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

Austen Z. Fan ✉ 📧
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Paraschos Koutris ✉ 📧
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Hangdong Zhao ✉ 📧
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

—— Abstract ——

We study the fine-grained complexity of evaluating Boolean Conjunctive Queries and their generalization to sum-of-product problems over an arbitrary semiring. For these problems, we present a general *semiring-oblivious* reduction from the k-clique problem to any query structure (hypergraph). Our reduction uses the notion of *embedding* a graph to a hypergraph, first introduced by Marx [20]. As a consequence of our reduction, we can show tight conditional lower bounds for many classes of hypergraphs, including cycles, Loomis-Whitney joins, some bipartite graphs, and chordal graphs. These lower bounds have a dependence on what we call the *clique embedding power* of a hypergraph H, which we believe is a quantity of independent interest. We show that the clique embedding power is always less than the submodular width of the hypergraph, and present a decidable algorithm for computing it. We conclude with many open problems for future research.

1 Introduction

In a seminal paper, Marx proved the celebrated result that $\mathrm{CSP}(\mathcal{H})$ is fixed-parameter tractable (FPT) if and only if the hypergraph $\mathcal{H}$ has a bounded submodular width [20]. In the language of database theory, a Boolean Conjunctive Query (BCQ) can be identified as the problem of $\mathrm{CSP}(\mathcal{H})$ where $\mathcal{H}$ is the hypergraph associated with the query [11]. Thus, Marx's result implies that a class of Boolean Conjunctive Queries is FPT if and only if its submodular width is bounded above by some universal constant. Built on this result, Khamis, Ngo, and Suciu introduced in [17] the PANDA (Proof-Assisted eNtropic Degree-Aware) algorithm, which can evaluate a BCQ[1] in time $\tilde{O}(|I|^{\mathsf{subw}(H)})$, where $|I|$ is the input size and $\mathsf{subw}(\mathcal{H})$ is the submodular width of $\mathcal{H}$ (here $\tilde{O}$ hides polylogarithmic factors). Remarkably, the running time of PANDA achieves the best known running time of *combinatorial algorithm*[2] for *all* BCQs. It is thus an important open question whether there exists a faster combinatorial algorithm than PANDA for some Boolean CQ.

---

[1] Technically, the PANDA algorithm works for Boolean or full CQs.

[2] Informally speaking, this requires the algorithm does not leverage fast matrix multiplication techniques.

To show that large submodular width implies not being FPT, Marx introduced the notion of an *embedding*, which essentially describes a reduction from one CSP problem to another. Our key insight in this work is that we can apply the notion of an embedding to measure how well *cliques* of different sizes can be embedded to a hypergraph $\mathcal{H}$. By taking the supremum over all possible clique sizes, we arrive at the definition of *clique embedding power*, denoted $\mathsf{emb}(\mathcal{H})$. The use of cliques as the starting problem means that we can use popular lower bound conjectures in fine-grained complexity (the Boolean k-Clique conjecture, the Min-Weight k-Clique conjecture) to obtain (conditional) lower bounds for the evaluation of BCQs that depend on $\mathsf{emb}(\mathcal{H})$.

Equipped with the new notion of the clique embedding power, we can show tight lower bounds for several classes of queries. That is, assuming the Boolean k-Clique Conjecture, we derive (conditional) lower bounds for many queries that meet their submodular width, and therefore the current best algorithm, up to polylogarithmic factors. In particular, we show that for cycles [2], Loomis-Whitney joins [22], and chordal graphs, among others, the current combinatorial algorithms are optimal.

We further extend the embedding reduction to be independent of the underlying (commutative) semiring[3]. It was observed by Green, Karvounarakis, and Tannen [10] that the semantics of CQs can be naturally generalized to sum-of-product operations over a semiring. This point of view unifies a number of database query semantics that seem unrelated. For example, evaluation over set semantics corresponds to evaluation over the Boolean semiring $\sigma_{\mathbb{B}} = (\{0,1\}, \vee, \wedge, 0, 1)$, while bag semantics corresponds to the semiring $(\mathbb{N}, +, \times, 0, 1)$. Interestingly, following this framework, the decision problem of finding a k-clique in a graph can be interpreted as the following sum-of-product operation: consider the input graph $G = (V, E)$ as the edge-weighted graph of the complete graph with $|V|$ vertices where $\mathsf{weight}(e) = \mathbb{1}_{e \in E}$; then the problem is to compute $\bigvee_{V' \subseteq V : |V'| = k} \bigwedge \mathsf{w}(\{v, w\})$. Observe that by changing the underlying semiring to be the tropical semiring $\mathsf{trop} = (\mathbb{R}^{\infty}, \min, +, \infty, 0)$, this formulation computes the min-weight k-clique problem. Indeed, given an edge-weighted graph (where the weight of non-existence edges is 0), the minimum weight of its k-clique is exactly $\min_{V' \subseteq V : |V'| = k} \sum \mathsf{w}(\{v, w\})$. We prove that the clique embedding reduction is *semiring-oblivious*, i.e., the reduction holds for arbitrary underlying semirings. This enables one to transfer the lower bound result independent of the underlying semiring and should be of independent interest.

Recent years have witnessed emerging interests in proving lower bounds for the runtime of database queries (see Durand [8] for a wonderful survey). Casel and Schmid consider the fine-grained complexity of regular path queries over graph databases [7]. Joglekar and Ré prove a full dichotomy for whether a 1-series-parallel graph admits a subquadratic algorithm [13]. Their proof is based on the hardness hypothesis that 3-XOR cannot be solved in subquadratic time. Perhaps the line of work in spirit closest to ours is the characterization of queries which can be enumerated by linear preprocessing time and constant delay [3, 6, 4]. However, their results focus on the enumeration problem and therefore are different from the main subject of our paper. Furthermore, their characterization mainly classifies queries based on the existence of a linear preprocessing time and constant delay algorithm. In contrast, our method can provide a lower bound for *every* query.

---

[3] A triple $(\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a commutative semiring if $\oplus$ and $\otimes$ are commutative binary operators over $\mathbf{D}$ with the following properties: (*i*) $(\mathbf{D}, \oplus)$ is a commutative monoid with an additive identity $\mathbf{0}$. (*ii*) $(\mathbf{D}, \otimes)$ is a commutative monoid with a multiplicative identity $\mathbf{1}$. (*iii*) $\otimes$ distributes over $\oplus$. (*iv*) For any element $e \in \mathbf{D}$, we have $e \otimes \mathbf{0} = \mathbf{0} \otimes e = \mathbf{0}$.

Our Contributions. We summarize our contributions as follows:

- We introduce the notion of the *clique embedding power* $\mathsf{emb}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ (Section 3). We show several interesting properties of this notion; most importantly, we show that it is always upper-bounded by the submodular width, $\mathsf{subw}(\mathcal{H})$. This connection can be seen as additional evidence of the plausibility of the lower bound conjectures for the k-clique problem.
- We show how to construct a reduction from the k-clique problem to any hypergraph $\mathcal{H}$ for any semiring, and discuss how the clique embedding power provides a lower bound for its running time (Section 4).
- We study how to compute $\mathsf{emb}(\mathcal{H})$ (Section 5). In particular, we prove that it is a decidable problem, and give a Mixed Integer Linear Program formulation. One interesting consequence of this formulation is that to achieve $\mathsf{emb}(\mathcal{H})$ it suffices to consider clique sizes that depend on the hypergraph size.
- We identify several classes of hypergraphs for which $\mathsf{emb}(\mathcal{H}) = \mathsf{subw}(\mathcal{H})$ (Section 6). For these classes of queries, our lower bounds match the best-known upper bounds if we consider the Boolean semiring with combinatorial algorithms or the tropical semiring. The most interesting class of hypergraphs we consider is the class of *chordal hypergraphs* (which captures chordal graphs).
- Finally, we identify a hypergraph with six vertices for which there is a gap between its clique embedding power and submodular width (Section 7). We believe that the existence of this gap leaves many open questions.

2 Background

In this section, we define the central problem, and notions necessary for our results.

The SumProduct Problem. We define this general problem following the notation in [16, 14]. Consider ℓ variables $x_1, x_2, \ldots, x_\ell$, where each variable x_i takes values in some discrete domain $\mathsf{Dom}(x_i)$. A *valuation v* is a function that maps each x_i to $\mathsf{Dom}(x_i)$. For a subset $S \subseteq [\ell]$, we define the tuple $\mathbf{x}_S = (x_i)_{i \in S}$ and $v(\mathbf{x}_S) = (v(x_i))_{i \in S}$.

The SumProduct Problem is parameterized by:

1. a commutative semiring $\sigma = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\mathbf{D}$ is a fixed domain.
2. a hypergraph $\mathcal{H} = (V, E)$ where $V = [\ell]$.

The input I specifies for every hyperedge $e \in E$ a function $R_e : \prod_{i \in e} \mathsf{Dom}(x_i) \to \mathbf{D}$. This function is represented in the input as a table of all tuples of the form $(\mathbf{a}_e, R_e(\mathbf{a}_e))$, such that $R_e(\mathbf{a}_e) \neq \mathbf{0}$. This input representation is standard in the CSP and database settings. We use $|I|$ to denote the input size, which is simply the sum of sizes of all tables in the input.

The SumProduct Problem then asks to compute the following function:

$$\bigoplus_{v:\text{valuation}} \bigotimes_{e \in E} R_e(v(\mathbf{x}_e)).$$

We will say that v is a *solution* for the above problem if $\bigotimes_{e \in E} R_e(v(\mathbf{x}_e)) \neq \mathbf{0}$.

Within this framework, we can capture several important problems depending on the choice of the semiring and the hypergraph. If we consider the Boolean semiring $\sigma_{\mathbb{B}} = (\{0,1\}, \vee, \wedge, 0, 1)$, then each R_e behaves as a relational instance (R_e is 1 if the tuple is in the instance, otherwise 0) and the SumProduct function captures Boolean Conjunctive Query evaluation. If $\sigma = (\mathbb{N}, +, \times, 0, 1)$ and R_e is defined as above, then the SumProduct function computes the number of solutions to a Conjunctive Query. Another important class of

problems is captured when we consider the min-tropical semiring $\mathsf{trop} = (\mathbb{R}^\infty, \min, +, \infty, 0)$ and we assign each tuple to a non-negative weight; this computes a minimum weight solution that satisfies the structural properties.

The Complexity for SumProduct Problems. We adopt the *random-access machine (RAM)* as our computation model with $O(\log n)$-bit words, which is standard in fine-grained complexity. The machine has read-only input registers and it contains the database and the query, read-write work memory registers, and write-only output registers. It is assumed that each register can store any tuple, and each tuple is stored in one register. The machine can perform all "standard" [4] operations on one or two registers in constant time.

In this paper, we are interested in the computational complexity of a SumProduct problem for a fixed hypergraph $\mathcal{H}$. (This is typically called *data complexity*). We will consider two different ways of treating semirings when we think about algorithms.

In the first variant, we fix the semiring σ along with the hypergraph $\mathcal{H}$. This means that the representation of the semiring is not part of the input and is known a priori to the algorithm. We denote this problem as $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$. In the second variant, we consider algorithms that access the semiring only via an oracle. In particular, the algorithm does not know the semiring a priori and can only access it during runtime by providing the values for the $\oplus, \otimes$ operations. We assume that each of these operations takes a constant amount of time. We denote this problem as $\mathsf{SumProd}\langle\mathcal{H}\rangle$.

Our goal in this paper is to specify the exact exponent of $|I|$ in the polynomial-time runtime cost of an algorithm that computes $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$ or $\mathsf{SumProd}\langle\mathcal{H}\rangle$.

Tree Decompositions. A *tree decomposition* of a hypergraph $\mathcal{H}$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and χ maps each node $t \in V(\mathcal{T})$ of the tree to a subset $\chi(t)$ of $V(\mathcal{H})$ such that:
1. every hyperedge $e \in E(\mathcal{H})$ is a subset of $\chi(t)$ for some $t \in V(\mathcal{T})$; and
2. for every vertex $v \in V(\mathcal{H})$, the set $\{t \mid v \in \chi(t)\}$ is a non-empty connected subtree of $\mathcal{T}$.

We say that a hypergraph $\mathcal{H}$ is *acyclic* if it has a tree decomposition such that each bag corresponds to a hyperedge.

Notions of Width. Let $\mathcal{H}$ be a hypergraph and F be a set function over $V(H)$. The F-width of a tree decomposition $(\mathcal{T}, \chi)$ is defined as $\max_t F(\chi(t))$. The F-width of $\mathcal{H}$ is the minimum F-width over all possible tree decompositions of $\mathcal{H}$.

A *fractional independent set* of a hypergraph $\mathcal{H}$ is a mapping $\mu : V(\mathcal{H}) \to [0, 1]$ such that $\sum_{v \in e} \mu(v) \leq 1$ for every $e \in E(\mathcal{H})$. We naturally extend functions on the vertices of $\mathcal{H}$ to subsets of vertices of $\mathcal{H}$ by setting $\mu(X) = \sum_{v \in X} \mu(v)$.

The *adaptive width* $\mathsf{adw}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is defined as the supreme of F-width$(\mathcal{H})$, where F goes over all fractional independent sets of $\mathcal{H}$. Hence if $\mathsf{adw}(\mathcal{H}) \leq w$, then for every μ, there exists a tree decomposition of $\mathcal{H}$ with μ-width at most w.

A set function F is *submodular* if for any two sets A, B we have $F(A \cup B) + F(A \cap B) \leq F(A) + F(B)$. It is monotone if whenever $A \subseteq B$, then $F(A) \leq F(B)$. The *submodular width* $\mathsf{subw}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is defined as the supreme of F-width$(\mathcal{H})$, where F now ranges over all non-negative, monotone, and submodular set functions over $V(\mathcal{H})$ such that for every hyperedge $e \in E(\mathcal{H})$, we have $F(e) \leq 1$. A non-negative, monotone, and submodular set function F is *edge-dominated* if $F(e) \leq 1$, for every $e \in E$.

---

[4] This includes all arithmetic (e.g. $+, -, \div, *$) and logical operations.

The *fractional hypertree width* of a hypergraph $\mathcal{H}$ is $\mathsf{fhw}(\mathcal{H}) = \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \rho^*(\chi(t))$, where ρ^* is the minimum fractional edge cover number of the set $\chi(t)$. It holds that $\mathsf{adw}(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H}) \leq \mathsf{fhw}(\mathcal{H})$.

It is known that $\mathsf{SumProd}\langle\sigma_\mathbb{B}, \mathcal{H}\rangle$ can be computed in time $\tilde{O}(|I|^{\mathsf{subw}(\mathcal{H})})$ using the PANDA algorithm [17]. However, we do not know of a way to achieve the same runtime for the general $\mathsf{SumProd}\langle\mathcal{H}\rangle$ problem. For this, the best known runtime is $\tilde{O}(|I|^{\#\mathsf{subw}(\mathcal{H})})$, where $\mathsf{subw}(\mathcal{H}) \leq \#\mathsf{subw}(\mathcal{H}) \leq \mathsf{fhw}(\mathcal{H})$ [14]. On the other hand, there are hypergraphs for which we can compute $\mathsf{SumProd}\langle\sigma_\mathbb{B}, \mathcal{H}\rangle$ with runtime better than $\tilde{O}(|I|^{\mathsf{subw}(\mathcal{H})})$ using non-combinatorial algorithms. For example, if $\mathcal{H}$ is a triangle we can obtain a runtime $\tilde{O}(|I|^{2\omega/(\omega+1)})$, where ω is the matrix multiplication exponent (the submodular width of the triangle is $3/2$).

Conjectures in Fine-Grained Complexity. Our lower bounds will be based on the following popular conjectures in fine-grained complexity. To state the conjectures, it will be helpful to define the *k-clique problem over a semiring* σ: given an undirected graph $G = (V, E)$ where each edge has a weight in the domain of the semiring, we are asked to compute the semiring-product over all the k-cliques in G, where the weight of each clique is the semiring-sum of clique edge weights.

▶ **Definition 1** (Boolean k-Clique Conjecture). *There is no real $\epsilon > 0$ such that computing the k-clique problem (with $k \geq 3$) over the Boolean semiring in an (undirected) n-node graph requires time $O(n^{k-\epsilon})$ using a combinatorial algorithm.*

▶ **Definition 2** (Min-Weight k-Clique Conjecture). *There is no real $\epsilon > 0$ such that computing the k-clique problem (with $k \geq 3$) over the tropical semiring in an (undirected) n-node graph with integer edge weights can be done in time $O(n^{k-\epsilon})$.*

When $k = 3$, min-weight 3-clique is equivalent to the All-Pairs Shortest Path (APSP) problem under subcubic reductions. The Min-Weight Clique Conjecture assumes the Min-Weight k-Clique conjecture for every integer $k \geq 3$ (similarly for the Boolean Clique Conjecture).

3   The Clique Embedding Power

In this section, we define the clique embedding power, a quantity central to this paper.

3.1   Graph Embeddings

We introduce first the definition of embedding a graph G to a hypergraph $\mathcal{H}$, first defined by Marx [20, 19]. We say that two sets of vertices $X, Y \subseteq V(\mathcal{H})$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or there is a hyperedge $e \in E(\mathcal{H})$ that intersects both X and Y. We say a hypergraph is connected if its underlying clique graph is connected.

▶ **Definition 3** (Graph Embedding). *Let G be an undirected graph, and $\mathcal{H}$ be a hypergraph. An embedding from G to $\mathcal{H}$, denoted $G \mapsto \mathcal{H}$, is a mapping ψ that maps every vertex $v \in V(G)$ to a non-empty subset $\psi(v) \subseteq V(\mathcal{H})$ such that the following hold:*
1. *$\psi(v)$ induces a connected subhypergraph;*
2. *if $u, v \in V(G)$ are adjacent in G, then $\psi(u), \psi(v)$ touch in $\mathcal{H}$.*

It will often be convenient to describe an embedding ψ by the reverse mapping $\psi^{-1}(x) = \{i \mid x \in \psi(i)\}$, where x is a vertex in $V(\mathcal{H})$. Given an embedding ψ and a vertex $v \in V(\mathcal{H})$, we define its *vertex depth* as $d_\psi(v) = |\psi^{-1}(v)|$. For a hyperedge $e \in E(\mathcal{H})$, we define its *weak edge depth* as $d_\psi(e) = |\{v \in V(G) \mid \psi(v) \cap e \neq \emptyset\}|$, i.e., the number of vertices of G that map to some variable in e. Moreover, we define the *edge depth* of e as $d_\psi^+(e) = \sum_{v \in e} d_\psi(v)$.

The *weak edge depth* of an embedding ψ can then be defined as $\mathsf{wed}(\psi) = \max_e d_\psi(e)$, and the edge depth as $\mathsf{ed}(\psi) = \max_e d_\psi^+(e)$. Additionally, we define as $\mathsf{wed}(G \mapsto \mathcal{H})$ the minimum weak edge depth of any embedding ψ from G to $\mathcal{H}$. Similarly for $\mathsf{ed}(G \mapsto \mathcal{H})$. It is easy to see that $\mathsf{wed}(G \mapsto \mathcal{H}) \leq \mathsf{ed}(G \mapsto \mathcal{H})$.

It will be particularly important for our purposes to think about embedding the k-clique graph C_k to an arbitrary hypergraph $\mathcal{H}$. In this case, it will be simpler to think of the vertices of G as the numbers $1, \ldots, k$ and the embedding ψ as a mapping from the set $\{1, \ldots, k\}$ to a subset of $V(\mathcal{H})$. We can now define the following quantity, which captures how well we can embed a k-clique to H for an integer $k \geq 3$:

$$\mathsf{emb}_k(\mathcal{H}) := \frac{k}{\mathsf{wed}(C_k \mapsto \mathcal{H})}.$$

▶ **Example 4.** Consider the hypergraph $\mathcal{H}$ with the following hyperedges:

$$\{x_1, x_2, x_3\}, \{x_1, y\}, \{x_2, y\}, \{x_3, y\}$$

We can embed the 5-clique into $\mathcal{H}$ as follows:

$$1 \to \{x_1\}, 2 \to \{x_2\}, 3 \to \{x_3\}, 4, 5 \to \{y\}.$$

It is easy to check that this is a valid embedding, since, for example, $1, 4$ touch at the edge $\{x_1, y\}$. Moreover, $\mathsf{wed}(C_5 \mapsto G) = 3$, hence $\mathsf{emb}_5(G) = 5/3$.

3.2 Embedding Properties

In this part, we will explore how $\mathsf{wed}(C_k \mapsto \mathcal{H})$ and $\mathsf{emb}_k(\mathcal{H})$ behave as a function of the size of the clique k. We start with some basic observations.

▶ **Proposition 5.** *For any hypergraph $\mathcal{H}$ and integer $k \geq 3$:*
1. $\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq k$.
2. $\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq \mathsf{wed}(C_{k+1} \mapsto H) \leq \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$.
3. *If $k = m \cdot n$, where $k, m, n \in \mathbb{Z}_{\geq 0}$, then $\mathsf{emb}_k(\mathcal{H}) \geq \mathsf{emb}_m(\mathcal{H})$.*

Proof. (1) We define an embedding ψ from a k-clique where $\psi(i) = V(\mathcal{H})$ for every $i = 1, \ldots, k$. It is easy to see that ψ is an embedding with weak edge depth k.

(2) For the first inequality, take any ψ_{k+1}, we can construct a ψ_k by only preserving the mapping ψ_{k+1} for $[k]$. Then, for any $e \in E(\mathcal{H})$, we have

$$\{y \in V(C_k) \mid \psi_k(y) \cap e \neq \emptyset\} \subseteq \{y \in V(C_{k+1}) \mid \psi_{k+1}(y) \cap e \neq \emptyset\}$$

Thus,

$$\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq \mathsf{wed}(\psi_k) := \max_{e \in E(\mathcal{H})} d_{\psi_k}(e) \leq \mathsf{wed}(\psi_{k+1}).$$

For the second inequality, take any ψ_k, we construct a ψ_{k+1} by preserving the mapping ψ_k and ψ_{k+1} maps $k+1$ to $V(\mathcal{H})$. Then, for any $e \in E(\mathcal{H})$, we have

$$d_{\psi_{k+1}}(e) = d_{\psi_k}(e) + 1$$

so $\mathsf{wed}(\psi_{k+1}) = \mathsf{wed}(\psi_k) + 1$ and in particular, we can take ψ_k such that

$$\mathsf{wed}(\psi_{k+1}) \leq \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$$

which implies that

$$\mathsf{wed}(C_{k+1} \mapsto \mathcal{H}) \le \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$$

(3) Suppose ψ is the embedding that achieves $\mathsf{emb}_m(\mathcal{H})$ for C_m. It suffices to construct an embedding ψ' for C_k which achieves the same quantity $\mathsf{emb}_m(\mathcal{H})$. To do so, we simply bundle every n vertices in C_k to be a "hypernode". That is, label the bundles as $b_1, \ldots, b_n$. and $\psi'(v) = \psi(i)$ if and only if $v \in B_i$. The embedding power given by ψ' is then

$$\frac{k}{\mathsf{wed}(\psi')} = \frac{mn}{\mathsf{wed}(\psi)n} = \frac{m}{\mathsf{wed}(\psi)} = \mathsf{emb}_m(\mathcal{H}). \qquad \blacktriangleleft$$

The first item of the above proposition tells us that $\mathsf{emb}_k(\mathcal{H}) \ge 1$ for any k. But how does $\mathsf{emb}_k(\mathcal{H})$ behave as k grows? We next show that $\mathsf{emb}_k(\mathcal{H})$ is always upper bounded by the submodular width of $\mathcal{H}$.

▶ **Lemma 6.** *Let $\mathcal{H}$ be a hypergraph. Take an embedding $\psi : C_k \mapsto \mathcal{H}$. Let $(\mathcal{T}, \chi)$ be a tree decomposition of $\mathcal{H}$. Then, there exists a node $t \in T$ such that for every $i = 1, \ldots, k$, $\psi(i) \cap \chi(t) \neq \emptyset$.*

Proof. For $i = 1, \ldots, k$, let $\mathcal{T}_i$ be the subgraph of $\mathcal{T}$ that includes all nodes $t \in V(\mathcal{T})$ such that $\psi(i) \cap \chi(t) \neq \emptyset$.

We first claim that $\mathcal{T}_i$ *forms a tree*. To show this, it suffices to show that $\mathcal{T}_i$ is connected. Indeed, take any two nodes t_1, t_2 in $\mathcal{T}_i$. This means that there exists $x_1 \in \chi(t_1) \cap \psi(i)$ and $x_2 \in \chi(t_2) \cap \psi(i)$. Since $x_1, x_2 \in \psi(i)$ and $\psi(i)$ induces a connected subgraph in $\mathcal{H}$, there exists a sequence of vertices $x_1 = z_1, \ldots, z_k = x_2$, all in $\psi(i)$, such that every two consecutive vertices belong to an edge of $\mathcal{H}$. Let $S_1, \ldots, S_k$ be the trees in $\mathcal{T}$ that contain $z_1, \ldots, z_k$ respectively. Take any two consecutive z_i, z_{i+1}: since they belong to the same edge, there exists a bag that contains both of them, hence S_i, S_{i+1} intersect. This means that there exists a path between t_1, t_2 in $\mathcal{T}$ such that every node is in T_i.

Second, we claim that *any two $\mathcal{T}_i, \mathcal{T}_j$ have at least one common vertex*. Indeed, $\psi(i), \psi(j)$ must touch in $\mathcal{H}$. If there exists a variable $x \in \psi(i) \cap \psi(j)$, then any vertex that contains x is a common vertex between $\mathcal{T}_i, \mathcal{T}_j$. Otherwise, there exists $x \in \psi(i), y \in \psi(j)$ such that x, y occur together in a hyperedge $e \in E(\mathcal{H})$. But this means that some node $t \in \mathcal{T}$ contains both x, y, hence $\mathcal{T}_i, \mathcal{T}_j$ intersect at t.

Finally, we apply the fact that a family of subtrees of a tree satisfies the *Helly property* [12], i.e. *a collection of subtrees of a tree has at least one common node if and only if every pair of subtrees has at least one common node.* Indeed, the trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$ satisfy the latter property, so there is a vertex t common to all of them. Such t has the desired property of the lemma. ◀

We can now state the following Theorem 7 on the embedding power of a hypergraph.

▶ **Theorem 7.** *For any hypergraph $\mathcal{H}$ and integer $k \ge 3$, the following holds:*

$$\mathsf{wed}(C_k \mapsto \mathcal{H}) \ge \frac{k}{\mathsf{subw}(\mathcal{H})}$$

Proof. Let $\mathsf{wed}(C_k \mapsto \mathcal{H}) = \alpha$. Then, there is an embedding $\psi : C_k \mapsto \mathcal{H}$ with weak edge depth α. We will show that $\mathsf{subw}(\mathcal{H}) \ge k/\alpha$.

First, we define the following set function over subsets of $V(\mathcal{H})$: for any $S \subseteq V(\mathcal{H})$, let $\mu(S) = |\{i \mid \psi(i) \cap S \neq \emptyset\}|/\alpha$. This is a coverage function, and hence it is a submodular function. It is also edge-dominated, since for any hyperedge e, we have $\mu(e) = |\{i \mid \psi(i) \cap e \neq \emptyset\}|/\alpha \le 1$.

Now, consider any decomposition $(\mathcal{T}, B_t)$ of $\mathcal{H}$. From Lemma 6, there is a node $t \in \mathcal{T}$ such that or every $i = 1, \ldots, k$, $\psi(i) \cap B_t \neq \emptyset$. Hence, $\mu(B_t) = |\{i \mid \psi(i) \cap B_t \neq \emptyset\}|/\alpha = k/\alpha$. Thus, the submodular width of the decomposition is at least k/α. ◀

The above result tells us that $\mathsf{emb}_k(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H})$ for any $k \geq 3$. Hence, taking the supremum of $\mathsf{emb}_k(\mathcal{H})$ for $k \geq 3$ is well-defined since the set is bounded. This leads us to the following definition:

▶ **Definition 8** (Clique Embedding Power). *Given a hypergraph* $\mathcal{H}$, *define the* clique embedding power *of* $\mathcal{H}$ *as*

$$\mathsf{emb}(\mathcal{H}) := \sup_{k \geq 3} \mathsf{emb}_k(\mathcal{H}) = \sup_{k \geq 3} \frac{k}{\mathsf{wed}(C_k \mapsto \mathcal{H})}.$$

The following is immediate:

▶ **Corollary 9.** *For any hypergraph* $\mathcal{H}$, $1 \leq \mathsf{emb}(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H})$.

For the connection between edge depth width and adaptive width, we have the following theorem analogous to Theorem 7. The proof can be found in [9].

▶ **Theorem 10.** *For any hypergraph* $\mathcal{H}$, *the following holds:*

$$\mathsf{ed}(C_k \mapsto \mathcal{H}) \geq \frac{k}{\mathsf{adw}(\mathcal{H})}$$

4   Lower Bounds

In this section, we show how to use the clique embedding power to obtain conditional lower bounds for SumProduct problems. Our main reduction follows the reduction used in [20], but also has to account for constructing the appropriate values for the semiring computations.

▶ **Theorem 11.** *For any hypergraph* $\mathcal{H}$ *and semiring* σ, *if* $\mathsf{SumProd}\langle \sigma, \mathcal{H} \rangle$ *can be solved in time* $O(|I|^c)$ *with input* I, *then* k-*Clique over* σ *can be solved in time* $O(n^{c \cdot \mathsf{wed}(C_k \mapsto \mathcal{H})})$ *where* n *is the number of vertices.*

Proof. We will show a reduction from the k-clique problem with n vertices over a semiring σ to $\mathsf{SumProd}\langle \sigma, \mathcal{H} \rangle$. Without loss of generality, we will assume that the input graph G to the k-clique problem is k-partite, with partitions $V_1, \ldots, V_k$. Indeed, given any graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, consider the k-partite graph $G^k = (V^k, E^k)$ where $V^k = \{v_i^j \mid 1 \leq i \leq n, 1 \leq j \leq k\}$ and for any two vertices $v_i^j, v_p^q \in V^k$, $\{v_i^j, v_p^q\} \in E^k$ iff $\{v_i, v_p\} \in E$ and $j \neq q$. Then there is a one-to-one mapping from a k-clique in G to a k-clique in G^k.

Let ψ be an embedding from C_k to $\mathcal{H}$ that achieves a weak edge depth $\lambda = k/\mathsf{emb}_k(\mathcal{H})$. As we mentioned before, it is convenient to take $V(C_k) = \{1, \ldots, k\}$. We now construct the input instance I for $\mathsf{SumProd}\langle \sigma, \mathcal{H} \rangle$. More explicitly, the task is to construct the function R_e for each hyperedge $e \in E$.

To this end, we first assign to each pair $\{i, j\} : i \neq j, i, j, \in \{1, 2, \ldots, k\}$ a hyperedge $\theta(\{i, j\}) = e \in E(\mathcal{H})$ satisfying the following conditions: $\psi(i) \cap e \neq \emptyset$ and $\psi(j) \cap e \neq \emptyset$. Such an e must exist by the definition of an embedding. If there is more than one hyperedge satisfying the condition, we arbitrarily choose one.

For every variable $x \in V(\mathcal{H})$, let $\psi^{-1}(x)$ be the subset of $\{1, \ldots, k\}$ mapping to x. Then, we define the domain $\mathsf{Dom}(x_i)$ of each variable x_i in the input instance as vectors over $[n]^{|\psi^{-1}(x_i)|}$. Let $S_e = \{i \in [k] \mid \psi(i) \cap e \neq \emptyset\}$. Note that $|S_e| = d_\psi(e) \leq \lambda$. Also, note

that $\psi^{-1}(x) \subseteq S_e$ for all $x \in e$. Then, we compute all cliques in the graph G between the partitions $V_i, i \in S_e$; these cliques will be of size $|S_e|$ and can be computed in running time $O(n^\lambda)$ by brute force.

For every clique $\{a_i \in V_i \mid i \in S_e\}$, let t be the tuple over $\prod_{i \in S_e} \mathsf{Dom}(x_i)$ such that its value at position x is $\langle a_i \mid i \in \psi^{-1}(x)\rangle$. Then, we set

$$R_e(t) = \mathbf{1} \otimes \bigotimes_{\{i,j\}:\theta(\{i,j\})=e} w(\{i,j\}).$$

In other words, we set the value as the semiring product of all the weights between the edges $\{a_i, a_j\}$ in the clique whenever the pair $\{i, j\}$ is assigned to the hyperedge e. All the other tuples are mapped to $\mathbf{0}$. By construction, the size of the input is $|I| = O(n^\lambda)$.

We now show that the two problems will return exactly the same output. To show this, we first show that there is a bijection between k-cliques in G and the solutions of the SumProduct instance.

$\Leftarrow$ Take a clique $\{a_1, \ldots, a_k\}$ in G. We map the clique to the valuation $v(x) = \langle a_i \mid i \in \psi^{-1}(x)\rangle$. This valuation is a solution to the SumProduct problem, since any subset of $\{a_1, \ldots, a_k\}$ forms a sub-clique. Hence for any hyperedge e, $R_e(v(\mathbf{x}_e)) \neq \mathbf{0}$.

$\Rightarrow$ Take a valuation v. Consider any $i \in \{1, \ldots, k\}$ and consider any two variables $x, y \in \psi(i)$ (recall that $\psi(i)$ must be nonempty). Recall that $x, y \in V(\mathcal{H})$. We claim that the i-th index in the valuation $v(x), v(y)$ must take the same value, which we will denote as a_i; this follows from the connectivity condition of the embedding. Indeed, since $x, y \in \psi(i)$, there exists a sequence of hyperedges $e_1, e_2, \ldots, e_m$ where $m \geq 1$ such that $e_j \cap e_{j+1} \neq \emptyset$ for $1 \leq j \leq m - 1$ and $x \in e_1$, $y \in e_m$. By the construction, the i-th index in $v(x)$ will then "propagate" to that in $v(y)$. This proves the claim. It then suffices to show that $\{a_1, \ldots, a_k\}$ is a clique in G. Indeed take any $i, j \in \{1, \ldots, k\}$. Since i and j are adjacent as two vertices in C_k, we know $\psi(i)$ and $\psi(j)$ touch. Therefore, there exists a hyperedge e that contains some $x \in \psi(i)$ and $y \in \psi(j)$. But this means that $\{a_i, a_j\}$ must form an edge in G.

We next show that the semiring product of the weights in the clique has the same value as the semiring product of the corresponding solution. Indeed:

$$\bigotimes_{e \in E} R_e(v(\mathbf{x}_e)) = \mathbf{1} \otimes \bigotimes_{e \in E} \bigotimes_{\{i,j\}:\theta(\{i,j\})=e} w(\{i,j\}) = \bigotimes_{\{i,j\}:i \neq j} w(\{i,j\})$$

where the last equality holds because each edge of the k-clique is assigned to exactly one hyperedge of $\mathcal{H}$.

The above claim together with the bijection show that the output will be the same; indeed, each the semiring sums will go over exactly the same elements with the same values.

To conclude the proof, suppose that $\mathsf{SumProd}\langle \sigma, \mathcal{H}\rangle$ could be answered in time $O(|I|^c)$ for some $c \geq 1$. This means that we can solve the k-clique problem over σ in time $O(n^\lambda + n^{c\lambda}) = O(n^{c \cdot \mathsf{wed}(C_k \mapsto \mathcal{H})})$. ◄

As an immediate consequence of Theorem 11, we can show the following lower bound.

▶ **Proposition 12.** *Under the Min-Weight k-Clique conjecture, $\mathsf{SumProd}\langle \mathsf{trop}, \mathcal{H}\rangle$ (and thus $\mathsf{SumProd}\langle \mathcal{H}\rangle$) cannot be computed in time $O(|I|^{\mathsf{emb}_k(\mathcal{H})-\epsilon})$ for any constant $\epsilon > 0$.*

Proof. Indeed, if $\mathsf{SumProd}\langle \mathsf{trop}, \mathcal{H}\rangle$ can be computed in time $O(|I|^{\mathsf{emb}_k(\mathcal{H})-\epsilon})$ for some constant $\epsilon > 0$, then by Theorem 11 the k-Clique problem over the tropical semiring can be solved in time $O(n^{(\mathsf{emb}_k(\mathcal{H})-\epsilon) \cdot \mathsf{wed}(C_k \mapsto \mathcal{H})}) = O(n^{k-\delta})$ for some $\delta > 0$. However, this violates the Min-Weight k-Clique conjecture. ◄

Similarly, we can show the following:

▶ **Proposition 13.** *Under the Boolean k-Clique conjecture,* $\mathsf{SumProd}\langle \sigma_\mathbb{B}, \mathcal{H} \rangle$ *(and thus* $\mathsf{SumProd}\langle \mathcal{H} \rangle$*) cannot be computed via a combinatorial algorithm in time* $O(|I|^{\mathsf{emb}_k(\mathcal{H}) - \epsilon})$ *for any constant* $\epsilon > 0$*.*

The above two results imply that to obtain the best lower bound, we need to find the clique size with the largest $\mathsf{emb}_k(\mathcal{H})$. However, the function $k \mapsto \mathsf{emb}_k(\mathcal{H})$ is really intriguing. It is not clear whether in the definition supremum is ever needed, i.e., whether there exists a hypergraph where the embedding power is achieved in the limit.

In Section 5, we show that for every hypergraph $\mathcal{H}$, there exists a natural number k such that $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_k(\mathcal{H})$. We also demonstrate how to compute $\mathsf{emb}(\mathcal{H})$ through a MILP and locate the complexity of computing the embedding power within the class 2-EXPTIME (double exponential time). The insight of our method is that, instead of computing the "integral" embedding power, one can consider the "fractional" embedding power and then recover the "integral" one by letting the clique size k to be sufficiently large.

5   Decidability of the Clique Embedding Power

To illustrate the algorithm for computing $\mathsf{emb}(\mathcal{H})$, it is instructive to first show how to compute $\mathsf{emb}_k(\mathcal{H})$ for a fixed clique size k.

5.1   An Integer Linear Program for $\mathsf{wed}(C_k \mapsto \mathcal{H})$

The following ILP formulation computes the minimum weak edge depth $w = \mathsf{wed}(C_k \mapsto \mathcal{H})$.

$$
\begin{aligned}
\min \quad & w \\
\text{s.t.} \quad & \sum_{S \subseteq V} x_S & = \ & k \\
& x_S & = \ & 0 & \forall S \subseteq V & \quad \text{where } S \text{ is not connected} \\
& \min\{x_S, x_T\} & = \ & 0 & \forall S, T \subseteq V & \quad \text{where } S, T \text{ do not touch} \\
& \sum_{S \subseteq V : e \cap S \neq \emptyset} x_S & \leq \ & w & \forall e \in \mathcal{E} \\
& x_S & \in \ & \mathbb{Z}_{\geq 0} & \forall S \subseteq V
\end{aligned} \tag{1}
$$

Each integer variable x_S, $S \subseteq V$, indicates how many vertices in C_k are assigned to the subset S. For example, if $x_{\{1,2\}} = 3$, this means in the embedding ψ, three vertices are mapped to the subset $\{1, 2\} \subseteq V$. It is sufficient to record only the number of vertices in C_k because of the symmetry of the clique. That is, since any two vertices are connected in C_k, one can arbitrarily permute the vertices in C_k so that the resulting map ψ' is still an embedding (given ψ is). Moreover, since the clique size k is fixed, to compute $\mathsf{emb}_k(\mathcal{H})$ it suffices to minimize w.

Observe that the condition $\min\{x_S, x_T\} = 0$ is not a linear condition. To encode it as such, we perform a standard transformation. We introduce a binary variable y_S for every set $S \subseteq V$. Then, we can write it as

$$
\left\{
\begin{array}{llll}
x_S & + & k \cdot y_S & \leq k \\
x_T & + & k \cdot y_T & \leq k \\
y_S & + & y_T & \geq 1
\end{array}
\right.
$$

Indeed, since y_S and y_T are binary variables, at least one of them is 1. Without loss of generality assume $y_S = 1$. Then $x_S = 0$ since $x_S \in \mathbb{Z}_{\geq 0}$. Therefore two subsets that do not touch cannot both be chosen in the embedding.

5.2 A Mixed Integer Linear Program for emb($\mathcal{H}$)

The above ILP construction does not directly yield a way to compute the clique embedding power, since the latter is defined to be the supremum for all k.
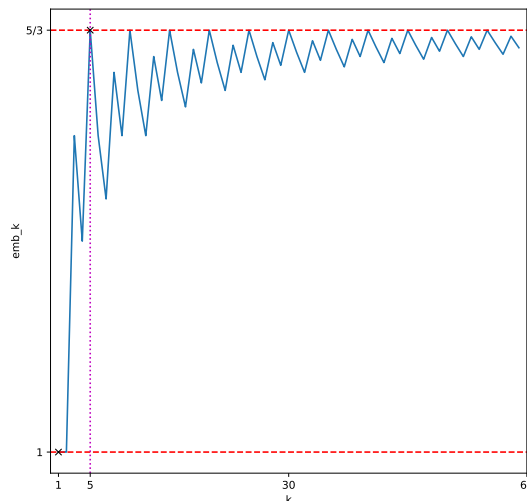
Figure 1 $\mathsf{emb}_k(\mathcal{H})$ for the 6-cycle.

As alluded before, the behavior of $\mathsf{emb}_k(\mathcal{H})$ as a function of k is non-trivial (and certainly not monotone). Figure 1 depicts how the clique embedding power changes with respect to different clique sizes for the 6-cycle, where the horizontal line represents the clique size.

To compute the supremum, the key idea is to change the integer variables x_S to be continuous (so they behave as fractions) and "normalize" the clique size k to 1. Specifically, we can write the following mixed integer linear program (MILP).

$$
\begin{aligned}
\min \quad & w \\
\text{s.t.} \quad & \sum_{S \subseteq V} x_S && = \; 1 \\
& x_S && = \; 0 && \forall S \subseteq V && \text{where } S \text{ is not connected} \\
& \min\{x_S, x_T\} && = \; 0 && \forall S, T \subseteq V && \text{where } S, T \text{ do not touch} \\
& \sum_{S \subseteq V : e \cap S \neq \emptyset} x_S && \leq \; w && \forall e \in \mathcal{E} \\
& x_S && \in \; \mathbb{R}_{\geq 0} && \forall S \subseteq V
\end{aligned}
\tag{2}
$$

▶ **Proposition 14.** *Let w^* be the optimal solution of MILP* (2). *Then, $\mathsf{emb}(\mathcal{H}) = 1/w^*$. Additionally, there exists an integer $K \geq 3$ such that $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H})$.*

Proof. We first show for any k, $\mathsf{emb}_k(\mathcal{H}) \geq 1/w^*$. Indeed, any embedding $\psi : C_k \to \mathcal{H}$ determines the values of the variables x_S in MILP (1). Let $\hat{x_S} = \frac{x_S}{k}$ and $\hat{w} = \frac{w}{k}$ be an assignment of the variables in MILP (2). It is easy to see that this is a feasible assignment. Thus, $w^* \leq \mathsf{wed}(C_k \mapsto \mathcal{H})/k$. Therefore $\mathsf{emb}_k(\mathcal{H}) = k/\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq 1/w^*$.

Next, observe that $\mathsf{emb}(\mathcal{H})$ is a rational number. In fact, the solution w^* for MILP (2) is a rational number, since every constant is a rational number [24]. Let K be the least common multiplier of their denominators of the fractions in the set $\{x_S\}$. Then, the assignment $K \cdot x_S, K \cdot w$ is a feasible solution for MILP (1) for $k = K$. This implies that $K \cdot w^* \geq \mathsf{wed}(C_K \mapsto \mathcal{H})$, so $\mathsf{emb}_K(\mathcal{H}) \geq 1/w^*$.

Table 1 Summary of emb and subw for some classes of queries.

	emb		subw	
Acyclic	1	[Theorem 29]	1	[27]
Chordal	=	[Theorem 24]	=	[22]
ℓ-cycle	$2 - 1/\lceil \ell/2 \rceil$	[Proposition 18]	$2 - 1/\lceil \ell/2 \rceil$	[2]
$K_{2,\ell}$	$2 - 1/\ell$	[Proposition 19]	$2 - 1/\ell$	[17]
$K_{3,3}$	2	[Proposition 20]	2	[17]
A_ℓ	$(\ell - 1)/2$	[Proposition 26]	$(\ell - 1)/2$	[22]
$\mathcal{H}_{\ell,k}$	ℓ/k	[Proposition 27]	ℓ/k	[22]
Q_b	17/9		2	[13]
Q_{hb}	7/4		2	[Proposition 30]

Thus, we have shown that $1/w^*$ is an upper bound for $\{\mathsf{emb}_k(\mathcal{H})\}_k$, but also $\mathsf{emb}_K(\mathcal{H}) = 1/w^*$. Hence, $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H}) = 1/w^*$. ◀

This leads to the following theorem (whose proof can be found in [9]).

▶ **Theorem 15.** *The problem of computing* $\mathsf{emb}(\mathcal{H})$ *for a hypergraph* $\mathcal{H}$ *is in 2-*EXPTIME *and, in particular, is decidable.*

Unfortunately, our method does not yield an upper bound on how large the K in Proposition 14 might be. There is no reason that K cannot be very large, e.g. doubly exponential to the size of $\mathcal{H}$. Some knowledge of that could be very useful in computing the clique embedding power. For example, one can compute all the embeddings from C_k, for k not greater than the upper bound, and output the one with the largest embedding power. The best-known upper bound we have so far is the following. The proof can be found in [9].

▶ **Proposition 16.** *For any hypergraph* $\mathcal{H}$*, there is a constant* $K = O((2^{|V|})!)$ *such that* $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H})$*.*

6 Examples of Tightness

In this section, we identify several classes of queries where the clique embedding power coincides with the submodular width. For brevity, we write emb, subw, fhw, and adw when the underlying hypergraph is clear under context. Table 1 summarizes our results.

6.1 Cycles

For the cycle query of length $\ell \geq 3$, we show that $\mathsf{emb} = \mathsf{subw} = 2 - 1/\lceil \ell/2 \rceil$. The best-known algorithm for ℓ-cycle detection (and counting) of Alon, Yuster, and Zwick [2] runs in time $O(|I|^{\mathsf{subw}})$. First, we show the following lemma.

▶ **Lemma 17.** *Consider the cycle query of length* $\ell \geq 3$*. Then* $\mathsf{emb} \geq 2 - 1/\lceil \frac{\ell}{2} \rceil$*.*

Proof. We start with the case where ℓ is odd and name the variables of the cycle query as $x_1, \ldots, x_\ell$. Then, we define $\lambda = (\ell + 1)/2$ and an embedding from a ℓ-clique as follows:

$$
\begin{aligned}
\psi^{-1}(x_1) &= \{1, 2, \ldots, \lambda - 1\} \\
\psi^{-1}(x_2) &= \{2, 3, \ldots, \lambda\} \\
&\quad \cdots \\
\psi^{-1}(x_\ell) &= \{2\lambda - 1, 1, \ldots, \lambda - 2\}
\end{aligned}
\tag{3}
$$

In other words, ψ maps each $i \in [\ell]$ into a consecutive segment consisting of $\lambda - 1$ vertices in the cycle. To see why ψ is an embedding, we observe that for any $i, j \in [\ell]$ such that $\psi(i) \cap \psi(j) = \emptyset$, $|\psi(i) \cup \psi(j)| = 2\lambda - 2 = \ell - 1$, so there is an edge that intersects both $\psi(i)$ and $\psi(j)$. It is easy to see that $\mathsf{wed}(\psi) = \lambda = (\ell + 1)/2$. Thus, $\mathsf{emb} \geq \ell/\lambda = 2\ell/(\ell + 1) = 2 - 2/(\ell + 1)$.

If ℓ is even, we define $\lambda = \ell/2$ and an embedding from a $(\ell - 1)$-clique as follows:

$$\psi^{-1}(x_1) = \{1, 2, \ldots, \lambda - 1\}$$
$$\psi^{-1}(x_2) = \{2, 3, \ldots, \lambda\}$$
$$\ldots$$
$$\psi^{-1}(x_{\ell-1}) = \{2\lambda - 2, 2\lambda - 1, 1, \ldots, \lambda - 3\}$$
$$\psi^{-1}(x_\ell) = \psi^{-1}(x_{\ell-1})$$

where $\psi^{-1}(x_i), i \in [\ell - 1]$ is exactly the embedding we constructed for $(\ell - 1)$-cycle. We show that this is a valid embedding. Let $i, j \in [\ell]$ such that $\psi(i) \cap \psi(j) = \emptyset$.
1. If $i \in \psi^{-1}(x_{\ell-1})$ (or $j \in \psi^{-1}(x_{\ell-1})$), then $|\psi(i) \cup \psi(j)| = \ell - 1$, so there is an edge that intersects both $\psi(i)$ and $\psi(j)$.
2. If $i, j \notin \psi^{-1}(x_{\ell-1})$, then $\psi(i)$ and $\psi(j)$ do not contain $x_{\ell-1}$ and x_ℓ. There is an edge that intersects both $\psi(i)$ and $\psi(j)$ since $|\psi(i) \cup \psi(j)| = \ell - 2$.

For this embedding, we have $\mathsf{wed}(\psi) = \lambda = \ell/2$, so $\mathsf{emb} \geq (\ell - 1)/\lambda = 2 - 2/\ell$. ◄

Thus, we have the following proposition.

▶ **Proposition 18.** *Consider the cycle query of length $\ell \geq 3$. Then we have*

$$\mathsf{emb} = \mathsf{subw} = 2 - \frac{1}{\lceil \frac{\ell}{2} \rceil}$$

Proof. It can be shown using Example 7.4 in [17] (setting $m = 1$) that $\mathsf{subw} \leq 2 - 1/\lceil \frac{\ell}{2} \rceil$ (technically the Example 7.4 in [17] only deals with cycles of even length, but their argument can be easily adapted to cycles of odd length). We thus conclude by applying Lemma 17 and Theorem 7. ◄

6.2 Complete Bipartite Graphs

We consider a complete bipartite graph $K_{m,n}$ where the two partitions of its vertices are $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$. We study two of its special cases, $K_{2,\ell}$ and $K_{3,3}$. The proofs of the following two propositions can be found in [9].

▶ **Proposition 19.** *For the bipartite graph $K_{2,\ell}$, $\mathsf{emb}(K_{2,\ell}) = \mathsf{subw}(K_{2,\ell}) = 2 - 1/\ell$.*

▶ **Proposition 20.** *For $K_{3,3}$, we have $\mathsf{emb}(K_{3,3}) = \mathsf{subw}(K_{3,3}) = 2$.*

Finding $\mathsf{emb}(K_{m,n})$ and $\mathsf{subw}(K_{m,n})$ in the most general case is still an open question.

6.3 Chordal Queries

In this section, we identify a special class of queries, called *choral queries*. We introduce necessary definitions and lemmas to prove that for a chordal query, emb, subw, fhw, and adw all coincide, as stated in Theorem 24.

Let G be a graph. A chord of a cycle C of G is an edge that connects two non-adjacent nodes in C. We say that G is *chordal* if any cycle in G of length greater than 3 has a chord. We can extend chordality to hypergraphs by considering the clique-graph of a hypergraph $\mathcal{H}$, where edges are added between all pairs of vertices contained in the same hyperedge.

Let $(\mathcal{T}, \chi)$ be a tree decomposition of a hypergraph $\mathcal{H}$ and $\mathsf{bags}(\mathcal{T}) \stackrel{\text{def}}{=} \{\chi(t) \mid t \in V(\mathcal{T})\}$. We say that $(\mathcal{T}, \chi)$ is *proper* if there is no tree decomposition $(\mathcal{T}', \chi')$ such that
1. for every bag $b_1 \in \mathsf{bags}(\mathcal{T}')$, there is a bag $b_2 \in \mathsf{bags}(\mathcal{T})$ such that $b_1 \subseteq b_2$;
2. $\mathsf{bags}(\mathcal{T}') \not\supseteq \mathsf{bags}(\mathcal{T})$,

The following important properties hold for chordal graphs.

▶ **Lemma 21** ([5]). *If G is a chordal graph and $(\mathcal{T}, \chi)$ is a proper tree decomposition of G, then the bags of $(\mathcal{T}, \chi)$, i.e. $\mathsf{bags}(\mathcal{T})$, are the maximal cliques in G.*

For chordal hypergraphs, we can show the following lemma:

▶ **Lemma 22.** *Let $\mathcal{H}$ be a hypergraph. Then, $(\mathcal{T}, \chi)$ is a (proper) tree decomposition of $\mathcal{H}$ if and only if it is a (proper) tree decomposition of the clique-graph of $\mathcal{H}$.*

Proof. We first show that $(\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$ if and only if it is also a tree decomposition of the clique-graph of $\mathcal{H}$. The forward direction is straightforward. For the backward direction, let $(\mathcal{T}, \chi)$ be a decomposition of the clique-graph of $\mathcal{H}$. Then for any hyperedge $e \in \mathcal{H}$ and any pair of vertices $u, v \in e$, we know that $\{t \mid u \in \chi(t)\} \cap \{t \mid v \in \chi(t)\} \neq \emptyset$. By the *Helly Property*, there is a bag that contains all vertices in the hyperedge e. Therefore, $(\mathcal{T}, \chi)$ is a tree decomposition for $\mathcal{H}$. It is easy to extend the proof for proper tree decompositions. ◀

The following corollary is immediate from both Lemma 21 and Lemma 22:

▶ **Corollary 23.** *If $\mathcal{H}$ is a chordal hypergraph and $(\mathcal{T}, \chi)$ is a proper tree decomposition of $\mathcal{H}$, then the bags of $(\mathcal{T}, \chi)$ are the maximal cliques in the clique-graph of $\mathcal{H}$.*

The above corollary tells us that every proper tree decomposition has the same set of bags, with the only difference being the way the bags are connected in the tree. From this, we can easily obtain that $\mathsf{subw} = \mathsf{fhw}$. However, we have an even stronger result:

▶ **Theorem 24.** *If $\mathcal{H}$ is a chordal hypergprah, then $\mathsf{emb} = \mathsf{adw} = \mathsf{subw} = \mathsf{fhw}$.*

Proof. Since $\mathcal{H}$ is chordal, by Corollary 23, the bags of any proper tree decomposition $(\mathcal{T}, \chi)$ of $\mathcal{H}$ are the maximal cliques in the clique-graph of $\mathcal{H}$. Then, there is a node $t \in V(\mathcal{T})$ such that the minimum fractional edge cover (also the maximum fractional vertex packing) of $\chi(t)$ is fhw. In particular, let $\{u_i \mid i \in \chi(t)\}$ be the optimal weights assigned to each vertex in $\chi(t)$ that obtain the maximum fractional vertex packing, so $\sum_{i \in \chi(t)} u_i = \mathsf{fhw}$. We let $\hat{u}_i = u_i / \sum_{i \in \chi(t)} u_i$ and k be the smallest integer such that $k \cdot \hat{u}_i$ is an integer for every $i \in \chi(t)$. Now we construct an embedding ψ from C_k to $\mathcal{H}$ so that every $\psi(j)$, for $j \in [k]$ is a singleton and for each $i \in \chi(t)$, let $d_\psi^{-1}(i) \stackrel{\text{def}}{=} k \cdot \hat{u}_i$. This assignment uses up all k vertices in C_k, since $\sum_{i \in \chi(t)} k \cdot \hat{u}_i = k$. Then,

$$\mathsf{wed}(\psi) = \max_{e \in E(\mathcal{H})} \sum_{i \in e} k \cdot \hat{u}_i = \frac{k}{\sum_{i \in \chi(t)} u_i} \cdot \max_{e \in E(\mathcal{H})} \sum_{i \in e} u_i \leq \frac{k}{\sum_{i \in \chi(t)} u_i}$$

and thus, we get $\mathsf{emb} = \mathsf{fhw}$ since

$$\mathsf{emb} \geq \mathsf{emb}(C_k \mapsto \mathcal{H}) \geq \frac{k}{\mathsf{wed}(\psi)} \geq \sum_{i \in \chi(t)} u_i = \mathsf{fhw}.$$

For adw, we define the following modular function over subsets of $V(\mathcal{H})$: for any $S \subseteq V(\mathcal{H})$, let $\mu(S) \stackrel{\text{def}}{=} \sum_{i \in S} u_i$. It is edge-dominated since for every hyperedge e, $\mu(e) = \sum_{i \in e} u_i \leq 1$. Moreover, we have that $\mu(\chi(t)) = \sum_{i \in \chi(t)} u_i = \mathsf{fhw}$. That is, $\mathsf{adw} \geq \mathsf{fhw}$, so $\mathsf{adw} = \mathsf{fhw}$. As a remark, it is also viable to use Lemma 3.1 in [17] to prove the claim for adaptive width. ◀

Recall that Corollary 23 implies if $\mathcal{H}$ is chordal, then every proper tree decomposition of $\mathcal{H}$ has the same set of bags. We show the converse is also true, which could be of independent interest. The proof is in [9].

▶ **Lemma 25.** *Let $\mathcal{H}$ be a hypergraph. If every proper tree decomposition of $\mathcal{H}$ has the same set of bags, then $\mathcal{H}$ is chordal.*

We identify three classes of hypergraphs (almost-cliques, hypercliques, and acyclic hypergraphs) that are chordal and find their clique embedding powers and submodular widths.

Almost-cliques. Consider the ℓ-clique where one vertex, say x_1, connects to k vertices only, where $1 \leq k < \ell - 1$ (hence it is the missing edges from being a ℓ-clique). We denote such a hypergraph as A_ℓ. To show that A_ℓ is chordal, we observe that for any cycle of length ≥ 4 that contains x_1, the two adjacent vertices of x_1 in the cycle must be connected by an edge in A_ℓ and that edge is a chord to the given cycle. We also show the following proposition.

▶ **Proposition 26.** *For an almost-cliques A_ℓ, $\mathsf{emb} = \mathsf{subw} = \mathsf{fhw} = (\ell - 1)/2$.*

Proof. To prove the claim, suppose WLOG x_i connects only to x_i, where $i \in 2, 3, \ldots, k$. Then, we take the decomposition with two bags: $\{x_1, x_2, \ldots, x_k\}$, $\{x_2, x_3, \ldots, x_\ell\}$, where each bag has an edge cover of at most $(\ell - 1)/2$ since the first bag induces an k-clique and the second bag induces an $(\ell - 1)$-clique. Hence, $\mathsf{fhw} \leq (\ell - 1)/2$.

On the other hand, consider the embedding ψ from the $(\ell - 1)$-clique, where $\psi(i) = x_{i+1}$, $1 \leq i \leq \ell - 1$; it is easy to verify that this is a valid embedding such that $\mathsf{wed}(\psi) = 2$, hence $\mathsf{emb} \geq (\ell - 1)/2$. Therefore, we have shown that $\mathsf{emb} = \mathsf{subw} = (\ell - 1)/2$ by Theorem 7. ◀

Hypercliques. Next, we consider the (ℓ, k)-hyperclique query $\mathcal{H}_{\ell,k}$, where $1 < k \leq \ell$. This query has ℓ variables, and includes as atoms all possible subsets of $\{x_1, \ldots, x_\ell\}$ of size exactly k. When $k = \ell - 1$, the query simply becomes a Loomis-Whitney join [22]. It is easy to see that $\mathcal{H}_{\ell,k}$ is chordal since the clique-graph of $\mathcal{H}_{\ell,k}$ is a ℓ-clique.

▶ **Proposition 27.** *For $\mathcal{H}_{\ell,k}$, we have $\mathsf{emb} = \mathsf{subw} = \mathsf{fhw} = \ell/k$.*

Proof. First, we show that $\mathsf{fhw} \leq \ell/k$. Indeed, there is a fractional edge cover that assigns a weight of $1/k$ to each hyperedge that contains k consecutive vertices in $\{x_1, \ldots, x_\ell\}$ (let the successor of x_ℓ be x_1). The fractional edge cover is then ℓ/k. We show next that this bound coincides with emb.

We simply define the embedding ψ from a ℓ-clique as $\psi(i) = x_i, i \in [\ell]$. Then, $\mathsf{wed}(\psi) = k$ since every hyperedge has exactly k vertices. Therefore, $\mathsf{emb} \geq \ell/k \geq \mathsf{fhw}$ and we conclude by applying Lemma 17 and Theorem 7. ◀

Acyclic Hypergraphs. First, we claim that acyclic queries are indeed chordal queries.

▶ **Lemma 28.** *An acyclic hypergraph $\mathcal{H}$ is chordal.*

Proof. We prove by induction on the number of hyperedges in the hypergraph $\mathcal{H} = (V, E)$. If $|E| = 1$, it is clique-graph is a clique, thus it is chordal. The induction hypothesis assumes the claim for acyclic hypergraphs with $|E| \leq k$ hyperedges. Let $\mathcal{H}$ be an acyclic hypergraph such that $|E| = k+1$. Since $\mathcal{H}$ is acyclic, it has a join forest whose vertices are the hyperedges of $\mathcal{H}$. Let $e_\ell \in E$ be a leaf of the join forest and it is easy to show that $\mathcal{H}' = (V, E \setminus \{e_\ell\})$ is an acyclic hypergraph with k hyperedges. For any cycle in the clique-graph of $\mathcal{H}$ having length ≥ 4, we discuss the following two cases.

If every edge of the cycle is in the clique-graph of $\mathcal{H}'$: by the induction hypothesis, there is a chord for this cycle in the clique-graph of $\mathcal{H}'$ (thus also in $\mathcal{H}$).

Otherwise, there is an edge e in the cycle that is in the clique-graph of $\mathcal{H}$, but not in the clique-graph of $\mathcal{H}'$: therefore, the edge e is only contained by e_ℓ. This implies that there is a vertex u that is only contained by e_ℓ, not by any other edges in E. Let $\{u, v\}$ and $\{u, w\}$ be the edges connecting u in the given cycle, we know that $\{u, v, w\} \subseteq e_\ell$ and thus, $\{v, w\}$ is a chord for the given cycle.   ◄

Now we prove the following theorem for acyclic hypergraphs:

▶ **Theorem 29.** *For an acyclic hypergprah $\mathcal{H}$,* emb = adw = subw = fhw = 1.
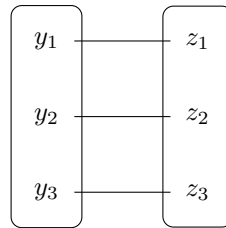
Proof. From Proposition 5, we know that emb ≥ 1. Since it is known that subw = fhw = 1, the theorem is then a direct result from Lemma 28 and Theorem 24.   ◄

7   Gap Between Clique Embedding Power and Submodular Width

In this section, we discuss the boat query and its variant depicted in Figure 2, where gaps between the clique embedding power and submodular width can be shown.
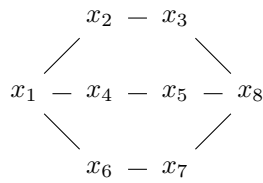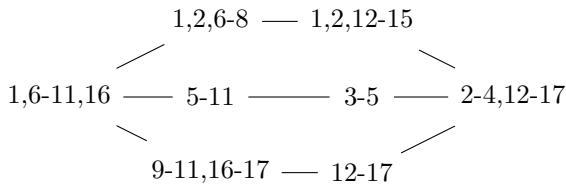

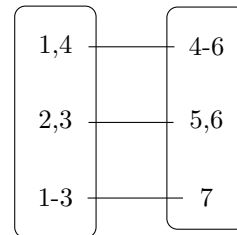**(a)** Boat Query Q_b.   **(b)** Hyper-boat Query Q_{hb}.

Figure 2 The Boat query and its variant, the Hyper-boat Query.

7.1   Clique Embedding Power, Submodular Width and Adaptive Width

(a) emb for Q_b.   **(b)** emb for Q_{hb}.

Figure 3 Optimal embedding for the boat query and its variants.

Using MILP (2), we find the optimal clique embedding for Q_b and Q_{hb}, as illustrated in Figure 3. The numbers represent the vertices from the clique, and we adopt the shorthand notation, say, 6-8 to refer to the set $\{6,7,8\}$. The clique embedding powers for Q_b and Q_{hb} are $\frac{17}{9}$ and $\frac{7}{4}$, respectively. However, [13] proves that for the boat query, subw$(Q_b) = 2$. This

implies a gap since $\mathsf{emb}(Q_b) = 17/9 < \mathsf{subw}(Q_b) = 2$. We show that for the hyper-boat query Q_{hb}, there is also a gap between the optimal clique embedding power and submodular width. In particular, we show that $\mathsf{subw}(Q_{hb}) = 2$ in the following proposition, which implies the following gap: $\mathsf{emb}(Q_{hb}) = \frac{7}{4} < \mathsf{subw}(Q_{hb}) = 2$. Its proof can be found in [9].

▶ **Proposition 30.** *For Q_{hb}, we have $\mathsf{subw}(Q_{hb}) = 2$.*

7.2 Subquadratic Equivalence Between Boat Queries

In this section, we demonstrate an interesting connection between the two boat queries. To start, let's consider Q_b and Q_{hb}. Both queries admit an algorithm that runs in time $O(|I|^2)$. Informally, we are going to show that either both queries can be executed significantly faster, or neither can. Following the seminal paper by Williams and Williams [25], we define *truly subquadratic algorithm* and *subquadratic equivalence.*

▶ **Definition 31.** *An algorithm is said to be* truly subquadratic *if it runs in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$ (m is the input size).*

Two problems A and B are *subquadratic equivalent* if A admits a truly subquadratic algorithm iff B admits a truly subquadratic algorithm. We show that the two boat queries are subquadratic equivalent.

▶ **Theorem 32.** *Q_b is subquadratic equivalent to Q_{hb}.*

Proof. It's easy to see that a truly subquadratic algorithm for Q_b gives a truly subquadratic algorithm for Q_{hb}. Indeed, given an input instance I_{hb} of Q_{hb}, we can form an input instance I_b of Q_b where the table (x_1, x_2) is the projection of the table (y_1, y_2, y_3) in I_{hb}, and similar for the tables $(x_1, x_4), (x_1, x_6), (x_3, x_8), (x_5, x_8)$ and (x_7, x_8). We then solve I_b by the algorithm for Q_b. It is easy to see that this algorithm is correct and runs in truly quadratic time.

The converse direction needs more work, since if we were to simply create the table (y_1, y_2, y_3) for Q_{hb} by joining the tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) for Q_b, the size of the result might be significantly greater than all previous tables. For example, if the sizes of the tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) are all m, then joining them could result in a table of size $m^{\frac{3}{2}}$ and therefore calling the algorithm for Q_{hb} on this instance does not necessarily yield a truly subquadratic algorithm for Q_b.

We perform our fine-grained reduction based on *heavy-light split*. Our goal is to give a subquadratic algorithm for Q_b assuming there is one such algorithm for Q_{hb}. Suppose the subquadratic algorithm for Q_b runs in time $O(m^{2-\delta})$ for some $\delta > 0$, where m is the size of all tables. Our algorithm for Q_b runs as follows. First, it checks whether there are entries of attribute x_1 that has degree more than $\Delta := m^\epsilon$ in tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) for some $\epsilon > 0$ to be specified later. Those are called *heavy* and there are at most $\frac{m}{\Delta}$ many of them. For those entries, we fix each one so that the remaining query becomes acyclic, and thus can be solved in linear time by Yannakakis algorithm [27]. We do the same procedure for heavy entries of attribute x_8. Therefore, any result of Q_b that contains a heavy entry in attributes x_1 or x_8 will be detected in time $O(m^{2-\epsilon})$. It remains to consider the case where the entries of attributes x_1 and x_8 have degrees less than Δ, which are called *light*. In this case, we loop over all light entries of x_1 in the table (x_1, x_2) and directly join them with the tables (x_1, x_4) and (x_1, x_6) and project the result to build a table (x_2, x_4, x_6). We then do the same procedure for joining x_8. This will cost time $O(m \cdot \Delta \cdot \Delta) = O(m^{1+2\epsilon})$. We then call the $O(m^{2-\delta})$ algorithm for Q_{hb}, which cost time $O(m^{(1+2\epsilon)(2-\delta)})$. By choosing $0 < \epsilon < \frac{\delta}{4-2\delta}$ (note that $\delta < 2$), we observe that the whole algorithm for Q_b takes time $O(m^{2-\epsilon}) + O(m^{(1+2\epsilon)(2-\delta)}) = O(m^{2-\epsilon'})$ for some $\epsilon' > 0$. ◀

We remark that the reduction from Q_{hb} to Q_b is parametrized by the running time of the algorithm for Q_{hb}. That is, the reduction is not uniform in the sense that only after given $\delta > 0$ can we specify a suitable ϵ. Theorem 32 implies that either both boat queries admit a truly subquadratic algorithm or none of them does.

The fact that there is a gap between $\mathsf{subw}(Q_{hb}) = 2$ and $\mathsf{emb}(Q_{hb}) = \frac{7}{4}$ suggests currently our lower bound does not match with the best upper bound, i.e., PANDA. This implies either that PANDA is not universally optimal, or that we are missing the best possible lower bound. We leave this as an open question.

Finally, we note that Theorem 4 in [13], which proves there does not exist a $\tilde{O}(m^{2-\epsilon} + |\mathrm{OUT}|^5)$ algorithm for the boat query unless 3-XOR can be solved in time $\tilde{O}(m^{2-t})$ for a $t > 0$, does not directly translate into the quadratic hardness for the boat query in our case. This is because their reduction uses the output of the boat query in an essential way to "hack back the collision" which is not available in the Boolean case.

8   Related Work

Fine-Grained Complexity.   The study of fine-grained complexity aims to show the (conditional) hardness of easy problems. Recent years have witnessed a bloom of development into this fascinating subject, resulting in many tight lower bounds which match exactly, or up to poly log factors, the running time of best-known algorithms [18, 26, 25, 1]. Among many others, popular hardness assumptions include the Strong Exponential Time Hypothesis (SETH), Boolean Matrix Multiplication (BMM), and All-Pairs Shortest Paths (APSP). Our work can be seen as a particular instance under this framework, i.e. using Boolean or Min-Weight k-Clique Conjecture to show conditional lower bounds for BCQs. Interestingly, our reduction of k-cycles essentially mirrors the construction in the proof of Theorem 3.1 in [18].

Conjunctive Queries (CQs) Evaluation.   The efficient evaluation of CQs constitutes the core theme of database theory. Khamis, Ngo, and Suciu introduced in [17] the PANDA algorithm that runs in time as predicted by the submodular width of the query hypergraph. This groundbreaking result establishes a profound connection between various lines of work on tree decompositions [19, 20], worst-case optimal join algorithms [23, 22], and the interplay between CQ evaluation and information theory [15, 28, 14].

Functional Aggregate Queries (FAQs).   FAQs [16] provides a Sum-of-Product framework that captures the semantics of conjunctive queries over arbitrary semirings. The semiring point-of-view originated from the seminal paper [10]. Khamis, Ngo, and Rudra [16] initiate the study of the efficient evaluation of FAQs. [14] introduces the FAQ version of the submodular width #subw and the #PANDA algorithm (as the FAQ version of the PANDA algorithm) that achieves the runtime as predicated by #subw. We show that the embedding from a k-clique into a hypergraph holds for arbitrary semirings, which enables one to transfer the hardness of k-clique to FAQ independent of the underlying semiring. To the best of our knowledge, this is the first *semiring-oblivious* reduction.

Enumeration and Preprocessing.   Bagan, Durand and Grandjean characterized in [3] when a constant delay and linear preprocessing algorithm for self-join-free conjunctive queries is possible. A recent paper [6] makes an initial foray towards the characterization of conjunctive

---

[5] $|\mathrm{OUT}|$ is the size of the output.

queries with self-joins. Also recently, [4] identifies new queries which can be solved with linear preprocessing time and constant delay. Their hardness results are based on the Hyperclique conjecture, the Boolean Matrix Multiplication conjecture, and the 3SUM conjecture.

9 Conclusion

In this paper, we study the fine-grained complexity of BCQs. We give a semiring-oblivious reduction from the k-clique problem to an arbitrary hypergraph. Assuming the Boolean k-Clique Conjecture, we obtain conditional lower bounds for many queries that match the combinatorial upper bound achieved by the best-known algorithms, possibly up to a poly-logarithmic factor.

One attractive future direction is to fully unravel the gap between the clique embedding power and submodular width, where improved lower bounds or upper bounds are possible. The Boolean k-Clique Conjecture states that there is no $O(n^{k-\epsilon})$ *combinatorial* algorithm for detecting k-cliques. One future direction is to base the hardness assumption over Nešetřil and Poljak's algorithm [21], which solves the k-clique problem in $O\left(n^{(\omega/3)k}\right)$ by leveraging fast matrix multiplication techniques and show lower bounds for any algorithm.

References

1   Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *FOCS*, pages 98–117. IEEE Computer Society, 2015.

2   Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

3   Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007.

4   Karl Bringmann and Nofar Carmeli. Unbalanced triangle detection and enumeration hardness for unions of conjunctive queries. *CoRR*, abs/2210.11996, 2022. `arXiv:2210.11996`.

5   Nofar Carmeli, Batya Kenig, Benny Kimelfeld, and Markus Kröll. Efficiently enumerating minimal triangulations. *Discret. Appl. Math.*, 303:216–236, 2021.

6   Nofar Carmeli and Luc Segoufin. Conjunctive queries with self-joins, towards a fine-grained complexity analysis. *CoRR*, abs/2206.04988, 2022. `arXiv:2206.04988`.

7   Katrin Casel and Markus L. Schmid. Fine-grained complexity of regular path queries. In *ICDT*, volume 186 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

8   Arnaud Durand. Fine-grained complexity analysis of queries: From decision to counting and enumeration. In *PODS*, pages 331–346. ACM, 2020.

9   Austen Z. Fan, Paraschos Koutris, and Hangdong Zhao. The fine-grained complexity of boolean conjunctive queries and sum-product problems. *CoRR*, 2023. `arXiv:2304.14557`.

10   Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40. ACM, 2007.

11   Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM (JACM)*, 54(1):1–24, 2007.

12   Pinar Heggernes. Treewidth, partial k-trees, and chordal graphs. *Partial curriculum in INF334-Advanced algorithmical techniques, Department of Informatics, University of Bergen, Norway*, 2005.

13   Manas Joglekar and Christopher Ré. It's all a matter of degree - using degree information to optimize multiway joins. *Theory Comput. Syst.*, 62(4):810–853, 2018.

14   Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. On functional aggregate queries with additive inequalities. In *PODS*, pages 414–431. ACM, 2019.

15   Mahmoud Abo Khamis, Phokion G. Kolaitis, Hung Q. Ngo, and Dan Suciu. Bag query containment and information theory. In *PODS*, pages 95–112. ACM, 2020.

16   Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: questions asked frequently. In *PODS*, pages 13–28. ACM, 2016.

17   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? In *PODS*, pages 429–444. ACM, 2017.

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

19   Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010.

20   Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, 2013.

21   Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

22   Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *J. ACM*, 65(3):16:1–16:40, 2018.

23   Hung Q. Ngo, Christopher Ré, and Atri Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Rec.*, 42(4):5–16, 2013.

24   Alexander Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

25   Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018.

26   Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *FOCS*, pages 786–797. IEEE, 2020.

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

28   Hangdong Zhao, Shaleen Deep, and Paraschos Koutris. Space-time tradeoffs for conjunctive queries with access patterns. *CoRR*, 2023. `arXiv:2304.06221`.