# General Space-Time Tradeoffs via Relational Queries

Shaleen Deep[1(✉)], Xiao Hu[2], and Paraschos Koutris[3]

[1] Microsoft GSL, Redmond, USA
shaleen.deep@microsoft.com
[2] University of Waterloo, Waterloo, Canada
xiaohu@uwaterloo.ca
[3] University of Wisconsin-Madison, Madison, USA
paris@cs.wisc.edu

**Abstract.** In this paper, we investigate space-time tradeoffs for answering Boolean conjunctive queries. The goal is to create a data structure in an initial preprocessing phase and use it for answering (multiple) queries. Previous work has developed data structures that trade off space usage for answering time and has proved conditional space lower bounds for queries of practical interest such as the path and triangle query. However, most of these results cater to only those queries, lack a comprehensive framework, and are not generalizable. The isolated treatment of these queries also fails to utilize the connections with extensive research on related problems within the database community. The key insight in this work is to exploit the formalism of relational algebra by casting the problems as answering join queries over a relational database. Using the notion of boolean *adorned queries* and *access patterns*, we propose a unified framework that captures several widely studied algorithmic problems. Our main contribution is three-fold. First, we present an algorithm that recovers existing space-time tradeoffs for several problems. The algorithm is based on an application of the *join size bound* to capture the space usage of our data structure. We combine our data structure with *query decomposition* techniques to further improve the tradeoffs and show that it is readily extensible to queries with negation. Second, we falsify two proposed conjectures in the existing literature related to the space-time lower bound for path queries and triangle detection for which we show unexpectedly better algorithms. This result opens a new avenue for improving several algorithmic results that have so far been assumed to be (conditionally) optimal. Finally, we prove new conditional space-time lower bounds for star and path queries.

## 1 Introduction

Recent work has made remarkable progress in developing data structures and algorithms for answering set intersection problems [12], reachability oracles and directed reachability [3,4,9], histogram indexing [7,18], and problems related to document retrieval [2,20]. This class of problems splits an algorithmic task


P. Morin and S. Suri (Eds.): WADS 2023, LNCS 14079, pp. 309–325, 2023.
https://doi.org/10.1007/978-3-031-38906-1_21

into two phases: the *preprocessing phase*, which computes a space-efficient data structure, and the *answering phase*, which uses the data structure to answer the requests to minimize the answering time. A fundamental algorithmic question related to these problems is the tradeoff between the space $S$ necessary for data structures and the answering time $T$ for requests.

For example, consider the 2-Set Disjointness problem: given a universe of elements $U$ and a collection of $m$ sets $C_1, \ldots, C_m \subseteq U$, we want to create a data structure such that for any pair of integers $1 \leq i, j \leq m$, we can efficiently decide whether $C_i \cap C_j$ is empty or not. Previous work [9,12] has shown that the space-time tradeoff for 2-Set Disjointness is captured by the equation $S \cdot T^2 = N^2$, where $N$ is the total size of all sets. The data structure obtained is conjectured to be optimal [12], and its optimality was used to develop conditional lower bounds for other problems, such as approximate distance oracles [3,4]. Similar tradeoffs have been independently established for other data structure problems as well. In the $k$-Reachability problem [8,12] we are given as an input a directed graph $G = (V, E)$, an arbitrary pair of vertices $u, v$, and the goal is to decide whether there exists a path of length $k$ between $u$ and $v$. In the edge triangle detection problem [12], we are given an input undirected graph $G = (V, E)$, and the goal is to develop a data structure that takes space $S$ and can answer in time $T$ whether a given edge $e \in E$ participates in a triangle or not. Each of these problems has been studied in isolation and, as a result, the algorithmic solutions are not generalizable.

In this paper, we cast many of the above problems into answering *Conjunctive Queries (CQs)* over a relational database. CQs are a powerful class of relational queries with widespread applications in data analytics and graph exploration [11,30,31]. For example, by using the relation $R(x, y)$ to encode that element $x$ belongs to set $y$, 2-Set Disjointness can be captured by the following CQ: $\varphi(y_1, y_2) = R(x, y_1) \wedge R(x, y_2)$. The insight of casting data structure problems into CQs over a database allows for a unified treatment for developing algorithms within the same framework. In particular, we can leverage the techniques developed by the data management community through a long line of research on efficient join evaluation [22,23,32], including worst-case optimal join algorithms [22] and tree decompositions [13,26]. Building upon these techniques, we achieve the following:

- We obtain in a simple way general space-time tradeoffs for any Boolean CQ (a Boolean CQ is one that outputs only true or false). As a consequence, we recover state-of-the-art tradeoffs for several existing problems (e.g., 2-Set Disjointness as well as its generalization $k$-Set Disjointness and $k$-Reachability) as special cases of the general tradeoff. We can even obtain improved tradeoffs for some specific problems, such as edge triangles detection, thus falsifying existing conjectures. This also gives us a way to construct data structures for any new problem that can be cast as a Boolean CQ (e.g., finding any subgraph pattern in a graph).
- Space-time tradeoffs for enumerating (non-Boolean) query results under static and dynamic settings have been a subject of previous work [1,11,14,16,17,24].

The space-time tradeoffs from [11] can be applied to the setting of this paper by stopping the enumeration after the first result is observed. We improve upon this result by (*i*) showing a much simpler data structure construction and proofs, and (*ii*) shaving off a polylogarithmic factor from the tradeoff.

We next summarize our three main technical contributions.

1. We propose a unified framework that captures several widely-studied data structure problems. More specifically, we use the formalism of CQs and the notion of *Boolean adorned queries*, where the values of some variables in the query are fixed by the user (denoted as an *access pattern*), and aim to evaluate the Boolean query. We then show how this framework captures the 2-Set Disjointness and $k$-Reachability problems. Our first main result (Theorem 1) is an algorithm that builds a data structure to answer any Boolean CQ under a specific access pattern. We show how to recover existing and new tradeoffs using this general framework. The first main result may sometimes lead to suboptimal tradeoffs since it does not take into account the structural properties of the query. Our second main result (Theorem 2) combines tree decompositions of the query structure with access patterns to improve space efficiency. We then show how this algorithm can handle Boolean CQs with negation.
2. We explicitly improve the best-known space-time tradeoff for the $k$-Reachability problem for $k \geq 4$. For any $k \geq 2$, the tradeoff of $S \cdot T^{2/(k-1)} = O(|E|^2)$ was conjectured to be optimal by [12], where $|E|$ is the number of edges in the graph, and was used to conditionally prove other lower bounds on space-time tradeoffs. We show that for a regime of answer time $T$, it can be improved to $S \cdot T^{2/(k-2)} = O(|E|^2)$, thus breaking the conjecture. To the best of our knowledge, this is the first non-trivial improvement for the $k$-Reachability problem. We also refute a lower bound conjecture for the edge triangles detection problem established by [12] that appeared at WADS'17.
3. Our third main contribution applies our framework to CQs with negation. This allows us to construct space-time tradeoffs for tasks such as detecting open triangles in a graph. We also show a reduction between lower bounds for the problem of $k$-Set Disjointness for $k \geq 2$, which generalizes the 2-Set Disjointness to computing the intersection between $k$ given sets.

## 2 Notation and Preliminaries

***Data Model.*** A *schema* is defined as a collection of relation names, where each relation name $R$ is associated with an arity $n$. Assuming a (countably infinite) domain **dom**, a tuple $t$ of relation $R$ is an element of $\mathbf{dom}^n$. An instance of relation $R$ with arity $n$ is a finite set of tuples of $R$; the size of the instance will be denoted as $|R|$. An input database $D$ is a set of relation instances over the schema. The size of the database $|D|$ is the sum of sizes of all its instances.

***Conjunctive Queries.*** A *Conjunctive Query* (CQ) is an expression of the form $\varphi(\mathbf{y}) = R_1(\mathbf{x}_1) \wedge R_2(\mathbf{x}_2) \wedge \ldots \wedge R_n(\mathbf{x}_n)$. The expressions $\varphi(\mathbf{y}), R_1(\mathbf{x}_1), R_2(\mathbf{x}_2), \ldots, R_n(\mathbf{x}_n)$ are called *atoms*. The atom $\varphi(\mathbf{y})$ is the *head* of the query, while the atoms $R_i(\mathbf{x}_i)$ form the *body*. Here, $\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_n$ are vectors where each position is a variable (typically denoted as $x, y, z, \ldots$) or a constant from **dom** (typically denoted $a, b, c, \ldots$). Each $\mathbf{x}_i$ must match the arity of the relation $R_i$, and the variables in $\mathbf{y}$ must occur in the body of the query. We use $\mathsf{vars}(\varphi)$ to denote the set of all variables occurring in $\varphi$, and $\mathsf{vars}(R_i)$ to denote the set of variables in atom $R_i(\mathbf{x}_i)$. A CQ is *full* if every variable in the body appears also in the head, and *Boolean* if the head contains no variables. Given variables $x_1, \ldots, x_k$ from $\mathsf{vars}(\varphi)$ and constants $a_1, \ldots, a_k$ from **dom**, we define $\varphi[a_1/x_1, \ldots, a_k/x_k]$ to be the CQ where every occurrence of a variable $x_i$, $i = 1, \ldots, k$, is replaced by the constant $a_i$. Given an input database $D$ and a CQ $\varphi$, we define the query result $\varphi(D)$ as follows. A *valuation* $v$ is a mapping from $\mathbf{dom} \cup \mathsf{vars}(\varphi)$ to **dom** such that $v(a) = a$ whenever $a$ is a constant. Then, $\varphi(D)$ is the set of all tuples $t$ such that there exists a valuation $v$ for which $t = v(\mathbf{y})$ and for every atom $R_i(\mathbf{x}_i)$, we have $R_i(v(\mathbf{x}_i)) \in D$.[1]

*Example 1.* Suppose that we have a directed graph $G$ that is represented through a binary relation $R(x, y)$: this means that there exists an edge from node $x$ to node $y$. We can compute the pairs of nodes that are connected by a directed path of length $k$ using the following CQ, which we call a *path query*: $P_k(x_1, x_{k+1}) = R(x_1, x_2) \wedge R(x_2, x_3) \wedge \cdots \wedge R(x_k, x_{k+1})$.

***Output Size Bounds.*** Let $\varphi(\mathbf{y}) = R_1(\mathbf{x}_1) \wedge R_2(\mathbf{x}_2) \wedge \ldots \wedge R_n(\mathbf{x}_n)$ be a CQ. A weight assignment $\mathbf{u} = (u_i)_{i=1,\ldots,n}$ is called a *fractional edge cover* of $S \subseteq \mathsf{vars}(\varphi)$ if $(i)$ for every atom $R_i$, $u_i \geq 0$ and $(ii)$ for every $x \in S$, $\sum_{i:x \in \mathsf{vars}(R_i)} u_i \geq 1$. The *fractional edge cover number* of $S$, denoted by $\rho^*(S)$ is the minimum of $\sum_{i=1}^{n} u_i$ over all fractional edge covers of $S$. Whenever $S = \mathsf{vars}(\varphi)$, we call this a fractional edge cover of $\varphi$ and simply use $\rho^*$. In a celebrated result, Atserias, Grohe and Marx [5] proved that for every fractional edge cover $\mathbf{u}$ of $\varphi$, the size of the output is bounded by the *AGM inequality*: $|\varphi(D)| \leq \prod_{i=1}^{n} |R_i|^{u_i}$. The above bound is constructive [22,23]: there exists an algorithm that computes the result $\varphi(D)$ in $O(\prod_i |R_i|^{u_i})$ time for every fractional edge cover $\mathbf{u}$.

***Tree Decompositions.*** Let $\varphi(\mathbf{y}) = R_1(\mathbf{x}_1) \wedge R_2(\mathbf{x}_2) \wedge \ldots \wedge R_n(\mathbf{x}_n)$ be a CQ. A *tree decomposition* of $\varphi$ is a tuple $(\mathcal{T}, (\mathcal{B}_t)_{t \in V(\mathcal{T})})$ where $\mathcal{T}$ is a tree, and every $\mathcal{B}_t$ is a subset of $\mathsf{vars}(\varphi)$, called the *bag* of $t$, such that

- For every atom $R_i$, the set $\mathsf{vars}(R_i)$ is contained in some bag; and
- For each variable $x \in \mathsf{vars}(\varphi)$, the set of nodes $\{t \mid x \in \mathcal{B}_t\}$ form a connected subtree of $\mathcal{T}$.

The *fractional hypertree width* of a decomposition is defined as $\max_{t \in V(\mathcal{T})} \rho^*(\mathcal{B}_t)$, where $\rho^*(\mathcal{B}_t)$ is the minimum fractional edge cover of the

[1] Here we extend the valuation to mean $v((a_1, \ldots, a_n)) = (v(a_1), \ldots, v(a_n))$.

vertices in $\mathcal{B}_t$. The fractional hypertree width of a query $\varphi$, denoted $\mathsf{fhw}(\varphi)$, is the minimum fractional hypertree width among all tree decompositions. We say that a query is *acyclic* if $\mathsf{fhw}(\varphi) = 1$.

***Computational Model.*** To measure the running time of our algorithms, we will use the uniform-cost RAM model [15], where data values and pointers to databases are of constant size. Throughout the paper, all complexity results are with respect to data complexity, where the query is assumed fixed.

## 3 Framework

### 3.1 Adorned Queries

In order to model different access patterns, we will use the concept of *adorned queries* introduced by [28]. Let $\varphi(x_1, \ldots, x_k)$ be the head of a CQ $\varphi$. In an adorned query, each variable in the head is associated with a binding type, which can be either *bound* ($\mathsf{b}$) or *free* ($\mathsf{f}$). We denote this as $\varphi^\eta$, where $\eta \in \{\mathsf{b}, \mathsf{f}\}^k$ is called the *access pattern*. The access pattern tells us for which variables the user must provide a value as input. Concretely, let $x_1, x_2, \ldots, x_\ell$ be the bound variables. An *access request* is sequence of constants $a_1, \ldots, a_\ell$, and it asks to return the result of the query $\varphi^\eta[a_1/x_1, \ldots, a_\ell/x_\ell]$ on the input database. We next demonstrate how to capture several data structure problems in this way.

*Example 2 (Set Disjointness and Set Intersection).* In the set disjointness problem, we are given $m$ sets $S_1, \ldots, S_m$ drawn from the same universe $U$. Let $N = \sum_{i=1}^{m} |S_i|$ be the total size of input sets. Each access request is a pair of indexes $(i, j), 1 \leq i, j, \leq m$, for which we need to decide whether $S_i \cap S_j$ is empty or not. To cast this problem as an adorned query, we encode the family of sets as a binary relation $R(x, y)$, such that element $x$ belongs to set $y$. Note that the relation will have size $N$. Then, the set disjointness problem corresponds to: $\varphi^{\mathsf{bb}}(y, z) = R(x, y) \wedge R(x, z)$. An access request in this case specifies two sets $y = S_i, z = S_j$, and issues the (Boolean) query $\varphi(S_i, S_j) = R(x, S_i) \wedge R(x, S_j)$. In the related set intersection problem, given a pair of indexes $(i, j)$ for $1 \leq i, j, \leq m$, we instead want to enumerate the elements in the intersection $S_i \cap S_j$, which can be captured by the following adorned query: $\varphi^{\mathsf{bbf}}(y, z, x) = R(x, y) \wedge R(x, z)$.

*Example 3 (k-Set Disjointness).* The $k$-set disjointness problem is a generalization of 2-set disjointness problem, where each request asks whether the intersection between $k$ sets is empty or not. Again, we can cast this problem into the following adorned query: $\varphi^{\mathsf{b}\ldots\mathsf{b}}(y_1, \ldots, y_k) = R(x, y_1) \wedge R(x, y_2) \wedge \cdots \wedge R(x, y_k)$

*Example 4 (k-Reachability).* Given a direct graph $G$ , the $k$-reachability problem asks, given a pair vertices $(u, v)$, to check whether they are connected by a path of length $k$. Representing the graph as a binary relation $R(x, y)$ (which means that there is an edge from $x$ to $y$), we can model this problem through the following adorned query: $\varphi^{\mathsf{bb}}(x_1, x_{k+1}) = R(x_1, x_2) \wedge R(x_2, x_3) \wedge \cdots \wedge R(x_k, x_{k+1})$ Observe that we can also check whether there is a path of length at most $k$ by combining the results of $k$ such queries (one for each length $1, \ldots, k$).

*Example 5 (Edge Triangles Detection).* Given a graph $G = (V, E)$, this problem asks, given an edge $(u, v)$ as the request, whether $(u, v)$ participates in a triangle or not. This task can be expressed as the following adorned query $\varphi^{\mathsf{bb}}_{\triangle}(x, z) = R(x, y) \wedge R(y, z) \wedge R(x, z)$ In the reporting version, the goal is to enumerate all triangles participated by edge $(x, z)$, which can also be expressed by the following adorned query $\varphi^{\mathsf{bbf}}_{\triangle}(x, z, y) = R(x, y) \wedge R(y, z) \wedge R(x, z)$.

We say that an adorned query is *Boolean* if every head variable is bound. In this case, the answer for every access request is also Boolean, i.e., true or false.

### 3.2 Problem Statement

Given an adorned query $\varphi^{\eta}$ and an input database $D$, our goal is to construct a data structure, such that we can answer any access request that conforms to the access pattern $\eta$ as fast as possible. In other words, an algorithm can be split into two phases:

- **Preprocessing phase:** we compute a data structure using space $S$.
- **Answering phase:** given an access request, we compute the answer using the data structure built in the preprocessing phase, within time $T$.

In this work, our goal is to study the relationship between the space of the data structure $S$ and the answering time $T$ for a given adorned query $\varphi^{\eta}$. We will focus on Boolean adorned queries, where the output is just true or false.

## 4 Space-Time Tradeoffs via Worst-Case Optimal Algorithms

Let $\varphi^{\eta}$ be an adorned query and $\mathcal{V}_{\mathsf{b}}$ denote its bound variables. For any fractional edge cover $\mathbf{u}$, we define the *slack* of $\mathbf{u}$ [11] as:

$$\alpha(\mathbf{u}) := \min_{x \in \mathsf{vars}(\varphi) \setminus \mathcal{V}_{\mathsf{b}}} \left( \sum_{i : x \in \mathsf{vars}(R_i)} u_i \right).$$

In other words, the slack is the maximum factor by which we can scale down the fractional cover $\mathbf{u}$ so that it remains a valid edge cover of the non-bound variables in the query[2]. Hence $\{u_i/\alpha(\mathbf{u})\}_i$ is a fractional edge cover of the nodes in $\mathsf{vars}(\varphi) \setminus \mathcal{V}_{\mathsf{b}}$. We always have $\alpha(\mathbf{u}) \geq 1$.

*Example 6.* Consider $\varphi^{\mathsf{b}\ldots\mathsf{b}}(y_1, \ldots, y_k) = R_1(x, y_1) \wedge R_2(x, y_2) \wedge \ldots R_k(x, y_k)$ with the optimal fractional edge cover $\mathbf{u}$, where $u_i = 1$ for $i \in \{1, \ldots, k\}$. The slack is $\alpha(\mathbf{u}) = k$, since the fractional edge cover $\hat{\mathbf{u}}$, where $\hat{u}_i = u_i/k = 1/k$ covers the only non-bound variable $x$.

[2] We will omit the parameter $\mathbf{u}$ from the notation of $\alpha$ whenever it is clear from the context.

**Theorem 1.** *Let $\varphi^{\eta}$ be a Boolean adorned query. Let* $\mathbf{u}$ *be any fractional edge cover of $\varphi$. Then, for any input database $D$, we can construct a data structure that answers any access request in time $O(T)$ and takes space*

$$S = O\left(|D| + \prod_{i=1}^{n} |R_i|^{u_i} / T^{\alpha}\right)$$

We should note that Theorem 1 applies even when the relation sizes are different; this gives us sharper upper bounds compared to the case where each relation is bounded by the total size of the input. Indeed, if using $|D|$ as an upper bound on each relation, we obtain a space requirement of $O(|D|^{\rho^*}/T^{\alpha})$ for achieving answering time $O(T)$, where $\rho^*$ is the fractional edge cover number. Since $\alpha \geq 1$, this gives us at worst a linear tradeoff between space and time, i.e., $S \cdot T = O(|D|^{\rho^*})$. For cases where $\alpha \geq 1$, we can obtain better tradeoffs. The full proofs for all results in this paper can be found in [10].

*Example 7.* Continuing the example in this section $\varphi^{\mathsf{b}\cdots\mathsf{b}}(y_1, \ldots, y_k) = R_1(x, y_1) \wedge R_2(x, y_2) \wedge \cdots \wedge R_k(x, y_k)$. We obtain an improved tradeoff: $S \cdot T^k = O(|D|^k)$[3]. Note that this result matches the best-known space-time tradeoff for the $k$-Set Disjointness problem [12]. (Note that all atoms use the same relation symbol $R$, so $|R_i| = |D|$ for every $i = 1, \ldots, k$. )

*Example 8 (Edge Triangles Detection).* For the Boolean version, it was shown in [12] that – conditioned on the strong set disjointness conjecture – any data structure that achieves answering time $T$ needs space $S = \Omega(|E|^2/T^2)$. A matching upper bound can be constructed by using a fractional edge cover $\mathbf{u} = (1, 1, 0)$ with slack $\alpha = 2$. Thus, Theorem 1 can be applied to achieve answering time $T$ using space $S = O(|E|^2/T^2)$. Careful inspection reveals that a different fractional edge cover $\mathbf{u} = (1/2, 1/2, 1/2)$ with slack $\alpha = 1$, achieves a better tradeoff. Thus, Theorem 1 can be applied to obtain the following corollary.

**Corollary 1.** *For a graph $G = (V, E)$, there exists a data structure of size $S = O(|E|^{3/2}/T)$ that can answer the edge triangles detection problem in $O(T)$.*

The data structure implied by Theorem 1 is always better when $T \leq \sqrt{|E|}$[4], thus refuting the conditional lower bound in [12]. We should note that this does not imply that the strong set disjointness conjecture is false, as we have observed an error in the reduction used by [12].

*Example 9 (Square Detection).* Beyond triangles, we consider the edge square detection problem, which checks whether a given edge belongs in a square pattern in a graph $G = (V, E)$, $\varphi^{\mathsf{bb}}_{\square}(x_1, x_2) = R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_4) \wedge R_4(x_4, x_1)$. Using the fractional edge cover $\mathbf{u} = (1/2, 1/2, 1/2, 1/2)$ with slack $\alpha = 1$, we obtain a tradeoff $S = O(|E|^2/T)$.

[3] For all results in this paper, $S$ includes the space requirement of the input as well. If we are interested in only the space requirement of the constructed data structure, then the $|D|$ term in the space requirement of Theorem 1 can be removed.

[4] All answering times $T > \sqrt{|E|}$ are trivial to achieve using linear space by using the data structure for $T' = \sqrt{E}$ and holding the result back until time $T$ has passed.

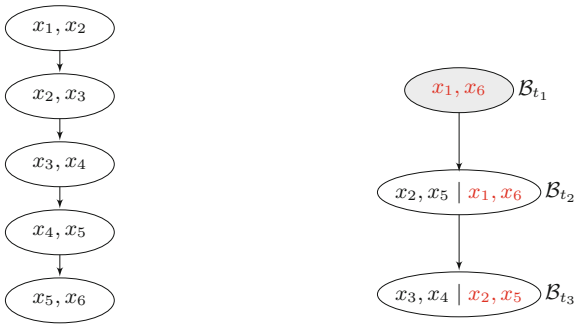



**Fig. 1.** Two tree decompositions for the length-5 path query: the left is unconstrained, while the right is a $C$-connex decomposition with $C = \{x_1, x_6\}$. The bound variables are colored red. The nodes in $A$ are colored grey.

## 5 Space-Time Tradeoffs via Tree Decompositions

Theorem 1 does not always give us the optimal tradeoff. For the $k$-reachability problem with the adorned query $\varphi^{\mathsf{bb}}(x_1, x_{k+1}) = R_1(x_1, x_2) \wedge \cdots \wedge R_k(x_k, x_{k+1})$, Theorem 1 gives a tradeoff $S \cdot T = |D|^{\lceil (k+1)/2 \rceil}$, by taking the optimal fractional edge covering number $\rho^* = \lceil (k+1)/2 \rceil$ and slack $\alpha = 1$, which is far from efficient. In this section, we will show how to leverage tree decompositions to further improve the space-time tradeoff in Theorem 1.

Again, let $\varphi^\eta$ be an adorned query. Given a set of nodes $C \subseteq \mathcal{V}$, a $C$-connex tree decomposition of $\varphi$ is a pair $(\mathcal{T}, A)$, where $(i)$ $\mathcal{T}$ is a tree decomposition of $\varphi$, and $(ii)$ $A$ is a connected subset of the tree nodes such that the union of their variables is exactly $C$. For our purposes, we choose $C = \mathcal{V}_\mathsf{b}$. Given a $\mathcal{V}_\mathsf{b}$-connex tree decomposition, we orient the tree from some node in $A$. We then define the bound variables for the bag $t$, $\mathcal{V}_\mathsf{b}^t$ as the variables in $\mathcal{B}_t$ that also appear in the bag of some ancestor of $t$. The free variables for the bag $t$ are the remaining variables in the bag, $\mathcal{V}_\mathsf{f}^t = \mathcal{B}_t \setminus \mathcal{V}_\mathsf{b}^t$.

*Example 10.* Consider the 5-path query $\varphi^{\mathsf{bb}}(x_1, x_6) = R_1(x_1, x_2) \wedge \cdots \wedge R_5(x_5, x_6)$. Here, $x_1$ and $x_6$ are the bound variables. Figure 1 shows the unconstrained decomposition as well as the $C$-connex decomposition for $\varphi^{\mathsf{bb}}(x_1, x_6)$, where $C = \{x_1, x_6\}$. The root bag contains the bound variables $x_1, x_6$. Bag $\mathcal{B}_{t_2}$ contains $x_1, x_6$ as bound variables and $x_2, x_5$ as the free variables. Bag $\mathcal{B}_{t_3}$ contains $x_2, x_5$ as bound variables for $\mathcal{B}_{t_3}$ and $x_3, x_4$ as free variables.

Next, we use a parameterized notion of width for the $\mathcal{V}_\mathsf{b}$-connex tree decomposition that was introduced in [11]. The width is parameterized by a function $\delta$ that maps each node $t$ in the tree to a non-negative number, such that $\delta(t) = 0$ whenever $t \in A$. The intuition here is that we will spend $O(|D|^{\delta(t)})$ in the node $t$ while answering the access request. The parameterized width of a bag $\mathcal{B}_t$ is now defined as: $\rho_t(\delta) = \min_{\mathbf{u}} (\sum_F u_F - \delta(t) \cdot \alpha)$ where $\mathbf{u}$ is a fractional edge cover of the bag $\mathcal{B}_t$, and $\alpha$ is the slack (on the bound variables of the bag). The

$\delta$-width of the decomposition is then defined as $\max_{t \notin A} \rho_t(\delta)$. Finally, we define the $\delta$-height as the maximum-weight path from the root to any leaf, where the weight of a path $P$ is $\sum_{t \in P} \delta(t)$. We now have all the necessary machinery to state our second main theorem.

**Theorem 2.** *Let $\varphi^\eta$ be a Boolean adorned query. Consider any $\mathcal{V}_\mathsf{b}$-connex tree decomposition of $\varphi$. For some parametrization $\delta$ of the decomposition, let $f$ be its $\delta$-width, and $h$ be its $\delta$-height. Then, for any input database $D$, we can construct a data structure that answers any access request in time $T = O(|D|^h)$ with space $S = O(|D| + |D|^f)$.*

The function $\delta$ allows us to trade off between time and space. If we set $\delta(t) = 0$ for every node $t$ in the tree, then the $\delta$-height becomes $O(1)$, while the $\delta$-width equals to the fractional hypetree width of the decomposition. As we increase the values of $\delta$ in each bag, the $\delta$-height increases while the $\delta$-width decreases, i.e., the answer time $T$ increases while the space decreases. Additionally, we note that the tradeoff from Theorem 2 is at least as good as the one from Theorem 1. Indeed, we can always construct a tree decomposition where all variables reside in a single node of the tree. In this case, we recover exactly the tradeoff from Theorem 1.

*Example 11.* We continue with the 5-path query. Since $\mathcal{B}_{t_1} = \{x_1, x_6\} \in A$, we assign $\delta(t_1) = 0$. For $\mathcal{B}_{t_2} = \{x_1, x_2, x_5, x_6\}$, the only valid fractional edge cover assigns weight 1 to both $R_1, R_5$ and has slack 1. Hence, if we assign $\delta(t_2) = \tau$ for some parameter $\tau$, the width is $2 - \tau$. For $\mathcal{B}_{t_3} = \{x_2, x_3, x_4, x_5\}$, the only fractional cover also assigns weight 1 to both $R_2, R_4$, with slack 1 again. Assigning $\delta(t_3) = \tau$, the width becomes $2 - \tau$ for $t_3$ as well. Hence, the $\delta$-width of the tree decomposition is $2 - \tau$, while the $\delta$-height is $2\tau$. Plugging this to Theorem 2, it gives us a tradeoff with answering time $T = O(|E|^{2\tau})$ and space usage $S = O(|E| + |E|^{2-\tau})$, which matches the state-of-the-art result in [12].

For the $k$-reachability problem, a general tradeoff $S \times T^{2/(k-1)} = O(|D|^2)$ was also shown by [12] using a careful recursive argument. The data structure generated using Theorem 2 is able to recover the tradeoff. In particular, we obtain the answering time as $T = O(|E|^{(k-1)\tau/2})$ using space $S = O(|E| + |E|^{2-\tau})$.

*Example 12.* Consider a variant of the square detection problem: given two vertices, the goal is to decide whether they occur in two opposites corners of a square, which can be captured by the following adorned query:

$$\varphi^{\mathsf{bb}}(x_1, x_3) = R_1(x_1, x_2) \wedge R_1(x_2, x_3) \wedge R_3(x_3, x_4) \wedge R_4(x_4, x_1).$$

Theorem 1 gives a tradeoff with answering time $O(T)$ and space $O(|E|^2/T)$. But we can obtain a better tradeoff using Theorem 2. Indeed, consider the tree decomposition where we have a root bag $t_1$ with $\mathcal{B}_{t_1} = \{x_1, x_3\}$, and two children of $t_1$ with Boolean $\mathcal{B}_{t_2} = \{x_1, x_2, x_3\}$ and $\mathcal{B}_{t_3} = \{x_1, x_3, x_4\}$. For $\mathcal{B}_{t_2}$, we can see that if assigning a weight of 1 to both hyperedges, we get a slack of 2. Hence, if $\delta(t_2) = \tau$, the $\delta$-width is $2-2\tau$. Similarly for $t_3$, we assign $\delta(t_3) = \tau$, for a $\delta$-width

with $2 - 2\tau$. Applying Theorem 2, we obtain a tradeoff with time $T = O(|E|^{\tau})$ (since both root-leaf paths have only one node), and space $S = O(|E| + |E|^{2-2\tau})$. So the space usage can be improved from $O(|E|^2/T)$ to $O(|E|^2/T^2)$.

## 6 CQs with Negation

In this section, we present a simple but powerful extension of our result to adorned Boolean CQs with negation. A CQ with negation, denoted as $CQ^{\neg}$, is a CQ where some of the atoms can be negative, i.e., $\neg R_i(\mathbf{x}_i)$ is allowed. For $\varphi \in CQ^{\neg}$, we denote by $\varphi^+$ the conjunction of the positive atoms in $\varphi$ and $\varphi^-$ the conjunction of all negated atoms. A $CQ^{\neg}$ is said to be *safe* if every variable appears in at least some positive atom. In this paper, we restrict our scope to the class of safe $CQ^{\neg}$, a standard assumption [21,29] ensuring that query results are well-defined and do not depend on domains.

Given a query $\varphi \in CQ^{\neg}$, we build the data structure from Theorem 2 for $\varphi^+$ but impose two constraints on the decomposition: $(i)$ no leaf node(s) contains any free variables, $(ii)$ for every negated atom $R^-$, all variables of $R^-$ must appear together as bound variables in some leaf node(s). In other words, there exists a leaf node such that $\mathsf{vars}(R^-)$ is present in it. It is easy to see that such a decomposition always exists. Indeed, we can fix the root bag to be $C = \mathcal{V}_{\mathsf{b}}$, its child bag with free variables as $\mathsf{vars}(\varphi^+) \setminus C$ and bound variables as $C$, and the leaf bag, which is connected to the child of the root, with bound variables as $\mathsf{vars}(\varphi^-)$ without free variables. Observe that the bag containing $\mathsf{vars}(\varphi^+)$ free variables can be covered by only using the positive atoms since $\varphi$ is safe. The intuition is the following: during the query answering phase, we wish to find the join result over all variables $\mathcal{V}_{\mathsf{f}}$ before reaching the leaf nodes; and then, we can check whether there the tuples satisfy the negated atoms or not, in $O(1)$ time. The next example shows the application of the algorithm to adorned path queries containing negation.

*Example 13.* Consider the query $Q^{\mathsf{bb}}(x_1, x_6) = R(x_1, x_2) \wedge \neg S(x_2, x_3) \wedge T(x_3, x_4) \wedge \neg U(x_4, x_5) \wedge V(x_5, x_6)$. Using the decomposition in Fig. 2, we can now apply Theroem 2 to obtain the tradeoff $S = O(|D|^3/\tau)$ and $T = O(\tau)$. Both leaf nodes only require linear space since a single atom covers the variables. Given an access request, we check whether the answer for this request has been materialized or not. If not, we proceed to the query answering phase and find at most $O(\tau)$ answers after evaluating the join in the middle bag. For each of these answers, we can now check in constant time whether the tuples formed by values for $x_2, x_3$ and $x_4, x_5$ are not present in relations $S$ and $U$ respectively.

For adorned queries where $\mathcal{V}_{\mathsf{b}} \subseteq \mathsf{vars}(\varphi^-)$, we can further simplify the algorithm. In this case, we no longer need to create a constrained decomposition since the check to see if the negated relations are satisfied or not can be done in constant time at the root bag itself. Thus, we can directly build the data structure from Theorem 2 using the query $\varphi^+$.

*Example 14 (Open Triangle Detection).* Consider the query $\varphi^{\mathsf{bb}}(x_2, x_3) = R_1(x_1, x_2) \wedge \neg R_2(x_2, x_3) \wedge R_3(x_1, x_3)$, where $\varphi^-$ is $\neg R_2(x_2, x_3)$ and $\varphi^+$ is $R_1(x_1, x_2) \wedge R_3(x_1, x_3)$ with the adorned view as $\varphi^{+\mathsf{bb}}(x_2, x_3) = R_1(x_1, x_2) \wedge R_3(x_1, x_3)$. Observe that $\{x_2, x_3\} \subseteq \mathsf{vars}(\varphi^-)$. We apply Theroem 2 to obtain the tradeoff $S = O(|E|^2/\tau^2)$ and $T = O(\tau)$ with root bag $C = \{x_2, x_3\}$, its child bag with $\mathcal{V}_\mathsf{b} = C$ and $\mathcal{V}_\mathsf{f} = \{x_1\}$, and the leaf bag to be $\mathcal{V}_\mathsf{b} = C$ and $\mathcal{V}_\mathsf{f} = \emptyset$. Given an access request $(a, b)$, we check whether the answer for this request has been materialized or not. If not, we traverse the decomposition and evaluating the join to find if there exists a connecting value for $x_1$. For the last bag, we simply check whether $(a, b)$ exists in $R_2$ or not in $O(1)$ time.

***A Note on Optimality.*** It is easy to see that the algorithm obtained for Boolean CQs with negation is conditionally optimal assuming the optimality of Theroem 2. Indeed, if all negated relations are empty, the join query is equivalent to $\varphi^+$ and the algorithm now simply applies Theroem 2 to $\varphi^+$. In Example 14, assuming relation $R_2$ is empty, the query is equivalent to set intersection whose tradeoffs are conjectured to be optimal.

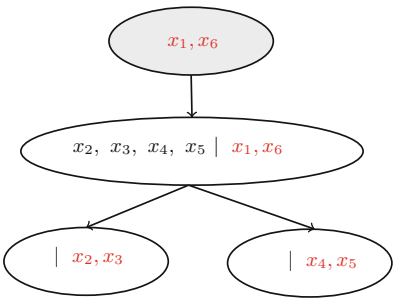



**Fig. 2.** $C$-connex decomposition for Example 13.

## 7 Path Queries

In this section, we present an algorithm for the adorned query $P_k^{\mathsf{bb}}(x_1, x_{k+1}) = R_1(x_1, x_2) \wedge \cdots \wedge R_k(x_k, x_{k+1})$ that improves upon the conjectured optimal solution. Before diving into the details, we first state the upper bound on the tradeoff between space and query time.

**Theorem 3 (due to [12]).** *There exists a data structure for solving $P_k^{\mathsf{bb}}(x_1, x_{k+1})$ with space $S$ and answering time $T$ such that $S \cdot T^{2/(k-1)} = O(|D|^2)$.*

Note that for $k = 2$, the problem is equivalent to SetDisjointness with the space/time tradeoff as $S \cdot T^2 = O(N^2)$. [12] also conjectured that the tradeoff is essentially optimal.

*Conjecture 1 (due to* [12]*).* Any data structure for $P_k^{\mathsf{bb}}(x_1, x_{k+1})$ with answering time $T$ must use space $S = \tilde{\Omega}(|D|^2/T^{2/(k-1)})$.

Building upon Conjecture 1, [12] also showed a result on the optimality of approximate distance oracles. Our result implies that Theroem 3 can be improved further, thus refuting Theorem 1. The first observation is that the tradeoff in Theroem 3 is only useful when $T \leq |D|$. Indeed, we can always answer any Boolean path query in linear time using breadth-first search. Surprisingly, it is also possible to improve Theroem 3 for the regime of small answering time as well. In what follows, we will show the improvement for paths of length 4; we will generalize the algorithm for any length later.

### 7.1 Length-4 Path

**Lemma 1.** *There exists a parameterized data structure for solving* $P_4^{\mathsf{bb}}(x_1, x_5)$ *that uses space* $S$ *and answering time* $T \leq \sqrt{|D|}$ *that satisfies the tradeoff* $S \cdot T = O(|D|^2)$.

For $k = 4$, Theorem 3 gives us the tradeoff $S \cdot T^{2/3} = O(|D|^2)$ which is always worse than the tradeoff in Lemma 1. We next present our algorithm in detail.

***Preprocessing Phase.*** Consider $P_4^{\mathsf{bb}}(x_1, x_5) = R(x_1, x_2) \wedge S(x_2, x_3) \wedge T(x_3, x_4) \wedge U(x_4, x_5)$. Let $\Delta$ be a degree threshold. We say that a constant $a$ is *heavy* if its frequency on attribute $x_3$ is greater than $\Delta$ in both relations $S$ and $T$; otherwise, it is *light*. In other words, $a$ is heavy if $|\sigma_{x_3=a}(S)| > \Delta$ and $|\sigma_{x_3=a}(T)| > \Delta$. We distinguish two cases based on whether a constant for $x_3$ is heavy or light. Let $\mathcal{L}_{\mathsf{heavy}}(x_3)$ denote the unary relation that contains all heavy values, and $\mathcal{L}_{\mathsf{light}}(x_3)$ the one that contains all light values. Observe that we can compute both of these relations in time $O(|D|)$ by simply iterating over the active domain of variable $x_3$ and checking the degree in relations $S$ and $T$. We compute two views:

$$V_1(x_1, x_3) = R(x_1, x_2) \wedge S(x_2, x_3) \wedge \mathcal{L}_{\mathsf{heavy}}(x_3)$$
$$V_2(x_3, x_5) = \mathcal{L}_{\mathsf{heavy}}(x_3) \wedge T(x_3, x_4) \wedge U(x_4, x_5)$$

We store the views as a hash index that, given a value of $x_1$ (or $x_5$), returns all matching values of $x_3$. Both views take space $O(|D|^2/\Delta)$. Indeed, $|\mathcal{L}_{\mathsf{heavy}}| \leq |D|/\Delta$. Since we can construct a fractional edge cover for $V_1$ by assigning a weight of 1 to $R$ and $\mathcal{L}_{\mathsf{heavy}}$, this gives us an upper bound of $|D| \cdot (|D|/\Delta)$ for the query output. The same argument holds for $V_2$. We also compute the following view for light values: $V_3(x_2, x_4) = S(x_2, x_3) \wedge \mathcal{L}_{\mathsf{light}}(x_3) \wedge T(x_3, x_4)$. This view requires space $O(|D| \cdot \Delta)$, since the degree of the light constants is at most $\Delta$ (i.e. $\sum_{x \in \mathcal{L}_{\mathsf{light}}(x_3)} |S(x_2, x) \wedge T(x, x_4)| \leq \sum_{x \in \mathcal{L}_{\mathsf{light}}(x_3)} |S(x_2, x)| \cdot |T(x, x_4)| \leq \sum_{x \in \mathcal{L}_{\mathsf{light}}(x_3)} |S(x_2, x)| \cdot \Delta \leq |D| \cdot \Delta$). We can now rewrite the original query as $P_4^{\mathsf{bb}}(x_1, x_5) = R(x_1, x_2) \wedge V_3(x_2, x_4) \wedge U(x_4, x_5)$.

The rewritten query is a three path query. Hence, we can apply Theorem 1 to create a data structure with answering time $T = O(|D|/\Delta)$ and space $S = O(|D|^2/(|D|/\Delta)) = O(|D| \cdot \Delta)$.

***Query Answering.*** Given an access request, we first check whether there exists a 4-path that goes through some heavy value in $\mathcal{L}_{\mathsf{heavy}}(x_3)$. This can be done in time $O(|D|/\Delta)$ using the views $V_1$ and $V_2$. Indeed, we obtain at most $O(|D|/\Delta)$ values for $x_3$ using the index for $V_1$, and $O(|D|/\Delta)$ values for $x_3$ using the index for $V_3$. We then intersect the results in time $O(|D|/\Delta)$ by iterating over the $O(|D|/\Delta)$ values for $x_3$ and checking if the bound values for $x_1$ and $x_5$ from a tuple in $V_1$ and $V_2$ respectively. If we find no such 4-path, we check for a 4-path that uses a light value for $x_3$. From the data structure we have constructed in the preprocessing phase, we can do this in time $O(|D|/\Delta)$.

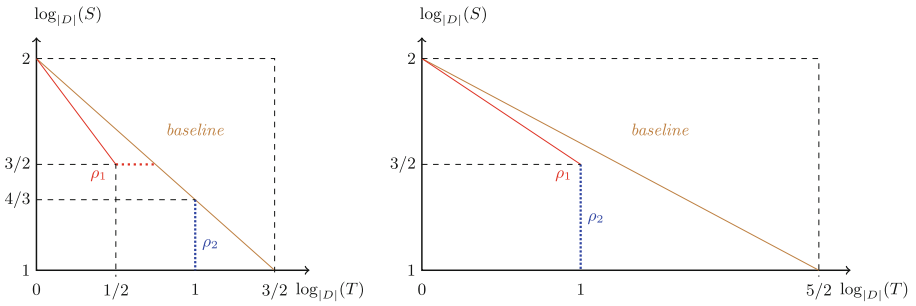
***Tradeoff Analysis.*** From the above, we can compute the answer in time $T = O(|D|/\Delta)$. From the analysis in the preprocessing phase, the space needed is $S = O(|D|^2/\Delta + |D| \cdot \Delta)$. Thus, whenever $\Delta \geq \sqrt{|D|}$, the space becomes $S = O(|D| \cdot \Delta)$, completing our analysis.

### 7.2 General Path Queries

We can now use the algorithm for the 4-path query to improve the space-time tradeoff for general path queries of length greater than four.

**Theorem 4.** *Let $D$ be an input instance. For $k \geq 4$, there is a data structure for $P_k^{\mathsf{bb}}(x_1, x_{k+1})$ with space $S = O(|D| \cdot \Delta)$ and answer time $T = O\left((\frac{|D|}{\Delta})^{\frac{k-2}{2}}\right)$ for $\Delta \geq \sqrt{|D|}$.*

The space-time tradeoff obtained from Theroem 4 is $S \cdot T^{2/(k-2)} = O(|D|^2)$, but only for $T \leq |D|^{(k-2)/4}$. To compare it with the tradeoff of $S \cdot T^{2/(k-1)} = O(|D|^2)$ obtained from Theroem 3, it is instructive to look at Figures Fig. 3a and 3b, which plot the space-time tradeoffs for $k = 4$ and $k = 6$ respectively. In general, as $k$ grows, the new tradeoff line (labeled as $\rho_1$) becomes flatter and approaches Theorem 3.




(a) 4-reachability CQAP. (b) 6-reachability CQAP.

**Fig. 3.** Space/time tradeoffs for path query of length $k \in \{4, 6\}$. The line in brown (baseline) shows the tradeoff obtained from Theroem 3. The red curve ($\rho_1$) is the new tradeoff obtained using Theorem 4 and $\rho_2$ shows the transition to when BFS takes over as the best algorithm.

## 8 Lower Bounds

In this section, we study the lower bounds for adorned star and path queries. We first present conditional lower bounds for the $k$-Set Disjointness problem using the conditional optimality of $\ell$-Set Disjointness where $\ell < k$. First, we review the known results from [12] starting with the conjecture for $k$-Set Disjointness .

*Conjecture 2. (due to* [12]*).* Any data structure for $k$-Set Disjointness problem that answers queries in time $T$ must use space $S = \Omega(|D|^k/T^k)$.

Conjecture 2 was shown to be conditionally optimal based on conjectured lower bound for the $(k+1)$-Sum Indexing problem, however, it was subsequently showed to be false [19], which implies that Conjecture 2 is still an open problem. 2 can be further generalized to the case when input relations are of unequal sizes as follows.

*Conjecture 3.* Any data structure for $\varphi_*^{\mathsf{b}\cdots\mathsf{b}}(y_1, \ldots, y_k) = R_1(x, y_1) \wedge \cdots \wedge R_k(x, y_k)$ that answers queries in time $T$ must use space $S = \Omega(\Pi_{i=1}^k |R_i|/T^k)$.

We now state the main result for star queries.

**Theorem 5.** *Suppose that any data structure for $\varphi_*^{\mathsf{b}\cdots\mathsf{b}}(y_1, \ldots, y_k)$ with answering time $T$ must use space $S = \Omega(\Pi_{i=1}^k |R_i|/T^k)$. Then, any data structure for $Q_*^{\mathsf{b}\cdots\mathsf{b}}(y_1, \ldots, y_\ell)$ with answering time $T$ must use space $S = \Omega(\Pi_{i=1}^\ell |R_i|/T^\ell)$, for $2 \leq \ell < k$.*

Theorem 5 creates a hierarchy for $k$-Set Disjointness , where the optimality of smaller set disjointness instances depends on larger set disjointness instances. Next, we show conditional lower bounds on the space requirement of path queries. We begin by proving a simple result for optimality of $P_2^{\mathsf{bb}}$ (equivalent to 2-Set Disjointness) assuming the optimality of $P_3^{\mathsf{bb}}$ query.

**Theorem 6.** *Suppose that any data structure for $P_3^{\mathsf{bb}}$ that answers queries in time $T$, uses space $S$ such that $S \cdot T = \Omega(|D|^2)$. Then, for $P_2^{\mathsf{bb}}$ , for any data structure that uses space $S = O(|D|^2/T^2)$, the answering time is $\Omega(T)$.*

Using a similar argument, it can be shown that the conditional optimality of Theroem 4 for $k = 4$ implies that $S \cdot T = \Omega(|D|^2)$ tradeoff for $P_3^{\mathsf{bb}}$ is also optimal (but only for the range $T \leq \sqrt{|D|}$ when the result is applicable).

## 9 Related Work

The study of fine-grained space/time tradeoffs for query answering is a relatively recent effort in the algorithmic community. The study of distance oracles over graphs was first initiated by [25] where lower bounds are shown on the size of a distance oracle for sparse graphs based on a conjecture about the best possible data structure for a set intersection problem. [9] also considered the problem of set intersection and presented a data structure that can answer boolean set

intersection queries which is conditionally optimal [12]. There also exist another line of work that looks at the problem of approximate distance oracles. Agarwal et al. [3,4] showed that for stretch-2 and stretch-3 oracles, we can achieve $S \times T = O(|D|^2)$ and $S \times T^2 = O(|D|^2)$. They also showed that for any integer $k$, a stretch-$(1 + 1/k)$ oracle exhibits $S \times T^{1/k} = O(|D|^2)$ tradeoff. Unfortunately, no lower bounds are known for non-constant query time. The authors in [12] conjectured that the tradeoff $S \times T^{2/(k-1)} = O(|D|^2)$ for $k$-reachability is optimal which would also imply that stretch-$(1+1/k)$ oracle tradeoff is also optimal. A different line of work has considered the problem of enumerating query results [27] of a non-boolean query. [9] presented a data structure to enumerate the intersection of two sets with guarantees on the total answering time. This result was generalized to incorporate *full* adorned views over CQs [11]. Our work extends the results to the setting where the join variables are projected away from the query result (i.e. the adorned views are *non-full*) and makes the connection between several different algorithmic problems that have been studied independently. Further, we also consider boolean CQs that may contain negations. In the non-static setting, [6] initiated the study of answering conjunctive query results under updates. More recently, [16] presented an algorithm for counting the number of triangles under updates. There have also been some exciting developments in the space of enumerating query results with delay for a proper subset of CQs known as *hierarchical queries*. [17] presented a tradeoff between preprocessing time and delay for enumerating the results of any (not necessarily full) hierarchical queries under static and dynamic settings. It remains an interesting problem to find improved algorithms for more restricted set of CQs such as hierarchical queries.

## 10 Conclusion

In this paper, we investigated the tradeoffs between answering time and space required by the data structure to answer boolean queries. Our main contribution is a unified algorithm that recovers the best known results for several boolean queries of practical interests. We then apply our main result to improve upon the state-of-the-art algorithms to answer boolean queries over the four path query which is subsequently used to improve the tradeoff for all path queries of length greater than four and show conditional lower bounds. There are several questions that remain open. We describe the problems that are particularly engaging.

**Unconditional Lower Bounds.** It remains an open problem to prove unconditional lower bounds on the space requirement for answering boolean star and path queries in the RAM model. For instance, 2-Set Disjointness can be answered in constant time by materializing all answers using $\Theta(|D|^2)$ space but there is no lower bound to rule out if this can be achieved using sub-quadratic space.

**Improved Approximate Distance Oracles.** It would be interesting to investigate whether our ideas can be applied to existing algorithms for constructing distance oracles to improve their space requirement. [12] conjectured that the $k$-reachability tradeoff is optimal and used it to prove the conditional optimality

of distance oracles. We believe our framework can be used to improve upon the bounds for $k$-reachability in conjunction with other techniques used to prove bounds for join query processing in the database theory community.

## References


1. Abo Khamis, M., Kolaitis, P.G., Ngo, H.Q., Suciu, D.: Decision problems in information theory. In: ICALP (2020)
2. Afshani, P., Nielsen, J.A.S.: Data structure lower bounds for document indexing problems. In: ICALP (2016)
3. Agarwal, R.: The space-stretch-time tradeoff in distance oracles. In: Schulz, A.S., Wagner, D. (eds.) ESA 2014. LNCS, vol. 8737, pp. 49–60. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44777-2_5
4. Agarwal, R., Godfrey, P.B., Har-Peled, S.: Approximate distance queries and compact routing in sparse graphs. In: INFOCOM, pp. 1754–1762. IEEE (2011)
5. Atserias, A., Grohe, M., Marx, D.: Size bounds and query plans for relational joins. SIAM J. Comput. **42**(4), 1737–1767 (2013)
6. Berkholz, C., Keppeler, J., Schweikardt, N.: Answering conjunctive queries under updates. In: PODS, pp. 303–318. ACM (2017)
7. Chan, T.M., Lewenstein, M.: Clustered integer 3SUM via additive combinatorics. In: STOC, pp. 31–40 (2015)
8. Cohen, H., Porat, E.: Fast set intersection and two-patterns matching. Theoret. Comput. Sci. **411**(40–42), 3795–3800 (2010)
9. Cohen, H., Porat, E.: On the hardness of distance oracle for sparse graph. arXiv preprint arXiv:1006.1117 (2010)
10. Deep, S., Hu, X., Koutris, P.: General space-time tradeoffs via relational queries. arXiv preprint arXiv:2109.10889 (2021)
11. Deep, S., Koutris, P.: Compressed representations of conjunctive query results. In: PODS, pp. 307–322. ACM (2018)
12. Goldstein, I., Kopelowitz, T., Lewenstein, M., Porat, E.: Conditional lower bounds for space/time tradeoffs. In: WADS 2017. LNCS, vol. 10389, pp. 421–436. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_36
13. Gottlob, G., Greco, G., Scarcello, F.: Treewidth and hypertree width. In: Tractability: Practical Approaches to Hard Problems, vol. 1 (2014)
14. Greco, G., Scarcello, F.: Structural tractability of enumerating CSP solutions. Constraints **18**(1), 38–74 (2013)
15. Hopcroft, J.E., Ullman, J.D., Aho, A.: The Design and Analysis of Computer Algorithms (1975)
16. Kara, A., Ngo, H.Q., Nikolic, M., Olteanu, D., Zhang, H.: Counting triangles under updates in worst-case optimal time. In: ICDT (2019)
17. Kara, A., Nikolic, M., Olteanu, D., Zhang, H.: Trade-offs in static and dynamic evaluation of hierarchical queries. In: PODS, pp. 375–392 (2020)
18. Kociumaka, T., Radoszewski, J., Rytter, W.: Efficient indexes for jumbled pattern matching with constant-sized alphabet. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 625–636. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40450-4_53
19. Kopelowitz, T., Porat, E.: The strong 3SUM-indexing conjecture is false. arXiv preprint arXiv:1907.11206 (2019)

20. Larsen, K.G., Munro, J.I., Nielsen, J.S., Thankachan, S.V.: On hardness of several string indexing problems. Theoret. Comput. Sci. **582**, 74–82 (2015)
21. Nash, A., Ludäscher, B.: Processing unions of conjunctive queries with negation under limited access patterns. In: Bertino, E., et al. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 422–440. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24741-8_25
22. Ngo, H.Q., Porat, E., Ré, C., Rudra, A.: Worst-case optimal join algorithms. In: PODS, pp. 37–48. ACM (2012)
23. Ngo, H.Q., Ré, C., Rudra, A.: Skew strikes back: new developments in the theory of join algorithms. SIGMOD Rec. **42**(4), 5–16 (2013)
24. Olteanu, D., Schleich, M.: Factorized databases. ACM SIGMOD Rec. **45**(2), 5–16 (2016)
25. Patrascu, M., Roditty, L.: Distance oracles beyond the Thorup-zwick bound. In: FOCS, pp. 815–823. IEEE (2010)
26. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**(3), 309–322 (1986)
27. Segoufin, L.: Enumerating with constant delay the answers to a query. In: Proceedings of the 16th ICDT, pp. 10–20. ACM (2013)
28. Ullman, J.D.: An approach to processing queries in a logic-based query language. In: Brodie, M.L., Mylopoulos, J. (eds.) On knowledge base management systems. Topics in Information Systems, pp. 147–164. Springer, New York (1986). https://doi.org/10.1007/978-1-4612-4980-1_16
29. Wei, F., Lausen, G.: Containment of conjunctive queries with safe negation. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 346–360. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36285-1_23
30. Xirogiannopoulos, K., Deshpande, A.: Extracting and analyzing hidden graphs from relational databases. In: SIGMOD, pp. 897–912. ACM (2017)
31. Xirogiannopoulos, K., Khurana, U., Deshpande, A.: GraphGen: exploring interesting graphs in relational data. Proc. VLDB Endowment **8**(12), 2032–2035 (2015)
32. Yannakakis, M.: Algorithms for acyclic database schemes. In: VLDB, pp. 82–94 (1981)