal{C} \leftarrow \mathcal{C} \cup Q$
    - Show that every certificate $\mathcal{C}'$ contains $\geq 1/m$ of $Q$

- Analysis
Fagin et al, JCSS’03: Database aggregation problem

Meta-algorithm

$C \leftarrow \emptyset$ (The certificate)
While $C$ does not yet prove the output

- $Q \leftarrow$ Some query to the input
- $C \leftarrow C \cup Q$
- Show that every certificate $C'$ contains $\geq 1/m$ of $Q$

Analysis

- $|C| \leq m \cdot |C'|$, for any certificate $C'$
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Instance-optimal Set Intersection

- Input: Two sets $R$ and $S$ of numbers

Output: $Q$: 

$$R_x S_p Q$$

$$R^p x Q^S$$

Worst-case Runtime:

$$O(p \min |R|, |S|)$$

Some instances are easier than others.

Instance 1:

$R$ $S$

Instance 2:

$R$ $S$
Instance-optimal Set Intersection

- **Input:** Two sets $R$ and $S$ of numbers
- **Output:** $Q := R \cap S$
Instance-optimal Set Intersection

- Input: Two sets \( R \) and \( S \) of numbers
- Output: \( Q := R \cap S \)
  - \( Q(X) = R(X) \land S(X) \)

Worst-case Runtime: 
\[ O(\min\{|R|, |S|\}) \]

Some instances are easier than others
Instance-optimal Set Intersection

- **Input:** Two sets $R$ and $S$ of numbers
- **Output:** $Q := R \cap S$
  - $Q(X) = R(X) \land S(X)$
- **Worst-case Runtime:** $O(\min(|R|, |S|))$
Instance-optimal Set Intersection

- **Input:** Two sets $R$ and $S$ of numbers
- **Output:** $Q := R \cap S$
  
  $Q(X) = R(X) \land S(X)$

- **Worst-case Runtime:** $O(\min(|R|, |S|))$

- **Some instances are easier than others**

  ![Diagram](image-url)
Instance-optimal Set Intersection

- **Input:** Two sets $R$ and $S$ of numbers
- **Output:** $Q := R \cap S$
  - $Q(X) = R(X) \land S(X)$
- **Worst-case Runtime:** $O(\min(|R|, |S|))$
- **Some instances are easier than others**

![Instance 1](image1)

![Instance 2](image2)
Instance-optimal Set Intersection

- Hwang and Lin, SIAM’72: “Leap-frogging” intersection
- Demaine et al., SODA’00: A form of comparison certificates
- Barbay and Kenyon, SODA’02: “Partition” certificates
- Ngo et al., PODS’14: “Stronger” comparison certificates
Instance-optimal Set Intersection

- **Algorithm** ⇒ **Decision Tree**

\[
\{x_1 < x_2\} \cap \{y\}
\]

```
\[
\begin{array}{c}
y < x_1 \\
Yes \\
∅ \\
No \\
y > x_2 \\
∅ \\
y > x_1 \\
∅ \\
y < x_2 \\
∅ \\
\{x_2\} \\
\{x_1\}
\end{array}
\]
```
Instance-optimal Set Intersection

- **Algorithm** ⇒ Decision Tree
- **Worst-case runtime** ⇒ Tree depth

\[
\{x_1 < x_2\} \cap \{y\} = \begin{cases} 
\emptyset & \text{Yes} \\
\emptyset & \text{No}
\end{cases}
\]

```
\[
\begin{array}{c}
y < x_1 \\
\text{Yes} \quad \text{No}
\end{array}
\]

\[
\begin{array}{c}
y > x_2 \\
\emptyset
\end{array}
\]

\[
\begin{array}{c}
y > x_1 \\
\emptyset
\end{array}
\]

\[
\begin{array}{c}
y < x_2 \\
\emptyset \quad \{x_1\}
\end{array}
\]

\[
\begin{array}{c}
\emptyset \\
\{x_2\}
\end{array}
\]
Instance-optimal Set Intersection

- **Algorithm** ⇒ Decision Tree
- **Worst-case runtime** ⇒ Tree depth
- **Instance-specific runtime** ⇒ Leaf depth
Instance-optimal Set Intersection

- Algorithm ⇒ Decision Tree
- Worst-case runtime ⇒ Tree depth
- Instance-specific runtime ⇒ Leaf depth
- Instance Certificate ⇒ Leaf-to-root path

\[
\{x_1 < x_2\} \cap \{y\}
\]

\[
\begin{align*}
&\emptyset \\
y < x_1 &\quad \text{Yes} \\
y > x_2 &\quad \text{No} \\
\emptyset &
\end{align*}
\]

\[
\begin{align*}
&\emptyset \\
y > x_1 &
\end{align*}
\]

\[
\begin{align*}
&\emptyset \\
y < x_2 &
\end{align*}
\]

\[
\{x_1\}
\]

\[
\{x_2\}
\]
Consider the class of algorithms that access the input only through comparisons
Consider the class of algorithms that access the input only through **comparisons**

\[ R[i] \theta S[j] \]
Instance-optimal Set Intersection

Consider the class of algorithms that access the input only through comparisons

- \( R[i] \theta S[j] \)
  - \( R[i] \) is the \( i \)-th smallest element in \( R \)
Consider the class of algorithms that access the input only through comparisons

- $R[i] \theta S[j]$
- $R[i]$ is the $i$-th smallest element in $R$
- $S[j]$ is the $j$-th smallest element in $S$
Instance-optimal Set Intersection

Consider the class of algorithms that access the input only through **comparisons**

- \( R[i] \ \theta \ S[j] \)
  - \( R[i] \) is the \( i \)-th smallest element in \( R \)
  - \( S[j] \) is the \( j \)-th smallest element in \( S \)
  - \( \theta \in \{<, =, >\} \)
Comparison-based Certificates

Input
- \( R = \{1, 5, 7\} \)
- \( S = \{2, 3, 4, 7, 9, 10\} \)

Output
- \( Q = \{7\} \)

Diagram:

- \( R = \{1, 5, 7\} \)
- \( S = \{2, 3, 4, 7, 9, 10\} \)
- \( Q = \{7\} \)
Comparison-based Certificates

- **Input**
  - $R = \{1, 5, 7\}$
  - $S = \{2, 3, 4, 7, 9, 10\}$

- **Output**
  - $Q = \{7\}$

- **Comparison-based certificate**
  - $R[4] = \infty$

![Diagram showing sets R and S with points marked on a number line.]
Gap-based Certificates

$\mathcal{C}$ is a collection of gap intervals from $R$ and $S$ that cover every point not in $R \cap S$
Gap-based Certificates

- $C$ is a collection of gap intervals from $R$ and $S$ that cover every point not in $R \cap S$
  - Input
    - $R = \{1, 5, 7\}$
    - $S = \{2, 3, 4, 7, 9, 10\}$
  - Output
    - $Q = \{7\}$
Gap-based Certificates

- $C$ is a collection of gap intervals from $R$ and $S$ that cover every point not in $R \cap S$
  
  - **Input**
    - $R = \{1, 5, 7\}$
    - $S = \{2, 3, 4, 7, 9, 10\}$
  
  - **Output**
    - $Q = \{7\}$

![Diagram showing gap intervals for R and S]
From $C_<$ to $C_{\square}$

$|C_{\square}| + Z = O(|C_<|)$
From $C_<$ to $C_\square$

$|C_\square| + Z = O(|C_<|)$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C_\square \leftarrow \emptyset$, $Z \leftarrow \emptyset$

From $C_<$ to $C_{\square}$

$$|C_{\square}| + Z = O(|C_<|)$$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C_{\square} \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $C_{\square} \cup Z$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take $(R, S)$ and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $\mathcal{C}_\square \cup Z$
  - If $t$ is in the output

From $C_<$ to $C_{\square}$

$|C_{\square}| + Z = O(|C_<|)$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C_{\square} \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $C_{\square} \cup Z$
  - If $t$ is in the output
    - There is $R[i] = S[j] = t$

From $C_<$ to $C_{\square}$

$$|C_{\square}| + Z = O(|C_<|)$$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C_{\square} \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $C_{\square} \cup Z$
  - If $t$ is in the output
    - There is $R[i] = S[j](= t)$
    - Add $t$ to $Z$

From $C_<$ to $C_{\Box}$

$|C_{\Box}| + Z = O(|C_＜|)$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C_{\Box} \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $C_{\Box} \cup Z$
  - If $t$ is not in the output

From $\mathcal{C}_<$ to $\mathcal{C}$

$|\mathcal{C}| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take $(R, S)$ and $\mathcal{C}_<$
- $\mathcal{C} \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $\mathcal{C} \cup Z$
  - If $t$ is not in the output
    - There are immovable $R[i] < t < R[i + 1]$

$$
\begin{align*}
R & \quad 1 \quad | \quad 5 \quad | \quad 7 \\
S & \quad 2 \quad 3 \quad 4 \quad | \quad 7 \quad 9 \quad 10
\end{align*}
$$

From $C_<$ to $C\Box$

$|C\Box| + Z = O(|C_<|)$

Proof idea:

- Take $(R, S)$ and $C_<$
- $C\Box \leftarrow \emptyset$, $Z \leftarrow \emptyset$
- Repeat: Find $t$ outside $C\Box \cup Z$
  - If $t$ is not in the output
    - There are immovable $R[i] < t < R[i + 1]$
    - Add $(R[i], R[i + 1])$ to $C\Box$

An Instance-Optimal Algorithm for $\cap$

- $C_\square \leftarrow \emptyset$
- $Z \leftarrow \emptyset$

Repeat: Find the smallest $t$ outside $C_\square \cup Z$
  - If $t$ is in the output
    - Add $t$ to $Z$
  - Otherwise
    - Find $R[i] < t < R[i + 1]$
    - Find $S[j] < t < S[j + 1]$
    - Add $(R[i], R[i + 1])$ and $(S[j], S[j + 1])$ to $C_\square$

Lemma:
$$|C_2| \leq 2 \cdot |C_1^2|,$$ for any $C_1^2$

Runtime:
$$O(p|C_2| \log |Z|).$$
An Instance-Optimal Algorithm for \( \cap \)

- \( C_\Box \leftarrow \emptyset \)
- \( Z \leftarrow \emptyset \)
- **Repeat:** Find the smallest \( t \) outside \( C_\Box \cup Z \)
  - If \( t \) is in the output
    - Add \( t \) to \( Z \)
  - Otherwise
    - Find \( R[i] < t < R[i + 1] \)
    - Find \( S[j] < t < S[j + 1] \)
    - Add \((R[i], R[i + 1])\) and \((S[j], S[j + 1])\) to \( C_\Box \)

**Lemma:** \( |C_\Box| \leq 2 \cdot |C'_\Box| \), for any \( C'_\Box \)
An Instance-Optimal Algorithm for \( \bigcap \)

\[
\begin{align*}
\text{Repeat: Find the smallest } t \text{ outside } C_\square \cup Z \\
\quad \text{If } t \text{ is in the output} \\
\quad \quad \text{Add } t \text{ to } Z \\
\quad \text{Otherwise} \\
\quad \quad \text{Find } R[i] < t < R[i + 1] \\
\quad \quad \text{Find } S[j] < t < S[j + 1] \\
\quad \quad \text{Add } (R[i], R[i + 1]) \text{ and } (S[j], S[j + 1]) \text{ to } C_\square
\end{align*}
\]

Lemma: \( |C_\square| \leq 2 \cdot |C'_\bigcap| \), for any \( C'_\bigcap \)

Runtime: \( O(|C_\square| + Z) = O(|C_\bigcap|) \)
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Instance-optimal Database Joins

▶ Database Join Query

\[ Q(X) = \bigwedge_{F} R_F(X_F) \]

▶ Examples

▶ \( Q(A, B) = R(A, B) \land S(A) \land T(B) \)
▶ \( Q(A, B, C) = R(A, B) \land S(B, C) \land T(C, A) \)
▶ \( Q(A) = R(A) \land S(A) \)
Relation Indices ⇒ Comparison Certificates

- $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
- Suppose $R(A, B)$ is indexed first on $A$ and then on $B$
Relation Indices ⇒ Comparison Certificates

- $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
- Suppose $R(A, B)$ is indexed first on $A$ and then on $B$
Relation Indices ⇒ Comparison Certificates

- \( R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \)
- Suppose \( R(A, B) \) is indexed first on \( A \) and then on \( B \)

\[
\begin{align*}
R[1] &= 2 \\
R[2] &= 4 \\
R[1, 1] &= 1 \\
R[2, 1] &= 2 \\
R[2, 2] &= \infty
\end{align*}
\]
Relation Indices ⇒ Comparison Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]
Relation Indices ⇒ Comparison Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]

\[ R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \]
Relation Indices ⇒ Comparison Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]

\[ R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \]

\[ S = \{1, 2, 3\} \]
Relation Indices ⇒ Comparison Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]

- \[ R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \]
- \[ S = \{1, 2, 3\} \]
- \[ T = \{2, 4\} \]
Relation Indices ⇒ Comparison Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]

\[ R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \]

\[ S = \{1, 2, 3\} \]

\[ T = \{2, 4\} \]

\[ [A, B]\text{-Comparison Certificate:} \]


\[ S[4] = \infty \]

\[ T[1] = R[1, 2] \]

\[ T[2] > R[1, 3] \]

\[ R[1, 4] = \infty \]
Relation Indices $\Rightarrow$ Gap Certificates

$Q(A, B) = R(A, B) \land S(A) \land T(B)$
Relation Indices $\Rightarrow$ Gap Certificates

$Q(A, B) = R(A, B) \land S(A) \land T(B)$

$R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
Relation Indices $\Rightarrow$ Gap Certificates

$Q(A, B) = R(A, B) \land S(A) \land T(B)$

- $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
- $S = \{1, 2, 3\}$
Relation Indices $\Rightarrow$ Gap Certificates

$Q(A, B) = R(A, B) \land S(A) \land T(B)$

- $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
- $S = \{1, 2, 3\}$
- $T = \{2, 4\}$
Relation Indices ⇒ Gap Certificates

\[ Q(A, B) = R(A, B) \land S(A) \land T(B) \]

- \( R = \{(2, 1), (2, 2), (2, 3), (4, 2)\} \)
- \( S = \{1, 2, 3\} \)
- \( T = \{2, 4\} \)

\([A, B]\)-Gap Certificate
Background

- Ngo et al, PODS’14:
  - $|C_{gao}^\square| + Z = O(|C_{<}^{gao}|)$
  - Minesweeper algorithm
    - First Instance-optimal Join Algorithm
    - $O(|C_{<}^{gao}| + Z)$ for $\beta$-acyclic queries
    - $O(|C_{<}^{gao}|^{w+1} + Z)$ for treewidth $w$-queries
Background

▶ Abo Khamis et al, PODS’15:
  ▶ A tighter notion of certificate $|C_{\Box}| \leq |C_{\text{gao}}|$  
  ▶ *Tetris* algorithm
    ▶ works over different kinds of indexes.
    ▶ achieves the fractional hypertree-width bound.
    ▶ achieves a series of instance-optimality results.
  ▶ A **proof system** for joins where
    ▶ proof complexity lower bounds/upper bounds are developed.
    ▶ proof sizes precisely capture the runtime of *Tetris*. 

Multiple Indexes $\Rightarrow C$ \\

$$R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$$
Multiple Indexes ⇒ $C$.

$R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$

[A,B]-Index
Multiple Indexes ⇒ $C$
Multiple Indexes ⇒ $C_\square$

$$R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$$

Quad tree
Box Cover Problem

Problem (BCP)
Box Cover Problem

Problem (BCP)
Given a set $A$ of (multi-dimensional rectangular) boxes,
Box Cover Problem

Problem (BCP)
Given a set $\mathcal{A}$ of (multi-dimensional rectangular) boxes,

- list all tuples *not* covered by any box in $\mathcal{A}$. 
Box Cover Problem

Problem (BCP)
Given a set $A$ of (multi-dimensional rectangular) boxes,

- list all tuples *not* covered by any box in $A$.

Relational Join can be reduced to BCP
Definition (Box Certificate)

Given a set of boxes $A$, a *box certificate* $C_\square$ for $A$ is a *minimum-sized* subset of $A$ such that

$$\bigcup_{c \in C_\square} c = \bigcup_{a \in A} a.$$
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Dyadic Boxes

▶ Suppose $|\text{Domain}(A_i)| = 2^d$, for simplicity.
Dyadic Boxes

- Suppose $|\text{Domain}(A_i)| = 2^d$, for simplicity.
- A dyadic interval is a binary string of length $\leq d$.

\[
\begin{array}{c|c|c}
\lambda & 0 & 1 \\
00 & 01 & 10 & 11 \\
\end{array}
\]
Dyadic Boxes

- Suppose $|\text{Domain}(A_i)| = 2^d$, for simplicity.
- A **dyadic interval** is a binary string of length $\leq d$.
- A **dyadic box** is an $n$-tuple of binary strings of length $\leq d$. 

![Diagram of dyadic boxes and intervals]
Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into \( \leq (2d)^n = \tilde{O}(1) \) dyadic boxes.
Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into \( \leq (2d)^n = \tilde{O}(1) \) dyadic boxes.

Gap boxes for \( R(A, B) \)
Every (not necessarily dyadic) box can be decomposed into $\lesssim (2d)^n = \tilde{O}(1)$ dyadic boxes.

Gap boxes for $R(A, B)$

Corresponding dyadic boxes
Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into $\leq (2d)^n = \tilde{O}(1)$ dyadic boxes.

Gap boxes for $R(A, B)$

- Every $n$-tuple is contained in $\leq d^m = \tilde{O}(1)$ dyadic boxes.
Geometric Resolution ...

... is an inference system for BCP.
Geometric Resolution ... 

... is an inference system for BCP.

\[ \langle 10, 01 \rangle \]
Geometric Resolution ...

... is an inference system for BCP.

\[ \langle 10, 01 \rangle \]
\[ \langle \lambda, 00 \rangle \]
Geometric Resolution ...

... is an inference system for BCP.

\[ \langle 10, 01 \rangle \]
\[ \langle \lambda, 00 \rangle \]
\[ \langle 10, 0 \rangle \]
Geometric Resolution ... 

... is an inference system for BCP.

... is analogous to traditional resolution in logic.
Geometric Resolution ...

... is an inference system for BCP.

... is analogous to traditional resolution in logic.
Geometric Resolution ...

... is an inference system for BCP.

... is analogous to traditional resolution in logic.
Geometric Resolution ...

... is an inference system for BCP.

... is analogous to traditional resolution in logic.
Geometric Resolution

- Geometric Resolution is complete.
Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes \( A \) that covers some box \( b \), we can infer from \( A \) a box \( b' \) that covers \( b \).
Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $A$ that covers some box $b$, we can infer from $A$ a box $b'$ that covers $b$.
- Three main variations:
Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $A$ that covers some box $b$, we can infer from $A$ a box $b'$ that covers $b$.

- Three main variations:
  - **GEOMETRIC RESOLUTION**
Geometric Resolution is complete.

- Given a set of boxes $A$ that covers some box $b$, we can infer from $A$ a box $b'$ that covers $b$.

Three main variations:

- Geometric Resolution
- Ordered Geometric Resolution
Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $A$ that covers some box $b$, we can infer from $A$ a box $b'$ that covers $b$.

- Three main variations:
  - GEOMETRIC RESOLUTION
  - ORDERED GEOMETRIC RESOLUTION
  - TREE ORDERED GEOMETRIC RESOLUTION
(General) Geometric Resolution

\[ w = \text{Resolve}(w_1, w_2) \]

\[ w_1 = \langle y_1, \ldots, y_{\ell-1}, x_{\ell 0}, y_{\ell+1}, \ldots, y_n \rangle \]

\[ w_2 = \langle z_1, \ldots, z_{\ell-1}, x_{\ell 1}, z_{\ell+1}, \ldots, z_n \rangle \]

\[ w = \langle \ldots, y_{\ell-1} \cap z_{\ell-1}, x_{\ell}, y_{\ell+1} \cap z_{\ell+1}, \ldots \rangle \]
Ordered Geometric Resolution

\[ w = \text{Resolve}(w_1, w_2) \]

\[
\begin{align*}
    w_1 & = \langle y_1 , \ldots , y_{\ell-1} , x_{\ell 0} , \lambda , \ldots , \lambda \rangle \\
    w_2 & = \langle z_1 , \ldots , z_{\ell-1} , x_{\ell 1} , \lambda , \ldots , \lambda \rangle \\
\end{align*}
\]

\[
\begin{align*}
    w & = \langle \ldots , y_{\ell-1} \cap z_{\ell-1} , x_{\ell} , \lambda , \ldots , \lambda \rangle \\
\end{align*}
\]
Tree-Ordered Geometric Resolution

- Proof is a Tree (as opposed to DAG)
  - No caching
Tetris: a recursive algorithm for BCP
Tetris: a recursive algorithm for BCP

is b covered by the union of boxes in A?

b
Tetris: a recursive algorithm for BCP

split b into two halves $b_1, b_2$, 

\[ b \rightarrow b_1 \mid b_2 \]
Tetris: a recursive algorithm for BCP

recursively verify that $b_1$ and $b_2$ are covered through finding two witnesses $w_1, w_2$ that cover $b_1, b_2$. 
$w = \text{Resolve}(w_1, w_2)$, then $w$ covers $b$, add $w$ to $A$, $w$ is a witness for $b$. 

Diagram:

- $b$
- $w$
- $b_1, b_2$
- $w_1, w_2$
Two main analytical components

- runtime $= \Theta(#\text{resolutions})$
Two main analytical components

- runtime = $\Theta(#\text{resolutions})$
- #resolutions is a function of dimension ordering
Two main analytical components

- runtime = $\Theta(#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
Two main analytical components

- runtime $= \Theta(#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)
Two main analytical components

- runtime = Θ(#resolutions)
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)
  - Tetris-Reloaded (load as needed)
Two main analytical components

- runtime = Θ(#resolutions)
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)
  - Tetris-Reloaded (load as needed)
  - Tetris-Balanced (work under a transformed space)
Tetris-Preloaded: Example

Input Boxes

\(\langle 0, \lambda \rangle\)
\(\langle 1, 0 \rangle\)
\(\langle \lambda, 11 \rangle\)
\(\langle 11, 1 \rangle\)
Tetris-Preloaded: Example

Is $\langle \lambda, \lambda \rangle$ covered?
No
Split into $\langle 0, \lambda \rangle$ and $\langle 1, \lambda \rangle$
Is $\langle 0, \lambda \rangle$ covered?
Tetris-Preloaded: Example

Is $\langle 0, \lambda \rangle$ covered?
Yes by $\langle 0, \lambda \rangle$
Tetris-Preloaded: Example

Is \( \langle 0, \lambda \rangle \) covered? Yes by \( \langle 0, \lambda \rangle \)
Is $\langle 1, \lambda \rangle$ covered?
No
Split into $\langle 10, \lambda \rangle$ and $\langle 11, \lambda \rangle$
Is $\langle 10, \lambda \rangle$ covered?
No
Split into $\langle 10, 0 \rangle$ and $\langle 10, 1 \rangle$
Tetris-Preloaded: Example

Is $\langle 10, 0 \rangle$ covered?
Is \langle 10, 0 \rangle covered? Yes by \langle 1, 0 \rangle
Is $\langle 10, 0 \rangle$ covered?
Yes by $\langle 1, 0 \rangle$
Is $\langle 10, 1 \rangle$ covered?  
No  
Split into $\langle 10, 10 \rangle$ and $\langle 10, 11 \rangle$
Tetris-Preloaded: Example

Is $\langle 10, 10 \rangle$ covered?
No
It cannot be split
Tetris-Preloaded: Example

Is \( \langle 10, 10 \rangle \) covered?
No
It cannot be split
Output \( \langle 10, 10 \rangle \)
Add a box \( \langle 10, 10 \rangle \)
Tetris-Preloaded: Example

Is $\langle 10, 10 \rangle$ covered?
No
It cannot be split
Output $\langle 10, 10 \rangle$
Add a box $\langle 10, 10 \rangle$
Tetris-Preloaded: Example

Is \( \langle 10, 11 \rangle \) covered?
Tetris-Preloaded: Example

Is \( \langle 10, 11 \rangle \) covered?
Yes by \( \langle \lambda, 11 \rangle \)
Tetris-Preloaded: Example

Is $\langle 10, 11 \rangle$ covered?
Yes by $\langle \lambda, 11 \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, 1 \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, 1 \rangle$
Resolve $\langle \lambda, 11 \rangle$
$\langle 10, 10 \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, 1 \rangle$
Resolve $\langle \lambda, 11 \rangle$
$\langle 10, 10 \rangle$
$\langle 10, 1 \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, 1 \rangle$
Resolve $\langle \lambda, 11 \rangle$
$\langle 10, 10 \rangle$
$\langle 10, 1 \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to \( \langle 10, \lambda \rangle \)
Resolve
\( \langle 10, 1 \rangle \)
\( \langle 1, 0 \rangle \)
Tetris-Preloaded: Example

Backtrack to $\langle 10, \lambda \rangle$
Resolve
$\langle 10, 1 \rangle$
$\langle 1, 0 \rangle$
$\langle 10, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 10, \lambda \rangle$
Resolve
$\langle 10, 1 \rangle$
$\langle 1, 0 \rangle$
$\langle 10, \lambda \rangle$
Is $\langle 11, \lambda \rangle$ covered?
No
Split into $\langle 11, 0 \rangle$ and $\langle 11, 1 \rangle$
Tetris-Preloaded: Example

Is \((11, 0)\) covered?
Is $\langle 11, 0 \rangle$ covered?
Yes by $\langle 1, 0 \rangle$
Is $\langle 11, 0 \rangle$ covered?
Yes by $\langle 1, 0 \rangle$
Is $\langle 11, 1 \rangle$ covered?
Tetris-Preloaded: Example

Is \( \langle 11, 1 \rangle \) covered?
Yes by \( \langle 11, 1 \rangle \)
Is \langle 11, 1 \rangle covered?
Yes by \langle 11, 1 \rangle
Tetris-Preloaded: Example

Backtrack to $\langle 11, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 11, \lambda \rangle$
Resolve
$\langle 11, 1 \rangle$
$\langle 1, 0 \rangle$
Backtrack to $\langle 11, \lambda \rangle$

Resolve

$\langle 11, 1 \rangle$

$\langle 1, 0 \rangle$

$\langle 11, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 11, \lambda \rangle$

Resolve

$\langle 11, 1 \rangle$

$\langle 1, 0 \rangle$

$\langle 11, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to \( \langle 1, \lambda \rangle \)
Tetris-Preloaded: Example

Backtrack to \(\langle 1, \lambda \rangle\)
Resolve
\(\langle 10, \lambda \rangle\)
\(\langle 11, \lambda \rangle\)
Tetris-Preloaded: Example

Backtrack to $\langle 1, \lambda \rangle$
Resolve $\langle 10, \lambda \rangle$
$\langle 11, \lambda \rangle$
$\langle 1, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle 1, \lambda \rangle$
Resolve
$\langle 10, \lambda \rangle$
$\langle 11, \lambda \rangle$
$\langle 1, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle \lambda, \lambda \rangle$
Tetris-Preloaded: Example

Backtrack to $\langle \lambda, \lambda \rangle$

Resolve

$\langle 0, \lambda \rangle$

$\langle 1, \lambda \rangle$
Tetris-Preloaded: Example

\[ \begin{array}{ccccc}
00 & 01 & 10 & 11 \\
\hline
0 & \lambda & 1 \\
11 & 10 & 01 & 00
\end{array} \]

Backtrack to \( \langle \lambda, \lambda \rangle \)
Resolve
\[ \langle 0, \lambda \rangle \]
\[ \langle 1, \lambda \rangle \]
\[ \langle \lambda, \lambda \rangle \]
Tetris-Preloaded: Example

Backtrack to $\langle \lambda, \lambda \rangle$

Resolve

$\langle 0, \lambda \rangle$

$\langle 1, \lambda \rangle$

$\langle \lambda, \lambda \rangle$

Done!
Upper Bounds

$\tilde{O}(|\mathcal{C}|^{\frac{n}{2}} + Z)$: any

$\tilde{O}(|\mathcal{C}| + Z)$: $tw\ 1$

$\tilde{O}(|\mathcal{C}|^{w+1} + Z)$: $tw\ w$

$\tilde{O}(N^{fhtw} + Z)$: any

Tetris-Preloaded

$\tilde{O}(AGM)$: any

Lower Bounds

$\Omega(|\mathcal{C}|^{\frac{n}{2}} + Z)$: $n$-clique

$\Omega(|\mathcal{C}|^{n-1} + Z)$: any

$\Omega(|\mathcal{C}|^{w+1} + Z)$: $tw\ w$

$\Omega(N^{\frac{n}{2}} + Z)$: $tw\ 1$

**Geometric Resolution**

**Ordered Geometric Resolution**

**Tree Ordered Geometric Resolution**
Tetris-Reloaded

▶ Algorithm
1. \( C_{\Box} \leftarrow \emptyset \)
2. **Fix** a dimension ordering
3. Run Tetris. If an uncovered point \( b \) is found
   ▶ Query for boxes covering \( b \) \( (\tilde{O}(1)) \)
   ▶ Load them into \( C_{\Box} \)
   ▶ Repeat
Tetris-Reloaded

Algorithm
1. $\mathcal{C}_\square \leftarrow \emptyset$
2. **Fix** a dimension ordering
3. Run Tetris. If an uncovered point $b$ is found
   - Query for boxes covering $b$ \[O(1)\]
   - Load them into $\mathcal{C}_\square$
   - Repeat

Analysis
- $|\mathcal{C}_\square| = \tilde{O}(|\mathcal{C}'_\square|)$, \text{ for any } $\mathcal{C}'_\square$
<table>
<thead>
<tr>
<th>Upper Bounds</th>
<th>Lower Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{O} \left(</td>
<td>C</td>
</tr>
<tr>
<td>$\tilde{O} \left(</td>
<td>C</td>
</tr>
<tr>
<td>$\tilde{O} \left(</td>
<td>C</td>
</tr>
<tr>
<td>$\tilde{O} \left( N^{fhtw} + Z \right)$: any</td>
<td>$\tilde{O} \left( AGM \right)$: any</td>
</tr>
<tr>
<td>$\Omega \left( N^{\frac{n}{2}} + Z \right)$: tw 1</td>
<td>$\Omega \left( N^{\frac{n}{2}} + Z \right)$: tw 1</td>
</tr>
</tbody>
</table>

**Geometric Resolution**

**Ordered Geometric Resolution**

**Tree Ordered Geometric Resolution**
Upper Bounds

Tetris-Balanced
\[ \tilde{O}(|\mathcal{C}|^{\frac{n}{2}} + Z) : \text{any} \]
\[ \tilde{O}(|\mathcal{C}| + Z) : \text{tw } 1 \]
\[ \tilde{O}(|\mathcal{C}|^{w+1} + Z) : \text{tw } w \]
\[ \tilde{O}(N^{\text{fhtw}} + Z) : \text{any} \]
\[ \tilde{O}(AGM) : \text{any} \]

Lower Bounds

\[ \Omega(|\mathcal{C}|^{\frac{n}{2}} + Z) : n\text{-clique} \]
\[ \Omega(|\mathcal{C}|^{n-1} + Z) : \text{any} \]
\[ \Omega(|\mathcal{C}|^{w+1} + Z) : \text{tw } w \]
\[ \Omega(N^{\frac{n}{2}} + Z) : \text{tw } 1 \]

**Geometric Resolution**

**Ordered Geometric Resolution**

**Tree Ordered Geometric Resolution**
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Open Problems

- A $\tilde{O}(P)$-algorithm?
Open Problems

▶ A $\tilde{O}(P)$-algorithm?
  ▶ $P$ is the Geometric Resolution-proof size
Open Problems

▶ A $\tilde{O}(P)$-algorithm?
  ▶ $P$ is the Geometric Resolution-proof size
  ▶ $P$ could be anywhere from $|\mathcal{C}|$ to $\tilde{\Theta}(|\mathcal{C}|^{n/2})$
Open Problems

- A $\tilde{O}(P)$-algorithm?
  - $P$ is the \textsc{Geometric Resolution}-proof size
  - $P$ could be anywhere from $|C\Box|$ to $\tilde{\Theta}(|C\Box|^{n/2})$

- A more practical alternative to dyadic encoding?
Open Problems

- A $\tilde{O}(P)$-algorithm?
  - $P$ is the \textsc{Geometric Resolution}-proof size
  - $P$ could be anywhere from $|\mathcal{C}_{\square}|$ to $\tilde{\Theta}(|\mathcal{C}_{\square}|^{n/2})$

- A more \textbf{practical} alternative to dyadic encoding?
  - Shave polylog factors
Open Problems

- A $\tilde{O}(P)$-algorithm?
  - $P$ is the Geometric Resolution-proof size
  - $P$ could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$

- A more practical alternative to dyadic encoding?
  - Shave polylog factors

- Other models for instance optimality?
Open Problems

- A $\tilde{O}(P)$-algorithm?
  - $P$ is the Geometric Resolution-proof size
  - $P$ could be anywhere from $|C|\leq k$ to $\tilde{\Theta}(|C|^{n/2})$

- A more practical alternative to dyadic encoding?
  - Shave polylog factors

- Other models for instance optimality?
- A notion of certificates for algebraic algorithms?
Open Problems

- A $\tilde{O}(P)$-algorithm?
  - $P$ is the Geometric Resolution-proof size
  - $P$ could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$

- A more practical alternative to dyadic encoding?
  - Shave polylog factors

- Other models for instance optimality?

- A notion of certificates for algebraic algorithms?
  - Algebraic algorithms can break the $\Omega(|\mathcal{C}_\square|^{n/2})$-lower bound
Open Problems

- A \(\tilde{O}(P)\)-algorithm?
  - \(P\) is the \textsc{Geometric Resolution}-proof size
  - \(P\) could be anywhere from \(|C_\Box|\) to \(\tilde{\Theta}(|C_\Box|^{n/2})\)

- A more \textbf{practical} alternative to dyadic encoding?
  - Shave polylog factors

- \textbf{Other models} for instance optimality?

- \textbf{A notion of certificates for algebraic algorithms}?
  - Algebraic algorithms can break the \(\Omega(|C_\Box|^{n/2})\)-lower bound
    - e.g.
      - listing triangles in \(O(N^{1.408} + N^{1.222}Z^{0.186})\)
        [Björklund et al, ICALP’14]
Open Problems (cont.)

- Count/Aggregate queries?
Open Problems (cont.)

- **Count/Aggregate queries?**
  - What is a natural notion of certificates here?

- **Recursive queries?**
  - e.g. instance-optimal transitive closure?

- **Instance-optimal all-pairs shortest-paths?**

- **Instance Optimality under Updates (IVM)?**
Open Problems (cont.)

▶ **Count/Aggregate** queries?
  ▶ What is a natural notion of certificates here?
  ▶ **Corresponding instance-optimal algorithm?**
Open Problems (cont.)

- **Count/Aggregate queries?**
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?

- **Recursive queries?**
Open Problems (cont.)

- **Count/Aggregate** queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?

- **Recursive** queries?
  - e.g. instance-optimal transitive closure?
Open Problems (cont.)

- **Count/Aggregate queries?**
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- **Recursive queries?**
  - e.g. instance-optimal transitive closure?
  - Instance-optimal all-pairs shortest-paths?
Open Problems (cont.)

- **Count/Aggregate** queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- **Recursive** queries?
  - e.g. instance-optimal transitive closure?
  - Instance-optimal all-pairs shortest-paths?
- **Instance Optimality under Updates (IVM)**?
Many Thanks!
Any Questions/Comments?
Table of Contents

Instance Optimality

Instance-optimal Set Intersection

Instance-optimal Database Joins

Geometric Resolution

The Tetris Algorithm

Open Problems

Appendix
Algorithm

1. Load all gap boxes
2. Fix a dimension ordering
3. Run Tetris
Tetris-Preloaded

▶ Algorithm
1. Load all gap boxes
2. Fix a dimension ordering
3. Run Tetris

▶ Underlying Proof System
  ▶ Ordered Geometric Resolution
Algorithm
1. Load all gap boxes
2. Fix a dimension ordering
3. Run Tetris

Underlying Proof System
- **ORDERED GEOMETRIC RESOLUTION**

Runtime Bounds
- $\tilde{O}(N + N^{\text{fh}tw} + Z)$
  - $\tilde{O}(N + Z)$ for acyclic queries
  - $\tilde{O}(\text{AGM})$ even *without caching*
(TREE ORDERED GEOMETRIC RESOLUTION)
Tetris-Reloaded: More details

- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION
Tetris-Reloaded: More details

- Underlying Proof System
  - **ORDERED GEOMETRIC RESOLUTION**

- Runtime Bounds
  - $\tilde{O}(|C_H| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|C_H|^{w+1} + Z)$
Tetris-Reloaded: More details

- Underlying Proof System
  - \textbf{Ordered Geometric Resolution}

- Runtime Bounds
  - $\tilde{O}(|C| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|C|^w + Z)$

- Lower Bounds for \textbf{Ordered Geometric Resolution}
  - $\Omega(|C| + Z)$ for treewidth $w = 1$
  - $\Omega(|C|^w + Z)$
  - $\Omega(|C|^n + Z)$ for $n$-clique
Tetris-Reloaded: More details

- **Underlying Proof System**
  - **ORDERED GEOMETRIC RESOLUTION**

- **Runtime Bounds**
  - $\tilde{O}(|\mathcal{C}_\square| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|\mathcal{C}_\square|^{w+1} + Z)$

- **Lower Bounds for ORDERED GEOMETRIC RESOLUTION**
  - $\Omega(|\mathcal{C}_\square| + Z)$ for treewidth $w = 1$
  - $\Omega(|\mathcal{C}_\square|^{w+1} + Z)$
  - $\Omega(|\mathcal{C}_\square|^{n-1} + Z)$ for $n$-clique
    - *But AGM bound for an $n$-clique is $\tilde{O}(N^{n/2})$*
$\Omega(|\mathcal{C}_{\square}|^{n-1})$ for Ordered Geometric Resolution
\( \Omega(|C_{\square}|^{n-1}) \) for ORDERED GEOMETRIC RESOLUTION

Consider the above certificate \( C_{\square} \)
$\Omega(\left| \mathcal{C}_{\square} \right|^{n-1})$ for Ordered Geometric Resolution

Ordered resolution under any order starting with $Z$ results in $\left| \mathcal{C}_{\square} \right|^2$
Ordered resolution under any order starting with $Z$ results in $|\mathcal{C}_\square|^2$
$\Omega(|C_{\square}|^{n-1})$ for Ordered Geometric Resolution

Ordered resolution under any order starting with $Z$ results in $|C_{\square}|^2$
\( \Omega(|C_\square|^{n-1}) \) for **Ordered Geometric Resolution**

Ordered resolution under any order starting with \( Z \) results in \( |C_\square|^2 \)
\( \Omega(|\mathcal{C}_\square|^{n-1}) \) for Ordered Geometric Resolution

By concatenating together 3 rotated instances of the above, we get a lower bound of \( |\mathcal{C}_\square|^2 \) for any fixed order
$\Omega(|\mathcal{C}_\square|^n/2)$ for (Unordered) Geometric Resolution

Let $m := \sqrt{|\mathcal{C}_\square|}/3$
Ω(|C□|^n/2) for (Unordered) Geometric Resolution

- Let \( m := \sqrt{|C□|}/3 \)
- \( m \times m \) red boxes

Resolving any two boxes results in a box of size 2 (This does NOT prove the lower bound! Just for intuition..)
\( \Omega(|\mathcal{C}_\square|^n/2) \) for (Unordered) \textbf{Geometric Resolution}

Let \( m := \sqrt{|\mathcal{C}_\square|}/3 \)

- \( m \times m \) red boxes
- \( m \times m \) green boxes
$\Omega(|C_{\square}|^{n/2})$ for (Unordered) Geometric Resolution

Let $m := \sqrt{|C_{\square}|}/3$

- $m \times m$ red boxes
- $m \times m$ green boxes
- $m \times m$ blue boxes

Resolving any two boxes results in a box of size 2 (This does NOT prove the lower bound! Just for intuition.)
\[ \Omega(|\mathcal{C}_\Box|^n/2) \] for (Unordered) **Geometric Resolution**

- Let \( m := \sqrt{|\mathcal{C}_\Box|}/3 \)
- \( m \times m \) red boxes
- \( m \times m \) green boxes
- \( m \times m \) blue boxes
\[ \Omega(|\mathcal{C}_\square|^{n/2}) \text{ for (Unordered) GEOMETRIC RESOLUTION} \]

Let \( m := \sqrt{|\mathcal{C}_\square|/3} \)

- \( m \times m \) red boxes
- \( m \times m \) green boxes
- \( m \times m \) blue boxes

Resolving any two boxes results in a box of size 2
\[ \Omega(|C_{\square}|^{n/2}) \text{ for (Unordered) Geometric Resolution} \]

Let \( m := \sqrt{|C_{\square}|}/3 \)

- \( m \times m \) red boxes
- \( m \times m \) green boxes
- \( m \times m \) blue boxes

Resolving any two boxes results in a box of size 2

(This does NOT prove the lower bound! Just for intuition..)
Tetris-Balanced

- Algorithm

1. Suppose the input space has $n$-dimensions $A_1, ..., A_n$.

2. For each $i \in \{1, ..., n\}$:
   "Split" dimension $A_i$ into two smaller dimensions $pA_{i1}, A_{i2}$ in a "balanced" way.

3. Fix the dimension order:
   $pA_{11}, A_{12}, ..., A_{1n/2}, A_{n1}, A_n, A_{2n/2}, A_{2n-3}, ..., A_{21}$.

4. Run Tetris-Reloaded in the new space of dimension $2n/2$. 

- Underlying Proof System
- Geometric Resolution
- Runtime Bound

\[ \tilde{O}(p|C|^2) \]
Tetris-Balanced

▶ Algorithm

1. Suppose the input space has $n$-dimensions $A_1, ..., A_n$. 
Algorithm

1. Suppose the input space has $n$-dimensions $A_1, \ldots, A_n$.
   - For each $i \in \{1, \ldots, n-2\}$
     - “Split” dimension $A_i$ into two smaller dimensions $(A'_i, A''_i)$ in a “balanced” way.
Tetris-Balanced

Algorithm

1. Suppose the input space has $n$-dimensions $A_1, \ldots, A_n$.
   - For each $i \in \{1, \ldots, n - 2\}$
     - “Split” dimension $A_i$ into two smaller dimensions $(A'_i, A''_i)$ in a “balanced” way.

2. Fix the dimension order:

   $$(A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1)$$
Tetris-Balanced

Algorithm

1. Suppose the input space has \( n \)-dimensions \( A_1, \ldots, A_n \).
   - For each \( i \in \{1, \ldots, n-2\} \)
     “Split” dimension \( A_i \) into two smaller dimensions \((A'_i, A''_i)\) in a “balanced” way.

2. Fix the dimension order:

\[
(A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1)
\]

3. Run Tetris-Reloaded (in the new space of dimension \( 2n - 2 \))
Algorithm

1. Suppose the input space has \( n \)-dimensions \( A_1, \ldots, A_n \).
   ▶ For each \( i \in \{1, \ldots, n - 2\} \)
     “Split” dimension \( A_i \) into two smaller dimensions \((A'_i, A''_i)\) in a “balanced” way.

2. Fix the dimension order:
   \[
   (A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1)
   \]

3. Run Tetris-Reloaded (in the new space of dimension \( 2n - 2 \))

Underlying Proof System
Tetris-Balanced

Algorithm
1. Suppose the input space has $n$-dimensions $A_1, ..., A_n$.
   - For each $i \in \{1, ..., n - 2\}$
     “Split” dimension $A_i$ into two smaller dimensions $(A_i', A_i'')$ in a “balanced” way.
2. Fix the dimension order:

\[(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')\]
3. Run Tetris-Reloaded (in the new space of dimension $2n - 2$)

Underlying Proof System

Geometric Resolution
Tetris-Balanced

Algorithm

1. Suppose the input space has $n$-dimensions $A_1, \ldots, A_n$.
   - For each $i \in \{1, \ldots, n - 2\}$
     “Split” dimension $A_i$ into two smaller dimensions $(A'_i, A''_i)$ in a “balanced” way.

2. Fix the dimension order:

   $$ (A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1) $$

3. Run Tetris-Reloaded (in the new space of dimension $2n - 2$)

Underlying Proof System

- Geometric Resolution

Runtime Bound
Algorithm

1. Suppose the input space has \( n \)-dimensions \( A_1, \ldots, A_n \).
   - For each \( i \in \{1, \ldots, n - 2\} \)
     “Split” dimension \( A_i \) into two smaller dimensions \((A'_i, A''_i)\) in a “balanced” way.

2. Fix the dimension order:

\[
(A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1)
\]

3. Run Tetris-Reloaded (in the new space of dimension \( 2n - 2 \))

Underlying Proof System

- Geometric Resolution

Runtime Bound

- \( \tilde{O}(|C\Box|^{n/2} + Z) \)
Consider the above box set $C_{\Box}$.
Split $Z$ into $\sqrt{|C|}$ slices where each slice has $\sqrt{|C|}$ boxes fully contained in the slice
Do resolution over $Z$ only within each slice
Then resolve over $X$ and $Y$
Then resolve the slices together over $Z$
Some Followup Works

  ▶ Given a relation $R$ with $N$ tuples, generate all maximal dyadic gap boxes of $R$ in time $\tilde{O}(N)$.
  ▶ Strengthens the notion of $C_\square$.