

Finding Smallest Witnesses for Conjunctive Queries

XIAO HU, Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

STAVROS SINTOS, Department of Computer Science, University of Illinois Chicago, Chicago, United States

A witness is a sub-database that preserves the query results of the original database, but of a much smaller size. It has wide applications in query rewriting and debugging, query explanation, IoT analytics, multi-layer network routing, and so on. In this article, we study the smallest witness problem (SWP) for the class of conjunctive queries (CQs) without self-joins.

We first establish the dichotomy that SWP for a CQ can be computed in polynomial time if and only if it has *head-cluster property*, unless $P = NP$. Furthermore, we discover the dichotomy that SWP for a CQ with head-cluster property can be computed in linear time if and only if it is acyclic, assuming some well-known conjectures. We next turn to the approximated version by relaxing the size of a witness from being minimum. We surprisingly find that the *head-domination property*—that has been identified for the deletion propagation problem [40]—can also precisely capture the hardness of the approximated smallest witness problem. In polynomial time, SWP for any CQ with head-domination property can be approximated within a constant factor, while SWP for any CQ without such a property cannot be approximated within a logarithmic factor, unless $P = NP$.

We further explore efficient approximation algorithms for CQs without the head-domination property: (1) we show a trivial algorithm that achieves a polynomially large approximation ratio for general CQs; (2) for any CQ with only one non-output attribute, such as star CQs, we show a greedy algorithm with a logarithmic approximation ratio; (3) for line CQs, which contain at least two non-output attributes, we relate SWP problem to the directed Steiner forest problem, whose algorithms can be applied to line CQs directly. Meanwhile, we establish an exponentially larger lower bound than above. It remains open to close the gap between the lower and upper bounds of the approximated SWP for CQs without the head-domination property.

CCS Concepts: • **Theory of computation** → **Data provenance**;

Additional Key Words and Phrases: Conjunctive query, smallest witness, head-cluster, head-domination

ACM Reference Format:

Xiao Hu and Stavros Sintos. 2026. Finding Smallest Witnesses for Conjunctive Queries. *ACM Trans. Datab. Syst.* 51, 4, Article 24 (June 2026), 31 pages. <https://doi.org/10.1145/3786785>

1 Introduction

To deal with large-scale data in analytical applications, people have developed a large body of data summarization techniques to reduce computational as well as storage complexity, such as sampling [16, 18, 55], sketch [22], coresets [50], and factorization [47]. The notion of witness

Authors' Contact Information: Xiao Hu (corresponding author), Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada; e-mail: xiaohu@uwaterloo.ca; Stavros Sintos, Department of Computer Science, University of Illinois Chicago, Chicago, Illinois, United States; e-mail: stavros@uic.edu.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 0362-5915/2026/06-ART24

<https://doi.org/10.1145/3786785>

has been studied as one form of why-provenance [6, 15, 36] that provides a *proof* for output results, with wide applications in explainable data-intensive analytics. The **smallest witness problem (SWP)** was first proposed by [45] that given a query Q , a database D and one specific query result $t \in Q(D)$, the target is to find the smallest sub-database $D' \subseteq D$ such that t is witnessed by D' , i.e., $t \in Q(D')$. In this article, we consider an extended notion for all query results, i.e., our target is to find the smallest sub-database $D' \subseteq D$ such that all query results can be witnessed by D' , i.e., $Q(D) = Q(D')$. Our extended smallest witness has many useful applications in practice, such as helping students learn SQL queries [45], query rewriting, query explanation, multi-layer network routing, IoT analytics on edge devices[49]. We mention three application scenarios:

Example 1.1. Alice, located in Seattle, wants to send the query results of Q over a database D (which is also stored in Seattle) to Bob, located in New York. Unfortunately, the number of query results could be polynomially large in terms of the number of tuples in D . An alternative is to send the entire database D , since Bob can retrieve all query results by executing Q over D at New York. However, moving the entire database is also expensive. A natural question arises: is it necessary for Alice to send the entire D or $Q(D)$? If not, what is the smallest subset of tuples to send?

Example 1.2. Charlie is a novice at learning SQL in an undergraduate database course. Suppose there is a huge test database D , a correct query Q that Charlie is expected to learn, and a wrong query Q' submitted by him, where some answers in $Q(D)$ are missed from $Q'(D)$. To help Charlie debug, the instructor can simply show the whole test database D to him. However, Charlie will have to dive into such a huge database to figure out where his query goes wrong. A natural question arises: Is it necessary to show the entire D to Charlie? If not, what is the smallest subset of tuples to show so that Charlie can quickly find all missing answers by his wrong query?

Example 1.3. In a multi-layer communication network, clients and servers are connected by routers organized into layers, such that links (or edges) exist between routers residing in consecutive layers. What is the smallest subset of links needed for building a fully connected network, i.e., every client-server pair is connected via a directed path? For a given network, what is the maximum number of links that can be broken while the connectivity with respect to the client-server pairs does not change? This information could help evaluate the inherent robustness of a network to either malicious attacks or even just random failures.

Recall that our SWP finds the smallest sub-database $D' \subseteq D$ such that $Q(D) = Q(D')$. It would be sufficient for Alice to send D' , while Bob can retrieve all query results by executing Q over D' , saving much transmission cost. Also, it would be sufficient for the instructor to show D' to Charlie, from which all correct answers in $Q(D)$ can be recovered, saving Charlie much effort in exploring a huge test database.¹ Moreover, the connectivity of a multi-layer network D can be modeled as a line query Q (formally defined in Section 6.3) with connected client-server pairs as $Q(D)$, such that D' is a smallest subset of links needed for maintaining the desired connectivity, and $D - D'$ is a maximum subset of links that can be removed safely. In this article, we aim to design algorithms that can efficiently compute or approximate the smallest witness for **conjunctive queries (CQs)**, and understand the hardness of this problem when such algorithms do not exist.

¹The smallest witness for a single query result [45] has been incorporated into an educational tool (<https://dukedb-hnrrq.github.io/>), successfully employed in Duke database courses with 1,000+ undergraduate users. Our extended version can also be incorporated and save more efforts by showing one small witness for all answers.

1.1 Problem Definition

Conjunctive Queries. Let $\mathbb{R}$ be a database schema that contains m relations $R_1, R_2, \dots, R_m$. Let $\mathbb{A}$ be the set of all attributes in $\mathbb{R}$. Each relation R_i is defined on a subset of attributes $\mathbb{A}_i \subseteq \mathbb{A}$. We use $A, B, C, A_1, A_2, A_3, \dots$ and so on, to denote the attributes in $\mathbb{A}$ and $a, b, c, \dots$ and so on, to denote their values. Let $\text{dom}(A)$ be the domain of attribute $A \in \mathbb{A}$. The domain of a set of attributes $X \subseteq \mathbb{A}$ is defined as $\text{dom}(X) = \prod_{A \in X} \text{dom}(A)$. Given the database schema $\mathbb{R}$, let D be a given database of $\mathbb{R}$, and let the corresponding relations of $R_1, \dots, R_m$ be $R_1^D, \dots, R_m^D$, where R_i^D is a collection of tuples defined on $\text{dom}(\mathbb{A}_i)$. The *input size* of database D is denoted as $N = |D| = \sum_{i \in [m]} |R_i^D|$. Where D is clear from the context, we will drop the superscript.

We consider the class of CQs *without self-joins*:

$$Q(\mathbf{A}) : -R_1(\mathbb{A}_1), R_2(\mathbb{A}_2), \dots, R_m(\mathbb{A}_m),$$

where $\mathbf{A} \subseteq \mathbb{A}$ is the set of *output attributes* (a.k.a. *free attributes*) and $\mathbb{A} - \mathbf{A}$ is the set of *non-output attributes* (a.k.a. *existential attributes*). For brevity, throughout the paper, we refer to a CQ without self-joins as just “CQ”. A CQ is *full* if $\mathbf{A} = \mathbb{A}$, indicating the natural join among the given relations; otherwise, it is *non-full*. Each R_i in Q is distinct. When a CQ Q is evaluated on database D , its query result denoted as $Q(D)$ is the projection of natural join result of $R_1(\mathbb{A}_1) \bowtie R_2(\mathbb{A}_2) \bowtie \dots \bowtie R_m(\mathbb{A}_m)$ onto $\mathbf{A}$ (after removing duplicates).

Extending the notation above, we use $\text{rels}(Q)$ to denote all the relations that appear in the body of Q , and use $\text{attr}(Q)$, $\text{head}(Q) \subseteq \text{attr}(Q)$ to denote all the attributes that appear in the body, head of Q separately (so, $\text{head}(Q) = \mathbf{A}$ in Section 1.1). Moreover, $\text{head}(R_i) = \text{head}(Q) \cap \text{attr}(R_i)$. For any attribute $A \in \text{attr}(R_i)$, $\pi_{A,t}$ denotes the value over attribute A of tuple t . Similarly, for a set of attributes $X \subseteq \text{attr}(R_i)$, $\pi_{X,t}$ denotes values over attributes in X of tuple t .

We mention two important classes of CQs that will be discussed throughout the paper. A CQ Q is α -acyclic [10, 29] if there exists a tree $\mathbb{T}$ such that (1) there is a one-to-one correspondence between the nodes of $\mathbb{T}$ and relations in Q and (2) for every attribute $A \in \text{attr}(Q)$, the set of nodes containing A forms a connected subtree of $\mathbb{T}$. A CQ is *free-connex* [9] if it is acyclic and the resulting CQ after adding a relation containing all output attributes is still acyclic.

Witness and Smallest Witness. Now, we are ready to formally define the notion of *witness* and the SWP problem. Below, we introduce a generalized version of the witness as well as the SWP problem to help clarify the literature.

Definition 1.4 (Witness). For a CQ Q , database D and a subset of query result $S \subseteq Q(D)$, a set $D' \subseteq D$ of tuples is a *witness* to (Q, D, S) if the query result of Q over D' include every tuple in S , i.e., $S \subseteq Q(D')$.

Definition 1.5 (Elastic Smallest Witness Problem (ESWP)). For a CQ Q , database D and a subset of query result $S \subseteq Q(D)$, it asks to find a subset of tuples $D' \subseteq D$ such that $S \subseteq Q(D')$, while there exists no subset of tuples $D'' \subseteq D$ such that $S \subseteq Q(D'')$ and $|D''| < |D'|$.

Given Q, D , and S , we denote the above problem by $\text{ESWP}(Q, D, S)$. One interesting case of $\text{ESWP}(Q, D, S)$ arises when S contains exactly a single query result, which has been studied in the literature [45]. The other interesting case of $\text{ESWP}(Q, D, S)$ occurs when S contains all query results of $Q(D)$, which will be the primary focus of our paper:

Definition 1.6 (SWP). For CQ Q and database D , it asks to find a subset of tuples $D' \subseteq D$ such that $Q(D) = Q(D')$, while there exists no subset of tuples $D'' \subseteq D$ such that $Q(D) = Q(D'')$ and $|D''| < |D'|$.

The problem described above can be represented as $\text{ESWP}(Q, D, Q(D))$, or more succinctly, as $\text{SWP}(Q, D)$. An illustrative example is shown in Figure 1. We note that the solution to $\text{SWP}(Q, D)$ may

R_1		R_2		R_3		R_4		$Q(D)$		
A	B	B	C	C	F	C	H	A	C	F
a1	b1	b1	c1	c1	f1	c1	h1	a1	c1	f1
a2	b2	b2	c3	c2	f3	c2	h1	a2	c3	f3
a3	b2	b3	c2	c3	f3	c3	h1	a3	c3	f3
		b3	c3			c3	h2			

Fig. 1. An example of database schema $\mathbb{R} = \{R_1, R_2, R_3, R_4\}$ with $\mathbb{A} = \{A, B, C, F, H\}$, $\text{attr}(R_1) = \{A, B\}$, $\text{attr}(R_2) = \{B, C\}$, $\text{attr}(R_3) = \{C, F\}$ and $\text{attr}(R_4) = \{C, H\}$. An example of database D and the result of CQ $Q(A, C, F) : -R_1(A, B), R_2(B, C), R_3(C, F), R_4(C, H)$ over D . A sub-database $D' = \{(a_1, b_1), (b_1, c_1), (c_1, f_1), (c_1, h_1)\}$ is a solution to $\text{SWP}(Q, D, (a_1, c_1, f_1))$. D' together with tuples $\{(a_2, b_2), (a_3, b_2), (b_2, c_3), (c_3, f_3), (c_3, h_2)\}$ is a solution to $\text{SWP}(Q, D)$.

not be unique, hence our target simply finds one such solution. We study the data complexity [52] of ESWP as well as SWP i.e., the sizes of the database schema are considered as constants, and the complexity is in terms of the database size. For any CQ Q and database D , $|Q(D)|$ is polynomially large in terms of $|D|$, and $Q(D)$ can also be computed in polynomial time in terms of $|D|$. In contrast, the size of a witness is always smaller than $|D|$, while, as we see later, the smallest witness may not be computed in polynomial time in terms of $|D|$. Again, our target is to compute the smallest witness instead of the query results.

We say that SWP is *poly-time solvable* for Q if, for an arbitrary database D , $\text{SWP}(Q, D)$ can be computed in polynomial time in terms of database size, i.e., $|D|$. Moreover, SWP is *linear-time solvable* for Q if, for an arbitrary database D , $\text{SWP}(Q, D)$ can be computed in $O(|D|)$ time. As shown later, SWP is not poly-time solvable for a large class of CQs, so we consider approximated solutions to SWP. More specifically, for a CQ Q and database D , a θ -approximated solution to SWP is a witness D' to (Q, D) such that $|D'| \leq \theta \cdot |\text{SWP}(Q, D)|$. Also, SWP is θ -approximatable for Q if, for an arbitrary database D , there is a θ -approximated solution to $\text{SWP}(Q, D)$ that can be computed in polynomial time in terms of $|D|$.

1.2 Previous Results

For both ESWP and SWP, various special cases have been investigated in prior work. We now provide a brief review of these results below.

Smallest Witness for One Query Result. As mentioned, $\text{ESWP}(Q, D, S)$ problem has been studied for the special case when S contains a single query result.

LEMMA 1.7 ([45]). *Given an arbitrary CQ Q and a database D , $\text{ESWP}(Q, D, \{t\})$ can be computed in polynomial time in terms of $|D|$, for an arbitrary tuple $t \in Q(D)$.*

The solution to $\text{ESWP}(Q, D, \{t\})$ can be computed by first identify an arbitrary full join result $t' \in \bowtie_{R_i \in \text{rels}(Q)} R_i$ such that $\pi_{\text{head}(Q)} t' = t$, and then return all participating tuples $\{\pi_{\text{attr}(R_i)} t' : R_i \in \text{rels}(Q)\}$. Such a solution always has a size of $|\text{rels}(Q)|$. Moreover, if the CQ is *free-connex*, we can get a much faster result.

LEMMA 1.8 ([9, 54]). *Given a free-connex CQ Q and a database D , after $O(|D|)$ preprocessing time, $\text{ESWP}(Q, D, \{t\})$ can be computed in $O(1)$ time for an arbitrary tuple $t \in Q(D)$.*

Smallest Witness for Boolean CQs. The SWP problem for a Boolean CQ (indicating whether the result of the underlying natural join is empty or not) can be solved by finding the smallest witness for an arbitrary join result.

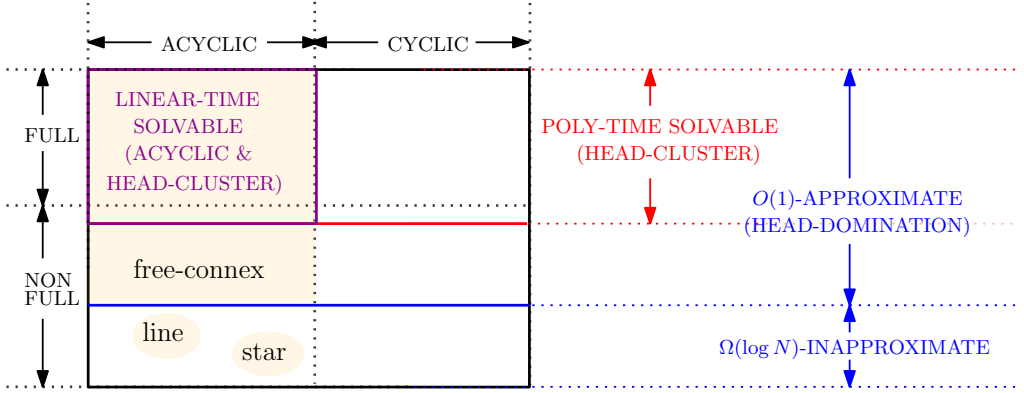


Fig. 2. Summary of our main results. The three disjoint shadow areas represent the class of free-connex CQs (Section 2), line CQs (Section 6), and star CQs (Section 6).

Smallest Witness for Full CQs. For a full CQ Q and a database D , we identify a tuple t as *dangling* if it does not appear in any query result of $Q(D)$, and *non-dangling* otherwise. The SWP for a full CQ is equivalent to computing exactly the set of non-dangling tuples:

LEMMA 1.9. *For any full CQ Q and database D , the solution to $\text{SWP}(Q, D)$ is exactly the set of non-dangling tuples that participate in at least one join result, i.e., $\bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D)$.*

The argument is not complicated. For any witness S to (Q, D) , we must have $\bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D) \subseteq S$. Suppose not, let $t \in R_i$ be an arbitrary tuple such that $t \in \bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D) - S$. Let $t' \in Q(D)$ be a query result such that $\pi_{\text{attr}(R_i)} t' = t$. As $t \notin S$, $t' \notin Q(S)$, coming to a contradiction that S is a witness to (Q, D) . Hence, $\bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D) \subseteq S$ holds for any witness S to (Q, D) . On the other hand, $\bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D)$ itself is a witness to (Q, D) . Hence, $\bigcup_{R_i \in \text{rels}(Q)} \pi_{\text{attr}(R_i)} Q(D)$ is the solution to $\text{SWP}(Q, D)$.

1.3 Our Results

Our main results obtained in this article can be summarized as follows (see Figure 2 for an illustration), which also serves as a roadmap of the whole article:

In Section 3, we start by examining how a witness can be decomposed using the structural properties of the input CQ. We then introduce a simple algorithm that, for any CQ, database, and subset of query results, computes a witness—albeit not necessarily the smallest one. This algorithm will be commonly used in later sections.

In Section 4, we establish the dichotomy of computing SWP in polynomial time. More specifically, SWP for any CQ with the head-cluster property can be computed in polynomial time, while SWP for any CQ without the head-cluster property is NP-hard by resorting to the NP-hardness of the set cover problem (Section 4.2). In Section 4.3, we further investigate the fine-grained complexity of the SWP problem for CQs with the head-cluster property. More specifically, for any CQ with the head-cluster property, if the CQ is acyclic, then SWP is linear-time solvable, and otherwise, SWP is not linear-time solvable by resorting to the strong triangle detection conjecture.

In Section 5, we build the dichotomy of approximating SWP for CQs without the head-cluster property. The head-domination property that has been identified for the **deletion propagation (DP)** problem [40], also captures the hardness of approximating SWP. We show that the ESWP-algorithm returns an $O(1)$ -approximated solution to SWP for CQs with the head-domination property. On the other hand, we prove that SWP cannot be approximated within a factor of $(1 - o(1)) \cdot \log N$ for

every CQ without the head-domination property, unless $P = NP$, by resorting to the logarithmic inapproximability of the set cover problem. Interestingly, the class of acyclic CQs for which the SWP problem can be approximated within $O(1)$ factor coincides with the class of free-connex CQs (Section 5.1), which has played an important role in the literature.

In Section 6, we further explore approximation algorithms for CQs without the head-domination property. Firstly, we show that the unified algorithm returns a $O(N^{1-1/\rho^*})$ -approximated solution for a general CQ Q , where ρ^* is the fractional edge covering number² of the input CQ [7]. Furthermore, for any CQ with only one non-output attribute, which includes the commonly-studied *star* CQs, we show a greedy algorithm that can approximate SWP within an $O(\log N)$ factor, matching the lower bound. However, for another commonly-studied class of *line* CQs, which contains more than two non-output attributes, we prove a much higher lower bound $\Omega(2^{(\log N)^{1-\epsilon}})$ for any $\epsilon > 0$ in approximating SWP, by resorting to the *label cover* problem (Appendix D). Meanwhile, we observe that SWP problem for line queries is a special case of the *directed Steiner forest* (DSF) problem (Section 6.3), and therefore existing algorithms for DSF can be applied to SWP directly. But, how to close the gap between the upper and lower bounds on approximating SWP for line CQs remains open.

Remark. An earlier version of this work was published in [48]. There are multiple new results in this new version of our work. The main differences from the previous version are:

- We introduce a generalized notion of witness as well as an extended version of SWP problem called ESWP. These extensions have brought various benefits:
 - We characterize the decomposability property of witness, which leads us to find a straightforward procedure—Algorithm 1—for finding a witness to an arbitrary CQ, a database, and a subset of query results (Section 3).
 - This witness returned enjoys very nice properties when the input CQ has certain structural properties: (1) if Q has the head-cluster property, then Algorithm 1 returns exactly a solution to ESWP; (2) if Q has the head-domination property, then Algorithm 1 returns an $O(1)$ -approximated solution to ESWP; (3) for arbitrary CQ Q , then Algorithm 1 returns an $O(|D|^{1-1/\rho^*})$ -approximated solution to ESWP. All these results apply to our original SWP problem as corollaries, hence the results achieved for SWP are much simpler than our previous presentation.
- In Section 1, we add Section 1.2 by summarizing a few special cases of ESWP and SWP that have been studied before.
- In Section 2, we remove the notion of non-output-connectivity, which is essentially implied by the existential connectivity. Hence, the presentation becomes simpler. Instead, we add Section 2.3 to clarify the connection between these two properties (head-cluster and head-domination) with the existing classifications of CQs. More specifically, we show the equivalence between the head-domination property and the free-connex property for acyclic CQs.
- In Section 4.3, we discuss the fine-grained complexity of computing SWP for CQs with the head-cluster property. Note that in the previous version, we focused on the poly-time solvability of SWP, while in this version, we further show a dichotomy of linear-time solvability of SWP for CQs with the head-cluster property. More specifically, for a CQ Q with head-cluster property, SWP is linear-time solvable if Q is acyclic and not linear-time solvable otherwise, based on some well-known conjectures.

²For a CQ Q , a fractional edge covering is a function $W : \text{rels}(Q) \rightarrow [0, 1]$ with $\sum_{R_i: A \in \text{attr}(R_i)} W(R_i) \geq 1$ for every $A \in \mathbb{A}$. The fractional edge covering number is the minimum value of $\sum_{R_i: R_i \in \text{rels}(Q)} W(R_i)$ over all fractional edge coverings.

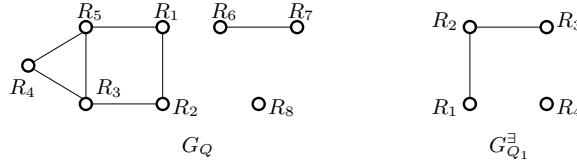


Fig. 3. The left is an illustration of $G_Q^\exists$ for $Q(A_1, A_2, A_3, A_4, A_5) : - R_1(A_1, B_1), R_2(B_1, B_2), R_3(A_2, B_2, B_3), R_4(A_2, A_3, B_4), R_5(A_1, A_2), R_6(A_4, B_5), R_7(B_5, A_5), R_8(B_6, B_7)$ with three derived subqueries $Q_1(A_1, A_2, A_3) : - R_1(A_1, B_1), R_2(B_1, B_2), R_3(A_2, B_2, B_3), R_4(A_2, A_3, B_4), R_5(A_1, A_2)$, and $Q_2(A_4, A_5) : - R_6(A_4, B_5), R_7(B_5, A_5)$ and $Q_3 : - R_8(B_6, B_7)$. The right is an illustration of $G_{Q_1}^\exists$ for Q_1 . It has two connected components $\{R_1, R_2, R_3\}, \{R_4\}$, with the corresponding dominant relation as R_5, R_4 . The two connected components containing non-output attributes $\{B_1, B_2, B_3\}, \{B_4\}$, respectively.

- We added all the missing proofs and details from the previous version. More specifically, in the new version, we included: proof of Lemma 5.17, proof of Lemma 5.15, proof of Lemma 5.16, all the details in Section 5.4, proof of Theorem 6.7, a detailed comparison between SWP and DP (Appendix B).

2 Head-Domination and Head-Cluster Properties

2.1 Notions of Connectivity

We give two important notions of connectivity for CQs, which will play an important role in characterizing the structural properties used in SWP. See an example in Figure 3.

Connectivity of CQ. We capture the *connectivity* of a CQ Q by modeling it as a graph G_Q , where each relation R_i is a vertex and there is an edge between $R_i, R_j \in \text{rels}(Q)$ if $\text{attr}(R_i) \cap \text{attr}(R_j) \neq \emptyset$. A CQ Q is *connected* if G_Q is connected, and *disconnected* otherwise. For a disconnected CQ Q , we can decompose it into multiple connected subqueries by applying search algorithms on G_Q , and finding all connected components for G_Q . The set of relations corresponding to the set of vertices in one connected component of G_Q forms a connected subquery of Q . Given a disconnected CQ Q , let $Q_1, Q_2, \dots, Q_k$ be its connected subqueries. Given a database D over Q , let $D_i \subseteq D$ be the corresponding sub-databases defined for Q_i . Observe that every witness for $Q(D)$ is the disjoint union of a witness for $Q_i(D_i)$, for $i \in [k]$. Hence, Lemma 2.1 follows. In the remainder of this article, we assume that Q is connected.

LEMMA 2.1. *For a disconnected CQ Q of k connected components $Q_1, Q_2, \dots, Q_k$, SWP is poly-time solvable for Q if and only if SWP is poly-time solvable for every Q_i , where $i \in [k]$.*

Existential-Connectivity of CQ [40]. We capture the existential-connectivity of a CQ Q by modeling it as a graph $G_Q^\exists$, where each relation R_i with $\text{attr}(R_i) - \text{head}(Q) \neq \emptyset$ is a vertex, and there is an edge between $R_i, R_j \in \text{rels}(Q)$ if $\text{attr}(R_i) \cap \text{attr}(R_j) - \text{head}(Q) \neq \emptyset$. We can find the connected components of $G_Q^\exists$ by applying a search algorithm on $G_Q^\exists$. Let $E_1, E_2, \dots, E_k \subseteq \text{rels}(Q)$ be the connected components of $G_Q^\exists$, each corresponding to a subset of relations in Q . Let $A_1, A_2, \dots, A_k$ be the output attributes appearing in relations from $E_1, E_2, \dots, E_k$ separately. Let $Q_1, Q_2, \dots, Q_k$ be the subqueries induced by $E_1, E_2, \dots, E_k$ separately.

2.2 Definition of Head-Domination and Head-Cluster Properties

These two important structural properties identified for characterizing the hardness of (A)SWP are directly built on the existential connectivity of a CQ Q and the notion of dominant relation. For a CQ Q with a subset $E \subseteq \text{rels}(Q)$ of relations, $R_i \in \text{rels}(Q)$ is a *dominant* relation for E if every output attribute appearing in any relation of E also appears in R_i , i.e., $\cup_{R_j \in E} \text{head}(R_j) \subseteq \text{head}(R_i)$.

Definition 2.2 (Head-domination Property [40]). For CQ Q , let $E_1, E_2, \dots, E_k$ be the connected components of $G_Q^\exists$. Q has head-domination property if for any $i \in [k]$, there exists a dominant relation from $\text{rels}(Q)$ for E_i .

The notion of head-domination property was first identified for DP with the side effect problem [40], which studied the smallest number of tuples to remove so that a subset of desired query results must disappear while maintaining as many remaining query results. We give a detailed comparison between SWP and DP in Appendix B, although they solve completely independent problems for CQs. As the head-cluster property implies the head-domination property, we give the following definition based on dominant relations.

Definition 2.3 (Head-cluster Property). For a CQ Q , let $E_1, \dots, E_k$ be connected components of $G_Q^\exists$. Q has head-cluster property if for any $i \in [k]$, every $R_j \in E_i$ is a dominant relation for E_i .

There is an equivalent but more straightforward definition for the head-cluster property:

LEMMA 2.4. *A CQ Q has head-cluster property if and only if for every pair of relations $R_i, R_j \in \text{rels}(Q)$ with $\text{head}(R_i) \neq \text{head}(R_j)$, there is $\text{attr}(R_i) \cap \text{attr}(R_j) \subseteq \text{head}(Q)$.*

2.3 Connection with Existing Classification of CQs

Now, we investigate how these two properties interact with the existing structural properties of CQs.

Head-cluster Property. As shown in Figure 2, every full CQ has the head-cluster property. Note that every relation is a singular connected component of $G_Q^\exists$, since every relation is assumed to have a distinct set of attributes. The head-cluster property automatically follows. Moreover, some non-full (acyclic or cyclic) CQs can also have the head-cluster property, for example, $Q(A_1) : -R_1(A_1, B_1, B_2), R_2(A_1, B_2, B_3)$ and $Q(A_1) : -R_1(A_1, B_1, B_2), R_2(A_1, B_2, B_3), R_3(A_1, B_1, B_3)$.

LEMMA 2.5. *Any full CQ has the head-cluster property.*

Head-domination Property. We find an interesting connection between the class of acyclic CQs with the head-domination property and the class of free-connex CQs [9], which has played an important role in query evaluation. Two important properties of (non) free-connex CQs have been studied:

LEMMA 2.6 ([9]). *A free-connex CQ Q has a free-connex join tree $\mathbb{T}$ such that (1) each node of $\mathbb{T}$ corresponds to $\text{attr}(R_i)$ or $\text{head}(R_i)$ for some relation $R_i \in \text{rels}(Q)$; and each relation $R_i \in \text{rels}(Q)$ corresponds to a node in $\mathbb{T}$; (2) for every attribute $A \in \text{attr}(Q)$, the set of nodes containing A form a connected subtree of $\mathbb{T}$; (3) there is a connected subtree $\mathbb{T}_{\text{con}}$ of $\mathbb{T}$ including the root of $\mathbb{T}$, such that all attributes appearing in $\mathbb{T}_{\text{con}}$ is exactly $\text{head}(Q)$.*

LEMMA 2.7 ([9]). *An acyclic but non-free-connex CQ Q contains a sequence of attributes $P = \langle A_1, A_2, \dots, A_k \rangle$ such that*

- $A_1, A_k \in \text{head}(Q)$ and $A_2, A_3, \dots, A_{k-1} \in \text{attr}(Q) - \text{head}(Q)$;
- for every $i \in [k-1]$, there exists a relation $R_j \in \text{rels}(Q)$ such that $A_i, A_{i+1} \in \text{attr}(R_j)$;
- for any relation $R_j \in \text{rels}(Q)$, either $\text{attr}(R_j) \cap P = \emptyset$, or $|\text{attr}(R_j) \cap P| = 1$, or $\text{attr}(R_j) \cap P = \{A_i, A_{i+1}\}$ for some $i \in [k-1]$.

Such a sequence of attributes is called a free path of Q .

LEMMA 2.8. *For an acyclic CQ Q , Q has the head-domination property if and only if Q is free-connex.*

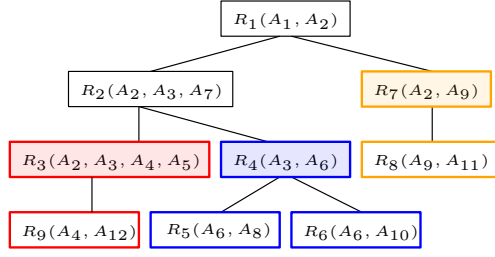


Fig. 4. An illustration of a free-connex join tree for a free-connex CQ $Q(A_1, A_2, A_3, A_7) : -R_1(A_1, A_2), R_2(A_2, A_3, A_7), R_3(A_2, A_3, A_4, A_5), R_4(A_3, A_6), R_5(A_6, A_8), R_6(A_6, A_{10}), R_7(A_2, A_9), R_8(A_9, A_{11})$. Relations $R_3, R_4, R_5, R_6, R_7, R_8, R_9$ contain non-output attributes. A partition for these relations is $\{\{R_3, R_9\}, \{R_4, R_5, R_6\}, \{R_7, R_8\}\}$ with dominants R_3, R_4, R_7 respectively.

PROOF OF LEMMA 2.8. Direction If. Given a tree structure $\mathbb{T}$ for a free-connex CQ Q as described by Lemma 2.6, every relation R_i with $\text{attr}(R_i) - \text{head}(R_i) \neq \emptyset$ and all of its ancestors fully containing output attributes, is a dominant. Suppose R_i corresponds to node u . As $\text{attr}(R_i) - \text{head}(R_i) \neq \emptyset$, $u \notin \mathbb{T}_{\text{con}}$. Consider any descendant v of u , that corresponds to a relation in Q , say R_j . Implied by the fact that $\mathbb{T}_{\text{con}}$ is a connected subtree, $v \notin \mathbb{T}_{\text{con}}$. Hence, $\text{head}(R_j) \subseteq \text{head}(R_i)$. For every relation $R_j \in \text{rels}(Q)$ with $\text{attr}(R_j) - \text{head}(Q) \neq \emptyset$, either itself is a dominant relation, or it has an ancestor as a dominant relation, since the root node fully contains output attributes. In our construction above, every relation $R_j \in \text{rels}(Q)$ with $\text{attr}(R_j) - \text{head}(Q) \neq \emptyset$ is associated with only one dominant relation.

Direction Only-If. Consider an arbitrary free path $P = \langle A_1, A_2, \dots, A_k \rangle$ in an acyclic but non-free-connex CQ Q . As for every $i \in [k - 1]$, there exists a relation $R_j \in \text{rels}(Q)$ such that $A_i, A_{i+1} \in \text{attr}(R_j)$, all relations contains at least one attribute in $A_2, A_3, \dots, A_{k-1}$ must fall into the same connected component in $G_Q^\exists$. Also, these relations contain output attributes A_1, A_k . As there exists no relation containing both A_1, A_k in Q , there is no dominant relation for this connected component, hence Q does not have the head-domination property. $\square$

An example of free-connex CQs is given in Figure 4.

2.4 Practical Implications

We have examined the 22 queries in the TPC-H benchmark [1] and focused on the 17 queries that neither involve self-joins nor complicated IN/EXISTS subqueries. Our findings are as follows:

- there are 5 queries with the head-cluster property: $Q_1, Q_4, Q_6, Q_{13}, Q_{15}$;
- there are 7 queries with the head-domination property: $Q_3, Q_{11}, Q_{12}, Q_{14}, Q_{17}, Q_{18}, Q_{19}$;
- there are 5 queries without the head-domination property: $Q_2, Q_5, Q_9, Q_{10}, Q_{16}$, where Q_{16} is a line query.

In total, 5 queries have the head-cluster property, and 12 queries have the head-domination property (including the 5 queries with the head-cluster property). For each of these 12 queries with the head-domination property—as well as for Q_{16} , which, although lacking the head-domination property, is a line query—we demonstrate that SWP can be computed exactly or approximated tightly.

3 Decomposability of Witness

In this section, we give a straightforward algorithm for computing a witness for an arbitrary CQ Q , database D , and a subset $S \subseteq Q(D)$ of query results. Our algorithm is based on the *decomposability* property of the witness stated below. Intuitively, any witness of the input CQ can be partitioned

into a group of witnesses, each for the subproblem derived by one connected component of CQ defined by existential connectivity.

THEOREM 3.1. (Decomposability of Witness) *For an arbitrary CQ Q , a database D , and a subset $S \subseteq Q(D)$ of query results, let $E_1, E_2, \dots, E_k \subseteq \text{rels}(Q)$ as the connected components of $G_Q^\exists$. For each $i \in [k]$, let Q_i be the subquery induced by E_i with output attributes A_i . For an arbitrary database D for Q and $i \in [k]$, let D_i be the sub-databases of D induced by E_i . The following properties hold:*

- *Let D' be an arbitrary witness to (Q, D, S) and $D'_i \subseteq D'$ be the subset of tuples from relations in E_i for any $i \in [k]$. Then, D'_i is also a witness to $(Q_i, D_i, \pi_{A_i} S)$ for any $i \in [k]$;*
- *Let D'_i be an arbitrary witness to $(Q_i, D_i, \pi_{A_i} S)$ for any $i \in [k]$. Then, $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is also a witness to (Q, D, S) .*

PROOF OF THEOREM 3.1. On one hand, let D' be an arbitrary witness to (Q, D, S) and $D'_i \subseteq D'$ be the subset of tuples from relations in E_i for any $i \in [k]$. By contradiction, we assume that there exists $i \in [k]$ such that $\pi_{A_i} S - Q_i(D'_i) \neq \emptyset$. Then, there must exist a tuple $t_i \in \pi_{A_i} S$ such that $t_i \notin Q_i(D'_i)$. Let $t \in S$ be an arbitrary query result such that $\pi_{A_i} t = t_i$. For an arbitrary witness $W \subseteq D'$ for t , the subset of tuples $W_i \subseteq W$ from relations in E_i forms a witness for tuple t_i . Hence, $t \notin Q(D')$ in this case, contradicting the fact that D' is a witness to (Q, D, S) .

On the other hand, let D'_i be an arbitrary witness to $(Q_i, D_i, \pi_{A_i} S)$ for any $i \in [k]$. Consider an arbitrary query result $t \in S$. Let $W_i \subseteq D'_i$ be the subset of tuples that form a witness for $\pi_{A_i} t$. Then, the disjoint union of $W_1 \uplus W_2 \uplus \dots \uplus W_k$ forms a witness of t , since (1) every tuple in $W_1 \uplus W_2 \uplus \dots \uplus W_k$ has the same value $\pi_{A_i} t$ over any output attribute A_i , if it contains attribute A_i ; and (2) no pair of tuples across W_i and W_j for $i \neq j$ can join on non-output attributes. Together, $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is also a witness to (Q, D, S) . $\square$

Algorithm. Following the decomposability property, we propose Algorithm 1 that takes an input CQ Q , a database D , and a subset $S \subseteq Q(D)$ of query results and returns a witness to (Q, D, S) . We distinguish two cases on the input CQ Q . If $G_Q^\exists$ is connected, we return the union of witnesses for every query result in S . Otherwise, we apply the decomposability property by recursively invoking the whole algorithm for each sub-query in $G_Q^\exists$ and then returning the union of their solutions as the final result.

We next show the correctness and complexity of Algorithm 1 in Lemma 3.2 and Lemma 3.3.

LEMMA 3.2. *For an arbitrary CQ Q , a database D , and a subset $S \subseteq Q(D)$ of query results, Algorithm 1 returns a valid witness to (Q, D, S) .*

PROOF OF LEMMA 3.2. Let D' be the set of tuples returned by Algorithm 1. In the base case, when $k = 1$, this trivially holds, as $\text{ESWP}(Q, D, \{t\})$ is included by D' for every query result in $t \in S$. In general, let D'_i be the set of tuples returned by line 10 of Algorithm 1. By hypothesis, we assume D'_i is a witness to $(Q_i, \{R_j : R_j \in E_i\}, \pi_{A_i} S)$. Implied by Theorem 3.1, $D' = D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is a witness to (Q, D, S) , completing the whole proof. $\square$

LEMMA 3.3. *Algorithm 1 runs in polynomial time in terms of $|D|$.*

PROOF OF LEMMA 3.3. The primitive at line 7 takes $O(1)$ time. The size of S is bounded by the size of the query result $Q(D)$, which is polynomial in terms of $|D|$. By hypothesis, assume line 10 runs in polynomial time in terms of $|D|$. All other operations take $O(1)$ time. Putting everything together, Algorithm 1 runs in polynomial time in terms of $|D|$. $\square$

ALGORITHM 1: FINDWITNESS(Q, D, S)

Input : A CQ Q , a database D and a subset $S \subseteq Q(D)$ of query result;
Output : A subset of tuples $D' \subseteq D$ such that $S \subseteq Q(D')$;

```

1  $D' \leftarrow \emptyset$ ;
2 if  $|S| = 1$ , say  $S = \{t\}$  then
3    $t' \leftarrow$  an arbitrary join results in  $\bowtie_{R_i \in \text{rels}(Q)} R_i$  such that  $\pi_{\text{head}(Q)} t' = t$ ;
4    $D' \leftarrow \{\pi_{\text{attr}(R_i)} t : R_i \in \text{rels}(Q)\}$ ;
5 else
6    $E_1, E_2, \dots, E_k \leftarrow$  connected components of  $G_Q^\exists$ ;
7    $Q_1, Q_2, \dots, Q_k \leftarrow$  sub-queries derived by relations in  $E_1, E_2, \dots, E_k$  separately;
8    $A_1, A_2, \dots, A_k \leftarrow$  output attributes appearing in relations from  $E_1, E_2, \dots, E_k$  separately;
9   if  $k = 1$  then
10    foreach  $t \in S$  do
11       $D' \leftarrow D' \uplus \text{FINDWITNESS}(Q, D, \{t\})$ ;
12  else
13    foreach  $i \in [k]$  do
14       $D' \leftarrow D' \uplus \text{FINDWITNESS}(Q_i, \{R_j : R_j \in E_i\}, \pi_{A_i} S)$ ;
15 return  $D'$ ;
```

4 Dichotomy of Exact SWP

In this section, we first establish the following dichotomy for computing SWP exactly:

THEOREM 4.1. *For a CQ Q , if it has head-cluster property, SWP is poly-time solvable; otherwise, SWP is not poly-time solvable, unless $P = NP$.*

In Section 4.1, we revisit Algorithm 1 on CQs with the head-cluster property. In Section 4.2, we prove the hardness of computing SWP exactly for CQs without the head-cluster property, by resorting to the hardness of set cover [11]: given a universe $\mathcal{U}$ of n elements and a family $\mathcal{S}$ of subsets of $\mathcal{U}$, it asks to find a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of sets whose union is $\mathcal{U}$ (called “cover”), while using the fewest sets. In Section 4.3, we then look into the fine-grained complexity of computing SWP exactly in linear time, which is determined by the *acyclicity* of the input CQ:

THEOREM 4.2. *For a CQ Q with the head-cluster property, if Q is acyclic, SWP is linear-time solvable; otherwise, SWP is not linear-time solvable, assuming the triangle detection conjecture and the hyperclique conjecture.*

4.1 A Poly-time Algorithm

We prove the first part of Theorem 4.1 with the help of Algorithm 1. To start, we show an analysis of Algorithm 1 for CQs with the head-cluster properties. Recall that the head-cluster property implies that if two relations have different output attributes, they share no common non-output attributes. Algorithm 1 will simply partition all relations based on their output attributes.

LEMMA 4.3. *For a CQ Q with head-cluster property, Algorithm 1 finds a solution to $\text{ESWP}(Q, D, S)$ for any database D and any subset of query result $S \subseteq Q(D)$ in polynomial time in terms of $|D|$.*

PROOF OF LEMMA 4.3. Let D' be the set of tuples returned by Algorithm 1. Implied by Lemma 3.2, D' is a witness to (Q, D, S) . It suffices to show that there exists no subset $D'' \subseteq D$ such that $S \subseteq Q(D'')$ and $|D''| < |D'|$. Along with Lemma 3.3, this completes the proof for Lemma 4.3.

In the base case when $k = 1$, $|D'| = |\text{rels}(Q)| \cdot |S|$. Implied by the head-cluster property, all relations share the same output attributes. More specifically, for any pair of query results $t, t' \in Q(D)$, their witnesses must be tuple-disjoint. This way, any witness of (Q, D, S) should include at least $|\text{rels}(Q)|$ distinct tuples for each query result in $Q(D)$. Hence, any solution to $\text{ESWP}(Q, D, S)$ includes at least $|\text{rels}(Q)| \cdot |S|$ tuples, i.e., D' is a solution to $\text{ESWP}(Q, D, S)$.

In the general case, let D'_i be the set of tuples returned by Algorithm 1 with input parameters $(Q_i, D_i, \pi_{A_i}S)$. By hypothesis, D'_i is a solution to $\text{ESWP}(Q_i, D_i, \pi_{A_i}S)$. Implied by Theorem 3.1, $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is a witness to (Q, D, S) . By contradiction, assume $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is not a solution to $\text{ESWP}(Q, D, S)$, i.e., there exists a subset $D'' \subseteq D$ such that $S \subseteq Q(D'')$ and $|D''| < \sum_{j \in [k]} |D'_j|$. Let $D''_1, D''_2, \dots, D''_k \subseteq D''$ be the subset of tuples from relations in $E_1, E_2, \dots, E_k$ separately. As $|D''| = \sum_{j \in [k]} |D''_j|$, there must exist some $i \in [k]$ such that $|D''_i| < |D'_i|$. Implied by Theorem 3.1, D''_i is also a witness to $(Q_i, D_i, \pi_{A_i}S)$, contradicting the fact that D'_i is a solution to $\text{ESWP}(Q_i, D_i, \pi_{A_i}S)$. Hence, $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is a solution to $\text{ESWP}(Q, D, S)$. $\square$

COROLLARY 4.4. *For a CQ Q with head-cluster property, SWP is poly-time solvable for Q .*

PROOF. Given an arbitrary database D for Q , we first compute the query result $Q(D)$. Also, the size of the query result $Q(D)$ is polynomial in terms of $|D|$. Then, we invoke Algorithm 1 with the input parameters $(Q, D, Q(D))$. From Lemma 4.3, the invocation of Algorithm 1 with input parameters $(Q, D, Q(D))$ can find a solution to $\text{SWP}(Q, D)$. As computing the query result $Q(D)$ takes polynomial time in terms of $|D|$, and combining this with 3.3, the entire procedure takes polynomial time in terms of $|D|$. $\square$

4.2 Hardness

We next prove the second part of Theorem 4.1 by showing the hardness for CQs without the head-cluster property. We start with $Q_{\text{cover}}(A) : -R_1(A, B), R_2(B)$ and then extend to any CQ without head-cluster property.

LEMMA 4.5. *SWP is not poly-time solvable for $Q_{\text{cover}}(A) : -R_1(A, B), R_2(B)$, unless $P = NP$.*

PROOF OF LEMMA 4.5. We show a reduction from set cover to SWP for Q_{cover} . Given an arbitrary instance $(\mathcal{U}, \mathcal{S})$ of set cover, we construct a database D for Q_{cover} as follows. For each element $u \in \mathcal{U}$, we add a value a_u to $\text{dom}(A)$; for each subset $S \in \mathcal{S}$, we add a value b_S to $\text{dom}(B)$, and a tuple (b_S) to R_2 . Moreover, for each pair $(u, S) \in \mathcal{U} \times \mathcal{S}$ with $u \in S$, we add a tuple (a_u, b_S) to R_1 . Note that $Q_{\text{cover}}(D) = \mathcal{U}$. Now, we can show that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\text{SWP}(Q_{\text{cover}}, D)$ has a solution D' of size $\leq |\mathcal{U}| + k$. Then, if SWP is poly-time solvable for Q_{cover} , set cover is also poly-time solvable, which is impossible unless $P = NP$. $\square$

LEMMA 4.6. *For a CQ Q without head-cluster property, SWP is not poly-time solvable for Q , unless $P = NP$.*

PROOF OF LEMMA 4.6. Consider such a CQ Q with a desired pair of relations $R_i, R_j \in \text{rels}(Q)$. We next show a reduction from Q_{cover} to Q . Given an arbitrary database D_{cover} over Q_{cover} , we construct a database D over Q as follows. From Lemma 2.4, it is always feasible to identify attribute $A' \in \text{head}(R_i) - \text{attr}(R_j)$ and attribute $B' \in \text{attr}(R_i) \cap \text{attr}(R_j) - \text{head}(Q)$. Conceptually, we would like to use A' to simulate A and B' to simulate B . All remaining attributes just contain a dummy value. More specifically, we distinguish the following four cases for each relation $R_h \in \text{rels}(Q)$:

- if $A', B' \in \text{attr}(R_h)$, for each tuple $t \in R_1$, we add a tuple t' to R_h such that $\pi_{A'}t' = \pi_{A'}t$, $\pi_{B'}t' = \pi_{B'}t$ and $\pi_Xt' = \{*\}$ for any remaining attribute $X \in \text{attr}(R_h) - \{A', B'\}$.
- if $B' \in \text{attr}(R_h)$ but $A' \notin \text{attr}(R_h)$, for each tuple $t \in R_2$, we add a tuple t' to R_h such that $\pi_{B'}t' = \pi_{B'}t$ and $\pi_Xt' = \{*\}$ for any remaining attribute $X \in \text{attr}(R_h) - \{B'\}$.

- if $A' \in \text{attr}(R_h)$ but $B' \notin \text{attr}(R_h)$, for each tuple $t \in Q_{\text{cover}}$, we add a tuple t' to R_h such that $\pi_{A'}t' = \pi_A t$ and $\pi_X t' = \{*\}$ for any remaining attribute $X \in \text{attr}(R_h) - \{A'\}$.
- if $A', B' \notin \text{attr}(R_h)$, we just add a tuple t' to R_h such that $\pi_X t' = \{*\}$ for any remaining attribute $X \in \text{attr}(R_h)$.

It can be easily checked that there is a one-to-one correspondence between solutions to $\text{SWP}(Q, D)$ and solutions to $\text{SWP}(Q_{\text{cover}}, D_{\text{cover}})$. Thus, if SWP is poly-time solvable for Q , then SWP is also poly-time solvable for Q_{cover} , coming to a contradiction of Lemma 4.5. $\square$

4.3 Fine-Grained Dichotomy of Exact SWP

So far, we only discussed the poly-time solvability of SWP. Investigating the fine-grained complexity of SWP for the class of CQs with head-cluster property is also interesting by resorting to the literature on query processing. Our main result is a dichotomy of linear-time solvability of SWP for CQs with the head-cluster property.

THEOREM 4.7. *For a CQ Q with head-cluster property, if Q is acyclic, SWP is linear-time solvable; otherwise, SWP is not linear-time solvable, assuming the triangle detection conjecture and the hyperclique conjecture.*

CONJECTURE 4.8 (TRIANGLE DETECTION CONJECTURE [3]). *Given a graph with m edges, no algorithm can decide whether a triangle exists in $O(m)$ time.*

CONJECTURE 4.9 (HYPERCLIQUE CONJECTURE [42]). *Given a k -uniform hypergraph (for $k \geq 3$), no algorithm can decide whether a hyper-clique of size $k + 1$ exists in $O(m)$ time, i.e., a set of $k + 1$ vertices where every subset of size k forms a hyperedge, where m is the number of edges in the graph.*

LEMMA 4.10 ([9, 12]). *For any cyclic CQ Q , no truly linear algorithm can detect for an arbitrary input database D whether there is a query result in $Q(D)$ or not, assuming the triangle detection conjecture and the hyperclique conjecture.*

PROOF OF THEOREM 4.7. Part If. We slightly adapt Algorithm 1 to Algorithm 2 for computing SWP over CQs with the head-cluster property. At line 1 of Algorithm 2, all dangling tuples (that do not participate in any underlying join result) are removed via semi-joins in $O(|D|)$ time [54]. The remaining tuples are non-dangling, i.e., participate in at least one underlying join result. We decompose the input CQ Q into $Q_1, Q_2, \dots, Q_k$ based on $G_Q^\exists$. As Q has the head-cluster property, all relations in each sub-query share the same output attributes. Consider an arbitrary sub-query Q_i . If relations in E_i do not have non-output attributes, E_i contains a single relation. In this case, we include all non-dangling tuples from this relation in our final witness. Otherwise, we look into the results of Q_i , essentially the projection of non-dangling tuples onto output attributes A_i . Implied by Lemma 1.8, after $O(|D|)$ preprocessing time, we can obtain the smallest witness for each query result within $O(1)$ time at line 11. As there are $O(|D|)$ query results for Q_i , this step takes $O(|D|)$ time. Algorithm 2 runs in $O(|D|)$ time in total.

Part Otherwise. Assume that there is an algorithm $\mathcal{A}$ that takes an arbitrary cyclic CQ Q with head-cluster property and a database D , and output a solution D' to $\text{SWP}(Q, D)$ in $O(|D|)$ time. Note that $D' \neq \emptyset$ if and only if $Q(D) \neq \emptyset$. Hence, this algorithm can solve the detection problem for cyclic CQs (as described in Lemma 4.10) in $O(|D|)$ time, coming to a contradiction. $\square$

Although no linear-time algorithm exists for the SWP on cyclic CQs with the head-cluster property, several results in the literature achieve greater efficiency when the cyclic CQ is full. In particular, Algorithm 2 applies directly to cyclic full CQs with the head-cluster property. Note that every relation is a singular connected component of $G_Q^\exists$. Moreover, for each $i \in [k]$, line 8 is always invoked since Q is full. Hence, the witness returned is exactly $D' = \bar{D}_1 \uplus \bar{D}_2 \uplus \dots \uplus \bar{D}_k$, i.e., the

ALGORITHM 2: FASTERFINDWITNESS(Q, D)

Input : An acyclic CQ Q with head-cluster property, a database D ;
Output: A witness D' to (Q, D) ;

```

1 Remove dangling tuples in  $D$ ;
2  $D' \leftarrow \emptyset$ ;
3  $E_1, E_2, \dots, E_k \leftarrow$  connected components of  $G_Q^\exists$ ;
4  $Q_1, Q_2, \dots, Q_k \leftarrow$  sub-queries derived by relations in  $E_1, E_2, \dots, E_k$  separately;
5  $A_1, A_2, \dots, A_k \leftarrow$  output attributes appearing in relations from  $E_1, E_2, \dots, E_k$  separately;
6  $\bar{D}_1, \bar{D}_2, \dots, \bar{D}_k \leftarrow$  non-dangling tuples from relations in  $E_1, E_2, \dots, E_k$  separately;
7 foreach  $i \in [k]$  do
8   if  $\text{attr}(Q_i) = A_i$  then  $D' \leftarrow D' \uplus \bar{D}_i$ ;
9   else
10     foreach  $t \in \pi_{A_i} \bar{D}_i$  do
11        $D' \leftarrow D' \uplus \text{FINDWITNESS}(Q_i, \bar{D}_i, \{t\})$ ;
12 return  $D'$ ;
```

set of all non-dangling tuples, thus verifying Lemma 1.9. However, it is much more expensive to identify non-dangling tuples for cyclic full CQs. For example, it takes $O(N^{\text{subw}})$ time to identify non-dangling tuples for any full CQ [4], where $\text{subw} \geq 1$ is the sub-modular width of the input query [4]. It is possible to further accelerate this result by incorporating fast matrix multiplication techniques. Let $\omega(a, b, c)$ be the smallest exponent for multiplying two rectangular matrices of sizes $n^a \cdot n^b$ and $n^b \cdot n^c$ within $O(n^{\omega(a,b,c)})$ time. For simplicity, let $\omega = \omega(1, 1, 1)$ be the exponent of fast square matrix multiplication. For example, consider the class of clique full CQs $Q_{\text{clique}}^k(A_1, A_2, \dots, A_k) : -\{R_{ij}(A_i, A_j)\}_{i \in [k], j \in [k], i \neq j}$. When $k = 3$ (a.k.a. triangle join), it takes $O(N^{\frac{2\omega}{\omega+1}})$ time to identify non-dangling tuples [5]. When $k = 4$, it takes $O(N^{\frac{\omega+1}{2}})$ time to identify non-dangling tuples [23]. When $k = 5$, it takes $O(N^{\frac{\omega}{2}+1})$ time to identify non-dangling tuples [23]. When $k \geq 6$, it takes $O(N^{\omega(\frac{1}{2}, \lceil \frac{k}{3} \rceil + \frac{1}{2}, \lceil \frac{k-1}{3} \rceil + \frac{1}{2}, \lfloor \frac{k}{3} \rfloor)})$ time to identify non-dangling tuples [27]. It is left as an interesting open question to compute SWP for cyclic full CQs with the head-cluster property more efficiently.

5 Dichotomy of Approximated SWP

As it is inherently difficult to compute SWP exactly for general CQs (without the head-cluster property), the next interesting question is to explore approximated solutions for SWP. In this section, we establish the following dichotomy for approximating SWP:

THEOREM 5.1. *If a CQ Q has head-domination property, SWP is $O(1)$ -approximatable; otherwise, SWP is not $(1 - o(1)) \cdot \log N$ -approximatable, where N is the size of the input database, unless $P = NP$.*

In Section 5.1, we show that Algorithm 1 returns an $O(1)$ -approximated solution to all CQs with the head-domination property. Next, we show the hardness of CQs without head-domination property by resorting to inapproximability of set cover [25, 30]: there is no poly-time algorithm for approximating set cover of input size n within factor $(1 - o(1)) \cdot \log n$, unless $P = NP$. We identify two hardcore structures (Definition 5.7 and Definition 5.10), and show the logarithmic inapproximability in Section 5.2 and Section 5.3, respectively. Lastly, we complete the proof of Theorem 5.1 by establishing the connection between the non-existence of the head-domination property for CQs and the existence of a hardcore.

5.1 An $O(1)$ -approximation Algorithm

Let's start by revisiting $Q_{\text{cover}}(A) : -R_1(A, B), R_2(B)$. Although it is hard to compute SWP exactly for Q_{cover} , it is surprisingly easy to approximate SWP for Q_{cover} within a factor of 2. Consider an arbitrary database D for Q_{cover} . Let D^* be a solution to $\text{SWP}(Q_{\text{cover}}, D)$. We can simply construct an approximated solution D' by picking a pair of tuples $(a, b) \in R_1, (b) \in R_2$ for every $a \in Q_{\text{cover}}(D)$. It can be shown that $|D'| \leq 2 \cdot |Q_{\text{cover}}(D)| \leq 2 \cdot |D^*|$. This is not a violation of the inapproximability of the set cover problem. If revisiting the proof of Lemma 4.5, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\text{SWP}(Q_{\text{cover}}, D)$ has a solution of size $\leq |\mathcal{U}| + k$. Because $k \leq |\mathcal{U}|$, the inapproximability of set cover does not carry over to SWP for Q . A similar argument applies to every CQ with the head-domination property.

LEMMA 5.2. *For a CQ Q with head-domination property, Algorithm 1 finds a $O(1)$ -approximated solution to $\text{ESWP}(Q, D, S)$ for any database D and any subset of query result $S \subseteq Q(D)$ in polynomial time in terms of $|D|$.*

PROOF OF LEMMA 5.2. Let D' be the set of tuples returned by Algorithm 1. Below, we will show that D' is a $|\text{rels}(Q)|$ -approximated solution to $\text{ESWP}(Q, D, S)$. As the query size is constant, we complete the whole proof with Lemma 3.3.

In the base case when $k = 1$, $|D'| = |\text{rels}(Q)| \cdot |S|$. Implied by the head-domination property, there exists a dominant $\hat{R}_1 \in \text{rels}(Q)$ such that $\text{head}(R_j) \subseteq \text{head}(\hat{R}_1)$ for every relation $R_j \in \text{rels}(Q)$. Any witness of (Q, D, S) should include one distinct tuple from $\hat{R}_1$ for each query result in $Q(D)$. Hence, any solution to $\text{ESWP}(Q, D, S)$ includes at least $|S|$ tuples, i.e., D' is a $|\text{rels}(Q)|$ -approximated solution to $\text{ESWP}(Q, D, S)$.

In the general case, let D'_i be the set of tuples returned by Algorithm 1 with input parameters $(Q_i, D_i, \pi_{A_i}S)$. By hypothesis, D'_i is a $|\text{rels}(Q_i)|$ -approximated solution to $\text{ESWP}(Q_i, D_i, \pi_{A_i}S)$. Implied by Theorem 3.1, $D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is a witness to (Q, D, S) . For any $i \in [k]$, let D_i^* be an arbitrary solution to $\text{ESWP}(Q_i, D_i, \pi_{A_i}S)$, i.e., $|D'_i| \leq |\text{rels}(Q_i)| \cdot |D_i^*|$. Moreover, $D^* = D_1^* \uplus D_2^* \uplus \dots \uplus D_k^*$ is a solution to $\text{ESWP}(Q, D, S)$. Then, we have

$$|D'| = \left| D'_1 \uplus D'_2 \uplus \dots \uplus D'_k \right| = \sum_{i \in [k]} |D'_i| \leq \max_i |\text{rels}(Q_i)| \cdot \sum_{i \in [k]} |D_i^*| \leq |\text{rels}(Q)| \cdot |D^*|.$$

Hence, $D' = D'_1 \uplus D'_2 \uplus \dots \uplus D'_k$ is a $|\text{rels}(Q)|$ -approximated solution to $\text{ESWP}(Q, D, S)$. $\square$

COROLLARY 5.3. *For a CQ Q with head-domination property, SWP is $O(1)$ -approximatable for Q .*

5.2 Logarithmic Inapproximability for Free Sequence

Let's start with the simplest acyclic but non-free-connex CQ $Q_{\text{matrix}}(A, C) : -R_1(A, B), R_2(B, C)$. We show a reduction from set cover to SWP for Q_{matrix} while preserving its logarithmic inapproximability. The essence of is the notion of *integral* witness, which is formally defined as below:

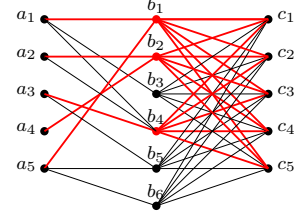
Definition 5.4 (Integral Database). For any database D over Q_{matrix} with $R_2 = (\pi_B R_2) \times (\pi_C R_2)$, a sub-database $(R'_1, R'_2) \subseteq D$ is integral if $R'_2 = (\pi_B R'_2) \times (\pi_C R'_2)$.

LEMMA 5.5. *For Q_{matrix} and a database D where $R_2 = (\pi_B R_2) \times (\pi_C R_2)$, there is an integral solution to $\text{SWP}(Q_{\text{matrix}}, D)$, i.e., a smallest witness to (Q, D) , that is, also integral.*

PROOF OF LEMMA 5.5. Consider a database D where $R_2 = (\pi_B R_2) \times (\pi_C R_2)$. Assume no dangling tuples exist in R_1, R_2 , i.e., every tuple can join with some tuple from the other relation; otherwise, we simply remove these dangling tuples. With a slight abuse of notation, we denote $A = \pi_A R_1$, $B = \pi_B R_1 (= \pi_B R_2)$ and $C = \pi_C R_2$. We consider the optimization problem O1 in Figure 5. Intuitively, it asks to assign a subset of elements $C_b \subseteq C$ to each value $b \in B$, such that each a is "connected" to

$$\begin{aligned}
(O1) \quad & \min \sum_{b \in B} |C_b| \\
& \text{s.t.} \quad \bigcup_{b: (a,b) \in R_1} C_b = C, \forall a \in A \\
& \quad C_b \subseteq C, \forall b \in B \\
(O2) \quad & \min \sum_{b \in B} x_b \\
& \text{s.t.} \quad \sum_{b: (a,b) \in R_1} x_b \geq 1, \forall a \in A \\
& \quad x_b \in \{0, 1\}, \forall b \in B
\end{aligned}$$

Fig. 5. Optimization problems in the proof of Lemma 5.5.

Fig. 6. A database D for Q_{matrix} , where each vertex represents a value of attributes and each edge (in red or in black) represents a tuple. An integral sub-database of D is highlighted in red.

all values in C via tuples in R_1, R_2 , while the total size of the assignment defined as $\sum_{b \in B} |C_b|$ is minimized. See Figure 6 for an example.

We rewrite the objective function as: $\sum_{b \in B} |C_b| = \sum_{c \in C} |\{b \in B : c \in C_b\}|$, where $\{b \in B : c \in C_b\}$ indicates the subset of values from B to which c is assigned. Together with the constraint that every $a \in A$ must be connected to c , we note that minimizing $|\{b \in B : c \in C_b\}|$ is equivalent to solving the optimization problem (O2) in Figure 5. Let x^* be the optimal solution of the program above, which only depends on the input relation R_1 , and is completely independent of the specific value c . Hence, we conclude that $\sum_{b \in B} |C_b| = |C| \cdot \sum_{b \in B} x_b^*$.

We next construct an integral sub-database D' as follows. For each $b \in B$ with $x_b^* = 1$, we add tuples in $\{(b, c) \in R_2 : \forall c \in C\}$ to D' . For each $a \in A$, we pick an arbitrary $b \in B$ with $(a, b) \in R_1$ and $x_b^* = 1$, and add (a, b) to D' . Any solution to $\text{SWP}(Q_{\text{matrix}}, D)$ must contain at least $|A|$ tuples from R_1 and at least $|C| \cdot \sum_{b \in B} x_b^*$ tuples from R_2 . Hence, D' is an integral solution to $\text{SWP}(Q_{\text{matrix}}, D)$. $\square$

LEMMA 5.6. *No poly-time algorithm approximates SWP for Q_{matrix} within a factor of $(1-o(1)) \cdot \log N$, unless $P = NP$.*

PROOF OF LEMMA 5.6. Consider an arbitrary instance of set cover $(\mathcal{U}, \mathcal{S})$, where $|\mathcal{U}| = n$ and $|\mathcal{S}| = n^c$ for some constant $c \geq 1$.³ We construct a database D for Q_{matrix} as follows. For each element $u \in \mathcal{U}$, we add a value a_u to $\text{dom}(A)$ and c_u to $\text{dom}(C)$; for each subset $S \in \mathcal{S}$, we add a value b_S to $\text{dom}(B)$. Moreover, for each pair $(u, S) \in \mathcal{U} \times \mathcal{S}$ with $u \in S$, we add tuple (a_u, b_S) to R_1 . R_2 is a Cartesian product between $\text{dom}(B)$ and $\text{dom}(C)$. Every relation contains at most n^{c+1} tuples. Hence $N \leq 2n^{c+1}$. Note that $Q_{\text{matrix}}(D)$ is the Cartesian product between A and C . Implied by Lemma 5.5, it suffices to consider an integral witness to $\text{SWP}(Q_{\text{matrix}}, D)$. Here, we show that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if the $\text{SWP}(Q_{\text{matrix}}, D)$ has an integral solution of size $\leq |\mathcal{U}| + k \cdot |V| = n(k+1)$.

Direction only-if. Suppose we are given a cover $\mathcal{S}'$ of size k to $(\mathcal{U}, \mathcal{S})$. We construct an integral solution D' to $\text{SWP}(Q_{\text{matrix}}, D)$ as follows. Let $B' \subseteq \text{dom}(B)$ be the set of values that correspond to $\mathcal{S}'$. For every $b_S \in B'$, i.e., $S \in \mathcal{S}'$, we add tuple (b_S, c_u) to D' for every $u \in \mathcal{U}$. For every $a_u \in \text{dom}(A)$, we choose an arbitrary value $b_S \in B'$ such that $u \in S$, and add tuple (a_u, b_S) to D' . This is always feasible since $\mathcal{S}'$ is a valid set cover. It can be easily checked that n tuples from R_1 and $k \cdot n$ tuples from R_2 are added to D' .

³The inapproximability of set cover holds even when the size of the family of subsets is only polynomially large with respect to the size of the universe of elements [46].

Direction if. Suppose we are given a integral solution D' of size k' to $\text{SWP}(Q_{\text{matrix}}, D)$. Let B' be the subset of values whose incident tuples in R_2 are included by D' . We argue that all subsets corresponding to B' form a valid cover of size $\frac{k'}{n-1}$. By definition of integral solution, $|B'| = \frac{k'}{n} - 1$. Moreover, for every value $a \in \text{dom}(A)$, at least one edge (a, b) is included by D' for some $b \in B'$, hence B' must be a valid set cover.

Hence, if SWP is $(1 - o(1)) \cdot \log N$ -approximable for Q_{matrix} , there is a poly-time algorithm that approximates the set cover of input size n within a $(1 - o(1)) \cdot \log N$ factor, which is impossible unless $P = NP$. $\square$

Definition 5.7 (Free Sequence). In a CQ Q , a free sequence is a sequence of attributes $P = \langle A_1, A_2, \dots, A_k \rangle$ such that⁴

- $A_1, A_k \in \text{head}(Q)$ and $A_2, A_3, \dots, A_{k-1} \in \text{attr}(Q) - \text{head}(Q)$;
- for every $i \in [k - 1]$, there exists a relation $R_j \in \text{rels}(Q)$ such that $A_i, A_{i+1} \in \text{attr}(R_j)$;
- there exists no relation $R_j \in \text{rels}(Q)$ such that $A_1, A_k \in R_j$.

LEMMA 5.8. *For a CQ Q containing a free sequence, there is no poly-time algorithm that can approximate SWP for Q within a factor of $(1 - o(1)) \cdot \log N$, unless $P = NP$.*

PROOF OF LEMMA 5.8. Let $P = \langle A_1, A_2, \dots, A_k \rangle$ be such a sequence. For simplicity, let $P' = \{A_2, A_3, \dots, A_{k-1}\}$. We next show a reduction from set cover to $\text{SWP}(Q, D)$. Consider an arbitrary instance of set cover with a universe $\mathcal{U}$ and a family $\mathcal{S}$ of subsets of $\mathcal{U}$ where $|\mathcal{U}| = n$ and $|\mathcal{S}| = n^c$ for some constant $c \geq 1$. We construct a database D for Q as follows. For each $u \in \mathcal{U}$, we add a value a_u to $\text{dom}(A_1)$ and $\text{dom}(A_k)$. For each subset $S \in \mathcal{S}$, we add a value b_S to $\text{dom}(B)$ for every $B \in P'$. We set the domain of remaining attributes in $\text{attr}(Q) - P$ as $\{*\}$. For each relation $R_j \in \text{rels}(Q)$, we distinguish the following cases:

- If $A_1 \in \text{attr}(R_j)$, we further distinguish two more cases:
 - if $\text{attr}(R_j) \cap P' = \emptyset$, we add tuple t with $\pi_{A_1} t = a_u$ for every $u \in \mathcal{U}$;
 - otherwise, we add tuple t with $\pi_{A_1} t = a_u$ and $\pi_{A_i} t = b_S$ for $A_i \in \text{attr}(R_j) \cap P'$, for every pair $(u, S) \in \mathcal{U} \times \mathcal{S}$ such that $u \in S$;
- If $A_k \in \text{attr}(R_j)$, we further distinguish two more cases:
 - if $\text{attr}(R_j) \cap P' = \emptyset$, we add tuple t with $\pi_{A_k} t = a_u$ for every $u \in \mathcal{U}$;
 - otherwise, we add tuple t with $\pi_{A_k} t = a_u$ and $\pi_{A_i} t = b_S$ for $A_i \in \text{attr}(R_j) \cap P'$, for every pair $(u, S) \in \mathcal{U} \times \mathcal{S}$;
- If $P \cap \text{attr}(R_j) \subseteq P - \{A_1, A_k\}$, we add a tuple t with $\pi_{A_i} t = b_S$ for $A_i \in \text{attr}(R_j) \cap P$, for every $S \in \mathcal{S}$;
- If $P \cap \text{attr}(R_j) = \emptyset$, then we add a tuple $\{*\}$;

It can be easily checked that every relation contains at most n^{c+1} tuples, hence $\log N = \Theta(\log n)$. The query result $Q(D)$ is exactly the Cartesian product of $\mathcal{U} \times \mathcal{U}$.

Consider a sub-database D' of D constructed above. Let R'_j be the corresponding sub-relation of R_j in D' . A solution D' to $\text{SWP}(Q, D)$ is integral if $R'_j = (\pi_{\text{attr}(R_j) \cap P'} R'_j) \times (\pi_{A_k} R'_j)$ holds for every relation R_j with $A_k \in \text{attr}(R_j)$ and $\text{attr}(R_j) \cap P' \neq \emptyset$. Applying a similar argument as Lemma 5.5, we can show that there always exists an integral solution to $\text{SWP}(Q, D)$. Below, it suffices to focus on integral solutions. It can be easily proved that $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\text{SWP}(Q, D)$ has an integral solution of size $\leq nq_1 + knq_2 + q_3$ where $q_1, q_2, q_3 \leq |\text{rels}(Q)|$ are query-dependent parameters. If SWP is $(1 - o(1)) \cdot \log N$ -approximatable for Q , there is a poly-time algorithm that

⁴Note that the notion of a free path is strictly more restrictive than that of a free sequence: every free path induces a free sequence, but not every free sequence forms a free path.

can approximate set cover instances of input size n within a $\log n$ -factor, which is impossible unless $P = NP$. $\square$

5.3 Logarithmic Inapproximability for Extended Pyramid

Although the free sequence suffices to capture the hardness of approximating SWP for acyclic CQs, it is not enough for cyclic CQs. Let's start with the simplest cyclic CQ $Q_{\text{pyramid}}(A, B, C) : -R_1(A, B), R_2(A, C), R_3(B, C), R_4(A, F), R_5(B, F), R_6(C, F)$ that does not contain a free sequence, but SWP is still difficult to approximate.

LEMMA 5.9. *No poly-time algorithm approximates SWP for Q_{pyramid} within a factor of $(1-o(1)) \cdot \log N$, unless $P = NP$.*

PROOF OF LEMMA 5.9. Consider an instance $(\mathcal{U}, \mathcal{S})$ of set cover, with $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$, where $m = n^c$ for some constant $c > 1$. We construct a database D for Q as follows. Let $\text{dom}(A) = \{a_1, a_2, \dots, a_n\}$, $\text{dom}(B) = \{b_1, b_2, \dots, b_n\}$, $\text{dom}(C) = \{c_1, c_2, \dots, c_n\}$ and $\text{dom}(F) = \text{dom}(F^-) \times \text{dom}(F^+)$, where $\text{dom}(F^-) = \{f_1^-, f_2^-, \dots, f_m^-\}$ and $\text{dom}(F^+) = \{f_1^+, f_2^+, \dots, f_n^+\}$. Relations R_1, R_2, R_3 and R_6 are Cartesian products of their corresponding attributes. For each pair $(u_\ell, S_j) \in \mathcal{U} \times \mathcal{S}$ with $u_\ell \in S_j$, we add tuples of $\{(a_\ell, f_j^-)\} \times \text{dom}(F^+)$ to R_4 . For each $i \in [n]$, we add tuples $\{b_i\} \times \text{dom}(F^-) \times \{f_i^+\}$ to R_5 . It can be easily checked that the input size of D is $O(n^{2c})$, hence $\log N = \Theta(\log n)$. $Q(D)$ is the Cartesian product between A, B and C . Hence, every solution to SWP(Q, D) includes all tuples in R_1, R_2, R_3 . Below, we focus on R_4, R_5, R_6 .

Note that D has a highly symmetric structure over $\text{dom}(B)$. More specifically, each value $b_i \in \text{dom}(B)$ induces a subquery $Q_i(A, C) = R_4(A, F_i) \bowtie R_5(F_i) \bowtie R_6(C, F_i)$, where $\text{dom}(F_i) = \text{dom}(F^-) \times \{f_i^+\}$, and a sub-database $D^i = \{R_4^i, R_5^i, R_6^i\}$, where $R_4^i = \{(a_\ell, f_j^-, f_i^+) : \forall \ell \in [n], j \in [m], u_\ell \in S_j\}$, $R_5^i = \text{dom}(F_i)$ and $R_6^i = \text{dom}(C) \times \text{dom}(F_i)$. Let D^* be a solution to SWP(Q, D). For each $i \in [n]$, let D_i^* be a solution to SWP(Q_i, D_i). It can be easily checked that $D^* = R_1 \uplus R_2 \uplus R_3 \uplus (\biguplus_{i \in [n]} D_i^*)$. For any $i \in [n]$, computing D_i^* is almost the same as Q_{matrix} . Moreover, the solution to each SWP(Q_i, D_i) shares the same structure, which is independent of the specific value $b_i \in \text{dom}(B)$. In a sub-database $D' \subseteq D$, let R'_i be the corresponding sub-relation of R_i . A solution D' to SWP(Q, D) is *integral* if $R'_4 = (\pi_{A, F^-} R'_4) \times \text{dom}(F^+)$, $R'_5 = \text{dom}(B) \times (\pi_{F^-} R'_5) \times \text{dom}(F^+)$, and $R'_6 = \text{dom}(C) \times (\pi_{F^-} R'_6) \times \text{dom}(F^+)$. Following Lemma 5.5, there always exists an integral solution to SWP(Q, D).

Moreover, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if SWP(Q, D) has an integral witness of size $\leq 3n^2 + n(n + k + kn) = (k + 4)n^2 + kn$. If SWP is $(1 - o(1)) \cdot \log N$ -approximable for Q , then there is a poly-time algorithm that can approximate set cover instances of input size n within a factor of $(1 - o(1)) \cdot \log n$, which is impossible unless $P = NP$. $\square$

Now, we are ready to introduce *extended pyramid* and the *shrink* procedure for capturing the hardness of cyclic CQs:

Definition 5.10 (Extended Pyramid). In a CQ Q , an extended pyramid is a subset of attributes $P \subseteq \text{attr}(Q)$ such that

- for any pair of attributes $A, B \in P$, there is some $R_j \in \text{rels}(Q)$ with $A, B \in \text{attr}(R_j)$;
- $P \cap \text{head}(Q) \neq \emptyset$ and $P - \text{head}(Q) \neq \emptyset$;
- there is no relation $R_j \in \text{rels}(Q)$ with $P \cap \text{head}(Q) \subseteq \text{head}(R_j)$.

Definition 5.11 (Shrink Procedure). Given the existential-connectivity graph $G_Q^\exists$ of a CQ Q with connected components $E_1, E_2, \dots, E_k$, the shrink procedure assigns one distinct attribute to all non-output attributes in the same component. The resulted CQ Q' contains the same output attributes as Q , and each $R_i \in \text{rels}(Q)$ defines a new relation $R'_i \in \text{rels}(Q')$ with $\text{attr}(R'_i) = \text{head}(R_i) \cup \{F_j : \forall j \in [k], H_j \cap \text{attr}(R_i) \neq \emptyset\}$.

As an example, recall $Q_1(A_1, A_2, A_3) : -R_1(A_1, B_1), R_2(B_1, B_2), R_3(A_2, B_2, B_3), R_4(A_2, A_3, B_4), R_5(A_1, A_2)$ in Figure 3. There are two connected components in $G_{Q_1}^\exists$, each containing non-output attributes $\{B_1, B_2, B_3\}, \{B_4\}$, respectively. The shrink procedure assigns F_1 to $\{B_1, B_2, B_3\}$ and assigns F_2 to B_4 . The resulting CQ is $Q'_1(A_1, A_2, A_3) : -R_1(A_1, F_1), R_2(F_1), R_3(A_2, F_1), R_4(A_2, A_3, F_2), R_5(A_1, A_2)$.

THEOREM 5.12. *For a CQ Q , if its shrunk counterpart Q' contains an extended pyramid, there is no poly-time algorithm that can approximate SWP for Q within a factor of $(1 - o(1)) \cdot \log N$, unless $P = NP$.*

PROOF OF THEOREM 5.12. Let $P \subseteq \text{attr}(Q)$ be the subset of attributes corresponding to the extended pyramid in the shrunk counterpart of Q . We identify the relation with the most number of output attributes of P , say $R_2 = \arg \max_{R_i \in \text{rels}(Q)} |\text{attr}(R_i) \cap P \cap \text{head}(Q)|$. As there exists no relation $R_k \in \text{rels}(Q)$ with $\text{head}(Q) \cap P \subseteq \text{attr}(R_k)$, it is always feasible to identify an attribute $A \in \text{head}(Q) \cap P - \text{attr}(R_2)$. For simplicity, we denote $P' = \text{head}(Q) \cap P - \text{attr}(R_2) - \{A\} = \{A_1, A_2, \dots, A_\ell\}$, for some integer ℓ . All non-output attributes in $P - \text{head}(Q)$ collapse to a single attribute F . All other attributes in $\text{attr}(Q) - P$ contain a dummy value $\{*\}$.

Consider an arbitrary instance of set cover $(\mathcal{U}, \mathcal{S})$ with $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$, where $m = n^c$ for some constant $c > 1$. We construct a database D for Q . Let $\text{dom}(A) = \{a_1, a_2, \dots, a_n\}$, $\text{dom}(B) = \{b_1, b_2, \dots, b_n\}$, $\text{dom}(C) = \{c_1, c_2, \dots, c_n\}$ for every $C \in P'$, and $\text{dom}(F) = \text{dom}(F^-) \times \text{dom}(F^+)$ where $\text{dom}(F^-) = \{f_1^-, f_2^-, \dots, f_m^-\}$ and $\text{dom}(F^+) = \{f_{h_1, h_2, \dots, h_\ell}^+ : \forall h_1, h_2, \dots, h_\ell \in [n]\}$. We distinguish the following cases:

- If $\text{attr}(R_i) \cap P \subseteq \text{head}(Q)$, R_i is a Cartesian product over all attributes in $\text{head}(R_i) \cap P$;
- Otherwise, we further distinguish the following three cases:
 - R_2 is a Cartesian product over all attributes in $\text{attr}(R_2) \cap P$;
 - If $A, F \in \text{attr}(R_i)$, we construct sub-relation $(\pi_{A,F} R_i)$ such that for each pair $(u_\ell, S_j) \in \mathcal{U} \times \mathcal{S}$ with $u_\ell \in S_j$, we add tuples $\{a_\ell, f_j^-\} \times \text{dom}(F^+)$ to $(\pi_{A,F} R_i)$. If $\text{attr}(R_i) \cap P' \neq \emptyset$, for each tuple $(a_\ell, f_{h_1, h_2, \dots, h_\ell}^+, f_j^-) \in (\pi_{A,F} R_i)$, we extend it by attaching value c_{h_j} for attribute $A_j \in \text{attr}(R_i) \cap P'$. This way, we already obtain $(\pi_{A,F, \text{attr}(R_i) \cap P'} R_i)$. At last, we construct R_i as the Cartesian product of remaining attributes in $\text{head}(R_i) \cap \text{head}(R_2) \cap P$ and $(\pi_{A,F, \text{attr}(R_i) \cap P'} R_i)$.
 - Otherwise, $F \in \text{attr}(R_i)$ but $A \notin \text{attr}(R_i)$. We construct sub-relation $(\pi_F R_i) = \text{dom}(F)$. If $\text{attr}(R_i) \cap P' \neq \emptyset$, for each tuple $(f_{h_1, h_2, \dots, h_\ell}^+, f_j^-) \in \pi_F R_i$, we extend it by attaching value c_{h_j} for attribute $A_j \in \text{attr}(R_i) \cap P'$. This way, we already obtain $(\pi_{F, \text{attr}(R_i) \cap P'} R_i)$. At last, we construct R_i as the Cartesian product of remaining attributes in $\text{head}(R_i) \cap \text{head}(R_2) \cap P$ and $(\pi_{F, \text{attr}(R_i) \cap P'} R_i)$.

It can be checked that every relation contains $O(n^{|\text{head}(Q) \cap P|} \cdot m)$ tuples, hence $\log N = \Theta(\log n)$. Meanwhile, the query result $Q(D)$ is the Cartesian product over attributes in $\text{head}(Q) \cap P$, so every solution to $\text{SWP}(Q, D)$ must contain all tuples in R_i if $\text{attr}(R_i) \cap P \subseteq \text{head}(Q)$, and the dummy tuple $\{*\}$ in every relation R_i if $\text{attr}(R_i) \cap P = \emptyset$.

Here, D also enjoys a highly symmetric structure over every attribute $C \in P'$. More specifically, every tuple $t = (c_{i_1}, c_{i_2}, \dots, c_{i_\ell}) \in \times_{C \in P'} \text{dom}(C)$ induces a subquery by removing all attributes in P' , and restricting $\text{dom}(F)$ as $\text{dom}(F_t) = \text{dom}(F^-) \times \{f_{i_1}^+, f_{i_2}^+, \dots, f_{i_\ell}^+\}$. In a sub-database $D' \subseteq D$, let R'_i be the corresponding sub-relation R_i . A solution D' to $\text{SWP}(Q, D)$ is *integral* if:

- for every relation $R_i \in \text{rels}(Q)$ with $A, F \in \text{attr}(R_i)$,

$$\pi_{A,F, \text{head}(R_i) \cap \text{head}(R_2) \cap P} R'_i = (\pi_{A,F} R'_i) \times \text{dom}(F^+) \times \left(\bigotimes_{C \in \text{head}(R_i) \cap \text{head}(R_2) \cap P} \text{dom}(C) \right)$$

- for every relation $R_i \in \text{rels}(Q)$ with $A \notin \text{attr}(R_i)$ and $F \in \text{attr}(R_i)$,

$$\pi_{F, \text{head}(R_i) \cap \text{head}(R_2) \cap P} R'_i = (\pi_F R'_i) \times \text{dom}(F^+) \times \left(\bigtimes_{C \in \text{head}(R_i) \cap \text{head}(R_2) \cap P} \text{dom}(C) \right)$$

It can be easily shown that there always exists an integral solution to $\text{SWP}(Q, D)$. Moreover, $(\mathcal{U}, \mathcal{S})$ has a cover of size $\leq k$ if and only if $\text{SWP}(Q, D)$ has an integral solution of size $(q_1 k + q_2) \cdot n^{|\text{head}(Q) \cap P| - 1}$, where $q_1, q_2 \leq |\text{rels}(Q)|$ are some query-dependent parameters. Applying a similar argument as Lemma 5.5, we can show that if SWP is $(1 - o(1)) \cdot \log N$ -approximatable for Q , there is a poly-time algorithm that can approximate set cover instances of input size n within a $\log n$ -factor, which is impossible unless $P = NP$. $\square$

5.4 Logarithmic Inapproximability for CQs without Head-domination Property

At last, we complete the proof of Theorem 5.1 by establishing the connection between the non-existence of the head-domination property and the existence of a hardness gadget. As this proof is quite technical, we first introduce two structures in cyclic CQs and prove two helper lemmas below:

Definition 5.13 (Cycle). In a CQ Q , a sequence of attributes $P = \langle A_1, A_2, \dots, A_k \rangle$ is a cycle such that

- for every $i \in [k]$, there exists a relation $R_j \in \text{rels}(Q)$ such that $A_i, A_{(i+1) \bmod k} \in \text{attr}(R_j)$;
- for every $R_j \in \text{rels}(Q)$, either $\text{attr}(R_j) \cap P = \emptyset$, or $|\text{attr}(R_j) \cap P| = 1$ or $\text{attr}(R_j) \cap P = \{A_i, A_{(i+1) \bmod k}\}$ for some $i \in [k]$.

Definition 5.14 (Non-conformal Clique). In a CQ Q , a non-conformal clique is a subset of attributes $P \subseteq \text{attr}(Q)$ such that

- for every pair of attributes A_i, A_j , there exists a relation $R_k \in \text{rels}(Q)$ with $A_i, A_j \in \text{attr}(R_k)$;
- there is no relation $R_\ell \in \text{rels}(Q)$ such that $P \subseteq \text{attr}(R_\ell)$.

LEMMA 5.15. *In a cyclic CQ Q without free sequences and extended pyramids in its shrunk counterpart, for any cycle or non-conformal clique P , if $P \cap \text{head}(Q) \neq \emptyset$ and $P - \text{head}(Q) \neq \emptyset$, there must exist a relation $R_k \in \text{rels}(Q)$ such that removing $\text{head}(R_k)$ turns Q into a disconnected CQ, where $\{R_i : \text{attr}(R_i) \cap P - \text{head}(Q) \neq \emptyset\}$ is a connected subquery.*

PROOF OF LEMMA 5.15. Consider an arbitrary non-output attribute $A \in \text{attr}(Q) - \text{head}(Q)$. Let $V_A \subseteq \text{attr}(Q)$ be the set of attributes shrunk to A . Let $N_A = \{B \in \text{attr}(Q) - V_A : \exists B' \in V_A, R_i \in \text{attr}(Q), \text{ s.t. } B, B' \in \text{attr}(R_i)\}$ be the set of attributes that appear together with some attribute of V_A in some relation. Note that $N_A \subseteq \text{head}(A)$; otherwise, any attribute in $N_A - \text{head}(A)$ will be shrunk as A , coming to a contradiction. Moreover, for every pair of attributes $B, B' \in N_A$, there must exist a relation R_i such that $B, B' \in \text{attr}(R_i)$; otherwise, a free sequence is identified, coming to a contradiction. Thus, N_A is a clique. In addition, there must exist one relation R_k such that $N_A \subseteq \text{attr}(R_k)$; otherwise, an extended pyramid $N_A \cup \{A\}$ is identified in the shrunk query, coming to a contradiction. In this way, removing all attributes in V_A turns Q into a disconnected CQ, where $\{R_i \in \text{rels}(Q) : \text{head}(R_i) \subseteq V_A\}$ is a connected subquery.

Let's come to cycle P . For simplicity, denote $P = \langle A_1, A_2, \dots, A_k \rangle$. As there is no free sequence, $P \cap \text{head}(Q) = \{A_i, A_{(i+1) \bmod k}\}$ for some $i \in [k]$. Then, we can identify an arbitrary output attribute in $P - \text{head}(Q)$ as A , and therefore

- $P - \text{head}(Q) \subseteq V_A$;
- for every relation $R_j \in \text{rels}(Q)$ with $\text{attr}(R_j) \cap P - \text{head}(Q) \neq \emptyset$, $\text{head}(R_j) \subseteq V_A$.

Removing $\text{head}(R_k)$ makes Q into disconnected, where $\{R_i : \text{attr}(R_i) \cap P - \text{head}(Q) \neq \emptyset\}$ is a connected subquery.

We next turn to non-conformal clique P and identify an arbitrary output attribute in $P - \text{head}(Q)$ as A , and therefore

- $P - \text{head}(Q) \subseteq V_A$;
- for every relation $R_j \in \text{rels}(Q)$ with $\text{attr}(R_j) \cap P - \text{head}(Q) \neq \emptyset$, $\text{head}(R_j) \subseteq V_A$.

Removing $\text{head}(R_k)$ makes Q disconnected, where $\{R_i : \text{attr}(R_i) \cap P - \text{head}(Q) \neq \emptyset\}$ is a connected subquery. $\square$

LEMMA 5.16. *In a cyclic CQ Q without a free sequence or an extended pyramid in its shrunk counterpart, if every cycle or non-conformal clique contains only output attributes or only non-output attributes, Q has the head-domination property.*

PROOF OF LEMMA 5.16. We apply the following procedure to Q : if there exists a cycle or non-conformal clique P , we add a *virtual* relation R_p with $\text{attr}(R_p) = P$ to $\text{rels}(Q)$. We apply this procedure iteratively until no more cycle or non-conformal cliques can be found. Let Q' be the resulting query.

We first prove that for each virtual relation R_p added, either $\text{attr}(R_p) \subseteq \text{head}(Q)$ or $\text{attr}(R_p) \subseteq \text{attr}(Q) - \text{head}(Q)$. Adding a virtual relation does not increase the number of cycles, but some possible non-conforming cliques. We distinguish the following three cases:

- If R_p is added for a cycle P , P must exist in Q and therefore $\text{attr}(R_p) \subseteq \text{head}(Q)$ or $\text{attr}(R_p) \subseteq \text{attr}(Q) - \text{head}(Q)$;
- If R_p is added for a non-conformal clique P , and P exists in Q , then $\text{attr}(R_p) \subseteq \text{head}(Q)$ or $\text{attr}(R_p) \subseteq \text{attr}(Q) - \text{head}(Q)$;
- Otherwise, R_p is added for a non-conformal clique P but P does not exist in Q . It must be the case that after R_p is added for a cycle P' , a non-conformal clique P forms with $P' \subsetneq P$. If $P' \subseteq \text{head}(Q)$, then $P \subseteq \text{head}(Q)$; otherwise, P is an extended pyramid in the shrunk query of Q . If $P' \subseteq \text{attr}(Q) - \text{head}(Q)$, then $P \subseteq \text{attr}(Q) - \text{head}(Q)$; otherwise, any pair of non-consecutive attributes in P' will form a free sequence together with an arbitrary attribute in $P \cap \text{head}(Q)$, contradicting the fact that Q does not contain a free sequence. In either case, R_p is added as a virtual relation with $\text{attr}(R_p) \subseteq \text{head}(Q)$ or $\text{attr}(R_p) \subseteq \text{attr}(Q) - \text{head}(Q)$.

We point out some observations on Q' : (1) $\text{head}(Q) = \text{head}(Q')$; (2) Q' is acyclic, as it does not contain any cycle or non-conformal clique; and (3) Q' also does not contain any free sequence as Q . Together, Q' is free-connex. It is feasible to find a free-connex join tree, and identify connected components $E'_1, E'_2, \dots, E'_k$ of $G_{Q'}^\exists$ with dominants $\hat{R}_1, \hat{R}_2, \dots, \hat{R}_k \in \text{rels}(Q')$. Moreover, as Q' is connected, $\text{head}(\hat{R}_i) \neq \emptyset$ holds for every $i \in [k]$. We next show that virtual relations are not dominant relations. By contradiction, we assume $R_p = \hat{R}_1$ for some virtual relation R_p . If $\text{attr}(R_p) - \text{head}(Q) = \emptyset$, $E'_1 \cup \{R_p\}$ form a connected subquery, so Q' is disconnected, leading to a contradiction. If $\text{attr}(R_p) \subseteq \text{head}(Q)$, we further distinguish two more cases. If R_p is added for a non-conformal clique, then E'_1 forms a nested-clique in the shrunk counterpart of Q ; and if R_p is added for a cycle, then any non-consecutive attributes of P together with non-output attributes in relations of E'_1 form a free sequence. Hence, virtual relations cannot be dominants, and $\hat{R}_1, \hat{R}_2, \dots, \hat{R}_k \in \text{rels}(Q)$. Let $E_i \subseteq E'_i$ be the set of non-virtual relations in E'_i . It can be easily checked that $E_1, E_2, \dots, E_k$ are also the connected components of $G_Q^\exists$ with dominants $\hat{R}_1, \hat{R}_2, \dots, \hat{R}_k$. Hence, Q has the head-domination property. $\square$

Finally, we are ready to prove the equivalence between the non-existence of the head-domination property and the existence of a hardness gadget:

LEMMA 5.17. *If a CQ Q does not have the head-domination property, there must exist a free sequence or an extended pyramid in its shrunk counterpart.*

PROOF OF LEMMA 5.17. We distinguish the input CQ into two cases. From Theorem 2.8, if Q is acyclic but does not have the head-domination property, Q must be non-free-connex, and hence Q contains a free path, that is, also a free sequence. In the remaining, it suffices to focus on cyclic CQs.

We note that every cyclic CQ contains a *cycle* or a *non-conformal clique* [13]. Consider an arbitrary cycle or clique P . As shown in Lemma 5.15, if $P \cap \text{head}(Q) \neq \emptyset$ and $P - \text{head}(Q) \neq \emptyset$, there must exist a relation $R_k \in \text{rels}(Q)$ such that removing $\text{head}(R_k)$ turns Q into a disconnected CQ, where relations in $\{R_i : \text{attr}(R_i) \cap P - \text{head}(Q) \neq \emptyset\}$ is a connected subquery. Then, it is safe to remove all relations in $\{R_i : \text{attr}(R_i) \cap P - \text{head}(Q) \neq \emptyset\}$ from Q , and show that the residual query (where P disappears) has the head-domination property. Applying this removal procedure iteratively, we are left with cycles and non-conformal cliques that contain only output attributes or only non-output attributes. Lemma 5.16 shows that Q must have the head-domination property in this case, thus completing the whole proof. $\square$

6 Approximation Algorithms for General CQs

In this section, we explore approximation algorithms for general CQs without the head-domination property. All missing proofs are given in Appendix D.

6.1 Revisit Algorithm 1 for General CQs

We give an analysis of Algorithm 1 on computing the SWP for general CQs:

THEOREM 6.1. *For a CQ Q , Algorithm 1 finds a $O(|D|^{1-1/\rho^*})$ -approximated solution to $\text{ESWP}(Q, D, S)$ for any database D and subset of query result $S \subseteq Q(D)$, where ρ^* is the fractional edge covering number of Q .*

PROOF OF THEOREM 6.1. We first show that Algorithm 1 returns a solution of size at most $\max\{|D|, |\text{rels}(Q)| \cdot |S|\}$. Let D' be the tuples returned by Algorithm 1. In the base case, when $k = 1$, this trivially holds. In the general case, let D'_i be the set of tuples returned by Algorithm 1 with input parameters $(Q_i, D_i, \pi_{A_i} S)$. By hypothesis, $|D'_i| \leq \max\{|D_i|, |\text{rels}(Q_i)| \cdot |\pi_{A_i} S|\}$. Note that $|D'| = \sum_{i \in [k]} |D'_i| \leq \sum_{i \in [k]} |D_i| = |D|$. Moreover,

$$|D'| = \sum_{i \in [k]} |\text{rels}(Q_i)| \cdot |\pi_{A_i} S| \leq \sum_{i \in [k]} |\text{rels}(Q_i)| \cdot |S| = |\text{rels}(Q)| \cdot |S|.$$

So, we have $|D'| \leq \max\{|D|, |\text{rels}(Q)| \cdot |S|\}$. Meanwhile, the AGM bound [8] implies that at least $O(|S|^{1/\rho^*})$ tuples are needed to reproduce S results, where ρ^* is the fractional edge covering number of Q .

To show the approximation ratio of D' , we distinguish two cases: (1) if $|S| \leq |D|$, the approximation ratio (skipping $|\text{rels}(Q)|$) can be bounded as $\frac{|D'|}{|S|^{1/\rho^*}} \leq |S|^{1-1/\rho^*} \leq |D|^{1-1/\rho^*}$; and (2) if $|S| > |D|$, the approximation ratio (skipping $|\text{rels}(Q)|$) can be bounded as $\frac{|D'|}{|S|^{1/\rho^*}} \leq \frac{|D|}{|D|^{1/\rho^*}} = |D|^{1-1/\rho^*}$. Hence, D' is a $|D|^{1-1/\rho^*}$ -approximated solution to $\text{ESWP}(Q, D, S)$ for any subset of query result $S \subseteq Q(D)$. $\square$

COROLLARY 6.2. *For a CQ Q , a $O(|D|^{1-1/\rho^*})$ -approximated solution to $\text{SWP}(Q, D)$ for any database D can be computed in polynomial time in terms of $|D|$, where ρ^* is the fractional edge covering number of Q .*

This upper bound is polynomially larger than the logarithmic lower bound proved in Section 5. Next, we will explore better approximation algorithms for some commonly used CQs.

ALGORITHM 3: MATRIXFINDWITNESS(Q_{matrix}, D)

Input : A database D for Q_{matrix} ;
Output : A witness D' to (Q_{matrix}, D) ;

```

1  $D' \leftarrow \emptyset, C \leftarrow \emptyset;$ 
2 foreach  $b_i \in \text{dom}(B)$  do
3    $(X_i, Y_i) \leftarrow (\sigma_{B=b_i} R_1, \sigma_{B=b_i} R_2);$ 
4 while  $C \neq Q_{\text{matrix}}(D)$  do
5    $b_j \leftarrow \arg \min_{b_i \in \text{dom}(B)} f(C, X_i, Y_i);$ 
6    $D' \leftarrow D' \cup X_j \cup Y_j, C \leftarrow C \cup \pi_{A,C}(X_j \bowtie Y_j);$ 
7   foreach  $b_i \in \text{dom}(B)$  do
8      $(X_i, Y_i) \leftarrow \arg \min_{X \subseteq \sigma_{B=b_i} R_1, Y \subseteq \sigma_{B=b_i} R_2} f(C, X, Y);$ 
9 return  $D';$ 

```

6.2 Star CQs

We look into one commonly-used class of CQs noted as *star* CQs:

$$Q_{\text{star}}(A_1, A_2, \dots, A_m) : -R_1(A_1, B), R_2(A_2, B), \dots, R_m(A_m, B).$$

Our approximation algorithm follows the greedy strategy developed for the weighted set cover problem, where each element to be covered is a query result, and each subset is a collection of tuples. Intuitively, the greedy strategy always picks a subset that minimizes the “price” for covering the remaining uncovered elements. The question boils down to specifying the universe $\mathcal{U}$ of elements and the family $\mathcal{S}$ of subsets and their weights. However, naively taking every possible sub-collection of tuples from the input database as a subset would generate an exponentially large $\mathcal{S}$, which leads to a greedy algorithm running in exponential time. Hence, keeping the size of $\mathcal{S}$ small is critical.

Greedy Algorithm for Q_{matrix} . Given Q_{matrix} , a database D , and a subset of query results $C \subseteq Q_{\text{matrix}}(D)$, the price of a collections of tuples (X, Y) for $X \subseteq R_1$ and $Y \subseteq R_2$ is defined as $f(C, X, Y) = \frac{|X|+|Y|}{|(\pi_{A,C}(X \bowtie Y)) - C|}$. To shrink the space of candidate subsets, a critical observation is that the subset with minimum price chosen by the greedy algorithm cannot be *divided* further, i.e., all tuples should have the same join value as captured by Lemma 6.3.

LEMMA 6.3. *Given an arbitrary database D for Q_{matrix} , a pair of distinct values $b, b' \in \text{dom}(B)$, and two pairs of subsets of tuples $(X_1, Y_1) \in 2^{\sigma_{B=b} R_1} \times 2^{\sigma_{B=b} R_2}$, $(X_2, Y_2) \in 2^{\sigma_{B=b'} R_1} \times 2^{\sigma_{B=b'} R_2}$, for an arbitrary $C \subseteq Q_{\text{matrix}}(D)$, we have $\min\{f(C, X_1, Y_1), f(C, X_2, Y_2)\} \leq f(C, X_1 \cup X_2, Y_1 \cup Y_2)$.*

We are ready to present a greedy algorithm that runs in polynomial time. As shown in Algorithm 3, we always maintain a pair (X_i, Y_i) for every value $b_i \in \text{dom}(B)$, such that (X_i, Y_i) minimize the function $f(C, X, Y)$ for $X \subseteq \sigma_{B=b_i} R_1$ and $Y \subseteq \sigma_{B=b_i} R_2$. The greedy algorithm always chooses the one pair with the minimum price, picks all related tuples in this pair, and updates the coverage C . After this step, we must update the candidate pair (X_i, Y_i) for each value $b_i \in \text{dom}(B)$ and enter into the next iteration. We will stop until all query results are covered.

The remaining question is how to compute (X_i, Y_i) with minimum price (line 8) efficiently. We mention a problem known as *densest subgraph in the bipartite graph* in the literature [35, 39], for which a poly-time algorithm based on max-flow has been proposed.

Definition 6.4 (Densest Subgraph in Bipartite Graph). Given a bipartite graph (X, Y, E) with $E : X \times Y \rightarrow \{0, 1\}$, it asks to find $X' \subseteq X, Y' \subseteq Y$ to maximize $\frac{\sum_{x \in X', y \in Y'} E(x, y)}{|X'| + |Y'|}$.

Back to our problem, each value $b_i \in \text{dom}(B)$ induces a bipartite graph with $X = \sigma_{B=b_i} R_1$, $Y = \sigma_{B=b_i} R_2$, and $E = \{(x, y) \mid x \in X, y \in Y, \pi_{A,C}(x \bowtie y) \notin C\}$. This way, the densest subgraph in this bipartite graph corresponds to a subset of uncovered query results with minimum price, and vice versa.

The approximation ratio of our greedy algorithm follows the standard analysis of weighted set cover [53]. We next focus on the time complexity. The while-loop proceeds in at most $O(|Q(D)|)$ iterations since $|C|$ increases by at least 1 in every iteration. It takes $O(|D|)$ time to find the pair with the minimum price since there are $O(|D|)$ distinct values in $\text{dom}(B)$. Moreover, it takes polynomial time to update (X_i, Y_i) for each $b_i \in \text{dom}(B)$. Overall, Algorithm 3 runs in polynomial time in terms of $|D|$.

Extensions. Our algorithm for Q_{matrix} can be extended to star CQs, and further to all CQs with only one non-output attribute. A similar property as Lemma 6.3 holds. Our greedy algorithm needs a generalized primitive, noted as *densest subgraph in hypergraph*⁵ [37] for finding the “set” with the smallest price. Following a similar analysis, we obtain:

THEOREM 6.5. *For a CQ Q with $|\text{attr}(Q) - \text{head}(Q)| = 1$, $\text{SWP}(Q, D)$ is $O(\log |D|)$ -approximable for any database D .*

6.3 Line CQs

We turn to another commonly-used class of CQs noted as *line* CQs:

$$Q_{\text{line}}(A_1, A_{m+1}) : -R_1(A_1, A_2), R_2(A_2, A_3), \dots, R_m(A_m, A_{m+1}),$$

with $m \geq 3$. We surprisingly find that SWP for line CQs is closely related to DSF problem [17, 24, 26, 31] in the network design: Given an edge-weighted directed graph $G = (V, E)$ of $|V| = n$ and a set of k demand pairs $\{(s_i, t_i) \in V \times V : i \in [k]\}$ and the goal is to find a subgraph G' of G with minimum weight such that there is a path in G' from s_i to t_i for every $i \in [k]$. Observe that $\text{SWP}(Q_{\text{line}}, D)$ is a special case of DSF. This way, all existing algorithms proposed for DSF can be applied to SWP for Q_{line} . The best approximation ratio achieved is $O(\min\{k^{\frac{1}{2}+o(1)}, n^{0.5778}\})$ [2, 17, 31]. Combining the baseline in Section 6.1 (with $\rho^* = 2$ for line CQs) and existing algorithms for DSF, we obtain:

THEOREM 6.6. *For Q_{line} and any database D , there is a poly-time algorithm that can approximate $\text{SWP}(Q_{\text{line}}, D)$ within a factor of $O(\min\{|Q_{\text{line}}(D)|^{\frac{1}{2}+o(1)}, \text{dom}^{0.5778}, |D|^{\frac{1}{2}}\})$, where dom is the number of values that participates in at least one full join result.*

There is a large body of works investigating the lower bounds of DSF, and we refer interested readers to [24] for details. We mention a polynomial lower bound $\Omega(k^{1/4-o(1)})$ for DSF, but it cannot be applied to SWP, since SWP is a special case with its complexity measured by the number of edges in the graph. Instead, we built a reduction from *label cover* to SWP for Q_{line} directly (which is an adaption of reduction proposed in [19, 26]) and prove the following:

THEOREM 6.7. *No poly-time algorithm approximates SWP on an arbitrary database D for Q_{line} within $\Omega(2^{(\log |D|)^{1-\epsilon}})$ factor for any constant $\epsilon > 0$, unless $P = NP$.*

7 Related Work

Factorized Database [47]. Factorized database studies nested representations of query results that can be exponentially more succinct than flat query results, which has the same goal as SWP.

⁵Given a hypergraph $H = (V, E)$ for $E \subseteq 2^V$, it asks to find a subset of nodes $S \subseteq V$ such that the ratio $|\{e \in E : e \subseteq S\}|/|S|$ is maximized.

They built tree-based representations by exploiting the distributivity of the product over the union and the commutativity of the product and union. This notion is quite different from SWP. They measure the regularity of factorizations by readability, the minimum over all its representations of the maximum number of occurrences of any tuple in that representation, while SWP measures the size of the witness. More significantly, they preserve the provenance instead of all query results as SWP, hence the hardness of factorization does not come from non-output attributes.

Data Synopses [21, 50]. In approximate query processing, people studied a lossy, compact synopsis of the data such that queries can be efficiently and approximately executed against the synopsis rather than the entire dataset, such as random samples, sketches, histograms, and wavelets. These data synopses differ in terms of what class of queries can be approximately answered, space usage, accuracy and so on. In computational geometry, a corset is a small set of points that approximate the shape of a larger point set, such as for shape-fitting, density estimation, high-dimensional vectors or points, clustering, graphs, Fourier transforms and so on. SWP can also be viewed as data synopsis, since it also selects a representative subset of tuples. But, it is more query-dependent, as different CQs over the same database (such as $Q_1(x) : -R_1(x, y), R_2(y, z)$ and $Q_2(x, y, z) : -R_1(x, y), R_2(y, z)$) can have dramatically different SWP. Furthermore, all query results must be preserved by SWP, but this is never guaranteed in other data synopses. There is no space-accuracy tradeoff in SWP as well.

Related to Other Problems in Database Theory. The SWP problem also outputs the smallest provenance that can explain why all query results are correct. This notion of why-provenance has been extensively investigated in the literature [6, 15, 36], but not from the perspective of minimizing the size of the witness. Very recently, [43] studied how to find the minimal-size factorization of the provenance that minimizes the number of variable repetitions. The SWP problem is also related to the resilience problem [32, 33, 51], which intuitively finds the smallest number of tuples to remove so that the query answer turns into false. Our SWP problem essentially finds the maximum number of tuples to remove while the query answer does not change. We can observe a clear connection here, but their solutions do not imply anything to each other. Very recently, [44] proposed a unified framework for the generalized DP problem as well as some related problems. In particular, their framework recovers the dichotomy for computing SWP exactly in polynomial time (Theorem 4.1). However, it cannot deal with approximating SWP in polynomial time or even the hardness gadgets. Moreover, their framework can also encode the ESWP problem, but does not directly give tractability results, which is left open. People also explore the *synthetic smallest witness problem* [28]. Instead of giving the database as input, this synthetic problem asks for a given CQ and a set of tuples S , how to compute a synthetic witness D such that $Q(D) = S$ while there does not exist another database D' with $Q(D') = S$ and $|D'| < D$. Although these two problems share similar definitions, their results and techniques are not related to each other.

8 Conclusion

In this article, we study the data complexity of SWP problem for CQs without self-joins. There are several interesting problems left:

- (1) *Approximating SWP for CQs without head-domination property.* So far, the approximation of SWP is well-understood only on some specific class of CQs without the head-domination property. For the large remaining class of CQs, both upper and lower bounds remain to be improved, which may lead to fundamental breakthroughs for other related problems, such as DSF.

- (2) *SWP for CQs with self-joins*. It becomes much more challenging when self-joins exist, as one tuple appears in multiple logical copies of the input relation. Similar observation has been made for the related resilience problem [33, 40].
- (3) *Aggregated SWP for CQs*. It is possible to explore the aggregated version of SWP, i.e., given a CQ Q , database D , and parameter k , it asks to find a sub-database D' such that $|Q(D')| \geq k$ while there does not exist another sub-database D'' with $|Q(D'')| \geq k$ and $|D''| < |D'|$. Note that this aggregated version is different from ESWP, in that it does not require specifying which particular query answers to recover, only that at least k answers be obtained. This problem generalizes SWP as a special case when $k = |Q(D)|$. This formulation is closely related to the *partial set cover* problem, for which a variety of approximation algorithms have been studied [34].

References

- [1] TPC-H. Retrieved from <https://www.tpc.org/tpch/>. (TPC-H).
- [2] Amir Abboud and Greg Bodwin. 2018. Reachability preservers: New extremal bounds and approximation algorithms. In *SODA*. SIAM, 1865–1883.
- [3] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 434–443.
- [4] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. FAQ: Questions asked frequently. In *PODS*. 13–28.
- [5] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-coding. *Journal of the ACM (JACM)* 42, 4 (1995), 844–856.
- [6] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for aggregate queries. In *PODS*. 153–164.
- [7] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size bounds and query plans for relational joins. In *FOCS (FOCS '08)*. 739–748.
- [8] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size bounds and query plans for relational joins. In *FOCS*. IEEE, 739–748.
- [9] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. 2007. On acyclic conjunctive queries and constant delay enumeration. In *CSL*. Springer, 208–222.
- [10] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. 1983. On the desirability of acyclic database schemes. *JACM* 30, 3 (1983), 479–513.
- [11] Bernhard Korte and Jens Vygen. 2008. *Combinatorial Optimization: Theory and Algorithms*. Springer.
- [12] Johann Brault-Baron. 2013. *On the Relevance of énumération: Complexité en Propositional and First Order Logics*. Ph.D. Dissertation. University of Caen.
- [13] Johann Brault-Baron. 2016. Hypergraph acyclicity revisited. *CSUR* 49, 3 (2016), 1–26.
- [14] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. 2002. On propagation of deletions and annotations through views. In *PODS*. 150–158.
- [15] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *ICDT*. Springer, 316–330.
- [16] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 1999. On random sampling over joins. *ACM SIGMOD Record* 28, 2 (1999), 263–274.
- [17] Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. 2011. Set connectivity problems in undirected graphs and the directed steiner network problem. *TALG* 7, 2 (2011), 1–17.
- [18] Yu Chen and Ke Yi. 2020. Random sampling and size estimation over cyclic joins. In *ICDT*.
- [19] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. 2021. Parameterized approximation algorithms for bidirected steiner network problems. *ACM Trans. Algorithms* 17, 2, Article 12 (apr 2021), 68 pages.
- [20] Gao Cong, Wenfei Fan, and Floris Geerts. 2006. Annotation propagation revisited for key preserving views. In *CIKM*. 632–641.
- [21] Graham Cormode, Minos Garofalakis, Peter J. Haas, Chris Jermaine. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases* 4, 1–3 (2011), 1–294.
- [22] Graham Cormode and Ke Yi. 2020. *Small Summaries for Big Data*. Cambridge University Press.
- [23] Mina Dalirrooyfard, Surya Mathialagan, Virginia Vassilevska Williams, and Yinzhan Xu. 2024. Towards optimal output-sensitive clique listing or: Listing cliques from smaller cliques. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 923–934.
- [24] Irit Dinur and Pasin Manurangsi. 2018. ETH-hardness of approximating 2-CSPs and directed steiner network. In *ITCS (LIPICs)*, Anna R. Karlin (Ed.), Vol. 94. 36:1–36:20.
- [25] Irit Dinur and David Steurer. 2014. Analytical approach to parallel repetition. In *STOC*. 624–633.

- [26] Yevgeniy Dodis and Sanjeev Khanna. 1999. Design networks with bounded pairwise distance. In *STOC*. 750–759.
- [27] Friedrich Eisenbrand and Fabrizio Grandoni. 2004. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science* 326, 1-3 (2004), 57–67.
- [28] Aryan Esmailpour, Boris Glavic, Xiao Hu, and Stavros Sintos. 2025. Smallest synthetic witnesses for conjunctive queries. *Proceedings of the ACM on Management of Data* 3, 2 (2025), 1–25.
- [29] R. Fagin. 1983. Degrees of acyclicity for hypergraphs and relational database schemes. *JACM* 30, 3 (1983), 514–550.
- [30] Uriel Feige. 1998. A threshold of $\ln n$ for approximating set cover. *JACM* 45, 4 (1998), 634–652.
- [31] Moran Feldman, Guy Kortsarz, and Zeev Nutov. 2012. Improved approximation algorithms for directed steiner forest. *JCSS* 78, 1 (2012), 279–292.
- [32] Cibebe Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. 2015. The complexity of resilience and responsibility for self-join-free conjunctive queries. *PVLDB* 9, 3 (2015), 180–191.
- [33] Cibebe Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. 2020. New results for the complexity of resilience for binary conjunctive queries with self-joins. In *PODS*. 271–284.
- [34] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. 2004. Approximation algorithms for partial covering problems. *Journal of Algorithms* 53, 1 (2004), 55–84.
- [35] Andrew V Goldberg. 1984. Finding a maximum density subgraph. (1984).
- [36] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *PODS*. 31–40.
- [37] Shuguang Hu, Xiaowei Wu, and TH Hubert Chan. 2017. Maintaining densest subsets efficiently in evolving hypergraphs. In *CIKM*. 929–938.
- [38] Xiao Hu, Shouzhao Sun, Shweta Patwa, Debmalaya Panigrahi, and Sudeepa Roy. 2020. Aggregated deletion propagation for counting conjunctive query answers. *PVLDB* 14, 2 (2020), 228–240.
- [39] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *ICALP*. Springer, 597–608.
- [40] Benny Kimelfeld, Jan Vondrák, and Ryan Williams. 2012. Maximizing conjunctive views in deletion propagation. *TODS* 37, 4 (2012), 1–37.
- [41] Benny Kimelfeld, Jan Vondrák, and David P Woodruff. 2013. Multi-tuple deletion propagation: Approximations and complexity. *PVLDB* 6, 13 (2013), 1558–1569.
- [42] Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. 2018. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1236–1252.
- [43] Neha Makhija and Wolfgang Gatterbauer. 2024. Minimally factorizing the provenance of self-join free conjunctive queries. *Proceedings of the ACM on Management of Data* 2, 2 (2024), 1–24.
- [44] Neha Makhija and Wolfgang Gatterbauer. 2025. Is integer linear programming all you need for deletion propagation? A unified and practical approach for generalized deletion propagation. In *Proceedings of the VLDB Endowment* 18, 8 (2025), 2667–2680.
- [45] Zhengjie Miao, Sudeepa Roy, and Jun Yang. 2019. Explaining wrong queries using small examples. In *SIGMOD*. 503–520.
- [46] Dana Moshkovitz. 2012. The projection games conjecture and the NP-hardness of $\ln n$ -approximating set-cover. In *APPROX-RANDOM*. Springer, 276–287.
- [47] Dan Olteanu and Maximilian Schleich. 2016. Factorized databases. *ACM SIGMOD Record* 45, 2 (2016), 5–16.
- [48] Xiao Hu and Stavros Sintos. 2024. Finding smallest witnesses for conjunctive queries. In *27th International Conference on Database Theory (ICDT’24)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 24–1.
- [49] John Paparrizos, Chunwei Liu, Bruno Barbarioli, Johnny Hwang, Ikradya Edian, Aaron J Elmore, Michael J Franklin, and Sanjay Krishnan. 2021. VergeDB: A database for IoT analytics on edge devices.. In *CIDR*.
- [50] Jeff M Phillips. 2017. Coresets and sketches. In *Proceedings of the Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC, 1269–1288.
- [51] Biao Qin, Deying Li, and Chunlai Zhou. 2022. The resilience of conjunctive queries with inequalities. *Information Sciences* 613 (2022), 982–1002.
- [52] Moshe Y Vardi. 1982. The complexity of relational query languages. In *STOC*. 137–146.
- [53] Vijay V Vazirani. 2001. *Approximation Algorithms*. Vol. 1. Springer.
- [54] Mihalís Yannakakis. 1981. Algorithms for acyclic database schemes. In *VLDB*, Vol. 81. 82–94.
- [55] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random sampling over joins revisited. In *SIGMOD*. 1525–1539.

Appendices

A Discussion on Our Improvement over Naive Solutions

At last, let’s see what can be benefited from our algorithmic results, such as in Examples 1.1, 1.2 and 1.3. Below, we compare $\text{SWP}(Q, D)$ with $|D|$ and $|Q(D)|$. For a CQ Q with head-cluster property, let $Q_1, Q_2, \dots, Q_k$ be the subqueries defined on relations partitioned by the output attributes. For a

CQ Q with head-domination property, let $Q_1, Q_2, \dots, Q_k$ be the subqueries defined by the connected components of $G_Q^\exists$. Consider an arbitrary database D . Let D'_i be the projection of $Q(D)$ onto the output attributes of Q_i . Let $D' = \sum_{i=1}^k D'_i$. We can prove for any D : (i) $|D'| \leq |Q(D')| = |Q(D)|$; (ii) $\text{SWP}(Q, D) = |D'|$, if Q has head-cluster property; (iii) $\text{SWP}(Q, D)$ can be approximated within a factor of $|\text{rels}(Q)|$ if Q has head-domination property. In general, $|Q(D)|$ can be as large as $\Theta(|D'|^\rho)$, where ρ is fractional edge covering number of the residual CQ by removing all non-output attributes from Q , and $|D|$ can also be polynomially larger than $|D'|$, so the (approximated) solution to $\text{SWP}(Q, D)$ could be polynomially smaller than both $|D|$ and $|Q(D)|$, while never be larger than them by a query-dependent constant.

For arbitrary CQ Q with database D , let D' be the projection of $Q(D)$ onto output attributes for each relation. We have proved that the approximated solution to $\text{SWP}(Q, D)$ is always smaller than $|D|$ and no larger than $|Q(D')|$ by a constant factor. The improvement would heavily depend on the specific database. As noted, in some applications (such as Example 1.2 and 1.3), a naive solution of size $|Q(D)|$ does not exist.

B Deletion Propagation

In the literature, the classic DP of CQs without self-joins shows resemblance with SWP studied in this work. Here, we perform a comprehensive comparison between them with respect to problem formulation, complexity results, and techniques.

Definition B.1 (Deletion Propagation with View Side Effect [40, 41]). Given a CQ Q , a database D and a subset of query results $J \subseteq Q(D)$, the problem asks to find a subset of tuples $D' \subseteq D$ such that $J \cap Q(D - D') = \emptyset$, and there exists no subset $D'' \subseteq D$ with $J \cap Q(D - D'') = \emptyset$ and $|Q(D - D'')| > |Q(D - D')|$.

DP with view side effects is different from SWP in terms of their objective functions. DP with view side effects focuses on the remaining query results. However, DP with source side effects (as well as ADP and resilience introduced below) also take the underlying input tuples into account - aiming to minimize the number of tuples removed.

Definition B.2 (Deletion Propagation with Source Side Effect [14, 20]). Given a CQ Q , a database D and a subset of query results $J \subseteq Q(D)$, the problem asks to find a subset of tuples $D' \subseteq D$ such that all query results J just disappear after removing all tuples in D' , i.e., $J \cap Q(D - D') = \emptyset$, while there exists no subset $D'' \subseteq D$ with $J \cap Q(D - D'') = \emptyset$ and $|D''| < |D'|$.

We note that if $J = \emptyset$, i.e., no query result is desired to be removed, the trivial solution $D' = \emptyset$ is optimal, and if $J = Q(D)$, DP with source side effect just degenerates to the resilience problem (see Definition B.4).

Definition B.3 (Aggregated Deletion Propagation (ADP) [38]). Given a CQ Q , a database D and a parameter $1 \leq k \leq |Q(D)|$, the problem asks to find a subset of tuples $D' \subseteq D$ such that at least k query results disappear, i.e., $|Q(D - D')| \leq |Q(D)| - k$, while there exists no subset $D'' \subseteq D$ with $|Q(D - D'')| \leq |Q(D)| - k$ and $|D''| < |D'|$.

Similarly, if $k = 0$, i.e., no query result is desired to be removed, the trivial solution $D' = \emptyset$ is optimal, and if $k = |Q(D)|$, ADP degenerates to the resilience problem. We next focus on the difference between resilience and SWP:

Definition B.4 (Resilience [32, 33]). Given a Boolean CQ Q and a database D with $Q(D)$ is true, the problem asks to find a subset of tuples $D' \subseteq D$ such that $Q(D - D')$ is false, while there exists no subset $D'' \subseteq D$ with $Q(D - D'')$ as false and $|D''| < |D'|$.

These two problems have completely different constraints on D' , such that $Q(D') = Q(D)$ is always required by SWP, but $Q(D - D') = \emptyset$ is required by the resilience problem. Note that $Q(D - D') \neq Q(D) - Q(D')$! Hence, there is no connection between solutions to the resilience problem and solutions to SWP.

Complexity. As head domination property also precisely captures the hardness of DP with side effect, we next focus on the difference between our paper and [40]:

- Poly-time solvability (assume $P \neq NP$):
 - SWP is poly-time solvable iff Q has head-cluster property;
 - DP is poly-time solvable iff Q has head-domination property;
- Approximability:
 - SWP is $N^{1-1/\rho}$ -approximable for any Q ;
 - DP is $1/\min\{|\text{attr}(Q)|, |\text{rels}(Q)|\}$ -approximable for any Q ;
- Inapproximability for CQ without head-domination property:
 - SWP is NP-hard to approximate within a factor of $(1 - o(1)) \cdot \log N$;
 - SWP is NP-hard to approximate for line CQs within a factor of $O(2^{(\log N)^{1-\epsilon}})$ for any constant $\epsilon > 0$;
 - DP is NP-hard to approximate within a factor of α , for some $\alpha \in (0, 1)$, which may vary over different CQs;
- Greedy Algorithms:
 - SWP is $O(\log N)$ -approximable for star CQs.
 - SWP is $O(\min\{|Q_{\text{line}}(D)|^{\frac{1}{2}+o(1)}, \text{dom}^{0.5778}, N^{\frac{1}{2}}\})$ -approximable for line CQs (see Theorem 6.6).
 - DP is $\frac{1}{2}$ -approximable for star CQs.

Techniques. We are not aware of similar techniques used for upper and lower bounds between this article and [40].

C Missing Materials of Computing Non-dangling Tuples for Acyclic CQs

Given an acyclic CQ Q and a database D , the set of non-dangling tuples can be computed in two phases. Let $\mathbb{T}$ be a join tree of Q . Note that there is a one-to-one correspondence between relations in Q and nodes in $\mathbb{T}$.

- Phase I: A bottom-up phase of semi-joins. We start from an arbitrary leaf node R_i of $\mathbb{T}$. Let R_j be the parent of R_i . We update R_j with $R_j \bowtie R_i$, and then conceptually remove R_i from $\mathbb{T}$.
- Phase II: A top-down phase of semi-joins. We start from the root R_j of $\mathbb{T}$. Let R_i be one child of R_j . We update R_i with $R_i \bowtie R_j$, and then iteratively apply this step for each node in a top-down way.

D Proof of Theorem 6.7

We focus on line-3 query $Q_{\text{line3}}(A_1, A_4) : -R_1(A_1, A_2), R_2(A_2, A_3), R_3(A_3, A_4)$, which can be generalized to line- m query $Q_{\text{line}}(A_1, A_m) : -R_1(A_1, A_2), R_2(A_2, A_3), \dots, R_m(A_m, A_{m+1})$, by putting $A_3, A_4, \dots, A_{m-1}$ as identical copies of A_m .

Definition D.1 (Label Cover). Given a complete bipartite graph (U, V, E) with $|U| = |V| = n$, a finite alphabet Σ with $|\Sigma| > n$, constraints $C_{u,v}$ for every edge $e = (u, v)$, the goal is find assignments to the vertices $M_U : U \rightarrow 2^\Sigma$ and $M_V : V \rightarrow 2^\Sigma$ that every edge is satisfied, i.e., there exists $x \in M_U(u), y \in M_V(v)$ such that $(x, y) \in C_{u,v}$, and $\sum_{u \in U} |M_U(u)| + \sum_{v \in V} |M_V(v)|$ is minimized. The problem size m is defined as $n \times |\Sigma|$.

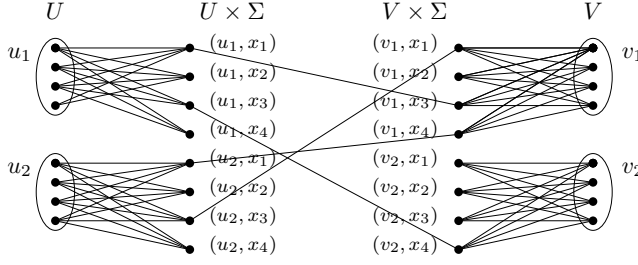


Fig. 7. An example of hard instance constructed for Q_{line3} .

LEMMA D.2 ([25]). *There is no poly-time algorithm that can approximate the label cover instances of size m within $O(2^{(\log m)^{1-\epsilon}})$ factor for any $\epsilon > 0$.*

We set $\text{dom}(A) = U \times [n^2]$, $\text{dom}(B) = U \times \Sigma$, $\text{dom}(C) = V \times \Sigma$ and $\text{dom}(D) = V \times [n^2]$. The database D includes:

- $R_1 = \{(\langle u, i \rangle, \langle u, x \rangle) : u \in V, i \in [n^2], x \in \Sigma\}$;
- $R_2 = \{(\langle u, x \rangle, \langle v, y \rangle) : u, v \in V, x, y \in \Sigma, (x, y) \in C_{uv}\}$;
- $R_3 = \{(\langle v, y \rangle, \langle v, i \rangle) : v \in V, y \in \Sigma, i \in [n^2]\}$;

We have $N = O(n^2 \cdot |\Sigma|^2 + n^3 \cdot |\Sigma|) = O(m^2)$. It can be easily checked that $Q_{\text{line3}}(D) = U \times [n^2] \times V \times [n^2]$ with n^6 query results. For any witness D' for $\text{SWP}(Q_{\text{line3}}, D)$, let D'_1, D'_2, D'_3 be the set of tuples chosen from R_1, R_2, R_3 separately. An example is given in Figure 7.

For a sub-database $D' \subseteq D$, let R'_1, R'_3 are the corresponding subset of tuples from R_1, R_3 in D' . A solution D' to $\text{SWP}(Q_{\text{line3}}, D)$ is integral if for any $(u, x) \in U \times \Sigma$, either $\sigma_{B=(u,x)} R'_1 = \emptyset$ or $\sigma_{B=(u,x)} R'_1 = \{\langle u, i \rangle, \langle u, x \rangle : i \in [n^2]\}$, and for any $(v, y) \in V \times \Sigma$, either $\sigma_{C=(v,y)} R'_3 = \emptyset$ or $\sigma_{C=(v,y)} R'_3 = \{\langle v, i \rangle, \langle v, y \rangle : i \in [n^2]\}$.

LEMMA D.3. *Every non-integral solution to $\text{SWP}(Q_{\text{line3}}, D)$ can be transformed into an integral solution to $\text{SWP}(Q_{\text{line3}}, D)$.*

PROOF. Consider an arbitrary non-integral solution D' . W.o.l.g., assume there exists some $(u, x) \in U \times \Sigma$, such that $\sigma_{B=(u,x)} R'_1 \neq \emptyset$ and $\sigma_{B=(u,x)} R'_1 \neq \{\langle u, i \rangle, \langle u, x \rangle : i \in [n^2]\}$. There must exist some pair of (i, j) such that $\langle u, i \rangle, \langle u, x \rangle \in D'_1$ but $\langle u, j \rangle, \langle u, x \rangle \notin D'_1$. Let $X_i = \{x \in \Sigma : \langle u, i \rangle, \langle u, x \rangle \in D'_1\}$ and $X_j = \{x \in \Sigma : \langle u, j \rangle, \langle u, x \rangle \in D'_1\}$. W.l.o.g., assume $|X_i| \leq |X_j|$. As D' is a solution to $\text{SWP}(Q_{\text{line3}}, D)$, it is feasible to remove $\langle u, j \rangle, \langle u, x \rangle \in D'_1$ for $x \in X_j$ from D' and add $\langle u, j \rangle, \langle u, x \rangle \in D'_1$ for $x \in X_i$ to D' , which yields another solution to $\text{SWP}(Q_{\text{line3}}, D)$ without increasing the number of tuples. After applying this step iteratively, we can obtain a solution such that every (u, i) is incident to the same values in $\text{dom}(B)$ over $i \in [n^2]$, i.e., an integral solution. $\square$

From now on, it suffices to consider the integral solution to $\text{SWP}(Q_{\text{line3}}, D)$. Moreover, we can assume $|D'_2| = n^2$ by picking an arbitrary edge $(\langle u, x \rangle, \langle v, y \rangle) \in R_2$ with $(x, y) \in C_{uv}$ for $(u, v) \in U \times V$. Observe that there is a solution to label cover of cost at most c if and only if there is an integral solution to SWP for Q_{line3} of cost at most $n^2 c + n^2 = n^2(c + 1)$.

- *Direction Only-If:* Let $S = (M_U, M_V)$ be the solution to label cover. We define an integral solution D_S to $\text{SWP}(Q_{\text{line3}}, D)$ based S . For each vertex $u \in U$, we add the set of tuples $(\langle u, * \rangle, x) \in R_1$ to D_S if $x \in M_U(u)$. Similarly, for each vertex $v \in V$, we add the set of tuples $(\langle v, * \rangle, y) \in R_3$ to D_S if $y \in M_V(v)$. Together with the n^2 tuples chosen from R_2 , we have obtained a witness D_S of size $c \cdot n^2 + n^2 = (c + 1)n^2$.

- *Direction If:* Let D_S be an integral witness to $\text{SWP}(Q_{\text{line3}}, D)$ of cost c' . We construct a solution (M_U, M_V) to label cover as follows. If $(\langle u, * \rangle, x) \in R_1$ is included by D_S , we add x to $M_U(u)$. Similarly, if $(\langle v, * \rangle, y) \in R_3$ is included by D_S , then we add y to $M_V(v)$. It can be easily checked that we have obtained a solution to label cover of size $\frac{c'-1}{n^2}$.

If there is a poly-time algorithm that can approximate SWP for Q_{line3} within $2^{(\log \sqrt{N})^{1-\epsilon}}$ factor, there is a poly-time algorithm that can approximate the label cover problem within $2^{(\log \sqrt{N})^{1-\epsilon}} = 2^{(\log m)^{1-\epsilon}}$ factor, coming to a contradiction of Lemma D.2.

Received 1 December 2024; revised 6 July 2025; accepted 1 December 2025