Space-Time Tradeoffs for Conjunctive Queries with Access Patterns

Hangdong Zhao
University of Wisconsin-Madison
Madison, WI, USA
hangdong@cs.wisc.edu

Shaleen Deep
Microsoft Gray Systems Lab
Madison, WI, USA
shaleen.deep@microsoft.com

Paraschos Koutris
University of Wisconsin-Madison
Madison, WI, USA
paris@cs.wisc.edu

ABSTRACT

In this paper, we investigate space-time tradeoffs for answering conjunctive queries with access patterns (CQAPs). The goal is to create a space-efficient data structure in an initial preprocessing phase and use it for answering (multiple) queries in an online phase. Previous work has developed data structures that trades off space usage for answering time for queries of practical interest, such as the path and triangle query. However, these approaches lack a comprehensive framework and are not generalizable. Our main contribution is a general algorithmic framework for obtaining space-time tradeoffs for any CQAP. Our framework builds upon the PANDA algorithm and tree decomposition techniques. We demonstrate that our framework captures all state-of-the-art tradeoffs that were independently produced for various queries. Further, we show surprising improvements over the state-of-the-art tradeoffs known in the existing literature for reachability queries.

CCS CONCEPTS

• **Theory of computation** → **Database query processing and optimization (theory)**; • **Information systems** → **Query planning**; **Query optimization**; **Join algorithms**.

KEYWORDS

Tradeoffs, Access Patterns, Submodular Width, Tree Decompositions, Disjunctive Datalog, Shannon-type Inequalities

ACM Reference Format:
Hangdong Zhao, Shaleen Deep, and Paraschos Koutris. 2023. Space-Time Tradeoffs for Conjunctive Queries with Access Patterns. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3584372.3588675

1 INTRODUCTION

We study a class of problems that splits an algorithmic task into two phases: the *preprocessing phase*, which computes a space-efficient data structure from the input, and the *online phase*, which uses the data structure to answer requests of a specific form over the input as fast as possible. Many important algorithmic tasks such as set intersection problems [8, 15], reachability in directed graphs [2, 3, 9], histogram indexing [7, 25], and problems related to document retrieval [1, 26] can be expressed in this way. The fundamental algorithmic question related to these problems is to find *the tradeoff between the space S necessary for storing the data structures and the time T for answering a request.*

Let us look at one of the simplest tasks in this setup. Consider the 2-Set Disjointness problem: given a universe of elements U and a collection of m sets $S_1, \ldots, S_m \subseteq U$, 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 $S_i \cap S_j$ is empty or not. It is well-known that the space-time tradeoff for 2-Set Disjointness is captured by the equation $S \cdot T^2 = O(N^2)$, where N is the total size of all sets [8, 15]. Similar tradeoffs have also been established for other data structure problems. In the k-Reachability problem [8, 15] we are given as 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. The data structure obtained was conjectured to be optimal by [15], and the conjectured optimality was used to develop conditional lower bounds for other problems, such as approximate distance oracles [2, 3] where no progress has been made in improving the upper bounds in the last decade. In the edge triangle detection problem [15], we are given as input a graph $G = (V, E)$, and the goal is to develop a data structure that can answer whether a given edge $e \in E$ participates in a triangle or not. Each of these problems has been studied in isolation and therefore, the algorithmic insights are not readily generalizable into a comprehensive framework.

In this paper, we cast many of the above problems into answering *Conjunctive Queries with Access Patterns (CQAPs)* over a relational database. 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 CQAP: $\varphi(\mid y_1, y_2) \leftarrow R(x, y_1) \wedge R(x, y_2)$. The expression $\varphi(\mid y_1, y_2)$ can be interpreted as follows: given values for y_1, y_2, compute whether the query returns true or not. Different access patterns capture different ways of accessing the result of the CQ and result in different tradeoffs.

Tradeoffs for enumerating Conjunctive Query results under static and dynamic settings have been a subject of previous research [13, 17, 19–21, 30]. However, previous work either focuses on the tradeoff between preprocessing time and answering time [19–21], or the tradeoff between space and delay in enumeration [13, 30]. In this paper, we focus explicitly on the tradeoff between space and answering time, without optimizing for preprocessing time. Most closely related to our setting is the problem of answering Boolean CQs [12]. In that work, the authors slightly improve upon the data structure proposed in [13] and adapt it for Boolean CQ answering. Further, [12] identified that the conjectured tradeoff for the

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06…$15.00
https://doi.org/10.1145/3584372.3588675

k-reachability problem is suboptimal by showing slightly improved tradeoffs for all $k \geq 3$. The techniques used in this paper are quite different and a vast generalization of the techniques used in [12]. The proposed improvements in [12] for k-reachability are already captured in this work and in many cases, surpass the ones from [12].

Our Contribution. Our key contribution is a general algorithmic framework for obtaining space-time tradeoffs for any CQAP. Our framework builds upon the PANDA algorithm [24] and tree decomposition techniques [16, 28]. Given any CQAP, it calculates a tradeoff that can find the best possible time for a given space budget. To achieve this goal, we need two key technical contributions.

First, we introduce the novel notion of *partially-materialized tree decompositions* (PMTDs) that allow us to capture different possible materialization strategies on a given tree decomposition (Section 3). At a high level, a PMTD augments a tree decomposition with information on which bags should be materialized and which should be computed online. To use a PMTD, we propose a variant of the Yannakakis algorithm (Subsection 3.1) such that during the online phase we incur only the cost of visiting the non-materialized bags.

The second key ingredient is an extension of the PANDA algorithm [24] that computes a disjunctive rule in two phases. The computation of a disjunctive rule allows placing an answer to any of the targets in the head of the rule. A key technical component in the PANDA algorithm is the notion of a *Shannon-flow inequality*. For any Shannon-flow inequality, one can construct a proof sequence that has a direct correspondence with relational operators. Consequently, a proof sequence can be transformed into a join algorithm. The disjunctive rules we consider are computed in two phases: in the first phase (preprocessing), we can place an answer only to targets that will be materialized during the preprocessing phase. In the second phase (online), we place an answer to the remaining targets. We call these rules *2-phase disjunctive rules* (Subsection 4.1). To achieve this 2-phase computation, we introduce a type of Shannon-flow inequalities, called *joint Shannon-flow inequalities* (Section 5), such that each inequality gives rise to a space-time tradeoff. The joint Shannon-flow inequality generates two parallel proof sequences, one proof sequence for the preprocessing phase and another proof sequence for the answering phase. This transformation allows us to use the PANDA algorithm as a blackbox on each of the proof sequences independently and is instrumental in achieving space-time tradeoffs.

We demonstrate the versatility of our framework by recovering state-of-the-art space-time tradeoffs for Boolean CQAPs, 2-Set Disjointness as well as its generalization k-Set Disjointness , and k-Reachability (Section 6). We also apply our framework to the previously unstudied setting of space-time tradeoffs (in the static setting) for access patterns over a subset of *hierarchical queries*, a fragment of acyclic CQs that is of great interest [5, 6, 10, 11, 18, 20]. Interestingly, we can recover strategies that are very similar to how specialized enumeration algorithms with provable guarantees work for this class of CQs [11, 20]. More importantly, we improve state-of-the-art tradeoffs. Our most interesting finding is that we can obtain complex tradeoffs for k-Reachability that exhibit different behavior for different regimes of S. For the 3-Reachability problem, we show how to improve the tradeoff for a significant part of the spectrum. For the 4-Reachability problem, we are able to show (via a rather involved analysis) that the space-time tradeoff can be improved *everywhere* when compared to the conjectured optimal! These results falsify the proposed optimal tradeoff of $S \cdot T^{2/(k-1)} = \widetilde{O}(|E|^2)$ for k-Reachability for regimes that are even larger than what was shown in [12].

Organization. We introduce the basic terminology and problem definition in Section 2. In Section 3, we describe the augmented tree decompositions that are necessary for our framework. Section 4 introduces the general framework while Section 5 presents the algorithms used in our framework. We present the applications of the framework in Section 6. The related work is described in Section 7 and we conclude with a list of open problems in Section 8.

2 BACKGROUND

Conjunctive Query. We associate a Conjunctive Query (CQ) φ with a hypergraph $\mathcal{H} = ([n], \mathcal{E})$, where $[n] = \{1, \ldots, n\}$ and $\mathcal{E} \subseteq 2^{[n]}$. The body of the query has atoms R_F, where $F \in \mathcal{E}$. To each node $i \in [n]$, we associate a variable x_i. The CQ is then

$$\varphi(\mathbf{x}_H) \leftarrow \bigwedge_{F \in \mathcal{E}} R_F(\mathbf{x}_F),$$

where $\mathbf{x}_I$ denotes the tuple $(x_i)_{i \in I}$ for any $I \subseteq [n]$. The variables in $\mathbf{x}_H$ are called the *head variables* of the CQ. The CQ is *full* if $H = [n]$ and *Boolean* if $H = \emptyset$. We use φ to denote the output of the CQ φ.

Degree Constraints. A *degree constraint* is a triple $(X, Y, N_{Y|X})$ where $X \subset Y \subseteq [n]$ and $N_{Y|X}$ is a natural number. A relation R_F is said to *guard* the degree constraint $(X, Y, N_{Y|X})$ if $X \subset Y \subseteq F$ and for every tuple $\mathbf{t}_X$ (over X), $\max_{\mathbf{t}_X} \deg_F(Y|\mathbf{t}_X) \leq N_{Y|X}$, where $\deg_F(Y|\mathbf{t}_X) = \left|\Pi_Y(\sigma_{X=\mathbf{t}_X} R_F)\right|$. We use DC to denote a set of degree constraints and say that DC is guarded by a database instance $\mathcal{D}$ if every $(X, Y, N_{Y|X}) \in$ DC is guarded by some relation in $\mathcal{D}$. A degree constraint $(X, Y, N_{Y|X})$ is a *cardinality constraint* if $X = \emptyset$. Throughout this work, we make the following assumptions on DC guarded by a database instance $\mathcal{D}$:

- *(best constraints assumption)* w.l.o.g, for any $X \subset Y \subseteq [n]$, there is at most one $(X, Y, N_{Y|X}) \in$ DC. This assumption can be maintained by only keeping the minimum $N_{Y|X}$ if there is more than one.
- for every relation $R_F \in \mathcal{D}$, there is a *cardinality constraint* $(\emptyset, F, |R_F| \stackrel{\text{def}}{=} N_{F|\emptyset}) \in$ DC. The *size* of the database $\mathcal{D}$ is denoted as $|\mathcal{D}| \stackrel{\text{def}}{=} \max_{R_F \in \mathcal{D}} |R_F|$.

In this work, we use degree constraints to measure data complexity. All logs are in base 2, unless otherwise stated.

2.1 CQs with Access Patterns

We define CQs with access patterns following the definition from [21]:

Definition 2.1 (CQ with access patterns). A Conjunctive Query with *Access Patterns* (CQAP) is an expression of the form

$$\varphi(\mathbf{x}_H \mid \mathbf{x}_A) \leftarrow \bigwedge_{F \in \mathcal{E}} R_F(\mathbf{x}_F),$$

where $A \subseteq [n]$ is called the *access pattern* of the query.

The access pattern tells us how a user accesses the result of the CQ. In particular, the user will provide an instance of a relation

$Q_A(\mathbf{x}_A)$, which we call an *access request*. The task is then to return the result of the following CQ, denoted as φ, where

$$\varphi(\mathbf{x}_H) \leftarrow Q_A(\mathbf{x}_A) \wedge \bigwedge_{F \in \mathcal{E}} R_F(\mathbf{x}_F).$$

We call φ the *access CQ*. The most natural access request is one where $|Q_A| = 1$; in other words, the user provides only one fixed value for every variable $x_i, i \in A$. This can be thought of as using the CQ result as an index with search key $\mathbf{x}_A$. By allowing the access request Q_A to consist of more tuples, we can capture other scenarios. For example, one can take a stream of access requests of size 1 and batch them together to obtain a (possibly faster) answer for all of them at once. Prior work [13, 21] has only considered the case where $|Q_A| = 1$.

2.2 Problem Statement

Let $\varphi(\mathbf{x}_H \mid \mathbf{x}_A)$ be a CQAP under degree constraints DC guarded by the input relations. In addition, we denote by AC another set of degree constraints known in prior, guarded by any access request Q_A. Similar to DC, we that assume there is a cardinality constraint $(\emptyset, A, |Q_A| = N_{A|\emptyset}) \in \text{AC}$ guarded by Q_A. For example, the case where $|Q_A| = 1$ can be interpreted as a cardinality constraint $(\emptyset, A, 1) \in \text{AC}$. Given a database instance $\mathcal{D}$ guarding DC, our goal is to construct a data structure, such that we can answer any access request as fast as possible. More formally, we split query processing into two phases:

Preprocessing phase: it constructs a data structure in space $\widetilde{O}(S)$[1]. The overall space cost takes the form $\widetilde{O}(S + |\mathcal{D}|)$, where S is called the *intrinsic space cost* of the data structure and $|\mathcal{D}|$ is the (unavoidable) space cost for storing the database.

Online phase: given an access request Q_A (guarding AC), it returns the results of the access CQ φ using the data structure built in the preprocessing phase. The (worst-case) answering time is then $\widetilde{O}(T+|Q_A|)+O(|\varphi|)$, where T is called the *intrinsic time cost* and $|Q_A| + |\varphi|$ is the (unavoidable) time cost of reading the access request Q_A and enumerating the output. For the Boolean case and when $|Q_A| = 1$, the answering time simply becomes $\widetilde{O}(T)$.

In this work, we study the tradeoffs between the two intrinsic quantities, S and T, which we will call an *intrinsic tradeoff*. At one extreme, the algorithm stores nothing, thus $S = O(1)$, and we answer each access request from scratch. At the other extreme, the algorithm stores the results of the CQ $\varphi_M(\mathbf{x}_{H\cup A}) \leftarrow \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F)$ as a hash table with index key $\mathbf{x}_A$. For any access request Q_A, we simply evaluate the query $\varphi(\mathbf{x}_H) \leftarrow Q_A \wedge \varphi_M$ in the online phase by probing each tuple of Q_A in the hash table. If $H \supseteq A$, then any access request can be answered in (instance-optimal) time $O(|Q_A| + |\varphi|)$, in which case $T = O(1)$.

Example 2.2 (k-Set Disjointness). In this problem, we are given sets $S_1, \ldots, S_m$ with elements drawn from the same universe U. Each access request asks whether the intersection between k sets is empty or not. By encoding the family of sets as a binary relation $R(y, x)$ such that element y belongs to set x, we can express the problem as the following CQAP:

$$\varphi(\mid \mathbf{x}_{[k]}) \leftarrow \bigwedge_{i\in[k]} R(y, x_i). \tag{1}$$

If we also want to enumerate the elements in their intersection, we would instead use the non-Boolean version:

$$\varphi(y \mid \mathbf{x}_{[k]}) \leftarrow \bigwedge_{i\in[k]} R(y, x_i). \tag{2}$$

Example 2.3 (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 $R(x, y)$, we can model this problem through the following CQAP (the k-path query):

$$\phi_k(\mid x_1, x_{k+1}) \leftarrow \bigwedge_{i\in[k]} R(x_i, x_{i+1}).$$

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 $1, \ldots, k$).

In this work, we focus on the CQAP such that $H \supseteq A$. If we are given a CQAP where $H \not\supseteq A$, we replace the head of the CQAP with $\varphi(\mathbf{x}_{H\cup A} \mid \mathbf{x}_A)$, and simply project on the desired results in the end.

3 PARTIALLY MATERIALIZED TREE DECOMPOSITIONS

In this section, we introduce a type of tree decomposition that augments a decomposition with information about what bags we want to materialize.

Definition 3.1 (Tree Decomposition). A *tree decomposition* of a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ is a pair $(\mathcal{T}, \chi)$ where (i) $\mathcal{T}$ is an undirected tree, and (ii) $\chi : V(\mathcal{T}) \to 2^{[n]}$ is a mapping that assigns to every node $t \in V(\mathcal{T})$ a subset of $[n]$, called the *bag* of t, such that

(1) For every hyperedge $F \in \mathcal{E}$, the set F is contained in some bag; and
(2) For each vertex $x \in [n]$, the set of nodes $\{t \mid x \in \chi(t)\}$ forms a (non-empty) connected subtree of $\mathcal{T}$.

Take a tree decomposition $(\mathcal{T}, \chi)$ and a node $r \in V(\mathcal{T})$. We define $\text{TOP}_r(x)$ as the highest node in $\mathcal{T}$ containing x in its bag if we root the tree at r. We now say that $(\mathcal{T}, \chi)$ is *free-connex w.r.t.* r if for any $x \in H$ and $y \in [n] \setminus H$, $\text{TOP}_r(y)$ is not an ancestor of $\text{TOP}_r(x)$ [34]. We say that $(\mathcal{T}, \chi)$ is free-connex if it is free-connex w.r.t. some $r \in V(\mathcal{T})$.

We can now introduce our key concept of a partially materialized tree decomposition, tailored for CQAPs. Let $\varphi(\mathbf{x}_H \mid \mathbf{x}_A)$ be a CQAP such that $A \subseteq H$. Let $\mathcal{H}$ be the hypergraph associated with $\varphi(\mathbf{x}_H)$, the access CQ.

Definition 3.2 (PMTD). A *Partially Materialized Tree Decomposition (PMTD)* of the CQAP $\varphi(\mathbf{x}_H \mid \mathbf{x}_A)$ with $H \supseteq A$ is a tuple $(\mathcal{T}, \chi, M, r)$ such that the following properties hold:

(1) $(\mathcal{T}, \chi)$ is a free-connex tree decomposition of $\mathcal{H}$ w.r.t. node r, called the root ; and
(2) $A \subseteq \chi(r)$; and
(3) $M \subseteq V(\mathcal{T})$ such that whenever $t \in M$ then all the nodes of its subtree (w.r.t. orienting the tree away from r) are in M.

[1]The notation $\widetilde{O}$ hides a polylogarithmic factor in $|\mathcal{D}|$.

Given a PMTD $(\mathcal{T}, \chi, M, r)$, we call M the *materialization set*. We also associate with each node $t \in V(\mathcal{T})$ a *view* with variables $\mathbf{x}_{v(t)}$, where the mapping $v : V(\mathcal{T}) \to 2^{[n]}$ is defined as follows. If the node $t \notin M$, then $v(t) \stackrel{\text{def}}{=} \chi(t)$ and the view is of the form $T_{v(t)}(\mathbf{x}_{v(t)})$, called a *T-view*. Otherwise, $t \in M$. If $t = r \in M$, define $v(r) \stackrel{\text{def}}{=} \chi(t) \cap H$. Let p be the parent node of a non-root node $t \in M$ and define

$$v(t) \stackrel{\text{def}}{=} \begin{cases} \chi(t) \cap (H \cup \chi(p)) & \text{if } p \notin M \\ \chi(t) \cap H & \text{if } p \in M \text{ and } \chi(t) \cap H \not\subseteq \chi(p) \cap H \\ \emptyset & \text{if } p \in M \text{ and } \chi(t) \cap H \subseteq \chi(p) \cap H. \end{cases}$$

The view (for each $t \in M$) then is of the form $S_{v(t)}(\mathbf{x}_{v(t)})$, called the *S-view*. This definition of S-views corresponds to running a bottom-up semijoin-reduce pass of the Yannakakis algorithm in the materialization set M of the free-connex tree decomposition $(\mathcal{T}, \chi)$. Indeed, any variables in $\chi(t) \setminus v(t)$ are safely projected out after the semijoin-reduce.

On a high level, M specifies the type of views associated with each bag (S-view or T-view), and $v(\cdot)$ pinpoints the schema of that view (possibly empty). A PMTD appoints its S-views to be materialized in the preprocessing phase and its T-views to be computed in the online phase. In the case where $M = \emptyset$, every view in the decomposition is obtained in the online phase. When $H = A$ or $H = [n]$, the free-connex property does not put any additional restrictions on the tree decompositions for a PMTD.

Example 3.3. We use the CQAP for 3-reachability as an example:

$$\phi_3(x_1, x_4 \mid x_1, x_4) \leftarrow R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_4).$$

Here, (x_1, x_4) is the access pattern. Figure 1 shows three PMTDs for the above query, along with the associated views of each bag in each PMTD. The leftmost PMTD has an empty materialization set. The middle PMTD materializes the bag $\{x_1, x_2, x_3\}$ but the associated view S_{13} projects out x_2. The rightmost PMTD materializes the only bag $\{x_1, x_2, x_3, x_4\}$ but the view S_{14} keeps only the variables x_1, x_4.

Redundancy & Domination. We say that a tree decomposition is *non-redundant* if no bag is a subset of another bag. We say that a tree decomposition $(\mathcal{T}_1, \chi_1)$ is *dominated* by another tree decomposition $(\mathcal{T}_2, \chi_2)$ if every bag of $(\mathcal{T}_1, \chi_1)$ is a subset of some bag of $(\mathcal{T}_2, \chi_2)$. Here, we will generalize both notions to PMTDs.

Definition 3.4 (PMTD Redundancy). A PMTD $(\mathcal{T}, \chi, M, r)$ is *non-redundant* if (1) for $t \in M$, $v(t) \neq \emptyset$ and no $v(t)$ is a subset of another; and (2) for $t \notin M$, no $v(t)$ is a subset of another.

Definition 3.5 (PMTD Domination). A PMTD $(\mathcal{T}_1, \chi_1, M_1, r_1)$ is dominated by another PMTD $(\mathcal{T}_2, \chi_2, M_2, r_2)$ if (1) for every node $t_1 \in M_1$, there is some node $t_2 \in M_2$ such that $v(t_1) \subseteq v(t_2)$, and (2) for every node $t_1 \in V(\mathcal{T}_1) \setminus M_1$, there is some node $t_2 \in V(\mathcal{T}_2) \setminus M_2$ such that $v(t_1) \subseteq v(t_2)$.

For PMTDs, both redundancy and domination are defined using the materialization set and views instead of the bags. For PMTDs with $M = \emptyset$, both PMTD redundancy and domination become equivalent to the standard definition.

Example 3.6. Continuing Example 3.3, suppose we consider a PMTD with that takes the same tree decomposition as the left PMTD, but with both bags in the materialization set. The S-view associated with the root bag is S_{14}, and $S_\emptyset$ for the child bag; thus, this PMTD is redundant. Moreover, suppose we consider a PMTD with one bag $\{x_1, x_2, x_3, x_4\}$ which is the root, but is not in M. The T-view associated with this bag is T_{1234}; thus, this PMTD dominates the left PMTD in Figure 1. On the other hand, all PMTDs in Figure 1 are non-redundant and non-dominant to each other.

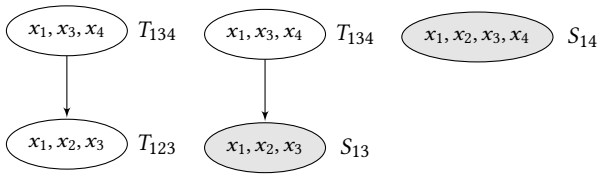


Figure 1: Three PMTDs for the 3-reachability CQAP. The materialized nodes are shaded and labeled as S-views.

As we later suggest in our general framework, we mostly focus on sets of non-redundant and non-dominant PMTDs. Note that a non-redundant PMTD $(\mathcal{T}, \chi, M, r)$ satisfies $v(t) \neq \emptyset$, for any $t \in V(\mathcal{T})$, thus we can safely assume that all views are non-empty.

3.1 Online Yannakakis for PMTDs

We introduce an adaptation of the Yannakakis algorithm [36] for a non-redundant PMTD (so no empty views), called *Online Yannakakis*. Recall that for a non-redundant PMTD, the S-views, one for each $t \in M$, are stored in the preprocessing phase, while the T-views, one for each $t \in \mathcal{T} \setminus M$, and the access request Q_A are accessible only in the online phase.

Theorem 3.7. *Consider a PMTD $(\mathcal{T}, \chi, M, r)$ and its view $v(\cdot)$. Given S-views, we can preprocess them in space linear in their size such that we can compute the free-connex acyclic CQ*

$$\psi(\mathbf{x}_H) \leftarrow Q_A \wedge \bigwedge_{t \in M} S_{v(t)} \wedge \bigwedge_{t \in V(\mathcal{T}) \setminus M} T_{v(t)} \tag{3}$$

for any T-view and Q_A in time $O(\max_{t \in V(\mathcal{T}) \setminus M} |T_{v(t)}| + |Q_A| + |\psi|)$, where $|\psi|$ is the output size of (3).

Note that the time cost has no dependence on the size of S-views, because throughout Online Yannakakis, S-views will be only used for hash probing in semijoin operations. We defer the details of the algorithm and the proof of its correctness to the full version of the paper [37].

4 GENERAL FRAMEWORK

Consider a CQAP $\varphi(\mathbf{x}_H \mid \mathbf{x}_A) \leftarrow \bigwedge_{F \in \mathcal{E}} R_F(\mathbf{x}_F)$ with $H \supseteq A$. Recall that our goal is to find the best space-time tradeoffs under degree constraints DC (guarded by input relations) and AC (guarded by any access requests Q_A), as specified in Subsection 2.2. Our main algorithm is parameterized by:

- $\mathcal{P} = \{P_i\}_{i \in I}$, a (finite) indexed set of non-redundant and non-dominant PMTDs such that $P_i = (\mathcal{T}_i, \chi_i, M_i, r_i)$ for every $i \in I$. Including all such PMTDs in $\mathcal{P}$ (which are finite) will result in the best possible tradeoff. However, as we will see later, it is meaningful to consider smaller sets of PMTDs that result in more interpretable space-time tradeoffs.

- S, the space budget.

4.1 2-Phase Disjunctive Rules

In this section, we define a specific type of disjunctive rule that will be necessary to acquire the S-views and T-views for PMTDs. We start by recalling the notion of a disjunctive rule. A disjunctive rule has the exact body of a CQ, while the head is a disjunction of output relations $T_B(\mathbf{x}_B)$, which we call *targets*. Let $\mathsf{BT} \subseteq 2^{[n]}$ be a non-empty set, then a *disjunctive rule* ρ takes the form:

$$\rho: \quad \bigvee_{B\in\mathsf{BT}} T_B(\mathbf{x}_B) \leftarrow \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F). \tag{4}$$

Given a database instance $\mathcal{D}$, a *model* of ρ is a tuple $(T_B)_{B\in\mathsf{BT}}$ of relations, one for each target, such that the logical implication indicated by (4) holds. More precisely, for any tuple $\mathbf{a}$ that satisfies the body, there is a target $T_B \in (T_B)_{B\in\mathsf{B}}$ such that $\Pi_B(\mathbf{a}) \in T_B$. The *size* of a model is defined as the maximum size of its output relations and the *output size* of a disjunctive rule ρ, denoted as $|\rho|$, is defined as the minimum size over all models.

For our purposes, we define a type of disjunctive rules, called *2-phase disjunctive rules*.

Definition 4.1 (2-phase Disjunctive Rules). A *2-phase disjunctive rule* ρ defined by a CQAP $\varphi(\mathbf{x}_H \mid \mathbf{x}_A)$ is a single disjunctive rule that takes the body of the access CQ φ, while the head has two sets of output relations. In other words, ρ takes the form

$$\rho: \quad \bigvee_{B\in\mathsf{BS}} S_B(\mathbf{x}_B) \vee \bigvee_{B\in\mathsf{BT}} T_B(\mathbf{x}_B) \leftarrow Q_A(\mathbf{x}_A) \wedge \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F), \tag{5}$$

where $\mathsf{BS}, \mathsf{BT} \subseteq 2^{[n]}$ and at most one can be empty. A *model* of ρ thus consists of two sets of output relations, i.e. the *S-targets* $(S_B)_{B\in\mathsf{BS}}$ and the *T-targets* $(T_B)_{B\in\mathsf{BT}}$.

As the name suggests, a model of a 2-phase disjunctive rule ρ is computed in two phases, the preprocessing and online phase:

Preprocessing phase: we obtain the S-targets $(S_B)_{B\in\mathsf{BS}}$ using a *preprocessing disjunctive rule*

$$\rho_S: \quad \bigvee_{B\in\mathsf{BS}} S_B(\mathbf{x}_B) \leftarrow \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F), \tag{6}$$

The space cost for storing the S-targets is $\widetilde{O}(S_\rho)$, and the overall space cost is $\widetilde{O}(S_\rho + |\mathcal{D}|)$. The preprocessing phase has no knowledge of Q_A except for the degree constraints AC, so as to explicitly force the S-targets to be universal for any instance of access request.

Online phase: given an access request Q_A (under AC), we obtain the T-targets $(T_B)_{B\in\mathsf{BT}}$ using an *online disjunctive rule*

$$\rho_T: \quad \bigvee_{B\in\mathsf{BT}} T_B(\mathbf{x}_B) \leftarrow Q_A(\mathbf{x}_A) \wedge \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F) \tag{7}$$

in time and space $\widetilde{O}(T_\rho)$. The overall time is $\widetilde{O}(T_\rho + |Q_A|)$.

If $\mathsf{BS} = \emptyset$, then the model is computed from scratch in the online phase (and vice versa). As in Subsection 2.2, our focus is on analyzing the space-time tradeoffs between the two intrinsic quantities, S_ρ and T_ρ.

For the next part, assume that we have a 2-phase algorithm (called 2PP) that, given a space budget S, has a preprocessing procedure 2PP-Preprocess using space $S_\rho \le S$ and an online procedure 2PP-Online using time (and space) T_ρ. We will discuss this algorithm in the next section.

4.2 Preprocessing Phase

As a first step, we construct from $\mathcal{P}$ a set of 2-phase disjunctive rules as follows. Let v_i be the mapping for associated views of P_i. Let us define the cartesian product $\mathbf{A} = \times_{i\in I}\{V(\mathcal{T}_i)\}$ and let $M = |\mathbf{A}|$. Informally, every element $\mathbf{a} \in \mathbf{A}$ picks one view from every PMTD in the indexed set. For every $\mathbf{a} \in \mathbf{A}$, we construct the following 2-phase disjunctive rule (recall that M_i is the materialization set of PMTD $(\mathcal{T}_i, \chi_i, M_i, r_i)$):

$$\bigvee_{\mathbf{a}_i\in M_i} S_{v_i(\mathbf{a}_i)}(\mathbf{x}_{v_i(\mathbf{a}_i)}) \vee \bigvee_{\mathbf{a}_i\notin M_i} T_{v_i(\mathbf{a}_i)}(\mathbf{x}_{v_i(\mathbf{a}_i)}) \leftarrow Q_A(\mathbf{x}_A) \wedge \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{x}_F)$$

The body of the rule is the same independent of $\mathbf{a} \in \mathbf{A}$. The head of the rule introduces an S-target whenever the corresponding bag is in the materialization set of the PMTD (using the corresponding view); otherwise, it introduces a T-target. There are exactly M 2-phase disjunctive rules constructed from the given set of PMTDs, which is a query-complexity quantity.

Example 4.2. Continuing our running example, consider the three PMTDs in Figure 1. These result in four 2-phase disjunctive rules (after removing redundant T-targets and S-targets):

$$\begin{aligned} T_{134}(x_1,x_3,x_4) \vee S_{14}(x_1,x_4) &\leftarrow \text{body} \\ T_{134}(x_1,x_3,x_4) \vee S_{13}(x_1,x_3) \vee S_{14}(x_1,x_4) &\leftarrow \text{body} \\ T_{123}(x_1,x_2,x_3) \vee T_{134}(x_1,x_3,x_4) \vee S_{14}(x_1,x_4) &\leftarrow \text{body} \\ T_{123}(x_1,x_2,x_3) \vee S_{13}(x_1,x_3) \vee S_{14}(x_1,x_4) &\leftarrow \text{body} \end{aligned}$$

where

$$\text{body} = Q_{14}(x_1,x_4) \wedge R_1(x_1,x_2) \wedge R_2(x_2,x_3) \wedge R_3(x_3,x_4)$$

For each 2-phase disjunctive rule ρ_k, where $k \in [M]$, we run 2PP-Preprocess with the space budget S. 2PP-Preprocess generates the S-targets for ρ_k. Next, we compute each S-view of a PMTD P_i by unioning all S-targets with the same schema as the S-view (possibly from outputs of different disjunctive rules). Then, we semijoin-reduce every S-view with the full join $\bowtie_{F\in\mathcal{E}} R_F$. This semijoin-reduce can be accomplished by tentatively storing $\bowtie_{F\in\mathcal{E}} R_F$ as an intermediate truth table and remove it after the semijoin-reduce of all S-views. This step guarantees that any tuple in a S-view participates in $\bowtie_{F\in\mathcal{E}} R_F$. Finally, we preprocess the S-views as described in Theorem 3.7.

4.3 Online Phase

Recall that upon receiving an instance of access request Q_A, we need to return the results of the CQ, $\varphi(\mathbf{x}_H)$. We obtain $\varphi(\mathbf{x}_H)$ as follows. First, we apply 2PP-Online for every ρ_k to get its T-targets (of size $\widetilde{O}(T_{\rho_k})$) in time $\widetilde{O}(T_{\rho_k} + |Q_A|)$. Let $T_{\max} = \max_{k\in[M]} T_{\rho_k}$.

Next, we compute each T-view of a PMTD P_i by unioning all T-targets with the same schema as the T-view (possibly from outputs of different disjunctive rules). We semijoin-reduce every T-view (of size $\widetilde{O}(T_\rho)$) with every input relation and Q_A. Then, for every PMTD in $\{P_i\}_{i\in I}$, we compute the free-connex acyclic CQ

$$\psi_i(\mathbf{x}_H) \leftarrow Q_A \wedge \bigwedge_{t\in M_i} S_{v_i(t)} \wedge \bigwedge_{t\in V_i(\mathcal{T}_i)\setminus M_i} T_{v_i(t)} \tag{8}$$

by applying Online Yannakakis as described in Section 3.1 in time $\widetilde{O}(T_{\max}) + O(|Q_A| + |\phi_i \cap \varphi|)$. We obtain the final result by unioning the outputs across all PMTDs in our set, $\varphi = \bigcup_{i \in I} \psi_i$. In total, we answer the access request Q_A in time $\widetilde{O}(T_{\max} + |Q_A|) + O(|\varphi|)$.

5 CONSTRUCTING THE TRADEOFFS

Let ρ be a 2-phase disjunctive rule taking the form (5), under degree constraints DC (guarded by input relations) and degree constraints AC (guarded by Q_A). In this section, we will discuss how we can obtain a model of ρ in two phases using PANDA, and the resulting space-time tradeoff. Due to limited space, we will keep the presentation informal and introduce the key ideas through an example. The full details and proofs are deferred to [37]. We will use the following rule as our running example, where $|R_1| = |R_2| = |\mathcal{D}|$:

$$T_{123} \vee S_{13} \leftarrow Q_{13}(x_1, x_3), R_1(x_1, x_2), R_2(x_2, x_3).$$

This rule is the only rule we obtain from considering two PMTDs for the 2-reachability query. To compute a disjunctive rule, PANDA starts with a *Shannon-flow inequality*, which is an inequality over set functions $h : 2^{[n]} \to \mathbb{R}_+$ that must hold for any set function that is a polymatroid[2]. For our purposes, we need a *joint Shannon-flow inequality*, which holds over two set functions h_S, h_T that must be polymatroids. Intuitively, h_S governs the preprocessing phase, while h_T governs the online phase. The joint Shannon-flow inequality for our example is:

$$\underbrace{h_S(1) + h_T(2|1)}_{R_1} + \underbrace{h_S(3) + h_T(2|3)}_{R_2} + 2\underbrace{h_T(13)}_{Q_{13}} \geq \underbrace{h_S(13)}_{S_{13}} + 2\underbrace{h_T(123)}_{T_{123}}$$

where $h(Y|X) = h(Y) - h(X)$. The right-hand side includes terms of h_S that correspond to S-targets and terms of h_T that correspond to T-targets. The left-hand side includes a term of h_T that corresponds to the access request Q_A, and possibly terms of h_S (h_T) that encode the degree constraints DC (DC $\cup$ AC). More importantly, it contains terms that correlate the two polymatroids by splitting an input relation with attributes Y into two parts, either (i) $h_S(X) + h_T(Y|X)$, or (ii) $h_T(X) + h_S(Y|X)$, where $X \subseteq Y$. Intuitively, the first split materializes the heavy X-values and sends everything else to the online phase, while the second split preprocesses the light X-values and sends the heavy X-values to the online phase. In our example, relation R_1 is split into $h_S(1) + h_T(2|1)$, and each part is sent to a different polymatroid. Using the coefficients of the above joint Shannon-flow inequality, we get the following intrinsic space-time tradeoff:

$$S \cdot T^2 \cong |Q_{13}|^2 \cdot |\mathcal{D}|^2$$

We will use the $\cong$ notation to mean that $S \cdot T^2 = \widetilde{O}(|Q_{13}|^2 \cdot |\mathcal{D}|^2)$. Generally, we show (for a formal definition, see the full version of the paper [37]):

Theorem 5.1 (Informal). *Every joint Shannon-flow inequality for a 2-phase disjunctive rule implies a space-time tradeoff computed by reading the coefficients of the inequality.*

The above theorem requires that we are given a joint Shannon-flow inequality to obtain a space-time tradeoff. We additionally show that, given a space budget S, we can also compute via a linear program the optimal inequality that will result in the best possible answering time.

[2]A polymatroid is a set function $h : 2^{[n]} \to \mathbb{R}_+$ that is non-negative, monotone, and submodular, with $h(\emptyset) = 0$.

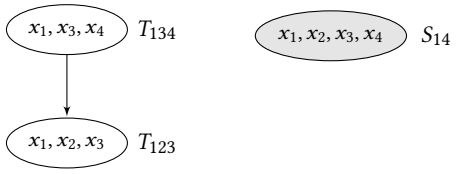


Figure 2: Two PMTDs for the square CQAP. The materialized nodes are shaded and labeled as S-views.

The* 2PP *algorithm. We now present how our main algorithm works (see [37] for a detailed description). For the running example, we take $|Q_{13}| = 1$, and S is a fixed space budget.

As a first step, 2PP scans the joint Shannon-flow inequality and partitions $R_1(x_1, x_2)$ (on x_1) into R_1^H and R_1^L, where R_1^H contains all (x_1, x_2) tuples where $|\sigma_{x_1 = t}(R_{12})| \geq |\mathcal{D}|/\sqrt{S}$, and R_1^L contains the tuples that satisfy $\deg_{12}(x_2|x_1) \leq |\mathcal{D}|/\sqrt{S}$. R_2 is partitioned symmetrically (on x_3) into R_2^H and R_2^L. This creates four subproblems, $\{R_1^H, R_2^H\}$, $\{R_1^H, R_2^L\}$, $\{R_1^L, R_2^H\}$ and $\{R_1^L, R_2^L\}$. In general, these splits will be done according to the correlated terms in the joint flow.

The preprocessing phase (2PP-Preprocess) is governed by the Shannon-flow inequality for h_S, which is $h_S(1) + h_S(3) \geq h_S(13)$. We now follow PANDA and construct a *proof sequence* for this inequality. A proof sequence proves the inequality via a sequence of smaller steps, such that each step can be interpreted as a relational operator. The proof sequence for our case is:

$$\begin{aligned} h_S(1) + h_S(3) &\geq h_S(13|3) + h_S(3) && (\textit{submodularity}) \\ &= h_S(13) && (\textit{composition}) \end{aligned}$$

In this case, PANDA attempts to join the two relations in each subproblem. However, we allow this to happen only if the resulting space is at most S. Because R_1^H and R_2^H have size at most $|\mathcal{D}|/(|\mathcal{D}|/\sqrt{S}) = \sqrt{S}$ values for x_1, x_3 respectively, the subproblem $\{R_1^H, R_2^H\}$ can be stored in S_{13} in space at most $\sqrt{S} \cdot \sqrt{S} = S$.

The online phase (2PP-Online) takes an access request $Q_{13}(x_1, x_3)$ that contains one tuple. Now, 2PP-Online follows the second proof sequence for the polymatroid h_T:

$$\begin{aligned} h_T(2|1) + h_T(2|3) + 2h_T(13) &\geq 2h_T(2|13) + 2h_T(13) && (\textit{submod.}) \\ &= 2h_T(123) && (\textit{comp.}) \end{aligned}$$

For the other 3 subproblems, 2PP-Online computes the following 3 joins: $Q_{13}(x_1, x_3) \bowtie R_2^L(x_2, x_3)$, $Q_{13}(x_1, x_3) \bowtie R_1^L(x_1, x_2)$ and $Q_{13}(x_1, x_3) \bowtie R_1^L(x_1, x_2)$. In the submodularity step, 2PP-Online identifies that for the first join, $\deg_{23}(x_2|x_3) \leq |\mathcal{D}|/\sqrt{S}$, so this join takes time $|Q_{13}| \cdot |\mathcal{D}|/\sqrt{S} \leq |\mathcal{D}|/\sqrt{S}$; and since $\deg_{12}(x_2|x_1) \leq |\mathcal{D}|/\sqrt{S}$, the last two identical joins take time $|Q_{13}| \cdot \deg_{12}(x_2|x_1) \leq |\mathcal{D}|/\sqrt{S}$. Therefore, the overall online computing time is $|\mathcal{D}|/\sqrt{S}$.

Example 5.2 (The square query). We now give a comprehensive example of how to construct tradeoffs for the following CQAP:

$$\varphi(x_1, x_3 \mid x_1, x_3) \leftarrow R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_4) \wedge R_4(x_4, x_1).$$

This captures the following task: given two vertices of a graph, decide whether they occur in two opposite corners of a square. We consider two PMTDs. The first PMTD has a root bag $\{1,3,4\}$ associated with a T-view T_{134}, and a bag $\{1,3,2\}$ associated with a T-view T_{132}. The second PMTD has one bag $\{1,2,3,4\}$ associated with an S-view S_{13}. The two PMTDs are depicted in Figure 2. This in turn generates two disjunctive rules:

$$T_{134} \vee S_{13} \leftarrow \text{body}, \qquad T_{132} \vee S_{13} \leftarrow \text{body}$$

where body $= Q_{13}(x_1,x_3) \wedge R_1(x_1,x_2) \wedge R_2(x_2,x_3) \wedge R_3(x_3,x_4) \wedge R_4(x_4,x_1)$. We can construct the following joint Shannon-flow inequality (and its proof sequence) for the first rule:

$$\begin{aligned}
&\underbrace{h_S(1) + h_T(4|1)}_{R_4} + \underbrace{h_S(3) + h_T(4|3)}_{R_3} + 2 \cdot \underbrace{h_T(13)}_{Q_{13}} \\
&\quad \geq h_S(13) + h_T(4|1) + h_T(4|3) + 2 \cdot h_T(13) \\
&\quad \geq h_S(13) + h_T(4|13) + h_T(13) + h_T(4|13) + h_T(13) \\
&\quad = \underbrace{h_S(13)}_{S_{13}} + 2 \cdot \underbrace{h_T(134)}_{T_{134}}.
\end{aligned}$$

For the second rule, we symmetrically construct a proof sequence for $2\log|\mathcal{D}| + 2\log|Q_{13}| \geq h_S(13) + 2 \cdot h_T(132)$. Hence, reading the coefficients of the above joint Shannon-flow inequalities, we obtain the following intrinsic space-time tradeoff $S \cdot T^2 \cong |\mathcal{D}|^2 \cdot |Q_{13}|^2$ for the given square CQAP.

6 APPLICATIONS

In this section, we apply our framework to obtain state-of-the-art space-time tradeoffs for several specific problems, as well as obtain new tradeoff results. We defer the discussion for hierarchical CQAPs to the full version of the paper [37].

6.1 Tradeoffs for k-Set Intersection

We will first study the CQAP (2) that corresponds to the non-Boolean k-Set Disjointness problem (set $y = x_{k+1}$)

$$\varphi(\mathbf{x}_{[k+1]} \mid \mathbf{x}_{[k]}) \leftarrow \bigwedge_{i \in [k]} R(x_{k+1}, x_i)$$

From the decomposition with a single node t with $\chi(t) = [k+1]$, we construct two PMTDs, one with $M_1 = \emptyset$, another with $M_2 = \{t\}$. Thus, $v_1(t) = v_2(t) = [k+1]$. This gives rise to the following (only) two-phase disjunctive rule:

$$T_{[k+1]} \vee S_{[k+1]} \leftarrow Q_{[k]}(\mathbf{x}_{[k]}) \wedge \bigwedge_{i \in [k]} R(x_{k+1}, x_i)$$

For this rule, we have the following joint Shannon-flow inequality:

$$\begin{aligned}
&h_S(k,k+1) + \sum_{i \in [k-1]} \{h_S(i|k+1) + h_T(k+1)\} + (k-1) \cdot h_T([k]) \\
&\quad \geq h_S([k+1]) + (k-1) \cdot h_T([k+1]).
\end{aligned}$$

By Theorem 5.1, we get the tradeoff $S \cdot T^{k-1} \cong |\mathcal{D}|^k \cdot |Q_A|^{k-1}$.

6.2 Tradeoffs via Fractional Edge Covers

Let $\varphi(\mathbf{x}_A \mid \mathbf{x}_A)$ be a CQAP with hypergraph $([n], \mathcal{E})$ of φ. A fractional edge cover of $S \subseteq [n]$ is an assignment $\mathbf{u} = (u_F)_{F \in \mathcal{E}}$ such that (i) $u_F \geq 0$, and (ii) for every $i \in S$, $\sum_{F: i \in F} u_F \geq 1$. For any fractional edge cover $\mathbf{u}$ of $[n]$, we define the *slack* of $\mathbf{u}$ w.r.t. $A \subseteq [n]$:

$$\alpha(\mathbf{u}, A) \stackrel{\text{def}}{=} \min_{i \notin A} \sum_{F \in \mathcal{E}: i \in F} u_F.$$

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 variables not in A. Hence $(u_F/\alpha(\mathbf{u},A))_{F \in \mathcal{E}}$ is a fractional edge cover of $[n] \setminus A$. We always have $\alpha(\mathbf{u}, A) \geq 1$.

Theorem 6.1. *Let $\varphi(\mathbf{x}_A \mid \mathbf{x}_A)$ be a CQAP. Let $\mathbf{u}$ be any fractional edge cover of the hypergraph of φ. Then, for any input database $\mathcal{D}$, and any access request, the following intrinsic tradeoff holds:*

$$S \cdot T^{\alpha(\mathbf{u},A)} \cong |Q_A|^{\alpha(\mathbf{u},A)} \cdot \prod_{F \in \mathcal{E}} |R_F|^{u_F}$$

The above theorem can also be shown as a corollary of Theorem 1 in [13]. However, the data structure used in [13] is much more involved, since its goal is to also bound the delay during enumeration (while we are interested in total time instead). A simpler construction with the same tradeoff was shown in [12]. Our framework recovers the same result using a simple materialization strategy with two PMTDs.

Example 6.2. Consider $\varphi(\mathbf{x}_{[k]} \mid \mathbf{x}_{[k]}) \leftarrow \bigwedge_{i \in [k]} R(y, x_i)$ (corresponds to the k-Set Disjointness problem) with the fractional edge cover $\mathbf{u}$, where $u_j = 1$ for $j \in \{1, \ldots, k\}$. The slack w.r.t. $[k]$ is k, since the fractional edge cover $\hat{\mathbf{u}}$, where $\hat{u}_i = u_i/k = 1/k$ covers x. Applying Theorem 6.1, we obtain a tradeoff of $S \cdot T^k \cong |Q_A|^k \cdot |\mathcal{D}|^k$. When $|Q_A| = 1$, this matches the best-known space-time tradeoff for the k-Set Disjointness problem.

6.3 Tradeoffs via Tree Decompositions

Let $\varphi(\mathbf{x}_A \mid \mathbf{x}_A)$ be a CQAP. In the previous section, we recovered a space-time tradeoff using two trivial PMTDs. Here, we will show how our framework recovers a better space-time tradeoff by considering a larger set of PMTDs that corresponds to one decomposition.

Pick any arbitrary non-redundant free-connex decomposition $(\mathcal{T}, \chi, r)$. We start by taking any set of nodes that are not ancestors of each other in the decomposition as a materialization set. Then, for each node t in the materialization set, we merge all bags in the subtree of t into the bag of t (and truncate the subtree). By ranging over all such materialization sets, we construct a fixed (finite) set of PMTDs. We say that this set of PMTDs is *induced* from $(\mathcal{T}, \chi, r)$. We now input the induced set of PMTDs to our general framework. To discuss the obtained space-time tradeoff, take any assignment of a fractional edge cover $\mathbf{u}_t$ to each node $t \in V(\mathcal{T})$ and let u_t^* be its total weight. Let A_t denote the common variables between node t and its parent (for the root, $A_r = A$), and define $\alpha_t = \alpha(\mathbf{u}_t, A_t)$ to be the slack in node t w.r.t. A_t. Now, take the nodes P of any root-to-leaf path in $\mathcal{T}$. We can show that any such path P generates the following intrinsic tradeoff:

$$S^{\sum_{t \in P} 1/\alpha_t} \cdot T \cong |Q_A| \cdot |\mathcal{D}|^{\sum_{t \in P} u_t^*/\alpha_t}$$

To obtain the final space-time tradeoff, we take the worst such tradeoff across all root-to-leaf paths. We show in [37] how to obtain this tradeoff, and also show why it recovers prior results [12]. Our framework guarantees that adding PMTDs to the induced set we considered here can only make this tradeoff better.

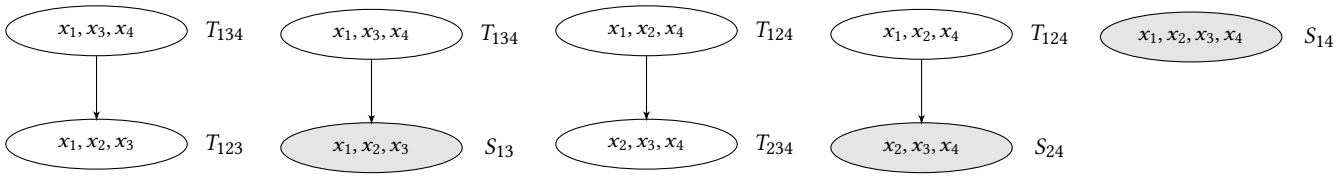


Figure 3: The PMTDs for the 3-reachability CQAP.

Table 1: 2-phase disjunctive rules for 3-reachability

rule	head	tradeoff
ρ_1	$T_{134} \vee T_{124} \vee S_{14}$	$S \cdot T^2 \cong \lvert\mathcal{D}\rvert^2 \cdot \lvert Q_A\rvert^2$
ρ_2	$T_{123} \vee S_{13} \vee T_{124} \vee S_{14}$	$S^2 \cdot T^3 \cong \lvert\mathcal{D}\rvert^4 \cdot \lvert Q_A\rvert^3$ $T \cong \lvert\mathcal{D}\rvert \cdot \lvert Q_A\rvert$
ρ_3	$T_{134} \vee T_{234} \vee S_{24} \vee S_{14}$	$S^2 \cdot T^3 \cong \lvert\mathcal{D}\rvert^4 \cdot \lvert Q_A\rvert^3$ $T \cong \lvert\mathcal{D}\rvert \cdot \lvert Q_A\rvert$
ρ_4	$T_{123} \vee S_{13} \vee T_{234} \vee S_{24} \vee S_{14}$	$S \cdot T \cong \lvert\mathcal{D}\rvert^2 \cdot \lvert Q_A\rvert$ $S^4 \cdot T \cong \lvert\mathcal{D}\rvert^6 \cdot \lvert Q_A\rvert$ $T \cong \lvert\mathcal{D}\rvert \cdot \lvert Q_A\rvert$

Example 6.3. Consider the 4-reachability CQAP. Here, $H = A = \{x_1, x_5\}$. We will consider the tree decomposition with bags $t_1 = \{x_1, x_2, x_4, x_5\} \rightarrow t_2 = \{x_2, x_3, x_4\}$.

Take the edge cover $u_1 = 1, u_4 = 1$ for the bag t_1, and the edge cover $u_2 = 1, u_3 = 1$ for the bag t_2. The first bag has slack $\alpha_1 = 1$ (w.r.t. x_1, x_5), while the second has slack $\alpha_2 = 2$ (w.r.t. x_2, x_4). Here we have one root-to-leaf path, hence we get the tradeoff $S^{1+1/2} \cdot T \cong |Q_A| \cdot |\mathcal{D}|^{2/1+2/2}$, or equivalently $S^{3/2} \cdot T \cong |Q_A| \cdot |\mathcal{D}|^3$.

6.4 Tradeoffs for k-Reachability

In this part, we will consider the CQAP that corresponds to the k-reachability problem described in Example 2.3:

$$\phi_k(x_1, x_{k+1} \mid x_1, x_{k+1}) \leftarrow \bigwedge_{i \in [k]} R(x_i, x_{i+1}).$$

Prior work [15] has shown the following tradeoff for a input $\mathcal{D}$, which was conjectured to be asymptotically optimal for $|Q_A| = 1$:

$$S \cdot T^{2/(k-1)} \cong |\mathcal{D}|^2 \cdot |Q_A|^{2/(k-1)}.$$

We will show that the above tradeoff can be significantly improved for $k \geq 3$ by applying our framework.

3-reachability. As a first step, we consider the set of all non-redundant and non-dominant PMTDs (five in total), as seen in Figure 3. The five PMTDs will lead to $2^4 = 16$ disjunctive rules, but we can reduce the number of rules we analyze by discarding rules with strictly more targets than other rules. For example, the disjunctive rule with head $T_{134} \vee S_{13} \vee T_{124} \vee S_{14}$ can be ignored, since there is another disjunctive rule which has a strict subset of targets, i.e., $T_{134} \vee T_{124} \vee S_{14}$. We list out the heads of the two-phase disjunctive rules we need to consider (we omit the variables for simplicity), along with the intrinsic tradeoffs for each rule in Table 1.

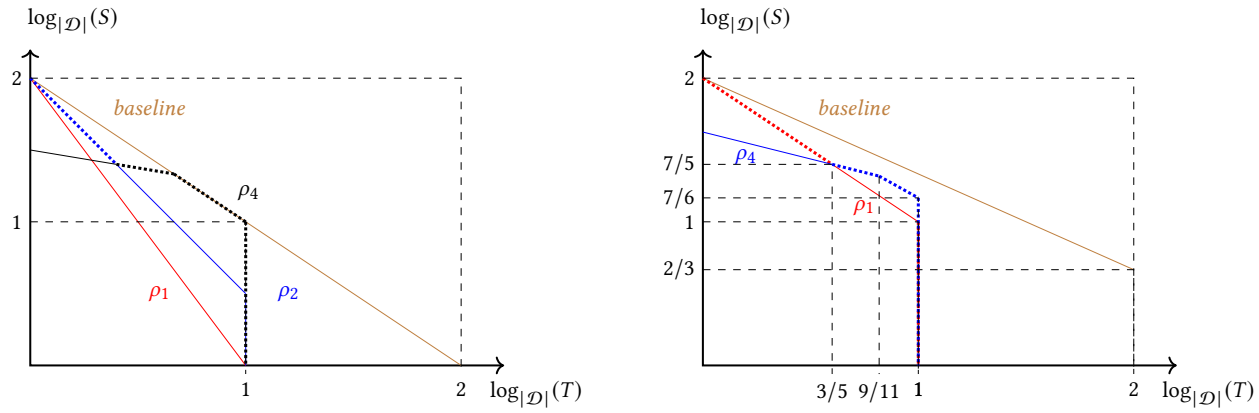
Note that rules can admit two (or more) tradeoffs that do not dominate each other; hence, we need to pick the best tradeoff depending on the regime we consider (see [37] for how we prove each tradeoff). To understand the combined tradeoff we obtain from our analysis, we plot in Figure 4a each tradeoff curve by taking $\log_{|\mathcal{D}|}$ and then taking the x-axis as $\log_{|\mathcal{D}|}(T)$ and the y-axis as $\log_{|\mathcal{D}|}(S)$ (fixing $|Q_A| = 1$). The dotted line in the figure shows the resulting tradeoff, which is a piecewise linear function. Note that each linear segment denotes a different strategy that is optimal for that regime of space. Note that Figure 4a is not necessarily optimized for $|Q_A| > 1$. Suppose that $S = |\mathcal{D}|$ and we receive $|\mathcal{D}|$ single-tuple access requests in the online phase. Answering them one-by-one costs time $\widetilde{O}(|\mathcal{D}|^2)$ using the above tradeoffs. However, one better strategy is to batch the $|\mathcal{D}|$ tuples (into a 4-cycle query) and use PANDA to answer it from scratch, which costs time $\widetilde{O}(|\mathcal{D}|^{3/2})$.

4-reachability. We also study the CQAP for the 4-reachability problem. We leave the (quite complex) calculations to [37], but we include here a plot (Figure 4b) similar to the one in Figure 4a. One surprising observation is that the new space-time tradeoff is better than the prior state-of-the-art for *every regime of space*. We should also point out that the tradeoff can possibly be further improved by including even more PMTDs (our analysis involved 11 PMTDs!), but the calculations were beyond the scope of this work.

General reachability. Analyzing the best possible tradeoff for $k \geq 5$ becomes a very complex proposition. However, from Subsection 6.3 and the analysis of [12], our framework can at least obtain the $S \cdot T^{2/(k-1)} \cong |\mathcal{D}|^2$ tradeoff, and can likely strictly improve it.

7 RELATED WORK

Set Intersection and Distance Oracles. Space-time tradeoffs for query answering (exact and approximate) has been an active area of research across multiple communities in the last decade [1, 8, 25, 26]. Cohen and Porat [8] introduced the fast intersection problem and presented a data structure to enumerate the intersection of two sets with guarantees on the total answering time. Goldstein et. al [15] formulated the k-reachability problem on graphs, and showed a simple recursive data structure which achieves the $S \cdot T^{2/(k-1)} = O(|\mathcal{D}|^2)$ tradeoff. They also conjectured that the tradeoff is optimal and used it to justify the optimality of an approximate distance oracle proposed by [2]. The study of (approximate) distance oracles over graphs was initiated by Patrascu and Roditty [32], 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. Cohen and Porat [9] also connected set intersection to distance oracles. Agarwal et al. [2, 3] introduced the notion of *stretch* of an oracle that controls the error allowed



(a) 3-reachability CQAP. **(b) 4-reachability CQAP.**

Figure 4: Space-time tradeoffs for the 3- and 4-reachability CQAP. The new tradeoffs obtained from our framework are depicted via the dotted segments. The brown lines (baseline) show the previous state-of-the-art tradeoffs.

in the answer. Further, for stretch-2 and stretch-3 oracles, we can achieve tradeoffs $S \cdot T = O(|\mathcal{D}|^2)$ and $S \cdot T^2 = O(|\mathcal{D}|^2)$ respectively, and for any integer $k > 0$, a stretch-$(1 + 1/k)$ oracle exhibits an $S \cdot T^{1/k} = O(|\mathcal{D}|^2)$ tradeoff. Unfortunately, no lower bounds are known for non-constant query time.

Space/Delay Tradeoffs. A different line of work considers the problem of enumerating query results of a non-Boolean query, with the goal of minimizing the *delay* between consecutive tuples of the output. In constant delay enumeration [4, 33], the goal is to achieve constant delay for a CQ after a preprocessing step of linear time (and space); however, only a subset of CQs can achieve such a tradeoff. Factorized databases [31] achieve constant delay enumeration after a more expensive super-linear preprocessing step for any CQ. If we want to reduce preprocessing time further, it is necessary to increase the delay. Kara et. al [20] presented a tradeoff between preprocessing time and delay for enumerating the results of any hierarchical CQ under static (and dynamic) settings. Deng et.al [14] initiates the study of the space-query tradeoffs for range subgraph counting and range subgraph listing problems. The problem of CQs with access patterns was first introduced by Deep and Koutris [13] (under the restriction $|Q_A| = 1$), but the authors only consider full CQAPs. Previous work [35] considered the problem of constructing space-efficient views of graphs to perform graph analytics, but did not offer any theoretical guarantees. More recently, Kara et. al [21] studied tradeoffs between preprocessing time, delay, and update time for CQAPs. They characterized the class of CQAPs that admit linear preprocessing time, constant delay enumeration, and constant update time. All of the above results concern the tradeoff between space (or preprocessing time) and delay, while our work focuses on the total time to answer the query. Our work is most closely related to the non-peer-reviewed work in [12]. There, the authors also study the problem of building tradeoffs for Boolean CQs. The authors propose two results that slightly improve upon [13]. They were also the first to recognize that the k-reachability tradeoff is not optimal by proposing a small improvement for $k \geq 3$. The results in our work are a vast generalization that is achieved using a more comprehensive framework. For the dynamic setting, [5] initiated the study of answering CQs under updates. Recently, [19] presented an algorithm for counting the number of triangles under updates. [21] proposed dynamic algorithms for CQAPs and provided a syntactic characterization of queries that admit constant time per single-tuple update and whose output tuples can be enumerated with constant delay.

CQ Evaluation. Our proposed framework is based on recent advances in efficient CQ evaluation, and in particular the PANDA algorithm [24]. This powerful algorithmic result follows a long line of work on query decompositions [16, 27, 28], worst-case optimal algorithms [29], and connections between CQ evaluation and information theory [22, 23].

8 CONCLUSION

In this paper, we present a framework for computing general space-time tradeoffs for answering CQs with access patterns. We show the versatility of our framework by demonstrating how it can capture state-of-the-art tradeoffs for problems that have been studied separately. The application of our framework has also uncovered previously unknown tradeoffs. Many open problems remain, among which are obtaining (conditional) lower bounds that match our upper bounds, and investigating how to make our approach practical.

Many open problems remain that merit further work. In particular, there are no known lower bounds to prove the optimality of the space-time tradeoffs. The optimality of existing space-time tradeoffs for approximate distance oracles is also now an open problem again and we believe our proposed framework should be able to improve the upper bounds. It would also be very interesting to see the applicability of this framework in practice. In particular, our framework can be extended to also include views that have been precomputed, which is a common setting. In this regard, challenges remain to optimize the constants in the time complexity to ensure implementation feasibility.

REFERENCES

[1] Peyman Afshani and Jesper Asbjørn Sindahl Nielsen. Data structure lower bounds for document indexing problems. In *ICALP*, 2016.

[2] Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In *ESA*, pages 49–60. Springer, 2014.

[3] Rachit Agarwal, P Brighten Godfrey, and Sariel Har-Peled. Approximate distance queries and compact routing in sparse graphs. In *INFOCOM*, pages 1754–1762. IEEE, 2011.

[4] 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.

[5] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. Answering conjunctive queries under updates. In *PODS*, pages 303–318. ACM, 2017.

[6] Angela Bonifati, Wim Martens, and Thomas Timm. An analytical study of large sparql query logs. *The VLDB Journal*, 29(2):655–679, 2020.

[7] Timothy M Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In *STOC*, pages 31–40, 2015.

[8] Hagai Cohen and Ely Porat. Fast set intersection and two-patterns matching. *Theoretical Computer Science*, 411(40-42):3795–3800, 2010.

[9] Hagai Cohen and Ely Porat. On the hardness of distance oracle for sparse graph. *arXiv preprint arXiv:1006.1117*, 2010.

[10] Nilesh Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Communications of the ACM*, 52(7):86–94, 2009.

[11] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Enumeration algorithms for conjunctive queries with projection. In *24th International Conference on Database Theory*, page 1, 2021.

[12] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Space-time tradeoffs for answering boolean conjunctive queries. *arXiv preprint arXiv:2109.10889*, 2021.

[13] Shaleen Deep and Paraschos Koutris. Compressed representations of conjunctive query results. In *PODS*, pages 307–322. ACM, 2018.

[14] Shiyuan Deng, Shangqi Lu, and Yufei Tao. Space-query tradeoffs in range subgraph counting and listing. In *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, pages 6:1–6:25, 2023.

[15] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. Conditional lower bounds for space/time tradeoffs. In *WADS*, pages 421–436. Springer, 2017.

[16] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Treewidth and hypertree width. *Tractability: Practical Approaches to Hard Problems*, 1, 2014.

[17] Gianluigi Greco and Francesco Scarcello. Structural tractability of enumerating csp solutions. *Constraints*, 18(1):38–74, 2013.

[18] Muhammad Idris, Martín Ugarte, and Stijn Vansummeren. The dynamic yannakakis algorithm: Compact and efficient query processing under updates. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1259–1274, 2017.

[19] Ahmet Kara, Hung Q Ngo, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. Counting triangles under updates in worst-case optimal time. In *ICDT*, 2019.

[20] Ahmet Kara, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. Trade-offs in static and dynamic evaluation of hierarchical queries. In *PODS*, pages 375–392, 2020.

[21] Ahmet Kara, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. Conjunctive queries with free access patterns under updates. In *Proceedings of the 26th International Conference on Database Theory (ICDT 2023)*, 2022. The 26th International Conference on Database Theory, 2023, ICDT 2023 ; Conference date: 28-03-2023 Through 31-03-2023.

[22] 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.

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

[24] 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.

[25] Tomasz Kociumaka, Jakub Radoszewski, and Wojciech Rytter. Efficient indexes for jumbled pattern matching with constant-sized alphabet. In *ESA*, pages 625–636. Springer, 2013.

[26] Kasper Green Larsen, J Ian Munro, Jesper Sindahl Nielsen, and Sharma V Thankachan. On hardness of several string indexing problems. *Theoretical Computer Science*, 582:74–82, 2015.

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

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

[29] 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.

[30] Dan Olteanu and Maximilian Schleich. Factorized databases. *ACM SIGMOD Record*, 45(2):5–16, 2016.

[31] Dan Olteanu and Jakub Závodný. Size bounds for factorised representations of query results. *ACM Trans. Database Syst.*, 40(1):2:1–2:44, 2015.

[32] Mihai Patrascu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. In *FOCS*, pages 815–823. IEEE, 2010.

[33] Luc Segoufin. Enumerating with constant delay the answers to a query. In *ICDT*, pages 10–20. ACM, 2013.

[34] Yilei Wang and Ke Yi. Secure yannakakis: Join-aggregate queries over private data. In *SIGMOD Conference*, pages 1969–1981. ACM, 2021.

[35] Konstantinos Xirogiannopoulos and Amol Deshpande. Extracting and analyzing hidden graphs from relational databases. *CoRR*, abs/1701.07388, 2017.

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

[37] Hangdong Zhao, Shaleen Deep, and Paraschos Koutris. Space-time tradeoffs for conjunctive queries with access patterns. *arXiv preprint arXiv:2304.06221*, 2023.