Lower Bounds for Sum-Product Queries

Austen Z. Fan

University of Wisconsin, Madison

Preliminary Exam

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

1. Constraint Satisfaction Problem (CSP)

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

1. Constraint Satisfaction Problem (CSP)
2. Query Evaluation in Relational Databases

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

1. Constraint Satisfaction Problem (CSP)
2. Query Evaluation in Relational Databases
3. Inference in Bayesian Networks and Probabilistic Graphical Model

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

1. Constraint Satisfaction Problem (CSP)
2. Query Evaluation in Relational Databases
3. Inference in Bayesian Networks and Probabilistic Graphical Model
4. Chain Matrix Multiplication

Introduction

Sum-Product Queries are ubiquitous in theory and practice:

1. Constraint Satisfaction Problem (CSP)
2. Query Evaluation in Relational Databases
3. Inference in Bayesian Networks and Probabilistic Graphical Model
4. Chain Matrix Multiplication
5. ...

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

1. Fixing the relations

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

1. Fixing the relations $\rightarrow$ Dichotomy Theorems

   Decision CSP [Bul17, Zhu20] & #CSP [Bul13, DR13, CC17]

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

1. Fixing the relations $\rightarrow$ Dichotomy Theorems

   Decision CSP [Bul17, Zhu20] & #CSP [Bul13, DR13, CC17]

2. Fixing the induced hypergraph

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

1. Fixing the relations $\rightarrow$ Dichotomy Theorems

   Decision CSP [Bul17, Zhu20] & #CSP [Bul13, DR13, CC17]

2. Fixing the induced hypergraph $\rightarrow$ Class of queries

Introduction

Two ways to study Sum-Product Queries in theoretical literature:

1. Fixing the relations → Dichotomy Theorems

   Decision CSP [Bul17, Zhu20] & #CSP [Bul13, DR13, CC17]

2. Fixing the induced hypergraph → Class of queries

   Bounded arity [Gro07, Mar10] & Unbounded arity [Mar13]

What is the exact lower bound for a given Sum-Product Query?

We partially answer the above question:

We partially answer the above question:

1. Conditional lower bound via fine-grained complexity

We partially answer the above question:

1. Conditional lower bound via fine-grained complexity
2. Unconditional lower bound via monotone circuits

Outline

Definition

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Example

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Example

Deciding a (colored) 4-cycle

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \dots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \dots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Example

Deciding a (colored) 4-cycle

$$Q() \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Example

Deciding a (colored) 4-cycle

$$Q() \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

Listing (colored) 4-cycles

Definition

A *Conjunctive Query* Q is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each R_e is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$.

It is called *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

Example

Deciding a (colored) 4-cycle

$$Q() \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

Listing (colored) 4-cycles

$$Q(x_1, x_2, x_3, x_4) \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

Hypergraph

Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

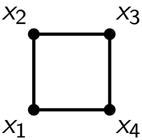
$Q() :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$

Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

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

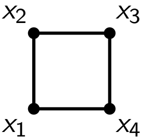


Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

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



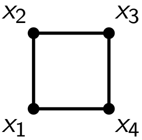
Remark

Hypergraph

For every (Boolean) CQ Q, we associate a *hypergraph* $\mathcal{H}$ to it, where the vertices are variables and the hyperedges are atoms.

Example

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



Remark

We are implicitly considering CQ without self-join. We will come back to this point for further work.

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

SAT: 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

SAT: 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

SAT: 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

SAT: 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.

Remark

*Fixing relations (*NP*) v.s. fixing hypergraphs (*P*).*

Semiring

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,
3. $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$.

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,
3. $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$.

Example

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,
3. $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$.

Example

$\mathbb{B} = (\{\textsc{False}, \textsc{True}\}, \vee, \wedge, \textsc{False}, \textsc{True})$

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,
3. $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$.

Example

$\mathbb{B} = (\{\text{False}, \ \text{True}\}, \vee, \wedge, \text{False}, \text{True})$
$\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$

Semiring

A (commutative) *semiring* is an algebraic structure $\mathbb{S} = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that:

1. $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids,
2. $\otimes$ is distributive over $\oplus$,
3. $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$.

Example

$\mathbb{B} = (\{\textsc{False}, \textsc{True}\}, \vee, \wedge, \textsc{False}, \textsc{True})$
$\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$
$\mathbb{C} = (\mathbb{N}, +, *, 0, 1)$

Sum-Product Computation over Semirings, I

Sum-Product Computation over Semirings, I

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

Sum-Product Computation over Semirings, I

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

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

Sum-Product Computation over Semirings, I

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

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

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

Sum-Product Computation over Semirings, 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

Sum-Product Computation over Semirings, 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

$\mathbb{B} \leftrightarrow$ set semantics

Sum-Product Computation over Semirings, 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

$\mathbb{B} \leftrightarrow$ set semantics

$\mathbb{C} \leftrightarrow$ bag semantics

Sum-Product Computation over Semirings, 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

$\mathbb{B} \leftrightarrow$ set semantics

$\mathbb{C} \leftrightarrow$ bag semantics

$\mathbb{T} \leftrightarrow$ optimization

Sum-Product Computation over Semirings, II

Sum-Product Computation over Semirings, II

Example

Sum-Product Computation over Semirings, II

Example

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

Sum-Product Computation over Semirings, II

Example

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

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

Sum-Product Computation over Semirings, II

Example

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

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

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

Sum-Product Computation over Semirings, II

Example

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

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

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

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

Provenance Polynomial, I

Provenance Polynomial, I

The *provenance polynomial* for a full CQ Q is parameterized by an underlying semiring $\mathbb{S}$, a hypergraph $\mathcal{H}$, and an instance I:

$$p_I^Q := \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x^e_{t[e]}$$

where $x^e_{t[e]}$ is a variable that captures the value of the tuple $t[e] \in R_e$ in the semiring domain $\mathbf{D}$ [GKT07].

Provenance Polynomial, I

The *provenance polynomial* for a full CQ Q is parameterized by an underlying semiring $\mathbb{S}$, a hypergraph $\mathcal{H}$, and an instance I:

$$p_I^Q := \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x^e_{t[e]}$$

where $x^e_{t[e]}$ is a variable that captures the value of the tuple $t[e] \in R_e$ in the semiring domain **D** [GKT07].

When we work over the counting semiring, the provenance polynomial becomes a polynomial:

$$p_I^{\mathcal{H}} := \sum_{t \in Q(I)} \prod_{e \in \mathcal{E}} x^e_{t[e]}$$

Provenance Polynomial, II

Example

Provenance Polynomial, II

Example

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

Provenance Polynomial, II

Example

$Q(x_1, x_2, x_3, x_4) : -R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$
$R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\}$

Provenance Polynomial, II

Example

$$Q(x_1, x_2, x_3, x_4) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$
$$R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\}$$
$$S(x_2, x_3) = \{(a_2, a_3), (c_2, d_3)\}$$

Provenance Polynomial, II

Example

$$
\begin{aligned}
&Q(x_1, x_2, x_3, x_4) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1) \\
&R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\} \\
&S(x_2, x_3) = \{(a_2, a_3), (c_2, d_3)\} \\
&T(x_3, x_4) = \{(a_3, a_4), (a_3, b_4), (d_3, c_4)\}
\end{aligned}
$$

Provenance Polynomial, II

Example

$$Q(x_1, x_2, x_3, x_4) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$
$$R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\}$$
$$S(x_2, x_3) = \{(a_2, a_3), (c_2, d_3)\}$$
$$T(x_3, x_4) = \{(a_3, a_4), (a_3, b_4), (d_3, c_4)\}$$
$$U(x_4, x_1) = \{(a_4, a_1), (b_4, a_1), (c_4, c_1)\}$$

Provenance Polynomial, II

Example

$$Q(x_1, x_2, x_3, x_4) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$
$$R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\}$$
$$S(x_2, x_3) = \{(a_2, a_3), (c_2, d_3)\}$$
$$T(x_3, x_4) = \{(a_3, a_4), (a_3, b_4), (d_3, c_4)\}$$
$$U(x_4, x_1) = \{(a_4, a_1), (b_4, a_1), (c_4, c_1)\}$$

$$Q(I) = \{(a_1, a_2, a_3, a_4), (a_1, a_2, a_3, b_4), (c_1, c_2, d_3, c_4)\}$$

Provenance Polynomial, II

Example

$$Q(x_1, x_2, x_3, x_4) :- R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$
$$R(x_1, x_2) = \{(a_1, a_2), (c_1, c_2)\}$$
$$S(x_2, x_3) = \{(a_2, a_3), (c_2, d_3)\}$$
$$T(x_3, x_4) = \{(a_3, a_4), (a_3, b_4), (d_3, c_4)\}$$
$$U(x_4, x_1) = \{(a_4, a_1), (b_4, a_1), (c_4, c_1)\}$$

$$Q(I) = \{(a_1, a_2, a_3, a_4), (a_1, a_2, a_3, b_4), (c_1, c_2, d_3, c_4)\}$$

$$\begin{aligned} p_I^Q = & (x_{a_1,a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,a_4} \otimes x_{a_4,a_1}) \oplus \\ & (x_{a_1.a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,b_4} \otimes x_{b_4,a_1}) \oplus \\ & (x_{c_1,c_2} \otimes x_{c_2,d_3} \otimes x_{d_3,c_4} \otimes x_{c_4,c_1}) \end{aligned}$$

Tree Decomposition

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

Tree Decomposition

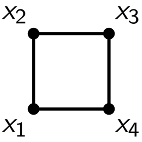
A *tree decomposition* of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and $\chi : V(\mathcal{T}) \rightarrow 2^{\mathcal{V}}$, such that (1) $\forall e \in \mathcal{E}$ is a subset for some $\chi(t), t \in V(\mathcal{T})$ and (2) $\forall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

Example

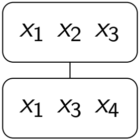
Tree Decomposition

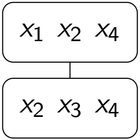
A *tree decomposition* of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and $\chi : V(\mathcal{T}) \rightarrow 2^{\mathcal{V}}$, such that (1) $\forall e \in \mathcal{E}$ is a subset for some $\chi(t), t \in V(\mathcal{T})$ and (2) $\forall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

Example

Two tree decompositions for 

are





Widths for CQs

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

The $\mathcal{F}$-width of a $\mathcal{H}$ is $\sup\{f\text{-width}(\mathcal{H}) \mid f \in \mathcal{F}\}$ [Mar13].

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

The $\mathcal{F}$-width of a $\mathcal{H}$ is $\sup\{f\text{-width}(\mathcal{H}) \mid f \in \mathcal{F}\}$ [Mar13].

Example

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

The $\mathcal{F}$-width of a $\mathcal{H}$ is $\sup\{f\text{-width}(\mathcal{H}) \mid f \in \mathcal{F}\}$ [Mar13].

Example

Let $s(B) = |B| - 1$. The *treewidth* of $\mathcal{H}$ is $\mathrm{tw}(\mathcal{H}) := s\text{-width}(\mathcal{H})$.

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

The $\mathcal{F}$-width of a $\mathcal{H}$ is $\sup\{f\text{-width}(\mathcal{H}) \mid f \in \mathcal{F}\}$ [Mar13].

Example

Let $s(B) = |B| - 1$. The *treewidth* of $\mathcal{H}$ is $\mathrm{tw}(\mathcal{H}) := s\text{-width}(\mathcal{H})$.

Let $\rho^*(\mathcal{H}) = \min_{\gamma} \sum_{e \in \mathcal{E}(\mathcal{H})} \gamma(e)$ where $\gamma : \mathcal{E}(\mathcal{H}) \to [0, 1]$ is a fractional edge cover. The *fractional hypertree width* of $\mathcal{H}$ is $\mathrm{fhw}(\mathcal{H}) := \rho^*\text{-width}(\mathcal{H})$.

Widths for CQs

The f-width of a TD $(\mathcal{T}, \chi)$ is $\max\{f(\chi(t)) \mid t \in V(\mathcal{T})\}$.

The f-width of a $\mathcal{H}$ is the minimum of f-widths of all its TDs.

The $\mathcal{F}$-width of a $\mathcal{H}$ is $\sup\{f\text{-width}(\mathcal{H}) \mid f \in \mathcal{F}\}$ [Mar13].

Example

Let $s(B) = |B| - 1$. The *treewidth* of $\mathcal{H}$ is $\mathrm{tw}(\mathcal{H}) := s\text{-width}(\mathcal{H})$.

Let $\rho^*(\mathcal{H}) = \min_{\gamma} \sum_{e \in \mathcal{E}(\mathcal{H})} \gamma(e)$ where $\gamma : \mathcal{E}(\mathcal{H}) \to [0, 1]$ is a fractional edge cover. The *fractional hypertree width* of $\mathcal{H}$ is $\mathrm{fhw}(\mathcal{H}) := \rho^*\text{-width}(\mathcal{H})$.

Remark

The famous Worst-Case Optimal Join achieves $O(m^{\mathrm{fhw}(\mathcal{H})})$ running time for computing $\mathcal{H}$ [NPRR18].

Submodular Width

Submodular Width

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

Submodular Width

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

Let $\mathcal{F}$ contain every edge-dominated monotone submodular function b on $\mathcal{V}(\mathcal{H})$ with $b(\emptyset) = 0$.

Submodular Width

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

Let $\mathcal{F}$ contain every edge-dominated monotone submodular function b on $\mathcal{V}(\mathcal{H})$ with $b(\emptyset) = 0$.

The *submodular width* of $\mathcal{H}$ is $\mathrm{subw}(\mathcal{H}) := \mathcal{F}\text{-width}(\mathcal{H})$.

Submodular Width

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

Let $\mathcal{F}$ contain every edge-dominated monotone submodular function b on $\mathcal{V}(\mathcal{H})$ with $b(\emptyset) = 0$.

The *submodular width* of $\mathcal{H}$ is $\mathrm{subw}(\mathcal{H}) := \mathcal{F}\text{-width}(\mathcal{H})$.

Theorem (Khamis, Ngo & Suciu, 16')

Any CSP($\mathcal{H}$) *can be computed in time* $\tilde{O}(m^{\mathrm{subw}(\mathcal{H})})$.

Submodular Width

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

Let $\mathcal{F}$ contain every edge-dominated monotone submodular function b on $\mathcal{V}(\mathcal{H})$ with $b(\emptyset) = 0$.

The *submodular width* of $\mathcal{H}$ is $\mathrm{subw}(\mathcal{H}) := \mathcal{F}\text{-width}(\mathcal{H})$.

Theorem (Khamis, Ngo & Suciu, 16')

Any CSP$(\mathcal{H})$ *can be computed in time* $\tilde{O}(m^{\mathrm{subw}(\mathcal{H})})$.

Remark

This will be the benchmark for our conditional lower bound.

Entropic Width

Entropic Width

A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n]$.

Entropic Width

A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n]$.

A set function is *entropic* if there exist random variables $A_1, \ldots, A_n$ such that $h(S) = H((A_i)_{i \in S})$ for any $S \subseteq [n]$, where H is the joint entropy of a set of variables.

Entropic Width

A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n]$.

A set function is *entropic* if there exist random variables $A_1, \ldots, A_n$ such that $h(S) = H((A_i)_{i \in S})$ for any $S \subseteq [n]$, where H is the joint entropy of a set of variables.

Let Γ_n^* be the set of all entropic functions of order n, and $\overline{\Gamma}_n^*$ the topological closure of Γ_n^*.

Entropic Width

A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n]$.

A set function is *entropic* if there exist random variables $A_1, \ldots, A_n$ such that $h(S) = H((A_i)_{i \in S})$ for any $S \subseteq [n]$, where H is the joint entropy of a set of variables.

Let Γ_n^* be the set of all entropic functions of order n, and $\overline{\Gamma}_n^*$ the topological closure of Γ_n^*.

The *entropic width* of $\mathcal{H}$ is $\mathrm{entw}(\mathcal{H}) := \overline{\Gamma}_n^*\text{-width}(\mathcal{H})$.

Entropic Width

A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n]$.

A set function is *entropic* if there exist random variables $A_1, \ldots, A_n$ such that $h(S) = H((A_i)_{i \in S})$ for any $S \subseteq [n]$, where H is the joint entropy of a set of variables.

Let Γ_n^* be the set of all entropic functions of order n, and $\overline{\Gamma}_n^*$ the topological closure of Γ_n^*.

The *entropic width* of $\mathcal{H}$ is $\mathrm{entw}(\mathcal{H}) := \overline{\Gamma}_n^*\text{-width}(\mathcal{H})$.

Remark

It remains open whether computing $\mathrm{entw}(\mathcal{H})$ *is even decidable.*

Degree Aware Entropic Width, I

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*.

Degree Aware Entropic Width, I

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*.

An instance I satisfies the constraints if $|\pi_Y(R_e \ltimes t_X)| \leq N_{Y|X}$ for every relation R_e in I with $X \subseteq Y \subseteq e$ and every tuple t_X.

Degree Aware Entropic Width, I

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*.

An instance I satisfies the constraints if $|\pi_Y(R_e \ltimes t_X)| \leq N_{Y|X}$ for every relation R_e in I with $X \subseteq Y \subseteq e$ and every tuple t_X.

Example

Degree Aware Entropic Width, I

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*.

An instance I satisfies the constraints if $|\pi_Y(R_e \ltimes t_X)| \leq N_{Y|X}$ for every relation R_e in I with $X \subseteq Y \subseteq e$ and every tuple t_X.

Example

A constraint of the form $(\emptyset, e, N_e)$ is simply a cardinality constraint.

Degree Aware Entropic Width, I

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*.

An instance I satisfies the constraints if $|\pi_Y(R_e \ltimes t_X)| \leq N_{Y|X}$ for every relation R_e in I with $X \subseteq Y \subseteq e$ and every tuple t_X.

Example

A constraint of the form $(\emptyset, e, N_e)$ is simply a cardinality constraint.

A constraint of the form $(X, Y, 1)$ is a Functional Dependency.

Degree Aware Entropic Width, II

Degree Aware Entropic Width, II

The degree constraints on an instance can be translated as constraints on entropic functions as follows:

$$\mathrm{HDC} := \left\{ h : 2^{[n]} \to \mathbb{R}_+ \mid \bigwedge_{(X,Y,N_{Y|X}) \in \mathrm{DC}} h(Y|X) \leq \log N_{Y|X} \right\}$$

where $h(Y|X) := h(Y) - h(X)$ [KNS17].

Degree Aware Entropic Width, II

The degree constraints on an instance can be translated as constraints on entropic functions as follows:

$$\mathrm{HDC} := \left\{ h : 2^{[n]} \to \mathbb{R}_+ \mid \bigwedge_{(X,Y,N_{Y|X}) \in \mathrm{DC}} h(Y|X) \leq \log N_{Y|X} \right\}$$

where $h(Y|X) := h(Y) - h(X)$ [KNS17].

The *degree-aware entropic width* of $\mathcal{H}$ is

$$\text{da-entw}(\mathcal{H}, \mathrm{HDC}) := (\overline{\Gamma}^*_n \cap \mathrm{HDC})\text{-width}(\mathcal{H}).$$

Degree Aware Entropic Width, II

The degree constraints on an instance can be translated as constraints on entropic functions as follows:

$$\mathrm{HDC} := \left\{ h : 2^{[n]} \to \mathbb{R}_+ \mid \bigwedge_{(X,Y,N_{Y|X}) \in \mathrm{DC}} h(Y|X) \leq \log N_{Y|X} \right\}$$

where $h(Y|X) := h(Y) - h(X)$ [KNS17].

The *degree-aware entropic width* of $\mathcal{H}$ is

$$\text{da-entw}(\mathcal{H}, \mathrm{HDC}) := (\overline{\Gamma}_n^* \cap \mathrm{HDC})\text{-width}(\mathcal{H}).$$

Remark

This will be the benchmark for our unconditional lower bound.

Fine-Grained Conjectures

Fine-Grained Conjectures

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.

Fine-Grained Conjectures

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.

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

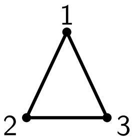
Definition (Touch)

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

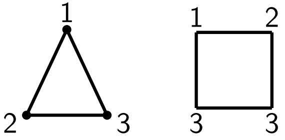
Definition (Touch)

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

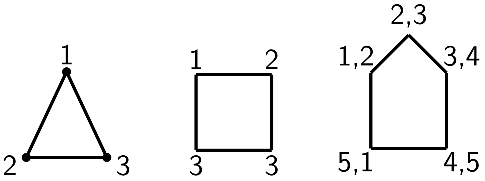
Definition (Touch)

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

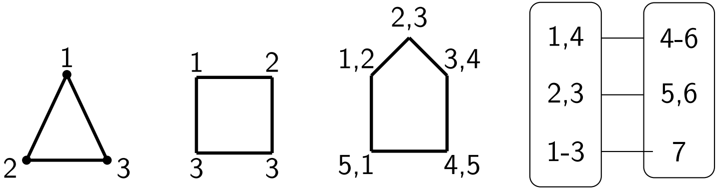
Definition (Touch)

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

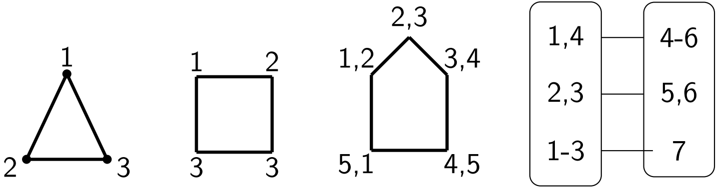
Definition (Weak Edge Depth)

$\forall e$ 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



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 $O(|I|^{\mathrm{emb}(\mathcal{H})-\epsilon})$ 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.

Remark

In a very recent work, Bringmann and Gorbachev showed that emb($\mathcal{H}$) *is tight for all* CSP($\mathcal{H}$) *that admits sub-quadratic algorithm [BG24].*

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.

Remark

In a very recent work, Bringmann and Gorbachev showed that $\mathrm{emb}(\mathcal{H})$ is tight for all $\mathrm{CSP}(\mathcal{H})$ that admits sub-quadratic algorithm [BG24].

In fact, it captures all $\mathcal{H}$ that admits sub-quadratic algorithm: If $\mathrm{CSP}(\mathcal{H})$ admits a sub-quadratic algorithm, then $\mathrm{emb}(\mathcal{H}) < 2$, and in that case there exists an $O(|I|^{\mathrm{emb}(\mathcal{H})})$ algorithm.

Semiring Oblivious Reduction

Semiring Oblivious Reduction

The proof can be adapted to tropical semiring (min k-clique) by assigning each pair $\{u, v\}$ 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 $\{u, v\}$ 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.

Examples

	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

Examples

	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

Remark

Bringmann and Gorbachev showed $\Omega(m^2)$ lower bound for both Q_b and Q_{hb} through MinConv conjecture [BG24].

Circuits over Semirings

Recall our provenance polynomial $p_I^Q = \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x_{t[e]}^e$.

Circuits over Semirings

Recall our provenance polynomial $p_I^Q = \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x^e_{t[e]}$.

A *circuit* F *over a semiring* $\mathbb{S}$ is a Directed Acyclic Graph (DAG) with input nodes variables in a set S_x containing $x^e_{t[e]}$'s and the constants $\mathbf{0}, \mathbf{1}$. Every other node is labelled by $\oplus$ or $\otimes$ and has fan-in 2; these nodes are called $\oplus$-*gates* and $\otimes$-*gates*, respectively.

Circuits over Semirings

Recall our provenance polynomial $p_I^Q = \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x_{t[e]}^e$.

A *circuit F over a semiring* $\mathbb{S}$ is a Directed Acyclic Graph (DAG) with input nodes variables in a set S_x containing $x_{t[e]}^e$'s and the constants $\mathbf{0}, \mathbf{1}$. Every other node is labelled by $\oplus$ or $\otimes$ and has fan-in 2; these nodes are called $\oplus$*-gates* and $\otimes$*-gates*, respectively.

A circuit F is said to *compute* a polynomial p if F and p coincide as functions (interpreted over the semiring $\mathbb{S}$), and is said to *produce* a polynomial p if F and p have exactly the same terms, i.e. monomials with their coefficients, syntactically.

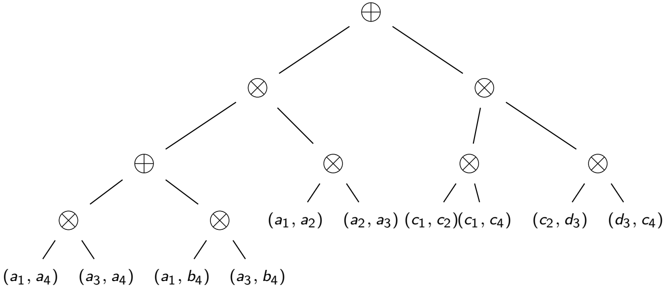
Example

$$p_I^Q = (x_{a_1,a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,a_4} \otimes x_{a_4,a_1}) \oplus$$
$$(x_{a_1.a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,b_4} \otimes x_{b_4,a_1}) \oplus$$
$$(x_{c_1,c_2} \otimes x_{c_2,d_3} \otimes x_{d_3,c_4} \otimes x_{c_4,c_1})$$

Example

$$
\begin{aligned}
p_I^Q = & (x_{a_1,a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,a_4} \otimes x_{a_4,a_1}) \oplus \\
& (x_{a_1.a_2} \otimes x_{a_2,a_3} \otimes x_{a_3,b_4} \otimes x_{b_4,a_1}) \oplus \\
& (x_{c_1,c_2} \otimes x_{c_2,d_3} \otimes x_{d_3,c_4} \otimes x_{c_4,c_1})
\end{aligned}
$$

$$\boxed{Q(x_1, x_2, x_3, x_4) \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)}$$



Motivation

1. Circuits can be seen as a computational model that corresponds to algorithms that *solely* exploit the algebraic semiring structure [Juk15].

Motivation

1. Circuits can be seen as a computational model that corresponds to algorithms that *solely* exploit the algebraic semiring structure [Juk15].

2. Circuits that compute the provenance polynomial of a CQ can be viewed as a concise representation of the corresponding provenance polynomial interpreted over the given semiring [OZ15, GKT07].

Main Results

Main Results

Theorem (F., Koutris & Zhao, 24')

For any $\epsilon > 0$ *and any hypergraph* $\mathcal{H}$*, there exists an instance* I *and* $k > 0$ *that satisfies the constraints* $\mathsf{HDC} \times k$ *such that any circuit* F *that computes the polynomial* $p_I^{\mathcal{H}}$ *over* $\{\mathbb{B}_{\mathsf{lin}}, \mathbb{T}, \mathbb{C}\}$ *has size*

$$\log |F| \geq (1 - \epsilon) \cdot \textit{da-entw}(\mathcal{H}, \mathsf{HDC} \times k).$$

Main Results

Theorem (F., Koutris & Zhao, 24')

For any $\epsilon > 0$ and any hypergraph $\mathcal{H}$, there exists an instance I and $k > 0$ that satisfies the constraints $\mathsf{HDC} \times k$ such that any circuit F that computes the polynomial $p_I^{\mathcal{H}}$ over $\{\mathbb{B}_{\mathsf{lin}}, \mathbb{T}, \mathbb{C}\}$ has size

$$\log |F| \geq (1 - \epsilon) \cdot \textit{da-entw}(\mathcal{H}, \mathsf{HDC} \times k).$$

Theorem (F., Koutris & Zhao, 24')

Let I be any instance that satisfies the degree constraint DC. There exists a multilinear and homogeneous circuit F of size $O(2^{\textit{da-entw}(\mathcal{H}, \mathsf{HDC})})$ that produces the polynomial $p_I^{\mathcal{H}}$ over any idempotent semiring.

Parse Tree

Parse Tree

A *parse tree* pt is a rooted tree in a circuit F defined inductively as follows:

Parse Tree

A *parse tree* pt is a rooted tree in a circuit F defined inductively as follows:

1. The root of pt is an output gate.

Parse Tree

A *parse tree* pt is a rooted tree in a circuit F defined inductively as follows:

1. The root of pt is an output gate.
2. If a $\otimes$-gate is in pt, include all of its children in F as its children in pt.

Parse Tree

A *parse tree* pt is a rooted tree in a circuit F defined inductively as follows:

1. The root of pt is an output gate.
2. If a $\otimes$-gate is in pt, include all of its children in F as its children in pt.
3. If a $\oplus$-gate is in pt, include exactly one of its children in F as its children in pt.

Parse Tree

A *parse tree* pt is a rooted tree in a circuit F defined inductively as follows:

1. The root of pt is an output gate.
2. If a $\otimes$-gate is in pt, include all of its children in F as its children in pt.
3. If a $\oplus$-gate is in pt, include exactly one of its children in F as its children in pt.

Remark

This notion has been extensively used to prove circuit lower bound [JS82, AI03].

TD from a Parse Tree

For a monomial q in $p_I^{\mathcal{H}}$, we define a structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ inductively in a bottom-up fashion from its parse tree:

1. For an input gate g of the variable $x^e_{t[e]}$, add a node v_g in $\mathcal{T}$ with $\chi(v_g) = e$. The input gate g is said to be *associated to* the node v_g.
2. For a $\oplus$-gate g, associate g to the node that is associated to g's single child in the parse tree.
3. For a $\otimes$-gate g, let g_1 and g_2 be its children. Let q_1, q_2 be the monomials computed at g_1, g_2 respectively; and B_g be the set of vertices $v \in \mathcal{V}(\mathcal{H})$ such that all hyperedges incident to v appear either exclusively in q_1 *or* exclusively in q_2. We add a node v_g with $\chi(v_g) = (\chi(v_{g_1}) \cup \chi(v_{g_2})) \setminus B_g$ as the parent of v_{g_1} and v_{g_2} in $\mathcal{T}$. We associate g with the new node v_g.

Key Lemmas

Key Lemmas

Lemma

For any monomial q in $p_I^{\mathcal{H}}$, the structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$.

Key Lemmas

Lemma

For any monomial q in $p_I^{\mathcal{H}}$, the structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$.

Lemma

Let q_1, q_2 be two monomials in $p_I^{\mathcal{H}}$ and $\mathcal{T}_{q_1} = (\mathcal{T}_1, \chi_1)$, $\mathcal{T}_{q_2} = (\mathcal{T}_2, \chi_2)$ be their corresponding tree decompositions. If the parse trees of q_1, q_2 share a common $\otimes$-gate g, then $\chi_1(v_g) = \chi_2(v_g)$.

Key Lemmas

Lemma

For any monomial q in $p_I^{\mathcal{H}}$, the structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$.

Lemma

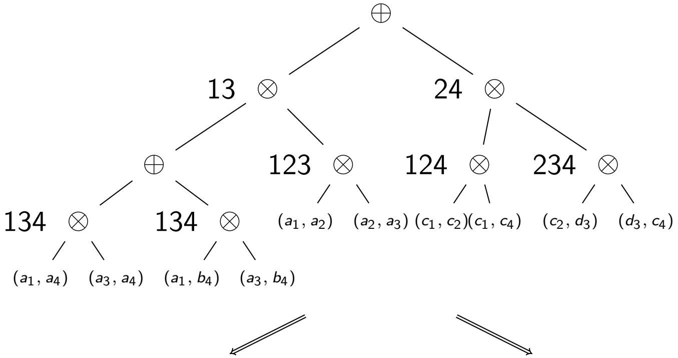
Let q_1, q_2 be two monomials in $p_I^{\mathcal{H}}$ and $\mathcal{T}_{q_1} = (\mathcal{T}_1, \chi_1)$, $\mathcal{T}_{q_2} = (\mathcal{T}_2, \chi_2)$ be their corresponding tree decompositions. If the parse trees of q_1, q_2 share a common $\otimes$-gate g, then $\chi_1(v_g) = \chi_2(v_g)$.

Remark

It is thus possible to assign a type $\mathrm{tp}(g)$ to each $\otimes$-gate g as $\chi(v_g)$ for some decomposition $\mathcal{T}_q = (\mathcal{T}, \chi)$ of a monomial q. In other words, the circuit F yields a globally consistent type assignment to each $\otimes$-gate in F.

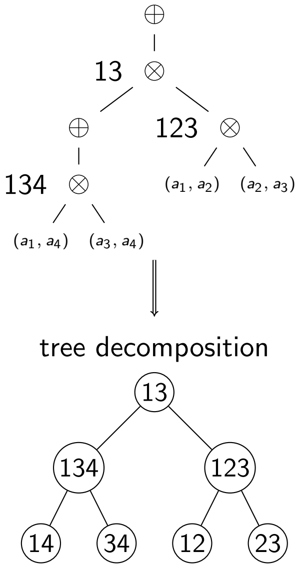
Example

$$Q(x_1, x_2, x_3, x_4) \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

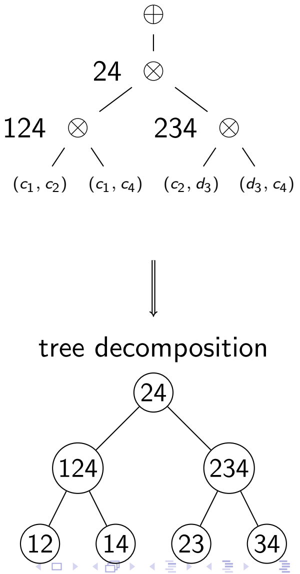


Example

parse tree for (a_1, a_2, a_3, a_4)



parse tree for (c_1, c_2, d_3, c_4)



Limit of Clique Embedding Power

? Does Bringmann and Gorbachev's characterization of clique embedding power for sub-quadratic queries extend [BG24]?

Limit of Clique Embedding Power

- ? Does Bringmann and Gorbachev's characterization of clique embedding power for sub-quadratic queries extend [BG24]?
- ✓ For planar graphs, a variant of clique embedding power is only constant factor away from tree width.

Limit of Clique Embedding Power

- ? Does Bringmann and Gorbachev's characterization of clique embedding power for sub-quadratic queries extend [BG24]?
- ✓ For planar graphs, a variant of clique embedding power is only constant factor away from tree width.
- ✗ There exists classes of graphs (e.g. expanders) where the gaps between that variant of clique embedding power and the tree widths are at least quadratic [GM09].

Limit of Clique Embedding Power

- ? Does Bringmann and Gorbachev's characterization of clique embedding power for sub-quadratic queries extend [BG24]?
- ✓ For planar graphs, a variant of clique embedding power is only constant factor away from tree width.
- ✗ There exists classes of graphs (e.g. expanders) where the gaps between that variant of clique embedding power and the tree widths are at least quadratic [GM09].
- ? What is the gap between the clique embedding power and submodular width [Mar13]?

Circuit for CQ with self-joins

? Can we provide tight circuit lower bounds for CQ with self-joins?

Circuit for CQ with self-joins

- **?** Can we provide tight circuit lower bounds for CQ with self-joins?
- ✓ Interesting connection to the notion of “minimal” queries [CS23] and the characterization of query containment parametrized by the underlying semiring [KRS12].

Circuit for Datalog

? Is the $O(n^3)$ size circuit for *st*-reachability given by Floyd-Warshall or Bellman-Ford optimal [KW90]?

Circuit for Datalog

- ? Is the $O(n^3)$ size circuit for *st*-reachability given by Floyd-Warshall or Bellman-Ford optimal [KW90]?
- ✓ We have obtained some results on dichotomies of regular language reachability ($\Omega(n^3)$ v.s. $O(n)$ circuit size).

Circuit for Datalog

? Is the $O(n^3)$ size circuit for *st*-reachability given by Floyd-Warshall or Bellman-Ford optimal [KW90]?

✓ We have obtained some results on dichotomies of regular language reachability ($\Omega(n^3)$ v.s. $O(n)$ circuit size).

? We are investigating the generalization of Bellman-Ford to arbitrary linear Datalog programs to construct logarithmic-depth circuit.

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.

Micah Adler and Neil Immerman, *An n! lower bound on formula size*, ACM Trans. Comput. Log. **4** (2003), no. 3, 296–314.

Karl Bringmann and Egor Gorbachev, *A fine-grained classification of subquadratic patterns for subgraph listing and friends*, arXiv preprint arXiv:2404.04369 (2024).

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.

Andrei A. Bulatov, *The complexity of the counting constraint satisfaction problem*, J. ACM **60** (2013), no. 5, 34:1–34:41.

References II

- ———, *A dichotomy theorem for nonuniform CSPs*, FOCS, IEEE Computer Society, 2017, pp. 319–330.
- Jin-Yi Cai and Xi Chen, *Complexity of counting CSP with complex weights*, J. ACM **64** (2017), no. 3, 19:1–19:39.
- Nofar Carmeli and Luc Segoufin, *Conjunctive queries with self-joins, towards a fine-grained enumeration complexity analysis*, PODS, ACM, 2023, pp. 277–289.
- Martin E. Dyer and David Richerby, *An effective dichotomy for the counting constraint satisfaction problem*, SIAM J. Comput. **42** (2013), no. 3, 1245–1274.
- Todd J. Green, Gregory Karvounarakis, and Val Tannen, *Provenance semirings*, PODS, ACM, 2007, pp. 31–40.

References III

- Martin Grohe and Dániel Marx, *On tree width, bramble size, and expansion*, J. Comb. Theory, Ser. B **99** (2009), no. 1, 218–228.
- 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.
- Mark Jerrum and Marc Snir, *Some exact complexity results for straight-line computations over semirings*, J. ACM **29** (1982), no. 3, 874–897.
- Stasys Jukna, *Lower bounds for tropical circuits and dynamic programs*, Theory Comput. Syst. **57** (2015), no. 1, 160–194.

References IV

- 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.
- Egor V. Kostylev, Juan L. Reutter, and András Z. Salamon, *Classification of annotation semirings over query containment*, PODS, ACM, 2012, pp. 237–248.
- Mauricio Karchmer and Avi Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, SIAM J. Discret. Math. **3** (1990), no. 2, 255–265.
- 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.

References V

- Dániel Marx, *Can you beat treewidth?*, Theory Comput. **6** (2010), no. 1, 85–112.
- ———, *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.
- Dan Olteanu and Jakub Závodný, *Size bounds for factorised representations of query results*, ACM Trans. Database Syst. **40** (2015), no. 1, 2:1–2:44.
- Mihalis Yannakakis, *Algorithms for acyclic database schemes*, VLDB, IEEE Computer Society, 1981, pp. 82–94.

References VI

Dmitriy Zhuk, *A proof of the CSP dichotomy conjecture*, J. ACM **67** (2020), no. 5, 30:1–30:78.

Sum-Product Computation over Semirings, III

Sum-Product Computation over Semirings, III

Example

Sum-Product Computation over Semirings, III

Example

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

Sum-Product Computation over Semirings, III

Example

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

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

Sum-Product Computation over Semirings, III

Example

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

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

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

Sum-Product Computation over Semirings, III

Example

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

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

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

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

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 $\mathsf{FPT} \neq \mathsf{W[1]}$ *the following are equivalent:*

1. $\mathsf{CSP}(\mathcal{C})$ *is polynomial-time solvable.*
2. $\mathsf{CSP}(\mathcal{C})$ *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 $\mathsf{FPT} \neq \mathsf{W}[1]$ *the following are equivalent:*

1. $\mathsf{CSP}(\mathcal{C})$ *is polynomial-time solvable.*
2. $\mathsf{CSP}(\mathcal{C})$ *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, $\mathsf{CSP}(\mathcal{C})$ *parametrized by $\mathcal{H}$ is fixed-parameter tractable if and only if $\mathcal{C}$ has bounded submodular width.*

Fine-Grained Complexity

Fine-Grained Complexity

"Hardness in easy problems"

Fine-Grained Complexity

"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

"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

"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

"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-$\textsc{Sat}$ requires $2^{\delta n}$ time.

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.

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-$\textsc{Sat}$ 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 $\epsilon > 0$.

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-$\textsc{Sat}$ 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 $\epsilon > 0$.

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.

...