The Fine-Grained Complexity of Boolean Conjunctive Queries and Sum-Product Problems

Austen Z. Fan Paraschos Koutris Hangdong Zhao

University of Wisconsin, Madison

ICALP 2023

Outline

Boolean Conjunctive Queries

Sum-of-Product Computation

Fine-Grained Complexity

Our Work

Boolean Conjunctive Queries
- Preliminaries
- Algorithms

Sum-of-Product Computation
- BCQ as CSP
- Semiring framework

Fine-Grained Complexity

Our Work
- Clique embedding power
- Main results
- Tightness and gaps

Boolean Conjunctive Queries
- Preliminaries
- Algorithms

Sum-of-Product Computation
- BCQ as CSP
- Semiring framework

Fine-Grained Complexity

Our Work
- Clique embedding power
- Main results
- Tightness and gaps

Preliminaries, I

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

It is called *Boolean* if its head is empty.

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

It is called *Boolean* if its head is empty.

Example

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

It is called *Boolean* if its head is empty.

Example

Listing 3-cycles

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

It is called *Boolean* if its head is empty.

Example

Listing 3-cycles

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

Preliminaries, I

A *conjunctive query* q is an expression of the form

$$q\left(x_1, \ldots, x_k\right) :- R_1\left(\vec{y_1}\right), \ldots, R_n\left(\vec{y_n}\right).$$

It is called *Boolean* if its head is empty.

Example

Listing 3-cycles

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

Detecting (the existence of) a 3-cycle

Preliminaries, I

A *conjunctive query* q is an expression of the form

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

It is called *Boolean* if its head is empty.

Example

Listing 3-cycles

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

Detecting (the existence of) a 3-cycle

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

Preliminaries, II

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

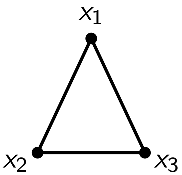
$q() : -R(x_1, x_2), S(x_2, x_3), T(x_3, x_1)$

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

$q() : -R(x_1, x_2), S(x_2, x_3), T(x_3, x_1)$



Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

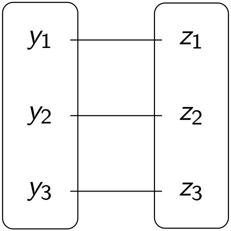
$q() :- R(y_1, z_1), S(y_2, z_2), T(y_3, z_3), U(y_1, y_2, y_3), V(z_1, z_2, z_3)$

Preliminaries, II

For every CQ q, we associate a *hypergrpah* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

$q() : -R(y_1, z_1), S(y_2, z_2), T(y_3, z_3), U(y_1, y_2, y_3), V(z_1, z_2, z_3)$



Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Yannakakis Algorithm

Yannakakis Algorithm

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

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 Boolean CQ q has a join tree, then we can evaluate q in linear time.

Yannakakis Algorithm

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 Boolean CQ q has a join tree, then we can evaluate q in linear time.

Example

Yannakakis Algorithm

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 Boolean CQ q has a join tree, then we can evaluate q in linear time.

Example

$q() : -R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$

Yannakakis Algorithm

Definition (Join Tree)

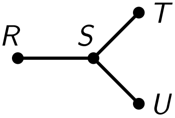
A *join tree* for a CQ q is a tree $\mathcal{T}$ whose vertices are the atoms in q such that, for any pair of atoms R, S, all variables common to R and S occur on the unique path connecting R and S.

Theorem (Yannakakis, 81')

If a Boolean CQ q has a join tree, then we can evaluate q in linear time.

Example

$q() : -R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_3, x_5)$



Worst-Case Optimal Joins, I

Worst-Case Optimal Joins, I

Definition (Fractional Edge Cover)

A *fractional edge cover* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is an assignment from each hyperedge $e \in \mathcal{E}$ to a weight $u_e \in \mathbb{R}_{\geq 0}$, such that for any vertex $v \in \mathcal{V}$ we have $\sum_{e \in \mathcal{E} : v \in e} u_e \geq 1$.

Worst-Case Optimal Joins, I

Definition (Fractional Edge Cover)

A *fractional edge cover* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is an assignment from each hyperedge $e \in \mathcal{E}$ to a weight $u_e \in \mathbb{R}_{\geq 0}$, such that for any vertex $v \in \mathcal{V}$ we have $\sum_{e \in \mathcal{E} : v \in e} u_e \geq 1$.

Theorem (AGM Bound)

Let q be a full CQ with associated $\mathcal{H}$. For every fractional edge cover of $\mathcal{H}$, the output size of q is bounded by $\prod_{e \in \mathcal{E}} N_e^{u_e}$.

Worst-Case Optimal Joins, I

Definition (Fractional Edge Cover)

A *fractional edge cover* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is an assignment from each hyperedge $e \in \mathcal{E}$ to a weight $u_e \in \mathbb{R}_{\geq 0}$, such that for any vertex $v \in \mathcal{V}$ we have $\sum_{e \in \mathcal{E}: v \in e} u_e \geq 1$.

Theorem (AGM Bound)

Let q be a full CQ with associated $\mathcal{H}$. For every fractional edge cover of $\mathcal{H}$, the output size of q is bounded by $N^{\rho^*(\mathcal{H})}$.

Worst-Case Optimal Joins, I

Definition (Fractional Edge Cover)

A *fractional edge cover* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is an assignment from each hyperedge $e \in \mathcal{E}$ to a weight $u_e \in \mathbb{R}_{\geq 0}$, such that for any vertex $v \in \mathcal{V}$ we have $\sum_{e \in \mathcal{E} : v \in e} u_e \geq 1$.

Theorem (AGM Bound)

Let q be a full CQ with associated $\mathcal{H}$. For every fractional edge cover of $\mathcal{H}$, the output size of q is bounded by $N^{\rho^*(\mathcal{H})}$.

Example

Worst-Case Optimal Joins, I

Definition (Fractional Edge Cover)

A *fractional edge cover* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is an assignment from each hyperedge $e \in \mathcal{E}$ to a weight $u_e \in \mathbb{R}_{\geq 0}$, such that for any vertex $v \in \mathcal{V}$ we have $\sum_{e \in \mathcal{E}: v \in e} u_e \geq 1$.

Theorem (AGM Bound)

Let q be a full CQ with associated $\mathcal{H}$. For every fractional edge cover of $\mathcal{H}$, the output size of q is bounded by $N^{\rho^*(\mathcal{H})}$.

Example

The minimum fractional edge cover number ρ^* of Δ is $\frac{3}{2}$.

Worst-Case Optimal Joins, II

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

Example (Listing Triangles)

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

Example (Listing Triangles)

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

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

Example (Listing Triangles)

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

Directed 2-paths with intermediate vertices being light:

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

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.

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')

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

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:

Worst-Case Optimal Joins, II

Theorem (Ngo, Porat, Ré & Rudra, 12')
Any full CQ q can be computed in time $O(N^{\rho^*(\mathcal{H})})$.

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.

Fractional Hypertree Width

Fractional Hypertree Width

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 $\mathcal{T}$.

Fractional Hypertree Width

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 $\mathcal{T}$.

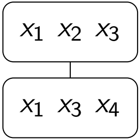
Example

Fractional Hypertree Width

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 $\mathcal{T}$.

Example

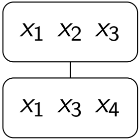
A tree decomposition for  is 

Fractional Hypertree Width

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 $\mathcal{T}$.

Example

A tree decomposition for  is 

Definition (Fractional Hypertree Width)

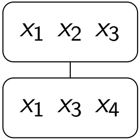
The *fractional hypertree width* $\text{fhtw}(\mathcal{H}) := \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \rho^*(\chi(t))$.

Fractional Hypertree Width

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 $\mathcal{T}$.

Example

A tree decomposition for  is 

Definition (Fractional Hypertree Width)

The *fractional hypertree width* $\mathrm{fhtw}(\mathcal{H}) := \min\limits_{(\mathcal{T},\chi)} \max\limits_{t \in V(\mathcal{T})} \rho^*(\chi(t))$.

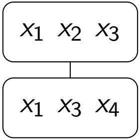
Example

Fractional Hypertree Width

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 $\mathcal{T}$.

Example

A tree decomposition for  is 

Definition (Fractional Hypertree Width)

The *fractional hypertree width* $\text{fhtw}(\mathcal{H}) := \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \rho^*(\chi(t))$.

Example

The fractional hypertree width of  is 2.

Proof-Assisted eNtropic Degree-Aware

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \rightarrow \mathbb{R}^+$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \to \mathbb{R}^+$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Definition (Submodular Width)

The *submodular width* $\mathrm{subw}(\mathcal{H}) := \max_{b} \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} b(\chi(t))$.

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \to \mathbb{R}^+$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Definition (Submodular Width)

The *submodular width* $\text{subw}(\mathcal{H}) := \max_b \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} b(\chi(t))$.

Lemma (Marx, 10')

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

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \to \mathbb{R}^+$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Definition (Submodular Width)

The *submodular width* $\text{subw}(\mathcal{H}) := \max_{b} \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} b(\chi(t))$.

Lemma (Marx, 10')

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

Theorem (Khamis, Ngo & Suciu, 16')

Any BCQ q can be computed in time $\tilde{O}(N^{\text{subw}(q)})$.

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \rightarrow \mathbb{R}^{+}$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Definition (Submodular Width)

The *submodular width* $\mathrm{subw}(\mathcal{H}) := \max_{b} \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} b(\chi(t))$.

Lemma (Marx, 10')

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

Theorem (Khamis, Ngo & Suciu, 16')

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

Example

Proof-Assisted eNtropic Degree-Aware

Definition (Submodularity)

A function $b : 2^{\mathcal{V}} \to \mathbb{R}^+$ is *submodular* if for any $X, Y \subseteq \mathcal{V}$, we have $b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y)$.

Definition (Submodular Width)

The *submodular width* $\text{subw}(\mathcal{H}) := \max_{b} \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} b(\chi(t))$.

Lemma (Marx, 10')

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

Theorem (Khamis, Ngo & Suciu, 16')

Any BCQ q can be computed in time $\tilde{O}(N^{\text{subw}(q)})$.

Example

The submodular width of $\square$ is $\frac{3}{2}$.

Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Constraint Satisfaction Problem

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Example

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Example

3SAT: V the set of variables, $D = \{0, 1\}$, C the set of clauses.

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Example

3SAT: V the set of variables, $D = \{0, 1\}$, C the set of clauses.

Example

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Example

3SAT: V the set of variables, $D = \{0, 1\}$, C the set of clauses.

Example

BCQ: V the set of variables, D the active domain, C the set of database relations.

Constraint Satisfaction Problem

A constraint satisfaction problem consists of (V, D, C), where each constraint is a relation on a subset of the variables.

Example

3SAT: V the set of variables, $D = \{0, 1\}$, C the set of clauses.

Example

BCQ: V the set of variables, D the active domain, C the set of database relations.

Definition (Fixed-Parameter Tractable)

Let $\mathcal{C}$ be a class of hypergraphs. CSP($\mathcal{C}$) is said to be *fixed parameter tractable* if there is an algorithm solving every instance I of CSP($\mathcal{H}$) in time $f(\mathcal{H})(||I||)^{O(1)}$, where f is a computable function.

Dichotomies

Dichotomies

Theorem (Grohe, 03')

If $\mathcal{C}$ is a recursively enumerable class of hypergraphs with bounded edge size, then assuming $\mathsf{FPT} \neq \mathsf{W}[1]$ *the following are equivalent:*

Dichotomies

Theorem (Grohe, 03')

If $\mathcal{C}$ is a recursively enumerable class of hypergraphs with bounded edge size, then assuming FPT $\neq$ W[1] *the following are equivalent:*

1. CSP($\mathcal{C}$) *is polynomial-time solvable.*

Dichotomies

Theorem (Grohe, 03')

If $\mathcal{C}$ is a recursively enumerable class of hypergraphs with bounded edge size, then assuming FPT $\neq$ W[1] *the following are equivalent:*

1. CSP($\mathcal{C}$) *is polynomial-time solvable.*
2. CSP($\mathcal{C}$) *is fixed-parameter tractable.*

Dichotomies

Theorem (Grohe, 03')

If $\mathcal{C}$ is a recursively enumerable class of hypergraphs with bounded edge size, then assuming FPT $\neq$ W[1] *the following are equivalent:*

1. CSP($\mathcal{C}$) *is polynomial-time solvable.*
2. CSP($\mathcal{C}$) *is fixed-parameter tractable.*
3. $\mathcal{C}$ *has bounded treewidth.*

Dichotomies

Theorem (Grohe, 03')

If $\mathcal{C}$ is a recursively enumerable class of hypergraphs with bounded edge size, then assuming FPT $\neq$ W[1] *the following are equivalent:*

1. CSP($\mathcal{C}$) *is polynomial-time solvable.*
2. CSP($\mathcal{C}$) *is fixed-parameter tractable.*
3. $\mathcal{C}$ *has bounded treewidth.*

Theorem (Max, 13')

Let $\mathcal{C}$ be a recursively enumerable class of hypergraphs. Assuming the Exponential Time Hypothesis, CSP($\mathcal{C}$) *parametrized by $\mathcal{H}$ is fixed-parameter tractable if and only if $\mathcal{C}$ has bounded submodular width.*

Boolean Conjunctive Queries
- Preliminaries
- Algorithms

Sum-of-Product Computation
- BCQ as CSP
- Semiring framework

Fine-Grained Complexity

Our Work
- Clique embedding power
- Main results
- Tightness and gaps

Semiring Framework, I

Semiring Framework, I

$$q\left(\right) :- R_1\left(\vec{x}_1\right), R_2\left(\vec{x}_2\right), \ldots, R_n\left(\vec{x}_n\right)$$

Semiring Framework, I

$$q() :- R_1(\vec{x}_1), R_2(\vec{x}_2), \ldots, R_n(\vec{x}_n)$$

$$q(I) := \bigvee_{v:\text{valuation}} \bigwedge_{i=1}^{n} R_i(v(\vec{x}_i))$$

Semiring Framework, I

$$q() : -R_1(\vec{x}_1), R_2(\vec{x}_2), \ldots, R_n(\vec{x}_n)$$

$$q(I) := \bigvee_{v\text{:valuation}} \bigwedge_{i=1}^{n} R_i(v(\vec{x}_i))$$

$$q(I) := \bigoplus_{v\text{:valuation}} \bigotimes_{i=1}^{n} R_i(v(\vec{x}_i))$$

Semiring Framework, I

$$q\left(\right) : -R_1\left(\vec{x}_1\right), R_2\left(\vec{x}_2\right), \ldots, R_n\left(\vec{x}_n\right)$$

$$q(I) := \bigvee_{v:\text{valuation}} \bigwedge_{i=1}^{n} R_i(v\left(\vec{x}_i\right))$$

$$q(I) := \bigoplus_{v:\text{valuation}} \bigotimes_{i=1}^{n} R_i(v\left(\vec{x}_i\right))$$

Example

Semiring Framework, I

$$q() : -R_1(\vec{x}_1), R_2(\vec{x}_2), \ldots, R_n(\vec{x}_n)$$

$$q(I) := \bigvee_{v:\text{valuation}} \bigwedge_{i=1}^{n} R_i(v(\vec{x}_i))$$

$$q(I) := \bigoplus_{v:\text{valuation}} \bigotimes_{i=1}^{n} R_i(v(\vec{x}_i))$$

Example

$(\{\textsc{True}, \textsc{False}\}, \vee, \wedge) \leftrightarrow$ set semantics

Semiring Framework, I

$$q\,() :- R_1\,(\vec{x}_1)\,, R_2\,(\vec{x}_2)\,, \ldots, R_n\,(\vec{x}_n)$$

$$q(I) := \bigvee_{v:\text{valuation}} \bigwedge_{i=1}^{n} R_i(v\,(\vec{x}_i))$$

$$q(I) := \bigoplus_{v:\text{valuation}} \bigotimes_{i=1}^{n} R_i(v\,(\vec{x}_i))$$

Example

$(\{\textsc{True},\ \textsc{False}\}, \vee, \wedge) \leftrightarrow$ set semantics
$(\mathbb{N}, +, *) \leftrightarrow$ bag semantics

Semiring Framework, I

$$q() : -R_1(\vec{x}_1), R_2(\vec{x}_2), \ldots, R_n(\vec{x}_n)$$

$$q(I) := \bigvee_{v\text{:valuation}} \bigwedge_{i=1}^{n} R_i(v(\vec{x}_i))$$

$$q(I) := \bigoplus_{v\text{:valuation}} \bigotimes_{i=1}^{n} R_i(v(\vec{x}_i))$$

Example

$(\{\textsc{True}, \textsc{False}\}, \vee, \wedge) \leftrightarrow$ set semantics
$(\mathbb{N}, +, *) \leftrightarrow$ bag semantics
$([0,1], +, *) \leftrightarrow$ probabilistic database

Semiring Framework, II

Semiring Framework, II

Example

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Example

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Example

Given an edge-weighted graph $G = (V, \mathrm{weight})$

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Example

Given an edge-weighted graph $G = (V, \mathrm{weight})$

Compute $\bigvee_{\substack{V' \subseteq V \\ |V'| = k}} \bigwedge_{\{v,w\} \in V'} \mathrm{weight}(\{v, w\}) \leftrightarrow$ Boolean k-clique

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Example

Given an edge-weighted graph $G = (V, \mathrm{weight})$

Compute $\bigvee_{\substack{V' \subseteq V \\ |V'|=k}} \bigwedge_{\{v,w\} \in V'} \mathrm{weight}(\{v, w\}) \leftrightarrow$ Boolean k-clique

Compute $\sum_{\substack{V' \subseteq V \\ |V'|=k}} \prod_{\{v,w\} \in V'} \mathrm{weight}(\{v, w\}) \leftrightarrow$ Counting k-clique

Semiring Framework, II

Example

Given an n-by-n square matrix $A = (a_{ij})$

Compute $\mathrm{perm}(A) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ #P-hard

Compute $\mathrm{asgmt}(A) := \min_{\sigma \in S_n} \sum_{i=1}^{n} a_{i,\sigma(i)} \Rightarrow$ P-time

Example

Given an edge-weighted graph $G = (V, \mathrm{weight})$

Compute $\bigvee_{\substack{V' \subseteq V \\ |V'|=k}} \bigwedge_{\{v,w\} \in V'} \mathrm{weight}(\{v,w\}) \leftrightarrow$ Boolean k-clique

Compute $\sum_{\substack{V' \subseteq V \\ |V'|=k}} \prod_{\{v,w\} \in V'} \mathrm{weight}(\{v,w\}) \leftrightarrow$ Counting k-clique

Compute $\min_{\substack{V' \subseteq V \\ |V'|=k}} \sum_{\{v,w\} \in V'} \mathrm{weight}(\{v,w\}) \leftrightarrow$ Minimum k-clique

Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Fine-Grained Complexity in 5 Minutes...

Fine-Grained Complexity in 5 Minutes...

"Hardness in easy problems"

Fine-Grained Complexity in 5 Minutes...

"Hardness in easy problems"

The edit distance between two strings := min # insertions, deletions or substitutions to transfrom from one to the other

Fine-Grained Complexity in 5 Minutes...

"Hardness in easy problems"

The edit distance between two strings := min # insertions, deletions or substitutions to transfrom from one to the other

Can be solved in $O(n^2)$ by simple dynamic programming

Fine-Grained Complexity in 5 Minutes...

"Hardness in easy problems"

The edit distance between two strings $:=$ min $\#$ insertions, deletions or substitutions to transfrom from one to the other

Can be solved in $O(n^2)$ by simple dynamic programming

Theorem (Backurs & Indyk, 15')

If the edit distance can be solved in time $O(n^{2-\delta})$ for some constant $\delta > 0$, then the Strong Exponential Time Hypothesis is wrong.

Fine-Grained Complexity in 5 Minutes...

"Hardness in easy problems"

The edit distance between two strings := min # insertions, deletions or substitutions to transfrom from one to the other

Can be solved in $O(n^2)$ by simple dynamic programming

Theorem (Backurs & Indyk, 15')

If the edit distance can be solved in time $O(n^{2-\delta})$ for some constant $\delta > 0$, then the Strong Exponential Time Hypothesis is wrong.

Informally, ETH says that 3-SAT cannot be solved in $2^{o(n)}$ time and SETH says that k-SAT needs 2^n for large k (when $k \to \infty$).

Conjectures

Conjectures

ETH: $\exists\delta > 0$ such that 3-SAT requires $2^{\delta n}$ time.

Conjectures

ETH: $\exists\delta > 0$ such that 3-$\textsc{Sat}$ requires $2^{\delta n}$ time.

SETH: $\forall\epsilon > 0, \exists k$ such that k-$\textsc{Sat}$ on n variables cannot be solved in $O(2^{(1-\epsilon)n})$ time.

Conjectures

ETH: $\exists\delta > 0$ such that 3-$\textsc{Sat}$ requires $2^{\delta n}$ time.

SETH: $\forall\epsilon > 0, \exists k$ such that k-$\textsc{Sat}$ on n variables cannot be solved in $O(2^{(1-\epsilon)n})$ time.

3-SUM: No randomized algorithm can solve 3-SUM on n integers in $\{-n^4, \ldots, n^4\}$ cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$.

Conjectures

ETH: $\exists \delta > 0$ such that 3-SAT requires $2^{\delta n}$ time.

SETH: $\forall \epsilon > 0, \exists k$ such that k-SAT on n variables cannot be solved in $O(2^{(1-\epsilon)n})$ time.

3-SUM: No randomized algorithm can solve 3-SUM on n integers in $\{-n^4, \ldots, n^4\}$ cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$.

APSP: No randomized algorithm can solve APSP in $O(n^{3-\epsilon})$ time for $\epsilon > 0$ on n node graphs with edge weights $\{-n^c, \ldots, n^c\}$ and no negative cycles for large enough c.

Conjectures

ETH: $\exists \delta > 0$ such that 3-SAT requires $2^{\delta n}$ time.

SETH: $\forall \epsilon > 0, \exists k$ such that k-SAT on n variables cannot be solved in $O(2^{(1-\epsilon)n})$ time.

3-SUM: No randomized algorithm can solve 3-SUM on n integers in $\{-n^4, \ldots, n^4\}$ cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$.

APSP: No randomized algorithm can solve APSP in $O(n^{3-\epsilon})$ time for $\epsilon > 0$ on n node graphs with edge weights $\{-n^c, \ldots, n^c\}$ and no negative cycles for large enough c.

...

Conjectures related to k-Clique

Conjectures related to k-Clique

Hypothesis (Combinatorial k-Clique; Lincoln, Vassilevska-Williams & Williams, 17')

Any combinatorial algorithm to detect a k-clique in a graph with n nodes requires $n^{k-o(1)}$ time on a Word RAM model.

Conjectures related to k-Clique

Hypothesis (Combinatorial k-Clique; Lincoln, Vassilevska-Williams & Williams, 17')

Any combinatorial algorithm to detect a k-clique in a graph with n nodes requires $n^{k-o(1)}$ time on a Word RAM model.

Hypothesis (Min Weight k-Clique; Lincoln, Vassilevska-Williams & Williams, 17')

Any randomized algorithm to find a k-clique of minimum total edge weight requires $n^{k-o(1)}$ time on a Word RAM model.

Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Clique Embedding Power, I

Clique Embedding Power, I

Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Clique Embedding Power, I

Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Clique Embedding Power, I

Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Example

Clique Embedding Power, I

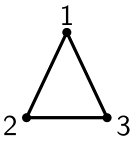
Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Example



Clique Embedding Power, I

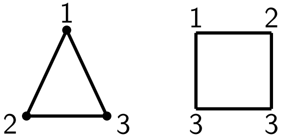
Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Example



Clique Embedding Power, I

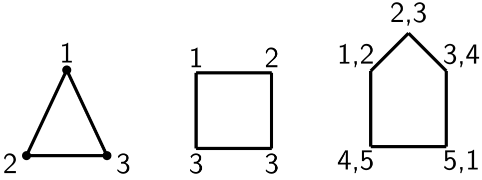
Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Example



Clique Embedding Power, I

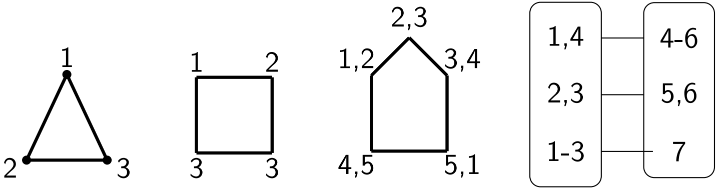
Definition (Touch)

We say $X, Y \subseteq \mathcal{V}$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or $\exists e \in \mathcal{E}$ such that $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition (K-Clique Embedding)

A *k-clique embedding* from C_k to $\mathcal{H}$ is a mapping ψ from $v \in [k]$ to a non-empty subset $\psi(v) \subseteq \mathcal{V}$ such that (1) $\forall v, \psi(v)$ induces a connected subhypergraph and (2) $\forall \{v, u\}$, $\psi(v), \psi(u)$ touch in $\mathcal{H}$.

Example



Clique Embedding Power, II

Clique Embedding Power, II

Definition (Weak Edge Depth)

$\forall e$ the *weak edge depth of e* is $d_\psi(e) := |\{v \in [k] \mid \psi(v) \cap e \neq \emptyset\}|$.
The *weak edge depth of* ψ $\mathrm{wed}(\psi) := \max_e d_\psi(e)$.

Clique Embedding Power, II

Definition (Weak Edge Depth)

$\forall e$ the *weak edge depth of* e is $d_\psi(e) := |\{v \in [k] \mid \psi(v) \cap e \neq \emptyset\}|$. The *weak edge depth of* ψ $\mathrm{wed}(\psi) := \max_e d_\psi(e)$.

Definition (Clique Embedding Power)

The *k-clique embedding power* is $\mathrm{emb}_k(\mathcal{H}) := \max_\psi \frac{k}{\mathrm{wed}(\psi)}$. The *clique embedding power* is $\mathrm{emb}(\mathcal{H}) := \sup_{k \geq 3} \mathrm{emb}_k(\mathcal{H})$.

Clique Embedding Power, II

Definition (Weak Edge Depth)

$\forall e$ the *weak edge depth of* e is $d_\psi(e) := |\{v \in [k] \mid \psi(v) \cap e \neq \emptyset\}|$.
The *weak edge depth of* ψ $\mathrm{wed}(\psi) := \max_e d_\psi(e)$.

Definition (Clique Embedding Power)

The *k-clique embedding power* is $\mathrm{emb}_k(\mathcal{H}) := \max_\psi \frac{k}{\mathrm{wed}(\psi)}$. The *clique embedding power* is $\mathrm{emb}(\mathcal{H}) := \sup_{k \geq 3} \mathrm{emb}_k(\mathcal{H})$.

Example

Clique Embedding Power, II

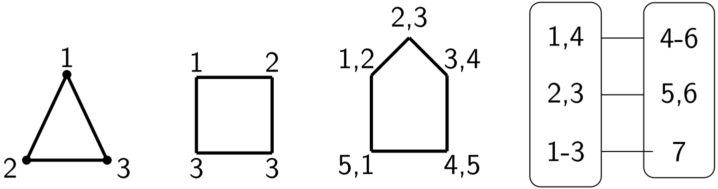
Definition (Weak Edge Depth)

$\forall e$ the *weak edge depth of* e is $d_\psi(e) := |\{v \in [k] \mid \psi(v) \cap e \neq \emptyset\}|$.
The *weak edge depth of* ψ $\mathrm{wed}(\psi) := \max_e d_\psi(e)$.

Definition (Clique Embedding Power)

The *k-clique embedding power* is $\mathrm{emb}_k(\mathcal{H}) := \max_\psi \frac{k}{\mathrm{wed}(\psi)}$. The *clique embedding power* is $\mathrm{emb}(\mathcal{H}) := \sup_{k \geq 3} \mathrm{emb}_k(\mathcal{H})$.

Example



Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Main Theorem

Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

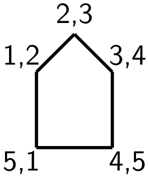
Proof.

Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.

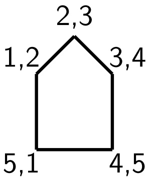


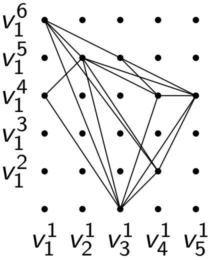
Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.





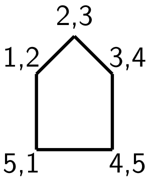
$\square$

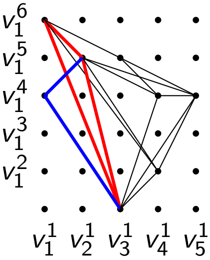
Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.





x_1	x_2
$\langle v_1^6, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$
$\langle v_1^4, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$

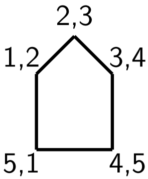
$\square$

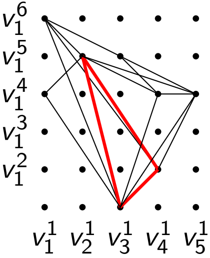
Main Theorem

Theorem (F., Koutris & Zhao, 23')

For any $\mathcal{H}$, CSP($\mathcal{H}$) cannot be computed via a combinatorial algorithm in time $O(|I|^{\text{emb}(\mathcal{H})-\epsilon})$ unless the Combinatorial k-Clique Conjecture is false.

Proof.





x_1	x_2
$\langle v_1^6, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$
$\langle v_1^4, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$

x_2	x_3
$\langle v_2^5, v_3^1 \rangle$	$\langle v_3^1, v_4^2 \rangle$

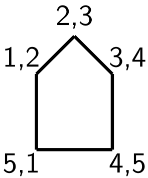
$\square$

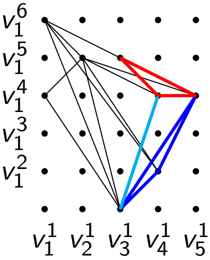
Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.





x_1	x_2
$\langle v_1^6, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$
$\langle v_1^4, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$

x_2	x_3
$\langle v_2^5, v_3^1 \rangle$	$\langle v_3^1, v_4^2 \rangle$

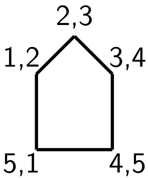
x_3	x_4
$\langle v_3^5, v_4^4 \rangle$	$\langle v_4^4, v_5^4 \rangle$
$\langle v_3^1, v_4^2 \rangle$	$\langle v_4^2, v_5^4 \rangle$
$\langle v_3^1, v_4^4 \rangle$	$\langle v_4^4, v_5^4 \rangle$

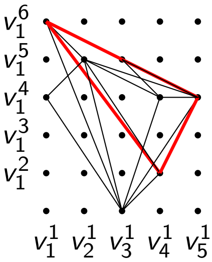
Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.





x_1	x_2
$\langle v_1^6, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$
$\langle v_1^4, v_2^5 \rangle$	$\langle v_2^5, v_3^1 \rangle$

x_2	x_3
$\langle v_2^5, v_3^1 \rangle$	$\langle v_3^1, v_4^2 \rangle$

x_3	x_4
$\langle v_3^5, v_4^4 \rangle$	$\langle v_4^4, v_5^4 \rangle$
$\langle v_3^1, v_4^2 \rangle$	$\langle v_4^2, v_5^4 \rangle$
$\langle v_3^1, v_4^4 \rangle$	$\langle v_4^4, v_5^4 \rangle$

x_4	x_5
$\langle v_4^2, v_5^4 \rangle$	$\langle v_5^4, v_1^6 \rangle$

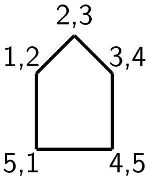
□

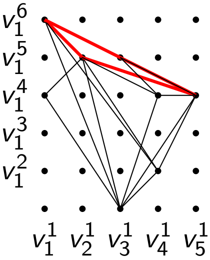
Main Theorem

Theorem (F., Koutris & Zhao, 23')

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

Proof.





x_1	x_2
$\langle v_1^6, v_2^5\rangle$	$\langle v_2^5, v_3^1\rangle$
$\langle v_1^4, v_2^5\rangle$	$\langle v_2^5, v_3^1\rangle$

x_2	x_3
$\langle v_2^5, v_3^1\rangle$	$\langle v_3^1, v_4^2\rangle$

x_3	x_4
$\langle v_3^5, v_4^4\rangle$	$\langle v_4^4, v_5^4\rangle$
$\langle v_3^1, v_4^2\rangle$	$\langle v_4^2, v_5^4\rangle$
$\langle v_3^1, v_4^4\rangle$	$\langle v_4^4, v_5^4\rangle$

x_4	x_5
$\langle v_4^2, v_5^4\rangle$	$\langle v_5^4, v_1^6\rangle$

x_5	x_1
$\langle v_5^4, v_1^6\rangle$	$\langle v_1^6, v_2^5\rangle$

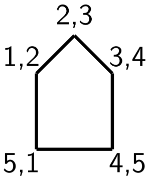
$\square$

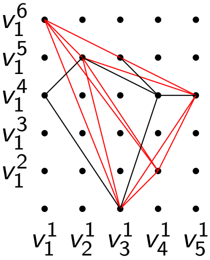
Main Theorem

Theorem (F., Koutris & Zhao, 23')

For any $\mathcal{H}$, CSP($\mathcal{H}$) cannot be computed via a combinatorial algorithm in time $O(|I|^{\text{emb}(\mathcal{H})-\epsilon})$ unless the Combinatorial k-Clique Conjecture is false.

Proof.





x_1	x_2
$\langle v_1^6, v_2^5\rangle$	$\langle v_2^5, v_3^1\rangle$
$\langle v_1^4, v_2^5\rangle$	$\langle v_2^5, v_3^1\rangle$

x_2	x_3
$\langle v_2^5, v_3^1\rangle$	$\langle v_3^1, v_4^2\rangle$

x_3	x_4
$\langle v_3^5, v_4^4\rangle$	$\langle v_4^4, v_5^4\rangle$
$\langle v_3^1, v_4^2\rangle$	$\langle v_4^2, v_5^4\rangle$
$\langle v_3^1, v_4^4\rangle$	$\langle v_4^4, v_5^4\rangle$

x_4	x_5
$\langle v_4^2, v_5^4\rangle$	$\langle v_5^4, v_1^6\rangle$

x_5	x_1
$\langle v_5^4, v_1^6\rangle$	$\langle v_1^6, v_2^5\rangle$

□

Semiring Oblivious Reduction

Semiring Oblivious Reduction

The proof can be adapted to tropical semiring (min k-clique) by assigning each pair $\{u, v\} \subseteq [k]$ to a unique hyperedge according to ψ.

Semiring Oblivious Reduction

The proof can be adapted to tropical semiring (min k-clique) by assigning each pair $\{u, v\} \subseteq [k]$ to a unique hyperedge according to ψ.

Theorem (F., Koutris & Zhao, 23')

For any $\mathcal{H}$, $\mathsf{CSP}(\mathcal{H})$ over tropical semiring cannot be computed via any randomized algorithm in time $O(|I|^{\mathsf{emb}(\mathcal{H})-\epsilon})$ unless the Min Weight k-Clique Conjecture is false.

Boolean Conjunctive Queries
Preliminaries
Algorithms

Sum-of-Product Computation
BCQ as CSP
Semiring framework

Fine-Grained Complexity

Our Work
Clique embedding power
Main results
Tightness and gaps

Summary

	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: Clique embedding power and submodular width for some classes of queries

Boolean Conjunctive Queries
- Preliminaries
- Algorithms

Sum-of-Product Computation
- BCQ as CSP
- Semiring framework

Fine-Grained Complexity

Our Work
- Clique embedding power
- Main results
- Tightness and gaps

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.
- Arturs Backurs and Piotr Indyk, *Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)*, SIAM J. Comput. **47** (2018), no. 3, 1087–1097.
- Todd J. Green, Gregory Karvounarakis, and Val Tannen, *Provenance semirings*, PODS, ACM, 2007, pp. 31–40.
- Martin Grohe, *The complexity of homomorphism and constraint satisfaction problems seen from the other side*, J. ACM **54** (2007), no. 1, 1:1–1:24.

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.

Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams, *Tight hardness for shortest cycles and paths in sparse graphs*, SODA, SIAM, 2018, pp. 1236–1252.

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.

References III

Mihalis Yannakakis, *Algorithms for acyclic database schemes*, VLDB, IEEE Computer Society, 1981, pp. 82–94.